وزارت علوم، تحقیقات و فن آوری

مؤسسه آموزش عالی
جهاد دانشگاهی یزد

دانشکده فنی - مهندسی

پایان نامه جهت اخذ درجه کاردانی
در رشته برق – الکترونیک

عنوان :

حافظه داخلی تلفن

استاد راهنما : جناب آقای محمدرضافتاحی

توسط: فاطمه السادات محمودی

تابستان ۱۳۸۶

فهرست مطالب

بنام خدا

چکیده :

"حافظه داخلی تلفن" شاید هنوز برای عده ای این اسم نا مفهوم باشد یا اینکه آن را با ***CALLERID***اشتباه بگیرند.

حافظه داخلی تلفن یعنی ذخیره سازی تعدادی شماره روی تلفن تا با زدن تنها دکمه ای شماره مورد نظر ما گرفته .

پروژه‌من در این رابطه توسط برنامه ای به عنوان میکرو کنترلر ***AVR*** طراحی شده است.

این دستگاه قابلیت ذخیره سازی ۱۰ شماره را داراست.

این پروژه طی سه مرحله انجام شده است.

۱. مرحله نرم افزاری که مربوط به بخش برنامه نویسی است (مهمترین مرحله)
۲. مرحله سخت افزاری
۳. مرحله مطالعهو پژوهش جهت تهیه پایان نامه

تاریخچه و مقدمه :

ریزپردازنده وسیله ای است که می توان با دادن فرمان آن را به عملیات مختلف واداشت . یعنی یک کنترل کننده قابل برنامه ریزی است . همه ریزپردازنده ها سه عمل اساسی یکسانی را انجام می دهند : انتقال اطلاعات ، حساب و منطق ، تصمیم گیری ، اینها سه کار یکسان هستند که به وسیله هر ریزپردازنده ، کامپیوتر کوچک یا کامپیوتر مرکزی انجام می شود .

اولین ریزپردازنده تک تراشه ای ، ریزپردازنده ***Intel 4004*** بود که توانست دو عدد ۴ بیتی دودویی را جمع کند و عملیات متعدد دیگری را انجام دهد .

۴۰۰۴ با معیارهای امروزی یک وسیله کاملا ابتدایی بود که می توانست ۴۰۹۶ مکان مختلف را آدرس دهد. برای حل این مسئله بود که ریزپردازنده ۸ بیتی (۸۰۰۸) به وسیله شرکت ***Intel*** معرفی شد .

Intel 8008:

Intel 8008 توانست اعداد ۸ بیتی را (که بایت نامیده می شوند) به کار گیرد ، که این خود پیشرفت بزرگی نسبت به ۴۰۰۴ بود . تقریبا در همان زمان گشایشی در ساختن مدارهای منطقی ***NMOS*** (نیمه هادی اکسید فلز از نوع ***N***)پیش آمد . منطق ***NMOS*** بسیار سریع تر از ***PMOS*** است . به علاوه از یک منبع تغذیه مثبت استفاده می کند که آن را برای اتصال به مدارهای منطقی ***TTL*** سازگارتر می کند . خصوصیات مذکور از این جهت دارای اهمیت است که بسیاری از مدارهای جنبی ریزپردازنده از نوع ***TTL*** هستند . ***NMOS*** سرعت ریزپردازنده را با ضریبی در حدود ۲۵ بار افزایش می دهد که رقم چشمگیری است .

این تکنولوژی جدید درساختمان ریزپردازنده معروف امروزی یعنی ***Intel 8080*** به کار برده شد .

Intel 8080 :

Intel 8080 در ۱۹۷۳ و معرفی آن دنیا را به دوره ریزپردازنده وارد کرد . ۸۰۸۰ نوع بسیار غنی شده ای از ۸۰۸۰ بود که می توانست ۵۰۰۰۰۰ عمل را در ثانیه انجام دهد و ۶۴ کیلو بایت از حافظه را آدرس می دهد و ۵۰۰۰۰۰ دستورالعمل را در ثانیه اجرا کند . امتیاز اصلی ***Z80*** نسبت به ۸۰۸۰ این است که می تواند از دستورالعمل هایی که برای ۸۰۸۰ می شوند نیز استفاده کند . نرم افزاری که برای ۸۰۸۰ استفاده می شود بدون پیچیدگی بر روی ***Z80*** قابل اجرا است . یک مشخصه سخت افزاری مهم ***Z80*** در مقایسه با ۸۰۸۰ آرایش کامل تر ثبات هاست . ***Z80*** همچنین مکانیزمی را به کار می گیرد که حافظه ***RAM*** دینامیکی را به طور خورکار تازه می کند . این دو مشخصه اضافی موجب برتری ***Z80*** نسبت به ***Intel 8080*** شده است.

سایر ریزپردازنده های اولیه :

تا سال ۱۹۷۳ ، ***Intel*** تولید کننده اصلی ریزپردازنده ها بود . بعد از آن تولید کنندگان دیگر متوجه شدند که این وسیله جدید دارای آینده است و شروع به تولید انواع اصلاح شده دیگری از ریزپردازنده ***Intel 8080*** کردند .

ریزپردازنده های امروزی :

به نظر می رسد که آینده توجه ریزپردازنده در دست سه شرکت ***Intel*** ، ***Motorola*** و ***Zilog*** است . این شرکت ها هر یک با دو سال یک بار انواع پیشرفته تری از ریزپردازنده ها را تولید می کنند . امروزه ریزپردازنده ها از نظر اندازه بین ۴ تا ۳۲ بیت دارند .

تولید کننده	شماره قطعه	اندازه کلمه
Intel	8048	8
Intel	8051	8
Intel	8085A	8
Intel	8086	16
Intel	8088	16
Intel	8096	16
Intel	80186	16
Intel	80188	16
Intel	80286	16
Intel	80386	32
Motorola	6800	8
Motorola	6805	8
Motorola	6809	8,16
Motorola	68000	16,32
Motorola	68008	16,32
Motorola	68010	16,32
Motorola	68020	32
Zilog	Z8	8
Zilog	Z80	8
Zilog	Z8000	16
Zilog	Z80000	32

انواع میکروپروسسورها :

۱. ***Genela*** (که خود شامل ***cpu*** می باشد که بر اساس برنامه وظیفه آنها تغییر می کند) و ***µ.c*** که از تکنولوژی ***RISC*** سود می برد .

۲.پروسسورهای صوتی : سری ***VP*** ساخت شرکت ***QUICK*** و سری ***ISD***

۳.پروسسورهای مخابراتی (شرکت ***MITEL*** فقط پروسسورهای مخابراتی می زند) .

۴. پروسسورهای خاص (برای کاربردهای خاص استفاده می شود)

در معماری ***CPU*** از تکنولوژی ***CISC*** و ***RISC*** استفاده شده که تکنولوژی ***CISC*** (***Complex INSTROCTION set Computer***)دستورات پیچیده را در داخل خود اجرا می کند و تکنولوژی ***RISC***(***Reduce INSTROCTION set Computer***)

SET کامپیوتری است که دستورات ساده ای دارد که از این نوع تکنولوژی در میکرو کنترلرها نیز استفاده شده و خواص آن تعداد کم دستورالعمل ها می باشد .

تعریف ***µ.c*** :

تراشه هایی هستند که واسطهای صفحه کلید ، دیسک و در بسیاری از دیگر دستگاهها استفاده می شود . این نوع تراشه ها به علت حجم بسیار کوچک که دارند به نام ***single µ.c chip*** معروفند .

تفاوت میان ریزپردازنده با ریز کنترل کننده (***µ.c***) :

ریز کنترل کننده ها علاوه بر ***cpu*** شامل حافظه ، خطوط ***I/O*** تایمر ، کانتر و در برخی از آنها حتی ***A/D*** نیز دارند.حال به مروری بر میکروهای ***AVR*** و انواع آنها می پردازیم.

مقدمه :

- الکترونیک در زندگی امروز

امروزه پیشرفت در الکترونیک ای امکان را به ما داده است تا بتوانیم انواع وسایل الکترونیکی مانند ماشین حساب های جیبی ، ساعت رقمی ، کامپیوتر برای کاربرد در صنعت در تحقیقات پزشکی و یا طریقه تولید کالا به طور اتوماتیک در کارخانجات و بسیاری از موارد دیگر را مستقیم یا غیر مستقیم مورد استفاده قرار دهیم .

اینها همه به خاطر آن است که فن آوری توانسته مدارهای الکترونیکی را که شامل اجزاء کوچک الکترونیکی هستند ، بر روی یک قطعه کوچک سیلیکن که شاید سطح آن به ۵ میلی متر مربع بیشتر نیست ، جای دهد . فن آوری میکروالکترونیک که به مدارهای یکپارچه معروف به آی سی یا تراشه مربوط می گردد ، در بهبود زندگی بشر تاثیر به سزایی داشته و آن را بطور کلی دگرگون نموده است . تراشه ها همچنین برای مصارفی چون کنترل رباتها در کارخانجات ، یا کنترل چراغهای راهنمایی و یا وسایل خانگی مانند ماشین لباس شویی و غیره مورد استفاده قرار می گیرند . از طرفی تراشه ها را می توان مغز دستگاه هایی چون میکرو کامپیوترها و رباتها به حساب آورد .

