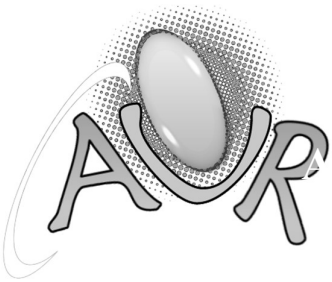
۵۳ پروژه کاربردی با میکروکنترلرهای



مولف: دکتر حامد سقایی

بسم الله الرحمن الرحیم

نهال را باران باید
تا بشوید غبار نشسته بر برگهایش
و سیرابش کند از آب حیات
و آفتاب باید
تا بتاباند
نیرو را
و محکم کند
شاخه‌های تازه روییده را

بر نام مادر
بوسه‌ای باید زد
دست‌هایی که
می‌شویند غبار خستگی روزگار را
و سیراب می‌کنند روح تشنه را
بر نام پدر
بوسه‌ای باید زد
دست هایی را
که می‌تابانند
نیرو را
و محکم می‌کنند
استواری پایه‌های زیستن را

مقدمه:

امروزه استفاده از تراشه‌های مختلف میکروکنترلر در ساخت و کنترل پروژه‌های مختلف آزمایشگاهی و صنعتی به عنوان ابزاری قدرتمند در خدمت طراحان قرار گرفته است و با ظهور این تراشه‌های منطقی برنامه‌پذیر، مدارات دیجیتال پیچیده، جای خود را به این تراشه‌ها داده‌اند.

اگرچه قابلیت‌ها و سرعت اجرای برنامه‌های نوشته شده به زبان اسمبلی غیر قابل انکار است ولی استفاده از زبان C در برنامه‌نویسی میکروکنترلرها به دلیل نزدیکی زیاد به سخت‌افزار (سطح میانی بودن زبان)، ساختار یافتگی و امکان استفاده از توابع نوشته شده آن در سایر پروژه‌ها مورد توجه بسیاری از کاربران قرار گرفته است. زبان دیگری که جذابیت زیادی را در برنامه‌نویسی میکروکنترلرهای AVR ایجاد کرده است، Basic می‌باشد. این زبان ساده‌ترین زبان برنامه‌نویسی میکروکنترلرهای AVR است.

در این کتاب سعی شده، قابلیت‌های میکروکنترلرهای سری Mega در قالب مثال های متنوع و کاربردی تشریح شوند. همچنین فرض شده است که خوانندگان این کتاب از آشنایی مقدماتی با میکروکنترلرهای AVR برخوردار هستند. مثال‌های کاربردی مطرح شده در این کتاب غالباً به گونه‌ای است که در کمتر کتابی یافت می شوند و کاربردهای این میکروکنترلرها را در مباحث مختلف الکترونیک، مخابرات، الکترونیک قدرت، رباتیک و نظایر آن نشان می‌دهند.

در هر فصل سعی شده است ضمن معرفی رجیسترهای مربوط به هر یک از قسمت‌های میکروکنترلر، تنظیمات مورد نیاز آن در محیط Wizard مربوط به نرم‌افزار CodeVision توضیح داده شوند و سپس مثال‌های متنوعی از کاربردهای آن بخش به زبان C ذکر گردند و نکات مهم آن برنامه‌ها در متن کتاب تشریح شوند.

مثال‌های مطرح شده در آن برای انواع میکروکنترلرهای سری Mega صادق هستند ولی به دلیل تنوع محصولات، قابلیت‌های موجود در تراشه ATmega16 به عنوان مبنای اصلی مطالب در این کتاب در نظر گرفته شده است. هر چند که در برخی از مثال‌ها از تراشه‌های دیگری استفاده شده است. به دلیل تنوع زیاد کامپایلرها در این کتاب، بیشترین تاکید بر کامپایلرهای زبان C و مخصوصاً CodeVision است. هر چند در فصل آخر کتاب، کامپایلر زبان Basic یعنی Bascom و کامپایلر قدرتمند دیگر زبان C یعنی WinAVR آموزش داده و پروژه‌های کاربردی مرتبط با آن‌ها ارائه می‌شوند.

به لطف ایزد منان، این مجموعه که حاصل تجربیات و مطالعات چندین ساله مولف بر روی سیستم‌های دیجیتال برنامه‌پذیر است، تکمیل شده است. لازم به ذکر است که:

تمانی برنامه‌های نوشته شده در کتاب، در CD همراه کتاب نیز قرار داده شده و به طور کامل در آزمایشگاه ریزپردازنده آزمایش و بررسی شده‌اند. در CD همراه کتاب، علاوه بر برنامه‌های کتاب، تمامی فایل‌های مرتبط و فایل‌های اجرایی توسط نرم‌افزار شبیه‌ساز Proteus نیز وجود دارند تا خواننده کتاب، تنها به تحلیل، بررسی و اجرای آن‌ها بپردازد. همچنین متناسب با نیاز خوانندگان کتاب: آموزش نرم‌افزارهای مرتبط با کتاب نیز در CD قرار داده شده تا خواننده در میان دنیای تراشه‌های برنامه‌پذیر احساس تنهایی نکند. شایان ذکر است که پروژه‌های این کتاب به سادگی قابل ترکیب و استفاده در پروژه‌های بزرگ‌تر می‌باشند. امید است به لطف خداوند یکتا، این کتاب، راه را برای یادگیری، طراحی و پیاده‌سازی رهیافت‌های نوین علمی و عملی برای شما هموار سازد.

در پایان بر خود لازم می‌دانم، از اساتید بزرگواری که در دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)، از آنان بسیار آموختم، تقدیر به عمل آورم و این کتاب را نتیجه زحمات بسیار آن عزیزان بدانیم.

با توجه به این مهم که این کتابی عاری از اشکال نیست، در صورتی که اشکال یا اشکالاتی را در کتاب مشاهده فرمودید، ما را بخشیده و نظرات اصلاحی خود را به آدرس الکترونیکی h.saghaei@gmail.com یا از طریق وب سایت زیر برای ما ارسال نمائید تا در چاپ‌های بعدی اصلاح شوند. پیشاپیش از شما سپاسگزاریم.

فهرست

فصل اول میکروکنترلر و واسطهای جانبی

در این فصل ابتدا به اختصار به معرفی میکروکنترلر AVR مورد استفاده در این کتاب پرداخته می‌شـــود. ســـپس انواع روش‌های برنامه‌ریزی میکروکنترلرهای AVR بیان می شوند. همچنین روش نصب و کار با نرم‌افزار CodeVision آموزش داده می شود و در پایان، به منظور آشـــنایی مقدماتی خوانندگان کتاب، دســـتورات لازم زبان برنامه‌نویسی C برای کار با میکروکنترلرها به اختصار توضیح داده خواهند شد.

۱-۱- معرفی میکروکنترلرهای AVR

۱-۱-۱- مقدمه

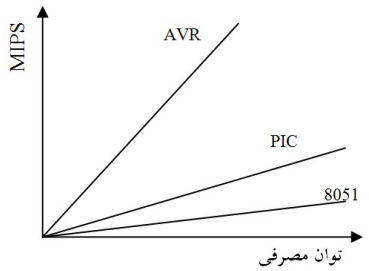
«زبان‌های سطح بالا»[1] به سرعت در حال تبدیل شدن به زبان برنامه‌نویسی استاندارد برای میکروکنترلرها (MCU) هستند. زبان برنامه‌نویسی Basic و C بیشترین استفاده را در برنامه‌نویسی میکروکنترلرها دارند ولی در اکثر کاربردها، کدهای بیشتری را نسبت به زبان برنامه‌نویسی اسمبلی تولید می‌کنند. شرکت ATMEL با ایجاد تحولی در معماری ساخت تراشه‌های برنامه‌پذیر، از معماری RISC[2] برای ساخت میکروکنترلرهای AVR استفاده می‌کند که این معماری در مقایسه با معماری CISC دارای دستورات با اندازه (سایز) ثابت، رجیسترهای (ثبات‌های) همه منظوره بیشتر و دستورات ساده‌تر و قابل اجرا تنها در یک یا دو پالس ساعت هستند، می‌باشد که در نهایت منجر به افزایش سرعت اجرای دستورات توسط میکروکنترلر می‌شود. سرعت اجرای دستورات در میکروکنترلرهای AVR در مقایسه با میکروکنترلرهای ۸۰۵۱ و PIC به ترتیب ۱۲ و ٤ برابر می‌باشند.

تکنولوژی حافظه کم مصرف غیر فرار برای برنامه‌ریزی AVR مورد استفاده قرار گرفته است. در نتیجه حافظه‌های FLASH و EEPROM در داخل مدار قابل برنامه

---

[1] High Level Languages (HLL).
[2] Reduced Instruction Set Computer (RISC).

ریزی هستند. میکروکنترلرهای اولیه AVR دارای ۱ ، ۲ و ۸ کیلوبایت حافظه FLASH و به صورت کلمه‌های ۱٦ بیتی سازماندهی شده بودند. AVR ها به عنوان میکروکنترلرهای RISC با دستورات فراوان طراحی شده‌اند که باعث می‌شود حجم کد تولید شده کم و سرعت بالاتری حاصل شود. با انجام دستورات تک سیکلی، اسیلاتور با کلاک داخلی سیستم یکی می‌شود. هیچ تقسیم‌کننده‌ای در داخل AVR قرار ندارد که منجر به اختلاف فاز کلاک سیستم با سرعت اجرای دستورات شود. بیشتر میکروکنترلرها، فرکانس کلاک اسیلاتور سیستم را بر ٤ یا ۱۲ تقسیم می‌کنند که این امر منجر به کاهش سرعت اجرای دستورات می‌شود. بنابراین AVR ها ٤ تا ۱۲ بار سریعتر و توان مصرفی آن‌ها نیز ٤ تا ۱۲ بار نسبت به میکروکنترلرهای کنونی کمتر است. زیرا در تکنولوژی CMOS استفاده شده در میکروکنترلر های AVR، مصرف توان سطح منطقی متناسب با فرکانس است. شکل (۱-۱)، افزایش MIPS[1] را به علت انجام عملیات تک سیکلی AVR در مقایسه با PIC و ۸۰۵۱ نشان می‌دهد.



شکل (۱-۱): نمودار توان مصرفی برحسب اجرای میلیون دستور بر ثانیه برای میکروکنترلرهای AVR، PIC و ۸۰۵۱

۱-۱-۲- طراحی برای زبان‌های Basic و C:

زبآن‌های Basic و C بیشترین استفاده را در دنیای امروز به عنوان زبآن‌های HLL دارند. پیش از طراحی چنین معماری، بیشتر میکروکنترلرها برای زبان اسمبلی طراحی شده و کمتر از زبآن‌های HLL حمایت می‌کردند. هدف شرکت ATMEL طراحی معماری بود که هم برای زبان اسمبلی و هم برای زبآن‌های HLL مفید باشد. به عنوان مثال در زبآن‌های Basic و C می‌توان یک متغیر محلی به جای متغیر سراسری در داخل زیربرنامه تعریف کرد. بنابراین تنها در زمان اجرای زیربرنامه مکانی از حافظه

[1] Million Instruction Per Seconds (MIPS).

SRAM برای متغیر اشغال می‌شود. در صورتی که اگر متغیر به عنوان متغیر سراسری تعریف گردد، در تمام زمان اجرای برنامه مکانی از حافظه SRAM را اشغال کرده است. برای دسترسی سریع‌تر به متغیرهای محلی و کاهش کد، نیاز به افزایش رجیستر های همه منظوره است. AVR ها دارای ۳۲ رجیستر همه منظوره هستند که مستقیما به «واحد محاسبه و منطق»[1] متصل شده‌اند و تنها در یک پالس ساعت به این واحد دسترسی پیدا می‌کنند. سه جفت از این رجیسترها می‌توانند به عنوان رجیسترهای ١٦ بیتی استفاده شوند. نتیجه تمام موارد بیان شده، این است که میکروکنترلرهای AVR با سرعت بالا و سازماندهی RISC هستند.

میکروکنترلرهای AVR به سه خانواده اصلی AT90S AVR، Tiny AVR و Mega AVR تقسیم‌بندی می‌شوند. البته خانواده XMega نیز توسط شرکت سازنده، به صنعت معرفی شده است. در این کتاب برای پیاده‌سازی بسیاری از پروژه‌ها از میکروکنترلر ATmega16 استفاده شده است و همانطور که از نامش مشخص است از خانواده Mega AVR می‌باشد. در ادامه به خلاصه‌ای از خصوصیات و پیکربندی این میکروکنترلر می‌پردازیم.

۱-۱-۳- خصوصیات ATmega16 و ATmega16L

- با استفاده از معماری RISC طراحی و ساخته شده‌اند.
  - کارایی بالا و توان مصرفی کم.
  - دارای ۱۳۱ دستورالعمل، با کارایی بالا که بیشتر دستورات، تنها در یک سیکل ساعت اجرا می‌شوند.
  - دارای ۳۲ رجیستر همه‌منظوره هشت بیتی.
  - دارای بیشینه سرعت تا 16 MIPS در فرکانس 16 MHz.
- حافظه برنامه و داده غیرفرار
  - ١٦ کیلوبایت حافظه FLASH داخلی قابل برنامه‌ریزی.
    پایداری حافظه FLASH با قابلیت ۱۰۰۰۰ بار نوشتن و پاک کردن.
  - ١٠٢٤ بایت حافظه داخلی SRAM .
  - ۵۱۲ بایت حافظه EEPROM داخلی قابل برنامه‌ریزی.
    پایداری حافظه EEPROM با قابلیت ۱۰۰۰۰۰ بار نوشتن و پاک کردن.
  - قفل برنامه FLASH و EEPROM .
- خصوصیات جانبی
  - دو تایمر/کانتر ۸ بیتی با مقسم مجزا.

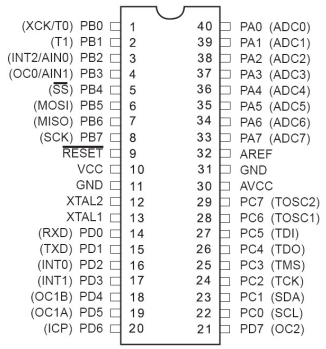
[1] Arithmetic Logic Unit (ALU).

- یک تایمر/کانتر ۱٦ بیتی با مقسم مجزا و دارای حالت‌های تسخیر و مقایسه.
- ٤ کانال مدولاسیون عرض پالس (PWM).
- ۸ کانال مبدل آنالوگ به دیجیتال ۱۰ بیتی.
- یک مقایسه کننده آنالوگ داخلی.
- Watchdog قابل برنامه‌ریزی با اسیلاتور داخلی.
- قابلیت ارتباط با پروتکل «سریال دو سیمه»[1] و I^2C.
- قابلیت ارتباط سریال SPI[2] به صورت Master یا Slave.
- USART سریال قابل برنامه‌ریزی.

- خصوصیات ویژه میکروکنترلر
  - دارای اسیلاتور RC داخلی کالیبره شده.
  - منابع وقفه[3] داخلی و خارجی.
  - توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS .
  - ولتاژ کاری 2.7 تا 5.5 ولت برای Atmega16L و 4.5 تا 5.5 ولت برای Atmega16 .
  - فرکانس‌های کاری 0 تا 8MHz برای Atmega16L و 0 تا 16MHz برای Atmega16 .
  - ۳۲ خط ورودی خروجی قابل برنامه ریزی.

شکل (۱-۲) شمای پایه‌های میکروکنترلر Atmega16 را نشان می‌دهد. همانطور که در شکل (۱-۲) مشاهده می‌شود ATmega16 دارای چهار پورت A، B، C و D می باشد که افزون بر این‌که به عنوان ورودی خروجی مورد استفاده قرار می‌گیرند، کاربردهای جانبی دیگری نیز دارند. که در فصل‌های بعد به آن‌ها خواهیم پرداخت.

[1] TWO-WIRE
[2] Serial Peripheral Interface (SPI).
[3] Interrupt

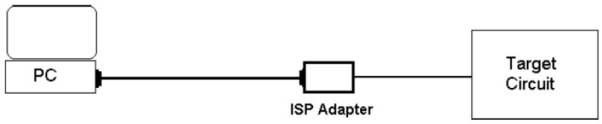


شکل (۱-۲): شمای پایه‌های میکروکنترلر ATmega16

۱-۲- برنامه‌ریزی میکروکنترلرهای AVR

به طور کلی، چهار روش برای برنامه‌ریزی میکروکنترلرهای AVR وجود دارد که به ترتیب: برنامه‌ریزی موازی، برنامه‌ریزی توسط پروتکل JTAG، خود برنامه‌ریز[1] و ISP[2] می‌باشند.

برای برنامه‌ریزی میکروکنترلر AVR، از روش ISP استفاده می‌کنیم. در این روش: بدون برداشتن تراشه میکروکنترلر AVR از روی برد مربوطه، اقدام به پروگرم نمودن آن می‌نمائیم. انتقال کدهای برنامه از کامپیوتر به میکروکنترولر، مطابق شکل (۱-۳) به روش سریال می‌باشد. در روش ISP، با استفاده از ارتباط SPI (که در فصل‌های بعد توضیح داده می‌شود) می‌توانیم علاوه بر برنامه‌ریزی میکروکنترولر (Programming)، صحت برنامه‌ریزی را نیز تشخیص دهیم (Verify). در این روش با استفاده از سه خط سیگنال، برنامه‌ریزی میکروکنترلر انجام می‌شود (یک خط ورودی، یک خط خروجی و یک خط پالس ساعت (Clock). البته: پایه Reset میکروکنترلر می‌بایست در زمان برنامه‌ریزی پایین (صفر منطقی) نگه داشته شود، یعنی به صفر ولت متصل شود).

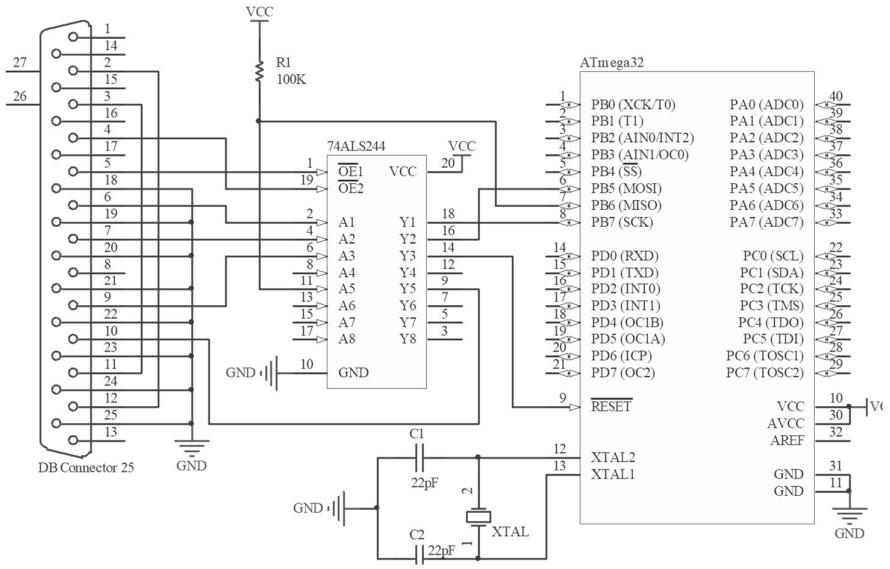


شکل (۱-۳): ساختار برنامه‌ریزی میکروکنترولر.

[1] Self-Programming
[2] In System Programming

ساخت پروگرمر ISP با استفاده از کانکتور DB25 بسیار ساده است. اتصال مستقیم میکروکنترلر به کامپیوتر، ممکن است باعث ایجاد مشکلات ناشی از جریان کشی توسط میکروکنترلر و کامپیوتر شود که در نهایت منجر به سوختن میکروکنترلر می‌شود. بنابراین، با استفاده از یک تراشه راه‌انداز[1] به منظور تامین جریان لازم در هنگام برنامه‌ریزی میکروکنترلر، مشکل مذکور برطرف خواهد شد. همچنین به این دلیل که پورت[2] موازی در مقابل اتصال کوتاه و یا اضافه بار چندان مقاوم نیست و ممکن است با قطع و وصل تغذیه‌ی بورد، آسیب ببیند. بنابراین، استفاده از یک بافر جریان ضروری است. در این روش، نمی‌توان به طور قاطع از برنامه‌ریزی کاملاً مطمئن میکرو صحبت کرد.

مدار واسط STK200/300 به منظور برنامه‌ریزی میکروکنترولر به روش ISP استفاده شده است که مطابق شکل (۱-٤) می‌باشد. تراشه 74ALS244 به عنوان بافر جریان، جهت حفاظت از پورت‌های میکروکنترلر و کامپیوتر استفاده شده است. شما می‌توانید مطابق نقشه شکل (۱-٤) مدار پروگرمر را بسازید و یا آن را از فروشگاه‌های الکترونیکی تهیه نمائید. شایان ذکر است که صحت عملکرد صحیح مدار زیر تائید می‌شود.



[1] Driver
[2] Port

شکل (۱-٤): پروگرامر STK200/300

۱-۳- آشنایی با روش نصب و کار با نرم‌افزار CodeVisionAVR

۱-۳-۱- مقدمه

برای کار با میکروکنترلرهای AVR باید برنامه‌ای به یکی از زبآن‌های Assembly، C یا Basic در محیط نرم افزار مربوط به آن نوشت. سپس آن را کامپایل نمود. **کامپایل نمودن برنامه:** عملی است که در آن، برنامه از زبان نوشتاری به زبان صفر و یک که توسط میکروکنترلر قابل فهم باشد، تبدیل می‌شود. در صورتی که برنامه هیچ خطایی، شامل: خطای املایی، ساختاری و نظایر آن را نداشته باشد به درستی کامپایل شده و یک فایل به زبان صفر و یک (زبان ماشین) توسط کامپایلر تولید می‌شود. پسوند فایل‌هایی که حاوی برنامه به زبان ماشین هستند، HEX می‌باشد. اکنون برای انتقال فایل HEX ایجاد شده به درون آی‌سی، نیازمند یک دستگاه جانبی یا واسط سخت افزاری هستیم که کامپیوتر را به تراشه میکروکنترلر متصل کند و فایل HEX مربوطه را از کامپیوتر بر روی میکروکنترلر بارگذاری نماید. این واسط سخت افزاری، اصطلاحاً پروگرمر نامیده می‌شود. پس از برنامه‌ریزی کردن (پروگرم کردن)، میکروکنترلر را از پروگرمر جدا کرده و در مدار مورد نظر قرار داده (و یا اگر پروگرمر ساخته شده مطابق شکل (۱-٤) باشد بدون جدا نمون میکروکنترلر از مدار به برنامه‌ریزی آن اقدام می‌کنیم). پس از آن، عملکرد سخت افزاری آن‌را بررسی می‌کنیم.

در این قسمت، نرم افزار CodeVisionAVR که یکی از کامپایلرهای قوی برای برنامه‌نویسی به زبان C می‌باشد، معرفی می‌شود. افزون بر این، روش نصب و قسمت‌های مختلف آن نیز آموزش داده می‌شود.

۱-۳-۲- آشنایی با محیط CodeVisionAVR

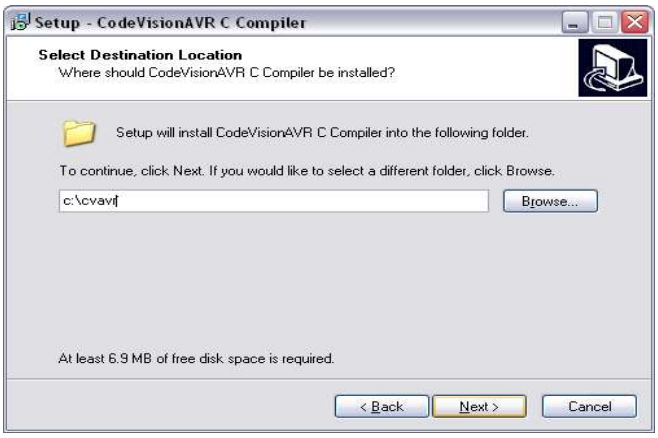
نرم افزار CodeVisionAVR دارای محیطی برای برنامه‌نویسی به زبان C است. که در این محیط، کاربر با تسلط نسبی بر زبان C قادر به نوشتن برنامه‌های بسیار کاربردی می‌شود. یکی از دلایل انتخاب این نرم‌افزار، قابلیت Wizard یا محیط راهنمای گام به گام است. محیط راهنمای گام به گام که به آن به اختصار ویزارد گفته می‌شود. این قابلیت در مقدار دهی اولیه رجیسترهای مختلف میکروکنترلر، همچنین فراخوانی برخی کتابخانه‌های موجود، کمک بسیار زیادی به کاربران می‌کند. بنابراین، به کاربران مبتدی، برنامه‌ریزی میکروکنترلرهای AVR با استفاده از این محیط توصیه می‌شود. این نرم‌افزار دارای یک کامپایلر بوده که توسط آن کدهای برنامه با پسوند Hex جهت

برنامه‌ریزی میکروکنترلر تولید می‌شوند. توسط این نرم افزار و یک پروگرمر از نوع ISP می‌توان کلیه‌ی میکروکنترلرهای AVR را برنامه‌ریزی نمود. نرم‌افزار CodevisionAVR علاوه بر حمایت از کتابخانه‌های استاندارد زبان C، دارای کتابخانه‌های دقیقی برای کار با LCD های کارکتری، تولید وقفه، تنظیمات توان مصرفی و نظایر آن می‌باشد.

۱-۳-۳- روش نصب نرم‌افزار CodevisionAVR

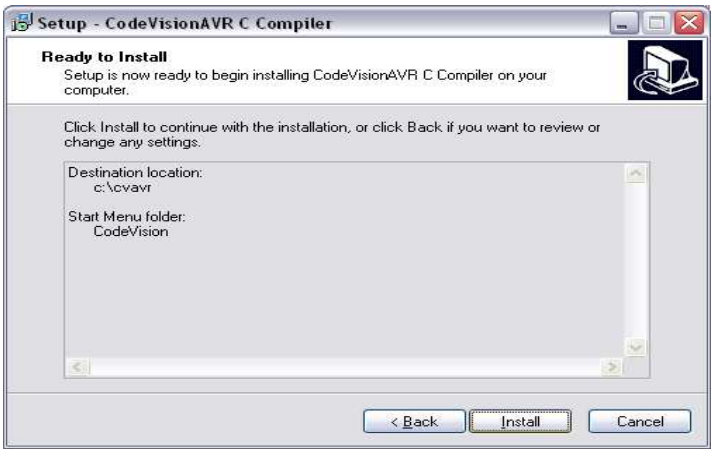
ابتدا با مراجعه به CD (که همراه با کتاب عرضه می‌شود) فایل مربوط به نرم افزار CodeVisionAVR را باز کنید و با اجرای فایل setup.exe مراحل نصب را مطابق شکل‌های زیر تا پایان ادامه دهید.

۱- بر روی گزینه Next کلیک نمائید.



۲- با انتخاب مسیر مناسب جهت نصب نرم‌افزار بر روی گزینه Next کلیک نمائید.

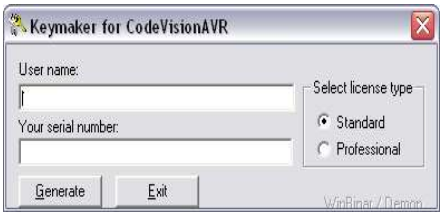
۳- بر روی گزینه Next کلیک نمائید.



٤- بر روی گزینه Install کلیک نمائید. در صورتی که نرم‌افزار CodeVision نسخه V2.03.4 را نصب می‌کنید، این نسخه دارای قفل نرم‌افزاری نبوده و در این مرحله عملیات نصب پایان می‌پذیرد. در غیر این صورت مراحل زیر را دنبال کنید.

٥- پس از نصب برنامه، برای اجرای صحیح آن ضروری است که قفل برنامه باز شود. برای این کار، ابتدا برنامه را اجرا نمائید. از پنجره ظاهر شده شماره سریال را یادداشت نمائید و در مرحله بعد وارد نمائید.

٦- فایل License Generator.exe را از مسیر دایرکتوری اصلی برنامه، اجرا نمائید. اکنون با وارد کردن یک نام دلخواه در قسمت User name، و در قسمت Serial number، شماره سریال مرحله‌ی ٥ را وارد کرده و دکمه‌ی Generate را فشار دهید و فایل تولید شده با پسوند dat. را در مسیر دلخواه ذخیره نمائید.



۷- اکنون با استفاده از پنجره ظاهر شده در مرحله‌ی ۵، بر روی Import کلیک کرده و آدرس فایل تولید شده در مرحله‌ی ٦ را وارد کنید. اکنون، قفل برنامه باز شده و برنامه قابل اجرا و استفاده است.

۱-۳-٤- ایجاد یک پروژه جدید

برای آشنایی با محیط CodeVisionAVR با ایجاد یک پروژه شروع می‌کنیم و مراحل زیر را مطابق شکل‌ها ادامه می‌دهیم:

۱- برنامه CodeVisionAVR را اجرا می کنیم.

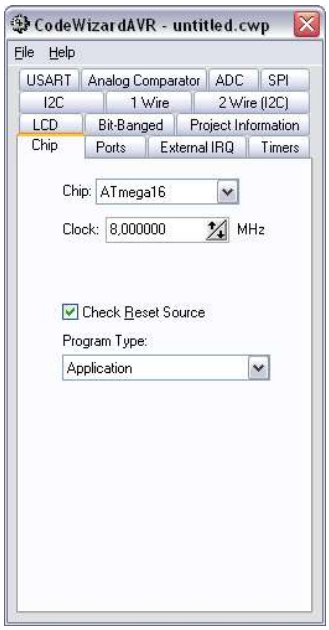
۲- در صفحه‌ی باز شده (صفحه اصلی) پس از انتخاب گزینه File، گزینه New را انتخاب کرده تا پنجره‌ای مطابق شکل (۱-۵)، ظاهر شود.

شکل (۱-۵): ایجاد پروژه جدید

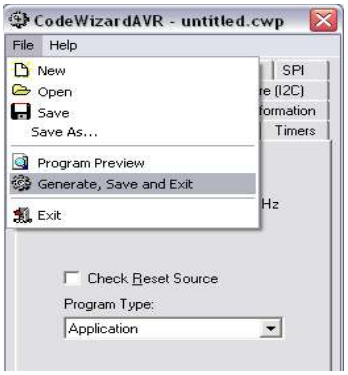
۳- از پنجره ظاهر شده در شکل (۱-۵) گزینه Project را که مربوط به انتخاب پروژه است انتخاب نموده و بر روی OK کلیک نمائید. اکنون، پنجره‌ی دیگری باز می‌شود و از شما سوال می‌شود که: آیا می‌خواهید پروژه جدید را توسط محیط ویزارد ایجاد کنید؟ با فشردن کلید Yes این عمل انتخاب می‌شود و با فشردن کلید No جهت ایجاد پروژه از ویزارد استفاده نمی‌شود.

شکل (۱-٦): انتخاب CodeWizardAVR

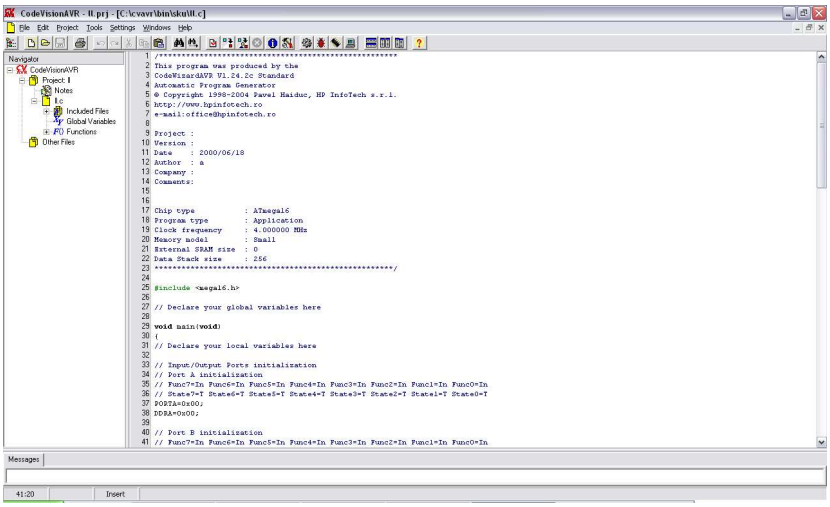
۳-۱- با کلیک بر روی گزینه Yes، مطابق شکل (۱-٦) محیط ویزارد مطابق شکل (۱-۷) باز می‌شود و توسط این محیط می‌توان با استفاده از سربرگ‌های موجود، با انتخاب میکروکنترلر مورد نظر، رجیسترهای آن را به‌صورت اولیه مقداردهی نمود. پس از اتمام مقدار دهی اولیه، با انتخاب گزینه File از منوی مربوطه، مطابق شکل (۱-۸)، بر روی گزینه Generate, Save and Exit کلیک کرده و با انتخاب یک نام برای آن، پروژه مورد نظر را ایجاد نمائید تا شکل (۱-۹) ظاهر شود. اکنون برنامه شما به صورت اولیه مقدار دهی شده است.



شکل (۱-۷): محیط CodeWizardAVR برای مقداردهی اولیه میکروکنترلرهای AVR.

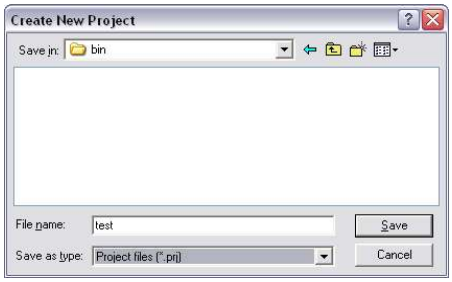


شکل (۱-۸): ذخیره‌سازی و تولید کد.



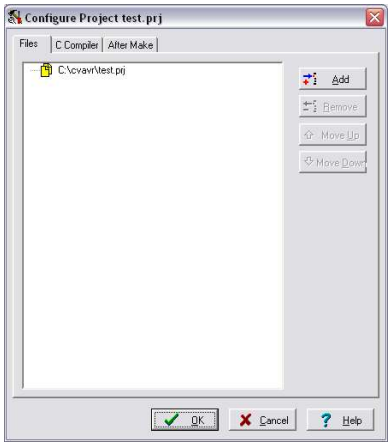
شکل (۱-۹): کدهای ایجاد شده توسط ویزارد در محیط CodevisionAVR

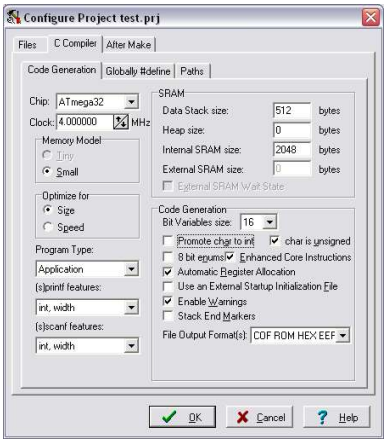
۳-۲- با کلیک بر روی گزینه No در شکل (۱-٦) پنجره زیر جهت ذخیره پروژه ظاهر می‌شود. پس از انتخاب یک نام برای پروژه مورد نظر، بر روی گزینه Save کلیک نمائید.



شکل (۱-۱۰): انتخاب یک نام جهت ذخیره پروژه

۳-۲-۱- پس از ذخیره پروژه، شکل (۱-۱۱) ظاهر می‌شود که دارای سه سربرگ File، C Compiler و After Make می‌باشد. با انتخاب سربرگ C Compiler میکروکنترلر و فرکانس کریستال آن را انتخاب نمائید. به عنوان مثال، در شکل (۱-۱۲) میکروکنترلر ATmega32 و کریستال ٤ مگاهرتز انتخاب شده است (سایر سربرگ‌ها در قسمت های مربوطه به تفصیل توضیح داده می‌شوند). حال، بر روی گزینه OK کلیک نمائید تا پروژه ایجاد شود.





شکل (۱-۱۱): پیکربندی پروژه **شکل (۱-۱۲): پیکربندی میکروکنترلر**

۳-۲-۲- در صفحه‌ی باز شده (صفحه اصلی) پس از انتخاب گزینه File، گزینه New را انتخاب کرده تا پنجره‌ای ظاهر شود. از پنجره ظاهر شده، گزینه Source را انتخاب و سپس فایل مورد نظر را با نام دلخواه ذخیره نمائید. دقت شود که پسوند این فایل c. است. یعنی محیطی جهت برنامه‌نویسی به زبان C ایجاد شده است. اکنون، از منوی بالای صفحه، بر روی گزینه Project و سپس بر روی Configure کلیک نمائید تا مجدداً شکل (۱-۱۱) ظاهر شود. سپس از سربرگ Files بر روی گزینه Add کلیک نمائید و فایل ایجاد شده با پسوند c. را به پروژه اضافه نمائید و بر روی گزینه OK کلیک نمائید. اکنون پروژه مورد نظر، ایجاد شده است.

۱-٤- آشنایی با زبان C

۱-٤-۱- ساختار برنامه‌نویسی

به منظور برنامه‌نویسی به زبان C، آشنایی اولیه با دستورات آن الزامی است. برای این منظور: با معرفی مختصر ساختارها و دستورات مرتبط، مطابق زیر با یک برنامه ساده شروع می‌کنیم.

```
#include <stdio.h>
 int main() {
            printf("Hello World\n");
```

```
    return 0;
}
```

شرح برنامه:

#include <stdio.h> : فایلی به اسم stdio.h را ضمیمه می‌کند که این فایل به ما اجازه استفاده از توابع خاصی را می‌دهد. stdio کوتاه شده‌ی عبارت Standard Input/Output است. این فایل شامل **توابع ورودی:** مانند خواندن از صفحه کلید و **توابع خروجی:** مانند نمایش دادن بر روی صفحه نمایش است.

int main() :

int : عبارتی است که یک مقدار را بر می‌گرداند (return) و در ادامه بیشتر به توضیح آن می‌پردازیم.

main : نام نقطه‌ای است که برنامه از آن نقطه شروع می‌شود. پرانتزها در جلوی عبارت main به این معنی است که این تابع آرگومان ورودی ندارد.

```
int main() {
        دستورها
    }
```

{ } (آکلادها) برای این است که تمام نوشته‌ها را در یک گروه خاص قرار دهد. در مثال بالا، آکلاد مشخص می‌کند که نوشته‌ها متعلق به تابع main است. شایان ذکر است که آکلادها در زبان C کاربرد زیادی دارند.

printf ("Hello World \n");

تابع printf : یک متن را در صفحه نمایش می‌دهد. اطلاعاتی که باید این تابع نمایش دهد بین دو پرانتز قرار می‌گیرد. دقت شود که: کلمه‌ها بین دو double contention (") قرار گیرند، زیرا آن‌ها در واقع یک رشته هستند. هر یک از حروف یک کاراکتر می‌باشد و مجموعه‌ی آن‌ها یک گروه با نام رشته (string) را تشکیل می‌دهند. رشته ها همیشه باید بین دو " قرار می‌گیرند.

\n : به کامپایلر دستور می‌دهد به خط جدید برود، وقتی شما در متن خود enter بزنید به خط جدید نمی‌رود برای همین ما مجبور هستیم از این دستور استفاده کنیم.

باید بعد از هر دستوری یک semicolon (;) قرار دهید برای این که نشان دهد آن دستور تمام شده است.

جدول (۱-۱): دستورات برای printf

\a	Audible signal

\b	Backspace
\t	رفتن به یک tab جلوتر
\n	رفتن به یک خط جدید
\v	Vertical tab
\f	پاک کردن صفحه / رفتن به صفحه جدید
\r	Carriage return

return 0 : مقدار آرگومان برگشتی توسط دستور return مشخص می‌شود و return 0 به معنی آن است که تابع ما مقدار صفر را باز می‌گرداند.

پس از کامپایل نمودن برنامه، اگر کدهای شما اشتباه باشند، کامپایلر به شما می‌گوید که اشتباه در کدام خط رخ داده است. به منظور اجرای صحیح برنامه، اصلاح خطا شامل: خطای املایی و ساختاری الزامی است. پس از کامپایل صحیح، برنامه‌ی شما تبدیل به فایل اجرایی می‌شود. اکنون باید عبارت "Hello World" را در صفحه ملاحظه کنید.

استفاده از توضیحات (comments) : برای تشخیص بهتر هر یک از خطوط برنامه، استفاده از comment در جلوی خطوط پیشنهاد می‌شود. توضیحات را باید بعد از // یا بین /*.....*/ بنویسید. توضیحات توسط کامپایلر خوانده نمی‌شوند. توضیحات استفاده شده در اول برنامه عملکرد برنامه را نشان می‌دهند. شما همچنین می‌توانید: بین قسمت‌های مختلف برنامه از توضیحات استفاده کنید تا آن قسمت را توضیح دهید در ذیل، مثالی از قرار دادن توضیح قرار داده شده است:

```
/* Author: Hamed Saghaei
 Date: 2009/07/15
 Description:
 Writes the words "Hello World" on the screen */

#include <stdio.h>

int main()
 {
    printf("Hello World\n");      //prints "Hello World"
    return 0;
}
```

۱-٤-۲- متغیرها

متغیرها در زبان برنامه‌نویسی C، مکانی از حافظه هستند که نامی به آن‌ها تخصیص داده می‌شود و می‌توانند مقداری را در بر داشته باشند. از متغیرها برای ذخیره کردن مقادیر در حافظه استفاده می‌شود. دو نوع متغیر اصلی در زبان C وجود دارد که به ترتیب (عدد) Numeric و(حروف) Character می‌باشند.

- **متغیرهای عددی:** این نوع متغیرها می‌توانند عدد صحیح باشند و اعداد کسری یا اعشاری نمی‌توانند در این گروه قرار بگیرند.

- **متغیرهای کاراکتری:** در این نوع متغیرها: حروف و اعداد می‌توانند قرار بگیرند. البته اعداد در اینجا کمی متفاوت هستند.

- **عبارات عددی و رشته‌ای ثابت:** این نوع عبارات ثابت هستند و مقدار آن‌ها قابل تغییر نیست. گاهی برای جلوگیری از اشتباه از آن‌ها استفاده می‌شود.

تعریف متغیرها: برای تعریف متغیر در برنامه ابتدا نوع آن را باید مشخص کنیم در جدول (۱-۲) نوع متغیرها و محدوده‌ی آن‌ها مشخص شده است. البته متغیرها را به هر نحوی که مایل باشید می‌توانید نام گذاری کنید. ولی بهتر است کمتر از ۳۲ حرف داشته باشند و در ابتدای برنامه تعریف شوند.

جدول (۱-۲): انواع داده

نام	نوع	محدوده
bit	یک بیت	از ۰ و ۱
char	هشت بیت	از ۱۲۸- تا ۱۲۷
unsigned char	هشت بیت بدون علامت	از ۰ تا ۲۵۵
int	اعداد صحیح ۱٦ بیتی	از ۳۲۷٦۷- تا ۳۲۷٦۸
unsigned int	اعداد صحیح بدون علامت ۱٦ بیتی	از ۰ تا ٦۵۵۳۵
long int	اعداد صحیح ۳۲ بیتی	از ۲۱۷٤۸۳٦٤۸- تا ۲۱۷٤۸۳٦٤۷
unsigned long int	اعداد صحیح بدون علامت ۳۲ بیتی	از ۰ تا ٤۲۹٤٦۷۲۹۵
float	اعداد اعشاری	از مقادیر مثبت و منفی ۳۸-e× ۱,۱۷۵ تا مقادیر مثبت و منفی ۳,٤۰۲×e۳۸

double	اعداد اعشاری	از مقادیر مثبت و منفی ۲,۲×e-۳۰۸ تا مقادیر مثبت و منفی ۱,۸×e۳۰۸

```
int main()
{
    int a;
    char b;
    return 0;
}
```

در عبارت بالا: متغیر a از نوع int و متغیر b از نوع char تعریف شده‌اند. و مطابق زیر برنامه ذیل می‌توان در یک مکان، چندین متغیر را تعریف کرد.

```
int main()
{
    int a, b, c;
    return 0;
}
```

با توجه به این‌که تابع main، تابع اصلی برنامه است. بنابراین بازگشت مقادیر توسط این تابع صحیح نمی‌باشد. در ادامه به جای تعریف آن به روش مثال‌های قبل، به صورت زیر تعریف می‌شود. همچنین برای تعریف متغیرهای Constant (ثابت) تنها لازم است: عبارت const را قبل از نوع متغیر قرار داد.

```
void main()
{
    const float pi = 3.1415;
}
```

- متغیرهای علامت‌دار و بدون علامت:

متغیرهای علامت‌دار (signed) می‌توانند دارای مقادیر مثبت یا منفی باشند ولی متغیرهای بدون علامت (unsigned)، تنها می‌توانند مقادیر مثبت و صفر را اختیار کنند. این عبارات، قبل از نوع متغیر تعریف می‌شوند و نبود آن دلیل بر علامت‌دار بودن (Signed) متغیر است.

```
void main()
{
    unsigned int a;
```

```
    signed int b;
}
```

- **استفاده از متغیر در محاسبات:**

برای دادن مقداری به متغیر از علامت تساوی (=) استفاده می‌شود.

```
void main()
{
    int a=4;     //  a=4
    char b;      //  b=0
    a = 3;       //  a=3
    b = 'H';     //  b='H'
}
```

به منظور انجام محاسبات از عملگرهای محاسباتی زیر استفاده می‌شود.

جدول (۱-۳): عملگرهای محاسباتی

عملیات	عمگر
جمع	+
تفریق	–
ضرب	*
تقسیم	/
باقیمانده	%

برای انجام عملیات به متغیری نیاز داریم که حاصل عملیات در آن ذخیره شود.

```
void main()
{
    int a, b;
    a = 5;
    b = a + 3;    // b=8
    a = a % 3;    // a=2
}
```

در زبان برنامه‌نویسی C، می‌توان متغیری از keyboard را با استفاده از دستور scanf گرفت و توسط printf آن را چاپ کرد.

```
#include <stdio.h>
void main()
{
```

```
    int a;
    scanf("%d", &a);
    a = a * 2;
    printf("The answer is %d", a);
}
```

٪d برای خواندن و چاپ کردن متغیرها از نوع int استفاده می‌شود و سایر متغیرها مطابق جدول زیر هستند.

جدول (۱-٤): خواندن و چاپ متغیرها

٪d or %i	int
٪c	char
٪f	float
٪lf	double
٪s	string

۱-٤-۳- دستور If

در بیشتر مواقع لازم است که مقدار متغیری در برنامه کنترل شود. برای این منظور استفاده از حلقه If به حل مساله کمک می‌کند.

- ساختار کلی دستور If

```
if (شرط)
  {
    دستورات;
  }
else if (شرط)
  {
    دستورات;
  }
else
  {
    دستورات;
  }
```

```
#include <stdio.h>
 void main()
```

```
{
    int mark;
    char pass;
    scanf("%d", &mark);
    if (mark > 40)
        pass = 'y';
}
```

در برنامه بالا ابتدا متغیری از ورودی گرفته می‌شود و سپس شرط mark > ٤٠ امتحان می‌شود. اگر پاسخ مثبت بود، حرف Y در متغیر pass قرار داده می‌شود.

```
#include <stdio.h>
 void main()
{
    int mark;
    char pass;
    scanf("%d", &mark);
    if (mark > 40)
        pass = 'y';
    else
        pass = 'n';
}
```

در برنامه بالا ابتدا متغیری از ورودی گرفته می‌شود و سپس شرط mark > ٤٠ امتحان می‌شود. اگر پاسخ مثبت بود، حرف Y در متغیر pass قرار می‌گیرد، در غیر این صورت حرف n در pass قرار داده می‌شود.

در صورتی که در برنامه، اجرای چند دستور پس از امتحان شرط نیاز داشت، می‌باید مطابق زیر برنامه ذیل، دستورات را درون { } قرار دهیم.

```
#include <stdio.h>
void main()
{
    int mark;
    char pass;
    scanf("%d", &mark);
    if (mark > 40)
      {
        pass = 'y';
        printf("You passed");
     }
```

```
  else
    {
        pass = 'n';
        printf("You failed");
    }
}
```

همچنین می‌توان در هر دستور if چند شرط را امتحان نمود:

```
#include<stdio.h>
 void main()
  {
    int a, b;
    scanf("%d", &a);
    scanf("%d", &b);
    if (a > 0 && b > 0)
        printf("Both numbers are positive\n");
    if (a = 0 || b = 0)
        {
         printf("At least one of the numbers = 0\n");
         a++;
        }
    if (!(a > 0) && !(b > 0))
        printf("Both numbers are negative\n");
}
```

- عملگرهای منطقی:

جدول (۱-۵): عملگرهای منطقی

مساوی	==
نا مساوی	=!
بزرگتر از	<
بزرگتر یا مساوی	=<
کوچکتر از	>
کوچکتر یا مساوی	=>
و	&&
یا	||
نقیض	!

و: هر دو شرط درست باشند حاصل درست.
یا: یکی یا هر دو شرط درست باشند حاصل درست است.

۱-٤-٤- ساختار دستور switch

```
switch(variable)
{
    case var 1:
    دستور;
    break;
    case var 2:
    دستور;
    break;
    .
    .
    .
    default:
    دستور;
}
```

```
#include <stdio.h>
 void main()
{
    char fruit;
    printf("Which one is your favourite fruit:\n");
    printf("a) Apples\n");
    printf("b) Bananas\n");
    printf("c) Cherries\n");
    scanf("%c", &fruit);
    switch (fruit) {
    case 'a':
        printf("You like apples\n");
        break;
   case 'b':
        printf("You like bananas\n");
        break;
    case 'c':
        printf("You like cherries\n");
        break;
```

```
    default:
        printf("You entered an invalid choice\n");
    }
}
```

در برنامه بالا: fruit متغیر برنامه است که آن را با استفاده از case کنترل می‌کنیم اگر برابر a بود، متن You like apples در خروجی چاپ می‌شود سپس از دستور beak استفاده شده است که باعث خروج از حلقه می‌شود در نهایت اگر با هیچ‌یک از متغیرها برابر نبود با استفاده از default متن You entered an invalid choice چاپ می‌شود.

۱-٤-٥- دستورات حلقه

در برخی مواقع لازم است دستوری به صورت متوالی اجرا شود. برای این کار از دستورات حلقه استفاده می‌شود که به سه دسته تقسیم می‌شوند:

for, do while, while:

- ساختار for

```
for (عدد شروع ;شرط ;اضافه شدن متغیر)
{
دستورات ;
}
```

---

for اجازه می‌دهد از عددی تا عدد دیگر تکرار داشته باشیم. در برنامه زیر اعدد ۱ تا ٢٤ در خروجی چاپ می‌شوند.

```
void main()
{
    int i;
    for (i = 1; i == 24; i++)
        printf("H\n");
    return 0;
}
```

در مثال بالا از i++ استفاده کردیم که برابر است با i=i+1 یعنی در هر بار اجرا، یک واحد به i اضافه می‌شود یا می‌توان از i-- استفاده کرد که هر بار اجرا، یک واحد از i کم می‌شود.

- ساختار while

```
while(condition)
{
```

```
    دستورات ;
}
```

تفاوت این حلقه با قبلی در این است که ما نمی‌دانیم قرار است چند بار حلقه اجرا شود در مثال زیر تعداد اجرا شدن حلقه از ورودی گرفته می‌شود در times ذخیره و در هر بار اجرا شدن حلقه یک واحد به i افزوده می‌شود تا زمانی که به تعداد times برسد از حلقه خارج شده و برنامه تمام می‌شود.

```
include <stdio.h>
void main()
{
    int i, times;
    scanf("%d", &times);
    i = 0;
    while (i <= times)
    {
        i++;
        printf("%d\n", i);
    }
}
```

– ساختار do while

ساختار do while مانند while است با این تفاوت که شرط آن در آخر امتحان می شود.

```
do {
     دستورات ;
   }
while(Condition)
```

```
#include <stdio.h>
 void main()
{
    int i, times;
    scanf("%d", &times);
    i = 0;
    do
    {
        i++;
        printf("%d\n", i);
    }
while (i <= times);
```

:Break and continue

Continue برای شروع Loop از ابتدا و برای خاتمه دادن از Break استفاده می‌شود. در مثال زیر هرگز Hello چاپ نمی‌شود.

```
#include <stdio.h>
 int main()
{
    int i;
    while (i < 10)
    {
        i++;
        continue;
        printf("Hello\n");
    }
}
```

۱-٤-٦- اشاره‌گرها

اشاره‌گر[1] متغیرهایی هستند که درون خود آدرسی از حافظه را نگهداری می‌کنند و به متغیر درون آن آدرس، اشاره می‌کنند به این دلیل به آن‌ها اشاره‌گر می‌گویند. شما می توانید از اشاره‌گرها برای کارهای خاصی استفاده کنید. مثلاً برای گرفتن مقدار آدرسی که به آن اشاره می‌کند. اشاره‌گرها می‌توانند مشخص یا نامشخص باشند. اشاره‌گرهای مشخص به یک متغیر مشخص مثلاً int اشاره می‌کنند، و اشاره‌گرهای نامشخص می توانند به انواع اطلاعات اشاره کنند. برای این‌که شما عبارت یا کاراکتری را به عنوان اشاره‌گر مشخص کنید باید یک * قبل از اسم آن بگذارید، در اینجا مثالی از اشاره‌گر آورده شده است.

```
void main()
{
    int *p;
    void *up;
 }
```

شما می‌توانید آدرس یک int را درون یک اشاره‌گر قرار دهید و اشاره‌گر با استفاده از علامت & آدرس int را می‌گیرد.

```
void main()
{
    int i;
```

[1] Pointer

```
    int *p;
    i = 5;
    p = &i;
    return 0;
}
```

شما می‌توانید به مقدار int که اشاره‌گر به آن اشاره کرده دسترسی داشته باشید.

* باعث می‌شود که اشاره‌گر دوباره بازگشت داده شود، یعنی در واقع همان متغیر می شود و تغییر در آن به منزله‌ی تغییر در متغیر است.

```
void main()
{
    int i, j;
    int *p;
    i = 5;
    p = &i;
    j = *p;   //j = i
    *p = 7;  //i = 7
    return 0;
}
```

استفاده از اشاره‌گر برای بیان i=j در بالا راهی طولانی است. شما در ادامه با کاربردهای بیشتر اشاره‌گرها آشنا خواهید شد ولی در این قسمت، هدف آشنایی مقدماتی با این مبحث است.

۱-٤-۷- آرایه‌ها

اگر شما می‌خواستید ۵ متغیر داشته باشید، مانند زیر عمل می‌کردید:

```
int i1, i2, i3, i4, i5;
```

حال اگر ۱۰۰۰ متغیر بود چه می‌کردید؟ مسلماً فرایند بالا زمان زیادی می‌برد، ولی با استفاده از آرایه می‌توانید: با نام یک متغیر، هر تعداد که بخواهید متغیر بسازید. آرایه مانند متغیر معمولی است و برای تعریف آن کافی است فقط بعد از نام آن یک جفت براکت بیاورید و درون آن تعداد متغیر مورد نظر را بنویسید. مثال:

```
int a[5]
```

برای این‌که به مقدار متغیرهای هر خانه دست یابید شما باید نام آرایه و سپس شماره ی خانه‌ی مورد نظر را بیاورید. فقط به یاد داشته باشید که شماره‌ی خانه‌های آرایه از صفر شروع می‌شود. برای مثال یک آرایه به طول ۵ دارای خانه‌هایی از شماره‌ی ۰ تا ٤ است.

```
int a[5];
a[0] = 12; a[1] = 23;
a[2] = 34; a[3] = 45;
```

```
a[4] = 56;
printf("%d",a[0]);
```

– استفاده از آرایه با حلقه

استفاده از آرایه در حلقه کاربرد زیادی دارد. زیرا خانه‌های آرایه یک دنباله را طی می کنند که این فرایند مشابه حلقه‌ها است. برای مثال: وقتی خانه‌های آرایه صفر نشده‌اند و شما نیاز دارید که مقدار آن ها را صفر کنید باید مانند زیر از حلقه استفاده کنید:

```
int a[10];
for (i = 0;i < 10;i++)  a[i] = 0;
```

– آرایه‌های چند بعدی

آرایه‌هایی که ما قبلاً استفاده کردیم آرایه‌های یک بعدی نام دارند زیرا تنها از یک سطر تشکیل شده‌اند. آرایه‌های ۲ بعدی از چند سطر و ستون تشکیل شده‌اند، در برای تسلط بیشتر بر موضوع شکل (۱-۱۳) را ملاحظه نمائید.

آرایه‌های یک بعدی

مقدار	شماره سطر
٤	۰
۳	۱
-۱	۲

آرایه ی دو بعدی

	۰	۱	۲
۰	۱	۲	۳
۱	٤	٥	٦
۲	۷	۸	۹

شکل (۱-۱۳): آرایه‌های چند بعدی

شما می‌توانید از آرایه‌های سه بعدی یا بیشتر نیز استفاده کنید ولی اغلب کاربردی ندارند. در ادامه برنامه‌ای آورده شده که شما را با روش تعریف یک آرایه‌ی دو بعدی و چگونه کار کردن آن آشنا می‌کند. توجه داشته باشید که مثال دارای ۲ حلقه است زیرا می‌خواهیم مقدار متغیرهای درون سطر و ستون های مختلف را تغییر دهیم.

```
int a[3][3], i, j;
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            a[i][j] = 0;
```

۱-٤-۸- رشته

رشته آرایه‌ای از کاراکترهاست که به یک ۰ یا یک کاراکتر خالی (null) ختم شود تا نشان دهد که رشته کجا تمام شده است. توجه کنید که کاراکتر null جزو رشته به حساب نمی‌آید. برای استفاده از رشته دو راه وجود دارد:

۱- **استفاده از آرایه‌ای از کاراکترها**

۲- **استفاده از اشاره‌گر رشته**

آرایه کاراکتری مانند آرایه زیر تعریف می شود:

```
char ca[10];
```

شما باید مقدار هر خانه‌ی آرایه را به کاراکتر مورد نظر خود اختصاص، و کاراکتر پایانی خود را ۰ قرار دهید. به یاد داشته باشید که برای اشاره کردن به یک رشته باید از s٪ استفاده کنید.

```
 char ca[10];
   ca[0] = 'H'; ca[1] = 'e';
   ca[2] = 'l';  ca[3] = 'l';
   ca[4] = 'o'; ca[5] = 0;
 printf("%s", ca);
```

وقتی که شما یک مقدار خاص را به یک اشاره‌گر رشته نسبت می‌دهید کامپایلر یک صفر در انتهای آن قرار می‌دهد و دیگر نیازی نیست تا مانند آرایه کاراکتری، ۰ را در انتهای آن قرار دهید.

```
char ca[10], *sp;
scanf("%s", ca);
  sp = ca;
scanf("%s", sp);
```

شما می‌توانید یک رشته را در یک آرایه کاراکتری با استفاده از scanf بخوانید ولی برای خواندن یک اشاره‌گر رشته، کامپایلر باید آن را به یک آرایه کاراکتری برگرداند. فایل سرآمد string.h دارای توابع پرکاربردی برای کار با رشته‌هاست. در اینجا به شرح تعدادی از آن‌ها می‌پردازیم:

strcpy (منبع ، مقصد)

شما نمی‌توانید در زبان C از چنین دستوری استفاده کنید: string1 = string2. بلکه شما باید از تابع strcpy برای کپی کردن یک رشته درون یک رشته‌ی دیگر استفاده کنید. تابع strcpy رشته‌ی منبع را در رشته‌ی مقصد کپی می‌کند.

```
s1 = "abc";
s2 = "xyz";
```

```
strcpy(s1, s2); // s1 = "xyz"
```

strcat (منبع، مقصد) :

رشته‌ی منبع و مقصد را با هم ترکیب نموده و در رشته‌ی مقصد قرار می‌دهد.

```
s1 = "abc";
s2 = "xyz";
strcat(s1, s2); // s1 = "abcxyz"
```

strcmp(دوم، اول) :

این تابع رشته‌ی اول را با رشته‌ی دوم مقایسه می‌کند، اگر رشته‌ی اول بزرگتر از رشته ی دوم بود، تابع عددی مثبت را برگشت می‌دهد، اگر هر دو رشته مساوی بودند تابع مقدار عددی صفر را برگشت می‌دهد و اگر رشته اول کوچکتر از رشته‌ی دوم بود تابع عددی منفی را برگشت می‌دهد.

```
s1 = "abc";
s2 = "abc";
i = strcmp(s1, s2); // i = 0
```

strlen(رشته)

این تابع تعداد کاراکترهای رشته را بازگشت می‌دهد.

```
s = "abcde";
i = strlen(s); // i = 5
```

۱-٤-۹- توابع

توابع[1] در واقع، زیر برنامه می‌باشند. شما قبلا از توابع استفاده کردید، منظور همان تابع main است. شما قبل از این‌که یک تابع را فراخوانی کنید باید آن را در ابتدای برنامه و قبل از main تعریف کنید. در مورد هر تابع می‌باید نوع return مشخص شود (یعنی نوع آرگومان برگشتی تابع مشخص شود مثلا از نوع int است یا از نوع char یا نظایر آن). در صورتی که نمی‌خواهید مقدار خاصی را return کنید از void استفاده کنید. بعد اسم منحصر به فرد تابع را بنویسید و پس از آن یک جفت پرانتز قرار دهید. در صورتی که تابع شما آرگومان ورودی دارد با مشخص کردن نوع و تعداد آرگومآن‌ها، آن‌ها را درون پرانتز جلوی اسم تابع قرار دهید و در صورتی که تابع شما آرگومان ورودی ندارد درون پرانتز عبارت void را نوشته یا آن را خالی باقی بگذارید. در آخر نیز دو عدد { } بگذارید و برنامه ی خود درون آن بنویسید. تابع تعریف شده در مثال زیر را بررسی نمائید:

```
#include <stdio.h>
 void Hello()
```

[1] Functions

```
{
    printf("Hello\n");
}

void main()
{
    Hello();
}
```

پارامتر های تابع، مقادیری هستند که ما به یک تابع می‌دهیم تا بتواند آن‌ها را محاسبه کند. شما باید در داخل پرانتزهای پارامتر، متغیرها را بیاورید تا مقدار پارامتر را که قابل قبول است در تابع قرار دهد. دراینجا ما Function Add که دو پارامتر را با هم جمع می کند آورده شده است:

```
#include<stdio.h>
int Add(int a, int b)
{
    return  a+ b;
}

void main()
{
    int answer;
    answer = Add(5, 7);
}
```

شما می‌توانید آدرس متغیر را در یک تابع قرار دهید، در اینجا کپی صورت نمی‌گیرد و شما نیاز به اشاره‌گر دارید.

```
#include<stdio.h>

int Add(int *a, int *b)
 {
    return *a + *b;
 }

int main()
{
    int answer;
    int num1 = 5;
    int num2 = 7;
    answer = Add(&num1, &num2);
    printf("%d\n", answer);
    return 0;
```

```
}
```

۱-٤-۱۰- متغیرهای سراسری و محلی

متغیرهای محلی[1] فقط در داخل یک تابع قابل استفاده هستند و بیرون از آن نمی‌شود از این نوع متغیرها استفاده کرد. اگر شما نیاز دارید که بتوانید از یک متغیر در تمامی قسمت‌های برنامه استفاده کنید، باید آن را سراسری[2] تعریف کنید.

```
#include<stdio.h>
 int a;      // Global variables
int b;      // Global variables

int Add()
{
    return a + b;
}

int main() {
    int answer;  // Local variable
    a = 5;
    b = 7;
    answer = Add();
    printf("%d\n", answer);
    return 0;
}
```

[1] Local
[2] Global

فصل دوم آشنایی با پورت‌ها

با توجه به این‌که این کتاب با عنوان پروژه‌های کاربردی با استفاده از میکروکنترلر AVR است، فرض بر آن است که خوانندگان کتاب از آشنایی مقدماتی با میکروکنترلرهای AVR برخوردار هستند. بر مبنای این فرض، جزئیات بیشتر در رابطه با معماری ساخت این تراشه‌ها و انواع خانواده‌های آن‌ها بحث نخواهند شد. در صورت تمایل به یادگیری بیشتر می‌توانید به مراجع آخر کتاب مراجعه نمائید.

در این فصل، کار با پورت‌های موازی میکروکنترلر AVR به صورت پروژه‌های کاربردی آموزش داده می‌شود. دقت شود: کلیه برنامه‌ها به زبان برنامه‌نویسی C و در محیط نرم‌افزار CodeVisionAVR نوشته و کامپایل می‌شوند. سپس توسط نرم افزار Proteus شبیه‌سازی شده و با اجرای آن‌ها، صحت برنامه‌های نوشته شده، آزمایش و تائید می‌شود.

پروژه اول: چراغ چشمک‌زن

برنامه‌ای بنویسید که یک LED متصل به یکی از پایه‌های میکروکنترلر را به صورت متوالی روشن و خاموش کند (بدون استفاده از محیط ویزارد برنامه نوشته شود)؟

فرضیات پروژه: میکروکنترلر مورد استفاده ATmega16 است و از کریستال داخلی ٤ مگاهرتز استفاده شده است. LED به پین صفرم از پورت B متصل شده است و LED به صورت متوالی یک ثانیه روشن و یک ثانیه خاموش می‌شود.

☑ **حل:** توضیحات لازم برای نوشتن برنامه: میکروکنترلر ATmega16 دارای چهار پورت است که به ترتیب آن‌ها را A، B، C و D می‌نامیم. هر یک از آن‌ها دارای هشت پایه (پین) هستند که به صورت صفر تا هفت شماره‌گذاری می‌شوند. به **عنوان مثال:** پین صفرم پورت B با PB.0 نشان داده می‌شود.

❋ **یادآوری:** رجیسترهای مورد استفاده برای تنظیم پورت‌ها مطابق زیر هستند: رجیسترهای پورت‌های ورودی خروجی (I/O):

۱- رجیستر DDRX : این رجیستر برای تنظیم جهت ورودی یا خروجی پایه استفاده می‌شود. به این ترتیب که: اگر بیتی در رجیستر DDRX یک شده باشد، پین متناظر با آن در حالت خروجی و با مقدار صفر در حالت ورودی قرار می‌گیرد. که مطابق جدول (۲-۱) است.

۲- رجیستر PORTX : اگر DDRX.pin=1 باشد (پایه در حالت خرجی باشد) این رجیستر برای نوشتن یک مقدار منطقی در پین استفاده می‌شود و اگر DDRX.pin=0 باشد (پایه در حالت ورودی باشد) برای تنظیم امپدانس پایه به کار می‌رود. که مطابق جدول (۲-۱) است.

جدول (۲-۱): وضعیت پایه‌ها بر اساس محتوای رجیسترهای متناظر

DDRX.PIN	PORTX.PIN	وضعیت یا مقدار پایه (پین)
1	0	"0"
	1	"1"
0	0	Tri State
	1	Pull-up (بالاکش)

- در حالت Tristate، امپدانس پایه‌ی مورد نظر بی نهایت شده و ارتباط بین پایه و مدار داخلی پورت قطع می‌گردد.
- در حالت Pull-up، پین مورد نظر با یک مقاومت بالاکش داخلی به Vcc متصل می‌گردد. این حالت ورودی برای اتصال مدارهایی که خروجی سه حالته دارند به پایه ورودی میکروکنترلر مناسب است.

۳- رجیستر PINX : این رجیستر، تنها یک رجیستر خواندنی است که مقدار منطقی روی پایه‌ها را مستقل از این که در حالت ورودی یا خروجی پیکربندی شده باشند، نشان می‌دهد.

- حداکثر جریانی که هر پین در حالت خروجی تامین می‌کند به اندازه‌ی است که بتواند یک LED را روشن نماید. به همین دلیل، می‌توان LED ها را

مستقیماً به پایه‌های مورد نظر از میکروکنترلر متصل نمود (جریانی به اندازه ی 40mA در ولتاژ تغذیه‌ی پنج ولتی و جریان 20mA در ولتاژ تغذیه‌ی سه ولتی).

- با فعال‌سازی بیت PUD (یعنی نوشتن یک در بیت PUD) در رجیستر SFIOR می‌توان مقاومت‌های بالاکش کلیه پورت‌ها را غیر فعال نمود.
- در نرم‌افزار CodevisionAVR باید رجیستر ها با حروف بزرگ نوشته شوند.

ابتدا کلیه مراحل لازم را به منظور ایجاد یک پروژه بدون استفاده از ویزارد انجام دهید، سپس درمحیط ایجاد شده، برنامه‌ای مطابق زیر بنویسید:

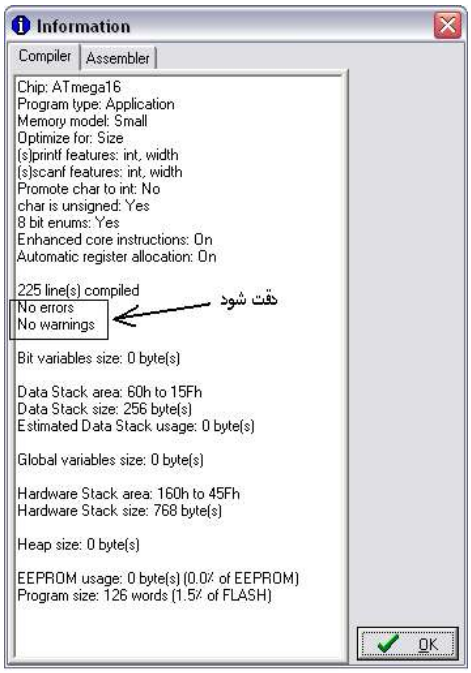
✓ برنامه:

```
#include <mega16.h>
#include <delay.h>
void main(void)
{DDRB.0=1;
PORTB.0=0;
while (1)
    {
        PORTB.0=1;
        delay_ms(1000);
        PORTB.0=0;
        delay_ms(1000);
  };}
```

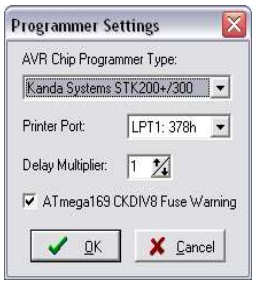
شرح برنامه: همان‌طور که مشاهده می‌شود ابتدا کتابخانه مربوط به میکروکنترلر AVR (که در اینجا ATmega16 است) و کتابخانه تاخیر، در ابتدای برنامه قرار داده شده‌اند. با استفاده از DDRB.0=1 پین صفرم از پورت B به عنوان خروجی و توسط PORTB.0=0 با مقدار صفر مقداردهی اولیه شده است. پس از اجرای برنامه، دستورات درون حلقه while برای همیشه تکرار می‌شوند. به این صورت که: ابتدا PORTB.0 یک و پس از یک ثانیه صفر می‌شود و دوباره پس از یک ثانیه یک می‌شود. در صورتی که پایه مثبت (آند) یک LED به PORTB.0 میکرو متصل و پایه منفی (کاتد) LED به زمین وصل شود این LED به‌صورت متناوب و با تاخیر یک ثانیه روشن و خاموش می‌شود.

کامپایل و برنامه‌ریزی تراشه: پس از اتمام نوشتن برنامه بر روی گزینه Compile and Make کلیک کرده یا کلیدهای Shift و F9 را به صورت همزمان فشار داده تا فایل Hex آن تولید شود. پنجره‌ای مطابق شکل (۲-۱) ظاهر می‌شود که

وضعیت عمل کامپایل و ساخت فایل Hex را نشان می‌دهد و در صورتی که برنامه اشکالی (املایی، ساختاری و نظایر آن) نداشته باشد، عبارت No errors در آن ظاهر می‌شود. اکنون که برنامه به درستی کامپایل و فایل Hex ساخته شده است، می‌باید فایل Hex به میکروکنترولر انتقال یابد (میکروکنترولر پروگرم شود). با توجه به پروگرمر ساخته شده در فصل اول، که از نوع STK200/300 است، در محیط CodeVisionAVR از گزینه‌ی Setting بر روی Programmer کلیک کرده تا پنجره‌ی شکل (۲-۲) ظاهر شود و از AVR programmer گزینه‌ی STK 200/300 را انتخاب کرده و دکمه‌ی OK را فشار داده تا پروگرمر مورد نظر انتخاب شود.

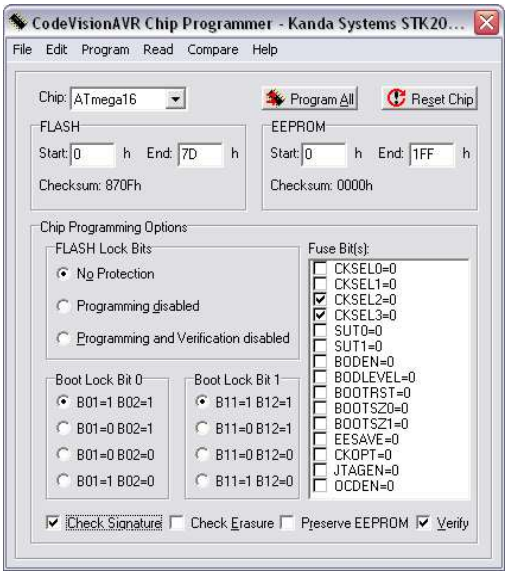


شکل (۲-۱): نتیجه‌ی عملیات کامپایل



شکل (۲-۲): انتخاب نوع پروگرمر

- اکنون، نوبت پروگرم نمودن میکروکنترلر است. مطابق شکل (۱-٤) از فصل اول، میکروکنترولر را به پروگرمر متصل نموده و منبع تغذیه را روشن نمائید. با فشردن همزمان کلیدهای Shift و F4 پنجره‌ای مطابق شکل (۲-۳) ظاهر می‌شود، فیوز بیت‌ها را برای نوسان ساز داخلی ٤ مگاهرتز و سایر قسمت‌ها را مطابق شکل (۲-۳) تنظیم و سپس بر روی Program All کلیک نمائید تا میکروکنترلر مورد نظر، برنامه‌ریزی شود. به منظور آشنایی کامل با قسمت‌های مختلف نرم‌افزار، همچنین روش دقیق برنامه ریزی میکروکنترلر به CD همراه کتاب مراجعه شود.



شکل (۲-۳): روش تنظیمات فیوز بیت‌ها و برنامه‌ریزی میکروکنترلر

تمرین: برنامه زیر، برنامه یک فلاشر است: ابتدا آن را تحلیل و سپس در محیط Codevision بنویسید و اجرای آن را توسط نرم‌افزار Proteus مشاهده نمائید؟

```
#include <mega16.h>
#include <delay.h>
#define xtal 4000000
int i;
void main (void)
{DDRD = 0xFF;
while(1)
  {   for(i = 1; i <= 128; i = i*2)
          {PORTD = i;delay_ms(100);}
      for(i = 128; i > 1; i = i/2)
          {PORTD = i;delay_ms(100);}
   }
}
```

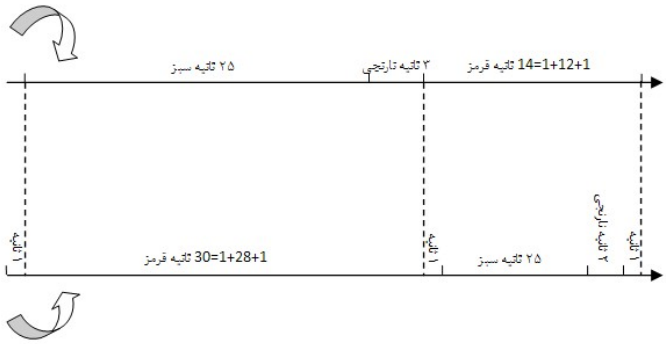
پروژه دوم: کنترل چراغ راهنمایی با نمایش مدت انتظار

بدون استفاده از وقفه‌ها و تایمرهای میکروکنترلر، برنامه‌ای بنویسید که چراغ‌های راهنمایی دو زمانه یک چهار راه را کنترل کند و با پیشنهاد یک روش بهینه به منظور استفاده از پایه‌های میکروکنترلر برنامه دو شمارنده معکوس شمار دو رقمی را برای دو سمت چهار راه بنویسید؟

فرضیات پروژه: میکروکنترلر مورد نظر ATmega16 است و از کریستال داخلی ٤ مگاهرتز استفاده شده است. برای این منظور ٦ عدد LED به پایه‌های پورت D میکروکنترلر مطابق جدول (٢-٢) متصل شده‌اند. وضعیت رنگ دو چراغ نسبت به هم مانند شکل (٢-٤) است. یک شستی تغییر وضعیت (Toggle) برای این چراغ لازم است که مامور راهنمایی و رانندگی بتواند با هر بار فشردن آن وضعیت چراغ‌ها را به طور همزمان عوض کند.

جدول (٢-٢): نحوه اتصال چراغ‌های راهنمایی به پورت‌های میکروکنترلر

	پایه درگاه	رنگ LED
چراغ خیابان عریضتر	PD0	سبز
	PD1	نارنجی
	PD2	قرمز
	PD3	
چراغ خیابان باریکتر	PD4	سبز
	PD5	نارنجی
	PD6	قرمز
	PD7	



شکل (۲-٤): وضعیت و مدت زمان روشن و خاموش بودن چراغ‌های راهنما

برای این کار، شستی متصل به پایه‌ی PD3 درنظر گرفته شده است. با فشردن این شستی، تنها در زمانی که یکی از چراغ‌ها سبز و دیگری قرمز است، باید باعث شود که: چراغی که سبز است، سه ثانیه نارنجی و سپس قرمز شود و چراغی که قرمز است، چهار ثانیه بعد سبز شود.

- **راهنمایی:** در هر بار پایان یافتن کوچکترین حلقه‌ای که برای ساختن زمآن‌ها نوشته‌اید، وضعیت کلید را چک کنید. دقت کنید که فاصله‌ی بین دوبار چک کردن این وضعیت در صورت فشرده نشدن در حدود چند میلی ثانیه و در صورت فشرده شدن در حد چند ثانیه باشد، که این خود یک نوع Debounce نرم افزاری است.

❋ یادآوری: برای نمایش اطلاعات روی چند نمایش‌گر «هفت قسمتی»[1] توسط میکروکنترلر و برای کاهش تعداد پایه‌های به کار رفته، معمولاً از روش تسهیم «خط – داده»[2] استفاده می‌شود. این نحوه نمایش بر پایه این مطلب استوار است که سیستم بینایی انسان قادر به تشخیص و تفکیک رویدادهایی که با سرعت بیش از حدود ۱۷ بار در ثانیه روی می‌دهد، نیست. مثلاً چشم ما، لامپی را که ۱۰۰ بار در ثانیه روشن و خاموش می‌شود، به طور پیوسته روشن می‌بیند، ولی با شدت نور کمتر از حالت روشن بودن دائم آن قابل رویت است.

یک روش معمول در اتصال و بسته‌بندی[3] چند نمایش‌گر هفت قسمتی، اتصال قسمت‌های متناظر همه‌ی نمایش‌گرها به هم و در نهایت خارج کردن ۸ پایه (هفت قسمت به همراه یک نقطه) از بسته‌بندی، به عنوان پایه‌های داده‌ی نمایش‌گر و خارج کردن جداگانه‌ی خط مشترک هر نمایش‌گر به عنوان پایه‌ی فعال‌ساز (یا همان آدرس) آن نمایش‌گر، است. بنابراین از یک بسته‌ی ٤ تایی از نمایش‌گرهای ۷ قسمتی که به این روش بسته‌بندی شده است، ۱۲ پایه شامل ۸ پایه داده مشترک میان همه نمایش‌گرها و ٤ پایه فعال‌ساز مخصوص هر نمایش‌گر (یا همان پایه های آدرس) خارج می‌شود. بنابراین، در این نوع اتصال در هر لحظه از زمان تنها یک داده‌ی واحد می‌تواند روی تعداد دلخواهی از آن‌ها به نمایش در آید. مالتی پلکس خط–داده همان روشی است که برای نمایش یک داده‌ی دلخواه (که لزوماً برای همه‌ی نمایش‌گرها مشابه نیست.) روی این نمایش‌گرها به کار می‌رود. اگر با سرعت بالا و در کسری از ثانیه، داده‌ی مربوط به هر نمایش‌گر را همزمان با فعال کردن آن، روی خط داده ارسال کنیم، سپس داده‌ی نمایش‌گر بعد را همزمان با فعال کردن آن و به همین ترتیب تا آخرین نمایش‌گر پیش برویم و مجدداً به نمایش‌گر اول بازگشته و این کار را به طور مداوم ادامه دهیم، با توجه به همان نکته‌ای که ذکر شد، کل داده به صورت ثابت و در یک زمان، روی مجموعه‌ی نمایش‌گرها مشاهده می‌شوند. با این کار به جای استفاده از ۳۲ پایه‌ی میکروکنترلر می‌توان از ۱۲ پایه‌ی آن برای راه اندازی[4] کامل ٤ نمایش‌گر هفت قسمتی استفاده کرد.

[1] 7-Segment
[2] Line-Data Multiplex
[3] Package
[4] Drive

☑ حل: به منظور کاهش بیشتر تعداد پایه‌های به کار رفته، می‌توان از یک «مبدل کد باینری به ۷ قسمتی»[1] (تراشه ۷٤٤۷) به عنوان واسط خط داده و از یک «کدگشای ۲ به ٤ خطی»[2] به عنوان واسط خط آدرس (تراشه ۷٤۱۳۹) استفاده کرد که این امر منجر به استفاده بهینه از پایه‌های میکرو می‌شود و در شکل (۲-۵) نشان داده شده است. جهت پیاده‌سازی نرم افزاری، ابتدا کلیه مراحل لازم را به منظور ایجاد یک پروژه بدون استفاده از ویزارد انجام داده، سپس درمحیط ایجاد شده، برنامه‌ای مطابق زیر بنویسید:

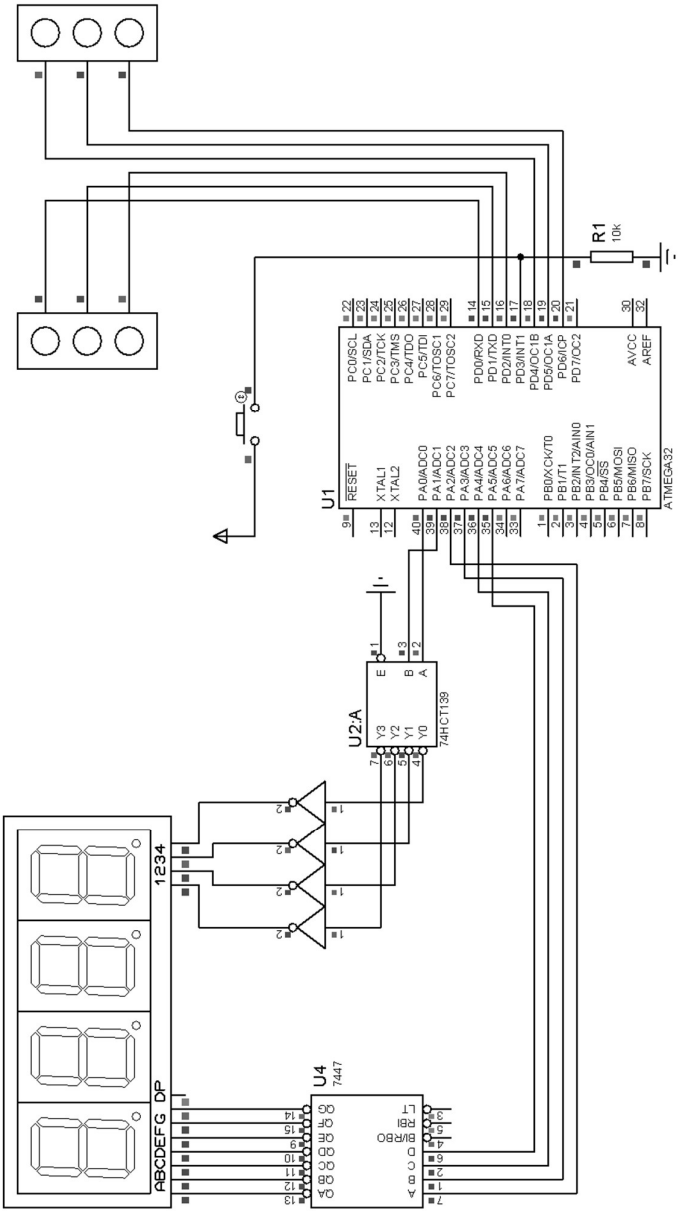
✓ برنامه:

```
#include <mega32.h>
#include <delay.h>
int s,x,m,n,q,r,i,b,t,d;
void output(void);
void time(void);
void change(void);
void main()
{ DDRA=0x3F; DDRD=0x77; t=2;
 while(1){
 if(t==2){s=25;x=29;}
 if(t==1){s=14;x=10;}
 m=s%10;
 n=s/10;
 q=x%10;
 r=x/10;
 time();
 }//end of while
 }
//*****************************************************
void time(void){
int f=1;
if(t==1)PORTD=0x14;
if(t==2)PORTD=0x41;
while(f){d=25;
while(d){
for(i=0;i<4;i++){
```

[1] BCD to 7-segment Decoder
[2] 2-to-4-Line Decoder/Demultiplexer

```
if(PIND.3==1)change();
switch(i){
case 3:b=n; break;
case 2:b=m; break;
case 1:b=r; break;
case 0:b=q; break;}//end of switch
output();
delay_ms(5);}//end of for
d--;}//end of while
if(q==0 & r==3)r=0;
if(m==0 & n!=0){n--;m=9;} else if(n==0 & m!=0){n=0;m--;} else
if(m!=0 & n!=0)m--;
if(q==0 & r!=0){r--;q=9;} else if(r==0 & q!=0){r=0;q--;} else if(q!=0 &
r!=0)  q--;
if(m==0 & n==0){if(t==1){t=2;f=0;  goto exit;}
else if(q<5&q>1)PORTD=0x42; else if(q<2)PORTD=0x44;}
if(q==0 & r==0){if(t==2){t=1;f=0;} else if(m<4 & m>1)PORTD=0x24;
else if(m==1){PORTD=0x44;r=3;q=0;}}
exit:
//end of exit
}//end of while
}
//*****************************************************
void output(void){
int c;
switch(b){
case 0:c=0x03;break;
case 1:c=0x07;break;
case 2:c=0x0B;break;
case 3:c=0x0F;break;
case 4:c=0x13;break;
case 5:c=0x17;break;
case 6:c=0x1B;break;
case 7:c=0x1F;break;
case 8:c=0x23;break;
case 9:c=0x27;break;
}//end of switch
if(i==0) PORTA=c&0xFC;
if(i==1) PORTA=c&0xFD;
if(i==2) PORTA=c&0xFE;
if(i==3) PORTA=c&0xFF;
```

```
}//end of output
//*****************************************************
void change(void){
if(t==2 & m>0){PORTD=0x42;m=0;n=0;r=0;q=4;d=25;}
if(t==1 & q>0){PORTD=0x24;m=3;n=0;r=0;q=0;d=25;}
}
```



شکل (۲-۵): سخت افزارکنترل چراغ راهنمایی و نمایش مدت انتظار با میکروکنترلر

پروژه سوم: نمایش کاراکترها بر روی نمایش‌گرهای ماتریسی

برنامه‌ای بنویسید که بر روی نمایش‌گر ماتریسی ۸×۸ حروف A، B و C را نمایش دهد؟

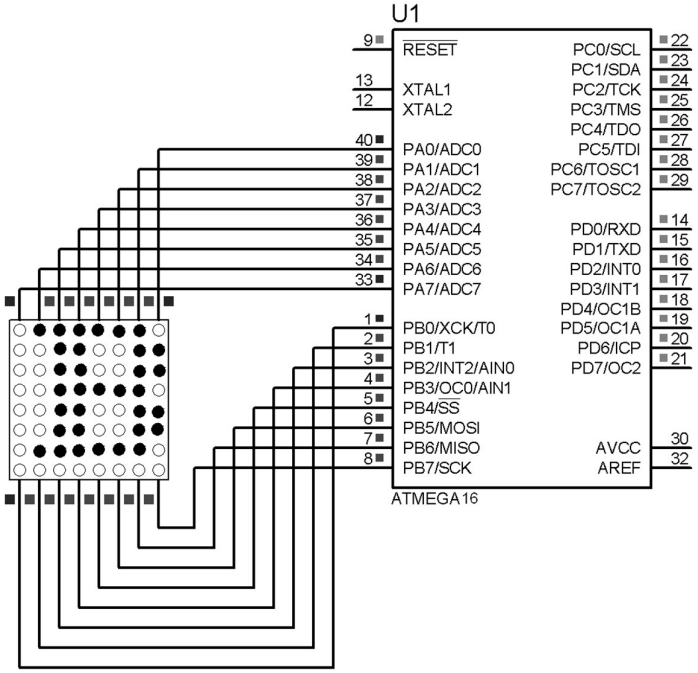
فرضیات پروژه: میکروکنترولر مورد نظر ATmega16 است و از کریستال داخلی ٤ مگاهرتز استفاده شده است. برای این منظور یک نمایش‌گر ماتریسی ۸×۸ به پایه‌های پورت A و B میکروکنترلر مطابق شکل (۲-٦) متصل شده‌اند.

حل: به منظور نمایش حروف بر روی نمایش‌گر ماتریسی، از روش جاروب نمودن سطرها و ستون‌ها استفاده می‌شود و می‌باید مطابق پروژه دوم عمل نمود.

