



دستور کار آزمایشگاه ریزپردازنده AVR

تهیه و تنظیم:

سارا السادات زمانی

مهر ۱۳۹۹

فهرست مطالب

فصل اول : آشنایی با برد آزمایشگاه و میکروکنترلرهای AVR	۲
۱-۱- مقدمه	۲
۲-۱- میکروکنترلرهای AVR	۲
۳-۱- میکروکنترلر Atmega16A	۳
۴-۱- تجهیزات آزمایشگاه ریزپردازنده	۴
۱-۴-۱- برد آموزشی میکروکنترلر AVR	۴
۲-۴-۱- پروگرامر USB میکروکنترلر AVR	۶
۳-۴-۱- نحوه شروع کار با برد آموزشی	۶
فصل دوم : آشنایی با نرم افزار Codevision AVR	۷
۱-۲- مقدمه	۷
۲-۲- ساخت فایل جدید در نرم افزار CodeVision AVR	۷
۳-۲- استفاده از محیط wizard	۹
۴-۲- نوشتن برنامه به زبان اسمبلی	۱۳
۵-۲- روش برنامه ریزی ریزپردازنده	۱۳
فصل سوم : آزمایش ها	۱۷
آزمایش شماره ۱: راه اندازی LED ها	۱۷
آزمایش شماره ۲: تولید صوت توسط Buzzer	۱۹
آزمایش شماره ۳: راه اندازی نمایشگر 7-Segment	۲۰
آزمایش شماره ۴: راه اندازی LCD کاراکتری	۲۲
آزمایش شماره ۵: راه اندازی موتور پله ای	۲۴
آزمایش شماره ۶: راه اندازی صفحه کلید ماتریسی	۲۸
آزمایش شماره ۷: راه اندازی حسگر دما	۳۳
آزمایش شماره ۸: راه اندازی USART	۳۴
منابع	۳۹

فصل اول

آشنایی با برد آزمایشگاه و میکروکنترلرهای AVR

۱-۱- مقدمه

ریزپردازنده، واحد پردازش مرکزی یا مغز رایانه است. این بخش شامل مدار الکترونیکی بسیار گسترده و پیچیده‌ای است که دستورات برنامه‌های ذخیره شده را انجام می‌دهد. با توجه به حرکت جوامع بشری به سوی هر چه کوچک‌تر کردن وسایل کاربردی، طراحان الکترونیک به تبعیت از این قانون، سعی در کوچک کردن مدار کنترلی یک پروسه و کاهش هزینه‌های مربوطه نمودند؛ که این امر موجب پیدایش میکروکنترلرها به عنوان وسیله‌ای که دارای حافظه، CPU، پورت‌های ورودی و خروجی و ... در یک چیپ گردید. اولین میکروکنترلر در سال ۱۹۷۱ توسط شرکت نام آشنای Intel ساخته شد. این شرکت، اولین میکروکنترلر کاربردی خود را در سال ۱۹۸۰ با نام ۸۰۸۰ روانه بازار کرد. با ورود خانواده میکروکنترلر ۸۰۵۱ از شرکت Intel، تحولی عظیم در این صنعت رخ داد و میکروکنترلرها تنها یک پردازشگر نبودند و عملیات محاسبه و منطق تنها بخشی از این تراشه بود. امکاناتی نظیر حافظه‌ها، تایمرها و ارتباطات به این تراشه افزوده شد و این تراشه مانند یک کامپیوتر کوچک به بازار عرضه شد. طولی نکشید که شرکت‌هایی نظیر MicroChip و Atmel سری جدیدتری از میکروکنترلرها را عرضه کردند. میکروکنترلرهای ۸ بیتی AVR ساخت شرکت Atmel، بدلیل امکانات متمایزشان، در دسته پرکاربردترین نوع میکروکنترلرهای موجود در دنیا قرار گرفتند. ما امروزه شاهد معماری‌های مختلفی از میکروکنترلرها هستیم که مهم‌ترین آن‌ها عبارتند از: AVR، PIC، ARM و 8051. تفاوت میکروکنترلرهای این ۴ خانواده، علاوه بر تکنولوژی ساخت، در برنامه نویسی مورد نیاز و نحوه پروگرام کردن آن‌ها است.

۱-۲- میکروکنترلرهای AVR

AVRها، میکروکنترلرهایی ۸ بیتی از نوع Cmos با توان مصرفی پایین هستند که بر اساس ساختار پیشرفته RISC، با معماری Harvard ساخته شده‌اند. در میکروکنترلرهای AVR دستورات تنها در یک پالس ساعت اجرا می‌شوند. بنابراین این میکروکنترلرها می‌توانند به ازای هر یک مگاهرتز، یک مگا دستور را در ثانیه اجرا کنند. در نتیجه برنامه از لحاظ سرعت پردازش و مصرف توان بهینه می‌شود. این میکروکنترلرها دارای ۳۲ رجیستر همه منظوره و مجموعه دستورات قدرتمندی هستند، که تمام این ۳۲ رجیستر مستقیماً به ALU متصل شده‌اند. میکروکنترلرهای AVR با دو معماری ۸ بیتی و ۱۶ بیتی ساخته می‌شوند که در اینجا به شرح کارکرد مدل ۸ بیتی می‌پردازیم. میکروکنترلرهای ۸ بیتی AVR به سه دسته تقسیم می‌شوند:

Tiny AVR ✓

Mega AVR ✓

Xmega AVR ✓

تفاوت بین این سه نوع به امکانات موجود در آن‌ها مربوط می‌شود. Tiny AVR ها غالباً تراشه‌هایی با تعداد پین و مجموعه دستورات کمتری نسبت به Mega AVR ها هستند، به عبارتی از لحاظ پیچیدگی حداقل امکانات را

دارند. Xmega AVR ها شامل حداکثر امکانات هستند، و نسبت به Mega AVR ها، تعداد پین ها و دستورات بیشتری دارند. ما در آزمایشگاه ریزپردازنده با میکروکنترلر Atmega16A کار خواهیم کرد. بنابراین در ادامه توضیح مختصری از این میکروکنترلر ارائه می شود.

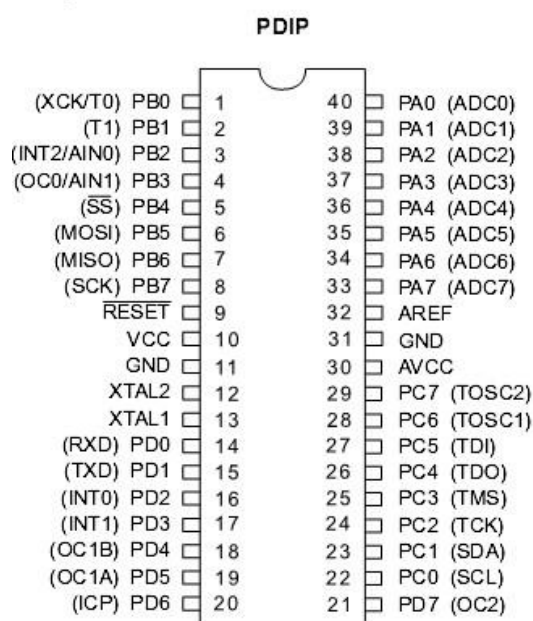
۱-۳- میکروکنترلر Atmega16A

میکروکنترلر Atmega16 یک میکروکنترلر پر کاربرد در بازار است و در پروژه های زیادی استفاده می شود. بیشترین استفاده این میکروکنترلر در پکیج PDIP است که همانند Atmega32 دارای ۴۰ پین و ۳۲ پین ورودی و خروجی دارد. این میکروکنترلر AVR در پکیج ۴۴ پایه TQFP نیز برای مصارف SMD یافت می شود. میکروکنترلر Atmega16A یکی از سری های Atmega16 می باشد. پسوند A دارای این معنی است که این میکرو بر خلاف Atmega16 که از ولتاژ ۴,۵ تا ۵,۵ ولت می تواند کار نماید، همانند سری L می تواند با ولتاژ ۲,۷۵ تا ۵,۵ ولت کار کند. اما بر خلاف سری L که دارای ماکزیمم فرکانس گارانتی شده ۸ مگاهرتز است، Atmega16 همانند Atmega16A می تواند دارای منبع کلاک تا سرعت ۱۶ MHz باشد.

ویژگی های میکروکنترلر Atmega16A :

- پایداری بالا
- مصرف توان کم
- میکروکنترلر ۸ بیتی Atmel
- معماری RISC پیشرفته ، ۱۳۱ دستورالعمل قدرتمند ، اجرای اغلب دستورالعمل ها در یک کلاک ، ۳۲ رجیستر ۸ بیتی با کاربرد عمومی ، بیش از ۱۶ میلیون دستورالعمل بر ثانیه (MIPS) با کلاک ۱۶ مگاهرتز (MHz)
- ۱۶ کیلوبایت حافظه فلش قابل برنامه ریزی
- ۵۱۲ بایت EEPROM
- ۱ کیلوبایت SRAM
- قابلیت برنامه ریزی حافظه فلش تا ۱۰,۰۰۰ بار و حافظه ی EEPROM تا ۱۰۰,۰۰۰ بار
- ماندگاری برنامه تا ۲۰ سال در دمای ۸۵ درجه و ۱۰۰ سال در دمای ۲۵ درجه سانتی گراد
- دارای قفل برنامه برای حفاظت از نرم افزار
- رابط JTAG مطابق استاندارد IEEE 1149.1
- دارای ۲ تایمر ۸ بیتی
- دارای یک تایمر ۱۶ بیتی
- دارای RTC با اسیلاتور مجزا
- ۴ کانال PWM
- ۸ کانال ADC ده بیتی
- رابط سریال TWO WIRE یا TWI
- USART

- رابط سریال SPI در حالت Master/Slave
 - دارای تایمر دیده بان با اسلاتور مجزای داخلی
 - مقایسه گر آنالوگ داخلی
 - دارای اسلاتور RC کالیبره شده داخلی
 - ۳۲ پورت ورودی و خروجی
 - ولتاژ تغذیه ۲,۷۵ تا ۵,۵ ولت
 - پشتیبانی از فرکانس ۰ تا ۱۶ مگاهرتز
- همانطور که در شکل زیر دیده می‌شود، میکروکنترلر Atmega16A دارای ۴ پورت A, B, C, D می‌باشد که علاوه بر اینکه به عنوان ورودی-خروجی مورد استفاده قرار می‌گیرند، کاربردهای جانبی دیگری نیز دارند. توجه داشته باشید که بر روی برد موجود در این آزمایشگاه، فقط پورت‌های A, B و C قابل دسترسی هستند.



