پلتفرم اختصاصی
مهندسی کنترل

https://controlengineers.ir

https://t.me/controlengineers

https://www.instagram.com/controlengineers.ir

سرشناسه	:	طالبی، حسین، ۱۳۶۴-
عنوان و نام پدیدآور	:	پروژه های میکروکنترلر/نویسنده حسین طالبی.
مشخصات نشر	:	زنجان : تخت سلیمان، ۱۲۸۸.
مشخصات ظاهری	:	۴۷۴ص.:مصور، جدول: ۲۲×۲۹ س م.
شابک	:	978-964-8070-25-5
وضعیت فهرست نویسی	:	فیپا
یادداشت	:	ص.ع به انگلیسی: microcontroller HOSSEIN TALEBI. Project of AVR.
موضوع	:	میکروکنترلر ا. وی. ار. اتمل
موضوع	:	کنترل کننده های برنامه پذیر
رده بندی کنگره	:	TJ ۲۲۲/ ک ۹ط ۲ ۱۳۸۸
رده بندی دیویی	:	۶۲۹/۸۹۵
شماره کتابشناسی ملی	:	۱۸۵۰۸۱۲

مرکز پخش : زنجان – میدان انقلاب – پاساژ سعدی – انتشارات تخت سلیمان

تلفن : ۰۲۴۱۳۲۳۷۲۱۵

همراه مؤلف : ۰۹۱۲۷۴۳۸۹۷۶

همراه ناشر : ۰۹۱۲۷۴۰۸۶۶۱

عنوان کتاب ..... پروژه های میکروکنترلر

مؤلف ..... حسین طالبی

ناشر ..... تخت سلیمان

ناشر همکار ..... سها دانش (۰۹۱۲۱۲۶۱۴۱۹)

سال چاپ ۱۳۸۸

نوبت چاپ ..... اول

ناظر چاپ ..... مقصود قرایی

تیراژ ..... ۱۰۰۰ نسخه

قیمت ..... ۹۵۰۰۰ ریال

شابک : ۴-۶۲-۲۷۴۸-۹۶۴-۹۷۸ ISBN : 978-964-2748-62-4

تقدیم به ساحت مقدس آقا امام زمان (عجل ا... فرجه الشریف)

باشد که این حقیر را از دوستداران و منتظران خویش بدانند.

گفته بودم چو بیایی غم دل با تو بگویم

چه بگویم که غم از دل برود چون تو بیایی

مقدمه..... ۱۰

فصل اول

معرفی چهار میکروکنترلر برگزیده در این کتاب ۱۱

میکروکنترلرهای خانواده AVR ۱۳

میکروکنترلرهای برگزیده در این کتاب..... ۱۴

معرفی میکروکنترلر ATMEGA32 ..... ۱۴

ترکیب پایه های ATMEGA32 ..... ۱۶

بلوک دیاگرام داخلی ATMEGA32 ..... ۱۷

ساختار هسته مرکزی ATMEGA32 ..... ۱۸

حافظه FLASH ۱۹

فضای حافظه SRAM ۲۰

حافظه EEPROM ۲۱

واحد INTERRUPT UNIT ۲۱

ALU ۲۱

کلاک سیستم در ATMEGA32 ..... ۲۱

بلوک دیاگرام توزیع کلاک سیستم در ATMEGA32 ..... ۲۲

مدت زمان START UP ۲۳

معرفی منابع کلاک خارجی در ATMEGA32 ..... ۲۴

اسیلاتور خارجی کریستالی ۲۴

کریستال خارجی فرکانس پایین ۲۵

نوسان ساز RC داخلی ۲۶

نوسان ساز RC کالیبر شده داخلی ۲۶

کلاک خارجی ۲۷

منابع ریست میکروکنترلر ATMEGA32 ..... ۲۸

ریست POWER ON ۲۸

ریست خارجی ۲۸

ریست BROUN-OUT ۲۸

بلوک دیاگرام ریست ATMEGA32 ..... ۲۹

ریست WATCH DOG ۲۹

ریست JTAG ۲۹

رجیستر کنترل وضعیت (MCUCR) ۳۰

منابع وقفه در میکروکنترلر ATMEGA32 ..... ۳۰

رجیستر GIFR ۳۱

وقفه های خارجی ATMEGA32L ۳۲

حالتهای مختلف تحریک وقفه INT0 ..... ۳۲

حالتهای مختلف تحریک وقفه INT1 ..... ۳۳

رابط های سریال در میکروکنترلر های AVR ۳۳

معرفی رابط برنامه ریزی JTAG ۳۴

معرفی رابط سریال I2C ۳۵

مبدل آناکوگ به دیجیتال (ADC) داخلی میکرو..... ۳۶

خصوصیات مبدل آنالوگ به دیجیتال داخلی AVR ۳۶

مدیریت تغذیه و مدهای SLEEP ۳۶

فیوز بیت های میکروکنترلر ATMEGA32 ..... ۳۸

بررسی کاربرد پورت های ATMEGA32 ..... ۴۰

معرفی میکروکنترلر ATMEGA16 ..... ۴۶

ترکیب پایه های ATMEGA16 ..... ۴۸

منابع کلاک در میکروکنترلر ATMEGA16 ..... ۴۹

منابع ریست در میکروکنترلر ATMEGA16 ..... ۴۹

منابع وقفه در میکروکنترلر ATMEGA16 ..... ۴۹

وقفه های خارجی ATMEGA16 ..... ۵۰

مدیریت تغذیه و مدهای SLEEP در ATMEGA16 ..... ۵۰

فیوز بیت های میکروکنترلر ATMEGA16 ..... ۵۱

معرفی میکروکنترلر ATMEGA8 ..... ۵۲

ترکیب پایه های ATMEGA8 ..... ۵۴

منابع ریست در میکروکنترلر ATMEGA8 ..... ۵۵

منابع وقفه در میکروکنترلر ATMEGA8 ..... ۵۵

فیوز بیت های میکروکنترلر ATMEGA8 ..... ۵۶

خصوصیات AT90S2313 ..... ۵۷

ترکیب پایه های AT90S2313 ..... ۵۹

منابع ریست در AT90S2313 ..... ۵۹

کاربرد پورت های AT90S2313 ..... ۵۹

فیوز بیت های میکروکنترلر AT90S2313 ..... ۶۰

فصل دوم

معرفی دستورات و آموزش برنامه نویسی ، پیکره بندی و کار با امکانات AVR در محیط BASCOM ۶۳

معرفی محیط AVR SIULATOR ۶۵
آموزش برنامه نویسی در محیط BASCOM..... ۶۹
تعریف نام میکرو ۶۹
تعریف فرکانس کاری میکرو و نحوه
تنظیم فیوز بیت های مربوطه ۷۰
آدرس شروع نوشتن برنامه در
حافظه FLASH ۷۱
دستور END ۷۱
تعریف متغیر ۷۱
پیکره بندی پورت ها به صورت ورودی
و خروجی ۷۳
حلقه های FOR-NEXT..... ۷۵
دستورات تاخیر ۷۵
تفاوت دستورات تاخیر با تایمرها ۷۶
نحوه دریافت ورودی ۷۷
نحوه طراحی یک کلید ورودی به
صورت ACTIVE LOW ۷۷
دستورالعمل ALIAS..... ۷۹
حلقه DO-LOOP ۷۹
دستورالعمل IF ۷۹
نحوه خواندن مقادیر لحظه ای یا LATCH ۸۰
نحوه طراحی یک کلید ورودی به
صورت ACTIVE HIGH ۸۱
نحوه تعیین حساسیت ورودی به سطح یا لبه..... ۸۲
تنظیم حساسیت ورودی به لبه بالا رونده ۸۴
دستورالعمل BITWAIT ۸۵
حلقه WHILE-WEND ۸۷
تنظیم حساسیت ورودی به لبه پایین رونده ۸۸
محافظت ورودی در برابر نویز های لحظه ای ۹۱
نحوه ایجاد تاخیر در حین دریافت ورودی ۹۳
طراحی پروژه رقص نور میکروکنترلری ۹۵
دستور SELECT CASE ۱۰۱
زیر برنامه ونحوه استفاده از آن ۱۰۲
عملگرهای ریاضی در محیط BASCOM ۱۰۲
نحوه پیکره بندی صفحه کلید
4*4 (KEYPAD) ۱۰۵
دستور GETKBD() ۱۰۶
نحوه پیکره بندی LCD کاراکتری ۱۰۸
دستورات و توابع مربوط به LCD کاراکتری..... ۱۰۹
پروژه نمایش نام کلید فشرده شده بر روی LCD
کاراکتری ۱۱۰
جداول LOOKUP ۱۱۳
جداول LOOKUPSTR ۱۱۳
نحوه اطلاع از اعداد برگردانده شده توسط
KEYPAD ۱۱۵
پروژه تشخیص بزرگترین عدد دو رقمی وارد شده و
نمایش آن بر روی LCD ۱۱۷
دستور SOUND ۱۲۱
پروژه ماشین حساب ساده ۱۲۳
پروژه ای که دو عدد A و B را گرفته و بزرگترین مقسوم
علیه مشترک آنها را بر روی LCD نشان دهد ۱۲۹
دستور MOD ۱۳۴
دستور DEFLCDCHOR ۱۳۴
پروژه که یک معادله درجه دو را گرفته و ریشه های آن را
بر روی LCD نمایش دهد ۱۳۷
پروژه ای که ۱۰ عدد را گرفته بزرگترین عدد را به عنوان
MAX1 و کوچکتر از آن را به عنوان MAX2 بر روی
LCD نمایش دهد ۱۴۷
متغیر های آرایه ای ۱۵۲
نحوه پیکره بندی LCD گرافیکی ۱۵۳
نحوه نمایش تصویر بر روی LCD گرافیکی ۱۵۵
دستور SHOWPIC ۱۵۷
دستور $BGF ۱۵۷
نحوه نوشتن بر روی LCD گرافیکی ۱۵۹
نحوه پیکره بندی و استفاده از مبدل آنالوگ به
دیجیتال (ADC) داخلی AVR ۱۶۱
نحوه کار با وقفه ADC ۱۶۳
پروژه نمایش دما بر روی LCD گرافیکی با

فونت فارسی ۱۶۴
مشخصات سنسور دمای LM35 و ترتیب
پایه های آن ۱۶۶
دستورات EXIT ۱۷۱
دستور STR ۱۷۱
دستور LEN ۱۷۱
دستور MID ۱۷۲
دستور VAL ۱۷۲
تایمر/کانترها ۱۷۳
پیکره بندی تایمر/کانتر صفر به صورت
تایمر در محیط BASCOM ۱۷۴
پیکره بندی تایمر/کانتر صفر به صورت کانتر ۱۷۶
دستور TOGGLE ۱۷۸
طراحی پروژه فرکانس متر دیجیتال ۱۷۸
پیکره بندی تایمر/کانترها در مد مقایسه ای
(COMPARE) ۱۸۲
پیکره بندی تایمر 1 در مد مقایسه ای ۱۸۲
پیکره بندی کانتر 1 در مد مقایسه ای ۱۸۳
طرز کار با وقفه تطابق مقایسه در تایمر/کانتر 1 ..... ۱۸۳
پیکره بندی تایمر دو در مد مقایسه ای ۱۸۶
طرز کار با وقفه تطابق مقایسه در تایمر/کانتر 2 ..... ۱۸۷
پروژه تولید سیگنال PWM با استفاده از مد
مقایسه ای تایمر 2 ..... ۱۸۷
معرفی مد PWM و پیکره بندی تایمر/کانترها
در این مد ۱۹۱
نحوه ایجاد PWM آنالوگ ۱۹۲
نحوه پیکره بندی و ایجاد سیگنال PWM توسط
تایمر/کانتر 2 ..... ۱۹۴
پروژه ایجاد PWM هشت بیتی با ضریب چسبندگی
قابل تنظیم ۱۹۶
نحوه طراحی دمدولاتورPWM ۱۹۹
پروژه ارسال یک عدد با استفاده از مدولاسیون PWM و
دریافت آن توسط میکروکنترلر دمدولاتور ۲۰۰
پروژه ارسال و دریافت یک سیگنال سینوسی با
استفاده از مدولاسیون ۲۰۳
نحوه عملکرد تایمر/کانتر 1 در مد PWM ۲۰۷
پیکره بندی تایمر/کانتر 1 در مد PWM ۲۰۸
پروژه تولید PWM با دقت 10بیتی با خروجی
INVERTED و NONINVERTED ۲۰۸
ارتباط سریال ۲۱۰
پیکره بندی UART سخت افزاری برای
ارسال داده ۲۱۱
پیکره بندی UART سخت افزاری برای
دریافت داده ۲۱۱
پروژه نوشتن ، ارسال و دریافت پیام کوتاه از طریق
UART سخت افزاری ۲۱۲
پیکره بندی UART نرم افزاری ۲۲۰
نحوه استفاده از UART نرم افزاری ۲۲۱
ارتباط سریال SPI ۲۲۳
پیکره بندی سخت افزاری SPI ۲۲۴
دستورات و توابع مربوط به ارتباط SPI ۲۲۵
پروژه تبادل اطلاعات تمام دو طرفه
(FULL DUPLEX) ۲۲۶
پروژه ارسال اطلاعات به صورت یک طرفه از طریق
باس SPI ۲۲۸
کار با حافظه EEPROM داخلی AVR ۲۳۱
پروژه ذخیره پیام کوتاه در EEPROM
داخلی AVR ۲۳۱
وقفه های خارجی و نحوه پیکره بندی آن ها
در محیط BASCOM ۲۳۶
مقایسه کننده آنالوگ
(ANALOG COMPARATOR) ۲۳۷
پیکره بندی مقایسه کننده آنالوگ در محیط
BASCOM ۲۳۸
مدهای SLEEP ۲۳۹
دستورات اجرای مدهای SLEEP در
محیط BASCOM ۲۴۰
موارد استفاده از مدهای SLEEP ۲۴۱

فصل سوم

اطلاعات کاربردی ۲۴۳
سوئیچینگ با ترانزیستور ۲۴۵
نحوه تشخیص ناحیه اشباع ۲۴۶
محاسبه RB برای این که ترانزیستور در نواحی
قطع و اشباع کار کند ۲۴۸
نحوه طراحی درایور SPEAKER ونحوه طراحی ولوم
برای تغییر دامنه صوتی SPEAKER ۲۴۹
نحوه طراحی درایور SPEAKER با خروجی
INVERTED ۲۴۹
مدت زمان های ton و toff ترانزیستور ۲۵۲
مدار های پایه مربوط به LED ۲۵۳
نحوه سوئیچ کردن LED با استفاده از ترانزیستور ۲۵۵
مدار های مربوط به رله ۲۵۶
نحوه راه اندازی رله با استفاده از ترانزیستور ۲۵۷
رابط تحریک رله برای مدارهای دارای جریان
خروجی پایین ۲۵۸
منابع تغذیه DC ۲۵۹
مدار های تثبیت کننده ولتاژ توسط آی سی های
رگلاتور 3 پایه ۲۶۰
نحوه ساختن ولتاژ خروجی رگوله شده متغیر با
استفاده از آی سی های سری 78 و79 ..... ۲۶۲
نحوه ساختن ولتاژ گرفته شده با خروجی قابل
تنظیم با استفاده از LM317 ..... ۲۶۲
منبع تغذیه متقارن 12V ساده بدون استفاده از
ترانسفورماتور ۲۶۴
تولید ولتاژ مثبت و منفی توسط آی سی
LM386 ..... ۲۶۴
کلید ها و سلکتورهای انتخابگر دو طرفه ۲۶۵
نحوه بایاسینگ کلید های دو طرفه ۲۶۶
پروژه میکروکنترلری کنترل دیجیتالی ضریب تقویت
با استفاده از کلید های دو طرفه CMOS ۲۶۹
پروژه کنترل دیجیتالی ولتاژ برش در مدار برشگر
قله با استفاده از کلید های CMS ۲۷۳
نحوه طراحی مدار موج مربعی با استفاده از
آی سی NE555 ..... ۲۷۶
نحوه طراحی مولد موج سینوسی با
میکروکنترلر AVR ۲۷۷
پروژه تولید موج سینوسی با دامنه متغیر با استفاده از
میکروکنترلر AVR ۲۷۹
طراحی پروژه اهمتر دیجیتال با میکروکنترلر
AVR ۲۸۲
حذف پرش های ناخواسته جریان در مدارات
میکروکنترلری ۲۸۳
تکنیک های کاهش نویز ADC ۲۸۴
حفاظت اسیلاتور مدارات میکروکنترلری
در مقابل نویز ۲۸۵
مدیریت توان (صرفه جویی در توان مصرفی) در
میکروکنترلرهای AVR ۲۸۵

فصل چهارم

AVR در الکترونیک نوری ۲۸۹
نور و خواص آن در الکترونیک نوری ۲۹۱
نمایشگرهای هفت قسمتی یا
7-SEGMENT ۲۹۲
نحوه راه اندازی 7-SEGMENT ۲۹۳
طراحی پروژه نمایش نام کلید فشرده شده توسط
KEYPAD برروی 7-SEGMENT ۲۹۴
راه اندازی 7-SEGMENT های چند رقمی ۲۹۵
پاسخ معمول ترکیب مغز انسان با چشم او به
چشمک نوری ۲۹۵
طراحی یک مبدل باینری به 7-SEGMENT هشت
بیتی میکروکنترلری ۲۹۵
طراحی چراغ راهنمایی با استفاده از میکروکنترلر
AVR ۲۹۸
طراحی ساعت الکترونیکی با نمایشگر 7-SEGMENT
و با استفاده از کریستال ساعت 32.768Hz ۳۰۲
طراحی تابلو روان با استفاده از میکروکنترلر
AVR ۳۰۶

طراحی تابلو روان با ماتریس 8*16 ..... ۳۰۸
برنامه اجرای متن فارسی بر روی ماتریس
8*16 ..... ۳۱۱
برنامه اجرای انیمیشن بر روی ماتریس
8*16 ..... ۳۱۷
برنامه اجرای متن انگلیسی بر روی ماتریس
8*16 ..... ۳۱۹
برنامه اجرای متن فارسی ، متن انگلیسی و انیمیشن به
صورت پشت سرهم بر روی ماتریس 8*16 ..... ۳۲۳
نحوه افزایش ستون های تابلو روان ۳۲۶
طراحی تابلو روان با ماتریس 8*32 ..... ۳۲۸
برنامه نوشته شده برای اجرای متن فارسی بر روی
ماتریس 8*32 ..... ۳۲۸
طراحی تابلو روان با ماتریس 16*32 ..... ۳۳۲
برنامه اجرای متن فارسی بر روی ماتریس
16*32 ..... ۳۳۴
برنامه اجرای انیمیشن بر روی ماتریس
16*32 ..... ۳۳۷
برنامه اجرای متن انگلیسی بر روی ماتریس
16*32 ..... ۳۴۲
برنامه اجرای متن فارسی ، متن انگلیسی و انیمیشن در
تابلوی 16*32 ..... ۳۴۶
برنامه اجرای متن فارسی در تابلوی
16*64 ..... ۳۵۴
طراحی تابلو 16*32 با قابلیت وارد کردن متن
با کی پد ۳۵۸

فصل پنجم

AVR در ارتباطات WIRELESS ۳۷۱
WIRELESS چیست ۳۷۳
تشریح کامل یک ریموت کنترل رادیویی ۳۷۳
طراحی یک ریموت کنترل RF ، هشت
کاناله میکروکنترلری ۳۷۸
پروژه ارسال و دریافت اطلاعات توسط
ماژول های RF ۳۸۶
پروژه ارسال و دریافت پیام کوتاه با استفاده
از ماژول های RF ۳۹۶
طراحی یک ریموت کنترل مادون قرمز
میکروکنترلری ۴۱۰
پروژه ارسال و دریافت پیام کوتاه از طریق کانال ارتباطی
مادون قرمز ۴۱۷

فصل ششم

AVR در پروژه های حفاظتی و کنترل ۴۲۵
پروژه کنترل وسایل برقی توسط کامپیوتر ۴۲۷
طراحی قفل رمز دیجیتالی توسط میکروکنترلر
AVR ۴۳۲
پروژه کنترل سرعت موتور DC با استفاده از
سیگنال PWM ۴۴۲
تایمر میکروکنترلری دقیق با مدت زمان قابل تنظیم از یک
دقیقه تا 250 ساعت ۴۴۸
دماسنج دیجیتالی با قابلیت تحریک رله هنگام خارج شدن
از محدوده دمائی تعیین شده ۴۵۲
کنترل وسایل برقی با استفاده از خط تلفن
TELE REMOTE CONTROL ۴۵۶

فصل هفتم

AVR در مدارات و پروژه های صوتی ۴۶۳
ارگ الکترونیکی با استفاده از AVR ۴۶۵
طراحی تلفن داخلی (INTERCOM)
دوطرفه ۴۶۶
ارسال دیجیتالی سیگنال صوتی با استفاده از
مدولاسیون PWM ۴۶۸

ضمیمه۱

مقادیر استاندارد مقاومت و خازن ۴۷۴

پیشگفتار :

امروزه میکروکنترلرهای AVR به دلیل مصرف توان کم، زبان های برنامه نویسی متعدد، امکانات گسترده ، فناوری حافظه پیشرفته با ظرفیت بالا و بسیاری مزایای دیگر یکی از پر طرفدارترین میکروکنترلرها در میان هنرآموزان و دانشجویان الکترونیک و کامپیوتر می باشد. از این رو بنده در کتاب پروژه های میکروکنترلر AVR به آموزش پروژه های کاربردی پیاده سازی شده توسط میکروکنترلرهای AVR پرداخته ام. در پروژه های این کتاب از چهار میکروکنترلر خانواده AVR با نامهای AT90S2313، ATMEGA8، ATMEGA16 ، ATMEGA32 و برای برنامه نویسی آن ها از کامپایلر BASCOMAVR استفاده شده است به همین دلیل در فصل اول به معرفی کامل میکروکنترلرهای برگزیده پرداخته، سپس در فصل دوم برنامه نویسی، پیکره بندی و کار با امکانات AVR در محیط BASCOM با متدی کاملا جدید (در قالب چندین پروژه کاربردی) آموزش داده می شود. پس از آن در فصل سوم با ارائه اطلاعات میکروکنترلری و غیر میکروکنترلری پیش نیاز از قبیل طراحی رگلاتورهای ولتاژ، نحوه استفاده از ترانزیستور در نواحی قطع و اشباع در پروژه های سویچینگ، نحوه راه اندازی انواع رله توسط ترانزیستور و میکروکنترلر و... خواننده را آماده طراحی پروژه های پیچیده تر کرده سپس در فصل های بعدی کاربردی ترین پروژه های الکترونیک در زمینه های الکترونیک نوری، ارتباطات WIRELESS، پروژه های حفاظتی و کنترل و پروژه های صوتی ارائه می شود. خوانندگان عزیز توجه داشته باشید که پروژه های این کتاب در صورت امکان با سیمولاتور پروتئوس و در غیر این صورت به صورت عملی تست شده و عاری از هر گونه اشغال می باشند. از خوانندگان عزیز تقاضا دارم اگر مواردی یافتید که قابل بحث بیشتر بود و یا مطالب جدیدتر را حتما با بنده در میان بگذارید هر گونه پروژه های عملی و ابتکارات مربوط به میکروکنترلرهای AVR توسط دوستانی که برایم ارسال شوند اگر جالب و برجسته باشند در نشرهای بعدی این کتاب با نام خودشان چاپ خواهد شد. امیدوارم شرایط این امکان را برایم بوجود بیاورد که بتوانم پروژه های جدیدتری را در جلد های بعدی کتاب به محضر شریفتان ارائه کنم.

در پایان شایسته است از برادر عزیزم محمد که زحمت تایپ،صفحه آرایی و طراحی جلد کتاب را بر عهده داشت ، از پدر ، مادر و خواهر عزیزم که همواره مشوق من بوده اند ، از آقای مقصود قرایی مدیریت انتشارات تخت سلیمان به خاطر همکاری صمیمانه ، از دوست عزیزم آقای سعید نظری خمارکی به خاطر طراحی نرم افزار LED DISPLAY CONVERTER و همچنین از اساتید بزرگوارم آقای مهندس علی بدرخانی مدیر گروه و مدرس الکترونیک در دانشکده فنی الغدیر زنجان و آقای مهندس بیاتی مدرس الکترونیک در دانشکده فنی الغدیر زنجان و همچنین آقای مهندس سعید جنگی بهادر مدرس الکترونیک و عضو هیئت علمی در دانشگاه آزاد اسلامی واحد میانه کمال تشکر و قدردانی را داشته باشم.

آدرس اینترنتی : HOSSEIN.ELECTOR_AYSUN@YAHOO.COM

فصل اول

معرفی چهار میکروکنترلر برگزیده در این کتاب

بسمه تعالی

پیشرفت علم الکترونیک در طراحی و ساخت مدارهای مجتمع در طول چند دهه اخیر، منجر به ساخت قطعات دیجیتالی پیشرفته و میکرو پروسسورها شده است. هنگام استفاده میکروپروسسورها ، با توجه به ساختار و معماری آنها ، به کارگیری قطعات جانبی نظیر PIO وحافظه های ROM ، RAM و تجهیزات دیگر ، امری اجتناب ناپذیر است.همچنین در بعضی از شرایط فرکانس کاری میکروپروسسورها نیاز کاربر را تامین نمی کند.از این رو پس از میکروپروسسورها، میکروکنترلرها که قطعات جانبی مورد نیاز را در خود جای داده و سرعت کار آنها نسبت به میکروپروسسورها بیشتر است ، به بازار عرضه شده اند. اما میکروکنترلرهایی نظیر خانواده 8X51 که بیشتر مورد استفاده قرار میگیرند ، دارای کمبودهایی بودند که از آن جمله می توان به عدم وجود WATCHDOG ، نداشتن پروتکل سریالTWO WIRE (I2C) و عدم تنوع زیاد این خانواده از میکرو کنترلرها اشاره نمود. ازآنجایی که در کاربردهای صنعتی ، کاربر تمایل دارد میکروکنترلرها را بر حسب نیاز خود انتخاب نمایدو از طرف دیگر در محیطهای پرنویز صنعتی امکان هنگ کردن میکروکنترلر وجود دارد و میکروکنترلرهای خانواده X851 امکان WATCHDOG ویک سری امکانات دیگر را ندارد ، این خانواده از میکروکنترلرها در حال جایگزین شدن بامیکرو کنترلرهای خانواده AVR و PIC هستند.