برنامه:

```
#include <mega16.h>
#include <delay.h>
unsigned char k;  int l;
unsigned char arr1[8]={0x18, 0x3C, 0x66, 0x66, 0x7E, 0x66, 0x66, 0x00};
unsigned char arr2[8]={0x7E, 0x33, 0x33, 0x3E, 0x33, 0x33, 0x7E, 0x00};
unsigned char arr3[8]={0x1E, 0x33, 0x60, 0x60, 0x60, 0x33, 0x1E, 0x00};
void main(void)
{PORTA=0xFF; DDRA=0xFF;
 PORTB=0xFF; DDRB=0xFF;
while (1)
      {for(l=0;l<1000;l++)
          {for(k=0;k<=7;k++)
           {     PORTA=arr1[k];
                 PORTB&=~(1<<k);
                 delay_us(100);
                 PORTB=0xFF;
           }
          }
      for(l=0;l<1000;l++)
          {for(k=0;k<=7;k++)
           {     PORTA=arr2[k];
                 PORTB&=~(1<<k);
                 delay_us(100);
                 PORTB=0xFF;
           }
          }
      for(l=0;l<1000;l++)
          {for(k=0;k<=7;k++)
```

```
        {     PORTA=arr3[k];
              PORTB&=~(1<<k);
              delay_us(100);
              PORTB=0xFF;
        }
      }
    }
}
```



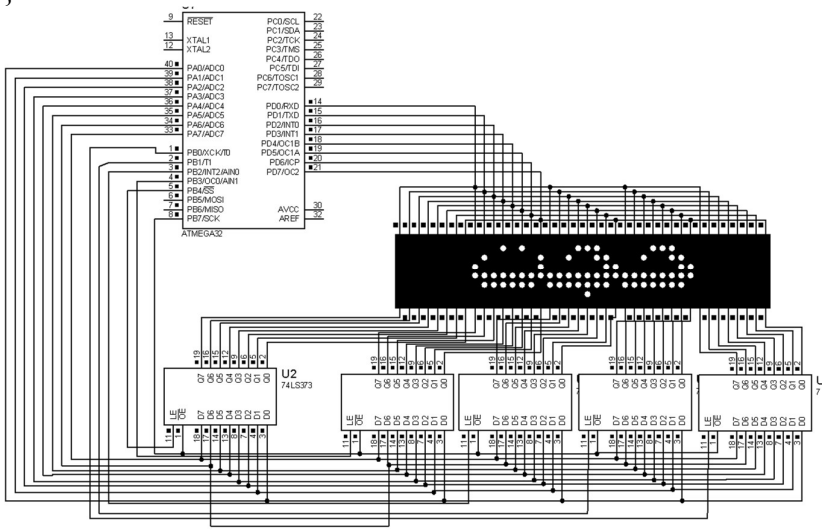
شکل (۲-٦): نمایش‌گر ماتریسی برای نمایش حروف

تمرین: برنامه زیر را تحلیل نمائید و آن را در توسط نرم‌افزار CodeVision بازنویسی کنید. سپس نتایج آن را در محیط نرم‌افزار Proteus با نتایج شکل (۲-۷) مقایسه نمائید؟

```
#include <mega32.h>
#include <delay.h>
unsigned char i,j,k,n,r, lsh, pos,char1,char2,charnum;
```

```
unsigned char ar1[17][8]={
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x1C,0x26,0x03,0xFF,0xFE,0x00,0x00,
0x00,0x1C,0x26,0x03,0xFF,0xFE,0x08,0x00,
0x00,0x14,0x41,0x81,0xFF,0x7E,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

void main(void)
{
charnum=13;
charnum=charnum-4;
PORTA=0x00;DDRA=0xFF;
PORTB=0x00;DDRB=0xFF;
PORTD=0xFF;DDRD=0xFF;

while(1)
{
pos=0;
  for(i=0;i<=charnum;i++)
    {
      lsh=8;
        for(r=1;r<=8;r++)
          {
            lsh--; pos++;
                for(n=1;n<=33;n++)
                  {
                    for(k=0;k<=7;k++)
                      {
                          for(j=0;j<=4;j++)
                            {
                              char1=ar1[i+j][k]>>r;
                              char2=ar1[i+j+1][k]<<lsh;
                              PORTA=char1|char2;
```

```
                    PORTB|=1<<j;PORTB=0x00;
                }
            PORTD&=~(1<<k);
            delay_us(66);
            PORTD=0xFF;
            }
        }
    }
  }
 }
}
```



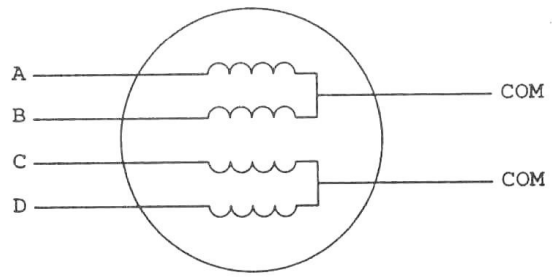
شکل (۲-۷): پیکربندی تابلوی روان ماتریسی

بازی و سرگرمی: با مراجعه به CD همراه کتاب، علاوه بر مشاهده برنامه بازی مار (Snake) در محیط CodeVision با اجرای آن در محیط Proteus سرعت عمل خود را آزمایش نمائید و در ساختار برنامه را تحلیل کنید.

موتورهای پله‌ای دارای کاربرد زیاد به منظور کنترل موقعیت است. زیرا، میزان چرخش شفت موتور متناسب با تعداد پالس‌های اعمال شده به هر فاز آن می‌باشد. این موتورها، در کاربردهایی همچون درایورهای سخت، پرینترهای ماتریسی، کنترل بازوهای ربات و نظایر آن استفاده می‌شوند. هر موتور پله‌ای از یک هسته متحرک مغناطیسی دائمی که به آن روتور (یا شفت) می‌گویند و یک بخش ثابت که استاتور

نامیده می‌شود، تشکیل شده است. موتورهای پله‌ای عموماً دارای چهار سیم‌پیچ استاتور هستند و به موتورهای چهار فاز معروف می‌باشند. یک موتور پله‌ای علاوه بر چهار سیم ذکر شده، ممکن است یک یا دو سیم برای اتصال به ولتاژ تغذیه مثبت یا منفی داشته باشد. شکل (۲-۸) ارتباط بین سیم‌پیچ‌های استاتور را نشان می‌دهد.

برای راه‌اندازی موتور پله‌ای به طور کلی دو روش «گام کامل»[1] و «نیم گام»[2] وجود دارد. که روش گام کامل، به دو رهیافت هدایت تک‌فاز و هدایت دو فاز تقسیم‌بندی می‌گردد که در ادامه به توضیح هر یک از این روش‌ها می‌پردازیم.

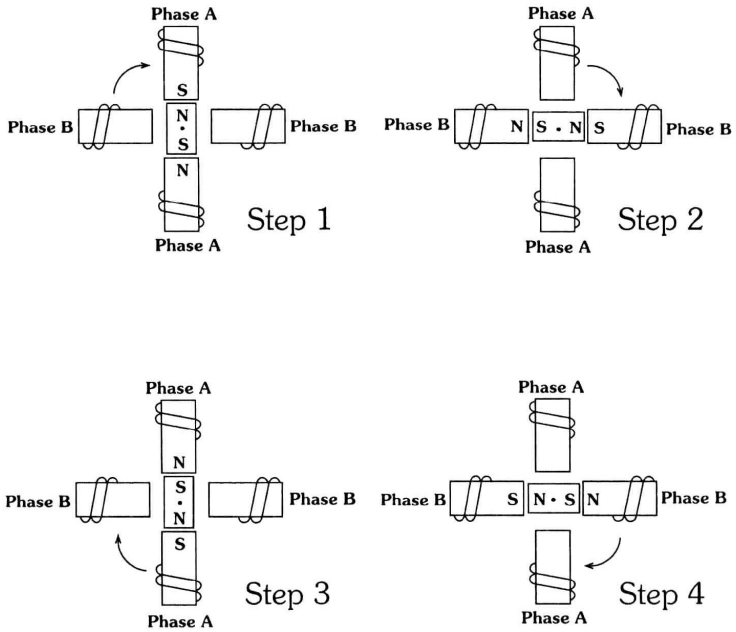


شکل (۲-۸): نحوه‌ی پیکربندی سیم‌پیچ‌های استاتور

الف- راه اندازی موتور به صورت گام کامل

۱- راه اندازی موتور به روش هدایت تک‌فاز: در این روش مطابق شکل (۲-۹)، در هر لحظه تنها یک جفت از سیم‌پیچ‌های استاتور و روتور همدیگر را جذب می‌کنند. در این صورت با فعال‌سازی مناسب سیم‌پیچ‌های استاتور چهار حالت شکل (۲-۹)، بوجود می‌آیند. به منظور راه‌اندازی موتور به صورت هدایت تک‌فاز، باید از جدول (۲-۳) استفاده کرد. با اجرای هر یک از این حالت‌ها شفت موتور یک پله جابجا می‌شود.

[1] Full Step
[2] Half Step



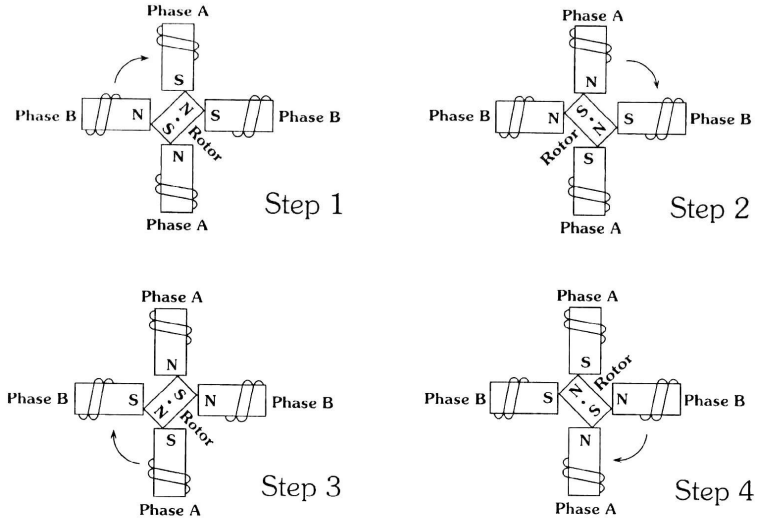
شکل (۲-۹): راه‌اندازی موتور پله‌ای به صورت هدایت تک‌فاز

جدول (۲-۳): مراحل راه‌اندازی موتور پله‌ای به صورت هدایت تک‌فاز

در جهت ساعت ↓	#مرحله	سیم‌پیچ A	سیم‌پیچ B	سیم‌پیچ C	سیم‌پیچ D	خلاف جهت ساعت ↑
	۱	۱	۰	۰	۰	
	۲	۰	۱	۰	۰	
	۳	۰	۰	۱	۰	
	٤	۰	۰	۰	۱	

۲- راه‌اندازی موتور پله‌ای به صورت هدایت دو فاز: در این روش، مطابق شکل (۲-۱۰) در هر لحظه هر دو جفت از سیم‌پیچ‌های استاتور با قطب‌های ناهمنام فعال می‌شوند که در این صورت قطب‌های روتور در جهت برایند قطب مخالف استاتور قرار می‌گیرند. در این حالت نیز با فعال‌سازی مناسب سیم‌پیچ‌های استاتور چهار حالت شکل (۲-۱۰) بوجود می‌آیند. تفاوت این روش با روش قبل در این است که: در این حالت گشتاور ایجاد شده افزایش می‌یابد و برای

کاربردهایی که به قدرت بیشتری نیاز دارند مفید است. در این حالت، برای راه‌اندازی موتور از جدول (۲-٤) استفاده می‌شود. با اجرای هر یک از این حالت‌ها شفت موتور یک پله جابجا می‌شود.



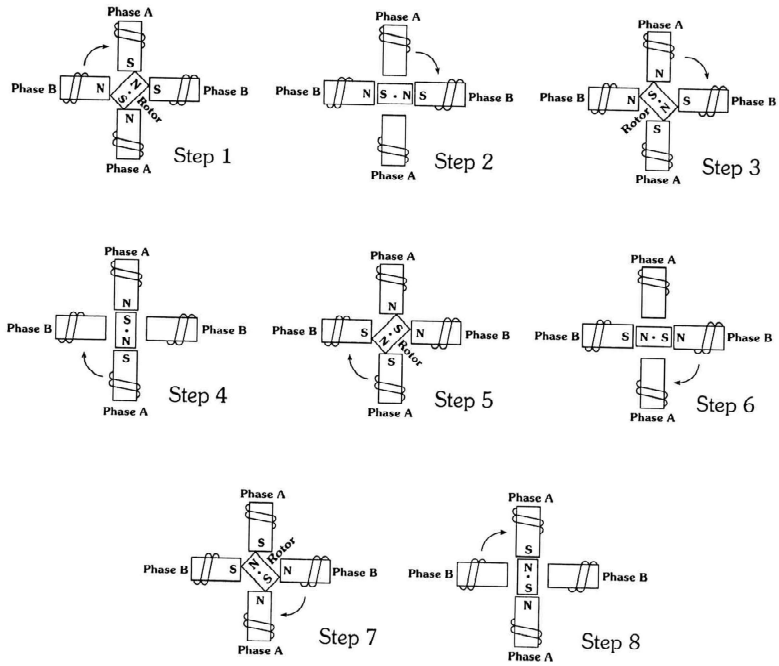
شکل (۲-۱۰): راه‌اندازی موتور پله‌ای به صورت هدایت دو فاز

جدول (۲-٤): مراحل راه‌اندازی موتور پله‌ای به صورت هدایت دو فاز

خلاف جهت ساعت	سیم‌پیچ D	سیم‌پیچ C	سیم‌پیچ B	سیم‌پیچ A	#مرحله	در جهت ساعت
↑	۱	۰	۰	۱	۱	↓
	۰	۰	۱	۱	۲	
	۰	۱	۱	۰	۳	
	۱	۱	۰	۰	٤	

ب- راه‌اندازی موتور به صورت نیم گام

در این حالت در هر بار تغییر وضعیت سیم‌پیچ‌های استاتور، میزان گردش موتور نصف حالت‌های قبل خواهد بود. در این صورت به جای چهار حالت مختلف، هشت حالت وجود خواهد داشت که با فعال شدن مناسب آن‌ها، می‌توان موتور را راه‌اندازی نمود. چگونگی عملکرد موتور در این روش، مطابق شکل (۲-۱۱) است



شکل (۲-۱۱): راه‌اندازی موتور پله‌ای به صورت نیم گام

جدول (۲-۵): مراحل راه‌اندازی موتور پله‌ای به صورت نیم گام

در جهت ساعت	#مرحله	سیم‌پیچ A	سیم‌پیچ B	سیم‌پیچ C	سیم‌پیچ D	خلاف جهت ساعت
	۱	۱	۰	۰	۰	
	۲	۱	۱	۰	۰	
	۳	۰	۱	۰	۰	
	۴	۰	۱	۱	۰	
↓	۵	۰	۰	۱	۰	↑
	۶	۰	۰	۱	۱	
	۷	۰	۰	۰	۱	
	۸	۱	۰	۰	۱	

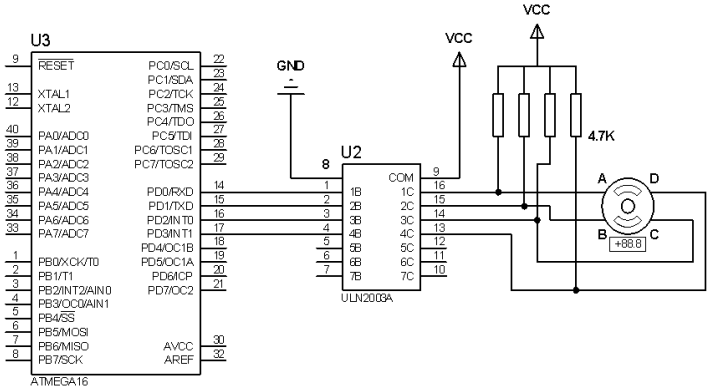
و از جدول (۲-۵) به منظور راه‌اندازی موتور استفاده می‌شود. تفاوت این روش با روش راه اندازی گام کامل هدایت دو فاز این است که: در این روش دقت موتور به دو برابر افزایش می‌یابد ولی میزان گشتاور تولید شده در این حالت، به اندازه‌ی ۱۵ تا ۳۰ درصد کاهش می‌یابد.

زاویه پله: موتورهای پله‌ای مختلف، مشخصات گشتاور، ولتاژ و تعداد پله در دور متفاوتی دارند. تعداد پله در دور، میزان دقت موتور را نشان می‌دهد. برای بدست آوردن زاویه هر پله، می‌توان ۳٦۰ درجه را بر تعداد پله در دور تقسیم نمود. جدول (۲-٦) تعداد پله در دور و زاویه هر پله را در چند موتور مختلف را نشان می‌دهد.

جدول(۲-٦): زاویه پله برای انواع موتورهای پله‌ای

زاویه‌ی پله (درجه)	پله در دور
0.72	500
1.8	200
2.0	180
2.5	144
5	72
7.5	48
15	24

سخت‌افزار مورد نیاز برای اتصال یک موتور پله‌ای چهار فاز به میکروکنترلر در شکل(۲-۱۲) ترسیم شده است. از آنجایی که جریان دهی میکرو برای تحریک موتور پله‌ای مناسب نیست، از تراشه ULN2003 که حاوی بافر جریان می‌باشد، استفاده شده است. مشخصات تراشه‌های خانواده ULN200xA در جدول (۲-۷) آورده شده است.



شکل(۲-۱۲): سخت‌افزار مدار کنترل موتور پله‌ای ٤ فاز

جدول (۲-۷): مشخصات تراشه‌های خانواده ULN200xA

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V, PMOS
ULN2003A	5V TTL , CMOS
ULN2004A	6-15 V ,CMOS , PMOS

پروژه چهارم: کنترل موتورهای پله‌ای

الف- برنامه‌ای بنویسید که یک موتور پله‌ای ٤ فاز را با دقت 1.8^0 به صورت گام کامل هدایت تک‌فازی به اندازه‌ی 90^0 در جهت ساعت‌گرد بچرخاند؟

حل: از آنجا که در تحریک به صورت پله کامل، هر بار تحریک روتور، منجر به چرخش روتور به اندازه 1.8^0 می‌شود. برای چرخش 90^0 باید 90/1.8=50 بار پالس فرمان تولید گردد.

برنامه:

```
#include <mega16.h>
#include <delay.h>
int i=0, k=0,num;
void main(void)
{
DDRD=0x0F;
ACSR=0x80;
SFIOR=0x00;
    for(i=1;i<=13;i++)
            {
        PORTD=0b00000001;
            delay_ms(30);
              num++;
            for(k=1;k<=3;k++)
                    {
                    PORTD=PORTD<<1;
                    delay_ms(30);
                num++;
                if (num==51) k=4;
                    }
    }
```

```
}
```

✍ ب- برنامه‌ای بنویسید که یک موتور پله‌ای ٤ فاز را با دقت 1.8^0 به صورت گام کامل، به اندازه‌ی 90^0 در جهت ساعت‌گرد بچرخاند؟

☑ حل: برای تحریک نیم گام برای چرخش به اندازه 90^0 باید تعداد تحریک‌های هدایت تک‌فاز ۵۰ مرتبه و تعداد تحریک‌های هدایت دو فاز نیز ۵۰ مرتبه باشند به طوری که تحریک هدایت تک‌فاز و هدایت دو فاز به صورت یک در میان به طور متوالی صورت پذیرد.

✓ برنامه:

```
#include <mega16.h>
#include <delay.h>
int d,a,i=0,k=0,num;
void main(void)
{
DDRD=0x0F;
ACSR=0x80;
SFIOR=0x00;
for(i=1;i<=13;i++)
        {
                d=0b00000001;
                a=0b00000011;
        for(k=1;k<=4;k++)
                        {
                        PORTD=d;
                        d=d<<1;
                        delay_ms(30);
                         PORTD=a;
                         a=a<<1;
                         if(a==24) a=9;
                        delay_ms(30);
                        num++;
                        if(num==51) k=4;
                        }
            }
}
```

✍ پ- برنامه‌ای بنویسید که یک موتور پله‌ای ٤ فاز را با دقت 1.8^0 به صورت پله کامل، به اندازه‌ی 360^0 در جهت ساعت‌گرد با سرعت ۵۰ دور در دقیقه (RPM) بچرخاند؟

☑ **حل:** برای این که بتوان موتور را کنترل نمود، کافی است که تاخیرهای مناسب بین تحریک‌های متوالی فازها وجود داشته باشد.

- در هر دقیقه، موتور ۵۰ دور می‌چرخد. یعنی، هر دور چرخش موتور ۱۲۰۰ میلی‌ثانیه زمان می‌برد.
- در روش گام کامل، هر دور کامل یعنی چرخش ۳٦۰ درجه نیازمند ۲۰۰ مرتبه تحریک موتور پله‌ای است (تعداد تحریک‌ها در حالت هدایت تک‌فازی و هدایت دو فازی یکسان هستند). یعنی میزان تاخیر هر تحریک معادل ٦ میلی‌ثانیه است. رابطه کلی در رابطه با زمان تاخیر مطابق زیر است:

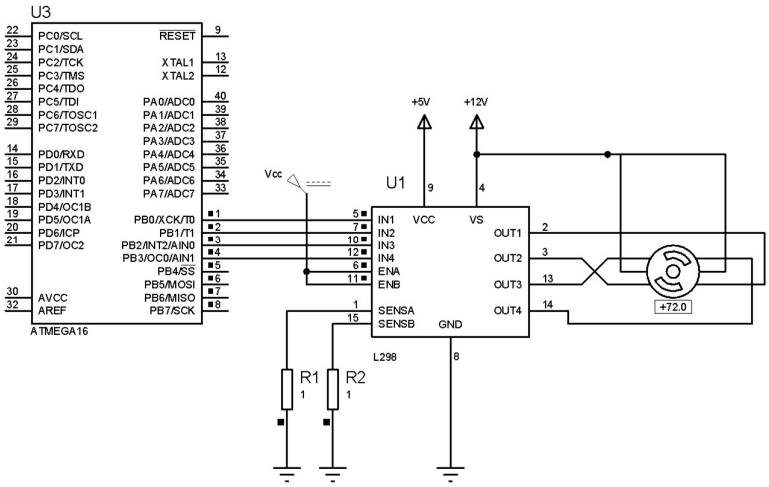
(تعداد پله در دور × سرعت برحسب RPM) /٦۰ = میزان تاخیر در تحریک با گام کامل

- **برای مطالعه:** در روش نیم گام، هر دور کامل یعنی چرخش ۳٦۰ درجه نیازمند ٤۰۰ مرتبه تحریک موتور پله‌ای است (تعداد تحریک‌های هدایت تک‌فاز ۲۰۰ مرتبه و تعداد تحریک‌های هدایت دو فاز نیز ۲۰۰ مرتبه می‌باشد). یعنی میزان تاخیر هر تحریک معادل ۳ میلی‌ثانیه است.

(تعداد پله در دور×۲× سرعت برحسب RPM) /٦۰ = میزان تاخیر در تحریک نیم گام

✓ **برنامه:**

```
#include <mega16.h>
#include <delay.h>
int i=0, k=0,num;
void main(void)
{
DDRD=0x0F;
ACSR=0x80;
SFIOR=0x00;
    for(i=1;i<=50;i++)
                {PORTD=0b00000001;
                delay_ms(6);num++;
                for(k=1;k<=3;k++)
                        { PORTD=PORTD<<1;
                        delay_ms(6);
                    num++;
                    if(num==51) k=4; }
                }
}
```



شکل(۲-۱۳): مدار کنترل موتور پله‌ای ٤ فاز با استفاده از تراشه L298

- **نکته:** اگر بخواهیم سرعت موتور را روی مقدار نسبتاً زیاد تنظیم نمائیم، نمی‌توان از لحظه راه‌اندازی موتور، تاخیر بین تحریک‌های متوالی را بر اساس سرعت مطلوب نهایی محاسبه کرد چرا که به دلیل لختی زیاد، آرمیچر نمی‌تواند میدان مغناطیسی استاتور را دنبال نماید بنابراین باید به تدریج سرعت میدان استاتور (تحریک استاتور) را افزایش دهیم تا روتور بتواند میدان استاتور را دنبال کند و اصطلاحاً از حالت سنکرون خارج نشود.

فصل سوم نمایش‌گرهای کاراکتری و گرافیکی

امروزه استفاده از «نمایشگر با کریستال مایع»[1] یا به اختصار LCD ها دارای کاربرد فراوانی در صنایع مهندسی هستند و جایگزین خوبی برای نمایش‌گرهای «هفت قسمتی»[2] می‌باشند. مزیت آن‌ها نسبت به نمایش‌گرهای هفت قسمتی (که در فصل دوم به تفصیل توضیح داده شد)، بر اساس موارد زیر است:

۱- برخلاف نمایش‌گرهای هفت قسمتی که فقط قادر به نمایش اعداد و چند حرف خاص هستند، LCD ها توانایی نمایش اعداد، کاراکترها و تصاویر گرافیکی را دارند.

۲- برخلاف نمایش‌گرهای هفت قسمتی که برای نمایش پیوسته اعداد و حروف خاص، باید به طور متوالی تازه‌سازی[3] شوند، LCD ها دارای یک کنترل کننده ی تازه‌سازی درونی می‌باشند.

۳- تعداد پایه‌های اشغال شده از میکروکنترلر برای کار با LCD ثابت بوده و ارتباطی به تعداد کاراکترهای آن ندارد.

۳-۱- LCD های کاراکتری

این نوع LCD ها در دو نوع سریال و موازی به بازار عرضه شده‌اند. در نوع سریال، LCD ها دارای سه پایه برای GND، VCC و انتقال سریال می‌باشند. نرخ ارسال در این نوع از LCD ها ۲٤۰۰ یا ۹٦۰۰ بیت بر ثانیه و قالب دیتا به صورت ۸ بیت داده، بدون بیت توازن و یک بیت پایان می‌باشد.

LCD های نوع موازی از ۸ خط داده به منظور ارسال اطلاعات و ۳ خط دیگر برای تعیین نوع اطلاعات (شامل دستور و داده) و همزمان‌سازی آن‌ها استفاده می‌کنند. LCD مورد بحث در این قسمت ۱٦ پایه دارای ابعاد ۱٦×۲ (دو سطر در شانزده ستون) می‌باشد. جدول (۳-۱) مکان پایه‌ها ی LCD را مشخص می‌کند.

---

[1] Liquid Crystal Display (LCD)
[2] 7-Segments
[3] Refreshing

جدول(۳-۱): مشخصات پایه‌های LCD

شماره پایه	نام	جهت داده	نوع عملکرد
۱	GND	-	0V
۲	VCC	-	5V
۳	VEE	-	تنظیم شدت نور
٤	RS	ورودی	انتخاب رجیستر
٥	R/W	ورودی	خواندن/نوشتن
٦	E	ورودی/خروجی	فعال سازی
۷	DB0	ورودی/خروجی	باس داده ۸ بیتی
۸	DB1	ورودی/خروجی	باس داده ۸ بیتی
۹	DB2	ورودی/خروجی	باس داده ۸ بیتی
۱۰	DB3	ورودی/خروجی	باس داده ۸ بیتی
۱۱	DB4	ورودی/خروجی	باس داده ۸ بیتی
۱۲	DB5	ورودی/خروجی	باس داده ۸ بیتی
۱۳	DB6	ورودی/خروجی	باس داده ۸ بیتی
١٤	DB7	ورودی/خروجی	باس داده ۸ بیتی
۱۵	A	–	تنظیم نور زمینه‌ی آند
١٦	K	–	تنظیم نور زمینه‌ی کاتد

شرح پایه‌ها مطابق زیر است:

RS، انتخاب‌گر رجیستر: این پایه برای انتخاب یکی از دو رجیستر موجود در LCD استفاده می‌شود. چنانچه RS صفر باشد، رجیستر دستورالعمل فرمان انتخاب می‌شود که به این ترتیب می‌توان فرمآن‌هایی همچون پاک کردن LCD، تغییر مکان‌نما و نظایر آن را صادر کرد و چنانچه RS یک باشد، رجیستر داده انتخاب می‌شود که به این ترتیب می‌توان داده‌ای را به LCD ارسال کرد.

R/W، خواندن/نوشتن: چنانچه این پایه صفر باشد، کاربر می‌تواند اطلاعات را بر روی LCD بنویسد و اگر یک باشد، کاربر اجازه ی خواندن اطلاعات را دارد.

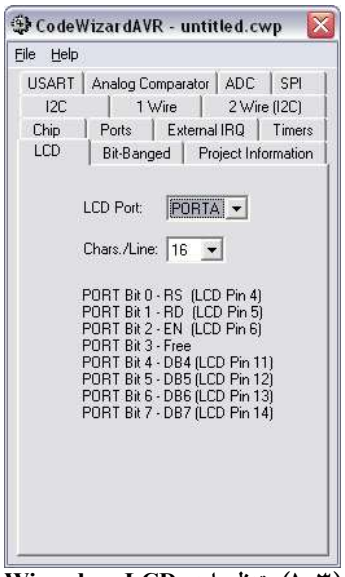
E، **فعال‌ساز:** هنگام ارسال داده یا فرمان به LCD باید یک پالس پایین رونده با حداقل پهنای 450ns به این پایه اعمال شود تا LCD اطلاعات موجود در پایه‌های داده را لچ کند.

پایه‌های D0-D7 : از این پایه‌ها برای ارسال داده یا فرمان به LCD یا خواندن محتوای رجیستر داخلی LCD استفاده می‌شود اگر RS=1 باشد اطلاعات ارسالی داده و در غیر این صورت دستورالعمل تلقی می‌شوند. جدول (۳-۲) کدهای مجاز دستورالعمل و فرمان را نشان می‌دهد. این نوع ارتباط می‌تواند به صورت ارتباط گذرگاه[1] ۸ سیمه باشد که در این صورت هر ۸ پایه‌ی D0 تا D7 مورد استفاده قرار می‌گیرند و یا این که می‌تواند به صورت گذرگاه ٤ سیمه باشد که در این صورت فقط ٤ پایه‌ی D4 تا D7 استفاده می‌شوند در میکروکنترلرهای AVR معمولاً از باس ٤ سیمه استفاده می شود.

جدول(۳-۲): کدهای فرمان LCD

کد	شرح دستور
1	پاک کردن صفحه‌ی نمایش
2	بازگشت به مکان اول (شروع)
4	کاهش مکان‌نما (جابجایی مکان‌نما به چپ)
5	افزایش مکان‌نما (جابجایی مکان‌نما به راست)
6	جابجایی نمایش به راست
7	جابجایی نمایش به چپ
8	صفحه‌ی نمایش خاموش و مکان‌نما خاموش
A	صفحه‌ی نمایش خاموش و مکان‌نما روشن
C	صفحه‌ی نمایش روشن و مکان‌نما خاموش
E	نمایش روشن مکان نما روشن
F	صفحه‌ی نمایش روشن و مکان‌نما در حالت چشمک زن
10	جابجایی محل مکان‌نما به چپ
14	جابجایی محل مکان‌نما به راست
18	کل صفحه نمایش به چپ جابجا شود
CO	مکان نما به آغاز خط دوم برود
38	سازمان دهی دو خط و ماتریس ۷×۵

[1] BUS



شکل (۳-۱): تنظیمات LCD در CodeWizard

۳-۱-۱- برنامه‌ریزی LCD توسط CodeWizard :

با استفاده از محیط wizard مطابق شکل (۳-۱) می‌توان LCD را برنامه‌ریزی کرد.

- **LCD Port**: پورت مورد نظر برای LCD مشخص می‌شود.
- **Char/Line**: نوع LCD مورد نظر مشخص می‌شود.

(با انتخاب پورت مورد نظر wizard نحوه اتصال LCD به آن پورت را مشخص می‌کند).

۳-۱-۲- دستورات مربوط به کار با LCD به شرح زیر می‌باشند:

- **void lcd_init(unsigned char lcd_columns)** : از این تابع برای پیکربندی LCD استفاده می‌شود و چنانچه از wizard استفاده می‌کنید به طور خودکار به برنامه اضافه می‌شود.
- **void lcd_clear(void)**: از این تابع برای پاک کردن صفحه نمایش LCD استفاده می‌شود.
- **void lcd_gotoxy(unsigned char x, unsigned char y)** : این تابع، مکان‌نما را به ستون x و سطر y انتقال می‌دهد. دقت شود که: شماره‌های سطرها و ستون‌ها از صفر شروع می‌شوند.

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)

- **void lcd_putchar(char c)** : این تابع کاراکتر c را در محل فعلی مکان‌نما چاپ می کند.
- **void lcd_puts(char *str)** : این تابع رشته str را در محل فعلی مکان‌نما چاپ می‌کند.
- **void lcd_putsf(char flash *str)** : این تابع رشته str را در محل فعلی مکان‌نما چاپ می‌کند. با این تفاوت که: رشته str در حافظه flash قرار دارد.
- **void lcd_ready(void)** : این تابع آن قدر صبر می‌کند تا LCD خود را برای دریافت دستور آماده کند. دقت شود که: این تابع باید قبل از LCD_write_data استفاده شود.
- **void_lcd_write_data(unsigned char data)**: برای نوشتن داده data در رجیستر LCD به کار می‌رود.
- **void lcd_write_byte(unsigned char addr, unsigned char data)**: این تابع برای نوشتن یک بایت داده data در آدرس addr حافظه RAM به کار می‌رود. دقت شود که: از این تابع برای نوشتن کاراکترهای دلخواه استفاده می‌شود.

پروژه پنجم: نمایش گردشی یک عبارت بر روی LCD

برنامه‌ای بنویسید که کلمه Hello را به طور گردشی روی LCD نشان دهد؟

حل: در این برنامه رشته Hello، توسط دستور lcd_gotoxy()، ۱۲ مرتبه در ۱۲ ستون LCD با تاخیر ۵۰۰ میلی ثانیه به نمایش در می‌آید و دستورهای پایانی برنامه برای به نمایش درآوردن حالت چرخشی رشته بر روی LCD مطابق شکل (۳-۲) تا (۳-۵) استفاده شده است.

o Hell	lcd _gotoxy (12,0);
	lcd _putsf("Hell");
	lcd _gotoxy (0,0);
	lcd _putsf("o");

شکل(۳-۲): برنامه‌نویسی ستون‌های پایانی LCD

lo Hel	lcd _gotoxy (13,0);
	lcd _putsf("Hel");
	lcd _gotoxy (0,0);
	lcd _putsf("lo");

شکل(۳-۳): برنامه‌نویسی ستون‌های پایانی LCD

llo He	lcd _gotoxy (14,0) ;
	lcd _putsf("he");
	lcd _gotoxy (0,0);
	lcd _putsf("llo");

شکل (۳-٤): برنامه‌نویسی ستون‌های پایانی LCD

ello H	LCD_putsf("h"); LCD_gotoxy (0,0); LCD_putsf("ello");

شکل (۳-۵): برنامه‌نویسی ستون‌های پایانی LCD

✓ برنامه:

```
#include <mega16.h>
#asm
.equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
#include<delay.h>
void main(void)
{
char i;
DDRA=0xFF;
ACSR=0x80;
lcd_init(16);
while (1)
    {
    i = 0;
    for(i=0;i<12;i++)
        {
        lcd_clear();
        lcd_gotoxy (i,0);
        lcd_putsf("Hello");
        delay_ms(500);
      }
    lcd_gotoxy (12,0 );
    lcd_putsf("Hell");
    lcd_gotoxy (0,0);
    lcd_putsf("o") ;
    delay_ms(500);
    lcd_clear();
```

```
    lcd_gotoxy (13,0);
    lcd_putsf("Hel");
    lcd_gotoxy (0,0);
    lcd_putsf("lo");
    delay_ms(500);
    lcd_clear();
    lcd_gotoxy (14,0) ;
    lcd_putsf("He");
    lcd_gotoxy (0,0);
    lcd_putsf("llo");
    delay_ms(500);
    lcd_clear() ;
    lcd_gotoxy (15,0) ;
    lcd_putsf("H");
    lcd_gotoxy (0,0);
    lcd_putsf("ello");
    delay_ms(500);
    lcd_clear();
    };
}
```

پروژه ششم: نمایش حرف به حرف روی LCD

برنامه‌ای بنویسید که روی LCD ابتدا پیغام hello و سپس نام هشت تن از دوستان خود را با توجه به نمونه‌ی زیر چاپ کند؟

فرضیات: فرض کنید که اولین نام Reza Teymoorfar است. این نام باید حرف به حرف چاپ شود. یعنی، ابتدا حرف R، سپس Re و سایر حروف تا اسم و فامیل به طور کامل چاپ شود. بعد از چاپ کامل اسم و فامیل، می‌باید حروف چاپ شده پاک شوند. پاک کردن نیز باید از آخر به اول و حرف به حرف باشد. یعنی ابتدا چاپ شود: Reza Teymoorfar، سپس Reza Teymoorfa و سایر حروف تا اسم و فامیل کاملاً پاک شود.

حل: در این برنامه از آرایه‌ی دو بعدی استفاده شده است، به این ترتیب که: اگر آرایه‌ی دو بعدی را همانند یک ماتریس یا جدول در نظر بگیرید، سطرهای این ماتریس (بعد اول) تعداد اسامی و ستون‌های این ماتریس (بعد دوم) حروف اسامی را شامل می‌شوند. این برنامه شامل سه حلقه "for" می‌باشد که حلقه اول، ۸ اسم مورد نظر را به ترتیب برای نمایش بر روی LCD وارد حلقه دوم می‌کند. با هر بار تکرار حلقه دوم یکی از حروف اسم مورد نظر توسط دستور

sprintf(s,"%c",s1[i][t]) وارد متغیر s شده و سپس توسط دستور lcd_puts(s) روی LCD چاپ می‌شود. پس از چاپ تک تک حروف، برنامه وارد حلقه دیگری می‌شود که در آنجا برای پاک کردن کلمه چاپ شده، به ترتیب از آخرین حرف تا اولین حرف آن کاراکتر" " چاپ می‌گردد.

✓ برنامه:

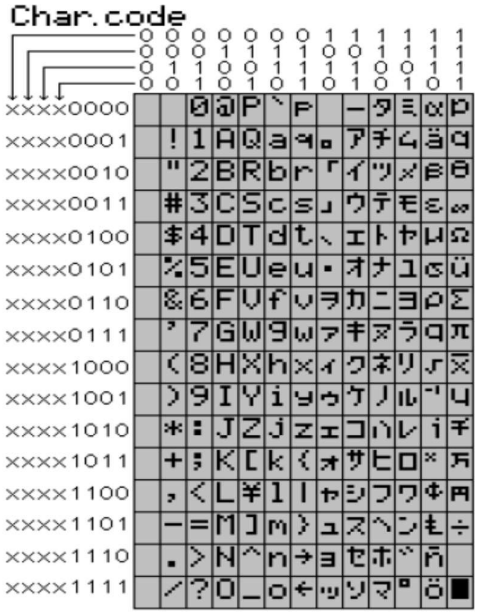
```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
// Alphanumeric LCD Module functions
#asm
.equ __lcd_port=0x1B
#endasm
#include <lcd.h>
int t,z;
char i;
char s[2] ;
char s1[20][16];
void main(void)
{
PORTA=0x00;DDRA=0xFF;
PORTB=0x00;DDRB=0xff;
ACSR=0x80;SFIOR=0x00;
// LCD module initialization
lcd_init(16); lcd_putsf("     Hello"); delay_ms(2000);
lcd_clear();lcd_gotoxy(0,0);
sprintf(s1[0],"Reza Teymoorfar");
sprintf(s1[1],"Hamed Saghaei");
sprintf(s1[2],"Farshid Koohi");
sprintf(s1[3],"Saeed Bahrami");
sprintf(s1[4],"Sima Aref");
sprintf(s1[5],"Sayeh Tehrani");
sprintf(s1[6],"Ramin Amiri");
sprintf(s1[7],"Arsalan Saei");
while (1)
    {
for(i=0;i<8;i++)
        {
         lcd_clear();lcd_gotoxy(0,0);
            for(t=0;t<15;t++)
                {
```

```
                sprintf(s,"%c",s1[i][t]);lcd_puts(s);delay_ms(1000);
            }
        for(z=15;z>0;z--)
            {
                lcd_gotoxy(z,0);lcd_putsf(" ");delay_ms(500);
            }
        }
    }
}
```

۳-۱-۲- کاراکترهای تعریف شده:

کاراکترهایی که از قبل برای LCD تعریف شده‌اند در جدول (۳-۳) آورده شده‌اند. در این جدول، هر کاراکتر دارای ٤ بیت در ستون و ٤ بیت در سطر می‌باشد که ٤ بیت ستون، به عنوان ٤ بیت با ارزش و ٤ بیت سطر به عنوان ٤ بیت کم ارزش کد کاراکتر مورد نظر به حساب می‌آیند. به عنوان مثال: کد کاراکتر # برابر 00100011 است که برابر 23H می‌باشد.

جدول (۳-۳): کد کاراکترهای موجود در ROM نمایش‌گر LCD



برای نمایش کاراکتر نظیر 23H می‌توان به ترتیب زیر اقدام نمود.

```
lcd_clear();
sprintf(s,"your character is  :\x23");
lcd_puts(s);
```

۳-۱-۳- نمایش کاراکتر جدید بر روی LCD کاراکتری:

در LCD، هر پیکسل به صورت یک ماتریس ۵×۸ نقطه در نظر گرفته می‌شود. برای ایجاد کاراکتر دلخواه باید مراحل زیر را دنبال کنید:

- تعریف آرایه‌ای ۸ عنصری که هر عنصر آن بیانگر سیاه یا سفید بودن نقطه‌های یک سطر از پیکسل باشد. به عنوان مثال: کاراکتر جدید "آ" فارسی را می‌توان به صورت جدول (۳-٤) در یک پیکسل ۵×۸ نقطه تعریف نمود.

جدول (۳-٤) نحوه تعریف کاراکتر "آ"

				■	0b0000001
■	■	■	■	■	0b0011111
■		■			0b0010100
		■			0b0000100
		■			0b0000100
		■			0b0000100
		■			0b0000100
		■			0b0000100
		■			0b0000100

```
flash unsigned char char_make[8]={
0b0000001,
0b0011111,
0b0010100,
0b0000100,
0b0000100,
0b0000100,
0b0000100,
0b0000100};
```

- با استفاده از تابع define_char() اطلاعات مربوط به کاراکتر تعریف شده جدید در آدرس char_code حافظه LCD ذخیره می‌شود و برای دسترسی به این کاراکتر جدید، کافی است با استفاده از تابع putchar (char_code) آن کاراکتر فراخوانی شود که این تابع در ادامه آمده است.

```
void define_char(flash unsigned char *pc,unsigned char_code)
{unsigned i,a;
a=(char_code<<3) | 0x40;
for (i=0; i<8; i++) lcd_write_byte(a++,*pc++);}
```

در تابع فوق اطلاعات کاراکترهای جدید به ترتیب در آدرس 40H تا 47H و کاراکتر بعدی در 48H تا 4FH و نظایر آن نوشته می‌شوند.

پروژه هفتم: فارسی نویسی بر روی LCD کاراکتری

✍ برنامه‌ای بنویسید که بر روی LCD ابتدا هشت حرف اول الفبای فارسی را نمایش دهد. سپس LCD پاک شود و در سطر دوم آن هشت حرف دوم الفبای فارسی چاپ شوند؟

☑ **حل: استفاده از نرم افزار LCD CHAR به منظور تولید کد کاراکتر دلخواه**

در نرم افزار LCD CHAR که یک نسخه آن در CD همراه کتاب وجود دارد، می‌توان با انتخاب پیکسل‌های یک کاراکتر به صورت شکل (۳-٦)، کاراکتر جدید را تولید نمود. بعد از ساخت کد مورد نظر باید دکمه "کپی شدن کد " را فشار دهید و سپس در محیط برنامه آن را Paste کنید. هنگامی که کد را در محیط برنامه paste می کنید علامت "؟" را باید تبدیل به یک عدد دلخواه انتخاب کنید.

شکل(۳-٦): نرم افزار LCD CHAR برای تولید کاراکتر جدید

➢ **توجه:** با استفاده از تابع define_char() به کاراکتر تعریف شده، یک کد اختصاص داده می‌شود که از این کد برای نمایش کاراکتر جدید بروی LCD توسط تابع lcd_putchar() استفاده می‌شود. دقت شود که در هر لحظه LCD قادر به نمایش حداکثر هشت کاراکتر است.

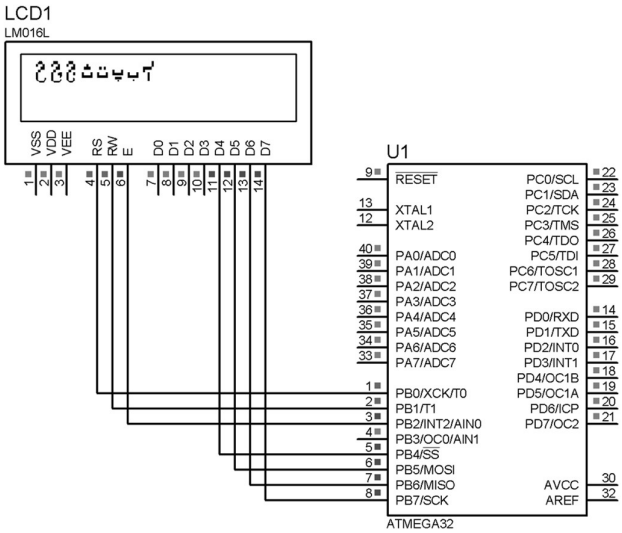
در ادامه، برنامه‌ای که با استفاده از تابع فوق کاراکتر تعریف شده جدید را روی LCD کاراکتری نمایش می‌دهد، ارائه می‌گردد:

✓ **برنامه:**

```
#include <mega32.h>
#include <delay.h>
#asm
```

```
  .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
char i;
flash unsigned char char0[8] = { 0x1, 0x1, 0x1E, 0x10, 0x4, 0x4, 0x4,0x4 }; //آ
flash unsigned char char1[8] = { 0x0, 0x0, 0x0, 0x0, 0x11, 0x1F, 0x0,0x4 }; //ب
flash unsigned char char2[8] = { 0x0, 0x0, 0x0, 0x11, 0x1F, 0x0, 0xE,0x4 }; //پ
flash unsigned char char3[8] = { 0x0, 0x0, 0xA, 0x0, 0x11, 0x1F, 0x0,0x0 }; //ت
flash unsigned char char4[8] = { 0x0, 0x4, 0xE, 0x0, 0x11, 0x1F, 0x0,0x0 }; //ث
flash unsigned char char5[8] = { 0xC, 0x12, 0x6, 0x8, 0x12, 0x10, 0x12,0xC }; //ج
flash unsigned char char6[8] = { 0xC, 0x12, 0x6, 0x8, 0x13, 0x11, 0x10,0xD }; //چ
flash unsigned char char7[8] = { 0xC, 0x12, 0x6, 0x8, 0x10, 0x10, 0x11,0xD }; //ح
flash unsigned char char8[8] = { 0x8, 0x0, 0xC, 0x16, 0x8, 0x10, 0x11,0xD }; //خ
flash unsigned char char9[8] = { 0x0, 0x0, 0x0, 0x4, 0x2, 0x1, 0x1,0xD }; //د
flash unsigned char char10[8] = { 0x0, 0x8, 0x0, 0x4, 0x2, 0x1, 0x1,0xD };//ذ
flash unsigned char char11[8] = { 0x0, 0x0, 0x0, 0x2, 0x2, 0x2, 0x4,0x8 }; //ر
flash unsigned char char12[8] = { 0x2, 0x0, 0x0, 0x2, 0x2, 0x2, 0x4,0x8 };//ز
flash unsigned char char13[8] = { 0x2, 0x6, 0x0, 0x2, 0x2, 0x2, 0x4,0x8 };//ژ
flash unsigned char char14[8] = { 0x0, 0x0, 0xA, 0xF, 0x8, 0x8, 0x18,0x0 };//س
flash unsigned char char15[8] = { 0x2, 0x7, 0x0, 0xA, 0xF, 0x8, 0x18,0x0 };//ش,...
void define_char(flash unsigned char *pc,unsigned char_code)
{unsigned i,a;
a=(char_code<<3) | 0x40;
for (i=0; i<8; i++) lcd_write_byte(a++,*pc++);
}
void main(void)
PORTB=0x00; DDRB=0x00; ACSR=0x80;
SFIOR=0x00; lcd_init(16);
while (1)
{define_char(char0,0); define_char(char1,1);
define_char(char2,2); define_char(char3,3);
define_char(char4,4);define_char(char5,5);
define_char(char6,6); define_char(char7,7);
lcd_clear(); lcd_gotoxy(0,0);
     for (i=8;i>0;i--)
     {lcd_putchar(i-1);delay_ms(1000);}
       define_char(char8,0); define_char(char9,1);
```

```
define_char(char10,2);define_char(char11,3);
define_char(char12,4);define_char(char13,5);
define_char(char14,6);define_char(char15,7);
lcd_clear();lcd_gotoxy(0,1);
    for (i=0;i<9;i++)  {lcd_putchar(i);delay_ms(1000);}
   };}
```



شکل(۳-۷): شماتیک مدار LCD

پروژه هشتم: منو نویسی در LCD

برنامه یک ساعت دیجیتالی را بنویسید که درسطر اول ساعت، دقیقه و ثانیه و در سطر دوم آن تاریخ را نشان دهد؟

☑ حل: هنگامی که یک پالس پایین رونده از طریق کلید تنظیم K_1 به int0 وارد می شود، LCD وارد منویی می‌شود که در سطر اول "-- > Year " و در سطر دوم " Month " را نشان می‌دهد (علامت -- > نشان دهنده این است که در حال حاضر تنظیمات بر روی Year انجام می گیرد) پس اگر LCD شما ۱۶×۲ کاراکتری باشد، با فشردن int0 باید به شکل (۳-۸) تبدیل شود.

```
Year < --
Month
```

شکل (۳-۸): منوی سال و ماه با فشردن کلید K_1

علامت " -- > " باید توسط کلیدهای K_3 و K_4 که به ترتیب به: PIND.5 و PIND.6 متصل هستند، پایین یا بالا برود. فرض کنید: LCD شما تنها قادر به تنظیم سال و ماه می‌باشد. پس اگر کلید K_2 یک بار زده شود باید علامت " -- > " به Month اشاره کند. در این منو باید گزینه exit نیز وجود داشته باشد تا پس از تنظیمات لازم بر روی هر یک از دو گزینه " Year " یا "Month " بتواند از منو خارج و به صفحه‌ی اصلی که نشان دهنده ساعت و تاریخ است، بازگردد. پس اگر دو مرتبه کلید K_2 فشرده شود، باید LCD به شکل (۳-۹) تبدیل شود.

```
Month

< --
```

شکل (۳-۹): منوی ماه و خروج با فشردن کلید K_2

پس از این که گزینه مورد نظر انتخاب شد، باید بتوان توسط کلید متصل به PIND.4 ، K_2 آن را انتخاب کرد. پس از آن که گزینه مورد نظر انتخاب شد، LCD وارد منوی جدیدی می‌شود که در آن منو می‌توان به تنظیم گزینه انتخاب شده پرداخت. فرض کنید که شما گزینه Year را انتخاب کردید حال در منوی جدید باید بتوان با زدن کلیدهای متصل به K_3 و K_4 به ترتیب مقدار Year را افزایش یا کاهش داد و اگر کلید K_2 فشرده شد دوباره به منوی قبلی (نه صفحه‌ی اصلی) باز گردد. در ضمن هنگامی که وارد منوی تنظیمات می‌شوید ساعت باید به کار عادی خود ادامه دهد، یعنی: مثلاً اگر کاربر درحال تنظیم ماه است همزمان ساعت نیز کار کند و متوقف نشود.

➢ **نکته‌های ضروری در برنامه:**

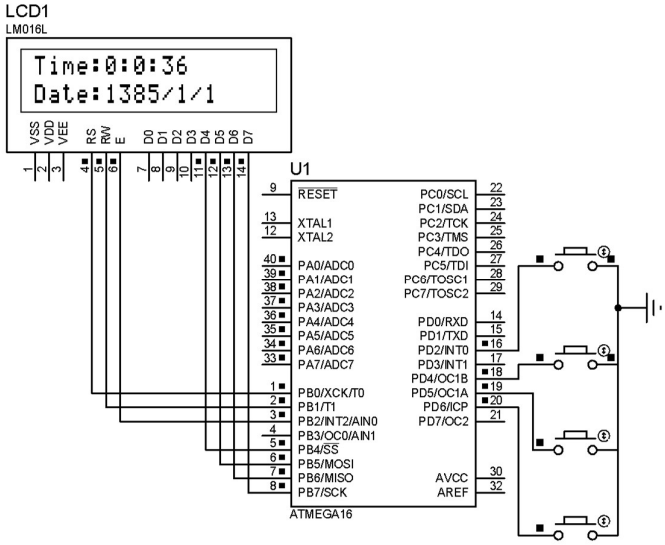
- ✓ از دستور اسمبلی ("sei") #asm استفاده شده است، تا هنگام اجرای وقفه‌ی int0 وقفه ovf نیز کار کند و در زمان انجام تنظیمات مورد نظر، ساعت نیز به طور هم‌زمان کار کند.
- ✓ متغیر i، هنگام ورود به وقفه، ۱ می‌شود، تا هنگام تنظیمات ساعت، LCD مقادیر ثانیه، دقیقه و نظایر آن را، مانند حالت عادی خود نمایش ندهد.
- ✓ توسط متغیر ord تعیین می‌شود که روی LCD چه چیزی نمایش داده شود. (عدد موجود در این متغیر به برنامه‌نویس کمک می‌کند تا متوجه شود که علامت -- > در مقابل کدام گزینه قرار دارد).
- ✓ نوشتن تنظیمات مربوط به Day (ord=1) و بقیه تنظیمات مانند تنظیم ثانیه، دقیقه و ساعت به خواننده واگذار می‌شود.

✓ برنامه:

```
#include <mega16.h>
#asm
 .equ __LCD_port=0x18 ;PORTB
#endasm
#include <lcd.h>
#include<stdio.h>
#include<delay.h>
char aray[16],i=0,araya[32];
int secound=0,minute=0,d=1,m=1,y=1385,hour=0;
interrupt [EXT_INT0] void ext_int0_isr(void)
{int select =0,ord=0,exit=0;
#asm("sei")
i=1;lcd_clear();lcd_gotoxy(0,0);
lcd_putsf("up down select");
delay_ms(500);
while(exit == 0)
  {   if(ord == 0)
            {
            lcd_clear();
            lcd_putsf("year <-- \nmonth");
            }
      else if(ord == 1)
            {
            lcd_clear();
            lcd_putsf("month <-- \nexit");
            }
      else
            {
            lcd_clear();
            lcd_putsf("exit <-- \nyear ");
            }
    while(PIND.4 && PIND.5 &&PIND.6);

        if(!PIND.5){ord=ord+1;  if(ord==3)ord=0;}
      else if(!PIND.6) {ord=ord-1; if(ord==-1)ord=2;}
      else if(!PIND.4) select = 1;

    if(select == 1)
    {         if(ord == 0)
                  {sprintf(aray,"year = %i",y);  lcd_clear(); lcd_puts(aray); }
```

```
            if(ord == 1)
                {sprintf(aray,"month = %i",m);  lcd_clear();
lcd_puts(aray); }
    }
            delay_ms(500);
   while(select ==1)
          {
          while(PIND.4 && PIND.5 && PIND.6);
          if( ord == 1)
                {   if(!PIND.5)  m=m+1;
                    else if(!PIND.6) m=m-1;
                    else  select = 0;
                  sprintf(aray,"month = %i",m);
                  lcd_clear();
                  lcd_puts(aray);
                }

          else if(ord==0)
               { if(!PIND.5) y=y+1;
                 else if(!PIND.6)y=y-1;
                 else select = 0;
               sprintf(aray,"year = %i",y);
               lcd_clear();
               lcd_puts(aray);
               }
             else  {exit = 1;select =0;}
          delay_ms(1000);
          }
   }
i=0;
}
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{ secound = secound + 1;
  if(secound == 60) {secound = 0;minute = minute + 1;}
  if (minute==60){minute=0;hour=hour+1;}
  if(i==0)
{sprintf(araya,"Time:%i:%i:%i\nDate:%i/%i/%i",hour,minute,secound,y,
m,d);
      lcd_clear();
      lcd_puts(araya);}
 TCNT1H=0x85; TCNT1L=0xED;
```

```
}
void main(void)
{
PORTA=0x00;DDRA=0xFF;
PORTD=0b01110100;DDRD=0x00;
TCCR1A=0x00;TCCR1B=0x04;TCNT1H=0x85;
TCNT1L=0xED;ASSR=0x00;GICR|=0x40;
MCUCR=0x02;MCUCSR=0x00;GIFR=0x40;
TIMSK=0x04;ACSR=0x80;SFIOR=0x00;
lcd_init(16);
#asm("sei")
while (1);}
```



شکل (۳-۱۰) نحوه اتصال LCD و کلیدها به میکرو

۳-۲- آشنایی و برنامه‌نویسی با LCD های گرافیکی

LCD های گرافیکی در اندازه‌های 64×64، 64×128، 128× 240 و نظایر آن‌ها توسط شرکت‌های مختلف و به صورت رنگی و تک رنگ عرضه شده‌اند. در ادامه بحث یک LCD تک رنگ با اندازهٔ 64×128 (128 پیکسل در سطر و 64 پیکسل در ستون که هر واحد تصویر را پیکسل گویند) معرفی شده است.

۳-۲-۱- معرفی پایه‌های LCD – 128 GO64A:

این LCD، دارای ۲۰ پایه است که به ترتیب: ٤ پایه برای تغذیه، ۵ پایه به منظور کنترل و ۸ پایه برای تبادل داده در نظر گرفته شده که به صورت زیر هستند:

پایه ۱ (VSS) : این پایه زمین شود.

پایه ۲ (VDD) : توسط این پایه تغذیهٔ LCD تامین می‌شود که معمولاً این تغذیه، ۵ ولت می‌باشد.

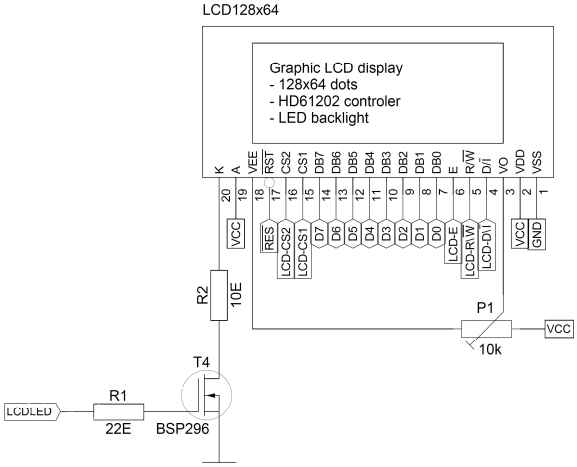
پایه ۳ (Vo) : با تغییر ولتاژ روی این پایه شفافیت LCD تغییر می‌کند.

پایه ٤ (RS یا D/I) : چنانچه این پایه صفر باشد: اطلاعات ارسالی دستور العمل، چنانچه یک باشد: اطلاعات ارسالی داده در نظر گرفته می‌شود.

پایه ۵ (R/W) : چنانچه این پایه صفر باشد می‌توان اطلاعات را روی LCD نوشت و چنانچه یک باشد اطلاعات ارسالی داده در نظر گرفته می‌شود.

پایهٔ ٦(E) : هنگام ارسال داده (یا دستورالعمل) باید یک پالس پایین رونده به این پایه اعمال شود تا LCD داده‌های موجود در پایه‌های داده را لچ کند.

پاپه ١٤-١٧ (DB0-DB7) : از این پایه‌ها برای ارسال داده (Data) به LCD یا خواندن محتوای رجیستر داخلی LCD استفاده می‌شود.



شکل(۳-۱۱) سخت‌افزار LCD – 128 GO64A

پایه ١٦-١٥ (CS1-CS2) : از این پایه برای انتخاب نیم صفحه‌ها استفاده می‌شود چنانچه CS1=0 باشد نیم صفحهٔ چپ و چنانچه CS2=0 باشد نیم صفحهٔ راست انتخاب می‌شود.

پایه ۱۷ (RES) : پایهٔ ریست LCD می‌باشد.
پایه ۱۸ (VOUT یا VEE) : توسط این پایه ولتاژ منفی مورد نیاز برای پتانسیومتر تنظیم پایهٔ V_o تامین می‌شود.
باید ۱۹ (A) : این پایه آند LED داخلی چراغ زمینه LCD می‌باشد که باید به ولتاژ +4.2V متصل باشد.
پایه ۲۰ (K): این پایه کاتد LED داخلی چراغ زمینهٔ LCD می‌باشد که باید به زمین متصل شود.
LCD مورد بحث در این بخش از دو نیم صفحه تشکیل شده است. هریک از دو نیم صفحه‌ی فوق توسط دو تراشه KS0108B کنترل می‌شوند. برای هماهنگی این دو با هم تراشهٔ دیگری به نام KS0108 در نظر گرفته شده است. هر یک از این نیم صفحه‌ها به ۸ قسمت مساوی سطری (Page) و ٦٤ قسمت مساوی ستونی تقسیم شده‌اند.

جدول (۳-۵) نحوه تقسیم‌بندی LCD گرافیکی

CS1 = 0 CS2=1	Cs1= 1 CS2= 0
صفحهٔ ۰	صفحهٔ ۰
صفحه ۱	صفحهٔ ۱
صفحهٔ ۲	صفحهٔ ۲
صفحهٔ ۳	صفحهٔ ۳
صفحهٔ ٤	صفحهٔ ٤
صفحهٔ ۵	صفحهٔ ۵
صفحهٔ ٦	صفحهٔ ٦
صفحهٔ ۷	صفحهٔ ۷

الگوریتم ارسال و دریافت داده به این ترتیب است که ابتدا باید توسط دو پایهٔ CS1 و CS2 نیم صفحه‌ی مورد نظر انتخاب شود و سپس آدرس سطر (صفحه) و ستون پیکسل مورد نظر ارسال شود و در پایان دادهٔ مورد نظر را برای LCD ارسال و چنانچه قصد خواندن از LCD را داشته باشیم دادهٔ مذکور را از حافظهٔ RAM آن دریافت می کنیم.

روش خواندن از LCD :

۱ـ یک کردن پایه‌های R/W, D /I

۲ـ ارسال یک پالس پایین رونده

۳ـ خواندن داده‌ای که بر روی DB0 تا DB7 قرار می‌گیرد.

مراحل نوشتن بر روی LCD :

۱ـ قرار دادن داده‌ی مورد نظر روی پایه‌های DB0 تا DB7

۲ـ یک کردن پایه D/I و صفر کردن پایهٔ R/W

۳ـ اعمال پالس پایین رونده به پایه E

پس از ۳ گام فوق دادهٔ مذبور وارد حافظهٔ RAM نمایش‌گر می‌شود.

روش خواندن وضعیت LCD :

۱ـ صفر کردن پایه D/I و یک کردن پایه R/W

۲ـ ارسال یک پالس پایین رونده به پایه E

۳ـ خواندن پایه‌های DB4, DB5, DB7

٤ـ بررسی وضعیت LCD طبق جدول (۳-٦)

جدول (۳-٦) وضعیت های LCD گرافیکی

پایه	0	۱
DB7	Normal	Busy
DB5	Display off	Display on
DB4	Reset off	Reset on

چگونگی تعیین آدرس ستون (Y):

۱ـ فعال کردن هر یک از دونیم صفحه توسط پایه‌های CS1 و CS2

۲ـ صفر کردن پایه D/I و صفرکردن پایهٔ R/W

۳ـ قرار دادن آدرس ستون مورد نظر که آدرس ستون صفرم 0x40 و آدرس ستون شصت و سوم 0x7f می‌باشد.

٤ـ ارسال پالس پایین رونده به پایهٔ E

➢ **نکته:** چنانچه ستون ابتدایی مشخص شود شماره ستون برای ارسال بایت های بعدی توسط یک شمارنده به صورت خودکار افزایش داده می‌شود.

جدول(۳-۷): آدرس Y

0 × 40	CS1=0 CS2=1	0 × 7F 0× 40	CS1 = 1 SC2=0	0 × 7F

چگونگی تعیین آدرس سطر (X):

۱ـ فعال کردن هر یک از دو نیم صفحه توسط پایه‌های CS1 و CS2

۲ـ صفر کردن پایه D/I و صفر کردن پایهٔ R/W

۳ـ قرار دادن آدرس نیم صفحه مورد نظر که آدرس نیم صفحه صفرم 0xBF و آدرس نیم صفحهٔ هفتم 0xBF می‌باشد.

جدول(۳-۸): آدرس X

0x B8	0x B8
0x B9	0x B9
0x BA	0x BA
0x BB	0x BB
0x BC	0x BC
0x BD	0x BD
0x BE	0x BE
0x BF	0x BF

روشن/ خاموش کردن LCD

۱ـ فعال کردن هر یک از دو نیم صفحه توسط پایه‌های CS1 و CS2

۲ـ صفر کردن D/I و R/W

۳ـ تنظیم پایه‌های D7 – D0 به صورت 00111110

٤ـ ارسال پالس پایین رونده به پایهٔ E برای ارسال یا دریافت دادهٔ فوق

۵ـ تنظیم دوبارهٔ پایه‌های D7-D0 به صورت 00111111

٦ـ ارسال پالس پایین رونده به پایهٔ E برای ارسال بایت دادهٔ فوق

پس از برداشتن٦ گام فوق LCD روشن خواهد شد.

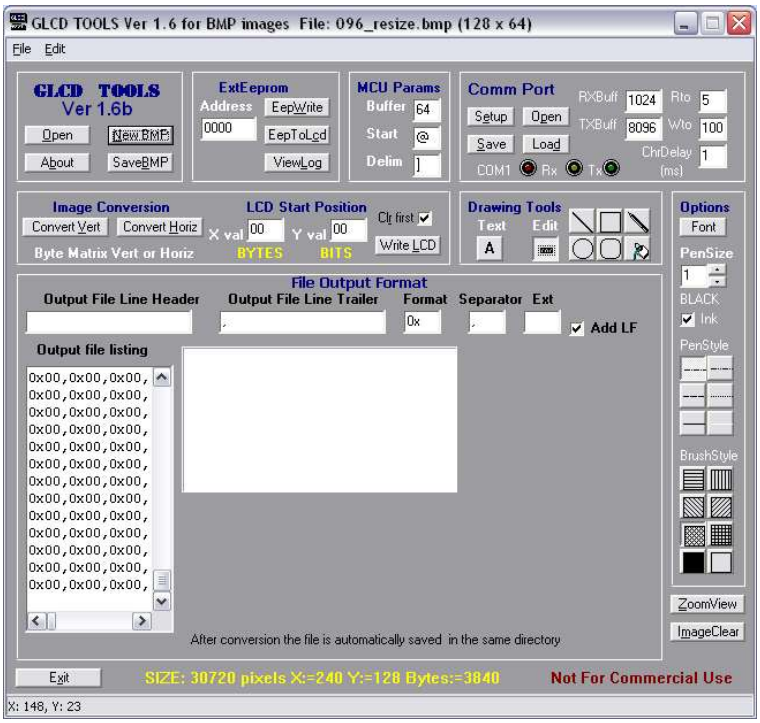
۳-۲-۲- شبیه سازی LCD گرافیکی با استفاده از شبیه ساز Dincer's Graphic LCD

نرم افزار Dincer برای آموزش روش خواندن و نوشتن روی LCD قابل استفاده است که نسخه Dincer's Graphic آن برای LCD های گرافیکی طراحی شده است و در پوشه LCD\djgfxLCDsim1 در CD همراه کتاب آورده شده است.

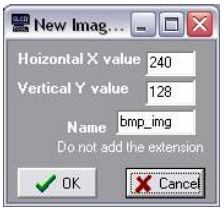
۳-۲-۳- نرم افزار مبدل فرمت عکس

نرم افزار GLCD Tools تحت سیستم عامل ویندوز است و به صورت یک فایل اجرایی بوده، یعنی نیازی به نصب ندارد. با اجرای این نرم‌افزار، محیطی مطابق شکل (۳-۱۲) ظاهر می‌شود و با توجه به مراحل زیر کد مطلوب ایجاد می‌شود.

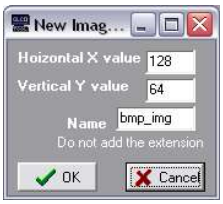
۱- بر روی گزینه New BMP کلیک نماید یا کلید N از صفحه کلید را فشار دهید و از پنجره ظاهر شده، سایز عکس مورد نظر را متناسب با LCD استفاده شده تنظیم نمائید. **به عنوان مثال:** برای LCD KS108 دارای ۱۲۸ ستون و ٦٤ سطر به صورت شکل (۳-۱۳) و برای LCD Toshoiba دارای ۲٤۰ ستون و ۱۲۸ سطر به صورت شکل (۳-۱٤) تنظیمات انجام می‌شوند.



شکل(۳-۱۲): نرم افزار GLCD Tools



شکل(۳-۱۳): تنظیمات LCD KS108



شکل(۳-۱٤): تنظیمات LCD Toshiba

۲- در محیط ایجاد شده مطابق شکل (۳-۱۲) نقاشی کشیده و متن‌های مورد نظر را اضافه کنید.

۳- قسمت File Output Format دارای شش گزینه است که می‌باید به منظور سازگار بودن کدهای تولید شده با نرم‌افزار CodeVision مطابق شکل (۳-۱۵) تنظیم شوند. دقت شود، هرگونه تغییر در ساختار زیر منجر به عدم نمایش تصویر بر روی LCD می‌شود.

شکل(۳-۱۵): تنظیمات File Output Format

٤- در صورتی که از LCD Ks108 128×64 استفاده می‌کنید: بر روی گزینه Convert Vert کلیک نمائید و در صورتی که از LCD Toshiba 240×128 استفاده می‌کنید: بر روی گزینه Convert Horiz کلیک نمائید تا کد مطلوب تولید شود.

٥- قسمت Output File Listing تمام کد ایجاد شده را کپی کنید.

٦- کد کپی شده را در محیط CodeVision قرار دهید و عبارت نوشته شده را که در اول کد ایجاد شده، قرار گرفته است را از اول آن حذف کنید (آدرس عکس به عنوان مثال: bmp_img.bmp (128 x 64);). همچنین آخرین کاما را از کد ایجاد شده حذف کنید.

پروژه نهم: نمایش تصویر بر روی LCD گرافیکی Ks108 128×64

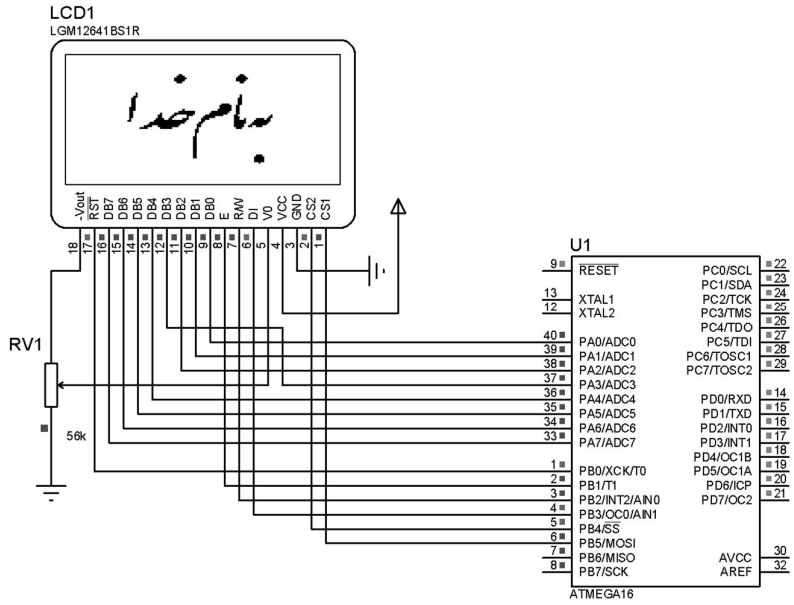
برنامه‌ای بنویسید که یک عکس بر روی LCD Ks108 128×64 نشان دهد؟

✓ برنامه

```
#include<mega32.h>
#include<delay.h>
#define LCD_PORT  PORTA
#define LCD_RST PORTB.0
#define LCD_E PORTB.1
#define LCD_RW PORTB.2
#define LCD_RS PORTB.3
#define LCD_CS2 PORTB.4
#define LCD_CS1 PORTB.5
// -------------------------------------------------------
// GLCD Picture name: p1.bmp
// GLCD Model: KS0108 128x64
// -------------------------------------------------------
flash char p1_bmp[] = {    //  کدهای ایجاد شده توسط نرم افزار به منظور نمایش عکس
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,128,128,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
```

```
  ,… };
// ----------------------------------------------------
void write_lcd(char columns,char page,char data);
void LatchLCD();
//*****************************************//
void main()
{
  unsigned int i,x,y;
  PORTA=0x00;
  DDRA=0xFF;
  PORTB=0;
  DDRB=0xFF;
  LCD_E=0;
  LCD_RW=0;
  LCD_RST=1;
  //******************************
  while(1){
  for(i=0;i<1024;i++){
          if(x>=128){x=0;y++;}
          write_lcd(x,y,p1_bmp[i]);
          x++;
  }
  delay_ms(2000);
  }
}
//---------------------------
void write_lcd(char columns,char page,char data)
{
          if(columns >= 64){
                    LCD_CS1=0;
                    LCD_CS2=1;
          }else{
                    LCD_CS1=1;
                    LCD_CS2=0;
          }
          LCD_RS=0;
          LCD_RW=0;
          LCD_PORT=0xB8 | page;
          LatchLCD();
          LCD_RS=0;
          LCD_RW=0;
```

```
        LCD_PORT=0x40 | columns;
        LatchLCD();
        LCD_RS=1;
        LCD_RW=0;
        LCD_PORT=data;
        LatchLCD();
}
//------------------------
void LatchLCD()
{
        delay_us(5);
        LCD_E=1;
        delay_us(5);
        LCD_E=0;
}
//------------------------
```



شکل(۳-۱٦): نحوه سیم‌بندی LCD Ks108 در محیط شبیه‌ساز Proteus

- دقت شود که پایه‌های LCD در شکل بالا، به منظور شبیه سازی در محیط نرم‌افزار Proteus صحیح است ولی در عمل پایه‌های LCD مطابق شکل (۲-۱۱) می‌باشند.

LCD گرافیکی مدل Toshiba : این مدل از LCD های گرافیکی از تراشـــه راه انداز T6963C اســتفاده می‌کنند. T6963C یک تراشــه کنترلی راه انداز LCD گرافیکی برای اندازه‌های کوچک و متوسط است و توانایی تولید سیگنال‌های کنترلی به منظور راه اندازی LCD گرافیکی را دارد. این تراشه از یک حافظه ROM داخلی با ظرفیت ۱۲۸ کلمه جهت نمایش کاراکترهای استاندارد (CG-ROM) استفاده می‌کند. همچنین می‌تواند یک RAM نمایش تصـــویر (VRAM) خارجی تا ظرفیت 64KB را کنترل نماید. اطلاعاتی که قرار است بر روی صفحه LCD نمایش داده شوند: ابتدا بر روی حافظه VRAM قرار گرفته، سپس توسط تراشه راه انداز روی صفحه، نمایش داده می‌شوند. اطلاعات موجود در VRAM می‌تواند فرمت‌های مختلفی همچون، متن، گرافیک و کاراکترهای خارجی (CG-RAM) باشـــد (منظور از کاراکترهای خارجی، هر کاراکتری به غیر از کاراکترهای استاندارد موجود در حافظه ROM داخلی (CG-ROM) است که توسط کاربر ایجاد می شود). از خصوصیات تراشه T6963C می توان به موارد زیر اشاره کرد:

- هشت پایه موازی جهت ارتباط با میکروکنترلر جانبی
- قابلیت انتخاب فرمت نمایش
- قابلیت انتخاب طول و عرض فونت (۸×۵، ۸×٦ و ۸×۷)
- ۱۲۸ کاراکتر داخلی (CG-ROM)
- قابلیت نمایش فرمت‌های متنی، گرافیک و کاراکتر های دلخواه (CG-RAM) قابلیت ترکیب و ادغام فرمت‌های متنی و گرافیکی ساخته شده بر اساس تکنولوژی

پروژه دهم: نمایش تصویر بر روی LCD گرافیکی Toshiba 240×128

برنامه‌ای بنویسید که یک عکس بر روی LCD Toshiba 240×64 نشان دهد؟

✓ برنامه

```
#include <mega32.h>
#include <delay.h>
#define  LCD_WR    PORTB.5
#define  LCD_RD    PORTB.4
#define  LCD_CE    PORTB.3
#define  LCD_CD    PORTB.2
#define  LCD_RST   PORTB.1
```

```
#define  LCD_FS1   PORTB.0
#define EightbyEight            0
#define Busy1Busy2              0x03
#define DAWRDY                  0x08
#define CursorPointerSet        0x21
#define OffsetRegisterSet       0x22
#define AddressPointerSet       0x24
#define TextHomeAddress         0x40
#define TextAreaSet      0x41
#define GraphicHomeAddress      0x042
#define GraphicAreaSet          0x043
#define CGROMMode               0x80
#define CGRAMMode               0x88
#define ORMode                  0x80
#define EXORMode                0x81
#define ANDMode                 0x83
#define Textonly                0x84
#define GraphicsOff             0x90
#define GraphicsOn              0x98
#define TextOff                 0x90
#define TextOn                  0x94
#define CursorOff               0x90
#define CursorOn                0x92
#define CursorBlinkOff          0x90
#define CursorBlinkOn           0x91
#define CursorPattern           0xA0
#define BottomLine              0x0
#define DataAutoWrite           0xB0
#define DataAutoRead            0xB1
#define AutoReset               0xB2
#define DataRdWr                0xC0
#define BitReset                0xF0
#define BitSet                  0xF8
#define GraphBase               0x0000
#define TextBase                0x1000
#define Column          240
#define Row             128
flash char picture1[]={ // کدهای ایجاد شده توسط نرم افزار به منظور نمایش عکس
// (240*128)
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,0x00,…};
char FontSize ;
unsigned int charsPerRow ;
void lcd_init();
void StatBusy12(void);
void StatDAWRDY(void);
void DataOut(char b);
void CommandOut(char b);
void lcd_clear_graphic();
void lcd_clear_text();
void lcd_Graphic1();
void lcd_Graphic2();
void SetAutoMode();
void AutoOut(char B);
void ResetAutoMode();
void lcd_cursorXY(char X, char Y);
void DataOut2(int b);
void AutoZero(unsigned int Terminate);
//--------------------------------------
void main()
{
  lcd_init();
  lcd_Graphic1();
  delay_ms(2000);
  while(1);
}
//-----------------------------------------
void lcd_Graphic1()
{
        int i;
        lcd_clear_graphic();
  CommandOut(GraphicsOn | TextOff | CursorOff);
  lcd_clear_graphic();
```

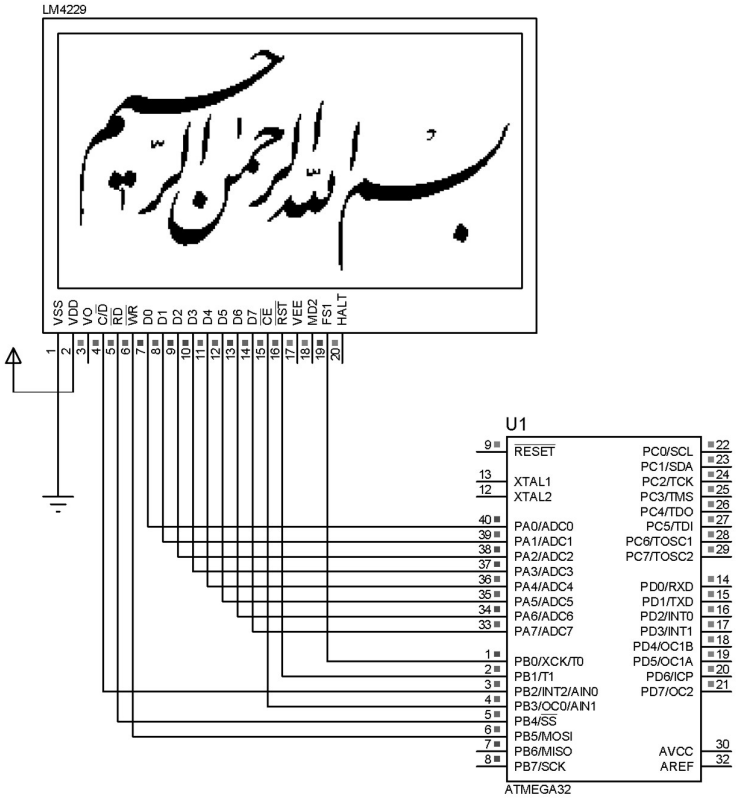
```
  DataOut(0);
  CommandOut(AddressPointerSet);
  SetAutoMode();
  for(i=0;i<(Row*Column)/8 ;i++)
    AutoOut(picture1[i]);
  ResetAutoMode();
}
//-----------------------------------------------------
void lcd_Graphic2()
{
          int i;
          lcd_clear_graphic();
  CommandOut(GraphicsOn | TextOff | CursorOff);
  lcd_clear_graphic();
  DataOut(0);
  CommandOut(AddressPointerSet);
  SetAutoMode();
  for(i=0;i<(Row*Column)/8 ;i++)
    AutoOut(picture2[i]);
  ResetAutoMode();
}

//----------------------------------------

void lcd_init()
{
  PORTB = 0x1F;
  DDRB = 0xff;
  PORTA = 0;
  DDRA = 0xff;
  LCD_CE = 1;
  LCD_RD = 1;
  LCD_WR = 1;
  LCD_CD = 1;
  LCD_RST = 0;
  delay_us(1);
  LCD_RST = 1;
  delay_us(1);
  LCD_FS1 = EightbyEight ;
  if ( LCD_FS1 == EightbyEight)
   {
```

```
    FontSize = 8 ;
    charsPerRow = Column / FontSize ;
   }
  else
   {
    FontSize = 6 ;
    charsPerRow = Column / FontSize ;
   }
  CommandOut(GraphicsOff | TextOff  | CursorOff );
  DataOut2(GraphBase);
  CommandOut(GraphicHomeAddress);
  DataOut2(charsPerRow);
  CommandOut(GraphicAreaSet);
  DataOut2(TextBase);
  CommandOut(TextHomeAddress);
  DataOut2(charsPerRow);
  CommandOut(TextAreaSet);
  CommandOut(EXORMode | CGROMMode);
  CommandOut(CursorPattern | 7);
  lcd_clear_graphic();
  lcd_clear_text();
  lcd_cursorXY(0,0);
}
//--------------------------------
void lcd_clear_graphic()
{
    DataOut2(GraphBase);
    CommandOut(AddressPointerSet);
    AutoZero( charsPerRow * Row  );
}

//----------------------------------
void lcd_clear_text()
{
  DataOut2(TextBase);
  CommandOut(AddressPointerSet);
  AutoZero( charsPerRow * ( Row /8 ));
}
//---------------------------------

void lcd_cursorXY(char X, char Y)
```

```
{
   DataOut(X);
   DataOut(Y);
   CommandOut(CursorPointerSet);
}
//--------------------------------------
void AutoZero( unsigned int Terminate)
{
  SetAutoMode();
  while(Terminate--)
    AutoOut(0);
  ResetAutoMode() ;
}

//--------------------------------------
void StatBusy12()
{
   char lcd_status;
   DDRA = 0x00;  // Input porta
   do
   {
      LCD_CE = 0;
      LCD_RD = 0;
      delay_us(1);
      lcd_status = PINA;
      LCD_CE = 1;
      LCD_RD = 1;
   }
   while ((lcd_status & Busy1Busy2) != Busy1Busy2);
   DDRA = 0xff; // output porta

}
//--------------------------------
void StatDAWRDY()
{
   char lcd_status;
   DDRA = 0x00;
   do
   {
      LCD_CE = 0;
      LCD_RD = 0;
```

```
    delay_us(1);
    lcd_status = PINA;
    LCD_CE = 1;
    LCD_RD = 1;
  }
  while ((lcd_status & DAWRDY) != DAWRDY);
  DDRA = 0xff;

}
//-------------------------------
void DataOut(char b)
{
  StatBusy12();
  LCD_CD = 0;
  PORTA = b;
  LCD_CE = 0;
  LCD_WR = 0;
  LCD_CE = 1;
  LCD_WR = 1;
  LCD_CD = 1;
}
//-------------------------------
void DataOut2(int data)
{
   DataOut(data & 0xFF);
   DataOut(data >> 8);
}
//-------------------------------
void CommandOut(char b)
{
  StatBusy12();
  PORTA = b;
  LCD_CE = 0;
  LCD_WR = 0;
  LCD_CE = 1;
  LCD_WR = 1;
}
//-------------------------------
void SetAutoMode()
{
 CommandOut(DataAutoWrite);
```

```
}
//--------------------------------

void AutoOut(char b)
{
  StatDAWRDY();
  LCD_CD = 0;
  PORTA = b;
  LCD_CE = 0;
  LCD_WR = 0;
  LCD_CE = 1;
  LCD_WR = 1;
  LCD_CD = 1;
}
//---------------------------------
void ResetAutoMode()
{   StatDAWRDY();
  PORTA = AutoReset;
  LCD_CE = 0;
  LCD_WR = 0;
  LCD_CE = 1;
  LCD_WR = 1;}
```



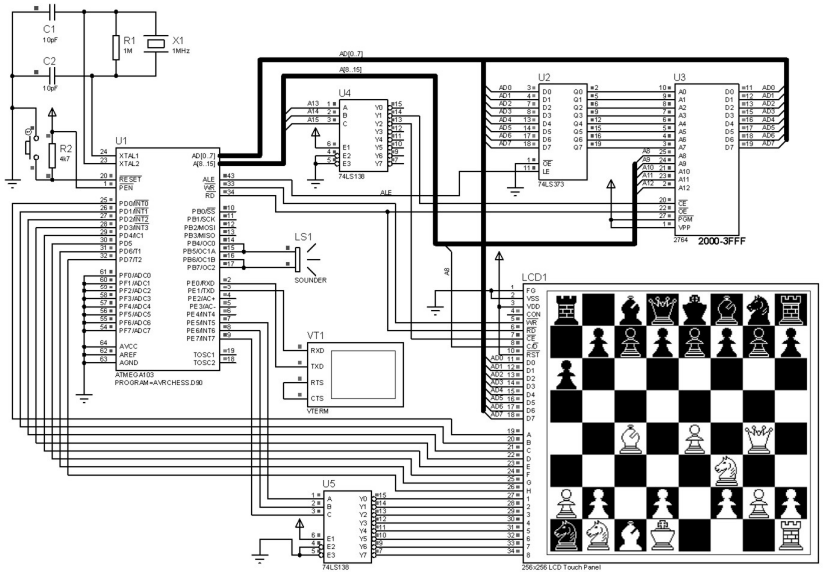
شکل(۳-۱۷): نحوه سیم‌بندی LCD Toshiba در محیط شبیه‌ساز Proteus

- دقت شود که پایه‌های LCD در شکل بالا، به منظور شبیه سازی در محیط نرم‌افزار Proteus صحیح است ولی در عمل پایه‌های LCD مطابق زیر است.

۱- GND، ۲- GND، ۳- VDD = 5volt، ٤- Vo سر وسط پتانسیومتر، ۵- W/R، ٦-R/D، ۷-CS، ۸-RS، ۹-Reset، پایه‌های ۱۰ تا ۱۷ به Data0 تا Data7، ۱۸-FS، ۱۹- یک سر پتانسیومتر ۱۰۰ کیلو اهمی، ۲۰- آند چراغ پس زمینه، ۲۱- کاتد چراغ پس زمینه، ۲۲- بدون اتصال.

مطالب دیگری نیز، به منظور راه اندازی LCD های دیگر شامل TFT موبایل وجود دارد که در فصل‌های بعدی به تفصیل بیان می‌شوند.

بازی و سرگرمی: در شکل (۳–۱۸) نحوه سیم بندی یک LCD 256*256 آورده شده که به **نمایش‌گر تاچ پنل** معروف است و برنامه بازی شطرنج بر روی میکروکنترلر پیاده‌سازی شده است. با مراجعه به CD همراه کتاب، ابتدا برنامه را مشاهده و سپس آن را تحلیل نمائید و با اجرای برنامه در محیط Proteus مطابق شکل (۳–۱۸)، مهارت خود در بازی شطرنج را آزمایش کنید.



شکل(۳–۱۸): نحوه سیم‌بندی Touch Panel LCD در محیط شبیه‌ساز Proteus

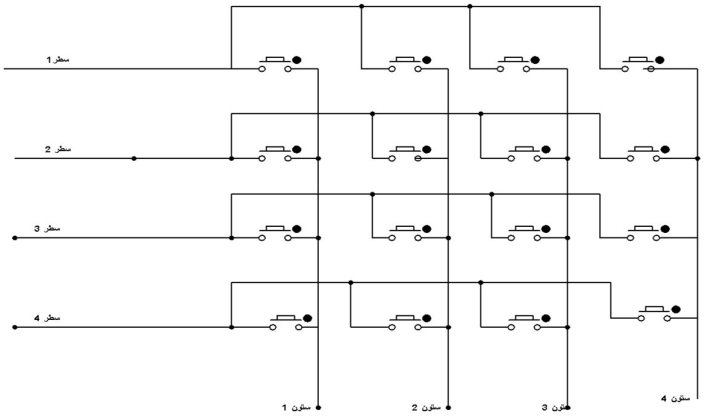
فصل چهارم کاربرد وقفه‌های خارجی

در بیشتر مدارهای میکروکنترلری، پردازنده مرکزی نیازمند دریافت اطلاعات از محیط خارج است. ممکن است این داده‌ها از منبع داده (غیر انسانی) ارسال شوند و یا ممکن است توسط کاربر (انسانی) ارسال شوند. در صورتی‌که، داده‌ها توسط کاربر ارسال شوند، پردازنده مرکزی پس از دریافت آن‌ها عملیات مورد نظر را بر روی آن‌ها انجام می‌دهد. ممکن است داده‌های دریافتی توسط مبدل آنالوگ به دیجیتال دریافت شوند و یا توسط صفحه کلید ماتریسی متصل به یکی از پورت‌های میکروکنترلر انجام شود. شکل (٤-١)، صفحه کلید ماتریسی (یا به اختصار کی‌پد[1])، را نشان می‌دهد که در آن کلیه کلیدهای یک سطر با یک سیم مشترک به هم متصل می‌شوند و کلیه کلیدهای هر ستون نیز با یک سیم مشترک به یکدیگر وصل می‌شوند. بنابراین، یک صفحه کلید ٤×٤ دارای ٨ سیم خروجی خواهد بود. روش‌های زیادی به منظور خواندن داده‌های کی‌پد وجود دارد که در روش اول: با اسکن صفحه کلید به صورت چرخش صفر بر روی سطرها و خواندن ستون‌ها انجام می‌شود. البته در این روش بهتر است از وقفه خارجی میکروکنترلر استفاده شود. روش دوم: با استفاده از تراشه‌های انکدر مانند MM74C922 انجام می‌شود. این انکدرها برای هر کلید فشرده شده یک کد چهار بیتی تولید می‌کند و به این ترتیب کلید فشرده شده شناسایی می‌شود. در روش سوم: خواندن داده‌های کی‌پد با استفاده از مبدل آنالوگ به دیجیتال انجام می‌شود و در روش چهارم: خواندن داده‌ها با استفاده از ثابت زمانی مدار RC صورت می‌پذیرد.