- سیستم های الکترونیکی

پس از یک نظر اجمالی در داخل یک سیستم الکترونیکی مانند یک دستگاه رادیو ، تلویزیون و یا کامپیوتر ممکن است انسان از پیچیدگی آن و از یادگیری الکترونیک دلسرد شود ، اما در واقع آن طور که به نظر می رسند ، دشوار نیستند و این به دو دلیل است .

ا ول اینکه اگرچه سیستم های الکترونیکی اجزاو قطعات زیادی را در خود جای می دهند ، اما باید دانست که انواع کلی این اجزا اغلب محدود و انگشت شمار هستند .

از مهم ترین گروه های این اجزا می توان مقاومت ها ، خازن ها ، القا گرها ، دیودها ، ترانزیستورها ، کلیدها و مبدل ها را نام برد . این اجزا زمانی که به صورت یکپارچه در یک تراشه قرار می گیرند ، هر یک

همان وظیفه خود را به عنوان یک قطعه مجزا انجام می دهند و فقط اندازه فیزیکی آن کوچکتر شده است

.

دوم اینکه انواع سیستم های الکترونیکی از تعداد محدودی مدارهای اصولی و یا بلوک هایی که وظیفه هر کدام به کاراندازی قسمتی از سیستم مثلا تقویت یا شمارش است ، تشکیل یافته اند که به منظور عملکرد کل سیستم ، آن را به یکدیگر متصل می نمایند .

- مدارهای خطی و مدارهای رقمی

بسیاری از سیستم های الکترونیکی طوری طراحی شده اند تا با دریافت یک ورودی الکتریکی و با پردازش آن ، یک خروجی الکتریکی تولید کرده تا بتوانند کار معینی را انجام دهند (که این کار بدون سیستم مورد نظر ، به تنهایی از عهده ورودی الکتریکی مذکور ساخته نخواهد بود .)

مدارهای الکترونیکی که در سیستم ها کاربرد دارند به دو دسته مهم تقسیم می شوند : مدارهای خطی (یا قیاسی) و مدارهای رقمی یا دیجیتال .

مدارهای خطی ار نوع مدارهای تقویت کننده هستند که با سیگنال هایی سرو کار دارند که این سیگنال ها معرف کمیت هایی مانند تغییرات صوتی ، صدای انسان یا موسیقی و غیره هستند . در بسیاری از مدارهای خطی از ترانزیستور به عنوان تقویت کننده صوتی استفاده می کنند . مدارهای دیجیتال از نوع مدارهای کلیدزنی هستند ، که مقدار ورودی یا خروجی آنها در هر زمان فقط می تواند دارای یکی از دو حالت صفر یا یک باشد و اگر قرار است این دو حالت به هم تبدیل شوند این تبدیل حالت بسیار سریع اتفاق می افتد ، در حالی که مدارهای خطی دارای حالت مداوم بوده و این حالات به تدریج در واحد زمان قابل تغییر هستند .

مدارهای رقمی دارای فقط دو حالت هستند و ورودی و خروجی آنها به اصطلاح (***high***) به معنی بالا ، یعنی نزدیک به میزان ولتاژ منبع مدار و یا (***low***) به معنی پایین ، یعنی نزدیک صفر ولت هستند .

در این مدارها عمل کلیدزنی به وسیله ترانزیستور انجام می گیرد . دستگاه شمارش گر در واقع یک مدار رقمی است که در آن سیگنال تولید شده توسط سلول نوری ، یا در حالت صفر و یا در حالت یک قرار می گیرد و این امر بستگی به قطع شدن یا نشدن نور دارد . بنابراین مدارهای رقمی علائم الکتریکی را به

صورت پالس یا ضربه با خود حمل می کنند . سیستمی که در آن یک لامپ توسط دیمر کنترل و کم و زیاد می شود ، یک سیستم حالت مداوم و سیستمی که همان لامپ را خاموش و روشن می کند یک سیستم دو حالته است ، چون که توسط آن لامپ مذکور یا کاملا روشن یا کاملا خاموش می شود .

مختصری راجع به AVR :

زبانهای سطح بالا یا همان HLL (HIGH LEVEL LANGUAGES) به سرعت درحال تبدیل شدن به زبان برنامه نویسی استاندارد برای میکروکنترلرها (MCU)حتی برای میکروهای ۸ بیتی کوچک هستند.زبان برنامه نویسی BASIC و C بیشترین استفاده را در برنامه نویسی میکرو ها دارند، ولی در اکثر کاربردها کدهای بیشتری را نسبت به زبان برنامه نویسی اسمبلی تولید می کنند.

ATMEL ایجاد تحولی در معماری ،جهت کاهش کد به مقدار مینیمم را درک کرد که نتیجه این تحول میکروکنترلرها AVR هستند که علاوه بر کاهش و بهینه سازی مقدار کدها به طور واقع عملیات را تنها در یک کلاک سیکل توسط معماری (REDUCED RISC INSTRUCTION SET COMPUTER) انجام می دهند و از ۳۲ رجیستر همه منظوره (ACCUMULATORS) استفاده می کنند که باعث شده ۴ تا ۱۲ بار سریعتر از میکروهای مورد استفاده کنونی باشند.

تکنولوژی حافظه کم مصرف غیر فرار شرکت ATMEL برای برنامه ریزی AVR ها مورد استفاده قرار گرفته است در نتیجه حافظه های FLASH و EEPROM در داخل مدار قابل برنامه ریزی (ISP) هستند. میکروکنترلرهای اولیه AVR دارای ۱،۲و۸ کیلو بایت حافظه FLASH و به صورت کلمات ۱۶ بیتی سازماندهی شده بودند.

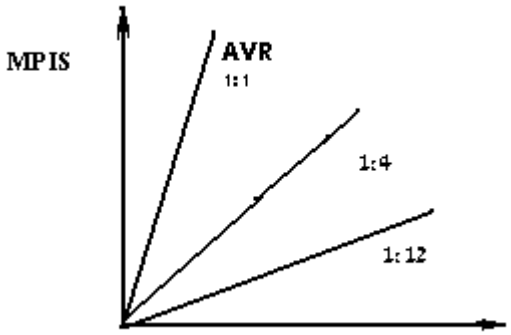
AVR ها به عنوان میکروهای RISC با دستورات فراوان طراحی شده اند که باعث می شود حجم کد تولید شده کم وسرعت بالاتری بدست آید.

عملیات تک سیکل:

با انجام تک سیکل دستورات،کلاک اسیلاتور با کلاک داخلی سیستم یکی می شود. هیچ تقسیم کننده ای در داخل AVR قرار ندارد که ایجاد اختلاف فاز کلاک کند. اکثر میکرو ها کلاک اسیلاتور به سیستم را با

نسبت ۱:۴ یا ۱:۱۲ تقسیم می کنند که خود باعث کاهش سرعت میشود. لذا AVR ها ۴ تا ۱۲ بار سریعتر و مصرف آنها نیز ۴-۱۲ بار نسبت به میکروکنترلرهای مصرفی کنونی کمتر است زیرا در تکنولوژی CMO استفاده شده در میکروهای AVR ، مصرف توان سطح منطقی متناسب با فرکانس است.

نمودار زیر افزایش MIPS (MILLION INSTRUCTION PER SECONDS) را به علت انجام عملیات تک سیکل AVR (نسبت ۱:۱) در مقایسه با نسبت های ۱:۴ و ۱:۱۲ در دیگر میکرو ها نشان می دهد.



طراحی برای زبان های BASIC و C :

زبان های BASIC و C بیشترین استفاده را در دنیای امروز به عنوان زبان های HLL دارند تا امروزه معماری بیشتر میکرو ها برای زبان اسمبلی طراحی شده و کمتر از زبان های HLL حمایت کرده اند.

هدف ATMEL طراحی معماری بود که هم برای زبان اسمبلی و هم زبان های HLL مفید باشد. به طور مثال در زبان های C و BASIC می توان یک متغیر محلی به جای متغیر سراسری در داخل زیر برنامه تعریف کرد ، در این صورت فقط در زمان اجرای زیر برنامه مکانی از حافظه RAM برای متغیر اشغال می

شود در صورتی که اگر متغیری به عنوان سراسری تعریف گردد در تمام وقت مکانی از حافظه FLASH ROM را اشغال کرده است.

برای دسترسی سریعتر به متغیرهای محلی و کاهش کد ، نیاز به افزایش رجیسترهای همه منظوره است .

AVR ها دارای ۳۲-رجیستر هستند که مستقیما به LOGIC ALU (ARITHMETIC UNIT) متصل شده اند ،و تنها در یک کلاک سیکل به این واحد دسترسی پیدا می کنند. سه جفت از این رجیسترها می توانند به عنوان رجیسترهای ۱۶ بیتی استفاده شوند.

برنامه صفحه بعد نشان میدهد که چگونه تعداد مناسب رجیسترهای همه منظوره (در AVRها) می توانند با معماری CISC با یک ACCOMULATOR مقایسه گردند.برای این منظور ما می خواهیم از معادله ی صفحه ی بعد A را به دست بیاوریم. می بینیم که با کدهای AVR این در عرض ۴ کلاک سیکل و با کدهای CISC در عرض ۹۶-۴۸ کلاک سیکل انجام می گیرد.