میکرو کنترلرهایAVR علاوه بر اینکه امکانات تمامی میکروکنترلرهای قبلی را دارا هستند مشکلات آن ها را رفع کرده و امکانات جالبی را نیز فراهم کرده اند.

میکرو کنترلرهای خانواده AVR

مصرف توان کم ، زبان های برنامه نویسی متعدد شامل زبان های سطح پایین و هم چنین سطح بالا (HIGH LEVEL LANGUAGE) امکانات گسترده فناوری حافظه پیشرفته با ظرفیت بالا ، دستورالعمل های قوی و توانایی اجرای آن ها در یک سیکل ساعت و توانایی های دیگر ، میکروکنترلرهای خانواده AVR را به یکی از پر طرفدارترین میکروکنترلر ها تبدیل کرده است .

این میکروکنترلرها دارای امکانات گسترده ای از جمله 32 رجیستر 8 بیتی همه منظوره (GENERAL PURPOSE) حافظه فلش داخلی قابل برنامه ریزی ، حافظه EEPROM داخلی با قابلیت 100000 بار نوشتن و پاک کردن (ERASE , WRITE) و حافظه RAM داخلی ، چندین تایمر/کانتر ، وقفه های سخت افزاری و نرم افزاری متعدد ، ارتباط سریال SPI ، I2C (2WIRE) ، JTAG ، استفاده از تکنولوژی CMOS و معماری RICS (کم کردن دستورالعمل های میکروکنترلرها) در ساختار آن ها و ... می باشند .

میکرو کنترلر های برگزیده AVR در این کتاب

در طراحی پروژه های این کتاب از 4 میکروکنترلر خانواده AVR استفاده شده است که عبارتند از AT90S2313، ATMEGA8 ،ATMEGA16 ، ATMEGA32 ، که در ادامه به معرفی این میکرو کنترلرهای ساخت شرکت اتمل می پردازیم.

این میکروکنترلرهای هشت بیتی دارای توان مصرفی پایینی بوده، در معماری آن ها از ساختار پیشرفته RICS (کم کردن دستورالعملهای میکروکنترلرها) بهره گرفته شده است. به عبارت دیگر این میکرو کنترلرها دارای 118تا 131 دستورالعمل ساده هستند که اغلب آنها در یک سیکل ساعت اجرا می شود.یعنی مدت زمان اجرای هر دستور برابر با (فرکانس کاری میکرو/1) خواهد بود ،اجرا شدن دستورالعملها در یک سیکل باعث افزایش سرعت این میکروکنترلرها گردیده است، همچنین ATMEGA16،ATMEGA32 وAT90S2313,ATMEGA8 دارای سی ودو رجیستر همه منظوره هشت بیتی اند و در صورت استفاده از نوسان ساز خارجی شانزده مگا هرتز و قابلیت اجرای حداکثر شانزده میلیون دستورالعمل در ثانیه را دارند. این قابلیت یکی دیگر از دلایل افزایش سرعت این میکرو کنترلر هاست.

به دلیل این که همه امکانات خانواده AVR در میکروکنترلر نمونه ATMEGA32 وجود دارد ابتدا به معرفی کامل آن می پردازیم.

معرفی میکرو کنترلر ATMEGA32

ATMEGA32 از سری میکرو کنترلر های MEGA AVR می باشد، میکروکنترلرهای MEGA نسبت به انواع دیگر (AT90S,TINY) دارای امکانات و قابلیت های بیشتری هستند.

خصوصیات ATMEGA32L,ATMEGA32

یک میکروکنترلر 8 بیتی که از معماری AVR RISC (کاهش دستورالعمل های میکرو کنترلر) استفاده می کند.

- کارایی بالا و توان مصرفی کم
- دارای 131 دستورالعمل با کارایی بالا که اکثرآنها در یک کلاک سیکل اجرا می شوند
- 8 * 32 رجیستر کاربردی
- سرعتی تا 16MIPS در فرکانس 16MHZ
- 32K بایت حافظه FLASH داخلی قابل برنامه ریزی.

پایداری حافظه FLASH :

قابلیت 10,000 بار نوشتن و پاک کردن(ERASE,WRITE) .

- 2K بایت حافظه داخلی SRAM

پایداری حافظه EEPROM :

قابلیت 100,000 بار نوشتن و پاک کردن(ERASE,WRITE).

– قفل برنامه FLASH و حفاظت داده EEPROM هنگام ERASE کردن میکروکنترلر.

قابلیت ارتباط JTAG (IEEE STD)

– برنامه ریزی برنامه FUSE BITS,EEPROM,FLASH و LOCK BITS از طریق ازتباط JTAG

خصوصیات جانبی (TIMER/COUNTER)

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

–یک تایمر/کانتر(TIMER/COUNTER) 16 بیتی با PRESCALER مجزا و دارای

مدهایCAPTURE , COMPARE

– چهار کانال PWM (PULSE WIDTH MODULATOR)

– هشت کانالA/D, (مبدل آنالوگ به دیجیتال) 10 بیتی

– یک مقایسه کننده آنالوگ داخلی(INTERNAL ANALIG COMPARATOR) .

– دارای RTC (REAL – TIME CLOCK) با اسیلاتور داخلی و توانایی ایجاد REAL TIME با استفاده از کریستال ساعت خارجی 32.768KHz .

– WATCHDOG قابل برنامه ریزی بااسیلاتور داخلی.

– ارتباط سریال SPI برای برنامه ریزی داخل مدار (IN SYSTEM PROGRAMMING) .

– قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE .

– قابلیت ارتباط با پروتکل سریال دو سیمه (TWO – WIRE)

– USART سریال قابل برنامه ریزی

خصوصیات ویژه میکروکنترلر

– POWER – ON RESET CIRCUIT .

– BROWN – OUT DETECTION قابل برنامه ریزی

– دارای شش حالت اسلیپ (EXTENDED STANDBY, STANDBY , POWER – SAVE

IDLE , POWER – DOWN و ADC NOISE REDUCTION)

– منابع وقفه (INTERRUPT) داخلی و خارجی.

– دارای اسیلاتور RC داخلی کالیبره شده

- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

ولتاژهای کاری

-2.7V تا 5.5V برای (ATMEGA32L)

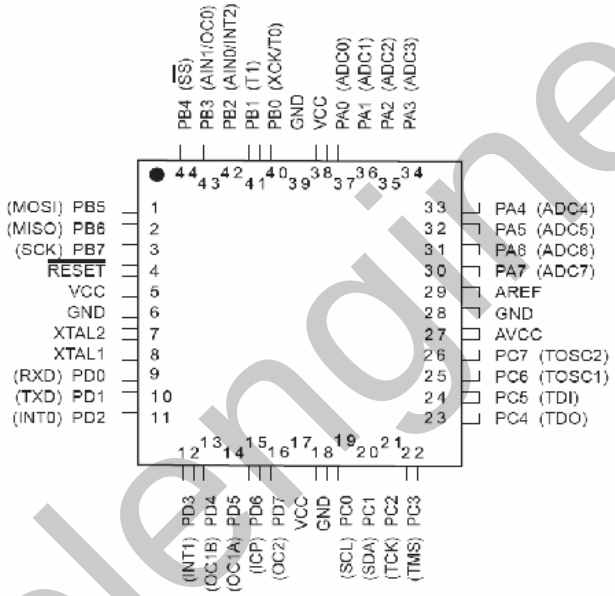
- 4.5V تا 5.5V برای (ATMEGA32)

خصوصیات I/O و انواع بسته بندی

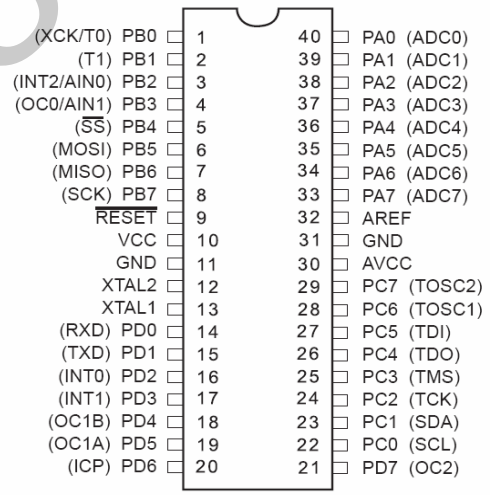
- 32 خط ورودی ، خروجی I/O قابل برنامه ریزی .

- در بسته بندی های 40 پایه PDIP , 44 پایه TQFP و 44 پایه MLF

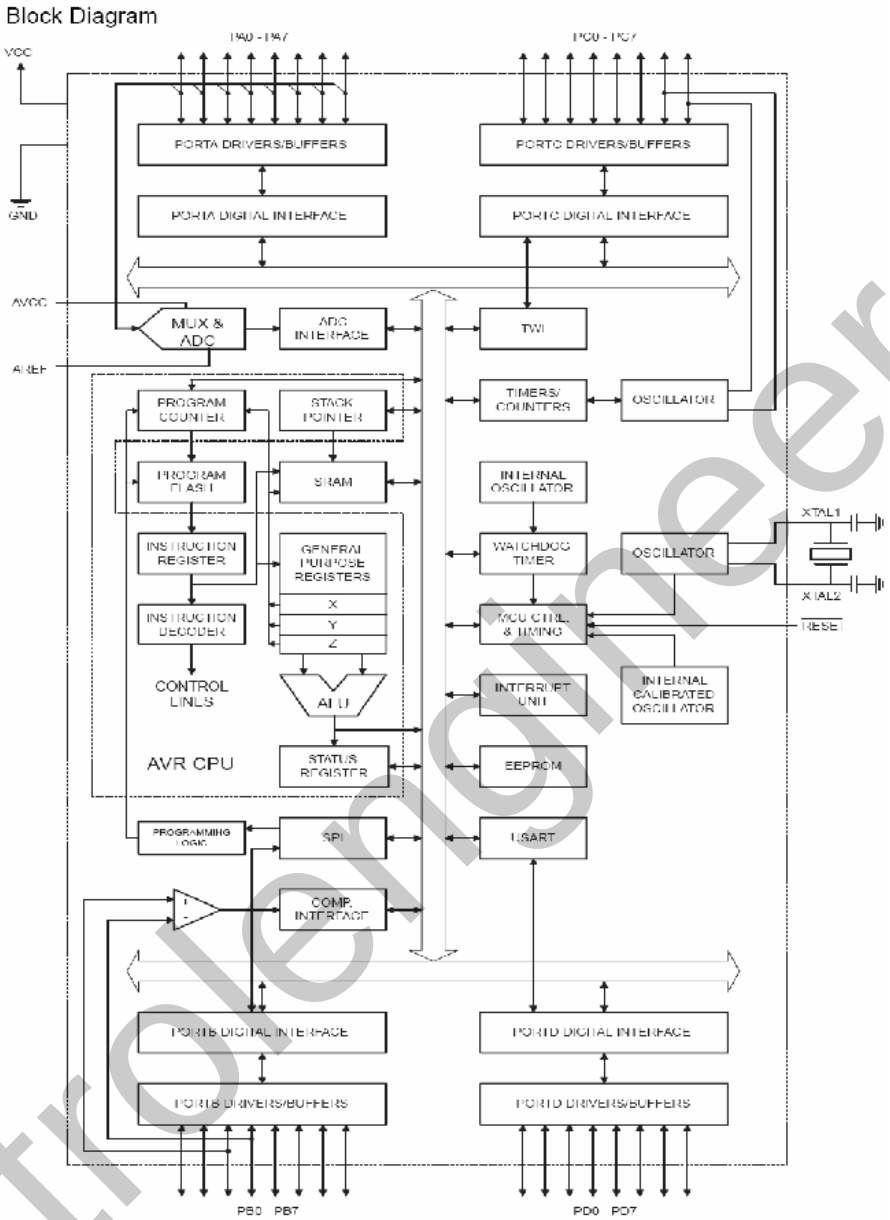
ترکیب پایه های میکروکنترلر ATMEGA32 (PIN CONFIGURATION)



PDIP



بلوک دیاگرام داخلی میکرو کنترلر ATMEGA32



شکل 1-1 بلوک دیاگرام داخلی میکروکنترلر در ATMEGA 32

کارکردن قسمتهای مختلف نشان داده در شکل **1-1** به مدهای مختلف SLEEP بستگی دارد . چنانچه قبلا بیان شد ATMEGA32 شش مد SLEEP دارد. در فصل دوم کتاب مدهای SLEEP و نحوه استفاده از آنها به صورت کامل بررسی شده است در مد IDLE، CPU متوقف شده ولی رابط سریال USART ، ADC ،

TIMER/COUNTER ، SRAM ، I2C (TWI) ، رابط SPI و سسیستم وقفه به کار خود ادامه می دهند.

در مد POWER – DOWN مقادیر رجیسترها حفظ شده ، اسیلاتور خارجی و بعضی از عملیات میکرو تا رسیدن وقفه بعدی یا RESET سخت افزاری متوقف می شود. مد POWER – SAVE مشابه مد POWER DOWN است ولی در این مد شمارنده آسنکرون به کار خود ادامه می دهد .

منظور از شمارنده آسنکرون تایمر/کانتر دو یا صفر می باشد در صورتی که پالس کلاک خود را به صورت آسنکرون با پالس کلاک میکرو و سنکرون با کریستال ساعت خارجی 32.768KHz دریافت کند .

مدهای STANDBY و EXTENDED – STANDBY نیز مشابه مد POWER DOWN هستند با این تفاوت که در این مد ها اسیلاتور میکروکنترلر با نوسان ساز کریستالی یا رزونانسی خارجی کار میکند. در مد ADC NOISE REDUCTION ، CPU به منظور مینیمم کردن نویز در هنگام تبدیل ADC متوقف می شود ولی قسمتهایی مانند شمارنده آسنکرون و مبدل ADC فعال هستند.

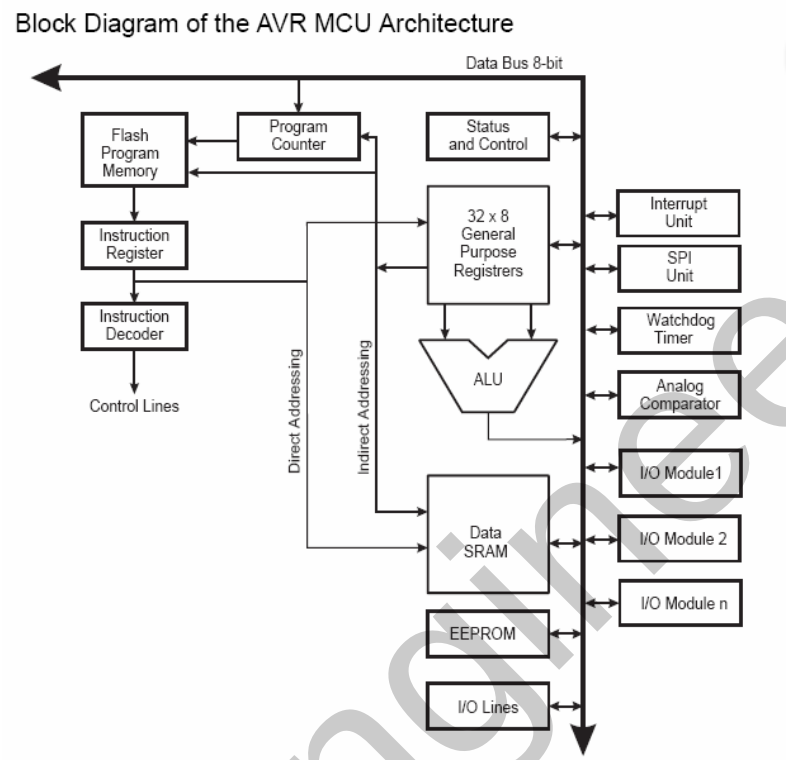
ساختار هسته مرکزی ATMEGA32

در این بخش ساختار MCU که وظیفه آن انجام دادن دستورالعملها، ارتباط با حافظه، انجام محاسبات ریاضی و منطقی، کنترل تجهیزات جانبی و پاسخ دادن به وقفه هاست ، به صورت اجمالی تشریح می شود.شکل 1-2 بلوک دیاگرام قسمت اصلی یا MCU میکروکنترلرهای AVR را نشان می دهد.

ATMEGA32 یک میکروکنترلر CMOS هشت بیتی توان پایین می باشد که بر اساس ساختار RISC طراحی شده است. با اجرای دستورالعملهای ساده در یک سیکل ساعت سرعت این میکرو بسیار افزایش یافته است.در میکروکنترلر های AVR اجرای دستورالعملها در یک سیکل ، واکشی یک دستور از حافظه و اجرای دستور قبلی در یک پالس ساعت انجام می شود. یا به عبارت ساده تر هنگام اجرای یک دستور در یک پالس ، دستور بعدی در حال واکشی شدن می باشد . واکشی یک دستورالعمل و اجرای همزمان دستورالعمل دیگر، یکی از دلایلی است که سرعت میکروکنترلرهای AVR را افزایش داده است.

چنان چه در شکل 1-2 دیده می شود سی و دو رجیستر هشت بیتی به صورت مستقیم با ALU در ارتباط هستند. که این ویژگی نیز منجر به افزایش سرعت میکروکنترلر می شود.در یک کار معمولی ALU ، دو عملوند به رجیستر ها فرا خوانده شده و عمل مربوط بر روی آن ها در یک پالس ساعت انجام و نتیجه در یک رجیستر ذخیره می شود. شش عدد از این رجیستر ها دو به دو با هم ، به صورت سه رجیستر 16 بیتی عمل می کنند ، که

اغلب ازآن ها برای آدرس دهی DATA استفاده می شود این رجیسترها با نامهای X , Y و Z شناخته می شوند.



شکل 2-1 بلوک دیاگرام MCU در میکروکنترلر های AVR

حافظه FLASH :

حافظه FLASH برای ذخیره کدهای برنامه به کار می رود. قابلیت ایجاد امنیت برای کدهای برنامه ذخیره شده در حافظه FLASH یکی از مزیت های میکرو کنترلر های AVR است. حافظه های FLASH حافظه هایی هستند که اطلا عات آن ها با قطع تغذیه از بین نمی رود این نوع حافظه ها در میکروکنترلر های AVR محلی برای ذخیره کد های برنامه بوده و به دو قسمت APPLICATION وBOOTLOADER تقسیم می شوند میکرو کنترلرهای AVR با استفاده از حافظه BOOTLOADER دارای قابلیت خود برنامه ریزی بوده و می توانند با اجرای بر نامه BOOTLOADER برنامه کاربردی را که در قسمت APPLICATION نوشته شده است تغییر دهند.

برنامه BOOTLOADER برنامه ای است که در قسمت BOOTLOADER حافظه FLASH ذخیره شده و قادر است با دنیای خارج از طریق پورت های میکروکنترلر و رابط هایی نظیر RS232 ، USART ، I2C و SPI ارتباط برقرار نماید . میکروکنترلر با استفاده از این برنامه می تواند بدون استفاده از PROGRAMER کدها یا دیتاهای مورد نیاز را از دنیای خارج گرفته و محتوای قسمت

APPLICATION یا حتی خود BOOTLOADER را نیز تغییر دهد. برای درک این موضوع روند برنامه ریزی میکروکنترلر بررسی می شود. هنگامی که میکروکنترلر برای اولین بار برنامه ریزی می شود، رابطی که PROGRAMER نامیده می شود. برای مثال STK 200 / 300 بین میکروکنترلر و کامپیوتر قرار می گیرد . سپس کد های برنامه نوشته شده با استفاده از یک نرم افزار (به عنوان مثال BASCOM AVR) و PROGRAMER درون حافظه FLASH نوشته می شود.اگر این برنامه ، یک برنامه کاربردی باشد ، در قسمت APPLICATION و اگر برنامه BOOTLOADER باشد ، در قسمت BOOTLOADER ذخیره می شود. حال اگر زمانی نیاز به تغییر برنامه کاربردی باشد و برنامه BOOTLOADER وجود نداشته باشد ، باید میکروکنترلر از مدار مربوطه جدا شده ، به PROGRAMER متصل و این روند تکرار شود. ولی استفاده از برنامه BOOTLOADER کاربر را از جدا کردن میکروکنترلر و اتصال آن به PROGRAMER بی نیاز کرده و بسته به نظر کاربر می تواند از رابط های مختلف استفاده کند. در محیط BASCOM می توان با دستور JMP $BOOTADDRESS در هر جای برنامه APPLACATIN به برنامه BOOTLOADER پرش کرد ویا در صورت برنامه ریزی فیوز بیت BOOTRST می توان با استفاده از فیوز بیت های BOOTSZ ، آدرس بردار ریست را روی آدرس شروع برنامه BOOTLOADER تنظیم کرد تا پس از ریست میکرو برنامه BOOTLOADER اجرا شود و در برنامه BOOTLOADER می توان با استفاده از پورت های I/O و رابط های سریال مقدار اولیه برخی از متغیر ها را تعیین نموده سپس با استفاده از دستور JMP $0000 اجرا ی برنامه را به قسمت APPLICATION منتقل کرد ، یاد آوری میکنم که در قسمت حافظه BOOT فقط قادر به نوشتن برنامه به صورت ASSEMBELY هستیم . فیوز بیت های مربوطه در قسمت معرفی فیوز بیت ها به صورت کامل بررسی خواهد شد . البته در هیچ یک از پروژه های این کتاب از حافظه BOOTLOADER استفاده نشده است .

فضای حافظه SRAM

حافظه SRAM میکرو کنترلرATMEGA32 به چند قسمت تقسیم شده است. سی و دو بایت اول فضای حافظه به رجیسترهای عمومی R0 تا R31 اختصاص داده شده است و آدرس های بعدی به رجیسترهای کنترلی قسمتهای مختلف نظیر شمارنده ها ، رابط SPI و غیره اختصاص یافته است که این رجیستر ها تحت عنوان I/O REGISTER شناخته شده و به صورت مستقیم یا غیر مستقیم آدرس دهی می شوند با توجه به اختصاص سی و دو خانه اول به رجیسترهای عمومی ، اولین آدرس در رجیستر های بخش دوم $20 خواهد بود. پشته نیزدر حافظه SRAM قرار دارد و نحوه افزایش آن از آدرسهای بالا به پایین است.البته در BASCOM معمولا ما به طور مستقیم با رجیستر های I/O یا رجیستر های R0 تاR31 کار نخواهیم کرد و برای ذخیره داده به جای

رجیستر های عمومی از متغیر هایی که خودمان در برنامه تعریف کرده ایم استفاده خواهیم نمود ولی توجه داشته باشید که کامپایلر برای ذخیره محتوای متغیر هایی که مادر برنامه تعریف کرده ایم از رجیستر های عمومی استفاده می کند .

حافظه EEPROM :

حافظه EEPROM یکی دیگر از قسمت های مهم میکروکنترلرهای AVR است. با توجه به این که اطلاعات ذخیره شده در EEPROM با قطع تغذیه میکروکنترلر از بین نمی رود، از این نوع حافظه برای ذخیره اطلاعات همیشه ماندنی استفاده می شود.

واحد INTERRUPT UNIT :

این قسمت مدیریت وقفه را بر عهده دارد . برای پذیرفته شدن وقفه های مختلف در میکروکنترلرهای AVR بیت I که تحت عنوان بیت فعال ساز عمومی وقفه (GLOBAL INTERRUPT ENABLE BIT) نامیده می شود ودر رجیستر SREG قرار دارد، باید یک شود. . البته در BASCOM برای فعال کردن وقفه عمومی به جای یک کردن بیت I در رجیستر SREG از دستور ENABLE INTERRUPTS استفاده می شود که با نوشتن این دستور کاپایلر به صورت خودکار بیت I از رجیستر SREG را یک خواهد کرد .

ALU :

ALU یکی دیگراز قسمتهای مهم هسته مرکزی میکروکنترلر AVR است.ALU به طور مستقیم با رجیسترهای R0 – R31 در ارتباط بوده و قادر به انجام سه نوع عملیات ریاضی، منطقی وبیتی است.ALU قادر است در مدت یک سیکل یک عملیات ریاضی را بر روی محتویات دو رجیستر یا یک رجیستر و یا یک مقدار ثابت انجام دهد.

کلاک سیستم در ATMEGA32

کلاک CLKcpu – CPU :

این کلاک برای فعال کردن قسمت AVR CPU در شکل 1-1 مورد استفاده قرار می گیرد . وظیفه CPU انجام عملیات AVR و عملیات مربوط به رجیسترها می باشد. توقف و به مکث بردن این کلاک باعث می شود که عملیات و محاسبات AVR انجام نگیرد. با قطع این کلاک توسط مد های SLEEP ، CPU به حالت SHOT DOWN می رود .

کلاک CLK I/O_I/O :

این کلاک توسط بسیاری از ماژولهای I/O به طور مثال کانترها، SPI و USART استفاده می گردد و با قطع این کلاک توسط مد های SLEEP تمام قسمت های I/O به حالت SHOT DOWN می رود .

کلاک CLK FLASH – FLASH :

این کلاک عملیات ارتباطی با حافظه FLASH را کنترل می کند.کلاک FLASH معمولا با کلاک CPU فعال می شود.و با قطع این کلاک توسط مد های SLEEP حافظه FLASH و CPU به حالت SHOT DOWN می رود.

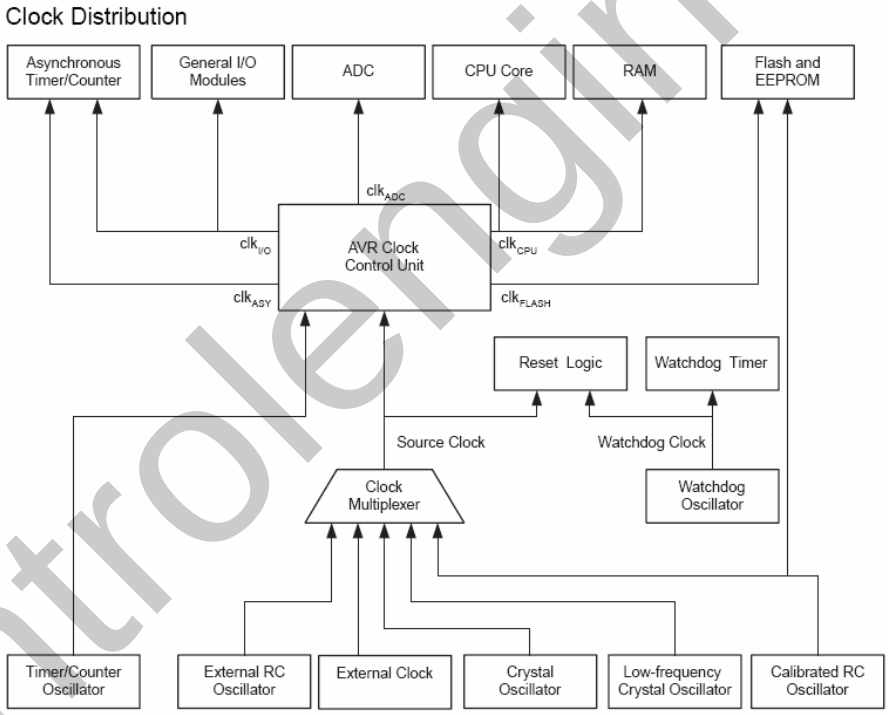
کلاک غیر همزمان تایمر - CLK ASY :

با این کلاک تایمر / کانتر 2 یا 0 به صورت آسنکرون با کلاک سیستم و سنکرون با کریستال ساعت 32768 Hz کار می کند حتی اگر سیستم در حالت SLEEP باشد.

