وزارت علوم تحقیقات و فناوری

دانشگاه صنعتی سجاد

غیردولتی-غیرانتفاعی

دستور کار آزمایشگاه

سیستم‌های دیجیتال ۲

تهیه و تنظیم:

محمدجواد مالکی عباس گلمکانی

پیشگفتار:

امروزه استفاده از تراشه‌های مختلف میکروکنترلر در ساخت و کنترل پروژه‌های مختلف آزمایشگاهی و صنعتی به‌عنوان ابزاری قدرتمند در خدمت طراحان قرارگرفته است. با ظهور این تراشه‌های منطقی و برنامه پذیر، مدارات دیجیتال پیچیده، جای خود را به این تراشه‌ها داده‌اند. میکروکنترلر حکم یک کامپیوتر در ابعاد کوچک و قدرت کمتر را دارد. بیشتر میکروکنترلرها برای کنترل و تصمیم‌گیری استفاده می‌شود. میکروکنترلر برای کنترل ربات‌ها تا استفاده در کارخانه‌های صنعتی کاربرد دارد .میکروکنترلر درواقع تراشه قابل‌برنامه‌ریزی هست که عملکرد آن‌ها از قبل تعیین نشده است.

در این دستور کار مطالب اساسی در مورد کار با میکروکنترلرهای AVR بیان‌شده است. در ابتدا مقدمه‌ای از میکروکنترلرها جهت آشنایی با آن‌ها بیان‌شده است. در ادامه به معرفی کامپایلر کدوی ژن پرداخته و سپس نحوه پروگرم کردن میکروکنترلر شرح داده‌شده است. این دستور کار شامل ۹ آزمایش می‌باشد. عمده مفاهیم میکروکنترلرها و راه‌اندازی بخش‌های مهم آن‌ها در قالب این ۹ آزمایش گنجانده‌شده است. با توجه به کم بودن زمان کلاس و حجم بالای مطالب، انتظار می‌رود تا دانشجویان عزیز پیش از ورود به کلاس آزمایش مربوط به جلسه پیش رو را به‌طور کامل مطالعه کرده و با آمادگی کامل در کلاس حضور پیدا کنند.

در پایان از تمام اساتید و دوستان بخصوص جناب آقای دکتر گلمکانی و دانشجویان عزیز سرکار خانم‌ها وحیده فرزانه، ریحانه بخشی و فائزه رضایی که بنده را در تهیه این اثر یاری نمودند کمال تشکر رادارم.

با توجه به اینکه هیچ کاری بدون اشکال نیست، اگر اشکالی در این دستور کار مشاهده فرمودید، بخشیده و نظرات اصلاحی خود را به آدرس الکترونیکی mjmaleki88@gmail.com ارسال نموده تا در نسخه‌های بعد اصلاح شوند. پیشاپیش از شما سپاسگزارم.

محمدجواد مالکی

تابستان ۹۹

فهرست

***لیست قطعات ثابت:**

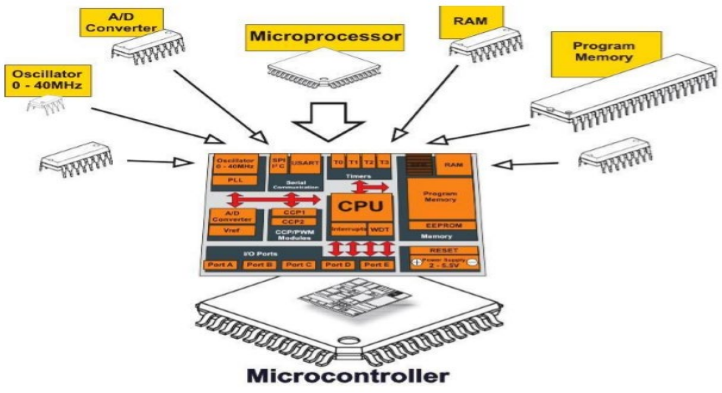
۱- میکرو کنترلر ATMEGA32/16 (A) ---------- ۱ عدد
۲- برد بورد ---------- ۱ عدد
۳- رگلاتور LM7805 ---------- ۱ عدد
۴- خازن عدسی ۱۰۰ نانو فاراد ---------- ۴ عدد
۵- خازن ۱ میکرو فاراد (۶/۳ ولت به بالا) ---------- ۲ عدد
۶- خازن ۱۰۰ میکرو فاراد (۶/۳ ولت به بالا) ---------- ۱ عدد
۷- خازن ۱۰ میکرو فاراد (۳۰ ولت) ---------- ۱ عدد
۸- LED (RED-GREEN) ---------- از هرکدام ۳ عدد

❋مقدمه:

میکروکنترلر یک تراشه الکترونیکی قابل‌برنامه‌ریزی است که استفاده از آن باعث افزایش سرعت و کار آیی مدار در مقابل کاهش حجم و هزینه مدار می‌گردد. با ساخت میکروکنترلرها تحول شگرفی در ساخت تجهیزات الکترونیکی نظیر لوازم‌خانگی، صنعتی، پزشکی، تجاری و ... به وجود آمده است که بدون آن تصور تجهیزات و وسایل پیشرفته امروزی غیرممکن است. یکی از میکروکنترلرهای پرکاربرد و معروف AVR نام دارد که ساخت Atmel شرکت می‌باشد.

هنگامی که قطعات سازنده یک میکروکامپیوتر در یک تراشه و در کنار هم قرار گیرند، یک میکروکنترلر به وجود می‌آید. مطابق با شکل ۱، در واقع میکروکنترلر یک آی‌سی شامل یک CPU، به همراه مقدار مشخصی از حافظه‌های ROM و RAM، پورت‌های ورودی/خروجی و همچنین واحدهای جانبی دیگری نظیر تایمر، رابط سریال و ... می‌باشد؛ به‌عبارت‌دیگر میکروکنترلر یک تراشه الکترونیکی قابل برنامه‌ریزی است که استفاده از آن باعث افزایش سرعت و کار آیی مدار در مقابل کاهش حجم و هزینه مدار می‌گردد.

در تلفن، موبایل، سیستم ایمنی، دربازکن گاراژ، دستگاه فاکس، کامپیوتر شخصی PC، ویدئو، دوربین ویدئویی، چرخ‌خیاطی، سیستم‌های تهویه، سرعت‌سنج و غیره از میکروکنترلرها استفاده‌شده است.



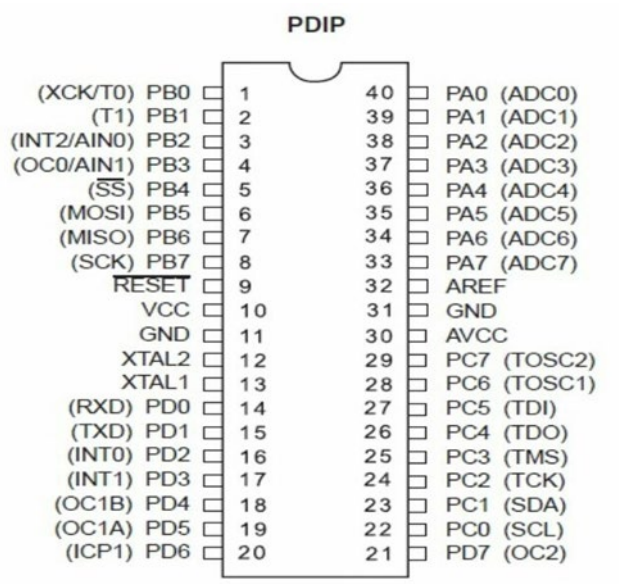
شکل ۱

❋آشنایی با میکرو کنترلرAtmega32 :

شکل ظاهری و پایه ها:

ATMEGA32 در سه نوع بسته‌بندی PDIP با ۴۰ پایه و TQFP با ۴۴ پایه و MLF با ۴۴ پایه ساخته می‌شود که نوع PDIP متداول‌تر است. ATMEGA32 دارای چهار پورت ۸ بیتی (۱ بایتی) است که علاوه بر اینکه به‌عنوان یک پورت معمولی می‌توانند باشند کارهای دیگری نیز انجام می‌دهند. به‌طور مثال PORT A می‌تواند به‌عنوان ورودی ADC (تبدیل ولتاژ آنالوگ به کد دیجیتال) استفاده شود که این خاصیت‌های مختلف پورت در برنامه‌ای که نوشته می‌شود تعیین خواهد شد. ولتاژ مصرفی این آی سی از 4.5 V تا 5.5 V می‌تواند باشد. فرکانس کار هم تا 16 MHz می‌تواند انتخاب شود که تا 8 MHz نیازی به کریستال خارجی نیست و در داخل خود آی سی می‌تواند تأمین شود. فرکانس کار ازجمله مواردی است که باید در برنامه تعیین شود. لازم به ذکر است که این فرکانس بدون هیچ تقسیمی به

CPU داده می‌شود؛ بنابراین این خانواده از میکروکنترلرها سرعت بیشتری نسبت خانواده‌های دیگر دارند. پایه‌ی شماره ۹ نیز ریست سخت‌افزاری است که برای عملکرد عادی آی سی نباید به‌جایی وصل شود و برای ریست کردن باید به زمین وصل شود. پایه‌های ۱۲ و ۳ نیز برای استفاده از کریستال خارجی تعبیه‌شده است.



شکل ۲

ساختار داخلی ATMGA32 :

برنامه‌ای که برای میکروکنترلر در کامپیوتر نوشته می‌شود وقتی‌که برای استفاده در آی‌سی ریخته می‌شود (توسط پروگرامر مخصوص آن خانواده) در ROM ذخیره خواهد شد. در ATMEGA32 مقدار این حافظه به ۳۲ (کیلوبایت) است. در این آی سی مکانی برای ذخیره موقت اطلاعات یا همان RAM هم وجود دارد که مقدارش ۲ کیلوبایت است. در RAM اطلاعات فقط تا زمانی که انرژی الکتریکی موجود باشد خواهد ماند و با قطع باتری اطلاعات از دست خواهند رفت. در ATMEGA32 مکانی برای ذخیره اطلاعات وجود دارد که با قطع انرژی از دست نخواهند رفت. به این نوع حافظه‌ها EEPROM گفته می‌شود که در این تراشه مقدارش یک کیلوبایت است و تا ۱۰۰۰۰۰ بار می‌تواند پر و خالی شود.

مهم‌ترین مشخصات این میکروکنترلر ۴۰ پایه عبارت است از:

- کارایی بالا و توان مصرفی کم
- ۳۲ رجیستر (ثبات) ۸ بیتی
- سرعت با سقف ۱۶ میلیون دستور در ثانیه در فرکانس 16 MHz
- ۳۲ کیلوبایت حافظه FLASH داخلی قابل‌برنامه‌ریزی باقابلیت ده هزار بار نوشتن و پاک کردن

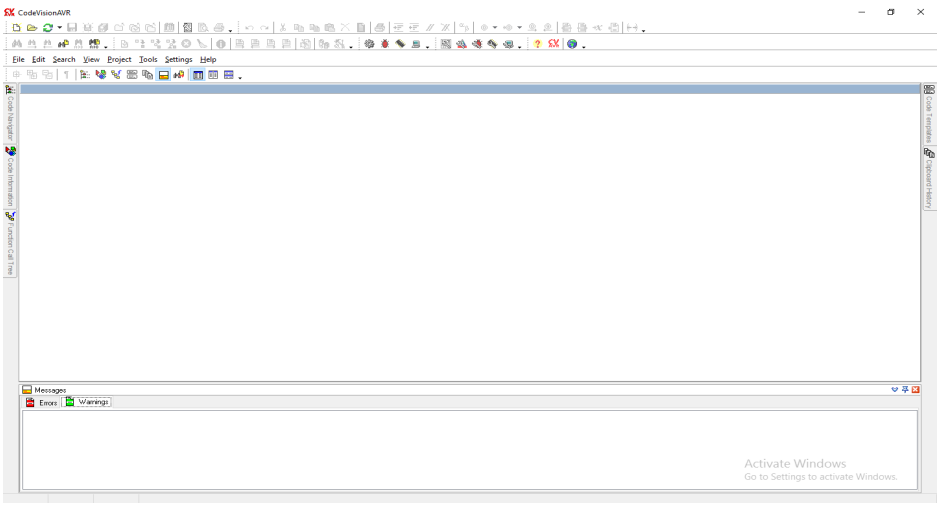
نکته: حافظه فلش حافظه‌ای است که برنامه نوشته‌شده توسط شما بعد از پروگرام کردن در آن قرار می‌گیرد.

- ۲ کیلوبایت حافظه داخلی SRAM
- ۱۰۲۴ بایت حافظه EEPROM داخلی قابل‌برنامه‌ریزی باقابلیت صد هزار بار نوشتن و خواندن

- قابلیت ارتباط JTAG
- دو تایمر/شمارنده هشت بیتی
- یک تایمر/شمارنده شانزده بیتی- چهار کانال PWM
- هشت کانال مبدل A/D ده بیتی
- ۱ مقایسه کننده آنالوگ داخلی
- فرکانس کاری ۰ تا ۱۶ مگاهرتز

***آشنایی با نرم‌افزار کدویژن**

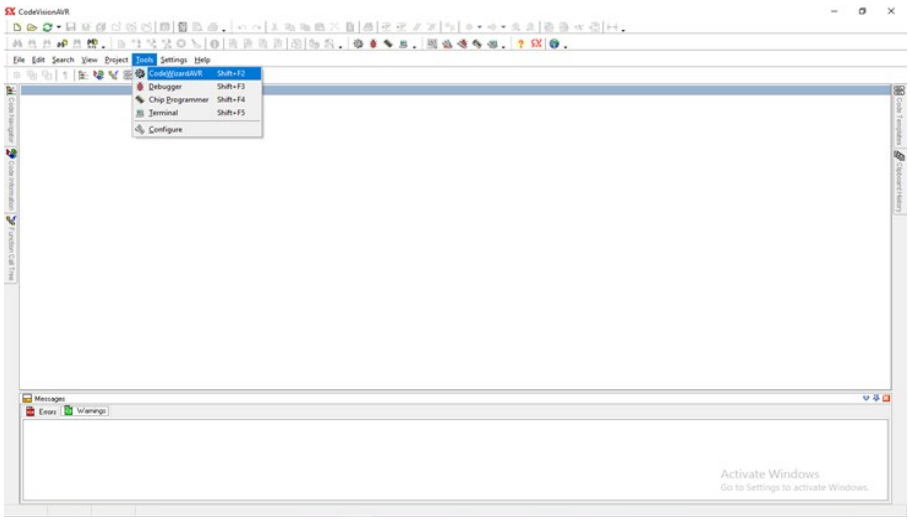
برای استفاده از میکروکنترلر های AVR نیازمند برنامه‌نویسی در محیط کدویژن هستیم. برای این‌که برنامه‌های نوشته‌شده به زبان C را بتوانیم به زبان قابل‌فهم برای ماشین و میکروکنترلر تبدیل کنیم به کامپایلر نیازمندیم. در کل برای کار با میکروکنترلرها بعد از انتخاب میروکنترلر مورد نظر برنامه خود را در محیط کامپایلر نوشته و آن را به فایل hex تبدیل کرده و بر روی میکرو توسط پروگرمر آپلود می‌کنیم.



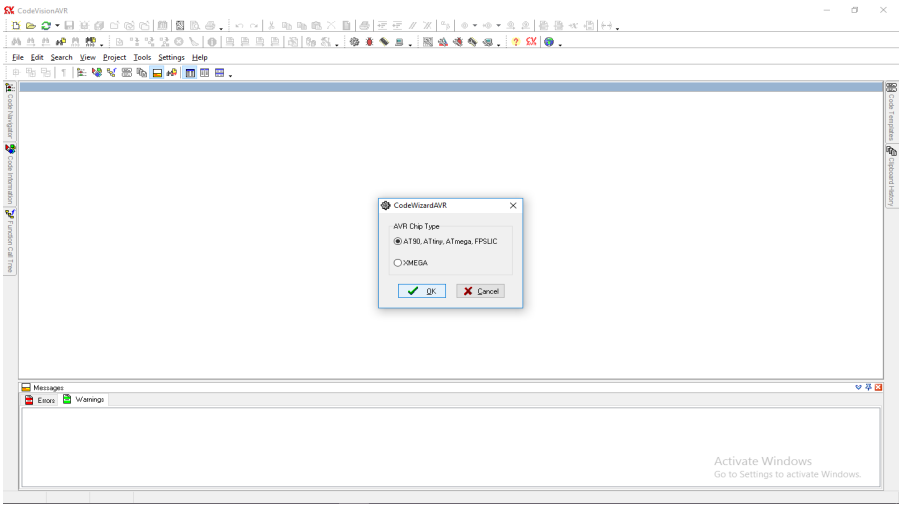
شکل ۳- محیط نرم‌افزار کدویژن

***ایجاد پروژه جدید در کدویژن:**

۱- برای راه‌اندازی کدویزارد در صفحه‌ی اصلی نرم‌افزار کدویژن از منوی tools گزینه‌ی codewizard AVR را انتخاب کنید. هم‌چنین می‌توانید بجای آن از منوی ابزار بالای نرم‌افزار گزینه چرخ‌دنده را کلیک کنید. در پنجره بازشده، می‌خواهد که نوع چیپ ازنظر سری ساخت را مشخص کنید. برای ایجاد پروژه جدید باید از project استفاده کنیم.

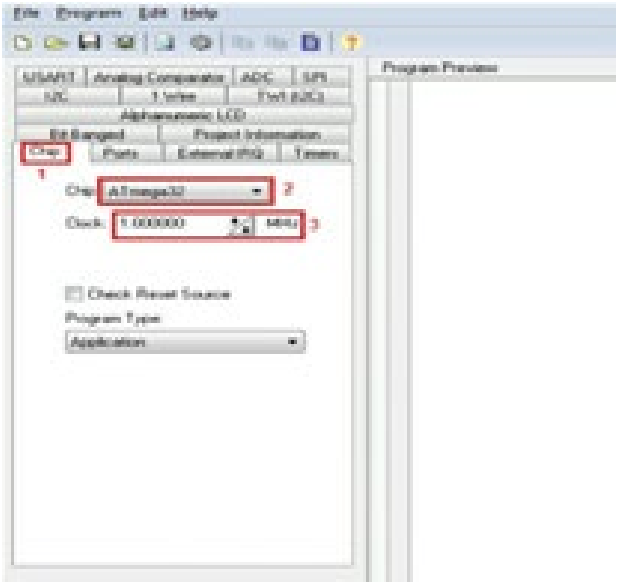


شکل ۴



شکل ۵

۲- در این قسمت نوع چیپ و فرکانس کاری میکرو را مشخص می‌کنیم فرکانس کاری میکرو پیش‌فرض 1 MHz می‌باشد.



شکل ۶

۳- پس از انتخاب امکانات مورد نظر و انجام تنظیمات مربوطه برای ساخت فایل پروژه، از منو program گزینه ی Generate, Save and Exit را کلیک می کنیم. اکنون ۳ بار باید فایل‌های مربوط به پروژه را بانام ترجیحاً یکسان ذخیره کنید. اکنون کدویزارد بخش اولیه برنامه ایجادشده و شما می‌توانید شروع به ادامه برنامه‌نویسی با اضافه کردن کدهای ساخته‌شده نمایید.

* ساختار برنامه میکروکنترلر به زبانC

برنامه میکروکنترلر باید توسط برنامه‌نویس روی یک کامپایلر نوشته شود و سپس توسط پروگرامر روی میکرو پروگرام شود. برنامه‌ای که نوشته می‌شود باید طوری نوشته شود که وقتی روی آی سی پروگرام شد دائماً اجرا شود و هیچ‌گاه متوقف نشود. راه‌حل این مسئله قرار دادن کدهای برنامه درون یک حلقه نامتناهی است. این عمل باعث می‌شود تا میکروکنترلر هیچ‌گاه متوقف نشود و به‌طور مداوم عملکرد مورد انتظار را اجرا کند؛ بنابراین ساختار یک برنامه به زبان c که قرار است در کامپایلر CodeVision نوشته شود به‌صورت زیر درمی‌آید:

```
#include < HeaderFiles.h>
محل معرفی متغیرهای عمومی، ثوابت و توابع
 Void main (void)
{
```

کدهایی که در این محل قرار می‌گیرند فقط یک‌بار اجرا می‌شوند معمولاً مقداردهی اولیه به رجیسترها در این ناحیه انجام می‌شود.

```
        While(1)
        {
کدهایی که باید مدام در میکروکنترلر اجرا شوند
        }
}
```

بنابراین همان‌طور که مشاهده می‌شود:

۱- خطوط ابتدایی برنامه، دستور فراخوانی فایل‌های سرآمد (Header Files) می‌باشد فایل‌های سرآمد فایل‌هایی با پسوندh. هستند که حاوی پیش تعریف‌ها و الگوهای توابع می‌باشند.

۲- قالب اصلی برنامه بر مبنای تابعی به نام main بناشده است. تابعی که اصولاً ورودی و خروجی ندارد و کدهای اصلی برنامه را در خود دارد.

معرفی تابع:

یک تابع همانند دستگاهی است که مواد اولیه را دریافت می‌کند و بعد از انجام عملیات موردنظر روی آن‌ها خروجی مطلوب را تحویل می‌دهد. توابع در زبان C یا توابع کتابخانه‌ای هستند یا توابعی هستند که کاربر برحسب نیاز برنامه خود اضافه می‌کند. تابع اصلی برنامه‌نویسی به زبان C تابع main نام داردکه در تمامی برنامه‌ها وجود داشته و بدون ورودی و خروجی است.

زبان C دارای توابعی است که از قبل نوشته‌شده‌اند و توابع کتابخانه‌ای نامیده می‌شوند. درواقع فرایندهایی که پرکاربرد هستند و در اغلب برنامه‌ها مورداستفاده قرار می‌گیرند. به‌صورت توابع مستقل قبلاً نوشته‌شده‌اند و درون فایل‌هایی قرار داده‌شده‌اند با اضافه کردن فایل‌های سرآمد که تعریف آن توابع در آن‌ها قرار دارد می‌توان از آن توابع استفاده کرد.

انواع متغیرها ازنظر محل تعریف در برنامه:

متغیرها ازنظر مکانی که در برنامه تعریف می‌شوند، به دودسته کلی تقسیم می‌شوند:

۱- متغیرهای عمومی (Global)

۲- متغیرهای محلی (Local)

متغیرهایی که قبل از تابع main تعریف می‌شوند را متغیرهای عمومی گویند و در همه جای برنامه می‌توان به آن دسترسی داشت؛ اما متغیرهای محلی در بدنه توابع تعریف می‌شوند و در بیرون از آن تابع، دسترسی به آن ممکن نیست. درواقع با تعریف یک متغیر عمومی در ابتدای برنامه، مقدار مشخصی از حافظه برای همیشه به آن متغیر تخصیص می‌یابد اما متغیرهای محلی تنها در زمان احتیاج تعریف‌شده و در حافظه می‌نشینند و بعد از مدتی از حافظه پاک می‌شوند.

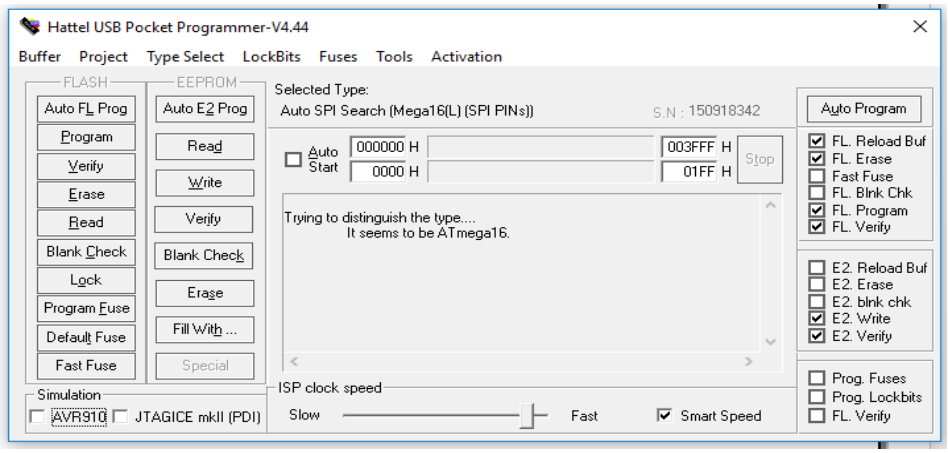
در اولین قدم باید مشخص کنیم که این تابع چه خروجی را به ما می‌دهد یعنی مثلاً اگر عدد صحیح برگرداند از int، اگر کاراکتر برگرداند از char و به همین ترتیب برای انواع دیگر و اگر هیچ مقداری را برنگرداند از استفاده void می‌کنیم.

***نحوه پروگرم کردن میکروکنترلر:**

۱- قرار دادن میکرو بر روی پروگرمر:

تذکر: به جهت میکرو دقت کنید.

بعد از قرار دادن میکرو، پروگرمر به‌طور خودکار مدل میکروکنترلر را شناسایی کرده و مدل میکروکنترلر شما رو نمایش می‌دهد (شکل ۷).