نتیجه تمام این موارد بحث شده ، میکروکنترلرها AVR با سرعت بالا و سازماندهی RISC هستند.

میکروکنترلرهای AVR به سه نوع AT90S یا AVR، TINY AVR و MEGA AVR تقسیم بندی شده اند. دیمانسیون متغیر

این دستور بعد یک متغیر را نشان میدهد. با این دستور می توانید متغیرهایی که در برنامه به کار برده می شوندتعریف کنید.

DIM var AS [XRAM/SRAM/ERAM] data type [AT location] [OVERLAY]

VAR نام متغیری که در برنامه به کار برده می شود. در صورت استفاده از حافظه جانبی آن را با XRAMمشخص کنید و SRAM را زمانی اختیار کنید که می خواهید متغیر ها را در حافظه SRAM قرار دهید و ERAM متغیر مورد نظر را در EERAM داخلی جای می دهد.

Data type نوع داده است که می تواند طبق جدول زیر STRING , WORD , LONG , INTERGER , BYPE , BIT یا SINGOL باشد. در صورت استفاده از متغیر SINGOL، بیشترین طول آن نیز باید نوشته شود. گزینه اختیاری OVERLAYمتغیر تعریف شده را به صورت POINTER در نظر می گیرد و فضایی را برای متغیر در نظر نمی گیرد.

AT LOCATION به شما اجازه می دهد که متغیّر تان را در آدرسی که می خواهید در حافظه ذخیره کنید. زمانی که محل آدرس دهی اشغال باشد ، اولین جای خالی در حافظه استفاده می شود.

- خصوصیات *Atmega16L* و *Atmega16*

ازمعماری *AVR RISC* استفاده می کند .

- کارایی بالا و توان مصرفی کم

- دارای ۱۳۱ دستورالعمل با کارایی بالا که اکثرا تنها در یک کلاک سیکل اجرا می شوند .

- ۳۲×۸رجیستر کاربردی

- سرعتی تا ***16 MIPS*** در فرکانس ***16 MHZ***

حافظه ، برنامه و داده غیر فرار

-***16K*** بایت حافظه ***FLASH*** داخلی قابل برنامه ریزی

پایداری حافظه ***FLASH*** : قابلیت ۱۰۰۰۰ بار نوشتن و پاک کردن (***WRITE / ERASE***)

-***1024*** بایت حافظه داخلی ***SRAM***

-***512*** بایت حافظه ***EEPROM*** داخلی قابل برنامه ریزی

پایداری حافظه ***EEPROM*** : قابلیت ۱۰۰۰۰۰ بار نوشتن و پاک کردن (***WRITE / ERASE***)

- قفل برنامه ***FLASH*** و حفاظت داده ***EEPROM***

قابلیت ارتباط *JTAG* (*IEEE Std .*)

- برنامه ریزی برنامه ***FLASH*** ، ***EEPROM*** ، ***FUSE BITS*** ، ***LOCK BITS*** از طریق ارتباط ***JTAG***

خصوصیات جانبی

- دو تایمر- کانتر (***TIMER / COUNTER***) ***8*** بیتی با ***PRESCALER*** مجزا و مد ***COMPARE***

- یک تایمر- کانتر (***TIMER / COUNTER***) ***16*** بیتی با ***PRESCALER*** مجزا و دارای مدهای ***COMPARE*** و ***CAPTURE***

- **4** کانال **PWM**

- ***8*** کانال مبدل آنالوگ به دیجیتال ***10*** بیتی

8 کانال ***SINGLE-ENDED***

دارای **7** کانال تفاضلی در بسته بندی ***TQFP***

دارای دو کانال تفاضلی با کنترل گین ***1x*** ، ***10x*** و ***200x***

- یک مقایسه کننده آنالوگ داخلی .

- ***WATCHDOG*** قابل برنامه ریزی با اسیلاتور داخلی .

- قابلیت ارتباط با پروتکل سریال دو سیمه (***TWO-WIRE***)

- قابلیت ارتباط سریال ***SPI*** (***SERIAL PERIPHERAL INTERFACE***) به صورت ***MASTER*** یا ***SLAVE***

-***USART*** سریال قابل برنامه ریزی

خصوصیات ویژه میکروکنترلر

-***POWER-ON RESET CIRCUIT*** و ***BROWN-OUT*** قابل برنامه ریزی .

- دارای اسیلاتور ***RC*** داخلی کالیبره شده

- دارای **6** حالت ***SLEEP*** (***POWER-DOWN*** ، ***IDLE*** ، ***POWER-SAVE*** ، ***STANDBY*** ، ***EXTENDED STANDBY*** و ***ADC NOISE REDUCTION***)

- منابع وقفه (***INTERRUPT***) داخلی و خارجی .

- عملکرد کاملا ثابت .

- توان مصرفی پایین و سرعت بالا توسط تکنولوژی ***CMOS***

توان مصرفی در *1MHZ* ، *3V* ، 25^0C برای *ATMEGA16L*

- حالت فعال ***1.1 mA*** (***ACTIVE MODE***)

- در حالت بی کاری ***0.35 mA*** (***IDLE MODE***)

- در حالت ***POWER-DOWN*** : ***1μA >***

ولتاژهای عملیاتی (کاری)

- ***2.7 V*** تا ***5.5 V*** برای (***Atmega16 L***)

- ***4.5 V*** تا ***5.5 V*** برای (***Atmega16***)

فرکانس های کاری

- ***0MHZ*** تا ***8MHZ*** برای(***Atmega16 L***)

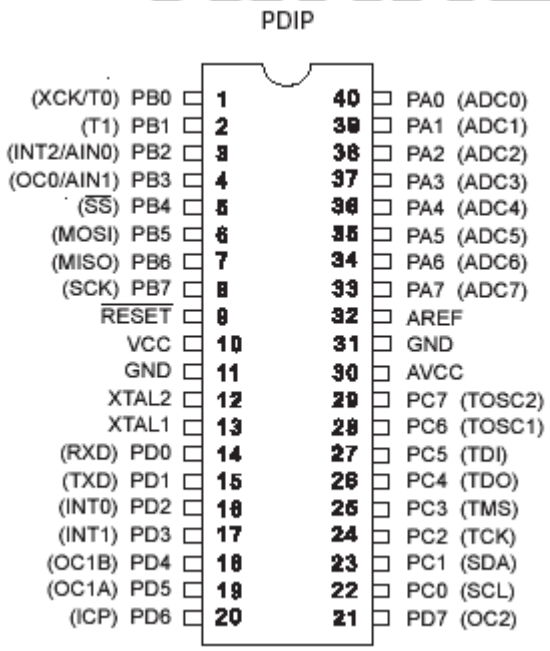
- ***0MHZ*** تا ***16MHZ*** برای (***Atmega16***)

خطوط ***I/O*** و انواع بسته بندی

- ***32*** خط ورودی/ خروجی (***I/O***) قابل برنامه ریزی .

- ***40*** پایه ***PDIP*** ، ***44*** پایه ***TQFP*** و ***44*** پایه ***MLF***

- ترکیب پایه ها:



- فیوز بیت های *ATMEGA16*

ATMEGA16 دارای دو بایت فیوز طبق جدول های زیر می باشد:

FUSE HIGT BYTE

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD	1(UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0(PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0(PROGRAMMED , SPI PROG.ENABLE)
CKOPT	4	OSCILLATOR OPTIONS	1(UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1(UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0(PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1(UNPROGRAMMED)

FUSE LOW BYTE

FUSE HIGH BYTE	BIT NO.	DESCRIPTION	DEFAULT VALUE
BODLEVEL	7	BROWN OUT DETECTOR TRIGGER LEVEL	1(UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1(UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START-UP TIME	1(UNPROGRAMMED)
SUT0	4	SELECT START-UP TIME	0(PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0(PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1(UNPROGRAMMED)

فیوز بیت ها با پاک کردن (***ERASE***) میکرو تاثیری نمی بینند ولی می توانند با برنامه ریزی بیت ***LB1*** قفل شوند . منطق ***0*** به معنای برنامه ریزی شدن و ***1*** به معنای برنامه ریزی نشدن بیت است .

OCDEN : در صورتی که بیت های قفل برنامه ریزی نشده باشند برنامه ریزی این بیت به همراه بیت ***JTAGEN*** باعث می شود که سیستم ***ON CHIP DEBUG*** فعال شود . برنامه ریزی شدن این بیت به قسمت هایی از میکرو امکان می دهد که در مدهای ***SLEEP*** کار کنند که این خود باعث افزایش مصرف سیستم می گردد . این بیت به صورتی پیش فرض برنامه ریزی نشده(*1*) است .

JTAGEN : بیتی برای فعال سازی برنامه ریزی میکرو از طریق استاندارد ارتباطی ***IEEE*** (***JTAG***) که در حالت پیش فرض فعال است و میکرو می تواند از این ارتباط برای برنامه ریزی خود استفاده نماید .