٤-١- اسکن صفحه کلید ماتریسی ٤×٤ با وقفه

سخت‌افزار اتصال صفحه کلید به میکروکنترلر در شکل (٤-٢) ترسیم شده است. در این سخت‌افزار ستون‌های صفحه کلید به چهار بیت کم ارزش پورت C و سطرهای آن به چهار بیت پر ارزش آن پورت متصل شده‌اند. توضیح این روش در پروژه یازدهم آورده شده است.

[1] Keypad



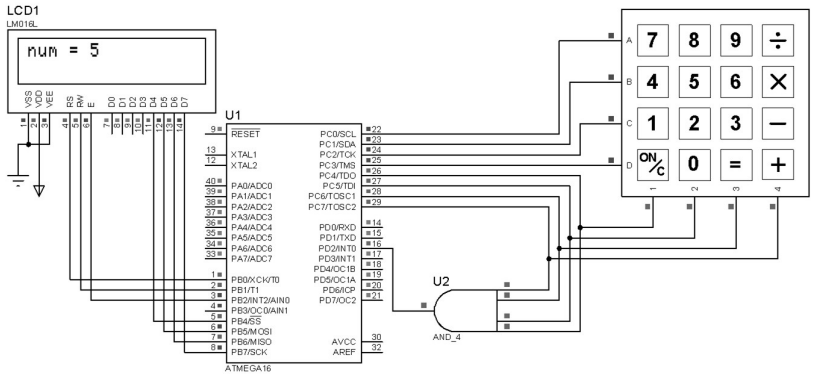
شکل (٤-١): صفحه کلید ماتریسی

پروژه یازدهم: اسکن صفحه کلید ماتریسی ۴×۴

✍ برنامه‌ای بنویسید که صفحه کلید ماتریسی ٤×٤ را اسکن نماید و با فشردن هر کلید از صفحه کلید ماتریسی، عدد متناظر را بر روی LCD کاراکتری نشان دهد؟

☑ **حل:** مطابق شکل (٤-٢)، پورت C میکروکنترلر AVR را به ٤ سطر و چهار ستون صفحه کلید متصل نموده و LCD کاراکتری را به پورت B و چهار ستون صفحه کلید ماتریسی را به گیت AND (گیت AND با چهار ورودی) متصل می‌نمائیم. سپس خروجی این گیت را به پایه int0 میکرو به منظور ایجاد وقفه‌ی خارجی وصل می‌کنیم. وقفه خارجی حساس به لبه پایین رونده تنظیم می‌شود. البته پیکربندی پورت‌ها، LCD و وقفه خارجی توسط محیط ویزارد انجام می‌شود. چهار بیت کم‌ارزش پورت C به عنوان خروجی با مقدار اولیه صفر و چهار بیت پر ارزش پورت C به عنوان ورودی و به صورت Pull-up تنظیم می‌شوند. به طوری که قبل از فشردن کلیدهای صفحه کلید، مقادیر سطرها صفر و ستون‌ها یک باشند. با فشردن هر یک از کلیدهای صفحه کلید، خروجی گیت AND از یک به صفر تغییر وضعیت می‌دهد و با توجه به تنظیم حساسیت وقفه به لبه پایین رونده، یک وقفه خارجی ایجاد می‌شود و برنامه وارد روتین وقفه می‌شود. در روتین وقفه، به ترتیب یکی از سطرها صفر و سه سطر دیگر یک می‌شوند. آنگاه، برنامه به بررسی وضعیت ستون‌ها می‌پردازد تا کلید فشرده شده را پیدا کند. برنامه به

صورت زیر است و این برنامه بر اساس کلیدهای صفحه کلید ماتریسی شکل (٤-٢) نوشته شده است.



شکل(٤-٢): شماتیک اسکن صفحه کلید ماتریسی ٤×٤ با استفاده از وقفه

✓ برنامه:

```
#include <mega16.h>
#asm
 .equ __LCD_port=0x18 ;PORTB
#endasm
#include <LCD.h>
#include <delay.h>
#include <stdio.h>
char x,aray[10];
interrupt [EXT_INT0] void ext_int0_isr(void)
{
delay_ms(50);
PORTC=0XFE;
if ((PINC&0XF0)==0XE0) {x=13;delay_ms(30);}
if ((PINC&0XF0)==0XD0) {x=8;delay_ms(30);}
if ((PINC&0XF0)==0XB0) {x=9;delay_ms(30);}
if ((PINC&0XF0)==0X70) {x=10;delay_ms(30);}
PORTC=0XFD;
if ((PINC&0XF0)==0XE0) {x=7;delay_ms(30);}
if ((PINC&0XF0)==0XD0) {x=5;delay_ms(30);}
if ((PINC&0XF0)==0XB0) {x=6;delay_ms(30);}
```

```
if ((PINC&0XF0)==0X70) {x=11;delay_ms(30);}
PORTC=0XFB;
if ((PINC&0XF0)==0XE0) {x=4;delay_ms(30);}
if ((PINC&0XF0)==0XD0) {x=2;delay_ms(30);}
if ((PINC&0XF0)==0XB0) {x=3;delay_ms(30);}
if ((PINC&0XF0)==0X70) {x=12;delay_ms(30);}
delay_ms(10);
PORTC=0XF7;
if ((PINC&0XF0)==0XE0) {x=1;delay_ms(30);}
if ((PINC&0XF0)==0XD0) {x=0;delay_ms(30);}
if ((PINC&0XF0)==0XB0) {x=14;delay_ms(30);}
if ((PINC&0XF0)==0X70) {x=15;delay_ms(30);}
sprintf(aray,"num = %i",x);
lcd_clear();
lcd_puts(aray);
delay_ms(100);
PORTC=0XF0;
}
void main(void)
{PORTA=0x0;DDRA=0x0;
PORTB=0x00;DDRB=0xFF;
PORTC=0xF0;DDRC=0x0F;
GICR|=0x40;MCUCR=0x02;
MCUCSR=0x00;GIFR=0x40;
TIMSK=0x00;ACSR=0x80;
SFIOR=0x00;lcd_init(16);
#asm("sei")
while (1);}
```

٤-٢- اسکن صفحه کلید ٤×٤ با انکدر MM74C922:

در این روش از انکدر M74C922 برای تشخیص کلید فشرده شده استفاده می‌شود. این انکدر ٨ ورودی دریافتی از صفحه کلید ماتریسی (داده‌های مربوط به ٤ سطر و ٤ ستون) را بر اساس جدول (٤-١) به یک کد ٤ بیتی تبدیل می‌کند به علاوه، هنگام تغییر وضعیت در ورودی‌ها یک پالس پایین رونده نیز تولید می‌نماید. جدول (٤-١) خروجی انکدر به ازای کلید فشرده شده را نشان می‌دهد که در آن X_n و Y_m مختصات کلید فشرده شده و ABCD عدد چهاربیتی خروجی است.

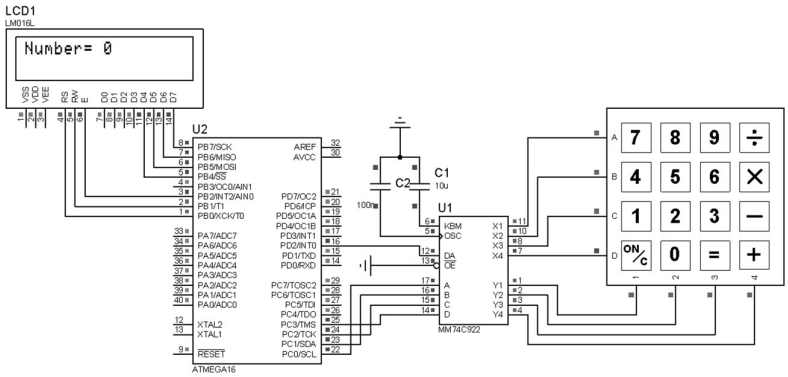
جدول(٤-١) خروجی انکدر به ازای کلید فشرده شده

موقعیت کلید فشرده شده	$Y_1 \cdot X_1$	$Y_1 \cdot X_2$	$Y_1 \cdot X_3$	$Y_1 \cdot X_4$	$Y_2 \cdot X_1$	$Y_2 \cdot X_2$	$Y_2 \cdot X_3$	$Y_2 \cdot X_4$	$Y_3 \cdot X_1$	$Y_3 \cdot X_2$	$Y_3 \cdot X_3$	$Y_3 \cdot X_4$	$Y_4 \cdot X_1$	$Y_4 \cdot X_2$	$Y_4 \cdot X_3$	$Y_4 \cdot X_4$
A	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
B	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
C	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
D	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

پروژه دوازدهم: اسکن صفحه کلید ماتریسی ۴×۴ توسط انکدر MM74C922

✍ برنامه‌ای بنویسید که صفحه کلید ماتریسی ٤×٤ را اسکن نماید و با فشردن هر کلید از صفحه کلید ماتریسی، عدد متناظر را بر روی LCD کاراکتری نشان دهد؟

☑ حل: در برنامه زیر، لبه بالا رونده ناشی از فشردن کلید منجر به درخواست وقفه خارجی می‌شود و در سرویس وقفه از روی جدول فوق، کلید فشرده شده تعیین می‌گردد و بر روی LCD نمایش داده می‌شود.



شکل(٤-٣): مدار اتصال صفحه کلید ٤×٤ به میکروکنترلر توسط انکدر MM74C922

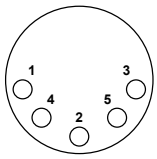
✓ برنامه

```
#include <mega16.h>
#include <stdio.h>
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>

// External Interrupt 0 service routine
```

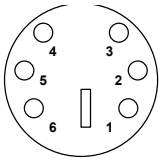
```
interrupt [EXT_INT0] void ext_int0_isr(void)
{
char code,array[10];
code=PINC;
sprintf(array,"Number= %u",code) ;
lcd_clear();
lcd_gotoxy(0,0);
lcd_puts(array);
}
void main(void)
{
PORTA=0x00; DDRA=0x00;
PORTB=0x00; DDRB=0x00;
PORTC=0x00; DDRC=0x00;
PORTD=0x00; DDRD=0x00;
TCCR0=0x00; TCNT0=0x00;
OCR0=0x00; TCCR1A=0x00;
TCCR1B=0x00; TCNT1H=0x00;
TCNT1L=0x00; ICR1H=0x00;
ICR1L=0x00; OCR1AH=0x00;
OCR1AL=0x00; OCR1BH=0x00;
OCR1BL=0x00; ASSR=0x00;
TCCR2=0x00; TCNT2=0x00;
OCR2=0x00; GICR|=0x40;
MCUCR=0x03; MCUCSR=0x00;
GIFR=0x40; TIMSK=0x00;
ACSR=0x80; SFIOR=0x00;
lcd_init(16);
#asm("sei")
while (1);}
```

٤-٣- اسکن صفحه کلید کامپیوتر:

صفحه کلید می‌تواند از طریق دو کانکتور: ps/2 و 5 pin DIN با ادوات خارجی مانند کامپیوتر یا میکرو ارتباط برقرار کند. در شکل (٤-٤) انواع کانکتورهای صفحه کلید ترسیم شده است و در جدول (٤-٢) و (٤-٣) توصیف پایه‌ها آمده است.



5 Pin DIN



PS/2

شکل (٤-٤): کانکتورهای صفحه کلید

جدول (٤-٢): پایه‌های کانکتور PS/2

Clock	1
GND	2
DATA	3
NC	4
VCC (5 ولت)	5
NC	6

جدول (٤-٣): پایه‌های کانکتور 5 pin DIN

Clock	1
DATA	2
NC	3
GND	4
VCC (5 ولت)	5

انتقال اطلاعات با لبه‌ی پایین رونده‌ی پالس ساعت همزمان (سنکرون) است: کلیدهای صفحه کلید به دو گروه یک بایتی و دو بایتی تقسیم می‌شوند:

الف- کلیدهای یک بایتی

هنگامی که یکی از کلیدهای یک بایتی فشرده می‌شود تا زمانی که کلید در حالت فشرده است کد اسکی آن کلید به کامپیوتر ارسال می‌شود و هنگامی که کلید رها شود ابتدا کد F0 H و سپس یک بار کد کلید فشرده شده ارسال می‌شود و تبادل اطلاعات این کلید به پایان می‌رسد. فرض کنید که کلید W برای مدتی فشرده شده و سپس رها شود در نتیجه کد ارسال شده به صورت زیر خواهد بود:

....1D1D1D1D1DF01D ☜

ب- کلیدهای دو بایتی

کد اسکی این کلیدها با E0 شروع می‌شود. **به عنوان مثال:** کد اسکی کلید END به صورت E069 می‌باشد به طور کلی عملکرد این کلیدها نیز مانند کلیدهای یک بایتی است با این تفاوت که هنگام رها شدن کلید ابتدا کد E0 و سپس F0 و در آخر هم بایت کم ارزش کد کلید فشرده شده ارسال می‌شود. فرض کنید که کلید END برای مدتی فشرده شده و سپس رها شود در نتیجه کد ارسال شده به صورت زیر خواهد بود:

...E069E069E069E069E0F069 ☜

جدول (٤-٤): کد اسکی کلیدهای صفحه کلید

نام کلید	کد اسکی	نام کلید	کد اسکی	نام کلید	کد اسکی	نام کلید	کد اسکی	نام کلید	کد اسکی	نام کلید	کد اسکی
A	1c	L	4B	W	1D	F8	0A	Left Alt	11	7,&	3D
B	32	M	3A	Y	35	F9	01	Right Alt	E0 11	6,^	36
C	21	N	31	Z	1A	F10	09	Tab	0D	5,%	2E
D	23	O	44	ESC	76	F11	78	Caps lock	58	4,$	25
E	24	P	4D	F1	05	F12	07	Back sp	66	3,#	26
F	2B	Q	15	F2	06	Space	29	\ ,|	5D	2,@	1E
G	34	R	2D	F3	04	Enter	5A	=,+	55	1,!	16
H	33	S	1B	F4	0c	Left shift	12	),0	45	`,~	0E
I	43	T	2C	F5	03	Right shift	59	(,9	46	Prt scr	E012 E07C
J	3B	U	3C	F6	0B	Left ctrl	14	-,_	4E	Scrl L	7E
K	42	X	22	F7	83	Right ctrl	E0 14	8,*	3E	NUM	77
+	79	→	E074	*	7C	3	7A	Insert	E070	Delete	E071
-	7B	←	E06B	.	71	2	72	Home	E06c	End	E069
↓	E072	↑	E075	7	6C	1	69	Page up	E07D	Page dow	E07A
/	E04A	4	6B	8	75	0	70	9	7D	Pause	E114 77E1 F014 F077

پروژه سیزدهم: اسکن صفحه کلید کامپیوتر توسط میکروکنترلر

برنامه‌ای بنویسید که صفحه کلید کامپیوتری را اسکن و کلید فشرده شده را بر روی LCD نمایش دهد. برنامه مورد نظر باید با فشرده شدن یک کلید همراه با کلید Shift، معادل حروف بزرگ کلید مربوطه را نمایش دهد؟

حل: در این برنامه پایه‌ی clock از کانکتور صفحه کلید به INT0 (PD.2) و پایه ی DATA نیز به PD.1 متصل شده است.

روند کلی به این صورت است که با هر پالس پایین رونده برنامه وارد سرویس وقفه ی INT0 می‌شود و در این سرویس بیت‌های توازن، خاتمه و شروع حذف شده و ۸ بیت داده از فریم داده جدا شده و در متغیری به نام my_data ذخیره می‌شوند.

توجه: روند ذخیره‌ی ۸ بیت داده را با مثال زیر شرح می‌دهیم:

فرض کنید که داده رسیده ۱۱۱۰۰۰۱۱ باشد: به تغییرات متغیر my_data توجه کنید:

- 10000000,11000000,01100000,00110000,00011000,10001100,11000110,11100011

در هر پالس پایین رونده که برنامه وارد سرویس وقفه شد: یک بیت در متغیر my_data قرار می گیرد و در انتهای این وقفه، INT0 در مد حساس به لبه‌ی بالا رونده پیکربندی می‌شود و در لبه بالا رونده، برنامه بررسی می‌کند که اگر ۱۱ بیت فریم داده را میکرو دریافت کرده باشد معادل کاراکتر کد ذخیره شده در متغیر my_data توسط تابع identify شناسایی شود.

مراحل تشخیص کد توسط تابع identify :

در یک تقسیم‌بندی دیگر کلیدهای صفحه کلید به ۳ دسته تقسیم می‌شوند:

۱. کلیدهای که همراه با آن‌ها کلید shift هم فشرده می‌شود (Shifted charactors)
۲. کلیدهایی که به تنهایی و بدون فشردن کلید shift فشرده شده‌اند (Unshifted charactors)
۳. کلیدهایی که دو بایتی هستند و کلیدهای shift چپ و راست هم نیستند (Extended charactors)

برای هر یک از این سه گروه، یک جدول جستجو در کتابخانه ای با نام codes.h طراحی گردیده است وظیفه‌ی تابع identify پس از تعیین نوع کلید فشرده شده معادل

کاراکتری آن را استخراج کرده و توسط تابع put_keyboardbuffer به بافر (keyboard_buffer[]) برای ذخیره‌ی موقت انتقال می‌دهد (برای استفاده از محتویات این بافر، دو اشاره گر با نام های my_input و my_output تعبیه شده است)
کلیدهای Extended نیز ابتدا توسط تابع put_extendedstring() به معادل رشته‌ای خود تبدیل شده و سپس توسط تابع put_keyboardbuffer() به بافر انتقال می‌یابند. تابعی با نام receive_key() طراحی گردیده است که کاراکتر ذخیره شده در بافر را به خروجی برده و سپس خروجی این تابع توسط دستور lcd_putchar(receive_key()) روی LCD چاپ می‌شود.

- **توجه:** به محض اینکه اولین کاراکتر وارد بافر شد تابع receive_key() شروع به کار می‌کند و این کاراکتر را به خروجی می‌برد. برای این منظور از دستور lcd_putchar که برای چاپ کاراکتر است استفاده شده است.

در این برنامه دو فایل کتاب خانه ای تعریف شده است: فایل کتابخانه ای codes.h را که قبلا توضیح داده شد، فایل pckeyboard.h که در آن توابع استفاده شده در برنامه تعریف شده است. این برنامه به روش چند فایلی نوشته شده است یکی از فایل ها main.c است و فایل دیگر pckeyboard.c است که شامل می‌باشد و به فایل main.c اضافه شده است. برای بهبود برنامه‌نویسی خود به نحوه ی استفاده ی مناسب از اشاره گرها، آرایه‌ها، تعریف بعضی متغیرها به صورت static، نحوه‌ی تعریف جدول جستجو و چگونگی جستجو در آن، پارتیشن بندی برنامه به صورت چند فایلی و ایجاد کتابخآن‌های مجزا برای توابع استفاده شده، توجه کنید. همچنین، کتابخانه‌های pc_keyboard.h و codes.h از CD همراه کتاب کپی و در فولدر inc از مسیر نصب نرم‌افزار CodeVision قرار بگیرند. اجرای این برنامه توسط نسخه ۱,۲۵,۳ یا ۱,۲٤,۲ پیشنهاد می‌شود.

- ✓ برنامه

```
#include <mega16.h>
#include <pc_keyboard.h>
#asm
  .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
#include <delay.h>

void main(void)
{
PORTA=0x00; DDRA=0xff;
PORTB=0x00; DDRB=0x00;
```

```
PORTC=0x00; DDRC=0x00;

PORTD=0x00; DDRD=0x00;
TCCR0=0x00; TCNT0=0x00; OCR0=0x00;

TCCR1A=0x00; TCCR1B=0x00; TCNT1H=0x00;
TCNT1L=0x00; OCR1AH=0x00; OCR1AL=0x00;
OCR1BH=0x00; OCR1BL=0x00;

// OC2 output: Disconnected
ASSR=0x00; TCCR2=0x00;
TCNT2=0x00; OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x40;


TIMSK=0x00; ACSR=0x80; SFIOR=0x00;
#asm("sei")
init_keyboard();
while (1)
     {
      lcd_putchar( receive_key() );
       delay_ms(50);
     };
}
```

✓ برنامه pc_keyboard.c مطابق زیر است:

```
#include <mega16.h>
#include <string.h>
#include "pc_keyboard.h"
#include "codes.h"
#define BUFFER_SIZE 60
unsigned char edge_state, bit_counter;          // 0 = negetive.  1 = positive.
unsigned char keyboard_buffer[BUFFER_SIZE];
```

```
unsigned char buffercounter;
unsigned char *my_input, *my_output;
//*********************************************************
void init_keyboard(void)
{
   MCUCR = 2;                          // INT0 interrupt on falling edge
   edge_state = 0;                       // 0 = falling edge  1 = rising edge
   bit_counter = 11;
      my_input =  keyboard_buffer;                    // Initialize buffer
   my_output = keyboard_buffer;
   buffercounter = 0;
}
//*********************************************************
interrupt [EXT_INT0] void ext_int0_isr(void)
{
   static unsigned char my_data;       // this var save the scancode receive
   if (edge_state)                     // if the edge is rising
   {
    MCUCR = 2;                          // configure interrupt on falling edge
      edge_state = 0;
     bit_counter=(bit_counter-1);
      if(bit_counter == 0)             // if the frame data receive compeletly
      {
         identify(my_data);
         bit_counter = 11;
      }
    }

    else {                  // if the edge is falling
      if(bit_counter < 11 && bit_counter > 2)    // if the data is bit3...10,
      {                                 //parity, start and stop bits are skipped.
        my_data = (my_data >> 1);
         if(PIND & 8)
            my_data = my_data | 0x80;   // put 1 into lastbit of var 'my_data'
      }

      MCUCR = 3;                          // configure interrupt on rising edge
      edge_state = 1;
        }
}
//*********************************************************
```

```
void identify(unsigned char key)
{
   static unsigned char key_push=0, is_shift = 0, mode = 0 , extrakey=0;
   unsigned char i;
   if (!key_push)              // if a key was pushed
   {
      switch (key)
      {
        case 0x12 :        // Left SHIFT is pushed
         is_shift = 1;
         break;

       case 0x59 :        // Right SHIFT is pushed
         is_shift = 1;
         break;
       case 0xF0 :        // if the key hold off
         key_push = 1;
         break;
       default:
           if(extrakey!=0xE0){
            if(!is_shift)                  // If shift not pushed
            {                          //  look up the table
                for(i = 0; noshift[i][0]!=key && noshift[i][0]; i++);
                if (noshift[i][0] == key) {
                    put_keyboardbuffer(noshift[i][1]);
                }}
               else {                  // If shift pushed
                for(i = 0; shift[i][0]!=key && shift[i][0]; i++);
                if (shift[i][0] == key) {
                    put_keyboardbuffer(shift[i][1]);
                } }
           } else{      //if the key does not find look up the extra table
            for(i = 0; extend[i][0]!=key && extend[i][0]; i++);
            if(extend[i][0]==key) {
                put_extendedstring(extend[i][1]);
            }   }
         break;
      }
   } else {
     key_push = 0;
```

```
      switch (key)
      {
       case 0x12 :                    // Left SHIFT is press
         key_push= 0;
         break;
         case 0x59 :                    // Right SHIFT  is press
         is_shift = 0;
         break;
      }
   }
extrakey=key;
}
//*************************************************************
void put_keyboardbuffer(unsigned char h)
{
   if (buffercounter<BUFFER_SIZE)                    // If buffer  is not full
   {
      *my_input =h;                          // character is placed into buffer
      my_input++;                             // pointer  is Incremented
      buffercounter++;
      if (my_input >= keyboard_buffer + BUFFER_SIZE)
         my_input = keyboard_buffer;
   }
}
//*************************************************************
int receive_key(void)
{
   int data;
   while(buffercounter == 0);                    // Wait for data
   data = *my_output;                          // Get byte
   my_output++;                             //  pointer Incremented
   if (my_output >= keyboard_buffer + BUFFER_SIZE)
     my_output = keyboard_buffer;
   buffercounter--;
   return data;
}
//*************************************************************
void put_extendedstring(flash unsigned char * b){
unsigned char z=0;
for(z=0;z<strlenf(b);z++){
  put_keyboardbuffer( *(b+z) ) ;}}  /***********The End**********/
```

فصل پنجم

کاربرد تایمرها

با توجه به عنوان کتاب، یعنی پروژه‌های کاربردی میکروکنترلرهای AVR، به منظور آشنایی مقدماتی با تایمرها مراجع پایانی کتاب پیشنهاد می‌شود و در این فصل پروژه‌های مرتبط با تایمرها بحث و برسی می‌شوند.

پروژه چهاردهم: فرکانس متر دیجیتال

✍ برنامه‌ای بنویسید که میکروکنترلر AVR فرکانس سیگنال ورودی شامل شکل موج سینوسی، مربعی، مثلثی و نظایر آن را اندازه‌گیری کند. همچنین، بیشینه فرکانس قابل اندازه‌گیری توسط میکرو را محاسبه نمائید؟

☑ حل: در این پروژه به منظور آشنایی بیشتر شما با میکروکنترلرهای AVR از تراشه ATmega32 استفاده شده است. این تراشه از لحاظ شکل ظاهری مشابه ATmega16 است با این تفاوت که حافظه‌های آن شامل حافظه Flash، SRAM و EEPROM دو برابر تراشه ATmega16 است. برنامه زیر تعداد لبه‌های بالا رونده پالس ورودی را در مدت یک ثانیه می‌شمارد. حال برای این کار: تایمر یک در مد شمارنده[1]، پالس‌های دریافتی را می‌شمارد و تایمر دو در مد Timer با استفاده از کریستال ساعت خارجی ۳۲/۷۶۸ کیلو هرتز، مطابق شکل (۵-۱)راه‌اندازی می‌شود و پس از مدت ۱ ثانیه تعداد پالس های دریافتی را بر روی LCD نشان می‌دهد.

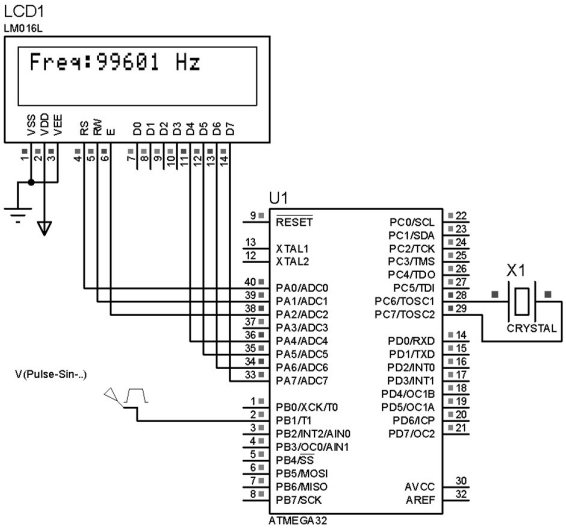
✓ برنامه:

```
#include <mega32.h>
#include <stdio.h>
#include <stdlib.h>
#include <delay.h>
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x1B ;PORTA
#endasm
```

[1] Counter

```
#include <lcd.h>
float x;
int y;
char s[15];
// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{y++;}
// Timer 2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{x=TCNT1+y*65536;
lcd_gotoxy(0,0);lcd_putsf("Freq:");
ftoa(x,0,s); TCNT1=0;   //lcd_gotoxy(1,1);
lcd_puts(s);lcd_putsf(" Hz");
x=0;y=0;}
void main(void)
{
PORTA=0x00; DDRA=0x00;PORTB=0x00; DDRB=0x00;
PORTC=0x00; DDRC=0x00;PORTD=0x00; DDRD=0x00;
TCCR0=0x00;TCNT0=0x00;OCR0=0x00;
// Timer/Counter 1 initialization
// Clock source: T1 pin Falling Edge
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00; TCCR1B=0x06;
TCNT1H=0x00; TCNT1L=0x00;
ICR1H=0x00; ICR1L=0x00;
OCR1AH=0x00; OCR1AL=0x00;
OCR1BH=0x00; OCR1BL=0x00;
// Timer/Counter 2 initialization
// Clock source: TOSC1 pin
// Clock value: PCK2/128
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x08; TCCR2=0x05;
```

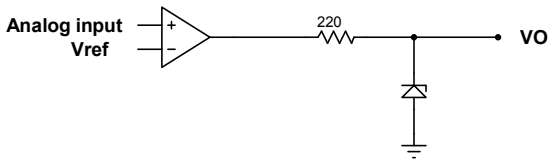
```
TCNT2=0x00; OCR2=0x00;
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00; MCUCSR=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x44;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80; SFIOR=0x00;
lcd_init(16);
#asm("sei")
x=0;
while (1) { };
}
```



شکل (۵-۱): مدار فرکانس‌متر دیجیتال

شکل (۵-۱) مدار فرکانس‌متر را نشان می‌دهد که در عمل با استفاده از اسیلاتور ٤ مگاهرتز، حداکثر قادر به اندازه‌گیری فرکانس سیگنال ورودی ۲ مگاهرتز می‌باشد. مدار عملی ساخته شده نشان می‌دهد که میکروکنترلر هر سیگنال ورودی را ابتدا توسط مدار اشمیت تریگر داخلی به موج مربعی تبدیل نموده و سپس آن را بررسی می‌نماید.

به دلیل افزایش ضریب اطمینان مدار شکل (۵-۲) پیشنهاد می‌شود که از شکل موج ورودی آنالوگ، یک پالس مربعی هم فرکانس با ورودی تولید می‌نماید:



شکل(۵-۲): مدار تولیدکننده پالس از شکل موج ورودی آنالوگ

پروژه پانزدهم: اندازه گیری درصد وظیفه و فرکانس ورودی با استفاده از وقفه

✍ برنامه‌ای بنویسید که با استفاده از وقفه‌ی صفر، درصد وظیفه و فرکانس پالس ورودی را اندازه‌گیری نماید و توسط LCD نمایش دهد؟

☑ حل:

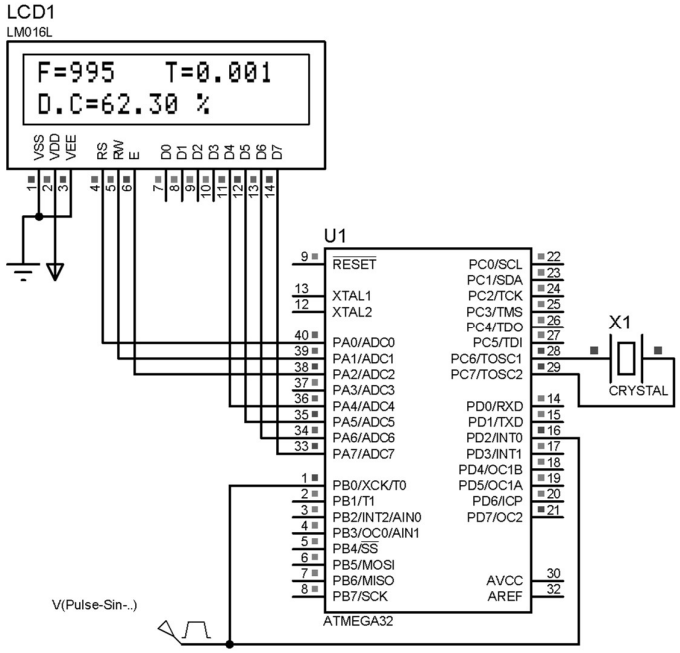
✓ برنامه

```
#include <mega32.h>
#include <stdio.h>
#include <stdlib.h>
#include <delay.h>
int y=0;;
float x; char c=0;
float t1,t2,t3,t4,dc,pp;
char s[9],s1[9],s2[9],strs[9] strs[9], array[14];
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
if (c){t2=TCNT1; TCNT1=0; MCUCR=0x02; c=0;}
else {t1=TCNT1; TCNT1=0; MCUCR=0x03; c=1;}
}
// Timer 0 overflow interrupt service routine
```

```
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
y++;
}
// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{}
// Timer 2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
x=y*256+TCNT0;
lcd_gotoxy(0,0);lcd_putsf("F=");ftoa(x,0,s);
lcd_puts(s);
pp=1/x;
ftoa(pp,3,s1);
lcd_gotoxy(8,0);lcd_putsf("T=");
lcd_puts(s1);
x=0;y=0;
TCNT0=0;
}
void main(void)
{
PORTA=0x00; DDRA=0x00;PORTB=0x00; DDRB=0x00;
PORTC=0x00; DDRC=0x00;PORTD=0x00; DDRD=0x80;

// Timer/Counter 0 initialization
// Clock source: T0 pin Falling Edge
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x06; TCNT0=0x00; OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 62,500 kHz
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: On
// Input Capture Interrupt: Off
```

```
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;TCCR1B=0x03;
TCNT1H=0x00;TCNT1L=0x00;
ICR1H=0x00;ICR1L=0x00;
OCR1AH=0x00;OCR1AL=0x00;
OCR1BH=0x00;OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: TOSC1 pin
// Clock value: PCK2/128
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x08;TCCR2=0x05;
TCNT2=0x00;OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;MCUCR=0x02;
MCUCSR=0x00;GIFR=0x40;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x45;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")
while (1)
    {
    t3=t1+t2; t4=t1;
```

```
dc=100*t1/t3;
ftoa(dc,1,strs);
lcd_gotoxy(0,1);
sprintf(array,"D.C=%s",strs);
lcd_puts(array);lcd_gotoxy(10,1);lcd_putsf("%");
delay_ms(100);
 };}
```



شکل (۵-۳): مدار مورد نظر برای اندازه‌گیری سیکل وظیفه و فرکانس موج ورودی

روش دوم: در این روش، برنامه با هر پالس پایین رونده وارد سرویس وقفه می‌شود و در سرویس وقفه تایمر یک روشن می‌شود و مدت زمانی که سیگنال در سطح منطقی صفر (PIND.2=0) و یا در سطح منطقی یک (PIND.2=1) می‌باشد را به طور جداگانه اندازه‌گیری می‌کند. سپس با توجه به زمان‌های بدست آمده، فرکانس و درصد وظیفه را اندازه‌گیری می‌کند و نتیجه را روی LCD نشان می‌دهد. فرکانس پالس ورودی با محاسبه معکوس مجموع زمان‌های high و low بدست می‌آید.

```
#include <mega32.h>
```

```
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#asm
   .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
float up,down,duty,f,n,sum;
char a1[10],a2[10],a3[32];
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{MCUCR=0x00;TCNT1=0x00;
//#asm("sei")
while(!PIND.2);
down=TCNT1;
TCNT1=0x00;
while(PIND.2);
up=TCNT1;
sum=up+down;
f=125000/sum;
duty = 100*up/sum;
ftoa(f,2,a1); ftoa(duty,2,a2);
sprintf(a3,"F=%s \nD.C=%s",a1,a2);
lcd_clear();lcd_puts(a3);delay_ms(100);
MCUCR=0x02; //n=0;
}
// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{n++;}
void main(void)
{PORTA=0x00;DDRA=0x00;
PORTB=0x00;DDRB=0x00;
PORTC=0x00;DDRC=0x00;
PORTD=0x00;DDRD=0x00;
// Timer/Counter 0 initialization
TCCR0=0x00;TCNT0=0x00;OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 125 kHz
// Mode: Normal top=FFFFh
```

```
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;TCCR1B=0x03;
TCNT1H=0x00;TCNT1L=0x00;
ICR1H=0x00;ICR1L=0x00;
OCR1AH=0x00;OCR1AL=0x00;
OCR1BH=0x00;OCR1BL=0x00;
// Timer/Counter 2 initialization
ASSR=0x00;TCCR2=0x00;
TCNT2=0x00;OCR2=0x00;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;MCUCR=0x02;
MCUCSR=0x00;GIFR=0x40;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x04;
// Analog Comparator initialization
ACSR=0x80;SFIOR=0x00;
// LCD module initialization
lcd_init(16);
// Global enable interrupts
#asm("sei")
while (1);}
```

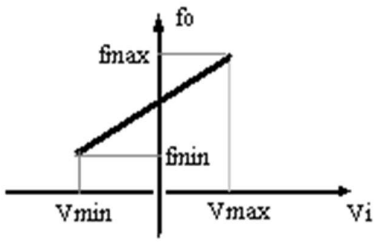
پروژه شانزدهم: ساخت یک نوسان‌ساز کنترل شونده با ولتاژ (VCO)

برنامه‌ای بنویسید که با توجه به دامنه سیگنال ورودی آنالوگ در AD0، یک پالس مربعی روی پایه PD.5 (OC1A) تولید کند که رابطه فرکانس آن با ولتاژ ورودی به صورت زیر باشد:

$$\text{Output frequency}=3000\times Vi+500 \qquad (۱-۵)$$

☑ **حل:** اسیلاتور کنترل شونده با ولتاژ، نوسان‌کننده‌ای است که فرکانس نوسان آن با تنظیم سطح ولتاژ ورودی کنترل می‌شود. از این اسیلاتور می‌توان در ساخت مدولاتور FM استفاده نمود.

❋ منحنی فرکانس خروجی بر حسب دامنه‌ی ولتاژ ورودی در VCO به صورت شکل (۵-٤) می‌باشد که در آن f_0 فرکانس مرکزی اسیلاتور می‌باشد و f_{max} و f_{min} به ترتیب حداکثر و حد اقل فرکانس نوسان را بیان می‌کنند.



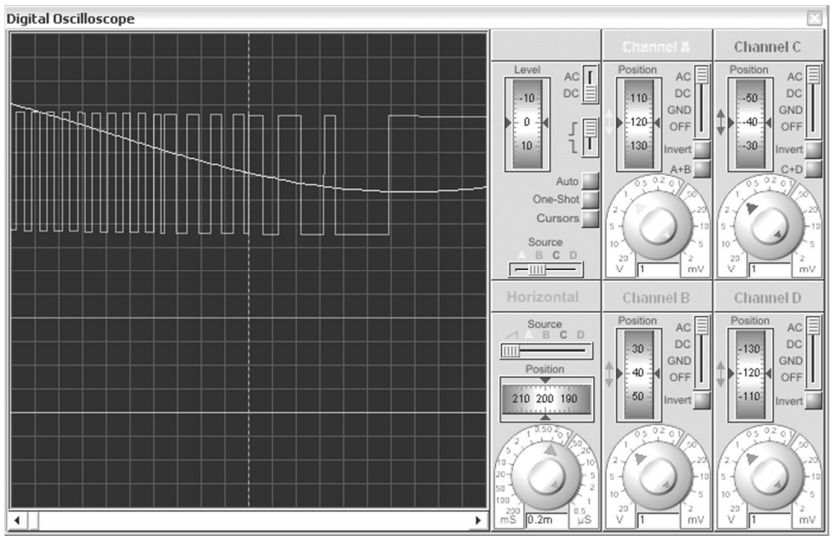
شکل (۵-٤): منحنی فرکانس خروجی بر حسب دامنه‌ی ولتاژ ورودی

➢ **توجه :** در این برنامه، فرکانس شمارش تایمر Hz ۳۱۲۵۰ انتخاب شده است و ADC در مد ۱۰ بیتی تنظیم شده است.

✓ برنامه:

```
#include <mega16.h>
#define ADC_VREF_TYPE 0x40
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
unsigned int code;
float volt,  frequency;
void main(void)
{
PORTA=0x00; DDRA=0x00;
PORTD=0x00; DDRD=0x20;
TCCR1A=0x40;
TCCR1B=0x0C;
```

```
ACSR=0x80;
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;
SFIOR&=0xEF;
while (1)
    { code=read_adc(0);
    volt=code*0.0048;
    frequency=(3000*volt+500);
    OCR1A=(31250/(2*frequency));
     };
}
```



شکل (۵-۵): فرکانس سیگنال خروجی وابسته به دامنه سیگنال ورودی

پروژه هفدهم: تنظیم و کنترل فرکانس و درصد وظیفه خروجی توسط کاربر

✍ برنامه‌ای بنویسید که کاربر توسط یک کی‌پد، فرکانس و درصد وظیفه سیگنال PWM خروجی از پایه OCR1A را کنترل کند.

☑ حل: مدار این پروژه در شکل (۵-٦) ترسیم شده است که در آن یک کی‌پد توسط انکدر MM74C922 به میکروکنترلر متصل شده و مقادیر فرکانس و درصد وظیفه توسط میکروکنترلر بر روی LCD نمایش داده می‌شوند. شکل موج خروجی که درصد وظیفه و فرکانس آن توسط کاربر قابل تنظیم است بر روی پایه PD.5 یا OCR1A قابل ملاحظه است.

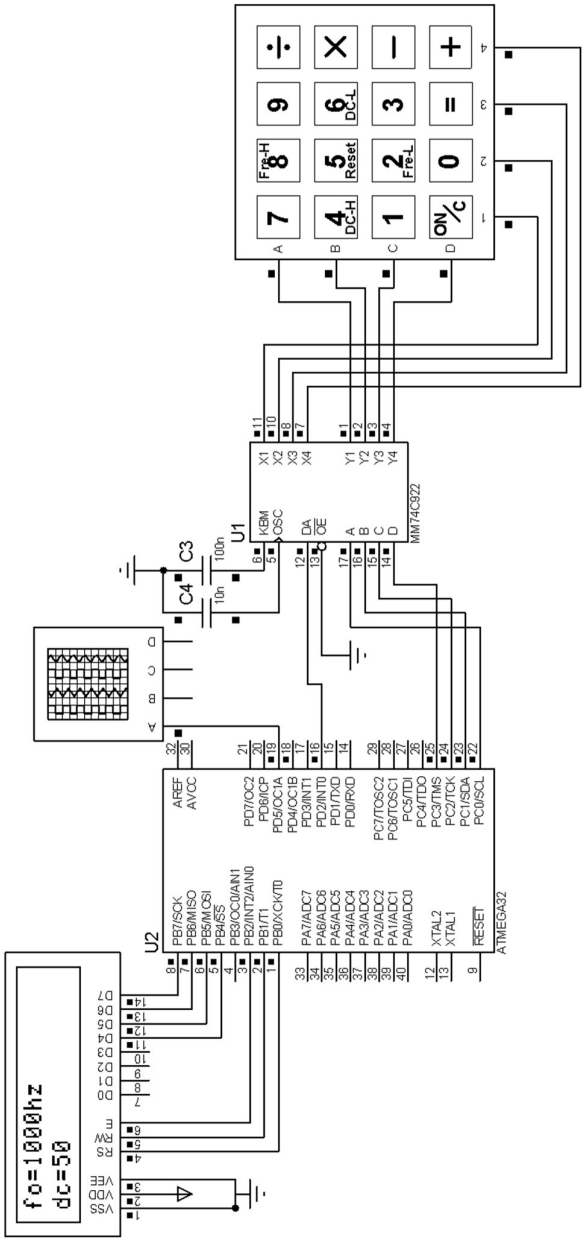
برنامه مطابق زیر است و در آن تایمر یک به صورت PWM برنامه‌ریزی شده است و با فشردن کلیدهای ۸ و ۲ به ترتیب فرکانس ۱۰ هرتز افزایش و کاهش می‌یابد. همچنین با فشردن کلیدهای ٤ و ٦ به ترتیب درصد وظیفه یک درصد افزایش و کاهش می‌یابد و با فشردن کلید ۵، فرکانس و درصد وظیفه سیگنال خروجی OCR1A به حالت اولیه باز می‌گردد.

✓ برنامه

```
#include <mega32.h>
#include<delay.h>
#include <stdio.h>
#asm
  .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
unsigned int c,d;
float v;
long int f;
int a,n,b,k,i,dc;
char s[20],s1[20];
interrupt [EXT_INT0] void ext_int0_isr(void)
{
a=1;k=0;i=0;n=PINC;
if(n==1)b=1;
else if(n==9)b=2;
else if(n==4)b=3;
else if(n==6)b=4;
else if(n==5)b=5;
l0:delay_ms(500);
  if(PIND.2==1){k++;goto l0;}
if(k==0)  i=1;
}
void main(void)
{
PORTA=0x00; DDRA=0x00;
PORTB=0x00;DDRB=0xF7;
PORTC=0x00; DDRC=0x00;
PORTD=0x00; DDRD=0x30;

// Timer/Counter 0 initialization
TCCR0=0x00;TCNT0=0x00;
```

```
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 8000.000 kHz
// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Inverted
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0xB2;
TCCR1B=0x19;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x1f;
 ICR1L=0x3f;
OCR1AH=0x0F;
OCR1AL=0xA0;
OCR1BH=0x00;
OCR1BL=0x00;
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x03;
MCUCSR=0x00;
GIFR=0x40;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
ACSR=0x80;
SFIOR=0x00;
lcd_init(16);
```

```
#asm("sei")

while (1)
{
f=1000; dc=50;
l1:a=0;
v=(float)dc/100;
d=v*c;
OCR1A=d;
sprintf(s,"fo=%uhz",f);
lcd_clear();
lcd_puts(s);
printf(s1,"dc=%u   ",dc );
lcd_gotoxy(0,1);
lcd_puts(s1);
while(a==0);
a=0;
        if(b==1)  f=f+1000*k+10*i;
        else if(b==2)   f=f-1000*k-10*i;
        else if(b==3)   dc=dc+5*k+i;
        else if(b==4)   dc=dc-5*k-i;
        else if(b==5)  {f=1000;dc=50;}
        else   goto l1;
       if(f>80000) {f=80000; goto l2;}
       else if(f<122) { f=122; goto l2;}
        if(dc>100) {dc=100; goto l2;}
        else if(dc<0){ dc=0; goto l2;}
l2:c=8000000/f;
goto l1;
  };
}
```



شکل (٥-٦): مدار تنظیم و کنترل درصد وظیفه و فرکانس توسط کاربر

پروژه هجدهم: مدار کنترل موتور DC

✍ برنامه‌ای بنویسید که با استفاده از چهار کلید از یک صفحه کلید ماتریسی (کی‌پد) مطابق شکل (۵-۷)، یک موتور DC را توسط تراشه راه‌انداز L298 بر مبنای قوانین زیر کنترل کند.

❶ با هر بار فشار دادن کلید ON/OFF، پایه‌ی خروجی PD5 و PD6 که به دو کانال در حال استفاده‌ی درایور متصل است، تغییر وضعیت دهند.

❷ با هر بار فشردن کلیدهای DC مقدار درصد وظیفه‌ی پالس PWM خروجی، که توسط درایور به یکی از پایه‌های تغذیه موتور وصل است و در ابتدای برنامه پنجاه درصد است، دو درصد تغییر کند.

❸ با هر بار فشار دادن کلید L/R موتور تغییر وضعیت بدهد.

❹ می‌خواهیم در هر حالت خاموش یا روشن بودن و جهت چرخش موتور به همراه درصد وظیفه خروجی میکروکنترلر مطابق شکل زیر روی LCD نمایش داده شود:

```
rotate:R  DC=50%
  motor is on.
```

☑ **حل:** یکی از بهترین و پرکاربردترین روش‌های کنترل دور موتورهای DC (به ویژه در اندازه‌های کوچکتر) کنترل موتور توسط ولتاژ آرمیچر است که یک کنترل وسیع خطی در اختیار کاربران قرار می‌دهد. ایجاد ولتاژهای متغیر DC در سطوح پایین، کار نسبتاً ساده‌ای است. دو روش مرسوم برای این کار: استفاده از مبدل دیجیتال به آنالوگ و تکنیک PWM است، که روش دوم ساده‌تر، ارزان‌تر و رایج تر می‌باشد.

سرعت کلیدزنی[1] موتورهای DC در مقایسه با سایر قطعات الکترونیکی، کمتر می‌باشد و قادر به دنبال کردن تغییرات لحظه‌ای ولتاژ ورودی در فرکانس‌های حتی در حدود کیلوهرتز و بالاتر نیستند و در این فرکانس‌ها به مقدار متوسط ورودی‌های خود پاسخ می‌دهند. بنابراین می‌توان به جای اعمال یک ولتاژ یکسو شده‌ی خاص به آن‌ها، ولتاژی دارای موجک AC، که همان مقدار متوسط (DC) را دارد، به آن‌ها اعمال کرد و خروجی مشابهی از موتور دریافت نمود. این قضیه، مبنای استفاده از مدولاسیون پهنای پالس در کنترل موتورهای DC است. در واقع، در این روش ما با تغییر درصد

[1] Switching

وظیفه‌ی یک موج مربعی، از صفر تا صد درصد، یک ولتاژ متغیر بین صفر تا مقدار حداکثر، برای کنترل سرعت موتور به آرمیچر آن اعمال می‌کنیم.

وجود چندین کانال خروجی PWM با حالت‌های مختلف موج خروجی، یکی از مشخصه‌های بارز میکروکنترلرهای AVR است که در این پروژه، از آن استفاده می نمائیم. اما از آن‌جا که در راه اندازی موتورهای DC، تنها از مقدار متوسط موج PWM استفاده می‌شود و از سایر ویژگی‌های آن از جمله تصحیح فاز و فرکانس و دوشیبه یا تک شیبه بودن موج مولد آن استفاده نمی‌شود، در این آزمایش، تنها از حالت PWM سریع تایمرهای میکروکنترلر استفاده می‌کنیم و تایمر دو را در حالت PWM برنامه‌ریزی می‌نمائیم.

برای اعمال تغذیه یک موتور توسط یک میکروکنترلر، از یک تراشه راه‌انداز[1] استفاده می‌شود. بنابراین ضروری است در انتخاب فرکانس موج اعمالی، به پاسخ فرکانسی بافرها و تقویت کننده‌های راه‌انداز دقت شود تا فرکانس در حدی نباشد که شکل موج پس از عبور از راه‌انداز تغییرات اساسی پیدا کند. فرکانس مطمئن برای موتورهای DC سایز کوچک و درایورهای رایج آن‌ها، تقریباً بین ۱ kHz تا ۵ kHz است.

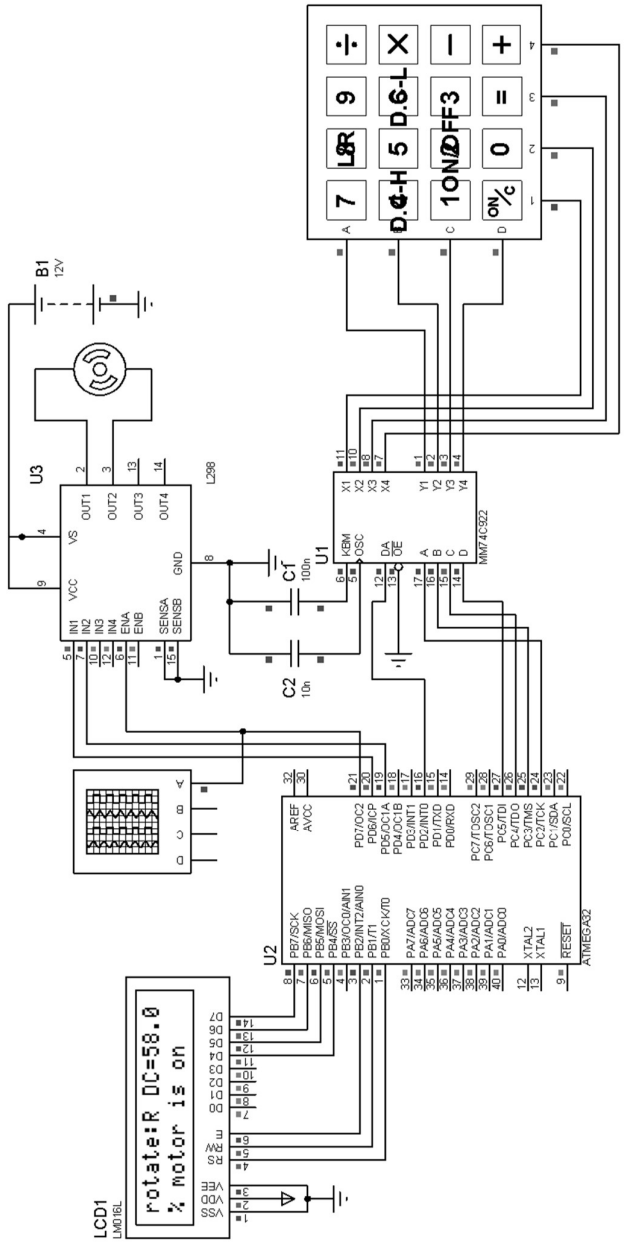
✓ **برنامه**

```
/*************************************************
Chip type           : ATmega32
Clock frequency     : 4.000000 MHz
*************************************************/
#include <mega32.h>
#include <stdio.h>
#include <stdlib.h>
#include <delay.h>
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
float e;
int dc;
char a,k,r,n,b;
char s[20],array[7];
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
```

[1] Driver

```
a=1; n=PINC; n=n>>2;
if(n==1) b=1;
else if(n==9) b=2;
else if(n==4) b=3;
else if(n==6) b=4;
else a=1;
}

void main(void)
{
PORTA=0x00;DDRA=0x00;
PORTB=0x00;DDRB=0xF7;
PORTC=0x00;DDRC=0x00;
PORTD=0x00;DDRD=0xE0;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 500,000 kHz
// Mode: Fast PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;TCCR2=0x6A;
TCNT2=0x00;OCR2=0x03;
TIMSK=0x00;

// External Interrupt(s) initialization
// INT0: On
GICR|=0x40;MCUCR=0x02;
MCUCSR=0x00;GIFR=0x40;
lcd_init(16);

// Global enable interrupts
#asm("sei")
while (1)
{
lcd_clear();
lcd_putsf("rotate:R DC=0  %  motor is off");
k=0;r=0;dc=0;
l:a=0;
  while(!a);
  if(b==3){
     if(dc<255)dc=dc+5;
```

```
    else {dc=100; goto l;}
        }
else if(b==4)
   {if(dc>5)dc=dc-5;
    else{dc=0; goto l;}
      }
else if(b==2){
      if(k==1){
        PORTD.5=0;PORTD.6=0;
        lcd_gotoxy(11,1);
        lcd_putsf("off");
        k=0;}
    else{
        PORTD.5=0;PORTD.6=1;dc=128;
        lcd_gotoxy(11,1);
        lcd_putsf("on ");
        k=1;}
         }
else if(b==1){
   if(r==1){
      PORTD.6=1;PORTD.5=0;
      lcd_gotoxy(7,0);
      lcd_putsf("R");
      r=0;}
   else{
      PORTD.6=0;PORTD.5=1;
      lcd_gotoxy(7,0);
      lcd_putsf("L");
      r=1;}
       }
else goto l;
OCR2=dc;
e=dc/2.55;
ftoa(e,1,array);
sprintf(s,"DC=%s%%",array);
lcd_gotoxy(9,0); lcd_puts(s);
goto l;
 };
}
```



شکل (۵-۷): مدار کنترل موتور DC

در کاربردهای صنعتی، اندازه‌گیری سرعت دوران شفت یک موتور معمولاً از روی سیگنال خروجی شفت انکدر صورت می‌پذیرد. در صورتی که سرعت چرخش موتور کمتر از ۲۰۰ دور در دقیقه (200 RPM) باشد: روش اندازه‌گیری دوره تناوب پالس خروجی شفت انکدر و سپس معکوس نمودن آن برای محاسبه فرکانس چرخش (یا RPM) خطای محاسباتی کمتری دارد و در سرعت‌های بالاتر از آن اندازه‌گیری مستقیم فرکانس مناسب‌تر است.

پروژه نوزدهم: محاسبه‌ی RPM موتور

✍ برنامه‌ای بنویسید که توسط یک انکدر ۱۰۰۰ «پالس در دور»[1]، تعداد دور موتور در دقیقه را محاسبه کند و روی LCD نشان دهد.

☑ حل: در این برنامه از تایمر صفر در مد کانتر برای شمارش تعداد پالس‌های انکدر و از تایمر یک در مد زمان سنج برای محاسبه‌ی دور در دقیقه موتور استفاده شده است و در پایان برنامه، تعداد پالس‌های شمارش شده در دقیقه بر ۱۰۰۰ تقسیم شده تا تعداد دورها در دقیقه حاصل شوند.

✓ برنامه

```
#include <mega16.h>
#include<stdio.h>
char array[31];
int n,k,m;
#asm
 .equ __LCD_port=0x1B
#endasm
#include <lcd.h>
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
n++;
}
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
m++;
if (m==60){
k=(((n*256)+TCNT0)/1000);
sprintf(array,'%d',k);
lcd_puts(array);
m=0;
      }
```

[1] Pulse Per Rotation (PPR).

```
TCNT1H=0x7A;
TCNT1L=0x12;
}
void main(void)
{
TCCR0=0x06;
TCCR1B=0x04;
TCNT1H=0x7A;
TCNT1L=0x12;
TIMSK=0x05;
ACSR=0x80;
lcd_init(16);
#asm("sei")
while (1);
```

یکی از روش‌های نمایش اعداد چند رقمی روی نمایش‌گرهای هفت قسمتی استفاده از روش مالتی پلکس کردن خط و داده می‌باشد. همانطور که در فصل دوم بیان شد: در این نوع نمایش‌گرها کلیه خطوط داده همه‌ی ارقام به هم متصل شده‌اند و برای آند یا کاتد مشترک هر رقم نیز یک پایه در نظر گرفته شده است. در این صورت برای یک نمایش‌گر ٤ رقمی، ١٢ پایه مورد نیاز است.

به منظور نمایش یک عدد ٤ رقمی بر روی نمایش‌گر کافی است، هر یک از ارقام این عدد در کسر کوچکی از ثانیه روی نمایش‌گر نشان داده شوند. برای این منظور: ابتدا داده رقم یکان روی پورت مربوطه قرار می‌گیرد و آند آن فعال می‌گردد. پس از مدت T ثانیه آند مربوطه غیر فعال شده، داده رقم بعدی روی پورت داده واقع شده و سپس آند مربوطه فعال می‌گردد. اگر این کار به طور مداوم تکرار شود و «فرکانس تازه‌سازی»[1] عدد روی نمایش‌گر از 25Hz بیشتر باشد دیتای نشان داده شده از دید بیننده ثابت خواهد بود.

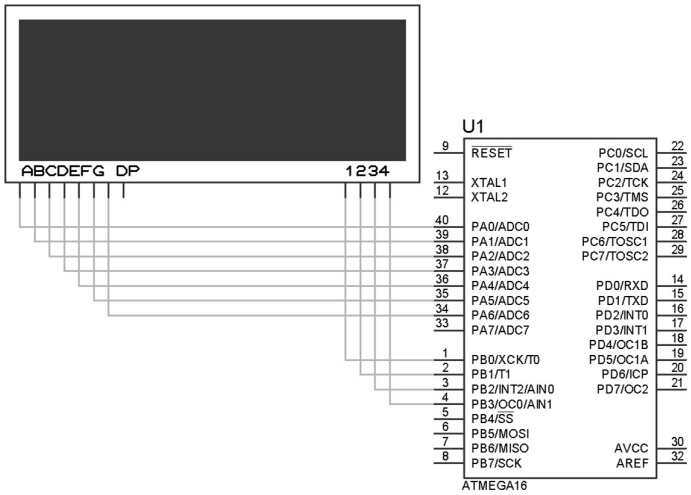
پروژه بیستم: اسکن نمایش‌گر هفت پارچه

✍ برنامه‌ای بنویسید که عدد 2567 را بر روی یک نمایش‌گر هفت قسمتی مالتی پلکس شده از نوع آند مشترک نشان دهد.

☑ حل: در این برنامه ابتدا یکان، دهگان، صدگان و هزارگان عدد مورد نظر توسط تابع Digit_Extract() استخراج شده و سپس هر ٢ میلی ثانیه یک بار عملیات

[1] Refreshing Frequency

تازه‌سازی درون سرویس وقفه سرریز تایمر یک با استفاده از DisplayRefresh() انجام می‌شود. همچنین، مطابق شکل (۵-۸)، آندهای ٤ نمایش‌گر به پایه‌های B.0 ... B.3 میکروکنترلر متصل شده‌اند.



شکل(۵-۸): سخت‌افزار اتصال نمایش‌گر هفت قسمتی ٤ رقمی به میکروکنترلر به روش مالتی پلکس کردن خط و داده.

❋ **چگونگی استخراج یکان، دهگان، صدگان و هزارگان عدد مورد نظر:**

ابتدا عدد مورد نظر بر ۱۰ تقسیم شده و باقیمانده به عنوان یکان در نظر گرفته می شود. سپس خارج قسمت دوباره بر ۱۰ تقسیم شده و باقیمانده به عنوان دهگان در نظر گرفته می‌شود و این عمل به همین ترتیب تکرار می‌شود تا هزارگان عدد مورد نظر حاصل شود.

☜ **معرفی توابع و آرایه‌های استفاده شده در برنامه:**

۱. در آرایه code، کدهای متناظر با ارقام ۰ تا ۹ که باید به PORTA نسبت داده شود، قرار داده شده است.

```
int code[11]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90}
```

۲. در آرایه segment_port عدد متناظر با PORTB برای روشن شدن هر کدام از نمایش‌گر ها قرار داده شده است.

```
int segment_port[5]={1,2,4,8}
```

۳. در آرایه DisplayMem یکان، دهگان، صدگان و هزارگان عدد مورد نظر قرار داده شده است.

٤. تابع Digit_Extract(void)، یکان، دهگان، صدگان و هزارگان عدد مورد نظر را استخراج کرده و درون آرایه‌ای به نام DisplayMem[] قرار می‌دهد.

۵. تابع void DisplayRefresh(void) عملیات تازه‌سازی را انجام می‌دهد به این ترتیب که توسط دستور PORTB=segment_port[segment_num] نمایش‌گری که قرار است روشن شود را روشن می‌کند و سپس توسط دستور PORTA=code[DisplayMem[segment_num]] عدد متناظر با آن نمایش‌گر را روی پورت قرار می‌دهد.

✓ برنامه

```
#include <mega16.h>
#include <delay.h>
void DisplayRefresh(void);
void Digit_Extract(void);
int code[11]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int segment_port[5]={8,4,2,1};
int num=2567,segment_num,DisplayMem[5];
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
DisplayRefresh();
segment_num++;
if (segment_num==4) segment_num=0;

TCNT1H=0xff;
TCNT1L=0xc0;
}
void main(void)
{
Digit_Extract();
DDRA=0xff;
PORTB=0x00;
DDRB=0xFF;
TCCR1A=0x00;
TCCR1B=0x04;
TCNT1H=0xff;
```

```
TCNT1L=0xc0;
TIMSK=0x04;
ACSR=0x80;
SFIOR=0x00;
#asm("sei")
while(1)
    {
    }
}
void DisplayRefresh(void)
{
PORTB=segment_port[segment_num];
PORTA=code[DisplayMem[segment_num]];
}
void Digit_Extract(void)
{
int a,r,i;
a=num/10;
r=num%10;
DisplayMem[0]=r;
for(i=1;i<4;i++)
    {
    r=a%10;
    a/=10;
    DisplayMem[i]=r;
    }
}
```

پروژه بیست‌ویکم: کسینوس φ متر

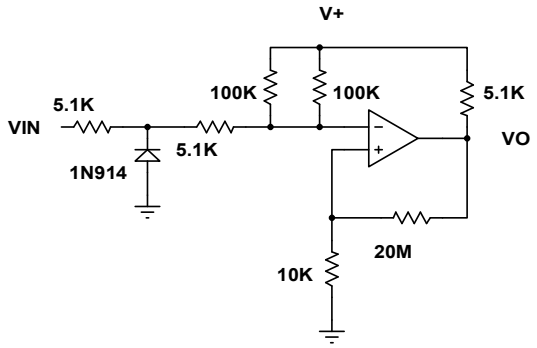
✍ برنامه‌ای بنویسید که کسینوس φ یک سیگنال ورودی را اندازه‌گیری کند؟

☑ حل: برای اندازه‌گیری کسینوس φ یک مدار الکتریکی، یک نمونه از ولتاژ و جریان گذرنده از مدار را بدست آورده و توسط مدار مبدل، سیگنال‌های مربعی هم فاز با این ورودی‌ها بدست می‌آیند. سپس اختلاف زمانی بین دو لبه ی بالا رونده‌ی این دو سیگنال را بدست آورده و از رابطه‌ی (۵-۲) اختلاف فاز آن‌ها بر حسب رادیان محاسبه می‌گردد.

$$\varphi = \frac{t_0}{T} \times \frac{\pi}{180} \qquad (۵-۲)$$

که t_0 اختلاف زمانی بین لبه‌های بالا رونده پالس های نمونه ولتاژ و جریان است و T نیز دوره تناوب آن پالس‌ها می‌باشد.

مدار مبدل مورد نظر یک مدار تشخیص گذر از صفر می‌باشد که با مثبت شدن سیگنال ولتاژ یا جریان "۱" و با منفی شدن آن صفر می‌شود. برای پیاده‌سازی این مدار می‌توان از مدار شکل (۵-۹) استفاده نمود. لازم به ذکر است که تقویت کننده عملیاتی استفاده شده در این مدار نیازی به منبع تغذیه دوبل ندارد.



شکل(۵-۹): مدار مبدل سینوسی به پالس بدون نیاز به منبع تغذیه دوبل

شرح برنامه: در این پروژه فرض می‌شود که ولتاژ و جریان به طور جداگانه در اختیار ما می‌باشند. در این برنامه سیگنال ولتاژ به پایه ی PD.2 و سیگنال جریان به پایه ی PD.3 از تراشه ATmega16 متصل شده است. با هر پالس بالا رونده‌ی سیگنال ولتاژ، برنامه وارد سرویس وقفه شده و در آنجا تایمر ۲ راه‌اندازی می شود. سپس در اولین پالس بالا رونده‌ی سیگنال جریان، تایمر ۲ خاموش شده و مقدار اختلاف زمانی (t_0) محاسبه و طبق رابطه‌ی (۵-۲) اختلاف فاز بر حسب رادیان محاسبه می‌شود و سپس کسینوس آن روی LCD نمایش داده می‌شود.

توجه:

۱. برای محاسبه cos از تابع زیر استفاده شده است

```
float cos(float x)
```

این تابع مقدار کسینوس زاویه x را (بر حسب رادیان) محاسبه کرده به خروجی می‌برد.

۲. از تایمر صفر و یک برای محاسبه‌ی دوره تناوب استفاده شده است که در برنامه های قبلی توضیح داده شده است.

برنامه

```
#include <mega16.h>
#include <stdio.h>
#include <math.h>
// Alphanumeric LCD Module functions
#asm
 .equ __LCD_port=0x1B
#endasm
#include <LCD.h>
long int f,t,angle,t0,n;
char array[31];

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
TCCR2=0x06;
}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
TCCR2=0;
t0=(TCNT2/31250);
angle=(t0/t)*3.14/180;
sprintf(array,"cos(Q)=%i",cos(angle));
LCD_puts(array);
TCNT2=0;
}

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
n=n+1;
}

// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
f=(256*n)+TCNT0;
t=(1/f);
// Reinitialize Timer 1 value
```

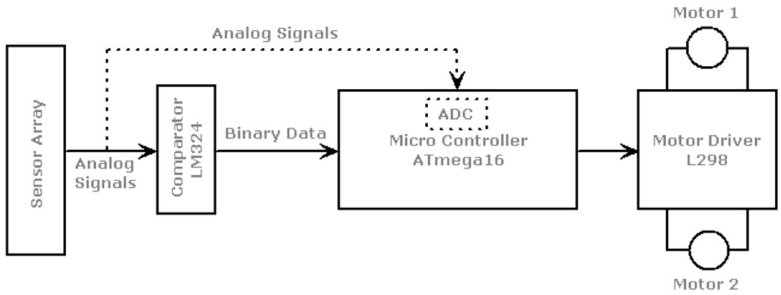
```
TCNT1H=0x85;
TCNT1L=0xED;
}

void main(void)
{
PORTA=0x00;DDRA=0xFF;
// Timer/Counter 0 initialization
// Clock source: T0 pin Falling Edge
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x06;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 31.250 kHz
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;TCCR1B=0x04;
TCNT1H=0x85;TCNT1L=0xED;
OCR1AH=0x00;OCR1AL=0x00;
OCR1BH=0x00;OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 31.250 kHz
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: On
```

```
// INT1 Mode: Rising Edge
// INT2: Off
GICR|=0xC0;
MCUCR=0x0F;
MCUCSR=0x00;
GIFR=0xC0;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x05;
ACSR=0x80;
SFIOR=0x00;
LCD_init(16);
#asm("sei")
while (1);
}
```

پروژه بیستودوم: ربات مسیریاب

✍ برنامه یک ربات مسیریاب با چیدمان هشت سنسوری را بنویسید؟

☑ حل: ربات مسیریاب ماشینی است که مسیر از پیش تعیین شده‌ای را دنبال می‌کند. این مسیر می‌تواند مانند خطوط سیاه روی زمینه سفید قابل مشاهده باشد و یا این‌که مانند زمین مغناطیسی غیر قابل مشاهده باشد. مسیر رباتی که در این پروژه طراحی شده است: از نوع خطوط سیاه بر روی زمینهٔ سفید و قابل مشاهده می باشد. جهت کنترل صحیح ربات از سنسورهای نوری مدل CNY70 استفاده شده است. از این سنسورها، برای تشخیص مسیر مشکی رنگ استفاده می‌شود که موقعیت ربات نسبت به مسیر را مشخص می‌کنند و بر اساس آن‌ها فرمان مناسب برای کنترل، جهت و میزان چرخش موتورها صادر می‌شود. شکل (۵-۱۰) بلوک دیاگرام کنترل ربات را نشان می‌دهد.



شکل(۵-۱۰): بلوک دیاگرام کنترل ربات

☜ **عملکرد ربات:** ربات از سنسورهای «مادون قرمز»[1] به منظور تشخیص مسیر استفاده می‌کند. بنابراین، آرایه‌ای از ۸ عدد سنسور فرستنده و ۸ عدد سنسور گیرنده استفاده شده است. سنسور فرستنده، امواج مادون قرمز ساطع می‌کند و گیرنده بر اساس میزان امواج مادون قرمز دریافتی سطح ولتاژی متناسب با آن تولید می‌کند. یعنی خروجی این سنسور گیرنده، سیگنال آنالوگی است که بستگی به میزان نور بازگشتی دارد. این سیگنال آنالوگ برای تولید صفر و یک منطقی به یک مقایسه کنندهٔ آنالوگ داده می‌شود و در نهایت داده خروجی مقایسه‌کننده آنالوگ به میکرو تحویل داده می‌شود. نحوه چینش زوج سنسورهای فرستنده و گیرنده در شکل (۵-۱۱) نشان داده شده است. چنانچه از وسط شروع کنیم سنسورهای سمت چپ به ترتیب: L3, L2, L1 و L4 و سنسورهای سمت راست به ترتیب: R4, R3, R2, R1 نام‌گذاری شده‌اند.

L4	L3	L2	L1	R1	R2	R3	R4

شکل(۵-۱۱): چیدمان سنسورها

همچنین فرض می‌کنیم که چنانچه سنسور روی خط باشد، سیگنال دریافتی در پایه میکرو صفر و چنانچه خارج از خط باشد، یک خوانده شود. میکرو بر اساس الگوریتم داده شده‌ی زیر حرکت بعدی خود را تعیین می‌کند. در این الگوریتم میکرو تلاش می کند تا ربات به نحوی قرار گیرد که سنسور R_1, L_1 مقدار صفر و بقیهٔ سنسورها مقدار ۱ را بخوانند.

L4	L3	L2	L1	R1	R2	R3	R4
1	1	1	0	0	1	1	1

[1] Infrared (IR)

شکل (۵-۱۲) :بهترین موقعیت برای ربات

الگوریتم: فرض کنید L شمارهٔ چپ‌ترین سنسوری است که مقدار صفر را می خواند و R شماره‌ی راست‌ترین سنسوری است که مقدار صفر را می‌خواند.

۱. چنانچه مقدار خوانده شدهٔ هیچ کدام از سنسورهای چپ (یا راست) برابر صفر نباشد L (یا R) را برابر صفر قرار بده. به مثال زیر توجه کنید.

L4	L3	L2	L1	R1	R2	R3	R4
1	0	0	1	1	1	1	1

L=3 R=0

L4	L3	L2	L1	R1	R2	R3	R4
1	1	0	0	0	0	0	0

L=2 R=4

L4	L3	L2	L1	R1	R2	R3	R4
1	0	1	0	1	0	0	1

L=3 R=3

شکل(۵-۱۳): شکل قرارگیری ربات در سه موقعیت متفاوت

۲. در صورتی که همهٔ سنسورها مقدار یک را بخوانند، به مرحلهٔ ۳ برو در غیر این صورت
   - اگر L>R به چپ برو.
   - اگر L < R به راست برو.
   - اگر L = R مستقیم برو.

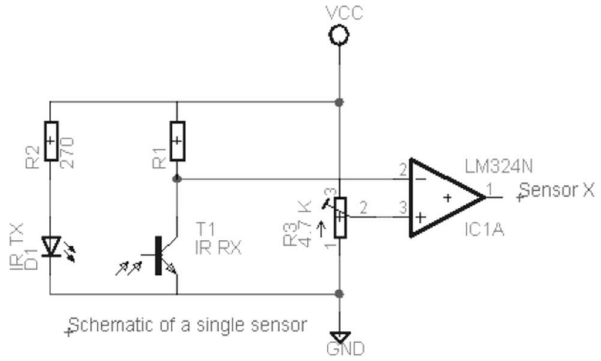
   به مرحلهٔ ٤ برو.

۳. اگر خط، آخرین بار در سمت راست دیده شده باشد، ساعت‌گرد حرکت کن.
   اگر خط، آخرین بار در سمت چپ دیده شده باشد، پاد ساعت‌گرد حرکت کن.
   مرحله ۳ را تکرار کن تا خط پیدا شود.

٤. به مرحلهٔ یک برو.

مدار بایاس سنسورها: چنانچه سنسورها امواج مادون قرمز را دریافت کنند مقاومت شان کاهش می‌یابد. یک سنسور خوب در حضور نور مادون قرمز مقاومتش صفر

و در غیاب آن مقاومتش شدیداً افزایش می‌یابد. همین ویژگی سبب می‌شود تا بتوان خط مشکی را تشخیص داد. رنگ مشکی نسبت به سایر رنگ‌ها میزان بیشتری از نور تابیده شده به آن‌را جذب می کند در نتیجه، سنسور گیرنده، نوری را دریافت نمی‌کند. اما رنگ سفید بیشتر نور تابیده شده به آن‌را باز می‌گرداند.



شکل (۵-۱٤): مدار بایاس سنسورهای فرستنده و گیرنده

در مدار فوق ولتاژ سر منفی آپ امپ برابر است با:

$$v_0 = vcc\frac{(Rsensor)}{Rsensor + R_1} \qquad (۳-۵)$$

هنگامی که سنسور روی خط مشکی باشد نوری بازتاب نمی‌شود بنابراین مقاومت سنسور بی‌نهایت است. در نتیجه: ورودی منفی آپ امپ برابر VCC و چنانچه سنسور روی زمینهٔ سفید باشد نور منعکس شده و مقاومت سنسور تقریباً صفر می شود و ورودی منفی آپ امپ نیز تقریباً برابر صفر خواهد شد. می‌توان پتانسیومتر را طوری تنظیم کرد که هنگامی که سنسور روی خط سیاه است خروجی آپ امپ برابر صفر و هنگامی که سنسور روی پیش زمینهٔ سفید رنگ است خروجی آن ۱ منطقی گردد. مقدار مقاومت R_2 باید به گونه‌ای انتخاب شود که جریان هدایت دیود فرستنده از حد مجاز فراتر نرود و مقدار **۲۷۰Ω** برای آن مناسب است. مقدار مقاومت R_1 بسیار مهم است چون حساسیت سنسور به آن بستگی دارد مقدار مقاومت 10kΩ نیز برای این مقاومت مناسب است. پیشنهاد می‌شود از سنسورهای فرستنده-گیرنده CNY70 استفاده شود. سعی شود که سنسورها را در فاصله 15mm قرار دهید و چنانچه از سنسورهای فرستنده-گیرندهٔ جداگانه استفاده می‌کنید فاصلهٔ دو سنسور باید حدوداً 2.5mm باشد.

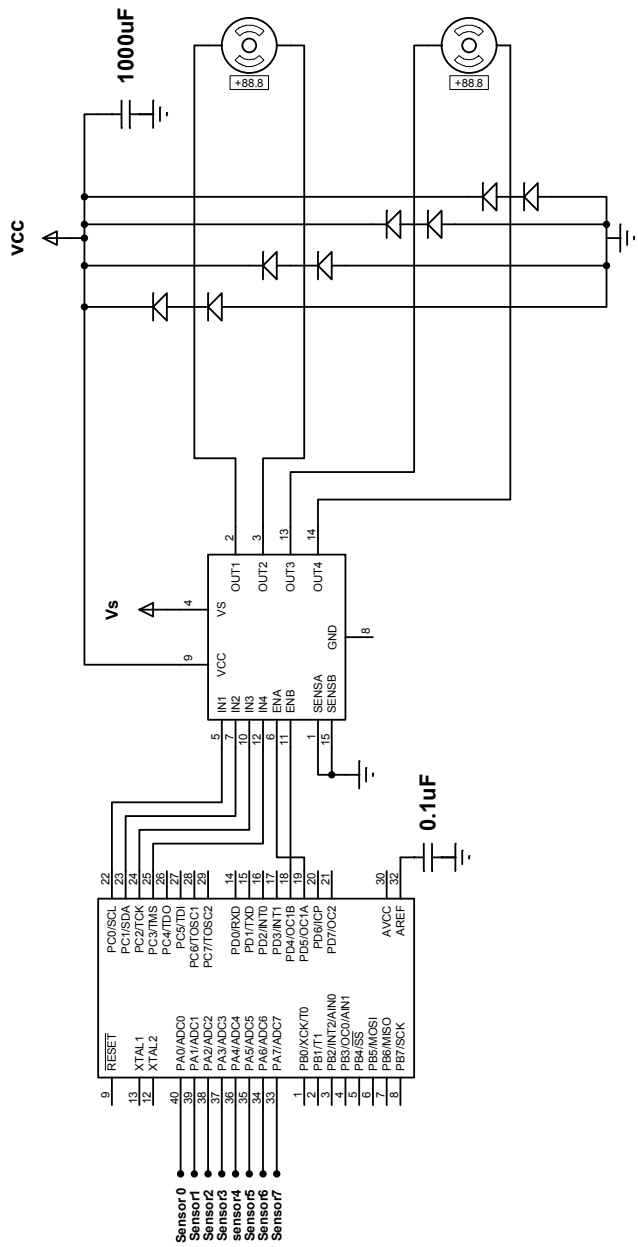
☜ **رابط موتور و مدار کنترل:** از آنجا که میکروکنترلر قادر به تامین جریان مورد نیاز موتورها نیست، بنابراین به منظور راه‌اندازی و کنترل موتورهای DC از یک درایور استفاده می‌شود. این درایور هم جریان مورد نیاز موتور را تامین می‌کند و هم می تواند موتور را در دو جهت راه‌اندازی نماید. برای این عمل دو نوع تراشهٔ معروف وجود دارد که به ترتیب L293D که می‌تواند ولتاژی تا 36V و با حداکثر جریان 1.2A را به موتور اعمال کند و L298 که می‌تواند ولتاژ 49V را با حداکثر جریان 4A به موتور اعمال کند که در این ربات نیز از L298 استفاده شده است. جدول (۵-۱) نحوه‌ی کنترل این موتور را در حالت‌های راست‌گرد، چپ‌گرد، توقف سریع و توقف عادی نشان می‌دهد.

☜ برای **کنترل پیوسته سرعت موتور** می‌توان یک شکل موج متغیر PWM با درصد وظیفه قابل تنظیم به پایهٔ Enable اعمال نمود. با کنترل درصد وظیفه این پالس می‌توان ولتاژ متوسط اعمال شده به موتورهای DC را کنترل نمود و بدین ترتیب سرعت موتورها با تغییر درصد وظیفه به صورت خطی تغییر می‌کند. با درصد وظیفهٔ ۱۰۰٪ می‌توان ولتاژ V_s و با ۵۰٪ می‌توان به ولتاژی حدود نصف V_s را به موتور اعمال کرد. همچنین چهار دیود هرزگرد 1N4004 نیز برای حفاظت از گیت‌های تراشه L298 و کنترل جریان در هنگام قطع هر فاز استفاده می‌شوند.

جدول(۵-۱) نحوه عملکرد درایور L298

ورودی		عملکرد و جهت حرکت
EN1=H	IN1=H ; IN2=L	راست‌گرد
	IN1=L ; IN2=H	چپ‌گرد
	IN1= IN2	ایست سریع
EN1=L	IN1=X ; IN1=L	ایست موتور به آرامی

L=Low H=High بدون اهمیت=X



شکل (۵-۱۵): سخت‌افزار ربات مسیریاب

توضیح برنامه: برای حرکت ربات یک تابع به نام move در نظر گرفته شده است. ورودی‌های این تابع جهت چرخش، میزان تاخیر و سرعت را مشخص می‌کنند. چنانچه متغیر جهت L باشد، ربات چرخ سمت چپ خود را متوقف نموده و چرخ سمت راست را به سمت جلو می‌چرخاند. در نتیجه، ربات به سمت چپ خواهد چرخید و چنانچه متغیر جهت R باشد ربات چرخ راست خود را متوقف نموده، چرخ چپ را به سمت جلو می‌چرخاند در نتیجه ربات به سمت راست می‌چرخد و چنانچه متغیر جهت FWD باشد ربات هر دو چرخ را به سمت جلو می‌چرخاند و ربات به جلوحرکت خواهد کرد و چنانچه این متغیر STOP باشد هر دو چرخ متوقف شده و ربات می‌ایستد. توسط متغیر ورودی تاثیر میزان تداوم حرکت مشخص می‌شود و با متغیر ورودی سرعت، سرعت حرکت ربات تعیین می‌شود. سرعت حرکت ربات بستگی به موقعیت سنسورها دارد که از یک رابطهٔ خطی پیروی می‌کنند.

✓ برنامه

```
/*************************************************
Project : Line Follower
Comments:
Chip type : ATmega16
Clock frequency : 7.372800 MHz
*************************************************/
//#define debug 1
#include <mega16.h>
#include <delay.h>
#ifdef debug
#include <stdio.h>
#endif
#define FWD 0xAA
#define REV 0x55
#define R 0x22
#define L 0x88
#define CW 0x99
#define CCW 0x66
#define STOP 0x00
#define B 0xFF
#define RSPEED OCR1AL
#define LSPEED OCR1BL
#define SPEED0 255
#define SPEED1 0
```

```
#define SPEED2 0
#define SPEED3 0
#define MAX 3
#define HMAX 1
void move (unsigned char dir,unsigned char delay,unsigned char power);
unsigned char i,rdev,ldev,ip, delay, dir, power, dirl, history[MAX],
unsigned char hcount=0,rotpow;
#ifdef debug
unsigned char rep=0,prev=0;
#endif

void main(void)
{
PORTA=0x00;DDRA=0x00;
PORTB=0x00;DDRB=0x00;
PORTC=0x00;DDRC=0xFF;
PORTD=0x00;DDRD=0x30;

// Timer/Counter 0 initialization
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 921.600 kHz
// Mode: Fast PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0xA1;TCCR1B=0x0A;
TCNT1H=0x00;TCNT1L=0x00;
ICR1H=0x00;ICR1L=0x00;
OCR1AH=0x00;OCR1AL=0xFF;
OCR1BH=0x00;OCR1BL=0xFF;

// Timer/Counter 2 initialization
ASSR=0x00; TCCR2=0x00;
TCNT2=0x00;OCR2=0x00;
```

```
// External Interrupt(s) initialization
MCUCR=0x00;MCUCSR=0x00;

#ifdef debug
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 57600
UCSRA=0x00;UCSRB=0x18;
UCSRC=0x86;UBRRH=0x00;
UBRRL=0x07;
#endif

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
ACSR=0x80;SFIOR=0x00;

while (1)
{
#ifdef debug
if(rep<255) rep++;
if(prev!=PINA)
  {
   prev=PINA;
   printf("%u\r",rep);
   for(i=0;i<8;i++)
   printf("%u\t",(prev>>i)&0x01);
   rep=0;
  }
#endif
if(PINA!=255)
  {
   rotpow=255;
   ldev=rdev=0;
  if(PINA.3==0) rdev=1;
  if(PINA.2==0)  rdev=2;
  if(PINA.1==0)  rdev=3;
```

```
  if(PINA.0==0)  rdev=4;
  if(PINA.4==0)  ldev=1;
  if(PINA.5==0)  ldev=2;
  if(PINA.6==0)  ldev=3;
  if(PINA.7==0)  ldev=4;
  if(rdev>ldev) move(R,0,195+12*rdev);
  if(rdev<ldev) move(L,0,195+12*ldev);
  if(rdev==ldev) move(FWD,0,200);
 }
else
 {
   for(i=0,dirl=0;i<MAX;i++)
     {
       if(history[i]==L) dirl++;
     }
if(rotpow<160) rotpow=160;
if(rotpow<255) rotpow++;
if(dirl>HMAX)  {move(CW,0,rotpow);}
else {move(CCW,0,rotpow);}
   }
 };
}

void move (unsigned char dir,unsigned char delay,unsigned char power)
{
PORTC=dir;
if(dir==L || dir==R)
    {
      hcount=(hcount+1)%MAX;
      history[hcount]=dir;
    }
LSPEED=RSPEED=255;//power;
}
```

فصل ششم مبدل آنالوگ به دیجیتال

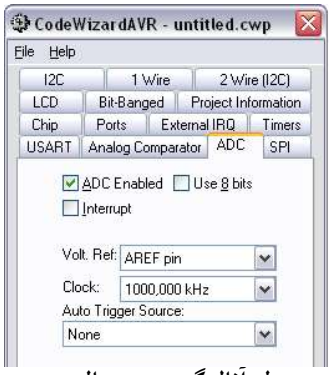
در این فصل، طراحی و پیاده‌سازی چند پروژه کاربردی توسط مبدل آنالوگ به دیجیتال بیان می‌شوند و مطابق ساختار کلی کتاب، در صورت عدم آشنایی مقدماتی با این قابلیت میکروکنترلر، مطالعه مراجع آورده شده در انتهای کتاب و CD همراه کتاب پیشنهاد می‌شود.

پروژه بیستم‌وسوم: ولت‌متر دیجیتال

✍ برنامه ای بنویسید که ولتاژ آنالوگ ورودی در پایه PA0 را اندازه‌گیری و مقدار آن را تا سه رقم اعشار روی LCD نمایش دهد (دامنه ولتاژ ورودی را بین صفر تا پنج ولت در نظر بگیرید)؟

☑ حل: به منظور پیاده‌سازی پروژه بیان شده، می‌باید در محیط wizard از نرم‌افزار CodeVision تنظیمات زیر را انجام داد:

- ابتدا مطابق شکل (٦-١) در پنجره باز شده: گزینه Adc Enable را فعال می کنیم. سپس در قسمت .Volt. Ref گزینه Avcc pin انتخاب می‌شود و سایر مراحل شامل اضافه کردن LCD و نظایر آن را انجام داده و پروژه را ایجاد می نمائیم.



شکل (٦-١): پیکربندی مبدل آنالوگ به دیجیتال در محیط CodeWizard

معرفی توابع استفاده شده در برنامه :

unsigned char read_adc( )

❶ ورودی این تابع شماره کانال‌های ADC مورد استفاده می‌باشد که عددی بین صفر تا هفت است و خروجی آن کد تبدیل یافته‌ی متناظر با ولتاژ مورد نظر است.

void ftoa(float n, unsigned char decimals, char *str)

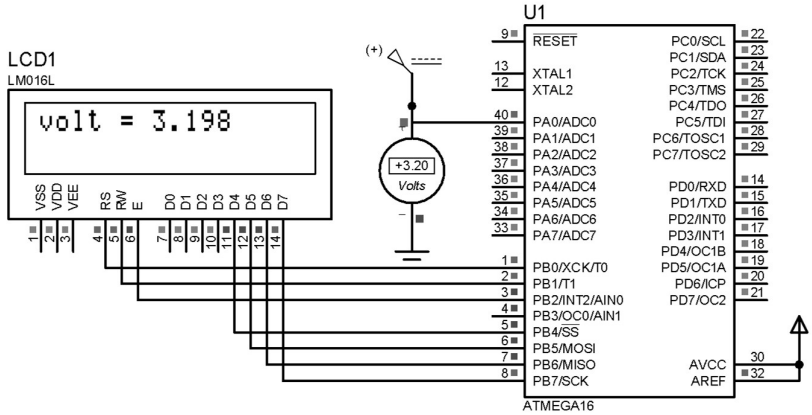
❷ این تابع متغیر اعشاری n را تا decimal رقم اعشار به رشته تبدیل کرده و در متغیر str قرار می‌دهد (آرگومان دوم تابع نشان دهنده تعداد رقم اعشار است).

➢ **توجه:** از آنجایی که LCD نمی‌تواند مقادیر اعشاری را نشان دهد از تابع ftoa() برای تبدیل عدد اعشاری به کاراکتر استفاده شده است. (LCD قادر است متغیرهایی از نوع char را نمایش دهد). در این برنامه متغیر f تا ۳ رقم اعشار به کاراکتر تبدیل شده و در متغیر d قرار می‌گیرد. با توجه به این امر که ولتاژ مرجع مبدل آنالوگ به دیجیتال به صورت داخلی به ولتاژ پنج ولت وصل شده است. بنابراین: در صورتی که بر روی کانال صفرم از ADC یعنی PA.0 ولتاژ پنج ولتی قرار گیرد مبدل آنالوگ به دیجیتال مقدار ۱۰۲۳ را باز می‌گرداند (زیرا مبدل آنااوگ به دیجیتال ۱۰ بیتی است و ولتاژ ورودی را به اعداد صفر تا ۱۰۲۳ تبدیل می‌نماید) و با همین نسبت سایر ولتاژها تقسیم می‌شوند (به عنوان مثال: ولتاژ ۳ ولت پس از تبدیل توسط مبدل آنالوگ به دیجیتال به عدد: ٦١٤=٥/(٣×١٠٢٣) تبدیل می‌شود و برای نمایش بر روی LCD ضروری است عکس عمل تبدیل انجام شود، یعنی عدد ٦١٤ در مقدار ٥/١٠٢٣=٠/٠٠٤٨٨ ضرب شود). مدار شبیه‌سازی شده توسط نرم‌افزار Proteus در شکل (٦-٢) نشان داده شده است.