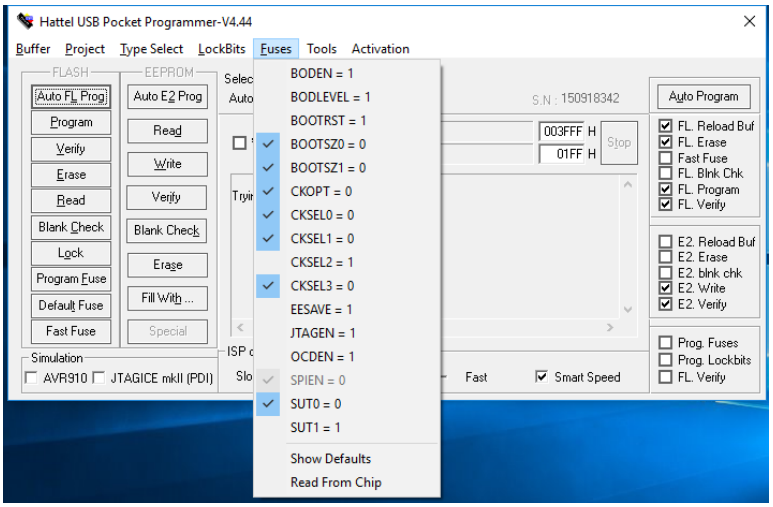


شکل ۷

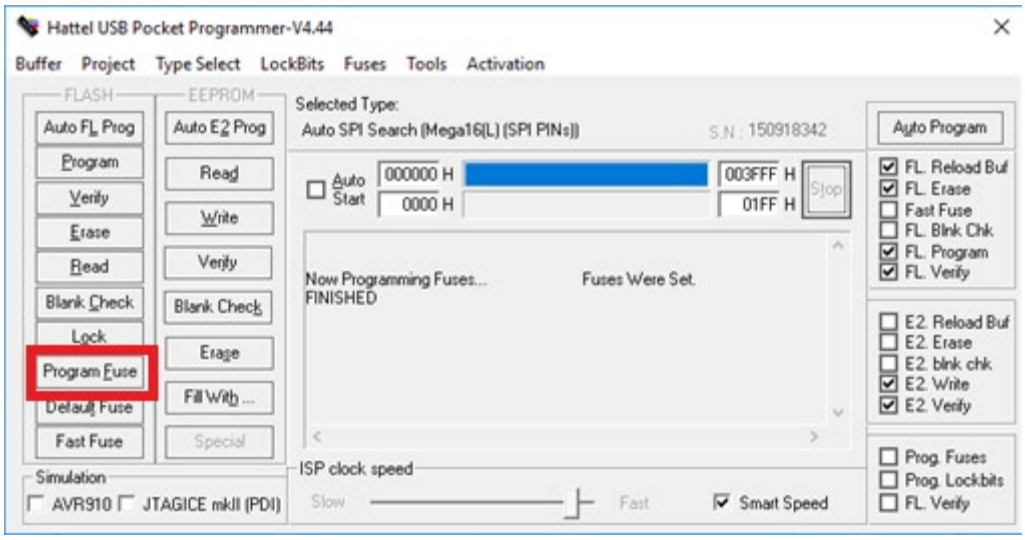
۲- تنظیم فیوزبیت‌ها:

درصورتی‌که تاکنون فیوز بیت‌های میکروکنترلر شما برنامه‌ریزی نشده باشد، از قسمت فیوز می‌توانید آن‌ها را برنامه‌ریزی کنید (شکل ۸).

بعد از تنظیم فیوز بیت‌های، توسط کلید program Fuse می‌توانید فیوزها رو برنامه‌ریزی کنید (شکل ۹).



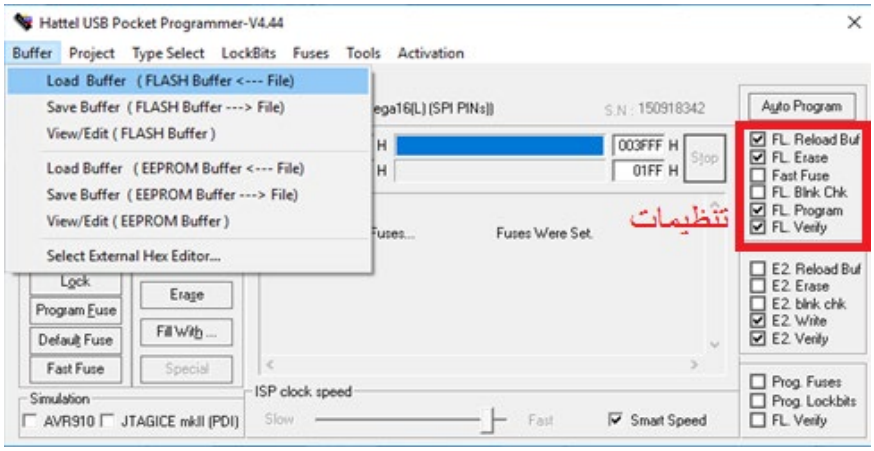
شکل ۸



شکل ۹

۳- پروگرام میکروکنترلر:

برای انتقال برنامه نوشته‌شده به میکروکنترلر از فایل با پسوندHEX. که توسط کامپایلر ایجادشده، استفاده می‌شود. از منوی Buffer آدرس فایلHEX. را می‌توان به میکروکنترلر داد (شکل ۴). سپس با زدن دکمه Auto FL Prog میکروکنترلر برنامه‌ریزی می‌شود. دقت کنید برای استفاده از کلید Auto FL Prog تنظیمات پروگرمر مانند شکل ۱۰ باشد.

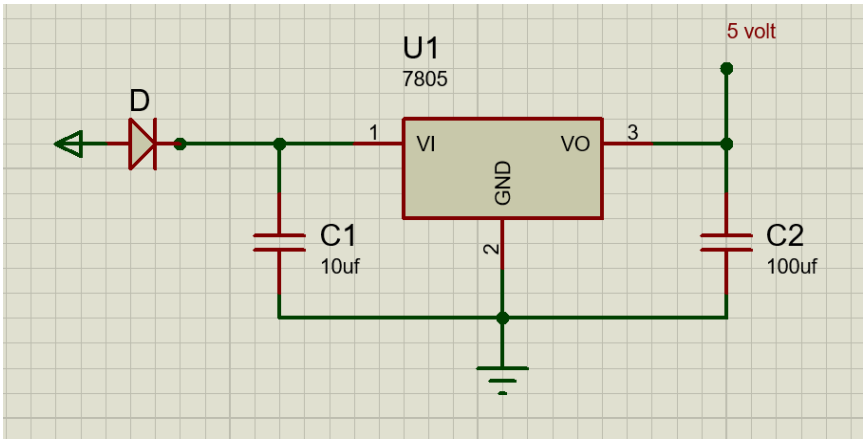


شکل ۱۰

طراحی تغذیه مدار با استفاده از رگلاتور

در اکثر سیستم‌های الکتریکی و الکترونیکی نیازمند داشتن ولتاژ ورودی (منبع تغذیه) با ولتاژ ثابت هستیم اما با افزایش جریانی که مدار می‌کشد، ولتاژ منبع تغذیه ورودی کاهش پیدا می‌کند. برای داشتن ولتاژی تنظیم‌شده و ثابت از المانی به نام رگلاتور که دارای سه‌پایه به‌صورت شکل زیر است (Regulator) استفاده می‌شود. درواقع این آی سی از نوسانات ولتاژی جلوگیری کرده و خروجی دقیق و تمیز به ما می‌دهد.

این بخش از مدار وظیفه‌ی تبدیل ولتاژ ورودی مدار به ولتاژ ۵ ولت را بر عهده دارد به این دلیل که میکروکنترلر و اکثر قطعات برد با برق ۵ ولت کار می‌کنند. برای این تبدیل ولتاژ از رگلاتور ۷۸۰۵ استفاده می‌کنیم و برای گرفتن نویز مدار یک خازن با ظرفیت بالا (۱۰۰ میکرو فاراد) و یک خازن با ظرفیت پایین (۱۰) با مدار موازی می‌کنیم بهتر است یک آل ای دی با این بخش موازی کنیم تا وضعیت روشن یا خاموش بودن مدار کاملاً مشخص باشد.



شکل ۱۱

آزمایش اول: راه‌اندازی سون‌سگمنت (7-Segment)

هدف:

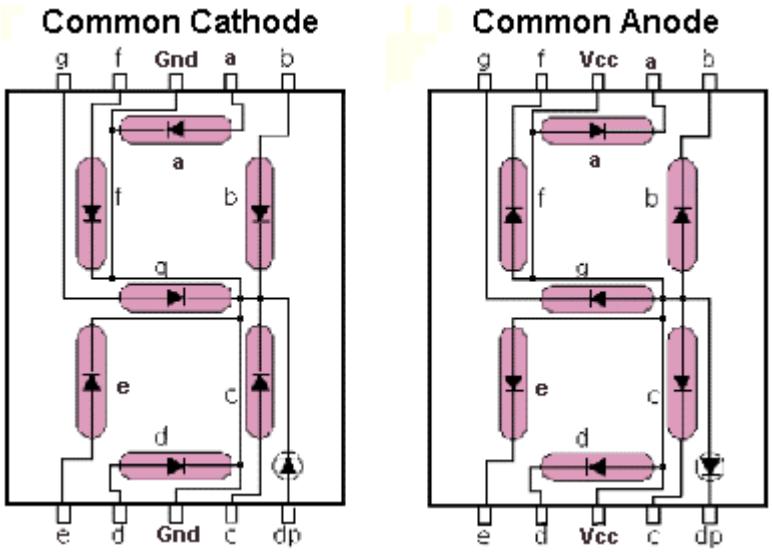
۱- آشنایی با پورت‌ها در میکروکنترلر
۲- آشنایی با دیکدر SN74LS47

قطعات مورد نیاز:

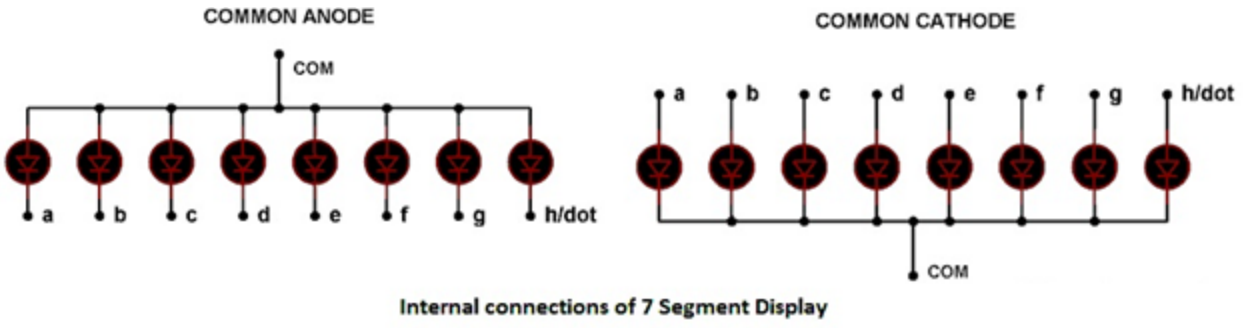
۱- سون‌سگمنت با دو بلاک (آند مشترک)
۲- SN74LS47 (BCD/7segment decoder)
۳- 2N2222 (NPN) ---------- ۲ عدد
۴- مقاومت ۱۰ کیلو اهم ---------- ۲ عدد
۵- مقاومت ۱ کیلو اهم ---------- ۱۰ عدد

شرح آزمایش:

سون‌سگمنت‌ها فطعاتی هستند که در آنها از ۸ LED استفاده شده است. ۷ LED مربوط به نمایش اعداد می‌باشد و یک LED جهت ممیز ارقام است. سون سگمنت‌ها چون از پایه‌های آند و کاتد دارند لذا آن‌ها را LED تشکیل‌شده و هرLED به دودسته آند مشترک و کاتد مشترک دسته‌بندی می‌کنند. اگر از داخل آندها به هم متصل باشند آند مشترک و اگر هم متصل باشند کاتد مشترک می‌باشند. کاتدها به



شکل ۱-۱: ساختار سون سگمنت



شکل ۲-۱: ساختار سون‌سگمنت

در سون سگمنت کاتد مشترک سر منفی یا کاتد تمامی LED ها از داخل به پایه‌ی مشترک متصل شده‌اند که باید به زمین وصل شود. به‌منظور روشن کردن LED موردنظر نیاز است تا ولتاژ ۵+ ولت به قطب مثبت و یا آند آن LED اعمال شود و لازم است که یک مقاومت محدودکننده‌ی جریان نیز به قطب مثبت متصل گردد.

در سون سگمنت آند مشترک سرهای مثبت یا آند همه‌ی LED 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 و لازم است که یک مقاومت محدودکننده‌ی جریان نیز سری گردد.

به جدول ۱ و ۲ دقت کنید، کد زیر برای عدد ۴ می‌باشد:

جدول ۱-۱

سگمنت کاتد مشترک	A	B	C	D	E	F	G
کد باینری	۰	۱	۱	۰	۰	۱	۱

جدول ۲-۱

سگمنت آند مشترک	A	B	C	D	E	F	G
کد باینری	1	0	0	1	1	0	0

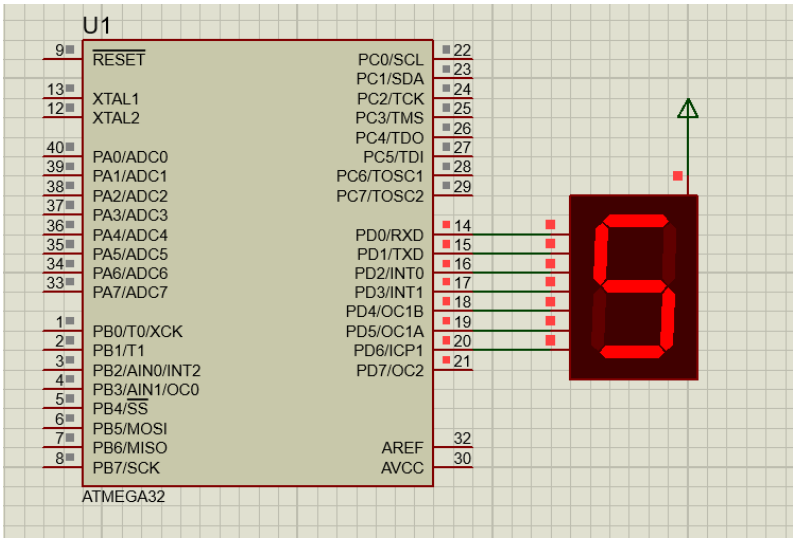
نکته: در سون سگمنت آند مشترک برای روشن شدن هر .باید آن را صفر کنیمLED
نکته: در سون سگمنت کاتد مشترک برای روشن کردن هر .باید آن را یک کنیمLED
نکته: در سون سگمنت آند مشترک پایه کنترل باید یک شود و در کاتد مشترک پایه کنترل باید صفر (زمین) شود.

جدول ۳ هرگز اعدادی که می‌توان روی یک سون سگمنت آند مشترک و کاتد مشترک نمایش داد را نشان می‌دهد:

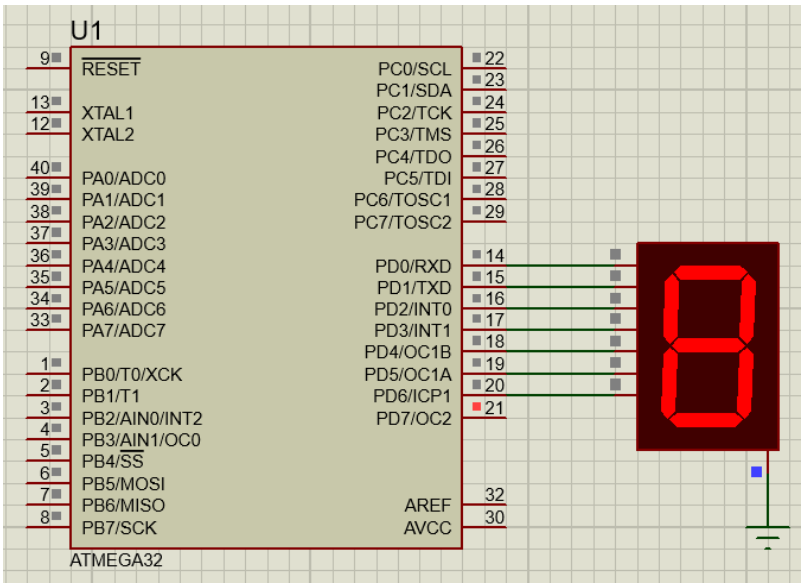
جدول ۳-۱

Numbers	Common Cathode		Common Anode	
	(DP)GFEDCBA	HEX Code	(DP)GFEDCBA	HEX Code
0	00111111	0x3F	11000000	0xC0
1	00000110	0x06	11111001	0xF9
2	01011011	0x5B	10100100	0xA4
3	01001111	0x4F	10110000	0xB0
4	01100110	0x66	10011001	0x99
5	01101101	0x6D	10010010	0x92
6	011111101	0x7D	10000010	0x82
7	00000111	0x07	11111000	0xF8
8	01111111	0x7F	10000000	0x80
9	01101111	0x6F	10010000	0x90

شکل ۲ و ۳ - نحوه اتصال دو مدل سون سگمنت را نشان می‌دهند. چون جریان خروجی پایه‌های میکروکنترلر محدود است مقاومت‌های محدودکننده جریان قرار داده نشده‌اند.



شکل ۲-۱: سون‌سگمنت آند مشترک



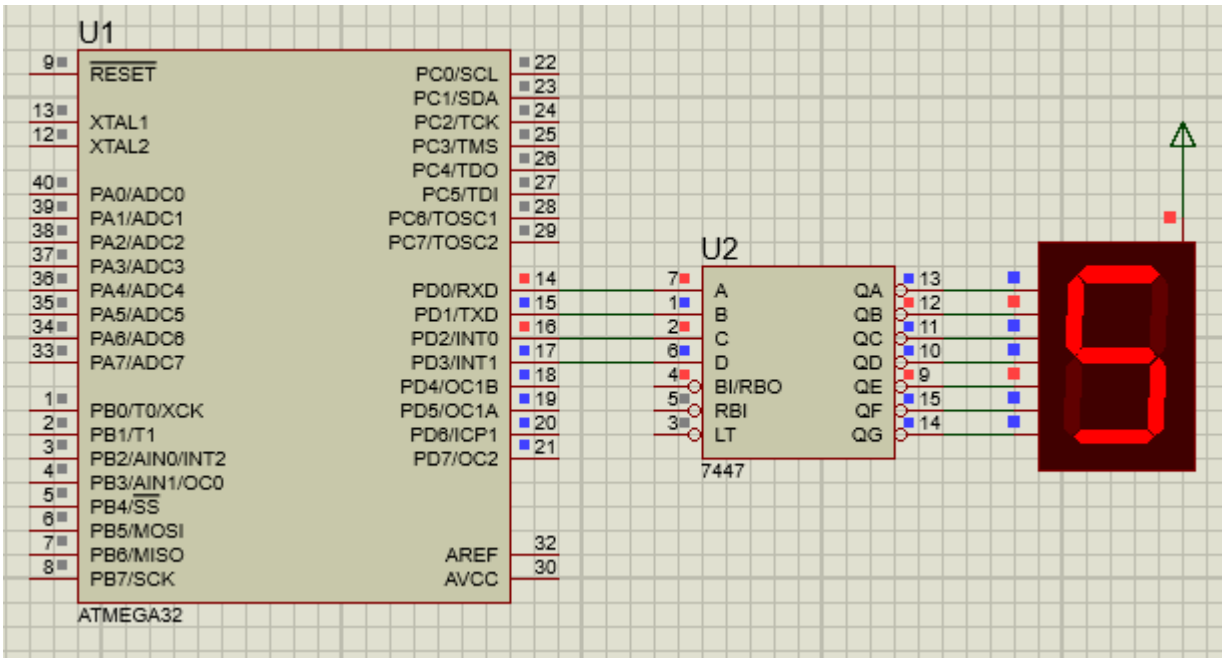
شکل ۳ سون‌سگمنت کاتد مشترک

```
#include <mega32.h>

void main (void)
 {
 //har 8 paye PORTD be surate khoruji tarif mishavad
 DDRD=0xff;
 //ba tavajoh be jadvale 1 mitavan har shomare ra bar ruye 7segment namayesh dad
 PORTD=0x92; //adade 5 bar ruye 7segment namayesh dade mishavad

 while (1)
  {
  // Place your code here

  }
}
```

مثال بالا یک مدار ساده برای روشن و راه‌اندازی یک سون سگمنت آند مشترک است. دقت شود که پورتی که سون سگمنت به آن متصل می‌شود را به‌عنوان خروجی در نظر بگیریم.

می‌توان به‌جای این‌که تک‌تک هر عدد را به‌صورت کد دربیاورید از یک تراشه به نام ۷۴۴۷ که ۱ مبدل به سونBCD ،سگمنت آند مشترک است استفاده کنیم لذا هم کار با آن ساده‌تر است و هم خطوط برنامه‌نویسی کمتری داریم و نیاز به کد کردن به‌صورت دستی نداریم علاوه بر آن برای نمایش هر رقم به چهار پورت خروجی نیاز خواهیم داشت. این آی سی

کار را برای ما بسیار ساده می‌کند. از طریق این مبدل کافی است در ورودی عدد دلخواه برفرض ۵ را بدهیم و در خروجی آن کد این عدد را که مربوط به سون سگمنت است را بگیریم. برای راه‌اندازی سون سگمنت‌های کاتد مشترک از مدل ۷۴۴۸ .استفاده می‌شودIC شکل زیر نحوه اتصال ۷۴۴۷ را به میکرو و یک سون سگمنت آند مشترک نشان می‌دهد.

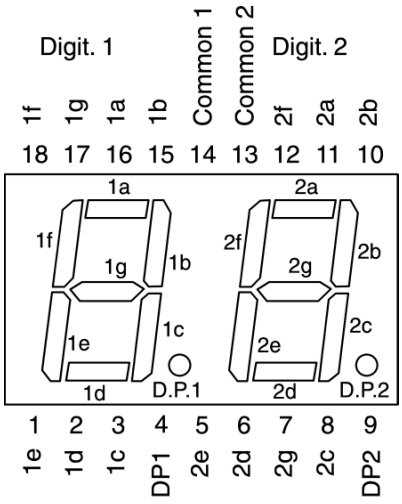


شکل ۴-۱

کد نوشته‌شده برای سخت‌افزار شکل ۴ به شرح زیر هست:

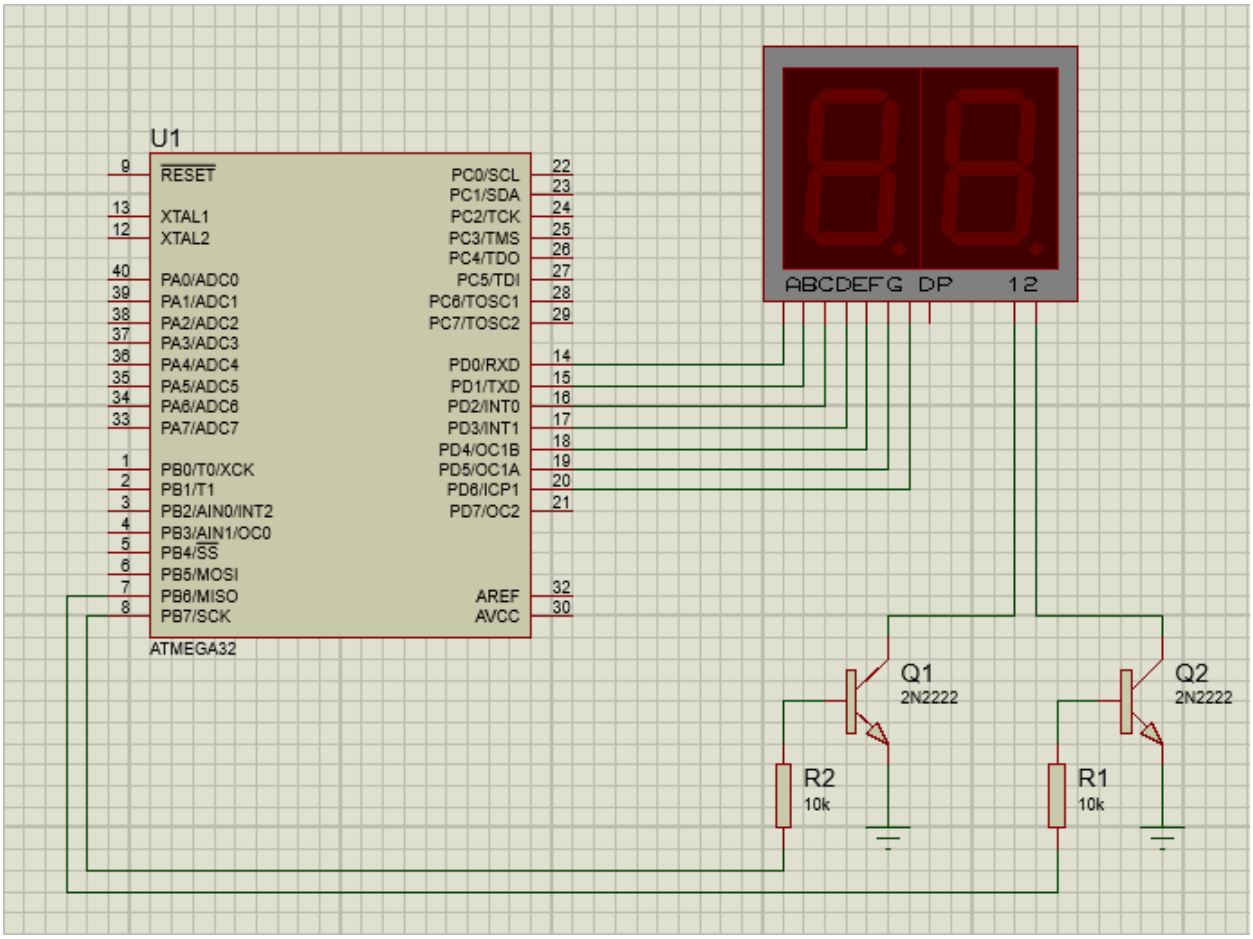
```
#include <mega32.h>

void main (void)
 {
 //har 8 paye PORTD be surate khoruji tarif mishavad
 DDRD=0xff;
 /*ba tavajoh be estefade az 7447, har adad ke bar ruye PORTD gharar dahim,
 bar ruye 7segment amayesh dade mishavad */
 PORTD=0x05; //adade 5 bar ruye 7segment namayesh dade mishavad

 while (1)
  {
  // Place your code here

  }
}
```

تمرین ۱-۱: برنامه یک شمارنده تک‌رقمی را بنویسید به‌گونه‌ای که از ۰ تا ۹ را به‌صورت صعودی شمارش کند و بعد از عدد ۹ دوباره از ۰ شروع به کار نماید.