۴-۱- تجهیزات آزمایشگاه ریزپردازنده

هدف از برگزاری این آزمایشگاه، آشنایی دانشجویان مقاطع کارشناسی مهندسی برق و کامپیوتر با سیستم‌های مبتنی بر ریزپردازنده و میکروکنترلر است. در این آزمایشگاه، دانشجویان با روش‌های ارتباط دهی میکروکنترلرها به وسایل ورودی/خروجی و برنامه ریزی آن‌ها در سطوح مقدماتی و پیشرفته آشنا خواهند شد. در انتهای ترم انتظار می‌رود که دانشجویان فوق الذکر از عهده طراحی سیستم‌های مبتنی بر میکروکنترلرهای AVR برآیند.

۱-۴-۱- برد آموزشی میکروکنترلر AVR

در این برد آموزشی سعی شده است تا اکثر تجهیزاتی که در کار با میکروهای AVR نیاز هست، گنجانده شود. یکی از مهمترین مزایای این برد، امکان شبیه سازی کلیه آزمایشات بر روی پورت‌های A, B, C و D است که توسط کاربر تعریف می‌شود. برای انجام آزمایش‌ها، توسط کابل‌های ارتباطی که در بسته بندی موجود می‌باشد، ارتباط بین قسمت‌های مختلف برد آموزشی و پورت‌ها برقرار می‌شود.

فصل دوم

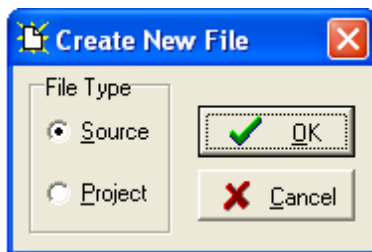
آشنایی با نرم افزار CodeVision AVR

۲-۱- مقدمه

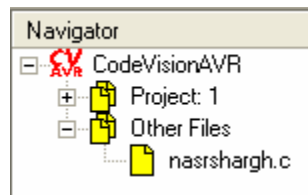
به منظور استفاده از میکروکنترلرهای AVR می‌بایست به طریقی برنامه‌های مورد نظر را توسعه و بر روی آن پیاده سازی نمود. این عمل از طریق نرم افزارهای مخصوصی که حاوی یک کامپایلر هستند، صورت می‌گیرد. در بازار انواع مختلفی از کامپایلرهای AVR معرفی شده‌اند، که از آن جمله می‌توان به CodeVision AVR، BASCOM و WIN AVR اشاره نمود. هر کدام از نرم افزارهای ارائه شده دارای قابلیت‌های مخصوص به خود است؛ اما از میان آن‌ها، نرم افزار CodeVision AVR از اهمیت و اعتبار بیشتری برخوردار است. این نرم افزار، امکانات قابل توجهی را به دانشجویان و کاربران صنعتی ارائه می‌دهد. CodeVision AVR توسط شرکت HP Info. Tech. معرفی شده است، که دارای قابلیت‌هایی نظیر ویرایشگر کد C و اسمبلی، کامپایلر، اسمبلر، پروگرامر و ... است. میکروکنترلر AVR که در این آزمایشگاه استفاده می‌شود، از نوع ATmega16A است.

۲-۲- ساخت فایل جدید در نرم افزار CodeVision AVR

پس از نصب نرم افزار، پوشه‌ای با نام دلخواه ایجاد کرده و از این پس فایل‌ها و پروژه‌های مورد نظر خود را در این پوشه ذخیره نمایید. با کلیک کردن بر روی آیکون نرم افزار، وارد محیط این نرم افزار می‌شوید. برای آغاز کار با نرم افزار ابتدا باید یک فایل جدید را ایجاد کرده و سپس کد خود را در آن وارد کنید. بنابراین به منظور ساخت فایل جدید از منوی File گزینه new و یا از نوار ابزار انتخاب کنید. پنجره زیر باز خواهد شد (شکل ۱-۲). همانطور که مشاهده می‌کنید، برای ایجاد فایل جدید دو انتخاب دارید: Project و Source. در ادامه این دو روش بررسی شده است.



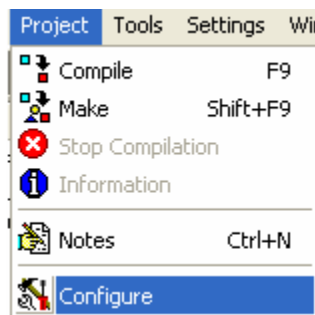
شکل ۱-۲



شکل ۲-۲

(۱) محیط Source

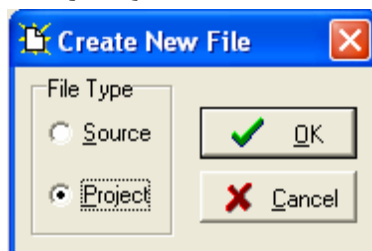
با انتخاب گزینه Source وارد محیط برنامه نویسی می‌شوید. پس از نوشتن برنامه موردنظر، با پسوند C ذخیره نمائید. فایل ذخیره شده، در قسمت Navigator جزء Other File قرار می‌گیرد. از منوی Project گزینه Configure را انتخاب کنید. شما می‌توانید همین گزینه را از نوار ابزار نیز انتخاب کنید. سپس با کلیک بر روی Add در پنجره Add File To Project و انتخاب فایل مورد نظر، فایل شما به لیست Project اضافه می‌شود.



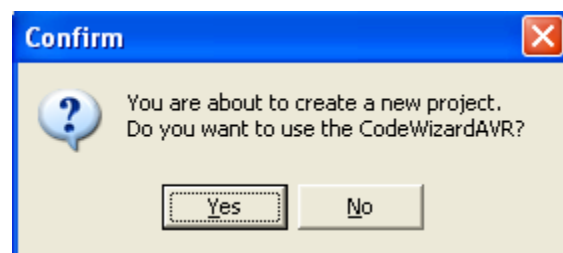
شکل ۳-۲

(۲) محیط Project

در قسمت Create New File با انتخاب گزینه Project، پنجره شکل ۲-۵ باز خواهد شد، با انتخاب Yes وارد محیط Wizard شده و با انتخاب No فقط می‌توانید از محیط Project استفاده نمائید.



شکل ۴-۲



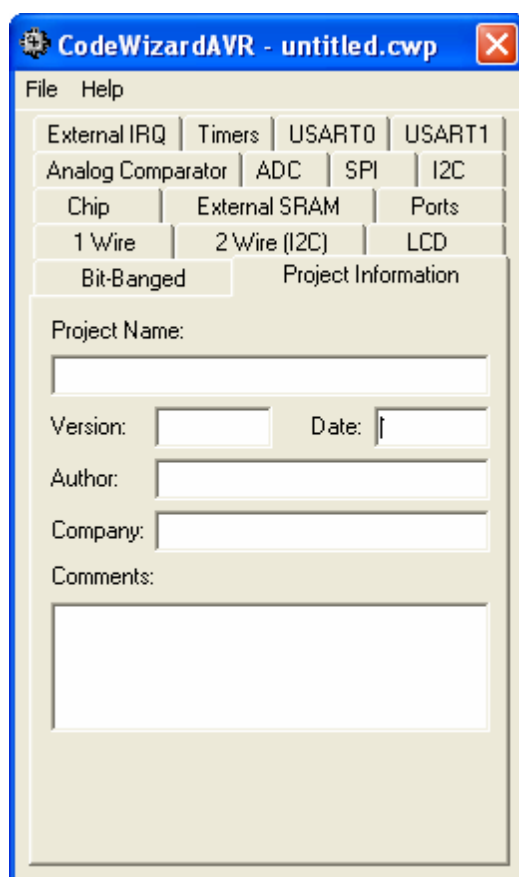
شکل ۵-۲

۳-۲- استفاده از محیط wizard

در نرم افزار CodeVision AVR، با ارائه قابلیت به نام Wizard توانایی تولید کد برای راه اندازی و پیکربندی اجزاء جانبی ریزپردازنده AVR در آن قرار داده شده است. پس از انتخاب Yes پنجره زیر باز می شود. با انتخاب از نوار ابزار نیز می توان پنجره Wizard را انتخاب کرد. این پنجره از برگه های مختلفی تشکیل شده است. با توجه به نوع پروژه، برگه های مورد نیاز را انتخاب و اطلاعات مورد نظر را وارد کنید.