SPIEN : در حالت پیش فرض برنامه ریزی شده و میکرو از طریق سریال ***SPI*** برنامه ریزی می شود .

CKOPT : انتخاب کلاک که به صورت پیش فرض برنامه ریزی نشده است . عملکرد این بیت به بیت های ***CKSEL*** بستگی دارد که در بخش کلاک سیستم (**1**) در انتهای همین فصل آمده است .

EESAVE : در حالت پیش فرض برنامه ریزی نشده و در زمان پاک شدن (***ERASE***) میکرو حافظه ***EEPROM*** پاک می شود ولی در صورتی که برنامه ریزی شود محتویات ***EEPROM*** در زمان پاک شدن میکرو محفوظ می ماند .

BOOTSZ0 , BOOTSZ1 برای انتخاب مقدار حافظه ***BOOT*** طبق جدول زیر برنامه ریزی می شوند و در زمان برنامه ریزی شدن فیوز بیت ***BOOTRST*** اجرای برنامه از آدرس حافظه ***BOOT*** آغاز خواهد شد .

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Addresses	Boot Flash Addresses	Boot Reset Address
1	1	128 words	2	0x0000 - 0x1F7F	0x1F80 -1FFF	0x1F80
1	0	256 words	4	0x0000 - 0x1EFF	0x1F00 -1FFF	0x1F00
0	1	512 words	8	0x0000 - x1DFF	0x1E00-x1FFF	0x1E00
0	0	1024 words	16	0x0000 - 0x1BFF	0x1C00-x1FFF	0x1C00

BOOTRST : بیتی برای انتخاب بردار ریست ***BOOT*** که در حالت پیش فرض برنامه ریزی نشده و آدرس بردار ریست ***$0000*** است و در صورت برنامه ریزی آدرس بردار ریست به آدرسی که فیوز بیت های ***BOOTSZ0*** و***BOOTSZ1*** مشخص کرده اند تغییر می یابد .

BOOTRST	RESET ADDRESS
1(UNPROGRAMMED)	RESET VECTOR = APPLICATION RESET (ADDRESS $0000)
0(PROGRAMMED)	RESET VECTOR = BOOT LOADER RESET

BODLEVEL : زمانی که این بیت برنامه ریزی نشده (پیش فرض) باشد ، اگر ولتاژ پایه ***VCC*** از **2.7V** پایین تر شود ریست داخلی میکرو فعال شده و سیستم را ریست می کند . زمانی که این بیت برنامه ریزی شده باشد ، اگر ولتاژ پایه ***VCC*** از **4V** پایین تر شود ریست داخلی میکرو فعال شده و میکرو را طبق شکل ۳-۲ ریست می شود.

BODEN : برای فعال کردن عملکرد مدار ***BROWN-OUT*** این بیت بایستی برنامه ریزی شده باشد . این بیت به صورت پیش فرض برنامه ریزی نشده است .

BODEN , BODLEVEL	BROWN- OUT DETECTION
11	DISABLE
10	DISABLE
01	AT VCC=2.7V
00	AT VCC=4.0V

SUT1 ,SUT0 : عملکرد این دو بیت برای انتخاب زمان ***START-UP*** در بخش ۳-۱۴ در انتهای همین فصل کاملا توضیح داده شده است .

CKSEL3 ...CKSEL0 : عملکرد این بیت ها در بخش ۳-۱۴ در انتهای همین فصل کاملا توضیح داده شده است . مقدار پیش فرض:

INTERNAL RC OSCILLATOR@ 1MHZ است .

بررسی پورت های میکرو *ATMEGA16*

در این بخش به بررسی عملکرد پورت های میکرو مورد نظر می پردازیم .

پورت ***B*** :

پورت ***B*** یک ***I/O*** دو طرفه ۸ بیتی است . سه آدرس از مکان حافظه ***I/O*** اختصاص به ***PORTB*** دارد . یک آدرس برای رجیستر داده ***PORTB*** دومی رجیستر جهت داده ***DDRB*** و سومی پایه ورودی پورت ***B*** ، ***PINB*** است . آدرس پایه های ورودی پورت ***B*** فقط قابل خواندن است . در صورتی که رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . پایه های پورت دارای مقاومت ***PULL-UP*** مجزا هستند . بافر خروجی پورت ***B*** می تواند تا ***20mA*** را ***Sink*** کند و در نتیجه ***LED*** را مستقیما راه اندازی می کند . هنگامی که ***PB0-PB7*** با مقاومت های ***PULL-DOWN*** خروجی استفاده می شوند آنها ***SOURCE*** جریان می شود . زمانی که مقاومت های ***PULL-UP*** داخلی ۳ فعال باشند .

استفاده از پورت ***B*** به عنوان یک ***I/O*** عمومی دیجیتال :

تمام ۸ پایه موجود زمانی که به عنوان پایه های ***I/O*** دیجیتال استفاده می شوند دارای عملکرد مساوی هستند .

PBn و پایه ***I/O*** عمومی : بیت ***DDBn*** در رجیستر ***DDRB*** مشخص کننده جهت پایه است . اگر ***DDBn*** یک باشد ، ***PBn*** به عنوان یک پایه خروجی مورد استفاده قرار می گیرد و اگر ***DDBn*** صفر باشد ، ***PBn*** به عنوان یک پایه ورودی در نظر گرفته می شود .

اگر ***PortBn*** یک باشد هنگامی که پایه به عنوان ورودی تعریف شود مقاومت ***Pull-UP*** فعال می شود برای خاموش کردن مقاومت ***Pull-UP*** باید ***Port Bn*** صفر باشد یا اینکه پایه به عنوان خروجی تعریف شود . پایه های پورت زمانی که ریست اتفاق می افتد به حالت ***Tristate*** می روند.

دیگر کاربردهای پورت ***B*** :

PORTB.7-TOSC2 :

TOSC2 زمانی که تایمر/ کانتر ۲ در مد آسنکرون کار می کند به این پایه و پایه ***TOSC1*** کریستال ساعت متصل می شود . در این حالت دیگر نمی توان از این پایه با عنوان ***I/O*** استفاده نمود .

PORTB.5-SCK :

SCK کلاک خروجی ***Master*** و کلاک ورودی ***Slave*** برای ارتباط ***SPI*** است . زمانی که ***SPI*** به عنوان ***Slave*** شکل دهی می شود این پایه با توجه به تنظیم ***DDB7*** ورودی و در حالت ***Master*** خروجی تعریف می شود .

PORTB.4-MISO :

MISO ورودی داده ***Master*** و خروجی داده ***slave*** که برای ارتباط ***SPI*** استفاده می شود . زمانی که ***SPI*** به عنوان ***Master*** شکل دهی می شود . این پایه با توجه به تنظیمات ***DDB6*** ورودی و در حالت ***Slave*** به عنوان خروجی استفاده می شود .

PORTB-MPS1,OC2 :

MIS1 ورودی داده ***Slave*** و خروجی داده ***Master*** که برای ارتباط ***SPI*** استفاده می شود . زمانی که ***SPI*** به عنوان ***Master*** شکل دهی می شود . این پایه با توجه به تنظیمات ***DDB3*** خروجی و در حالت ***Slave*** به عنوان ورودی استفاده می شود .

OC2 : خروجی مد مقایسه ای تایمر / کانتر ***PB3.2*** با یک شدن ***DDB3*** می توان به عنوان پایه خروجی مد مقایسه ای ***timer/counter2*** شکل دهی می شود . این پایه همچنین برای خروجی ***PWM*** تایمر استفاده می شود .

PORTB.2-SS.OC1B :

SS زمانی که ***SPI*** به عنوان ***Slave*** شکل دهی شود ***PB2*** با توجه به ***DDB2*** ورودی تعریف می شود و در ***Slave*** با ***Low*** شدن این پایه ***SPI*** فعال می شود . این پایه در ***Master*** می تواند خروجی یا ورودی تعریف شود .

OC1B خروجی مد مقایسه ای ***Timer/Counter 1*** پایه ***PB2*** با یک شدن ***DDB2*** می تواند برای خروجی مد مقایسه ای ***Timer/Counter*** شکل دهی می شود . این پایه همچنین برای خروجی ***PWM*** تایمر استفاده می شود .

PORTB.0.ICP :

PB0.ICP می تواند به عنوان پایه ورودی ***CAPTURE*** تایمر / کانتر ۱ عمل کند .

Port Pin	Alternate Functions
PB7	XTAL2 (Chip Clock Oscillator pin 2) TOSC2 (Timer Oscillator pin 2)
PB6	XTAL1 (Chip Clock Oscillator pin 1 or External clock input) TOSC1 (Timer Oscillator pin 1)
PB5	SCK (SPI Bus Master clock Input)
PB4	MISO (SPI Bus Master Input/Slave Output)
PB3	MOSI (SPI Bus Master Output/Slave Input) OC2 (Timer/Counter2 Output Compare Match Output)
PB2	$\overline{SS}$ (SPI Bus Master Slave select) OC1B (Timer/Counter1 Output Compare Match B Output)
PB1	OC1A (Timer/Counter1 Output Compare Match A Output)
PB0	ICP (Timer/Counter1 Input Capture Input)

پورت *C* :

پورت ***C*** یک ***I/O*** دو طرفه ۷ بیتی است . سه آدرس از مکان حافظه ***I/O*** اختصاص به ***PORTC*** دارد .