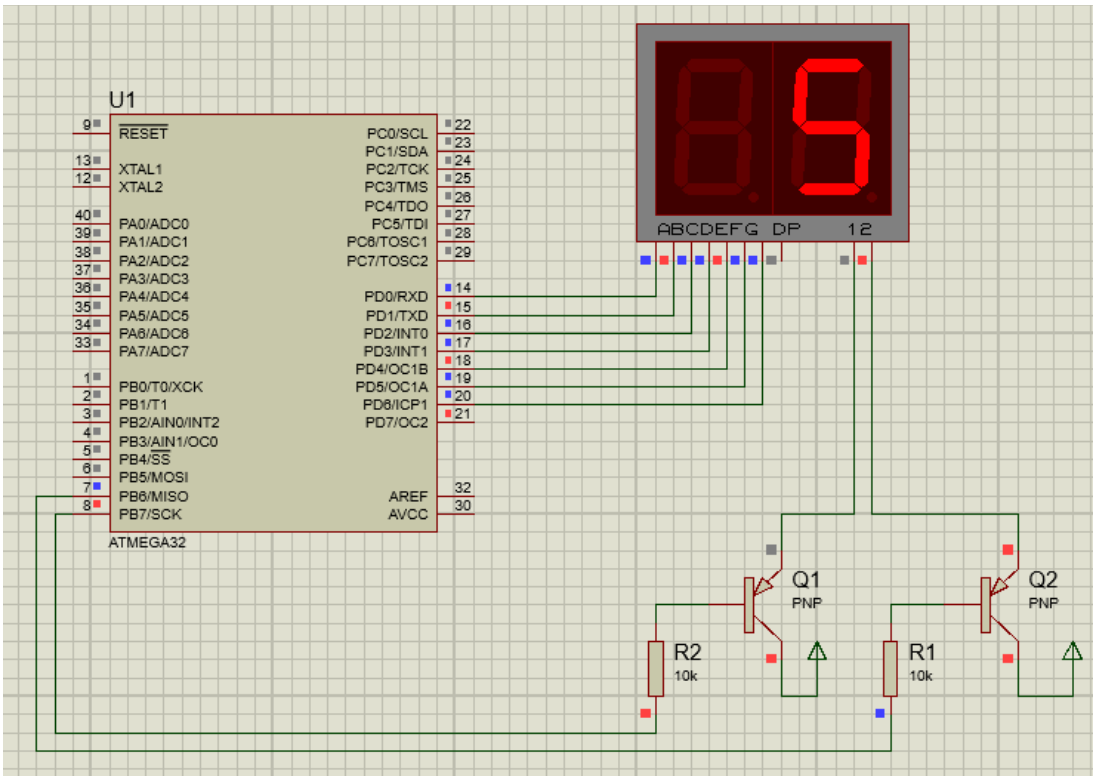


شکل ۱-۵: ترتیب پایه‌های سون‌سگمنت دو بلاکه با پایه‌های مجزا

مطابق با شکل ۶، برای تأمین جریان موردنظر برای سون سگمنت‌ها باید پایه‌های کنترلی (مشترک) آن‌ها را با ترانزیستور درایو کرد چون جریان کلیه سگمنت ها از این پایه عبور می‌کند و میکروکنترلر این مقدار جریان را نمی‌تواند تأمین کند. در شکل ۶ با فعال کردن هر پایه (یک کردن آن) یک رقم نمایش داده خواهد شد و اگر با سرعت‌بالا ۱۵ بار در ثانیه رقم‌ها را یک‌به‌یک فعال کنیم هر دو عدد ثابت دیده خواهند شد. ابتدا رقم اول را از طریق پورت D ارسال کنید و پایه ۱ را فعال کنید و سپس هر دو را غیرفعال کرده و داده رقم دوم را بفرستید و پایه ۲ را فعال کنید و این کار را حداقل ۱۵ بار در ثانیه تکرار کنید.



شکل ۶-۱: نحوه اتصال سون سگمنت دو بلاک کاتد مشترک



شکل ۷-۱: نحوه اتصال سون سگمنت دو بلاک آند مشترک

تمرین ۲-۱: مطابق با شکل ۶ یا ۷، برنامه یک شمارنده دورقمی را بنویسید به‌گونه‌ای که از ۰ تا ۹۹ را به‌صورت صعودی شمارش کند و بعد از عدد ۹۹ دوباره از ۰ شروع به کار نماید.

آزمایش دوم: راه‌اندازی صفحه‌کلید ۴×۴

هدف:

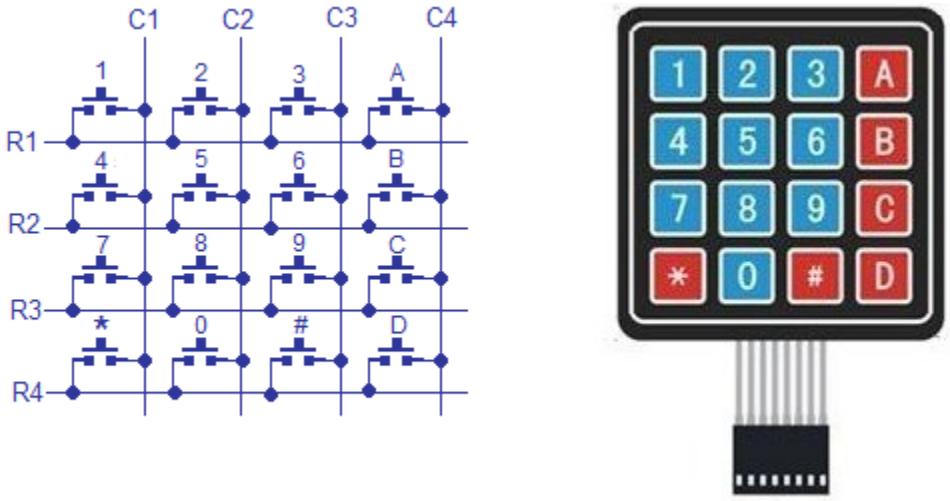
۱- آشنایی با پورت‌ها در میکروکنترلر
۲- آشنایی با صفحه‌کلید

قطعات موردنیاز:

۱- صفحه‌کلید ۴×۴
۲- مقاومت ۱۰ کیلو اهم ---------- ۴ عدد

شرح آزمایش:

صفحه‌کلیدها شامل یکسری سطر و ستون می‌باشند که با فشار هر کلید یک سطر به یک ستون متصل می‌شود. برای خواندن صفحه‌کلید توسط میکروکنترلر، معمولاً از روش اسکن صفحه‌کلید برای تشخیص کلید فشرده‌شده استفاده می‌شود.

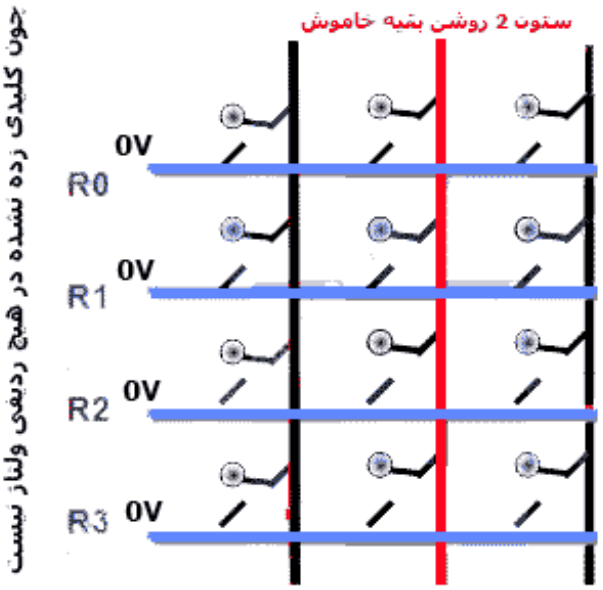


شکل ۱-۲: صفحه‌کلید ۴×۴ و نمایش نحوه سیم‌کشی داخلی آن

نحوه اسکن کردن کیپد و به دست آوردن شماره کلید زده‌شده:

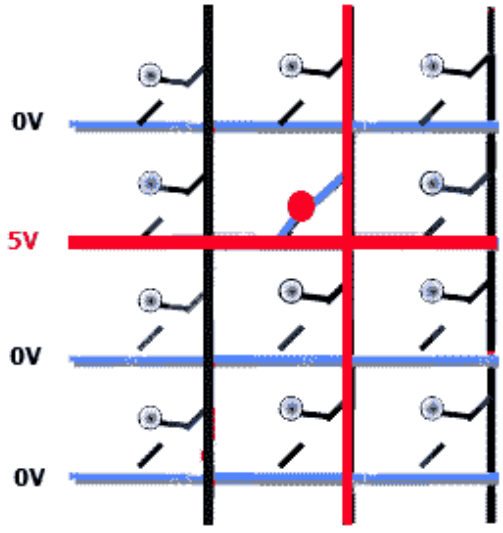
برای اسکن صفحه‌کلید باید سطر یا ستون را ورودی در نظر گرفته و دیگری را خروجی در اینجا ستون را ورودی صفحه‌کلید در نظر می‌گیریم و ردیف‌ها را خروجی مدار صفحه‌کلید، حال به روش زیر عمل می‌کنیم:

حال باید به ترتیب به ستون یک الی ۴ ولتاژ وصل کنیم و چک کنیم که آیا در خروجی صفحه‌کلید ولتاژی هست یا نه درصورتی‌که در خروجی ولتاژی وجود داشته باشد به معنی این است که کلیدی فشرده‌شده است و باید با توجه به‌ردیف شماره کلید را به دست آورد به شکل زیر نگاه کنید:



شکل ۲-۲

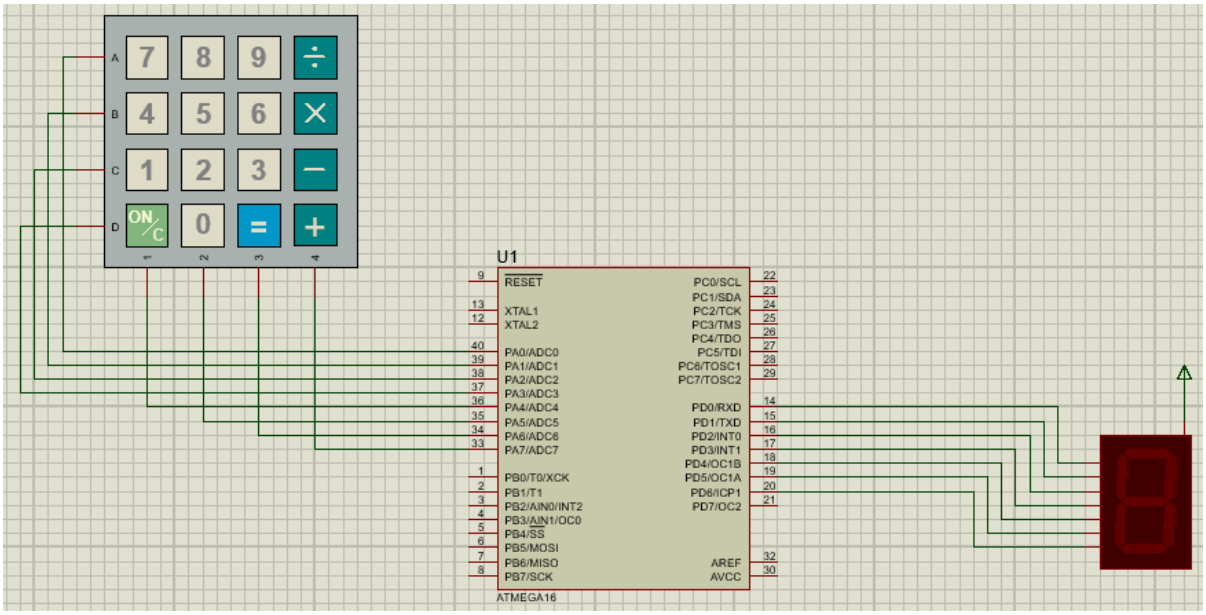
وقتی یکی از کلیدها زده شود مدار به شکل زیر تغییر کرده و دریکی از ردیف‌ها ولتاژ وجود خواهد داشت:



شکل ۳-۲

توضیح: درصورتی‌که از صفحه‌کلید شکل ۱ استفاده می‌کنید می‌توانید آن را با استفاده از پین هدر نری، داخل برد بورد قرار داده و مستقیم به میکروکنترلر متصل کنید.

نحوه اتصال کیپد به میکروکنترلر در شکل زیر نشان داده‌شده است:



شکل ۴-۲: شماتیک مدار

بعد از بستن سخت‌افزار سراغ نرم‌افزار آن در محیط کدویژن می‌رویم بعد از تنظیمات کدویزارد و باز شدن برنامه کدها و کتابخانه‌ها و توابع لازم را به آن اضافه می‌کنیم.

کد برنامه برای اسکن صفحه‌کلید به شکل زیر نوشته می‌شود:

```
unsigned char read_key()
{
    char data;

    PORTA=0b11101111;
    DDRA=0b00010000;
    delay_ms(10);
    data=PINA&0x0F;
    if(data==0b00001110)
        return 1;
    if(data==0b00001101)
        return 4;
    if(data==0b00001011)
        return 7;
    if(data==0b00000111)
        return 10;
    //====================================
    PORTA=0b11011111;
    DDRA=0b00100000;
    delay_ms(10);
    data=PINA&0x0F;
    if(data==0b00001110)
        return 2;
    if(data==0b00001101)
        return 5;
    if(data==0b00001011)
        return 8;
    if(data==0b00000111)
        return 0;
    //====================================
    PORTA=0b10111111;
    DDRA=0b01000000;
    delay_ms(10);
    data=PINA&0x0F;
    if(data==0b00001110)
        return 3;
    if(data==0b00001101)
        return 6;
    if(data==0b00001011)
        return 9;
    if(data==0b00000111)
        return 10;
    //====================================
    return 0xff;
    delay_ms(50);
}
```

در برنامه main کافی است با فراخوانی زیر تابع read_key، صفحه‌کلید چک شده و در صورت فشرده شدن کلید از کیپد، کد آن کلید توسط زیر برنامه برگشت داده شود. برای فراخوانی کافیست تا تنها تابع read_key به شکل زیر صدا زده شود:

w=read_key();

توسط این دستور، کد کلید زده‌شده در متغیر w قرار می‌گیرد. درصورتی‌که کلیدی فشرده نشده باشد کد FF توسط تابع برگردانده می‌شود.

تمرین ۱-۲: زیر برنامه read_key را با توجه به توضیحات داده‌شده تحلیل کرده و نحوه عملکرد آن را توضیح دهید.

تمرین ۲-۲: برنامه‌ای بنویسید که با استفاده از تابع read_key صفحه‌کلید را اسکن کرده و در صورت فشرده شدن کلید ۹-۰، آن را بر روی سون‌سگمنت نمایش دهد.

سؤال: در صورت فشرده شدن دو کلید هم‌زمان، انتظار دارید چه اتفاقی بیوفتد؟ به‌صورت عملی این اتفاق را تست کنید، آیا حدس شما درست بود؟ دلیل اتفاقی که می‌افتد را بنویسید.

آزمایش سوم: راه‌اندازی LCD۲×۱۶

هدف:

۱- آشنایی با نمایشگرها
۲- آشنایی انواع درایورهای LCD

قطعات موردنیاز:

۱- نمایشگر کاراکتری ۲×۱۶
۲- پین هدر نری تک ردیفه ---------- یک عدد ۴۰ تایی
۳- پتانسیومتر ۱ کیلو ---------- یک عدد

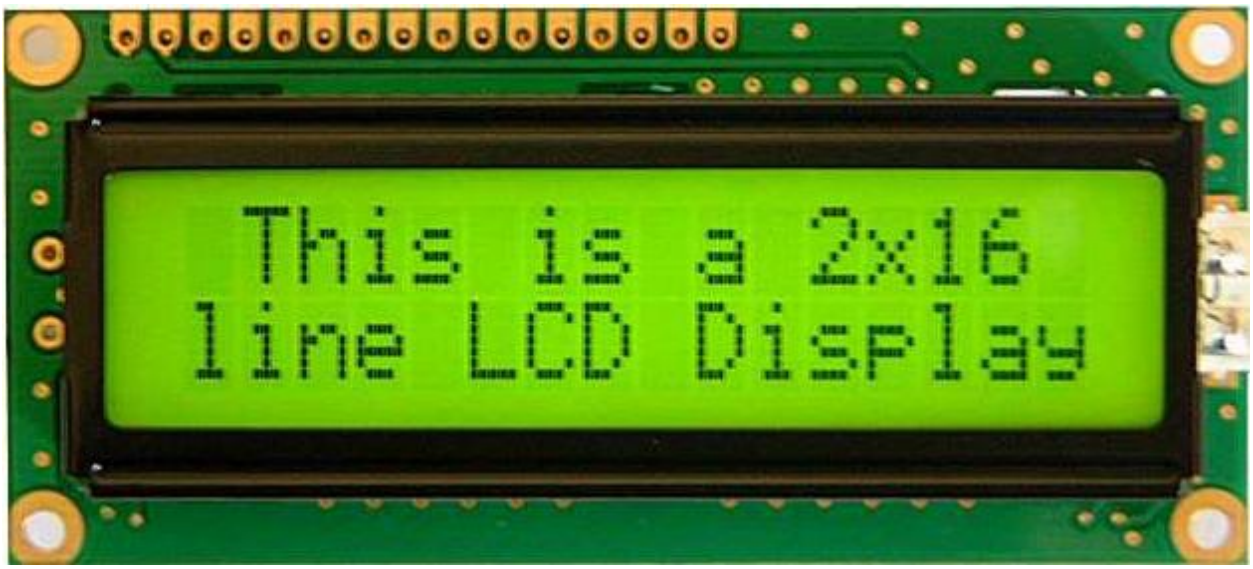
شرح آزمایش:

نمایشگرها در ایجاد ارتباط بین دنیای انسان‌ها و دنیای ماشین‌ها نقش مهمی را ایفا کرده و در سیستم‌های تعبیه‌شده نیز از بخش‌های بااهمیت هستند. نمایشگرها مستقل از این‌که کوچک یا بزرگ باشند دارای اصول یکسانی در کار کردن هستند. شخصی که در حوزه‌ی الکترونیک فعال است باید حداقل کار کردن با نمایشگرهای ساده‌ای مانند ال سی دی کاراکتری را بداند. در سیستم‌های تعبیه‌شده نمایشگرهای زیادی مانند نمایشگرهای گرافیکی یا نمایشگرهای سه‌بعدی مورداستفاده قرار می‌گیرند. LCD یکی از نمایشگرهای حرفه‌ای است که در این سیستم‌ها مورداستفاده قرار می‌گیرد. هنگامی‌که نحوه‌ی کار با ال سی دی را یاد گرفتید، خواهید دید که این نمایشگر یکی از آسان‌ترین و مورد اعتمادترین دستگاه‌های خروجی است! تا جایی که در پروژه‌های مبتنی بر میکروکنترلر با توجه به این‌که همیشه نمی‌توان از دیباگر استفاده کرد، ال سی دی برای تست خروجی‌ها بسیار کارآمد است. البته برای این کار نیاز است که به‌خوبی قادر به کار با این نمایشگرها باشید؛ و این امر با دنبال کردن این پست برایتان میسر خواهد شد.

LCD در مقایسه با LED و سون سگمنت برای نمایش اطلاعات به فرم حروف، اعداد، کاراکتر و یا پویانمایی‌ها ترجیح داده می‌شود. برنامه‌نویسی ال سی دی‌ها بسیار ساده بوده و کار را سریع و جذاب می‌کنند. این امر بدین دلیل است که ال سی دی ازلحاظ اقتصادی مقرون‌به‌صرفه بوده؛ به‌سادگی قابل‌برنامه‌ریزی است و برخلاف سون سگمنت محدودیتی در نمایش کاراکترهای خاص و پویانمایی‌ها ندارد.

LCDکاراکتری ۲×۱۶

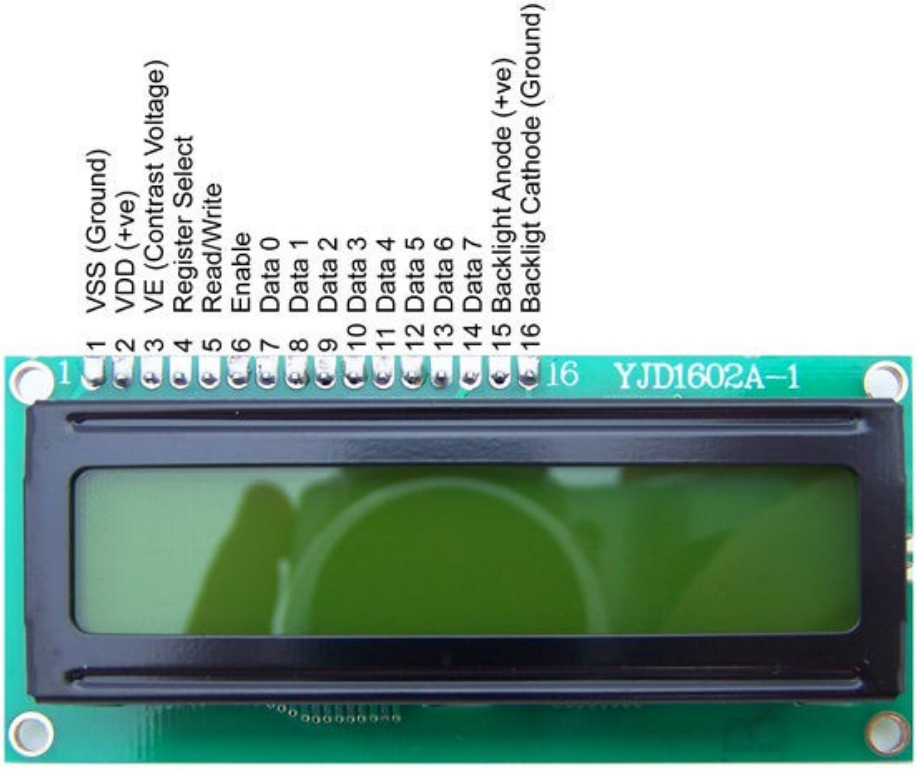
ال سی دی که ما می‌خواهیم از آن استفاده کنیم ال سی دی کاراکتری ۲×۱۶ است که قابلیت نمایش ۳۲ کاراکتر را در دو ردیف ۱۶ تایی دارا می‌باشد؛ یعنی در هر ردیف ۱۶ کاراکتر را می‌تواند نمایش دهد. شکل ۱ یک LCD ۲×۱۶ را نشان می‌دهد:



شکل ۱-۳: نمایش LCD ۲×۱۶

پین‌های ماژولLCD

ماژول ال سی دی دارای ۱۶ پین می‌باشد که هر پین برای وظیفه‌ای خاص تعبیه‌شده است. شماره بندی پین‌ها به صورتی که در شکل ۲ نشان داده‌شده است، صورت می‌پذیرد.



شکل ۲-۳: پایه‌های LCD۲×۱۶

در ادامه کاربرد هر پین به ترتیب شماره‌گذاری بالا توضیح داده خواهد شد:

جدول ۱-۳: پایه‌های LCD

Pin	Symbols and functions
1	GND
2	VCC (+5v)
3	Contrast adjust
4	(RS) ==>> 0 = Instruction input / 1 = Data input
5	(R/W) ==>> 0 = Write to LCD Module / 1 = Read from LCD module
6	(E) ==>> Enable signal
7	(DB0) ==>> Data Pin 0
8	(DB1) ==>> Data Pin 1
9	(DB2) ==>> Data Pin 2
10	(DB3) ==>> Data Pin 3
11	(DB4) ==>> Data Pin 4
12	(DB5) ==>> Data Pin 5
13	(DB6) ==>> Data Pin 6
14	(DB7) ==>> Data Pin 7
15	(VB+) ==>> back light (+5V)
16	(VB-) ==>> back light (GND)

پین‌های VSS، VDD و VEE

این سه پین به ترتیب دارای شماره‌های ۱، ۲ و ۳ می‌باشند. پین ۱ (VSS) پین زمین بوده و باید زمین شود تا LCD به‌خوبی کار کند. معمولاً پین‌های VEE و VDD به ولتاژهای ۵ ولت متصل می‌شوند. بدین‌صورت که پین VDD حتماً باید همیشه به ولتاژ ۵ ولت وصل باشد ولی پین VEE می‌تواند به‌جای این‌که مستقیماً به ولتاژ ۵ وصل گردد قبل از آن به پتانسیومتری متصل شود تا کنتراست و وضوح LCD قابل تنظیم باشد.

پین‌های Register Select (یا RS)، Read/Write (یا R/W) و Enable (یا E)

این سه پین نیز به ترتیب با شماره‌های ۴، ۵ و ۶ مشخص می‌شوند. پایه‌ی RS برای انتخاب بین رجیسترهای داده و دستور به کار می‌رود. وقتی RS=0 رجیستر دستور انتخاب می‌شود و هنگامی‌که RS=1 رجیستر داده انتخاب می‌گردد؛ و پین R/W برای انتخاب خواندن یا نوشتن است، به‌طوری‌که اگر قصد خواندن رجیستری را داشته باشیم باید R/W=1 و اگر تصمیم برنوشتن مقداری روی رجیستر داریم باید R/W=0 باشد.

پین‌های Data یعنی از DB0 تا DB7

این پین‌ها برای انتقال داده‌های هشت بیتی هستند که برای ارسال اطلاعات به LCD و یا خواندن محتویات رجیستر درونی LCD به کار می‌روند.

پین‌های آند و کاتد برای نور زمینه

پین ۱۵ و ۱۶ به ترتیب قطب آند و کاتد LED هستند که نور زمینه‌ی LCD را فراهم می‌کند و باید به ترتیب به ولتاژ ۵ ولت و زمین وصل گردند.

مُدهای کاریLCD

مد ۸ بیتی:

در مد ۸ بیتی پین‌های شماره ۷ تا ۱۴ از ماژول LCD به ۸ پین ورودی/خروجی میکروکنترلر متصل می‌شوند. درنتیجه در این مد ما برای تبادل اطلاعات نیاز به ۸ پین داریم. مزیت این مد در این است که برنامه‌نویسی برای این مد آسان بوده و داده‌ها به‌سرعت آپدیت می‌شوند.

مد ۴ بیتی:

در مد ۴ بیتی پین‌های شماره‌ی ۱۱ تا ۱۴ از ماژول LCD به چهار پین ورودی/خروجی میکروکنترلر وصل می‌گردند. ازاین‌رو در این مد برای تبادل داده فقط به ۴ پین نیاز داریم. دلیل اصلی استفاده از مد ۴ بیتی استفاده از ۴ پایه‌ی میکروکنترلر به‌جای استفاده از ۸ پایه است.