✓ **برنامه**

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#asm
 .equ __LCD_port=0x18 ;PORTB
#endasm
#include <LCD.h>
#define ADC_VREF_TYPE 0x40
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
```

```
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
void main(void)
{
unsigned int a;
char aray[32],d[10]; float f;
PORTB=0x00;DDRB=0xFF;
ACSR=0x80;SFIOR=0x00;
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;
SFIOR&=0xEF;
lcd_init(16);
while (1)
    {
    a = read_adc(0);  f = a;
    f = f * 0.0048828125;
    ftoa(f,3,d);
    sprintf(aray,"volt = %s",d);
    lcd_clear();
    lcd_puts(aray);
    delay_ms(500);
    };
}
```



شكل (٦-٢): مدار ولت‌متر ديجيتال

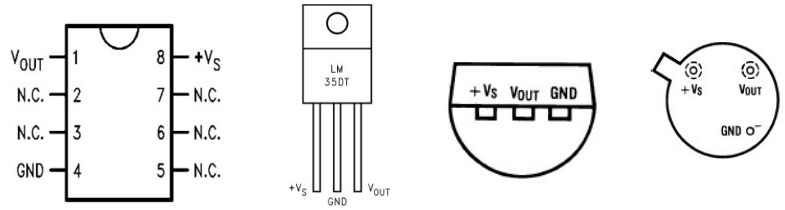
پروژه بیست‌وچهارم: اندازه‌گیری دما با استفاده از سنسور LM35

الف: برنامه‌ای بنویسید که دمای محیط را با استفاده از سنسور دمای LM35 اندازه گیری کند و نتیجه را بر روی LCD نمایش دهد؟

حل: خروجی اکثر سنسورهای اندازه‌گیری دما، ولتاژی است که متناسب با دمای بدنه آن‌ها می‌باشد. در سنسور LM35 ولتاژ خروجی آن به ازای هر یک درجه سانتی‌گراد ده میلی ولت تغییر می‌کند. دو نوع بسته‌بندی این سنسور در شکل (٦-٣) ترسیم شده است. در این پروژه نحوه‌ی اتصال یک LM35 به میکروکنترلر و نحوه‌ی نمایش دمای محیط بر حسب درجه بر روی LCD تشریح شده است.

شرح پروژه: با توجه به این‌که دامنه ولتاژ خروجی این سنسور در دمای 100^0 C برابر یک ولت می‌باشد برای این‌که دماسنج مورد نظر، حداکثر دقت اندازه‌گیری را داشته باشد ولتاژ مرجع A/D (پایه Aref) را به ولتاژ کاملاً ثابت یک ولتی متصل می‌نمائیم. در این صورت: دقت تبدیل آنالوگ به دیجیتال 1/1000 محدوده کامل دماسنج که همان صد درجه است خواهد شد. دقت اندازه‌گیری و نمایش دما با این دماسنج در دمای محیط($T_A=25^0$) برابر 0.50 -/+ خواهد بود.

برنامه مورد نیاز برای قرائت ولتاژ و نمایش دما مشابه برنامه ولت‌متر می‌باشد. کافی است پس از خواندن ولتاژ ورودی با استفاده از رابطه مربوطه، دمای سنسور محاسبه و نمایش داده شود.



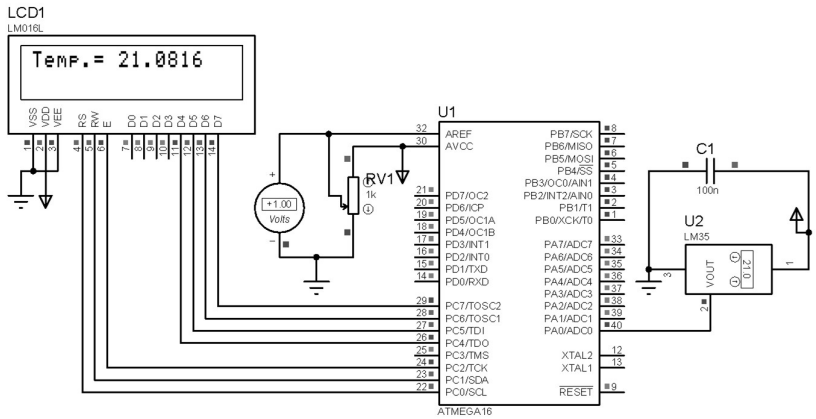
شکل(٦-٣): انواع بسته‌بندی سنسورهای دما LM35

✓ برنامه

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
// Alphanumeric LCD Module functions
#asm
 .equ __LCD_port=0x15  //PORTC
#endasm
```

```
#include <LCD.h>
#define ADC_VREF_TYPE 0x00
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

// Declare your global variables here

void main(void)
{
unsigned int a;
char aray[32],d[10];
float f;
PORTB=0x00;DDRB=0xFF;
ACSR=0x80;SFIOR=0x00;
// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;
SFIOR&=0xEF;
// LCD module initialization
lcd_init(16);
while (1)
      {
      a = read_adc(0);f = a;
      f = f * 0.000976*100;  //f*1*100/1024
      ftoa(f,4,d);
      sprintf(aray,"Temp.= %s",d);
      lcd_clear();  lcd_gotoxy(0,0);
 lcd_puts(aray);
```

```
    delay_ms(500);
    };
}
```



شکل (٦-٤): اتصال دماسنج LM35 به میکروکنترلر و نمایش‌گر LCD

✍ ب- برنامه قسمت الف را طوری تغییر دهید که با استفاده از سنسور LM35 دماهای منفی نیز اندازه‌گیری شوند به طوری که رنج دمایی ^{0}۳۰- تا ^{0}۱۵۰+ قابل اندازه‌گیری باشد؟

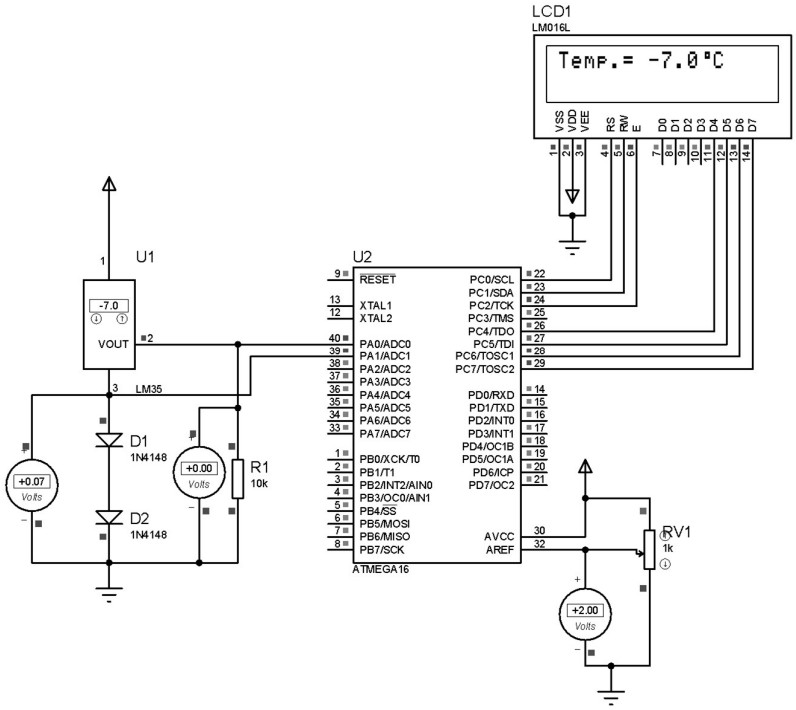
☑ **حل:** با توجه به این که سنسور LM35 دماهای ۵۵- تا ۱۵۰ را اندازه‌گیری می‌کند، بنابراین برای محاسبه دماهای منفی مدار شکل (٦-٥) پیشنهاد می‌شود. البته این مدار در عمل رنج دمایی ^{0}۳۰- تا ^{0}۱۵۰+ را اندازه گیری می‌کند.

✓ **برنامه**

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#define ADC_VREF_TYPE 0x00
char array[16], d[6];
float a,b,c;
```

```
flash unsigned char char0[8] = { 0x7, 0x5, 0x7, 0x0, 0x0, 0x0, 0x0,0x0};

void define_char(flash unsigned char *pc,unsigned char_code)
{
unsigned i,a;
a=(char_code<<3) | 0x40;
for (i=0; i<8; i++) lcd_write_byte(a++,*pc++);
}

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

void main(void)
{
PORTA=0x00;DDRA=0x00;PORTB=0x00;DDRB=0x00;
PORTC=0x00;DDRC=0x00;PORTD=0x00;DDRD=0x00;
// ADC initialization
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x83;

// LCD module initialization
lcd_init(16);
define_char(char0,0);
while (1)
    {
     a=read_adc(0);b=read_adc(1);
    c=(a-b);c=c/5;
```

```
    ftoa(c,1,d);
    sprintf(array,"Temp.= %s",d);
    lcd_clear(); lcd_gotoxy(0,0);
    lcd_puts(array);lcd_putchar(0);lcd_putchar('C');delay_ms(200);
    };
}
```



شکل (٦-٥): اتصال دماسنج LM35 به میکروکنترلر و نمایش‌گر LCD به منظور نمایش رنج دمایی -30^0 تا $+150^0$

فصل هفتم پورت سریال

در بیشتر مدارهای میکروکنترلری، پردازنده مرکزی نیازمند دریافت اطلاعات از محیط خارج است. ممکن است این داده‌ها از منبع داده (غیر انسانی) ارسال شوند و یا ممکن است توسط کاربر (انسانی) ارسال شوند. در صورتی که، داده‌ها توسط هر یک از منابع فوق ارسال شوند، پردازنده مرکزی پس از دریافت آن‌ها، عملیات مورد نظر را بر روی آن‌ها انجام می‌دهد. یکی از منابع ارسال و دریافت داده پورت سریال در میکروکنترلر است که به اختصار USART[1] نامیده می‌شود.

در این فصل: ابتدا پورت سریال معرفی، سپس توابع مورد نیاز به منظور راه‌اندازی، ارسال و دریافت داده از پورت سریال بیان می‌شوند و در ادامه، چند پروژه کاربردی معرفی، بحث و بررسی خواهند شد.

۷-۱- ارتباط سریال USART

ارتباط سریال USART یکی از پروتکل‌هایی است که توسط انواع کامپیوترها حمایت می‌شود. بنابراین برای برقراری ارتباط بین میکروکنترلرها و کامپیوترها از این روش استفاده می‌شود و به این دلیل مطالعه‌ی آن از اهمیت زیادی برخوردار است. بخش ارتباط سریال USART در میکروکنترلرها AVR، قابلیت‌های متنوعی دارد که از جمله می‌توان به موارد زیر اشاره کرد.

۱. عملکرد Full Duplex (رجیسترهای سریال مستقل برای ارسال و دریافت)
۲. عملکرد همزمان (سنکرون) و غیرهمزمان (آسنکرون)
۳. عمل به صورت Master و Slave
٤. تولیدکننده نرخ ارسال دقیق
۵. حمایت از قاب‌های با ارسال ۵، ٦، ۷، ۸ یا ۹ بیت داده و ۱ یا ۲ بیت توقف.
٦. تولید پریتی به صورت زوج یا فرد و امکان چک کردن سخت افزاری آن.
۷. تشخیص خطای سرریز

[1] Universal Synchronous-Asynchronous Receive Transmit (USART)

۸. تولید سه وقفه‌ی مجزا برای ارسال TX، خالی شدن رجیستر داده‌ی TX و اتمام دریافت RX.
۹. کار در ارتباط چند پردازنده.
۱۰. امکان دو برابر کردن سرعت در حالت آسنکرون.

۷-۲- سازگاری USART با UART

بعضی از انواع AVR تنها از ارتباط سریال UART حمایت می‌کنند، به این معنی که ارتباط سریال تنها به صورت غیرهمزمان یا به اصطلاح آسنکرون قابل انجام می‌باشد و انواع پیشرفته‌تر میکروکنترلر AVR می‌توانند به صورت سنکرون و آسنکرون ارتباط برقرار کنند این دو نوع ارتباط سریال از نظر محل بیت‌ها در داخل رجیسترها، نحوه‌ی تولید ارسال، نرخ ارسال و دریافت اطلاعات، همچنین عملکرد بافر مربوط به ارسال اطلاعات کاملاً مطابقت دارند و تنها عملکرد بافر مربوط به ارسال اطلاعات در ارتباط USART بهبود یافته است.

۷-۳- تولید کننده‌ی نرخ ارسال داخلی

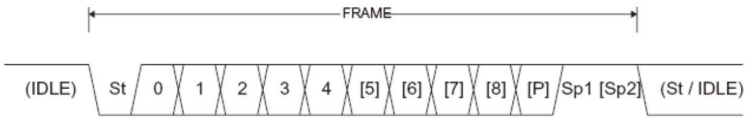
واحد ارتباط سریال، یک تولیدکننده‌ی کلاک داخلی را شامل می‌شود که از آن در حالت آسنکرون و در حالت سنکرون در مد Master استفاده می‌کند. در این واحد یک شمارنده یا به اصطلاح (کانتر) وجود دارد که به صورت کاهشی می‌شمارد. این کانتر کلاک خود را به صورت مستقیم از کلاک سیستم می‌گیرد. هر بار که محتوای این کانتر به صفر می‌رسد، یک کلاک تولید می‌شود و محتوای کانتر از رجیستر UBRR بار می‌شود. سپس وابسته به مد عملکرد ارتباط سریال، کلاک تولید شده بر ۲، ۸ یا ۱۶ تقسیم می‌گردد که در قسمت‌های مختلف از آن استفاده می‌شود. جدول (۷-۱) روابط لازم برای محاسبه «نرخ ارسال»[1] (بر حسب بیت بر ثانیه) و محاسبه‌ی مقدار UBRR را وابسته به مد عملکرد ارتباط سریال نشان می‌دهد که در آن BAUD نرخ ارسال (بر حسب تعداد بیت‌ها در یک ثانیه bps)، فرکانس کلاک اسیلاتور سیستم و UBRR محتوای رجیسترهای UBRRH، UBRRL تنظیم می‌شوند.

جدول (۷-۱): روابط لازم برای برای تنظیم رجیستر UBRR

[1] Baud Rate

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).



St Start bit, always low.

(n) Data bits (0 to 8).

P Parity bit. Can be odd or even.

Sp Stop bit, always high.

IDLE No transfers on the communication line (RxD or TxD). An IDLE line must be high.

شکل (۷-۱): نمودار زمانی قاب داده سریال

۷-٤- قاب داده

در یک قاب[1] اطلاعاتی که توسط بیت شروع و بیت پایان محصور شده است، معمولاً ۵ تا ۹ بیت داده قرار می‌گیرند و یک بیت توازن نیز به صورت اختیاری تعریف می‌شود. بیت شروع متناظر با صفر منطقی است و بیت پایان که ممکن است ۱ یا ۲ بیت باشد با مقدار منطقی یک شناسایی می‌شود. به عنوان مثال: در نمودار زمانی شکل (۷-۱) یک قاب بیت است که ۵، ٦، ۷، ۸ یا ۹ بیت آن شامل داده، یک بیت آغازین، یک بیت پایانی و یک بیت توازن قبل از بیت پایان می‌باشد.

با توجه به ساختار کتاب، از بیان جزئیات بیشتر در مورد پورت سریال صرفنظر نموده و در صورت عدم آشنایی شما با سایر تنظیمات، مراجع آخر کتاب پیشنهاد می‌شود.

[1] Frame

۷-۵- توابع پورت سریال

char getchar(void)
از این تابع برای خواندن یک کاراکتر از پورت سریال استفاده می‌شود.

void putchar(char c)
از این تابع برای ارسال کاراکتر c از طریق پورت سریال استفاده می‌شود.

void puts(char *str)
از این تابع برای ارسال رشته‌ی s که در SRAM وجود دارد، استفاده می‌شود.

void putsf(char flash *str)
این تابع مشابه تابع قبلی است، با این تفاوت که رشته‌ی s در حافظه ی Flash قرار دارد.

void printf(char flash *fmtstr [arg1, arg2,...])
از این تابع برای ارسال یک جمله با فرمت مشخص به پورت سریال استفاده می‌شود. فرمت پارامترها به شکل زبر است:

%[flags][width][.precision] [l]type_char

Flags ها عبارتند از :

'+' : این پرچم باعث می‌شود تا اعداد مثبت با علامت '+' چاپ شوند.

' ' : این پرچم باعث می‌شود تا اعداد مثبت با کاراکتر space و اعداد منفی با علامت '-' شروع شوند.

'-' : اگر طول چاپ معلوم باشد و طول عدد کمتر از طول چاپ باشد و از این علامت استفاده کنید زیرا باعث می‌شود تا به تعداد کافی کاراکتر space در سمت راست عدد قرار بگیرد و چاپ شود.

Width : حداقل کاراکترهای چاپی تعیین می‌شوند که به دو صورت است:

n : برای کاراکترهای کوچک‌تر، به تعداد کافی از کاراکتر space استفاده می‌کند.

on : برای کاراکترهای کوچک‌تر به تعداد کافی از کاراکتر ۰ استفاده می‌کند.

Precision : برای تعیین تعداد رقم های اعشاری استفاده می‌شود.

type_char : برای تعیین آرگومان چاپ شونده استفاده می‌شود.

'd' : عدد صحیح علامت‌دار

'u' : عدد صحیح بدون علامت

'i' : عدد صحیح علامت‌دار

'e' : نمایش علمی اعداد

'E' : نمایش علمی اعداد
'f' : برای اعداد اعشاری
'x' : نمایش به صورت هگزا دسیمال با حروف کوچک
'X' : نمایش به صورت هگزا دسیمال با حروف بزرگ
'c' : نمایش کاراکتر
'p' :نمایش رشته‌ی موجود در flash
's' : نمایش رشته‌ی موجود در SRAM
'٪' : نمایش کاراکتر

- **توجه:** برخی از flag ها غیر فعال هستند. برای فعال کردن آن‌ها گام های زیر را بردارید.

مسیر زیر را در نرم‌افزار CodeVision دنبال کنید:

project / configure / ccompiler

در صفحه‌ی ccompiler به قسمت (s)printf features بروید؛ حال گزینه‌ی مناسب را برای تنظیم این تابع انتخاب کنید:

Int : X,x,d,u,p,s,c,% و flag های '+' و ' ' پشتیبانی می‌شوند.
Int,with : علاوه بر موارد قبلی flag های 'on','-',with نیز حمایت می‌شوند.
With,long : مشابه قبلی
Long,with,prescision : علاوه بر موارد فوق prescision نیز حمایت می‌شود.
Float,with, prescision : از تمام flag ها حمایت می‌شود.

❖ **مثال:**

```
Int k=36;
Printf(“ number of the package is %+I “ , k )
```

خروجی: number of the package is 36

```
Char flash  float *fmt=” the height is approximately %05.2 f meter “
Float a= 12.3567 ;
Printf (fmt,a);
```

خروجی: the height is approximately 012.35

```
char *gets(char *str, unsigned char len)
signed char scanf(char flash *fmtstr [ , arg1 address, arg2 address,...])
```

این تابع ورودی را با توجه به فرمت بندی‌های موجود می‌خواند و در متغیرهای موجود قرار می‌دهد.

➢ توجه:

```
Int a;
Scanf("%i",&a);
```

اگر متغیر از نوع آرایه باشد علامت & لازم به ذکر نیست.

```
Char array[31]
Scanf (" %s " , array);
```

برای تنظیم پارامتر های تابع ()scanf از مسیر زیر را دنبال کنید:

project/configure/ccompiler

سپس به قسمت ()scanf(s) بروید و مطابق بالا تنظیمات لازم را انجام دهید.

پروژه بیست‌وپنجم: نمایش کاراکترهای دریافتی از پورت سریال بر روی LCD

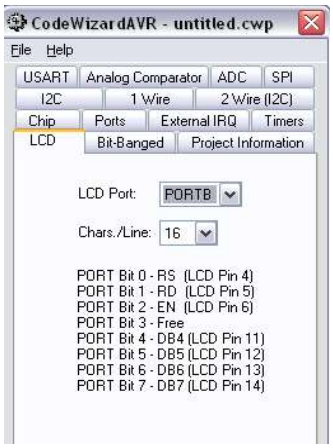
✍ برنامه‌ای بنوسید که توسط آن، کاراکترهای دریافتی از پورت سریال میکروکنترلر بر روی LCD نمایش داده شوند؟

فرضیات پروژه:

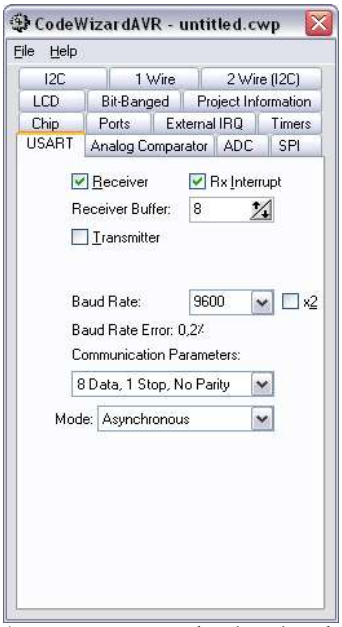
۱. میکروکنترولر مورد نظر ATmega16 است و از کریستال 8MHz استفاده شده که به صورت خارجی به میکروکنترلر متصل شده است.
۲. LCD به پورت B متصل است و USART به منظور ارسال و دریافت داده از پورت D استفاده می‌کند.
۳. از وقفه پورت سریال دریافت داده استفاده می‌شود.
٤. نرخ بیت ۹٦۰۰ بیت بر ثانیه، ۸ بیت داده، ۱ بیت پایان و بدون بیت توازن.
۵. از کتابخانه‌های CodeVision استفاده شده است.

☑ حل: با توجه به این‌که: ممکن است در تنظیمات CodeWizard ابهاماتی وجود داشته باشد، برای این منظور در این پروژه با یادآوری تنظیمات، نسبت به برنامه‌نویسی اقدام می‌نمائیم.

۱. مراحل ۱، ۲ و ۳ از بخش ۱-۳-٤ از فصل اول را انجام دهید.
۲. مطابق شکل (۷-۲) LCD را پیکربندی نمائید و مطابق شکل (۷-۳) پورت سریال را برنامه‌ریزی نمائید و کدهای برنامه را تولید و ایجاد نمائید.



شکل (۷-۲): پیکربندی LCD



شکل (۷-۳): پیکربندی پورت سریال

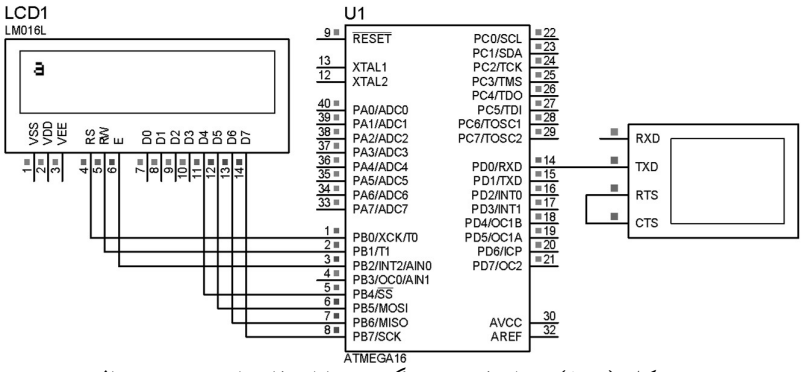
✓ برنامه

```
/*****************************************************
Chip type           : ATmega16
```

```
Clock frequency     : 8,000000 MHz
*************************************************/
#include <mega16.h>
#include <stdio.h>
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm

#include <lcd.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

bit rx_buffer_overflow;

interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data,str[3];
status=UCSRA;
data=UDR;
```

```
lcd_gotoxy(0,0);
sprintf(str,"%c",data);
lcd_puts(str);
if ((status & (FRAMING_ERROR | PARITY_ERROR |
DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    };
  };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+

char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

void main(void)
{
PORTB=0x00; DDRB=0x00;
PORTD=0x00; DDRD=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
```

```
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x90;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;

lcd_init(16);

#asm("sei")

while (1)
    {
    };
}
```



شکل (۷-٤): مدار فرستنده-گیرنده با استفاده از پورت سریال

پروژه بیست‌وششم: راه‌اندازی ماژول فرستنده- گیرنده بی سیم توسط USART

برنامه‌ای برای استفاده از ماژول فرستنده بی‌سیم بنویسید که در آن کاربر با استفاده از چهار کلید فشاری، چهار عدد کد شده را به سمت گیرنده بی‌سیم که در ۱۵ متری

از آن قرار دارد ارسال کند. همچنین، برنامه‌ای برای گیرنده مربوطه بنویسید که داده دریافتی از فرستنده را کدگشایی کند و سپس عدد مربوط به کلید فشرده شده را بر روی LCD نمایش دهد؟

☑ **حل:** این پروژه کاربرد زیادی به منظور ارسال و دریافت داده‌ها توسط کانال بی‌سیم دارد. با توجه به این مهم که نویز، تداخل، محوشدگی و اعوجاج از عوامل کاهش کیفیت سیگنال و تخریب آن در کانال بی‌سیم می‌باشند، استفاده از دو رهیافت **افزایش توان ارسالی** و **کدگذاری** داده منجر به افزایش کیفیت و امنیت داده‌های دریافتی در گیرنده می‌شوند. همچنین با توجه به استفاده اشتراکی از کانال بی‌سیم توسط کاربران زیاد، تداخل از مهمترین عوامل کاهش کیفیت سیگنال دریافتی توسط گیرنده است و به منظور کاهش این اثر: کدگذاری داده‌های ارسالی امری ضروریست. برای طراحی و پیاده‌سازی این پروژه، فرستنده و گیرنده هر یک به طور جداگانه بحث می‌شوند. شایان ذکر است که این پروژه با استفاده از ماژول‌های فرستنده و گیرنده ASK[1] که در بازار با قیمت نسبتاً مناسبی موجود بوده، پیاده‌سازی شده و صحت عملکرد آن تا ۳۰ متر آزمایش شده است.

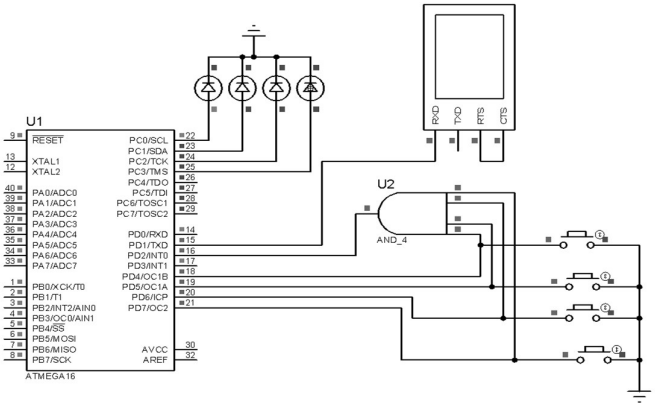
فرستنده: شکل (۷-۵) مدار فرستنده را نشان می‌دهد که در آن کاربر با استفاده از چهار کلید فشاری به ارسال عدد مطلوب خود، اقدام می‌نماید. جهت ارسال هر یک از اعداد مربوط به کلیدهای فشاری از پورت سریال (USART) میکروکنترلر ATmega16 استفاده شده است که نرخ بین فرستنده و گیرنده ۱۲۰۰ بیت بر ثانیه تنظیم شده است. به این ترتیب که با فشردن هر یک از کلیدها، کد مربوطه شامل: تعدادی کاراکتر از جمله کاراکتر رمز و شماره کلید به صورت سریال روی پورت سریال میکرو که به پایه داده ماژول فرستنده متصل می‌باشد قرار گرفته و برای اطمینان از ارسال و دریافت صحیح، داده در یک حلقه‌ی تکرار، پنج بار ارسال می شود. برای ارسال داده‌های ارسالی به صورت رشته در هنگام فشرده‌شدن یکی از کلیدهای فرمان، میکروکنترلر باید به صورت وقفه‌ای عمل کند و به دلیل این‌که تعداد پایه‌های وقفه خارجی موجود در میکرو محدود است، از تراشه ١٤ پایه 7408 که شامل چهار گیت AND است، استفاده می‌شود. همان‌طور که در شکل (۷-۵) مشاهده می‌شود، هر یک از کلیدها از قسمتی که زمین نشده، یک بار به ورودی AND و یک‌بار به یکی از پایه‌های پورت D میکرو (که با مقاومت داخلی Pull-up شده‌اند) متصل شده و خروجی AND نیز به پایه وقفه خارجی یک میکرو وصل می‌شود. به این

[1] Amplitude Shift Modulation (ASK).

ترتیب، با فشـردن هر کدام از کلیدها، برنامه وارد روتین وقفه شـده و با بررسـی پایه‌های پورت D متصل به کلیدها، کلید فشرده شده تشخیص داده و کد مربوط به آن ارسـال می‌شـود. همان‌طور که شـکل‌های (۷-٦) تا (۷-۹) نشـان می‌دهند، رمز این فرستنده 06#* است و عدد بعد از آن شماره کلید فشرده شده توسط کاربر اسـت که بر روی پورت سـریال میکرو قرار می‌گیرد و به ماژول فرسـتنده ارسـال می‌شـود. ماژولی که برای ارسـال داده به کار رفته ماژول FS1000A با مدولاسیون دیجیتال از نوع ASK در باند 315MHz می‌باشد که در شکل (۷-۱۰) نشان داده شده است. این فرستنده RF ارزان‌قیمت می‌تواند برای ارسال داده (تا حدود ۱۰۰ متر بدون وجود مانع) استفاده شود که برای استفاده در فاصله‌های کوتاه مناسب است. لازم به ذکر است طراحی آنتن، محیط کاری و منبع ولتاژ بر مسافت ارسال تاثیر گذارند (ماژول‌های دیگر با استفاده از مدولاسیون دیجیتال FSK به دلیل ماهیت انتقال داده در فرکانس از کیفت بالاتری برخوردارند. همچنین، به دلیل پیچیدگی سـاخت، قیمت بیشـتری نسـبت به ماژول‌های ASK دارند). ماژول فرسـتنده دارای ۳ پایه اسـت که مطابق شـکل (۷-۱۰) از سـمت راست: Data، VCC(2.5-12V) و Gnd همچنین یک پایه برای آنتن وجود دارد که بهتر است از یک سیم با طول بیش از ۳۰ سانتیمتر استفاده شود. برای کوتاه شدن ابعاد فرستنده، سیم آنتن را به صورت مارپیچ قرار دهید. مشخصات ماژول فرستنده بی‌سیم ASK در جدول (۷-۲) آورده شده است.

جدول (۷-۲): مشخصات ماژول فرستنده بی‌سیم ASK

Operating Voltage	2.5 V to 12 V
Operating Current	4mA @ 5V, 15mA @ 9V
Quiescent Current	10uA
Operating Temperature	-10C - 60C
Modulation	ASK
Max. Data Rate	9.6K
Data Input	TTL
RF Power	20 mW@5V



شکل (۷–۵): مدار فرستنده (در مدار عملی، پایه TXD میکرو به پایه Data از ماژول فرستنده وصل شود).

***#061*#061*#061*#061*#061**

شکل (۷–٦): داده‌های ارسالی پس از فشرده شدن اولین کلید.

***#062*#062*#062*#062*#062**

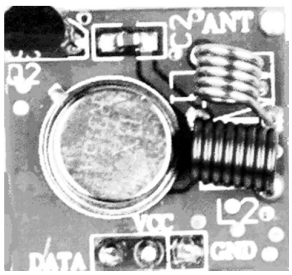
شکل (۷–۷): داده‌های ارسالی پس از فشرده شدن دومین کلید.

***#063*#063*#063*#063*#063**

شکل (۷–۸): داده‌های ارسالی پس از فشرده شدن سومین کلید.

***#064*#064*#064*#064*#064**

شکل (۷–۹): داده‌های ارسالی پس از فشرده شدن چهارمین کلید.



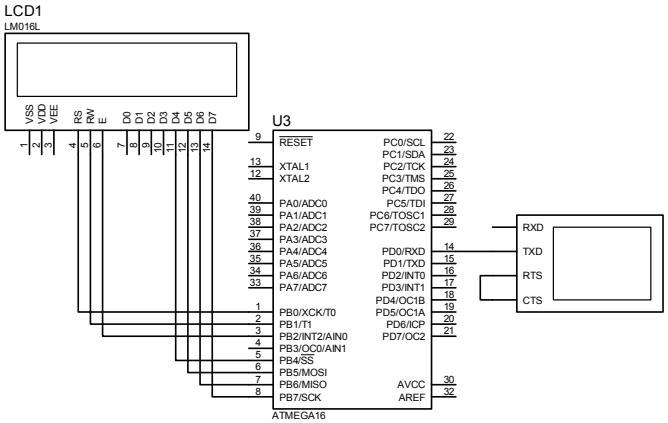
شکل (۷–۱۰): ماژول فرستنده بی‌سیم ASK در باند 315MHz

✓ برنامه فرستنده

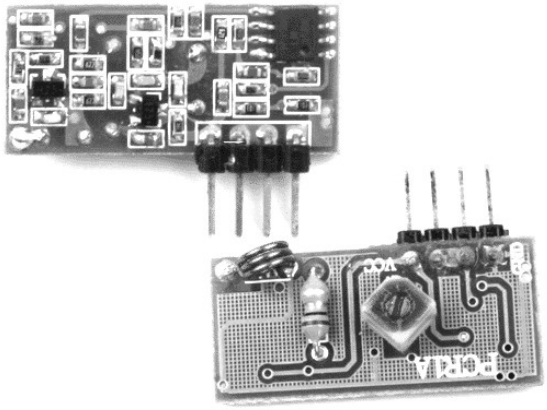
```
/*****************************************************
Chip type           : ATmega16
Clock frequency     : 8,000000 MHz
*****************************************************/
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
char a,b,c,d,j;
 bit i;
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{PORTC=0;delay_ms(10);
i=1;if(PIND.4==0)a=1;
else if(PIND.5==0)b=1;
else if(PIND.6==0)c=1;
else if(PIND.7==0)d=1;
}

void main(void)
{
PORTA=0x00;DDRA=0x00;
PORTB=0x00;DDRB=0x00;
PORTC=0x00;DDRC=0xFF;
PORTD=0xF0;DDRD=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40; MCUCR=0x02;
MCUCSR=0x00; GIFR=0x40;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 1200
```

```
UCSRA=0x00; UCSRB=0x08;
UCSRC=0x86; UBRRH=0x01; UBRRL=0xA0;

#asm("sei")
i=0;
while (1)
    {
    while(i)
    {i=0;
        if(a==1) {for(j=1;j<=5;j++)
                    {printf("*#061");PORTC.0=1;delay_ms(30);a=0;}
                 }
    else if(b==1) {for(j=1;j<=5;j++)
                    {printf("*#062");PORTC.1=1;delay_ms(30);b=0;}
                 }
    else if(c==1) {for(j=1;j<=5;j++)
                    {printf("*#063");PORTC.2=1;delay_ms(30);c=0;}
                 }
    else if(d==1) {for(j=1;j<=5;j++)
                     {printf("*#064");PORTC.3=1;delay_ms(30);d=0;}
                 }
     }
    };
}
```

گیرنده: مدار گیرنده شامل ماژول ASK گیرنده (ماژول گیرنده PCR1A نیز مانند ماژول فرستنده با مدولاسیون دیجیتال ASK و در باند 315MHZ کار می‌کند. این گیرنده ارزان قیمت می‌تواند برای دریافت سیگنال‌های RF ارسالی از فرستنده FS1000A در باند 315MHz استفاده شود)، میکروکنترلر و LCD است که سیگنال‌های دریافتی توسط ماژول گیرنده ابتدا دمدولاسیون شده و از فضای سیگنال به داده‌های دیجیتال تبدیل می‌شوند. سپس به میکروکنترلر تحویل داده می‌شوند. میکروکنترلر توسط برنامه‌ای که نوشته شده است، ابتدا داده‌های دریافتی را کد گشایی (دیکد) کرده و عدد متناظر با کلید فشرده شده را استخراج و بر روی LCD نمایش می‌دهد و سطح ولتاژی پایه مرتبط با کلید فشرده شده را یک می‌کند. بلوک دیاگرام مدار گیرنده در شکل (۷-۱۱) آورده شده، همچنین ماژول گیرنده در شکل (۷-۱۲) ترسیم شده است.



شکل (۷-۱۱): مدار فرستنده (در مدار عملی، پایه RXD میکرو به پایه Data از ماژول فرستنده وصل شود. دقت شود که هر دو پایه وسط شکل (۷-۱۲) پایه Data هستند).



شکل (۷-۱۲): ماژول فرستنده بی‌سیم ASK در باند 315MHz از نماهای پشت و جلو

جدول (۷-۳): مشخصات ماژول فرستنده بی‌سیم ASK

Operating Voltage	4.5V to 5.5V
Operating Current	4mA @ 5V
Operating Temperature	-10C - 60C
Sensitivity	-105dBm
Max. Data Rate	4.8K
Data Output	TTL

✓ برنامه گیرنده

```
/*****************************************************
Chip type           : ATmega16
Clock frequency     : 8,000000 MHz
*****************************************************/

#include <mega16.h>
#include <stdio.h>
#include <delay.h>
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)
// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char index,rx_buffer[RX_BUFFER_SIZE],code[6];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

bit rx_buffer_overflow;

// USART Receiver interrupt service routine
```

```
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
lcd_clear();lcd_gotoxy(0,0);
    if (data=='*'){code[0]=data;index++;}
else if (index==1 && data=='#'){code[index]=data; index++;}
else if (index==2 && data=='0'){code[index]=data; index++;}
else if (index==3 && data=='6'){code[index]=data; index++;}
else if (index==4 && data=='1'){code[index]=data;PORTC=1;
                        lcd_puts(code);index=0;delay_ms(1000);}
else if (index==4 && data=='2')
{code[index]=data;PORTC=2;lcd_puts(code);index=0;delay_ms(1000);}
else if (index==4 && data=='3')
{code[index]=data;PORTC=4;lcd_puts(code);index=0;delay_ms(1000);}
else if (index==4 && data=='4')
{code[index]=data;PORTC=8;lcd_puts(code);index=0;delay_ms(1000);}

if ((status & (FRAMING_ERROR | PARITY_ERROR |
DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    };
  };
}
#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
```

```
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

void main(void)
{
PORTA=0x00;DDRA=0x00;
PORTB=0x00;DDRB=0x00;
PORTC=0x00;DDRC=0xFF;
PORTD=0x00; DDRD=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud Rate: 1200
UCSRA=0x00;UCSRB=0x90;
UCSRC=0x86;UBRRH=0x01;
UBRRL=0xA0;

lcd_init(16);
#asm("sei")
while (1)  { };
}
```

پروژه بیست‌وهفتم: روش ساخت Caller ID تلفن بر اساس استاندارد FSK

برنامه‌ای بنویسید که شماره تلفن دریافتی در استاندارد FSK را بر روی LCD نمایش دهد؟

حل: برای طراحی و پیاده‌سازی این پروژه توضیحات زیر ضروری است.

شرح سطوح ولتاژی تلفن:

۱. ٤٨ ولت DC : هنگامی که از تلفن استفاده‌ای نمی‌شود و گوشی برداشته نشده است، اگر با یک ولت‌متر، ولتاژ پایانه‌های خطوط تلفن را اندازه بگیریم، خواهیم دید که اختلاف ولتاژی به اندازه ٤٨ ولت وجود خواهد داشت.

۲. ۱۲~۲۰ ولت DC : هنگامی که گوشی تلفن را برداریم یا مشغول استفاده و صحبت با تلفن باشیم اختلاف ولتاژی حدود ۱۲ تا ۲۰ ولت بین دو ترمینال خط تلفن وجود خواهد داشت.

پس در صورت استفاده یا عدم استفاده از تلفن، همیشه بین دو پایانه خط تلفن ولتاژ DC وجود خواهد داشت، که می‌توان توسط یک یکسو ساز، یک محدود کننده و یک رگولاتور ولتاژ از آن استفاده کرد. به همین علت است که ما هیچ‌گاه نمی‌بینیم که یک دستگاه تلفن الکترونیکی معمولی برای کار، نیاز به یک منبع ولتاژ خارجی داشته باشد.

۳. ۸۰~۱۲۰ ولت AC : هنگامی که تلفن زنگ می‌خورد، در واقع یک موج تک فرکانس سینوسی ۲۵ هرتز با دامنه ذکر شده روی ولتاژ ٤٨ ولت DC سوار می شود و این سیگنال توسط بخش Ring Detector تلفن، شناسایی و به زنگ تبدیل می‌شود.

سیستم‌های Tone و پالس:

DTMF: این کلمه که خلاصه Dual Tone Multi Frequency می‌باشد و همچنین به نام Touch Tone نیز شناخته می‌شود که در سیگنال‌های تلفنی استفاده می‌شود و در باند فرکانسی صوتی می‌باشد. DTMF نمونه‌ای از MFSK[1] است. قبل از DTMF سیستم های تلفن از سیستمی به نام Pulse dialing برای شماره‌گیری استفاده می کردند که با قطع و وصل سریع خط تلفن شماره گیرنده کار می‌کرد و شبیه خاموش و روشن کردن یک لامپ، که هنگام شماره‌گیری صداهای قطع وصل مشابه کلیک‌های متوالی بود. این سیستم فقط در جاهایی کار می‌کرد که سیم وجود داشت و به صورت بی‌سیم (موبایل) قابل استفاده نبود.

[1] Multi Frequency Shift Keying (MFSK).

DTMF در Bell Labs اختراع شد و توسعه یافت تا امکان شماره‌گیری راه دور مخصوصاً برای سیستم‌های بی‌سیم مانند لینک‌های MicroWave و ماهواره‌ای فراهم شود. Encoder/Decoder هایی به دو طرف خط اضافه شدند که سیگنال‌های استاندارد پالس را به DTMF تبدیل می‌کردند و آن‌ها را وارد خط تلفن به سوی مقصد دور دست ارسال می‌نمودند. در مقصد دور دست یک Encoder/Decoder دیگر Tone ها را Decode می‌کرد و به پالس تبدیل می‌نمود. DTMF برای نمایش هر کلیدی که روی صفحه کلید فشرده می‌شود از دو Tone استفاده می‌کند. وقتی که کلیدی فشرده می‌شود، Tone مربوط به سطر و Tone مربوط به ستون آن کلید تولید می‌شوند. به همین دلیل به آن Dual tone می‌گویند. به عنوان مثال مطابق شکل (۷-۱۳) فشردن "5"، Tone های 770 Hz و 1336 Hz را تولید می‌کند

DTMF Keypad Frequencies				
1	2	3	A	697 Hz
4	5	6	B	770 Hz
7	8	9	C	852 Hz
*	0	#	D	941 Hz
1209 Hz	1336 Hz	1477 Hz	1633 Hz	

شکل (۷-۱۳): فرکانس‌های اختصاصی به صفحه کلید تلفن

فرکانس‌ها به نحوی انتخاب شده‌اند تا مانع از ایجاد هارمونیک‌های دیگر شوند (هیچ فرکانسی ضریب دیگری نیست، اختلاف بین دو فرکانس ضریب هیچکدام از فرکانس‌ها نمی‌باشد و جمع هر دو فرکانس نیز ضریب هیچکدام از فرکانس‌ها نیست). تلورانس فرکانس‌های تولید شده باید حداکثر ٪۱,۵± مقدار نامی هر یک از فرکانس‌ها باشد. کلیدهای A، B، C و D برای سیستم تلفنی Autovon به کار برده می‌شدند که یک سیستم تلفنی در ارتش آمریکا بود تا از حمله‌های هسته‌ای در امان بمانند). هنوز هم با وجود این که این سیستم از منسوخ شده، برخی صفحه کلیدها از این حروف استفاده می‌کنند. سیگنال های بوق اشغال (Busy signal) و بوق شماره‌گیری (Dial tone) و بوق زنگ (Ring back) هریک دارای فرکانس خاص خود هستند که در جدول (۷-٤) آمده است.

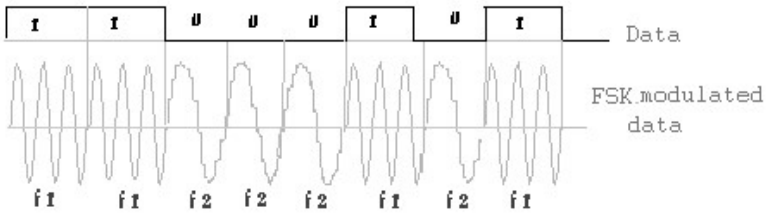
هم برای تولید و هم برای Decode کردن DTMF ، تراشه‌های مربوطه در بازار وجود دارند. با اتصال صفحه کلید به تراشه‌های مربوط به تولید DTMF می‌توان به

راحتی یک سیستم Tone Dialing ساخت؛ همچنین اگر یک Decoder به خط تلفن وصل کنیم، می‌توانیم Tone های مربوط به صفحه کلید (Valid Tones) را آشکار نموده و از آن‌ها استفاده نماییم.

جدول (۷-٤): محدوده فرکانس بوق‌های تلفن

DTMF Event Frequencies		
Event	Low frequency	High frequency
busy signal	480Hz	620Hz
dial tone	350Hz	440Hz
ring back	440Hz	480Hz

- کالر آی‌دی (Caller ID) طرفین یک ارتباط مخابراتی را قادر می‌سازد تا از شماره تماس گیرنده و مخاطب تماس آگاه شوند. اطلاعات تماس گیرنده توسط مخابرات از طریق مدولاسیون FSK[1] بر روی خط ارسال می‌گردد. شکل (۷-۱٤) نشان می‌دهد که ازای بیت یک، فرکانس 1200Hz و به ازای بیت صفر، فرکانس آنالوگ 2200Hz ارسال می‌شود. (استاندارد ۲۰۲ بل) و تمامی اطلاعات تماس گیرنده که حاوی اطلاعات گوناگونی است بر روی یک خط به صورت FSK مدوله و به دستگاه تلفن گیرنده ارسال می‌گردد. هنگامی تلفن زنگ می‌خورد، اطلاعات تماس گیرنده بعد از زنگ اول ارسال می‌شوند. سیگنال زنگ اول به مدت ۲ ثانیه طول می‌کشد و سپس اطلاعات تماس گیرنده در مدت زمان ۱/۵ ثانیه منتقل می‌شوند. که این در طول ۱/۵ ثانیه، ابتدا ۰/۵ ثانیه سکوت (صفر، سیگنال نداریم) و سپس در مدت زمان ۰/۸ ثانیه اطلاعات و پس از آن نیز ۰/۲ ثانیه سکوت و سپس سیگنال زنگ دوم به مدت ۲ ثانیه ارسال می‌شود.



شکل (۷-۱٤): مدولاسیون FSK

اطلاعاتی که در زمان ۰/۸ ثانیه ارسال می‌گردد حاوی بایت‌های زیر است:

[1] Frequency Shift Keying (FSK).

۱. ابتدا یک رشته صفر و یک به صورت متناوب جهت تصرف کانال (Channel Seizure) و فعال نمودن بلوک کالر آیدی ، ارسال می‌شود.
۲. ۱۸۰ بیت یک به عنوان مرحله نشانه گذاری (Mark State) ارسال می‌شود که تعیین کننده آغاز اطلاعات تماس گیرنده است.

بعد از مرحله نشانه‌گذاری ارسال اطلاعات آغاز می‌شود که به ترتیب حاوی بایت های زیر است:

۳. یک بایت که نشان دهنده طول پیغام است، ارسال می‌شود.
٤. اطلاعات ماه، روز، ساعت و دقیقه که هر کدام در دو بایت ارسال می‌شوند.
٥. شماره تلفن ده رقمی در ده بایت ارسال می‌شود.
٦. در نهایت بایت اصلاح خطا یا به عبارتی checksum ارسال می‌شود.

با استفاده از این اطلاعات دستگاه تلفن و یا دستگاه Caller ID مشخصات تماس گیرنده را تشخیص می‌دهد.

برای تبدیل سیگنال‌های دریافتی و دمدولاسیون FSK مربوط به شماره تماس، تراشه‌های زیادی وجود دارند و همه‌ی آن‌ها برای این پروژه مناسب می‌باشند. با توجه به حجم کم کدهای برنامه از تراشه میکرو از خانواده ATtiny استفاده شده است که جزئیات بیشتر آن در برگه‌های اطلاعاتی CD همراه با کتاب آورده شده است. این پروژه با استفاده از چهار تراشه کالر آی‌دی به نام‌های HT9032، MT88E43، SM8223 و MT8841 آزمایش شده است که در ادامه سه برنامه نوشته شده برای تراشه‌های HT9032، SM8223 و MT88E43 همراه با بلوک دیاگرام مربوطه آورده شده است.

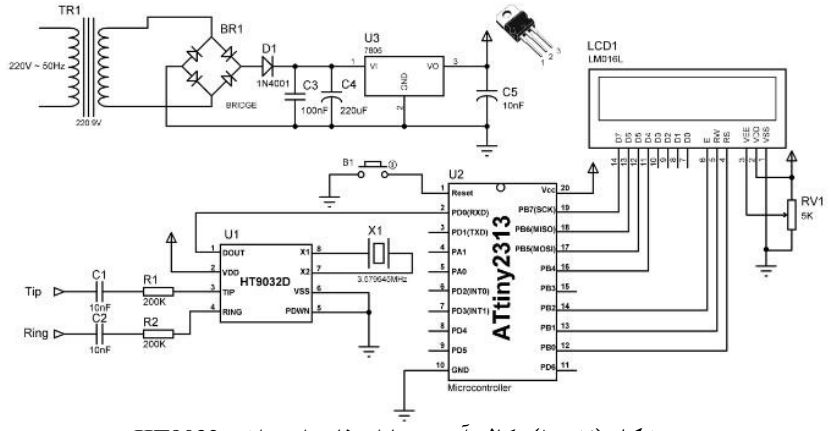
✓ برنامه کالر آی‌دی با استفاده از تراشه HT9032

```
#include <tiny2313.h>
#asm
  .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
#include <stdio.h>
#include <delay.h>
#define xtal 1000000
//-------------------------------------------
unsigned char a,b,d,e,buffer[10];
unsigned int c;
//-------------------------------------------
interrupt [2] void exint0 (void)
{
```

```
 PORTD.4=0;
}
//---------------------------------------------
void main(void)
{
 UCSRA=0x00;
 UCSRB=0x10;
 UCSRC=0x86;
 UBRRH=0x00;
 UBRRL=0x33;
 DDRD=0x10;
 PORTD.4=1;
 GIMSK=0b01000000;
 MCUCR=0b01100000;
 #asm ("sei")
 lcd_init(16);
 lcd_clear();
 lcd_gotoxy(0,0);
 //---------------------------------------------
 do
 {
  c=0;
  do
    {
     a=getchar();
    }while(a!=4);
  c=c+a;
  a=getchar();
  e=a-8;
  if(e>12){e=12;}
  c+=a;
  for(b=0;b<2;b++)
     {
      a=getchar();
      c+=a;
      a=a&0x0f;
      sprintf(buffer,"%d",a);
      lcd_puts(buffer);
     }
  lcd_putsf("/");
  for(b=0;b<2;b++)
```

```
   {
   a=getchar();
   c+=a;
   a=a&0x0f;
   sprintf(buffer,"%d",a);
   lcd_puts(buffer);
   }
lcd_putsf("    ");
for(b=0;b<2;b++)
   {
   a=getchar();
   c+=a;
   a=a&0x0f;
   sprintf(buffer,"%d",a);
   lcd_puts(buffer);
   }
lcd_putsf(":");
for (b=0;b<2;b++)
   {
   a=getchar();
   c+=a;
   a=a&0x0f;
   sprintf(buffer,"%d",a);
   lcd_puts(buffer);
   }
lcd_gotoxy(1,1);
for(b=0;b<e;b++)
   {
   a=getchar();
   c+=a;
   a=a&0x0f;
   sprintf(buffer,"%d",a);
   lcd_puts(buffer);
   }
a=getchar();
c+=a;
d=(unsigned char)c;
if(d!=0)
   {
   lcd_clear();
   lcd_putsf ("error");
```

```
    }
  delay_ms(40000);
  lcd_clear();
  PORTD.4=1;
  MCUCR=0b01100000;
 }while(1);
}
```



شکل (۷-۱۵): کالر آی‌دی با استفاده از تراشه HT9032

✓ برنامه کالر آی‌دی با استفاده از تراشه SM8223

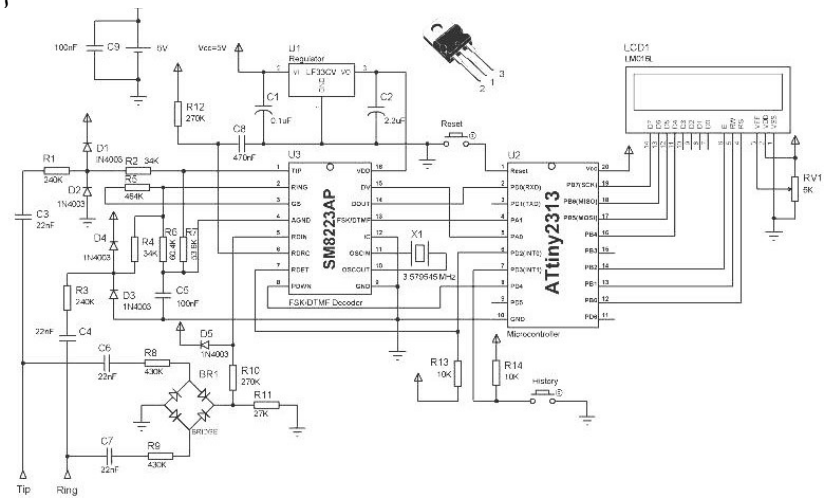
```
#include <tiny2313.h>
#asm
  .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
#include <stdio.h>
#include <delay.h>
#define xtal 1000000
//-------------------------------------------
unsigned char a,b,c=0,e,buffer[10];
//-------------------------------------------
void EEPROM_write(unsigned char Address, unsigned char Data)
{
 while(EECR & 0x02 );
 EEAR =Address;
```

```
 EEDR = Data;
 EECR =0x04;
 EECR =0x06;
}
//---------------------------------------------
unsigned char EEPROM_read(unsigned char Address)
{
 EEAR = Address;
 EECR = 0x01;
 return EEDR;
}
//---------------------------------------------
void print(unsigned char key)
{
 sprintf(buffer,"%d",key);
 lcd_puts(buffer);
}
//---------------------------------------------
unsigned char receive()
{
 a=getchar();
 a=a&0x0f;
 return a;
}
//---------------------------------------------
interrupt [2] void exint0 (void)
{
 PORTD.4=0;
}
//---------------------------------------------
interrupt [3] void exint1 (void)
{
 for (b=0;b<2;b++)
   {
    EEPROM_read(b);
    a=EEDR;
    print(a);
   }
 lcd_putsf ("/");
 for (b=2;b<4;b++)
   {
```

```
   EEPROM_read(b);
   a=EEDR;
   print(a);
  }
 lcd_putsf ("   ");
 for (b=4;b<6;b++)
  {
   EEPROM_read(b);
   a=EEDR;
   print(a);
  }
 lcd_putsf (":");
 for (b=6;b<8;b++)
  {
   EEPROM_read(b);
   a=EEDR;
   print(a);
  }
 lcd_gotoxy(1,1);
 EEPROM_read(30);
 e=EEDR;
 for (b=8;b<e;b++)
  {
   EEPROM_read(b);
   a=EEDR;
   print(a);
  }
 delay_ms(10000);
 lcd_clear();
 c=1;
 }
//---------------------------------------------
void main(void)
{
 UCSRA=0x00; UCSRB=0x10;
 UCSRC=0x86;  UBRRH=0x00;
 UBRRL=0x33;  DDRD=0x10;
 DDRA=0x00;
 PORTD.4=1;
 GIMSK=0b11000000;
 MCUCR=0b01001010;
```

```
#asm ("sei")
lcd_init(16);
lcd_clear();
lcd_gotoxy(0,0);
//-------------------------------------------
do
{
 if(c==1)
    {
    MCUCR=0b01001010;
    PORTD.4=1;  c=0;
    }
 if(!(PINA.0 )&&(PINA.1 ))
    {
    do
    {
     a=getchar();
    }while(a!=4);
    //-----------------
    e=getchar();
    EEPROM_write(30,e);
    e=e-8;
    if(e>12){e=12;}
     for(b=0;b<2;b++)
       {
       receive();
       EEPROM_write(b,a);   print(a);
       }
    lcd_putsf ("/");
    for (b=2;b<4;b++)
       {
       receive();
       EEPROM_write(b,a);   print(a);
       }
    lcd_putsf ("   ");
    for (b=4;b<6;b++)
       {
       receive();
       EEPROM_write(b,a);    print(a);
       }
    lcd_putsf (":");
```

```
    for (b=6;b<8;b++)
      {
       receive();
       EEPROM_write(b,a);  print(a);
      }
    lcd_gotoxy(1,1);
    for (b=0;b<e;b++)
      {
       receive();
       EEPROM_write(b+8,a);   print(a);
      }
    delay_ms(60000);   lcd_clear();
    PORTD.4=1;
    MCUCR=0b01001010;
   }
 }while(1);
}
```



شکل (٧-١٦): کالر آی‌دی با استفاده از تراشه SM8223

✓ برنامه کالر آی‌دی با استفاده از تراشه MT88E43

```
#include <mega8.h>
#include <delay.h>

// Alphanumeric LCD Module functions
#asm
```

```
   .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>

// Declare your global variables here
#define LATCH_OE PORTD.0
#define FSK_EN   PORTD.1
#define RINGING  PIND.3
#define LCD_EN   PORTD.4
#define FSK_DISABLE 0
#define FSK_ENABLE  1
#define LCD_DISABLE 1
#define LCD_ENABLE  0

void init(void);
unsigned char BUF[255];
unsigned char b=0 ;       // buffer counter

// External Interrupt 0 service routine
// this int will call when a data is ready to read from the LATCH output
interrupt [EXT_INT0] void ext_int0_isr(void)
{
 unsigned char i,j;
 LCD_EN=LCD_DISABLE;       // disable LCD
 i=DDRB;j=PORTB;           // save current DDR & PORT value's
 DDRB=0x00;PORTB=0xff;     // change port dir to input
 LATCH_OE=0;               // enalbe latch output's
 #asm("nop\nop\nop\nop\nop") // wait a bit
 BUF[b++]=PINB;            // read data
 if(b>254)b=0;             // prevent from overflow or over pointing
 LATCH_OE=1;               // disable latch output's
 DDRB=i;PORTB=j;           // restore DDR & PORT value's
 LCD_EN=LCD_ENABLE;        // enable LCD
}

void main(void)
{
unsigned char i=0,j=0;
init();
LATCH_OE=1;           // disable latch output's
FSK_EN=FSK_DISABLE;   // disable 8843 FSK demedolation
```

```
LCD_EN=LCD_ENABLE;
lcd_init(16);
lcd_clear();
lcd_putsf("Wait for Ring");
while(1){
	for(i=0;i<254;i++)BUF[i]=0; // fill the buffer with zero
	b=0;
	while(RINGING);  // wait for first ring
	#asm("sei")
	FSK_EN=FSK_ENABLE;
	lcd_clear();
	lcd_putsf("First ring");
	while(~RINGING);

	delay_ms(4000);

	#asm("cli")
	FSK_EN=FSK_DISABLE;
	lcd_clear();
	if(BUF[1]==0x55){
	 for(i=1;i<60;i++){
	  if(BUF[i]!=0x55)break; // 0x55 is chanle seizure
	 }
	 i+=2;
	 j=i+BUF[i]+1;  // BUF[i] contain data LEN.
		// i.e: at my town, SABZEVAR-IRAN it was 20(0x14)
		// this byte recived after a 0x04 witch I think it mean as a
message type indicator.
	 for(i=i+1;i<j;i++)
	  lcd_putchar(BUF[i]);
	}
	else{
	 lcd_putsf("UnKnow CID Data!");
	} // end if(BUF[1]=0x55)
	delay_ms(7000);
  }  // end while(1)
}  // end main

void init(void){
PORTB=0x00; DDRB=0x00;
PORTC=0x00; DDRC=0x00;
```

```
PORTD=0x0D; DDRD=0x13;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: Off
GICR|=0x40;
MCUCR=0x03;
GIFR=0x40;
ACSR=0x80;
SFIOR=0x00;
}
```

به منظور مشاهده شکل مدار به CD کتاب مراجعه نمائید.

پروژه بیست‌وهشتم: مد چند پردازنده ارتباط سریال USART

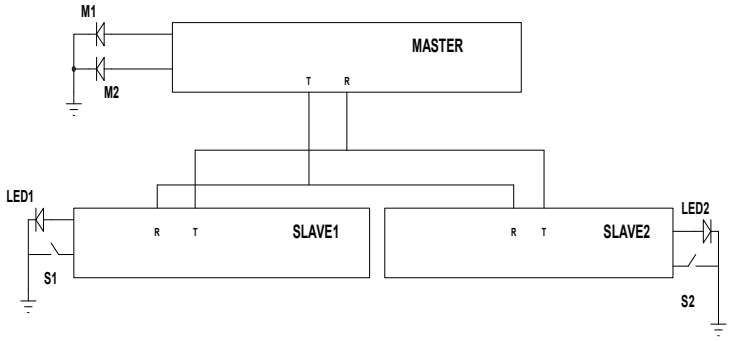
برنامه‌ای بنویسید که شامل اتصال دو میکروکنترلر Slave به یک میکروکنترلر Master باشد و مطابق شکل (۷-۱۷) به هر Slave یک LED و یک میکروسوئیچ متصل باشد و دو LED نیز به Master وصل شده باشند. هنگامی که میکروسوئیچ یکی از Slave ها تحریک می‌شود یکی از LED های متصل به Master روشن و LED متصل به Slave دیگر نیز روشن شود و هنگامی که تحریک قطع می‌شود LED ها خاموش شوند.

توجه: آدرس S1=100 و S2=110 فرض شود.

راهنمایی: بیت پر ارزش اطلاعات مبادله شده را برای تشخیص نوع اطلاعات دریافتی (صفر برای" داده" و یک برای "آدرس") در نظر بگیرید.

تنظیمات CodeWizard

```
UCPOL Slave = 0
UCPOL Master = 1
BAUD RATE = 9600
Master :  RECEIVER
          TRANSMITTER
Slave : RECEIVER
```



شکل (۷-۱۷): اتصال یک میکروکنترلر Master به دو Slave در مد چند پردازنده

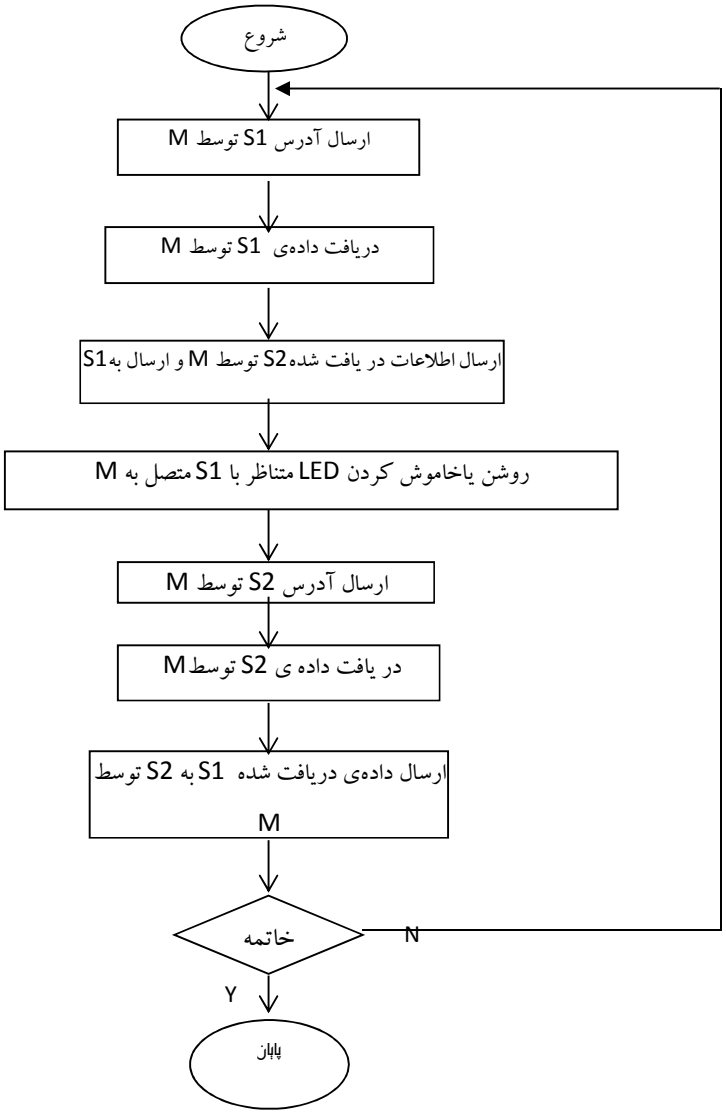
شرح کلی:

۱. پایه‌های R و T در Master فعال شده است. اما تنها پایه‌ی R در Slave ها فعال شده است و پایه‌ی T آن‌ها در حین برنامه توسط رجیستر UCSRB فعال می‌شود. همان طور که می‌دانید از آنجایی که پایه‌های T هر دو Slave به هم متصل است و پایه‌های R هر دو Slave نیز به یکدیگر متصل است، اگر آن‌ها بخواهند همزمان اطلاعات را به Master ارسال کنند باعث آسیب دیدن پورت سریال خواهد شد. لذا در برنامه مربوط به Slave ها، Slave پس از تطبیق آدرس دریافتی، پایه T خود را فعال و پس از ارسال داده به Master پایه‌ی T خود را غیرفعال می‌کند.

۲. پورت مربوط به کلیدها به صورت Pull-up و پورت LED ها در حالت خروجی تنظیم شده‌اند.

شرح برنامه مربوط به Master

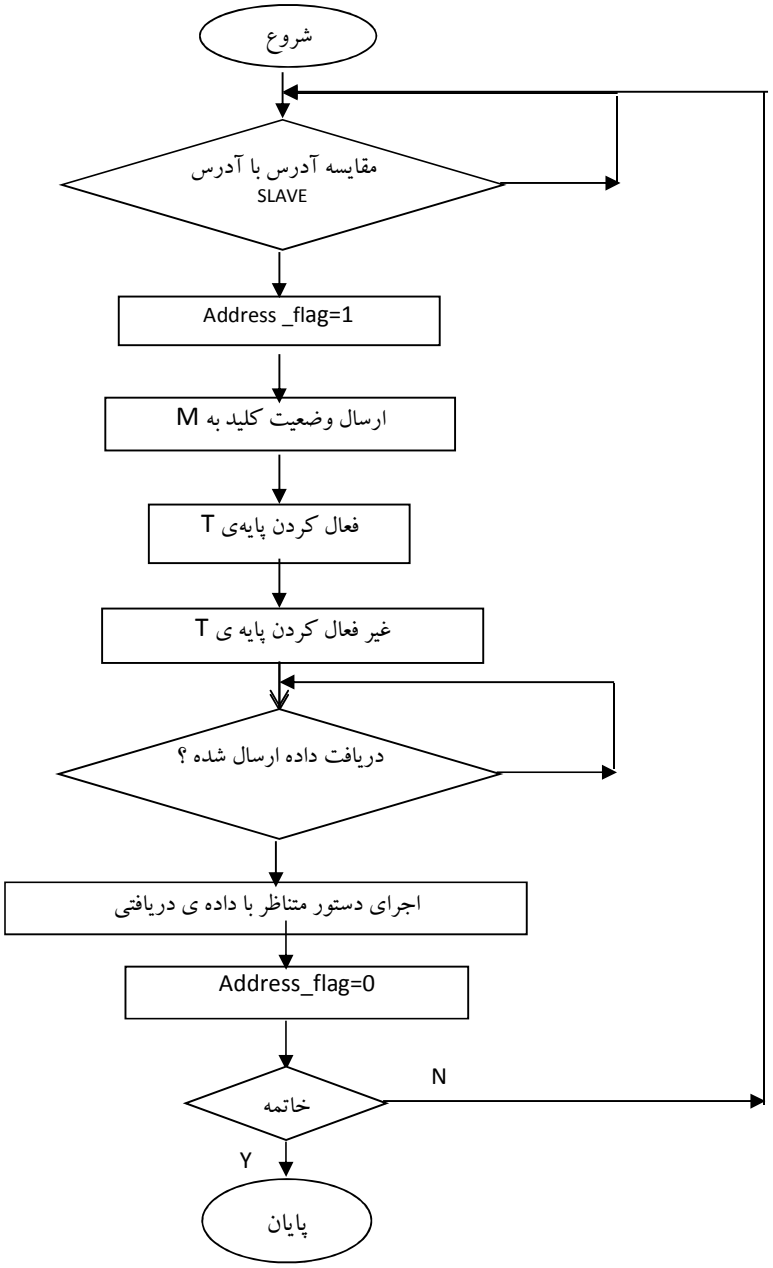
ابتدا Master توسط دستور putchar() آدرس Slave1 را روی خط می فرستد، سپس Slave1 اطلاعات کلید مورد نظر (خاموش یا روشن بودن کلید) را به Master ارسال می‌کند و Master پس از ذخیره آن‌ها اطلاعات مربوط به Slave2 (خاموش یا روشن بودن کلید مربوطه) را به Slave1 می‌فرستد سپس Master به بررسی داده‌ی دریافتی از Slave1 می‌پردازد، اگر کلید مورد نظر فشرده شده باشد LED متناظر با Slave1 (M1) را روشن در غیر این صورت خاموش می کند. سپس Master آدرس Slave2 را ارسال می‌کند و داده مربوطه را از Slave2 دریافت و داده Slave1 را به Slave2 می‌فرستد سپس به بررسی داده دریافتی از Slave2 می‌پردازد و وضعیت کلید مورد نظر را روی M2 نمایش می‌دهد. برای درک بهترعملیات فوق به روند شکل (۷-۱۸) توجه کنید:



شکل (۷-۱۸): الگوریتم پروژه برای میکروکنترلر Master

✓ برنامه Master

```
#include <mega16.h>
#include <stdio.h>
#include <delay.h>
void main(void)
{
char d1,
    d2,
    a1=0b11100100,//100
    a2=0b11101110;//110
PORTA=0x00;DDRA=0x03;
PORTB=0x00;DDRB=0x01;
PORTD=0x00;DDRD=0x01;
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0xC7;
UBRRH=0x00;
UBRRL=0xCF;
ACSR=0x80;
SFIOR=0x00;
while (1)
    {
    putchar(a1);
    d1 = getchar();
    putchar(d2);
    if(d1 == 0)
        PORTA.0 = 0;
    else
        PORTA.0 = 1;  delay_ms(100);
    putchar(a2);
    d2 = getchar();
    putchar(d1);
    if(d2 == 0)
        PORTA.1 = 0;
    else
        PORTA.1 = 1;  delay_ms(100);
    };
}
```



شکل (۷-۱۹): الگوریتم پروژه برای میکروکنترلرSlave

شرح برنامه‌های مربوط به Slave ها

میکروکنترلر Slave ابتدا آدرس دریافتی از Master را با آدرس خود مقایسه می‌کند، در صورت برابر بودن آدرس دریافتی با آدرس Slave، متغیر address_flag را ۱ می‌کند. در این صورت، Slave پایه T خود را توسط رجیستر UCSRB فعال می‌کند و وضعیت کلید خود را به Master ارسال می‌کند. سپس دوباره پایه ی T خود را غیر فعال می‌کند. سپس آنقدر منتظر می‌ماند تا داده‌ی بعدی دریافت شود و پس از دریافت داده به بررسی دستور متناظر با داده می‌پردازد و در آخر دوباره متغیر address_flag را صفر می‌کند. نمای کلی الگوریتم در شکل (۷-۱۹) به درک بیشتر برنامه کمک می‌کند.

- **توجه:** دستور if ((data & 0b10000000)!=0) برای این منظور نوشته شده است که اگر احتمالاً داده دومی، که از طرف Master به Slave ارسال می‌شود از بین برود و داده بعدی که آدرس است، ارسال بشود، این آدرس را فقط Slave بعدی دریافت کند و Slave جاری آن را به عنوان داده در نظر نگیرد.

✓ برنامه Slave1

```
#include <mega16.h>
#include <stdio.h>
void main(void)
{
char data,adress_flag=0;
PORTA=0x01;
DDRA=0x02;
UCSRA=0x00;
UCSRB=0x10;
UCSRC=0xC6;
ACSR=0x80;
SFIOR=0x00;
while (1)
    {
    if( adress_flag == 1)
            {
        //trans en
      UCSRB=0x18;
      if(PINA.0 == 1)
            {
```

```
                data = 0;
                putchar(data);
                }
        else
                {
                data = 1;
                putchar(data);
            }
        //trans dis
        UCSRB=0x10;
        data = getchar();
        if( (data & 0b10000000) != 0)
                {
                }
        else
                {
                if(data == 0)
                        {
                        PORTA.1 = 0;
                        }
                else
                        {
                        PORTA.1 = 1;
                        }
                }
        adress_flag = 0;


        }
    else
        {
        data = getchar();
      if( data == 0b11100100 )      adress_flag = 1;     //=100
         }
    };
}
```

✓ برنامه Slave2

```
#include <mega16.h>
#include <stdio.h>
#include <delay.h>
void main(void)
```

```
{
 char adress_flag = 0,data;
PORTA=0x01; DDRA=0x02;
UCSRA=0x00;
UCSRB=0x10;
UCSRC=0xC6;
ACSR=0x80;
SFIOR=0x00;
while (1)
     {
     if( adress_flag == 1)
            {
            //trans en
        UCSRB=0x18;
        if(PINA.0)
                    {
                    data = 0;
                    putchar(data);
                    }
            else
                    {
                    data = 1;
                    putchar(data);
                    }
            //trans dis
            UCSRB=0x10;
            data = getchar();
            if( (data & 0b10000000) != 0)
                    {
                    }
            else
                    {
                    if(data == 0)
                            {
                            PORTA.1 = 0;
                            }
                    else
                            {
                            PORTA.1 = 1;
                            }
                    }
```

```
            adress_flag = 0;

            }
    else
            {
            data = getchar();
        if( data == 0b11101110 )    adress_flag = 1        //==110
            }
    };
}
```

۷-٦- اتصال میکروکنترلر AVR به پورت سریال کامپیوتر با ساختار RS232

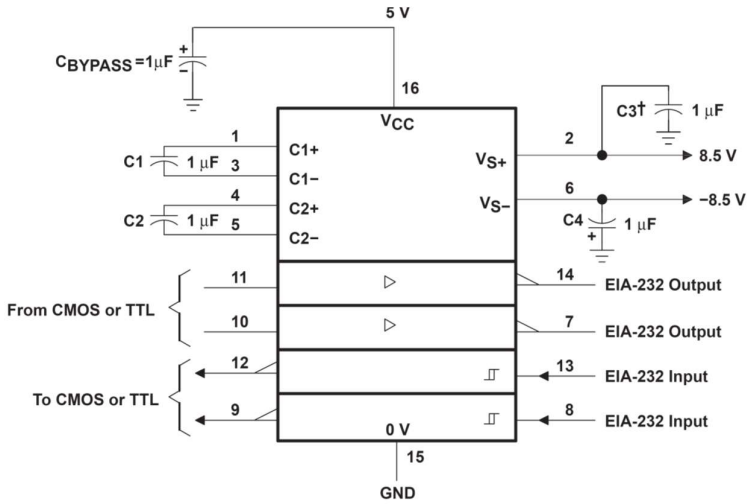
در روش تبادل اطلاعات سریال با استفاده از پروتکل RS232 یک بسته اطلاعاتی با یک بیت شروع و تعدادی بیت ختم می‌شود. همچنین، با نرخ تبادل قابل تنظیم ارسال و یا دریافت می‌شود. تعداد بیت‌های داده، تعداد بیت‌های خاتمه، نوع توازن بیت‌ها و نرخ تبادل داده از پارامترهای قابل تنظیم هستند. گذرگاه سریال میکروکنترلر AVR کلیه قابلیت‌های پروتکل RS232 را پشتیبانی می‌کند. تنها تفاوت آن‌ها در سطوح ولتاژ می‌باشد. سطح ولتاژ برای RS232 بین ۳ تا ۲۵V برای صفر منطقی و بین۳- تا ۲۵V- برای یک منطقی می‌باشد: در حالی که این مقادیر برای USART به ترتیب ۰ و ۵V می‌باشند.

پروژه بیست‌ونهم: ارتباط میکروکنترلر AVR و کامپیوتر

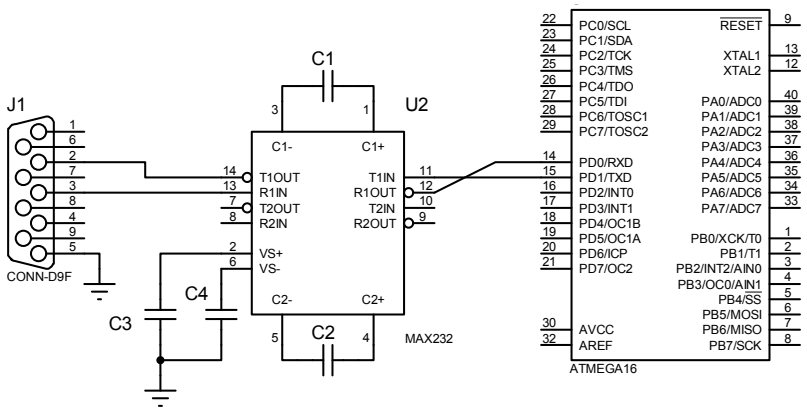
✍ برنامه‌ای بنویسید که میکروکنترلر AVR از طریق پورت سریال با کامپیوتر تبادل داده انجام دهد؟

☑ در این برنامه می‌خواهیم نحوه‌ی ارتباط میکروکنترلر AVR با کامپیوتر از طریق پورت سریال آن (COM) را بیان کنیم. از آنجایی که سطوح TTL ایجاد شده توسط میکرو RS232 با هم تفاوت می کنند باید به نحوی این سطوح ولتاژ به یکدیگر تبدیل شوند. برای این کار از تراشه‌های MAX232 یا MAX233 استفاده می‌شود؛ تنها تفاوت این دو تراشه در این است که MAX232 به ٤ خازن با ظرفیت‌های یکسان ۱ تا 22uF نیاز دارد در صورتی که MAX233 به این خازن ها نیاز ندارد. پیکربندی تراشه MAX232 در شکل (۷-۲۰) نشان داده شده است و در شکل (۷-۲۱)، نحوه اتصال تراشه MAX232 به میکرو و پورت COM کامپیوتر نشان داده شده است. همچنین پیکربندی تراشه MAX233 در شکل (۷-

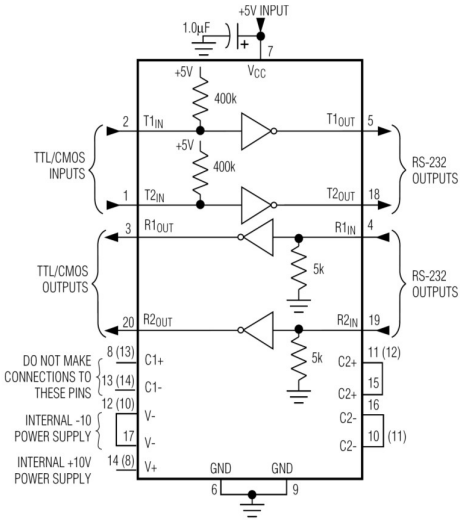
۲۲) نشان داده شده است و در شکل (۷-۲۳)، نحوه اتصال تراشه MAX233 به میکرو و پورت COM کامپیوتر نشان داده شده است.



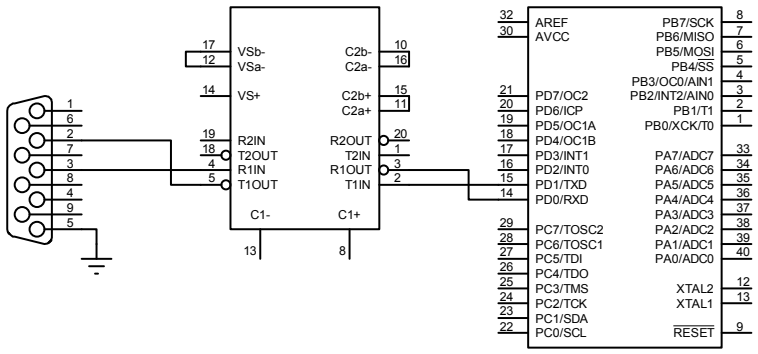
شکل(۷-۲۰) :پیکربندی تراشه MAX232



شکل(۷-۲۱): نحوه اتصال تراشه MAX232 به میکرو



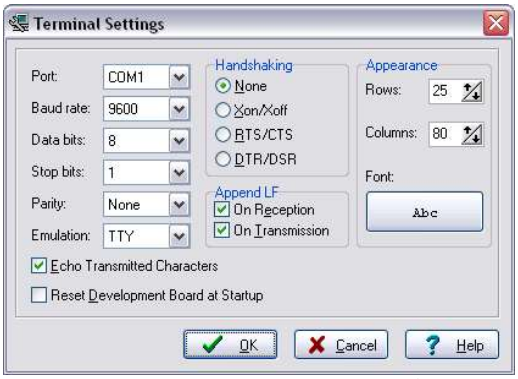
شکل(۷-۲۲): پیکربندی MAX233



شکل(۷-۲۳): نحوه اتصال تراشه MAX233 به میکرو

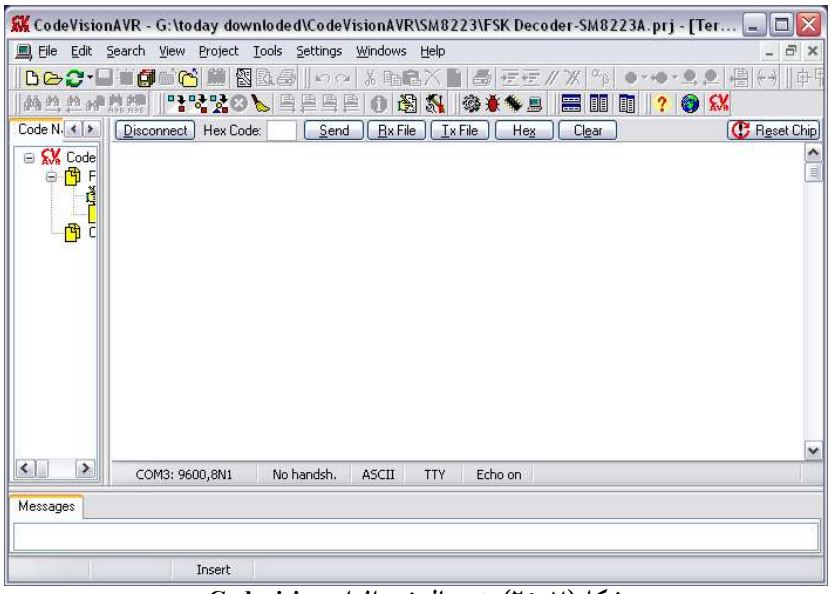
برای کنترل تبادل اطلاعات بین میکروکنترلر و کامپیوتر می‌توان در سیستم عامل DOS از زبان‌های برنامه‌نویسی سطح متوسط نظیر C یا پاسکال که قابلیت کار با سخت‌افزار را دارند استفاده نمود و یا می‌توان از محیط Terminal در داخل نرم افزار code vision، FAST AVR، Bosom و یا Hyper Terminal نرم افزار ویندوز و یا برنامه Advanced Serial Port Monitor استفاده کرد و یا این‌که خودتان می‌توانید برنامه‌هایی مشابه، البته با محدودیت‌های کاربردی، با نرم افزارهایی نظیر MATLAB و یا ویژوال بیسیک بنویسید.

چنانچه از محیط Terminal نرم‌افزار Codevison استفاده می‌کنید باید قبل از برقراری ارتباط، این محیط را تنظیم کنید، برای ایجاد تنظیمات باید به منوی Setting /Terminal بروید تا پنجره شکل (۷-۲٤) باز شود.



شکل(۷-۲٤): پنجره تنظیمات ترمینال

در این پنجره باید نوع پورت ارتباطی (که در این برنامه پورت COM می‌باشد)، نرخ ارسال و فرمت داده‌ها تعیین شود.



شکل(۷-۲۵): ترمینال نرم افزار Codevision

در این برنامه به دنبال آن هستیم که هر آن چه در محیط ترمینال نوشته شود روی LCD متصل شده به میکرو نشان داده شود و هنگامی که کلید k روی پایه Int0 فشرده شد عدد روی پورت B میکرو به کامپیوتر ارسال و در محیط ترمینال نمایش داده شود. در این برنامه نرخ ارسال داده را 4800bps فرض می‌شود.

- **شرح برنامه:** در این برنامه از تابع printf() برای ارسال عدد موجود روی پورت B و از تابع scanf() برای دریافت داده‌ها از پورت سریال میکرو که به پورت سریال کامپیوتر متصل است استفاده شده است. در این برنامه به طور پیش فرض عدد ۲۵۵ توسط مقاومت‌های بالاکش داخلی روی پورت B ساخته شده است.

✓ برنامه

```
#include <mega16.h>
#asm
 .equ __LCD_port=0x1B ;PORTA
#endasm
#include <lcd.h>
#include<stdio.h>
char    c[32];
int k;
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
k=PINB; printf("%i",k);
}

void main(void)
{
PORTA=0x00; DDRA=0xFF;
PORTB=0xff; DDRB=0x00;
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x40;
TIMSK=0x00;
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;
ACSR=0x80;
SFIOR=0x00;
```

```
lcd_init(16);
#asm("sei")
while (1)
    {
    scanf("%s",c);
   lcd_clear();
    lcd_puts(c);
        };
}
```

۷-۷- اتصال میکروکنترلر AVR به پورت سریال با استفاده از پروتکل RS485

استاندارد RS485 نسبت به RS432 دارای مزیت‌های فراوانی است، از جمله این مزیت ها می‌توان به موارد زیر اشاره کرد:

۱. RS232 بسیار نویز پذیر است اما RS485 به دلیل عملکرد دیفرانسیلی کمتر از نویز تاثیر می‌پذیرد.
۲. سرعت انتقال اطلاعات RS484 نسبت به RS232 بیشتر است.
۳. حداکثر طول مجاز سیم در RS485 تقریبا ۵۰ برابر RS232 است.

برای مقایسهٔ بهتر به جدول (۷-۵) توجه کنید.

جدول (۷-۵) : مقایسه ویژگی‌های تبادل سریال با ساختار RS232 و RS485

توضیح	RS232	RS485
عملکرد به صورت دیفرانسیلی	خیر	بله
مدهای ارتباطی	Half duplex Full duplex	Half duplex
حساسیت ورودی گیرنده	+/-3V	+/-200mv
حداکثر طول سیم	۲۵ متر	۱۲۰۰ متر
حداکثر سرعت در طول ۱۲m	20kbs	33Mbs
حداکثر سرعت در طول ۱۲۰۰m	1kbs	100kbs
حداکثر ولتاژ خروجی درایور	+/-25v	-7~12v

☜ تراشهٔ ADM485

از این تراشه برای تبدیل سطوح ولتاژ TTL و RS485 استفاده می‌شود. نحوهٔ تبدیل سیگنال TTL به RS485 مطابق زیر است:

۱. اتصال پایهٔ فرستندهٔ میکرو (TXD) به پایهٔ DI از DM4854
۲. فعال کردن ADM485 از طریق پایهٔ DE

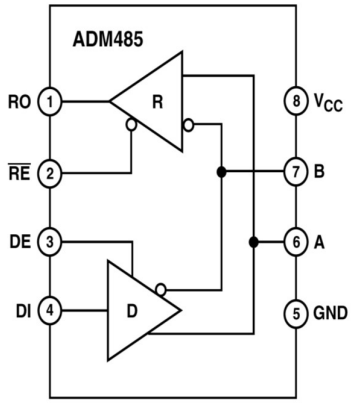
بعد از برداشتن دو گام فوق، سیگنال دیفرانسیلی بر اساس استاندارد RS485 روی پایه‌های A و B تولید خواهد شد.

نحوهٔ تبدیل سیگنال RS485 به سطوح TTL:

۱. اتصال پایه‌های دیفرانسیلی مثبت و منفی سیگنال RS485 به پایه‌های A و B
۲. فعال کردن گیرندهٔ داخلی تراشه توسط پایهٔ RE

پس از برداشتن دو گام فوق، سیگنال با سطوح TTL روی پایهٔ RO تولید خواهد شد.

- **توجه:** از آنجایی که RS485 به صورت Half duplex عمل می‌کند در هر لحظه تنها یکی از پایه‌های RXD و TXD فعال هستند. همان طور که در شکل (۷-۲۶) ملاحظه می‌کنید اگر پایه‌های DE و RE به هم متصل شوند با اعمال ولتاژ مناسب به این پایه می‌توان، جهت تبادل دیتا را کنترل نمود. چنانچه این پایه‌ها به صورت پیش فرض صفر باشد میکرو فقط می‌تواند اطلاعات را به صورت سریال دریافت کند.

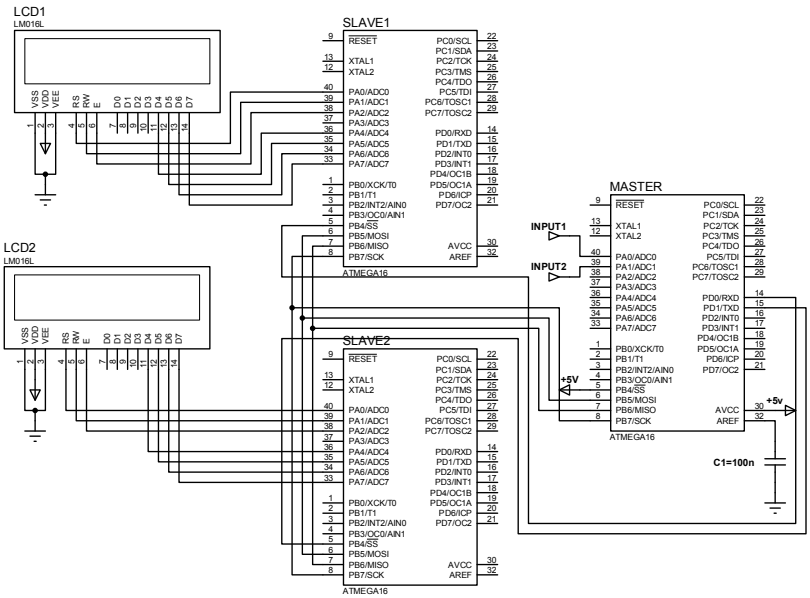


شکل(۷-۲۶): پایه‌های تراشه ADM485

پروژه سی‌ام: مد چند پردازنده SPI

✍ برنامه‌ای بنویسید که ولتاژ اندازه‌گیری شده از طریق دو کانال ADC1 و ADC0 در میکروکنترلر Master را با استفاده از پروتکل SPI به ترتیب برای Slave1 و Slave2 ارسال شود و Slave ها نیز ولتاژ دریافت شده را روی LCD نشان دهند.

☑ در این برنامه برای انتخاب هر یک از Slave ها، پایه SS آن‌ها به ترتیب توسط پایه PD.0 (برای Slave1) و پایه PD.1 (برای Slave2) پایین نگه داشته می‌شود.



شکل(۷-۲۷): بلوک دیاگرام اتصال یک Master و دو Slave با واسط سریال SP

✓ برنامه Master