یک آدرس برای رجیسترداده ***PORTC*** دومی رجیستر جهت داده ***DDRC*** و سومی پایه پورت ***PINC,C*** است .
آدرس پایه های ورودی پورت ***C*** فقط قابل خواندن است . در صورتی که رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . پایه های پورت دارای مقاومت (***Pull-up***) مجزا هستند . بافر

خروجی پورت ***C*** می تواند تا ***20mA*** را ***sink*** کند و در نتیجه ***LED*** را مستقیما راه اندازی می کند .

هنگامی که ***PC0-PC7*** که با مقاومت های ***Pull-Down*** خروجی استفاده می شوند آنها ***SOURCE*** جریان می شوند زمانی که مقاومت های ***Pull-up*** داخلی فعال باشند .

استفاده از پورت ***C*** به عنوان یک ***I/O*** عمومی دیجیتال :

تمام ۷ پایه موجود زمانی که به عنوان پایه های ***I/O*** دیجیتال استفاده می شوند دارای عملکرد مساوی هستند .

PCn پایه ***I/O*** عمومی : بیت ***DDCn*** در رجیستر ***DDRC*** مشخص کننده جهت پایه است . اگر ***DDCn*** یک باشد ، ***PCn*** به عنوان یک پایه خروجی مورد استفاده قرار می گیرد و اگر ***DDCn*** صفر باشد ، ***PCn*** به عنوان یک پایه ورودی در نظر گرفته می شود .

اگر ***Port Cn*** یک باشد هنگامی که پایه به عنوان ورودی تعریف می شود ، مقاومت ***Pull-up*** فعال می شود . برای خاموش کردن مقاومت ***Pull-up*** باید ***Port Cn*** صفر باشد یا اینکه پایه به عنوان خروجی تعریف شود . پایه های پورت زمانی که ریست اتفاق می افتد به حالت ***Tri-state*** می روند .

دیگر کاربردهای پورت ***C*** :

پورت ***C*** به عنوان ***ADC*** هم استفاده می شوند . اگر تعدادی از پایه های پورت ***C*** خروجی تعریف شوند این نکته بسیار مهم است که در زمان نمونه برداری از سیگنال آنالوگ توسط ***ADC*** ، سوییچ نشوند . این کار ممکن است عملیات تبدیل ***ADC*** را نا معتبر کند .

Port Pin	Alternate Function
PC6	$\overline{\text{RESET}}$ (Reset pin)
PC5	ADC5 (ADC Input Channel 5) SCL (Two-wire Serial Bus Clock Line)
PC4	ADC4 (ADC Input Channel 4) SDA (Two-wire Serial Bus Data Input/Output Line)
PC3	ADC3 (ADC Input Channel 3)
PC2	ADC2 (ADC Input Channel 2)
PC1	ADC1 (ADC Input Channel 1)
PC0	ADC0 (ADC Input Channel 0)

PORTC.6-RESET :

RESET این پایه برای ریست کردن میکرو استفاده می شود .

PORTC.5-SCL,ADC5 :

SCL در زمان ارتباط ***2-WIRE*** به عنوان خط کلاک استفاده می شود .

PORTC.4-SDA,ADC4 :

SDA در زمان ارتباط ***2-WIRE*** به عنوان خط داده استفاده می شود .

پورت D :

پورت ***D*** یک ***I/O*** دو طرفه ۸ بیتی است . سه آدرس از مکان حافظه ***I/O*** اختصاص به ***PORTD*** دارد. یک آدرس برای رجیستر داده ***PORTD*** ، دومی رجیستر جهت داده ***DDRD*** و سومی پایه ورودی پورت ***PIND,D*** است . آدرس پایه های ورودی پورت ***D*** فقط قابل خواندن است در صورتی که رجیستر داده

جهت داده هم خواندنی و هم نوشتنی هستند . پایه های پورت دارای مقاومت ***Pull-up*** مجزا هستند . بافر خروجی پورت ***D*** می تواند تا ***20mA*** را ***sink*** کند و در نتیجه ***LED*** را مستقیما راه اندازی می کند .

هنگامی که ***PD0-PD7*** با مقاومت های ***Pull-Down*** خروجی استفاده می شوند آنها ***SOURCE*** جریان می شوند زمانی که مقاومت های ***Pull-up*** داخلی هستند .

استفاده از پورت ***D*** به عنوان یک ***I/O*** عمومی دیجیتال :

تمام ۸ پایه موجود زمانی که به عنوان پایه های ***I/O*** دیجیتال استفاده می شوند دارای عملکرد مساوی هستند .

PDN پایه ***I/O*** عمومی : بیت ***DDDn*** در رجیستر ***DDRD*** مشخص کننده جهت پایه است . اگر ***DDDn*** یک باشد ، ***PDn*** به عنوان یک پایه خروجی مورد استفاده قرار می گیرد و اگر ***DDn*** به عنوان یک پایه ورودی در نظر گرفته می شود .

اگر ***PortDn*** یک باشد هنگامی که پایه به عنوان ورودی تعریف شود مقاومت ***Pull-up*** فعال می شود برای خاموش کردن مقاومت ***Pull-up*** باید ***PortDn*** صفر باشد یا اینکه پایه به عنوان خروجی تعریف شود . پایه های پورت زمانیکه ریست اتفاق می افتد به حالت ***Tri-state***می روند .

دیگر کاربردهای پورت ***D*** :

PORTD.7.AIN1 :

AIN1 ورودی منفی مقایسه کننده آنالوگ است .

PORTD.7.AIN0 :

AIN0 ورودی مثبت مقایسه کننده آنالوگ است .

POTRD.5.T1 :

T1 ورودی کلاک برای ***Timer / Counter1*** است .

PORTD.4-XCKT0 :

T0 ورودی کلاک برای ***Timer / Counter0*** است .

XCK : این پایه نیز می تواند به عنوان کلاک خارجی ***USART*** مورد استفاده قرار گیرد .

این پایه فقط زمانی که ***USART*** در مد آسنکرون کار می کند فعال می شود .

PORTD.3-INT1 :

INT1 : منبع وقفه خارجی یک

پایه ***PD3*** می تواند به عنوان منبع وقفه خارجی برای میکرو استفاده شود .

PORTD.3-INT0 :

INT0 : منبع وقفه خارجی یک

پایه ***PD2*** می تواند می تواند به عنوان منبع وقفه خارجی برای میکرو استفاده شود .

PORTD.1-TXD :

ارسال داده (پایه خروجی داده برای ***USART***) زمانی که ارسال ***USART*** فعال می شود پایه با توجه به ***DDD1*** به عنوان خروجی شکل دهی می شود.

PORTD.0.RXD :

RXD دریافت داده (پایه ورودی برای ***USART***)

زمانی که دریافت ***USART*** فعال می شود پایه با توجه به ***DDD0*** به عنوان ورودی شکل دهی میشود.

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input)
PD6	AIN0 (Analog Comparator Positive Input)
PD5	T1 (Timer/Counter 1 External Counter Input)
PD4	XCK (USART External Clock Input/Output) T0 (Timer/Counter 0 External Counter Input)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

- پیکره بندی LCD

اتصال پایه های **LCD** به میکرو

پایه های **LCD** برای اتصال به پایه های میکرو به صورت زیر پیکره بندی می شوند .

```
CONFIG  LCDPIN = PIN , DB4 = PN , DB5 = PN , DB6 = PN , _
 DB7 = PN , E = PN , Rs = PN
```

PN : پایه ای دلخواه از میکروکه پایه ی **LCD** به آن اتصال می یابد، به طور مثال **PORTB.1**

باید توجه داشت که پیکره بندی پایه های **LCD** باید در یک خط نوشته شود و یا ادامه آن با علامت _ (**UNDER LINE**) در خط بعد نوشته شود .

تشریح پایه های **LCD**

(Vss) پایه شماره۱: برای پیکره بندی **LCD** این پایه به زمین مدار **(GND)** متصل می گردد . **(Vcc)** پایه شماره ۲ : این پایه از **LCD** به مثبت ۵ ولت وصل می شود .

(CONT) پایه شماره ۳ : این پایه نیز به زمین مدار متصل می شود .

(Rs) پایه شماره ۴ : این پایه مخفف **Register select** می باشد بدین معنی که هرگاه بخواهیم به **LCD** دستور ارسال کنیم (مثلا دستور **CLS**) **Rs** برابر صفر قرار می گیرد و هرگاه بخواهیم اطلاعات به **LCD** ارسال کنیم **Rs** برابر یک قرار می گیرد . این پایه به یکی از پایه های میکرو متصل می گردد .

(R/W) پایه شماره ۵ : پایه **READ** و **WRITE** که با توجه اینکه از **LCD** می خوانیم یا روی آن می نویسیم مقدار دهی می شود . اگر از **LCD** بخوانیم این پایه باید برابر یک شود و اگر بخواهیم بر روی آن بنویسیم این پایه باید برابر صفر گردد . باید توجه داشت چون ما همواره برروی **LCD** می نویسیم این پایه برابر صفر قرار می گیرد .