چند تابع پرکاربرد نرم‌افزار کدویژن مربوط بهLCD

تابع (lcd_init)

قبل از هر کار دیگر، هنگامی‌که نیاز به استفاده از ال سی دی داریم باید این تابع را استفاده نماییم. ورودی این تابع نوع تعداد ستون‌های ال سی دی مورداستفاده می‌باشد. به‌عنوان‌مثال در اینجا ۱۶ می‌باشد. این تابع ال سی دی را راه‌اندازی نموده و صفحه‌ی آن را پاک‌کرده و نشانگر را به سطر و ستون صفر می‌برد.

تابع (lcd_clear)

این تابع صفحه‌ی نمایش را پاک‌کرده و نشانگر را به سطر و ستون می‌برد.

تابع (lcd_gotoxy)

این تابع نشانگر را به مکان داده‌شده می‌برد. ورودی‌های این تابع دو عدد بی علامت هستند که عدد اول شماره‌ی ستون و عدد دوم شماره سطر موردنظر است.

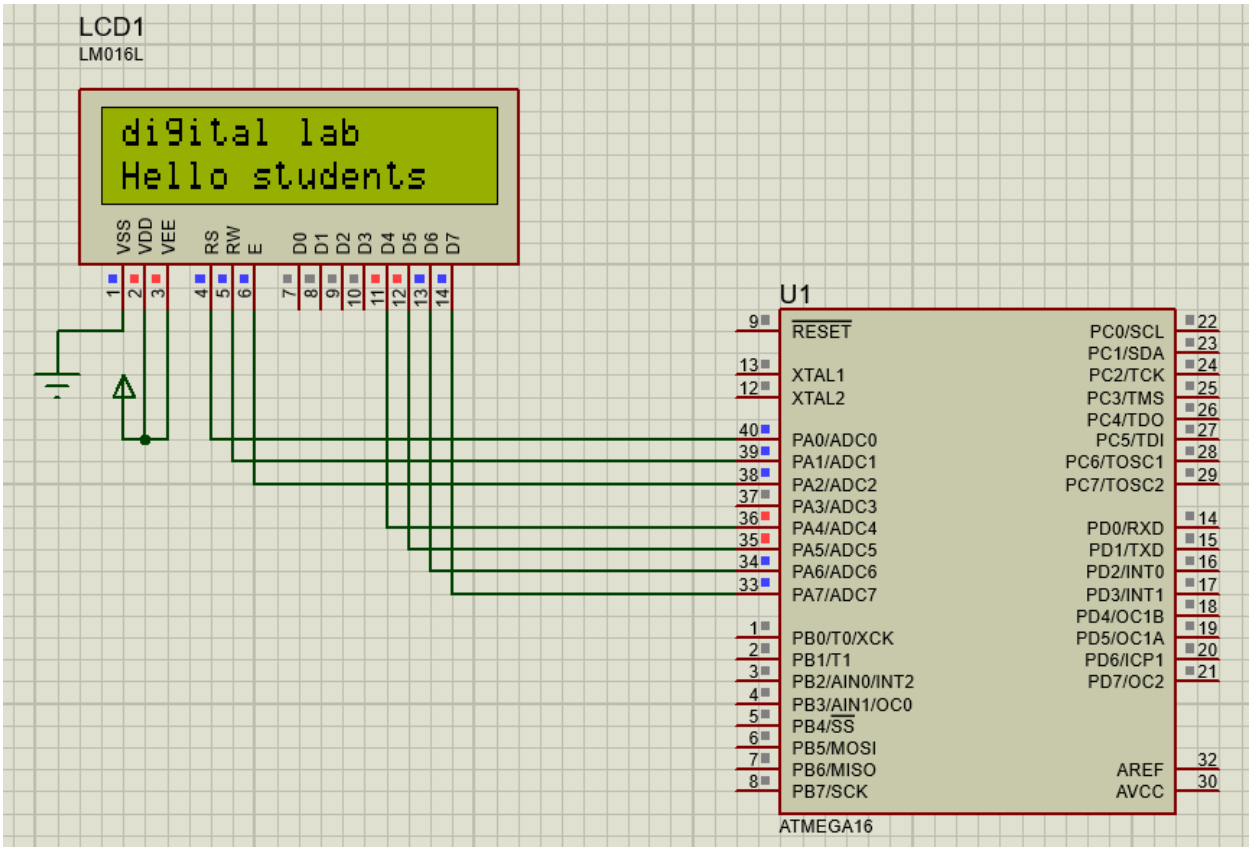
تابع (lcd_putchar)

این تابع کاراکتر موردنظر را نمایش می‌دهد. ورودی این تابع کد اسکی کاراکتری است که می‌خواهیم نمایش دهیم.

تابع)lcd_puts(

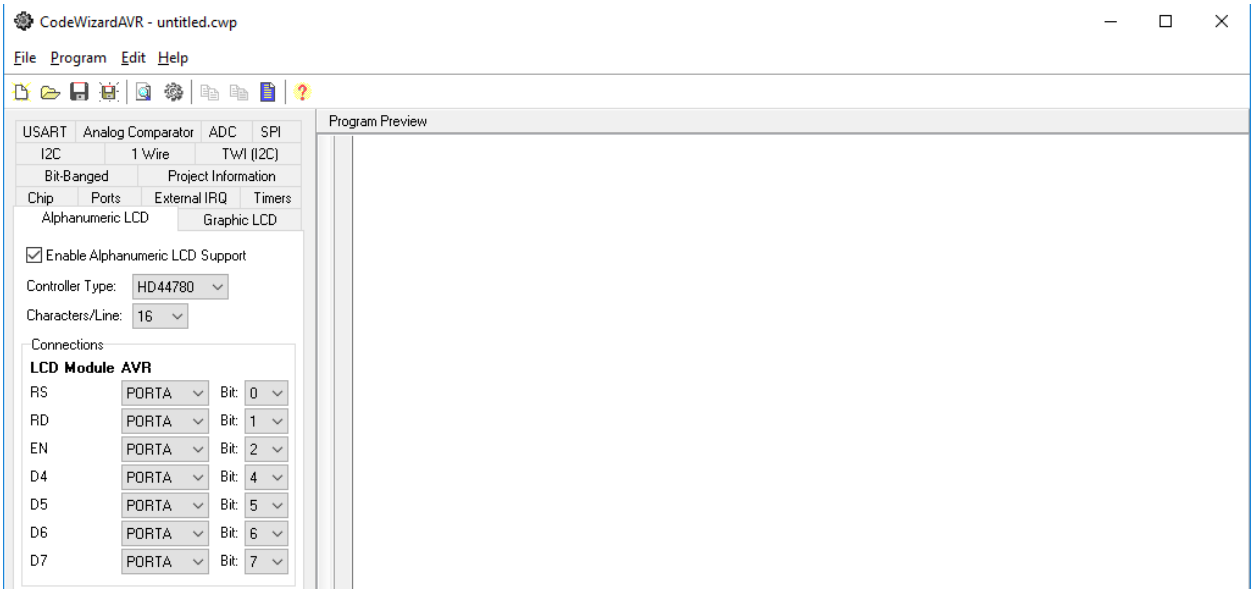
این تابع رشته‌ی موردنظر از روی ال سی دی نمایش می‌دهد. ورودی این تابع یک‌رشته می‌باشد.

نحوه اتصال نمایشگر به میکروکنترلر در شکل ۳ آمده است.



شکل ۳-۳

تنظیمات codewizard برای راه‌اندازی LCD به‌صورت زیر است.



شکل ۳-۴

تمرین ۱-۳: سخت‌افزار معرفی‌شده در دستور کار را آماده کرده و برنامه‌ای بنویسید که عبارت "Hello world" بر روی خط اول و نام شما در خط دوم نمایش داده شود.

تمرین ۲-۳: با توجه به آزمایش کیپد، صفحه‌کلید ۴×۴ را به پورت دیگر میکروکنترلر متصل کرده و برنامه‌ای بنویسید که نام شما را در خط اول و کلید فشرده‌شده توسط صفحه‌کلید را در خط دوم نمایش دهد.

تمرین ۳-۳: برنامه یک ماشین‌حساب بنویسد که ۴ عمل اصلی را انجام دهد.

آزمایش چهارم: راه‌اندازی ارتباط سریال

هدف:

۱- آشنایی با ارتباط سریال
۲- آشنایی با آی‌سی Max232
۳- آشنایی با وقفه‌ها
۴- معرفی پروتکل‌های ارتباطی سریال مانند RS232، RS485، CAN

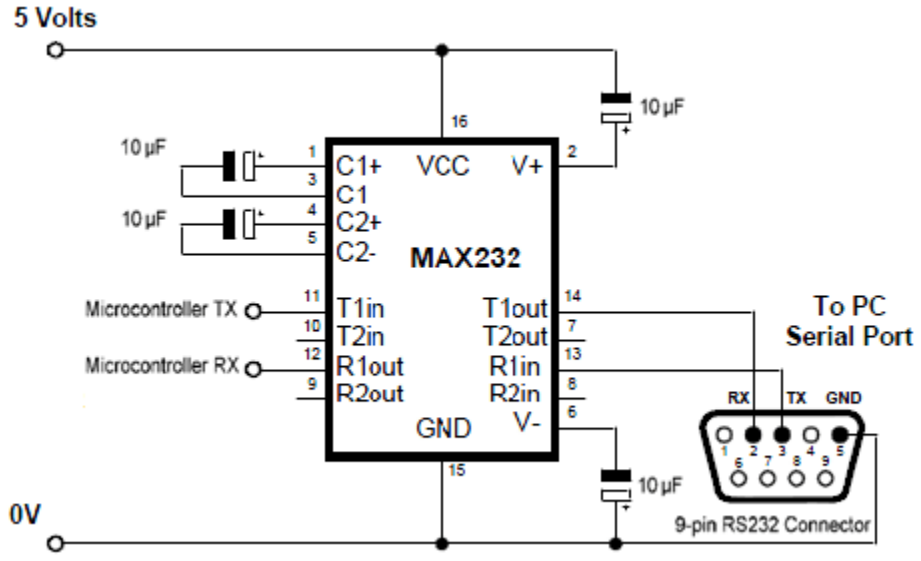
قطعات موردنیاز:

۱- MAX232 ---------- ۱ عدد
۲- خازن1uf (6.3v به بالا) ---------- ۵ عدد
۳- کابل ۳ رشته‌ای ---------- ۱/۵ متر
۴- کانکتور DB09 مادگی + قاب ------- ۱ عدد

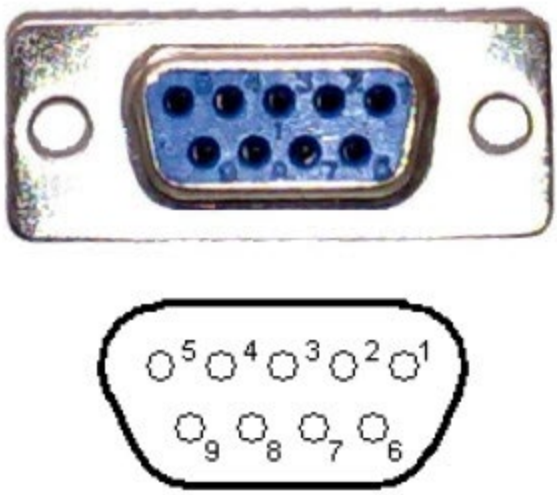
شرح آزمایش:

راه‌اندازی آی‌سیMAX232

این آی‌سی جهت ارتباط میکروکنترلر با پورت سریال کامپیوتر می‌باشد. طبق شکل ۱ باید این آی‌سی را به پورت سریال کامپیوتر وصل نمود. این آی‌سی با پروتکل USART با میکروکنترلر ارتباط برقرار می‌کند.



شکل ۱-۴



شکل ۲-۴: نمای پورت سریال DB09

پروتکل UART یک پروتکل سریال می‌باشد و فرق آن با پروتکل سریال RS232 این است که ولتاژ آن بین صفر ولت (برای عدد صفر) تا VCC (برای عدد یک) است.

در حال حاضر تمام میکروکنترلرهای موجود اعم از AVR, ARM و بردهای آدرینو حداقل یک درگاه USART و UART را دارا می‌باشند. در عین سادگی استفاده از پورت سریال می‌بایست یک سری اصول اصلی را در مورد آن بدانیم.

انواع ارتباط سریال:

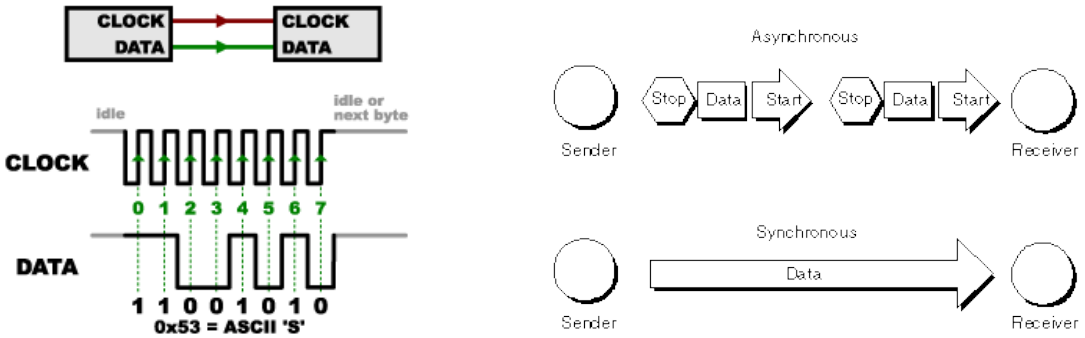
۱ - ارتباطه simplex (یک‌طرفه): فرستنده فقط ارسال می‌کند و گیرنده فقط دریافت می‌کند.

۲ – ارتباط half duplex (نیم دوطرفه): هر دستگاه می‌تواند هم فرستنده باشد و هم گیرنده ولی ارسال و دریافت هم‌زمان صورت نمی‌گیرد.

۳ – ارتباط full duplex (دوطرفه): هر دستگاه می‌تواند هم فرستنده باشد و هم گیرنده و ارسال و دریافت هم‌زمان صورت می‌گیرد.

تقسیم‌بندی دیگری وجود دارد که ارتباط سریال را به دودسته تقسیم می‌کند: سنکرون (synchronous) – آسنکرون (asynchronous)

به شکل ۳ دقت کنید:

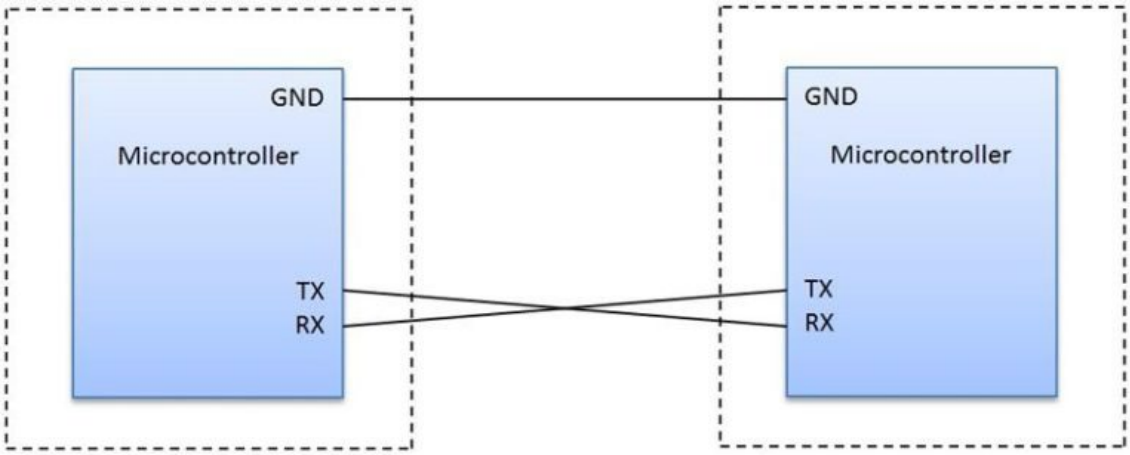


شکل ۳-۴

ارتباط سنکرون: در ارتباط سنکرون علاوه بر خط data یک خط کلاک هم وجود دارد. به ازای هر تغییر پالس بالارونده وضعیت یک bit را از روی خط data می‌خواند. باید به این نکته نیز توجه شود که احتمال وجود خطا در ارتباط سنکرون کمتر از اسنکرون می‌باشد.

ارتباط آسنکرون: در ارتباط آسنکرون دیگر خبری از خط clock نیست و انتقال دیتا بر اساس baud rate انجام می‌شود که به این منظور باید یک سری دسته‌بندی خاص در frame ارسالی وجود داشته باشد. سرعت انتقال دیتا در حالت سنکرون بیشتر از حالت اسنکرون می‌باشد.

UART: (**Universal Asynchronous Receiver / Transmitter**) این ارتباط از نوع آسنکرون می‌باشد.



شکل ۴-۴

این نوع ارتباط یکی از معروف و پرکاربردترین نوع ارتباط است. اکثر ماژول‌ها اعم از Bluetooth, ZigBee's, GPS و ... از این نوع ارتباط استفاده می‌کنند. تفاوت دیگر سنکرون و آسنکرون در نوع داده‌های ارسالی می‌باشد. بعضی از این بیت‌ها داده نیستند مثل start و stop و parity این بیت‌ها و قوانین ارسالی به‌منظور برقراری یک ارتباط امن خالی از هرگونه خطا می‌باشد. حال دراین‌باره بیشتر بحث خواهد شد به شکل ۵ دقت کنید:

شکل ۴-۵

شکل ۵ قالب bit های ارسالی در ارتباط UART را نشان می‌دهد. تذکر مهمی در این قسمت وجود دارد و در اصل تعریف یک پروتکل وجود دارد قوانین ارسال در فرستنده و گیرنده باید یکسان باشد. مهم‌ترین پارامتری که باید در فرستنده و گیرنده یکسان باشد نرخ انتقال دیتا (baud rate) است.

baud rate: سؤالی که حال در ذهن شما شکل‌گرفته که انتقال دیتا به چه سرعتی انجام می‌شود؟در ارتباط سریال پارامتری که میزان سرعت انتقال دیتا را نشان می‌دهد baud rate می‌باشد. برای مثال هنگامی‌که گفته می‌شود baud rate رو ۹۶۰۰ تنظیم‌شده است یعنی ۹۶۰۰ بیت در هر ثانیه انتقال دیتا صورت گرفته و واحد آن bit per second (bps) می‌باشد. نکته‌ی قابل‌ملاحظه‌ای که وجود دارد این است که نیازی نیست کلاک داخلی دو میکروکنترلری که قرار است باهم انتقال دیتا انجام دهند، باهم برابر باشد، شما فقط باید baud rate هرکدام را برقرار کنید.

baud rate ها مقداری از قبل مشخص‌شده و استاندارد می‌باشند. این مقادیر تقسیم‌هایی از اسیلاتور اصلی می‌باشد. این مقادیر عبارت‌اند از: ۱۲۰۰, ۲۴۰۰, ۴۸۰۰, ۱۹۲۰۰, ۳۸۴۰۰, ۵۷۶۰۰, ۱۱۵۲۰۰ البته مقدار ۹۶۰۰ از جایگاه ویژه‌ای برخوردار است و اکثر ماژول‌های موجود در بازار از این baud rate استفاده می‌کنند.

هر بلوک از دیتا در UART (به‌اصطلاح به این بلوک بایت byte می‌گویند) در قالب frame هایی انتقال می‌یابد. بخش اصلی این frame قسمت دیتا می‌باشد که از ۵ تا ۹ bit قابل تنظیم است به‌طور استاندارد روی ۸ بیت به‌عنوان بایت تنظیم می‌شود ولی برای فرستادن کد ASCII هفت بیت نیز کافی و مؤثر می‌باشد.

همان‌طور که مشاهده می‌شود آغاز ارسال اطلاعات باید از یک بیت شروع شود، طبق قوانین این پروتکل به حرکت bit start از ۱ به ۰ آغاز انتقال دیتا شروع می‌شود و برای پایان انتقال به حرکت بیت ۰ به ۱ پایان می‌یابد و روی خط انتقالی روی حالت آزاد که به‌اصطلاح idle می‌گویند.

همان‌طور که گفته شد در ارتباط از نوع اسنکرون احتمال خطا در دریافت اطلاعات بیشتر است. روش‌های زیادی در انواع ارتباطات برای مقابله با این مسئله وجود دارد. ولی در ارتباط UART یک بیت به‌عنوان بیت Parity در نظر گرفته‌شده است که به آن بیت توازن نیز گفته می‌شود.

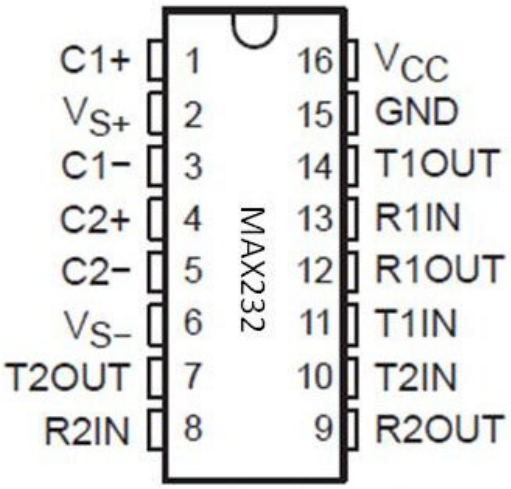
در پروتکل RS232 سطح ولتاژ عدد یک بین 3-ولت تا 15- ولت است و سطح ولتاژ عدد صفر نیز بین ۳ ولت تا ۱۵ ولت می‌باشد.

پروتکل RS232 در کانکتور ۹ پین پشت کامپیوتر (بیشتر در کامپیوترهای قدیمی) وجود دارد. برای ارتباط میکروکنترلر با این پورت باید سطح ولتاژ UART را به RS232 تبدیل نمود. با استفاده از تراشه‌ی MAX232 می‌توان این کار را انجام داد.

برای اینکه داده درست منتقل شود، باید علاوه بر پایه‌های TX و RX که داده را ارسال و دریافت می‌کنند، GND را هم برای هم‌پتانسیل کردن فرستنده و گیرنده وصل نمود تا داده درست منتقل شود.

استاندارد RS232 :

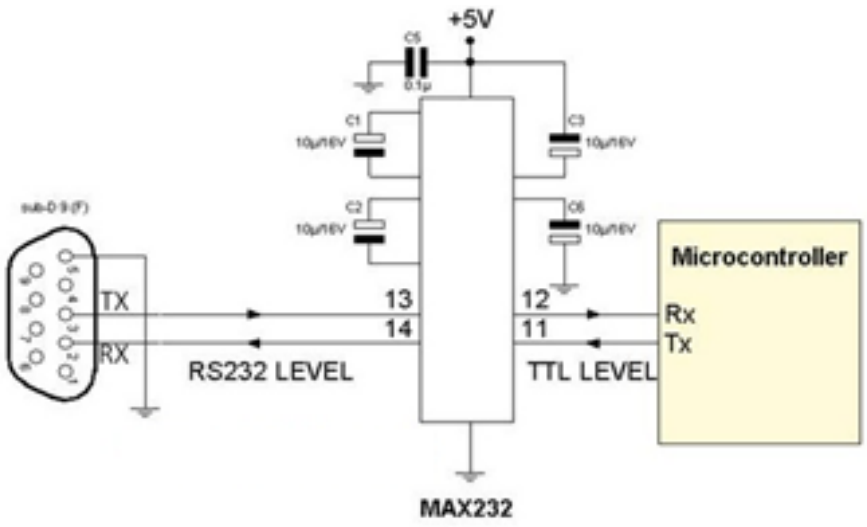
امروزه RS232 یکی از پرکاربردترین استانداردهای اتصال سریال است؛ اما ازآنجاکه این استاندارد خیلی پیش از ظهور خانواده منطقی TLL تنظیم‌شده بود سطوح ولتاژهای ورودی و خروجی آن با TLL سازگار نیستند. در RS232 منطق یک با 3v- تا 25v- و منطق صفر با 3+ v تا 25v+ مشخص می‌شود. فاصله بین 3v- تا 3v+ تعریف‌نشده است. درنتیجه برای اتصال RS232 به یک سیستم میکروکنترلر باید از مبدل‌های ولتاژی مانند MAX232 جهت تبدیل سطوح ولتاژ RS232 به TTL و بالعکس استفاده کنیم (سطح ولتاژ TTL پنج ولت است.)



شکل ۴-۶

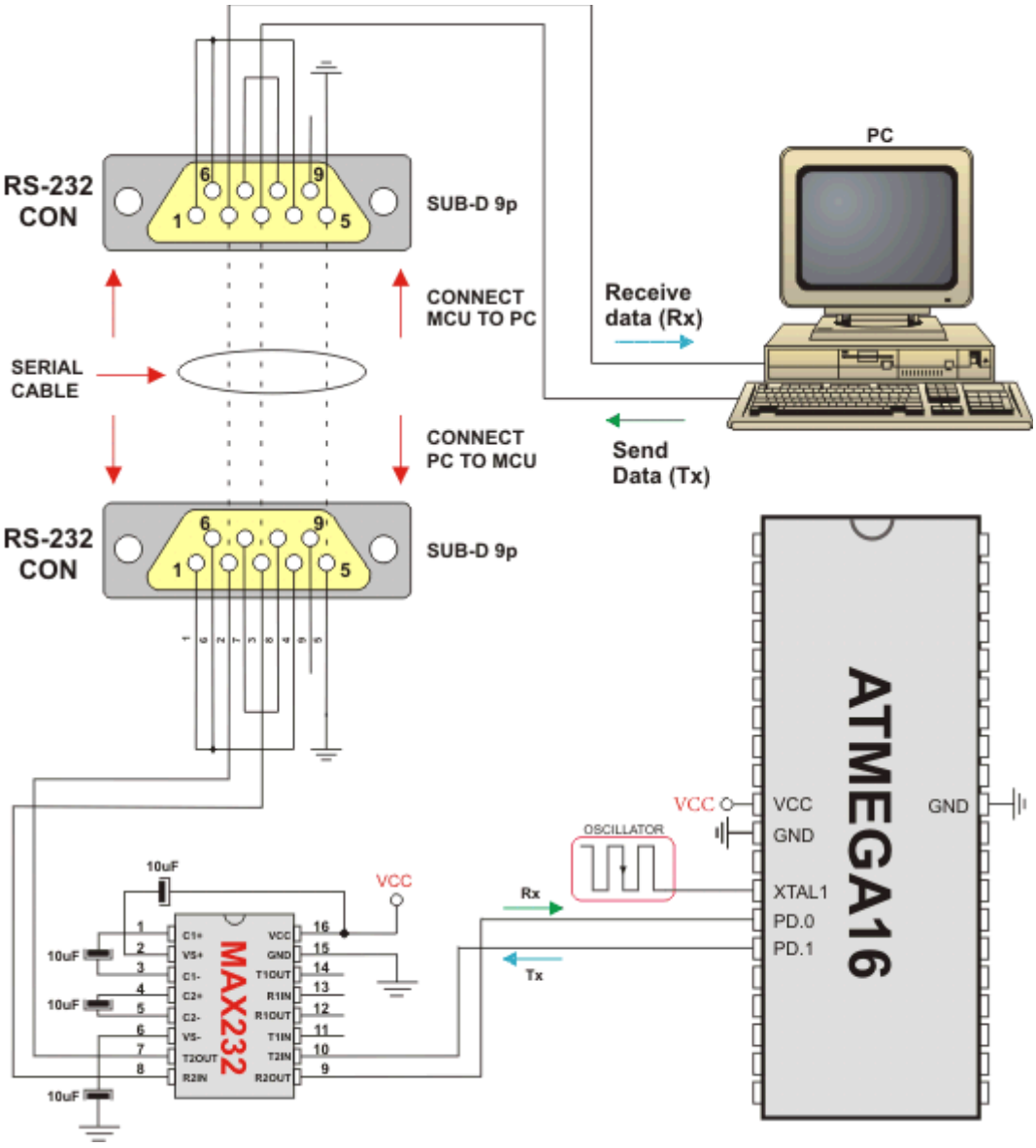
برای آزمایشات ارتباط سریال، پورت سریال AVR را به پورت COM در PC متصل کنیم. در صورت نبودن پورت COM می‌توانیم از یک مبدل COM به USB استفاده کنیم.