✓ برگه Project Information

در این برگه می توانید مشخصات پروژه نظیر نام، نسخه، تاریخ و ... را وارد نمایید. (شکل ۶-۷)



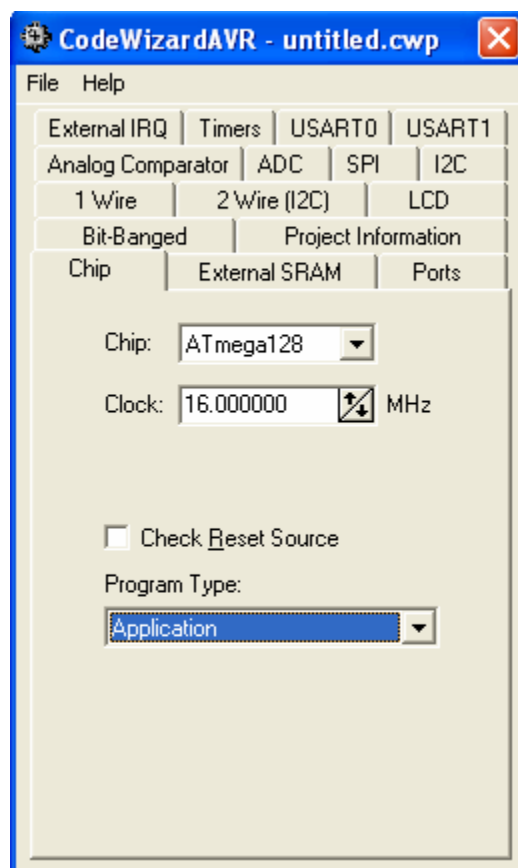
The screenshot shows the 'CodeWizardAVR - untitled.cwp' window with the 'Project Information' tab selected. The window has a menu bar with 'File' and 'Help'. Below the menu bar is a grid of tabs: External IRQ, Timers, USART0, USART1, Analog Comparator, ADC, SPI, I2C, Chip, External SRAM, Ports, 1 Wire, 2 Wire (I2C), LCD, Bit-Banged, and Project Information. The 'Project Information' tab is active, showing fields for Project Name, Version, Date, Author, Company, and Comments.

CodeWizardAVR - untitled.cwp			
File Help			
External IRQ	Timers	USART0	USART1
Analog Comparator	ADC	SPI	I2C
Chip	External SRAM	Ports	
1 Wire	2 Wire (I2C)	LCD	
Bit-Banged	Project Information		
Project Name: <input type="text"/>			
Version: <input type="text"/>		Date: <input type="text"/>	
Author: <input type="text"/>			
Company: <input type="text"/>			
Comments: <input type="text"/>			

شکل ۶-۲

✓ برگه Chip

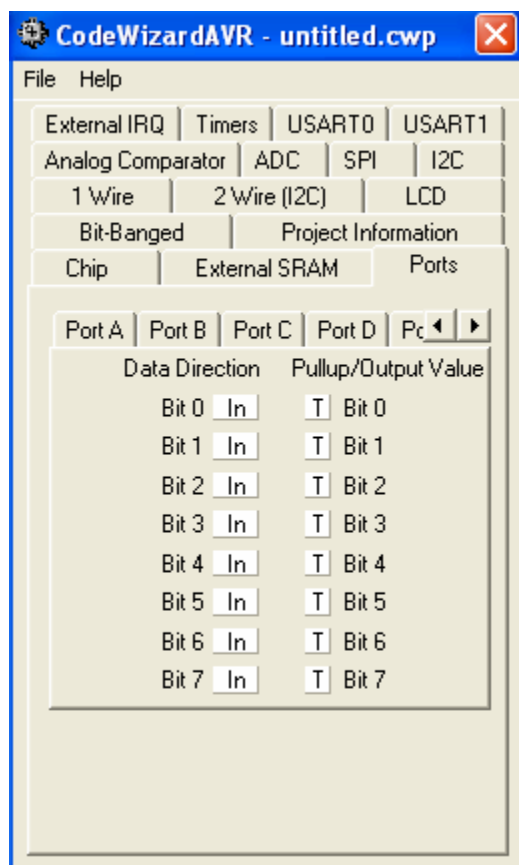
در این برگه، تراشه مورد استفاده در سخت افزار و فرکانس کاری مرتبط با آن را انتخاب کنید. (شکل ۷-۲)



شکل ۷-۲

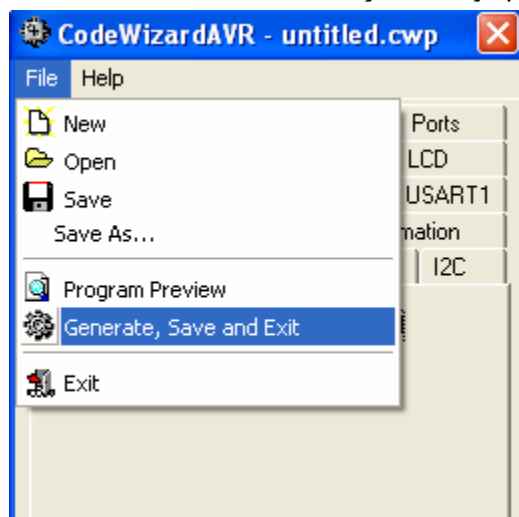
✓ برگه Ports

در این برگه مشخصات پورتهای تراشه را می توان مشخص نمود. پایه ورودی با In و پایه خروجی با Out مشخص می شود. برای تغییر حالت یک بار بر روی آن کلیک کنید. توجه داشته باشید زمانی که پورتی را به عنوان ورودی تعریف، مقدار آن صفر و زمانی که به عنوان خروجی تعریف شود، مقدار آن یک در نظر گرفته می شود.

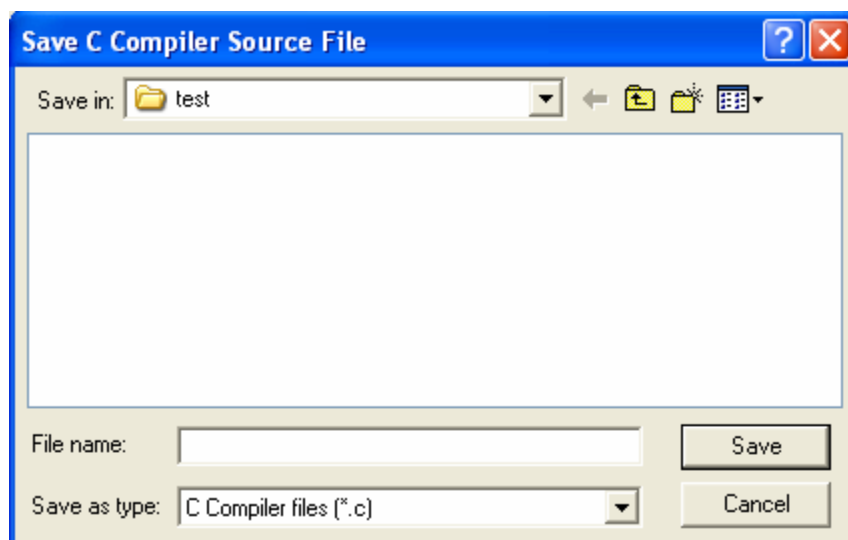


شکل ۸-۲

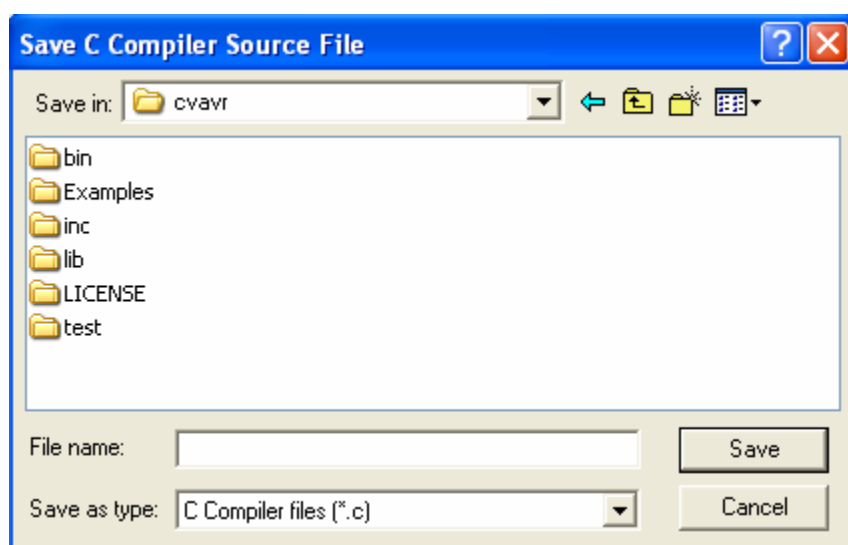
با توجه به نوع پروژه، چنانچه لازم است برگه های دیگر را نیز فعال نمائید. پس از این که اطلاعات لازم در برگه های مورد نظر وارد شدند، از منوی File، گزینه Generate, Save and Exit را انتخاب کنید. سپس پنجره زیر (شکل ۹-۲) باز خواهد شد. پوشه test را که به منظور ذخیره فایل ساخته اید، انتخاب و نام فایل مورد نظر را وارد و با پسوند ذخیره نمائید.