```
/****************************************
Chip type           : ATmega16
Clock frequency     : 8.000000 MHz
*****************************************/
#include <mega16.h>
#include <delay.h>
#define ADC_VREF_TYPE 0x40
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
```

```
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

// SPI functions
#include <spi.h>

void main(void)
{
PORTB=0x00; DDRB=0Xb0;
ACSR=0x80; SFIOR=0x00;
// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x86;
SFIOR&=0Xef;
// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 2000.000 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x50;
SPSR=0x00;
// configure slaves
DDRD.0=1; PORTD.0=1;
DDRD.1=1;  PORTD.1=1;
while (1)
    {
    PORTD.0=0; // Slave 1 select
    spi(read_adc(0));
    PORTD.0=1; // Slave 1 deselected
```

```
    PORTD.1=0; // Slave 2 select
    spi(read_adc(1));
    PORTD.1=1; // slave 2 deselected
    delay_ms(600);
    };
}
```

✓ برنامه slave ها که برای هر دو میکروکنترلر به صورت یکسانی نوشته می‌شود:

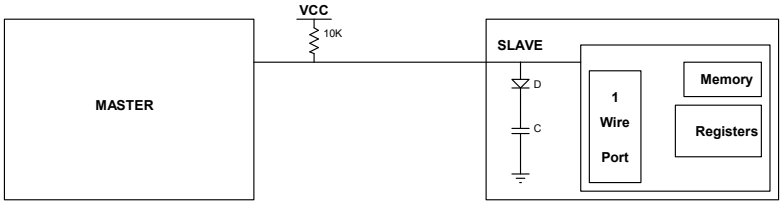
```
/*****************************************
Chip type           : Atmega16
Clock frequency     : 8.000000 MHz
*****************************************/
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
// Alphanumeric LCD Module functions
#asm
 .equ __LCD_port=0x1B
#endasm
#include <lcd.h>
char aray[32],d[10];
float code;
// SPI interrupt service routine
interrupt [SPI_STC] void spi_isr(void)
{
unsigned char data;
data=SPDR;
code = data;// f contain adc codes
code = code * 0.0048828125; // 0.0048828125==(5/1024)
ftoa(code,4,d);
sprintf(aray,"volt = %s",d);
lcd_clear();
lcd_puts(aray);
delay_ms(500);
// Place your code here
}

void main(void)
```

```
{
PORTA=0x00;
DDRA=0Xff;
PORTB=0x00;
DDRB=0x40;
ACSR=0x80;
SFIOR=0x00;
// SPI initialization
// SPI Type: Slave
// SPI Clock Rate: 2000.000 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0Xc0;
SPSR=0x00;

// Clear the SPI interrupt flag
#asm
   in   r30,spsr
   in   r30,spdr
#endasm

// LCD module initialization
lcd_init(16);
// Global enable interrupts
#asm("sei")
while (1);
}
```

۷-۸- ارتباط سریال یک سیمه (One Wire)

این نوع ارتباط، پروتکلی است که شرکت Dallas Semiconductor ارائه نموده است. در این نوع ارتباط بر خلاف ارتباط‌های SPI، TWI و نظایر آن که چندین خط برای انتقال داده، تغذیه Slave، سنکرون‌سازی داده‌ها و نظایر آن وجود دارد، تنها یک سیم برای تبادل داده به صورت Half Duplex استفاده می‌شود و منبع تغذیه لازم برای slave مطابق شکل (۷-۲۸) با استفاده از یک یکسو کننده دیودی و خازن از روی خط انتقال داده فراهم می‌شود و به این ترتیب در هزینه‌هایی که مربوط به سیم‌های اضافی و یا منبع تغذیهٔ slave است صرفه‌جویی می‌شود. هنگامی که سیم توسط مقاومت

بالاکش یک می‌شود، دیود روشن و خازن 800pf شارژ می‌شود و از انرژی ذخیره شده برای تغذیهٔ اسیلاتور داخلی و مدارات کنترلی استفاده می‌شود. در این پروتکل هر کدام از slave ها یک آدرس منحصر به فرد و یک کنترلر مستقل دارند. همچنین درایورهای Slave و Master به صورت درین باز همراه یک مقاومت بالاکش می‌باشند که به ولتاژ ۵ ولت متصل شده‌اند. این نوع پروتکل در سنسورهای دما، حافظه و مشابه آن به کار رفته است و در سال‌های اخیر کاربرد آن در وسایل گوناگون افزایش یافته است. در Codevision نیز کتابخانه‌های طراحی شده است که شامل توابع مختلفی برای ارتباط با این پروتکل می‌باشد.



شکل(۷-۲۸): نحوه ارتباط Master و Slave با پروتکل 1WIRE

به طور کلی با هر میکروکنترلی و با حداقل فرکانس 1.8MHz می‌توان با وسایلی که با این پروتکل کار می‌کنند ارتباط برقرار کرد و زمان‌بندی‌های لازم را پیاده‌سازی نمود. در این نوع ارتباط ولتاژهای کوچکتر از ۰/۸ ولت به عنوان صفر منطقی و ولتاژهای بالاتر از ۲/۲ ولت به عنوان یک منطقی در نظر گرفته می‌شوند. برای نوشتن یک منطقی توسط Master باید خط داده به مدت 15µs پایین نگه داشته شود و برای نوشتن صفر منطقی این خط باید به مدت $60\mu s$ توسط Master پایین نگه داشته شود. در هنگام شروع، باس داده به مدت 480µs به وسیلهٔ Master پایین نگه داشته می‌شود و به این ترتیب باس را Reset می‌کند و پس از آن شروع به آدرس دهی Slave ها می‌کند. پس از فراخوانی slave های مورد نظر، Master شروع به تبادل اطلاعات با slave های انتخاب شده می‌نماید. در داخل ROM داخلی هر کدام از وسایلی که با این پروتکل کار می‌کنند، یک کد ٦٤ بیتی وجود دارد که ۸ بیت اول این کد نوع وسیله را که می تواند EEPROM، سنسور دما و مشابه آن باشد را مشخص می کند. ٤٨ بیت بعدی این کد، مربوط به آدرس تجهیز می‌باشد که البته این آدرس یکتا است. ۸ بیت آخر این کد نیز مربوط به CRC (Cyclic Redundancy Check) می‌باشد که Master بوسیلهٔ آن بروز خطا در تبادل اطلاعات را تشخیص می‌دهد.

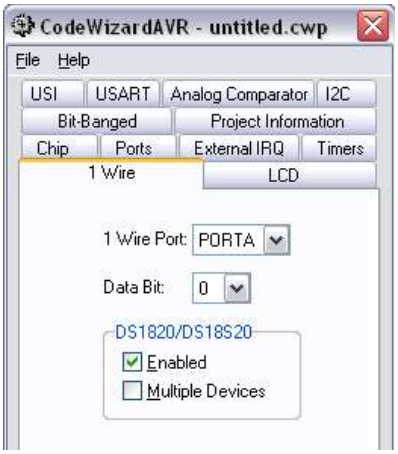
۷-۸-۱- پیکربندی 1Wire با Codewizard

نحوه تنظیم اتصال حسگر تک سیمه به میکرو در نرم افزار Codewizard در شکل (۷-۲۹) آمده است.

1wire Port : پورتی که برای ارتباط 1wire استفاده می‌شود در این قسمت، مشخص می گردد.

Data Bit : در این قسمت شماره ی بیتی از پورت که برای باس 1wire در نظر گرفته شده است مشخص می‌گردد.

اگر از سنسورهای دمای DS1820/DS1822 استفاده می‌کنید باید گزینهٔ Enabled را انتخاب کنید و چنانچه از چند سنسور استفاده می‌کنید باید گزینهٔ Multiple Device را انتخاب کنید که در این حالت می‌توانید هشت Slave را آدرس‌دهی کنید که کدهای ROM مربوط به این slave ها در آرایه ای به نام ds1820_rom _ address وارد خواهند شد.



شکل(۷-۲۹): تنظیمات 1Wire با Wizard

۷-۸-۲- معرفی توابع 1Wire

unsigned char w1_init(void)

این تابع برنامه را برای ارتباط 1wire پیکربندی می‌کند و اگر وسیله‌ای روی باس وجود داشته باشد مقدار یک و چنانچه وسیله ای نباشد مقدار صفر را به خروجی باز می‌گرداند.

unsigned char w1_read(void)

توسط این تابع می‌توان یک بایت را از باس 1wire خواند.

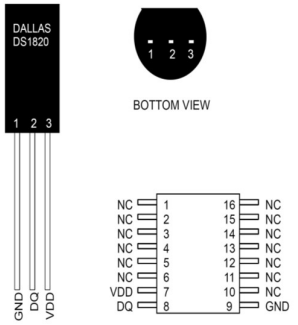
unsigned char w1_write(unsigned char data)
توسط این تابع می‌توان یک بایت داده را روی باس نوشت و چنانچه عمل نوشتن به صورت موفقیت‌آمیز انجام گیرد مقدار یک و در غیر این صورت صفر را به خروجی می‌برد

unsigned char w1_search(unsigned char cmd,void *p)
خروجی این تابع تعداد وسایل متصل به باس می‌باشد و چنانچه وسیله‌ای روی باس نباشد خروجی تابع صفر است.

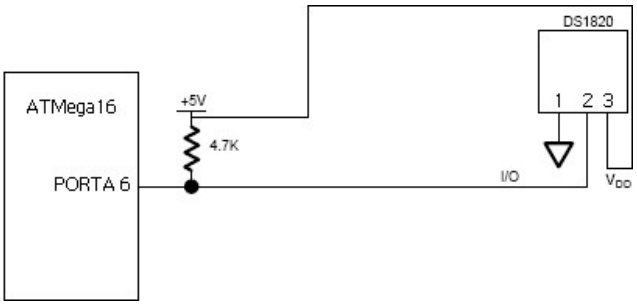
پروژه سی‌ویکم: اندازه‌گیری دما به کمک سنسور DS1820

✍ برنامه‌ای بنویسید که دما را توسط سنسور DS1820 اندازه‌گیری کند؟

☑ حل: در این پروژه از توابع معرفی شده در بخش 1wire استفاده شده است. سنسور دماسنج DS1820 ساخت شرکت Dallas Semiconducter سنسوری است که دمای محیط را با دقت 0.5 درجه سانتی گراد با فرمت سریال تک سیمه ارسال می نماید. محدوده دمایی این سنسور از -55^0c تا 125^0c می‌باشد و قابلیت تغذیه از طریق خط دیتا را دارد (نیازی به تغذیه خارجی ندارد). دمای قرائت شده آن به .صورت داده‌ی ۹ بیتی می‌باشد و زمان تبدیل دمای بدنه به معادل دیجیتال آن حدود ۲۰۰ میلی ثانیه است. در شکل (۷-۳۰) سه نوع بسته‌بندی این سنسور نشان داده شده است. نحوه اتصال آن به میکرو کنترلر ATmega16 در شکل (۷-۳۱) آمده است. می توان از مقاومت 4.7k به عنوان مقاوت بالا کش خروجی استفاده نمود.



شکل(۷-۳۰): بسته‌بندی سنسور دمای DS1820



شکل(۷-۳۱): اتصال سنسور دماسنج به میکروکنترلر

✓ برنامه:

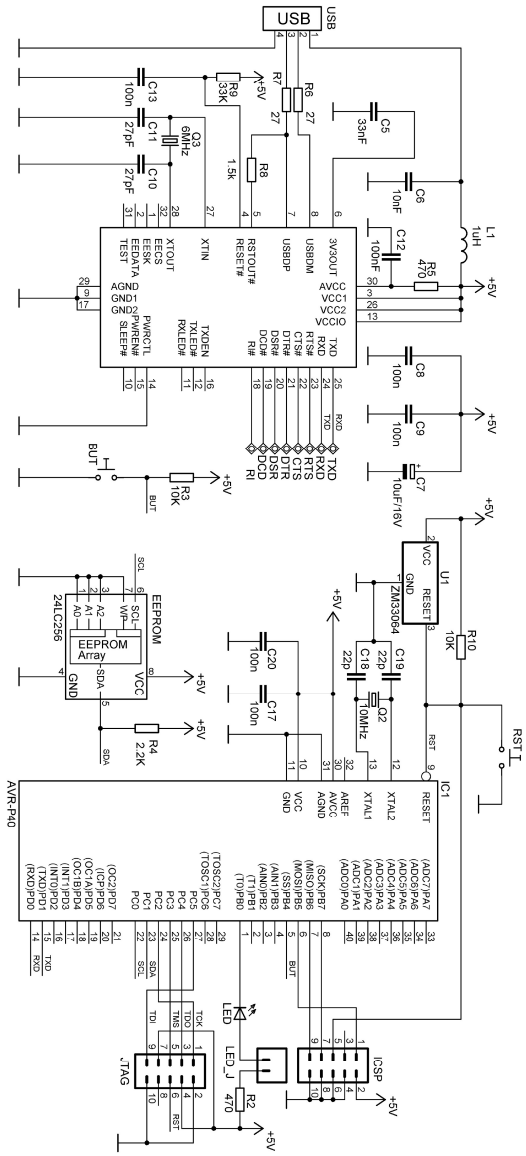
```
/* The DS1820/18S20 sensors are connected to
bit 6 of PORTA of the ATmega16 as follows:
[DS1820/18S20] [STK500 PORTA HEADER]
1 GND        -   9  GND
2 DQ         -   7  PA6
3 VDD        -  10 +5V
All the temperature sensors must be connected
in parallel
AN 4.7k (6.8k IN MY CASE) PULLUP RESISTOR MUST
BE CONNECTED BETWEEN DQ (PA6) AND +5V !
*/
#asm
.equ __w1_port=0×1b
.equ __w1_bit=6
#endasm
 #include <1wire.h>

#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#include <ds1820.h>
#include <delay.h>
#include <stdio.h>
char lcd_buffer[33];
 // maximum number of DS1820/DS18S20 connected to the 1 Wire bus
#define MAX_DEVICES 8
 // DS1820/DS18S20 devices ROM code storage area
```

```
unsigned char rom_code[MAX_DEVICES,9];

void main()
{
unsigned char i,j,devices;
int temp;
lcd_init(16);
lcd_putsf("CodeVisionAVR\n1 Wire Bus Demo");
delay_ms(2000);
lcd_clear();
/* detect how many DS1820/DS18S20 devices
   are connected to the 1 Wire bus */
devices=w1_search(0xf0,rom_code);
sprintf(lcd_buffer,"%u DS1820\nDevice detected",devices);
lcd_puts(lcd_buffer);
delay_ms(2000);
/* display the ROM codes for each device */
if (devices)
   {
   for (i=0;i<devices;i++)
       {
       sprintf(lcd_buffer,"Device #%u ROM\nCode is:",i+1);
       lcd_clear();
       lcd_puts(lcd_buffer);
       delay_ms(2000);
       lcd_clear();
       for (j=0;j<8;j++)
           {
           sprintf(lcd_buffer,"%02X ",rom_code[i][j]);
           lcd_puts(lcd_buffer);
           if (j==3) lcd_gotoxy(0,1);
           };
       delay_ms(5000);
       };
   }
else while (1); /* stop here if no devices were found */
while (1)
      {
        for (i=0;i<devices;)
          {
          temp=ds1820_temperature_10(&rom_code[i][0]);
```

```
        j='+';
        if (temp<0)     {   j='-'; temp=-temp;   };
        sprintf(lcd_buffer,"t%u=%c%i.%u\xdfC",++i,j,temp/10,temp%10);
        lcd_clear();
        lcd_puts(lcd_buffer);
        delay_ms(800);
        };
    };
}
```

۷-۹- ارتباط میکروکنترلر با پورت USB از طریق تراشه FT232

تراشه‌ی FT232 یک واسط USB به سریال RS232 با ویژگی‌های زیر است:
این تراشه از طریق پایه‌های Reset، RXD و TXD با میکرو ارتباط برقرار می‌کند. هنگامی که دستگاه برای اولین بار به کامپیوتر متصل می‌شود، کامپیوتر قادر به شناسایی آن نیست. برای معرفی دستگاه به کامپیوتر باید یکی از درایورهای شرکت FTDI که برای تراشه‌ی FT232 ساخته شده است را به کار بگیرید. در این پروژه درایور Comport پیشنهاد می‌شود که این درایور در CD همراه با کتاب ضمیمه شده است. البته درایورهای دیگر را نیز می‌توانید از سایت www.Atmel.com دریافت کنید. پس از نصب درایور مورد نظر با استفاده از پنجره‌ی Add New Hardware، دستگاه شما به عنوان یک پورت سریال جدید به سیستم معرفی می‌شود و چنانچه کامپیوتر شما دارای دو پورت Com2,Com1 باشد دستگاه موردنظر به عنوان پورت Com3 به کامپیوتر معرفی می‌شود. برنامه‌نویسی میکرو با این تراشه همانند برنامه‌نویسی میکرو برای پورت RS232 است. این تراشه از طریق باس دیتا تغذیه می‌شود، بنابراین باید جریان مصرفی آن تحت کنترل باشد.

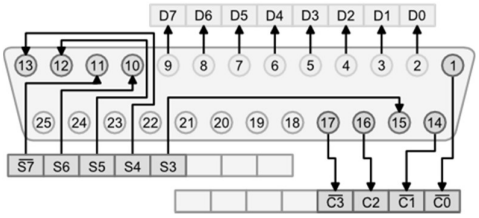


شکل(۷-۳۲): نحوه ارتباط FT232 با با ادواتی که با پروتکل USART کار می‌کنند

پروژه سی‌ودوم: ارتباط با پورت موازی کامپیوتر

برنامه‌ای بنویسید که توسط آن میکروکنترلر عدد ٤٠ را به پورت موازی کامپیوتر ارسال کند و چنانچه عددی روی این پورت قرار گرفته باشد روی LCD نشان داده شود. همچنین برای ارسال یک عدد به پورت موازی و خواندن عددی که توسط میکرو ارسال شده است برنامه‌ای در محیط ویژوال بیسیک بنویسید؟

حل: پورت موازی کامپیوتر دارای ۲۵ پایه به شرح زیرمی‌باشد:



شکل(۷-۳۳): پایه‌های پورت LPT

جدول (۷-٦): توصیف پایه‌های پورت LPT

شماره پایه	نام پایه
۱	strobe
۲ تا ۹	D0-D7
۱۰	ACK
۱۱	BUSY
۱۲	PE
۱۳	SELECT
١٤	AUTO FEED
۱۵	Error
١٦	init
۱۷	SLCT IN
۱۸ تا ۲۵	GND

بسیاری از کامپیوترها دارای ۳ پورت موازی با نام‌های LPT1، LPT2 و LPT3 می باشند و هر یک از این ۳ پورت دارای ۳ رجیستر با نام‌های «رجیستر داده»[1]، «رجیستر وضعیت»[2] و «رجیستر کنترل»[3] می‌باشد.

رجیستر داده

D7	D6	D5	D4	D3	D2	D1	D0

داده‌ی دریافتی یا ارسالی در این رجیستر قرار می‌گیرد. توسط این رجیستر می‌توان به پایه‌های ۲ تا ۹ گذرگاه LPT دسترسی پیدا کرد.

رجیستر وضعیت

Busy	Ack	PE	Select	Error	----	----	----

توسط این رجیستر می توان به پایه‌های select,PE,ACK,Busy و Error گذرگاه LPT دسترسی پیدا نمود.

رجیستر کنترل

---	---	Dir	----	SLCT IN	Init	Auto feed	Strobe

توسط این رجیستر می‌توان به پایه‌های strobe,Auto feed, SLCT IN و Init دسترسی پیدا نمود.

- **توجه:** بسیاری از کامپیوترهای جدید در بیت ششم (Control bit.5) رجیستر کنترلی خود دارای بیت Dir می‌باشند، که وظیفه‌ی این بیت برقراری ارتباط نیم دوطرفه با کامپیوتر است چنانچه این بیت یک باشد، پورت در حالت ورودی و در غیر این صورت در حالت خروجی پیکربندی می‌شود.

جدول(۷-۷): آدرس رجیسترهای پورت LPT

نام پورت	آدرس رجیستر اطلاعات	آدرس رجیستر وضعیت	آدرس رجیستر کنترل
LPT1	378H	379H	37AH
LPT2	278H	279H	27AH

[1] Data Register
[2] Status Register
[3] Control Register

- **توجه:** از آنجا که هیچ یک از خطوط پورت موازی کامپیوتر در برابر اضافه بار محافظت شده نیستند در نتیجه این پورت در مقایسه با پورت سریال بسیار آسیب پذیرتر است. پیشنهاد می‌شود قبل از استفاده از این پورت به توصیه‌های زیر توجه کنید:
  - ✓ محدوده ولتاژهای ورودی این پورت بین ۰ و ۵ ولت باشد.
  - ✓ پین‌های خروجی پورت موازی نباید به ولتاژ تغذیه وسایل دیگر وصل شوند و یا اتصال کوتاه شوند.
  - ✓ وقتی کامپیوتر خاموش است اتصالات خارجی را به آن متصل کنید.

استفاده از پورت موازی در ویژوال بیسیک:

قبل از شروع این بحث لازم است تا با مفهوم DLL[1] آشنا شوید.

همان طور که می‌دانید در ویندوز برای دسترسی به سخت‌افزارهای خارجی نیاز به نرم افزار راه‌انداز آن سخت‌افزار است. پیاده‌سازی یک راه‌انداز کار پیچیده‌ای است و تنها برنامه نویسان خبره این کار را می‌توانند انجام دهند. در نتیجه کاربران کامپیوتر برای دریافت اطلاعات از واسط های کامپیوتر از DLL استفاده می‌کنند. DLL ها توابع کتابخانه‌های ویندوز می‌باشند که برای کار با سخت‌افزار بورد مادر نوشته شده‌اند و از مجموعه‌ای از زیر روال‌ها تشکیل شده‌اند که می‌توان آن‌ها را از داخل برنامه‌های ویژوال بیسیک، c++، و یا هر زبان برنامه‌نویسی فراخوانی کرد.

DLL ای که در این برنامه استفاده شده است در ویژوال بیسیک می‌باشد (البته شما می توانید DLL مورد علاقه خود را به هر زبان برنامه‌نویسی‌ای بنویسید وتوجه داشته باشید که این DLL به هر زبانی که نوشته شود می‌تواند در هر زبان برنامه‌نویسی‌ای فراخوانی شود).

DLL مذکور POTR.DLL می‌باشد که برای استفاده از این DLL باید آن را در دایرکتوری system ویندوز کپی کنید. در این صورت می‌توانید توابع آن را در برنامه خود فراخوانی نمائید.

- **توجه:** DLL فوق برای ویندوز XP تضمین شده است چنانچه از ویندوز VISTA و یا XP SP3 استفاده می‌کنید ممکن است این DLL را قبول نکند. برای رفع این مشکل می‌توانید به جای استفاده از DLL فوق می‌توانید از ActiveX های ارائه شده در سایت http://www.logix4u.net استفاده کنید. ActiveX مربوط

[1] Dynamic Link Library (DLL).

Hwinterface برای این پروژه بسیار مناسب است که در CD همراه کتاب ضمیمه شده است. پس از نصب این DLL می توانید از دستورات آن برای کار با پورت موازی استفاده کنید. DLL دیگری به نام INPUT32.DLL نیز در CD همراه این کتاب آمده است که به جای Port.dll می توانید از آن نیز استفاده کنید. چنانچه از این DLL استفاده کنید می‌توانید از دو تابع OUT,INP برای خواندن ونوشتن در پورت موازی استفاده کنید.اما قبل از آن باید این دستورات را به شکل زیر در ویژوال بیسیک معرفی کنید:

```
Public Declare Function Inp Lib "inpout32.dll"_
Alias "Inp32" ( ByVal PortAddress As Integer ) As Integer
Public Declare Sub Out Lib "inpout32.dll"_
Alias "Out32" ( ByVal PortAddress As Integer,ByVal Value As Integer)
```

آرگومان توابع

```
OUT Addressport,value
Data=Inp(Addressport)
```

معرفی توابع PORT.DLL

تابع Opencom () : از این تابع برای باز کردن پورت LPT استفاده می‌شود.

```
Opencom('LPT1:')
```

نحوه معرفی آن در VB و VBA به صورت زیر است:

```
Declare sub OPENCOM lib "port" ( ByVal as )
Declare Function OPENCOM Lib "port" ( ByVal As ) As Integer
```

تابع Closecom : از این تابع برای بستن پورت LPT استفاده می‌شود و به صورت زیر در VB و VBA تعریف می‌شود:

```
Declare sub Closecom Lib "port" ()
```

تابع Outport() : از این تابع برای نوشتن روی پورت LPT استفاده می‌شود.

```
Outport BA,N
```

توجه : BA (آدرس پایه)برای پورت LPT1 برابر 0x378 می‌باشد.

تابع Outport() به صورت زیر در VB و VBA تعریف می‌شود:

```
Declare Function Inport Lib "port" (ByVal p%) As Integer
```

تابع Inport : از این تابع برای خواندن از پورت LPT استفاده می‌شود.

مثال:

```
A=Inport (BA+1) : REM Status port read
```

در مثال فوق مقدار رجیستر وضعیت خوانده می‌شود.

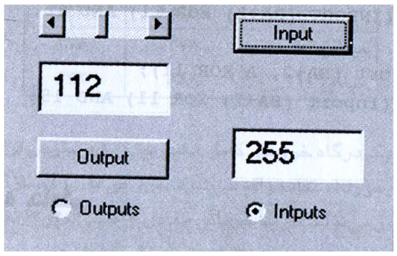
تابع ()Inport به صورت زیر در VB و VBA تعریف می‌شود:

```
Declare Function Inport Lib "port" (ByVal p%) As Integer
```

توجه: به جز بیت Init از رجیستر کنترل، ۳ بیت باقیمانده به صورت I/O معکوس شده‌اند در برنامه‌ها مطابق مثال ارائه شده زیر می توان بیت های مورد نظر را معکوس نمود.

```
Outport BA+2, A XOR 11
B=(Inport (BA+2) XOR 11) AND 15
```

پس از کپی کردن DLL در پوشه System ویندوز خود در ویژوال بیسیک رفته و مطابق شکل (۷-۳٤) طراحی نمائید.



شکل(۷-۳٤): پوسته ایجاد شده در ویژوال بیسیک

پس از آن باید برنامه زیر را در محیط ویرایشگر آن بنویسید.

```
Dim BA As Integer
Private Sub Command1_click()
OUTPORT BA,Hscroll1.value
OUTPORT (BA+2),1 'Strobe=1
OUTPORT (BA+2),0 'Strobe=0
End Sub
Private Sub Command2_click()
OUTPORT (BA+2),34 'Auto feed=1
OUTPORT (BA+2),32 'Auto feed=0
D=INPORT(BA)
D=D And 255
Text2.Text=Str$(D)
End sub

Private Sub Form_ Load()
If OPENCOM("LPT1:") = 0 Then Msgbox ("LPT1 not free")
```

```
Ba=&H378
OUTPORT (BA +2) , 0 'output
End Sub

Private Sub Form_Unload(Cancel As Integer)
CLOSECOM
End Sub

Private Sub Hscroll1_change()
 Text1.text = Str$(Hscroll1.value)
End sub

Private Sub Option1_Click()
 OUTPORT (BA +2) , 0 'output
End Sub

Private Sub Option2_Click()
OUTPORT(BA +2) , 32 'input
End Sub
```

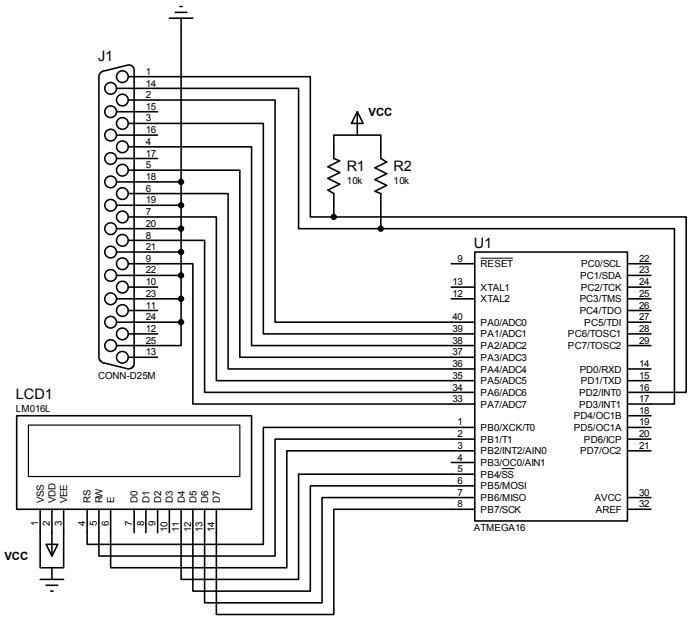
شرح برنامه: با ارسال هر پالس پایین رونده توسط پایه‌ی Auto feed برنامه وارد سرویس وقفه INT1 می‌شود. در این سرویس عدد ٤٠ به پورت موازی ارسال می شود و با ارسال هر پالس پایین رونده توسط پایه‌ی Strobe برنامه وارد سرویس وقفه ی INT0 شده و در این سرویس وقفه، عدد روی پورت موازی خوانده می‌شود.

✓ برنامه

```
/*****************************************************
Chip type           : Atmega16
Clock frequency     : 8.000000 MHz
*****************************************************/
#include <mega16.h>
// Alphanumeric LCD Module functions
#asm
 .equ __LCD_port=0x18
#endasm
#include <LCD.h>
#include <stdio.h>
 char array[10];
 unsigned char d;
```

```
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void) //Read data
{
DDRA=0;
d=PINA;
sprintf(array,"DATA IS %d",d);
lcd_clear();
lcd_puts(array);
}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)  //write Data
{
DDRA=0xff;
PORTA=40;
}

void main(void)
{
DDRA=0;
PORTA=0x00;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: On
// INT1 Mode: Falling Edge
// INT2: Off
GICR|=0Xc0;
MCUCR=0x0A;
MCUCSR=0x00;
GIFR=0Xc0;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;
```

```
// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")

while (1);
}
```



شکل(۷-۳۵): شماتیک اتصال میکرو به پورت موازی و LCD

فصل هشتم حافظه‌های جانبی

در این فصل، با استفاده از چندین پروژه کاربردی، علاوه بر معرفی حافظه‌های جانبی با دستورات مربوط به راه‌اندازی و کار با آن‌ها آشنا می‌شویم.

پروژه سی‌وسوم: ارتباط با حافظه سریال AT24C256 توسط I2C

برنامه‌ای بنویسید که توسط آن، ابتدا حافظه EEPROM سریال مدل AT24C256 راه‌اندازی شود و داده 0x55 را در آدرس 0xaa بنویسد و سپس داده‌ها همان آدرس را بخواند؟

حل: حافظه‌ی سریال AT24C256 دارای 256 KBit حافظه است که دارای پروتکل سریال دوسیمه[1] می‌باشد و دارای ویژگی‌های زیر می‌باشد:

۱. توان مصرفی کم و کار با ولتاژهای استاندارد همچون 2.5V، 2.7V، 5V، 5.5V
۲. استفاده از اشمیت تریگر و فیلتر برای حذف نویز.
۳. استفاده از پروتکل دو سیمه به صورت دو طرفه.
٤. حفاظت از داده‌ها در مقابل عمل نوشتن به صورت سخت افزاری توسط پایه WP

معرفی پایه‌ها

SCL: با لبه مثبت کلاک این پایه داده‌ها وارد EEPROM و با لبه منفی داده‌ها از EEPROM خارج می‌شوند.

SDA: از این پایه برای انتقال دو طرفه داده استفاده می‌شود. توجه کنید که این پایه به صورت Open-Drain است.

WP: چنانچه این پایه به Vcc متصل گردد از داده‌های موجود در حافظه در مقابل عمل نوشتن محافظت می‌گردد و چنانچه به زمین متصل گردد EEPROM عمل خواندن و نوشتن عادی را انجام می‌دهد.

A0 و A1: از این پایه‌ها برای آدرس‌دهی EEPROM استفاده می‌شود.

[1] Two-Wire

آدرس سخت افزاری این تراشه بر اساس بایت زیر تنظیم می‌شود:

1	0	1	0	0	A1	A0	R/W

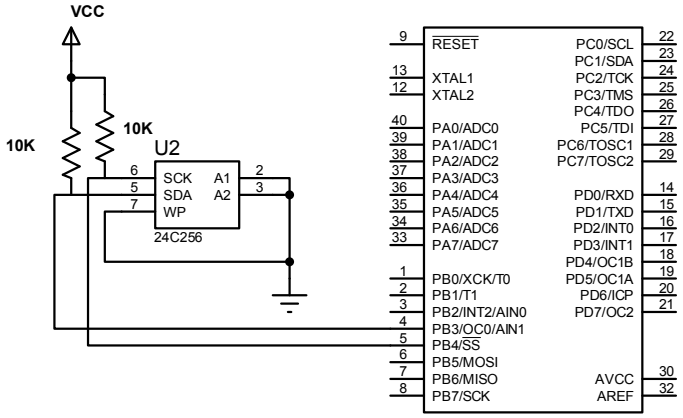
برای نوشتن در حافظه، بیت R/W را صفر و برای خواندن از حافظه یک می‌شود. به این ترتیب آدرس سخت افزاری نوشتن در حافظه ای که A0 و A1 آن زمین شده باشد برابر0x160 =0b10100000 و آدرس سخت‌افزاری خواندن حافظه برابر 0x161=0b10100001 می‌باشد.

برنامه‌ی خواندن از حافظه:

۱. ایجاد Start Condition توسط دستور i2c_start()
۲. ارسال آدرس سخت‌افزاری مورد نظر برای نوشتن توسط دستور i2c_write()
۳. ارسال آدرسی از حافظه که قرار است خوانده شود توسط دستور i2c_write()
٤. ایجاد Start Condition مجدد توسط دستور i2c_start()
۵. ارسال آدرس سخت افزاری حافظه‌ی مورد نظر برای خواندن توسط دستور i2c_write()
٦. دریافت داده از آدرس مورد نظر توسط دستور i2c_read()
۷. ایجاد یک Stop Condition توسط دستور i2c_Stop()

برنامه‌ی نوشتن در حافظه:

۱. ایجاد Start Condition توسط دستور i2c_start()
۲. ارسال آدرس سخت افزاری حافظه‌ی مورد نظر برای نوشتن توسط دستور i2c_ write()
۳. ارسال آدرسی از حافظه که قرار است در آن نوشته شود توسط دستور i2c_write()
٤. ارسال دیتای مورد نظر توسط دستور i2c_write()
۵. ایجاد یک Stop Condition توسط دستور i2c_stop()



شکل (۸-۱): نحوه اتصال EEPROM سریال مدل AT24C256 به میکرو

✓ برنامه

```
#include<mega16.h>
/* the I2C bus is connected to PORTB */
/* the SDA signal is bit 3 */
/* the SCL signal is bit 4 */
#asm
  .equ __i2c_port=0x18
  .equ __sda_bit=3
  .equ __scl_bit=4
#endasm

/* now you can include the I2C Functions */
#include <i2c.h>
/* function declaration for delay_ms */
#include <delay.h>
#define EEPROM_BUS_ADDRESS 0xa0

/* read a byte from the EEPROM */
unsigned char eeprom_read(unsigned char address) {
unsigned char data;
i2c_start();
i2c_write(EEPROM_BUS_ADDRESS);
i2c_write(address);
```

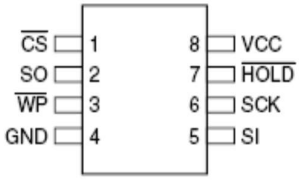
```
i2c_start();
i2c_write(EEPROM_BUS_ADDRESS | 1);
data=i2c_read(0);
i2c_stop();
return data;
}

/* write a byte to the EEPROM */
void eeprom_write(unsigned char address, unsigned char data) {

i2c_start();
i2c_write(EEPROM_BUS_ADDRESS);
i2c_write(address);
i2c_write(data);
i2c_stop();

/* 10ms delay to complete the write operation */
delay_ms(10);
}

void main(void) {
unsigned char i;

/* initialize the I2C bus */
i2c_init();

/* write the byte 55h at address AAh */
eeprom_write(0xaa,0x55);

/* read the byte from address AAh */
i=eeprom_read(0xaa);

while (1); /* loop forever */
}
```

پروژه سی‌وچهارم: ارتباط با حافظه سریال AT25256 توسط SPI

✍ برنامه‌ای بنویسید که توسط آن، ابتدا حافظه EEPROM سریال مدل AT25256 راه‌اندازی شود و داده‌ای را در آدرسی بنویسد و سپس از آدرس دیگری داده‌ای را بخواند؟

☑ حل: حافظه‌ی سریال AT25256 دارای ویژگی‌های زیر می‌باشد:

۱. نرخ کلاک 3MHZ
۲. حد اکثر زمان نوشتن ۵ میلی ثانیه
۳. حفاظت از اطلاعات به صورت سخت افزاری و نرم افزاری
٤. توان ۱۰۰۰۰۰ بار نوشتن و حفظ اطلاعات تا ۲۰۰ سال



شکل(۸-۲): EEPROM سریال مدل AT25256

معرفی پایه‌ها

جدول(۸-۱): معرفی پایه‌های AT25256

نام پایه	عملکرد پایه
$\overline{CS}$	انتخاب آی‌سی
SCK	کلاک داده سریال
SI	ورودی داده سریال
SO	خروجی داده سریال
GND	زمین
VCC	منبع تغذیه
$\overline{WP}$	حفاظت در مقابل نوشتن
$\overline{HOLD}$	معلق کردن ورودی سریال

$\overline{CS}$: با صفر کردن این پایه تراشه انتخاب و با یک کردن آن تراشه از حالت انتخاب خارج می‌شود و وارد حالت Standby می‌شود. در این حالت پایه‌ی so به حالت امپدانس بالا رفته و خط در اختیار سایر وسایلی که به این باس SPI متصل هستند قرار می‌گیرد.

- **توجه:** چنانچه یک سیکل برنامه در حال اجرا باشد این سیکل بدون توجه به سطح پایه‌ی CS کامل اجرا خواهد شد.

SI : از این پایه برای ورود داده‌ها به تراشه استفاده می‌شود.

SO : از این پایه برای خروج داده‌ها استفاده می‌شود.

Sck : پایه‌ای است که کلاک وارد آن می‌شود.

WP : توسط این بیت و بیت WPEN واقع در رجیستر وضعیت می‌توان عمل نوشتن در حافظه را غیر فعال نمود. برای غیر فعال نمودن عمل نوشتن باید پایه ی WP در سطح منطقی صفر و بیت WPEN یک باشد. هنگامی که بیت WPEN صفر باشد صرفنظر از وضعیت WP امکان نوشتن وجود دارد.

HOLD : صفر کردن این پایه باعث می‌شود تا ارتباط با تراشه بدون Reset کردن دنباله‌ی سریال متوقف شود. هنگامی که ارتباط با تراشه متوقف شد پایه‌های SO,SCK,SI به حالت امپدانس بالا می‌روند و چنانچه تغییری بر روی این پایه‌ها رخ دهد، این تغییر در نظر گرفته نمی‌شود. برای برقراری دوباره‌ی ارتباط باید پایه ی HOLD هنگامی که پایه‌ی SCK پایین است، یک شود.

مجموعه دستورات تراشه:

جدول(۸-۲): مجموعه دستورات تراشه

نام دستور	فرمت دستور	عملکرد
WREN	0000X110	
WRDI	0000X100	
RDSR	0000X101	خواندن رجیستر وضعیت
WRSR	0000X001	نوشتن رجیستر وضعیت
READ	0000X011	خواندن داده از آرایه حافظه
WRITE	0000X010	نوشتن داده از آرایه حافظه

نحوه ی عملکرد تراشه

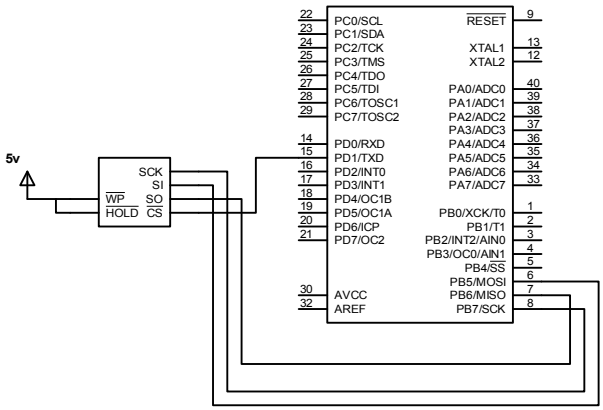
نحوه خواندن از حافظه:

۱. انتخاب تراشه با صفر کردن پایه ی CS
۲. ارسال دستور ۸ بیتی خواندن حافظه
۳. ارسال ۱٦ بیت آدرس خانه حافظه مورد نظر

٤. پس از اعمال مراحل فوق داده موجود در آدرس مورد نظر، توسط پایه‌ی SO، به میکرو ارسال می‌شود. چنانچه ارسال پالس ساعت ادامه پیدا کند می توان داده ی موجود در آدرس های بالاتر را دنباله‌وار خواند. با ارسال هر بایت، اشاره‌گر آدرس داخلی به صورت خودکار به آدرس بالاتر اشاره می‌کند. این روند به همین ترتیب ادامه خواهد داشت تا اشاره‌گر به بالاترین آدرس برسد، هنگامی که داده‌ی موجود در این آدرس هم ارسال شد اشاره‌گر به 0000H اشاره می‌کند. جهت به اتمام رساندن عمل خواندن پایه‌ی CS باید یک شود.

☜ **نحوه نوشتن بر روی حافظه:**

۱. زمین کردن پایه‌ی CS
۲. ارسال دستور ۸ بیتی WREN جهت فعال کردن عمل نوشتن
۳. بعد از ارسال ۸ بیت دستور، برای فعال کردن عمل نوشتن پایه‌ی CS باید یک شود. چنانچه بعد از دستور WREN پایه‌ی CS یک نشود، داده بر روی حافظه نوشته نخواهد شد.
٤. بعد از فعال کردن عمل نوشتن پایه‌ی CS باید زمین شود.
٥. ارسال ١٦ بیت آدرس خانه حافظه مورد نظر
٦. ارسال داده ی مورد نظر که قرار است در آدرس ارسالی ذخیره شود.



شکل(۸-۳): شماتیک سخت‌افزار ارتباط میکروکنترلر با حافظه AT25256 از طریق SPI

- **تنظیمات CodeWizard**

```
SPI initialization
SPI Type: Master
SPI Clock Rate: 2000.000 kHz
SPI Clock Phase: Cycle Half
SPI Clock Polarity: Low
SPI Data Order: MSB First
```

✓ برنامه

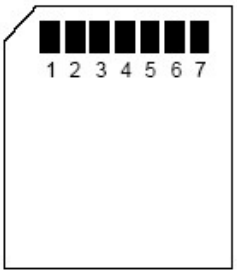
```
/*****************************************
Chip type           : ATmega16
Clock frequency     : 8.000000 MHz
Memory model        : Small
*****************************************/
#include <mega16.h>
#include <delay.h>
// SPI functions
#include <spi.h>

// Declare your global variables here
void write_data(unsigned int address,unsigned int data);
unsigned char read_data(unsigned int address);
void main(void)
{
unsigned int data_send,data_receive;
DDRD.1=1;
PORTB=0x00;
DDRB=0xB0;
ACSR=0x80;
SFIOR=0x00;
// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 2000.000 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x50;
SPSR=0x00;
data_send=45;
write_data(2,data_send);
```

```
delay_ms(200);
data_receive=read_data(3);
while (1);
}

void write_data(unsigned int address,unsigned int data)
{
PORTD.1=0;
spi(6);
PORTD.1=1;
PORTD.1=0;
spi(2);
spi((address & 0b1111111100000000)>>8);
spi(address & 0b0000000011111111);
spi(data);
PORTD.1=1;
}

unsigned char read_data(unsigned int address)
{
unsigned char mydata;
PORTD.1=0;
spi(3);
spi((address & 0b1111111100000000)>>8);
spi(address & 0b0000000011111111);
mydata=spi(250);
PORTD.1=1;
return mydata;
}
```

کارت‌های حافظه MMC و SD به طور وسیع به عنوان حافظه‌های Flash با امکان تبدیل سـریال داده، مورد اسـتفاده قرار می‌گیرند. این کارت‌ها به دلیل قیمت پایین، حجم کم، محدوده وسیع ولتاژ تغذیه و توان مصرفی پایین در موبایل‌ها، دستگاه‌های تغذیه شـده با باتری نظیر پخش صـوت، کتاب الکترونیکی، دایره المعارف، فرهنگ لغت و دفترچه یادداشت‌های الکترونیکی مورد استفاده قرار می‌گیرند. با بهره‌گیری از تکنیک‌های فشرده سازی داده، نظیر: Mpeg و mp3، کارت‌های حافظه ظرفیت کافی برای انواع داده‌های چند رسـانه را دارند. کارت‌های حافظه امروز با ظرفیت‌های ۱۲۸، ۲۵٦، ۵۱۲ مگابایت و یک، دو یا چند گیگابایت عرضـه می‌شـوند. کارت‌های

حافظه در دو نوع SD و MMC (کارت‌های چند رسانه) تولید می‌شوند که کارت‌های حافظه SD، اگر چه قیمت بالاتری دارند، ولی دارای سرعت بیشتر و توان مصرف کمتری نسبت به MMC ها هستند.

تمام واحدها، در کارت‌های چند رسانه، توسط کلاک تولید شده مولد پالس ساعت داخلی کار می‌کنند. واحد «راه انداز واسط»[1] سیگنال‌های DAT و CMD را از CLk خارجی به سیگنال کلاک داخلی سنکرون می‌کند. کارت توسط سیگنال‌های CLk، DAT، CMD کنترل می‌شود. برای تشخیص کارت MMC در یک پشته از MMC ها، یک رجیستر شناسایی کارت (CID) و یک رجیستر آدرس نسبی کارت (RCA) وجود دارد. یک رجیستر اضافی برای ذخیره‌سازی پارامترهای عملکرد مختلف کارت وجود دارد. این رجیستر CSD نام دارد. ارتباط از طریق خطوط MMC برای دستیابی به حافظه یا رجیسترهای آن بر اساس استانداردهای خاص MMC به روش SPI صورت می‌گیرد. برای راه اندازی کارت پس از وصل تغذیه آن نیازی به استفاده از سیگنال Reset نمی‌باشد و به صورت اتوماتیک در هنگام قرار دادن و خارج ساختن، کارت در برابر اتصال کوتاه محافظت می‌شود. برای برنامه‌ریزی آن نیازی به ولتاژ تغذیه خارجی اضافی ندارد و این ولتاژ به صورت داخلی تولید می‌شود. MMC ها حالت تبادل اطلاعات SPI را پشتیبانی می‌کنند. در این حالت واسط مورد نظر پس از اولین فرمان Reset (CMDO) بلافاصله پس از روشن شدن تغذیه شناسایی می‌شود و تا زمانی که تغذیه روشن است، قابل تغییر نمی‌باشد. حالت SPI در MMC کارت‌ها شامل مجموعه‌ای از پروتکل‌ها و فرمآن‌ها است. مزیت استفاده از SPI در MMC کارت‌ها کاهش هزینه طراحی و تولید انبوه محصولات می‌باشد. مشخصات پایه‌های آن در جدول (۸-۳) و شکل (۸-٤) آمده است.



شکل(۸-٤): کارت حافظه MMC

[1] Interface Driver

جدول(۸-۳): پایه‌های MMC

عملکرد پایه	نوع پایه	نام پایه	شماره پایه
انتخاب قطعه	ورودی	CS	1
داده ورودی	ورودی	DI	2
زمین	---	VSS	3
تغذیه مثبت	---	VDD	4
کلاک	ورودی	Clk	5
زمین	---	VSS2	6
دیتا خروجی	خروجی	DO	7

توابع کار با کارت MMC

void mmc_Init(void)

تنظیم سخت‌افزاری برای دسترسی به MMC کارت

unsigned char mmc_command(unsigned char cmd, unsigned long arg)

این تابع برای ارسال فرمان به کارت MMC و کد نتیجه را بر می‌گرداند.

unsigned char mmc_read(unsigned long sector, unsigned char* buffer):

یک سکتور ۵۱۲ بایتی با شماره سکتور مشخص از کارت MMC را به داخل بافر می خواند و در صورت موفقیت صفر بر می‌گرداند.

unsigned char mmc_write(unsigned long sector, unsigned char* buffer):

یک سکتور ۵۱۲ بایتی از بافر را در کارت MMC می‌نویسد و در صورت موفقیت صفر بر می‌گرداند.

پروژه سی‌وپنجم: ارتباط میکروکنترلر با حافظه MMC

برنامه‌ای بنویسید که هر بار با خواندن ولتاژ کانال اول ADC و مقایسه آن با یک حد آستانه ۳ ولتی، در صورت کوچکتر بودن مقدار 0x55 و در صورت بزرگتر

بودن، 0xAA بر روی خانه حافظه مربوط به MMC قرار دهد و پس از ۵۱۲ بار نوشتن داده، یعنی یک سکتور، آن‌ها را بخواند و بر روی پورت D قرار دهد؟

✓ برنامه

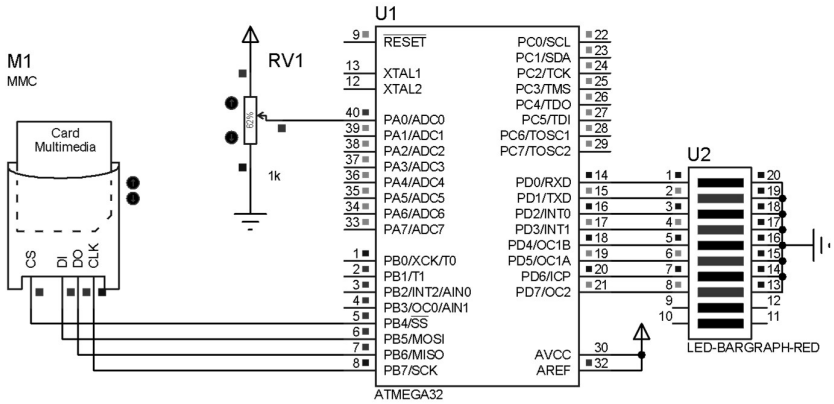
```
#include <mega32.h>
#include<delay.h>
#include <spi.h>
#define IDLE_STATE          0      // MMC to SPI
#define INIT_COMMAND        1      // initialize card
#define CRC_OFF             59     // CRC checking on/off (normally
automatically off in SPI)
#define SET_BLKLEN          16     // Set number of bytes to transfer per
block (default 512 in SPI)
#define WRITE_BLK           24     // write a block
#define READ_BLK            17     // read a block
#define STARTBYTE_RDWR      0xFE
#define DR_MASK             0x1F
#define DR_ACCEPT           0x05
#define BlockSize           512
#define ADC_VREF_TYPE 0x40
/*******************************************************/
// Read the AD conversion result
unsigned char buf_mmc[512];
unsigned long sector_num;
unsigned int i;
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
/*******************************************************/
unsigned char mmc_init(void);
unsigned char mmc_command(unsigned char cmd, unsigned long arg);
unsigned char mmc_write(unsigned long sector, unsigned char* buffer);
unsigned char mmc_read(unsigned long sector, unsigned char* buffer);
/*******************************************************/
unsigned char mmc_command(unsigned char cmd, unsigned long arg)
```

```
{      unsigned char resp;
       unsigned char retry=0;
       PORTB.0 = 1;
       PORTB.0 = 0;
       spi(cmd | 0x40);
       spi(arg>>24);
       spi(arg>>16);
       spi(arg>>8);
       spi(arg);
       spi(0x95);
       while((resp = spi(0xFF)) == 0xFF)
              if(retry++ > 8) break;
       return resp;   }
/*********************************************************/
unsigned char mmc_init(void)
{      unsigned char retry, i, resp=0;
      PORTB = 255;
       retry = 0;
        do
        {
              for(i=0;i<10;i++)
                     spi(0xFF);
              resp = mmc_command(IDLE_STATE, 0);
            //printf("\r\nIDLE_STATE-status:%x", resp);
            delay_ms(10);
              retry++;
              if(retry>10) return -1;
        } while(resp != 0x01);
        retry = 0;
    do
    {
      resp = mmc_command(INIT_COMMAND, 0);
            //printf("\r\nINIT_COMMAND:%x", resp);
            delay_ms(10);
      retry++;
      if(retry>10) return -1;
    } while(resp);
    resp = mmc_command(CRC_OFF, 0);
            //printf("\r\nCRC_OFF:%x", resp);
            delay_ms(10);
    resp = mmc_command(SET_BLKLEN, BlockSize);
```

```
        //printf("\r\nSET_BLKLEN:%x", resp);
         delay_ms(10);
    return 0; }
/*********************************************************/
 unsigned char mmc_write(unsigned long sector, unsigned char* buffer)
{       unsigned char resp;
        unsigned int i;

        PORTB.0 = 0;
        resp = mmc_command(WRITE_BLK, sector<<9);

        if(resp != 0x00)        return resp;
        spi(0xFF);
        spi(STARTBYTE_RDWR);
        for(i=0; i<0x200; i++)
        {
              spi(*buffer++);
        }
        spi(0xFF);
        spi(0xFF);
        resp = spi(0xFF);
        if( (resp & DR_MASK) != DR_ACCEPT)    return resp;


        while(!spi(0xFF));
        PORTB.0 = 1;
        return 0; }
/*********************************************************/
unsigned char mmc_read(unsigned long sector, unsigned char* buffer)
{       unsigned char resp;
        unsigned int i;

        PORTB.0 = 0;
        resp = mmc_command(READ_BLK, sector<<9);

        if(resp != 0x00)        return resp;
        while(spi(0xFF) != STARTBYTE_RDWR);
        for(i=0; i<0x200; i++)
        {
              *buffer++ = spi(0xFF);
        }
```

```
        spi(0xFF);
        spi(0xFF);
        PORTB.0 = 1;
        return 0; }
/*******************************************************/
void main(void)
{unsigned char a;
float v;
PORTA=0x00;DDRA=0x00;
PORTB=0x00;DDRB=0xB1;
PORTC=0x00;DDRC=0x00;
PORTD=0x00;DDRD=0xFF;
TCCR0=0x00;TCNT0=0x00;OCR0=0x00;
TCCR1A=0x00;TCCR1B=0x00;
TCNT1H=0x00;TCNT1L=0x00;
ICR1H=0x00;ICR1L=0x00;
OCR1AH=0x00;OCR1AL=0x00;
OCR1BH=0x00;OCR1BL=0x00;
ASSR=0x00;TCCR2=0x00;
TCNT2=0x00;OCR2=0x00;
MCUCR=0x00;MCUCSR=0x00;
TIMSK=0x00;ACSR=0x80;SFIOR=0x00;
ADMUX=ADC_VREF_TYPE;ADCSRA=0x85;
SPCR=0x50;SPSR=0x00;
mmc_init();
sector_num = 50;i=0;
while (1)
    {
      if  (i<512){v=read_adc(0);v=v*5/1024;a=v;
      if (v>3) buf_mmc[i]=0xAA;
    else buf_mmc[i]=0x55;
       i++; delay_ms(500);}
      if (i==512)
            {
                mmc_write(sector_num, buf_mmc);
                for (i = 0 ; i <= 511 ; i++) buf_mmc[i] = 0;
                mmc_read(sector_num, buf_mmc);
               for (i = 0 ; i <= 511 ; i++)
                 {
                    DDRD=0xff;
                    PORTD=buf_mmc[i] ;
```

```
                elay_ms(500);
            }
        }
    };
}
/*****************************************************/
```



شکل(۸-۵): ارتباط میکرو با کارت حافظه MMC

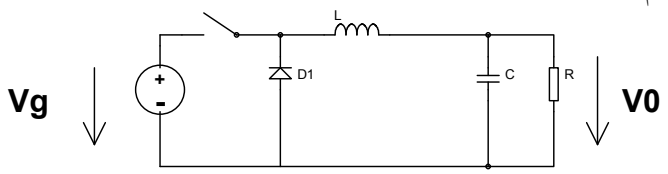
فصل نهم پروژه‌های الکترونیک قدرت

با توجه به اهمیت مبحث الکترونیک قدرت در پیاده‌سازی‌های صنعتی، در این فصل چندین پروژه کابردی بیان می‌شوند که در ادامه به توضیح، طراحی و پیاده‌سازی آن‌ها خواهیم پرداخت.

پروژه سی‌وششم: ساخت مبدل BUCK

یک مبدل ولتاژ DC به DC طراحی نمائید که ولتاژ ورودی آن ۲٤ ولت، ولتاژ خروجی ۱۲ ولت، میزان ریپل جریان سلف آن 0.1 آمپر، ریپل ولتاژ خروجی آن یک درصد، و جریان خروجی آن ۲ آمپر باشد؟

حل: یکی از انواع مبدل‌های الکترونیک قدرت، مبدل‌های ولتاژ DC به DC می باشد. این مبدل‌ها به انواع مختلف کاهنده، افزاینده، افزاینده-کاهنده، Cuck و نظایر آن تقسیم می‌شوند. در این پروژه قصد داریم: به طور مختصر نحوه عملکرد مبدل باک، روابط حاکم بر آن، کنترل ولتاژ خروجی و نحوه شبیه‌سازی آن با نرم افزار Proteus را مورد بررسی قرار دهیم. در شکل (۹-۱) مدار یک مبدل کاهنده ترسیم شده است.



شکل(۹-۱): مدار یک مبدل کاهنده

نحوه عملکرد: این مبدل، یکی از پرکاربردترین مبدل‌های الکترونیک قدرت می باشد و برای تولید یک ولتاژ DC قابل تنظیم از روی یک ولتاژ DC رگوله نشده مورد استفاده قرار می‌گیرد. در این مبدل، v_g یک منبع ولتاژ DC با میزان ریپل قابل توجه می‌باشد و v_o ولتاژ خروجی با دامنه کمتر نسبت به v_g و ریپل بسیار کمتر از آن می‌باشد. کلید sw با فرکانس ثابت و درصد وظیفه قابل کنترل قطع و وصل می‌شود. وقتی که کلید وصل است جریان سلف از مسیر مقاومت شروع به

افزایش می‌کند تا زمانی که کلید قطع شود، تا این لحظه دیود در مسیر جریان قرار ندارد. با قطع شدن کلید، به دلیل پیوستگی جریان سلف، دیود هرزگرد شروع به هدایت کرده و انرژی ذخیره شده در سلف به تدریج به خروجی منتقل می‌شود. اگر جریان سلف در زمان قطع کلید به صفر نرسد گوییم حالت هدایت پیوسته جریان [1](CCM) رخ داده است. در چنین حالتی رابطه ولتاژ خروجی با ولتاژ ورودی به صورت خطی خواهد بود: $V_0=DV_g$ که D درصد وظیفه پالس فرمان سوئیچ می‌باشد. نحوه پیاده‌سازی سخت‌افزاری این مبدل و دیاگرام بلوکی کنترل کننده آن به ترتیب در شکل‌های (۹-۲) و (۹-۳) آورده شده‌اند. با مراجعه به کتاب های مرجع الکترونیک صنعتی می‌توان حداقل مقادیر اندوکتانس و خازن مورد نیاز مبدل را با استفاده از روابط زیر بدست آورد.

$$C \geq \frac{\Delta I_L}{8f.\Delta V_o} \qquad (۹-۱)$$

$$L \geq \frac{D(1-D)V_s}{f.\Delta I_L} \qquad (۹-۲)$$

با استفاده از روابط فوق مقادیر انتخاب شده به صورت زیر می باشند.

$C=10\mu f$ و $L=2mH$

برای پیاده‌سازی دیجیتال سیستم کنترل کننده ولتاژ در مدل باک، باید با فرکانس نمونه‌برداری ثابت از ولتاژ خروجی مبدل نمونه برداری نموده و آن را به معادل دیجیتال تبدیل نمود. محاسبه مقدار خطا از روی مقایسه ولتاژ خروجی نمونه برداری شده با ولتاژ مرجع صورت می‌گیرد و پس از عبور از کنترل کننده PI دیجیتال مقدار درصد وظیفه (DC%) لازم برای پالس فرمان سوئیچ مشخص می‌گردد و با اعمال سیگنال گیت سوئیچ با DC% کنترل شده، ولتاژ خروجی (Vo) تنظیم می‌شود. لازم به ذکر است که برای عملکرد مناسب سیستم کنترل، باید فرکانس نمونه‌برداری از ولتاژ خروجی ثابت باشد و از طرف دیگر کلیه محاسبات مربوط به بدست آوردن سیگنال کنترل باید در یک دوره تناوب نمونه برداری انجام شوند. لذا برای کاهش زمان محاسبات باید حتی الامکان سعی شود از محاسبات ممیز شناور که بسیار زمان‌بر می باشند پرهیز شود. فرم کلی کنترل کننده PI آنالوگ مطابق زیر می‌باشد:

$$D(S)=(K_p+\frac{K_i}{S})E(s) \qquad (۹-۳)$$

که

$E = V_{ref} - V_0$

[1] Continues Current Mode (CCM).

برای تحقق کنترل کننده PI به صورت دیجیتال از گسسته‌سازی رابطه فوق استفاده می‌شود.

$$d((k+1)T) = K_p e(kT) + K_i T \sum_{k=0}^{n} e(kT) \qquad (۹-٤)$$

که k_p و k_i به ترتیب ضرایب تناسبی و انتگرالی کنترل کننده، e(KT) مقدار خطای ولتاژ خروجی در k امین لحظه نمونه‌برداری و T پریود نمونه‌برداری می‌باشد.

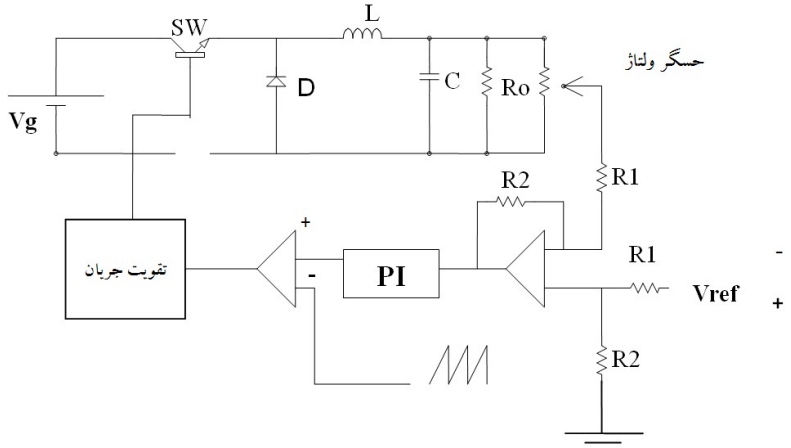
برای محاسبه $\sum_{K=0}^{n}$ متغیر sum به صورت زیر تعریف می‌شود:

$$\text{Sum} = \text{Sum} + e(kT) \times T \qquad (۹-۵)$$

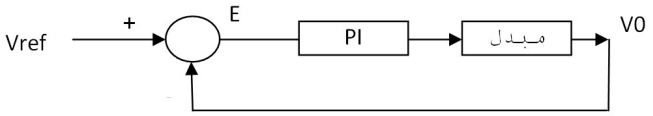
در این صورت داریم:

$$d((k+1)T) = K_p e(kT) + sum \qquad (۹-٦)$$

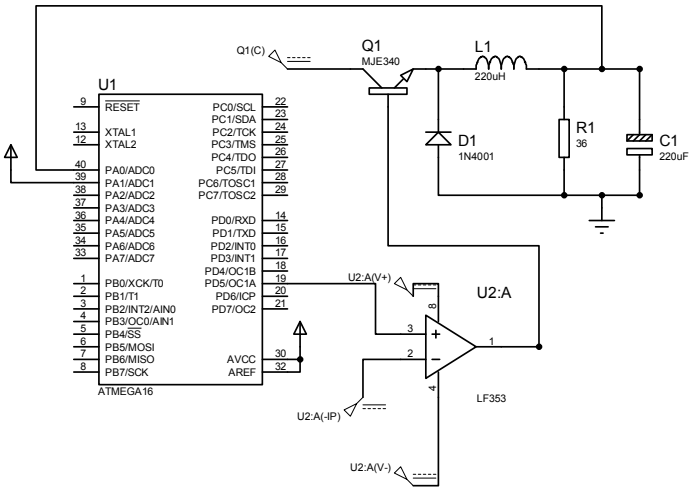
در برنامه ارائه شده در این بخش، از ورودی‌های آنالوگ به دیجیتال AD0 و AD1 به ترتیب برای نمونه برداری ولتاژ مرجع و ولتاژ خروجی مبدل استفاده شده است. مبدل آنالوگ به دیجیتال به صورت ۸ بیتی با قابلیت راه اندازی و توقف دستی تنظیم شده است. با توجه به نتایج شبیه‌سازی‌های انجام شده روی مبدل باک در نرم‌افزار Matlab، ضرایب k_p=0.01 و k_i=0.13 برای کنترل ولتاژ خروجی مبدل با مشخصات زیر مناسب بوده‌اند.



شکل(۹-۲): مبدل باک به همراه مدار فرمان



شکل(۹-۳): بلوک دیاگرام کنترل ولتاژ خروجی مبدل باک



شکل (۹-٤): سخت‌افزار مبدل باک توسط میکروکنترلر AVR

✓ برنامه

```
/*****************************************************
Chip type           : ATmega16
Clock frequency     : 8.000000 MHz
*****************************************************/
#include <mega16.h>
#include <delay.h>
#include <math.h>
#define b1 ADCH.7
#define b2 ADCH.0

// Declare your global variables here
bit b=0;
char n;
char d=100;
signed char a;
void main(void)
{
```

```
PORTA=0x00; DDRA=0x00;
PORTB=0x00; DDRB=0x00;
PORTC=0x00; DDRC=0x00;
PORTD=0x00; DDRD=0x20;

// Timer/Counter 0 initialization
TCCR0=0x00;TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x82;
TCCR1B=0x1A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=199;
OCR1AH=0x00;
OCR1AL=100;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
```

```
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

ADMUX=0x30;
ADCSRA=0x87;
while (1)
      {
start0:
delay_ms(10);
start:
for (n=0;n<3;n++)
{
ADCSRA=0xC6;
wait:
if (!ADCSRA.4)
goto wait;
ADCSRA.4=1;
if (b!=b1)
{
b=b1;
goto start;
}
delay_us(50);
}
a=ADCH;
if (b1)
{
```

```
#asm("neg r6")
d+=a;
}
else if(a>=d)
d=1;
else
d-=a;
if(d>199)
d=199;
OCR1AL=d;
goto start0;

    };
}
```

پروژه سی‌وهفتم: مولد موج PWM سینوسی

برنامه یک مولد موج PWM سینوسی را بنویسید؟

حل: یکی از روش‌های تولید موج PWM سینوسی به روش آنالوگ، مقایسه کردن سیگنال سینوسی مرجع با موج مثلثی یا دندانه اره‌ای می‌باشد. اگر اندازه‌ی لحظه ای موج سینوسی از موج دندانه اره‌ای بزرگتر باشد خروجی مقایسه کننده High و در غیر این صورت خروجی آن Low خواهد شد. سیگنال بدست آمده از این روش می‌تواند برای کنترل سوئیچ‌های قدرت یک اینورتر منبع ولتاژ استفاده شود و در این صورت می توان یک منبع ولتاژ سینوسی با قابلیت جریان دهی بالا ساخت. اگر f فرکانس موج سینوسی و f_c فرکانس موج دندانه اره‌ای باشد $m_f = \frac{fc}{f}$ به عنوان اندیس فرکانس مدولاتور شناخته می‌شود و در طیف سیگنال SPWM حاصل، مولفه‌های هارمونیک پایین حذف می‌شوند یا به عبارت دقیق‌تر بزرگترین مؤلفه‌های هارمونیکی قابل توجه در موج خروجی عبارتند از: m_f , m_f -2 , $2m_f$ -1 ,...

برای پیاده‌سازی این مولد به صورت دیجیتال باید با استفاده از نرم‌افزار Matlab زمان‌های تلاقی موج دندان اره‌ای با موج سینوسی را بدست آورده و این زمان‌ها در حافظه‌ی میکرو ذخیره شوند. سپس با استفاده از تایمرهای میکرو در حالت PWM این سیگنال را تولید نمود.

$$V_{tr} = \frac{t}{T_c} \quad \text{برای} \quad 0 < t < T_C \qquad (۷-۹)$$

$$V_{ref} = 0.5(1 + m \sin \omega t) \quad (۸-۹)$$

که $\omega = 2\pi f$ فرکانس زاویه‌ای موج سینوسی مرجع می‌باشد و m اندیس مدولاسیون دامنه نامیده می‌شود. از آنجا که زمان Low بودن سیگنال Spwm به سادگی از روی دوره تناوب (T_c) و زمان High آن بدست می‌آید، بنابراین کافی است که زمآن‌های Low در حافظه ذخیره شوند. اگر m_f به اندازه کافی بزرگ باشد، به دلیل تقارن نیم موج سیگنال خروجی Spwm زمان های Low و High در نیم سیکل مثبت، عکس این زمان ها در نیم سیکل منفی آن می‌باشند.

در برنامه نوشته شده فرکانس موج سینوسی 50 Hz و فرکانس موج دندانه اره‌ای 1KHz فرض شده است. بنابراین تعداد نمونه‌برداری‌ها از موج سینوسی ۲۰ نمونه می‌باشد. یک دوره تناوب موج سینوسی PWM تولیدی 1000 میکرو ثانیه می‌باشد و پس از تلاقی موج های سینوسی و دندانه اره‌ای ۲۰ مقدار مربوط به زمان های High موج PWM تولیدی بدست آمده‌اند که در آرایه High در برنامه زیر ذخیره شده‌اند.

✓ برنامه

```
#include <mega16.h>
  int i=1;
 flash unsigned int High [23]={ 593, 763, 895, 975, 999, 977, 913, 817, 699,
568, 433, 302, 184,
88, 24, 1, 26, 106, 238, 408 };
// Timer 1 output compare B interrupt service routine
interrupt [TIM1_COMPB] void timer1_compb_isr(void)
{
OCR1B=High[i];
i++;
if (i==20) (i=0);
}
void main(void)
{
PORTD=0x00;
DDRD=0x10;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
// Timer/Counter 1 initialization
```

```
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: Fast PWM top=OCR1A
// OC1A output: Discon.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x23;
TCCR1B=0x1A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x03;
OCR1AL=0xE8;
OCR1B=593;
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x08;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// Global enable interrupts
#asm("sei")

while (1);
}
```

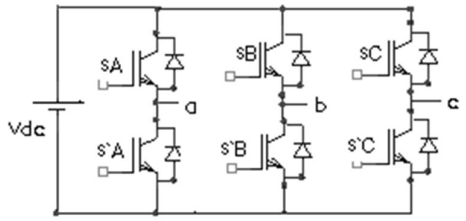
پروژه سی‌وهشتم: مولد پالس‌های اینورتر سه‌فاز SPWM

برنامه‌ای برای یک مولد پالس‌های اینورتر سه‌فاز SPWM بنویسید؟

حل: در شکل (۹-۵) یک اینورتر منبع ولتاژ سه فاز ترسیم شده است در این اینورتر سیگنال‌های فرمان دو سوئیچ در هر ساق اینورتر عکس یکدیگر می‌باشند. با توجه به این‌که در یک اینورتر سه فاز متقارن اختلاف فاز خروجی ها 120^0 می باشد، بنابراین کافی است، سیگنال‌های فرمان سوئیچ‌های هر فاز نسبت به هم اختلاف فاز 120^0 داشته باشند. به همین منظور اگر از یک جدول نظاره برای ذخیره زمآن‌های هدایت سوئیچ بالایی در فاز a اینورتر استفاده شود و تعداد درایه‌های این جدول N باشد، باید اشاره‌گر مربوط به زمآن‌های کلیدزنی هر فاز نسبت به فاز دیگر به اندازه N/3 اختلاف داشته باشد. به عنوان نمونه برای پیاده سازی SPWM سه فاز در پروژه قبل، اگر i اندیس مورد نظر برای زمآن‌های کلید زنی فاز a باشد، ۲۰٪ (i+7) و ۲۰٪ (i+14) به ترتیب اندیس‌های زمآن‌های کلیدزنی فازهای b و c خواهند بود. که ٪ به مفهوم باقیمانده تقسیم یک عدد صحیح بر عدد دیگر است و ۲۰، تعداد نمونه‌های مربوط به یک دوره تناوب موج سینوسی می‌باشد. لازم به ذکر است که سیگنال تحریک سوئیچ‌های پایینی هر ساق عکس سوئیچ‌های بالا می‌باشد.

توجه: در خاموش و روشن شدن سوئیچ‌ها دقت شود که اگر بلافاصله پس از خاموش کردن یک سوئیچ، سوئیچ دیگر آن ساق روشن شود، امکان اتصال کوتاه شدن منبع وجود دارد. لذا باید سعی شود با استفاده از یک مدار سخت افزاری در لبه بالا رونده هر سیگنال تحریک تاخیر مناسب (در حد یک یا چند میکرو ثانیه) ایجاد گردد.

توضیح برنامه: از آنجا که تولید SPWM سه فاز احتیاج به ۳ تایمر دارد که بتواند فرکانس 1KHz را تولید کند و با توجه به این‌که تایمر صفر و دو در مد PWM فرکانس های مشخصی را با پالس ورودی معین (در این مثال 8MHz) می‌توانند تولید کنند و فرکانس 1KHz را با کلاک پالس 8MHz نمی‌توانند تولید کنند، در این برنامه برای تولید شکل موج PWM از مد CTC تایمر 0، 1 و 2 استفاده شده است. به این ترتیب که ابتدا مقدار High در رجیستر OCRx بارگذاری شده و هنگامی که وقفه مقایسه تایمر رخ دهد، مقدار Low در رجیستر OCRx بارگذاری می‌شود. اما از آنجایی که شکل موج CTC با سطح منطقی Low شروع می‌شود و در رجیستر OCRx ابتدا مقدار High بارگذاری می‌شود باید خروجی سه پایه OCR0، OCR1A و OCR2 توسط یک گیت NOT معکوس شود.



شکل (۹-۵): اینورتر منبع ولتاژ سه فاز

✓ برنامه :

```
/*****************************************************
Chip type           : ATmega16
Clock frequency     : 8.000000 MHz
*****************************************************/

#include <mega16.h>
  int i=1,j=1,k=1;
 flash unsigned int High0 [43]={0x57, 0x25, 0x47, 0x36, 0x36, 0x46, 0x25,
0x57, 0x17, 0x66, 0xb, 0x72, 0x3, 0x7a,0x3, 0x79, 0xd, 0x6f , 0x1d, 0x5f,
0x33, 0x4a, 0x4a, 0x32 , 0x5f, 0x1d, 0x6f, 0xd, 0x9, 0x3, 0x7a, 0x2, 0x72,
0xa,0x66, 0x16};
flash unsigned int High1 [43]={593, 407, 763, 237, 895, 105, 975, 25, 977,
23, 913, 87, 817, 183, 699, 301, 568, 432, 433, 567 , 302, 698, 184, 816,
88, 912, 24, 976, 26, 974, 106, 894, 238, 762, 408 , 592};
 flash unsigned int High2 [43]={ 0x3, 0x79, 0xd, 0x6f , 0x1d, 0x5f, 0x33,
0x4a, 0x4a, 0x32 , 0x5f, 0x1d, 0x6f, 0xd, 0x9,0x3, 0x7a, 0x2, 0x72, 0xa,
0x66, 0x16, 0x57, 0x25, 0x47, 0x36, 0x36, 0x46, 0x25, 0x57, 0x17, 0x66,
0xb, 0x72, 0x3, 0x7a};

// Timer 0 output compare interrupt service routine
interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
OCR0=High0[j];
j++;
if (j==36) (j=0);
}

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
```

```
OCR1A=High1[i];
i++;
if (i==36) (i=0);
}

// Timer 2 output compare interrupt service routine
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
OCR2=High2[k];
k++;
if (k==36) (k=0);
}


void main(void)
{
PORTA=0x00; DDRA=0x00;
PORTB=0x00; DDRB=0x08;
PORTC=0x00; DDRC=0x00;
PORTD=0x00; DDRD=0xA0;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 125.000 kHz
// Mode: CTC top=OCR0
// OC0 output: Toggle on compare match
TCCR0=0x1B;
TCNT0=0x00;
OCR0=0x57;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Toggle
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x40;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
```

```
ICR1L=0x00;
OCR1A=593;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 125.000 kHz
// Mode: CTC top=OCR2
// OC2 output: Toggle on compare match
ASSR=0x00;
TCCR2=0x1C;
TCNT2=0x00;
OCR2=0x0d;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x92;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// Global enable interrupts
#asm("sei")

while (1);
}
```

پروژه سی‌ونهم: طراحی یک دیمر

برنامه یک دیمر را بنویسید؟

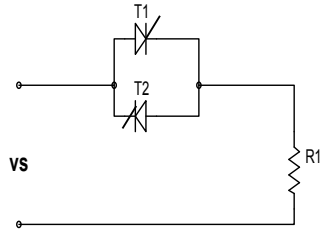
☑ حل: معمولا برای تنظیم شدت نور یک لامپ، از سوئیچ قابل کنترل الکترونیک قدرت که قابلیت قطع و وصل جریآن‌های AC را داراست، استفاده می‌شود. به همین منظور: معمولا سوئیچ تریاک (Triac) با مدار گیت قابل تنظیم به کار گرفته می‌شود. اگر ولتاژ برق شهر را به صورت سینوسی در نظر بگیریم ولتاژ دو سر لامپ مطابق شکل (۹-٦) به صورت سینوسی برش یافته خواهد بود. میزان تاخیر در روشن شدن سوئیچ نسبت به لحظه‌ی گذر از صفر ولتاژ برق شهر را زاویه‌ی

آتش می‌نامند. مقدار rms ولتاژ دو سر لامپ بر اساس رابطه (۹-۹) با تغییر زاویه ی آتش تغییر می‌کند.

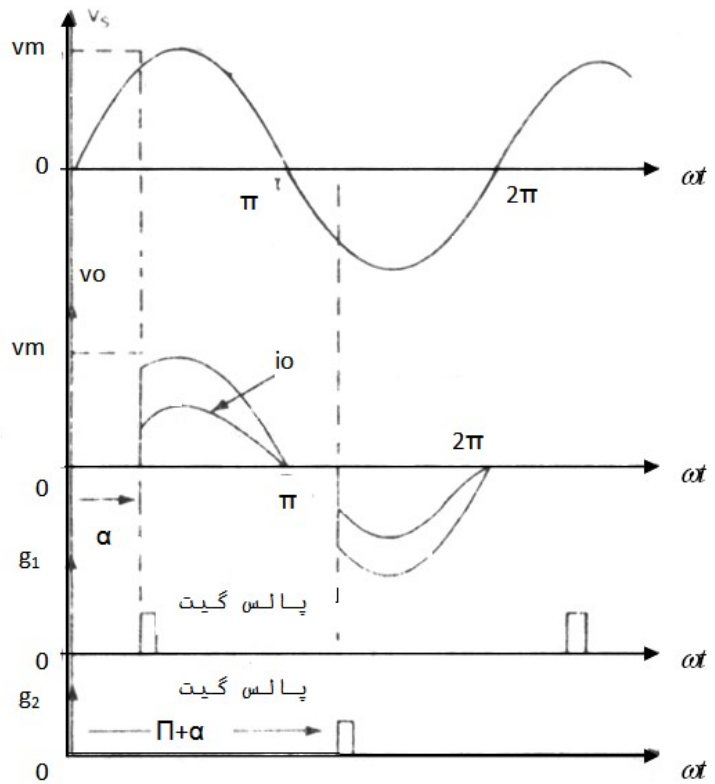
$$Vo = Vs[\frac{1}{\pi}(\pi - \alpha + \frac{\sin 2\alpha}{2})]^{\frac{1}{2}} \qquad (۹-۹)$$

در مدار شکل (۹-۷) از یک مقسم ولتاژ و مدار تشخیص گذر از صفر با استفاده از آپ- امپ LM339 برای ساخت پالس سنکرون کننده استفاده شده است و با استفاده از ترانزیستور A733 لحظات روشن شدن تریاک کنترل شده است.

توضیح برنامه: می‌توان زاویه‌ی آتش (شدت نور) را در ۱۰ سطح مختلف توسط کلیدهای up و down تنظیم نمود؛ سطح انتخاب شده در متغیر y که از نوع eeprom است ذخیره می‌شود. استفاده از eeprom به این دلیل است که اگر برق قطع شد پس از راه اندازی مجدد، دیمر به حالت قبل از قطع برق برگردد. متغیر y به متغیری به نام g نسبت داده می‌شود، این متغیر از جنس eeprom نیست و در طول برنامه به جای متغیر y استفاده می‌شود و چنانچه تغییری در متغیر g ایجاد گردد این تغییر توسط دستور y=g; به y انتساب داده می‌شود تا این متغیر بهنگام شود. با ارسال هر پالس بالا رونده توسط مدار آشکار ساز گذر از صفر، برنامه وارد سرویس وقفه می‌شود. در ابتدای سرویس وقفه تریاک با دستور PORTA.3=1 خاموش می‌شود (پورت A.3 به ترانزیستور کنترل کننده‌ی تریاک متصل می‌باشد) سپس با توجه به سطح زاویه‌ی آتش انتخاب شده تاثیری به اندازه ی 200×(g+1) میکروثانیه ایجاد می‌شود و در پایان سرویس وقفه تریاک توسط دستور PORTA.3=0 دوباره روشن می‌شود.



شکل(۹-۶): مدار کنترل نور لامپ با استفاده از یک تریستور دو طرفه یا تریاک



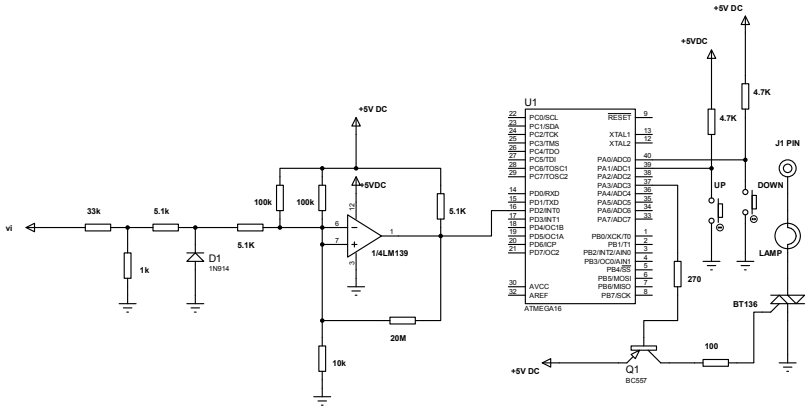
شکل(۹-۷): خروجی و پالسهای گیت در مدار دیمر

✓ برنامه

```
/*****************************************
Chip type           : ATmega16
Clock frequency     : 8.000000 MHz
*****************************************/

#include <mega16.h>
#include <delay.h>
eeprom int y;
int g,k;
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
 PORTA.3=1;
```

```
 for (k=0;k<=g;k++)
            {delay_us(200);}
 PORTA.3=0;
 }

void main(void)
{
DDRA.0=0;DDRA.1=0;DDRA.3=1;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x03;
MCUCSR=0x00;
GIFR=0x40;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
ACSR=0x80;
SFIOR=0x00;

// Global enable interrupts
#asm("sei")
if (y>9) (y=9);
g=y;
k=0;
PORTA.3=1;
while (1)
    {
    if (PINA.1==0) {
            if (g<9) g++;
            delay_ms(10);
            y=g;
            }
    // Place your code here
    if (PINA.0==0) {
            if (g>1) g--;
            delay_ms(10);
            y=g;
            }
```

```
    };
}
```



شکل(۹-۸): مدار کنترل دیمر با میکروکنترلر

فصل دهم پروژه‌های کاربردی دیگر

پروژه چهلم: اسیلوسکوپ دیجیتال با استفاده از میکروکنترلر

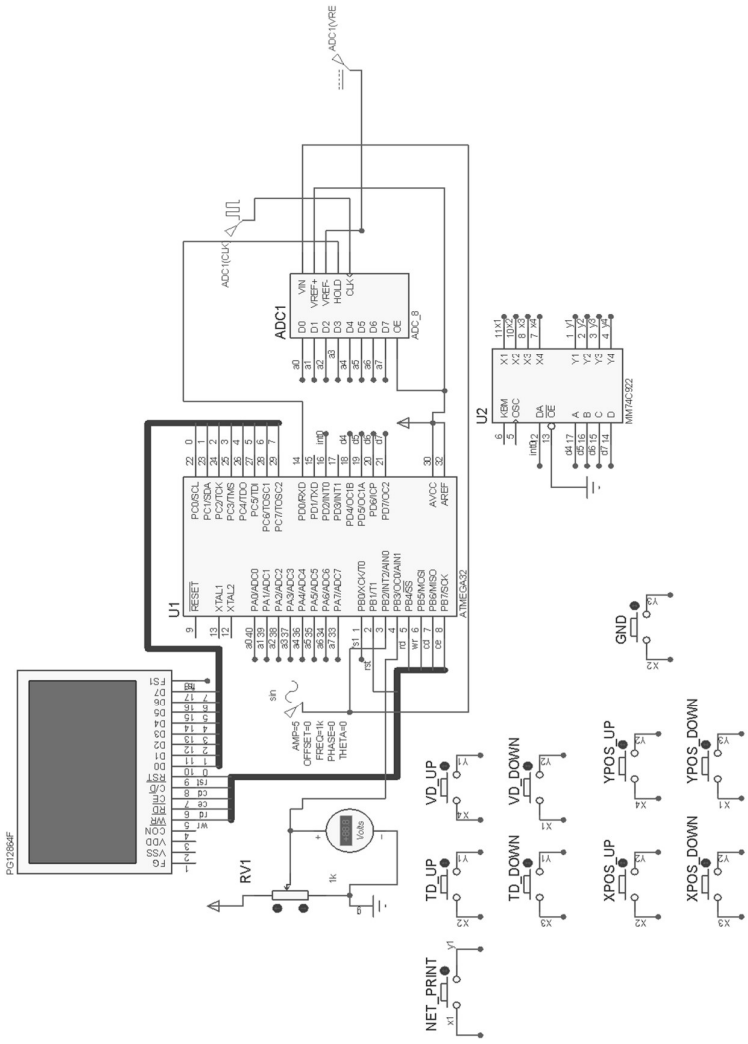
برنامه یک اسیلوسکوپ دیجیتال را با استفاده از میکروکنترلر AVR بنویسید؟

حل: اسیلوسکوپ، نوسان‌سازی است که برای نمایش دو بعدی سیگنال‌های متغیر با زمان استفاده می‌شود. در این پروژه، روش ساخت یک اسیلوسکوپ دیجیتال با میکروکنترلر ATmega32 را بیان می‌کنیم. این اسیلوسکوپ دارای یک LCD گرافیکی ۱۲۸×٦٤ و یک میکروکنترلر می‌باشد که توسط یک کریستال خارجی ١٦MHz با حداکثر سرعت کار می‌کند. برای افزایش سرعت نمونه‌برداری از یک ADC هشت بیتی خارجی استفاده شده است. تمام عملکرد سیستم شامل: تنظیم ضریب بهره، انتخاب کننده زمان، محور عمودی و نظایر آن به صورت دیجیتال انجام می‌شود. سیستم قابلیت دریافت ولتاژ مثبت و منفی بین۵- تا ۵ ولت را دارا می‌باشد و همچنین قابلیت تغییر Volt Division،Time Division و Trigger Level را نیز دارد. همچنین از امپدانس ورودی بالایی در حدود یک مگا اهم برخوردار است. ساختار سخت‌افزاری این اسیلوسکوپ مطابق شکل (۱۰-۱) است که هر قسمت در بخش‌های بعد معرفی می‌شوند.

مبدل آنالوگ به دیجیتال خارجی ADC804 : دلیل استفاده از مبدل خارجی در این پروژه: سرعت پایین مبدل ADC میکرو است. زیرا در صورت استفاده از حالت تک تبدیل، به دلیل تنظیمات اولیه‌ی ADC هر تبدیل حدود ۲۵ سیکل طول می‌کشد، درحالی که در وضعیت Free Running هر تبدیل برای کامل شدن حدود ۱۳ کلاک زمان نیاز دارد. با استفاده از این مبدل، ولتاژ ورودی به نسبتی از ولتاژ مرجع ماکزیمم و مینیمم تقسیم می‌شود و در مقدار ۲۵۵ ضرب می‌گردد و با لبه بالا رونده پایه hold روی خروجی قرار می‌گیرد.

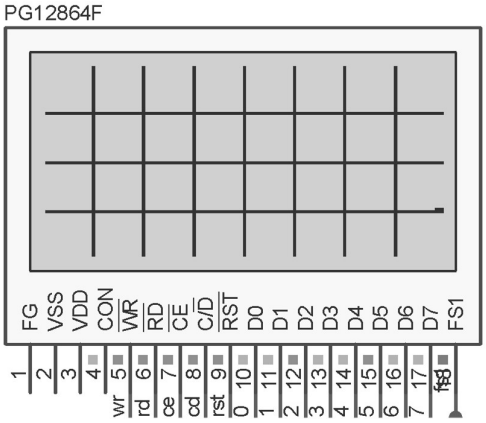
مبدل صفحه کلید MM74C922 : این مبدل در فصل سوم تشریح شده است. خروجی این تراشه (همان طور که در فصل چهار توضیح داده شد) به ٤ بیت با ارزش بیشتر پورت D متصل شده است. پایه da آن به وقفه صفر متصل شده است و کلیدها مطابق شکل بعدی با برچسب گذاری به آن متصل شده‌اند. با

فشرده‌شدن هر کلید عددی بین ۰ تا ۱۵ روی خروجی ظاهر می‌شود و یک وقفه سخت‌افزاری نیز در میکرو اتفاق می‌افتد.



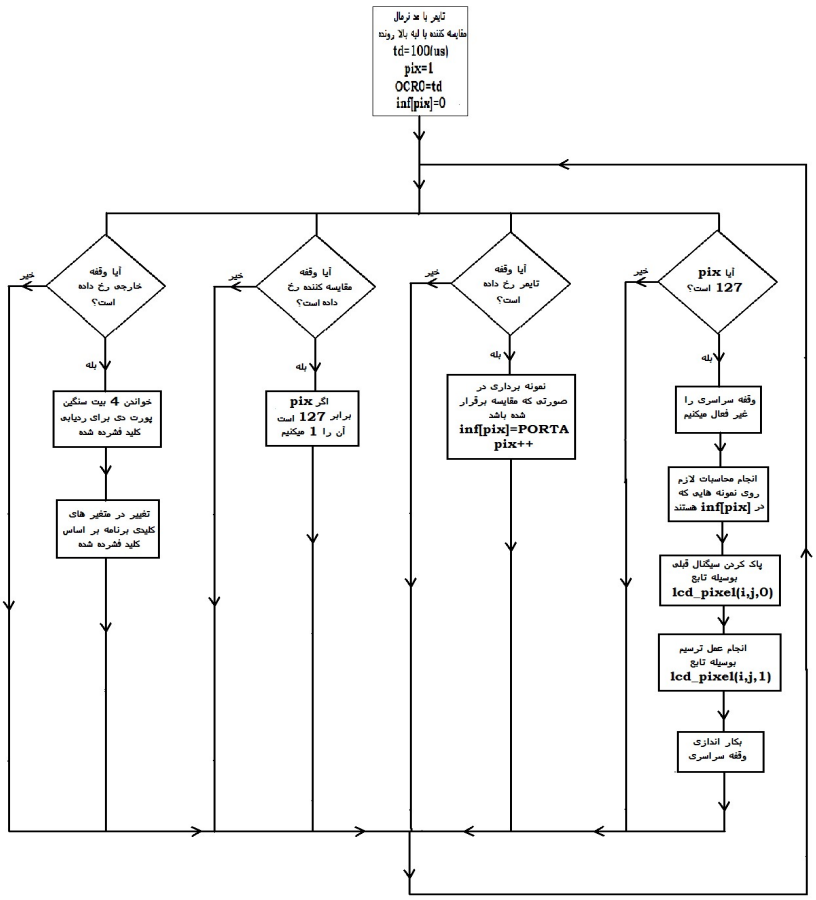
شکل (۱۰-۱): شمای کلی اسیلوسکوپ دیجیتال با میکروکنترلر در محیط Proteus

تقسیم نرم افزاری LCD : به دلیل عدم استفاده از برچسب بر روی LCD برای نمایش خطوط افقی و عمودی اسکوپ، به طور نرم‌افزاری تقسیم‌بندی LCD به صورت شکل (۱۰-۲) انجام می‌شود. بنابراین می‌توان دید که هر خانه اسکوپ ۱۶ در ۱۶ پیکسل خواهد داشت.



شکل (۱۰-۲): تقسیمات اسیلوسکوپ

ساختار برنامه نوشته شده در شکل (۱۰-۳) آورده شده است. توضیحات مربوط به هر بخش نیز در برنامه مربوطه آمده است. شکل مدار و فایل اجرایی آن نیز در CD ضمیمه قرار دارد.



شکل (۱۰-۳): فلوچارت برنامه اسیلوسکوپ دیجیتال با میکروکنترلر

شرح برنامه:

اضافه کردن کتابخانه‌های مورد نیاز:

```
#include <mega32.h>
#include <delay.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <delay.h>
```

به پایه‌های نمایش‌گر اسم‌هایی که برای کار ساده‌تر است، اختصاص داده شده است:

```
#define LCD_WR PORTB.5
#define LCD_RD PORTB.4
#define LCD_CE PORTB.7
#define LCD_CD PORTB.6
#define LCD_RST PORTB.1
#define LCD_FS1 PORTB.0
```

دستوراتی که به نمایش‌گر داده می‌شوند، هر کدام کد مخصوص به خود دارند که برای عدم نیاز به مراجعات پیاپی اسم‌های زیر به آن‌ها اختصاص داده شده است:

```
#define eightbyeight 0
#define busy1busy2 0x03
#define DAWRDY 0x08
#define cursorpointerset 0X21
#define offsetregisterset 0x22
#define addresspointerset 0x24
#define texthomeaddress 0x40
#define textareaset 0x41
#define graphichomeaddress 0x042
#define graphicareaset 0x43
#define cgrommode 0x80
#define cgrammode 0x88
#define ormode 0x80
#define exormode 0x81
#define andmode 0x83
#define textonly 0x84
#define graphicsoff 0x90
#define graphicson 0x98
#define textoff 0x90
#define texton 0x94
#define cursoroff 0x90
#define cursoron 0x92
#define cursorblinkoff 0x90
#define cursorblinkon 0x91
#define cursorpattern 0xa0
#define bottomline 0x0
#define dataautowrite 0xb0
#define dataautoread 0xb1
#define autoreset 0xb2
#define datardwr 0xc0
#define bitreset 0xf0
```

```
#define bitset 0xf8
#define graphbase 0x1000
#define textbase 0x0
#define column 128
#define row 64
char fontsize;
unsigned int charsperrow;
```

الگوی توابع برنامه به صورت زیر هستند:

```
void lcd_init();
void statbusy12(void );
void statdawardy(void );
void dataout(char b);
void commandout(char b);
void lcd_clear_graphic();
void lcd_clear_text();
void lcd_print(char *string);
void lcd_pixel(int x,int y,char setreset);
void lcd_xy(int x,int y);
void setautomode();
void autoout(char b);
void resetaotomode();
void lcd_cursorxy(char x,char y);
void dataout2(int b);
void autozero(unsigned int terminate);
void print_net();
void clear_signal();
void text_td_vd();
```

متغیرهای عمومی برنامه را به صورت زیر تعریف می‌کنیم:

```
unsigned char pix=127,inf[128],prev_inf[128],net=0,key;
unsigned long int td=200,compare,vd=5000;  //usec
char code[10];
```

در روال وقفه زیر که با تطابق وقفه تایمر رخ می‌دهد، متغیر پیکسل را افزایش می دهیم. در واقع، تایمر با توجه به مقدار Time Division می‌گوید که چه موقع برای پس خواندن اطلاعات پیکسل بعدی مناسب است.