ساده‌ترین اتصال AVR به پورت RS232 به‌وسیله سه اتصال امکان‌پذیر است. تنها کافیست پایه TXD میکروکنترلر به پایه RXD پورت RS232و پایه RXD میکروکنترلر به پایه TXD پورت RS232 متصل شود و یا اتصال مشترک GNDهم داشته باشند. این اتصال باید به‌وسیله رابط MAX232 باشد. در شکل ۷ نحوه اتصال را مشاهده می‌کنید.



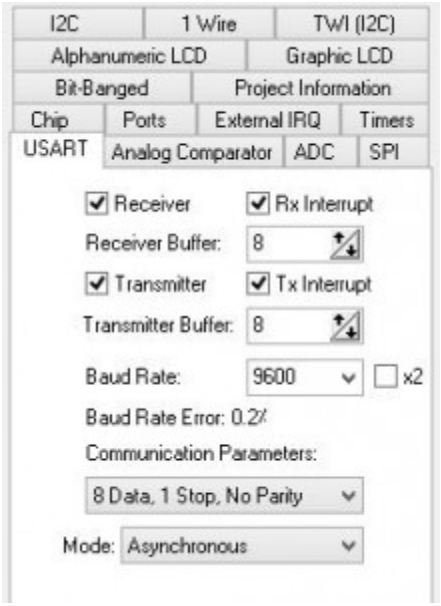
شکل ۷-۴

توصیه می‌شود طول کابلی که برای اتصال پورت سریال به میکرو استفاده می‌شود حدود ۱/۵ متر باشد.



شکل ۸-۴

مطابق شکل ۸ مدار را می‌بندیم. بعد در قسمت نرم‌افزار کدویژن و کدویزارد وقتی پنجره کد ویزارد را باز می‌کنید در قسمت ارتباط سریال تصویری مشابه عکس ۹ را خواهید دید:



شکل ۹-۴

وقتی شما می‌خواهید که میکرو فقط فرستنده باشد، **تیک دوم** (Transmitter) را بزنید و وقتی می‌خواهید میکرو فقط گیرنده باشد **تیک اول** (Receiver) را بزنید. اگر هم فرستنده و گیرنده باشد باید تیک هر دو را بزنید. بعد از انتخاب آن‌ها می‌توانید **وقفه‌های دریافت و ارسال** را فعال کنید. با فعال کردن آن (زدن تیک Rx Interrupt) گزینه ی Receiver Buffer نمایش داده می شود که در آن قسمت می توانید **حجم بافر گیرنده** را مشخص کنید. این بافر یک متغیر است که اطلاعات دریافتی را داخل خود نگه می دارد. (گزینه ی Receiver buffer بدون تغییر خواهد ماند)

قسمت **Mode**:

قسمت Mode برای زمانی است که بخواهید **ارتباط شبکه** داشته باشید؛ یعنی یک میکرو مرکزی و چند میکروی جانبی داشته باشید . به این قسمت در آینده خواهیم پرداخت. این قسمت (mode) نیز بدون تغییر و روی تنظیمات پیش فرض باشد .

انتخاب قالب رشته ارسالی: قسمت Communication Parameters چند گزینه دارد:

شکل ۱۰-۴

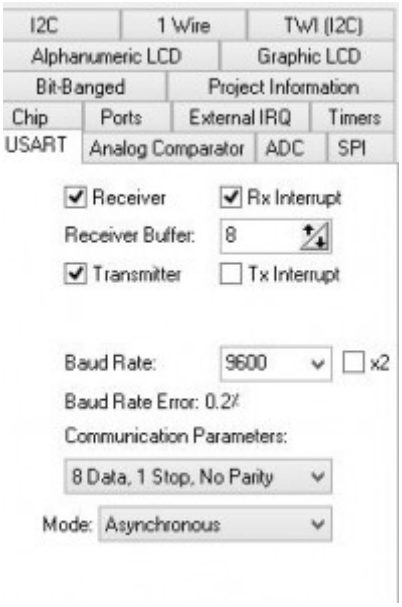
خط متصل به گیرنده در ارتباط سریال همیشه در **سطح منطقی ۱** است. به محض اینکه ارتباط شروع می‌شود این خط به **سطح منطقی صفر** می رود و به همین خاطر (**تغییر سطح منطقی از ۱ به ۰**) شما یک وقفه در میکروکنترلر خواهید داشت. این تغییر سطح به دستگاه می‌گوید که ارسال داده‌ها شروع شده است . بعد از بیت شروع، داده اصلی ارسال می‌شود که می‌تواند در قالب ۷، ۸ یا ۹ بیت باشد .

بعد از اینکه داده (Data) ارسال شد باید دو طرف بفهمند که داده ارسال و کار تمام‌شده است. به همین خاطر معمولاً یک بیت به عنوان بیت پایان ارسال می‌شود که به آن بیت توقف یا Stop می گویند. برای ارتباط ۱ یا ۲ بیت توقف می توان در نظر گرفت. شما همان ۱ بیت را در نظر بگیرید. ۲ بیت برای محکم‌کاری یا مقاصد خاص دیگری است که در اینجا کاربرد ندارد. اما اگر در زمان ارسال یکی از بیت‌ها براثر نویز محیط یا هرچیزه دیگری عوض شود، یعنی ۱ به ۰ یا ۰ به ۱، تغییر کند از کجا می توان فهمید که اشتباه صورت گرفته است؟ برای این کار از **Parity** استفاده می‌کنند.

تنظیمات نرخ ارسال اطلاعات (BaudRatet) :

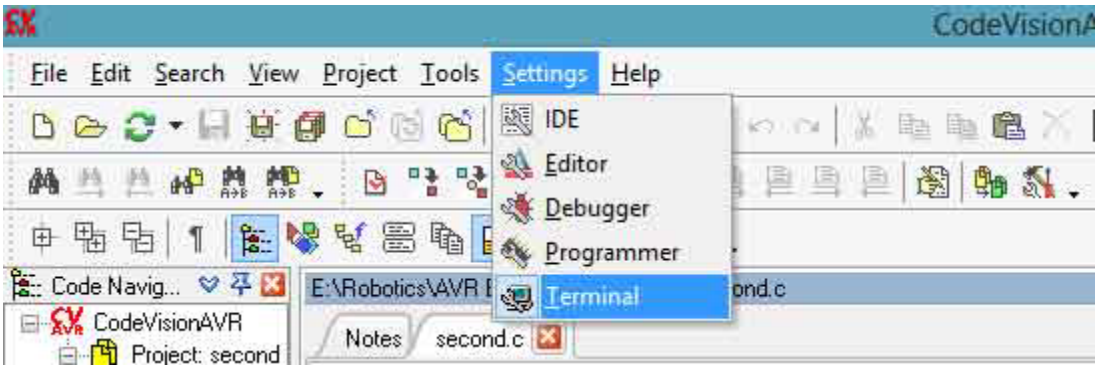
در ارتباط سریال نیازی نیست فرکانس قسمت ارتباط همه ی دستگاها یکسان باشد؛ مثلا یک طرف فرکانس شما 1Megو طرف دیگر8Meg هست . یا هر اعدادی که منطقی باشند! اگر فرکانس در دو طرف با هم متفاوت باشد و بخواهند با همان فرکانس ارسال را انجام دهند اطلاعات اشتباه ارسال می‌شود و به درد نمی‌خورد. علت اینکه اگر در دو طرف فرکانس‌های کاری متفاوت باشند اما ارتباط درست انجام شود این است که این کار را در ارتباط سریال تنظیم باودریت انجام می دهد یعنی با هر فرکانسی که میکرو کار می کند مهم نیست، فقط باید با سرعتی که باودریت مشخص می کند ارسال انجام شود. وقتی این کار انجام شود هر دو طرف فرستنده و گیرنده باهم هماهنگ میشوند و کارشان را انجام میدهند و اشتباهی صورت نمی‌گیرد. واحد **باودریت** همان BPS است که مخفف Bit Per Second هست؛ یعنی بیت بر ثانیه. معمولاً روی ۹۶۰۰ تنظیم می شود .

تنظیمات نهایی ارتباط سریال:



شکل ۴-۱۱

پس از پروگرام کردن و کامل کردن اتصالات، برای شروع ارتباط کامپیوتر با میکروکنترلر نیاز به یک اینترفیس نرم‌افزاری در PC داریم که از طریق آن با پورت سریال و میکرو ارتباط برقرار کنیم نرم‌افزارهای متعددی به نام Termina برای این منظور طراحی‌شده‌اند که می‌توان از همه آن‌ها استفاده نمود. در نرم‌افزار کدویژن ابزاری برای این منظور تعبیه‌شده است که در منوی Tools قرار دارد؛ بنابراین بعد از اتصال کابل سریال به کامپیوتر و وصل کردن منبع تغذیه به میکرو و روشن نمودن آن، از منوی Setting در نرم‌افزار کدویژن Terminal را انتخاب کرده و تنظیمات مربوط به پورتی که میکرو به آن وصل است و قاب دیتا و نرخ ارسال/دریافت را نیز مشخص می‌کنیم.



شکل ۴-۱۲

در قسمت کد نویسی داخل while کدهای زیر را قرار می‌دهیم:

```
While(1)
{
  Puts("hello world…\n\r");
```

```
  Delay_ms(1000);
}
```

تمرین ۱-۴:

الف) ابتدا سخت‌افزار مدار را مطابق با دستور کار آماده کرده

ب) برنامه‌ای بنویسید که هر عددی که توسط صفحه‌کلید وارد می‌شود در خط اول LCD با عبارت "keypad: --" نمایش دهد.

ج) همزمان با نمایش بر روی LCD عدد را بر روی پورت سریال فرستاده و در ترمینال ویندوز مشاهده کنید.

د) هر عدد یا حرفی که توسط صفحه‌کلید ویندوز وارد می‌شود را توسط پورت سریال میکرو دریافت کرده و در خط دوم LCD با عبارت "serial: --" نمایش دهید.

آزمایش پنجم: راه‌اندازی سنسور دما LM35

هدف:

۱- آشنایی با واحد ADC
۲- معرفی فیلتر آنتی‌الیاس

قطعات موردنیاز:

۱- سنسور دما LM35 ---------- ۱ عدد
۲- سلف ۱۰ میکرو هانری ---------- ۱ عدد
۳- خازن ۱۰۰ نانو فاراد ---------- ۱ عدد

چکیده: سنسور دما، همانند لودسل‌ها، ترانسمیترها و فلومترها از کاربردی‌ترین ابزارها در صنعت و اتوماسیون صنعتی بشمار می‌رود. اطلاعات این سنسورها به کنترلرهای ابزار دقیق چون plc وارد می‌شود و از این طریق پارامترهای یک محیط صنعتی کنترل می‌شود.

در پروژه‌های صنعتی همواره شرایط محیطی از عوامل مهم و تأثیرگذار است که می‌بایست در نظر گرفته شود. یکی از مهم‌ترین عوامل در این زمینه بی‌شک، دما می‌باشد. برای اندازه‌گیری دما از سنسورهای ویژه‌ای استفاده می‌گردد که در اصطلاح به سنسورهای دما معروف هستند.

به‌طورکلی روش‌های اندازه‌گیری دما را می‌توان به دودسته تماس (contact) و غیر تماسی (non-contact) تقسیم‌بندی نمود.

روش غیر تماسی:

برای اندازه‌گیری دما درروش غیر تماسی از نور مادون‌قرمز استفاده می‌گردد و بدون آنکه سنسور، برخوردی با ماده موردنظر داشته باشد، می‌تواند دمای آن را اندازه‌گیری نماید. به‌عنوان نمونه فرض کنید که شما می‌خواهید دمای مایعات درون یک مخزن را اندازه‌گیری نمایید و به دلیل بزرگی مخزن امکان ایجاد تماس با سیال درون مخزن را ندارید. برای اندازه‌گیری دما در این شرایط، به‌راحتی می‌توان از سنسورهای غیر تماسی و لیزری در بالای سطح مخزن استفاده نمود.

همچنین اگر ماده‌ای که قصد اندازه‌گیری دمای آن راداریم، خطرناک باشد و امکان نزدیک شدن به آن را نداشته باشیم، برای اندازه‌گیری دمای آن ماده به‌سادگی می‌توان از سنسورهای دمای غیر تماسی استفاده نمود.

شکل ۱-۵

روش تماسی:

روش‌های تماسی برای اندازه‌گیری دما را می‌توان به دودسته کلی تقسیم‌بندی نمود.

- ترموکوبل (TH)

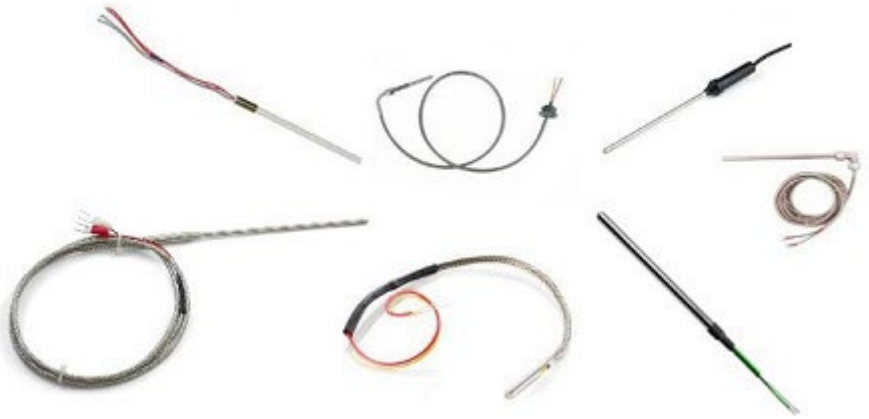
-آر تی دی (RTD)

***سنسور دمایی ترموکوپل:**

ترموکوپل از دو فلز با جنس‌های مختلف تشکیل‌شده است که این فلزات ضریب‌های انتقال دمایی متفاوتی باهم دارند و با توجه به این ویژگی زمانی که گرما را حس می‌کنند، رفتارهای متفاوتی از خود نشان می‌دهند که باعث ایجاد ولتاژ بین دو فلز می‌گردد. این اختلاف ولتاژ بیان‌کننده میزان دما است. با توجه به اینکه از چه فلزاتی برای ساخت ترموکوپل استفاده‌شده است؛ دسته‌های مختلفی از ترموکوپل ها ایجادشده است که از مهم‌ترین انواع ترموکوپل‌ها می‌توان به ترموکوپل نوع J, K,T, E, ... اشاره نمود.

سنسورهای دمای ترموکوپلی بسیار دقیق بوده و دقت آن در رنج یک درجه سانتی گراد است.

ترموکوپل‌ها محدوده دمایی خاص خود رادارند که این محدوده بیان‌کننده بازه دمایی است که ترموکوپل می‌تواند اندازه‌گیری دما را به نحو احسن انجام دهد. با توجه به محدوده‌های دمایی مختلف، ترموکوپل‌ها انواع مختلفی دارند که هر یک قیمت‌های متفاوتی دارند.



شکل ۲-۵

***سنسور دمایی آر تی دی (RTD):**

Resistance Temperature Detector که به اختصار RTD نامیده می‌شود یک کنترل‌کننده دما با استفاده از مقاومت است. در این سنسورها از یک مقاومت متغیر با دما استفاده می‌گردد که با استفاده از تغییر مقاومت بتوان به میزان تغییر دما پی برد.

به‌صورت کلی می‌توان RTD ها را به دسته‌های ۲ سیمه، ۳ سیمه و ۴ سیمه تقسیم‌بندی نمود. مدل‌های ۴ سیمه دارای قیمت بالا و مدل‌های ۲ سیمه دقت پایینی دارند و ازاین‌رو در صنعت بیشتر از RTD های ۳ سیمه استفاده می‌گردد.

سنسورهای RTD نسبت به سنسورهای ترموکوپل سنسورهای دقیق‌تری هستند و دقتی در حدود ۰/۱ درجه سانتی گراد دارند و همچنین زمان پاسخ‌گویی سریع‌تری دارند. از معروف‌ترین سنسورهای RTD می‌توان به سنسورها ۱۰۰PT، PT1000 و NTC اشاره نمود.

معروف‌ترین Sensor تشخیص دما موجود در بازار Lm35 می‌باشد، این سنسور تغییرات دمای مورد نظر را به ولتاژ آنالوگ تبدیل می‌کند.

این سنسور دارای سه‌پایه می‌باشد درصورتی‌که سنسور روبروی ما قرار گیرد (بتوانیم نوشته‌هایش را ببینیم). اولین پایه، سمت چپ (vcc) می‌باشد که به ۵ ولت وصل می‌شود.

پایه وسط، ولتاژ خروجی (Vout) است که به میکروکنترلر متصل می‌شود.

پایه سوم، زمین (GND) سنسور است.

محدوده دمایی که این سنسور قادر به اندازه‌گیری آن می‌باشد بین ۵۵- تا ۱۵۰+ درجه سانتی گراد است و این سنسور به ازای هر درجه سانتی‌گراد ۱۰ میلی ولت ولتاژ خروجی را تغییر می‌دهد؛ یعنی به ازای دمای ۱ درجه، ولتاژ خروجی سنسور ۱۰ میلی ولت و به ازای ۱۰۰ درجه خروجی سنسور ۱۰۰۰ میلی ولت می‌باشد.

همچنین به ازای دمای ۲۰- درجه خروجی سنسور ۲۰۰- میلی ولت می‌باشد.

نکته مهم: ازآنجاکه در سنسور پایه ولتاژ منفی به چشم نمی‌خورد، برای اندازه‌گیری دمای منفی باید پایه vo سنسور را توسط ۱ مقاومت ۱ کیلو به ولتاژ منفی متصل کرد (۵ – ولت)

ولتاژ تغذیه این سنسور ۵ ولت می‌باشد، همچنین بدنه آن قابلیت تحمل دما تا ۲۰۰ درجه سانتی‌گراد را دارد.

ازآنجاکه مبدل آنالوگ به دیجیتال داخل میکروکنترلر مانند میکرو AVR ده بیتی است و ولتاژ مرجع آن بین صفر تا ۵ ولت است درنتیجه ولتاژ اندازه‌گیری شده را به ۱۰۲۴ قسمت تقسیم می‌کند پس می‌تواند سنسور دمای موردنظر ما را به‌راحتی با دقت ۰/۵ درجه سانتی‌گراد بخواند.

کاربرد سنسور LM35 :

۱-در بعضی از پروژه‌ها مانند سیستم‌های تهویه مطبوع به دلیل مهم بودن دمای هوای در حال جریان، در کانال عبوری هوا از سنسورهای دمای متعددی استفاده می‌شود.

۲- می‌توان با سنسور Lm35 اندازه‌گیری و رسم منحنی دقیق میزان رطوبت را داشته باشیم. با توجه به رابطه‌ای که بین میزان رطوبت و دما می‌باشد می‌توان برای رسم صحیح منحنی رطوبت، از سنسور دمای Lm35 استفاده نمود که درواقع برای رسم منحنی و مانیتورینگ از نرم‌افزار لب ویو استفاده کرد.

۳- کنترل دمای تماسی بعضی تجهیزات مانند ترموالکتریک یا بدنه موتور و یا هیت سینک و ... را می‌توان داشت.

مبدل آنالوگ به دیجیتال:

مقدمه: در حالت کلی دنیای خارج از میکروکنترلر ماهیت آنالوگ (پیوسته) دارد. مبدل‌های آنالوگ به دیجیتال (ADC) و دیجیتال به آنالوگ (DAC) امکان ارتباط میکروکنترلر با سیگنال‌های آنالوگ را فراهم می‌کند.

مبدل ADC یک ولتاژ آنالوگ را گرفته و مقدار دیجیتالی برای آن ولتاژ فراهم می‌کند و مبدل DAC یک مقدار دیجیتال را گرفته و متناسب با آن‌یک ولتاژ آنالوگ تولید می‌کند.

برخی از میکروکنترلرها دارای یک واحد داخلی مبدل آنالوگ به دیجیتال هستند که می‌تواند سیگنال‌های آنالوگ بین صفر تا ۵ ولت را بدون نیاز به مدارات جانبی به دیجیتال تبدیل کند.

مبدل‌های آنالوگ به دیجیتال:

به‌طورکلی ۶ روش برای تبدیل سیگنال‌های آنالوگ به دیجیتال وجود دارد:

۱-مبدل به روش موازی یا همزمان

۲-مبدل به روش پله‌ای

۳-مبدل به روش تقریب متوالی (Successive Approximation)

۴-مبدل به روش تبدیل ولتاژ به زمان (تک‌شیب)

۵-مبدل به روش دو شیب

۶-مبدل به روش تبدیل ولتاژ به فرکانس

در میکروکنترلرهای AVR از روش تقریب متوالی برای تبدیل سیگنال‌های آنالوگ به دیجیتال استفاده می‌شود.

مبدل ADC به روش تقریب متوالی:

در این روش خروجی یک شمارنده که به ورودی یک مبدل DAC متصل شده است، باعث می‌شود که خروجی DAC متناسب با مقدار خروجی شمارنده دارای یک ولتاژ آنالوگ باشد. این ولتاژ (V_{DAC}) توسط یک مقایسه کننده با ولتاژ آنالوگ ورودی مقایسه شده و درصورتی‌که تساوی پایه Stop شمارنده را فعال کرده تا شمارنده متوقف شود. حال مقدار دیجیتالی خروجی همان مقدار متناسب با ولتاژ آنالوگ ورودی خواهد بود.

بنابراین رابطه مقدار دیجیتال خروجی متناسب با ولتاژ آنالوگ ورودی را می‌توان به‌صورت فرمول زیر نشان داد:

$$\frac{Vin}{Vref} = \frac{\text{مقدار دیجیتال}}{2^{n-1}}$$

با توجه به فرمول، ولتاژ خروجی DAC فقط می‌تواند مقادیر گسسته (با تقریب ± 1 بیت) را داشته باشد به این ویژگی درجه‌ی تفکیک می‌گویند و به‌صورت زیر محاسبه می‌شود:

$$\text{درجه تفکیک} = \frac{Vref}{2^{n-1}}$$

درجه تفکیک ، محدودکننده دقت خروجی ADC است. برای بهبود آن می‌توان Vref را کاهش یا تعداد بیت‌های شمارنده را زیاد نمود.

بررسی واحد ADC در میکروکنترلر AVR:

مبدل ADC در AVR ها بسته به نوع AVR از کانال‌های ورودی متعددی بهره می‌برد. به‌طوری‌که توسط یک مالتی پلکس می‌توان ورودی‌های مختلفی را به ADC متصل نمود. کانال‌های ورودی به‌صورت Singel-Ended عمل می‌کند؛ که در این حالت دامنه‌ی ولتاژ ورودی باید بین صفر تا Vcc باشد. ADC دارای دو مد تبدیل Single و Free می‌باشد.

ولتاژ مرجع (Vref) ADC از سه طریق تأمین می‌شود: (در برخی از AVR از یک یا دو طریق)

۱-ولتاژ مرجع داخلی ۲/۵۶ ولت

۲-ولتاژ روی پایه Aref

۳-ولتاژ روی پایه Avcc

باید توجه داشت که ADC دارای پایه تغذیه آنالوگ مجزا (Avcc) می‌باشد؛ که اختلاف آن به vcc نباید بیشتر از ± 0.3 ولت باشد. برای انتخاب ولتاژ مرجع می‌توان از بیت‌های کنترلی REFS0,1 استفاده نمود. درصورتی‌که از ولتاژ مرجع داخلی ۲/۵۶ ولت استفاده شود، با توجه به اینکه این ولتاژ روی پایه Aref نیز ظاهر می‌شود، برای کاهش نویز روی این پایه‌یک خازن (حدود 100nf) قرار می‌گیرد.

رجیسترهای ADC

۱-ریجستر کنترلی ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

شکل ۳-۵

REFS0,۱

این دو بیت جهت انتخاب ولتاژ مرجع مورد استفاده قرار می‌گیرد. جدول ۱ چگونگی عملکرد این دو بیت را نشان می‌دهد.

جدول ۱-۵: انتخاب ولتاژ مرجع

REFS0	REFS1	ولتاژ مرجع انتخاب‌شده
۰	۰	در این حالت ولتاژ موجود روی پایه Aref به‌عنوان ولتاژ مرجع انتخاب می‌شود.
۱	۰	در این حالت ولتاژ پایه Avcc به عنوان ولتاژ مرجع انتخاب می‌شود.
۰	۱	بدون استفاده
۱	۱	در این حالت ولتاژ مرجع داخلی ۲.۵۶ ولت به‌عنوان ولتاژ مرجع انتخاب می‌شود.