کلاک CLKADC – ADC :

ADC یا مبدل آنالوگ به دیجیتال داخلی از یک کلاک جداگانه حساس استفاده می کند که باعث می شود کلاک های CPU و I/O به حالت ایست (HALT) رفته تا نویز حاصل از مدار دیجیتال داخلی کاهش یافته ودر نتیجه عملیات تبدیل با دقت بیشتری انجام گیرد .

بلوک دیاگرام توزیع کلاک سیستم میکرو در ATMEGA32



شکل 1-3 بلوک دیاگرام توزیع کلاک سیستم میکرو

همان طور که در شکل 1-3 مشاهده می کنید منابع کلاک این میکروکنترلر عبارتند از :

۱- نوسان ساز RC کالیبره شده داخلی

۲- کریستال خارجی فرکانس پایین (معمولا 32.768 khz)

۳- کریستال خارجی یا رزوناتور (RESONATOR) سرامیکی

۴- کلاک خارجی

۵- اسیلاتور RC خارجی

۶- نوسان ساز کریستالی خارجی برای تایمر های 8 بیتی (تایمر مد 0 و2) که به کریستال ساعت نیز معروف است و مقدارآن (32.768 KHz) می باشد.(معمولا از این حالت برای ایجاد زمان های واقعی استفاده می شود)

۷- نوسان ساز داخلی WATCHDOG

منابع کلاک ATMEGA32 را می توان بوسیله بیت های قابل برنامه ریزی FLASH , FLASH FUSE) BITS) انتخاب کرد کلاک انتخاب شده به عنوان ورودی کلاک AVR طبق جدول زیر در نظر گرفته شده و کلاک مناسب به هر قسمت ارائه می شود.

توجه شود در تمام فیوز بیت ها (0) به معنای بیت برنامه ریزی شده (PROGRAMED) و (1) به معنای بیت برنامه ریزی نشده (UNPROGRAMED) می باشد.

DEVICE CLOCKING OPTIONS SELECT

DEVICE CLOCKING OPTION	CKSEL3..0
EXTERNAL CRYSTAL/CERAMIC RESONATOR کریستال یا رزوناتور خارجی	1111 -1010
EXTERNAL LOW-FREQUENCY CRYSTAL کریستال فرکانس پائین خارجی	1001
EXTERNAL RC OSCILLATOR اسیلاتور خارجی	1000 – 0101
CALIBRATED INTERNAL RC OSCILLATOR اسیلاتور کالیبره شده داخلی	0100 – 0001
EXTERNAL CLOCK کلاک خارجی	0000

NOTE: 1.FOR ALL FUSES "1"MEANS UNPROGRAMMED WHILE "0" MEANS PROGRAMED

جدول انتخاب انواع کلاک سیستم

تعریف مدت زمان STRART – UP :

وقتی که CPU از مداسلیپ POWER – DOWN یا POWER – SAVE به حالت WAKE UP (بیدار شدن از حالت SLEEP) بر میگردد پالس انتخاب شده توسط جدول بالا مدت زمان مشخصی به میکرو اعمال نمی شود، این زمان که تحت عنوان زمان START – UP شناخته می شود، به منظورپایدار شدن پالس ، قبل از اجرای دستورالعمل بعدی است، وقتی CPU از حالتRESET بر می گردد زمان تاخیر بیشتری به منظور ثابت شدن تغذیه و پایدار شدن پالس ساعت مورد نیاز است. برای ایجاد زمان بندی های START_UP مذکور از اسیلاتور WATCHDOG استفاده می شود .

اسیلاتور WATCHDOG برای تنظیم زمان START – UP مطابق جدول زیر به کار می رود، چنانچه دیده می شود زمان وفرکانس کاری WATCHDOG به ولتاژ تغذیه وابسته است.

Number of Watchdog Oscillator Cycles

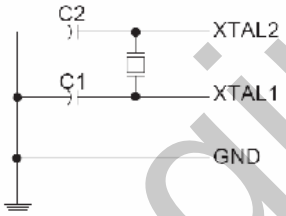
Typ Time-out (V_{CC} = 5.0V)	Typ Time-out (V_{CC} = 3.0V)	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

جدول زمان های START_UP نمونه

معرفی منابع کلاک خارجی MEGA32

اسیلاتور خارجی کریستالی(EXTERNAL CRYSTAL , CERAMIC RESONATOR) :

در این حالت یک کریستال کوارتز یا رزوناتور سرامیکی مانند شکل 1-4 به پایه های XTAL1 وXTAL2 متصل می شود.



Crystal Oscillator Connections

شکل 1-4 نحوه اتصال کریستال یا رزوناتور سرامیکی

در جدول زیر مدهای کاری کریستال خارجی ارائه شده اند نوسان ساز می تواند در 3 مد مختلف کار کند که هر کدام برای یک محدوده فرکانسی بهینه شده اند.

Crystal Oscillator Operating Modes

CKOPT	CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101(1)	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.

یادآور می شوم که گزینه اول فقط برای نوسانگر سرامیکی استفاده می شود و نباید آن را برا ی نوسانگر کریستالی به کار برد در جدول فوق مد کاری توسط فیوزبیت های CKSEL1..3 و CKOPT تعیین می شود. خازن های C1وC2 در هر دو حالت استفاده از کریستال یا رزوناتور دارای مقادیر متفاوتی هستند. در جدول بالا برخی از مقادیر نمونه برای انتخاب خازن C1 و C2 ارائه شده اند.

در این حالت مدت زمان (START–UP) توسط CKSEL0 و SUT0 و SUT1 تنظیم می شود برای تنظیم START-UP می توان از جدول زیر استفاده کرد.

Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK(1)	4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK(1)	65 ms	Ceramic resonator, slowly rising power
0	10	1K CK(2)	–	Ceramic resonator, BOD enabled
0	11	1K CK(2)	4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK(2)	65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	–	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

جدول انتخاب مدت زمان START_UP برای اسیلاتور کریستالی

(1)–این گزینه ها زمانی که سیستم در فرکانس های بالا کار نمی کند استفاده می گردد.انتخاب این گزینه ها برای کریستال مناسب نیست.

(2)– این گزینه ها برای نوسان گرهای سرامیکی استفاده می شود، همچنین می توان برای کریستال ها زمانی که در فرکانسهای پایین کار می کنند استفاده شود.

کریستال خارجی فرکانس پایین :

با برنامه ریزی فیوزبیتهای CKSEL به مقدار 1001 ، می توان از یک نوسان ساز کریستال فرکانس پایین 32768Hz (کریستال ساعت) برای فراهم کردن کلاک سیستم استفاده کرد. با برنامه ریزیCKOPT توسط کاربر، خازن های داخلی به کریستال متصل می شوند و نیاز به خازن های خارجی بر طرف می گردد. مقدار اسمی خازن های درونی 36PF است.

هنگامی که این نوع کریستال انتخاب می شود مدت زمان START– UP توسط فیوزبیت هایSUT طبق جدول زیر قابل انتخاب است.

Start-up Times for the Low-frequency Crystal Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	1K CK(1)	4.1 ms	Fast rising power or BOD enabled
01	1K CK(1)	65 ms	Slowly rising power
10	32K CK	65 ms	Stable frequency at start-up
11	Reserved		

Note: 1. These options should only be used if frequency stability at start-up is not important for the application.

جدول تنظیم مدت زمان START_UP برای اسیلاتور کریستالی فرکانس پایین

نوسان ساز RC خارجی

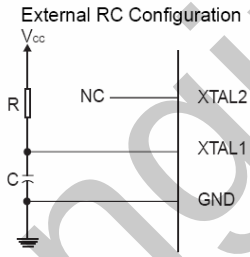
این حالت در کاربردهایی که دقت بالایی در آنها مد نظر نیست، به کار می رود. فرکانس تقریبی از رابطه F=1/(3RC) به دست می آید.با برنامه ریزی CKOPT، نیاز به خازن خارجی برطرف می شود.در این حالت نوسان ساز در 4 مد کاری متناسب با محدوده های فرکانسی بهینه کار می کند.در جدول زیر مد های کاری نوسان ساز RC خارجی ارائه شده اند.

External RC Oscillator Operating Modes

CKSEL3..0	Frequency Range (MHz)
0101	≤ 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0

جدول مد های عملیاتی اسیلاتور RC خارجی

نحوه اتصال اسیلاتور RC خارجی در شکل 1-5 ارائه شده است .



شکل 1-5 نحوه اتصال اسیلاتور RC خارجی

هنگامی که فرکانس کاری انتخاب می شود زمان شروع (START– UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است.

Start-up Times for the External RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	18 CK	–	BOD enabled
01	18 CK	4.1 ms	Fast rising power
10	18 CK	65 ms	Slowly rising power
11	6 CK(1)	4.1 ms	Fast rising power or BOD enabled

Note: 1. This option should not be used when operating close to the maximum frequency of the device.

جدول انتخاب زمان START_UP برای اسیلاتور RC خارجی

(1)– این گزینه زمانی که میکرو در فرکانسهای بالا کار می کند نباید انتخاب گردد.

نوسان ساز RC کالیبره شده داخلی (CALIBRATED INTERNAL RC OSCILLATOR) :

Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	Nominal Frequency (MHz)
0001(1)	1.0
0010	2.0
0011	4.0
0100	8.0

Note: 1. The device is shipped with this option selected.

جدول انتخاب اسیلاتور RC کالیبره شده داخلی

(1)-برای میکروکنترلر به صورت پیش فرض انتخاب شده است.

پس از انتخاب فرکانس کاری زمان شروع (START– UP) توسط فیوز بیت های SUT طبق جدول زیر قابل انتخاب است.

Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10(1)	6 CK	65 ms	Slowly rising power
11	Reserved		

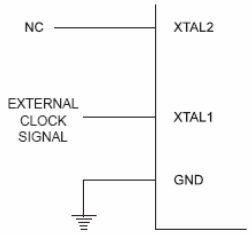
Note: 1. The device is shipped with this option selected.

جدول تنظیم زمان START_UP برای اسیلاتور RC داخلی

(۱)- برای میکروکنترلر بصورت پیش فرض انتخاب شده است.

کلاک خارجی

یک کلاک خارجی که به پایه XTAL متصل می شود، به عنوان منبع کلاک مورد استفاده قرار گیرد. در این حالت CKSEL3..0 برابر 0000 قرار داده می شود. با برنامه ریزی فیوزبیت CKOPT نیز یک خازن 36PF داخلی بین پایه XTAL1 و زمین قرار می گیرد.



External Clock Drive Configuration

شکل 6-1 نحوه اتصال کلاک خارجی به میکرو

هنگامی که این نوع کلاک انتخاب شود زمان شروع START-UP توسط فیوزبیت های SUT طبق جدول زیر قابل انتخاب است.

Start-up Times for the External Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10	6 CK	65 ms	Slowly rising power
11	Reserved		

جدول تنظیم مدت زمان START_UP برای کلاک خارجی

در این مد بایستی از تغییرات ناگهانی فرکانس کلاک (بیشتر از 20٪) جلوگیری کرد، زیرا ممکن است باعث رفتار های غیر قابل انتظار میکرو شود زمانی که قصد تغییر فرکانس کلاک را دارید ، بایستی میکرو در حالت RESET نگه داشته شود.

منابع ریست میکروکنترلر ATMEGA32

میکرو کنترلر ATMEGA32 دارای منابع ریست POWER ON ، ریست سخت افزاری خارجی ، ریست BROWN OUT ، ریست WACTHDOG و ریست JTAG می باشد.که در ادامه به معرفی آن ها می پردازیم.

ریست POWER ON :

یک مدارآشکار کننده داخلی، پالس ریست POWER–ON را هرگاه که سطح ولتاژ VCC از ولتاژآستانه مدارآشکار ساز پایین تر باشد تولید می کند ، هنگام وصل شدن تغذیه به میکروکنترلر ، این منبع ریست در لحظه روشن شدن باعث ریست تراشه خواهد شد.درحقیقت سیستم در موقع روشن شدن به طور خود کار ریست POWER–ON می شود.

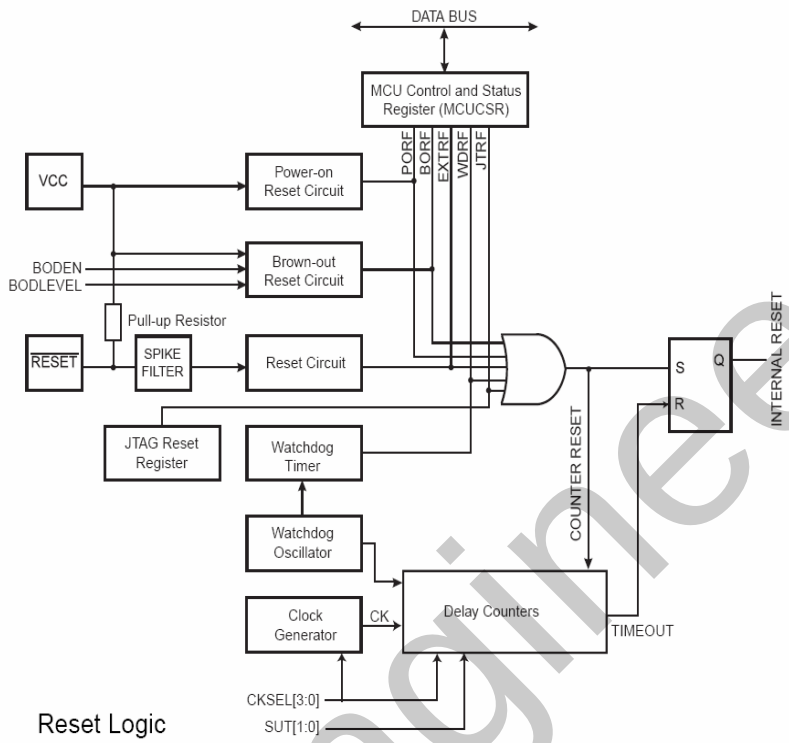
ریست خارجی:

ریست خارجی با LOW شدن پایه RESET اتفاق می افتد،حداقل پهنای پالس لازم برای ریست شدن تراشه 2.5US میباشد.

ریست BROWN–OUT :

میکرو کنترلر MEGA32 دارای یک مدار آشکار ساز BROWN–OUT می باشد این مدار در هر لحظه سطح ولتاژ VCC را برسی می کند و همواره آن را با یک سطح ولتاژ مرجع مقایسه می نماید.در صورتی که VCC از ولتاژ ریفرنس (مرجع) کمتر شود عمل ریست را انجام می دهد ، انتخاب سطح ولتاژ مرجع توسط فیوز بیت های BODLEVEL و BODEN طبق جدول مربوطه انجام می گیرد.جدول مربوط به تعیین ولتاژ مرجع در قسمت فیوز بیت ها به طور کامل بررسی شده است .

بلوک دیاگرام ریست میکرو کنترلر ATMEGA32



شکل 1-7 بلوک دیاگرام RESET میکرو

ریست WATCHDOG :

هنگامی که تایمر WATCHDOG سرریز کند یک پالس کوتاه به پهنای یک ماشین سایکل تولید میشود در لبه پایین رونده این پالس تایمر WATCHDOG شروع به شمارش بازه زمانی START_UP می کند که در این مدت میکرو در حالت RESET قرار میگیرد با اتمام این زمان میکروکنترلر به حالت عادی خود بر می گردد. تایمر WATCHDOG را می توان در محیط برنامه نویسی BASCOM به صورت نرم افزاری مقدار دهی کرد.

ریست JTAG :

رابط JTAG یکی از ماژول های جانبی ATMEGA32 است که اغلب برای پروگرم کردن میکروکنترلر با استفاده از سخت افزار های خاص نظیر پروگرمر STK500 به کار می رود. رابطJTAG یک رجیستر RESET دارد که پس از اجرای دستور مربوطه مقدار این رجیستر یک شده و میکرو کنترلر راRESET می کند.

رجیستر کنترل وضعیت (MCUCR)

پس از اعمال پالس های ریست از سوی منابع مختلف ریست، یک پرچم از رجیستر MCUCR برسی می شود ، این رجیستر نشان می دهد که کدام یک از منابع RESET منجربه این کار شده اند.

The MCU Control and Status Register provides information on which reset source caused an MCU Reset.

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

رجیستر کنترل وضعیت (MCUCR)

بیت **4** : هنگامی که ریست JTAG رخ داده باشد این بیت یک می شود، این بیت با ریست POWER-ON یا نوشتن صفر به صورت نرم افزاری پاک می شود.

بیت **3** : هنگامی که ریست WATCHDOG رخ داده باشد ، این بیت یک می شود، این بیت با ریست POWER–ON یا نوشتن صفر به صورت نرم افزاری پاک می شود.

بیت **2** : هنگامی که ریست BROWN-OUT رخ داده باشد این بیت یک می شود این بیت ریست POWER –ON یا نوشتن صفر به صورت نرم افزاری پاک می شود.

بیت **1** : پرچم ریست خارجی

بیت **0** : پرچم ریست POWER– ON

توجه داشته باشید که برای تشخیص شرایط ریست ، باید رجیستر MCUSR هر بار پس از باز خوانی پاک شود.

منابع وقفه در میکرو کنترلر ATMEGA32

تراشه MEGA32 دارای 21 منبع وقفه است که همگی در جدول زیر ارائه شده اند. که اکثریت آن ها به عنوان وقفه داخلی و تعدادی هم به عنوان وقفه خارجی مورد استفاده قرار می گیرند . جدول زیر منابع ایجاد کننده وقفه را در میکرو کنترلر ATMEGA32 نشان می دهد.

Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	INT2	External Interrupt Request 2
5	$008	TIMER2 COMP	Timer/Counter2 Compare Match
6	$00A	TIMER2 OVF	Timer/Counter2 Overflow
7	$00C	TIMER1 CAPT	Timer/Counter1 Capture Event
8	$00E	TIMER1 COMPA	Timer/Counter1 Compare Match A
9	$010	TIMER1 COMPB	Timer/Counter1 Compare Match B
10	$012	TIMER1 OVF	Timer/Counter1 Overflow
11	$014	TIMER0 COMP	Timer/Counter0 Compare Match
12	$016	TIMER0 OVF	Timer/Counter0 Overflow
13	$018	SPI, STC	Serial Transfer Complete
14	$01A	USART, RXC	USART, Rx Complete
15	$01C	USART, UDRE	USART Data Register Empty
16	$01E	USART, TXC	USART, Tx Complete
17	$020	ADC	ADC Conversion Complete
18	$022	EE_RDY	EEPROM Ready
19	$024	ANA_COMP	Analog Comparator
20	$026	TWI	Two-wire Serial Interface
21	$028	SPM_RDY	Store Program Memory Ready

جدول منابع وقفه در میکروکنترلر ATMEGA32

در میکروکنترلرهای خانواده AVR اولویت وقفه ها به صورت یک سطحی است یعنی در صورت رخ دادن همزمان دو وقفه ابتدا به وقفه ای که آدرس پایین تری در جدول بالا دارد پاسخ داده می شود سپس با توجه به ردیف اولویت طبق جدول بالا به وقفه بعدی پاسخ داده می شود.

برای فعال کردن وقفه ها قبل از هر کاری باید فعال ساز عمومی در رجیستر(SREG) را برنامه ریزی کنیم، با رخ دادن هر وقفه این بیت صفر می شود و وقفه های دیگر همگی غیر فعال خواهند شد. در زمان بازگشت از زیر برنامه وقفه (ISR) این بیت دوباره یک خواهد شد. یک وصفر کردن این بیت در BASCOM توسط دستورات DISSABLE INTERRUPTS , ENABLE INTERRUTS صورت می گیرد.

رجیستر GIFR :

در این رجیستر پس از اعمال پالس تریگر (پالس تحریک) به هر یک از وقفه های خارجی بیت مربوط به آن وقفه یک می شود و رخداد وقفه را گزارش می کند پس از اجرای زیر برنامه (ISR) دوباره صفر می شود. نکته مهم این که اگر بیت فعال ساز عمومی وقفه ها یک نشده باشد با آمدن هر وقفه تنها پرچم مربوط به آن در رجیستر

GIFR یک می شود ، اما وقفه ای اتفاق نمی افتد. در BASCOM فعال کردن وقفه ها توسط دستورات زیر صورت می گیرد .

ENABLE INTERRUPTS
ENABLE INT X

وقفه های خارجی ATMEGA32 :

وقفه های خارجی توسط پایه های INT0 ، INT1 و INT2 تریگر می شوند.وقفه های INT0 و INT1 با لبه بالا رونده ، پایین رونده یا با سطح پایین تریگر می شوند ولی وقفه INT2 فقط به لبه حساس است. تعیین حساسیت وقفه های خارجی به لبه یا سطح در رجیستر های MCUCSR و MCUCR انجام می شود. وقتی یک وقفه خارجی به صورت حساس به سطح (فقط INT0 و INT1) تنظیم می شود وقفه تا زمان پایین بودن سیگنال فعال خواهد بود .

برای فعال کردن این وقفه ها باید در رجیستر کنترلی GICR بیت مربوط به آن وقفه را یک کنیم. در تراشه ATMEGA32 از خانواده AVR ، سه وقفه خارجی INT0 , INT1 و INT2 وجود دارند. جدول زیر رجیستر کنترلی GICR را نشان می دهد.

General Interrupt Control Register – GICR

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

نوع تحریک هر یک از وقفه های INT0 و INT1 توسط چهار بیت کم ارزش (نیبل پایینی) رجیستر MCUCR تعیین می شود. در شکل زیر این رجیستر نشان داده شده است.

The MCU Control Register contains control bits for interrupt sense control and general MCU functions.

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

در زیر حالتهای مختلف تحریک وقفه خارجی INT0 با توجه به بیت های رجیستر MCUCR و در جدول بعدی حالتهای مختلف تحریک وقفه خارجی INT1 با توجه به بیت های رجیستر MCUCR ارائه شده اند.

حالتهای مختلف تحریک وقفه INT0 :

Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

حالتهای مختلف تحریک وقفه INT1 :

Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

وقفه INT2 در میکرو کنترلر AVR ، تنها به لبه پایین رونده یا بالا رونده حساس است. با نوشتن یک در بیت ISC2 از رجیستر MCUCSR ، وقفه INT2 در حالت بالا رونده پیکره بندی می شود و با نوشتن صفر در بیت ISC2 وقفه INT2 در حالت پایین رونده پیکره بندی می شود .

در هر حال ، برای تمام وقفه های خارجی حداقل عرض پالس باید50ns باشد.برای تعیین نحوه تریگر شدن وقفه خارجی در BASCOM از دستور CONFIG INTX=RISING/FALLING/LOW LEVEL استفاده می شود . به طور کلی در برنامه نویسی های سطح بالا از کار کردن مستقیم با رجیستر ها صرفه نظر می شود .

رابط های سریال در میکروکنترلر های AVR

به طور کلی ارتباط سریال می تواند به دو صورت سنکرون و آسنکرون انجام پذیرید ، در ارتباط سریال آسنکرون پالس ساعت (به منظور MATCH کردن فرستنده و گیرنده از نظر زمانبندی) فرستاده نمی شود و به جای آن از START BIT و STOP BIT برای ایجاد هم زمانی بین فرستنده و گیرنده استفاده می شود . ولی در ارتباط سریال سنکرون علاوه بر DATA پالس ساعت هم به منظور ایجاد هم زمانی بین فرستنده و گیرنده فرستاده می شود البته در فصل دوم کتاب به صورت مفصل به این موضوع خواهیم پرداخت . لازم به ذکر است که در پروژه های این کتاب از ارتباط سریال دو طرفه SPI به عنوان ارتباط سریال سنکرون و از ارتباط سریال UART (در حالت آسنکرون) به عنوان ارتباط سریال آسنکرون استفاده شده است .

به طور کلی رابط های سریال میکروکنترلر های AVR عبارتند از رابط برنامه ریزی JTAG ، رابط SPI ، رابط UART و رابط I2C ، با توجه به این که رابط های سریال UART و SPI در فصل دوم بررسی خواهند شد در این فصل فقط به معرفی را بط های JTAG وI2C می پردازیم.

معرفی رابط برنامه ریزی JTAG

رابط JTAG یکی دیگر از ماژولهای جانبی میکرو کنترلر های AVR است که اغلب برای PROGRAM کردن میکروکنترلر با استفاده از سخت افزارهای خاصی نظیر STK500 به کار می رود. رابط برنامه ریزی JTAG رابطی است مطابق با استاندارد IEEE STD.1149 که دارای قابلیت هایی به شرح زیر است.

- دارای قابلیت عیب یابی (DEBUGGING) کلیه قسمت های داخلی شامل RAM داخلی و خارجی ، رجیستر های عمومی ، شمارنده برنامه ، حافظه های FLASH و EEPROM
- دارای قابلیت پویش مرزی BOUNDARY SCAN میکروکنترلر. پویش مرزی JTAG دارای قابلیت برسی کامل پورت ها، مدارهای آنالوگ و کلیه قسمت های میکروکنترلر می باشد.
- دارای قابلیت BREAK (نقض کردن) دستورالعمل ها ، BREAK به هنگام تغییر روند برنامه ، قابلیت SINGLE STEP BREAK و قابلیت BREAK برنامه در یک آدرس مشخص.
- دارای قابلیت به کارگیری برای PROGRAM کردن حافظه های FLASH , EEPROM , فیوز ها و LOCK BIT ها .

کنترل کننده TAB رجیستر دستورالعمل های JTAG یا یکی از چندین رجیستر دیتای میکرو کنترلر های AVR را به عنوان زنجیره SCAN بین پایه های TDI(ورودی اطلاعات سریال) و TDO(خروجی اطلاعات سریال) انتخاب می کند.توسط یک مولتی پلکسر داخلی می توان خروجی رجیسترهای مختلف نظیر INSTRUCTION REGISTER , ID REGISTER , BYPASS REGISTER , BREAK POINT , SCAN CHAIN یا BOUNDARY SCAN را با استفاده از TAP CONTROLLER انتخاب و مقدار آن را بر روی پایه TDO مشاهده نمود.