(E) پایه شماره ۶ : این پایه پایه ی **ENABLE** یا فعال ساز **LCD** میباشد ، در صورتی که این پایه یک شود **LCD** فعال و در صورتی که صفر شود **LCD** غیرفعال است . این پایه به یکی از پایه های میکرو متصل می شود .

(DB0…DB7) پایه های ۷ تا ۱۱ : از این پایه ها برای فرستادن اطلاعات روی **LCD** استفاده می شود . برای فرستادن دیتا روی **LCD** می توان از مد هشت بیتی یا چهار بیتی استفاده کرد ، که اکثر مواقع از مد چهار بیتی استفاده می شود زیرا در این روش پایه های کمتری از میکرو اشغال می شود . در مد چهار بیتی از پایه های **DB4** تا **DB7** استفاده می شود که این پایه ها نیز به میکرو متصل می شوند .

(BACK LIGHT) پایه های شماره ۱۵ و ۱۶ : اگر از دو پایه استفاده شود لامپ زمینه **LCD** روشن می شود که این عمل به بهتر دیده شدن مطالب روی **LCD** کمک می کند . برای این کار پایه شماره ۱۵ به مثبت ۵ ولت و پایه شماره ۱۶ به زمین متصل می گردند .

* منظور از یک شدن پایه این است که این پایه به مثبت ۵ ولت وصل شود و منظور از صفر کردن یک پایه این است که آن پایه به زمین وصل گردد .*

تعیین نوع LCD

LCDTYPE می تواند انواع زیر باشد :
40*4 = دارای ۴۰ ستون و ۴ سطر
16*1a = دارای ۱۶ ستون و یک سطر است . این نوع LCD ، نوع ویژه ای است که به صورت LCD 2*8 استفاده می شود که دارای خط دومی در ستون نهم یا آدرس &H8 است .

16*2 = دارای ۱۶ ستون و ۲ سطر است که بصورت پیش فرض قرار می گیرد . اگر از این نوع LCD استفاده شود نیازی به تعیین نوع LCD نیست . ونیز متوان از انواع 16*4 ، 20*2 ، 20*4 ، 16*1 باشد .

پیکره بندی باس LCD

CONFIG LCDBUS = CONSTAN

در صورتی که بخواهیم از انتقال داده به LCD بصورت ۴ بیتی (پیش فرض) یا ۸ بیتی استفاده نماییم از این دستور استفاده می نماییم که CONSTANT می تواند عدد ۴ برای انتقال اطلاعات به صورت ۴ بیتی وعدد ۸ برای انتقال اطلاعات بصورت ۸ بیتی باشد . زمانی که از انتقال داده ۴ بیتی استفاده می نماییم نیازی به نوشتن این پیکره بندی نیست .

دستورات و توابع مربوط *LCD*

دستور *LCD* :

این دستور یک یا چند عبارت ثابت یا متغیر را بر روی ***LCD*** نمایش می دهد .

LCDx : **X** *متغیری است که نمایش داده می شود .*

LCDconstant : ***constant*** ثابتی است که نمایش داده می شود .

برای نمایش چند عبارت پشت سر هم بین آنها علامت (***semicolon***) را قرار می دهیم .

LCD a ; b1; "constant"

دستور *CLS* :

این دستور مخفف ***CLEAR SCREEN*** است که باعث می شود تمام صفحه نمایش ***LCD*** پاک شود .

دستور ***DISPLAY ON/OFF*** :

با این دستور می توانید صفحه نمایش را روشن یا خاموش کنید .

دستور *CURSOR* :

توسط این دستور می توان مکان نمای ***LCD*** را تنظیم کرد .

CURSOR ON/OFF BLINK NOBLINK

شما می توانید روشن (***ON***) یا خاموش (***OFF***) و چشمک زن (***BLINK***) یا چشمک نزدن مکان نما را تنظیم کنید .

در حالت پیش فرض مکان نما در حالت روشن و چشمک نزدن است .

دستور *HOME* :

این دستورات مکان نما زا به ترتیب در اولین ستون سطر اول ، سطر دوم ، سطر سوم یا سطر چهارم قرار می دهد .

HOME UPPER/LPWER/THIRD/FOURTH

دستورات فوق را به صورت ساده زیر نیز می توان نوشت .

HOMEU/L/T/F

اگر دستور ***HOME*** به تنهایی نوشته شود مکان نما در سطر و ستون اول قرار می گیرد دستور دستور ***LOCATE*** :

این دستور مکان نما را به مکان دلخواه در صفحه ***LCD*** می برد . ***X*** ثابت یا متغیری از (۴-۱) مشخص کننده سطر و ***Y*** ثابت یا متغیری از (۶۴-۱) که مشخص کننده ستون ***LCD*** است .

دستور *SHIFT CURSOR* :

این دستور مکان نمای ***LCD*** را یک واحد به چپ یا راست انتقال می دهد .

SHIFT CURSOR LEFT/RIGHT

دستور ***SHIFTLCD*** :

این دستور صفحه نمایش ***LCD*** را یک واحد به چپ یا راست انتقال می دهد .

دستور ***LOWERLINE*** :

این دستور مکان نما را به خط پایین تر می برد .

دستور ***UPPERLINE*** :

این دستور مکان نما را به خط بالاتر می برد .

دستور ***THIRDLINE*** :

این دستور مکان نما را به خط سوم می برد .

دستور ***FOURTH LINE*** :

در صرت استفاده از ***LCD*** چهار سطر این دستور کرزر را به اول خط چهارم می برد .

این دستور فقط برای ***LCD*** چهار معتبر است .

تابع ***DEELCDIHAR*** :

با این دستور می توانید حروف یا علامتی که خودتان در منوی ***TOOLS*** و قسمت ***LCD DESIGNER*** محیط ***BSCOM*** طراحی نموده اید بر روی صفحه ***LCD*** نمایش دهید . بعد از طراحی حرف یا علامت دلخواه در ***DESIGNER LCD*** و کلیک کردن بر روی دکمه ***OK*** خط زیر در محیط برنامه نویسی ظاهر خواهد شد .

DEFLCDCHAR ? , r1 , r2 , r3 , r4 , r5 , r6 , r7 , r8

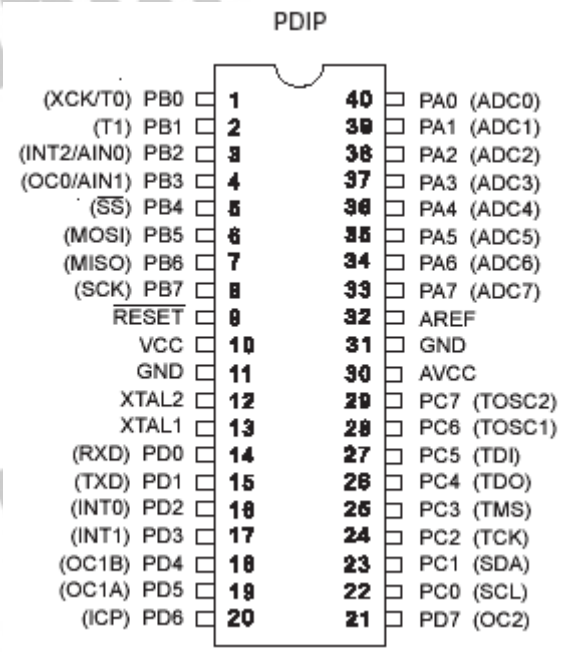
R1 تا ***R8*** باتوجه به طراحی توسط نرم افزار نوشته می شوند و شما می توانید به جای ***?*** عددی بین ۰ تا ۷ قرار دهید . بدین صورت شما می توانید تا ۸ کاراکتر و بر روی ***LCD*** نمایش دهید . نمایش کاراکتر طراحی شده توسط دستور (***?***) ***LCD CHR*** بعد از دستور***CLS*** انجام می گیرد.