توجه داشته باشید، تغییر بیت‌های REFS0,۱ زمانی مؤثر خواهد بود که ADC در حال تبدیل نباشد. در غیر این صورت پس از اتمام تبدیل، تغییر بیت‌ها مؤثر واقع‌شده و ولتاژ مرجع تغییر حالت می‌دهد.

ADLAR

این بیت روی رجیستر داده ADC(ADCH,ADCL) تأثیر می‌گذارد، بخش بعدی به عملکرد این بیت پرداخته خواهد شد.

MUX4:0

از این بیت‌ها جهت تعیین کانال ورودی و نیز انتخاب بهره تفاضلی بهره گرفته می‌شود. در برخی از انتخاب‌ها مشاهده می‌شود که ورودی مثبت و منفی برای حالت تفاضلی، یکی از ورودی‌هایی است که در عمل برای اندازه‌گیری ولتاژ آفست استفاده می‌شود.

۲-رجیستر ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

شکل ۴-۵

ADEN

با یک کردن این بیت مبدل ADC فعال و با غیر صفر کردن آن غیرفعال می‌شود. درصورتی‌که ADC در حال تبدیل باشد، با صفر کردن این بیت ADC غیرفعال شده و عمل تبدیل نیمه‌کاره رها خواهد شد.

ADSC

در مد عملکرد single با یک کردن این بیت، تبدیل شروع‌شده و پس از پایان تبدیل، به‌صورت خودکار صفر می‌شود. در مد Free, یک کردن این بیت برای شروع تبدیل الزامی است.

ADATE

با نوشتن یک در این بیت ADC می‌تواند به‌صورت خودکار تحریک شود (Auto Tregger) و با لبه بالارونده منبع تحریک‌کننده، شروع به تبدیل کند. منبع تحریک‌کننده ADC توسط بیت‌های ADTS از رجیستر SFIOR انتخاب می‌شود.

ADIF

با اتمام تبدیل ADC و تغییر محتویات رجیستر داده ADC(ADCH,ADCL)، این بیت یک خواهد شد.

ADIE

در صورت فعال کردن این بیت هنگام اتمام تبدیل (ADIF=1), وقفه اتمام تبدیل رخ خواهد داد (درصورتی‌که وقفه سراسری فعال باشد).

ADPS2:0

این بیت‌ها تعیین‌کننده پالس ساعت اعمالی به ورودی پالس ساعت ADC (شمارنده ۱۰ بیت) می‌باشند و با توجه به مقدار این بیت‌ها تقسیمی از فرکانس نوسان‌ساز به آن اعمال خواهد شد.

جدول ۲-۵: انتخاب ضرایب تقسیم فرکانس نوسان‌ساز

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
0	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

۳-رجیستر داده ADC (ADCH,ADCL)

این دو رجیستر حاوی اطلاعات خروجی ADC می‌باشند، اگر از کانال‌های تفاضلی استفاده‌شده باشد نتیجه به‌صورت مکمل ۲ در این رجیسترها قرار می‌گیرد. همان‌طور که قبلاً گفته شد توسط بیت ADLAR می‌توان وضعیت قرار گرفتن داده در این دو رجیستر را تعیین نمود.

اگر ADLAR=1 باشد نتیجه تبدیل به‌صورت تنظیم از چپ (Left Adjust) و اگر ADLAR=0 باشد نتیجه به‌صورت تنظیم از راست (Right Adjust) در رجیسترهای ADCH,ADCL قرار خواهد گرفت.

اگر به‌دقت تبدیل بیش از هشت بیت نیاز نباشد، می‌توانید ADLAR=1 کرده و فقط رجیستر ADCH را بخوانید.

۴-رجیستر SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

شکل ۵-۵

ADTS2:0

این سه بیت منبع تحریک‌کننده ADC را مشخص می‌کند. این بیت‌ها تنها زمانی مؤثر خواهند بود که بیت ADATE از رجیستر ADCSRA فعال باشد. هر تبدیل با یک لبه صعودی منبع تحریک‌کننده اتفاق می‌افتد.

توجه داشته باشید که اگر ADEN فعال باشد با تغییر منبع تحریک‌کننده و ایجاد یک لبه‌ی صعودی تبدیل آغاز می‌شود.

ولتاژ مرجع ADC:

همان‌طور که قبلاً اشاره شد ولتاژ مرجع ADC به سه طریق قابل انتخاب است.

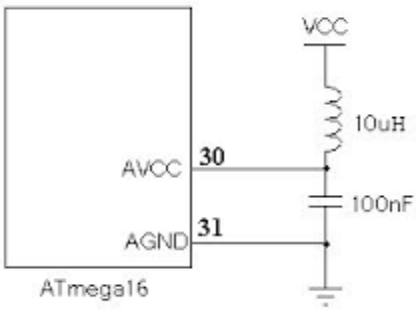
ولتاژ مرجع از طریق پایه AVCC ، ولتاژ ۲/۵۶ ولت داخلی یا ولتاژ از طریق پایه AREF

نکته که وجود دارد این است که ولتاژ مرجع تعیین‌کننده محدوده تبدیل است؛ و درصورتی‌که ولتاژ ورودی ADC از Vref تجاوز کند، خروجی حدود 3FFH خواهد شد.

اگر ولتاژ مرجعی به پایه AREF متصل شود، نباید ولتاژ مرجع داخلی یا پایه AVCC فعال باشد زیرا این عمل موجب اتصال کوتاه بین دو ولتاژ خواهد شد و امکان صدمه دیدن ADC وجود دارد. در صورت استفاده از کانال تفاضلی محدوده ولتاژ مرجع به‌صورت ۲<Vref<Avcc -۰/۵ باید رعایت شود.

*مبدل ADC دارای Noise Canceller می‌باشد که نویز حاصل از CPU و وسایل جانبی را بر روی ADC را کاهش دهد.

استفاده از شبکه LC برای تغذیه ADC

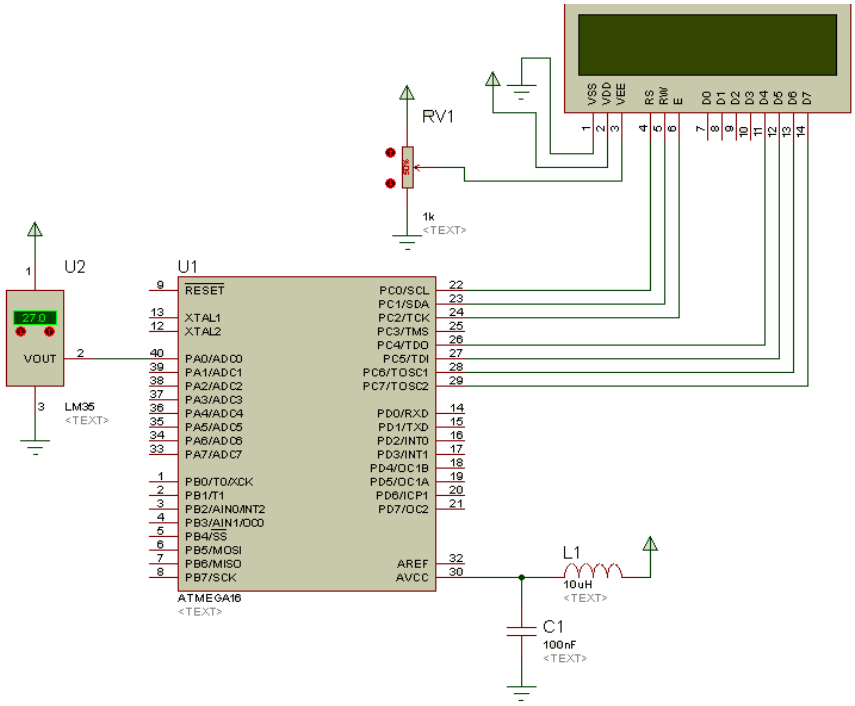


شکل ۶-۵

تمرین۱-۵: ابتدا سخت‌افزار نشان داده‌شده در شکل ۷ را بر روی برد بورد ببندید و سپس برنامه‌ای بنویسید که دمای محیط را با استفاده از سنسور lm35 خوانده و بر روی نمایشگر نشان دهد.

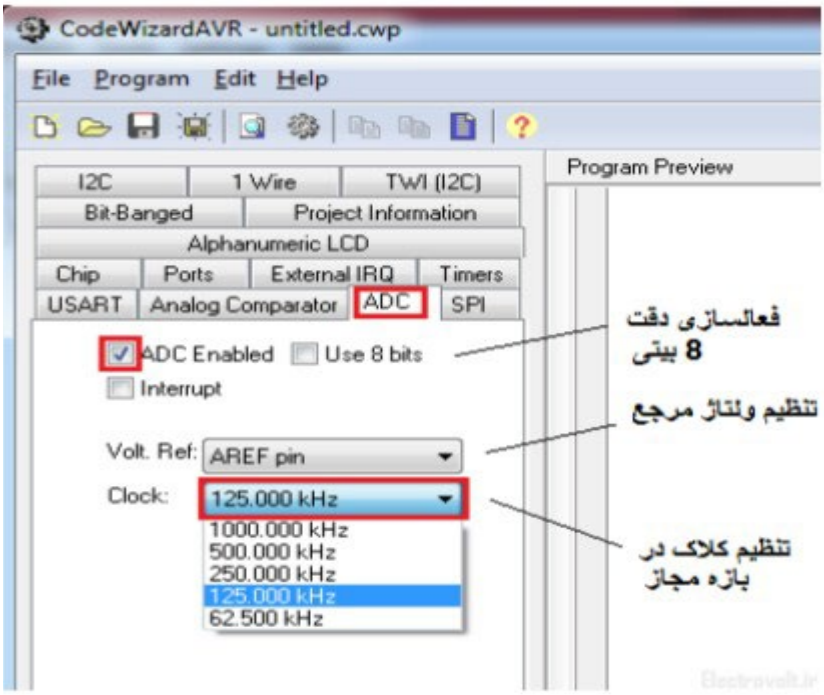
توضیح: استفاده از پورت‌های بکار گرفته‌شده در شکل ۷ الزامی نیست.

در این پروژه ما توسط مبدل آنالوگ به دیجیتال (ADC) میکرو ولتاژهای خروجی سنسور LM35 را اندازه‌گیری می‌کنیم و با قرار دادن این مقادیر در یک نسبت ساده آن‌ها را به مقادیر دما تبدیل می‌کنیم و بر روی LCD کاراکتری نمایش می‌دهیم. شماتیک به‌صورت زیر است:



شکل ۷-۵

تنظیمات کدویزارد:
در کدویزارد میکرو ATmega16 را انتخاب کنید و فرکانس آن را ۱ مگاهرتز قرار دهید پورت D را برای اتصال به LCD خروجی کنید و تنظیمات LCD کاراکتری را برای اتصال به پورت D انجام دهید. ADC میکرو را فعال کنید. برای نمایش مقادیر دما بر روی LCD کاراکتری لازم است که ابتدا این مقادیر به رشته‌ای از کاراکتر تبدیل شوند تا بتوانیم آن‌ها را بر روی LCD نمایش دهیم برای این کار از دستور sprintf استفاده می‌کنیم. این دستور یکی از توابع کتابخانه stdio می‌باشد، بنابراین برای استفاده از آن لازم است که در ابتدای برنامه این کتابخانه را فراخوانی کنیم.



شکل ۸-۵

همان‌طور که می‌دانید دستور read_adc عددی بین ۰ تا ۱۰۲۳ با توجه به ولتاژ آنالوگ ما از پین‌های میکرو می‌خواند (درصورتی‌که تیک Use 8 bits را نزده باشیم) و ازآنجایی‌که در دیتاشیت LM35 گفته‌شده که به ازای هر درجه سانتی‌گراد، ۱۰ میلی ولت تغییرات در خروجی سنسور داریم بنابراین می‌توانیم با یک تناسب ، ولتاژ خواند شده از سنسور را به مقادیر دما تبدیل کنیم. این کار در خط ۴۰ برنامه انجام‌شده و با تقسیم مقادیر خوانده‌شده از ADC بر ۲/۰۵۴ ولتاژ ما به مقادیر دما تبدیل می‌شود.

و درنهایت کدهای این پروژه به‌صورت زیر است:

While(1)

```
{
A=read_ADC(0)
V=(A*5.0)/255.0;
Temp=v*100;
Sprintf(buff, ”temp=%f”,temp)
puts(buff);
Delay_ms(1000);
}
```

آزمایش ششم: ساخت فرکانس‌متر

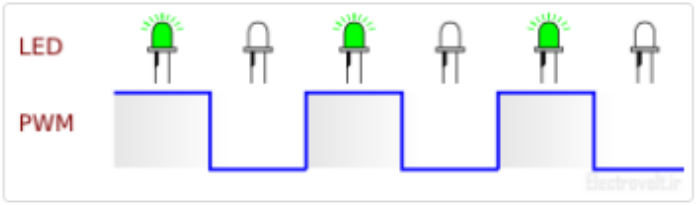
هدف:

۱- آشنایی با تایمرها
۲- آشنایی با وقفه‌های خارجی

شرح آزمایش:

مقدمه:

یکی از مهم‌ترین واحدهای میکروکنترلر **واحد تایمر/کانتر** می‌باشد که در اکثر پروژه‌های مهم وجود آن ضروری است. این واحد از نظر سخت‌افزاری متشکل از یک شمارنده اصلی و چندین رجیستر برای تنظیمات می‌باشد به‌طوری‌که با اعمال تنظیمات متفاوت چندین کاربرد مختلف از این سخت‌افزار خاص می‌شود. از مهم‌ترین کاربردهای این سخت‌افزار می‌توان به تایمر (زمان‌سنج)، کانتر (شمارنده)، Real Time Clock (زمان‌سنج حقیقی) و PWM (مدولاسیون عرض پالس) اشاره کرد.



شکل ۱-۶

مهم‌ترین ویژگی‌های یک واحد تایمر/کانتر در میکروکنترلرهای AVR:

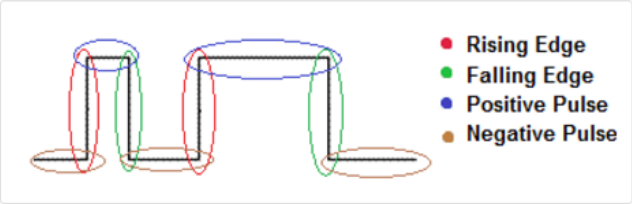
مهم‌ترین مسائلی که در هنگام کار با واحد تایمر/کانتر به وجود می‌آید به شرح زیر می‌باشد:

- **طول شمارنده واحد:** تعداد بیت‌های شمارنده (کانتر) می‌تواند ۸, ۱۶, ۳۲ و ... باشد. در میکروکنترلرهای AVR تنها طول ۸ بیتی و ۱۶بیتی وجود دارد.
- **رجیسترهای تنظیمات واحد:** همواره تعدادی رجیستر در کنار واحد به‌منظور انجام تنظیمات وجود دارد که بسته به ساده یا پیچیده بودن تعداد آنها کم یا زیاد می‌شود.
- **مد (Mode) یا حالت کار واحد:** برای این واحد دو حالت کار متفاوت وجود دارد. حالت کار تایمری و حالت کار کانتری.

- **تنظیمات کلاک ورودی واحد:** در حالت **تایمری** کلاک ورودی تقسیمی از کلاک اصلی می‌باشد و از **داخل** به واحد وارد می‌شود اما در حالت **کانتری** کلاک واحد از طریق پایه مربوطه (پایه های Tx) و از **خارج** میکروکنترلر به واحد اعمال می‌شود.
- **مشخص کردن رخداد در حالت کانتری:** زمانی که واحد در حالت کانتری کار می‌کند باید نوع رخدادی که می‌خواهیم شمرده شود را تنظیم نماییم این رخداد یکی از انواع رخدادها در سیگنال ورودی به شرح زیر است:

۱. **لبه بالارونده (Rising Edge)**
۲. **لبه پایین‌رونده (Falling Edge)**
۳. **پالس مثبت (Positive Pulse)**
۴. **پالس منفی (Negative Pulse)**

در شکل ۲ انواع رخدادهای فوق که در یک نمونه سیگنال مشخص‌شده است را مشاهده می‌کنید.



شکل ۲-۶

- **مشخص کردن حالت کار واحد تایمر/کانتر:** زمانی که واحد در حالت تایمری کار می‌کند می‌توان از آن در حالت مختلف زیر استفاده کرد:

۱. **حالت ساده (عادی)**
۲. **حالت مقایسه (CTC)**
۳. **حالت PWM سریع (تک‌شیب)**
۴. **حالت PWM تصحیح فاز (دو شیب)**
۵. **حالت PWM تصحیح فاز و فرکانس**

تذکر: نحوه عملکرد دقیق واحد تایمر/کانتر در حالت‌های فوق، در بخش مربوط به هر یک تشریح خواهد شد.

- **فعال یا غیرفعال بودن خروجی:** هر یک از واحدهای تایمر/کانتر، **حداکثر** سه خروجی دارد که هرکدام از خروجی‌ها به یکی از پایه‌های میکروکنترلر متصل می‌شود. پایه‌هایی که مربوط به واحد تایمر/کانتر هستند در میکروکنترلرهای AVR به نام **OCx** مشخص می‌شوند که در آن **x یک عدد بین ۰ تا ۴ است**. در صورت

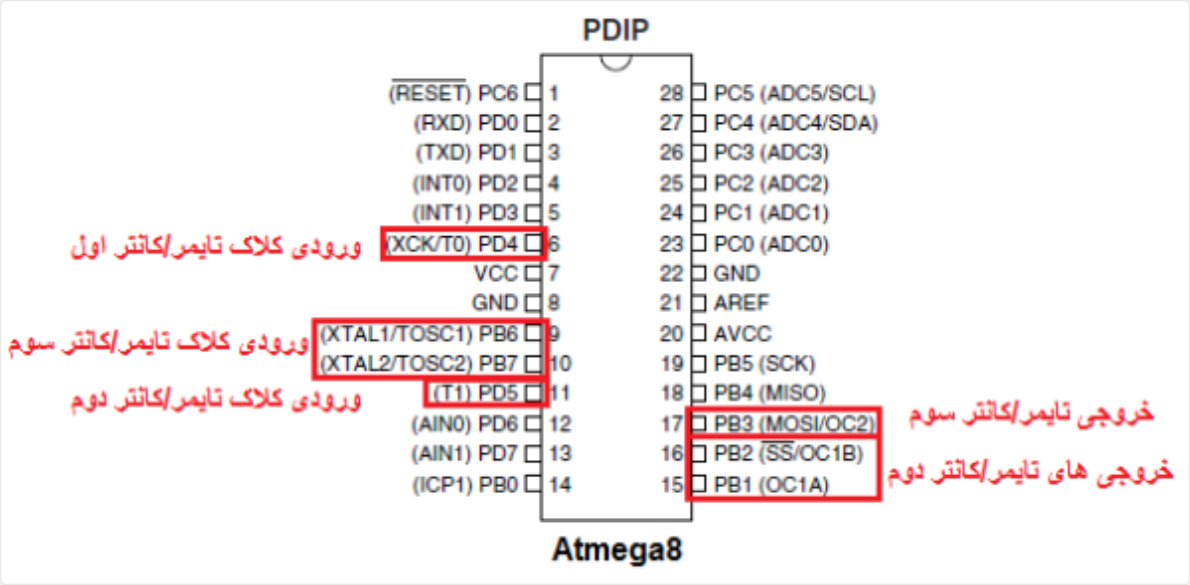
فعال بودن خروجی واحد، پایه مربوط از حالت I/O معمولی (واحد ورودی/خروجی) خارج می‌شود و به خروجی واحد تایمر/کانتر متصل می‌گردد.

انواع واحد تایمر/ کانتر در میکروکنترلرهای AVR

شکل زیر انواع واحد تایمر/ کانتر را به همراه میکروکنترلر AVR موردنظر نشان می‌دهد. لازم به **تذکر** است که هر میکروکنترلر AVR حداقل یک و حداکثر پنج واحد تایمر/کانتر دارد که به‌ترتیب از شماره ۰ نام‌گذاری می‌شود. هر یک از این واحدها به‌خودی‌خود می‌تواند ساده، پیشرفته، ۸ بیتی یا ۱۶ بیتی باشد.

شکل ۳-۶

مثال: میکروکنترلر Atmega8 دارای ۳ واحد تایمر/کانتر می‌باشد. تایمر/کانتر شماره ۰ به‌صورت ساده ۸ بیتی، تایمرکانتر شماره ۱ به‌صورت پیشرفته ۱۶ بیتی و تایمر/کانتر شماره ۲ به‌صورت پیشرفته ۸ بیتی می‌باشد. پایه‌های خروجی مربوط به هریک از واحدهای تایمر/کانتر را در شکل ۴ مشاهده می‌کنید.



شکل ۴-۶

نکته: همان‌طور که در شکل ۴ نیز مشاهده می‌کنید، در تایمر/کانترهای ساده ۸ بیتی پایه خروجی وجود ندارد. در تایمر/کانترهای پیشرفته ۸ بیتی ۱ خروجی و در تایمر/کانترهای پیشرفته ۱۶ بیتی، دو خروجی و در برخی میکروکنترلرها سه خروجی وجود دارد.

نکته: همان‌طور که در شکل ۴ نیز مشاهده می‌کنید کلیه ورودی‌های واحدهای تایمر/کانتر، کلاک حالت کار **کانتری** می‌باشند که در حالت کار تایمری بدون استفاده است. ضمناً ورودی کلاک تایمر/کانتر سوم (پایه‌های TOSC1 و TOSC2) تنها می‌تواند برای RTC (زمان سنج حقیقی) استفاده شود.

معرفی اجمالی رجیسترهای واحدهای تایمر/کانتر

این رجیسترها که نام آن‌ها در همه واحدهای تایمر/کانتر اعم از ساده، پیشرفته، ۸ بیتی یا ۱۶ بیتی یکسان هستند و به‌طور خودکار توسط کد ویزارد مقداردهی می‌شوند، به شرح زیر می‌باشد:

رجیستر کنترل تایمر/کانتر (TCCRX):

این رجیسترکه مخفف Timer Counter Control Register است، وظیفه‌ی کنترل کلیه تنظیمات واحد تایمر کانتر را بر عهده دارد. این تنظیمات که در کد ویزارد نیز انجام می‌شوند به شرح زیر است:

۱. فعال یا غیرفعال بودن واحد
۲. مشخص کردن حالت کار تایمری یا کانتری (منبع کلاک)
۳. تنظیم عدد تقسیم کلاک ورودی واحد در حالت تایمری

۴. مشخص کردن نوع رخداد در حالت کانتری
۵. تنظیم حالت‌های مختلف کاری واحد (CTC, PWM, ….)
۶. فعال یا غیرفعال کردن خروجی واحد پایه‌های OCX

ریجستر تایمر/کانتر (TCNTX):

این رجیستر ۸ بیتی، مقدار شمارنده در هرلحظه را به‌صورت خودکار در خود ذخیره می‌کند. این رجیستر امکان دسترسی مستقیم برای خواندن و نوشتن در شمارنده را فراهم می‌کند. به‌طوری‌که این رجیستر هنگام خواندن مقدار شمارش‌شده را برمی‌گرداند و به هنگام نوشتن مقدار جدید را به شمارنده انتقال می‌دهد.

رجیستر مقایسه خروجی (OCRX):

این رجیستر۸ بیتی خواندنی و نوشتنی بوده و به‌طور مستقیم با مقدار شمارنده TCNT مقایسه می‌شود. از تطابق این دو برای تولید وقفه خروجی یا تولید یک شکل موج روی پایه OCX می‌توان استفاده نمود.

رجیستر پوشش وقفه تایمر/کانتر (TIMSK):

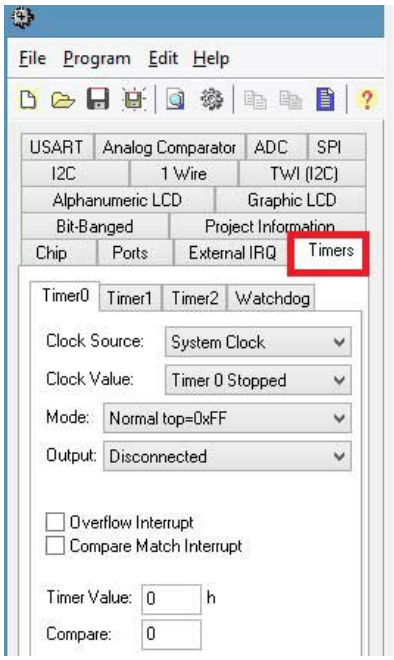
این رجیستر برای تنظیمات وقفه در هنگام سر ریز شدن تایمر/کانتر یا در هنگام تطبیق مقایسه (Compare Match) مورداستفاده قرار می‌گیرد.

رجیستر پرچم سرریز تایمر/کانتر (TIFR):

که در این رجیستر بیت TOV0 زمانی یک می‌شود که در یک سرریز تایمر یا کانتر صفر رخداده باشد و بیت‌های دیگر که همه توسط کد ویزارد تنظیم می‌شوند.

راه‌اندازی تایمر در کدویژن:

برای راه‌اندازی واحد Timer-Counter در میکروکنترلر AVR در نرم‌افزار CodeVision پس از ساخت یک پروژه در مرحله‌ی انتخاب ویژگی‌ها بخش Timers را همانند زیر انتخاب کرده است.