شکل ۹-۲

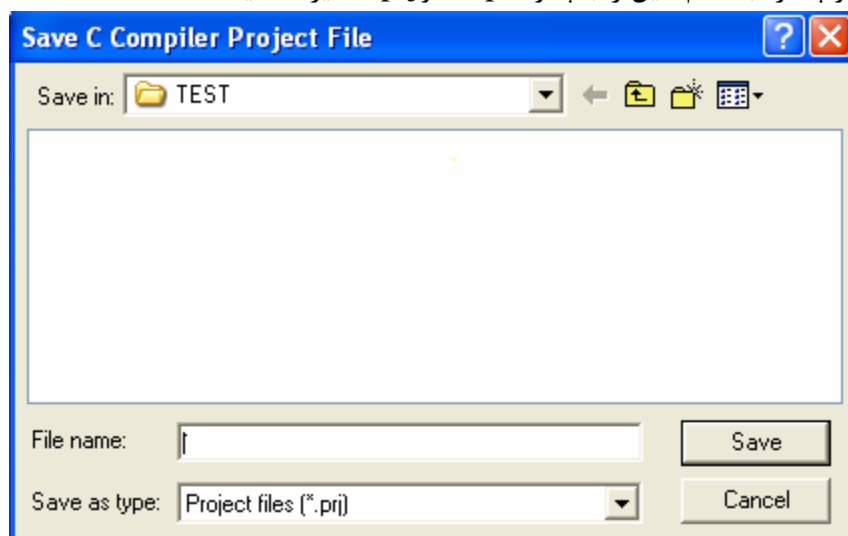


شکل ۱۰-۲

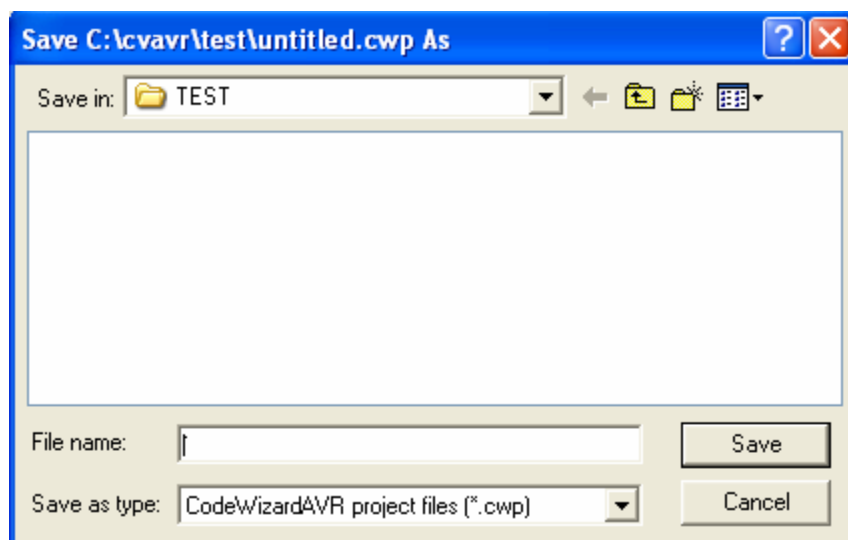


شکل ۱۱-۲

در این پنجره و پنجره بعد نام فایل را با پسوند .cwp و .prj ذخیره نمایید.



شکل ۱۲-۲



شکل ۲-۱۳

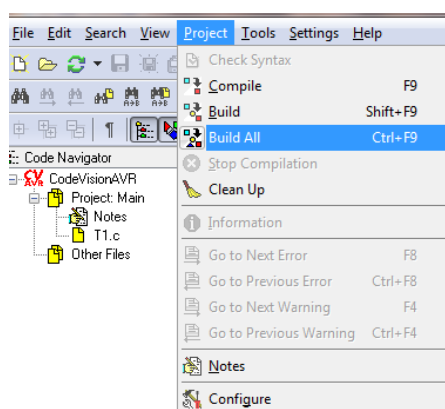
پس از ذخیره فایل، محیط لازم برای نوشتن برنامه آماده شده است. شما می‌توانید برنامه خود را به کد تولید شده اضافه کنید.

۲-۴- نوشتن برنامه به زبان اسمبلی

یکی از قابلیت‌های این نرم‌افزار نوشتن برنامه به زبان اسمبلی است و شما می‌توانید در هر نقطه از برنامه C کد اسمبلی بنویسید. برای این منظور تنها کافی است کد مورد نظر را بین دو دستور `#asm` و `#endasm` قرار دهید.

۲-۵- روش کامپایل ریزپردازنده

آخرین فرآیندی که باید پیش از برنامه‌ریزی کردن ریزپردازنده انجام داد، کامپایل و بررسی کد نوشته شده از نقطه نظر خطا است. برای این منظور از منوی **Project** گزینه **Build all** را انتخاب نمایید یا می‌توانید از نوار ابزار انتخاب کنید (شکل ۲-۱۴).



شکل ۲-۱۴

فصل سوم

آزمایش‌ها

آزمایش شماره ۱ راه‌اندازی LEDها

هدف از این آزمایش، راه‌اندازی LEDهای برد آموزشی و آشنایی با نحوه برنامه‌نویسی آن‌هاست.

در کنار برد آموزشی میکروکنترلر، چند سری کابل نواری که به دو طرف آن سوکت‌های IDC وصل شده، قرار دارد. یک سر این کابل را به پورت A و سر دیگر آن را به پورت LEDها متصل نمایید.

برنامه روشن و خاموش شدن یک LED:

```
#include <mega16.h>
# include <delay.h>
void main() {
DDRA=0xff;

//پورت A به سمت LEDها دیتا می‌فرستد، پس باید به صورت خروجی تنظیم شود.
PORTA=0x00;

//مقدار اولیه ی پورت را صفر قرار داده‌می‌شود. بدین ترتیب در شروع برنامه، تمام LEDها خاموش هستند.
while(1){
PORTA.0=~PORTA.0;
delay_ms(100);

// نقیض مقدار اولیه پورت (صفر) در پورت قرار می‌گیرد؛ بعد از ۱۰۰ میلی ثانیه تاخیر، دوباره مقدار پورت نقیض می‌شود.
}}
```

سوالات :

۱. برنامه را طوری تغییر دهید که بجای LED0، یک LED دیگری چشمک بزند.
۲. برنامه را طوری تغییر دهید که زمان تاخیر کمتر از ۱۰ میلی ثانیه باشد. چه نتیجه‌ای مشاهده می‌کنید؟
۳. برنامه را طوری تغییر دهید که زمان روشن بودن یک ثانیه و زمان خاموش بودن ۲ ثانیه باشد.
۴. برنامه را طوری تغییر دهید که LED روشن به ترتیب بر روی تمام LEDها در جهت راست چرخش پیدا کند.
۵. برنامه قسمت قبل را طوری تغییر دهید که LEDها در جهت معکوس چرخش پیدا کنند.
۶. برنامه قسمت قبل را طوری تغییر دهید که بجای یک LED، دو LED کنارهم چرخش پیدا کنند.
۷. برنامه قسمت قبل را طوری تغییر دهید که در هر لحظه دو LED به صورت یک در میان روشن باشد و چرخش پیدا کنند.
۸. برنامه چرخش LEDها را با تاخیرهای متفاوت انجام دهید؛ و مشاهدات خود را بیان نمایید.

آزمایش شماره ۲

تولید صوت توسط Buzzer

هدف از این آزمایش، راهاندازی Buzzer برد آموزشی و آشنایی با نحوه برنامه‌نویسی جهت تولید صوت است. در این آزمایش، یک سر کابل ارتباطی را به پورت B و سر دیگر آن را به پورت 7Segment متصل نمایید. با این کار، Buzzer توسط یک درایور به PortB.4 وصل خواهد شد.

برنامه راهاندازی Buzzer:

```
#include <mega16.h>
# include <delay.h>
void main() {
    DDRB=0x10;
    PORTB=0x10;
    while(1){
        PORTB.4=~PORTB.4;
        delay_ms(100);
    }
}
```

سوالات :

۱. برنامه فوق را با تاخیرهای متفاوت انجام دهید؛ و مشاهدات خود را بیان نمایید.
۲. برنامه فوق را با پورت‌های دیگر برد آموزشی امتحان کنید؛ و نتیجه را بیان کنید.
۳. برنامه فوق را با پین‌های دیگری غیر از پین شماره ۴ امتحان کنید؛ و نتیجه را بیان کنید.

آزمایش شماره ۳

راهاندازی نمایشگر 7-Segment

هدف از این آزمایش، راهاندازی نمایشگر 7-Segment برد آموزشی و آشنایی با نحوه برنامه‌نویسی آن است.

برای انجام آزمایش، یک سر کابل را به پورت A و سر دیگر آن را به پورت 7-Segment متصل نمایید. در این آزمایش، نمایشگر 7-Segment به صورت مالتی‌پلکسری راهاندازی می‌شود، به این صورت که پین‌های ۵، ۶ و ۷، از پورت A، به ترتیب 7-Segment های S1، S2 و S3 را فعال می‌کنند. در این حالت، توسط ورودی‌های A، B، C و D از دیکدر ۷۴۴۷، داده BCD از پایه‌های PORTA.0 تا PORTA.3 به سگمنت‌ها اعمال می‌شود.

برنامه شمارش از ۰ تا ۹۹ بر روی 7-Segment:

```
#include <mega16.h>
#include <delay.h>
int Yekan,Dahgan,num,i;
void main(void){
char seg[10]={0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09};
PORTA=0x00;
DDRA=0xFF;
num=0;
while (1) {
Yekan=num%10;
Dahgan=num/10;
for(i=0;i<20;i++) {
PORTA=seg[Yekan];
PORTA.5=1;
delay_ms(100);
PORTA=seg[Dahgan];
```



```
PORTA.6=1;
delay_ms(100);
}

num++;
if(num==100)
num=0;
};
}
```

سوالات :

۱. برنامه را طوری تغییر دهید که بجای سگمنت‌های ۱ و ۲، اعداد روی سگمنت‌های ۲ و ۳ نمایش داده شود.
۲. برنامه را طوری تغییر دهید که زمان تاخیر چند ده برابر شود؛ چه نتیجه‌ای مشاهده می‌کنید؟
۳. برنامه را طوری تغییر دهید که بر روی هر سه سگمت شمارش اعداد از ۰ تا ۹۹۹ نمایش داده شود.