رابط JTAG از طریق چهار پایه میکروکنترلر ATMEGA32 قابل دسترسی است.به این پایه ها اصطلاحا TAP گفته می شود. در ادامه به معرفی این چهار پایه می پردازیم.

۱- پایه TMS (TEST MODE SELECT) :

این پایه برای مدیریت JTAG از طریق کنترل کننده TAP به کار می رود.

۲- پایه TCK (TEST CLOCK) :

رابط JTAG به صورت سنکرون با این پالس کار می کند.

۳- پایه TDI (TEST DATA IN) :

اطلا عات سریال ورودی از طریق این پایه به داخل رجیستر های دستورالعمل یا رجیستر های دیتا انتقال می یابند.

۴- پایه TDO(TEST DATA OUTPUT) :

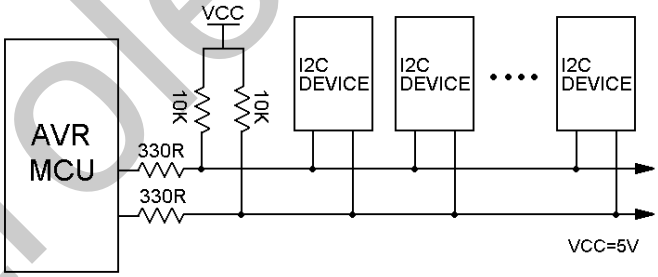
خروجی سریال اطلا عات از رجیستر های دستور العمل یا دیتا می باشد.

هنگامی که فیوز JTAG برنامه ریزی نشده باشد ، پایه های TAP به صورت پورتهای I/O در حالت عادی کارکرده و کنترل کننده TAP نیز غیر فعال است. اگر این فیوز بیت برنامه ریزی شود و بیت JTD در رجیستر MCUCR صفر شده باشد، سیگنالهای ورودی TAP به صورت داخلی HIGH شده ، JTAG برای برنامه ریزی و پویش مرزی فعال می شود. در میکروی MEGA32 رابط JTAG به صورت پیش فرض توسط فیوز بیت های JTAGEN ، PROGRAMED (برنامه ریزی شده) می باشد.

رابط سریال I2C (2-WIRE)

رابط سریال I2C یک پروتکل ارتباطی 2-WIRE است که برای انتقال اطلاعات با استفاده از دو باس و برای برقراری ارتباط با EEPROM های سریال سری AT24 و وسایل دیگری که دارای این پروتکل ارتباطی هستند به کار می رود . البته برای برقراری ارتباط نیاز به VCC و زمین است ، پس در واقع ارتباط I2C ارتباطی چهار سیمه می باشد.

I2C روش مناسبی برای برقراری ارتباطات میکروکنترلر هاست این پروتکل با استفاده از دو باس پالس(SCL) و دیتا (SDA) ، امکان برقراری ارتباط بین 128 وسیله مختلف یا میکروکنترلر را فراهم می سازد تمامی تجهیزات متصل شده به باس دارای آدرس منحصر به فرد هستند ، مکانیزم در اختیار گرفتن باس با استفاده از پروتکل I2C انجام می شود . شکل 1-8 طرز اتصال وسایل I2C دریک باس را نشان می دهد .



شکل 1-8 طرز قرار گرفتن I2C DEVICE در باس I2C

در بحث I2C ، MASTER (ارباب یا فرمانده) وسیله ای است که پس از آغاز به کار شروع به ارسال اطلاعات کرده و پالسSCL تولید می کند . SLAVE (برده یا فرمانبر) نیز تجهیزی است که توسط MASTER آدرس دهی می شود .

MASTER وظیفه شروع و به پایان رساندن انتقال داده را بر عهده دارد. انتقال اطلاعات زمانی آغاز می شودکه MASTER شرطSTART را بر روی باس قرار دهد و پایان انتقال اطلاعات با قرار دادن شرط STOP اعلام

می شود . بین شرایط START و STOP ، باس به صورت BUSY (اشغال) در نظر گرفته می شود و MASTER دیگری نباید سعی در اشغال باس را داشته باشد .

هنگام برقراری ارتباط I2C با EEPROM های سریال سری AT24 برای انتخابSLAVE (EEPROM) ابتدا آدرسی که به صورت سخت افزاری توسط پایه های A0 تا A2 در هر کدام از EEPROM ها تعیین شده است ، توسط MASTER به باس I2C فرستاده می شود ، سپس آدرس داده و پس از آن خود داده در باس I2C قرار می گیرد .

مبدل آنالوگ به دیجیتال (ADC) داخلی میکرو

ADC ولتاژ ورودی رابه مقدار دیجیتال ده بیتی با تقریب بسیار خوبی تبدیل می کند ، مینیمم مقدار خروجی صفر و ماکزیمم آن زمانی حاصل می شود که ولتاژ ورودی برابر ولتاژ مرجع مبدل باشد . ولتاژ مرجع مبدل از طریق پایه AREF یا ولتاژ مرجع داخلی (2.56V) تامین می شود .

در MEGA32 ، 8 کانال SINGLE ENDED به ADC اختصاص یافته است ، منظور از ولتاژ SINGLE ENDED ولتاژی است که نسبت به زمین سنجیده می شود .

خصوصیات مبدل آنالوگ به دیجیتال داخلی AVR :

- وضوح 10 بیتی (یعنی ADC داخلی AVR می تواند ولتاژهای اعمالی بین 0 تا ولتاژ مرجع را به 1024 قسمت تقسیم کند.)
- صحت مطلق ±2 LSB
- زمان تبدیل (CONVERSION TIME) 65-260US
- وضوح 15KSPS در بالاترین حد
- کانال های مولتی پلکس شده
- مد های تبدیل FREE(عملیات تبدیل برای همه کانال هایADC) و SINGLE (عملیات تبدیل برای یکی از کانال ها)
- ولتاژ ورودی از 0V تا VCC (یعنی ولتاژ مرجع و ولتاژ اعمالی به کانال ADC بایستی بین 0 تا VCC باشد.)
- پرچم وقفه پایان تبدیل ADC
- حذف کننده نویز(NOISE CANCELER) هنگام استفاده از مد SLEEP مربوطه

مدیریت تغذیه و مدهای SLEEP

فعال شدن ماژول های مختلف میکرو کنترلر های AVR با استفاده از منابع مختلف پالس صورت می گیرد با وارد شدن ماژول ها به مد های SLEEP ، این پالس ها غیر فعال شده در نتیجه ماژول مربوطه نیز غیرفعال می شود . مد های مختلف SLEEP (EXTRA STANDBY, IDLE , STANDBY , ADC NOISE REDUCTION , POWER – DOWN,POWRE SAVE) به منظور متوقف کردن ماژول های غیر فعال MCU به کار می روند تا توان تلفاتی را کا هش دهند. برای فعال کردن مد های SLEEP از بیت SE در رجیستر MCUCR استفاده می شود. با یک نمودن بیت SE در رجیستر فوق و اجرا نمودن دستورالعمل SLEEP ، میکروکنترلر وارد یکی از مد های SLEEP می شود. سه بیت انتخاب کننده مد SLEEP، یعنی بیت های SM2..0 در رجیستر MCUCR برای انتخاب یکی از مدهای SLEEP به کار می روند. جدول زیر مقادیر مختلف این بیت ها و مد های مختلف SLEEP را نشان داده است.

These bits select between the six available sleep modes as shown in Table 13.
Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Note: 1. Standby mode and Extended Standby mode are only available with external crystals or resonators.

در صورتی که MCU در یکی از مد های SLEEP باشد و وقفه ای رخ دهد، MCU فعال می شود. در ابتدا به مدت چهار سیکل و مدت زمان START UP متوقف مانده و سپس برنامه سرویس وقفه را اجرا و مجددا به حالت SLEEP برمی گردد. مقادیر رجیستر ها و SRAM هنگام WAKE UP (بیدار شدن از حالت SLEEP) شدن MCU تغییر نمی کنند. در صورت وقوع RESET نیز میکروکنترلر فعال شده و به بردار RESET پرش می کند.

The MCU Control Register contains control bits for power management.

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

با یک شدن بیت SE در رجیستر MCUCR (MCU CONTROL REGISTER) و اجرای دستورالعمل SLEEP , MCU وارد یکی از مد های SLEEP می شود که توسط بیت های SM2..0 تعیین شده است. توجه داشته باشید که در پروژه های این کتاب از کار کردن مستقیم با رجیستر ها صرفه نظر شده است . و برای تغییر محتوای رجیستر های کنترلی از دستورات نرم افزاری BASIC استفاده شده است . انواع مد های SLEEP و نحوه وارد شدن به آن ها توسط دستورات BASCOM در فصل دوم کتاب ارائه شده است .

فیوز بیت های میکروکنترلر ATMEGA32

میکروکنترلر ATMEGA32 دارای دو بایت فیوز بیت طبق جدول های زیر می باشد.

توجه شود که در تمامی فیوز بیت ها (0) به معنای پروگرم شدن (PROGRAMED) و (1) به معنای پروگرم نشدن (UNPROGRAMED) می باشد.

Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
OCDEN(4)	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN(1)	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT(2)	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 (see Table 100 for details)	0 (programmed)(3)
BOOTSZ0	1	Select Boot Size (see Table 100 for details)	0 (programmed)(3)
BOOTRST	0	Select reset vector	1 (unprogrammed)

جدول بایت بالای فیوز بیت های ATMEGA 32

فیوز بیت ها با پاک کردن (ERASE) میکرو تاثیری نمی بیند ولی می توانند با برنامه ریزی بیت LBI قفل شوند. (بیتهای قفل (LB) برای قفل کردن فیوزبیتها بر روی مقدار برنامه ریزی شده به کار می روند.)

فیوز بیت های بایت بالای MEGA32 :

: OCDEN

در صورتی که بیتهای قفل برنامه ریزی نشده باشند و رابط JTAG توسط فیوز مربوطه و سیستم عیب یاب درونی(ON CHIP DEBUG) توسط فیوز OCDEN فعال شده باشد ماژول JTAG قابلیت DEBUGGING (عیب یابی) کلیه قسمت های داخلی و خارجی ، رجیستر های شمارنده برنامه (PC) ، حافظه های فلش و EEPROM را خواهد داشت البته در این حالت اگر میکرو کنترلر وارد مد های SLEEP (POWER DOWN) و (POWER SAVE) شود پالس سیستم هنوز فعال بوده و منجر به تلفات توان

خواهد شد برای جلو گیری از اتلاف توان در این شرایط باید یکی از فیوز بیت های OCDEN یا JTAGEN ، UNPROGRAM شود.

JTAGEN :

مربوط به حالت برنامه ریزی MICRO از طریق استاندارد JTAG می باشد که در حالت پیش فرض PROGRAMED است، اگر بخواهیم از پایه های PC5..2 به عنوان I/O استفاده کنیم بایستی این بیت UNPROGRAM شود رابط JTAG یکی دیگر از ماژول های جانبی MEGA32 است که اغلب برای PROGRAM کردن میکروکنترلر با استفاده از سخت افزارهای خاصی نظیر پروگرمر STK500 به کار می رود.

SPIEN :

بیت فعال ساز برنامه ریزی MICRO از طریق استاندارد SPI می باشد و در حالت عادی به صورت PROGRAMED است.

CKOPT :

هنگامی که فرکانس کاری MICRO از8MHZ بالاترود برنامه ریزی می شود ودر حالت عادی به صورت UNPROGRAMED است در حقیقت برنامه ریزی این بیت بستگی به فیوز بیت های CKSEL3..0 دارد.

EESAVE :

اگر این بیت برنامه ریزی شود محتویات EEPROM در زمان ERASE میکرو محفوظ می ماند و اگر برنامه ریزی نشود در زمان ERASE میکرو EEPROM نیز پاک خواهد شد این بیت در حالت پیش فرض به صورت UNPROGRAM می باشد.

BOOTSZ0,BOOTSZ1 :

از این دو فیوز بیت برای تنظیم مقدار حافظه BOOT استفاده می شود، اگر فیوز بیت BOOTRST به صورت PROGRAMED باشد. اجرای برنامه از آدرسی شروع می شود که توسط فیوز بیت های BOOTSZ1 و BOOTSZ0 طبق جدول زیر تعیین شده است در حالت پیش فرض هر دو فیوز بیت به صورت PROGRAMED بوده وآدرس شروع برنامه $0000 خواهد بود.

BOOTSZ1	BOOTSZ0	BOOT SIZE	PAGES	APPLICATION FLASH ADDRESSES	BOOT FLASH ADDRESSES	BOOT RESET ADDRESS
1	1	256 WORDS	4	$0000-$3EFF	$3F00-$3FFF	$3F00
1	0	512 WORDS	8	$0000-$3DFF	$3E00-$3FFF	$3E00
0	1	1024 WORDS	16	$0000-$3BFF	$3C00-$3FFF	$3C00
0	0	2048 WORDS	32	$0000-$37FF	$3800-$3FFF	$3800

جدول انتخاب مقدار حافظه BOOT توسط فیوز بیت های BOOTSZ0,1

BOOTRST :

این فیوز بیت مربوط به انتخاب بردار ریست BOOT می باشد که در صورت PROGRAM شدن آدرس بردار ریست توسط فیوز بیت های BOOTSZ0 , BOOTSZ1 طبق جدول مر بوطه تعیین می شود که در حالت پیش فرض این فیوز بیت به صورت UNPROGRAMED می باشد و آدرس بردار ریست $0000 خواهد بود.

Fuse Low Byte

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown-out Detector trigger level	1 (unprogrammed)
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

جدول بایت پایین فیوز بیت های ATMEGA 32

BODLEVEL : با PROGRAM شدن این بیت اگر ولتاژ پایه VCC از 4 ولت پایین تر شود ریست داخلی میکرو فعال می شود و با UNPROGRAM شدن آن اگر ولتاژ پایه VCC از2.7V کمتر شود ریست داخلی فعال شده و میکرو ریست می شود.

BODEN :برای فعل کردن عملکرد مدار BROWN–OUT این بیت بایستی برنامه ریزی شده باشد این بیت به صورت پیش فرض UNPROGRAM است. میکروکنترلر MEGA32 دارای یک مدار آشکار ساز BROWN–OUT است. این مدار در هر لحظه سطح ولتاژ VCC را برسی می کند و همواره آن را با یک سطح ولتاژ مرجع مقایسه می نماید ولتاژ مرجع نیز توسط فیوز بیت BODLEVEL انتخاب می شود. در صورتی که VCC از ولتاژ مرجع کمتر شود عمل ریست را انجام می دهد.

SUT1,SUT0 : برای انتخاب زمان START–UP به کار می روند.

CKSEL3..CKSEL0 : فرکانس کاری MICRO توسط این فیوز بیت ها تعیین می شود که قبلا به صورت مفصل در این رابطه توضیح داده شده است مقدار پیش فرض اسیلاتور RC اینترنال 1MHz می باشد.

برسی کاربرد پورت های میکروی**ATMEGA32** :

کاربردهای پورت **A** :

هر کدام از پایه های پورت A را می توان به عنوان I/O (ورودی یا خروجی) ، CONFIG کرد. پورت A یک I/O دوطرفه 8 بیتی است سه آدرس از مکان حافظه I/O اختصاص به PORTA دارد یک آدرس برای رجیستر داده (PORTA) ، دومی رجیستر جهت داده (DDRA) و سومی بایت آدرس پایه های ورودی (PIN A) بایت آدرس پورت A فقط قابل خواندن است در حالی که رجیستر های PORT و DDR پورت A هم خواندنی و هم نوشتنی هستند ، تمام پایه های پورت A دارای مقاومت PULUP مجزا هستند. بافر خروجی پورت A می تواند تا 20mA را SINK کند در نتیجه یک IED را می توان مستقیما توسط پایه های پورت A راه اندازی کرد.

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

رجیستر DDRA جهت پایه های پورت A را به عنوان ورودی یا خروجی تعیین می کند. با یک کردن هر یک از بیت های DDRA همان بیت از پورت A به عنوان ورودی تعریف می شود برای مثال با یک کردن DDA1 می توانیم PA.1 را به عنوان خروجی CONFIG کنیم.

نکته: زمانی که پورت A خوانده می شود داده LATCH روی پورت A خوانده خواهد شد و زمانی که پین A خوانده می شود دسترسی به مقدار لحظه ای اطلا عات موجود بر روی هر یک از پایه ها را ممکن می سازد.

Port A Pins Alternate Functions

Port Pin	Alternate Function
PA7	ADC7 (ADC input channel 7)
PA6	ADC6 (ADC input channel 6)
PA5	ADC5 (ADC input channel 5)
PA4	ADC4 (ADC input channel 4)
PA3	ADC3 (ADC input channel 3)
PA2	ADC2 (ADC input channel 2)
PA1	ADC1 (ADC input channel 1)
PA0	ADC0 (ADC input channel 0)

جدول کاربرد های پورت A

پورت A به عنوان ADC (مبدل آنالوگ به دیجیتال) نیز مورد استفاده قرار می گیرد اگر تعدادی از پایه های پورتA به عنوان خروجی وتعدادی به عنوان ADC تعریف شوند پایه های خروجی نباید در زمان تبدیل مقدار دهی می شوند این کار ممکن است عملیات تبدیل ADC را نامعتبر کند.

کاربرد های پورت B : پورت B را نیز می توان به صورت یک I/O دو طرفه 8 بیتی مورد استفاده قرار داد. سه آدرس از مکان حافظه I/O به پورتB اختصاص دارد یک آدرس برای رجیسترداده (PORT B) دومی رجیستر جهت داده (DDRB) و سومی بایت آدرس پایه های ورودی(PINB) توضیحات مربوط به این 3 بایت در قسمت کاربرد های پورت A داده شده است.

Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Data Direction Register– DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

جدول کاربرد های دیگر پورت**B** به شرح زیر است

Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

PORTB.7 – SCK :

SCK : کلاک خروجی MASTER و کلاک ورودی SLAVE برای ارتباط SPI است.زمانی که SPI به عنوان SLAVE شکل دهی می شود.

PORTB.6 – MISO :

MISO : ورودی داده MASTER و خروجی داده SLAVE برای ارتباط SPI می باشد.

PORTB.5- MOSI :

MOSI : ورودی داده SLAVE و خروجی داده MASTER برای ارتباط SPI می باشد.

PORTB.4 – SS :

SS : زمانی که SPI به عنوان SLAVE شکل دهی می شود PB.4 با توجه به DDB4 ورودی تعریف می شود و در SLAVE با LOWE شدن این پایه SPI فعال می شود. این پایه در MASTER می تواند خروجی یا ورودی تعریف شود.

PORTB.3-OC0 , AIN1 :

AIN1 : ورودی منفی مقایسه کننده آنالوگ است.

OC0 : دیگر کار برد این پایه به عنوان خروجی مد مقایسه ای COUNTER0،TIMER است.

PORTB.3-OC0 , AINI0 :

AINI0 : ورودی مثبت مقایسه کننده آنالوگ است.

INT2 :دیگر کاربرد این پایه به عنوان منبع وقفه خارجی دو است. پایه PB2 می تواند به عنوان منبع وقفه خارجی برای میکرو (MCU) استفاده شود. توجه داشته باشید که وقفه خارجی INT2 فقط در لبه بالا رونده ، یا پایین رونده تریگ می شود.

PORTB.1 – T1 :

T1 : ورودی کلاک برای COUNTER0،TIMER است.

XCK : این پایه نیز می تواند به عنوان کلاک خارجی USART استفاده می شود.این پایه فقط زمانی که USART در مد آسنکرون کار می کند فعال می شود.

کاربرد های پورت C :

پورت C را نیز می توان به صورت یک I/O دوطرفه 8 بیتی مورد استفاده قرار داد، سه آدرس از مکان حافظه I/O به پورت C اختصاص دارد. یک آدرس برای رجیستر داده (PORT C) ، یک آدرس برای رجیستر جهت داده (DDRC) و سومی برای بایت آدرس پایه های ورودی (PINC) .

توضیحات مربوط به این 3 بایت در قسمت کاربرد های پورت A داده است.

لازم به ذکر است که برای استفاده از پایه های C.2 تا C.5 به عنوان I/O در MEGA32 بایستی فیوز بیت JTAG یک (UNPROGRAM) شود .

Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

کاربرد های دیگر پورت C :

Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC5	TDI (JTAG Test Data In)
PC4	TDO (JTAG Test Data Out)
PC3	TMS (JTAG Test Mode Select)
PC2	TCK (JTAG Test Clock)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC0	SCL (Two-wire Serial Bus Clock Line)

PORTC.7- TOSC2 :

TOSC2 : زمانی که تایمر / کانترهای 2 یا 0 در مد آسنکرون کار می کند به این پایه و پایه TOSC1 کریستال ساعت متصل می شود. در این حالت دیگر نمی توان این پایه را به عنوان I/O استفاده نمود.

PORTC.6- TOSC1 :

TOSC1 : زمانی که تایمر / کانتر 2 در مد آسنکرون کار می کند به این پایه و پایه TOSC2 کریستال ساعت متصل می شود. در این حالت دیگر نمی توان این پایه را به عنوان I/O استفاده نمود.

PORTC.5 – TDI :

اطلا عات سریال ورودی از طریق این پایه به داخل رجیستر های دستورالعمل یا رجیستر های دیتا انتقال می یابند.

POORT.4 – TDO :

خروجی سریال اطلا عات از رجیستر های دستور العمل یا دیتا می باشد.

PORTC.3 – TMS :

این پایه برای مدیریت JTAG از طریق کنترل کننده TAP به کار می رود.

PORTC.2 – TCK :

رابط JTAG به صورت سنکرون با این پالس کار می کند.

PORTC.1 – SDA :

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

PORTC.0 – SCL :

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

معرفی پورت D :

پورت D را نیز می توان به عنوان یکI/O دو طرفه 8 بیتی مورد استفاده قرار دارد 3 آدرس از مکان حافظهI/O به پورت C اختصاص دارد. یک آدرس برای رجیستر داده (PORTD) ، یک آدرس برای رجیستر جهت داده (DDRD) ویک آدرس برای بایت آدرس پایه های ورودی(PINC) .

توضیحات مربوطه به این 3 بایت در قسمت کاربرد های پورت A داده شده است.

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

کاربرد های دیگر پورت D :

Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

PORTD.7 –OC2 :

OC2 : خروجی مد مقایسه ای تایمر / کانتر می باشد. این پایه هم چنین برای خروجی PWM تایمر2 نیز استفاده می شود.

PORTD.6 –ICP :

به عنوان پایه ورودی CAPTURE تایمر/کانتر 1عمل کند.

PORTD.6 –OC1A :

OC1A : خروجی مد مقایسه ای تایمر/کانتر می باشد.

این پایه هم چنین برای خروجی PWM تایمر 1نیز استفاده می شود.

PORTD.6 –OC1B :

OC1A : خروجی مد مقایسه ای تایمر/کانتر1 می باشد.

این پایه هم چنین برای خروجی PWM تایمر1 استفاده می شود.

PORTD.3 –INT1 :

INT1 : منبع وقفه خارجی یک.

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

PORTD.2 –INT0 :

INT0 : منبع وقفه خارجی یک.

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

PORTD.1 –TXD :

TXD : ارسال داده (پایه خروجی داده برای USART)

PORTD.0 –RXD :

RXD : دریافت داده (پایه ورودی داده برای USART)

معرفی میکروکنترلر ATMEGA16 :

ATMEGA16 از سری میکروکنترلر های MEGA AVR می باشد ، میکروکنترلر های MEGA نسبت به انواع دیگر (AT90S,TINY) دارای امکانات و قابلیت های بیشتری هستند .

خصوصیات ATMEGA16 و ATMEGA16L :

- از معماری AVR RISC (کاهش دستورالعمل های میکروکنترلر) استفاده می کند.
- دارای131 دستورالعمل با کارایی بالا که همگی در یک کلاک سیکل ساعت اجرا می شوند .
- 32*8 رجیستر کاربردی.

- سرعتی تا 16MIPS در فرکانس 16MHZ
- 16K بایت حافظه FLASH داخلی قابل برنامه ریزی .