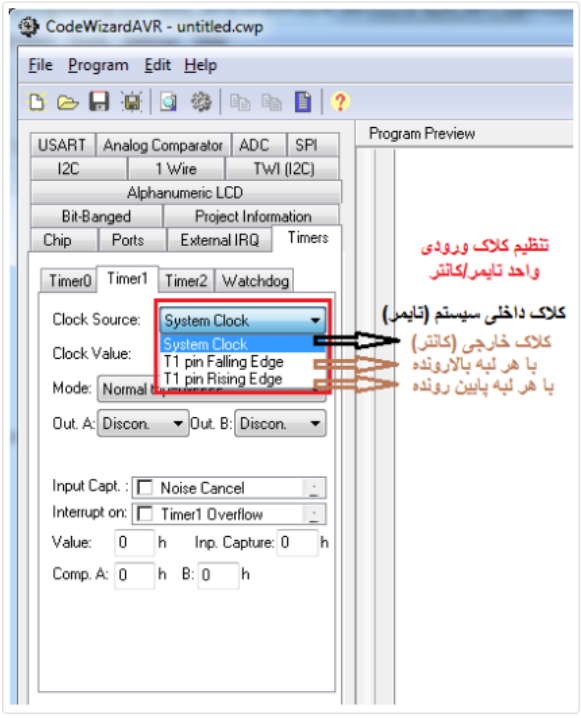


شکل ۵-۶

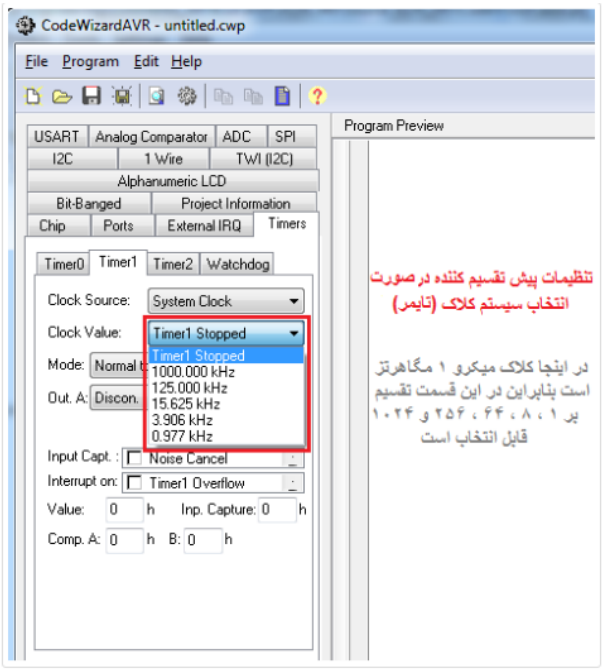
Timer0: این بخش مربوط به واحد Timer-Counter صفر است که از یک حافظه‌ی ۸ بیتی بهره می‌برد.

Timer1: این بخش مربوط به واحد Timer-Counter یک است که از یک حافظه‌ی ۱۶ بیتی بهره می‌برد.

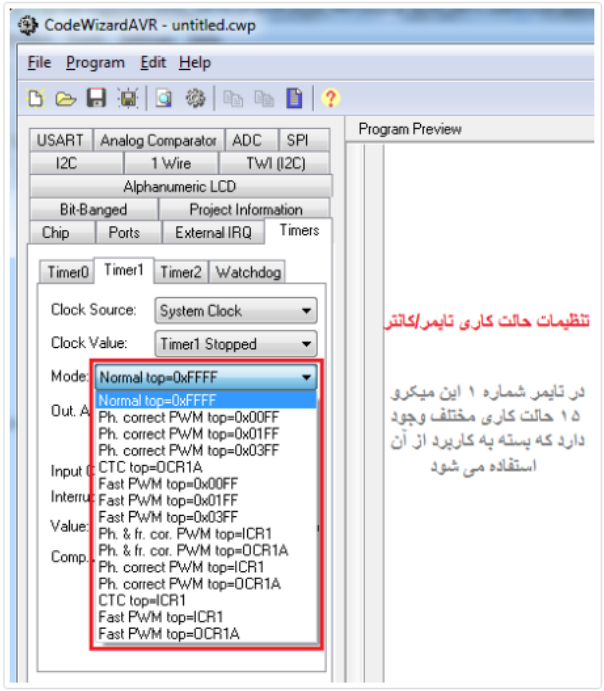
Timer2: این بخش مربوط به واحد Timer-Counter دو است که از یک حافظه‌ی ۸ بیتی بهره می‌برد.



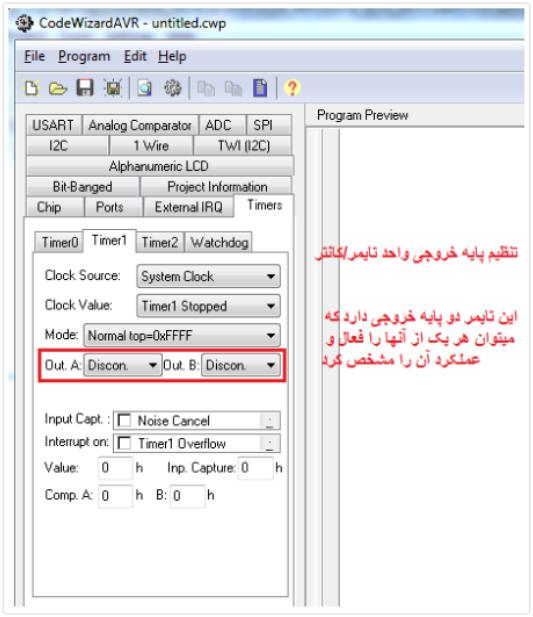
شکل ۶-۶



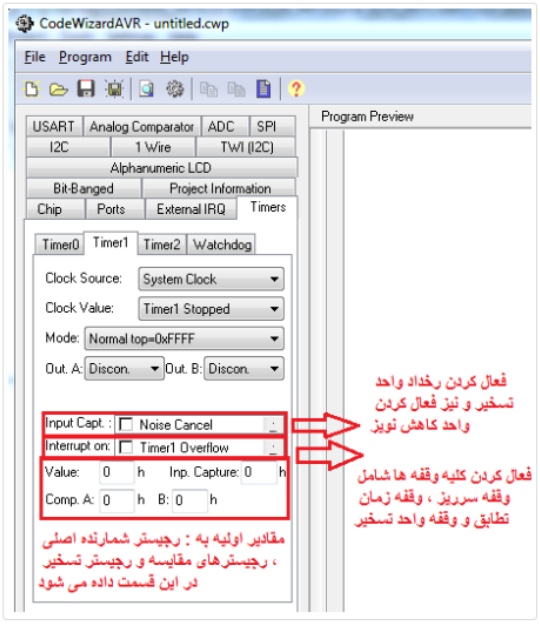
شکل ۶-۷



شکل ۶-۸



شکل-۹-۶



شکل ۱۰-۶

وقفه‌های خارجی

یک وقفه‌ی خارجی در اثر سیگنالی از یک وسیله جانبی خارج از میکرو که وارد پایه خاصی از میکرو می‌شود اتفاق می‌افتد. برای مثال در میکروکنترل ATMEGA32 سه پایه برای وقفه‌ی خارجی با نام‌های INT2, INT1, INT0 وجود دارد . برای اینکه بتوانیم از هرکدام از این پایه‌ها به‌عنوان وقفه خارجی استفاده کنیم باید در رجیسترهای مربوطه تنظیمات خاصی را انجام دهیم که در ادامه توضیح داده می‌شود.

رجیستر GICR

نام این رجیستر از Global interrupt control register گرفته‌شده است. با استفاده از بیت‌های شماره ۵، ۶ و ۷ که در شکل ۱۲ قابل‌مشاهده است می‌تواند قابلیت پاسخ‌گویی به وقفه‌های خارجی ۰ تا ۲ را فعال کند. INT0 برای وقفه خارجی صفر استفاده می‌شود و سایر وقفه‌های خارجی نیز به همین ترتیب فعال می‌شوند. البته باید توسط رجیستر SREG بیت فعال‌ساز وقفه عمومی فعال شود و توسط رجیستر MCUCR حساسیت به لبه یا سطح مشخص شود که در ادامه توضیح داده می‌شود.

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

شکل ۱۱-۶

رجیستر MCUCR

نام این رجیستر از MCU Control register گرفته‌شده است.

همان‌طور که گفتیم، وقفه‌های سخت‌افزاری یا حساس به سطح هستند یا حساس به لبه. با تنظیم بیت‌های رجیستر MCUCR می‌توان نوع حساسیت را تعیین کرد. برای INT0 از بیت‌های ۰, ۱ و برای INT1 از بیت‌های ۲ و ۳ استفاده می‌شود. در شکل ۱۲ این رجیستر را مشاهده می‌کنید:

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

شکل ۱۲-۶

در جدول ۱ نحوه تنظیم حساسیت به سطح یا لبه برای INT0 توضیح داده‌شده است:

جدول ۱-۶

ISC01	ISC00		توضیحات
0	0		سطح پایین روی پایه INT0 باعث تولید درخواست وقفه می گردد.
0	1		هر تغییر منطقی روی پایه INT0 باعث تولید درخواست وقفه می گردد.
1	0		لبه پایین رونده روی پایه INT0 باعث تولید درخواست وقفه می گردد.
1	1		لبه بالا رونده روی پایه INT0 باعث تولید درخواست وقفه می گردد.

تمام موارد گفته‌شده درباره INT0 در مورد INT1 هم صدق می‌کند. برای تنظیم نوع حساسیت INT1 کافی است در جدول ۱ بیت‌های ISC10 و ISC11 را به همین شیوه برای INT1 اعمال کنید.

رجیستر MCUCSR

نام این رجیستر از MCU Control and Status Register گرفته‌شده است.

نحوه حساسیت وقفه‌ی خارجی دو یا INT2 در ATMEGA32 توسط بیت ISC2 در ثبات MCUCSR تنظیم می‌شود. وقفه‌ی خارجی دو فقط در دو حالت تنظیم می‌شود.

- اگر ISC2 صفر باشد INT2 حساس به لبه پایین‌رونده می‌شود.
- اگر ISC2 یک باشد INT2 حساس به لبه بالارونده می‌شود.

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

شکل ۱۳-۶

رجیستر GIFR

نام این رجیستر از Global interrupt flag register گرفته‌شده است. این رجیستر پرچم وقوع وقفه‌های خارجی است. هر وقت یک وقفه‌ی خارجی رخ دهد، پرچم متناظر با آن‌یک می‌شود. همان‌طور که در شکل زیر می‌بینید، پرچم متناظر با وقفه‌ی خارجی صفر INTF0 است و برای وقفه‌های یک و دو نیز به همین ترتیب.

اما نکته‌ی مهمی که در رابطه با این پرچم‌ها وجود دارد صفر شدن آن‌ها است. اگر پاسخ‌گویی به وقفه را فعال کرده باشیم، با وقوع وقفه و پرش به آدرس بردار وقفه، پرچم متناظر هم صفر می‌شود؛ اما اگر به‌صورت نرم‌افزاری بخواهیم آن را صفر کنیم باید روی آن ۱ نوشته شود. این قانون برای تمامی پرچم‌های AVR صادق است.

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

شکل ۱۴-۶

طراحی فرکانس‌متر:

عملکرد فرکانس‌متر بدین‌صورت است که توسط تایمر صفر، زمانی به مدت یک ثانیه تولید و هم‌زمان با آن، تایمر یک نیز شروع به شمارش تعداد پالس‌های اعمالی بر روی پایه T1 می‌کند. پس از طی زمان یک ثانیه، مقدار شمارش‌شده در رجیستری تایمر ۱، معادل فرکانس موج روی پایه T1 است و می‌توان آن را بر روی LCD نمایش داد.

در برنامه برای تولید زمان یک ثانیه، می‌توان از تایمر صفر استفاده نمود. اگر فرکانس کاری میکرو ۸ مگاهرتز در نظر گرفته شود و فرکانس تایمر نیز ۸ مگاهرتز باشد، در این صورت اگر تایمر صفر (با توجه به اینکه در هر مرتبه سرریز خود ۲۵۶ عدد را با سرعت$\frac{1}{8000000}$ می‌شمارد)، ۳۱۲۵۰ مرتبه سرریز شود، ۱ ثانیه زمان طی خواهد شد:

```
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    cnt++;
    if(cnt>=31250)
    {
        frequency = TCNT1;
        TCNT1 = 0;
    }

}
```

کد زیر در تابع اینتراپت یک led را پس از ۱۰ ثانیه روشن می‌کند.

```
Interrupt [TIMO_OVF]void timer 0_ovf_isr(void)
{
Peak++;
If(peak==10)
{
Peak=0;
PORTD.5=1
}
Else
}
```

تمرین۱-۶: برنامه فرکانس‌متر را تکمیل نموده و با بستن سخت‌افزار مناسب، مدار را به‌گونه‌ای پیاده‌سازی کنید که فرکانس موج ورودی را اندازه گرفته و بر روی LCD با عبارت "Freq= xx Hz" نمایش دهد.

آزمایش هفتم: کنترل دور موتور (PWM)

هدف:

۱- آشنایی با PWM و تولید آن توسط میکروکنترلر
۲- آشنایی با درایورهای موتور DC

قطعات موردنیاز:

۱- موتور DC (آرمیچر) ---------- ۱ عدد
۲- آی‌سی L298 ---------- ۱ عدد
۳- 1N4007 ---------- ۴ عدد
۴- پتانسیومتر ۱۰ کیلو اهم --------- ۱ عدد

شرح آزمایش:

PWM چیست؟

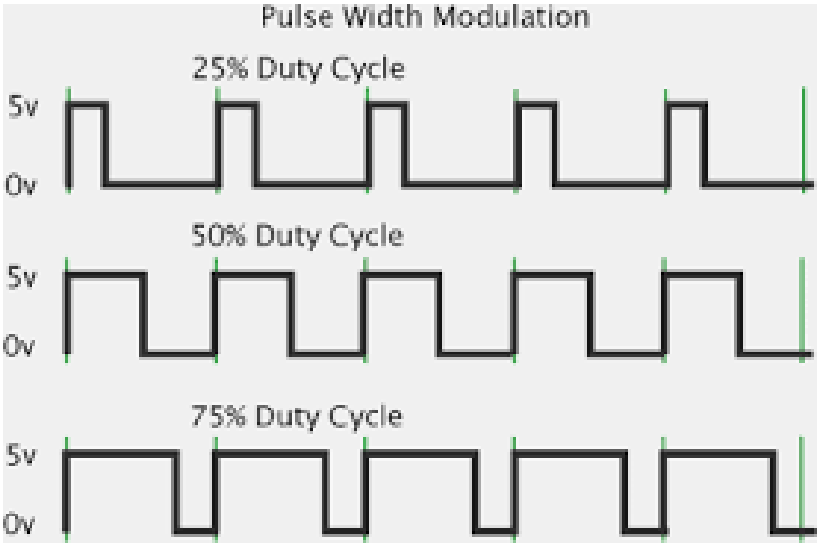
در بسیاری از موارد، ما نیاز به کنترل ولتاژ بر روی پایه‌های خروجی میکروکنترلر راداریم. مثلاً اگر بخواهیم سرعت موتور را کنترل کنیم، باید ولتاژی که بر روی موتور اعمال می‌شود را کنترل کرد. در حقیقت سرعت موتور تقریباً تابع مستقیمی از ولتاژی است که بر روی آن اعمال می‌شود. یعنی اگر ولتاژ کاریِ موتوری (ولتاژ استاندارد برای فعال‌سازی موتور که بر روی بدنه‌ی آن نوشته می‌شود.) ۱۲ ولت باشد، با اعمال ولتاژ ۶ ولت روی آن، می‌توانید سرعت چرخش آن (rpm)را حدوداً به نصف کاهش دهید. PWM تکنیکی است که به کمک آن می‌توانیم ولتاژ پایه‌های خروجی میکروکنترلر و درنتیجه سرعت موتور یا سایر قطعات جانبی که به میکروکنترلر متصل می‌شود را کنترل کنیم.

PWM مخفف واژه‌ی Pulse Width Modulation و به معنای مدولاسیون پهنای پالس است. همان‌طور که گفتیم PWM تکنیکی برای کنترل ولتاژِ پایه‌ی خروجی است. حال ببینیم چگونه با این تکنیک می‌توان ولتاژ خروجی را کنترل کرد. می‌دانیم که ولتاژ در پایه‌های خروجی میکروکنترلر یا ۰ است یا ۵ ولت، اما برای کنترل سرعت موتور، باید بتوانیم حداقل ولتاژ یکی از پایه‌ها را بین ۰ تا ۵ تغییر دهیم. PWM روشی است تا ما بتوانیم با استفاده از همین پایه‌ی خروجی معمولی، به‌نوعی ولتاژ را بین ۰ تا ۵ ولت تغییر دهیم.

به نسبت زمان یک بودن سیگنال به‌کل دوره سیگنال Duty cycle گویند:

$$Duty\ cycle = \frac{t_{on}}{T}$$

در رابطه بالا t_{on} مدت‌زمان یک بودن سیگنال در یک دوره آن و T، پریود سیگنال است. در زیر چند موج PWM با duty cycle های مختلف نشان داده‌شده است.

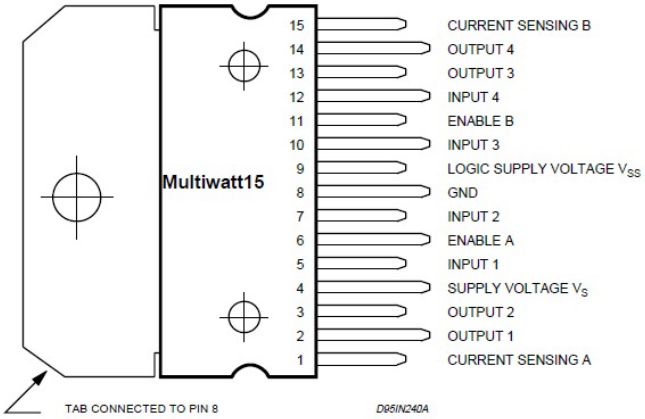


شکل ۱-۷

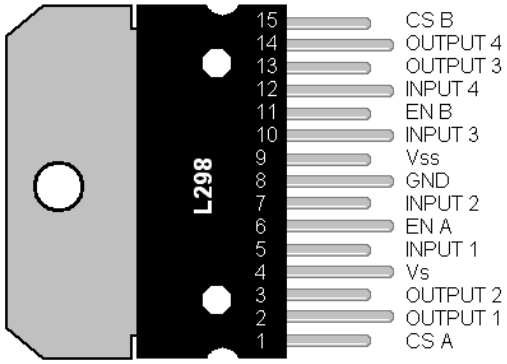
آی‌سی L298 شامل ۲ پل H برای کنترل موتور است. این IC به‌سادگی و با کمترین قطعات می‌تواند روشن و خاموش کردن موتور را برای شما مدیریت نماید. آی‌سی L298 هماهنگ با استاندارد TTL است. عموماً در راه‌اندازی موتورهای DC و پله‌ای استفاده می‌شوند.
دو کانال دارد که هر کانال برای چپ گرد و راست‌گرد کردن یک موتور DC کافی است.

پایه enable: هر کانال دارای یک‌پایه فعال‌ساز می‌باشد که با استفاده از آن می‌توان ولتاژ و جریان کانال مربوطه را قطع و وصل کرد. از این پایه معمولاً برای کنترل سرعت موتور استفاده می‌شود.

پایه CURRENT SENSING: امیتر ترانزیستورهای پایین آورنده باهم به یک‌پایه متصل شده‌اند که از این پایه برای اندازه‌گیری جریان استفاده می‌شود. هر کانال دارای یکی از این پایه‌ها است. این پایه‌ها باید با یک مقاومت اهم پایین (حدود ۰/۵ اهم) به زمین متصل شوند.



شکل ۲-۷



شکل ۳-۷

VS: ولتاژ تغذیه که برای خروجی (مثلاً موتور) استفاده می‌شود که حداکثر آن ۵۰ ولت برای این آی‌سی می‌باشد. البته توصیه‌شده از ۴۶ ولت بیشتر نشود

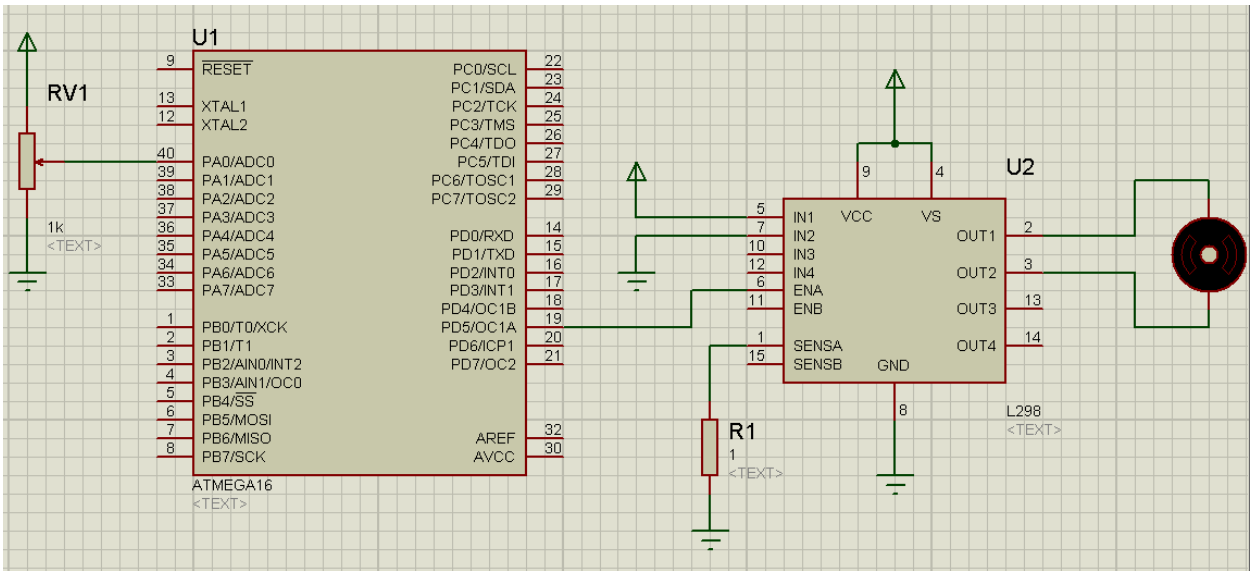
VSS: ولتاژ تغذیه‌ای که برای گیت‌های منطقی استفاده می‌شود. حداکثر ولتاژ آن می‌تواند ۷ ولت باشد.

OUTPUT: هر کانال دارای دو خروجی می‌باشد که هر خروجی به یک‌پایه موتور DC متصل می‌شود.

GND: به زمین وصل می‌شود.

سخت‌افزار نشان داده‌شده در شکل ۳ برای راه‌اندازی موتور توسط درایور l298 می‌باشد. توسط پایه‌های IN1 و IN2 می‌توان جهت موتور را تغییر داد و یا موتور را در حالت قفل قرارداد.

نکته مهم: در شکل ۴ فرض شده که ولتاژ کاری موتور با ولتاژ کاری میکرو برابر است (۵ ولت) درصورتی‌که ولتاژ موتور متفاوت باشد بایستی ولتاژ موتور به پایه VS متصل گردد و این پایه (VS) نباید به VCC آی سی متصل شود.



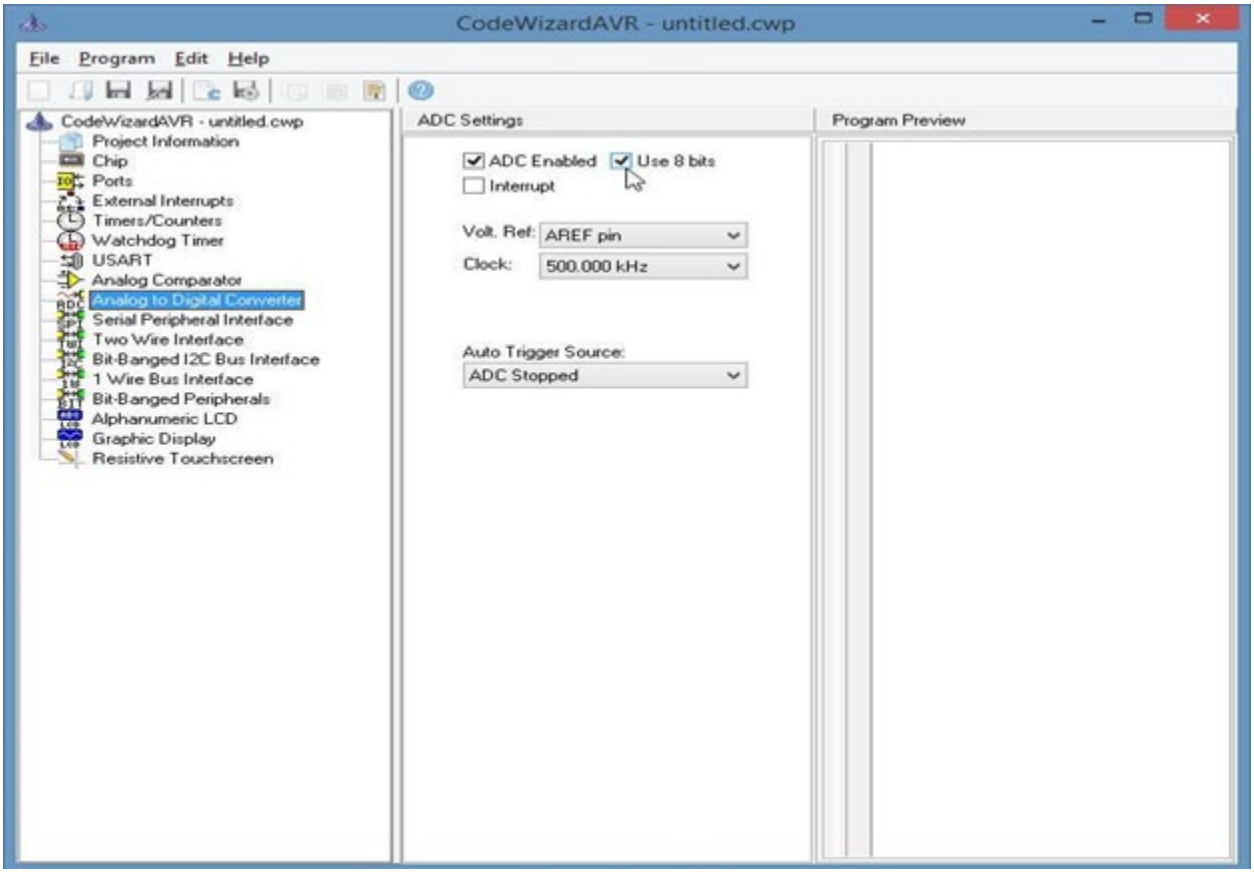
شکل ۴-۷: مدار اتصال موتور به میکروکنترلر

تمرین ۱-۷: جدول حالاتی که می‌توان با پایه‌های IN1 و IN2، موتور را کنترل کرد، رسم نمایید.

تمرین ۲-۷: سخت‌افزار نشان داده‌شده در شکل ۳ را ببندید و برنامه‌ای بنویسید که مقدار پتانسیومتر توسط ADC خوانده‌شده و بر اساس ولتاژ خروجی پتانسیومتر، سرعت موتور تغییر پیدا کند.