```
// Timer 0 output compare interrupt service routine
interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
if(OCR0<255){ #asm("sei")
```

با توجه به این که LCD ۱۲۸ پیکسل در طول خود دارد عدد صفر مبین شروع پیکسل‌ها و عدد ۱۲۷ نشان می‌دهد که پیکسل‌ها تمام شده است.

```
if(pix!=127)
{
 pix++;
TCNT0=0;
```

حال با دستورات زیر یک لبه به مبدل آنالوگ به دیجیتال خارجی می‌دهیم تا مقدار ولتاژ را روی پورت D به ما بدهد.

```
PORTD.0=0;
inf[pix]=PINA;
PORTD.0=1;}
```

برای Time Divisions بزرگ که در رجیستر ۸ بیتی مقایسه جا نمی‌شوند دستورات زیر مقدار مقایسه را به واحدهای ۲۵۵ تایی می‌شکنند. توجه کنید که باید مقدار مقایسه در رجیستر مقایسه تایمر ریخته شود تا تایمر در موقع لازم اعلام کند که زمان نمونه برداری فرا رسیده است.

```
Compare=td;}
if(compare>255)
{
OCR0=255;
compare-=255;
}
else
 OCR0=compare;
}
```

تا زمانی که شکل موج به نقطه‌ی خاصی نرسد، ۱۲۷ پیکسل می‌ماند. پس از رسیدن به مقدار تریگر با این روال وقفه ۱ پیکسل می‌شود و همان‌طور که خواهیم دید عمل ترسیم روی نمایش‌گر گرافیکی شروع می‌شود.

```
// Declare your global variables here
// Analog Comparator interrupt service routine
interrupt [ANA_COMP] void ana_comp_isr(void)
{
if(pix==127)
pix=1;
}
```

در وقفه ی صفر اگر کلید صفر فشرده باشد، یعنی: باید جدول اسکوپ را رسم کنیم.

```
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
```

```
if(PIND/16==0){
net=1-net;
if(net)print_net();
else
lcd_clear_graphic();
}
else
 while(PIND/16)
{
text_td_vd();
while(PIND.2);
```

در آنجا منتظر کلیدهای بعدی می‌مانیم تا اینکه کلید صفر فشرده شود و تغییراتی که در Time Division و .. داده شده، تایید گردد.

```
while(!PIND.2);
key=PIND/16;
```

کلید ۱ و۲ یعنی تغییر Time Division

```
if(key==1) td=td*10;
if(key==2) td=td/10;
```

کلید ۳ و٤ یعنی تغییر Volt Division

```
if(key==3) vd=vd*10;
if(key==4) vd=vd/10;
```

کلید ٥ و٦ و۷ و ۸ یعنی تغییر موقعیت.

```
if(key==5) xpos++;
if(key==6) xpos--;
if(key==7) ypos++;
if(key==8) ypos--;
```

حال برای برگشتن به برنامه آماده می‌شویم. یعنی از مد نوشتاری LCD که در ادامه گفته خواهد شد، خارج می‌شویم.

```
if(PIND/16==0)
{
lcd_clear_text();
commandout(graphicson|textoff|cursoroff|cursorblinkoff);}
//while(PIND.2);
}
}
char st[1024];
//**************************************************
```

```
void main(void)
{
```

متغیر های محلی تابع main را به صورت زیر تعریف می‌کنیم:

```
unsigned char y;
int i=0 , xpos=0,ypos=0;
float mf;
```

جهت دهی پورت ها به صورت زیر می‌باشد:

```
DDRB.2=0;
DDRB.3=0 ;
PORTB.2=0;
PORTB.3=0;
DDRD=3;
DDRA=0;
```

اطلاعات در این رشته ریخته می‌شوند. باید اول آن را ۰ کنیم.

```
for(i=0;i<1023;i++){
st[i]=0;
prev_inf[i]=0;
}
```

تنظیمات اولیه LCD انجام می‌شود.

```
lcd_init();
commandout(graphicson|textoff|cursoroff|cursorblinkoff);
```

وقفه با لبه بالا رونده حاصل از فشردن کلیدها فعال شده است.

```
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x03;
MCUCSR=0x00;
GIFR=0x40;
```

تایمر با رسیدن به مقدار مقایسه باید یک وقفه ایجاد کند.

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 16000.000 kHz
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x01;
TCNT0=0x00;
OCR0=td;
```

فعال‌سازی وقفه صفر:

```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x02;
// Analog Comparator initialization
// Analog Comparator: On
```

مقایسه کننده در هر بار تریگر شدن اسکوپ باید وقفه ایجاد کند. در این صورت خواهد گفت که شروع شکل موج اینجاست.

```
// Interrupt on Rising Output Edge
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x0B;
SFIOR=0x00;
```

فعال سازی وقفه سراسری:

```
#asm("sei")
while(1)
{
```

در بدنه دائمی برنامه، هر بار نوشتار لازم را انجام می‌دهیم.

```
lcd_print(code);
```

اگر پیکسل ۱۲۷ شود، یعنی اینکه تمام ۱۲۸ پیکسلی که در طول LCD هستند در وقفه تایمر ولتاژشان خوانده شده است. پس باید ترسیم را شروع کنیم.

```
if(pix==127){
```

وقفه کلی را از کار انداخته تا سرعت ترسیم کم نشود.

```
#asm(“cli”)
```

سیگنال قبلی را از LCD پاک می‌کنیم.

```
clear_signal();
```

با توجه به این که LCD را به ۸ در ٤ خانه تقسیم کرده‌ایم ولتاژ هر خانه را ترسیم می کنیم.

```
For(i=1;i<128;i++)
{
mf=inf[i];
mf=10*mf/255-5;
mf=(16*mf/vd)*1000;
y=32+mf;
prev_inf[i]=y;
x=x+xpos;
y=y+ypos;
lcd_pixel(I,y,1);
}
```

```
#asm("sei")
```

و تا شروع دوباره شکل موج صبر می‌کنیم.

```
while(pix!=1);
}
}
}
//**************************************************
```

با این تابع مقادیر تایم و ولت دیویژن را به رشته تبدیل کرده و می‌نویسیم.

```
void text_td_vd()
{
lcd_clear_graphic();
lcd_clear_text();
commandout(graphicsoff|texton|cursoroff|cursorblinkoff);
lcd_xy(1,6);
if(td<1000)sprintf(code,"td=%uus        vd=%umv",td,vd);
else
 sprintf(code,"td=%ums        vd=%umv",td/1000,vd);
lcd_print(code);
}
//**************************************************
```

با اطلاعات قبلی که در بدنه اصلی آن را در prev_inf ذخیره کردیم در این تابع سیگنال را پاک می‌کنیم.

```
void clear_signal()
{
int i;
for(i=1;i<128;i++)
if(i%16)if(prev_inf[i]%16)
lcd_pixel(i,prev_inf[i],0);
}
//**************************************************
```

در این تابع با رسم خطوطی جدول مشبک اسکوپ را رسم می‌کنیم.

```
void print_net()
{
int i,j;
for(i=1;i<8;i++)for(j=1;j<63;j++)lcd_pixel(16*i,j,1);
for(i=1;i<128;i++)for(j=1;j<8;j++)lcd_pixel(i,j*16,1);
}
```

//**********

راه اندازی LCD :

```
void lcd_init()
{
PORTB=0B11010011;
DDRB=0xf3;
PORTC=0;
DDRC=0xff;
LCD_CE=1;
LCD_RD=1;
LCD_WR=1;
LCD_CD=1;
LCD_RST=0;
```

با این کار LCD منتظر دستور می‌ماند. سپس آن را ریست کرده و پیکربندی فونت را انجام می‌دهیم.

```
delay_us(1);
LCD_RST=1;
delay_us(1);
LCD_FS1=eightbyeight;
if(LCD_FS1==eightbyeight)
{
fontsize=8;
charsperrow=column/fontsize;
}
else
{
fontsize=6;
charsperrow=column/fontsize;
}
```

سپس در RAM مربوط به LCD بخشی را به گرافیک و بخشی را به نوشتار اختصاص می‌دهیم.

```
 commandout(graphicsoff|textoff|cursoroff);
 dataout2(graphbase);
 commandout(graphichomeaddress);
  dataout2(charsperrow);
  commandout(graphicareaset);
  dataout2(textbase);
  dataout2(texthomeaddress);
  dataout2(charsperrow);
```

```
 commandout(textareaset);
 commandout(exormode|cgrommode);
 commandout(cursorpattern|7);
 lcd_clear_graphic();
 lcd_clear_text();
 lcd_cursorxy(0,0);
 }
```

در این تابع، گرافیک‌ها را از LCD پاک می‌کنیم.

```
 //**************************************************
 void lcd_clear_graphic()
{
  dataout2(graphbase);
 commandout(addresspointerset);
 autozero(charsperrow*row);
 }
//**************************************************
 void lcd_clear_text()
{
  dataout2(textbase);
  commandout(addresspointerset);
   autozero(charsperrow*row/8);
   }
```

یک رشته به این تابع وارد می‌شود و در حالت اتوماتیک LCD کاراکتر به کاراکتر چاپ می‌شود.

```
//**************************************************
void lcd_print(char *string)
{
char *p;
p=string;
setautomode();
while(*p)
autoout(*p++ -0x20);
resetaotomode();
}
//**************************************************
```

در مهمترین تابع، ما ابتدا جایگاه فونتی ۸ در ۸ برای پیکسل یافت شده و از این ٦٤ خانه یکی خاموش یا روشن می‌شود.

```
void lcd_pixel(int x,int y,char setreset)
{
```

```
char  Bit;
Bit=(fontsize-1)-(x%fontsize);
dataout2(graphbase+(y*charsperrow)+(x/fontsize));
commandout(addresspointerset);
if(setreset)
commandout(bitset |Bit);
else
commandout(bitreset|Bit);
}
```

تعیین موقعیت نشانگر LCD در این تابع انجام می‌شود.

```
//**************************************************
void lcd_xy(int x,int y)
{
dataout2(textbase+(y*charsperrow)+x);
commandout(addresspointerset);
}
//**************************************************
```

اگر نشانگر چشمک زن فعال باشد، با این تابع به موقعیت مطلوب می‌رود.

```
void lcd_cursorxy(char x,char y)
{
dataout(x);
dataout(y);
commandout(cursorpointerset);
}
//**************************************************
```

برای پاک کردن LCD از این تابع کمک می‌گیریم. با مد اتوماتیک LCD اطلاعات را پاک می‌کنیم.

```
void autozero(unsigned int terminate)
{
setautomode();
while(terminate--)  autoout(0);
resetaotomode();
}
//**************************************************
```

اگر LCD مشغول باشد این تابع برنامه را نگه میدارد تا تداخل دستوری رخ ندهد.

```
void statbusy12(void ){
char lcd_status;
DDRC=0x00;
do{
```

```
LCD_CE =0;
LCD_RD=0;
delay_us(1);
lcd_status=PINC;
LCD_CE =1;
LCD_RD=1;
}
while((lcd_status & busy1busy2)!=busy1busy2);
DDRC=0xff;
}
//**************************************************
```

برای آگاهی از موقعیت LCD داریم:

```
void statdawardy(void )
{
 char lcd_status;
DDRC=0x00;
do
{
LCD_CE =0;
LCD_RD=0;
delay_us(1);
lcd_status=PINC;
LCD_CE =1;
LCD_RD=1;
}
while((lcd_status & DAWRDY)!=DAWRDY);
DDRC=0xff;
}
//**************************************************
```

این تابع معلوم می‌کند که کد فرستاده شده به پورت LCD داده است.

```
void dataout(char b)
{
statbusy12();
LCD_CD=0;
PORTC=b;
LCD_CE=0;
LCD_WR=0;
LCD_CE=1;
LCD_WR=1;
LCD_CD=1;
```

```
}
//**************************************************
```

محتوای داده نیز با این تابع فرستاده می‌شود.

```
void dataout2(int b)
{
dataout(b & 0xff);
dataout(b >> 8);
}
//**************************************************
```

این تابع به جای داده، دستور می‌فرستد.

```
void commandout(char b){
statbusy12();
PORTC=b;
LCD_CE=0;
LCD_WR=0;
LCD_CE=1;
LCD_WR=1;
}
//**************************************************
```

در مد اتوماتیک بین داده‌های ارسالی نیاز به ایجاد دستور نداریم. اطلاعات به صورت داده‌ای فرض می‌شوند.

```
void setautomode()
{
commandout(dataautowrite);
}
//**************************************************
```

این تابع از تابع قبلی در مد اتوماتیک کمک می‌گیرد.

```
void autoout(char b)
{
statdawardy();
LCD_CD=0;
PORTC=b;
LCD_CE=0;
LCD_WR=0;
LCD_CE=1;
LCD_WR=1;
LCD_CD=1;
}
\\**************************************************
```

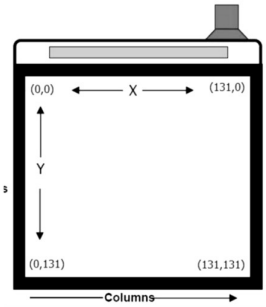
با این تابع LCD از حالت اتوماتیک خارج می‌شود تا داده‌های بعدی اعضای یک رشته فرض نشوند.

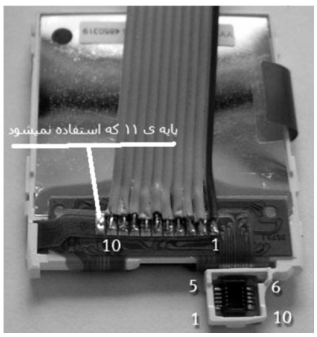
```
void resetaotomode()
{
statdawardy();
PORTC=autoreset;
LCD_CE=0;
LCD_WR=0;
LCD_CE=1;
LCD_WR=1;
}
```

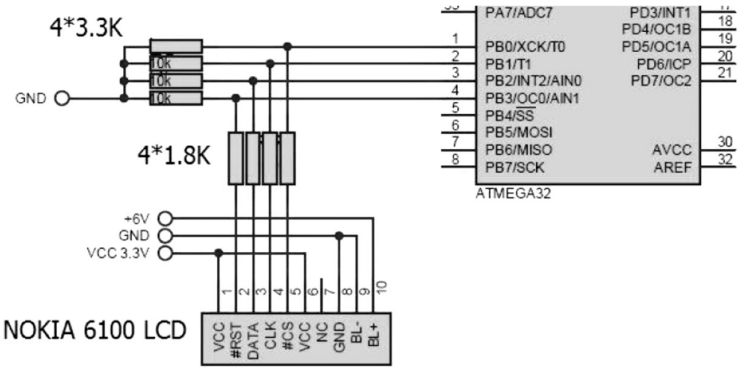
پروژه چهل‌ویکم: ارتباط میکروکنترلر و TFT موبایل (LCD موبایل)

الف- برنامه‌ای بنویسید که میکروکنترلر AVR عکس رنگی ذخیره شده بر روی حافظه Flash خود را خوانده و بر روی TFT موبایل (صفحه نمایش) متصل به آن نمایش دهد؟

شرح: در این پروژه از LCD موبایل مربوط به گوشی‌های شماره Nokia 6100 استفاده شده است. این LCD ها دارای ۱۳۲×۱۳۲ پیکسل هستند که در دو نوع PCF8833 و S1D15G00 در بازار موجود می‌باشند. در این پروژه، LCD دوم مورد نظر است و برنامه نوشته شده برای PCF8833 است. تنها راه تشخیص این دو LCD از یکدیگر، رنگ مدار چاپی پشت آن‌ها است که S1D15G00 دارای بورد قهوه‌ای یا نارنجی رنگ است و PCF8833 دارای بورد مدار چاپی سبز رنگ می‌باشد. شایان ذکر است که این LCD از پروتکل SPI پشتیبانی می‌کند ولی در این پروژه از این پروتکل به منظور ارسال تصاویر استفاده نمی‌شود. ولتاژ کاری این LCD برابر با ۳/۳ ولت است. بنابراین استفاده از رگولاتور ولتاژ الزامی است. مدار این پروژه مطابق شکل (۱۰-٤-الف) است و همچنین این شکل، نحوه قرار گرفتن پایه‌های LCD را مشخص می‌کند. به منظور تولید کدهای مربوط به LCD موبایل، نرم افزار موجود در CD همراه کتاب پیشنهاد می‌شود که توسط آن، ابتدا عکس مورد نظر طراحی یا بارگذاری می‌شود و سپس کد مربوط به آن تولید و توسط برنامه استفاده می‌شود.







شکل (۱۰-٤): نحوه ارتباط LCD موبایل به میکرو

✓ برنامه

```
#include <mega32.h>
#include <delay.h>
//          you have to select one of these three preprocessor's
```

```
#define COLOR_INTERFACE_16     // for use 16 bit color interface
#define COLOR_INTERFACE_12     // for use 12 bit color interface
#define COLOR_INTERFACE_8        // for use 8 bit color interface
#define CLCD_PORT PORTB
#define CLCD_DDR DDRB
#define CS 0
#define CLK 1
#define SDA 2
#define RST 3

#define CS_0  CLCD_PORT.CS=0
#define CS_1  CLCD_PORT.CS=1
#define CLK_0 CLCD_PORT.CLK=0
#define CLK_1 CLCD_PORT.CLK=1
#define SDA_0 CLCD_PORT.SDA=0
#define SDA_1 CLCD_PORT.SDA=1
#define RST_0 CLCD_PORT.RST=0
#define RST_1 CLCD_PORT.RST=1

#define SLEEP_OUT          0x11
#define INVON              0x21
#define NORMAL_DISPLAY     0x13
#define COLOR_MODE         0x3A
#define MEMORY_ACCESS_CTRL 0x36
#define DIPLAY_ON          0x29
#define ROW_ADDRESS_SET    0x2B
#define COLUMN_ADDRESS_SET 0x2A
#define RAM_WRITE          0x2C
#define SET_CONTRAST       0x25
#define SOFTWARE_RESET     0x01
#define BOOSTER_ON         0x03
#define DATA_ORDER         0xBA

#ifdef COLOR_INTERFACE_16
#define BLACK   0x0000
#define RED     0xF800
#define GREEN   0x07E0
#define BLUE    0x001F
#define WHITE   0xFFFF
#define PINK    0xF81F
#define YELLOW  0xFFE0
```

```
#endif
#ifdef COLOR_INTERFACE_12
#define BLACK   0x000
#define BLUE    0x00F
#define GREEN   0x0F0
#define CYAN    0x0FF
#define RED     0xF00
#define MAGENTA 0xF0F
#define YELLOW  0xFF0
#define WHITE   0xFFF
#define BROWN   0xB22
#define ORANGE  0xFA0
#define PINK    0xF6A
#endif
#ifdef COLOR_INTERFACE_8
#define BLACK   0x00
#define RED     0xE0
#define GREEN   0x3C
#define BLUE    0x03
#define WHITE   0xFF
#endif

void _Clcd_write_command_(unsigned char);
void _Clcd_write_data_(unsigned char);
void Clcd_clear_screen(void);
void Clcd_set_contrast(char);
void Clcd_write_pixel(unsigned char,unsigned char,int);
void Clcd_draw_line(unsigned char,unsigned char,unsigned char,unsigned
char,int);
void Clcd_draw_rectangle(unsigned char,unsigned char,unsigned
char,unsigned char,char,int);
void Clcd_write_pic(unsigned char,unsigned char,flash unsigned char *);

#ifdef COLOR_INTERFACE_8
flash unsigned char picture1[]=
{
132,132,
0xFF, 0xFF, 0xFF, 0xFF, xF7, 0xAD, 0xF2, 0xFF, 0xFF, 0xF7, 0xFF,
0xFF, 0xFF, 0xAE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF2,
0xAD, 0xFF, 0xFF, 0xF7, 0xC9, 0xF7, 0xD6, 0xF7, 0xFF, 0xFF, 0xFF,
0xFF, 0xF7, 0xA9, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xD2, 0xFF, 0xCE,
```

```
0xD2, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xAE, 0xF7, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xAE, 0xF7,
0xFF, 0xFF, 0xFF, 0xFF, 0xB1, 0xFF, 0xFF, 0xFF, 0xFF, 0xF7, 0xA5,
0xFF, 0xFF, 0xAE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xA9, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xF6, 0xA5, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xDB,
0xAE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF7,
0xA5, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC9,... };
#endif

void main (void)
  {
  unsigned int counter_1,counter_2;
  unsigned int color_8=0;
  CLCD_DDR=(1<<SDA)|(1<<CLK)|(1<<CS)|(1<<RST);
// Port Direction Setup

  // ********** STARTUP COLOR LCD **********
  CS_0;
  SDA_0;
  CLK_1;

  RST_1;
  RST_0;
  delay_ms(10);
  RST_1;

  CLK_1;
  SDA_1;
  CLK_1;
  delay_ms(10);

  // ********** INITIALIZATION COLOR LCD **********
  //Software RST
  _Clcd_write_command_(0x01);

  //Sleep Out
  _Clcd_write_command_(0x11);

  //Booster ON
  _Clcd_write_command_(0x03);
```

```
delay_ms(10);

//Display On
_Clcd_write_command_(0x29);

//Normal display mode
_Clcd_write_command_(0x13);

//Data order
_Clcd_write_command_(0xBA);

//Memory data access control
_Clcd_write_command_(0x36);
_Clcd_write_data_(0x00);     // no mirror Y and no mirror X

#ifdef COLOR_INTERFACE_16
_Clcd_write_command_(0x3A);
_Clcd_write_data_(5);     //16-Bit per Pixel
#endif
#ifdef COLOR_INTERFACE_12
_Clcd_write_command_(0x3A);
_Clcd_write_data_(3);     //12-Bit per Pixel (default)
#endif
#ifdef COLOR_INTERFACE_8
_Clcd_write_command_(0x3A);
_Clcd_write_data_(2);     //8-Bit per Pixel
#endif
Clcd_clear_screen();

// ********** PLACE YOUR CODE HERE **********
start:
Clcd_draw_rectangle(5,20,50,80,1,RED);
Clcd_write_pixel(5,5,GREEN);
Clcd_write_pixel(6,5,GREEN);
Clcd_write_pixel(5,6,GREEN);
Clcd_write_pixel(6,6,GREEN);
Clcd_draw_line(60,55,130,75,RED);
Clcd_draw_line(60,60,130,80,GREEN);
Clcd_draw_line(60,65,130,85,BLUE);
#ifdef COLOR_INTERFACE_8
Clcd_write_pic(0,0,picture1);
```

```
#endif
delay_ms(10000);
for(counter_2=0;counter_2<132;counter_2++)
  {
  for(counter_1=0;counter_1<132;counter_1++)
    {
    Clcd_write_pixel(counter_1,counter_2,color_8++);
    }
  }
delay_ms(10000);
for(color_8=65535;color_8;color_8--)
  {
  Clcd_draw_rectangle(0,0,131,131,1,color_8);
  delay_ms(500);
  }
Clcd_clear_screen();
goto start;
//*****************************************
}

void _Clcd_write_data_(unsigned char data)
  {
  CLK_0;
  SDA_1;     // send 1 for data
  CLK_1;

  CLK_0;
  if(data&0b10000000) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(data&0b01000000) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(data&0b00100000) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(data&0b00010000) SDA_1; else SDA_0;
  CLK_1;
```

```
  CLK_0;
  if(data&0b00001000) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(data&0b00000100) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(data&0b00000010) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(data&0b00000001) SDA_1; else SDA_0;
  CLK_1;
  }

void _Clcd_write_command_(unsigned char command)
  {
  CLK_0;
  SDA_0;    // send 0 for command
  CLK_1;

  CLK_0;
  if(command&0b10000000) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(command&0b01000000) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(command&0b00100000) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
  if(command&0b00010000) SDA_1; else SDA_0;
  CLK_1;

  CLK_0;
```

```
 if(command&0b00001000) SDA_1; else SDA_0;
 CLK_1;

 CLK_0;
 if(command&0b00000100) SDA_1; else SDA_0;
 CLK_1;

 CLK_0;
 if(command&0b00000010) SDA_1; else SDA_0;
 CLK_1;

 CLK_0;
 if(command&0b00000001) SDA_1; else SDA_0;
 CLK_1;
 }

void Clcd_set_contrast(char contrast)
 {
 _Clcd_write_command_(SET_CONTRAST);
 _Clcd_write_data_(contrast);
 }

void Clcd_clear_screen(void)
 {
 unsigned int counter;
 _Clcd_write_command_(ROW_ADDRESS_SET);
 _Clcd_write_data_(0);
 _Clcd_write_data_(131);
 _Clcd_write_command_(COLUMN_ADDRESS_SET);
 _Clcd_write_data_(0);
 _Clcd_write_data_(131);
 _Clcd_write_command_(RAM_WRITE);
 // it fills screen with white color
 #ifdef COLOR_INTERFACE_16
 for(counter=0;counter<17424;counter++)
  {
  _Clcd_write_data_(0xFF);
  _Clcd_write_data_(0xFF);
  }
 #endif
 #ifdef COLOR_INTERFACE_12
```

```
 for(counter=0;counter<8712;counter++)
   {
   _Clcd_write_data_(0xFF);
   _Clcd_write_data_(0xFF);
   _Clcd_write_data_(0xFF);
   }
 #endif
 #ifdef COLOR_INTERFACE_8
 for(counter=0;counter<17424;counter++)
   _Clcd_write_data_(0xFF);
 #endif
 }

void Clcd_write_pixel(unsigned char x,unsigned char y,int color)
 {
 _Clcd_write_command_(COLUMN_ADDRESS_SET);
 _Clcd_write_data_(x);
 _Clcd_write_data_(x);
 _Clcd_write_command_(ROW_ADDRESS_SET);
 _Clcd_write_data_(y);
 _Clcd_write_data_(y);
 _Clcd_write_command_(RAM_WRITE);
 #ifdef COLOR_INTERFACE_16
 // this is 16 bit color mode rrrrrgggggbbbbb
 // here we send 8 LSB bits (rrrrrggg) to lcd
 _Clcd_write_data_(color>>8);
 // and sending 8 MSB bits (gggbbbbb)
 _Clcd_write_data_(color);
 #endif
 #ifdef COLOR_INTERFACE_12
 // this is 12 bit color mode rrrrggggbbbb
 // it sets 1'st red and green colors
 _Clcd_write_data_(color>>4);
 // this line sets 1st pixel blue color and 2nd pixel red color
 _Clcd_write_data_((color<<4)|(color>>8));
 // and it sets 2'nd green and blue colors and also they must be 0
 _Clcd_write_data_(color);
 #endif
 #ifdef COLOR_INTERFACE_8
 // and last it's 8 bit color mode rrrgggbb
 // this line send pixel data color to lcd
```

```
 _Clcd_write_data_(color);
 #endif
 }
void Clcd_draw_line(unsigned char x1,unsigned char y1,unsigned char
x2,unsigned char y2,int color)
 {
 float m;
 unsigned int counter,temp;
 if(x1==x2)
  {
  if(y1>y2) {temp=y1;y1=y2;y2=temp;}
  for(counter=y1;counter<=y2;counter++)
{Clcd_write_pixel(x1,counter,color);}
  return;
  }
 if(x1>x2) {temp=x1;x1=x2;x2=temp;temp=y1;y1=y2;y2=temp;}
 m=(float)(y2-y1)/(x2-x1);
 if((m>0 && m<=1) || (m<0 && m>=-1) || m==0)
  {
  if(y1==y2)                    {for(counter=x1;counter<=x2;counter++)
{Clcd_write_pixel(counter,y1,color);}}
  for(counter=x1;counter<=x2;counter++)
{Clcd_write_pixel(counter,m*(counter-x1)+y1,color);}
  }
 else
  {
  if(y1>y2) {temp=x1;x1=x2;x2=temp;temp=y1;y1=y2;y2=temp;}
  for(counter=y1;counter<=y2;counter++)
{Clcd_write_pixel((float)(counter-y1+(m*x1))/m,counter,color);}
  }
 }

void Clcd_draw_rectangle(unsigned char x1,unsigned char y1,unsigned
char x2,unsigned char y2,char fill,int color)
 {
 unsigned int counter;
 unsigned char temp;
 #ifdef COLOR_INTERFACE_12
 unsigned char part_1,part_2,part_3;
 #endif
 if(x1>x2) {temp=x1;x1=x2;x2=temp;}
```

```
  if(y1>y2) {temp=y1;y1=y2;y2=temp;}
  if(fill)
    {
    _Clcd_write_command_(COLUMN_ADDRESS_SET);
    _Clcd_write_data_(x1);
    _Clcd_write_data_(x2);
    _Clcd_write_command_(ROW_ADDRESS_SET);
    _Clcd_write_data_(y1);
    _Clcd_write_data_(y2);
    _Clcd_write_command_(RAM_WRITE);
    #ifdef COLOR_INTERFACE_16
    for(counter=((int)((int)(x2-x1)+1)*((int)(y2-y1)+1));counter;counter--)
      {
      _Clcd_write_data_(color>>8);
      _Clcd_write_data_(color);
      }
    #endif
    #ifdef COLOR_INTERFACE_12
    part_1=(char)(color>>4);
    part_2=(char)(color<<4)|(color>>8);
    part_3=(char)color;
    for(counter=(((int)((int)(x2-x1)+1)*((int)(y2-
y1)+1))/2)+1;counter;counter--)
      {
      _Clcd_write_data_(part_1);
      _Clcd_write_data_(part_2);
      _Clcd_write_data_(part_3);
      }
    #endif
    #ifdef COLOR_INTERFACE_8
    for(counter=(((int)(x2-x1)+1)*((int)(y2-y1)+1));counter;counter--)
      _Clcd_write_data_(color);
    #endif
    }
  else
    {
    Clcd_draw_line(x1,y1,x2,y1,color);
    Clcd_draw_line(x2,y1,x2,y2,color);
    Clcd_draw_line(x2,y2,x1,y2,color);
    Clcd_draw_line(x1,y2,x1,y1,color);
    }
```

```
 }

void Clcd_write_pic(unsigned char x,unsigned char y,flash unsigned char
*pointer)
 {
 unsigned int counter;
 _Clcd_write_command_(COLUMN_ADDRESS_SET);
 _Clcd_write_data_(x);
 _Clcd_write_data_(x+((pointer[0])-1));
 _Clcd_write_command_(ROW_ADDRESS_SET);
 _Clcd_write_data_(y);
 _Clcd_write_data_(y+((pointer[1])-1));
 _Clcd_write_command_(RAM_WRITE);
 #ifdef COLOR_INTERFACE_16
 for(counter=(int)(pointer[0])*(pointer[1]),pointer+=2;counter;counter--)
  {
  _Clcd_write_data_(*pointer++);
  _Clcd_write_data_(*pointer++);
  }
 #endif
 #ifdef COLOR_INTERFACE_12

for(counter=((int)(pointer[0])*(pointer[1]))/2,pointer+=2;counter;counter-
-)
  {
  _Clcd_write_data_((*pointer)>>4);
  _Clcd_write_data_(((*pointer++)<<4)|((*pointer)>>8));
  _Clcd_write_data_(*pointer++);
  }
 #endif
 #ifdef COLOR_INTERFACE_8
 for(counter=(int)(*pointer++)*(*pointer++);counter;counter--)
  _Clcd_write_data_(*pointer++);
 #endif
 }
```

✍ ب- برنامه‌ای بنویسید که میکروکنترلر AVR عکس رنگی ذخیره شده بر روی MMC را خوانده و بر روی TFT موبایل (صفحه نمایش) متصل به آن نمایش دهد؟

✓ برنامه

```
#include <mega64.h>
#include <delay.h>
#include <math.h>
#include <stdlib.h>
#include <spi.h>
#include <string.h>
#include <stdio.h>
/*********************************************/
#define CS              PORTB.0
#define MMC             PINB.4
#define EN              0;
#define UN              1;
#define LCDCS           PORTD.2
#define LCDCLK                  PORTD.0
#define LCDSDA                  PORTD.1
#define LCDRESET        PORTD.3
#define KByte     1024      //Define as 1 Kilo Byte
#define MByte     1048576 //Define as 1 Mega Byte
#define byte unsigned char
/*********************************************/
byte s1,s2;
flash unsigned char image[678] = {0x00, 0x00, 0x03, 0x06, 0x02, 0x03, 0x06,
0x02, 0x03, 0x02, 0x02, 0x03, 0x08, 0x01, 0x03, 0x06, 0x01, 0x03, 0x05, 0x00,
0x03, 0x06, 0x01, 0x03, 0x04, 0x01, 0x03, 0x81, 0x81, 0x03, 0x00, 0x02, 0x03,
0x20, 0x00, 0x03, 0xA0, 0x00, 0x03, 0xA0, 0x00, 0x03, 0x20, 0x00, 0x03, …};
unsigned char LastX, LastY, TFName[9];
unsigned char FName[9],FExt[4],_Name[9],_Ext[4],FOName[9],buffer[512];
unsigned int n;
unsigned long FSize,FFat,FatArea1,FatArea2,RootArea,DataArea,ClusterSize;
unsigned long FileAddress;
long TST=0,TCO=0,TpX=0,TnX=0,TpY=0,TnY=0,TAX=0,TAY=0;
float Zoom=1;
/*************************************************************/
void MMC_CMD(unsigned char Command,unsigned long Arg);
unsigned char MMC_Response(unsigned char res);
unsigned char MMC_Read(unsigned long Add, unsigned char *ReadTemp);
unsigned char MMC_Write(unsigned long Add, unsigned char *WiteTemp);
unsigned char MMC_Init(void);
unsigned long File_Open(void);
```

```
unsigned long File_Parser(unsigned long add);
void header(void);
void parser(unsigned char BYTE1,unsigned char BYTE2,unsigned char BYTE3);
void sendData(byte data);
void sendCMD(byte data);
void setPixel(byte r,byte g,byte b);
void shiftBits(byte b);
void lcd_init(void);
void lcd_area(byte x1, byte x2, byte y1, byte y2);
void lcd_gotoxy(byte x, byte y);
/*****************************************************************/
void header(void)
{
unsigned char n,TFST[8],TFCO[4],TFPX[6],TFNX[6],TFPY[6];
  unsigned char TFNY[6],TFAX[7],TFAY[7];
        for(n=0;n<8;n++)
        TFName[n]=buffer[n+3];  //Tajima File Name
        TFName[8]='\n';

        for(n=0;n<7;n++)
        TFST[n]=buffer[n+23];    //Tajima ST
        TFST[7]='\n';

        for(n=0;n<3;n++)
        TFCO[n]=buffer[n+34];    //Tajima total used Color
        TFCO[3]='\n';

        for(n=0;n<5;n++)
        TFPX[n]=buffer[n+41];    //Tajima +X
        TFPX[5]='\n';

        for(n=0;n<5;n++)
        TFNX[n]=buffer[n+50];    //Tajima -X
        TFNX[5]='\n';

        for(n=0;n<5;n++)
        TFPY[n]=buffer[n+59];    //Tajima +Y
        TFPY[5]='\n';

        for(n=0;n<5;n++)
        TFNY[n]=buffer[n+68];    //Tajima -Y
        TFNY[5]='\n';

        for(n=0;n<6;n++)
        TFAX[n]=buffer[n+77];    //Tajima AX
        TFAX[6]='\n';
```

```
        for(n=0;n<6;n++)
        TFAY[n]=buffer[n+87];    //Tajima AY
        TFAY[6]='\n';

        TST=atol(TFST);//Tajima ST
        TCO=atol(TFCO); //Tajima total used Color
        TpX=atol(TFPX); //Tajima +X
        TnX=atol(TFNX); //Tajima -X
        TpY=atol(TFPY); //Tajima +Y
        TnY=atol(TFNY); //Tajima -Y
        TAX=atol(TFAX);  //Tajima AX
        TAY=atol(TFAY);  //Tajima AY
}
/*************************************************************/
void optimize(void)
{
        LastX=50;
        LastY=50;
   Zoom=.5;
}
/*************************************************************/
//send data
void sendData(byte data)
{
  LCDCLK=0;
  LCDSDA=1;//1 for param
  LCDCLK=1;
  shiftBits(data);
}
/*************************************************************/
//send cmd
void sendCMD(byte data)
{
  LCDCLK=0;
  LCDSDA=0;//1 for cmd
  LCDCLK=1;
  shiftBits(data);
}
/*************************************************************/
//converts a 3*8Bit-RGB-Pixel to the 2-Byte-RGBRGB Format of the Display
void setPixel(byte r,byte g,byte b)
 {
#ifdef MODE565
   sendData((r&248)|g>>5);
   sendData((g&7)<<5|b>>3);
```

```
#else
  if (n==0)
  {
    s1=(r & 240) | (g>>4);
    s2=(b & 240);
    n=1;
  }
else
{
    n=0;
    sendData(s1);
    sendData(s2|(r>>4));
    sendData((g&240) | (b>>4));
  }
#endif
}
/*****************************************************************/
void shiftBits(byte b)
{
  LCDCLK=0;
  if ((b&128)!=0) LCDSDA=1; else LCDSDA=0;
  LCDCLK=1;

  LCDCLK=0;
  if ((b&64)!=0) LCDSDA=1; else LCDSDA=0;
  LCDCLK=1;

  LCDCLK=0;
  if ((b&32)!=0) LCDSDA=1; else LCDSDA=0;
  LCDCLK=1;

  LCDCLK=0;
  if ((b&16)!=0) LCDSDA=1; else LCDSDA=0;
  LCDCLK=1;

  LCDCLK=0;
  if ((b&8)!=0) LCDSDA=1; else LCDSDA=0;
  LCDCLK=1;

  LCDCLK=0;
  if ((b&4)!=0) LCDSDA=1; else LCDSDA=0;
  LCDCLK=1;

  LCDCLK=0;
  if ((b&2)!=0) LCDSDA=1; else LCDSDA=0;
  LCDCLK=1;
```

```
 LCDCLK=0;
 if ((b&1)!=0) LCDSDA=1; else LCDSDA=0;
 LCDCLK=1;
}
/*************************************************************/
void lcd_init(void)
{
 LCDCS=0;
 LCDSDA=0;
 LCDCLK=1;

 LCDRESET=1;
 LCDRESET=0;
 LCDRESET=1;

 LCDCLK=1;
 LCDSDA=1;
 LCDCLK=1;

 delay_ms(10);
 sendCMD(0x01);
 sendCMD(0x11);
 sendCMD(0x03);

 delay_ms(10);

 sendCMD(0x29);
 sendCMD(0x13);
 sendCMD(0x21);
 sendCMD(0xBA);
 sendCMD(0x36);
 sendData(8|128);
}
/*************************************************************/
void lcd_area(byte x1, byte x2, byte y1, byte y2)
{
        //Column Adress Set
        sendCMD(0x2A);
        sendData(x1);
        sendData(x2);

        //Page Adress Set
        sendCMD(0x2B);
        sendData(y1);
        sendData(y2);
```

```
        //Memory Write
        sendCMD(0x2C);
}
/*************************************************************/
void lcd_gotoxy(byte x, byte y)
{
        if( (x>=0 && x<=131) && (y>=0 && y<=131) )
        {
        //Column Adress Set
        sendCMD(0x2A);
        sendData(x);
        sendData(x);

        //Page Adress Set
        sendCMD(0x2B);
        sendData(y);
        sendData(y);

        //Memory Write
        sendCMD(0x2C);
        }
}
/*************************************************************/
#define black     0
#define white     1
#define red       2
#define green     3
#define blue      4
#define yellow    5
flash unsigned char colorbox[6][3]={
  {0x00,0x00,0x00}, //Black
   {0xFF,0xFF,0xFF}, //White
   {0xFF,0x00,0x00}, //Red
   {0x00,0xFF,0x00}, //Green
   {0x00,0x00,0xFF}, //Blue
   {0xFF,0xFF,0x00}  //Yellow
};
/*************************************************************/
int sgn(long val)
{
        if(val>0)return +1;
        else if(val<0)return -1;
        else return 0;
}
/*************************************************************/
```

```
void line(int x1, int y1, int x2, int y2, int col)
 {
        long u,s,v,d1x,d1y,d2x,d2y,m,n;
        int  i;
    u   = x2-x1;
        v   = y2-y1;
        d1x = sgn(u);
        d1y = sgn(v);
        d2x = sgn(u);
        d2y = 0;
        m   = abs(u);
        n   = abs(v);
    if (m<=n)
        {
        d2x = 0;
        d2y = sgn(v);
        m   = abs(v);
        n   = abs(u);
        }
        s = (int)(m / 2);
        for (i=0;i<m;i++)
        {
        lcd_gotoxy(x1,y1);
           setPixel(colorbox[col][0],colorbox[col][1],colorbox[col][2]);

        s += n;
        if (s >= m)
        {
                s -= m;
                x1 += d1x;
                y1 += d1y;
        }
        else
        {
                x1 += d2x;
                y1 += d2y;
        }
        }
        lcd_gotoxy(x1,y1);
        setPixel(colorbox[col][0],colorbox[col][1],colorbox[col][2]);
        LastX=x2;
        LastY=y2;
}
/*****************************************************************/
```

```
void lineto(int x, int y, int col)
{

        line(LastX,LastY,x,y,col);
        LastX=x;
        LastY=y;
}
/*****************************************************************/
void linerto(int x, int y, int col)
{

        line(LastX,LastY,LastX+x,LastY+y,col);
}
/*****************************************************************/
void square(int x, int y, int w, int h, int col)
{
        line(x,y,x+w,y,col);
        lineto(x+w,y+h,col);
        lineto(x,y+h,col);
        lineto(x,y,col);

        LastX=x;
        LastY=y;
}
/*****************************************************************/
void parser(unsigned char BYTE1,unsigned char BYTE2,unsigned char BYTE3)
{
        unsigned char STAT=0;
        int SX=0,SY=0;

        if((BYTE3&0x03)==0x03)STAT=1; // Normal Stitch
        if((BYTE3&0x83)==0x83)STAT=2; // Jump Stitch
        if((BYTE3&0xC3)==0xC3)STAT=3; // Stop/Color
        if((BYTE3&0xF3)==0xF3)STAT=4; // End Design

        if((BYTE1&0x01)==0x01)SX+=1;
        if((BYTE1&0x02)==0x02)SX-=1;
        if((BYTE1&0x04)==0x04)SX+=9;
        if((BYTE1&0x08)==0x08)SX-=9;

        if((BYTE2&0x01)==0x01)SX+=3;
        if((BYTE2&0x02)==0x02)SX-=3;
        if((BYTE2&0x04)==0x04)SX+=27;
        if((BYTE2&0x08)==0x08)SX-=27;

        if((BYTE3&0x04)==0x04)SX+=81;
```

```
        if((BYTE3&0x08)==0x08)SX-=81;

        if((BYTE1&0x10)==0x10)SY-=9;
        if((BYTE1&0x20)==0x20)SY+=9;
        if((BYTE1&0x40)==0x40)SY-=1;
        if((BYTE1&0x80)==0x80)SY+=1;

        if((BYTE2&0x10)==0x10)SY-=27;
        if((BYTE2&0x20)==0x20)SY+=27;
        if((BYTE2&0x40)==0x40)SY-=3;
        if((BYTE2&0x80)==0x80)SY+=3;

        if((BYTE3&0x10)==0x10)SY-=81;
        if((BYTE3&0x20)==0x20)SY+=81;
        linerto(SX,SY,red);


}
// MMC Command in SPI Mode ////////////////////////////////////
void MMC_CMD(unsigned char Command,unsigned long int Arg)
{
        unsigned char CRC;

        if(Command==0) CRC=0x95;
        else CRC=0xFF;

        CS=EN;
    spi(0xFF);
    spi(Command|0x40);
    spi((unsigned char)(Arg>>24));
    spi((unsigned char)(Arg>>16));
    spi((unsigned char)(Arg>>8));
    spi((unsigned char)(Arg));
    spi(CRC);
    spi(0xFF);
}
/*****************************************************************/
unsigned char MMC_Response(unsigned char res)
{
        unsigned long count=0xFFFF        ;
        while( (spi(0xFF)!=res)&&(--count>0) );
        if(count<10) return 0; //Response timeout;
        else return 1; //Response before timeout;
}
/*****************************************************************/
// MMC Read a sector ////////////////////////////////////////
```

```
unsigned char MMC_Read(unsigned long Add, unsigned char *ReadTemp)
{
   unsigned long cnt;
   if(MMC==0)
   {
           MMC_CMD(17,Add);
           if(!MMC_Response(0xFE)) return 1;

           for(cnt=0;cnt<512;cnt++)
           ReadTemp[cnt]=spi(0xFF);

      spi(0xFF);
      spi(0xFF);
      CS=UN;

      return 0;
    }
    else return 1;
}
/*****************************************************************/
// MMC Write a sector ////////////////////////////////////////
unsigned char MMC_Write(unsigned long Add, unsigned char *WiteTemp)
{
   unsigned long cnt;
   if(MMC==0)
   {
     MMC_CMD(24,Add);
           if(!MMC_Response(0x00)) return 1; //Write Command Error
           spi(0xFE);
           for(cnt=0;cnt<512;cnt++)
           spi(WiteTemp[cnt]);
            spi(0xFF);
      spi(0xFF);
            if(!MMC_Response(0xFF)) return 2; //Timeout Error
           CS=UN;
            return 0;
    }
    else return 1;
}
/*****************************************************************/
unsigned char MMC_Init(void)
{
           unsigned char n;
           unsigned long timeout=0xFFFF;
           CS=UN;
           for(n=0;n<10;n++) spi(0xFF);
```

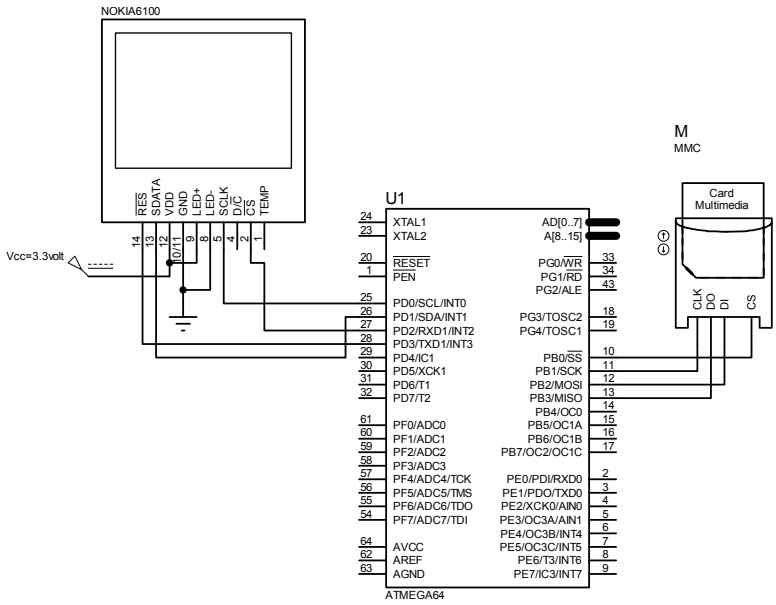
```
        CS=EN;

        MMC_CMD(0,0);
        if(!MMC_Response(0x01))
        {
                CS=UN;
                 return(1); //MMC not detect.
        }
                 while((spi(0xFF)!=0)&&(--timeout)) MMC_CMD(1,0);
        if(timeout<10)
        {
                CS=UN;
                 return 2; //Response timeout;
        }
        CS=UN;
        // SPI Clock Rate: 2*2000.000 kHz
        SPCR=0x50;
        SPSR=0x01;
        return 59; //Memory initialized.
        }
/*****************************************************************/
//MMC Info ////////////////////////////////////////////////
void MMC_Info(void)
{
        unsigned char BootSector[512],ABS[2],ARS[2],ASF[2],SectorCluster;
        unsigned long IBS,ISF,IRS;
 if(MMC==0)
   {
        MMC_Read(0x4000,BootSector);
        ABS[0]=BootSector[0x0C]; //Array[0] Byte per Sectror
        ABS[1]=BootSector[0x0B]; //Array[1] Byte per Sectror
        SectorCluster=BootSector[0x0D];
        ARS[0]=BootSector[0x0F]; //Array[0] Reserved Sectror
        ARS[1]=BootSector[0x0E]; //Array[1] Reserved Sectror
        ASF[0]=BootSector[0x17]; //Array[0] Sectror per Fat
        ASF[1]=BootSector[0x16]; //Array[1] Sectror per Fat
        IBS=(ABS[0]*0x100)+ABS[1]; //Integer Byte per Sectror
        IRS=(ARS[0]*0x100)+ARS[1]; //Integer Reserved Sectror
        ISF=(ASF[0]*0x100)+ASF[1]; //Integer Sectror per Fat
        FatArea1=0x4000+(IRS*512);
        FatArea2=0x4000+(ISF*512)+(IRS*512);
        RootArea=0x4000+((ISF*512)*2)+(IRS*512);
        DataArea=RootArea+0x4000;
        ClusterSize=(SectorCluster*IBS);
        }
   }
```

```
/*************************************************************/
//File Parser (Read a file details and parse) /////////////
unsigned long File_Parser(unsigned long add)
{
	unsigned char RootTemp[512],n,SizeAr[4],FatAr[2];
	unsigned long page,Pointer;
	page=(add*32)/512;
	Pointer=(add*32)%512;
	if (MMC_Read(RootArea+(page*512),RootTemp)==0)
	{
		for(n=0;n<8;n++)
		FName[n]=RootTemp[Pointer+n];
		FName[8]='\0';
		for(n=0;n<3;n++)
		FExt[n]=RootTemp[Pointer+n+8];
		FExt[3]='\0';
		for(n=0;n<4;n++)
		SizeAr[3-n]=RootTemp[Pointer+n+28];
		FSize=(SizeAr[0]);
		FSize=(FSize*0x100+SizeAr[1]);
		FSize=(FSize*0x100+SizeAr[2]);
		FSize=(FSize*0x100+SizeAr[3]);
		for(n=0;n<2;n++)
		FatAr[1-n]=RootTemp[Pointer+n+26];
		FFat=(FatAr[0]);
		FFat=(FFat*0x100+FatAr[1]);
		}
	if (FName[0]==0x00 && FName[1]==0x00) return 1;
	if (FName[0]==0xE5) return 2;
	return 0;
}
/*************************************************************
// File Open ////////////////////////////////////////////
unsigned long File_Open(void)
{
	unsigned char
TParser[3],col=0,Find=0,TBuffer[512],TempName[9],FatAr[3],SizeAr[5],mess[3
0];
unsigned long POINTER=0,n=0,FPage=0,F_Size,F_Fat,Page=0,
Position=0,Stack=0;
TempName[8]='\0';
	while(!Find && FPage<10)
	{	MMC_Read(RootArea+(FPage*512),TBuffer);
		while(!Find && POINTER<512)
		{	for(n=0;n<8;n++)
			TempName[n]=TBuffer[POINTER+n];
```

```
                        POINTER=POINTER+32;
                        if (strncmp(FOName,TempName,8)==0)Find=1;
                        }
                  FPage++;
        }
        if(Find==1)
        {
                FatAr[0] = TBuffer[POINTER-32+26];        // -- LL
                FatAr[1] = TBuffer[POINTER-32+27];        // HH --
                SizeAr[0] = TBuffer[POINTER-32+28];        // -- -- -- LL
                SizeAr[1] = TBuffer[POINTER-32+29];        // -- -- LH --
                SizeAr[2] = TBuffer[POINTER-32+30];       // -- HL -- --
                SizeAr[3] = TBuffer[POINTER-32+31];       // HH -- -- --
        F_Size=SizeAr[0]+SizeAr[1]*0x100+SizeAr[2]*0x10000+SizeAr[3]*0
x1000000;
                F_Fat=FatAr[0]+FatAr[1]*0x100;
                 Page=((F_Fat-2)*ClusterSize)/512;
                Position=((F_Fat-2)*ClusterSize)%512;
                Stack=F_Size;
                MMC_Read(DataArea+(Page*512),buffer);
    header();
                Position=512;
                Stack-=512;
                while( Stack>0 )
                {
                        if(Position<512)
                         {
                         TParser[col]=TBuffer[Position];
                         col++;
                         if(col>2)
                         {
                          parser(TParser[0],TParser[1],TParser[2]);
                         col=0;
                         }
                         Position++;
                         Stack--;
                }
                else
                         {
                         Position=0;
                         Page++;
                         MMC_Read(DataArea+(Page*512),TBuffer);
                         }
        }
```

```
  return (FPage*512+POINTER);
   }
}
/*************************************************************/
void main(void)
{long i=0;
 PORTA=0x00;DDRA=0xFF;
PORTB=0x00;DDRB=0x07;
PORTC=0x00;DDRC=0x00;
PORTD=0x00;DDRD=0x00;
PORTE=0x00;DDRE=0x00;
PORTF=0x00;DDRF=0x00;
PORTG=0x00;DDRG=0x00;
ASSR=0x00;
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;
EICRA=0x00;
EICRB=0x00;
```

```
EIMSK=0x00;
TIMSK=0x00;
ETIMSK=0x00;
SPCR=0x50;
SPSR=0x00;
ACSR=0x80;
SFIOR=0x00;
   delay_ms(10);
   lcd_init();
    delay_ms(10);
   while(1)
   {
   lcd_area(0,131,0,131);
   setPixel(255,255,255); //White Background
     lcd_gotoxy(5,5); sendData('h');
          square(10,10,50,50,red);
          delay_ms(1000);  square(50,50,50,50,blue);  delay_ms(1000);
    }
}
```



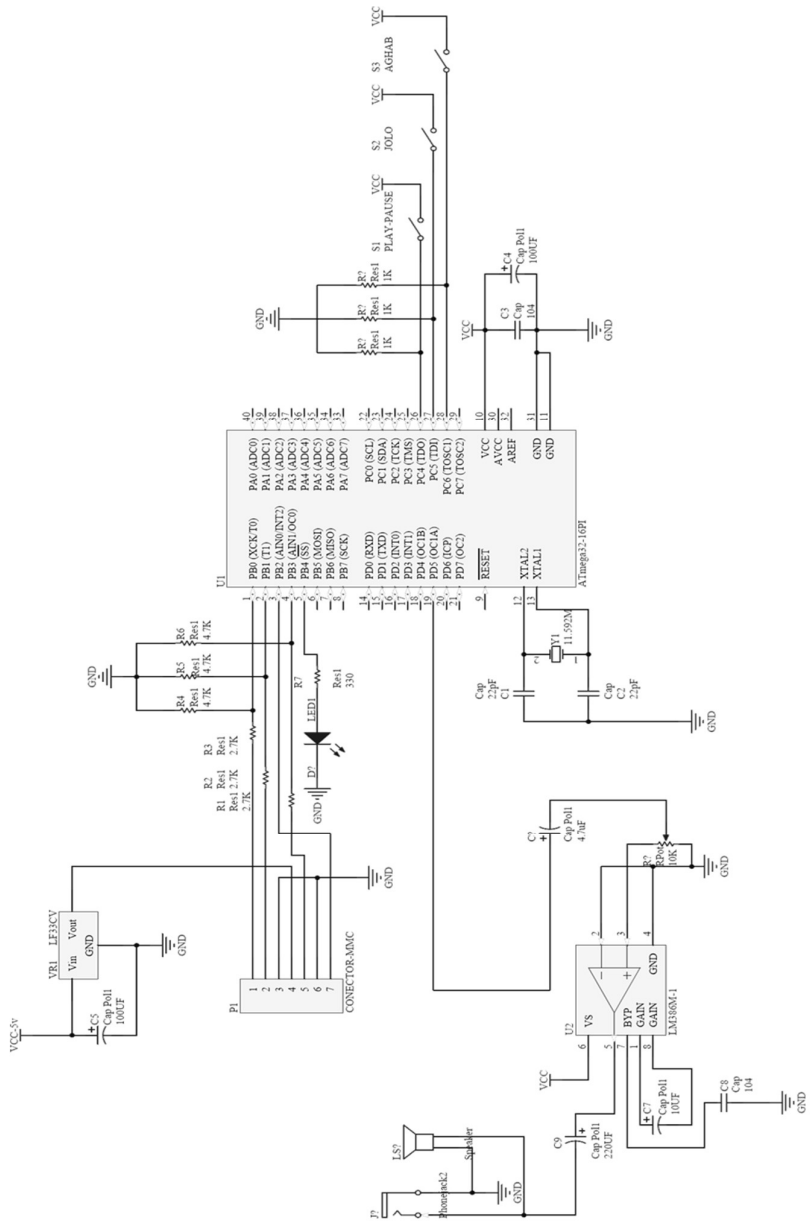
شکل(۱۰-۵): ارتباط میکرو با کارت حافظه MMC و TFT موبایل

تا این قسمت از کتاب، کلیه پروژه‌های بیان شده به زبان C و در محیط نرم‌افزار CodeVision انجام شدند. با توجه به وجود کتابخانه‌های بیشتر و کامل‌تر در نرم‌افزار Bascom، همچنین وجود برنامه‌های زیاد نوشته شده به زبان Basic، شبیه‌ساز و دیباگر بسیار قوی آن، آشنایی با زبان Basic و استفاده از نرم‌افزار Bascom راه را برای انجام پروژه‌های بسیار پیچیده هموار می‌سازد. با توجه به این مهم که زبان Basic، نسبت به زبان‌های C و اسمبلی به زبان ما نزدیک‌تر است، بنابراین: در هنگام ایجاد کدهای Hex، حجم کد بیشتری تولید می‌شود که در نتیجه، این زبان به صورت بهینه از حافظه محدود میکرو استفاده نمی‌کند. البته حجم زیاد کد برای برنامه‌های ارائه شده در این کتاب چندان مهم نیست. زیرا این کدها حجم زیادی از حافظه را اشغال نمی‌کنند. به منظور آشنایی مقدماتی با دستورات برنامه‌نویسی، ساختار برنامه‌نویسی، همچنین نحوه بررسی و اجرای برنامه‌ها در محیط دیباگر Bascom به CD همراه کتاب یا مراجع پایانی کتاب مراجعه نمایید. بار دیگر بر این مهم تاکید می‌نمائیم که اصرار پیاده‌سازی تمامی برنامه‌ها تنها به یک زبان و توسط یک نرم‌افزار کار معقولآن‌های نیست و یک مهندس می‌باید با استفاده از تمامی ابزارها به پیاده‌سازی و اجرای رهیافت خود به صورت بهینه بپردازد و از نرم‌افزارهای موجود بیشنه استفاده را به عمل آورد.

پروژه چهل‌ودوم: طراحی و ساخت Wav Player

✍ برنامه‌ای بنویسید که در آن میکروکنترلر، فایل‌های صوتی با فرمت wav را به ترتیب از mmc بخواند و توسط بلندگو متصل به مدار پخش نماید؟

☑ **حل:** ابتدا سخت‌افزار شکل (۱۰-٦) را مداربندی کرده، سپس برنامه‌ای که در ادامه آورده شده که به زبان Basic در محیط Bascom نوشته شده را بنویسید و پس از کامپایل نمودن برنامه، فایل ایجاد شده با پسوند Hex را بر روی میکروکنترلر توسط پروگرمر برنامه‌ریزی نمائید. سپس mmc را به کامپیوتر متصل نموده و فایل‌های صوتی با فرمت wav را بر روی آن کپی نمائید. در پایان با اتصال mmc به مدار میکروکنترلر، فایل‌های صوتی به ترتیب پخش می‌شوند. شایان ذکر است که تنها فایل‌های صوتی با پسوند wav قابلیت خواندن توسط mmc را دارند و سایر فرمت‌ها می‌بایست با استفاده از روشی که در ادامه بیان می‌شود، به فرمت wav تبدیل شوند.



شکل(۱۰-٦): مدار Wav Player

✓ برنامه

```
$regfile = "m16def.dat"
$crystal = 11059200
Config Pinc.4 = Input
Pause Alias Pinc.4
'Config Pinc.5 = Input
Kelid Alias Pinc.5
Config Pinc.6 = Input
Kelid2 Alias Pinc.6
Config Pinb.4 = Output
Config Pind.3 = Output
Set Portd.3
Config Debounce = 20
Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Up , Compare B
Pwm = Clear Down , Prescale = 1
Pwm1a = 255
Pwm1b = 255
Const Msbl = 0
Const Msbh = 1
Const Dly = 2
Const Bits8 = 8
Const Bits16 = 16
Const Bits32 = 32
Dim Dat As Byte
Dim Resp As Byte
Dim I As Word
Dim Addr As Long
Dim Adres As Long
Dim Tanzim As Byte
Dim Ali As Byte
Dim Test As Long
Dim Copy As Long
Dim Shomare As Long
Dim Hassan As Byte
'Aliases
Cs Alias Portb.0
Mosi Alias Portb.1
Clk Alias Portb.3
```

```
Miso Alias Pinb.2
'Declarations
Declare Sub Minit
Declare Sub Mread(byval Addr As Long )
'Configs
Config Portb.2 = Output
Config Portb.0 = Output
Config Portb.3 = Output
Config Pinb.1 = Input
Waitms 300
Config Spi = Soft , Din = Pinb.2 , Dout = Portb.1 , Ss = Portb.0 , Clock = Portb.3
Enable Interrupts
Enable Spi
Spiinit
 'MAIN PROGRAM*************
Reset Portd.3
Main:
'Initialize the MMCC
Minit
Adres = 1048576
Ali = 0
 Reset Portd.3
Inja:
Toggle Portb.4
Mread Adres
Debounce Pause , 1 , Stop_play , Sub
Debounce Kelid2 , 1 , Jolo2 , Sub
Debounce Kelid , 1 , Jolo , Sub
Adres = Adres + 512
If Ali = 0 Then
Adres = Adres + 5120000
Test = 512 * 7800
' Test = 512 * 8000
Adres = Adres - Test
Ali = 1
Test = Adres
End If
'---------------------------------------------
Goto Inja
```

```
Xloop:
Set Cs
Shiftout Mosi , Clk , Dat , Msbl
Endloop:
Goto Endloop
End                                        'end program
' =======''SUB ROUTINES AND FUNCTIONS=======
' ***'INITIALIZATION OF MMC***
Sub Minit
Set Cs
Dat = &HFF
For I = 1 To 10
  Shiftout Mosi , Clk , Dat , Msbl
Next I
Resp = 255
Reset Cs
Cmd0:
Dat = &H40
Shiftout Mosi , Clk , Dat , Msbl
Addr = &H00000000
Shiftout Mosi , Clk , Addr , Msbl
Dat = &H95
Shiftout Mosi , Clk , Dat , Msbl

While Resp <> &H01
Shiftin Miso , Clk , Resp , Msbl
Wend

Set Cs
Waitms 50
Reset Cs
Dat = &HFF

Cmd1:
While Resp <> &H00
'warter are FAM & Behnam
Set Cs
Shiftout Mosi , Clk , Dat , Msbl
Shiftin Miso , Clk , Resp , Msbl
```

```
Reset Cs
Dat = &H41
Shiftout Mosi , Clk , Dat , Msbl
Addr = 0
Shiftout Mosi , Clk , Addr , Msbl
Dat = &HFF
Shiftout Mosi , Clk , Dat , Msbl
Shiftout Mosi , Clk , Dat , Msbl
Shiftin Miso , Clk , Resp , Msbl
Wend
Dat = &HFF
Set Cs
End Sub

' ****'READ routine assumes ADDR uses Status subroutine*****
Sub Mread(byval Addr As Long)
Set Cs
Dat = &HFF
Shiftout Mosi , Clk , Dat , Msbl
Shiftin Miso , Clk , Resp , Msbl
Reset Cs

Dat = &H51
Shiftout Mosi , Clk , Dat , Msbl
Shiftout Mosi , Clk , Addr , Msbl

Dat = &HFF
Shiftout Mosi , Clk , Dat , Msbl
Shiftin Miso , Clk , Resp , Msbl


While Resp <> 0
Shiftin Miso , Clk , Resp , Msbl
Wend

While Resp <> &HFE
 Shiftin Miso , Clk , Resp , Msbl
Wend

For I = 1 To 512
```

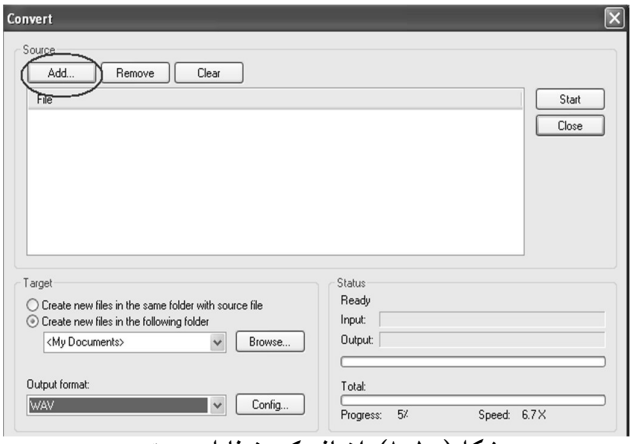
```
    Shiftin Miso , Clk , Resp , Msbl

      Pwm1a = Resp
      Pwm1b = Resp
      Waitus 29
'      Waitus 3
      Next I

Shiftin Miso , Clk , Resp , Msbl
Shiftin Miso , Clk , Resp , Msbl
Set Cs
'warter are FAM & Behnam
End Sub
```

☜ روش تبدیل فایل‌های صوتی به فرمت قابل پخش برای دستگاه WavPlayer :

برای تبدیل فایل های صوتی مختلف به فرمت Wav برنامه‌های متفاوتی وجود دارد که یکی از آن‌ها نرم افزار Jet Audio است. پس از اجرای برنامه Jet Audio روی قسمت Conversion کلیک نمائید:

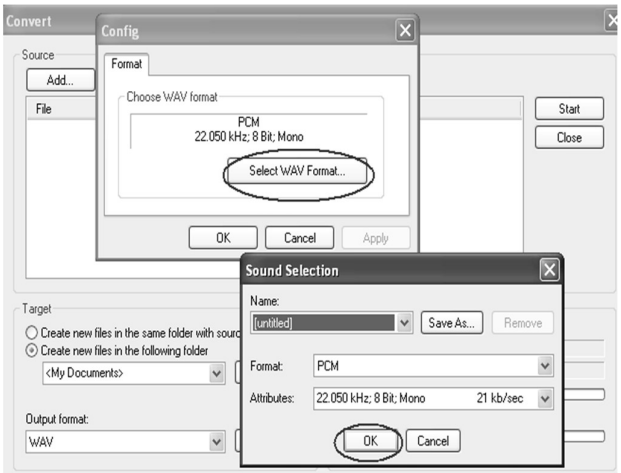
شکل(۱۰-۷): نرم‌افزار JetAudio

در پنجره باز شده در شکل (۱۰-۸)، روی دکمه Add کلیک کرده و فایل صوتی مورد نظر را انتخاب نمائید.



شکل(۱۰-۸): اضافه کردن فایل صوتی

در قسمت پایین یعنی Output format گزینه WAV را انتخاب کرده و کلید روبروی آن یعنی Config را انتخاب کنید تا شکل (۱۰-۹) ظاهر شود.



شکل(۱۰-۸): پیکربندی تبدیل

در این پنجره روی عبارت Select WAV format کلیک کنید و در پنجره جدید نوع PCM و مشخصات 22.050 kHz ; 8Bit ; Mono را انتخاب و سپس بر روی OK کلیک کنید. پس از پایان کار روی کلید Start کلیک کرده تا فایل یا فایل‌های صوتی به فرمت مورد نظر تبدیل شوند. تمامی فایل‌های تبدیل شده در پوشه My Documents ذخیره می‌شوند.

با توجه به اینکه این دستگاه برای خواندن اطلاعات از روی کارت MMC تمامی قوانین FAT16 و FAT32 و غیره را نادیده می‌گیرد، با پاک کردن و حتی فرمت کارت حافظه نیز تمامی فایل‌ها را پخش می‌کند مگر آن‌که فایل جدید را روی کارت کپی کنید. ترتیب پخش فایل‌ها بستگی به ترتیب کپی کردن آن‌ها روی حافظه دارد و هر فایلی که زودتر کپی شود در اول حافظه قرار می‌گیرد و در ابتدا پخش می گردد. هر بار که قصد دارید فایل جدیدی را در کارت ذخیره کنید حتماً قبل از آن کارت را Format کنید تا فایل‌های صوتی به ترتیب از اولین آدرس بعد از FAT قرار بگیرند و به ترتیب پخش شوند.

↻ پروژه چهل‌وسوم: ماشین حساب

✍ برنامه ماشین حساب با قابلیت‌های جمع، تفریق، ضرب و تقسیم را بنویسید؟

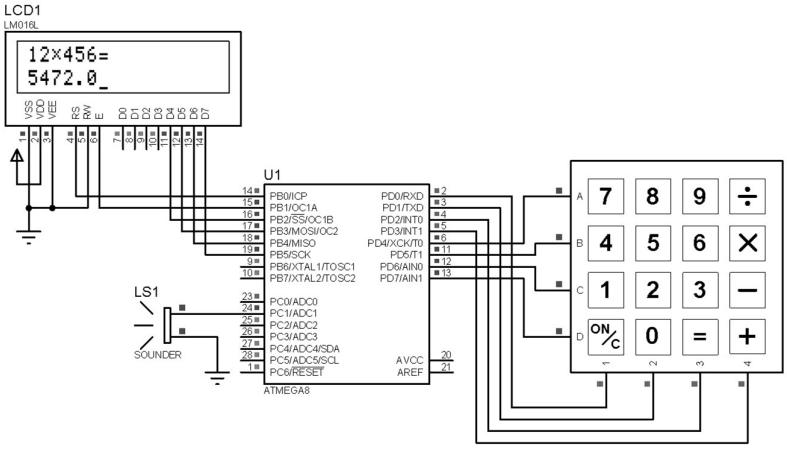
☑ حل: به منظور آشنایی بیشتر با زبان Basic، برنامه زیر به این زبان و در محیط Bascom نوشته شده است.

✓ برنامه

```
$regfile = "M8DEF.DAT"
$crystal = 1000000
Config Pinc.0 = Output
Config Pinc.1 = Output
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinb.2 , Db5 = Pinb.3 , Db6 = Pinb.4 , Db7 = Pinb.5
, Rs = Pinb.0 , E = Pinb.1
Config Kbd = Portd , Debounce = 100 , Delay = 10
Dim K As Byte                                   '
Dim A As Byte , D As Byte , B As Byte , C As Long , E As Byte , F As Long , G
As Long , H As Word , I As Single , J As Word
Deflcdchar 0 , 32 , 17 , 10 , 4 , 10 , 17 , 32 , 32       '*
Deflcdchar 1 , 32 , 4 , 32 , 31 , 32 , 4 , 32 , 32        '/
Deflcdchar 3 , 32 , 32 , 32 , 31 , 32 , 32 , 32 , 32      '-
Deflcdchar 4 , 32 , 4 , 4 , 31 , 4 , 4 , 32 , 32          '+
Deflcdchar 5 , 32 , 32 , 32 , 31 , 32 , 31 , 32 , 32      '=
Deflcdchar 6 , 32 , 17 , 10 , 4 , 10 , 17 , 32 , 32       'BINAHAYAT  VASAT
Deflcdchar 7 , 32 , 3 , 4 , 4 , 4 , 3 , 32 , 32           'BINAHAYAT  GHAP
Deflcdchar 2 , 32 , 24 , 4 , 4 , 4 , 24 , 32 , 32         'BINAHAYAT  RAST
Set Portc.0
Cls
Readeeprom K , 0
If K = 255 Then K = 0
```

```
Shoro:
Lcd "0" : B = 0 : C = 0 : E = 0

Main:
A = Getkbd()
If A = 16 Then Goto Main
Loop1:
D = Getkbd()
If D <> 16 Then Goto Loop1
B = Lookup(a , Keydata)
Select Case B
Case 0 To 9:
Cls
Sound Portc.1 , 100 , 25
I = 0 : J = 0 : C = C * 10 : C = C + B : Lcd C
If E = 0 Then F = C
If E = 1 Then G = C
Case 10:
Sound Portc.1 , 100 , 35
E = 1 : J = 0 : C = 0 : H = 0 : Locate 1 , 14 : Lcd Chr(1)
Goto Main
Case 11 :
Sound Portc.1 , 100 , 45
E = 1 : J = 0 : C = 0 : H = 1 : Locate 1 , 14 : Lcd Chr(0)
Goto Main
Case 12 :
Sound Portc.1 , 100 , 55
E = 1 : J = 0 : C = 0 : H = 2 : Locate 1 , 14 : Lcd Chr(3)
Goto Main
Case 13 :
Sound Portc.1 , 100 , 65 : Incr J
If J = 3 Then Goto Resetha
Cls : Locate 1 , 1 : Lcd "reset " : Locate 2 , 1
Lcd "calculator" : Waitms 600 : Cls
Goto Shoro
Case 14:
Sound Portc.1 , 100 , 150 : Waitms 30
Sound Portc.1 , 100 , 150 : Waitms 30
J = 0 : K = K + 1
Writeeeprom K , 0
Waitms 4
If H = 0 Then Goto Javab
If H = 1 Then Goto Javab1
If H = 2 Then Goto Javab2
If H = 3 Then Goto Javab3
Case 15 :
```

```
Sound Portc.1 , 100 , 75
J = 0 : E = 1 : C = 0 : H = 3
Locate 1 , 14 : Lcd Chr(4)
Goto Main
End Select
Goto Main
Javab:
I = F / G
If G = 0 Then Goto Error1
Cls : Lcd F ; Chr(1) ; G ; Chr(5)
Locate 2 , 1 : Lcd I : F = 0 : G = 0
Goto Main
Javab1:
I = F * G : Cls : Lcd F ; Chr(0) ; G ; Chr(5)
Locate 2 , 1 : Lcd I : F = 0 : G = 0
Goto Main
Javab2:
I = F - G : Cls : Lcd F ; Chr(3) ; G ; Chr(5)
Locate 2 , 1 : Lcd I : F = 0 : G = 0
Goto Main
Javab3:
I = F + G : Cls : Lcd F ; Chr(4) ; G ; Chr(5)
Locate 2 , 1 : Lcd I : F = 0 : G = 0
Goto Main

Error1:
Cls : F = 0 : G = 0 : Locate 1 , 1
Lcd "CANNOT DIVIDE BY " : Locate 2 , 1 : Lcd "ZERO !!!"

Sound Portc.1 , 100 , 50
 Waitms 50
Sound Portc.1 , 100 , 50
 Waitms 50
Sound Portc.1 , 100 , 50
 Waitms 50
Sound Portc.1 , 100 , 50
 Waitms 50
Sound Portc.1 , 100 , 50
Waitms 50
Sound Portc.1 , 100 , 50

Goto Main
End

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

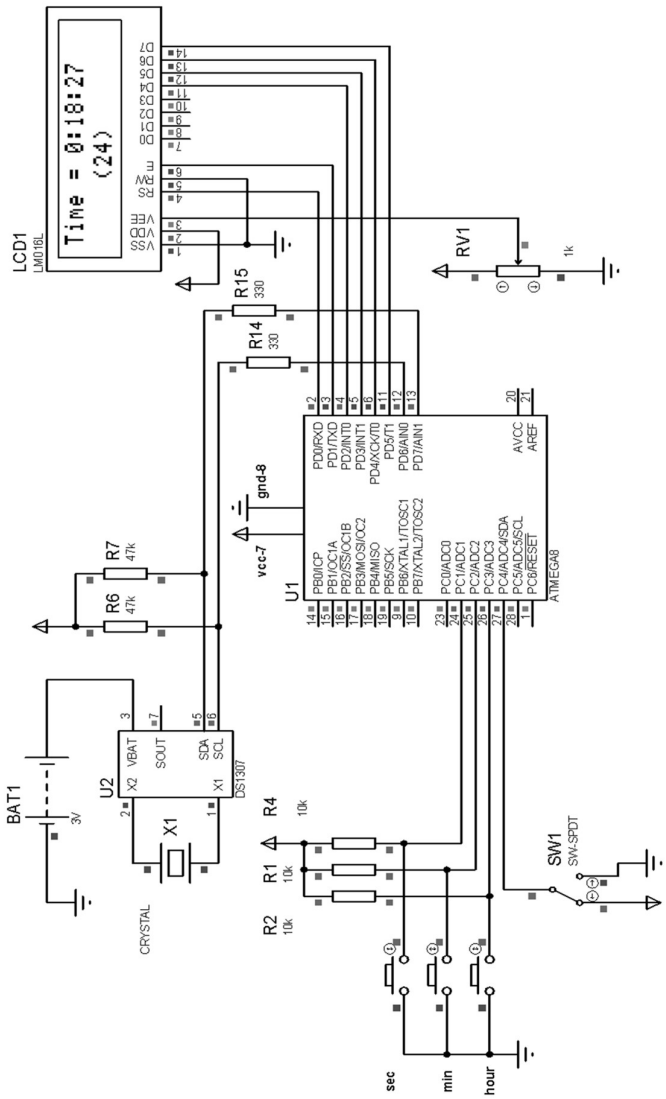


شکل(۱۰-۹): مدار ماشین حساب

پروژه چهل‌وچهارم: ساعت با استفاده از تراشه DS1307

الف- برنامه ساعتی را بنویسید که توسط آن: میکروکنترلر با اتصال به تراشه DS1307 قادر به نمایش زمان و تاریخ باشد. همچنین امکان تغییر تاریخ، زمان برای کاربر وجود داشته باشد؟

حل: در نگاه اول برای کسی که بر سیستم‌های میکروکنترلری تسلط دارد، ساخت ساعت دیجیتالی کار دشواری نیست. اما از آنجایی که می‌باید پردازنده سیستم تمام وقت کار کند و هیچ‌گاه تغذیه آن قطع نشود، استفاده از یک میکرو تنها برای تحقق چنین امری معقولانه نیست. برای حل این مشکل، برخی شرکت‌های تولیدکننده قطعات الکترونیکی اقدام به ساخت تراشه‌هایی نمودند که در بسته بندی‌های کوچک، امکانات کامل یک تقویم و ساعت واقعی را دارند. ازجمله این شرکت‌ها، شرکت Dallas Semiconductors می‌باشد که تراشه DS1307 را که یک تراشه کوچک ۸ پین می‌باشد، تولید نموده است. این تراشه از خانواده تراشه‌های ساعت زمان واقعی (RTC) می‌باشد و امکان تنظیم و خواندن ثانیه، دقیقه، ساعت، روز، ماه، سال و روز هفته را به کاربر می‌دهد. ارتباط این تراشه با پردازنده با استفاده از پروتکل سریال I^2C و فقط از طریق دو سیم انجام می‌شود. همچنین امکان استفاده از باتری پشتیبان (Backup Battery) نیز برای آن در نظر گرفته شده است تا در زمان قطع تغذیه، تراشه به کارخود ادامه دهد. برنامه این پروژه به زبان Basic و در محیط Bascom نوشته شده است.



شکل(۱۰-۱۰): مدار ساعت با استفاده از تراشه DS1307 و نمایش روی LCD

✓ برنامه

```
$regfile = "m8def.dat"
 $crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Rs = Pind.0 , E = Pind.1 , Db4 = Pind.2 , Db5 = Pind.3 ,
Db6 = Pind.4 , Db7 = Pind.5
$lib "ds1307clock.lib"
Config Sda = Portd.7 : Config Scl = Portd.6
Const Ds1307w = &HD0                    ' Addresses of Ds1307 clock
Const Ds1307r = &HD1
Config Pinc.1 = Input : Config Pinc.2 = Input : Config Pinc.3 = Input : Config
Pinc.4 = Input
Config Debounce = 30
Dim A As Byte , B As Byte , Data1 As Byte , C As Byte
Dim Seco As Byte , Mine As Byte , Hour As Byte
Cursor Off : Cls
Main:
Do
Gosub Ds1307
Gosub 24_12
Gosub Chekkey
Loop
Ds1307:
I2cstart                          ' Generate start code
I2cwbyte Ds1307w                          ' send address
I2cwbyte 0                        ' start address in 1307
I2cstart                            ' Generate start code
I2cwbyte Ds1307r                          ' send address
I2crbyte Seco , Ack                       'sec
I2crbyte Mine , Ack                       ' MINUTES
I2crbyte Hour , Nack                      ' Hours
I2cstop
Seco = Makedec(seco) : Mine = Makedec(mine) : Hour = Makedec(hour)
If Seco > 59 Then Seco = 0
If Mine > 59 Then Mine = 0
If Hour > 23 Then
Hour = 0
Gosub Seco
End If
Return
24_12:
If Pinc.4 = 1 Then Gosub Disply_24
If Pinc.4 = 0 Then Gosub Disply_12
Return
Disply_24:
Locate 1 , 1 : Lcd "Time = " ; Hour ; ":" ; Mine ; ":" ; Seco ; "      "
Locate 2 , 6 : Lcd "(24)"
```

```
Return
Disply_12:
If Hour = 0 Then Hour = 12
If Hour > 12 Then Hour = Hour - 12
Locate 1 , 1 : Lcd "Time = " ; Hour ; ":" ; Mine ; ":" ; Seco ; "      "
Locate 2 , 6 : Lcd "(12)"
Return
Chekkey:
Debounce Pinc.1 , 0 , Seco , Sub
Debounce Pinc.2 , 0 , Mine , Sub
Debounce Pinc.3 , 0 , Hour , Sub
Return
Seco:
Incr Seco
If Seco > 59 Then Seco = 0
Seco = Makebcd(seco)
I2cstart                        ' Generate start code
I2cwbyte Ds1307w                        ' send address
I2cwbyte 0                        ' starting address in 1307
I2cwbyte Seco
I2cstop
Return
Mine:
Incr Mine
If Mine > 59 Then Mine = 0
Mine = Makebcd(mine)
I2cstart                        ' Generate start code
I2cwbyte Ds1307w                        ' send address
I2cwbyte 1                        ' starting address in 1307
I2cwbyte Mine
I2cstop
Return
Hour:
Incr Hour
If Hour > 23 Then Hour = 0
Hour = Makebcd(hour)
I2cstart                        ' Generate start code
I2cwbyte Ds1307w                        ' send address
I2cwbyte 2                        ' starting address in 1307
I2cwbyte Hour
I2cstop
Return
End
```

ب- برنامه ساعتی را بنویسید که توسط آن: میکروکنترلر با اتصال به تراشه DS1307 قادر به نمایش زمان و تاریخ بر روی تابلو روان باشد؟

✓ برنامه

```
$regfile = "M32def.dat"
$crystal = 12000000

Dim Iloop As Word
Dim Row As Word
Dim Index As Word
Dim A As Word
Dim N As Word
Dim K As Word

Dim Ar1(512) As Byte
Dim Strg1 As String * 2
Dim Strg2 As String * 2
Dim Strg3 As String * 2
Dim Strg As String * 12