پایداری حافظه **FLASH** : قابلیت 10.000 بار نوشتن و پاک کردن (WRITE/ERASE)

- 1024بایت حافظه داخلی SRAM
- 512 بایت حافظه EEPROM داخلی قابل برنامه ریزی .
- پایداری حافظه **EEPROM** : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)
- قفل برنامه FLASH و حفاظت داده EEPROM هنگام ERASE کردن میکرو

قابلیت ارتباط **JTAG (IEEE STD.)** :

- برنامه ریزی برنامه FLASH ، EEPROM ، FUSE BITS و 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 مد COMPARE و CAPTURE
- چهار کانال PWM (PULSE WIDTH MODULATOR)
- هشت کانال مبدل آنالوگ به دیجیتال 10 بیتی
- یک مقایسه کننده آنالوگ داخلی(INTERNAL ANALOG COMPARATOR)
- 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 کالیبر شده
- دارای شش حالت SLEEP (POWER-DOWN ، IDLE ، POWER-SAVE ، STANDBY ، EXTENDED STANDBY و ADC NOISE REDUCTION)
- منابع وقفه (INTERRUPT) داخلی و خارجی .
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

توان مصرفی در 1MHz، 3V در 25 درجه سانتی گراد برای ATMEGA16L :

- حالت فعال 1.1ma (ACTIVE MODE)
- در حالت بی کاری 0.3ma (IDLE MODE)
- در حالت POWER-DOWN : <1UA

ولتاژ های عملیاتی (کاری)

- 2.7V تا 5.5V برای (ATMEGA16L)
- 4.5V تا 5.5V برای (ATMEGA16)

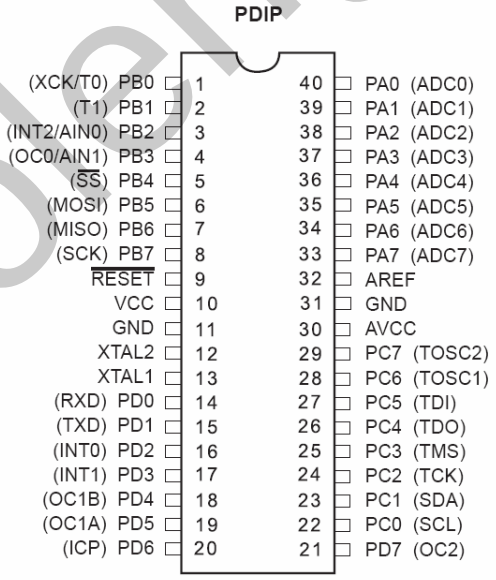
فرکانس های کاری

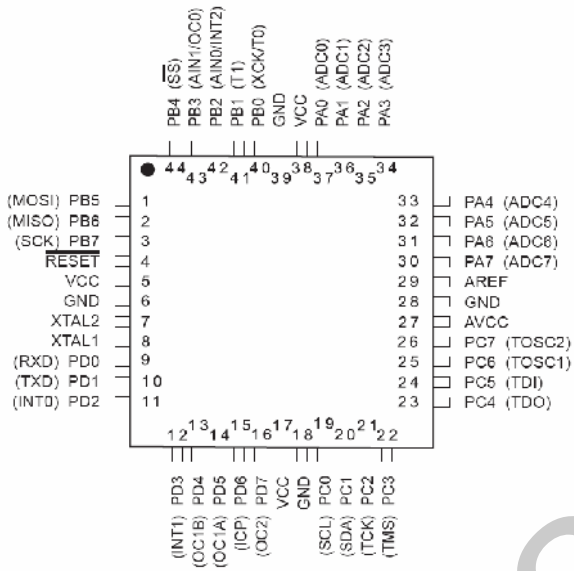
- 0MHZ تا 8MHZ برای (ATMEGA16L)
- 0MHZ تا 16MHZ برای (ATMEGA16)

خطوط **I/O** و انواع بسته بندی

- 32خط ورودی / خروجی (I/O) قابل برنامه ریزی
- 40 پایه PDIP, 44 پایه TQFP و 44 پایه MLF

ترکیب پایه های میکروکنترلر **ATMEGA16 (PIN CONFIGURATION)** :





منابع کلاک در میکروکنترلر ATMEGA16 عبارتند از :

۱- نوسان ساز RC کالیبره شده داخلی

۲- کریستال خارجی فرکانس پایین (معمولا 32.768 khz)

۳- کریستال خارجی یا رزوناتور (RESONATOR) سرامیکی

۴- کلاک خارجی

۵- اسیلاتور RC خارجی

۶- نوسان ساز کریستالی خارجی برای تایمر های 8 بیتی (تایمر مد 0 و2) که به کریستال ساعت نیز معروف است و مقدار آن (32.768 KHZ) می باشد.

جزئیات مربوط به تنظیم کلاک سیستم در قسمت های قبلی توضیح داده شده است .

منابع RESET میکروکنترلر ATMEGA16 :

میکروکنترلر ATMEGA16 دارای منابع ریست POWER-ON ، ریست سخت افزاری خارجی ، ریست BROWN–OUT ، ریستWATCHDOG و ریست JTAG می باشد . به منظور حفاظت در برابر خطا ها میکروکنترلر باید ریست شود ، در این حالت محتویات حافظه FLASH تغییر نمی کند و رجیستر شمارنده برنامه (PROGRAM COUNTER) به ابتدای حافظه فلش باز میگردد با آمدن هر یک از ریست ها یک پالس تحریک به DLAY COUNTER اعمال می شود و پس از یک زمان تاخیر که توسط فیوزبیت های CKSEL , SUT تنظیم می شود عمل ریست انجام می شود .

منابع وقفه در میکروکنترلر ATMEGA16 :

در میکروکنترلر های خانواده های AVR اولویت وقفه ها به صورت یک سطحی است یعنی در صورت رخ دادن همزمان دو وقفه ابتدا به وقفه ای که آدرس پایین تری در جدول بالا دارد پاسخ داده می شود سپس با توجه به ردیف طبق جدول زیر به وقفه بعدی پاسخ داده می شود.

Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

جدول منابع وقفه در ATMEGA 16

وقفه های خارجی ATMEGA16L :

وقفه های خارجی توسط پایه های INT0 , INT1 و INT2 تریگر می شوند.وقفه های INT0 و INT1 با لبه بالا رونده ، پایین رونده یا با سطح پایین تریگر می شوند ولی وقفه INT2 فقط به لبه حساس است. تعیین حساسیت وقفه های خارجی به لبه یا سطح در رجیستر های MCUCSR و MCUCR انجام می شود. وقتی یک وقفه خارجی به صورت حساس به سطح (فقط INT0 و INT1) تنظیم می شود وقفه تا زمان پایین بودن سیگنال فعال خواهد بود .

در رابطه با وقفه های خارجی و نحوه تحریک آن ها قبلا به صورت مفصل توضیح داده شده است .

مدیریت تغذیه و مد های SLEEP در ATMEGA16

فعال شدن ماژول های مختلف میکرو کنترلر های AVR با استفاده از منابع مختلف پالس صورت می گیرد و به منظور وارد شدن ماژول های مختلف به مد های SLEEP بایستی این پالسها غیر فعال شود . در ادامه مد های SLEEP مورد برسی قرار می گیرد.

انواع مد های SLEEP درتراشه MEGA16 عبارتند از:

POWER-DOWN ، IDLE ، POWER-SAVE ، STANNDBY ، EXTENDED-STANDBY ، ADC NOISE REDOCTION

مد های SLEEP در فصل دوم کتاب به صورت مفصل بررسی شده اند .

فیوز بیت های میکروکنترلر ATMEGA16

میکروکنترلر ATMEGA16 دارای دو بایت فیوز بیت طبق جدول های زیر می باشد.

توجه شود که در تمامی فیوز بیت ها 0 به معنای پروگرم شدن (PROGRAMED) و 1 به معنای پروگرم نشدن (UNPROGRAMED) می باشد.

جدول فیوز بیت های MEGA16

Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
OCDEN(4)	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN(1)	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT(2)	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 (see Table 100 for details)	0 (programmed)(3)
BOOTSZ0	1	Select Boot Size (see Table 100 for details)	0 (programmed)(3)
BOOTRST	0	Select reset vector	1 (unprogrammed)

فیوز بیت ها با پاک کردن (ERASE) میکرو تاثیری نمی بینند ولی می توانند با برنامه ریزی بیتهایLBI قفل شوند . (بیتهای قفل (LB) برای قفل کردن فیوزبیتها بر روی مقدار برنامه ریزی شده به کار می روند.) از آنجایی که فیوز بیت های MEGA16 دقیقا مشابه فیوز بیت های ATMEGA32 می باشد.در این قسمت از ارائه توضیح در رابطه با آن ها صرفه نظر شده است .

فیوز بیت های بایت پایین MEGA16

Fuse Low Byte

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown-out Detector trigger level	1 (unprogrammed)
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

معرفی پورت های ATMEGA16

آرایش پایه های MEGA16 دقیقا مشابه ATMEGA32 می باشد و کاربرد پایه های آنها نیز یکسان است . از این رو برای مطالعه کاربرد پایه های MEGA16 به قسمت کاربرد پایه های ATMEGA32 مراجعه شود.

معرفی میکروکنترلر ATMEGA8

ATMEGA8 از سری میکرکنترلر های MEGA AVR می باشد ، میکروکنترلر های MEGA نسبت به نواع دیگر (AT90S,TINY) دارای امکانات وقابلیت های بیشتری هستند .

خصوصیات ATMEGA8 و ATMEGA8L

- یک میکرو کنترلر 8 بیتی که در آن از معماری RICS استفاده شده است .
- کارایی بالا و توان مصرفی کم
- دارای 130 دستورالعمل با کارایی بالا که اکثرا تنها در یک کلاک سیکل اجرا می شوند .
- 8*32 رجیستر کاربردی.
- سرعتی تا 16MIPS در فرکانس 16 MHz
- 8K بایت حافظه FLASH داخلی قابل برنامه ریزی .

پایداری حافظه **FLASH** : قابلیت 10,000 بار نوشتن و پاک کردن (WRITE/ERASE)

- 1024بایت حافظه داخلی SRAM
- 512 بایت حافظه EEPROM داخلی قابل برنامه ریزی .
- پایداری حافظه **EEPROM** :

قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)

- قفل برنامه FLASH و حفاظت داده EEPROM

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 مد های COMPARE و CAPTURE
- 3 کانال PWM
- هشت کانال مبدل آنالوگ به دیجیتال در بسته بندی های TQFP و MLF
شش کانال با دقت 10 بیتی
دو کانال بادقت 8 بیتی
- شش کانال مبدل آنالوگ به دیجیتال در بسته بندی های PDIP
چهار کانال با دقت 10 بیتی
دو کانال بادقت 8 بیتی
دارای RTC (REAL-TIME CLOCK) با اسیلاتور داخلی.
- یک مقایسه کننده آنالوگ داخلی
- USART سریال قابل برنامه ریزی
- WATCH DOG قابل برنامه ریزی با اسیلاتور داخلی
- ارتباط سریال SPI برای برنامه ریزی داخل مدار(IN-SYSTEM PROGRAMMING)
- قابلیت ارتباط سریال SPI (SERIAL PERIPHERAL INTERFACE) به صورت MASTER یا SLAVE .
- قابلیت ارتباط با پروتکل سریال دوسیمه (TWO-WIRE)

خصوصیات ویژه میکروکنترلر :

- POWER ON RESET CIRCUIT
- دارای اسیلاتور RC داخلی کالیبر شده
- دارای 5 حالت SLEEP (POWER-DOWN ، IDLE ، POWER-SAVE ، 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 4MHz، 3V ، 25 درجه سانتی گراد
- حالت فعال 3.6mA (ACTIVE MODE)

- در حالت بی کاری 1.0mA (IDLE MODE)
- در حالت POWER-DOWN : <5UA

ولتاژ های عملیاتی (کاری)

- 2.7V تا 5.5V برای (ATMEGA8L)
- 4.5V تا 5.5V برای (ATMEGA8)

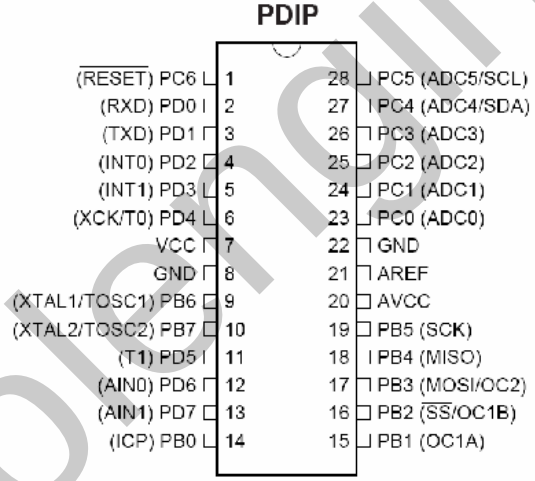
فرکانس های کاری

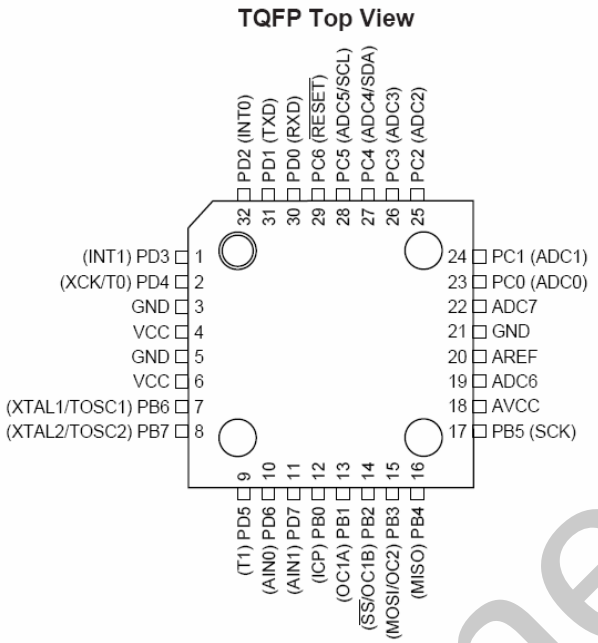
- 0MHz تا 8MHz برای (ATMEGA8L)
- 0MHz تا 16MHz برای (ATMEGA8)

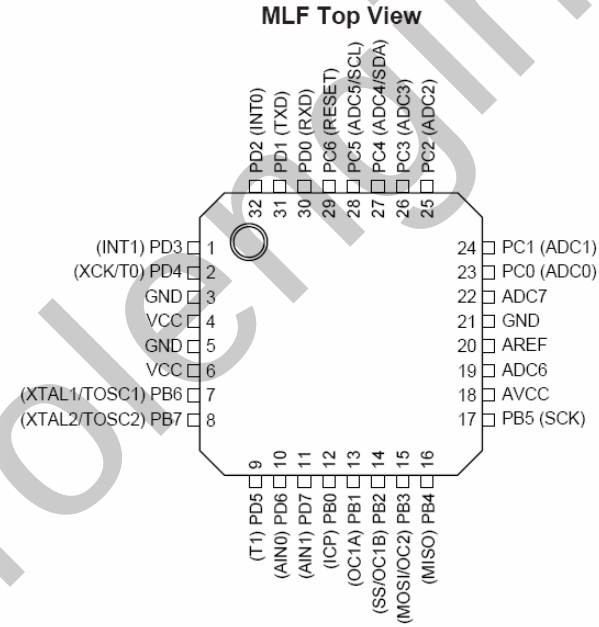
خطوط **I/O** و انواع بسته بندی

- 23خط ورودی / خروجی (I/O) قابل برنامه ریزی
- 28 پایه PDIP, 32 پایه TQFP و MLF

ترکیب پایه ها ی میکروکنترلر در ATMEGA8 (PIN CONFIGURATION)







منابع ریست درمیکروکنترلر ATMEGA8

میکروکنترلر ATMEGA8 دارای 4 منبع ریست است POWER-ON ، ریست خارجی ، ریست WATCH DOG و ریست BROWN-OUT

منابع وقفه در ATMEGA8

در جدول زیرتمامی منابع وقفه و ریست میکروکنترلر ATMEGA8 ارائه شده اند.

Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	0x000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	SPM_RDY	Store Program Memory Ready

جدول منابع وقفه در ATMEGA8

فیوز بیت های ATMEGA8

MEGA8 دارای دو بایت فیوز بیت است که در جداول زیر نشان داده شده است .

Fuse High byte

Fuse High Byte	Bit No.	Description	Default Value
RSTDISBL(4)	7	Select if PC6 is I/O pin or RESET pin	1 (unprogrammed, PC6 is RESET-pin)
WDTON	6	WDT always on	1 (unprogrammed, WDT enabled by WDTCR)
SPIEN(1)	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT(2)	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 (see Table 82 for details)	0 (programmed)(3)
BOOTSZ0	1	Select Boot Size (see Table 82 for details)	0 (programmed)(3)
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Fuse Low Byte

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown out detector trigger level	1 (unprogrammed)
BODEN	6	Brown out detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

توجه شود در تمام فیوز بیت ها (0) به معنای بیت برنامه ریزی شده (PROGRAMED) و (1) به معنای بیت برنامه ریزی نشده (UNPROGRAMED) می باشد.

RISTDISBL : در حالت پیش فرض PC6 پایه RESET می باشد با برنامه ریزی این بیت پایه PC6 به عنوان I/O استفاده می شود .

WDTON : در حالت پیش فرض WATCHDOG غیر فعال می باشد و کاربر بایستی به صورت نرم افزاری WATCH DOG راه اندازی کند ، ولی زمانی که این بیت برنامه ریزی می شود WATCHDOG همیشه روشن است . بقیه فیوز بیت ها قبلا توضیح داده شده اند .

خصوصیات AT90S2313

از معماری AVR RISC استفاده می کند .

- یک میکروکنترلر 8 بیتی با کارایی بالا و توان مصرفی کم
- دارای 118 دستورالعمل که اکثرا در یک کلاک سیکل اجرا می شوند .
- 8*32 رجیستر همه منظوره .
- سرعتی تا 10MIPS در فرکانس 10MHz

حافظه ، برنامه و داده غیر فرار :

- 2K بایت حافظه FLASH قابل برنامه ریزی داخلی .

پایداری حافظه FLASH : قابلیت 1000 بار نوشتن و پاک کردن (WRITE/ERASE)

- 128بایت حافظه SRAM

- 128 بایت حافظه EEPROM داخلی قابل برنامه ریزی
- پایداری حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن (WRITE / ERASE)
- قفل برنامه FLASH و حفاظت داده EEPROM

خصوصیات جانبی :

- ایجاد وقفه باتغییر وضعیت پایه
- یک تایمر – کانتر (TIMER/COUNTER) 8 بیتی با PRESCALER مجزا
- یک تایمر – کانتر (TIMER/COUNTER) 8 بیتی با PRESCALER مجزا و دارای مد های COMPARE ، CAPTURE و PWM ,9,8 یا 10 بیتی .
- یک مقایسه کننده آنالوگ داخلی
- WATCHDOG قابل برنامه ریزی با اسیلاتور داخلی
- ارتباط سریال SPI برای برنامه ریزی داخل مدار (IN-SYSTEM PROGRAMING) .
- UART دو طرفه (FULL DUPLEX) .

خصوصیات ویژه میکروکنترلر :

- تغذیه کم در مد های IDLE ، POWERDOWN
- منابع وقفه (INTERRUPT) داخلی و خارجی .
- عملکرد کاملا ثابت.
- توان مصرفی پایین و سرعت بالا توسط تکنولوژی CMOS

توان مصرفی در 4MHz ، 3V ، 25 درجه سانتی گراد .

- حالت فعال 2.8mA (ACTIVE)
- در حالت بی کاری 0.8mA (IDLE)
- در حالت POWER-DOWN : <1UA

ولتاژ های عملیاتی (کاری)

- 2.7V تا 6V برای (AT90S2313-4)
- 4V تا 6V برای (AT90S2313-10)

فرکانس های کاری

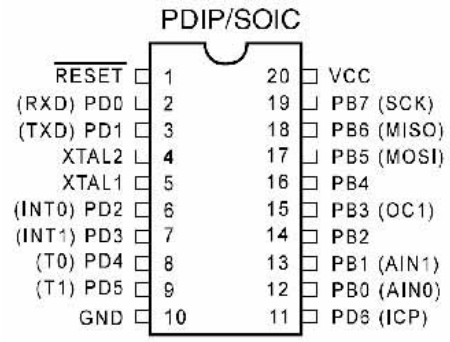
0MHz تا 4MHz برای (AT90S2313-4)

0MHz تا 12 MHz برای (AT90S2313-10)

خطوط I/O و انواع بسته بندی

- 15 خط ورودی / خروجی (I/O) قابل برنامه ریزی .
- 20 پایه (PIN) در انواع PDIP ، SOIC

ترکیب پایه ها در AT90S2313 :



منابع ریست در AT90S2313

منابع ریست در میکروکنترلر AT90S2313 عبارتند از

WATCHDOG RESET ، EXTERNAL RESET ، RESET POWER-ON

منابع وقفه در AT90S2313

Table 2. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin, Power-on Reset and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT1	Timer/Counter1 Capture Event
5	$004	TIMER1 COMP1	Timer/Counter1 Compare Match
6	$005	TIMER1 OVF1	Timer/Counter1 Overflow
7	$006	TIMER0 OVF0	Timer/Counter0 Overflow
8	$007	UART, RX	UART, RX Complete
9	$008	UART, UDRE	UART Data Register Empty
10	$009	UART, TX	UART, TX Complete
11	$00A	ANA_COMP	Analog Comparator

جدول منابع وقفه در AT90S2313

کاربرد های پورت B در جدول زیر ارائه شده است .

Table 17. Port B Pin Alternate Functions

Port Pin	Alternate Functions
PB0	AIN0 (Analog Comparator positive input)
PB1	AIN1 (Analog Comparator negative input)
PB3	OC1 (Timer/Counter1 Output Compare Match output)
PB5	MOSI (Data input line for memory downloading)
PB6	MISO (Data output line for memory uploading)
PB7	SCK (Serial clock input)

When the pins are used for the alternate function, the DDRB and PORTB Registers have to be set according to the alternate function description.

کاربرد های پورت D در جدول زیر ارائه شده است .

Table 19. Port D Pin Alternate Functions

Port Pin	Alternate Function
PD0	RXD (Receive data input for the UART)
PD1	TXD (Transmit data output for the UART)
PD2	INT0 (External interrupt 0 input)
PD3	INT1 (External interrupt 1 input)
PD4	TO (Timer/Counter0 external input)
PD5	T1 (Timer/Counter1 external input)
PD6	ICP (Timer/Counter1Input Capture pin)

When the pins are used for the alternate function, the DDRD and PORTD Registers have to be set according to the alternate function description.

فیوز بیت های AT90S2313

میکروکنترلر AT90S2313 دارای 2 فیوز بیت به شرح زیر است که در زمان برنامه ریزی به صورت سریال قابل دسترسی نمی باشد .

SPIEN : در حالت پیش فرض به صورت برنامه ریزی شده بوده و میکرو کنترلر می تواند از طریق سریال SPI برنامه ریزی شود .

FSTRT : با برنامه ریزی کردن این بیت کوتاه ترین زمان شروع (START UP) برای ریست و همچنین مد های SLEEP در نظر گرفته می شود . این بیت به صورت پیش فرض برنامه ریزی نشده بوده و طولانی ترین زمان در نظر گرفته شده است . جدول زیر مشخص کننده این زمان ها است .

VCC	PARAMETER	MIN	TYP	MAX	UNITS
VCC = 5.0V	RESET DELAY TIME-OUT PERIOD AT90S2313 FSTRT PROGRAMMED	0.25	0.28	0.31	ms
	RESET DELAY TIME-OUT PERIOD AT90S2313 FSTRT UNPROGRAMMED	11.0	16.0	21.0	ms

Beautiful Pictures Deloped From Negatives In a Dark Room . So, If You See Darkness In Your Life Be Sure That GOD Is Making Beautiful Picture For You!

زیبا ترین تصاویر توسط نگاتیو هایی در اتاق های تاریک ایجاد می شوند ، بنابراین اگر شما تاریکی را در زندگی تان دیدید مطمئن باشید که خداوند در حال ساختن تصویری زیبا برای شما ست .

فصل دوم

معرفی دستورات و آموزش برنامه نویسی ، پیکره بندی و کار با امکانات AVR در محیط (BASCOM)

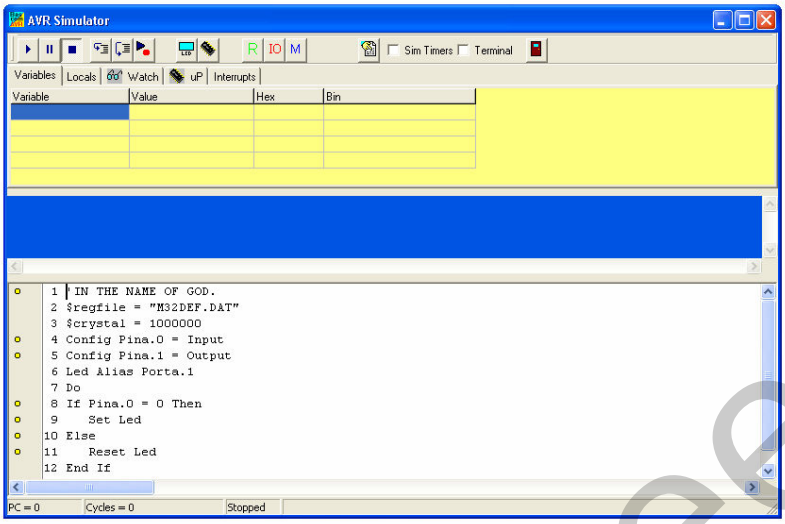
در پروژه های این کتاب از 4 میکروکنترلر معرفی شده در فصل اول استفاده شده است که میکروکنترلر ATMEGA32 نسبت به 3 میکرو کنترلر دیگر از امکانات و کارائی بالاتری برخوردار بوده و همه امکانات مربوط به خانواده AVR را داراست ، از این رو در این فصل برای آموزش برنامه نویسی و کار با امکانات خانواده AVR در محیط BASCOM ، از میکروکنترلر نمونه ATMEGA32 استفاده شده است،شما می توانید هنگام طراحی پروژه های مختلف با توجه به نیاز خودتان و با توجه به امکانات هر یک از میکروکنترلرهای خانواده AVR نمونه مورد نظر خود را انتخاب کنید . برای نوشتن برنامه های این فصل به غیر از برنامه های مربوط به LCD گرافیکی از کامپایلر BASCOM1.11.7.4 استفاده شده است . برای نوشتن برنامه های مربوط به LCD گرافیکی می توانید از کامپایلر BASCOM 1.11.8.7 استفاده کنید .

پیشنهاد می شود برای یادگیری و تسلط بیشتر پروژه های این فصل را توسط سیمولاتور PROTEUS شبیه سازی کرده و نتیجه کار خود را مشاهده کنید ، نرم افزار PROTEUS یک شبیه ساز کامل مدار های دیجیتال و آنالوگ می باشد ، از خصوصیات بارز این نرم افزار توانایی آن در شبیه سازی مدار های میکروکنترلری است . برای شبیه سازی پروژه های این کتاب به غیر از پروژه هایی که مربوط به LCD گرافیکی می باشد از PROTEUS7.1 استفاده شده است. برای شبیه سازی پروژه های مربوط به LCD گرافیکی می توانید از PROTEUS6.2 استفاده کنید .

برای مشاهده نتیجه نهایی سیمولاتور پروتئوس بهترین شبیه ساز می باشد ولی برای مشاهده محتوای متغیر ها در حین اجرای برنامه همچنین اجرای خط به خط برنامه و مشاهده تغییرات موجود بر روی متغیرهای تعریف شده در برنامه و رجیستر های R0 تاR31 ،رجیستر های I/O و منابع وقفه می توان از SIMULATOR داخلی BASCOM استفاده نمود .

معرفی محیط (AVR(SIMULATOR :

با زدن کلید F2 در محیط برنامه نویسی BASCOM پنجره شکل2-1 با نام AVR SIMULATOR باز می شود .



شکل 2-1 پنجره AVR SIMULATOR

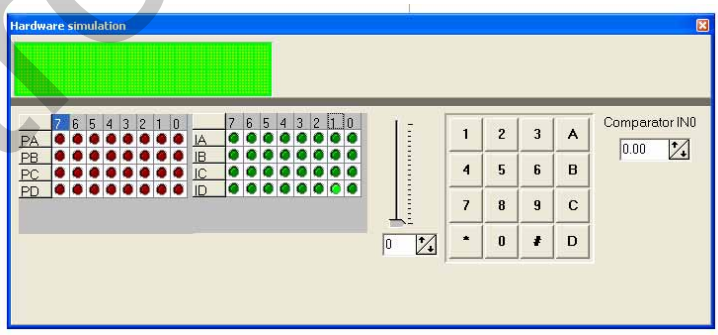
پنجره نمایش داده شده شامل کلید هایی به شرح زیر است .