آزمایش شماره ۴

راه اندازی LCD کاراکتری

هدف از این آزمایش، راه اندازی LCD کاراکتری برد آموزشی و آشنایی با نحوه برنامه نویسی آن است.

LCD کاراکتری روی برد آموزشی در مود چهار بیتی به میکروکنترلر وصل شده است. این LCD، دارای ۱۶ ستون و ۲ سطر است. با فرض اینکه در این آزمایش، پورت LCD به پورت B متصل باشد؛ پیکربندی آن در بخش تنظیمات LCD، هنگام ساختن پروژه از طریق wizard، به صورت زیر خواهد بود.

PB.0=RS

PB.1=E

PB.2=D4

PB.3=D5

PB.4=D6

PB.5=D7

یکی از پایه های LCD مربوط به نور زمینه (Back Light) است که به پایه ۶ پورت وصل می شود.

برنامه نمایش یک متن روی LCD:

```
#include <mega16.h>
```

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

```
void main(void) {
```

```
PORTB=0x00;
```

```
DDRB=0xFF;
```

```
PORTB.6=1;
```

```
Lcd_init(16);
```

```
while (1) {
```

```
lcd_clear();
```

```
lcd_gotoxy(0,0);
```

```
lcd_putsf("Hello");  
delay_ms(100);  
lcd_gotoxy(0,1);  
lcd_putsf("UUT university");  
delay_ms(100);  
}}
```

سوالات :

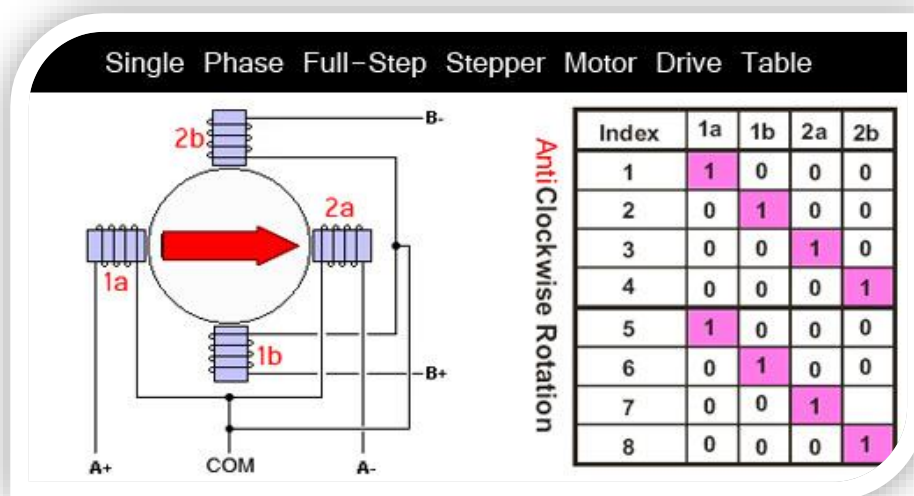
۱. برنامه را طوری تغییر دهید که متن پیام‌های هر خط، در وسط خط نمایش داده شود.
۲. برنامه را طوری تغییر دهید که متن پیام‌ها بر روی نمایشگر بصورت چشمک زن نمایش داده شود.
۳. برنامه را طوری تغییر دهید که زمان روشن بودن یک ثانیه و زمان خاموش بودن ۲ ثانیه باشد.
۴. برنامه را طوری تغییر دهید که متن روی نمایشگر LCD در هر ثانیه به سمت چپ چرخش یابد.
۵. برنامه قسمت قبل را طوری تغییر دهید که متن روی نمایشگر LCD در جهت معکوس چرخش پیدا کنند.

آزمایش شماره ۵

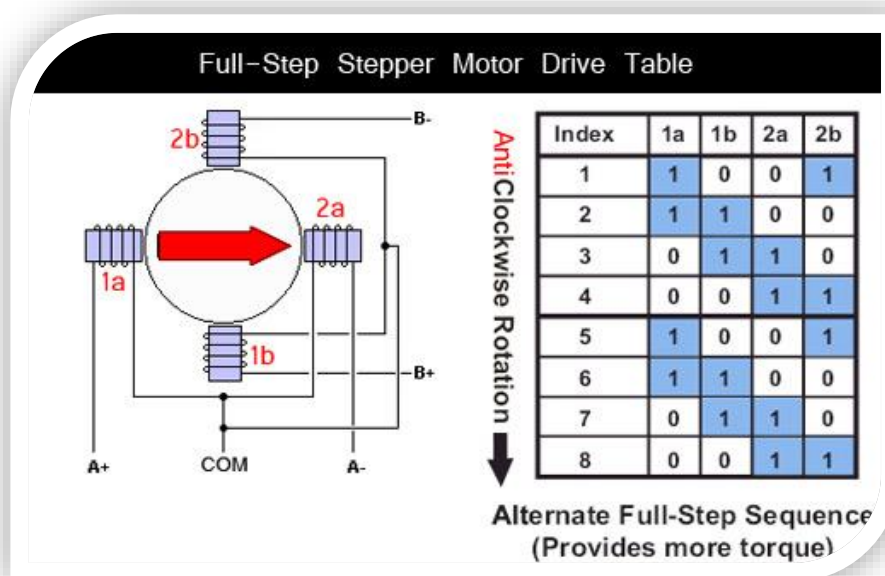
راهاندازی موتورپله‌ای

هدف از این آزمایش، راهاندازی موتورپله‌ای (Step Motor) برد آموزشی و آشنایی با نحوه برنامه‌نویسی آن است. موتورپله‌ای (Step Motor) نوعی موتور مشابه موتورهای DC است که حرکت دورانی تولید می‌کند. با این تفاوت که موتورپله‌ای دارای حرکت دقیق و حساب شده‌تری است. این موتورها به صورت درجه‌ای دوران می‌کنند و با درجه‌های مختلف در بازار موجود هستند. موتورپله‌ای‌های موجود در بازار معمولاً در دو نوع ۵ یا ۶ سیم یافت می‌شود. کاربرد اصلی این موتورها در کنترل موقعیت است. این موتورها ساختار کنترلی ساده‌ای دارند. لذا در ساخت ربات کاربرد زیادی دارند.

در این آزمایش، یک موتورپله‌ای تک‌قطبی (Unipolar) ۵ سیمه بوسیله میکروکنترلر AVR به زبان C و در محیط کدویژن راهاندازی می‌شود. موتور پله‌ای ۵ سیمه و موتور پله‌ای ۶ سیمه در حقیقت ساختمان داخلی یکسانی دارند؛ با این تفاوت که در موتور پله‌ای ۶ سیمه ۲ سیم مشترک (Common) وجود دارد، اما در موتور پله‌ای ۵ سیمه هر دو سیم COM به یکدیگر متصل شده‌اند. در موتورپله‌ای‌های تک‌قطبی، دارای ۴ ورودی است که با اعمال پالس به هر یک از آن‌ها، شفت موتور در موقعیت خاصی قرار می‌گیرد. موتورپله‌ای تک‌قطبی را به دو صورت نیم‌پله (half step) و تمام‌پله (full step) می‌توان راهاندازی نمود. برای چرخش شفت موتور در حالت تمام‌پله، کافیست تا سیم‌پیچ‌های داخل موتور را به ترتیب روشن و خاموش کنیم با روشن شدن هر سیم‌پیچ، شفت موتور به اندازه‌ی یک پله به جلو یا عقب رانده می‌شود. جهت حرکت موتور پله‌ای به ترتیب خاموش و روشن شدن سیم‌پیچ‌ها بستگی دارد. در جدول زیر نحوه‌ی روشن و خاموش کردن سیم‌پیچ‌ها در حالت تمام‌پله تک فاز در جهت پادساعتگرد نمایش داده شده است. برای حرکت ساعتگرد کفایست مراحل را از پایین به بالا انجام دهید.