Dim Digits(12) As Byte At Strg Overlay

Ddra = &B11110000
Ddrb = 255
Ddrc = 255
Ddrd = 255

Config Scl = Porta.0
Config Sda = Porta.1

Const Ds1307w = &HD0
Const Ds1307r = &HD1

Dim _sec As Byte , _min As Byte , _hour As Byte
Dim _day As Byte , _month As Byte , _year As Byte
Dim Weekday As Byte , _min1 As Byte , _hour1 As Byte
Dim _weekday As Byte
'-----------------------------------------------------------------------------------------
Do
 Gosub Getdatetime
 Gosub Getdatetime
 Strg1 = Bcd(_sec)
 Strg2 = Bcd(_min)
 Strg3 = Bcd(_hour)
```

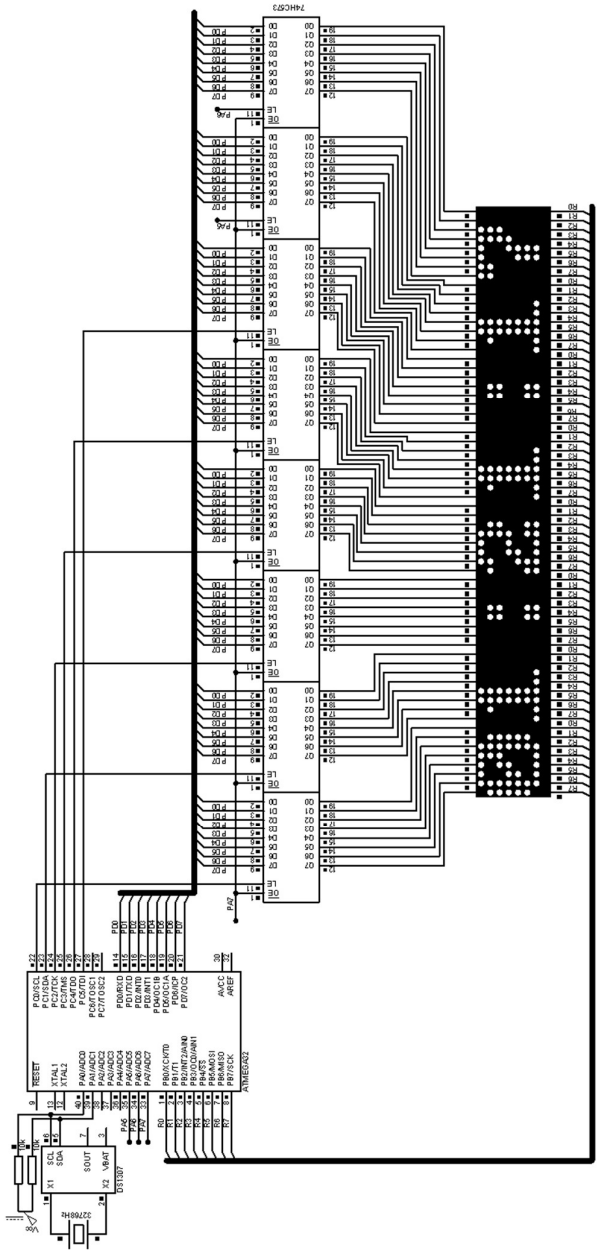
```
Strg = Strg3 + ":" + Strg2 + ":" + Strg1

A = 1
For Iloop = 1 To 8
  Gosub Filling
  For K = 1 To 8
     Read Ar1(a)
     Incr A
  Next
Next

For N = 1 To 100
    For Row = 0 To 7
        Portd = Ar1(1 + Row)
        Set Portc.0
        nop
        Reset Portc.0

        Portd = Ar1(9 + Row)
        Set Portc.1
        nop
        Reset Portc.1

        Portd = Ar1(17 + Row)
        Set Portc.2
        nop
        Reset Portc.2

        Portd = Ar1(25 + Row)
        Set Portc.3
        nop
        Reset Portc.3

        Portd = Ar1(33 + Row)
        Set Portc.4
        nop
        Reset Portc.4

        Portd = Ar1(41 + Row)
        Set Portc.5
        nop
        Reset Portc.5
        Portd = Ar1(49 + Row)
        Set Porta.5
        nop
        Reset Porta.5
```

```
        Portd = Ar1(57 + Row)
        Set Porta.6
        nop
        Reset Porta.6
      Reset Portb.row
      Waitms 1
      Set Portb.row
    Next
 Next
Loop
'-----------------------------------------------------------------------------------------
 Filling:
   Select Case Digits(iloop)
   Case 32 : Gosub Space_char
   Case 46 : Gosub Dot_char
   Case 47 : Gosub Forward_char
   Case 48 : Gosub 0_char
   Case 49 : Gosub 1_char
   Case 50 : Gosub 2_char
   Case 51 : Gosub 3_char
   Case 52 : Gosub 4_char
   Case 53 : Gosub 5_char
   Case 54 : Gosub 6_char
   Case 55 : Gosub 7_char
   Case 56 : Gosub 8_char
   Case 57 : Gosub 9_char
   Case 58 : Gosub Colon_char
   Case Else : Gosub Space_char
   End Select
 Return
'-----------------------------------------------------------------------------------------

Space_char:
 Restore Space_dta
 Return

Dot_char:
 Restore Point_dta
 Return

Forward_char:
 Restore Division_dta
 Return

0_char:
 Restore 0_dta
```

```
 Return

1_char:
 Restore 1_dta
 Return

2_char:
 Restore 2_dta
 Return

3_char:
 Restore 3_dta
 Return

4_char:
 Restore 4_dta
 Return

5_char:
 Restore 5_dta
 Return

6_char:
 Restore 6_dta
 Return

7_char:
 Restore 7_dta
 Return

8_char:
 Restore 8_dta
 Return

9_char:
 Restore 9_dta
 Return

Colon_char:
 Restore Colon_dta
 Return
'-------------------------------------------------------------------------------
Getdatetime:
I2cstart
I2cwbyte Ds1307w
I2cwbyte 0
```

```
I2cstart
I2cwbyte Ds1307r
I2crbyte _sec , Ack
I2crbyte _min , Ack
I2crbyte _hour , Ack
I2crbyte Weekday , Ack
I2crbyte _day , Ack
I2crbyte _month , Ack
I2crbyte _year , Nack
I2cstop
_min1 = Makedec(_min)
_hour1 = Makedec(_hour)
Return
Setdate:
_day = Makebcd(_day)
_month = Makebcd(_month)
_year = Makebcd(_year)
I2cstart
I2cwbyte Ds1307w
I2cwbyte 3
I2cwbyte _weekday
I2cwbyte _day
I2cwbyte _month
I2cwbyte _year
I2cstop
Return
Settime:
_sec = Makebcd(_sec)
_min = Makebcd(_min)
_hour = Makebcd(_hour)
I2cstart
I2cwbyte Ds1307w
I2cwbyte 0
I2cwbyte _sec
I2cwbyte _min
I2cwbyte _hour
I2cstop
Return
End
'-----------------------------------------------------------------------------------------
Space_dta:
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00

Point_dta:
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H18 , &H18 , &H00
```

```
Division_dta:
Data &H06 , &H0C , &H18 , &H30 , &H60 , &HC0 , &H80 , &H00

0_dta:
Data &H7C , &HCE , &HDE , &HF6 , &HE6 , &HC6 , &H7C , &H00

1_dta:
Data &H30 , &H70 , &H30 , &H30 , &H30 , &H30 , &HFC , &H00

2_dta:
Data &H78 , &HCC , &H0C , &H38 , &H60 , &HCC , &HFC , &H00

3_dta:
Data &H78 , &HCC , &H0C , &H38 , &H0C , &HCC , &H78 , &H00

4_dta:
Data &H1C , &H3C , &H6C , &HCC , &HFE , &H0C , &H1E , &H00

5_dta:
Data &HFC , &HC0 , &HF8 , &H0C , &H0C , &HCC , &H78 , &H00

6_dta:
Data &H38 , &H60 , &HC0 , &HF8 , &HCC , &HCC , &H78 , &H00

7_dta:
Data &HFC , &HCC , &H0C , &H18 , &H30 , &H30 , &H30 , &H00

8_dta:
Data &H78 , &HCC , &HCC , &H78 , &HCC , &HCC , &H78 , &H00

9_dta:
Data &H78 , &HCC , &HCC , &H7C , &H0C , &H18 , &H70 , &H00

Colon_dta:
Data &H00 , &H18 , &H18 , &H00 , &H00 , &H18 , &H18 , &H00
```



شکل(۱۰-۱۱): مدار ساعت با استفاده از تراشه DS1307 و نمایش توسط تابلو روان

تراشه DS1307 تاریخ را به صورت میلادی نشان می‌دهد و تاریخ شمسی مورد توجه می‌باشد. بنابراین می‌باید برنامه‌ای مطابق زیر نوشته شود که توسط آن تاریخ میلادی به شمسی تبدیل شود:

✓ برنامه

```
$regfile = "m16def.dat"
$crystal = 1000000
$lib "mcsbyteint.lbx"
Dim Myear As Integer
'input Miladi(gregorian) Year
Dim Mday As Integer
'input Miladi Day
Dim Mmonth As Integer
'input Miladi Month
Dim Mdayofyear As Integer
Dim Myear_ As Integer
Dim Jday As Integer                    'output Jalali(shamsi) Day
Dim Jmonth As Integer                   'output Jalali Month
Dim Jyear As Integer                   'output Jalali Year
Dim Jdayofyear As Integer
Dim Gnumdayofyear As Integer
Dim Isleapyear As Bit
Dim Isleapyearp As Bit
Dim Iloop As Byte
Dim Temp1 As Integer
Dim Temp2 As Integer
Dim Temp3 As Integer
Dim Temp4 As Integer
Dim Jmonth_a(12) As Byte
Dim Mmonth_a(12) As Byte
Mmonth_a(1) = 31
'———28 Ya 29 Agar Sall Kabiseh Bashad In Mah 29 Roz Mibashad————————
Mmonth_a(2) = 28
Mmonth_a(3) = 31
Mmonth_a(4) = 30
Mmonth_a(5) = 31
Mmonth_a(6) = 30
Mmonth_a(7) = 31
Mmonth_a(8) = 31
Mmonth_a(9) = 30
Mmonth_a(10) = 31
Mmonth_a(11) = 30
Mmonth_a(12) = 31
```

```
Gnumdayofyear = 365
Myear = 2008                                    'sample Miladi(gregorian) Year
Mday = 28                                       'sample Miladi Day
Mmonth = 8                                      'sample Miladi Month
'————————shart Kabiseh Bodane Sall———————— -
Temp1 = Myear Mod 4
Temp2 = Myear Mod 100
Temp3 = Myear Mod 400
Temp4 = Temp1 And Temp2
If Temp4 <> 0 Or Temp3 = 0 Then
Mmonth_a(2) = 28
Else
Mmonth_a(2) = 29
End If
Myear_ = Myear - 1
Temp1 = Myear_ Mod 4
Temp2 = Myear_ Mod 100
Temp3 = Myear_ Mod 400
Temp4 = Temp1 And Temp2
If Temp4 <> 0 Or Temp3 = 0 Then
Reset Isleapyearp
Else
Set Isleapyearp
End If
Temp1 = Mmonth - 1
For Iloop = 1 To Temp1
Mdayofyear = Mmonth_a(iloop) + Mdayofyear
Next
Mdayofyear = Mday + Mdayofyear
If Mdayofyear > 79 Then
Temp1 = Mdayofyear - 79
Jyear = Myear - 621
If Temp1 < 186 Then
Temp2 = Temp1 Mod 31
Temp3 = Temp1 / 31
If Temp2 = 0 Then
Jmonth = Temp3
Jday = 31
Else
Jmonth = Temp3 + 1
Jday = Temp2
End If
Else
Temp2 = Temp1 - 186
Temp3 = Temp2 Mod 30
Temp4 = Temp2 / 30
```

```
If Temp3 = 0 Then
Jmonth = Temp4 + 6
Jday = 30
Else
Jmonth = Temp4 + 7
Jday = Temp3
End If
End If
Else
Jyear = Myear - 622
If Isleapyearp = 1 Then
Mdayofyear = Mdayofyear + 11
Else
Mdayofyear = Mdayofyear + 10
End If
Temp2 = Mdayofyear Mod 30
Temp3 = Mdayofyear / 30
If Temp2 = 0 Then
Jmonth = Temp3 + 9
Jday = 30
Else
Jmonth = Temp3 + 10
Jday = Temp2
End If
Waitms 1
End If
```

پروژه چهل‌وپنجم: اجرای موزیک با میکرو

✍ برنامه‌ای بنویسید که میکروکنترلر با استفاده از داده‌های ذخیره شده در حافظه Flash خود به اجرای یکی از آهنگ‌های معروف اقدام نماید؟

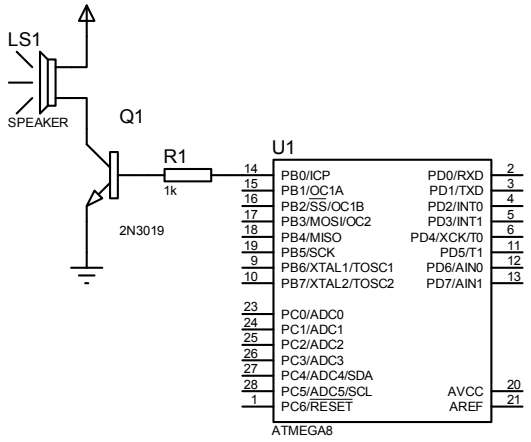
☑ حل: ابتدا برنامه زیر را در محیط برنامه‌نویسی Bascom نوشته و با اجرای آن در محیط شبیه‌ساز Proteus مطابق شکل (۱۰-۱۲) آن را اجرا نموده و به نواختن آهنگ گوش دهید

✓ برنامه

```
$regfile = "m8def.dat"
$crystal = 1000000
Config Pinb.0 = Output
Musicpin Alias Portb.0
Const La = 114
Const Lad = 107
Const Si = 101
Const Doo = 96
Const Dod = 90
```

```
Const Re = 85
Const Red = 80
Const Mi = 76
Const Fa = 72
Const Fad = 68
Const Sol = 64
Const Sold = 60
Const Mt = 1
'****************************
Dim I As Word
Dim Note As Byte
Dim Length As Byte
Dim Duration As Word
Do
For I = 0 To 149
 Note = Lookup(i , Notes)
 Note = Note * 2
 Length = Lookup(i , Lengths)
 Duration = 5000 / Note
 Duration = Duration * Length
 Sound Musicpin , Duration , Note
Next I
Waitms 500
Loop
End
'****************************
Notes:
Data La , Si , Doo , Mi , Fa , Mi
Data Fa , Mi , Fa , Mi , Fa , Mi , Re , Mi
Data Re , Mi , Re , Mi , Re , Mi
Data Re , Doo , Re , Doo , Re , Doo
Data Si , La , Sold , La , Si , Mt
Data Fa , Mi , Mt
Data La , Si , Doo , Mi , Fa , Mi
Data Fa , Mi , Fa , Mi , Fa , Mi , Re , Mi
Data Re , Mi , Re , Mi , Re , Mi
Data Re , Doo , Re , Doo , Re , Doo
Data Si , La , Sold , La , Si , Mt
Data Fa , Mi , Mt ,
Data La , Sol , Fa , Mi , Fa , Mi
Data Re , Fa , Mi , Re , Fa , Sol
Data Fa , Mi , Re , Mi , Re , Doo
Data Mi , Re , Doo , Mi , Fa , Mi
Data Re , Doo , Re , Doo , Si , Re
Data Doo , Si , Re , Re , Mi , Sol
Data Fa , Mi , Sold , La , Mt
```

```
Data La , Sol , Fa , Mi , Fa , Mi
Data Re , Fa , Mi , Re , Fa , Sol
Data Fa , Mi , Re , Mi , Re , Doo
Data Mi , Re , Doo , Mi , Fa , Mi
Data Re , Doo , Re , Doo , Si , Re
Data Doo , Si , Re , Mi , Mi , Doo
Data Si , La , Mt
'***************************
Lengths:
Data 4 , 4 , 4 , 8 , 4 , 8
Data 4 , 8 , 4 , 4 , 4 , 4 , 8 , 4
Data 8 , 4 , 8 , 4 , 4 , 4
Data 4 , 8 , 4 , 8 , 4 , 8
Data 4 , 4 , 4 , 4 , 6 , 4
Data 8 , 4 , 4
Data 4 , 4 , 4 , 8 , 4 , 8
Data 4 , 8 , 4 , 4 , 4 , 4 , 8 , 4
Data 8 , 4 , 8 , 4 , 4 , 4
Data 4 , 8 , 4 , 8 , 4 , 8
Data 4 , 4 , 4 , 4 , 6 , 4
Data 8 , 4 , 4
Data 4 , 4 , 4 , 4 , 6 , 2
Data 4 , 6 , 2 , 4 , 8 , 4
Data 4 , 4 , 4 , 6 , 2 , 4
Data 6 , 2 , 4 , 8 , 4 , 4
Data 4 , 4 , 6 , 2 , 4 , 6
Data 2 , 4 , 8 , 4 , 4 , 4
Data 4 , 12 , 12 , 8 , 4
Data 4 , 4 , 4 , 4 , 6 , 2
Data 4 , 6 , 2 , 4 , 8 , 4
Data 4 , 4 , 4 , 6 , 2 , 4
Data 6 , 2 , 4 , 8 , 4 , 4
Data 4 , 4 , 6 , 2 , 4 , 6
Data 2 , 4 , 8 , 4 , 4 , 4
Data 4 , 4 , 16
'***************************
```

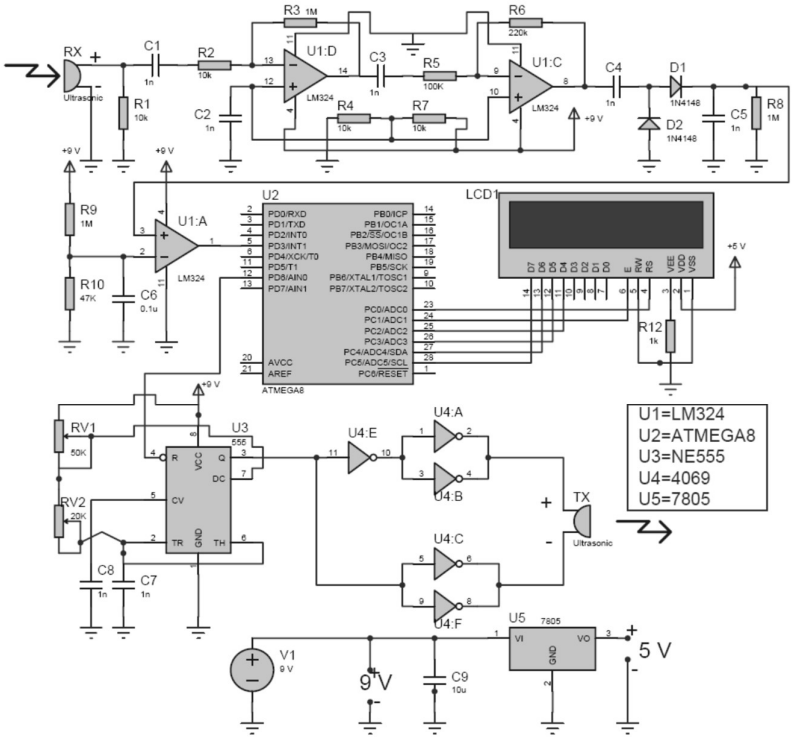


شکل(۱۰-۱۲): مدار پخش‌کننده آهنگ

پروژه چهل‌وششم: اندازه‌گیری فاصله توسط سنسورهای فراصوت (Ultrasonic)

✍ برنامه‌ای بنویسید که میکروکنترلر با استفاده از سنسورهای فراصوتی فاصله خود از مانع مقابلش را اندازه‌گیری و بر روی LCD نمایش دهد؟

☑ حل: یکی از روش‌های اندازه‌گیری فاصله یا اندازه‌گیری سطح مایع درون مخزن بدون تماس با آن، استفاده از روش آلتراسونیک می‌باشد. در این روش یک سیگنال مافوق صوت از طریق فرستنده آلتراسونیک به طرف سطح یا مانع مورد نظر ارسال شده و پس از برخورد با آن و انعکاس امواج، این سیگنال در گیرنده دریافت می شود. در این پروژه برای تحریک سنسور فرستنده از پالس‌های متقارن که توسط میکروکنترلر ATmega8 تولید می‌شوند، استفاده شده است. فرکانس این پالس‌ها برابر با فرکانس مرکزی سنسورها می‌باشد؛ که برای سنسورهای استفاده شده در این پروژه در حدود 40KHz است. همچنین از آنجایی که دامنه خروجی گیرنده به دلیل عوامل تضعیف کننده محیطی، در حدود میلی ولت است، نیاز به تقویت کننده چندطبقه و همچنین فیلترهایی برای حذف نویز و فرکانس‌های مزاحم محیط خواهیم داشت. در صورتی که مانعی در مقابل سنسورها به منظور انعکاس امواج داشته باشیم، خروجی طبقه گیرنده یک موج مربعی خواهد بود. این موج مربعی به پایه وقفه خارجی میکروکنترلر اعمال می‌شود که میکروکنترلر با محاسبه نصف زمان بین رفت و برگشت سیگنال و استفاده از سرعت صوت، فاصله فرستنده تا مانع را محاسبه کرده و بر روی LCD نمایش می‌دهد.



شکل(۱۰-۱۳): مدار فرستنده- گیرنده آلتراسونیک

✓ برنامه (به منظور توضیح بیشتر برنامه، خطوط برنامه شماره‌گذاری شده‌اند).

```
1_$regfile="m8def.dat"
2_$crystal=8000000
3_Config Lcdpin = Pin , Db4 = Pinc.2 , Db5 = Pinc.3 , -
4_db6 = Pinc.4 , Db7 = Pinc.5 , E = Pinc.1 , Rs = Pinc.0
5_Config Lcd = 16 * 2
6_Config Timer1 = Timer , Prescale = 8
7_Config Int1 = Rising
8_Enable Interrupts
9_Enable Int1
10_On Int0 Receive
11_Dim A As Single , B As Single
```

```
12_Config Pind.6 = Output
13_Cls
14_Cursor Off
15_Do
16_Start Timer1
17_Set Portd.6
18_Waitms 500
19_Reset Portd.6
20_Waitms 100
21_Loop
22_End

23_Receive :
24_Stop Timer1
25_Reset Portd.6
26_A = Timer1
27_B = A
28_If a > 8000 Then
29_Cls
30_Home
31_Lcd "OUT OF RANGE"
32_Goto Down
33_End If
34_A = A / 100
35_Cls
36_Home
37_Lcd A
38_Locate 2 , 1
39_Lcd B

40_Down :
41_Timer 1 = 0
42_A = 0
43_B = 0
44_Return
```

توضیح برنامه:

۱. معرفی تراشه مورد استفاده به کامپایلر جهت فراخوانی رجیسترهای مربوطه.
۲. معرفی فرکانس کاری اسیلاتور داخلی میکرو به کامپایلر.
۳. پیکربندی LCD مورد استفاده به منظور تعیین پایه‌های LCD متصل به میکرو که به پورت C وصل شده است.
٤. ادامه پیکربندی LCD
۵. پیکربندی نوع LCD به کار رفته که از نوع 2*16 است.

٦. پیکربندی تایمر یک، در مد تایمر که با فرکانس 1 MHz کار می‌کند.
٧. پیکربندی وقفه یک جهت دریافت پالس‌های رسیده از OPAMP که حساس به لبه بالا رونده است.
٨. فعال کردن کلیه پرچم وقفه سراسری
٩. فعال کردن وقفه یک
١٠. در صورت دریافت پالس به زیر برنامه Receive پرش کن
١١. تعریف کردن متغیرهای با نام A و B که از نوع Single هستند یعنی ٣٢ بیتی هستند
١٢. پیکربندی پین ششم از پورت D به عنوان خروجی داده جهت کنترل کردن تراشه 555
١٣. پاک کردن صفحه نمایش یا همان LCD
١٤. خاموش کردن مکان نما از روی LCD
١٥. تشکیل حلقه ای با تکرار بی نهایت توسط دستور DO – LOOP
١٦. شروع به کار تایمر یک
١٧. یک کردن پین ششم از پورت D میکرو جهت راه اندازی 555
١٨. به مدت 500 میلی ثانیه صبر کن یعنی 555 به مدت 500 میلی ثانیه روشن باشد.
١٩. صفر کردن پورت D پایه ٦ از میکرو جهت از کار افتادن 555
٢٠. به مدت 100 میلی ثانیه صبر کن.
٢١. تشکیل حلقه ای با تکرار بینهایت توسط دستور DO – LOOP
٢٢. پایان برنامه
٢٣. برچسبی با نام Receive که در وقفه خارجی رخ دهد، برنامه به این برچست پرش می‌کند.
٢٤. از کار افتادن تایمر یک
٢٥. صفر کردن پین ششم از پورت D میکرو جهت غیرفعال کردن 555
٢٦. مقدار تایمر یک را در متغیر A قرار بده
٢٧. مقدار متغیر A را برابر متغیر B قرار بده
٢٨. اگر A بزرگتر از 8000 بود سپس
٢٩. صفحه LCD را پاک کن
٣٠. مکان نما را به سطر و ستون اول ببر
٣١. برروی LCD متن OUT OF RANGE را نمایش بدهد
٣٢. به بر چسب DOWN پرش کن
٣٣. پایان دستور شرطی IF

۳٤. مقدار متغیر A را تقسیم بر 100 بکند نتیجه را در A ذخیره کن (جهت بد ست آوردن فاصله)
۳۵. صفحه نمایش را پاک بکن
۳٦. مکان نما را به سطر ستون اول ببر
۳۷. بر روی LCD مقدار متغیر A را نمایش بده.
۳۸. مکان نما را به سطر دوم و ستون اول ببر
۳۹. بر روی LCD مقدار متغییر B را نمایش بده
٤۰. برچسبی با نام DOWN
٤۱. مقدار Timer 1 را برابر صفر بکن
٤۲. مقدار متغیر A را برابر صفر بکن
٤۳. مقدار متغیر B را برابر صفر بکن
٤٤. بازگشت به حلقه DO – LOOP جهت اجرای دوباره برنامه.

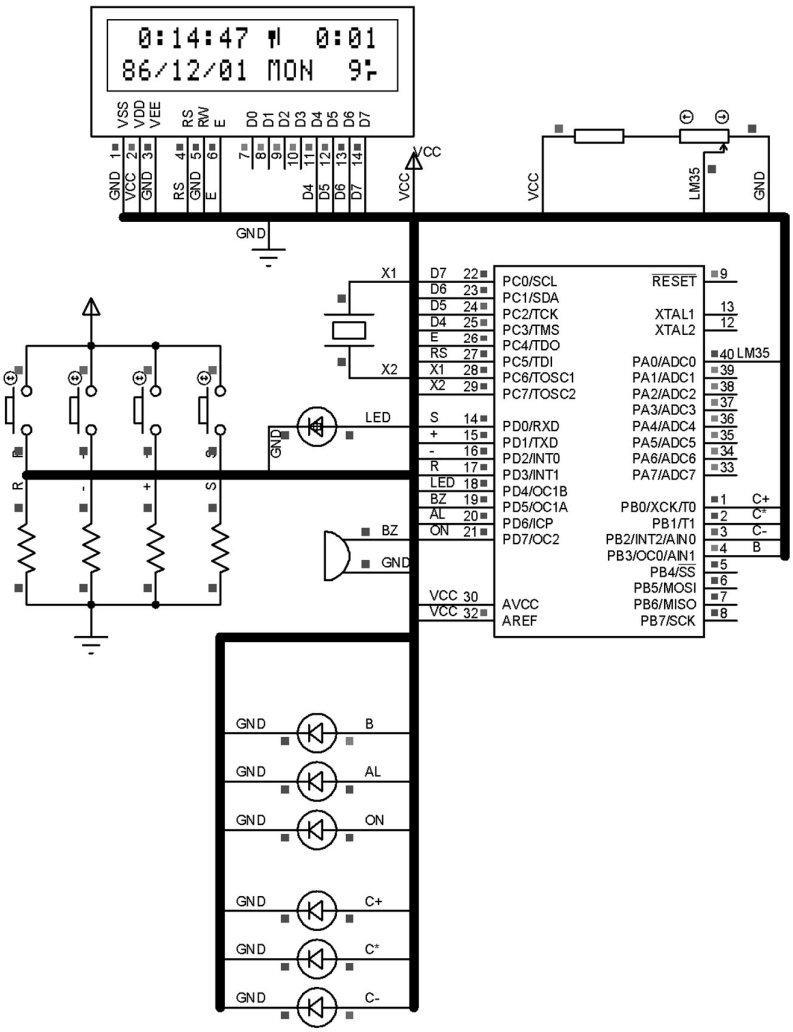
پروژه چهل‌وهفتم: مدار ترکیبی ترموستات وساعت

✍ برنامه یک ساعت با قابلیت‌های نمایش زمان، تاریخ، روز، زنگ اخبار و تنظیمات مربوط به یک ترموستات را بنویسید، سپس نتایج خود را در محیط نرم‌افزار شبیه‌ساز Proteus آزمایش کنید؟

☑ **حل:** برای برنامه‌نویسی و پیاده سازی این پروژه، مدار شکل (۱۰–۱٤) پیشنهاد می‌شود. این مدار نیازهای پروژه را تامین کرده و برنامه نوشته شده برای آن طوری طراحی شده است که دانستن نکات زیر به منظور تنظیمات آن الزامی است.

تنظیم تایمر: کلید SET را فشار دهید، مشاهده می‌شود که ساعت تایمر شروع به چشمک زدن می‌کند. با کلیدهای + و – می‌توانید زمان را تنظیم کنید. بعد از تنظیم ساعت، کلید SET را فشار داده و این بار دقیقه تایمر شروع به چشمک زدن می کند، بعد از تنظیم دقیقه تایمر با کلیدهای + و – ، کلید SET را برای برگشت به حالت عادی فشار دهید .

نکته: ساعت را به صورت ۲٤ ساعته تنظیم کنید.



شکل(۱۰-۱٤): مدار ترموستات و ساعت

- **تنظیم زمان:** کلید SET را برای ٦ ثانیه نگه دارید، می‌بینید که ساعت شروع به چشمک‌زدن می‌کند، بعد از تنظیم ساعت با فشار کلید SET می‌توانید دقیقه را تنظیم و با فشار مجدد کلید SET می‌توانید ثانیه و به همین صورت سال، ماه، روز و هفته را با کلیدهای + و - تنظیم کنید. با فشار مجدد کلید SET به حالت عادی بر می‌گردیم.

➢ نکته:

- ساعت را به صورت ٢٤ ساعت یعنی بین ٠ تا ٢٣ تنظیم کنید . (گزینه صبح و بعد از ظهر وجود ندارد).
- در هنگام تنظیم ثانیه اگر کلید + را فشار دهید و ثانیه فعلی بین ٣٠ تا ٥٩ ثانیه باشد یکی به دقیقه اضافه شده و بعد ثانیه صفر می شود . اگرکلید – را فشار دهیم و یا ثانیه فعلی بین ٠ و ٢٩ باشد و کلید + را فشار دهیم فقط ثانیه صفر میشود .
- خروجی PORTD.4 به یک LED برای نمایش ثانیه متصل می‌شود. LED با سرعت یک بار در ثانیه (٠/٥ ثانیه روشن و ٠/٥ ثانیه خاموش) چشمک می‌زند .
- در هنگام تنظیم روز اگر ماه بین ١ تا ٦ (٦ ماهه اول سال) با شد، می‌توانید روز را بین ١ تا ٣١ تنظیم کنید، در غیر این صورت یعنی اگر ماه بین ٧ تا ١٢ (٦ ماه دوم سال) با شد ، روز را می توانید بین ١ تا ٣٠ تنظیم کنید. در ضمن ا سفند نیز ٣٠ روزه در نظر گرفته شده است.
- در هنگام تنظیم روزهای هفته: SAT به معنی شنبه ، SUN بمعنی یک شنبه و به همین ترتیب FRI به معنی جمعه است.

تنظیم ترموستات: دو کلید + و– را فشار داده و برای ٦ ثانیه هر دو را نگه دارید، مشاهده می شود که میزان ترموستات دمای بالا شروع به چشمک زدن می کند . یعنی اگر دما از این حد بالاتر باشد خروجی PORTB.0 یک می شود، بعد از تنظیم دمای بالا، با فشار کلید SET می‌توانید دمای پایین را تنظیم کنید. در این صورت اگر دما از این میزان کمتر باشد خروجی PORTB.2 یک می شود، با فشار مجدد کلید SET به حالت عادی بر می گردیم .

➢ نکته :

- در کنار تنظیم دمای بالا علامت ↑ و برای تنظیم دمای پایین علامت ↓ نمایش داده می‌شود.
- دما ورودی هر یک ثانیه تست و نمایش داده می‌شود .
- اگر دمای فعلی بین دمای بالا و پایین باشد ، خروجی PORTB.1 یک می شود (اگر دمای بالا و پایین را معکوس تنظیم کرده با شید این خروجی یک نمی شود و تمام خروجی‌های ترموستات صفر می‌شوند).
- این مدار قادر به اندازه‌گیری دمای مثبت بین ٠ تا ٩٩ می‌باشد . اگر دما از ٩٩ درجه بالاتر باشد عبارت ERR نمایش داده می‌شود .
- دقت اندازه‌گیری یک درجه است.

- تست دما برای ترموستات هر ۳۰ ثانیه می‌باشد (به دلیل آن‌که خروجی‌ها در آستانه دمای ترموستات شروع به چشمک زدن نکند).

فعال کردن تایمر: با فشار کلید در هر جای برنامه که باشید می‌توانید تایمر را فعال یا غیرفعال کنید.

نکته:
- اگر مدار در حالت زنگ زدن و یا زمان SHORT SLEEP باشد، زنگ غیر فعال می‌شود و خروجی های مربوطه هم صفر می‌شوند.
- در هنگام فعال بودن تایمر خروجی PORTB.3 یک شده و در وسط خط اول LCD نیز علامت زنگ نمایش داده می‌شود.
- اگر مدار در حالت زنگ زدن با شد، خروجی PORTD.7 یک شده و تا زمانی که کلید را فشار ندهیم یک باقی می‌ماند. (به عنوان مثال برای اتصال به کتری برقی و یا رادیو).

تنظیم زمان زنگ زدن: اگر زمان ساعت با زمان تایمر برابر شود، مدار به ترتیب زیر شروع به زنگ زدن می‌کند.
ابتدا به مدت T1 ثانیه بیزر با صدای ملایم شروع به زنگ زدن می‌کند. (۰/۲۵ ثانیه بیزر روشن و ۰/۷۵ ثانیه خاموش) سپس به مدت T2 ثانیه مدار ساکت شده و هیچ صدایی ندارد و پس از آن به مدت T3 ثانیه بیزر با صدای معمولی شروع به زنگ زدن می‌کند، (۰/۵ ثانیه بیزر روشن و ۰/۵ ثانیه بیزر خاموش) بعد از طی زمان T3 مدار به مدت T4 ثانیه ساکت شده و هیچ صدایی ندارد و سپس به اندازه T5 ثانیه با صدای طولانی‌تر (۰/۷۵ ثانیه بیزر روشن و ۰/۲۵ ثانیه بیزر خاموش) بیزر شروع به زنگ زدن می‌کند ، پس از طی این مدت مدار ساکت شده و منتظر زمان بعدی تایمر می‌ماند. برای تنظیم زمان های T1 الی T5 به صورت زیر عمل می‌کنیم:
کلید را برای مدت ۷ ثانیه نگه داشته و مشاهده می شود که در مکان نمایش تایمر عبارت T1=01 نشان داده می شود بعد از تنظیم این زمان با کلیدهای + و - کلید SET فشار داده و زمان T2 و بهمین صورت زمان های T3 و T4 و T5 را تنظیم می‌کنیم و کلید SET را برای برگشت به حات عادی فشار می‌دهیم .

نکته:

- اگر در طی هر کدام از این زمان ها (T1 الی T5) کلید را فشـــار دهیم مدار ساکت می‌شود و زمآن‌ها غیر فعال می‌شوند و مدار دیگر هیچ صدایی ندارد.
- این زمآن‌ها بین ۰ تا ۱۵ دقیقه قابل تنظیم هستند.
- در زمان زمآن‌های T1 و T3 و T5 خروجی PORTD.6 یک می شود .
- گر هر زمانی را صفر در نظر بگیریم ، آن زمان اجرا نخواهد شد و اگر همه زمآن‌ها را صفر کنیم در لحظه تایمر فقط خروجی PORTD.7 یک خواهد شد.

خروجی های مدار:

Portd.4 برای نمایش ثانیه به یک LED متصل می‌شود.
Portd.5 خروجی تایمر جهت فعال‌سازی آلارم (خروجی زنگ مدار برای بیزر)
Portd.6 خروجی زنگ مدار به صورت ساده
Portd.7 خروجی تایمر که فقط با تایمر یک شده و صفر شدن آن به صورت دستی است . مثلا برای راه اندازی کتری برقی
Portb.0 برای دمای بالاتر از مقدار تعیین شده
Portb.1 برای دمای بین مقدار پایین و بالا از مقدار تعیین شده
Porb.2 برای دمای پایین‌تر از مقدار تعیین شده
Porb.3 برای نشان دادن فعال بودن تایمر

حجم برنامه بالا بوده و حدود 15 KB می‌باشد و نمی‌توان آن را در میکروهای کوچکتر از ATmega16 برنامه‌ریزی کرد.

✓برنامه

```
$regfile = "m16def.dat"
$crystal = 4000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.3 , Db5 = Pinc.2 , Db6 = Pinc.1 , Db7 = Pinc.0
, E = Pinc.4 , Rs = Pinc.5
Cursor Off
Config Portb = Output
Config Pind.7 = Output , Pind.6 = Output , Pind.5 = Output , Pind.4 = Output
Config Pind.3 = Input , Pind.2 = Input , Pind.2 = Input , Pind.0 = Input
Config Adc = Single , Prescaler = Auto

    Deflcdchar 0 , 24 , 24 , 32 , 32 , 15 , 8 , 8 , 15   ' .C
    Deflcdchar 1 , 1 , 29 , 29 , 29 , 9 , 9 , 8 , 8      'zang
    Deflcdchar 2 , 4 , 14 , 31 , 4 , 4 , 4 , 4 , 4       'UP
    Deflcdchar 3 , 4 , 4 , 4 , 4 , 4 , 31 , 14 , 4       'DOWN
```

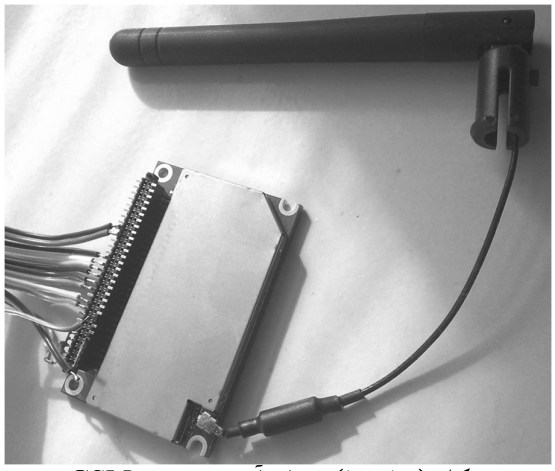
با توجه به حجم زیاد کدهای برنامه، برای مشاهده ادامه برنامه به CD همراه کتاب مراجعه نمائید و با اجرای برنامه، به بررسی موارد بیان شده در تنظیمات بپردازید.

پروژه چهل‌وهشتم: مودم GSM

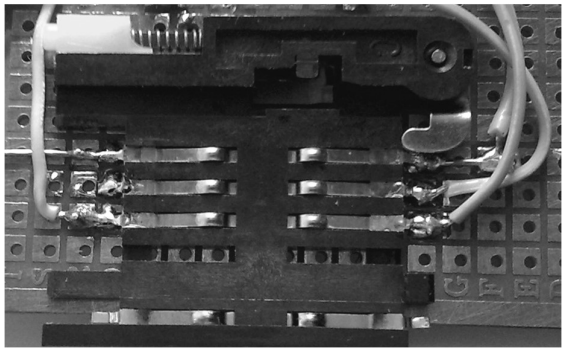
✍ برنامه‌ای بنویسید که توسط آن میکروکنترلر با اتصال به یک مودم GSM و سپس راه‌اندازی آن، پیام‌های کوتاه (SMS) آن را دریافت و پس از تحلیل آن‌ها عمل مطلوب را انجام دهد.

☑ حل: مودم‌های GSM یا به اصطلاح ماژول‌های موبایل در بازار با تنوع بسیار زیادی وجود دارند. بر روی این مودم‌ها یک عدد سیم‌کارت نصب می‌شود که با استفاده از آن، امکان ارسال و دریافت پیام کوتاه ممکن می‌شود. داده‌های دریافتی از سیم‌کارت از طریق پورت سریال مودم قابل دسترس است. همچنین داده‌های ارسالی توسط میکرو یا کامپیوتر از طریق همین پورت به مودم فرستاده می‌شوند تا مودم، آن‌ها را با استفاده از شناسه سیم کارت، ارسال نماید. مشابه هر تلفن همراه، مودم نیز دارای یک آنتن است که در شکل (۱۰-۱۵) آنتن آن همراه با سخت‌افزار مودم ترسیم شده‌اند. همچنین برای اتصال سیم کارت به مودم نیاز به استفاده از یک سوکت سیم کارت هستیم که این سوکت در شکل (۱۰-۱۶) نمایش داده شده است. نحوه اتصال سیم کارت به پایه‌های مودم بر اساس برگه‌های اطلاعاتی همان مودم تعیین می‌شود. زیر مودم‌ها از لحاظ عملکرد و شکل ظاهری و قابلیت‌ها، یکسان نمی‌باشند و بررسی آن‌ها تنها با استفاده از برگه‌های اطلاعاتی آن‌ها امکان‌پذیر است. در این پروژه برنامه‌ای به زبان Basic و در محیط Bascom نوشته شده است. هدف از این برنامه روشن نمودن و راه‌اندازی مودم است. همچنین این برنامه امکان دریافت پیام کوتاه را ممکن می‌سازد. طوری که مودم را قادر به دریافت پیام کوتاه می‌نماید و پس از آن، بر اساس پیام دریافت شده عمل مطلوب انجام می شود. سخت‌افزار کلی مدار در شکل (۱۰-۱۷) ترسیم شده است. همانطور که با مرور برنامه مشاهده می شود، در این برنامه دستوراتی با عبارت AT همراه شده‌اند که به این دستورات ATcommand می‌گویند. مودم GSM با استفاده از این دستورات پیکربندی و راه‌اندازی می‌شود و عمل ارسال و دریافت داده را انجام می‌دهد. با توجه به این‌که: برای ارسال و دریافت داده توسط مودم GSM از پورت سریال مودم استفاده شده است، بنابراین اطلاع از میزان نرخ بیت بر ثانیه الزامی است که در بیشتر مودم‌ها این نرخ برابر با ۹۶۰۰ بیت بر ثانیه است. دقت شود که به منظور راه‌اندازی مودم می‌باید شماره پین (Pin Code) سیم کارت توسط میکرو به

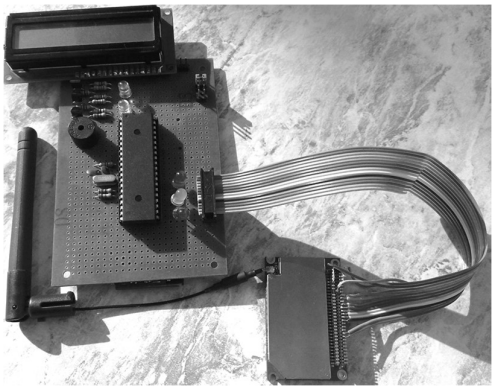
مودم ارسال شود. همچنین به منظور ارسال پیام کوتاه نیز، ارسال شماره مقصد توسط میکرو به مودم الزامی است. با مطالعه برنامه این پروژه، با دستورات راه‌اندازی و پیکربندی مودم، همچنین روش دریافت SMS آشنا شوید. شایان ذکر است که این برنامه آزمایش شده است و صحت عملکرد آن تایید می‌شود.



شکل (۱۰-۱۵): مدار آنتن و مودم GSM

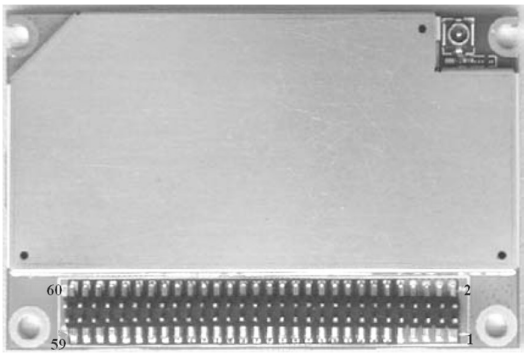


شکل (۱۰-۱٦): سوکت سیم کارت

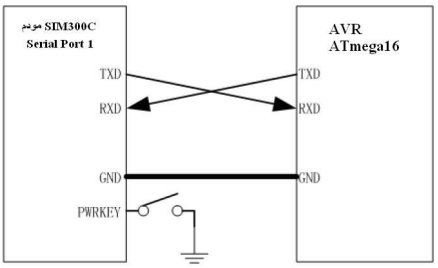


شکل (۱۰-۱۷): سخت‌افزار کلی مودم GSM و میکروکنترلر متصل به آن

مودم انتخاب شده برای این پروژه، مودم SIM300C است که برگه اطلاعاتی آن در CD همراه کتاب موجود است. در شکل (۱۰-۱۸) شماره پایه‌ها، در شکل (۱۰-۱۹) روش ارتباط سریال و در جدول (۱۰-۱) مشخصه هر پایه آورده شده است.



شکل (۱۰-۱۸): شماره پایه‌های مودم SIM300C



شکل (۱۰-۱۹): ارتباط پورت سریال میکرو و پورت سریال شماره (۱) مودم SIM300C

جدول (۱۰-۱): وضعیت و عملکرد پایه‌های مودم SIM300C

PIN NO.	PIN NAME	I/O	PIN NO.	PIN NAME	I/O
2	GND		1	VBAT	I
4	GND		3	VBAT	I
6	GND		5	VBAT	I
8	GND		7	VBAT	I
10	GND		9	VBAT	I
12	ADC1	I	11	CHG_IN	I
14	VRTC	I	13	TEMP_BAT	I
16	Network LED /GPIO12	O	15	VDD_EXT	O
18	KBC0	O	17	PWRKEY	I
20	KBC1	O	19	STATUS	O
22	KBC2	O	21	GPIO5	I/O
24	KBC3	O	23	BUZZER	O
26	KBC4	O	25	VSIM	O
28	KBR0	I	27	SIM_RST	O
30	KBR1	I	29	SIM_I/O	I/O
32	KBR2	I	31	SIM_CLK	O
34	KBR3	I	33	SIM_PRESENT	I
36	KBR4	I	35	GPIO32	I/O
38	SPI_EN	O	37	DCD	O
40	SPI_CLK	O	39	DTR	I
42	SPI_DO	I/O	41	RXD	I
44	SPI_AO	O	43	TXD	O
46	SPI_RESET	O	45	RTS	I
48	DBGRX	I	47	CTS	O
50	DBGTX	O	49	RI	O
52	AGND		51	AGND	
54	MIC1P	I	53	SPK1P	O
56	MIC1N	I	55	SPK1N	O
58	MIC2P	I	57	SPK2P	O
60	MIC2N	I	59	SPK2N	O

✓ برنامه

```
$regfile = "m16def.dat"
$crystal = 4000000
$baud = 9600
Config Portc = Output
```

```
Config Portb.0 = Output
Portb.0 = 0
Wait 4
Config Lcdpin = Pin , Db4 = Pina.4 , Db5 = Pina.5 , Db6 = Pina.6 , Db7 = Pina.7
, E = Pina.2 , Rs = Pina.1
Config Lcd = 16 * 2
Portb.0 = 1 : Portc = 0 : Wait 1
Portc = 1 : Wait 1
Portc = 2 : Wait 1
Portc = 3 : Wait 1
Portc = 4 : Wait 1
Portc = 5 : Wait 1
Portc = 6 : Wait 1
Portc = 7 : Wait 1
Declare Sub Getline(s As String)
Declare Sub Flushbuf()
Declare Sub Showsms(s As String )

Deflcdchar 0 , 32 , 32 , 32 , 21 , 21 , 31 , 32 , 32
Deflcdchar 1 , 32 , 20 , 20 , 20 , 20 , 31 , 32 , 32
Deflcdchar 2 , 32 , 32 , 32 , 6 , 30 , 16 , 16 , 16
Dim I As Byte , B As Byte
Dim Sret As String * 66 , Stemp As String * 6
Config Serialin = Buffered , Size = 12
'enable the interrupts because the serial input buffer works interrupts driven
Enable Interrupts
'define a constant to enable LCD feedback
Const Uselcd = 1
Const Senddemo = 1                          ' 1= send an sms
Const Pincode = "AT+CPIN=1392"        ' pincode change it into yours!
Const Phonenumber = "AT+CMGS=+989133835243"  ' phonenumber to
send                                                                  ' sms to
Wait 4
'Portb.0 = 1
#if Uselcd = 1
   Cls
   Lcd "SMS Demo"
#endif
'wait until the mode is ready after power up
Waitms 3000

#if Uselcd = 1
   Cls : Home : Lcd "Init modem"
#endif

Print "AT"        ' send AT command twice to activate the modem
```

```
Print "AT"
Flushbuf                        ' flush the buffer
Print "ATE0"
#if Uselcd = 1
   Home Lower
#endif
#if Uselcd = 1
   Cls : Home : Lcd "ATE0"
#endif

Do
 Print "AT"                                ' Waitms 100
 Getline Sret                              ' get data from modem
  #if Uselcd = 1
     Lcd Sret                              ' feedback on display
  #endif
Loop Until Sret = "OK"                     ' modem must send OK
Flushbuf                                   ' flush the input buffer
#if Uselcd = 1
   Home Upper : Lcd "Get pin mode"
#endif
Print "AT+cpin?"                           ' get pin status
Getline Sret
#if Uselcd = 1
   Home Lower : Lcd Sret
#endif
If Sret = "+CPIN: SIM PIN" Then
  Print Pincode                            ' send pincode
End If
Flushbuf
#if Uselcd = 1
   Home Upper : Lcd "set text mode"
#endif
Print "AT+CMGF=1"                         ' set SMS text mode
Getline Sret                               ' get OK status
#if Uselcd = 1
   Home Lower : Lcd Sret
#endif
'sms settings
Print "AT+CSMP=17,167,0,0"
Getline Sret
Print "AT+CNMI=0,1,2,0,0"
Getline Sret
Cls : Home : Lcd "hello user"
Wait 3
Cls : Home
```

```
'#if Senddemo = 1
 ' #if Uselcd = 1
  '   Home Upper : Lcd "send sms"
  '#endif
  'Print Phonenumber
  'Waitms 100
  'Print "BASCOM AVR SMS" ; Chr(26)
  'Getline Sret
  '#if Uselcd = 1
   '   Home Lower : Lcd Sret                          'feedback
  '#endif
'#endif
'main loop
Do
   Getline Sret                                  ' wait for a modem response
   #if Uselcd = 1
       Cls
       Lcd "Msg from modem"
       Wait 2
       Home Lower : Lcd Sret
   #endif
   I = Instr(sret , ":")                          ' look for :
   If I > 0 Then                                  'found it
      Stemp = Left(sret , I)
      Select Case Stemp
         Case "+CMTI:" : Showsms Sret                    ' we received an SMS
         ' hanle other cases here
      End Select
   End If
Loop                                          ' for ever
'subroutine that is called when a sms is received
's hold the received string
'+CMTI: "SM",5
Sub Showsms(s As String )
    #if Uselcd = 1
       Cls
    #endif
    I = Instr(s , ",")                            ' find comma
    I = I + 1
    Stemp = Mid(s , I)                              ' s now holds the index number
    #if Uselcd = 1
       Lcd "get " ; Stemp
       Waitms 1000                                 'time to read the lcd
    #endif

    Print "AT+CMGR=" ; Stemp                         ' get the message
```

```
  Getline S
' header   +CMGR: "REC READ","+316xxxxxxxx",,"02/04/05,01:42:49+00"
  #if Uselcd = 1
    Lowerline :   Lcd S
  #endif
  Do
    Getline S                                   ' get data from buffer
    Select Case S
      Case "PORT" :
                #if Uselcd = 1
                   Cls : Lcd S : For I = 0 To 7
                    Portc = I : Wait 2 : Next I
                #endif
      Case "Port" :
                #if Uselcd = 1
                   Cls : Lcd S : For I = 0 To 7
                    Portc = I : Wait 2 : Next I
                #endif
      Case "port" :
                #if Uselcd = 1
                   Cls : Lcd S : For I = 0 To 7
                    Portc = I : Wait 2 : Next I
                #endif
      Case "0" :
                #if Uselcd = 1
                   Cls : Lcd "0" : Portc = 0
                #endif
      Case "1" :
                #if Uselcd = 1
                   Cls : Lcd "1" : Portc = 1
                   #endif
      Case "2" :
                #if Uselcd = 1
                   Cls : Lcd "2" : Portc = 2
                #endif
     Case "3" :
                #if Uselcd = 1
                   Cls : Lcd "3" : Portc = 3
                #endif
     Case "4" :
                #if Uselcd = 1
                   Cls : Lcd "4" : Portc = 4
                #endif
     Case "5" :
                #if Uselcd = 1
                   Cls : Lcd "5" : Portc = 5
```

```
                #endif
        Case "6" :
                #if Uselcd = 1
                   Cls : Lcd "6" : Portc = 6
                #endif
        Case "7" :
                #if Uselcd = 1
                   Cls : Lcd "7" : Portc = 7
                #endif
        Case "8" : #if Uselcd = 1
                Locate 1 , 5
                Lcd Chr(2) ; Chr(1) ; Chr(0)
                #endif
        Case "OK" : Exit Do
         Case Else
      End Select
    Loop
    #if Uselcd = 1
      Home Lower : Lcd "remove sms"
    #endif
    Print "AT+CMGD=" ; Stemp                          ' delete the message
    Getline S                                  ' get OK
    #if Uselcd = 1
      Lcd S
    #endif
End Sub
'get line of data from buffer
Sub Getline(s As String)
  S = ""
  Do
    B = Inkey()
    Select Case B
      Case 0                                   'nothing
      Case 13                                   ' we do not need this one
      Case 10 : If S <> "" Then Exit Do              ' if we have received something
      Case Else
        S = S + Chr(b)                            ' build string
    End Select
  Loop
End Sub

'flush input buffer
Sub Flushbuf()
  Waitms 100                               'give some time to get data if it is there
  Do
    B = Inkey()                                  ' flush buffer
```

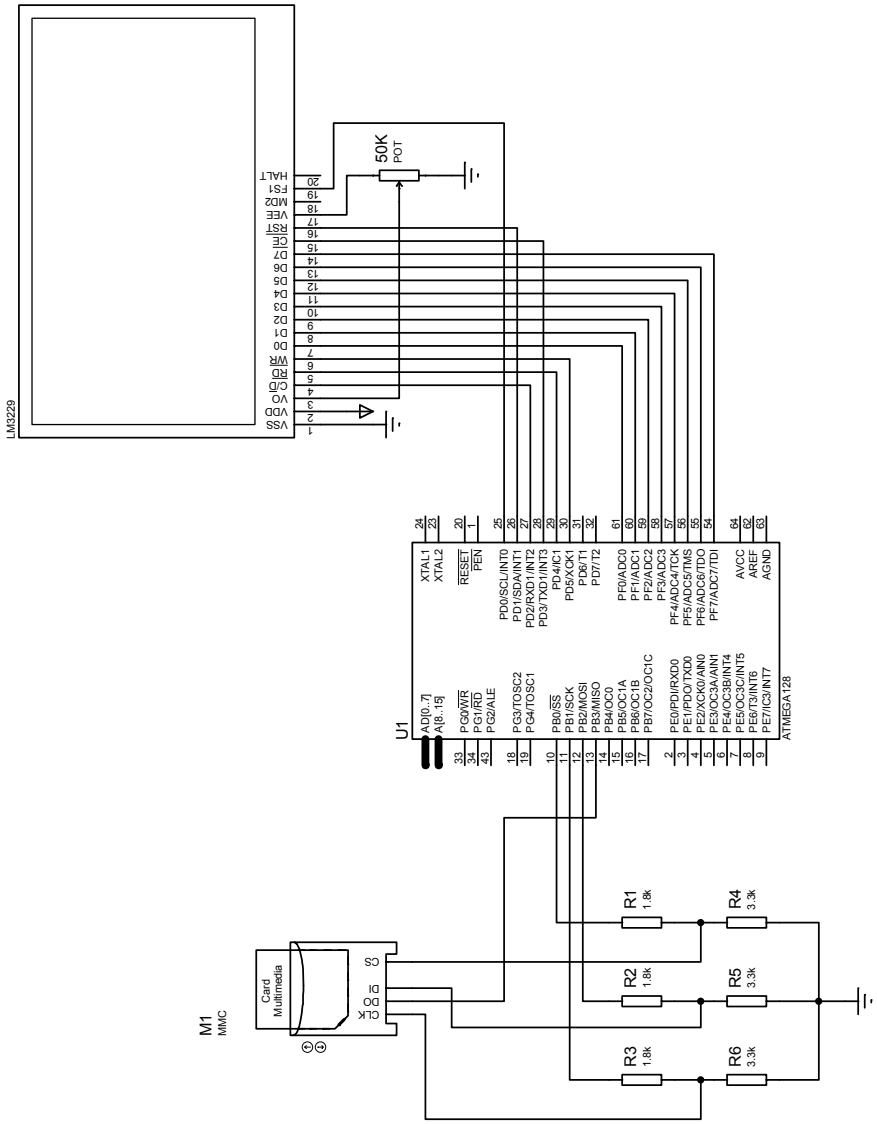
```
  Loop Until B = 0
End Sub
```

پروژه چهل‌ونهم: کنترل کننده LCD گرافیکی با قابلیت فایل خوانی از MMC

مدار کنترل کننده نمایش‌گرگرافیکی با قابلیت فایل‌خوانی از کارت حافظه چند رسانه (MMC) را طراحی و پیاده‌سازی نمائید؟

هدف از این پروژه: طراحی نرم‌افزار و سخت‌افزار لازم به منظور نمایش تصاویر و نوشته‌های مورد نظر کاربر بر روی یک نمایش‌گر گرافیکی است. در این پروژه، کاربر تصاویر مورد نظر خود را توسط برنامه طراحی شده با استفاده از نرم افزار ویژوال بیسیک ایجاد می‌کند. سپس با پردازش مناسب بر روی این تصاویر توسط برنامه، کد مناسب برای نمایش بر روی نمایش‌گر گرافیکی تولید و بر روی یک کارت MMC به صورت یک فایل ذخیره می شود. سپس کارت چند رسانه بر روی مدار سخت افزاری طراحی شده قرار می‌گیرد. میکروکنترلر AVR، ابتدا توسط یک برنامه جامع به منظور خواندن فایل از MMC و نمایش آن بر روی نمایش‌گر گرافیکی که در ادامه آورده شده است، برنامه‌ریزی می شود. سپس فایل موجود بر روی حافظه چند رسانه را می‌خواند و داده‌های مربوطه از فایل استخراج و بر روی نمایش‌گر گرافیکی ترسیم می شوند. در ادامه به اختصار سخت‌افزار مرتبط توضیح داده و نرم‌افزار مربوطه شرح داده می شود (برنامه‌های مربوط به نرم‌افزار طراحی شده توسط Visual Basic و میکروکنترلر در CD همراه کتاب موجود هستند).

شرح سخت‌افزار پروژه: در این پروژه، از میکروکنترلر ATmega128 استفاده شده است که وظیفه اصلی این میکرو دریافت داده‌ها از فایل‌های ذخیره شده در MMC و ارسال آن‌ها به تراشه T6963C، برای نمایش بر روی نمایش‌گر گرافیکی است. یکی از مهمترین دلایل انتخاب Atmega128، ظرفیت بالای حافظه SRAM آن می‌باشد که برابر با ٤ کیلوبایت است. زیرا استفاده از دو کتابخانه MMClib و AVR-DOS در برنامه Bascom به حافظه SRAM بیشتر از یک کیلوبایت نیازمند است. به همین دلیل در این پروژه از Atmega128 استفاده شده است. نمایش‌گرهای گرافیکی مدل T6963 در فصل سوم به طور کامل شرح داده شد. همچنین ساختار و عملکرد MMC نیز در فصل هشتم توضیح داده و به منظور آشنایی بیشتر مخاطبان کتاب، چندین پروژه توسط آن انجام شد. با توجه به مطالب بیان شده، مدار این پروژ در شکل(۱۰-۲۰) نشان داده شده است.



شکل(۱۰-۲۰): سخت‌افزار مدار نمایش‌گر گرافیکی

شرح نرم‌افزار:

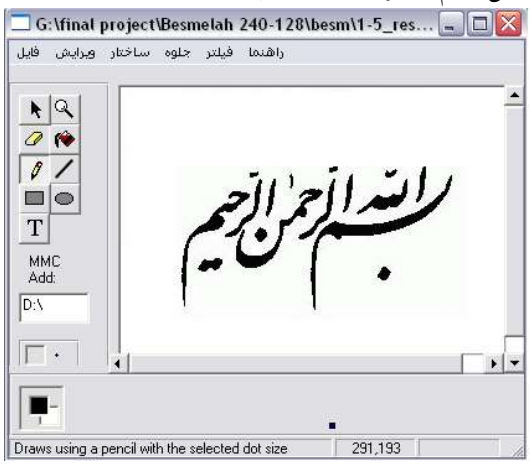
وظایف نرم‌افزار: از آنجایی که هر نمایش‌گر گرافیکی دارای ابعاد مشخصی است، در ابتدا لازم است که اندازه تصویر مورد نظر کاربر به ابعاد نمایش‌گر گرافیکی (که در اینجا ۱۲۸*۲٤۰ پیکسل است) تغییر داده شود. همچنین چون LCD مورد استفاده در این پروژه تک رنگ می‌باشد، باید تصویر مورد نظر کاربر، به یک تصویر تک رنگ تبدیل گردد. سپس وابسته به نوع نمایش‌گر گرافیکی مورد استفاده تصویر به یک کد باینری تبدیل شده تا بر روی MMC به صورت یک فایل ذخیره شود. بنابراین برنامه نرم‌افزاری نوشته شده، سه وظیفه عمده بر عهده دارد که عبارتند از:

۱- تغییر اندازه تصویر به اندازه ابعاد LCD مورد استفاده

۲- تبدیل تصویر مورد نظر کاربر به یک تصویر تک رنگ (سیاه یا سفید)

۳- تبدیل تصویر به کد باینری و ذخیره آن در یک فایل باینری

در نمایش‌گر گرافیکی مدل KS108 روش کد کردن بدین صورت انجام می‌شود که ۸ پیکسل اول در ستون اول به صورت یک بایت، ۸ پیکسل اول در ستون دوم، بایت بعدی و به همین صورت تا کلیه ستون‌ها پوشش داده شوند و در ادامه ۸ پیکسل دوم در ستون اول یک بایت، ۸ پیکسل دوم در ستون دوم بایت بعدی و به همین صورت تا آخر داده‌ها ذخیره می‌شـــوند. ولی در نمایش‌گر گرافیکی مدل T6963، روند کد کردن بدین صورت است که ۸ پیکسل در سطر اول به صورت یک بایت ، ۸ پیکسل بعدی در همان ســـطر، بایت دوم و به همین ترتیب ادامه پیدا کرده ، تا ســـطر اول پوشش داده شود و ادامه کار با سطر دوم خواهد بود تا سطر دوم نیز تمام شود و این روال تا پوشش کامل تمام سطرها ، ادامه خواهد یافت.



شکل(۱۰-۲۱): نمایی از برنامه

شرح منوهای نرم‌افزار طراحی شده توسط Visual Basic :

در این پروژه با استفاده از نرم افزار شیءگرای ویژوال بیسیک محیطی به صورت شکل (۱۰-۲۱) طراحی شده است البته برنامه بسیار کامل و جامعی برای آن نوشته شده است که با توجه به حجم زیاد کدهای آن، می‌توانید آن را در CD همراه کتاب ملاحظه نمایید. همچنین فایل اجرایی آن که مطابق شکل (۱۰-۲۱) است در CD همراه کتاب موجود و قابل استفاده است. این نرم‌افزار دارای امکانات زیر است:

: انتخاب یک ناحیه

: بزرگ نمایی یک ناحیه

: پاکن با ۸ سطح مقطع متفاوت

: رنگ‌آمیزی یک ناحیه

: مداد با ۸ سطح مقطع متفاوت

: کشیدن خط با ۸ سطح مقطع متفاوت

: کشیدن مستطیل و مربع و ایجاد جدول به سه شیوه زیر

: کشیدن بیضی و  دایره به سه شیوه زیر 

 : نوشتن یک عبارت متنی

: آدرس درایو مربوط به کارت MMC (محل ذخیره کدها) به عنوان مثال : F:\

منوی فایل شامل:

- ایجاد فایل جدید
- باز کردن فایل
- ذخیره فایل
- ذخیره مجدد فایل
- خروج از فایل

شکل(۱۰-۲۲): منوی فایل

منوی ویرایش شامل:

- لغو کردن
- دوباره انجام دادن
- برش
- کپی
- بازنشانی
- حذف
- برش دادن نما

شکل(۱۰-۲۳): منوی ویرایش

منوی ساختار شامل:

- نوع حاشیه:
  - یکسان
  - خط تیره
  - نقطه
  - خط تیره و نقطه
  - خط تیره و دو نقطه

شکل(۱۰-۲٤): منو ساختار

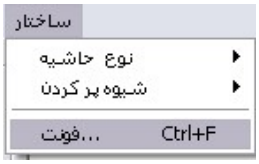
قسمت نوع حاشیه:

- شیوه پر کردن:
  - یکسان
  - خط افقی
  - خط عمودی
  - مورب \
  - مورب /
  - شطرنجی
  - شطرنجی مورب

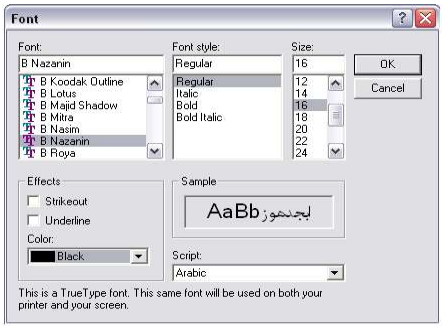
شکل(۱۰-۲۵): منوی ساختار قسمت شیوه پرکردن

فونت:

- انتخاب فونت فارسی و انگلیسی و نوع آن
- سایز فونت
- رنگ فونت
- افکت فونت و...



شکل(۱۰-۲٦): منوی ساختار قسمت فونت



شکل(۱۰-۲۷): انتخاب فونت

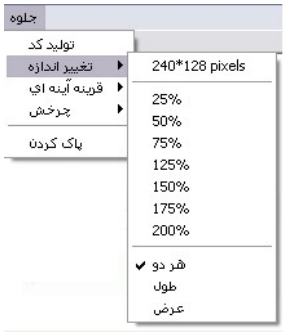
منوی جلوه:

- تولید کد

شکل(۱۰-۲۸): منوی جلوه قسمت تولید کد

- تغییر اندازه:

۱۲۸ * ۲٤۰ پیکسل
٪۲۵
٪۵۰
٪۷۵
٪۱۲۵
٪۱۵۰
٪۱۷۵
٪۲۰۰
تغییر اندازه هر دو (طول و عرض)
تغییر اندازه طول
تغییر اندازه عرض



شکل(۱۰-۲۹): منوی جلوه قسمت تغییر اندازه

- قرینه آینه‌ای
  - نسبت به خط افقی
  - نسبت به خط عمودی

شکل(۱۰-۳۰): منوی جلوه قسمت قرینه آینه‌ای

- چرخش:
  - ٤۵ درجه
  - ۹۰ درجه
  - ۱۳۵ درجه
  - ۱۸۰ درجه
  - ۲۲۵ درجه
  - ۲۷۰ درجه

- ۳۱۵ درجه
- چرخش ساعتگرد
- چرخش غیر ساعتگرد

شکل(۱۰-۳۱): منوی جلوه قسمت چرخش

- پاک کردن

شکل(۱۰-۳۲): منوی جلوه قسمت پاک کردن

منوی فیلتر شامل:

- تک رنگ کردن
- سیاه و سفید
- وارونه کردن رنگ

شکل(۱۰-۳۳): منوی فیلتر

راهنما:

شکل(۱۰-۳٤): راهنما

روش استفاده از نرم‌افزار: برای استفاده از این نرم افزار، ابتدا کاربر تصویر مورد نظر خود را ایجاد می‌نماید و یا می‌تواند یک تصـــویر موجود را باز کند، ســـپس آدرس کارت MMC را وارد کرده و در نهایت بر روی گزینه تولید کد از منوی جلوه

کلیک می‌کند. به این ترتیب، تصـــویر مورد نظر کاربر بر روی کارت MMC ذخیره می‌شـــود. به اســـتفاده از همین روش کاربر می‌تواند هر تعداد تصـــویر را که بخواهد تولید و ذخیره نماید.

برنامه نوشته شده برای میکروکنترلر:

✓ **برنامه**

```
$regfile = "M128def.dat "
$crystal = 4000000
$baud = 115200
$swstack = 64
$hwstack = 128
$framesize = 128
```

برای این‌کـه در BASCOM کـارت حـافظـه را راه انـدازی کنیم، بـایـد فـایـل "Config_MMC.bas" را در اختیار داشـــته باشـــیم. این فایل در مســـیر نصـــب برنامه BASCOM موجود است. مسیر زیر را دنبال کنید:

C:\Program Files\MCS Electronics\BASCOM-AVR\SAMPLES\AVRDOS

این فایل حاوی تنظیماتی برای پروتکل ارتباطی SPI می‌باشـــد. در خط ابتدایی برنامه ثابتی به نام CMMC_soft تعریف شده است که در صورت صفر بودن این پروتکل به صورت سخت افزاری پیکربندی می شود و اگر عددی غیر از صفر باشد SPI به صورت نرم افزاری پیکربندی می شود. پیکربندی نرم‌افزاری دارای سرعت کمتری بوده اما می‌توان از هر پایه ورودی/خروجی مورد نظر برای ارتباط با حافظه استفاده کرد که این موضوع در پیکربندی سخت افزاری صادق نبوده و اماکن‌پذیر نمی‌باشد. در صورت قرار دادن عدد غیر صفر، تنظیمات پایه‌های مورد نظر بعد از خط زیر در این فایل انجام می‌شود:

```
#else ' Config here SPI pins, if not using HW
```

حال برای این‌که بتوانیم در برنامه BASCOM طبق استاندارد FAT استفاده کنیم، باید به فایل "CONFIG_AVR-DOS.bas" دسترسی داشته باشیم که این فایل در مسیر زیر موجود است:

C:\Program Files\MCS Electronics\BASCOM-AVR\SAMPLES\AVRDOS

این فایل حاوی تعداد زیادی متغیر و ثابت بوده و ما برای سطح مبتدی وحتی متوسط کار چندانی با آن‌ها نداشـــته و فقط کامپایلر با آن‌ها مفهوم FAT را راه اندازی و پشتیبانی می‌کند. اما در این فایل دو ثابت بسیار مهم موجود است که ذکر توضیح در مورد آن‌ها ضروری است. اولین خط برنامه موجود cfilehandles می‌باشد که عدد متناظر با آن نشان دهنده تعداد فایلی است که می توان همزمان باز کرد و ثابت بعدی

csepfathandle می‌باشد که تعداد بافر برای کار با این استاندارد در آن مشخص می شود و بافر بزرگتر به شکل محدود روی سرعت کار تاثیر دارد.
نکته: توجه کنید که اعداد بالا هر کدام با دریافت ضـــریب، حجم قابل توجهی از SRAM داخلی میکروکنترلر را در اختیار می‌گیرد. برای تغییر دادن این اعداد به تو ضیحاتی که بالای هر ثابت نو شته شده کاملا دقت کنید زیرا رعایت نکردن این نکته باعث بروز خطا در کامپایلر می‌شود. (out of sram)

```
$include "Config_AVR-DOS.BAS"
$include "Config_MMC.bas"
"----------------------------------------
Lcd_wr Alias Portd.5
Lcd_rd Alias Portd.4
Lcd_ce Alias Portd.3
Lcd_cd Alias Portd.2
Lcd_rst Alias Portd.1
Lcd_fs1 Alias Portd.0
```

قبل از هر چیزی ذکر این نکته اهمیت دارد که میکروکنترلر در کار با FAT نیاز به یک ســاعت دارد. البته این موضــوع فقط برای modify کردن فایل و فولدرهاســت و به طور معمول تو سط RTC تامین می شود و تو صیه می شود که از این امکان ا ستفاده شود.

```
Config Clock = Soft
Config Date = Dmy , Separator = .
Enable Interrupts
Time$ = "20:22:00"
Date$ = "16.04.08"
Eightbyeight Alias 0
Busy1busy2 Alias &H03
Dawrdy Alias &H08
Cursorpointerset Alias &H21
Offsetregisterset Alias &H22
Addresspointerset Alias &H24
Texthomeaddress Alias &H40
Textareaset Alias &H41
Graphichomeaddress Alias &H042
Graphicareaset Alias &H043
Cgrommode Alias &H80
Cgrammode Alias &H88
Ormode Alias &H80
Exormode Alias &H81
Andmode Alias &H83
```

```
Textonly Alias &H84
Graphicsoff Alias &H90
Graphicson Alias &H98
Textoff Alias &H90
Texton Alias &H94
Cursoroff Alias &H90
Cursoron Alias &H92
Cursorblinkoff Alias &H90
Cursorblinkon Alias &H91
Cursorpattern Alias &HA0
Bottomline Alias &H0
Dataautowrite Alias &HB0
Dataautoread Alias &HB1
Autoreset Alias &HB2
Datardwr Alias &HC0
Bitreset Alias &HF0
Bitset Alias &HF8
Graphbase Alias &H0000
Textbase Alias &H1000
Column Alias 240
Row Alias 128
Dim K8 As Word
Dim K9 As Word
Dim K10 As Byte
Dim K11 As Byte
Dim K12 As Integer
Dim Fontsize As Byte
Dim Charsperrow As Word
Dim I As Word
Dim X As Byte , D As Byte
Dim Lcd_status As Byte
Dim Ff As Byte
Dim B As Byte
Dim Count As Long
Dim Le As Long
Dim Acc As Byte
Dim S1 As String * 30
Dim Test As Byte
Dim J As Byte
Dim S2 As String * 30
Dim Search As Byte
'---------------------------------------
Declare Sub Lcd_init
Declare Sub Statbusy12
Declare Sub Statdawrdy
Declare Sub Dataout(byval B As Byte)
```

```
Declare Sub Commandout(byval B As Byte)
Declare Sub Lcd_clear_graphic
Declare Sub Lcd_graphic
Declare Sub Lcd_pixel(byval X As Integer , Byval Y As Integer , Byval Setreset
As Byte)
Declare Sub Lcd_xy(byval X As Integer , Byval Y As Integer)
Declare Sub Setautomode
Declare Sub Autoout(byval B As Byte)
Declare Sub Resetautomode
Declare Sub Lcd_cursorxy(byval X As Byte , Byval Y As Byte)
Declare Sub Dataout2(byval B As Word)
Declare Sub Autozero(byval Terminate As Word)
Declare Sub Lcd_setup_page(byval Pages As Integer)
'---------------------------------------
Call Lcd_init
J = 1
Do
Call Lcd_graphic
Wait 2
Loop
End
'----------------------------------------
Sub Lcd_graphic
  Dim K1 As Byte
  Dim K2 As Byte
  Dim K3 As Word
  Dim K4 As Word
  K1 = Graphicson Or Textoff
  K2 = K1 Or Cursoroff
  Call Lcd_clear_graphic()
  Call Commandout(k2)
  Call Lcd_clear_graphic()
  Call Dataout(0)
  Call Commandout(addresspointerset)
  Call Setautomode()
```

بعد از مرحله فوق باید در برنامه از وجود کارت حافظه در مدار یقین حا صل کرد که این کار با استفاده از دستور زیر انجام می شود:

```
Drivecheck()
```

همان‌طور که در مثال زیر مشاهده می کنید در صورتی‌که کارت حافظه در مدار وجود نداشته باشد، خروجی تابع فوق عددی غیر از صفر خواهد شد:

```
A2:
If Drivecheck() <> 0 Then Goto A2
```

تابع بعدی تابعی است که در کارت حافظه را از نظر نرم‌افزاری ریست و init می‌کند:

```
Temp1 = Driveinit()
```

در صورت بروز هر گونه خطا، خروجی این تابع غیر صفر خواهد شد.
تابع بعدی بررسی فایل سیستم موجود در کارت حافظه را انجام می دهد. این امر اطلاعاتی از قبیل نوع FAT، ظرفیت حافظه و نظایر آن را در اختیار AVR-DOS قرار می‌دهد و انجام آن برای کار تحت پروتکل FAT ضروریست. این تابع نیز مانند بقیه توابع در صورت بروز خطا در خروجی عدد غیر صفر ایجاد می‌کند.

```
Temp1 = Initfilesystem(1)
```

```
Drivecheck()
A2:
If Drivecheck() <> 0 Then Goto A2
Ff = Drivereset()
Ff = Driveinit()
B = Initfilesystem(1)
```

دستور dir:

این دستور محتوای مسیر فعلی را نمایش می دهد. برای درک موضوع در مثال زیر تابعی را نوشته‌ایم که به دنبال فایل مورد نظر گشته و در صورت پیدا کردن آن عدد یک را بر می گرداند:

```
Function search (byval name as string) as byte
Local st as string*12
St=""
St=dir("*.*")
While len(st)>0
   If st=name then
      Search=1
      Exit function
    End if
    St=dir()
Wend
Search=0
End function
```

بدست آوردن طول یک فایل:

بوسیله دستور LOF می توان طول محتویات یک فایل را بدست آورد:

```
FileSize=LOF(FileNumber)
```

```
S1 = Str(j) + ".BIN"
S2 = ""
S2 = Dir( "*.*")
While Len(s2) > 0
  If S2 = S1 Then
```

```
    Goto Label
  End If
  S2 = Dir()
Wend
J = 1
Exit Sub
Label:
```

توضیح: برای کار با فایل‌ها، ابتدا باید به آن فایل یک عدد نسبت داد. این کار فقط برای راحتی و کاهش حجم برنامه می باشد به طوری که هر بار به جای اسم فایل عدد مربوط به آن را به کار می‌بریم. برای دریافت یک عدد آزاد که به فایل دیگری نسبت داده نشده باشد از تابع freefile به صورت زیر استفاده می‌شود:

```
Ff=freefile()
```

پس از این کار این عدد که در متغیرFF قرار گرفته به فایل نسبت می‌دهیم. این کار با استفاده از دستور open انجام می‌شود:

```
Open "ETC.txt" for binary as #ff
```

مثال فوق باعث ایجاد یک فایل با شماره ff و نوع دسترسی binary می‌شود.

```
Open S1 For Binary As #2
J = J + 1
For I = 1 To 14
  Call Autoout(0)
Next I
Count = 1
Do
  Le = Seek(#2)
```

دستور Get: از این دستور برای خواندن فایل‌های باینری با دسترسی تصادفی استفاده می‌شود:

```
Get #Filenumber,[Recordnum%],ReadData
```

این دستور رکورد شماره Recordnum را از فایلی با شماره Filenumber می‌خواند و در متغیر ReadData قرار می‌دهد. علامت کروشه نشان می‌دهد که پارامتر Recordnum اختیاری است و در صورتیکه ذکر نشود داده‌ها از رکورد بعدی فایل (جائی‌که اشاره گر فایل آنجا قرار دارد) خوانده می‌شوند.

```
  Get #2 , Acc , Le , Count
  Cls
  Lcd Acc
  Call Autoout(acc)
```

تشخیص انتهای فایل :

برای این‌که متوجه شویم به انتهای یک فایل ر سیده‌ایم از د ستور EOF ا ستفاده می کنیم. این دستور یکی از مقادیر True یا False را بر می‌گرداند که نشان می‌دهد به انتهای فایل رسیده‌ایم یا نه. از این تابع در حلقه های Do While استفاده می شود:

```
Do While Not (EOF(FileNumer))
.
.
.
Loop
```

حلقه فوق تا زمانی‌که فایل موردنظر به انتها نرسیده باشد اجرا خواهد شد.