RUN : با فشار این کلید شبیه سازی آغاز می شود .

PAUSE : با فشار این کلید شبیه سازی در خط جاری متوقف شده و با فشار مجدد آن اجرای برنامه شبیه سازی ادامه می یابد.

RUN TO CURRENT LINE : این کلید ، شبیه سازی را تا خطی که مکان نمای موس در آن قرار دارد ادامه می دهد .

SHOW HARDWARE EMULATION : این کلید شبیه ساز سخت افزاری AVR SIMULATOR را فعال می کند که پس از فشار آن پنجره نشان داده شده در شکل 2-2 با نام HARDWARE SIMULATION باز می شود .



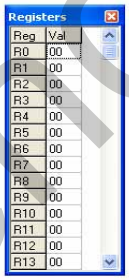
شکل 2-2 پنجره HARDWARE SIMULATION

در قسمت بالای پنجره ، سمت چپ، یک LCD وجود دارد که به منظور نمایش داده های ارسالی به LCD در نظر گرفته شده است ، اندازه این LCD با توجه به اندازه تعریف شده در برنامه خواهد بود برای مثال اگر LCD تعریف شده در برنامه 2*16 باشد این LCD نیز 2*16 خواهد بود .

LED های قرمز موجود در پنجره ، وضعیت پایه های خروجی و LED های سبز وضعیت پایه های ورودی را نشان می دهد . روشن شدن LED به منزله HIGH یا یک شدن پایه و خاموش شدن LED به منزله LOW یا صفر شدن پایه می باشد . برای تغییر وضعیت پایه های ورودی می توانید از LED های سبز استفاده کنید .

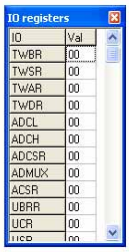
یک صفحه کلید مجازی نیز برای شبیه سازی KEYPAD ورودی درنظر گرفته شده ولی توصیه میشود برای شبیه سازی مداراتی که دارای KEYPAD می باشند از سیمولاتور PROTEUS استفاده کنید . در ضمن مقدارآنالوگ ورودی برای مقایسه کننده آنالوگ و همچنین کانال های مختلف ADC نیز توسط این پنجره قابل اعمال است .

DISPLAY REGISTER WINDOW : با زدن این کلید پنجره شکل 2-3 با نام REGISTERS باز می شوداین پنجره مقادیر هگز رجیستر های R0 تا R31 را در هنگام اجرای برنامه نشان می دهد



شکل 2-3 پنجره REGISTERS

DISPLAY I/O REGISTER WINDOW : بازدن این کلید پنجره شکل 2-4 با نام IO REGISTERS باز می شود این پنجره مقادیر هگز رجیستر های ورودی ، خروجی را در هنگام اجرای برنامه شان می دهد .

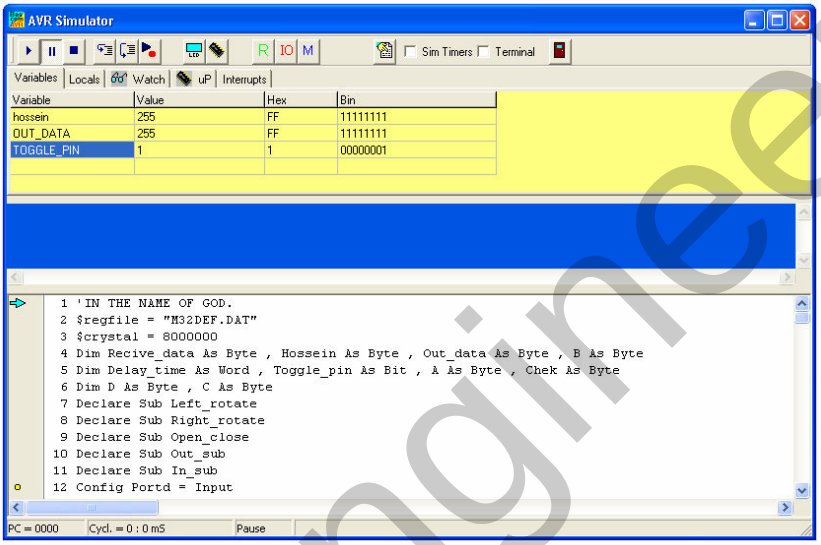


شکل 2-4 پنجره IO REGISTERS

گزینه **VARIABLES** :

برای مشاهده مقادیر موجود در متغیر های برنامه از این گزینه استفاده می شود شما می توانید متغیر مورد نظر خود را با دو بار کلیک کردن در ستون VARIABLE انتخاب کنید و یا این کار را با کلیک کردن بر روی ستون VARIABLE و نوشتن نام متغیر در این قسمت انجام دهید.

برای برسی مقدار یک متغیر آرایه ای ، می توانید نام متغیر همراه با اندیس آنرا در قسمت VARIABLE تایپ کنید



شکل **2-5** نمائی از پنجره **AVR SIMULATOR**

گزینه **INTERUPTS** :

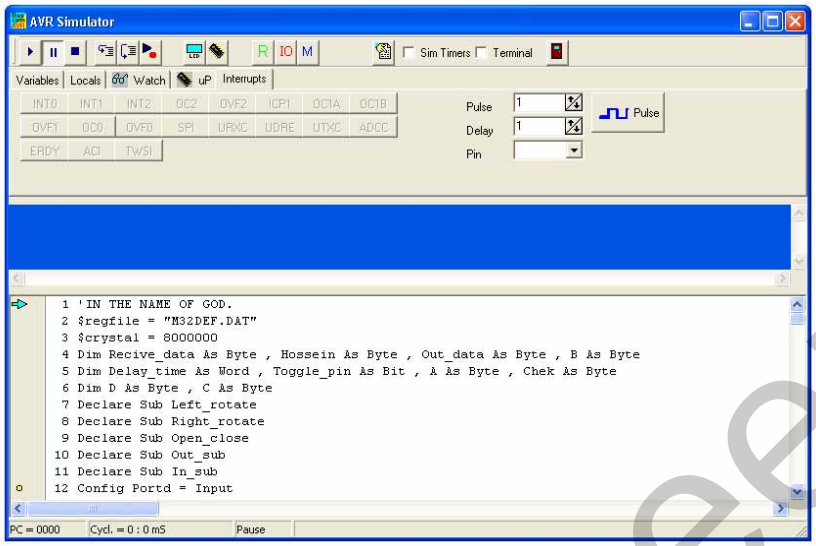
برای مشاهده منابع وقفه می توانید از این گزینه استفاده کنید ، اگر هیچ کدام از وقفه ها فعال سازی نشده باشد همه دکمه های مربوط به آنها غیرفعال خواهد بود . با کلیک بر روی هر کدام از این دکمه ها برنامه وقفه مربوطه اجرا می شود همچنین توسط این پنجره می توانید روی یک پایه خاص پالس مربعی ایجاد کنید .

STEP INTO CODE :

این گزینه باعث اجرای شبیه سازی به صورت دستور به دستور (STEP BY STEP) می شود ولی هنگام استفاده از دستورات CALL یا GOSUB برای فراخوانی زیر برنامه اجرای برنامه به داخل SUBROUTIN نخواهد رفت .

STEP OVER (GOSUB,CALL) :

نحوه عملکرد این کلید مانند کلید قبل می باشد با این تفاوت که می توان هنگام فراخوانی زیر برنامه با استفاده از دستور GOSUB به داخل زیربرنامه رفته و مراحل اجرای آن را بررسی کرد.



شکل 6-2 نمائی از پنجره AVR SIMULATOR

آموزش برنامه نویسی در محیط BASCOM :

برای نوشتن برنامه بیسیک در محیط BASCOM بایستی ابتدا نوع میکروکنترلر وسپس فرکانس کاری آن تعریف شود . برای معرفی میکرو کنترلر از دستور زیر استفاده می کنیم .

$REGFILE=AVR

AVR نام چیپ استفاده شده برای میکرو کنترلر مورد نظر در محیط BASCOM می باشد ، نام چیپ های مربوط به 4 میکروکنترلر معرفی شده در فصل اول در زیر ارائه شده است .

ATMEGA 32 =" M32DEF.DAT "
ATMEGA 16 =" M16DEF.DAT "
ATMEGA 8 =" M8DEF.DAT "
AT90S2313 ="2313DEF.DAT"

به عنوان مثال برای معرفی میکروکنترلر ATMEGA32 به صورت زیر عمل می کنیم .

$REGFILE = " M32DEF.DAT"

برای برنامه نویسی در محیط BASCOM همواره بایستی پس از تعریف نام میکرو ، کریستال استفاده شده برای میکروکنترلر معرفی شود .

برای مشخص کردن نوسانگر استفاده شده (چه داخلی و چه خارجی) ازدستور زیر استفاده می شود .

$CRYSTAL =AVR

AVR مقدار فرکانس انتخاب شده بر حسب هرتز می باشد .

البته فرکانس انتخاب شده علاوه بر این که در ابتدای برنامه تعریف می شود بایستی توسط فیوز بیت های مربوطه نیز تنظیم شود که در این قسمت مختصرا در مورد آن ها توضیح داده می شود . توجه شود که در تمامی فیوزبیت ها 0 به معنای برنامه ریزی شدن (PROGRAMED) و 1 به معنای برنامه ریزی نشدن (UNPROGRAMED)

می باشد . برا ی فرکانس های1 ،2 ، 4 و 8 مگا هرتز از نوسانگر RC کالیبره شده داخلی میکرو استفاده می کنیم نحوه برنامه ریزی فیوز بیت ها به صورت زیر خواهد بود . البته تنظیمات مربوط به فیوز بیت ها در فصل اول به طور کامل تشریح شده ولی برای یاد آوری و درک بهتر نحوه تنظیم فرکانس کاری سیستم توسط فیوز بیت ها به صورت مختصر در این قسمت توضیح داده می شود .

CKOPT	CKSEL 3	CKSEL 2	CKSEL 1	CKSEL 0	رنج فرکانسی
1	0	0	1	1	1.0MHz
1	0	0	0	0	2.0MHz
1	0	0	1	1	4.0MHz
1	0	1	0	0	8.0MHz

جدول تنظیم رنج فرکانس اسیلاتور کالیبره شده داخلی

توجه داشته باشید که رنج فرکانسی 1MHz به صورت پیش فرض برای میکروکنترلر انتخاب شده است . به عنوان مثال برای تنظیم فرکانس کاری میکرو روی 8 مگاهرتز ابتدا بایستی فرکانس مورد نظر در داخل برنامه به صورت زیر تعریف شود.

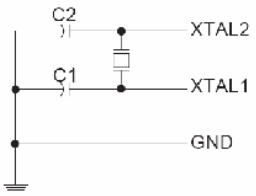
```
$CRYSTAL = 8000000
```

برای تنظیم فیوز بیت ها نیز با توجه به جدول بالا به صورت زیر عمل می کنیم .

```
CKOPT = 1 (UNPROGRAMED)
CKSEL 0 = 0  (PROGRAMED)
CKSEL1 = 0 (PROGRAMED)
CKSEL2 = 1 ( UNPROGRAMED)
CKSEL3 = 0 (PROGRAMED)
```

در صورتی که فرکانس مورد نظر بیشتر از 8 مگاهرتز باشد بایستی از نوسانگر خارجی استفاده کنیم . هنگام استفاده از نوسانگر خارجی به صورت زیر عمل خواهیم کرد .

در این حالت یک کریستال یا نوسانگر سرامیکی (CEROMIC RESONATOR) یا کریستال کوارتز (QUARTZ CRYSTAL) به صورتی که در شکل 7-2 دیده می شود به دو پایه XTAL1 , XTAL2 متصل میشود



Crystal Oscillator Connections

شکل 7-2 نحوه اتصال کریستال خارجی به میکروکنترلر

خازن های C1 و C2 برای کریستال ها و نوسانگر ها بایستی یک مقدار باشند و مقادیرآن ها نیز بستگی به مقدار نامی کریستال و نویز های الکترو مغناطیسی محیط دارد بعضی از خازن های مورد استفاده برای کریستال های مختلف در جدول زیر آمده است ، برای نوسانگر های سرامیکی بایستی از خازن هایی که توسط کارخانه سازنده پیشنهاد می شود استفاده نمود . هنگام استفاده از کریستال خارجی به همراه فیوز بیت های SUT برای تنظیم مدت زمان START-UP سیستم به کار می رود . (در فصل اول به صورت کامل توضیح داده شده است) که در این قسمت برای تنظیم فرکانس سیستم آن را 0 , (PROGRAMED) در نظر می گیریم جدول تنظیم فیوزبیت های کریستال خارجی به صورت زیر می باشد .

CKOPT	CKSEL3..1	رنج فرکانس (MHZ)	مقدار خازن های C2,C1
1	101	0.4 – 0.9	این گزینه مربوط به نوسانگرسرامیکی می باشد
1	110	0.9 – 3.0	12P - 22P
1	111	3.0 – 8.0	12P - 22P
0	111	بالای 8.0	12P - 22P

جدول تنظیم فیوز بیت ها برای کریستال یا نوسانگر سرامیکی خارجی

به عنوان مثال برای کریستال خارجی 12MHz به صورت زیر عمل می کنیم

نحوه معرفی در BASCOM

```
$CRYSTAL = 12000000

CKOPT = 0    (PROGRAMED)
CKSEL0 = 0   (PROGRAMED)
CKSEL1= 1    (ONPROGRAMED)
CKSEL2= 1    (ONPROGRAMED)
CKSEL3= 1    (ONPROGRAMED)
```

آدرس شروع نوشتن برنامه در حافظه FLASH :

گاهی نیاز است آدرس شروع نوشته شدن کد برنامه در (FLASH ROM) توسط برنامه نویس تعیین شود برای این منظور از دستور زیر استفاده می کنیم .

```
$ROMSTART = ADDRESS
```

ADDRESS مکانی از حافظه است که کد هگز برنامه از این آدرس در حافظه میکروکنترلر شروع به نوشتن می شود در صورت استفاده نکردن از این دستور کامپایلر ادرس &H0000 را در نظر می گیرد .

دستور END :

این دستور در انتهای برنامه قرار گرفته و اجرای آن را متوقف می کند توجه شود با اجرای دستور END توسط میکرو تمام وقفه ها غیر فعال شده و یک حلقه بی نهایت تولید می شود .

تعریف متغیر :

متغیر هایی که در برنامه نویسی بیسیک به کار برده می شوند بایستی قبل از استفاده شدن در برنامه با استفاده از دستور زیر تعریف شوند .

```
DIM AVR AS [XRAM/SRAM/ERAM] [VALUE RANGE]
```

VAR : نام متغیر می باشد . توجه داشته باشید که برای نام گذاری متغیر ها نبایستی از کلمات کلیدی BASCOM مانند START ، END و غیره استفاده نمود .

[XRAM/SRAM/ERAM] : در صورتی که می خواهید محتوای متغیر تعریف شده در حافظه جانبی ذخیره شود از گزینه XRAM و زمانی که بخواهید محتوای متغیر در حافظه EEPROM داخلی ذخیره شود از گزینه ERAM و برای ذخیره متغیر ها در حافظه رم داخلی از گزینه SRAM استفاده می کنید .

توجه شود اگر هیچ کدام از 3 گزینه بالا را استفاده نکنید محتوای متغیر در SRAM داخلی میکرو ذخیره می شود.

[VALUE RANGE] : نوع داده (محدوده متغیر) را مشخص می کند و می تواند یکی از گزینه های جدول زیر باشد .

DATA TYPE	STORES AS	VALUE RANGE
BIT	A BIT تک بیتی	0 OR 1
BYTE	UNSIGNED 8-BITS 8 بیتی	0 TO 255
INTEGER	SINGED 16-BITS16بیتی علامت دار	-32767 TO 32767
WORD	UNSINGED16-BITبیتی بدون علامت16	0 TO 65535
LONG	SINGED32-BITS 32 بیتی علامت دار	-2147483648 TO 214783647
SINGLE	SINGED 32-BITS 32 بیتی علامت دار و ممیز دار	1.5X 10^-45 TO 3.4X10^38
STRING	0-254 BYTES متغیر رشته ای	-

جدول انواع متغیر های موجود در BASCOM

توجه داشته بایستید که مقدار عدد ذخیره شده در متغیر بایستی در محدوده VALUE RANGE تعریف شده برای متغیر باشد .

اگر از متغیر STRING (رشته ای) استفاده می شود بایستی بیشترین طول آن را نیز مشخص کنید .

به عنوان مثال برای تعریف متغیر ی با نام S و طول رشته 10 کار اکتر به صورت زیر عمل می کنیم .

```
DIM S AS STRING * 10
```

و برای تعریف متغیری با نام A1 که عدد ذخیره شده در آن در محدوده 432767 تا 32767- قرار دارد به صورت زیر عمل می کنیم .

```
DIM A1 AS INTEGER
```

برای تعریف متغیری با نام A2 که قرار است در EEPROM داخلی ذخیره شود و محدوده عددی آن بین 0 تا 255 قرار دارد به صورت زیر عمل می کنیم .

```
DIM A2 AS ERAM BYTE
```

برای تعریف یک متغیر ممیز دار با نام H که قرار است محتوای آن در SRAM داخلی کامپیوتر ذخیره شود به صورت زیر عمل می کنیم .

```
DIM H AS SINGLE
```

از آنجایی که درآموزش ارائه مثال باعث یاد گری و تسلط بیشتر می شود از این به بعد آموزش برنامه نویسی و کار با امکانات و نحوه پیکره بندی آنها را با ارائه مثال های متعدد ادامه خواهیم داد . همان طور که قبلا گفته شد برای ارائه مثال های برنامه نویسی در این فصل از میکرو کنترلر نمونه ATMEGA32 استفاده خواهیم کرد .

در میکروکنترلر ATMEGA32 ، 4 پورت 8 بیتی داریم که هر کدام از این پورت ها می تواند به صورت ورودی یا خروجی پیکره بندی (CONFIG) شود . که برای تعیین جهت پورت ها یا پین ها در محیط BASCOM از دستورات زیر استفاده می شود.

برای تعین جهت پورت ها

```
CONFIG PORTX = INPUT/OUTPUT
```

برای تعیین جهت هر یک از پین ها X

```
CONFIG PINX.Y = INPUT/OUTPUT
```

Y, بسته به نوع میکروکنترلر می تواند به ترتیب پایه های 0 تا7 پورت های A,B,C,D باشد به عنوان مثال برای پیکره بندی پین B.7 به عنوان خروجی داریم .

```
CONFIG PINB.7 = OUTPUT
```

و برای پیکره بندی همه پین های پورت B به عنوان ورودی داریم .

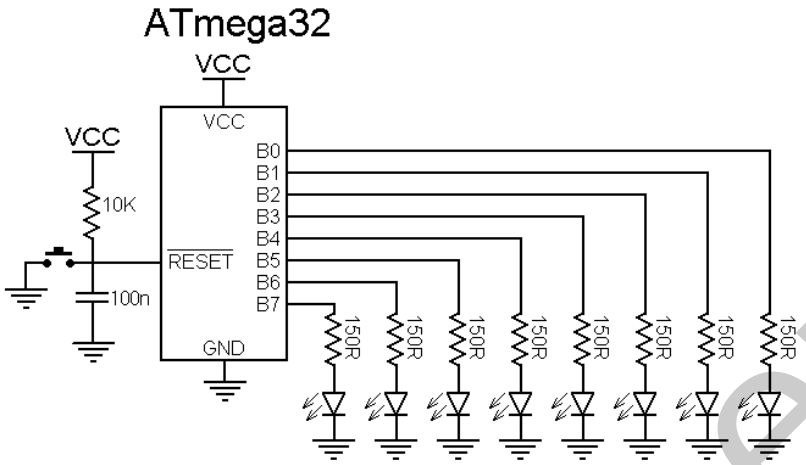
```
CONFIG PORTB = INPUT
```

توجه داشته باشید برای این که بتوانیم در میکروکنترلر های ATMEGA32 و ATMEGA16 از پورت C به عنوان I/O استفاده کنیم بایستی فیوزبیت JTAG ، UNPROGRAM یعنی یک شود .

زمانی که بخواهید از پورتی بخوانید بایستی از رجیستر PIN و برای نوشتن روی پورت از رجیستر PORT استفاده کنید (عملکرد 3 رجیستر DDR,PIN,PORT در فصل اول شرح داده شده است)

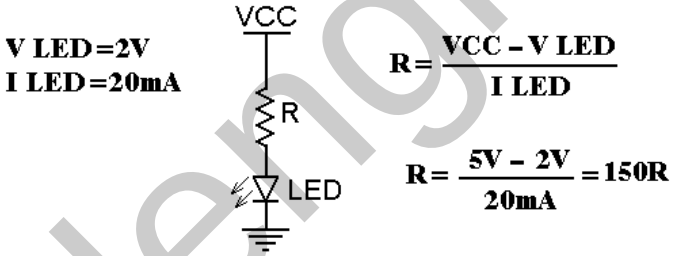
به عنوان مثال می خواهیم در مدارشکل 7-2 پورت A را به عنوان خروجی تعریف کرده و روی آن بنویسیم .

توجه داشته باشید تمام پورت های میکرو کنترلر AVR دارای مقاومت PULUP مجزا هستند و بافر خروجی آن ها می تواند بیشتر از 20mA را SINK کند در نتیجه یک LED را می توان مستقیما توسط پایه های خروجی میکروکنترلر راه اندازی کرد .



شکل 2-7 یک نمونه مدار طراحی شده برای استفاده از پورت B به عنوان خروجی

ولتاژ خروجی میکرو کنترلر در حالت یک منطقی برابر با 5 ولت است . پس برای راه اندازی LED با توجه به فرمول زیر بایستی از مقاومت 150R به صورت سری با LED استفاده کنیم.



شکل 2-8 نحوه محاسبه مقدار مقاومت محدود کننده جریان

برنامه نوشته شده در محیط BASCOM به شرح زیر است.

توجه داشته باشید شماره گذاری در ابتدای هر خط جزء متن برنامه نبوده و برای توضیح خط مربوطه استفاده شده است .

```
1-$REGFILE = " M32DEF.DAT"
2- $crystal = 1000000
3- Dim B As Byte
4-Config Porta = Output
5-Start_program :
6-For B = 0 To 200 Step 1
7-Porta = B
8-Waitms 100
9-Next B
10-For B = 200 To 0 Step -1
11-Porta = B
12-Waitms 100
```

```
13-Next B
14-Goto Start_program
15-End
```

توضیح برنامه :

۱- نوع میکرو به کامپایلر BASCOM معرفی شده است .

۲- ازدستور $CRYSTAL= VAR برای تعریف فرکانس کاری میکرو استفاده می شود .

۳- متغیری با نام B به صورت 8 بیتی تعریف شده است و حداکثر می تواند مقدار 255 را داشته باشد .

۴- تمام پایه های پورت A به صورت خروجی تعریف شده است .

۵- برچسب STSRT-PROGRAM به عنوان برچسب شروع برنامه استفاده شده است ، از بر چسب یا لیبل برای این منظور استفاده می کنیم که در مواقع لزوم اجرای برنامه را توسط دستور GOTO به آدرس لیبل مورد نظر انتقال دهیم . توجه داشته باشید که برای نام گذاری لیبل نمی توانیم از واژه های کلیدی BASCOM استفاده کنیم . برای اطمینان از این موضوع لیبل نوشته شده در BASCOM بایستی به رنگ سیاه باشد ، زیرا دستورات اصلی در BASCOM به رنگ آبی بوده و علائمی مانند + ، - ، = ، : به رنگ قرمز خواهد بود .

۶- از حلقه FOR-NEXT برای تکرار نوشتن بر روی PORTA استفاده شده که شکل کلی دستور FOR-NEXT به صورت زیر می باشد .

```
FOR VAR = START TO END STEP VALUE
```

VAR متغیری است که به عنوان کانتر عمل می کند و START مقدار اولیه و END آخرین مقدار یا مقدار پایانی آن می باشد VAR , END هر دو می توانند یک ثابت عددی یا یک متغیر عددی باشند VALUE مقدار عددی قدم ها را نشان می دهد که می تواند مثبت یا منفی باشد. عمل تکرار در حلقه FOR-NEXT توسط دستور NEXT VAR انجام می شود .

۷- مقدار متغیر B روی پورتA قرار می گیرد .

۸- اجرای برنامه به مدت صد میلی ثانیه در این خط متوقف می شود . برای ایجاد تاخیر از دستورات تاخیر و همچنین تایمر ها استفاده می شود .

دستورات تاخیر عبارتند از:

دستور DLAY : این دستور برای مدت کوتاهی به مقدار 100 میکرو ثانیه در اجرای برنامه تاخیر ایجاد می کند .

برای ایجاد تاخیر به میکرو ثانیه از دستور زیر استفاده می شود که در همه دستورات تاخیرVAR می تواند عددی بین 1تا 255 باشد .

```
WAITUS VAR
```

برای ایجاد تاخیر به میلی ثانیه از دستور زیر استفاده می شود .

WAITMS VAR

برای ایجاد تاخیر به ثانیه از این دستور استفاده می شود .

WAIT VAR

بعد از اجرای دستور بالا اجرای برنامه به مدت VAR ثانیه متوقف می شود پس از سپری شدن زمان تعیین شده اجرای برنامه از خط بعدی ادامه می یابد .

دستورات تاخیر زمان دقیق را به شما نمی دهد ، برای بدست آوردن زمان دقیق از تایمر ها استفاده کنید

تفاوت دستورات تاخیری با تایمر ها :

۱- دستورات تاخیری زمان دقیق را به شما نمی دهند ولی با تایمر ها می توانید زمان های تاخیر دقیق تری بدست بیاورید .

۲- هنگام استفاده از دستورات تاخیری اجرای برنامه تا پایان زمان تاخیر متوقف می شود ولی هنگام استفاده از تایمر ها تا زمان سرریز شدن تایمر (زمان تاخیر) میکروکنترلر قادر به اجرای دستورات دیگری نیز خواهد بود .