برچسب کلید	شماره کلید	کد کاراکتر	برچسب کلید	شماره کلید	کد کاراکتر
ESC	76	C:200	w	1D	119
F1	05	C:1	e	24	101
F2	06	C:2	r	2D	114
F3	04	C:3	t	2C	116
F4	0C	C:4	y	35	121
F5	03	C:5	u	3C	117
F6	0B	C:6	i	43	105
F7	83	C:7	o	44	111
F8	0A	C:8	p	4D	112
F9	01	C:9	[	54	91
F10	09	C:10	]	5B	93
F11	78	C:11	Caps lock	58	C:15
F12	07	C:12	a	1C	97
`	0E	&H5E	s	1B	115
1	16	49	d	23	100
2	1E	50	f	2B	102
3	26	51	g	34	103
4	25	52	h	33	104
5	2E	53	j	3B	106
6	36	54	k	42	107
7	3D	55	l	4B	108
8	3E	56	;	4C	59
9	46	57	‘	52	C:16
0	45	48	ENTER	5A	C:17
-	4E	176	z	1A	122
+	55	43	x	22	120
Back Space	66	C:13	c	21	99
TAB	0D	C:14	v	2A	118
q	15	113	b	32	98

NORMAL KEYS LOWER CASE

برچسب کلید	شماره کلید	کد کاراکتر	برچسب کلید	شماره کلید	کد کاراکتر
c	21	99	5	73	53
v	2A	118	6	74	54
b	32	98	+	79	43
n	31	110	1	69	49
m	3A	109	2	72	50
,	41	44	3	7A	51
.	49	46	0	70	48
/	4A	47	.	71	46
CTRL	14	C:18	ENTER	5A	C:17
ALT	11	C:19	\	5D	96
SPACE	29	C:20			
INSERT	70	C:21			
HOME	6C	C:22			
UP	7D	C:23			
DOWN	7A	C:24			
END	69	C:25			
DELET	71	C:26			
↑	75	C:27			
←	6B	C:28			
↓	72	C:29			
→	74	C:30			
NUM LOCK	77	C:31			
/	4A	47			
*	7C	42			
-	7B	176			
7	6C	55			
8	75	56			
9	7D	57			
4	6B	52			

" شماتیک آی سی **ATMEGA16**"



نرم افزار و سخت افزار

حا فظه داخلی تلفن

همان طور که قبلاً گفته شد،این پروژه قادر به ذخیره کردن ده شماره تلفن میباشد که این شماره ها در خانه های ۰ تا ۹ کیبرد ذخیره می شوندو دارای یک LCD برای نشان دادن شماره ها می باشد.

عملکرد این دستگاه به این گونه است که ابتدا برای SAVE کردن شماره از "*"استفاده می نماییم،با زدن این کلید و سپس شماره خانه مورد نظر دوباره "*"را فشار داده و اکنون شماره تلفن خود را وارد می کنیم.باید توجه داشته باشید که اگر در این حین دچار اشتباه شوید کلید F4عمل CLEAR را انجام می دهد.برای پایان عملیات از کلید "#" استفاده می کنیم.در صورت تعویض شماره وارد شده ابتدا "#"که کار آن CHANGEاست را زده، شماره خانه مورد نظر، * و سپس شماره جدید را وارد کرده،در آخرکلید "#"را می زنیم.

کلید * به عنوان نشان دادن شماره ما نیز به کار می رود. به اینگونه که بعد از * یکی از کلید های ۰ تا ۹ را زده که شماره های ذخیره شده در آن خانه ها را برای ما نمایان می کند.

اگر قصد شماره گیری را داشته باشیم ابتدا با زدن F1و شماره کلید،شماره روی LCD به نمایش در می آید که با زدن F2دستگاه شروع به شماره گیری می کند.

در صورت منصرف شدن از کلید F3 می توان استفاده کرد.

پس:

*.....SAVE

#.....CHANGE

F1.....SHOW

F2.....CONEECT

F3.....LAPSE

f4.....CLEAR

نرم افزار

```
<include <mega32.h#
<include <delay.h#
<include <stdio.h#
<include <stdlib.h#
<include <string.h#
define array 32                    //array - 1  number will be saeved#
define time 400#
define epromtime 3#

asm#
equ __lcd_port = 0x15.
endasm#
<include <lcd.h#
-------------------------------------------------------------------------------//
------
-pragma warn#
;[eeprom unsigned int savenumbers[array][15
;[eeprom unsigned int counter[array
+pragma warn#
unsigned int Dtmf = 0;                 //work like a temp register
unsigned int a[20];                    //array of decoded number
;[unsigned int b[20
;[unsigned int newnumber[20//
;unsigned int number = 0
;static   int column
unsigned int i,j,m,n,L=1;//find=1;          //variables
unsigned char str[20];                 //lcd buffer
;[signed char str1[18],str2[18//
;bit flag = 0
;bit ok = 1//
-------------------------------------------------------------------------------//
------
unsigned int findnumber(unsigned int Dtmf);      //function
```

```
;(void firstring (void
void ringing (void);                    //function
;(void filter (void
;(void ninefilter (void
;(void whatnumber (void
;([]void lcdshow (unsigned int b
;([]void save (unsigned int
;(void readnumberplus(void
;(void readnumberminus(void
;(void eraseeeprom (void
;(void initialize (void
//-------------------------------------------------------------------------------
------
interrupt [TIM1_OVF] void timer1_ovf_isr(void) //Timer 1 overflow interrupt
service routine
}
TCNT1H=0x0B;        //0B Reinitialize Timer 1 value for 4 second
TCNT1L=0xDB;        //DB

;flag = 0
TIMSK = 0x00;       //Timer(s)/Counter(s) Interrupt(s) initialization
;()ninefilter
;()whoisringed//
{
//-------------------------------------------------------------------------------
------
(void main(void
}

;PORTA=0x00
;DDRA=0x00
;PORTB=0x00
;DDRB=0x00
;PORTC=0x00
;DDRC=0x00
;PORTD=0x00
;DDRD=0x00
```

```
TCCR1A=0x00;                // Timer/Counter 1 initialization
TCCR1B=0x05;                // Clock source: System Clock
TCNT1H=0x0B;                // Clock value: 15.625 kHz
TCNT1L=0xDB;                // Mode: Normal top=FFFFh
.ICR1H=0x00;                // OC1A output: Discon
.ICR1L=0x00;                // OC1B output: Discon
OCR1AH=0x00;                // Noise Canceler: Off
OCR1AL=0x00;                // Input Capture on Falling Edge
;OCR1BH=0x00
;OCR1BL=0x00
;MCUCR=0x00
;MCUCSR=0x00
TIMSK=0x00;                 // Timer(s)/Counter(s) Interrupt(s) initialization
;ACSR=0x80
;SFIOR=0x00

;(lcd_init(16

asm("sei")                  // Global enable interrupts#
;(lcd_gotoxy(4,0
;("lcd_putsf("CallerId
//-------------------------------------------------------------------------------
(while (1

}

if (PINA.1 == 1)            //if Est is high
}
;()firstring
;()ringing
{

(if (PIND.5 == 1
}
;()readnumberplus
{
```

```
(if (PIND.6 == 1
}
;()readnumberminus
{

(if (PINA.3 == 1
}
;()eraseeeprom
{

;{

{
-------------------------------------------------------------------------------------------//
------
(void ringing (void
}

if (PINA.1 == 1)                    //if Est is high
}

;(delay_ms(20
TIMSK = 0x04;                  //int of timer1 on

if (PINA.0 == 1)                   //if Strob is high

}

PORTA.2 = 1;                  //Enable TOE
Dtmf = PINB;                  //read portB
PORTA.2 = 0;                  //Disable TOE and high ampedance the
output of mt8870

a[i] = findnumber(Dtmf);          //Call find number function
filter();                     //Call filter function
```

```
;(save(a

[i++;                    //i for a[i
j++;                     //j for nine filter

:again
(if (PINA.0 == 1
;goto again

{

{

end of ringing function//{
----------------------------------------------------------------------------------//
(unsigned int findnumber(unsigned int Dtmf
}

;unsigned char pop

Dtmf &= 0b00001111;      //bit AND

;pop = Dtmf
pop ^=0b00001010;        //bit Xor
(if (pop == 0
;return 0

;pop = Dtmf
;pop ^=0b00000001
(if (pop == 0
;return 1

;pop = Dtmf
;pop ^=0b00000010
(if (pop == 0
;return 2
```

```
;pop = Dtmf
;pop ^=0b00000011
(if (pop == 0
;return 3

;pop = Dtmf
;pop ^=0b00000100
(if (pop == 0
;return 4

;pop = Dtmf
;pop ^=0b00000101
(if (pop == 0
;return 5

;pop = Dtmf
;pop ^=0b00000110
(if (pop == 0
;return 6

;pop = Dtmf
;pop ^=0b00000111
(if (pop == 0
;return 7

;pop = Dtmf
;pop ^=0b00001000
(if (pop == 0
;return 8

;pop = Dtmf
;pop ^=0b00001001
(if (pop == 0
;return 9

end of find number function//{
```