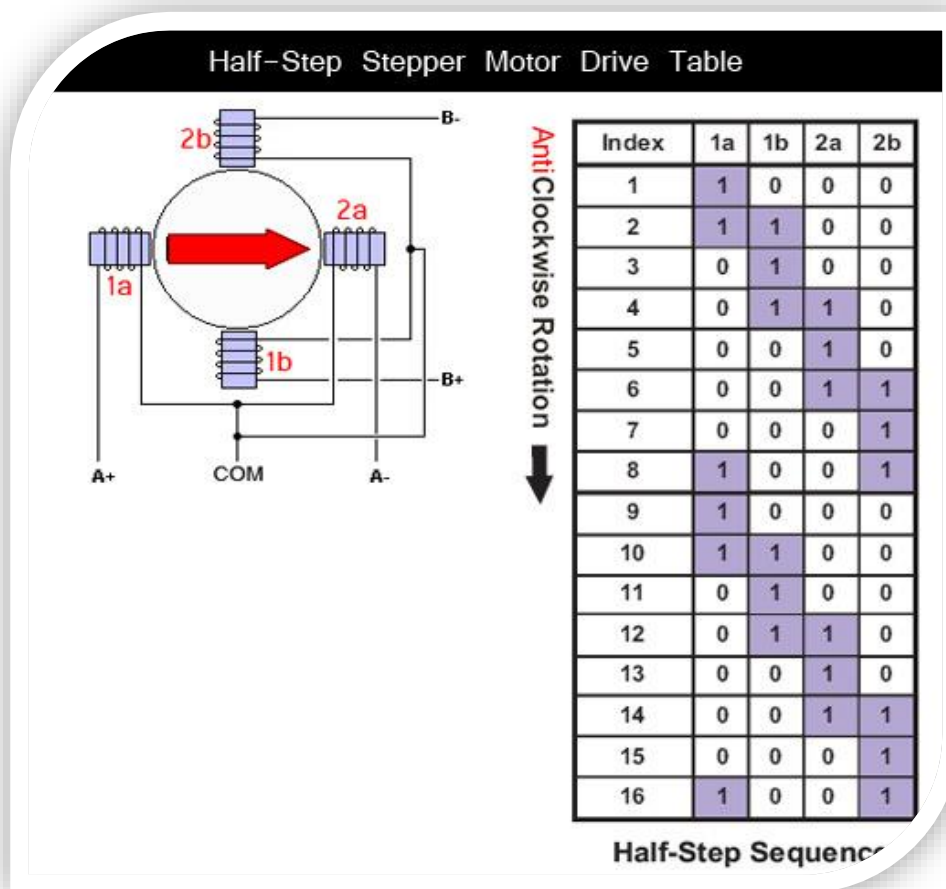


همانطور که در جدول بالا مشاهده می‌کنید، مراحل درایو موتورپله‌ای در حالت تمام‌پله (Full Step) طی ۴ مرحله انجام می‌شود. در حالت تک فاز در هر مرحله تنها یک سیم پیچ روشن و بقیه سیم‌پیچ‌ها خاموش هستند. این روش، آسانترین روش درایو کردن موتور پله‌ای است. شما می‌توانید پس از اتصال سیم مشترک به یکی از قطب‌های تغذیه، سایر سیم‌ها را به ترتیب متصل نمایید، و حرکت موتورپله‌ای را مشاهده نمایید. دقت کنید در موتورپله‌ای‌ها، جهت چرخش موتور ارتباطی به نحوه‌ی اتصال تغذیه (+ و -) ندارد، و با استفاده از ترتیب روشن و خاموش شدن سیم‌پیچ‌ها می‌توان جهت چرخش را مشخص نمود. در جدول بالا مشاهده نمودید که در هر مرحله یک سیم‌پیچ روشن است و به این روش تک فاز یا Single phase گفته می‌شود. اما روش دیگری نیز برای راه‌اندازی تمام‌پله وجود دارد که گشتاور یا Torque بیشتری را فراهم خواهد نمود. در این روش در هر مرحله ۲ سیم‌پیچ روشن است. موتور در این حالت توان بیشتری خواهد داشت. در هر مرحله موتور یک پله کامل را طی می‌کند. جدول راه‌اندازی موتورپله‌ای به این روش نیز در شکل زیر نمایش داده شده است.



همانطور که در جدول بالا مشاهده می‌کنید، این حالت نیز مانند حالت تمام‌پله تک فاز در ۴ مرحله انجام می‌شود. اما حالت دیگری از راه‌اندازی موتورپله‌ای وجود دارد که در هر گام، موتورپله‌ای نیم‌پله به جلو حرکت می‌کند، از این روش برای افزایش دقت موتورپله‌ای استفاده می‌شود. به عنوان مثال اگر شما یک موتورپله‌ای ۱۰ درجه را در حالت نیم‌پله (half step) راه‌اندازی کنید، می‌توانید شفت موتور را با دقت ۵ درجه حرکت دهید. این روش، برخلاف دو روش قبل در ۸ مرحله انجام می‌شود. در این روش پس از روشن شدن هر سیم‌پیچ، در مرحله بعد همان سیم‌پیچ و سیم‌پیچ بعدی روشن می‌شود. در مرحله‌ی بعد سیم‌پیچی که در ۲ مرحله قبل روشن بوده، خاموش می‌شود، و تنها سیم‌پیچی که در مرحله دوم اضافه شده به تنهایی روشن می‌ماند. برای درک ساده‌تر، فکر کنید با روشن شدن سیم‌پیچ ۱ شفت در موقعیت ۰ درجه و با روشن شدن سیم‌پیچ ۲ شفت در موقعیت ۱۰ درجه قرار می‌گیرد. حالا در مرحله اول، سیم‌پیچ ۱ روشن می‌شود؛ و شفت موتور در موقعیت ۰ قرار می‌گیرد. در مرحله بعد، سیم‌پیچ ۱ و ۲ با هم روشن می‌شوند؛ که به علت برابر بودن قدرت هر دو سیم‌پیچ، شفت در موقعیت وسط ۰ و ۱۰ درجه، یعنی در موقعیت ۵ درجه، توقف می‌نماید. در مرحله‌ی سوم، فقط سیم‌پیچ ۲ روشن می‌ماند؛ و موتور به موقعیت ۱۰ درجه

می‌رود. این روند برای هر ۴ سیم پیچ طی می‌شود تا موتور به حرکت خود ادامه دهد. جدول راه‌اندازی موتورپله‌ای به این روش نیز در شکل زیر نمایش داده شده است.



روش دیگری نیز برای درایو کردن موتورپله‌ای تحت عنوان درایو میکرو استپ وجود دارد. در این روش موتورپله‌ای به جای اینکه با موج مربعی راه‌اندازی شود، با موج سینوسی راه‌اندازی می‌شود. علت استفاده از این روش این است که موتورپله‌ای در صورت راه‌اندازی با موج مربعی، حرکت‌های بریده بریده به اندازه گام‌ها خواهد داشت. در این حالت، با لمس بدنه موتور متوجه این ضربه‌ها خواهید شد. اما در درایو میکرواستپ موتورپله‌ای، به علت درایو کردن با موج سینوسی، حرکت موتور یکنواخت، آرام و بدون ضربه خواهد بود. استفاده از این روش، متداول‌ترین روش راه‌اندازی صنعتی موتورپله‌ای است؛ اما نیازمند سخت‌افزار پیچیده‌تری است.

برنامه راه‌اندازی موتورپله‌ای در حالت تمام پله تک‌فاز در جهت ساعتگرد:

```
#include <mega16.h>
```

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

```
char data=0x01;
```

//تعریف متغیر data با مقدار اولیه ۱ برای تحریک قطب‌های موتور

```
void main(void) {
```

```
PORTB=0x00;
```

```
DDRB=0xff;
```

```
while (1) {
```

```
PORTB=data;
```

```
data=data<<1;
```

// شیفت به چپ. چرخش ساعتگرد.

```
If (data==0x10)
```

```
data==0x01;
```

```
delay_ms(100);
```

//وجود تاخیر الزامی است تا سیم پیچ ها زمان کافی برای پاسخ به تحریک را داشته باشند

```
}}
```

سوالات :

۱. برنامه را طوری تغییر دهید که موتورپله‌ای در جهت پادساعتگرد چرخش نماید.
۲. برنامه را طوری تغییر دهید که موتورپله‌ای با گشتاور بیشتری چرخش نماید.
۳. برنامه را طوری تغییر دهید که موتورپله‌ای در حالت نیم‌پله در جهت پادساعتگرد چرخش نماید.
۴. برنامه را طوری تغییر دهید که با افزودن متغیر Speed سرعت چرخش موتور قابل تنظیم باشد.

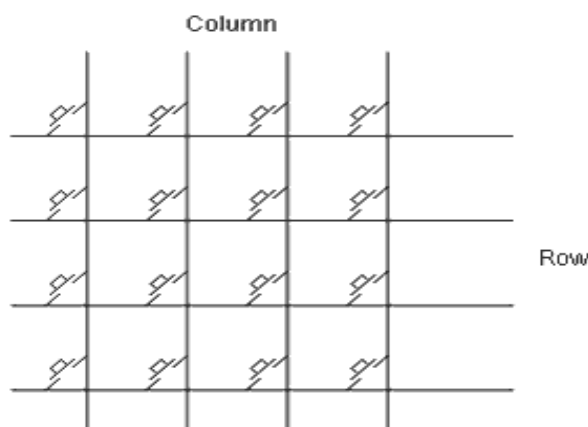
آزمایش شماره ۶

راه اندازی صفحه کلید ماتریسی

هدف از این آزمایش، راه اندازی صفحه کلید ماتریسی برد آموزشی و آشنایی با نحوه برنامه نویسی آن است.

آشنایی با سخت افزار صفحه کلید ماتریسی و مفهوم اسکن کردن :

ارتباط دهی تعداد کمی کلید با میکروکنترلر کاری نسبتاً ساده است؛ اما زمانی که کلیدهای بیشتری نیاز باشد، استفاده از یک صفحه کلید ماتریسی ضروری می شود. درواقع این صفحه کلیدهای توسعه یافته، تک کلیدهای فشاری هستند که برای صرفه جویی در تعداد ورودی - خروجی های مورد نیاز به فرم ماتریسی آرایش یافته اند.