۹- با اجرای دستور NEXT B یک واحد به کانتر (متغیر B) حلقه FOR-NEXT اضافه می شود اگر مقدار کانتر کوچکتر یا مساوی مقدار پایانی (200) باشد دور بعدی حلقه FOR-NEXT اجرا خواهد شد اما اگر مقدار کانتر حلقه FOR-NEXT بزرگتر از مقدار پایانی باشد اجرای برنامه از حلقه FOR-NEXT خارج خواهد شد . پس با این حساب اولین عددی که در این حلقه در متغیر B قرار می گیرد عدد (1) می باشد هر بار یک واحد به B اضافه شده (توسط STEP1 در حلقه FOR-NEXT) و مقدار متغیر B روی پورت A قرار می گیرد ، پس از 100 میلی ثانیه تاخیر دور بعدی حلقه توسط دستور NEXT B اجرا می شود این روند 200 بار ادامه پیدا کرده و در نهایت اجرای برنامه از حلقه خارج می شود .

۱۰ – شروع حلقه FOR با کانتر B ،با مقدار اولیه 200 و مقدار پایانی 1 و با STEP نزولی (1) .

بدین صورت که مقدار اولیه کانتر 200 بوده و با هر بار تکرار حلقه یک واحد از آن کم می شود.

۱۱ – مقدار کانتر حلقه (متغیر B) روی پورت A قرار می گیرد .

۱۲ – اجرای برنامه به مدت 100 میلی ثانیه در این خط متوقف می شود .

۱۳- با اجرای دستور NEXT B یک واحد از کانتر (متغیر B) کم می شود اگر مقدار کانتر بزرگتر یا مساوی مقدار پایانی (1) باشد دور بعدی حلقه FOR-NEXT اجرا خواهد شد اما اگر مقدار کانتر حلقه کوچکتر از مقدار پایانی باشد اجرای برنامه از حلقه FOR-NEXT خارج خواهد شد .

پس با این حساب اولین عددی که در این حلقه در متغیر B قرار می گیرد عدد (200) می باشد که هر بار یک واحد از B کم شده (توسط STEP-1 در حلقه FOR-NEXT) و مقدار متغیر B روی پورت A قرار می گیرد پس از 100 میلی ثانیه تاخیر دور بعدی حلقه توسط دستور NEXT B اجرا می شود این روند 200 بار ادامه پیدا کرده و در نهایت اجرای برنامه از حلقه خارج می شود .

۱۴ – با اجرای دستور GO TO LABLE اجرای برنامه به آدرس لیبل مورد نظر منتقل می شود . بطورکلی در برنامه بالا ابتدا معادل باینری اعداد 1 تا 200 به ترتیب و به صورت صعودی روی پورت A قرار می گیرند سپس معادل باینری اعداد 200 تا 1 به ترتیب و به صورت نزولی روی پورت A قرار گرفته و همین روند برای همیشه ادامه پیدا می کند . فایل BASCOM , PROTEUS برنامه بالا در CD پیوست کتاب ارائه شده است .

دریافت ورودی :

نکاتی که هنگام دریافت ورودی بایستی مورد توجه قرار گیرد عبارتند از

۱- نحوه دریافت ورودی
۲- تعیین حساسیت ورودی به سطح یا لبه
۳- محافظت ورودی در برابر نویز های لحظه ای
۴- نحوه ایجاد تاخیر در حین دریافت ورودی

نحوه دریافت ورودی :

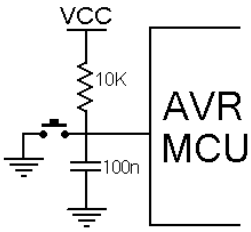
برای این که بتوانیم یکی از پایه های میکروکنترلر مانند PORTA را به عنوان ورودی در نظر بگریم ابتدا بایستی پایه مورد نظر را در BASCOM به صورت ورودی پیکره بندی (CONFIG) کنیم ، پیکره بندی توسط دستور زیر انجام می گیرد .

```
CONFIG PINA.0 = INPUT
```

اطلاعات ورودی می تواند خروجی یک میکرو کنترلر دیگر یا یک مدار مجتمع (IC) دیجیتال باشد .

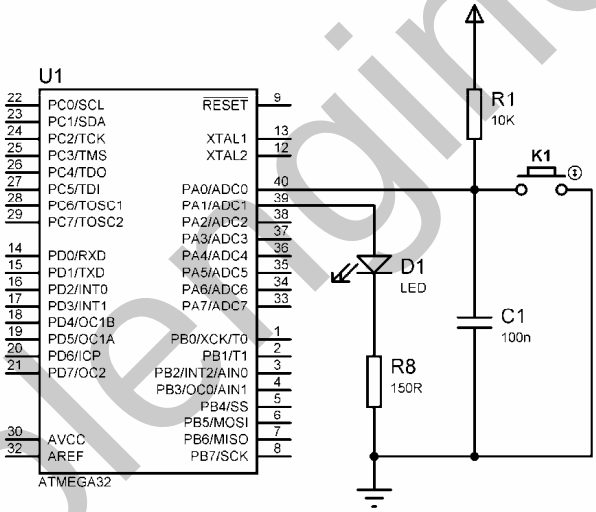
در صورتی که فرستنده یک مدار یا آی سی آنالوگ باشد ابتدا باید سطح ولتاژ خروخی را روی 5 ولت و زمین تنظیم نموده (توسط مدارات تریگر اشمیت و ...) سپس به ورودی میکروکنترلر اعمال کنیم.

در صورتی که از کلید به عنوان ورودی استفاده کنیم . کلید ورودی از نظر سخت افزاری می تواند به صورت (ACTIVE LOW) یا(ACTIVE HIGH) بسته شود اگر بخواهیم ورودی را به صورت ACTIVE-LOW در برنامه استفاده کنیم سخت افزار آن به شکل زیر خواهد بود .



شکل 2-9 نحوه استفاده از یک کلید به عنوان ورودی ACTIVE LOW در حالت سخت افزاری

اگر کلید ورودی بدین صورت بسته شود در حالت عادی پین A.1 در سطح منطقی یک قرار دارد و با فشار کلید K1 به سطح منطقی صفر می رود . به عنوان مثال در مدار زیر کلید K1 به صورت (ACTIVE LOW) مورد استفاده قرار گرفته است. برنامه نوشته شده به گونه ای است که در زمان بسته شدن کلید K1 , LED روشن و هنگامی که کلید باز است LED خاموش خواهد بود .



شکل 2-10 شماتیک طراحی شده برای استفاده از کلید به صورت ACTIVE LOW

برنامه نوشته شده برای سخت افزار بالا به شرح زیر است .

```
1-$regfile = "M32DEF.DAT"
2-$crystal = 1000000
3-Config Pina.0 = Input
4-Config Pina.1 = Output
5-Led Alias Porta.1
6-Do
7-If Pina.0 = 0 Then
8- Set Led
9-Else
10-Reset Led
11-End If
12-Loop
```

۱- نام میکروکنترلر استفاده شده تعریف شده است .

۲- فرکانس کاری میکرو 1MHz در نظر گرفته شده که در این صورت نیازی به تنظیم فیوز بیت های مربوطه نمی باشد .

۳- پین A.0 به عنوان خروجی پیکره بندی (CONFIG) شده است.

۴- پین A.1 به عنوان خروجی پیکره بندی (CONFIG) شده است.

۵- از دستور ALIAS برای تغییر نام PORTA.1 به LED استفاده شده است که شکل کلی دستور به صورت زیر می باشد .

```
DIRECTION ALIAS PORTX.Y
```

با نوشتن این دستور شما می توانید به جای PORTX.Y از نام DIRECTION استفاده کنید .

۶- شروع حلقه DO-LOOP که شکل کلی آن به شرح زیر است

```
DO
  STATEMENTS
LOOP UNTIL EXPRESSION
```

دستورالعمل STATEMENTS تا زمانی که EXPRESSION دارای ارزش TRUE یا غیر صفر است تکرار خواهد شد بنابر این ، این نوع حلقه بایستی حداقل یک بار تکرار شود اگراز حلقه DO-LOOP به شکل زیر استفاده کنیم

```
DO
  STATEMENTS
LOOP
```

تعداد تکرار حلقه DO-LOOP به صورت بی نهایت خواهد بود در این حالت برای خروج از حلقه می توانیم از دستور EXIT DO در داخل حلقه استفاده کنیم.

۷- استفاده از دستور IF که فرم کلی آن به صورت زیر است

دستوالعمل IF دارای 3 حالت کلی است

حالت اول :

```
IF EXPRESSION THEN STATEMENT
```

در صورتی که عبارت EXPRESSION دارای ارزش TRUE باشد دستورالعمل STATEMENT اجرا خواهد شد .

حالت دوم:

```
IF EXPRESSION THEN
STATEMENT 1
STATEMENT 2
        .
        .
STATEMENT N
END IF
```

در صورتی که عبارت EXPRESSION دارای ارزش TRUE باشد به ترتیب دستورالعمل های STATEMENT1 ، STATEMET2 و STATEMENTN اجرا خواهد شد .

حالت سوم:

```
IF EXPRESSION  THEN
STATEMENTS 1
ELSE
STATEMENTS 2
END IF
```

در صورتی که عبارت EXPRESSION دارای ارزش TRUE باشد دستورالعمل های STATEMENTS 1به ترتیب اجرا می شوند و در صورتی که عبارت EXPRESSION دارای ارزش FALSE باشد دستورالعمل های STATEMENTS2 به ترتیب اجرا خواهند شد . که در این جا از حالت سوم دستورالعمل IF استفاده شده است . اگر PIN A.0 برابر صفر باشد دستور SET LED اجرا می شود و گرنه دستور RESET LED اجرا خواهد شد .

توجه داشته باشید که برای خواندن از روی یک پین یا پورت به صورت لحظه ای بایستی از دستور VAR=PINX.Y استفاده کنیم که در این دستور VAR متغیری است که اطلاعات موجود برروی PIN X.Y در آن قرار می گیرد، به عنوان مثال برای خواندن مقدار لحظه ای پورت B از دستور زیر استفاده می کنیم

VAR = PINB

و برای خواندن مقدار لحظه ای PORTB.1 از دستور زیر استفاده می کنیم

VAR = PINB.1

و اگر بخواهیم مقدار LACH شده بر روی هر یک از پورت ها را بخوانیم از دستورات زیر استفاده خواهیم کرد

VAR = PORTB
VAR = PORTB.1

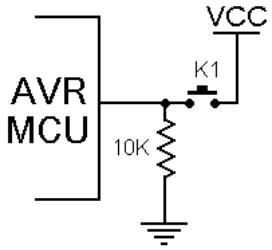
۸- با اجرای این دستور LED روشن می شود .

۹- مربوط به ساختار دستورالعمل IF در حالت سوم می باشد و اگر شرط PINA.0=0 برقرار نباشد دستور یا دستورات بعد از ELSE اجرا خواهد شد .

۱۰- با اجرای این دستور LED خاموش خواهد شد .

۱۱- END IF به معنای پایان دستور شرطی IF بوده و مربوط به ساختار دستورالعمل IF می باشد .

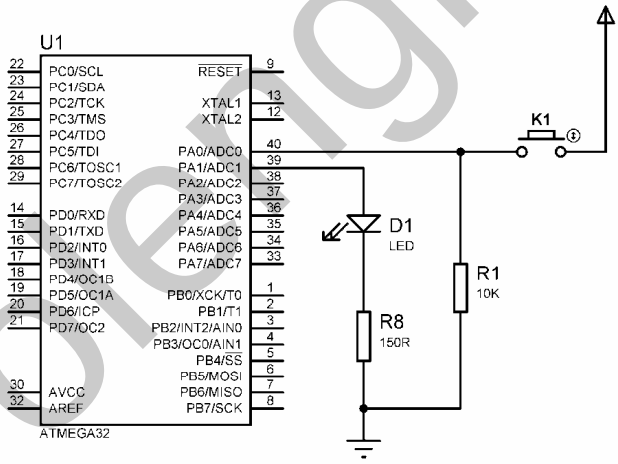
۱۲- دستور LOOP به معنای تکرار دور بعدی حلقه DO-LOOP می باشد . برای استفاده از ورودی به صورت ACTIVE HIGH از سخت افزار زیر استفاده خواهیم کرد .



شکل 2-11 نحوه استفاده از یک کلید به عنوان ورودی ACTIVE HIGH در حالت سخت افزاری

اگر کلید ورودی بدین صورت بسته شود در حالت عادی پین ورودی در سطح منطقی صفر قرار داشته و با فشار کلید به سطح منطقی یک می رود. در پروژه های این کتاب از این روش برای دریافت اطلاعات ورودی استفاده شده است .

به عنوان مثال در مدار زیر کلیدK1 به صورت (ACTIVE HIGH) بسته شد است . برنامه نوشته شده به گونه ای است که در زمان بسته شدن کلیدK1 ، LED روشن و در هنگامی که کلیدK1 باز است LED خاموش خواهد بود .



شکل 2-12 شماتیک طراحی شده برای استفاده از یک کلید به صورت ACTIVE HIGH

برنامه نوشته شده برای سخت افزار بالا به شرح زیر است :

```
1-$regfile = "M32DEF.DAT"
2-$crystal = 1000000
3-Config Pina.0 = Input
4-Config Pina.1 = Output
5-Led Alias Porta.1
6-Do
7-If Pina.0 = 1 Then
8-Set Led
9-Else
10-Reset Led
11-End If
```

12-Loop

۱- نام میکروکنترلر استفاده شده تعریف شده است .

۲- فرکانس کاری میکرو 1MHz در نظر گرفته شده است که در این حالت نیازی به تنظیم فیوز بیت های مربوطه نمی باشد .

۳- پین A.0 به عنوان ورودی پیکره بندی (CONFIG) شده است .

۴- پین A.1 به عنوان خروجی پیکره بندی (CONFIG) شده است .

۵- از دستور ALIAS برای تغییر نام PORTA.1 به LED استفاده شده است.

۶- دستور DO برای شروع حلقهDO-LOOP مورد استفاده قرار می گیرد .

۷- EXPRESSION یا شرط دستورالعمل IF در این خط بیان شده است (اگر پین A.0 برابر با یک است)

۸- اگر EXPERSSION دستورالعمل IF دارای ارزش TRUE باشددستور (SET LED) اجرا خواهد شد که با اجرای این دستور LED روشن می شود .

۹- اگر EXPERSSION دستورالعمل IF دارای ارزش FALSE باشد دستورالعمل بعد از ELSE اجرا خواهد شد .

۱۰- دستورالعمل RESET LED به معنای خاموش کردن LED می باشد . یعنی صفر کردن پورتی که LED به آن متصل شده است .

۱۱- دستورالعمل END IF به معنای پایان دستور شرطی IF می باشد .

۱۲- با اجرای دستورالعمل LOOP دور بعدی حلقه DO-LOOP اجرا خواهد شد .

نحوه تعیین حساسیت ورودی به سطح یا لبه :

توجه داشته باشید که ورودی میکروکنترلر می تواند به یکی از حالت های زیر حساس باشد .

۱- حساس به سطح یک منطقی

۲- حساس به سطح صفر منطقی

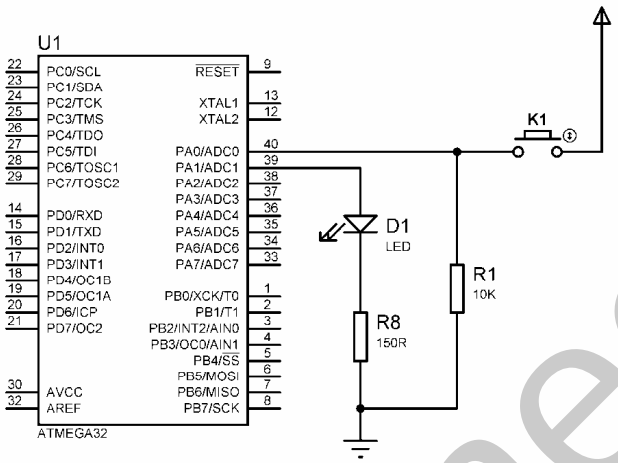
۳- حساس به لبه بالا رونده

۴- حساس به لبه پایین رونده

در حالتی که ورودی حساس به لبه نبوده و به سطح منطقی یک یا صفر حساس باشد و ورودی دائما چک شود اگر کلید ورودی بیش از مدت زمان معین در حالت بسته باقی بماند شرط بسته بودن کلید چندین بار اجرا خواهد شد .

برای درک بهتر این موضوع مدارمیکروکنترلری شکل 2-13 را در نظر بگیرید . دراین مدار با یک بار فشار کلید LED خروجی روشن شده و با فشار آن برای بار دوم خاموش می شود و ورودی نیز حساس به سطح یک منطقی

می باشد اگر کلید را فشار داده و آن را بیش از 100 میلی ثانیه در حالت بسته نگه داریم LED شروع به چشمک زدن خواهد کرد .



شکل 2-13 شماتیک طراحی شده برای استفاده از یک کلید به صورت ACTIVE HIGH

برنامه نوشته شده برای سخت افزار فوق به شرح زیر است .

```
1-$regfile = "M32DEF.DAT"
2-$crystal = 1000000
3-Config Pina.0 = Input
4-Config Pina.1 = Output
5-Dim Out_data As Bit
6-Led Alias Porta.1
7-Do
8-If Pina.0 = 1 Then Out_data = Not Out_data
9-If Out_data = 1 Then
10-Set Led
11-Else
12-Reset Led
13-End If
14-Waitms 100
15-Loop
```

توضیح برنامه :

برنامه به صورتی است که کلید ورودی در هر 100میلی ثانیه یک بار چک می شود و با یک بار فشار کلید LED روشن و با فشار آن برای بار دوم LED خاموش خواهد شد .

۱- نام میکروکنترلر استفاده شده تعریف شده است .

۲- فرکانس کاری میکرو 1MHz در نظر گرفته شده است که در این حالت نیازی به تنظیم فیوز بیت های مربوطه نمی باشد .

۳- پین A.0 به عنوان ورودی پیکره بندی (CONFIG) شده است .

۴- پین A.1 به عنوان خروجی پیکره بندی (CONFIG) شده است .

۵- متغیری با نام OUT-DATA به صورت بیتی تعریف شده است یعنی فقط می تواند دو مقوله (0) و (1) را داشته باشد .

۶- از دستور ALIAS برای تغییر نام PORTA.1 به LED استفاده شده است.

۷- دستور DO برای شروع حلقهDO-LOOP مورد استفاده قرار می گیرد .

۸- از دستورالعمل IF استفاده شده (اگر PINA.0 برابر با (1) باشد محتوای متغیر OUT-DATA معکوس می شود)

۹-EXPRESSION با شرط دستورالعمل IF در این خط بیان شده است (اگر متغیر OUT-DATA برابر با یک است)

۱۰- اگر EXPERSSION دستورالعمل IF دارای ارزش TRUE باشددستور (SET LED) اجرا خواهد شد که با اجرای این دستور LED روشن می شود .

۱۱- اگر EXPERSSION دستورالعمل IF دارای ارزش FALSE باشد دستورالعمل بعد از ELSE اجرا خواهد شد .

۱۲- با اجرای دستورالعمل RESET LED ، LED خاموش می شود .

۱۳- دستورالعمل END IF به معنای پایان دستور شرطی IF می باشد.

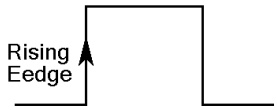
۱۴- اجرای برنامه به مدت 100میلی ثانیه در این خط متوقف می شود .

۱۵- با اجرای دستورالعمل LOOP دور بعدی حلقه DO-LOOP اجرا خواهد شد .

برای رفع مشکل چشمک زدنLED بایستی برنامه را طوری بنویسیم که ورودی حساس به لبه باشد برای این منظور می توانیم از دولبه بالا رونده (RISING EDEGE) یا پایین رونده (FALLING EDEGE) استفاده کنیم که در ادامه به توضیح هر دو مورد می پردازیم .

حساس به لبه بالا رونده :

در این حالت برنامه بایستی به گونه ای نوشته شود که پایه ورودی در لبه بالا رونده پالس تحریک شود لبه بالا رونده پالس ورودی در شکل 2-14 نشان داده شده است .



شکل 2-14 لبه بالا رونده (RISING EDGE) پالس ورودی

به عنوان مثال مدار میکروکنترلری شکل 13-2 را در نظر بگیرید در این مدار با یک بار فشار کلید LED خروجی روشن شده و با فشار آن برای بار دوم خاموش می شود ، ورودی حساس به لبه بالا رونده می باشد و با نگه داشتن کلید در حالت بسته LED بصورت چشمک زن نخواهد بود. برنامه نوشته شده در محیط BASCOM به صورت زیر است .

```
1-$regfile = "M32DEF.DAT"
2-$crystal = 1000000
3-Config Pina.0 = Input
4-Config Pina.1 = Output
5-Led Alias Porta.1
6-Do
7-If Pina.0 = 1 Then
8-Togle Led
9-Bitwait Pina.0,reset
10-End If
11-Loop
```

توضیح برنامه :

برنامه به گونه ای نوشته شده که به محض یک شدن ورودی ابتدا LED روشن شده و سپس توسط دستور BITWAIT تا زمان صفر شدن ورودی تاخیر ایجاد می شود .

۱- نام میکروکنترلر استفاده شده تعریف شده است .

۲- فرکانس کاری میکرو برابر 1MHz در نظر گرفته شده که در این حالت نیازی به تنظیم فیوز بیت های مربوطه نمی باشد .

۳- پین A.0 به عنوان ورودی پیکره بندی شده است .

۴- پین A.1 به عنوان خروجی پیکره بندی شده است .

۵- از دستور ALIAS برای تغییر نام PORTA.1 به LED استفاده شده است .

۶- شروع حلقه DO_LOOP

۷- از دستور IF استفاده شده اگر PINA.0 برابر با 1 باشد دستور نوشته شده در خط بعدی اجرا می شود .

۸- با اجرای این دستور وضعیت LED معکوس می شود .

۹- از دستور BITWAIT برای ایجاد تاخیر زمانی که PINA.0 برابر با صفر شود استفاده شده است و شکل کلی آن به صورت زیر است .

```
BITWAIT PIX,SET/RESET
```

۱۰- دستورالعمل END IF به معنای پایان دستور IF می باشد .

۱۱- با اجرای این دستور دور بعدی حلقه DO_LOOP اجرا می شود .

توجه داشته باشید که برنامه فوق برای آزمایش عملی نوشته شده و توسط سیمولاتور پروتئوس قابل اجرا نمی باشد اگر از سیمولاتور پروتئوس استفاده می کنید می توانید از برنامه زیر برای حساس کردن ورودی به لبه بالا رونده استفاده کنید .

برنامه نوشته شده در محیط BASCOM به شرح زیر است :

```
1-$regfile = "M32DEF.DAT"
2-$crystal = 1000000
3-Config Pina.0 = Input
4-Config Pina.1 = Output
5-Dim Out_data As Bit , Save As Bit , Input_pin As Bit
6-Led Alias Porta.1
7-Do
8-If Pina.0 = 1 Then Out_data = Not Out_data
9-Save = Pina.0
10-If Out_data = 1 Then
11-Set Led
12-Else
13-Reset Led
14-End If
15-Input_pin = Pina.0
16-While Save = Input_pin
17-Input_pin = Pina.0
18-Wend
19-Loop
```

۱- نام میکروکنترلر استفاده شده تعریف شده است .

۲- فرکانس کاری میکرو 1MHz در نظر گرفته شده است که در این حالت نیازی به تنظیم فیوز بیت های مربوطه نمی باشد .

۳- پین A.0 به عنوان ورودی پیکره بندی (CONFIG) شده است .

۴- پین A.1 به عنوان خروجی پیکره بندی (CONFIG) شده است .

۵- متغیری با نام OUT-DATA ، متغیری به نام SAVE و متغیری به نام INPUT-PIN به صورت بیتی تعریف شده اند .

۶- از دستور ALIAS برای تغییر نام PORTA.1 به LED استفاده شده است.

۷- دستور DO برای شروع حلقهDO-LOOP مورد استفاده قرار می گیرد .

۸- از دستورالعمل IF استفاده شده (اگر PINA.0 برابر با (1) باشد محتوای متغیر OUT-DATA معکوس می شود)

۹- با اجرای این دستور مقدار PINA.0 در متغیر SAVE قرار می گیرد .

۱۰-EXPRESSION با شرط دستورالعمل IF در این خط بیان شده است

۱۱- اگر EXPERSSION دستورالعمل IF دارای ارزش TRUE باشددستور (SET LED) اجرا خواهد شد که با اجرای این دستور LED روشن می شود .

۱۲- اگر EXPERSSION دستورالعمل IF دارای ارزش FALSE باشد دستورالعمل بعد از ELSE اجرا خواهد شد .

۱۳- با اجرای دستورالعمل RESET LED ، LED خاموش می شود .

۱۴- دستورالعمل END IF به معنای پایان دستور شرطی IF می باشد.

۱۵- با اجرای این دستور مقدار PINA.0 در متغیر INPUT-PIN قرار می گیرد .

۱۶- شروع حلقه WHILE-WEND که فرم کلی ان به شکل زیر است .

```
WHILE CONDITION
      STSTEMENTS
WEND
```

دستورات STATEMENTS تا وقتی که حاصل CONDITION ، TRUE می باشد تکرار خواهد شد .

در خط 16 این برنامه CONDITION برابر با SAVE = INPUT- PIN می باشد یعنی حلقه WHILE-WEND تا زمانی که متغیر های SAVE و INPUT-PIN برابر هستند تکرار خواهد شد.