```
  Loop Until Eof(#2) <> 0
  Close #2
  Call Resetautomode()
End Sub
'-----------------------------------------
Sub Lcd_init()
  Dim K5 As Byte
  Dim K6 As Byte
  Dim K7 As Byte
  Portd = &H1F
  Ddrc = &HFF
  Portf = &H00
  Ddra = &HFF A
  Lcd_ce = 1
  Lcd_rd = 1
  Lcd_wr = 1
  Lcd_cd = 1
  Lcd_rst = 0
  Waitus 1
  Lcd_rst = 1
  Waitus 1
  Lcd_fs1 = Eightbyeight
  If Lcd_fs1 = Eightbyeight Then
    Fontsize = 8
    Charsperrow = Column / Fontsize
   Else
    Fontsize = 6
    Charsperrow = Column / Fontsize
  End If
  K5 = Graphicsoff Or Textoff
  K6 = K5 Or Cursoroff
```

```
  Call Commandout(k6)
  Call Dataout2(graphbase)
  Call Commandout(graphichomeaddress)
  Call Dataout2(charsperrow)
  Call Commandout(graphicareaset)
  Call Dataout2(textbase)
  Call Commandout(texthomeaddress)
  Call Dataout2(charsperrow)
  Call Commandout(textareaset)
  K5 = Exormode Or Cgrommode
  Call Commandout(k5)
  K5 = Cursorpattern Or 7
  Call Commandout(k5)
  Call Lcd_clear_graphic
  Call Lcd_cursorxy(0 , 0)
End Sub
'----------------------------------
Sub Lcd_clear_graphic()
  Call Dataout2(graphbase)
  Call Commandout(addresspointerset)
  K8 = Charsperrow * Row
  Call Autozero(k8)
End Sub
'-------------------------------------
Sub Lcd_xy(byval X As Integer , Byval Y As Integer)
  Dim K13 As Word
  K8 = Y * Charsperrow
  K9 = K8 + Textbase
  K13 = K9 + X
  Call Dataout2(k13)
  Call Commandout(addresspointerset)
End Sub
'-------------------------------------
Sub Lcd_setup_page(byval Pages As Integer)
  Charsperrow = 16 * Pages
  Call Dataout2(textbase)
  Call Commandout(texthomeaddress)
  Call Dataout2(charsperrow)
  Call Commandout(textareaset)
End Sub
'-------------------------------------
Sub Lcd_cursorxy(byval X As Byte , Byval Y As Byte)
  Call Dataout(x)
  Call Dataout(y)
  Call Commandout(cursorpointerset)
End Sub
```

```
'-------------------------------------
Sub Autozero(byval Terminate As Word)
  Call Setautomode()
  For I = 0 To Terminate
    Call Autoout(0)
  Next I
  Call Resetautomode()
End Sub
'-------------------------------------
Sub Statbusy12()
  Ddra = &H00
  Do
    Lcd_ce = 0
    Lcd_rd = 0
    Waitus 1
    Lcd_status = Pina
    Lcd_ce = 1
    Lcd_rd = 1
    K10 = Lcd_status And Busy1busy2
  Loop Until K10 = Busy1busy2
  Ddra = &HFF
End Sub
'-------------------------------------
Sub Statdawrdy()
  Ddra = &H00
  Do
    Lcd_ce = 0
    Lcd_rd = 0
    Waitus 1
    Lcd_status = Pina
    Lcd_ce = 1
    Lcd_rd = 1
    K10 = Lcd_status And Dawrdy
  Loop Until K10 = Dawrdy
  Ddra = &HFF
End Sub
'-------------------------------------
Sub Dataout(byval B As Byte)
  Call Statbusy12()
  Lcd_cd = 0
  Portf = B
  Lcd_ce = 0
  Lcd_wr = 0
  Lcd_ce = 1
  Lcd_wr = 1
  Lcd_cd = 1
```

```
End Sub
'-------------------------------------
Sub Dataout2(byval B As Word)
  K10 = B And &HFF
  Call Dataout(k10)
  K11 = B / 256
  Call Dataout(k11)
End Sub
'-------------------------------------
Sub Commandout(byval B As Byte)
  Call Statbusy12()
  Portf = B
  Lcd_ce = 0
  Lcd_wr = 0
  Lcd_ce = 1
  Lcd_wr = 1
End Sub
'-------------------------------------
Sub Setautomode()
  Call Commandout(dataautowrite)
End Sub
'-------------------------------------
Sub Autoout(byval B As Byte)
  Call Statdawrdy()
  Lcd_cd = 0
  Portf = B
  Lcd_ce = 0
  Lcd_wr = 0
  Lcd_ce = 1
  Lcd_wr = 1
  Lcd_cd = 1
End Sub
'-------------------------------------
Sub Resetautomode()
  Call Statdawrdy()
  Portf = Autoreset
  Lcd_ce = 0
  Lcd_wr = 0
  Lcd_ce = 1
  Lcd_wr = 1
End Sub
'-------------------------------------
```

پروژه پنجاهم: صفحه لمسی (Touch Screen)

برنامه‌ای بنویسید که با قرار دادن انگشت بر روی صفحه لمسی نصب شده بر روی LCD گرافیکی ۶۴×۱۲۸ مختصات نقطه نمایش داده شود

شرح: صفحات لمسی به صفحاتی گفته می‌شود که بتوان تماس یک شیء و به‌طور خاص انگشت انسان با آن را با استفاده از خواص شیء از قبیل نیرو، گرما، هدایت الکتریکی، مقاومت اپتیک و نظایر آن تشخیص داد. صفحات لمسی دارای کاربردهای گسترده‌ای در صنایع مختلف و حتی کاربردهای عام میباشند. ترکیب یک صفحه لمسی شفاف با یک صفحه نمایش می‌تواند ابزار ایده‌آلی برای ارتباط با دستگاه‌های مختلف برای ما فراهم کند. صفحات لمسی برحسب خاصیتی که از آن برای تشخیص تماس مورد استفاده قرار می‌دهد، از تکنولوژی‌های مختلفی استفاده می‌کنند. در ادامه مهمترین این تکنولوژی‌ها به اختصار بیان می‌شوند.

تکنولوژی صفحات مقاومتی: در این تکنولوژی صفحه لمسی از چند لایه تشکیل شده است که مهمترین آن‌ها، دو لایه فلزی است. لایه‌های مقاومتی با فاصله کمی از هم جدا شده‌اند. وقتی این صفحه در نقطه خاصی توسط شیئ، لمس شود، صفحات مقاومتی در آن نقطه به یکدیگر متصل می‌گردند. در این حالت صفحه به صورت مقاومت عمل کرده و جریان الکتریکی صفحه تغییر می‌کند که توسط یک کنترلر، پردازش می‌شود. تغییر جریان به این نحو است که برحسب تعداد سیم‌های استفاده شده و مکان اتصال، مقاومت بین صفحات متفاوت خواهد بود. خروجی پنل لمسی معمولاً چهار تا هشت سیمه است و مکان لایه‌های مقاومتی برحسب تعداد خروجی مورد استفاده متفاوت خواهد بود. در حالت چهار سیمه، شیء می‌تواند در چپ، راست، بالا و پایین قرار گیرد. در حالت پنج سیمه، نقطه‌های قابل تشخیص شامل چهار گوشه و نقطه وسط خواهد بود. صفحات لمسی مقاومتی معمولاً قیمت مناسبی دارند ولی دارای قدرت تفکیک ٪۷۵ هستند (که با اضافه کردن فیلم‌های پلاستیکی و شیشه‌ای تا ٪۸۵ قابل افزایش است) و در مقابل اشیاء تیز مقاومت کمی دارند و صدمه می‌بینند. در مقابل این‌گونه صفحات از عوامل بیرونی مانند، آب و گرد و خاک تأثیر نمی‌پذیرند و امروزه بیشترین استفاده را دارند.

تکنولوژی صفحات خازنی: صفحه لمسی خازنی، صفحه‌ایست که با موادی همچون اکسید نازک ایندیم پوشانده شده است که جریان ثابتی را از سنسور عبور می‌دهد. بنابراین سنسور باعث میدان دقیقاً کنترل‌شده‌ای از الکترون‌های ذخیره‌شده در هر دو محور افقی و عمودی می‌شود که دارای ظرفیت خازنی خواهد بود. وقتی میدان خازنی

نرمال سنسور (در حالت پایه) توسط میدان خازنی دیگری (مثلاً انگشت انسان) تغییر کند، مدارات الکترونیکی که در گوشه‌های صفحه قرار دارند، برآیند تغییرات موج سینوسی میدان مرجع را اندازه‌گیری می‌کند و این اطلاعات را برای محاسبات ریاضی به کنترلر می‌فرستند. صفحات لمسی خازنی، می‌توانند توسط انگشت بدون پوشش یا با ابزاری رسانا که با دست گرفته شده باشد، لمس شوند. این‌گونه صفحات لمسی از عوامل خارجی تأثیر نمی‌پذیرند و قدرت تفکیک بالایی دارند. ولی مدارات آنالیز سیگنال آن‌ها قیمت آن‌ها را افزایش می‌دهند.

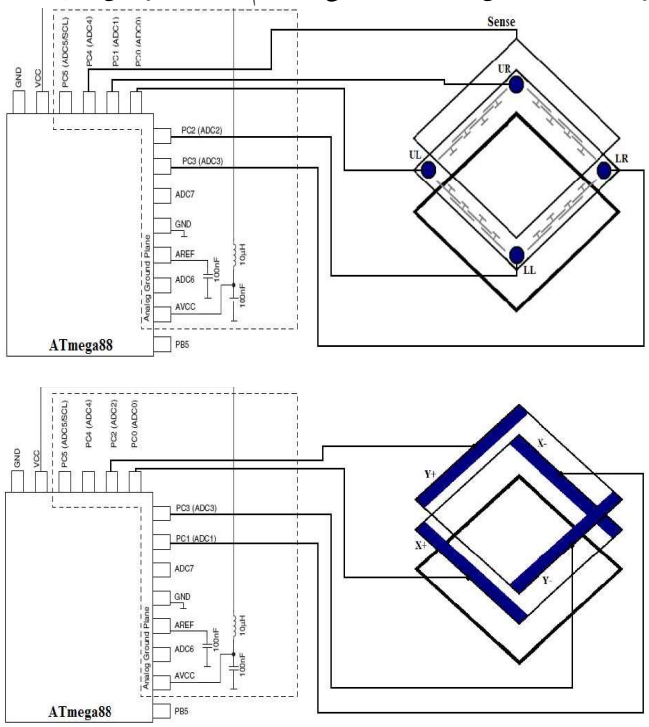
تکنولوژی مادون قرمز: این صفحات از یکی از دو روش کاملاً متفاوت استفاده می‌کنند. در یکی از این روش‌ها تغییرات دمایی سطح صفحه که توسط عوامل خارجی ایجاد شود را مورد استفاده قرار می‌دهد. این روش گاهی کند بوده و به دست‌هایی گرم نیاز دارند. در روش دیگر آرایه‌ای از سنسورهای مادون قرمز عمودی و افقی مورد استفاده قرار می‌گیرند که تغییرات در پرتوهای تنظیم‌شده نور مادون قرمز نزدیک سطح صفحه را تشخیص می‌دهند. این صفحات مقاوم‌ترین سطح را دارند و در کاربردهای نظامی که نیاز به صفحه نمایش لمسی دارند مورد استفاده قرار می‌گیرند.

تکنولوژی تصویرسازی نوری (پردازش تصویر): در این روش، تعداد دو عدد یا بیشتر سنسور تصویر در دور صفحه قرار می‌گیرند و نورهای پیش زمینه مادون قرمز در طرف دیگر صفحه در زاویه دید دوربین قرار می‌گیرند. یک تماس به صورت سایه بر روی هر جفت از دوربین‌ها می‌افتد که می‌تواند با مثلثی کردن برای یافتن مکان تماس مورد استفاده قرار گیرد.

شکل (۱۰-۳۵) نحوه اتصال صفحات لمسی ٤ سیمی و ۵ سیمی به AVR را نشان می‌دهد.

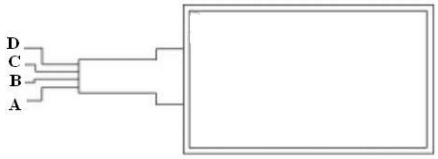
نحوه راه اندازی و اتصال قطعات: برای راه‌اندازی یک صفحه لمسی ۵ سیمی باید خط sense را به کانال ورودی ADC یک AVR متصل نمود. صفحات لمسی ٤ سیمی به دو کانال ADC ورودی نیاز دارند (-Xو-Y). بقیه خطوط باقیمانده باید به I/O ها وصل شوند(ترجیحا بر روی پورت مشابه) پیشنهاد می‌شود از AVCC به همراه یک خازن روی پایه AREF به عنوان مرجع ولتاژ استفاده گردد. جهت کاهش نویز صفحات

لمسی می‌توانید در درایور آن‌ها از خازن اتصال به زمین در حد 0.01uF استفاده کنید. اما در نظر داشته باشید این کار ثابت زمانی سیستم شما را افزایش خواهد داد.



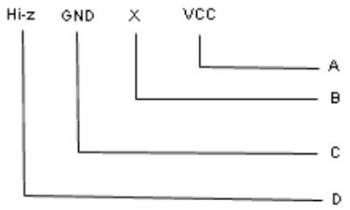
شکل (۱۰-۳۵): ارتباط صفحات لمسی با میکروکنترلر

☑ صفحه لمسی که در این قسمت معرفی می‌شود یک صفحه لمسی چهار سیمه به اندازه یک LCD با ابعاد ۱۲۸×٦٤ است که بر روی LCD گرافیکی متناظر آن، چسبیده می‌شود. مطابق شکل (۱۰-۳٦) این صفحه کاملا شفاف است به طوری که داده‌های نمایش داده شده بر روی LCD با قرار گرفتن این صفحه بر روی LCD با وضوح کامل مشخص می‌باشند.



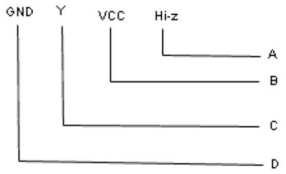
شکل (۱۰-۳٦): صفحه لمسی چهار سیمی

این نوع صفحه لمسی دارای ٤ سیم می‌باشد که توسط دو سیم میانی می‌توان مختصات x و y نقطه فشار داده شده بر روی صفحه را بدست آورد. خروجی x و y به صورت ولتاژ می‌باشد که بسته به نوع موقعیت مکان فشار داده شده به صفحه، ولتاژ خروجی تغییر می‌کند که در نهایت برای بدست آوردن مختصات باید این دو خروجی به A/D میکرو داده شوند. برای این که بتوانیم مختصات نقطه x را بدست آوریم باید سیم‌های تغذیه صفحه را به صورت شکل (۱۰–۳۷) را وصل نماییم:



شکل (۱۰–۳۷): سیم‌بندی صفحه نمایش برای خواندن مختصات x

سپس برای این که بتوانید مختصات Y را بدست آوردید، باید نحوه اتصال سیم‌های تغذیه به پایه‌ها را تغییر دهید و به صورت شکل (۱۰–۳۸) وصل نمایید.



شکل (۱۰–۳۸): سیم‌بندی صفحه نمایش برای خواندن مختصات y

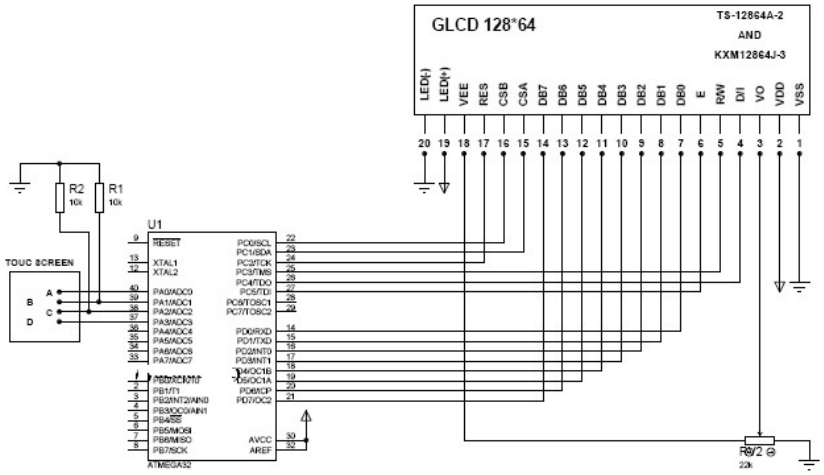
باید عمل تغییر دادن ولتاژ پایه‌ها برای بدست آوردن X,Y سریع صورت گیرد به همین دلیل، نمی توان به طور دستی این کار را انجام داد و باید با میکرو این عمل را انجام دهید. اگر به شکل‌های (۱۰–۳۷) و (۱۰–۳۸) توجه کرده باشید، می‌بینید که یکی از پایه‌ها باید HI-Z (امپدانس بالا) است. برای این که بتوانید با میکرو این حالت را به وجود آورید باید به صورت زیر عمل نمایید (این دستور برای میکروکنترلر AVR به زبان بیسیک می‌باشد). باید برای به وجود آوردن این حالت، پینی که به این پایه متصل شده است را به عنوان ورودی در نظر گرفته و سپس رجیستر پورت آن را صفر نمایید.

به عنوان مثال:

```
Config Porta.0 = input
Reset Porta.0
```

در مورد تغذیه هم که می‌توانید مدار را به ولتاژ ۵ ولت وصل نمایید. در شکل (۱۰-۳۹) روش اتصال صفحه لمسی به میکرو ATmega32 ترسیم شده است. اما باز هم اگر دقت کرده باشید در هر یک از دو حالت، سیم‌های صفحه لمسی مربوط به نقاط VCC، GND و همچنین HI-Z یکی نمی‌باشند. به منظور تنظیم حالت پایه‌ها برای بدست آوردن نقطه X و Y به صورت زیر عمل کنید:

```
Config Porta.0 = Output
Config Porta.1 = Input
Config Porta.2 = Output
Config Porta.3 = Input
Set Porta.0
Reset Porta.1
Reset Porta.2
Reset Porta.3
X = Getadc(1)
X = X / 4
Waitms 10
Config Porta.0 = Input
Config Porta.1 = Output
Config Porta.2 = Input
Config Porta.3 = Output
Reset Porta.0
Set Porta.1
Reset Porta.2
Reset Porta.3
Y = Getadc(2)
Y = Y / 8
```



شكل (١٠-٣٩): بلوك دياگرام كلى مدار

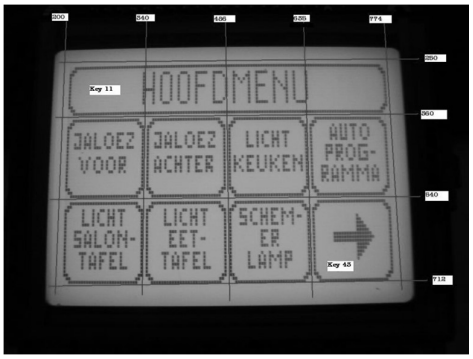
✓ برنامه كلى مدار به زبان Basic در محيط Bascom :

```
$regfile = "m32def.dat"                    ' specify the used micro
$crystal = 8000000                         ' used crystal frequency
' use baud rate
$hwstack = 32                              ' default use 32 for the hardware stack
$swstack = 10                              ' default use 10 for the SW stack
$framesize = 40                            ' default use 40 for the frame space
$lib "glcdKS108.lib"
Config Graphlcd = 128 * 64sed , Dataport = Portd , Controlport = Portc , Ce = 0 ,
Ce2 = 1 , Cd = 4 , Rd = 3 , Reset = 2 , Enable = 5

'The dataport is the portname that is connected to the data lines of the LCD
'The controlport is the portname which pins are used to control the lcd
'CE =CS1  Chip select
'CE2=CS2  Chip select second chip
'CD=Data/instruction
'RD=Read
'RESET = reset
'ENABLE= Chip Enable
 Config Adc = Single , Prescaler = Auto
Dim X As Word , Y As Word
```

```
Dim Aa As Word
Aa = 800
Dim S As Byte
S = 10
Setfont Font8x8
Stop Adc
Start Adc
Cls
Lcdat 1 , 1 , "yazdkit.com"
Wait 2
Cls
Main:
Config Porta.0 = Output
Config Porta.1 = Input
Config Porta.2 = Output
Config Porta.3 = Input

Set Porta.0
Reset Porta.1
Reset Porta.2
Reset Porta.3
X = Getadc(1)
X = X / 4
Waitms S
Config Porta.0 = Input
Config Porta.1 = Output
Config Porta.2 = Input
Config Porta.3 = Output

Reset Porta.0
Set Porta.1
Reset Porta.2
Reset Porta.3
Y = Getadc(2)
Y = Y / 8
Waitms S
Lcdat 1 , 5 , X ; "=x"
Lcdat 1 , 50 , Y ; "=y"
Waitms 200
```

```
Goto Main
'specify the font we want to use
End
 ==='Subroutines===
$include "font8x8.font"
```

تمرین: : برنامه زیر، برنامه یک صفحه لمسی قرار گرفته بر روی LCD است: ابتدا آن را تحلیل و سپس برنامه آن را در محیط Bascom بنویسید و اجرای آن را عمل مشاهده نمائید؟



شکل (۱۰-٤۰): منوی ایجاد شده بر روی نمایش‌گر گرافیکی

```
$regfile = "m128def.DAT"
$crystal = 7372800
$baud = 19200
$eepleave
$loadersize = 512
$hwstack = 100
$swstack = 75
$framesize = 40
$lib "glcdKS108.lib"          'Include the library for the KS-108 controler
Config Graphlcd = 128 * 64sed , Dataport = Porta , Controlport = Portc , Ce = 0 ,
Ce2 = 1 , Cd = 2 , Rd = 3 , Reset = 5 , Enable = 4
'The dataport is the portname that is connected to the data lines of the LCD
'The controlport is the portname which pins are used to control the lcd
'CE =CS1  Chip select
'CE2=CS2  Chip select second chip
'CD=Data/instruction
'RD=Read
'RESET = reset
'ENABLE= Chip Enable
```

```
Config Adc = Single , Prescaler = Auto , Reference = Internal
 'Setting up the a/d convertor
Config Timer1 = Timer , Prescale = 1024
 ' Timer 1 sets the screen back to the mainmenu after 10sec
Const Timer1preload = 58336
 'Timer 1 preload for 1 sec interrupt
Speaker Alias Portf.7
'Buzzer connected to portf.7, see circuit diagram
Dim Temp as byte , X as Word , Y as Word
Dim Row as byte , Keyarray(3) as Byte , Col as byte , Key as Byte , Keylus as
Byte
Dim Keypressed as Byte , Menu as byte
Dim Timecount as Byte
Setfont Font8x8
Enable Ovf1
Enable Interrupts
On Timer1 1secint                    ' on overflow jump to 1 sec int routine
Start adc                            ' Start the ad convertor
Stop timer1
Gosub ShowMainMenu                           ' Starts with the Mainmenu
' ===== 'Main=====
Main:
Do
************************'
 'Here your main prog
************************'
 Gosub Readtouch                             ' Read the touch cordinates
 Gosub WhichKey                              ' Which key is pressed
If menu > 1 then              ' Starts the timer if the menu is not the Mainmenu
   start timer1
Else
  Stop timer1
End if
If Keypressed > 0 then          ' Key is pressed
 Select case Menu
  ' Depends on the menu that we are in what to do with the pressed key
 Case 1 : Select case Keypressed                    'Mainmenu
            Case 11 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 12 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 13 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 21 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
```

```
            Case 22 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 23 : Gosub ShowLichtEettafelMenu     ' Shows a sub menu
            Case 31 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 32 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 33 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 41 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 42 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 43 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            End select
Case 13 : Select case Keypressed                        'LichtEetafelmenu (menu 13)
            Case 11 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 12 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 13 : Gosub ShowMainMenu              ' Back to the mainmenu
            Case 21 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 22 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 23 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 31 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 32 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 33 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 41 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 42 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            Case 43 : Print " You pressed key: " ; Keypressed ; " in menu: " ;
Menu
            End select
End select
 Keypressed = 0                                 ' Key is processed so put it back to 0
end if
Loop
 ==='Subroutines===
```

```
$include "font8x8.font"

secint:
' Interrupt routine will set the screen back to the mainmenu in 10 seconds
Incr Timecount                    ' increment every sec the counter with 1
If Timecount => 10 Then                    ' here is the 10 sec
 Stop Timer1 : Timecount = 0
 Gosub ShowMainMenu                              ' Sets the screen
Else
 Timer1 = Timer1preload
End If
Return
ShowMainMenu:                              'Shows the main menu
Menu = 1                                   'Menu number
Cls                                        'Clears the screen
Showpic 0 , 0 , HeaderHoofdmenu
 ' Draw the 9 pictures on the screen
Showpic 0 , 16 , Jalvoor
Showpic 32 , 16 , jalachter
Showpic 64 , 16 , Lichtkeuken
Showpic 96 , 16 , Autoprog
Showpic 0 , 40 , LichtSalontafel
Showpic 32 , 40 , LichtEetTafel
Showpic 64 , 40 , Schemerlamp
Showpic 96 , 40 , Pijlrechts
return

ShowLichtEettafelMenu:
' Shows the ShowLichtEetafel menu (sub menu 13(Menu = 13
' Menu number
Cls                                        'Clears the screen
Showpic 0 , 0 , HeaderLichteettafel
' Draw the 9 pictures on the screen
Showpic 0 , 16 , Stopbutton
Showpic 32 , 16 , PijlOmhoog
Showpic 64 , 16 , F1Preset
Showpic 96 , 16 , F2Preset
Showpic 0 , 40 , EscButton
Showpic 32 , 40 , PijlOmlaag
Showpic 64 , 40 , F3Preset
Showpic 96 , 40 , F4Preset
Return
WhichKey:                                  'Determins which key is pressed
Select Case X                                    ' For the x value
        Case 200 to 340 : Col = 10
' The cordinates on the touchscreen determins which key is pressed for example;
```

```
        Case 341 to 486 : Col = 20
 ' 341 to 486 are the cordinates where between the second column is
        Case 487 to 635 : Col = 30
        Case 636 to 774 : Col = 40
        Case else Col = 0
End select
Select Case Y                                   ' For the y value
        Case 250 to 360 : Row = 1
' The cordinates on the touchscreen determins which key is pressed for example;
        Case 361 to 540 : Row = 2
' 361 to 540 are the cordinates where between the second row is
        Case 541 to 730 : Row = 3
        Case else Row = 0
End select
Key = Col + row
' Add the row and column value so we get 1 key value
If Key > 0 then                                 ' There is a key pressed
  Keyarray(Keylus) = Key
  'Must read the same key 3 times in a row, to prefent bouncing
  Incr Keylus
  If Keylus > 3 then keylus = 1
   If Keyarray(1) = Keyarray(2) then
      If Keyarray(2) = Keyarray(3) then
  ' Key is correct read 3 times the same
            Sound Speaker , 1 , 65000             ' Give a key beep
            KeyPressed = Key
            Timecount = 0
      end if
   end if
 end if
Return

ReadTouch:
Config Pinf.0 = Output                          ' Makes port F.0 output
Config Pinf.2 = Output                          ' Makes port F.0 output
Set Portf.0                                     ' Sets port F.0 High
reset Portf.2                                   ' Sets port F.2 Low
ddrf.1 = 0                                      ' Sets port F.1 as input
ddrf.3 = 0                                      ' Sets port F.1 as input  because we need
it now as ad input
Waitms 20                                       ' Wait until the port is stable
Y = Getadc(3)                                   ' Read the ad value for the y
Y = 1024 - Y                                    ' Invert the reading
'Print "VALUE Y : " ; Y                          ' for debugging
Config Pinf.1 = Output                          ' Makes port F.1 output
Config Pinf.3 = Output                          ' Makes port F.3 output
```

```
reSet Portf.1                    ' Sets port F.1 Low
set Portf.3                     ' Sets port F.3 High
ddrf.0 = 0                      ' Sets port F.0 as input
ddrf.2 = 0                      ' Sets port F.2 as input  because we need
it now as ad input
Waitms 20                        ' Wait until the port is stable
X = Getadc(2)                     ' Read the ad value for the x
X = 1024 - X                      ' Invert the reading
'Print "VALUE  X : " ; X
return
====='The buttons images=====
HeaderHoofdmenu:
$Bgf "HeaderHoofdmenu.bgf"
HeaderLichteettafel:
$Bgf "HeaderLichtEettafel.bgf"
Jalvoor:
$Bgf "Jalvoor.bgf"
JalAchter:
$Bgf "Jalachter.bgf"
LichtSalonTafel:
$Bgf "LichtSalonTafel.bgf"
LichtEetTafel:
$Bgf "LichtEetTafel.bgf"
Lichtkeuken:
$Bgf "Lichtkeuken.bgf"
Schemerlamp:
$Bgf "Schemerlamp.bgf"
EscButton:
$Bgf "Esc.bgf"
PijlOmhoog:
$Bgf "PijlOmhoog.bgf"
PijlOmlaag:
$Bgf "PijlOmlaag.bgf"
PijlRechts:
$Bgf "PijlRechts.bgf"
PijlLinks:
$Bgf "PijlLinks.bgf"
Autoprog:
$Bgf "Autoprog.bgf"
StopButton:
$Bgf "Stop.bgf"
F1Preset:
$Bgf "F1preset.bgf"
F2Preset:
$Bgf "F2preset.bgf"
F3Preset:
```

```
$Bgf "F3preset.bgf"
F4Preset:
$Bgf "F4preset.bgf"
F3Leeg:
$Bgf "F3Leeg.bgf"
F4Leeg:
$Bgf "F4Leeg.bgf"
Leeg:
$Bgf "Leeg.bgf"
```

---

۲ تمرین: برنامه زیر برای راه‌اندازی صفحه لمسی در محیط CodeVision نوشته شده است آن‌را تحلیل نمائید؟

```
#include <mega32.h>
#include <delay.h>
#define ADC_VREF_TYPE 0x40
#define TOUCH_PORT    PORTA
#define TOUCH_DDR     DDRA
#define XP          0  //a pin of micro to connects to x+ of touch screen.example
PORTA.0
#define XN          1  //a pin of micro to connects to x- of touch screen.example
PORTA.1
#define YP          2  //a pin of micro to connects to y+ of touch screen.example
PORTA.2
#define YN          3  //a pin of micro to connects to y- of touch screen.example
PORTA.3
#define INPUT        0
#define OUTPUT       1
#define LOW          0
#define HIGH         1
//  the function of scan   touch-screen
void touch_scan(void);
//  global variables for X and Y position
unsigned int x,y;

void main(void)
  {
  // ADC initialization
  // ADC Clock frequency: 250.000 kHz
  // ADC Voltage Reference: AVCC pin
  ADMUX=ADC_VREF_TYPE & 0xFf;
  ADCSRA=0x86;

  while(1)
    {
    touch_scan();
```

```
   }
  }

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
  {
  ADMUX=adc_input | (ADC_VREF_TYPE & 0xFF);
  // Delay needed for the stabilization of the ADC input voltage
  delay_us(10);
  // Start the AD conversion
  ADCSRA|=0x40;
  // Wait for the AD conversion to complete
  while ((ADCSRA & 0x10)==0);
  ADCSRA|=0x10;
  // return ADC value
  return ADCW;
  }

void touch_scan(void)
  {
  // start scan of the screen
  // X-Coordinate-start-------------------------------------------------------
  //        |X+ = OUTPUT|  |X- = OUTPUT|  |Y+ = INPUT|  |Y- = INPUT|
  TOUCH_DDR= (OUTPUT<<XP) | (OUTPUT<<XN) | (INPUT<<YP) |
(INPUT<<YN);
  //         |X+ = GND|  |X- = VCC|   |Y+ = HI-Z| |Y- = HI-Z and to the ADC|
  TOUCH_PORT= (LOW<<XP) | (HIGH<<XN) | (LOW<<YP) | (LOW<<YN);
  x=read_adc(YN);
  //end----------------------------------------------------------------------
  // Y-Coordinate-start-------------------------------------------------------
  //        |X+ = INPUT|  |X- = INPUT|  |Y+ = OUTPUT|  |Y- = OUTPUT|
  TOUCH_DDR= (INPUT<<XP) | (INPUT<<XN) | (OUTPUT<<YP) |
(OUTPUT<<YN);
  //         |X+ = HI-Z| |X- = HI-Z and to the ADDC|
  TOUCH_PORT= (LOW<<XP) | (LOW<<XN) | (LOW<<YP) | (HIGH<<YN);
  //                              |Y+ = GND|  |Y- = VCC|
  y=read_adc(XN);
  //end----------------------------------------------------------------------
    // put touch-screen into standby mode-start---------------------------------
  TOUCH_DDR= (OUTPUT<<XP) | (INPUT<<XN) | (INPUT<<YP) |
(INPUT<<YN);
  TOUCH_PORT= (LOW<<XP) | (LOW<<XP) | (LOW<<YP) | (HIGH<<YN);
  //end----------------------------------------------------------------------
  // end of scan
  }
```

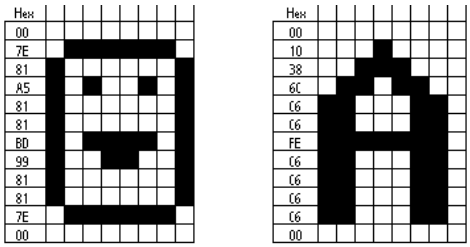
مهندس حسین لاچینی در وب‌سایت علمی و کاربردی خود (www.hlachini.com)، پروژه‌های زیادی در زمینه کار با میکروکنترلرها و راه‌اندازی تابلوهای روان با استفاده از این تراشه‌ها قرار داده که توجه بسیاری از علاقه‌مندان به این حوزه را جلب و ابهامات ذهنی آن‌ها را بر طرف کرده است. پیشنهاد می‌شود که با مشاهده این وب‌سایت (کاملاً جذاب)، با مفاهیم کلی تابلوهای روان آشنا شوید و اشکالات خود را از طریق تالار گفتمان مطرح نمائید. در زیر دو پروژه بسیار کابردی برگرفته از وب‌سایت مهندس لاچینی معرفی می‌شوند که در پروژه اول: به روش راه‌اندازی و کار با تابلوهای روان خواهیم پرداخت و در پروژه دوم: یک منبع تغذیه DC با محدوده ولتاژی صفر تا ۳۰ ولت توضیح داده می‌شود.

پروژه پنجاه‌ویکم: تابلو روان

برنامه یک تابلو روان را بنویسید که قابلیت حرکت عبارات نوشته شده از سمت چپ به سمت راست، از سمت راست به سمت چپ، از بالا به پائین و از پائین به بالا را داشته باشد؟

شرح ساختار کلی تابلو روان:

برای نمایش هر تصویر و یا متنی در تابلو روان ضروری است که ابتدا آن‌را به نقاط تشکیل دهنده تقسیم کنیم. در مورد حروف نیز به همین صورت عمل می‌کنیم و به ازای هر حرف یک جدول درست می‌کنیم. به مجموعه این جداول که شامل تمامی حروف می‌شوند، اصطلاحاً جدول گلایف می‌گویند. برای روشن شدن مطلب به تصویر شکل (۱۰-٤۱) دقت نمایید:



جدول گلایف نمونه برای کاراکتر A و صورتک خندان

شکل (۱۰-٤۱): جدول گلایف برای کاراکتر A و صورتک خندان

همان‌طور که مشاهده می‌کنید، در تصویر فوق نمونه‌ای از جدول گلایف برای حرف A و صورتک خندان ترسیم شده است و در ستون سمت چپ هر تصویر کد هگز (Hex) مربوط به هر سطر درج شده است و با این فرض که پهنای هر کاراکتر هشت پیکسل بوده ترسیم شده است. به ازای هر پیکسل فعال بیت مرتبط با آن یک

در نظر گرفته شده است. در نتیجه: در دو مثال فوق که ارتفاع هر کاراکتر ۱۲ سطر است، برای ذخیره اطلاعات هر کاراکتر به ۱۲ بایت نیاز داریم. حال بسته به زبان برنامه نویسی که شما از آن استفاده می‌کنید نحوه ذخیره بازیابی این جدول متفاوت خواهد بود. البته نرم‌افزارهایی جهت طراحی فونت نیز در این زمینه وجود دارد. نرم‌افزار استفاده شده در این پروژه، اجازه می‌دهد مستقیماً یک جدول گلایف کامل را طراحی کنیم و از طرفی خروجی فونت‌های طراحی شده به فرمت مختلف می‌باشند که اجازه استفاده در زبان‌های مختلف را می‌دهد. با استفاده از همین نرم‌افزار، فونت‌های مختلف فارسی طراحی شده است. که شامل ۱٦ فونت مختلف می‌باشد که دارای ارتفاع ۱٦ سطری است و همچنین دو مدل فونت انگلیسی ۸ و ۱۲ سطری را شامل می‌شود.

☑ **مدار عملی یک تابلو روان ساده:**

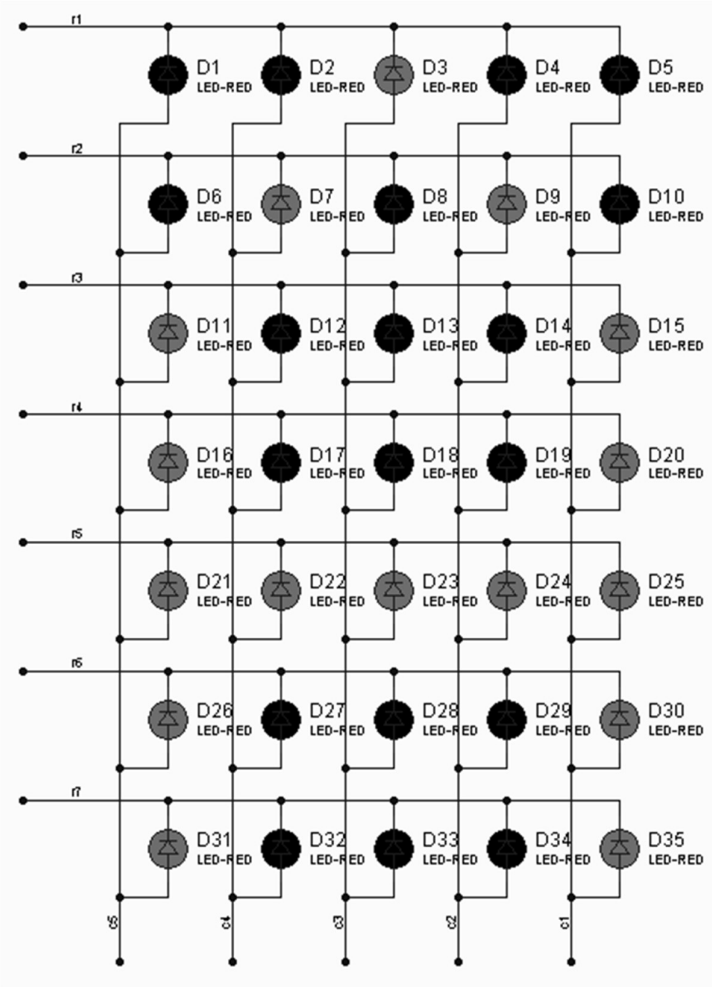
در این قسمت، نخستین مدار عملی تابلو روان، برای شما توضیح داده می‌شود. البته قبل از شروع این بحث باید عنوان شود که در طراحی این مدارات سعی شده که از حداقل قطعات ممکن استفاده شود تا مدار از نظر سخت‌افزاری تا حد امکان ساده و ارزان باشد.

لیست قطعات مدار

- میکروکنترلر ATmega8*
- LED قرمز ۳۵ عدد
- مقاومت ۳۳۰ اهمی ۵ عدد
- سوکت ۲۸ پین جهت میکرو ATmega8
- برد هزار سوراخ ۱۵ در ۱۰ سانتی متر

* قطعاتی که تعداد آنها مشخص نشده، مقدار آن یک عدد است.

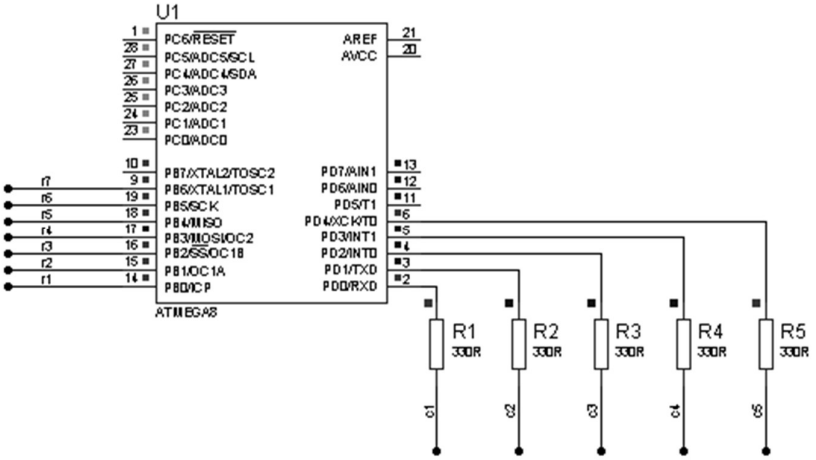
حال برای ساخت مدار بهتر است ابتدا ماتریس LED را بسازید. نقشه این ماتریس در شکل (۱۰-٤۲) آورده شده است.



شکل (۱۰-۴۲): ماتریس LED

همانطور که مشاهده می‌کنید در شکل (۱۰-۴۲) کاتدهای LED های موجود در یک سطر به هم و آندهای LED های موجود در یک ستون نیز به یکدیگر متصل شده‌اند. بعد از ساخت ماتریس و قبل از ادامه مونتاژ سایر قطعات ماتریس LED را توسط اعمال یک ولتاژ بین ۳ تا ۹ ولت به سطرها و ستون ها تست کنید تا از اتصال صحیح آن‌ها اطمینان حاصل کنید. اکنون نوبت به نصب سوکت ۲۸ پین میرسد، دلیل استفاده از سوکت، جلوگیری از صدمه دیدن میکروکنترولر در حین لحیم کاری است و در

عین حال به شما اجازه میدهد که از میکرو در پروژه های دیگر هم استفاده کنید. بعد از نصب سوکت به سراغ مقاومت‌های ۳۳۰ اهمی رفته و آن‌ها را به پین‌های صفر تا پنج Portd وصل نمایید و سر دیگر مقاومت‌ها را به ستون‌های ماتریس LED متصل نمائید. به نقشه شکل (۱۰-۴۳) دقت کنید.



شکل (۱۰-۴۳): اتصال سطرها و ستون‌ها به میکروکنترلر

همانطور که در شکل (۱۰-۴۳) نیز مشاهده می‌کنید: مقاومت R1 به ستون ۱ و مقاومت R2به ستون ۲ و ... مقاومت R5 به ستون ۵ ماتریس LED متصل می‌شود. اکنون مدار کامل شده است و شما صاحب یک تابلو روان واقعی هستید. و پس پروگرم کردن میکرو می‌توانید نتیجه کار خود را مشاهده نمائید. اکنون نوبت به توضیح برنامه است. برنامه این میکرو به زبان Basic نوشته شده است و از نرم افزار Bascom استفاده می‌شود.

☜ **جاروب سطری تابلو روان**

```
$regfile = "m8def.dat"
$crystal = 8000000
```

همانطور که می‌دانید، دستوراتی که با علامت "$"در BASCOM آغاز می‌شوند، جزو دستورات کامپایلر به حساب می‌آیند. و در زمان کامپایل کدی را تولید نمی‌کنند. دو دستور فوق نیز به همین صورت هستند. در دستور اول نوع میکرو برای کامپایلر تعریف می‌شود که در اینجا ATmega8 می‌باشد و در دستور بعدی فرکانس کریستال بر حسب هرتز مشخص می‌شود. در این برنامه مقدار فرکانس تعریفی هشت مگاهرتز

است. توجه داشته باشید که در مدار تابلو روان، از کریستال خارجی استفاده نشده است. بنابراین: این دستور تعیین کننده فرکانس اسیلاتور داخلی میکروکنترلر می‌باشد.

```
Config Portb = Output
Config Portd = Output
```

در دو دستور فوق پورت‌های B,D به‌عنوان خروجی پیکربندی گشته‌اند. در این مدار تابلو روان از پورت B برای راه‌اندازی و کنترل سطرها و از پورت D جهت راه‌اندازی ستون‌ها استفاده شده است.

```
Dim Row As Byte
Dim Scan As Byte
```

در این دو دستور از دو متغیر از نوع بایت تعریف شده است. متغییر Row جهت شمارش سطرها و متغیر Scan جهت تهیه سیگنال جاروب در سطرها استفاده می شود. بعد از موارد فوق در برنامه، به حلقه اصلی برنامه می‌رسیم. جهت ساخت این حلقه از دستور Do-Loop استفاده شده و به دلیل عدم ذکر هیچگونه شرطی در این دستور، دستورات موجود در بدنه این حلقه به تعداد بینهایت بار اجرا می‌شوند.

```
Scan = &B11111110
```

در ابتدای حلقه Do-Loop متغیر Scan، مقدار دهی اولیه می‌شود تا سیگنال مورد نیاز جهت فعال نمودن سطر نخست تولید گردد. با توجه به ساختار ماتریس LEDمورد استفاده در این تابلو روان (اتصال کاتد LED های موجود در یک سطر به یکدیگر)، جهت فعال‌سازی یک سطر باید پین مربوط به آن سطر در میکرو صفر شود و سایر پین‌های مربوط به دیگر سطرها، یک شوند .همانطور نیز که مشاهده کردید در دستور فوق نیز بیت نخست متغیر Scan نیز صفر شده که مربوط به سطر اول ماتریس است و سایر بیت‌ها نیز یک شده‌اند. در نتیجه فقط سطر اول فعال خواهد شد و سایر سطرها غیر فعال هستند. در ادامه برنامه به حلقه For-Nextمی‌رسیم. متغیر Row در این حلقه با صفر مقدار دهی اولیه می‌شود و اجرای دستورات حلقه تا رسیدن این متغییر به عدد ٦ تعریف شده است. بنابراین تعداد دفعات اجرای دستورات درون حلقه ٧ بار خواهد بود. در واقع ما در درون این حلقه یک بار کامل کل سطرهای ماتریس را که هفت عدد می‌باشند، جاروب می‌کنیم.

```
For Row = 0 To 6
    Portb = Scan
    Rotate Scan , Left
    Portd = Lookup(row , Gelayof)
    Waitus 20
    Portd = 0
Next Row
```

در اولین دستور در حلقه For-Next مقدار متغیر Scan بر روی پورت B میکروکنترلر قرار می‌گیرد تا سطر مورد نظر در ماتریس فعال شود. در دستور بعدی متغییر Scan به اندازه یک بیت به سمت چپ شیفت چرخشی داده می‌شود. با این شیفت صفر موجود در این متغیر به سمت چپ منتقل شده و جای آنرا یک بیت یک پر می‌کند. به عنوان مثال: در نخستین بار اجرای این دستور متغیر Scan از مقدار ۱۱۱۱۱۱۱۰ به مقدار ۱۱۱۱۱۱۰۱ تغییر می‌کند و در شیفت بعدی به ۱۱۱۱۱۰۱۱ تا اینکه بعد از هفتمین شیفت بصورت ۱۰۱۱۱۱۱۱ در می‌آید. که در هفتمین مرحله در واقع بیت هفتم، صفر شده است که منجر به فعال شدن سطر هفتم ماتریس خواهد شد. در این برنامه هدف: نمایش حرف A است، بنابراین جدولی با نام Gelayof در برنامه تعریف شده است. همان‌طور نیز که در زیر مشاهده می‌کنید، جهت ذخیره اطلاعات مربوط به حرف A، از هفت بایت استفاده شده و اطلاعات مربوط به هر سطر را در یک بایت قرار گرفته است. از طرفی چون در این مدار پهنای ماتریس LED، پنج است، فقط از پنج بیت اول هر بایت استفاده شده و سه بیت با ارزش آن صفر شده‌اند. شما بنابر نیاز خود می‌توانید با تغییر دادن وضعیت بیتها به نمایش هر شکل و یا کاراکتری بپردازید.

```
Gelayof:
Data &B00000100
Data &B00001010
Data &B00010001
Data &B00010001
Data &B00011111
Data &B00010001
Data &B00010001
```

حال اطلاعات این جدول، مرحله به مرحله و سطر به سطر خوانده شده و بر روی پورت D قرار می‌گیرند.این عمل توسط دستور Lookup در برنامه صورت می‌گیرد. در این دستور بایت مورد نظر (اطلاعات سطر مورد نظر) توسط متغیر Row تعیین می‌شود. بعد از قرار دادن اطلاعات مربوط هر سطر در پورت D به اندازه ۲۰ میکروثانیه این اطلاعات در پورت نگاه داشته می‌شوند تا LED های موجود در آن سطر روشن بمانند و اثر آن در چشم بیننده باقی بماند. سپس پورت D صفر می‌شود و اعمال فوق مجدداً برای سطر بعدی تکرار می‌شود. بعد از هر بار جاروب کامل تمامی سطرها، کنترل برنامه از حلقه For-Next خارج شده و مجدداً متغیر Scanمقدار دهی اولیه شده تا برای جاروب مجدد آماده گردد. به همین سادگی شما یک نمونه ساده از تابلو روان را ساختید!

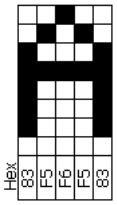
✓ **برنامه**

```
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Output
Config Portd = Output
Dim Row As Byte
Dim Scan As Byte
Do
   Scan = &B11111110
   For Row = 0 To 6
      Portb = Scan
      Rotate Scan , Left
      Portd = Lookup(row , Gelayof)
      Waitus 20
      Portd = 0
   Next Row
Loop
End 'end program

Gelayof:
Data &B00000100
Data &B00001010
Data &B00010001
Data &B00010001
Data &B00011111
Data &B00010001
Data &B00010001
```

☜ **جاروب ستونی تابلو روان**

در برنامه قبل، از روش جاروب سطری در برنامه نویسی تابلو روان استفاده شد. حال در این قسمت روش جاروب ستونی در تابلو روان را توضیح خواهیم داد و از همان مدار قبلی برای تست این روش استفاده می‌کنیم و تنها برنامه میکروکنترلر تغییر می کند. در جاروب ستونی ابتدا نخستین ستون را فعال می‌کنیم و سپس ستون دوم فعال می‌شود و این روال ادامه پیدا می‌کند تا به ستون آخر برسیم. اگر به مدار ماتریس LEDدقت کنید، می‌بینید که آند LED های هر ستون به هم متصل شده‌اند، پس جهت فعال نمودن هر سطر می‌باید: آن‌را به سطح ولتاژ مثبت متصل کنیم، از طرفی به دلیل این‌که در ماتریس LED، کاتد LED های هر سطر به هم متصل هستند، جهت روشن

نمودن هر LED سطر متناظر با آن LED بایستی به سطح ولتاژ صفر متصل گردد. از همین جا روشن می‌شود که در طراحی جدول گلایف بایستی بر خلاف برنامه قبلی به ازای نقاطی که می‌خواهیم LED در آن نقطه روشن باشد، بایستی عدد صفر را قرار دهیم. شکل (۱۰-٤٤) را با دقت ملاحظه نمائید.



شکل (۱۰-٤٤): جدول گلایف به صورت ستونی

در شکل (۱۰-٤٤) نحوه طراحی جدول گلایف برای جاروب ستونی در تابلو روان نشان داده شده است. که باعث به نمایش در آمدن کاراکتر A در ماتریس LED تابلو روان خواهد شد. همان‌طور که مشاهده می‌کنید، به ازای نقاط فعال (نقاطی که LED در آنجا روشن خواهد بود) مقدار بیت متناظر با آن صفر در نظر گرفته شده است. از طرفی اگر به یاد داشته باشید در برنامه قبلی تعداد بایت‌های مورد نیاز جهت نمایش کاراکتر A، هفت بایت بود ولی در اینجا ما فقط با استفاده از پنج بایت توانستیم همین کار را انجام دهیم. برنامه به زبان Basic و توسط نرم‌افزار Bascom نوشته شده است.

✓ **برنامه**

```
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Output
Config Portd = Output
Dim Col As Byte
Dim Scan As Byte
Do
   Scan = &B00000001
   For Col = 0 To 4
      Portd = Scan
      Rotate Scan , Left
      Portb = Lookup(col , Gelayof)
      Waitus 20
      Portb = &B11111111
   Next Col
Loop
End 'end program
```

```
Gelayof:
Data &B10000011
Data &B11110101
Data &B11110110
Data &B11110101
Data &B10000011
```

☜ حرکت در تصاویر از راست به چپ

در این قسمت چگونگی ایجاد حرکت دادن متون و تصاویر را در تابلو روان آموزش می‌دهیم. البته برنامه تابلو روان بسته به اینکه شما در سخت‌افزار خود از چه روشی جهت جاروب ماتریس LED استفاده کرده باشید فرق خواهد کرد. معمولاً اگر از جاروب ستونی استفاده کنید: نوشتن برنامه برای حرکت متن به صورت افقی ساده‌تر است و در جاروب سطری نوشتن برنامه برای حرکت عمودی ساده‌تر خواهد بود. البته مداری که در ابتدای این بحث برای تست عملی به شما معرفی شد، قابلیت اجرای جاروب سطری و ستونی را با توجه به ساختار حقیقتاً ساده خود دارد. حالا تصور کنید که ما به جای این‌که در این روش ابتدا اطلاعات ستون اول را در محل اصلی خود یعنی همان ستون اول نمایش دهیم، در ستون دوم نمایش دهیم و همین‌طور سایر اطلاعات را به اندازه یک ستون جابجا نمایش دهیم، چه اتفاقی می‌افتد؟ بله، متن به نمایش در خواهد آمد ولی بصورت افقی به اندازه یک ستون جابجا شده است. اساس کار حرکات افقی (چپ و راست) در تابلوهای روان با جاروب ستونی هم به همین شکل است. به برنامه زیر دقت کنید. برای شروع بحث، برنامه حرکت متن را در مدارات تابلو روان با جاروب سطری را آغاز می‌کنم. همان‌طور که می‌دانید در این روش در جدول گلایف، اطلاعات هر کاراکتر به صورت ستونی ذخیره شده و ما در هر لحظه فقط یک ستون را فعال می‌کنیم و سایر ستون ها غیر فعال هستند.

```
For S = 0 To 4
    For Refresh = 1 To 10
        Scan = &B00000001
        For Col = 0 To 4
            Index = S + Col
            Portb = Lookup(Index , Gelayof)
            Portd = Scan
            Waitus 250
            Rotate Scan , Left
            Portd = &H00
        Next Col
```

```
    Next Refresh
  Next S
```

همان‌طور که ملاحظه می‌نمائید: این برنامه از سه حلقه For-Next تو در تو تشکیل شده است. دو حلقه درونی صرفاً به نمایش اطلاعات می‌پردازند. حلقه داخلی یک‌بار تمام ستون‌ها را جاروب می‌کند. حلقه وسط این کار را تا ده مرتبه تکرار می‌کند تا متن به اندازه کافی دیده شود و حلقه خارجی هر بار محل خواندن اطلاعات از جدول گلایف را به اندازه یک بایت (یک ستون) جابجا می‌کند. که باعث به حرکت درآمدن متن در تابلو می‌شود.

دقت کنید که ما در دستور Lookup آدرس خواندن اطلاعات را از متغیر Index می خوانیم که این متغیر هم مقدارش توسط جمع زدن مقادیر Col و S محاسبه می‌شود که Col ستون در حال جاروب شدن بر روی ماتریس LED را نشان می‌دهد و S محل شروع خواندن مقادیر در جدول گلایف است. برنامه را اجرا کنید. تعجب کردید. یک کاراکتر عجیب که مثل دنباله به کاراکتر اصلی شما چسبیده و به دنبال آن حرکت میکند. این کاراکتر ناشناخته از کجا پیدا شد؟ بله دوباره به سراغ برنامه رفته و آن‌را با دقت بیشتری بررسی می‌کنیم. زمانی که مقدار متغیر S یک می‌شود یعنی لحظه‌ای کاراکتر به اندازه یک ستون بر روی ماتریس LED جابجا می‌شود مقدار متغییر Index برابر با ۵ خواهد شد (البته در زمانی که مقدار متغیر Col برابر با ٤ است). و در دستور بعدی توسط فرمان Lookup میکرو به سراغ برداشتن بایت ششم ذخیر شده در جدول Gelayofمی‌رود. در حالی که ما این جدول را تا ۵ بایت تعریف کردیم و عملاً در مکان ششم هیچ متغیری توسط ما تعریف نشده و محتویات آن ناحیه نامشخص است.

```
Gelayof:
Data &B10000011
Data &B11110101
Data &B11110110
Data &B11110101
Data &B10000011
```

و باعث می‌شود که شما آن کاراکتر نا آشنا را مشاهده کنید. برای رفع این مشکل باید به انتهای جدول گلایف به اندازه ابعاد ماتریس اضافه کنیم و مقادیری را در آن قرار دهیم که زمان نمایش، چیزی بر روی تابلو روان نمایان نشود. (در این برنامه تمامی خانه‌ها یک می‌شوند).

```
Gelayof:
Data &B10000011
Data &B11110101
Data &B11110110
```

```
Data &B11110101
Data &B10000011

Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
```

دوباره برنامه را با جدول گلایف جدید اجرا می‌کنیم. به نظر می‌رسد که دیگر مشکلی وجود ندارد. اما هنوز یک ایراد کوچک در تابلو باقی مانده است و این است که هنگام نمایش حرکت به سمت چپ کاراکتر A یک‌دفعه به صورت کامل ظاهر می‌شود و سپس به تدریج از سمت چپ تابلو روان خارج می‌شود. در حالی‌که بهتر این هست که به تدریج از سمت راست تابلو روان ظاهر شود و از سوی دیگر خارج شود. برای انجام این مهم، بهترین کار: قرار دادن یک فضای خالی در ابتدای کاراکتر در جدول گلایف است که ابعاد این فضا به اندازه ابعاد ماتریس LED تابلو روان می‌باشد. که در تابلوی ما این مقدار به اندازه یک کاراکتر است. پس جدول گلایف به شکل زیر تغییر می‌کند.

```
Gelayof:
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111

Data &B10000011
Data &B11110101
Data &B11110110
Data &B11110101
Data &B10000011

Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
```

اکنون باید در برنامه جاروب تغییر در ابعاد جدول گلایف را مشخص کرد. یعنی در حلقه For-Next اول مقدار نهایی متغیر S را از ٤ به ٩ افزایش دهیم (به اندازه طول

اضافه شده به جدول گلایف) و دستورات مربوط به جاروب به شکل زیر در خواهند آمد.

```
For S = 0 To 9
      For Refresh = 1 To 10
         Scan = &B00000001
         For Col = 0 To 4
            Index = S + Col
            Portb = Lookup(Index , Gelayof)
            Portd = Scan
            Waitus 250
            Rotate Scan , Left
            Portd = &H00
         Next Col
      Next Refresh
   Next S
```

متن کامل برنامه حرکت به سمت چپ در تابلو روان با جاروب ستونی:

```
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Output
Config Portd = Output
Dim Col As Byte
Dim Scan As Byte
Dim Refresh As Byte
Dim Index As Byte
Dim S As Byte
Do
   For S = 0 To 9
      For Refresh = 1 To 10
         Scan = &B00000001
         For Col = 0 To 4
            Index = S + Col
            Portb = Lookup(Index , Gelayof)
            Portd = Scan
            Waitus 250
            Rotate Scan , Left
            Portd = &H00
         Next Col
      Next Refresh
   Next S
Loop
End          'end program
```

```
Gelayof:
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111

Data &B10000011
Data &B11110101
Data &B11110110
Data &B11110101
Data &B10000011

Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
```

☜ حرکت در تصاویر از چپ به راست

✓ برنامه

```
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Output
Config Portd = Output
Dim Col As Byte
Dim Scan As Byte
Dim Refresh As Byte
Dim Index As Byte
Dim S As Byte
Dim R As Byte

Do
   For S = 0 To 9
      R = 9 - S
      For Refresh = 1 To 10
         Scan = &B00000001
         For Col = 0 To 4
            Index = R + Col
            Portb = Lookup(Index , Gelayof)
            Portd = Scan
```

```
            Waitus 250
            Rotate Scan , Left
            Portd = &H00
        Next Col
    Next Refresh
  Next S
Loop
End 'end program

Gelayof:
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111

Data &B11110111
Data &B11110111
Data &B11010101
Data &B11100011
Data &B11110111

Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
Data &B11111111
```

☜ حرکت در تصاویر از بالا به پایین

✓ برنامه

```
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Output
Config Portd = Output
Dim Col As Byte
Dim Scan As Byte
Dim Refresh As Byte
Dim Index As Byte
Dim S As Byte
Dim R As Byte
Dim Mask As Byte
Dim A As Byte
```

```
Do
   For S = 0 To 8
      A = 8 - S
      Mask = &HFF
      Shift Mask , Left , S
      For Refresh = 1 To 10
         Scan = &B00000001
         For Col = 0 To 4
            Index = S + Col
            Portb = Lookup(Index , Gelayof)
            Shift Portb , Right , A
            Portb = Portb Or Mask
            Portd = Scan
            Waitus 250
            Rotate Scan , Left
            Portd = &H00
         Next Col
      Next Refresh
   Next S

   For S = 0 To 8
      A = 8 - S
      Mask = &HFF
      Shift Mask , Right , A
      For Refresh = 1 To 10
         Scan = &B00000001
         For Col = 0 To 4
            Index = S + Col
            Portb = Lookup(Index , Gelayof)
            Shift Portb , Left , S
            Portb = Portb Or Mask
            Portd = Scan
            Waitus 250
            Rotate Scan , Left
            Portd = &H00
         Next Col
      Next Refresh
   Next S
Loop
End         'end program
```

```
Gelayof:
Data &B11110111
Data &B11101111
Data &B11000000
Data &B11101111
Data &B11110111
```

☜ حرکت در تصاویر از پایین به بالا

✓ برنامه

```
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Output
Config Portd = Output
Dim Col As Byte
Dim Scan As Byte
Dim Refresh As Byte
Dim Index As Byte
Dim S As Byte
Dim R As Byte
Dim Mask As Byte
Dim A As Byte

Do
   For S = 0 To 8
      A = 8 - S
      Mask = &HFF
      Shift Mask , Right , S
      For Refresh = 1 To 10
         Scan = &B00000001
         For Col = 0 To 4
            Index = S + Col
            Portb = Lookup(Index , Gelayof)
            Shift Portb , Left , A
            Portb = Portb Or Mask
            Portd = Scan
            Waitus 250
            Rotate Scan , Left
            Portd = &H00
         Next Col
      Next Refresh
```

```
  Next S

  For S = 0 To 8
    A = 8 - S
    Mask = &HFF
    Shift Mask , Left , A
    For Refresh = 1 To 10
      Scan = &B00000001
      For Col = 0 To 4
        Index = S + Col
        Portb = Lookup(Index , Gelayof)
        Shift Portb , Right , S
        Portb = Portb Or Mask
        Portd = Scan
        Waitus 250
        Rotate Scan , Left
        Portd = &H00
      Next Col
    Next Refresh
  Next S
Loop
End        'end program

Gelayof:
Data &B11110111
Data &B11111011
Data &B10000001
Data &B11111011
Data &B11110111
```

با استفاده از نرم‌افزار شکل (۱۰-٤٥) که در CD همراه کتاب موجود است، ابتدا کلیه عبارات مورد نظر برای نمایش بر روی تابلو روان را نوشته و جهت حرکت و سرعت آن‌ها را مشخص کنید. همچنین نوع فونت و عرض نمایش و نوع تراشه را مشخص نمائید. سپس بر روی گزینه Save کلیک نماید تا فایلی با پسوند bas. در مسیر مطلوب شما ذخیره شود و توسط نرم‌افزار Bascom این فایل را باز کنید و آن را کامپایل نمائید تا فایل Hex ایجاد شود. مطابق شکل (۱۰-٤٦) یا مدار موجود در CD همراه کتاب مداربندی نمائید و فایل Hex را بر روی میکروکنترلر پروگرم نمائید و اجرای برنامه خود را ملاحظه نمائید.

شکل (۱۰-٤٥): نرم‌افزار تولید کننده کد برای تابلو روان

✓ برنامه

```
$regfile = "m32def.dat"
$crystal = 1000000
Config Porta = Output
Config Portb = Output
Config Portc = Output
Config Portd = Output
Dim Scan As Byte
Dim I As Word
Dim Refresh As Byte
Dim T As Byte
Dim B As Byte
Dim D As Word
Dim U As Word
Dim S As Word
Dim M As Word
Dim E As Word
```

```
Dim A As Word

Do
' +----------------------------+
' | Scrolling Text00 to Left  |
' +----------------------------+
 For S = 0 To 240
  E = S + 7
  For Refresh = 1 To 5
   Scan = &H01
   For I = S To E

    For B = 0 To 15

     D = B * 8
     D = D + I
     Portd = Lookup(d , Text00)

     If B < 8 Then
      Portb = 2 ^ B
      Portb = 0
     Else
      U = B - 8
      Portc = 2 ^ U
      Portc = 0

     End If
    Next B

    Porta = Scan
    Rotate Scan , Left , 1
    Waitms 2
    Porta = 0

   Next I
  Next Refresh
 Next S
' +----------------------------+
' | Scrolling Text01 to Right |
' +----------------------------+
 For S = 0 To 384
  M = 384 - S
  E = M + 7
  For Refresh = 1 To 5
   Scan = &H01
   For I = M To E
```

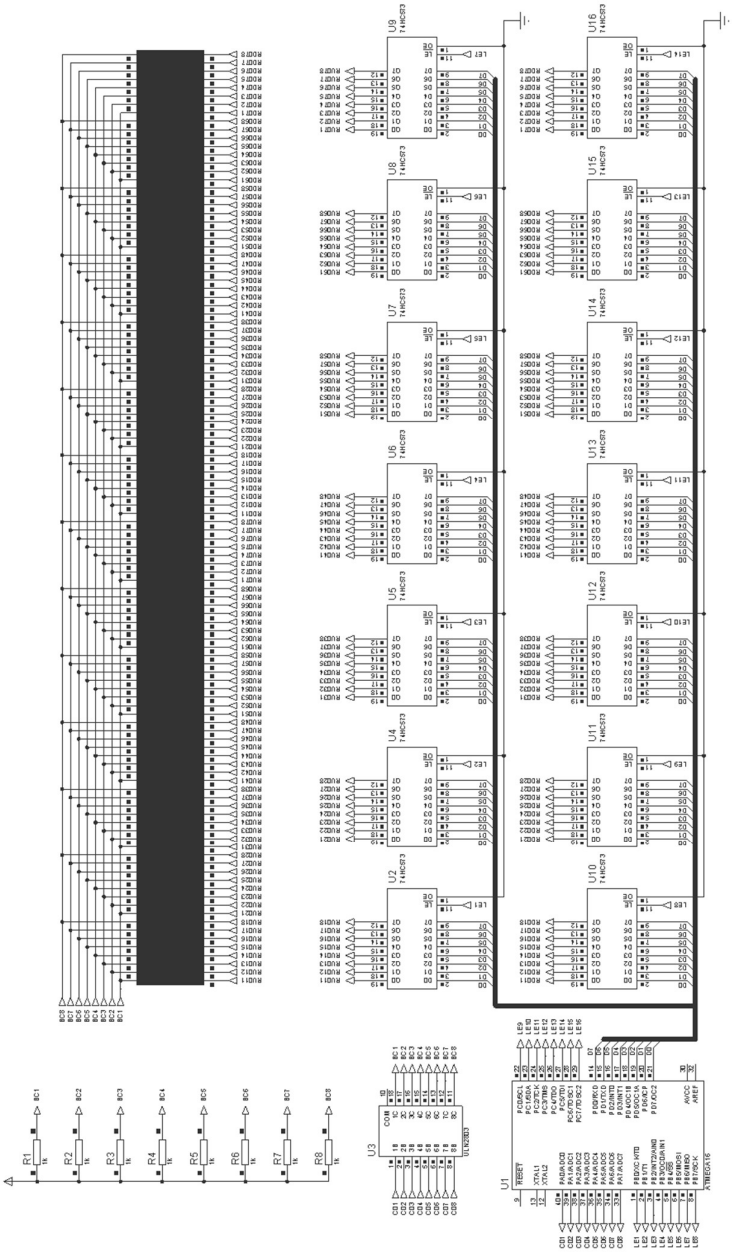
```
    For B = 0 To 15

     D = B * 8
     D = D + I
     Portd = Lookup(d , Text01)

     If B < 8 Then
      Portb = 2 ^ B
      Portb = 0
     Else
      U = B - 8
      Portc = 2 ^ U
      Portc = 0

     End If
    Next B

    Porta = Scan
    Rotate Scan , Left , 1
    Waitms 2
    Porta = 0

   Next I
  Next Refresh
 Next S
' +---------------------------+
' | Scrolling Text02 to Down  |
' +---------------------------+
 For S = 0 To 7
  M = 15
  If S = 0 Then M = 35
  For Refresh = 1 To M
   Scan = &H01
   For I = 0 To 7

    A = &HFF00
    Rotate , A , Right , S

    For B = 0 To 15

     D = B * 8
     D = D + I
     Portd = Lookup(d , Text02)
     T = Portd And A
```

```
    Shift Portd , Left , S
    Rotate T , Left , S
    If B < 8 Then
     Portb = 2 ^ B
     Portb = 0
    Else
     U = B - 8
     Portc = 2 ^ U
     Portc = 0

    End If
   Next B

   Porta = Scan
   Rotate Scan , Left , 1
   Waitms 2
   Porta = 0

  Next I
 Next Refresh
Next S
' +----------------------------+
' | Scrolling Text03 to Up    |
' +----------------------------+
 For S = 0 To 7
  M = 15
  If S = 0 Then M = 35
  For Refresh = 1 To M
   Scan = &H01
   For I = 0 To 7
    A = &HFF00
    Rotate , A , Left , S

    For B = 0 To 15

     D = B * 8
     D = D + I
     Portd = Lookup(d , Text03)
     T = Portd And A

     Shift Portd , Right , S
     Rotate T , Right , S
     If B < 8 Then
      Portb = 2 ^ B
      Portb = 0
     Else
```

```
      U = B - 8
      Portc = 2 ^ U
      Portc = 0

     End If
    Next B

    Porta = Scan
    Rotate Scan , Left , 1
    Waitms 2
    Porta = 0

   Next I
  Next Refresh
 Next S
' +---------------------------+
' | Scrolling Text04 to Right |
' +---------------------------+
 For S = 0 To 136
  M = 136 - S
  E = M + 7
  For Refresh = 1 To 1
   Scan = &H01
   For I = M To E

    For B = 0 To 15

     D = B * 8
     D = D + I
     Portd = Lookup(d , Text04)

     If B < 8 Then
      Portb = 2 ^ B
      Portb = 0
     Else
      U = B - 8
      Portc = 2 ^ U
      Portc = 0

     End If
    Next B

    Porta = Scan
    Rotate Scan , Left , 1
    Waitms 2
    Porta = 0
```

```
   Next I
  Next Refresh
 Next S

Loop
End                                         'end program

Text00:
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H38 , &H28 , &H18
'Farsi character
Data &H08 , &H08 , &H08 , &H08 , &H08 , &H14 , &H14 , &H08
'Farsi character
Data &H08 , &H08 , &H18 , &H28 , &H78 , &H28 , &H10 , &H00
'Farsi character
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00
 'Farsi character
Data &H00 , &H00 , &H00 , &H00 , &H00 , &H38 , &H28 , &H18 ,…
```

برای مشاهده سایر کدها به CD مراجعه شود.

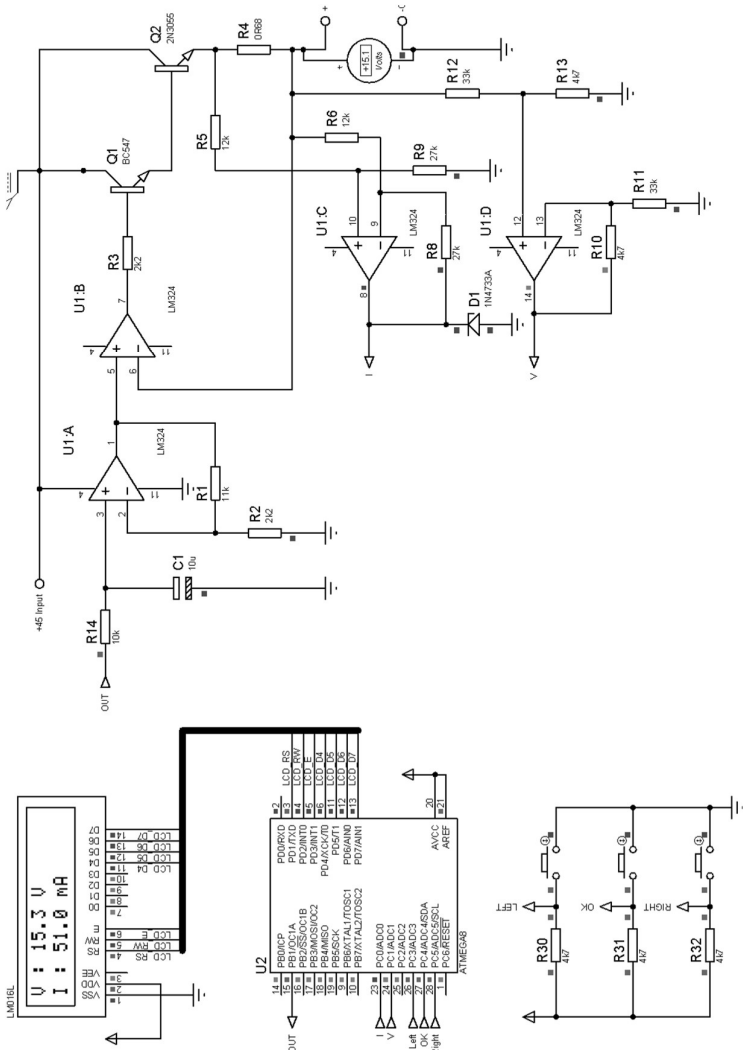


شکل (۱۰-٤٦): شماتیک سخت‌افزاری تابلو روان با استفاده از نمایش‌گر نقطه‌ای

پروژه پنجاه‌ودوم: منبع تغذیه دیجیتالی DC

برنامه و بلوک دیاگرام یک منبع تغذیه DC صفر تا ۳۰ ولت را نوشته و ترسیم نمائید؟

حل: با توجه به حجم زیاد برنامه برای مشاهده آن به CD همراه کتاب مراجعه شود .



شکل (۱۰-۴۷): شماتیک سخت‌افزاری منبع تغذیه دیجیتالی صفر تا سی ولت

در پایان این فصل: پیاده‌سازی یک پروژه با استفاده از نرم‌افزار WinAVR آورده شده است. زبان برنامه‌نویسی این نرم‌افزار C است و با توجه به رایگان بودن آن، مورد توجه کشورهای زیادی که قانون کپی برداری در آن‌ها اجرا می‌شود، است. آموزش روش کار با این نرم‌افزار در CD همراه کتاب آورده شده است. و در اینجا تنها به انجام یک پروژه خواهیم پرداخت. همچنین در CD همراه کتاب چندین پروژه عملی با WinAVR آورده شده است، همچنین روش ساخت پروگرمر USB همراه با برنامه‌ها و فایل‌های مورد نیاز آن قرار داده شده است.

پروژه پنجاه‌وسوم: راه‌اندازی LCD کاراکتری موازی به روش سریال

✍ برنامه‌ای بنویسید که در آن LCD کاراکتری موازی به صورت سریال راه‌اندازی شود؟

✓ برنامه

```
#include <avr/io.h>
#include <inttypes.h>
#include <stdlib.h>
#include <stdio.h>
#include <avr/pgmspace.h>
//#include <avrlibdefs.h>
//#include <avrlibtypes.h>
#include "LCD_3w.h"
const uint8_t welcomeln1[] PROGMEM="Hamed Saghaei\0";
const uint8_t backslash[] PROGMEM=
{
0b00000000,                    //back slash
0b00010000,
0b00001000,
0b00000100,
0b00000010,
0b00000001,
0b00000000,
0b00000000
};

int main (){
/*
u32 format;
u32 data;
LCDinit();
//LCDcursorOFF();
```

```
data="LCD4BIT 2*16",format="%s";
LCDprintData(format,data);
LCDGotoXY(0, 1);
data="kavirelectronic",format="%s";
LCDprintData(format,data);

*/
/*  send data byte
data=123,controll="%u";
LCDprintData(format,data);
LCDsendChar('m');         //send char to lcd
LCDcursorOFF();            //cursor off
LCDGotoXY(0,1);
LCDstring("kavirelectronic",15); // send string to lcd
CopyStringtoLCD(welcomeln1, 3, 1); //copy string to lcd
*/
LCDinit();
LCDclr();
LCDdefinechar(backslash, 0);

//LCDstring("kavirelectronic",15); // send string to lcd
CopyStringtoLCD(welcomeln1, 3, 1); //copy string to lcd

return(0);
}



/-----------------------------------------------------------------------
#include <inttypes.h>
#include <stdlib.h>
#include <stdio.h>
#include <avrlibtypes.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <util/delay.h>
#include "LCD_3w.h"
//max for _delay_ms(14)
//max for _delay_us(48)
//write char to LCD CGRAM

void LCDwritebyte(uint8_t a, uint8_t pc)
{
LCDsendCommand(a);
LCDsendChar(pc);
```

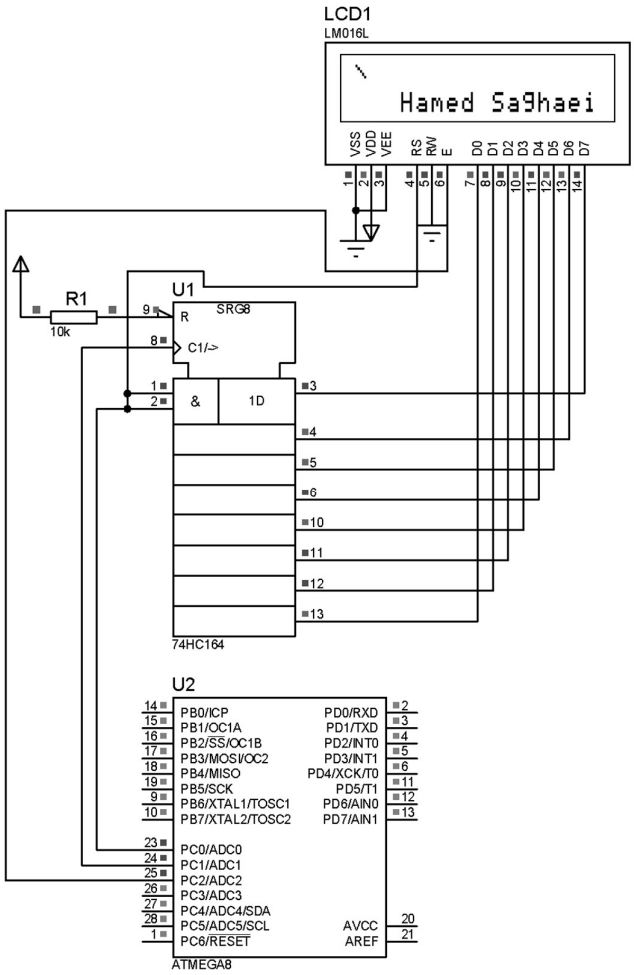
```
}
void LCDdefinechar(const uint8_t *pc,uint8_t char_code){
                        uint8_t a, pcc;
                        uint16_t i;
                        a=(char_code<<3)|0x40;
                        for (i=0; i<8; i++){
                                pcc=pgm_read_byte(&pc[i]);
                                LCDwritebyte(a++,pcc);
                                }
}
void LCDclr(void)
{
// clear LCD
                        LCDsendCommand(1<<LCD_CLR);
}

void LCDhome(void)
{
                        // move cursor to home
                        LCDsendCommand(1<<LCD_HOME);
}
void sendByteToRegister(uint8_t LCDdata)
{
uint8_t i, temp;
PORTC&=~_BV(datapin);                           // sets datapin to output a
LOW
for (i=0;i<=7;i++)                              //clear shift 74HC164
register
                        {
                                PORTC|=_BV(clockpin);           // sets clockpin to
output a HIGH
                                //_delay_ms(1);
                                PORTC&= ~_BV(clockpin);         // sets clockpin to
output a LOW
                                //_delay_us(40);
                        }
temp=LCDdata;
for (i=0;i<=7;i++)                      //write 8bit LCDdata to 74HC164 register
                        {
                                PORTC |= (LCDdata&1);
                                //_delay_us(20);
                                PORTC |= _BV(clockpin);         // sets clockpin to
output a HIGH
                                //_delay_ms(1);
                                PORTC&= ~_BV(clockpin);         // sets clockpin to
output a LOW
```

```
                    //_delay_us(40);
                    PORTC&=~_BV(datapin);          // sets datapin to
output a LOW
                    //_delay_us(40);
                    LCDdata=temp>>1;
                    temp=LCDdata;
          }
}
void LCDenableCommand()
{
PORTC |= _BV(Epin);
}
void LCDdisableCommand()
{
PORTC &= ~_BV(Epin);
}
void LCDenableData()
{
PORTC |= _BV(datapin);
PORTC |= _BV(Epin);
}
void LCDdisableData()
{
PORTC &= ~_BV(Epin);
PORTC &= ~_BV(datapin);
}
void LCDsendChar(uint8_t letter) //forms data ready to send to 74HC164
{
sendByteToRegister(letter);//sends char to shift register
LCDenableData();
_delay_ms(1);
LCDdisableData();
}
void LCDsendCommand(uint8_t cmd) //forms data ready to send to 74HC164
{
sendByteToRegister(cmd);//sends command to shift register
LCDenableCommand();
_delay_ms(1);
LCDdisableCommand();
}

void LCDstring(uint8_t* data, uint8_t nBytes)
{
          register uint8_t i;

          // check to make sure we have a good pointer
```

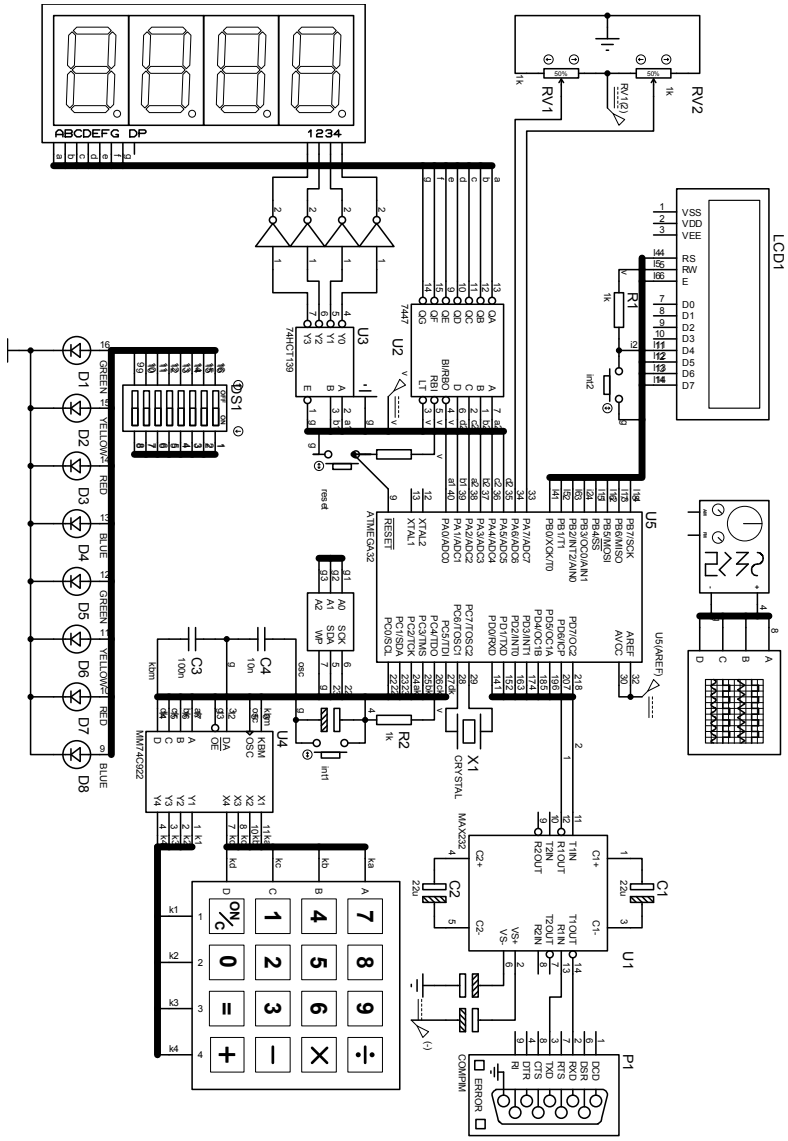
```
		if (!data) return;

		// print data
		for(i=0; i<nBytes; i++)
		{
			LCDsendChar(data[i]);
		}
}

void LCDGotoXY(uint8_t x, uint8_t y)
{
		register uint8_t DDRAMAddr;

		// remap lines into proper order
		switch(y)
		{
		case 0: DDRAMAddr = LCD_LINE0_DDRAMADDR+x;
break;
		case 1: DDRAMAddr = LCD_LINE1_DDRAMADDR+x;
break;
		case 2: DDRAMAddr = LCD_LINE2_DDRAMADDR+x;
break;
		case 3: DDRAMAddr = LCD_LINE3_DDRAMADDR+x;
break;
		default: DDRAMAddr = LCD_LINE0_DDRAMADDR+x;
		}

		// set data address
		LCDsendCommand(1<<LCD_DDRAM | DDRAMAddr);
}



void LCDinit(void)
{
PORTC&=~((1<<datapin)|(1<<clockpin)|(1<<Epin));//set outputs to zerro
DDRC|=(1<<datapin)|(1<<clockpin)|(1<<Epin); //enable output pins
_delay_ms(5); //wait for poverup
sendByteToRegister(0x30);//1
LCDenableCommand();
_delay_ms(1);
LCDdisableCommand();
sendByteToRegister(0x30);//2
LCDenableCommand();
_delay_ms(1);
LCDdisableCommand();
```

```
sendByteToRegister(0x30);//3
LCDenableCommand();
_delay_ms(1);
LCDdisableCommand();
_delay_ms(1);//wait for more
sendByteToRegister(0x38);//enable 8 bit mode dual line
LCDenableCommand();
_delay_ms(1);
LCDdisableCommand();
sendByteToRegister(0x0E);// increment adress counter. Cursor shift
LCDenableCommand();
_delay_ms(1);
LCDdisableCommand();
}

//Print data
void LCDprintData(u32 s,u32 v){
// fprint data
FILE lcd_str = FDEV_SETUP_STREAM(LCDsendChar, NULL,
_FDEV_SETUP_WRITE);
stderr = &lcd_str;
fprintf(stderr,s,v);

}

//Copies string from flash memory to LCD at x y position
//const uint8_t welcomeln1[] PROGMEM="AVR LCD DEMO\0";
//CopyStringtoLCD(welcomeln1, 3, 1);
void CopyStringtoLCD(const uint8_t *FlashLoc, uint8_t x, uint8_t y)
{
                uint8_t i;
                LCDGotoXY(x,y);
                for(i=0;(uint8_t)pgm_read_byte(&FlashLoc[i]);i++)
                {

                LCDsendChar((uint8_t)pgm_read_byte(&FlashLoc[i]));
                }
}
//-------------------------------------------------------------------------------
void LCDshiftLeft(uint8_t n)        //Scrol n of characters Right
{
                for (uint8_t i=0;i<n;i++)
                {
                        LCDsendCommand(0x1E);
                }
}
```

```
void LCDshiftRight(uint8_t n)		//Scrol n of characters Left
{
		for (uint8_t i=0;i<n;i++)
		{
			LCDsendCommand(0x18);
		}
}
void LCDcursorOn(void) //displays LCD cursor
{
		LCDsendCommand(0x0E);
}
void LCDcursorOnBlink(void)		//displays LCD blinking cursor
{
		LCDsendCommand(0x0F);
}
void LCDcursorOFF(void)		//turns OFF cursor
{
		LCDsendCommand(0x0C);
}
void LCDblank(void)		//blanks LCD
{
		LCDsendCommand(0x08);
}
void LCDvisible(void)		//Shows LCD
{
		LCDsendCommand(0x0C);
}
void LCDcursorLeft(uint8_t n)		//Moves cursor by n poisitions left
{
		for (uint8_t i=0;i<n;i++)
		{
			LCDsendCommand(0x10);
		}
}
void LCDcursorRight(uint8_t n)	//Moves cursor by n poisitions left
{
		for (uint8_t i=0;i<n;i++)
		{
			LCDsendCommand(0x14);
		}
}
```



شکل (۱۰-٤۸): شماتیک سخت‌افزاری راه‌اندازی سریال LCD



شکل (۱۰-۴۹): بورد طراحی شده در آزمایشگاه

پیوست توابع کتابخانه‌های نرم‌افزار CodeVision

توابع کتابخانه‌های نرم‌افزار CodeVision

unsigned char isalpha(char c): اگر ورودی این تابع یکی از حروف الفبا باشد، خروجی را به یک می‌برد در غیر این صورت خروجی را به صفر می‌برد.

unsigned char isalnum(char c): اگر ورودی این تابع الفبایی- عددی باشد، خروجی را به ۱ در غیر این صورت به صفر می‌برد.

unsigned char iscntrl(char c): اگر ورودی این تابع یکی از کاراکترهای کنترلی مانند: TAB,ENTER و... باشد خروجی را به یک می‌برد.

unsigned char isdigit(char c): اگر وروری این تابع یکی از کاراکترهای '۰' تا '۹' با شد، خروجی را به ۱ می‌برد.

unsigned char isprint(char c): اگر ورودی این تابع کاراکترهای قابل چاپ باشد، خروجی را به ۱ می‌برد.

unsigned char ispunct(char c): اگر ورودی این تابع یکی از کاراکترهای دستوری مانند "؟" , " ,", " , "!" باشد خروجی را به ۱ می‌برد.

unsigned char isupper(char c): به ازای حروف بزرگ ۱ را به خروجی می‌برد.

unsigned char isspace(char c): اگر ورودی این تابع (فضای خالی) باشد ۱ را به خروجی می‌برد.

unsigned char isxdigit(char c): به ازای ارقام هگزا دسیمال ۱ را به خروجی می‌برد.

char tolower(char c): ورودی را تبدیل به حروف کوچک کرده به خروجی می‌برد.

unsigned char toint(char c): به ازای کاراکترها ی '۰' تا '۹' خروجی را به ۱ می‌برد.

توابع کتابخانه‌های استاندارد

فایل سرآمد این توابع Stdlib.h است.

int atoi(char *str): ورودی این تابع یک رشته است که معادل عدد صحیح از نوع int آن را به خروجی ارسال می‌شود.

long int atol(char *str): ورودی این تابع یک رشته است که معادل عدد صحیح از نوع long int را به خروجی می‌برد.

void itoa(int n, char *str):این تابع عدد صحیح n را به آرایه رشته ای تبدیل می‌کند. و در متغیر str می‌ریزد.

void ltoa(long int n, char *str): این تابع عدد صحیح n از نوع long int را به آرایه رشته‌ای تبدیل می‌کند. و در متغیر str می‌ریزد.
void ftoa(float n, unsigned char decimals, char *str):این تابع متغیر اعشاری n را تا decimal رقم اعشار به رشته تبدیل کرده و در متغیر str می‌ریزد (آرگومان دوم تابع نشان دهنده تعداد رقم اعشار است.
float atof(char *str): ورودی این تابع یک رشته است که معادل اعشاری آن به خروجی برده می‌شود.
int rand (void): این تابع یک عدد تصادفی بین ۰ و ۳۲۷۶۷ تولید می‌کند.
void srand(int seed): این تابع برای تابع rand() تعیین می‌کند که ابتدا از چه عددی برای تولید اعداد تصادفی شروع کند (این تابع باید یک بار قبل از فراخوانی تابع rand() فراخوانی شود.

توابع ریاضی

unsigned char cabs(signed char x):این تابع قدر مطلق متغیر ۱ بایتی x را به خروجی می‌برد.
unsigned int abs(int x): این تابع قدر مطلق متغیر صحیح x را به خروجی می‌برد.
unsigned long labs(long int x): این تابع قدر مطلق متغیر ٤ بایتی x را به خروجی می‌برد.
float fabs(float x): این تابع قدر مطلق متغیر اعشاری x را به خروجی می‌برد.
signed char cmax(signed char a, signed char b): مقدار بیشینه دو بایت a و b را به خروجی می‌برد.
int max(int a, int b):مقدار بیشینه دو عدد صحیح a و b را به خروجی می‌برد.
long int lmax(long int a, long int b): مقدار بیشینه دو عد صحیح a و b که از نوع long int هستند را به خروجی می‌برد.
float fmax(float a,float b): بیشینه دو عدد اعشاری a و b را به خروجی می‌برد.
signed char cmin(signed char a, signed char b): مقدار مینیمم دو بایت a و b را به خروجی می‌برد.
int min(int a, int b): مقدار مینیمم دو عدد صحیح a و b را به خروجی می‌برد.
long int lmin(long int a, long int b): مقدار مینیمم دو عدد صحیح a و b را که از نوع long int هستند را به خروجی می‌برد.
float fmin(float a, float b): مینیمم دو عدد اعشار ی a و b را به خروجی می‌برد.
signed char csign(signed char x): اگر بایت مثبت باشد مقدار ۱، اگر منفی باشد ۱- و اگر بایت ۰ باشدمقدار ۰ را بر می‌گرداند.

signed char sign(int x): تابع علامت دو بایتی
signed char lsign(long int x): تابع علامت چهار بایتی
signed char fsign(float x): تابع علامت برای متغیر اعشاری
float sqrt(float x): این تابع جذر متغیر اعشار ی x را به خروجی می‌برد.
float floor(float x): این تابع کوچک‌ترین جزء صحیح عدد اعشار ی x رابه خروجی می‌برد (برای گرد کردن اعداد اعشاری استفاده می‌شود).
float ceil(float x): این تابع بزگترین جزء صحیح عدد اعشار ی x رابه خروجی می‌برد (برای گرد کردن اعداد اعشاری استفاده می‌شود).
float fmod(float x, float y): این تابع باقیمانده حاصل از تقسیم دو عدد x و y را به خروجی می‌برد.
float modf(float x, float *ipart): قسمت اعشار ی عدد x در آرایه ipart و قسمت صحیح عدد x به خروجی می‌برد.
float ldexp(float x, int expn): این تابع $x * 2^{expn}$ را محاسبه کرده به خروجی می‌برد.
float exp(float x):این تابع مقدار e^x را محاسبه کرده به خروجی می‌برد.
float log(float x):این تابع مقدار lnx() را به خروجی می‌برد.
:float log10(float x):این تابع لگاریتم اعشاری x را در مبنای ۱۰ محاسبه کرده به خروجی می‌برد.
float pow(float x, float y):این تابع مقدار x^y را محاسبه کرده به خروجی می‌برد.
float sin(float x): این تابع مقدار سینوس زاویه x را (بر حسب رادیان) محاسبه کرده به خروجی می‌برد.
float cos(float x): این تابع مقدار کسینوس زاویه x را (بر حسب رادیان) محاسبه کرده به خروجی می‌برد.
float tan(float x): این تابع مقدار تانژانت زاویه x را (بر حسب رادیان) محاسبه کرده به خروجی می‌برد.
توجه: تابع کتانژانت تعریف نشده و باید از ۱/ tan(x) استفاده کنیم.
float sinh(float x):این تابع مقدار سینوس هیپربولیک زاویه x را (بر حسب رادیان) محاسبه کرده و به خروجی می‌برد.
float cosh(float x): این تابع مقدار کسینوس هیپربولیک زاویه x را (بر حسب رادیان) محاسبه کرده و به خروجی می‌برد.
float tanh(float x): این تابع مقدار تانژانت هیپربولیک زاویه x را (بر حسب رادیان) محاسبه کرده و به خروجی می‌برد.

float asin(float x): مقدار آرک‌سینوس x را بر حسب رادیان در فاصله [$-\pi/2,\pi/2$] محاسبه کرده به خروجی می‌برد (توجه: x باید بین [-1,1] با شد.).
float acos(float x): مقدار آرک‌کسینوس x را بر حسب رادیان در فاصله [$0,\pi$] محاسبه کرده به خروجی می‌برد (توجه :x باید بین [-1,1] با شد).
float atan(float x): مقدار آرک‌تانژانت x را بر حسب رادیان در فاصله [$\pi/2,\pi/2$] محاسبه کرده و به خروجی می‌برد.

توابع رشته ای

char *strcat(char *str1, char *str2): این تابع انتهای str2 را به انتهای str1 می چسباند.
char *strcatf(char *str1, char flash *str2): این تابع رشته str2 را که در flash می باشد، به انتهای رشته str1 می‌چسباند.
char *strncat(char *str1, char *str2, unsigned char n): این تابع حداکثر n کاراکتر از رشته str2 را به انتهای str1 می‌چسباند.
char *strncatf(char *str1, char flash *str2, unsigned char n): این تابع حداکثر تعداد n کاراکتر از رشته s2 که در حافظه flash می‌باشد را به انتهای رشته s1 اضافه می‌کند.
signed char strcmp(char *str1, char *str2) : اگر str1<str2 باشد خروجی یک عدد منفی است (0>)، اگر str1>str2 باشد خروجی یک عدد مثبت است (0<) و اگر str1=str2 باشد خروجی 0 است.
signed char strcmpf(char *str1, char flash *str2): تنها تفاوت آن با تابع قبل در حافظه flash بودن تابع str است.
signed char strncmp(char *str1, char *str2, unsigned char n): حداکثر n کاراکتر از str1 را با str2 مقایسه می‌کند خروجی مانند تابع قبل است.
signed char strncmpf(char *str1, char flash *str2, unsigned char n): همانند تابع قبلی است با این تفاوت که رشته str2 در flash قرار دارد.
char *strcpy(char *dest, char *src): این تابع رشته src را در رشته dest کپی می‌کند.
char *strcpyf(char *dest, char flash *src): این تابع رشته src که در حافظه flash قرار دارد را در رشته dest کپی می‌کند.
char *strncpy(char *dest, char *src, unsigned char n): حد اکثر n کاراکتر از رشته src را در رشته dest کپی می‌کند.

unsigned char strlen(char *str) (in the range 0..255)
unsigned int strlen(char *str) (in the range 0..65535)
هر دو تابع فوق طول رشته str را به خروجی می‌برند.

void *memcpy(void *dest,void *src, unsigned char n): این تابع n کاراکتر از رشته str را در رشته dest کپی می‌کند. توجه: این تابع در میکروکنترلر های tiny استفاده می‌شود.

void *memcpy(void *dest,void *src, unsigned int n): همانند تابع فوق است با این تفاوت که در حافظه‌های small کاربرد دارد.

void *memcpyf(void *dest,void flash *src, unsigned char n): همانند تابع فوق است با این تفاوت که تابع src در حافظه flash قرار دارد و در میکرو کنترلر های tiny کاربرد دارد.

توابع تبدیل BCD

unsigned char bcd2bin(unsigned char n): این تابع برای تبدیل یک عدد BCD مثبت به باینری استفاده می‌شود.

unsigned char bin2bcd(unsigned char n): این تابع برای تبدیل یک عدد باینری به BCD به استفاده می‌شود که n باید بین ۰ تا ۹۹ باشد.

توابع Gray

unsigned char gray2binc(unsigned char n)
unsigned char gray2bin(unsigned int n)
unsigned char gray2binl(unsigned long n)
کلیه توابع فوق برای تبدیل یک کد گری به یک عدد باینری استفاده می‌شود.

unsigned char bin2grayc(unsigned char n)
unsigned char bin2gray(unsigned int n)
unsigned char bin2grayl(unsigned long n)
کلیه توابع فوق برای تبدیل یک عدد باینری به یک کدگری استفاده می‌شود.

مراجع

۱- برگه‌های اطلاعاتی میکروکنترلرهای AVR

۲- میکروکنترلرهای AVR و کاربردهای آن‌ها، تالیف: امیر ره‌افروز، انتشارات نص.
۳- میکروکنترلرهای AVR سری MEGA، مولفان: سعید شجاعی و نادر مهرا.
٤- مرجع کامل میکروکنترلرهای AVR، تالیف: محمد مهدی پرتوی‌فر، انتشارات نص
۵- میکروکنترلرهای AVR، تالیف: مهندس علی کاهه، انتشارات نص
٦- وب سایت www.atmel.com
۷- وب سایت استاد بهزاد خزاما www.khazama.com
۸- وب سایت مهندس حامد سقایی www.saghaei.com
۹- وب سایت مهندس حسین لاچینی www.hlachini.com
۱۰-وب سایت www.kavirelectronic.ir
۱۱-وب سایت www.yazdkit.com
۱۲-وب سایت www.ir-micro.com
۱۳-وب سایت www.iranled.com
۱٤-وب سایت www.eca.ir
۱۵-وب سایت www.edaboard.com