شکل فوق مربوط به آرایش یک صفحه کلید ماتریسی 4×4 است. این صفحه کلید دارای ۴ سطر و ۴ ستون می باشد؛ در نتیجه برای خواندن کلید فشرده شده، ۸ پین ورودی خروجی نیاز خواهد بود. این در صورتی است که اگر با اتصال مستقیم، کلیدها به میکروکنترلر متصل شوند؛ ۱۶ پین مصرف می شود. برنامه باید برای خواندن کلید فشرده شده صفحه کلید را اسکن کند. نحوه عملکرد صفحه کلید بدین ترتیب است که با فشردن هر کلید، سطر و ستون متناظر با آن به یکدیگر متصل می شوند. بنابراین اگر سطر مربوطه ورودی و ستون متناظر آن خروجی باشد، آنچه روی ستون نوشته می شود توسط سطر خوانده می شود. با توجه به این که صفحه کلید یک ماتریس دو بعدی 4×4 است، هر کلید دارای یک مختصات منحصر به فرد خواهد بود. بنابراین برای پیدا کردن کلید فشرده شده، باید مختصات X و Y آن را پیدا کنیم. با فرض اینکه سطرها و ستونها به یک پورت میکروکنترلر متصل شده اند، الگوریتم کار مطابق با فلوجارت زیر خواهد بود.



اکنون که با الگوریتم اسکن صفحه کلید ماتریسی آشنا شده‌اید، می‌خواهیم این الگوریتم را به برنامه تبدیل کنیم تا با استفاده از آن کلید فشرده شده را بخوانیم. با نوشتن تابعی به نام keypad_read یک روتین عمومی برای خواندن صفحه کلید ایجاد می‌کنیم.

گام اول: نوشتن تعاریف و اطلاعات عمومی برنامه (با فرض اینکه صفحه کلید به PORTC متصل شود):

```
#define keypad_port portc
#define keypad_pin pinc
#define keypad_DDR DDRC
Unsigned char keypad_read();
```

گام دوم: بدنه تابع keypad_read :

```
Unsigned char keypad_read()
{
Unsigned char scancode,butNum;
keypad_DDR=0x0f;           (1)
keypad_port=0xf0;          (2)
delay_us(1);                (3)
```

```

scancode = keypad_pin;          (4)
keypad_DDR=0xf0;                (5)
keypad_port=0x0f;               (6)
delay_us (5);                   (7)
scancode = scancode | keypad_pin; (8)
if (scancode==0xff) return 0;   (9)
switch (scancode)                (10)
{
    Case 0xee: butnum=1; break;
    Case 0xed: butnum=2; break;
    Case 0xeb: butnum=3; Break;
    Case 0xe7: butnum=4; Break;
    Case 0xde: butnum=5; Break;
    Case 0xdd: butnum=6 Break;
    Case 0xdb: butnum=7; Break;
    Case 0xd7: butnum=8 Break;
    Case 0xbe: butnum=9; Break;
    Case 0xbd: butnum=10; Break;
    Case 0xbb: butnum=11; Break;
    Case 0xb7: butnum=12; Break;
    Case 0x7e: butnum=13; Break;
    Case 0x7d: butnum=14; Break;
    Case 0x7b: butnum=15; Break;
    Case 0x77: butnum=16; Break;
    Default: butnum=0
}
Return butnum

```

توضیحات برنامه :

(۱) و (۲) : این خطوط مرحله 1 از الگوریتم بحث شده را پیاده سازی می کنند.

(۳) : تاخیر برای همزمان شدن رجیستر PIN با رجیستر PORT

(۴) : پیاده سازی مرحله 2 از الگوریتم

(۵) و (۶) و (۷) : پیاده سازی مرحله 3 از الگوریتم

(۸) : پیاده سازی مرحله 4 و 5 از الگوریتم

(۹) : در صورتی که کلید فشرده نشده باشد مقدار کد اسکن برابر 0xff است.

(۱۰) : تا این مرحله عدد ذخیره شده در متغیر scan code حاوی کد حاصل از اسکن است که این کد برای هر یک از کلیدهای فشرده شده منحصر به فرد است. از آنجایی که علایم روی صفحه کلید متفاوت با کد اسکن است نیاز به برقراری تناظر بین هر یک از کدهای اسکن با شماره کلید مورد نظر است بدین منظور می توان از دستور انتخاب switch استفاده کرد.

با فرض اینکه نحوه شماره گذاری صفحه کلید مطابق با جدول زیر است. دستور switch شماره کلید را در butnum ذخیره می کند. در صورتیکه چند کلید همزمان فشرده شوند کد اسکنی ایجاد می شود که پیش بینی نشده است. پس در خط default آن را بی ارزش تلقی کرده و همانند فشرده نشدن کلید آن را با صفر مشخص می کنیم.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

شناسایی اصول استفاده از صفحه کلیدهای ماتریسی :

- (۱) عدد فشرده شده بر روی LCD نمایش داده شود.
- (۲) کلیدی برای پاک کردن صفحه نمایش و رفتن به سر خط وجود داشته باشد.

سوالات :

۱. با استفاده از برنامه اسکن صفحه کلید، برنامه ای بنویسید که با فشردن کلیدهای صفحه کلید، اعداد مطابق جدول فوق بر روی LCD نمایش داده شود.
۲. برنامه را طوری تغییر دهید که کلیدهای فشرده شده اعداد و نشانه های متناظر با هر کلید بر روی برد آزمایشگاه، بر روی LCD نمایش داده شود.
۳. برنامه را طوری تغییر دهید که علاوه بر نمایش اعداد بر روی صفحه کلید، با فشردن کلیدهای F1 ، F2 ، F3 و Enter ، عبارت "UUT University" را در حالت های زیر نمایش دهد:
Enter : صفحه نمایش پاک شود.

- F1 : شیفت متن به چپ به تعداد یک کاراکتر
F2 : شیفت متن به راست به تعداد یک کاراکتر
F3 : نمایش متن در وسط صفحه
۴. برنامه فوق را طوری تغییر دهید که هنگام فشردن کلیدها، صدا تولید شود.

آزمایش شماره ۷

راهاندازی حسگر دما

هدف از این آزمایش، راهاندازی حسگر دما LM35 و آشنایی با نحوه برنامه‌نویسی مبدل آنالوگ به دیجیتال و حسگر دما است.

حسگر دما LM35 به ازای هر درجه سانتیگراد دما، مقدار 10mV خروجی می‌دهد که به ورودی آنالوگ ADC متصل می‌باشد. برای مثال اگر دما 15°C سانتیگراد باشد، خروجی LM35 برابر با 150mV خواهد بود.

سوالات :

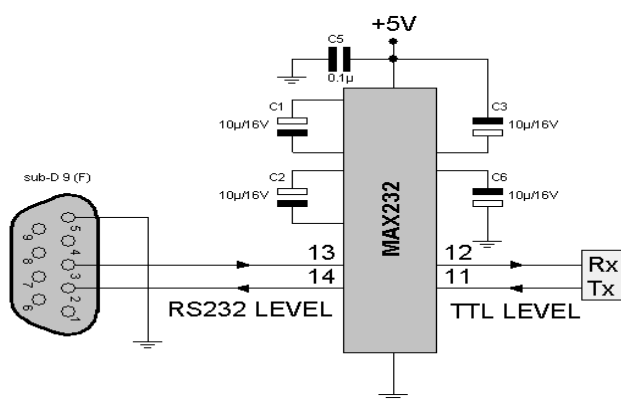
۱. برنامه‌ای بنویسید که دمای محیط را بر روی LCD نمایش دهد.
۲. با قرار دادن انگشتان بر روی سنسور دمای برد آزمایشگاه تغییرات دما را بر روی LCD مشاهده نمایید.

آزمایش شماره ۸

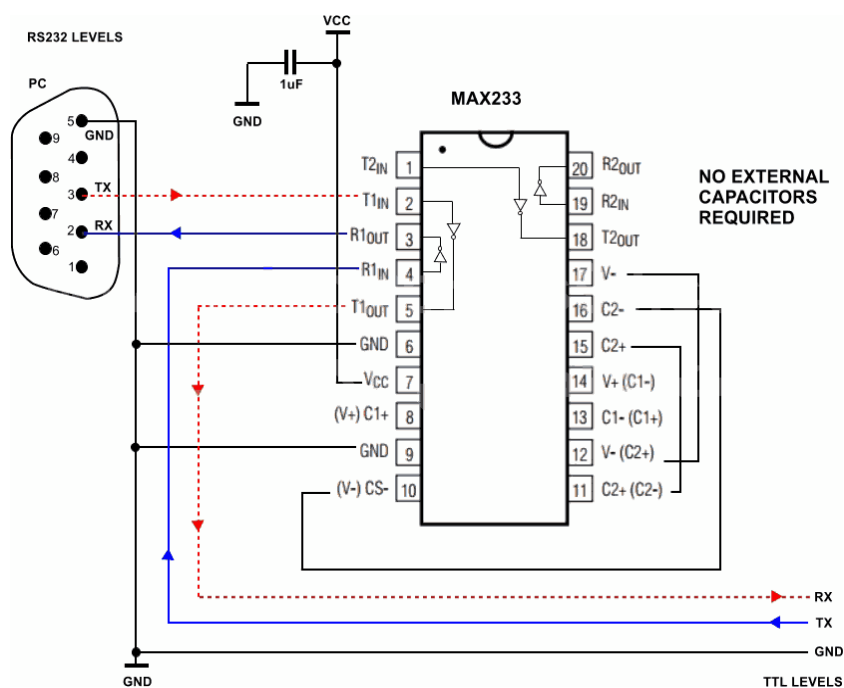
راهاندازی USART

هدف از این آزمایش، راهاندازی USART برد آموزشی و آشنایی با نحوه برنامه‌نویسی آن است.