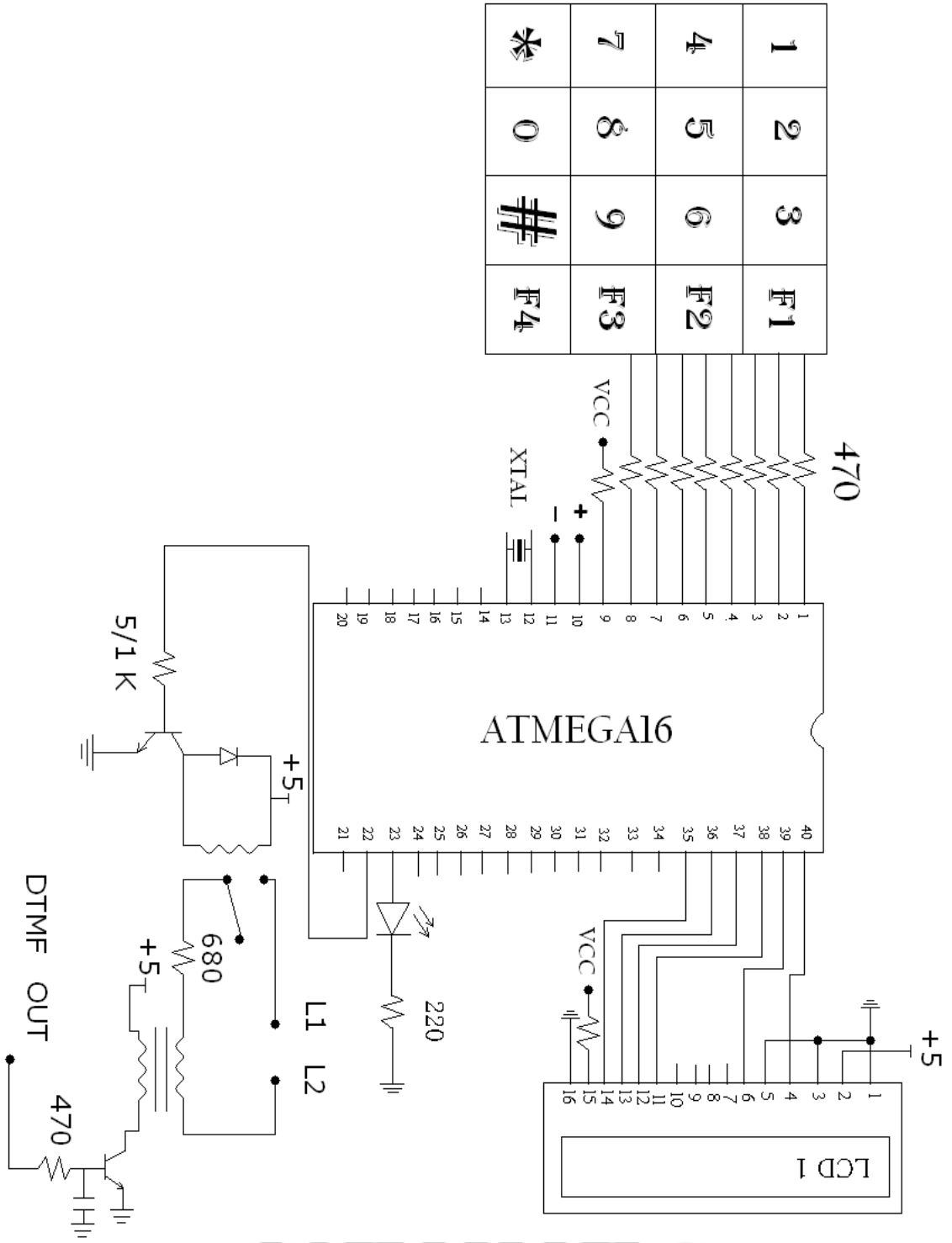
```
//-------------------------------------------------------------------------------
_____
void lcdshow (unsigned int b[]
{
lcd_gotoxy(column,1);
itoa(b[i],str);
lcd_puts(str);
column++;
}
//-------------------------------------------------------------------------------
_____
void firstring (void)
{
if (flag == 0)
{
flag = 1;

whatnumber();

i = 0;
j = 0;
column = 0;
{
{
//-------------------------------------------------------------------------------
_____
void whatnumber (void)
{
lcd_clear();
lcd_gotoxy(1,0);                //show the existing call number on lcd
itoa(number,str);               //itoa(number,str);
lcd_puts(str);
{
//-------------------------------------------------------------------------------
_____
```

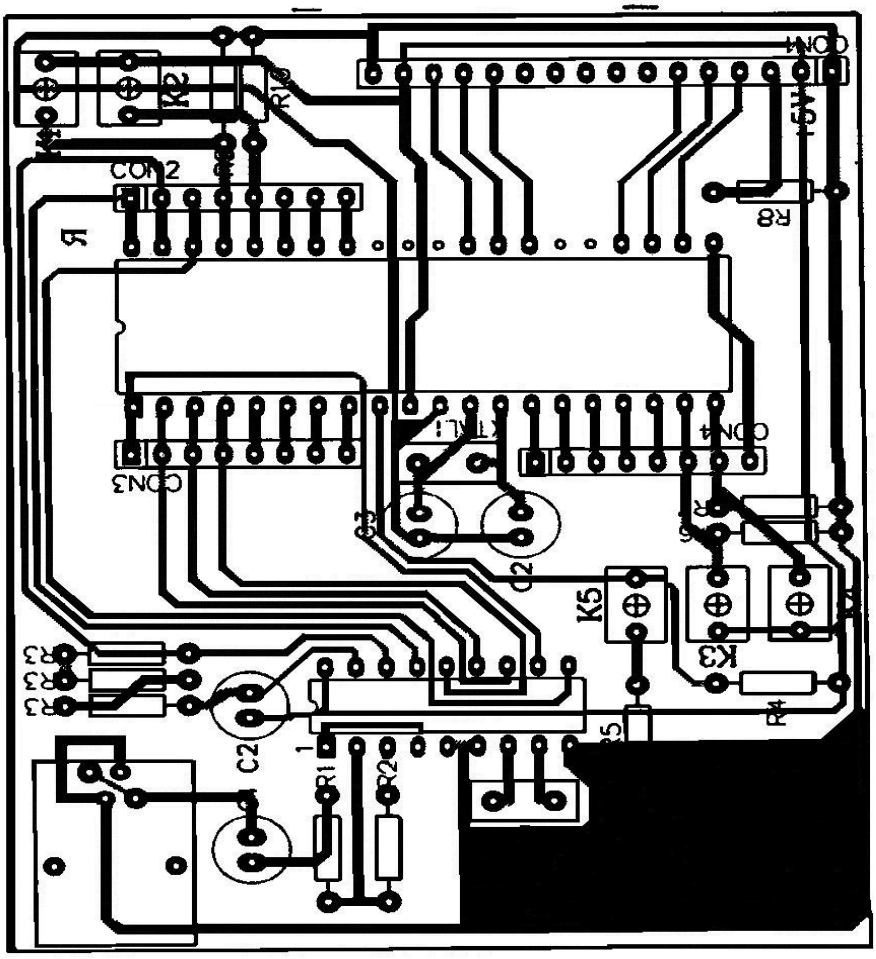
```
(void filter (void
}
(if (a[0]==9
}
number++;                //number of phone numbers
;ok = 0//
{
{
----------------------------------------------------------------------------//
_____
(void ninefilter (void
}
;--j
;(lcd_gotoxy(j,1
;(" ")lcd_putsf
{
----------------------------------------------------------------------------//
_____
([]void save (unsigned int a
}
(if (number <= array - 1
}
;[savenumbers[number][i] = a[i
;savenumbers[number][i+1] = 0x00
;[newnumber[i] = a[i//
;newnumber[i+1]=0x00//
;(delay_ms(epromtime
;counter[number] = i-1
;(delay_ms(epromtime
;[b[i] = a[i
;(lcdshow(b
{
else
}
;number = 1
```

```
;L = 1
;(save(a
{
{
-------------------------------------------------------------------------------//
------
(void readnumberplus(void
}
;()lcd_clear
;(lcd_gotoxy(4,0
;("lcd_putsf("CallerId

(if (L <= number
}

(++for (m = 0;m<=counter[L];m
}
;(lcd_gotoxy(m,1
;(itoa (savenumbers[L][m],str
;(lcd_puts(str
{

;(lcd_gotoxy(14,0
;(itoa (L,str
;(lcd_puts(str

;++L
;if (L >= array) L = array - 1
{

;(delay_ms(time
{
-------------------------------------------------------------------------------//
------
```

```
(void readnumberminus(void
}
;()lcd_clear
;(lcd_gotoxy(4,0
;("lcd_putsf("CallerId

;--L
;if (L <= 1) L=1

(++for (m = 0;m<=counter[L];m
}
;(lcd_gotoxy(m,1
;(itoa (savenumbers[L][m],str
;(lcd_puts(str
{

;(lcd_gotoxy(14,0
;(itoa (L,str
;(lcd_puts(str

;(lcd_gotoxy(9,0//
;(itoa(counter[L],str//
;(lcd_puts(str//

;(delay_ms(time
{
-------------------------------------------------------------------------------//
------
(void eraseeeprom (void
}
(++for (m = 0;m<=array-1;m
}
(++for (n = 0;n<=15;n
}
```

```
;savenumbers[L][m]=0x00
{
;counter[m]=0
{
;()lcd_clear
;()initialize
;i = 0
;j = 0
;L = 1
;number = 0
;flag = 0
{
//-------------------------------------------------------------------------------
------
(void initialize (void
}
;(lcd_gotoxy(4,0
;("lcd_putsf("CallerId
;(lcd_gotoxy(4,1
;("lcd_putsf("No Item
{
//-----------------------------------------------
```





Filename: Document1
Directory:
Template: C:\Documents and Settings\hadi tahaghoghi\Application Data\Microsoft\Templates\Normal.dotm
Title: □□□□□ □□□□ :
Subject:
Author: m
Keywords:
Comments:
Creation Date: 3/28/2012 4:45:00 PM
Change Number: 1
Last Saved On:
Last Saved By: H.H
Total Editing Time: 0 Minutes
Last Printed On: 3/28/2012 4:45:00 PM
As of Last Complete Printing
Number of Pages: 49
Number of Words: 6,268 (approx.)
Number of Characters: 35,730 (approx.)