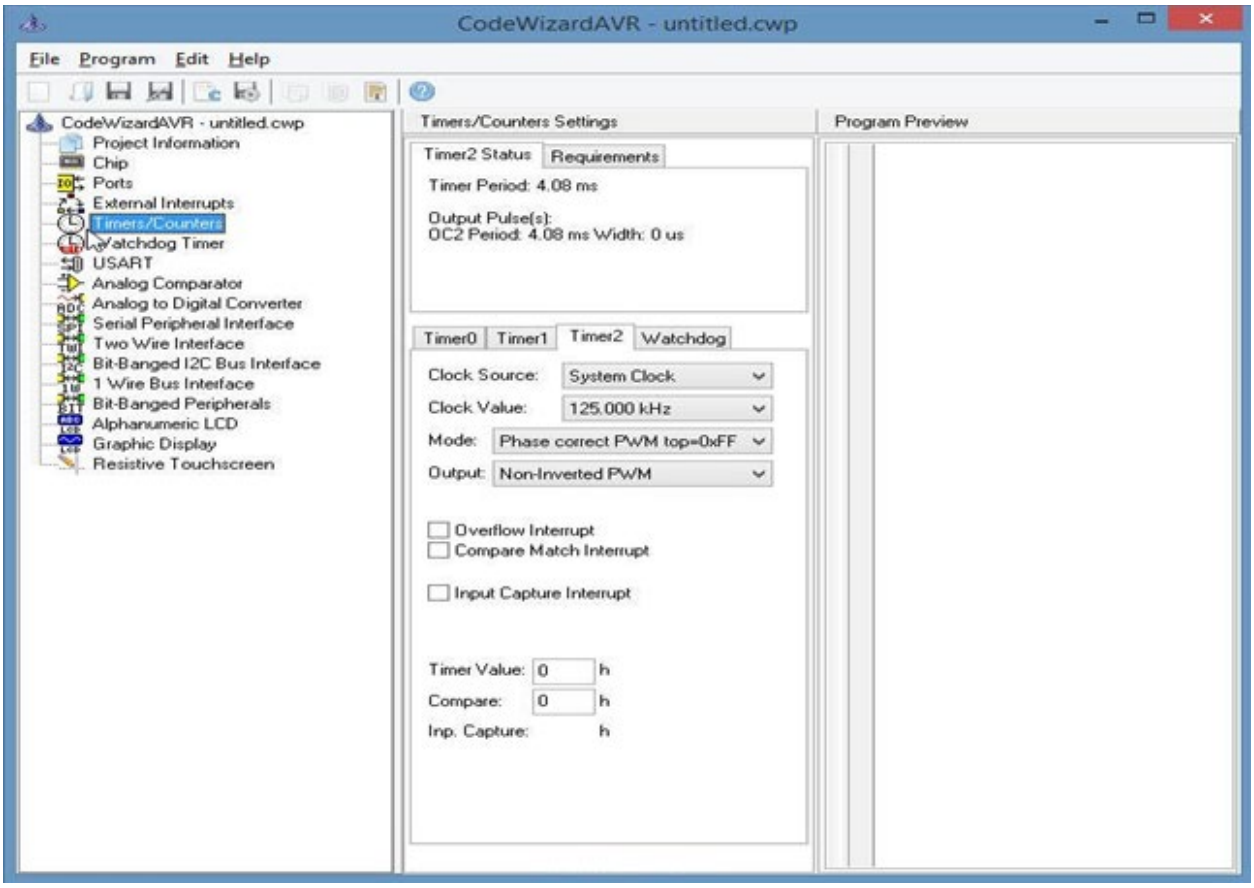
تنظیمات کدویزارد:

در نرم‌افزار کدویژن یک پروژه جدید ایجاد کنید و در محیط کدویزارد میکرو کنترلرATmea16 انتخاب کرده و فرکانس ۱ مگاهرتز تنظیم کنید و پین ۵ از پورت D را برای تولید موج PWM خروجی کنید. در این پروژه با اندازه‌گیری ولتاژ آنالوگی که به پین ۴۰ میکرو توسط پتانسیومتر اعمال می‌شود موج PWM متناسب با آن تولید می‌کنیم؛ بنابراین لازم است که ADC میکرو را هم فعال کنیم تنظیمات کدویزارد ADC به‌صورت زیر است:



شکل ۵-۷

توجه کنید که حتماً تیک Use8bits را فعال کنید. میکرو ATmega16 دارای ۳ تایمر می‌باشد که از آن‌ها می‌توان برای تولید موج PWM استفاده کرد:


CodeWizardAVR - untitled.cwp
File Program Edit Help
CodeWizardAVR - untitled.cwp
Project Information
Chip
Ports
External Interrupts
Timers/Counters
Watchdog Timer
USART
Analog Comparator
Analog to Digital Converter
Serial Peripheral Interface
Two Wire Interface
Bit-Banged I2C Bus Interface
1 Wire Bus Interface
Bit-Banged Peripherals
Alphanumeric LCD
Graphic Display
Resistive Touchscreen
Timers/Counters Settings
Program Preview
Timer2 Status
Requirements
Timer Period: 4.08 ms
Output Pulse(s):
OC2 Period: 4.08 ms Width: 0 us
Timer0
Timer1
Timer2
Watchdog
Clock Source: System Clock
Clock Value: 125.000 kHz
Mode: Phase correct PWM top=0xFF
Output: Non-Inverted PWM
Overflow Interrupt
Compare Match Interrupt
Input Capture Interrupt
Timer Value: 0 h
Compare: 0 h
Inp. Capture: h

شکل ۶-۷

آزمایش هشتم: راه‌اندازی آی‌سی ساعت

هدف:

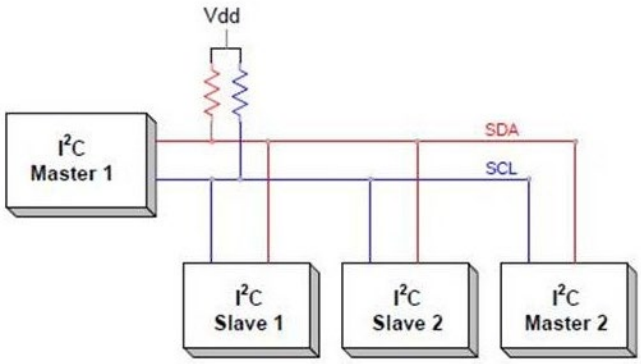
۱- آشنایی با ارتباط I2C
۲- راه‌اندازی آی‌سی DS1307 با ارتباط I2C

قطعات موردنیاز:

۱- آی‌سی DS1307 ---------- ۱ عدد
۲- کریستال 32.768k ---------- ۱ عدد
۳- باتری سکه‌ای ۳ ولت + جا باتری سکه‌ای ---------- ۱ عدد
۴- مقاومت ۱۰ کیلو اهم ---------- ۲ عدد
۵- کلید فشاری ۲ پایه ---------- ۳ عدد

شرح آزمایش:

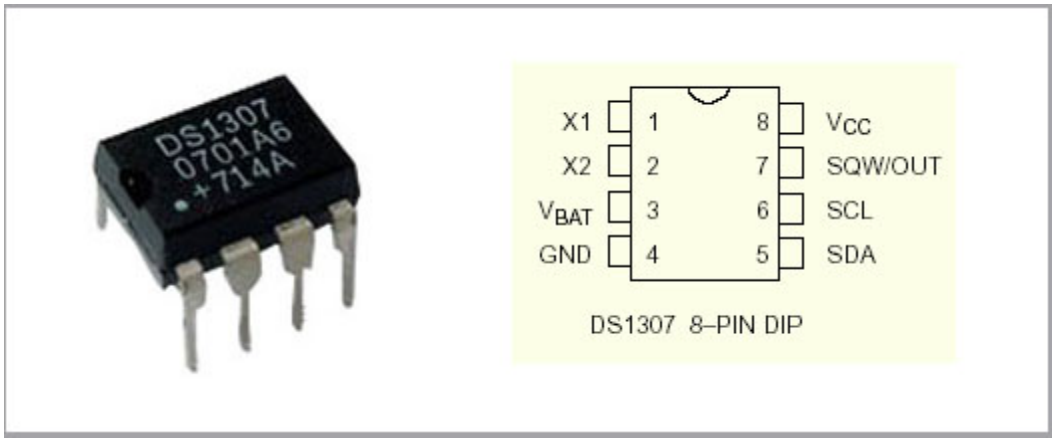
ارتباط I2C یا (Inter integrated circuits) یک نوع گذرگاه ارتباطی است که در بسیاری از مدارهای مجتمع ازجمله سنسورها، حافظه‌ها و rtc ها جهت تبادل داده پیاده‌سازی شده است. بسیاری این رابط را رابط دوسیمه یا (Two Wire Interface) می‌نامند. در میکرو کنترلر AVR دوپایه بانام‌های SCK و SDA به این گذرگاه اختصاص داده‌شده است. پایه SCK برای ایجاد پالس جهت هم‌زمانی ارتباط بکار می‌رود و پایه SDA جهت ارسال و دریافت داده بکار می‌رود. پایه‌های مذکور ازلحاظ الکتریکی کلکتور باز هستند؛ یعنی برای استفاده از آنها باید با یک مقاومت ۴/۷ کیلو اهمی به مثبت مدار متصل شوند. اگر چندین دستگاه توسط دو خط مربوط به i2c به یکدیگر متصل شوند هر خط تنها وقتی یک می‌ماند که هیچکدام از دستگاه‌های متصل، آن را یک نکرده باشند. هنگامی که خط در حالت بیکاری باشد دو پایه I2C در حالت یک قرار می‌گیرند. در avr تا ۱۲۰ دستگاه مختلف می‌توانند از طریق رابط i2c به آن متصل شوند که به هر کدام از این اتصال ها یک گره می‌گویند. هرکدام از این دستگاه‌ها می‌توانند یا فرمانده (Master) باشند و یا فرمانبر (Slave). دستگاه فرمانده گره‌ای است که وظیفه تولید پالس ساعت و آغاز و پایان تبادل داده را بر روی خط به عهده دارد. گره فرمانبر دریافت‌کننده پالس ساعت است و توسط فرمانده آدرس‌دهی می‌شود. هر فرمانبر و فرمانده می‌توانند در دو حالت دریافت‌کننده یا ارسال‌کننده عمل کنند.



شکل ۱-۸

توضیح مختصری درباره DS1307

DS 1307 یک آی‌سی ساعت و تقویم باقابلیت شمارش ثانیه، دقیقه، ساعت، روز هفته، روز ماه، ماه و سال می‌باشد. این آی‌سی علاوه بر قابلیت نگهداری ساعت و تقویم، ۵۶ بایت رم آزاد نیز دارد که می‌توان برای نگهداری داده از آن استفاده نمود. این آی‌سی توسط پروتکل ارتباط سریال I2C به میکروکنترلر و ... متصل می‌شود. ترتیب پایه‌های DS1307 مطابق شکل ۲ و به شرح زیر می‌باشد:



شکل ۲-۸

پایه ۱ و ۲ به کریستال ساعت (۳۲.۷۶۸kHz) متصل می‌شود.

پایه ۳ به باطری وصل می‌شود.

پایه ۴ به زمین است.

پایه ۵(SDA): ارسال و دریافت اطلاعات از طریق پورت I2C است.

پایه ۶(SCL) : کلاک هماهنگی بین میکروکنترلر AVR و آی‌سی DS1307 است.

پایه ۷ پایه خروجی پالس مربعی با فرکانس‌های4.096KHZ و 8.192KHZ و32.768KHZ و۱Hz است که می‌توان آن را در کدیژن راه اندازی کرد.

پایه ۸ به VCC وصل می‌شود (۵ ولت)

برای خواندن و نوشتن اطلاعات از دستورات زیر استفاده می‌کنیم.

با این دستور به ساعت، دقیقه و ثانیه مقدار اولیه می‌دهیم:

```
rtc_set_time(0,0,0);
```

این دستور مقدار فعلی ساعت، دقیقه و ثانیه را از آی‌سی ساعت می‌خواند.

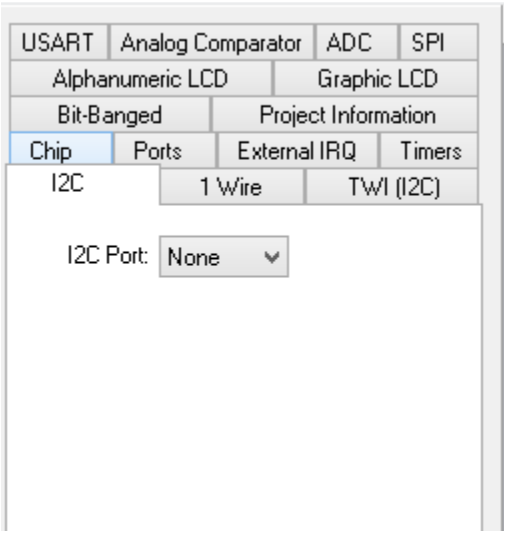
نکته: متغیرهای این دستور باید به‌صورت کاراکتری تعریف شوند: (char h, m, s)

```
rtc_get_time(&h,&m,&s);
```

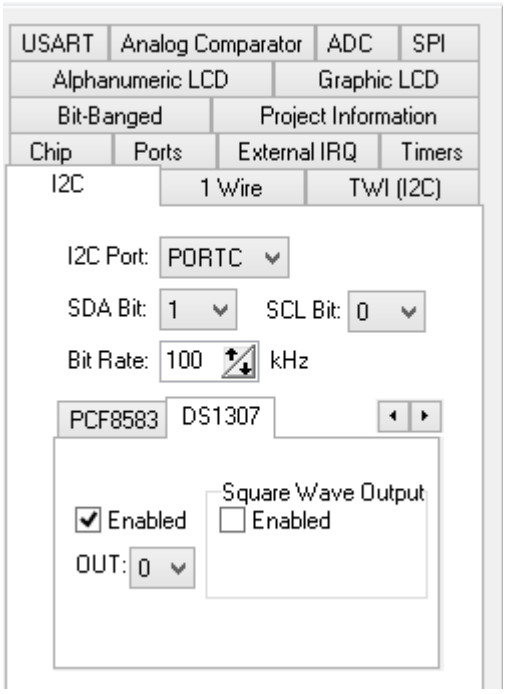
برای نوشتن کد مربوط به این قسمت می‌خواهیم از help کد ویژن استفاده کنیم پس بر روی علامت سؤال در نوار بالا کلیک کرده و در قسمتsearch آی‌سی ساعت یا همان DS1307 را می‌نویسیم پس از این کار کدهای آماده نشان داده می‌شود که برای این آزمایش به شرح زیر است:

```
while (1)
      {
      rtc_get_time(&hour,&min,&sec);

      rtc_get_date(&week_day,&day,&month,&year);

      sprintf(display_buffer,"Time: %2d:%02d:%02d\n",hour,min,sec);

      lcd_clear();

      lcd_puts(display_buffer);

      sprintf(display_buffer,"Date: %2d/%02d/%d",day,month,2000+year);

      lcd_puts(display_buffer);

      delay_ms(500);
      }
```

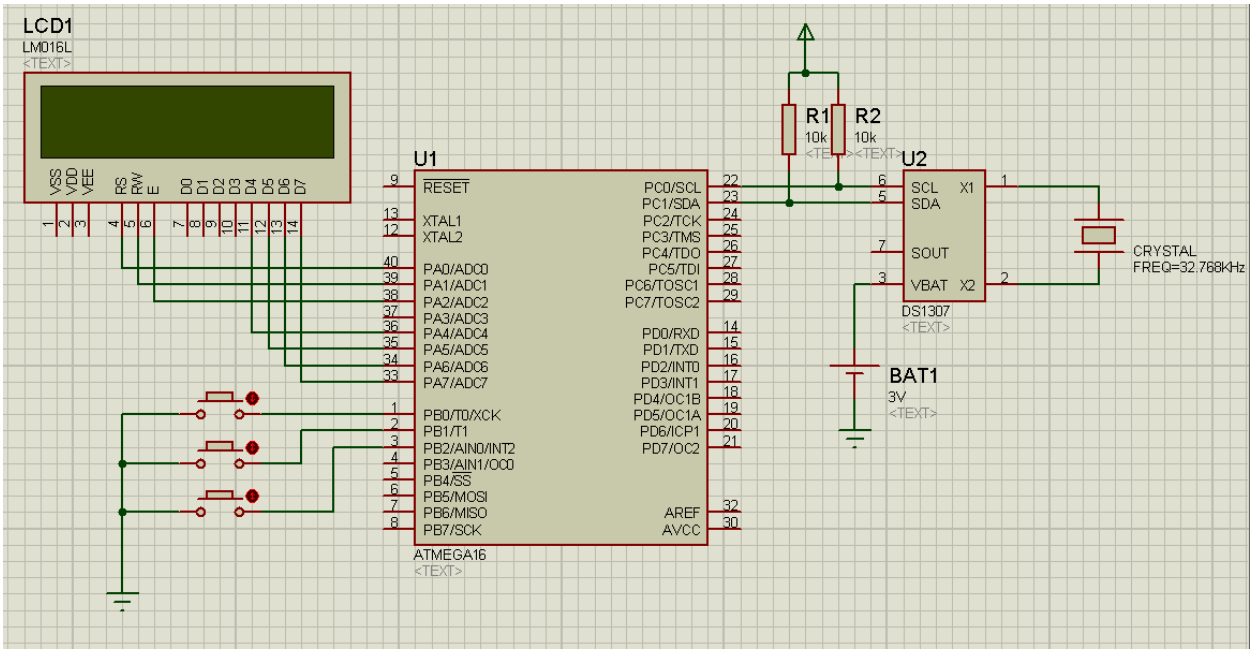
تنظیمات کدویژن برای راه‌اندازی DS1307 به‌صورت زیر انجام می‌گیرد:


USART
Analog Comparator
ADC
SPI
Alphanumeric LCD
Graphic LCD
Bit-Banged
Project Information
Chip
Ports
External IRQ
Timers
I2C
1 Wire
TWI (I2C)
I2C Port:
None

شکل ۳-۸


USART
Analog Comparator
ADC
SPI
Alphanumeric LCD
Graphic LCD
Bit-Banged
Project Information
Chip
Ports
External IRQ
Timers
I2C
1 Wire
TWI (I2C)
I2C Port:
PORTC
SDA Bit:
1
SCL Bit:
0
Bit Rate:
100
kHz
PCF8583
DS1307
Square Wave Output
Enabled
Enabled
OUT:
0

شکل ۴-۸



شکل ۵-۸

تمرین ۱-۸: شماتیک نشان داده‌شده در شکل ۵ را بسته و برنامه‌ای بنویسید که زمان و تاریخ را بر روی lcd نمایش دهد.

تمرین ۲-۸: نحوه استفاده از کلیدهای ۱، ۲ و ۳ را توضیح دهید؟ (خواندن کلیدها در برنامه به چه صورت انجام می‌گیرد؟)

تمرین ۳-۸: برنامه نوشته‌شده در تمرین ۱ را به‌گونه‌ای تکمیل نمایید تا به کمک سه کلید استفاده‌شده، بتوان ساعت، دقیقه و ثانیه را تنظیم کرد.

تمرین ۴-۸: علت استفاده از مقاومت‌های R1 و R2 را بنویسید.

تمرین ۵-۸: عملکرد مدار را با قطع و وصل تغذیه مدار در دو حالت با باتری و بدون باتری پشتیبان مقایسه کنید.

آزمایش نهم: تشخیص مانع با سنسور مادون‌قرمز

هدف:

۳- آشنایی با تایمرها و تولید موج با فرکانس مشخص توسط میکروکنترلر
۴- آشنایی با وقفه‌های خارجی
۵- آشنایی با سنسورهای مادون‌قرمز

قطعات موردنیاز:

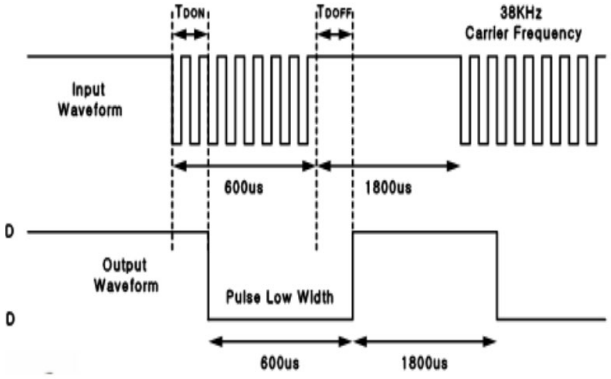
۱- فرستنده مادون‌قرمز ---------- ۱ عدد
۲- گیرنده مادون‌قرمز (سه‌پایه) TSOP-1837 ---------- ۱ عدد
۳- خازن uf۴/۷ (V۶/۳) ---------- ۱ عدد
۴- مقاومت ۱۰۰ اهم ---------- ۱ عدد

شرح آزمایش:

امروزه انتقال اطلاعات توسط پرتو مادون‌قرمز (Infra-Red) عملی رایج و پرطرفدار است، برای مثال، کنترل وسایل صوتی و تصویری منزل یکی از انبوه کاربردهای آن می‌باشد. انتقال اطلاعات بین گوشی‌های تلفن همراه از طریق واسط مادون‌قرمز، برقراری ارتباط در شبکه‌های کامپیوتری و حتی استفاده در اسباب‌بازی‌ها و ... از دیگر کاربردهای آن می‌باشند.

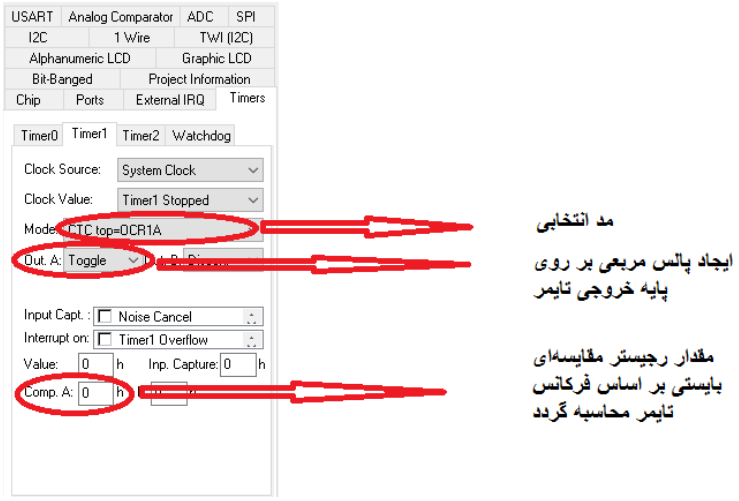
فرستنده مادون‌قرمز:

در یک طراحی برای استفاده از هر قطعه الکترونیکی بایستی از دیتاشیت آن قطعه استفاده گردد. با توجه به اینکه گیرنده مادون‌قرمز استفاده‌شده در این آزمایش یک گیرنده 38kHz است، بنابراین بایستی توسط فرستنده یک پالس 38KHz تولید شود. از طرف دیگر در دیتاشیت گیرنده TSOP-1837 شکل موج فرستنده مانند شکل ۱ رسم کرده است.



شکل ۱-۹: شکل موج ارسالی توسط فرستنده و نمایش خروجی گیرنده

برای تولید این شکل موج از مد CTC TOP= OCR1A استفاده می‌شود و خروجی تایمر را فعال می‌کنیم.

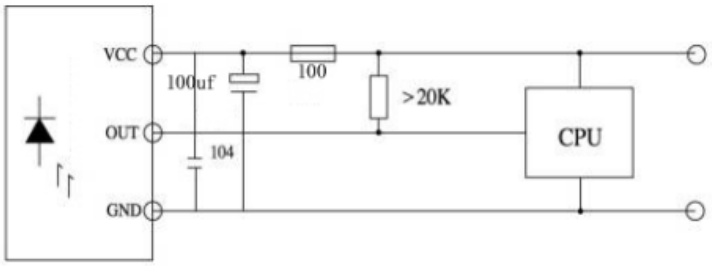


شکل ۲-۹: تنظیمات تایمر

دقت شود در شکل ۲، تنها تنظیمات مد و خروجی تایمر نشان داده‌شده است و سایر تنظیمات (مانند فرکانس تایمر و مقدار رجیستر OCR1A) در شکل نمایش داده نشده است.

گیرنده مادون‌قرمز (VB1837)

شماتیک راه‌اندازی این گیرنده در شکل ۳ نشان داده‌شده است.



شکل ۳-۹

تمرین ۱-۹- شکل موج موردنیاز برای گیرنده VB1837 را توسط تایمر میکرو ساخته و بر روی اسیلوسکوپ نمایش دهید.

تمرین ۲-۹- با اعمال شکل موج ساخته‌شده در تمرین ۱ به یک عدد گیرنده مادون‌قرمز و قرار دادن سنسور گیرنده در مقابل آن، خروجی سنسور گیرنده را بر روی اسیلوسکوپ نمایش دهید.

تمرین ۳-۹- برنامه‌ای بنویسید که وجود و یا عدم وجود مانع را تشخیص داده و بر روی LCD وضعیت را نمایش دهد.

مراحل کلی انجام یک پروژه میکروکنترلری:

به‌طورکلی وقتی‌که قرار است یک پروژه با میکروکنترلرهای انجام دهید، بعد از مشخص شدن هدف AVR پروژه و صرفه اقتصادی آن مراحل زیر به وجود می‌آید:

۱. طراحی سخت‌افزار: در این مرحله می‌بایست بر اساس هدف پروژه و شرایط مکانی به‌کارگیری پروژه، نوع و مقدار تک تک المان‌های سخت‌افزار موردنیاز طراحی و روی کاغذ آورده شود.
۲. طراحی نرم‌افزار: در این مرحله ابتدا الگوریتم یا فلوچارت موردنیاز رسم و سپس برنامه‌نویسی موردنظر بر اساس آن نوشته می‌شود.
۳. شبیه‌سازی: قبل از پیاده‌سازی عملی، تست صحت عملکرد مدار در این مرحله توسط نرم‌افزارهای مناسب (در اینجا CodeVision و Proteus) صورت می‌گیرد.
۴. پیاده‌سازی: پروگرام کردن میکروکنترلر و بستن مدار موردنظر روی برد بورد در این مرحله صورت می‌گیرد.
۵. تست و عیب‌یابی: با وصل منبع تغذیه به مدار، تست و عیب‌یابی مدار در این مرحله صورت می‌گیرد.