از آن جایی که برای اتصال میکرو به کامپیوتر از استاندارد RS232 استفاده می‌شود نیاز است که از یک عدد تراشه MAX232 استفاده گردد. نحوه اتصال این آی سی به میکروکنترلر را در شکل زیر مشاهده می‌کنید.



به علت اینکه برای راهاندازی تراشه MAX232 شما احتیاج به استفاده از چندین خازن دارید، می‌توان از تراشه MAX233 استفاده کرد که در آن احتیاج به استفاده از هیچ خازنی ندارید. نحوه ارتباط این آی سی را در شکل زیر مشاهده می‌کنید.

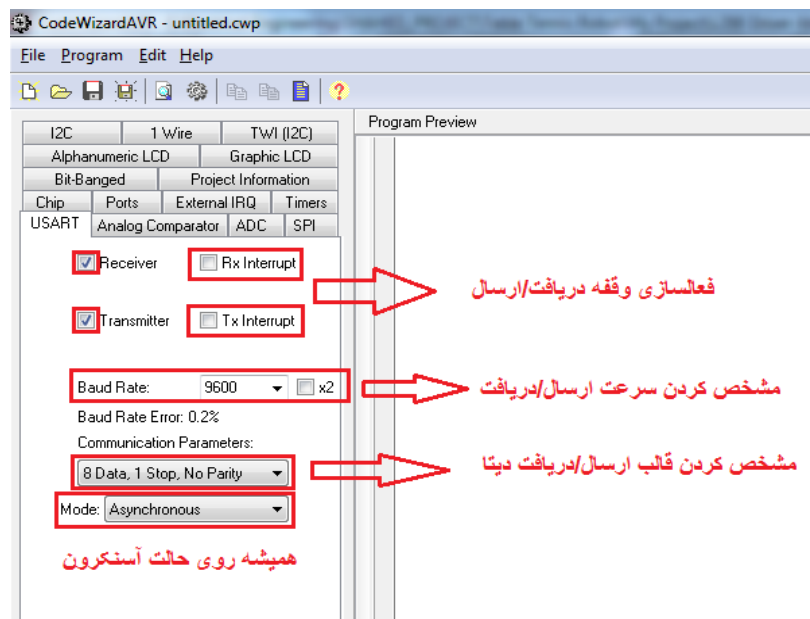


بعد از اتصال میکرو به پورت RS232، نیاز به یک کابل ارتباطی سریال (۹DB) برای اتصال میکرو به پورت سریال PC دارید که در شکل زیر آن را مشاهده می کنید.

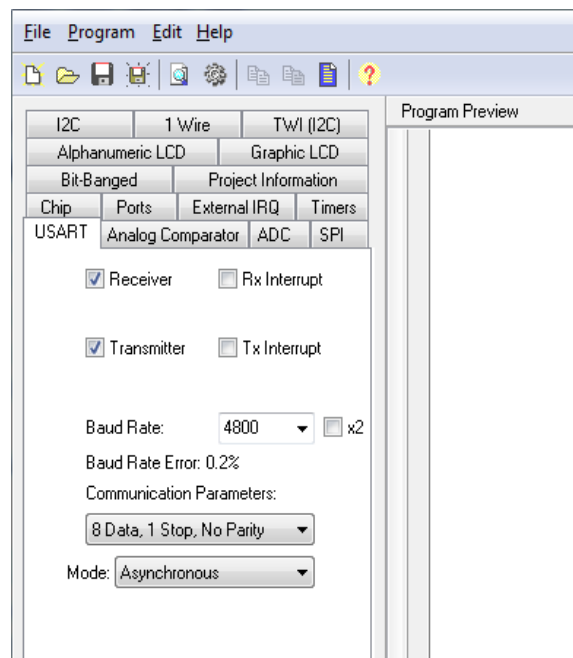


پس از پروگرام کردن و کامل کردن اتصالات، برای شروع ارتباط کامپیوتر با میکروکنترلر نیاز به یک اینترفیس نرمافزاری در PC داریم که از طریق آن با پورت سریال و میکرو ارتباط برقرار کنیم. نرم افزارهای مختلفی برای تبادل اطلاعات سریال از طریق پورت RS232 (یا همان پورت COM) وجود دارد. یکی از این برنامه ها که در محیط نرم افزار CodeVision AVR ارائه می شود، ابزار Terminal است. در نرم افزار CodeVision AVR، ابزاری برای این منظور تعبیه شده است که در منوی Tools قرار دارد. بنابراین بعد از اتصال کابل سریال به کامپیوتر و وصل کردن منبع تغذیه به میکرو و روشن نمودن آن، از منوی Setting در نرم افزار CodeVision AVR، Terminal را انتخاب کرده و تنظیمات مربوط به COM پورتهی که میکرو به آن وصل است و قاب دیتا و نرخ ارسال/دریافت را نیز مشخص می کنیم. سپس از طریق منوی Tools وارد ترمینال می شویم و حالا می توان برای میکرو کاراکتری را فرستاد یا هر آنچه میکرو ارسال می کند را مشاهده کرد.

تنظیمات واحد USART در کدویزارد Codewizard : پس از رفتن به سربرگ USART، بر حسب نیاز به حالت یک طرفه یا دوطرفه می توان Transmitter, Reciever یا هر دو را فعال کرد. اگر گزینه فعالسازی وقفه (interrupt) نیز فعال شود ، وقفه داخلی برای ارسال/دریافت فعال می شود و به برنامه تابع سابروتین مربوطه اضافه می گردد . سپس در قسمت BaudRate نرخ ارسال/دریافت دیتا را مشخص می کنیم . نرخ ارسال/دریافت باید طوری باشد که مقدار درصد خطا که زیر آن نوشته شده است ، مقدار قابل قبولی باشد . با فعال کردن گزینه ۲x نیز میتوان نرخ ارسال/دریافت را توسط مدار ضرب کننده دو برابر کرد . در قسمت Communication Parameter نوع قاب دیتا را مشخص می کنیم . توجه شود که قاب دیتا و نرخ ارسال/دریافت در گیرنده و فرستنده باید یکی باشد . در قسمت Mode نیز سنکرون یا آسنکرون بودن ارتباط را مشخص می کنیم (این قسمت همیشه روی آسنکرون است).



پس از تولید کد توسط برنامه کدویزارد مشاهده می شود کتابخانه `stdio.h` به برنامه اضافه می شود. درون این کتابخانه توابع کار با `USART` وجود دارد که در طول برنامه نویسی می توان از آنها برحسب نیاز استفاده کرد. برای انجام این آزمایش، پس از انجام تنظیمات کدویزارد مربوط به سربرگ های `chip`، `port`، `Alphanumeric` و `LCD` به همان صورت توضیح داده شده به سربرگ `USART` رفته و تنظیمات مربوطه را به صورت شکل زیر انجام دهید.



پس از تولید کد اولیه توسط کدویزارد و حذف کدها و کامنت های اضافی نوبت به برنامه نویسی پروژه می رسد. بعنوان مثال، برنامه زیر به صورتی نوشته شده است که ابتدا میکرو منتظر می ماند تا کلید `b` از صفحه کلید کامپیوتر زده شود سپس از `adc` مقدار سنسور دما را خوانده و ابتدا روی `lcd` نمایش داده و سپس به کامپیوتر ارسال می نماید.


```

#include <mega16.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>
#include <stdlib.h>

#define ADC_VREF_TYPE 0x40

int a,i;
char s[16],c='b';

unsigned int read_adc(unsigned char adc_input) {
    ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);

    delay_us;(1·)

    ADCSRA|=0x40;
    while ((ADCSRA & 0x10)==0);
    ADCSRA|=0x10;
    return ADCW;}

void main(void) {
    DDRC=0xF7;
    PORTC=0x00;

    //USART initialization

    //Communication Parameters: 8 Data, 1 Stop, No Parity

    //USART Receiver: On

    //USART Transmitter: On

    //USART Mode: Asynchronous

    //USART Baud Rate: 4800

    UCSRA=0x00;
    UCSRB=0x18;
    UCSRC=0x86;
    UBRRH=0x00;

```

```

UBRRL=0x67;

//ADC initialization

//ADC Clock frequency: 125/000 kHz

//ADC Voltage Reference: AVCC pin

//ADC Auto Trigger Source: None

ADMUX=ADC_VREF_TYPE & 0xff;

ADCSRA=0x86;

lcd_init(16);

lcd_clear();

lcd_putsf("hello");

lcd_gotoxy(0,0);

lcd_putsf("b to update temp");

while(1){

putsf("please insert b");

c=getchar();

a=read_adc(0);

sprintf(s," Temperature=%d ",a/2);

lcd_gotoxy(0,1);

lcd_puts(s);

if(c=='b'){

puts(s);

putchar(13);

putchar(10);

} } }

```

سوالات

- (۱) برنامه‌ای بنویسید که یک رشته کد اسکی را از کامپیوتر دریافت کرده و آن را بر روی LCD نمایش دهد.
- (۲) برنامه قسمت قبل را طوری تغییر دهید که علاوه بر نمایش رشته بر روی LCD، معکوس رشته را به کامپیوتر ارسال کند.

منابع

- [۱] مرجع جامع میکروکنترلر AVR، رضا سپاس یار و یداله مهریزی، چاپ دوم ۱۳۹۲
- [۲] امیر ره افروز، میکرو کنترلرهای AVR و کاربردهای آن، انتشارات نص، ۱۳۷۹
- [۳] مهندس سعید شجاعی و نادر مهرا، میکرو کنترلرهای AVR سری Mega ، موسسه فرهنگی هنری دیباگران تهران، ۱۳۸۴