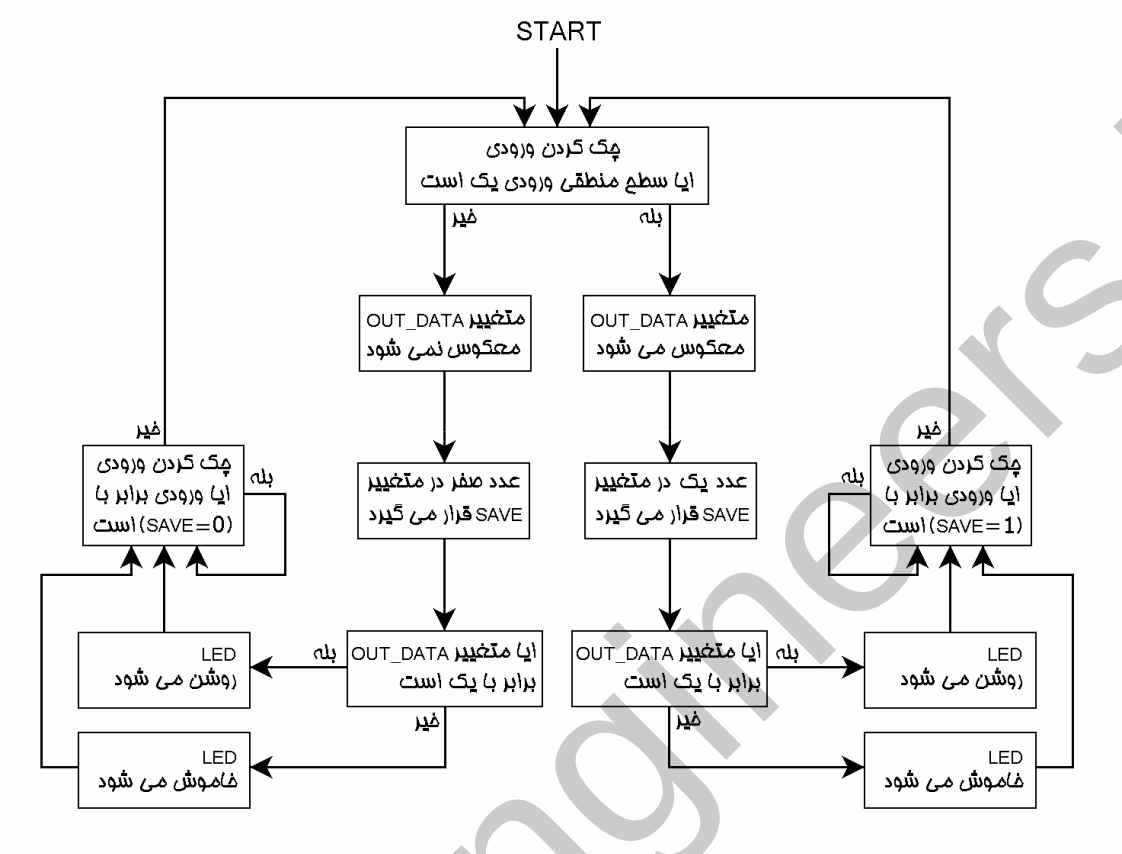
۱۷- با اجرای این دستور مقدار PINA.0 در متغیر INPUT-PIN قرار می گیرد .

۱۸- با اجرای این دستور دور بعدی حلقه WHILE-WEND اجرا خواهد شد .

۱۹- با اجرای این دستور دور بعدی حلقه DO-LOOP اجرا می شود .

در حالت کلی برای تنظیم حساسیت ورودی به لبه بالا رونده ، ورودی باید دائما چک شود و بلافاصله در زمان یک شدن دستورالعمل های مربوطه اجرا شود سپس بایستی اجرای برنامه تا صفر شدن پالس ورودی متوقف شود ، و میکروکنترلر دوباره شروع به چک کردن ورودی نماید و این روند برای همیشه ادامه پیدا کند.

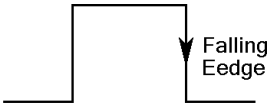
روند اجرای برنامه بالا در شکل 15-2 نشان داده شده است .



شکل2-15 نحوه اجرای برنامه حساس کردن کلید ورودی به لبه بالا رونده در سیمولاتور پروتئوس

حساس به لبه پایین رونده :

در این حالت برنامه بایستی به گونه ای نوشته شود که پایه ورودی در لبه پایین رونده پالس تحریک شود لبه پایین رونده پالس ورودی در شکل زیر نشان داده شده است .



شکل 2-16 لبه پایین رونده (FALLING EDGE) پالس ورودی

برای درک بهتر این موضوع مدار میکروکنترلری شکل 2-13 را در نظر بگیرید دراین مدار با یک بار فشار کلید LED خروجی روشن شده و با فشار آن برای بار دوم خاموش می شود ، ورودی حساس به لبه پایین رونده بوده و با فشار کلید و نگه داشتن آن در حالت بسته هیچ تغییری در وضعیت LED رخ نخواهد داد فقط زمانی LED خاموش یا روشن می شود که کلید از وضعیت بسته به وضعیت باز برگردد.

برنامه نوشته شده در محیط BASCOM به صورت زیر است .

```
1-$regfile = "M32DEF.DAT"
2-$crystal = 1000000
3-Config Pina.0 = Input
4-Config Pina.1 = Output
5-Led Alias Porta.1
6-Do
7-If Pina.0 = 1 Then
8-Bitwait Pina.0,reset
9-Togle Led
10-End If
11-Loop
```

توضیح برنامه :

برنامه به گونه ای نوشته شده که به محض یک شدن ورودی میکروکنترلر شروع به چک کردن ورودی می کند و تا زمانی که ورودی صفر نشده به چک کردن ورودی ادامه خواهد داد ، پس از صفر شدن ورودی بلافاصله و ضعیت LED را معکوس خواهد کرد .

۱- تعریف نام میکرو

۲- تعریف فرکانس کاری میکرو

۳- پیکره بندی PINA.0 به عنوان ورودی

۴- پیکره بندی PINA.1 به عنوان خروجی

۵- تغییر نام PORTA.1 به LED

۶- شروع حلقه DO_LOOP

۷- از دستورالعمل IF استفاده شده (اگر پین A.0 برابر با 1 باشد دستور نوشته شده در خط بعدی اجرا خواهد شد.)

۸- از دستور BITWAIT برای ایجاد تاخیر تا زمانی که PINA.0 برابر با صفر شود استفاده شده است .

۹- با اجرای این دستور وضعیت LED معکوس می شود .

۱۰- دستورالعمل END IF به معنای پایان دستور IF می باشد .

۱۱- با اجرای این دستور دور بعدی حلقه DO_LOOP اجرا می شود .

توجه داشته باشید که برنامه فوق برای آزمایش عملی نوشته شده و توسط سیمولاتور پروتئوس قابل اجرا نمی باشد ، اگر از سیمولاتور پروتئوس استفاده می کند می توانید از برنامه زیر برای حساس کردن ورودی به لبه پایین رونده استفاده کنید .

برنامه نوشته شده در محیط BASCOM به شرح زیر است .

```
1-$regfile = "M32DEF.DAT"
2-$crystal = 1000000
3-Config Pina.0 = Input
```

```
4-Config Pina.1 = Output
5-Dim Out_data As Bit , Save As Bit , Input_pin As Bit
6-Led Alias Porta.1
7-Do
8-If Pina.0 = 1 Then Out_data = Not Out_data
9-Save = Pina.0
10-Input_pin = Pina.0
11-While Save = Input_pin
12-Input_pin = Pina.0
13-Wend
14-If Out_data = 1 Then
15-Set Led
16-Else
17-Reset Led
18-End If
19-Loop
```

۱- تعریف نام میکرو

۲- تعریف فرکانس کاری میکرو

۳- پیکره بندی پین ورودی

۴- پیکره بندی پین خروجی

۵- تعریف متغیر های برنامه

۶- تغییر نام PORTA.1 به LED

۷- شروع حلقه DO-LOOP

۸- دستورالعمل IF (اگر پین A.0 برابر با یک است متغیر OUT-DATA معکوس شود)

۹- مقدار PINA.0 در متغیر SAVE قرار می گیرد .

۱۰- مقدار PINA.0 در متغیر INPUT-PIN قرار می گیرد .

۱۱- شروع حلقه WHILE-WEND (حلقه تا وقتی ادامه پیدا می کند که INPUT-PIN=PINA.0)

۱۲- مقدار PINA.0 در متغیر INPUT-PIN قرار می گیرد .

۱۳- پایان حلقه WHILE-WEND (با اجرای این دستور اگر INPUT-PIN=PINA.0 باشد دور بعدی حلقه اجرا می شود)

۱۴- دستورالعمل IF (اگر OUT-DATA=1 است دستورالعمل خط بعدی اجرا شود)

۱۵- با اجرای این دستور LED روشن می شود .

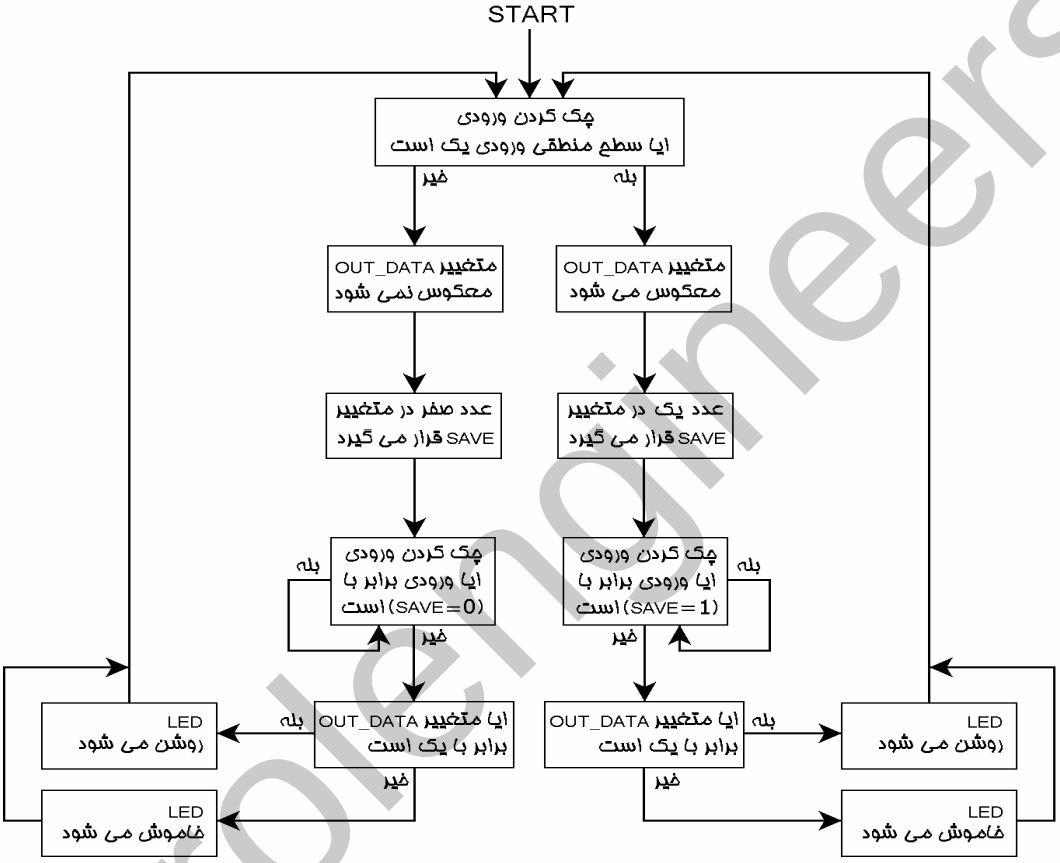
۱۶- مربوط به دستورIF (اگر OUT-DATA مخالف یک است دستورالعمل خط بعدی اجرا می شود)

۱۷- با اجرای این دستور LED خاموش می شود .

۱۸- پایان دستور شرطی IF

۱۹- با اجرای این دستور دور بعدی حلقه DO-LOOP اجرا می شود .

به طور کلی برای تنظیم حساسیت ورودی در لبه پایین رونده ، ورودی باید دائما چک شود و در زمان یک شدن ورودی اجرای برنامه تا زمان صفر شدن آن متوقف شود و بلافاصله پس از صفر شدن دستورالعمل های مربوطه اجرا شود ، سپس میکروکنترلر دوباره شروع به چک کردن ورودی نماید و این روند برای همیشه ادامه پیدا کند .



شکل 2-17 نحوه اجرای برنامه حساس کردن کلید ورودی به لبه پایین رونده در سیمولاتور پروتئوس

محافظت ورودی در برابر نویزهای لحظه ای :

نویز های لحظه ای معمولا در محیط های صنعتی ایجاد می شوند و دارای منابع مختلفی هستند برای مثال قطع و وصل یک رله مکانیکی می تواند نویز لحظه ای ایجاد کند .

پایه های ورودی میکروکنترلر AVR دارای حساسیت بالایی بوده و یک نویز لحظه ای می تواند سطح منطقی ورودی را برای مدت زمان بسیار کوتاه HIGH یا LOWE کند . برای رفع این مشکل به صورت نرم افزاری می توان سطح منطقی ورودی را چندین بار چک کرد .

برای درک بیشتر این موضوع مثال زیر را در نظر بگیرید . در این مثال نیز از مدار شکل 2-13 استفاده شده است .

در این مدار ورودی حساس به لبه بالا رونده بوده و با یک بار فشار کلید LED روشن و با فشارآن برای بار دوم LED خاموش می شود.

توجه داشته باشید میکروکنترلر ورودی را زمانی یک در نظر می گیرد که کلید به مدت 100ms بسته بماند بدین ترتیب ، اثر نویز های لحظه ای که مدت زمان آنها بسیارکمتر از 100ms می باشد، حذف خواهد شد.

برنامه نوشته شده در محیط BASCOM به شرح زیر است :

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Config Pina.1 = Output
Config Pina.0 = Input
Dim Count As Byte , A As Byte , Input_pin As Bit , Out_data As Bit , Save As Bit
Led Alias Porta.1
Do
'START OF INPUT BUTTON CHECK-----------------------------------------------------------------------------
For A = 1 To 10 Step 1
  If Pina.0 = 1 Then Incr Count
  Waitms 10
Next A
If Count > 9 Then
  Count = 0
  Input_pin = 1
Else
  Input_pin = 0
End If
'END OF INPUT BUTTON CHECK-------------------------------------------------------------------------------
If Input_pin = 1 Then Out_data = Not Out_data
Save = Input_pin
If Out_data = 1 Then
  Set Led
Else
  Reset Led
End If
A1:
'START OF INPUT BUTTON CHECK-----------------------------------------------------------------------------
For A = 1 To 10 Step 1
  If Pina.0 = 1 Then Incr Count
  Waitms 10
Next A
If Count > 9 Then
  Count = 0
  Input_pin = 1
Else
  Input_pin = 0
End If
'END OF INPUT BUTTON CHECK-------------------------------------------------------------------------------
If Input_pin = Save Then Goto A1
Loop
'-----------------------------------------------------------------------------------------------------------
```

توضیحات مربوط به دستورات برنامه در مثالهای قبلی ارئه شده است .

نکته مهم در رابطه با این برنامه نحوه دریافت ورودی است بدین صورت که در هر بار چک کردن پین A.0 به تعداد 10 بار و با فاصله زمانی 10 میلی ثانیه خوانده می شود . اگر در هر 10 بار مقدار خوانده شده 1 باشد کلید ورودی بسته در نظر گرفته شده و دستورات مربوطه اجرا می شود .

نحوه ایجاد تاخیر در حین دریافت ورودی :

گاهی اوقات لازم است که ورودی دائما چک شود از طرفی ممکن است در حین اجرای برنامه از دستورات WAIT استفاده کرده باشیم برای درک بهتر موضوع مثال زیر را در نظر بگیرید فرض کنید یک مدار میکروکنترلری داریم در آن ورودی دائما چک می شود و با فشار کلید K1، LED خروجی به مدت 5 ثانیه روشن می شود در زمان روشن بودن LED کلید K2 دائما چک می شود ، هر لحظه که کلید K2 فشار داده شد LED خاموش می شود حتی اگر مدت زمان 5 ثانیه تمام نشده باشد .

در چنین شرایطی حداکثر تاخیر برای چک کردن ورودی 100میلی ثانیه باشد همانطور که می دانید دستورات تاخیری اجرای برنامه را به مدت زمان مشخصی متوقف می کند برای مثال اگر شما از دستور WAIT 5 در بر نامه استفاده کنید به مدت 5 ثانیه نمی توانید کلید ورودی را چک کنید .

به طور کلی برای ایجاد تاخیر در حین دریافت ورودی می توان از دو روش زیر استفاده کرد.

۱- برای ایجاد تاخیر از تایمر ها استفاده کنیم و تا زمانی که تایمر سرریز نشده ورودی را چک کنیم . (در رابطه با تایمر ها در قسمت مربوطه توضیح داده خواهد شد)

۲-ابتدا تاخیر مورد نظر به تاخیرهای 100میلی ثانیه ای تقسیم کنیم برای مثال برای ایجاد تاخیر 5 ثانیه در مثال بالا می توانیم بدین صورت عمل کنیم به جای استفاده از دستور WAIT 5 از یک حلقه FOR-NEXT با تعداد تکرار 5/100ms=50 به صورت زیر استفاده کنیم .

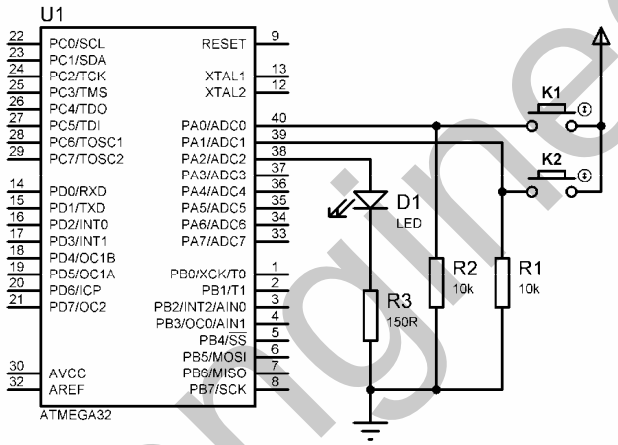
```
DIM A AS BYTE
FOR A= 1 TO 50 STEP1
WAITMS 100
IF PINX/Y = 1/0 THEN GO TO A1
NEXT A
A1:
RESET PORTX/Y
```

در برنامه بالا به جای 5 ثانیه تاخیر از 5 تا تاخیر ی 100 میلی ثانیه ای استفاده شده که بعد از هر تاخیر یک بار ورودی چک شده و دستور شرطی مربوط به یک یا صفر بودن آن اجرا می شود .

حالایک مدار میکروکنترلری طراحی می کنیم که دارای دو کلید K1 , K2 می باشد و ورودی حساس به سطح مثبت (HIGH ACTIVE) بوده و حساس به نویز لحظه ای نمی باشد و با فشار کلید K1 ، LED به مدت

10 ثانیه روشن بماند و سپس خاموش شود ، هنگامی که LED روشن است K2 دائما چک شود و هر لحظه که کلید K2 فشار داده شد LED خاموش شود حتی اگر زمان تاخیر 10 ثانیه تمام نشده باشد .

برای این که ورودی حساس به نویز لحظه ای نباشد بایستی چندین بار چک شود ، سپس یک یا صفر بودن ورودی تایید شود برای این کار می توانیم از 10 بار چک کردن با فاصله زمانی 10 میلی ثانیه ای استفاده کنیم که در این صورت کل زمان چک کردن برابر 100 میلی ثانیه خواهد بود . سخت افزار مدار در شکل 18-2 نشان داده شده است .



شکل 18-2 شماتیک طراحی شده برای مدار دریافت کننده ورودی در حین ایجاد تاخیر

در برنامه زیر ابتدا کلید K1 با استفاده از روش گفته شده چک می شود ، اگر برابر با یک بود LED روشن شده و به جای استفاده از دستور WAIT 10 برای مدت زمان روشن بودن LED ابتدا زمان 10 ثانیه را به مدت زمان چک کردن ورودی یعنی 100 میلی ثانیه تقسیم می کنیم 10/100mS=100 سپس برای بوجود آوردن زمان تاخیر 10 ثانیه، کلید ورودی K2 را 100 بار چک می کنیم در این حالت هم کلید K2 را چک کردیم و هم زمان تاخیر را بوجود آورده ایم .

متن برنامه نوشته شده به صورت زیر است :

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Config Pina.0 = Input
Config Pina.1 = Input
Config Pina.2 = Output
Dim Count As Byte , A As Byte , Input_pin As Bit
Dim B As Byte , C As Byte , D As Byte
Led Alias Porta.2
Do
A2:
```

```
'START OF K1 BUTTON CHECK-----------------------------------------------------------------------------
For A = 1 To 10 Step 1
  If Pina.0 = 1 Then Incr Count
  Waitms 10
Next A
If Count > 9 Then
  Count = 0
  Input_pin = 1
Else
  Count = 0
  Input_pin = 0
End If
'END OF K1 BUTTON CHECK-------------------------------------------------------------------------------
If Input_pin = 1 Then
  Set Led
Else
  Goto A2
End If
'100*100mS=10S
For B = 1 To 100 Step 1
'START OF K2 BUTTON CHECK-----------------------------------------------------------------------------
For A = 1 To 10 Step 1
'10*10mS=100mS
  If Pina.1 = 1 Then Incr Count
  Waitms 10
Next A
If Count > 9 Then
  Count = 0
  Input_pin = 1
Else
  Count = 0
  Input_pin = 0
End If
'END OF K2 BUTTON CHECK-------------------------------------------------------------------------------
If Input_pin = 1 Then Goto A1
Next B
A1:
Reset Led
Loop
'-----------------------------------------------------------------------------------------------------
```

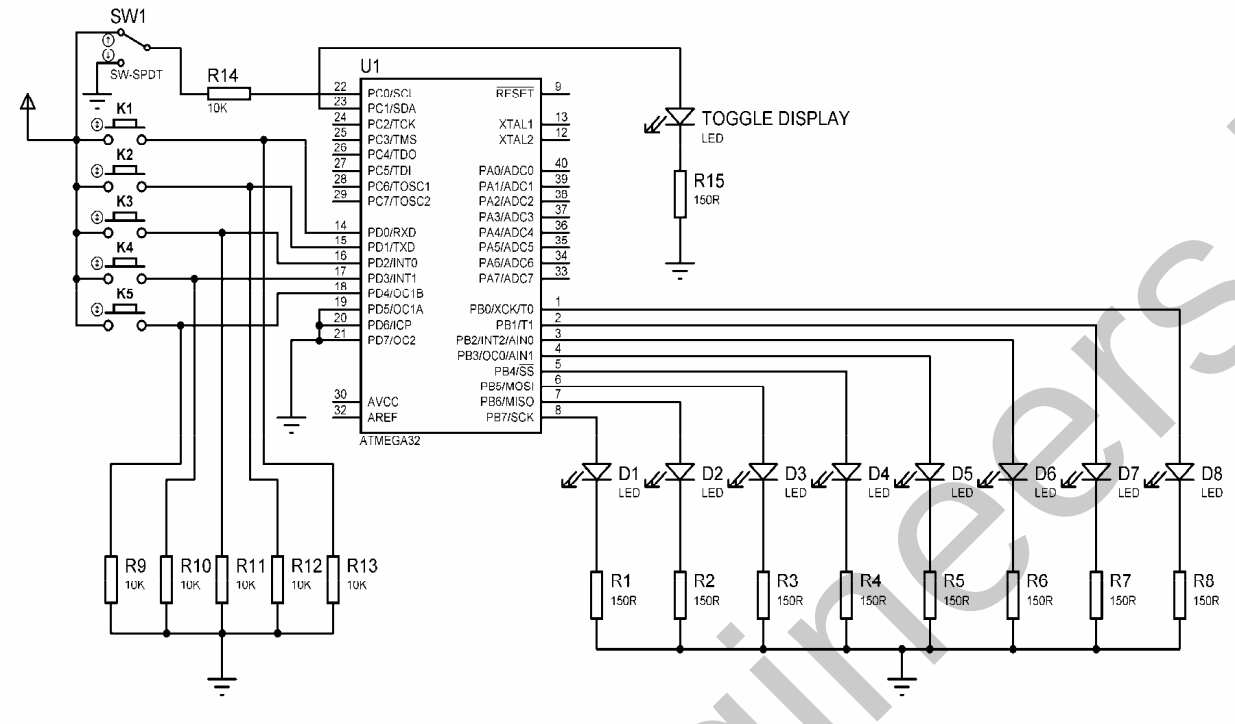
گاهی نیاز است یادداشتهایی برای اطلاعات بیشتر در برنامه اضافه کنیم در این حالت از ' یا REM استفاده خواهیم کرد . یادداشت ها و نوشته های بعد از این دستور غیر فعال بوده و توسط کامپایلر ، کامپایل نخواهد شد .

توجه داشته باشید که یادداشت های اختیاری BASCOM به رنگ سبز در می آیند .

در رابطه با دستورات استفاده شده قبلا توضیح داده شده است .

طراحی پروژه رقص نور میکروکنترلری :

حالا می خواهیم پروژه ای را طراحی کنیم که دارای 8 ، LED خروجی ، 5 کلید فشاری و یک کلید دو حالته می باشد سخت افزار پروژه در شکل 19-2 نشان داده شده است .



شکل 2-19 سخت افزار طراحی شده برای رقص نور میکروکنترلری

در سخت افزار فوق با فشار کلید K1 اگر کلید SW1 در حالت HIGH قرار داشته باشد LED ها به صورت یک نقطه نورانی از سمت راست به چپ حرکت می کنند . در این حالت مراحل حرکت نقطه نورانی به صورت زیر است .

	LED1	LED2	LED3	LED4	LED5	LED6	LED7	LED8
مرحله ۱	○	○	○	○	○	○	○	●
مرحله ۲	○	○	○	○	○	○	●	○
مرحله ۳	○	○	○	○	○	●	○	○
مرحله ۴	○	○	○	○	●	○	○	○
مرحله ۵	○	○	○	●	○	○	○	○
مرحله ۶	○	○	●	○	○	○	○	○
مرحله ۷	○	●	○	○	○	○	○	○
مرحله ۸	●	○	○	○	○	○	○	○

شکل 2-20 مراحل حرکت نقطه نورانی پس از فشار کلید K1

پس از اجرای مرحله 8 دوباره نقطه نورانی از مرحله یک شروع و تا مرحله 8 ادامه پیدا می کند . این روند در حالتی بود که کلید دو حالته SW1 در حالت HIGH قرار داشت اگر کلید SW1 را در حالت LOW قرار دهیم LED ها به صورت یک حفره از راست به چپ حرکت خواهند کرد . البته TAGGLE DISPLAY، LED نشان دهنده نقطه یا حفره بودن حرکت LED ها می باشد ، بدین صورت که اگر کلید SW1 در حالت HIGH

قرار داشته باشد TAGGLE DISPLAY روشن و اگر این کلید در حالت LOW باشد LED TAGGLE DISPLAY خاموش خواهد بود .

هنگامی که SW1 را در حالت LOW قرار داده و کلید K1 را فشار دهیم حرکت حفره به صورت زیر خواهد بود .

	LED1	LED2	LED3	LED4	LED5	LED6	LED7	LED8
مرحله ۱	●	●	●	●	●	●	●	○
مرحله ۲	●	●	●	●	●	●	○	●
مرحله ۳	●	●	●	●	●	○	●	●
مرحله ۴	●	●	●	●	○	●	●	●
مرحله ۵	●	●	●	○	●	●	●	●
مرحله ۶	●	●	○	●	●	●	●	●
مرحله ۷	●	○	●	●	●	●	●	●
مرحله ۸	○	●	●	●	●	●	●	●

شکل 2-21 مراحل حرکت حفره نوری پس از فشار کلید K1

اگر SW1 را در حالت HIGH قرار داده و کلید K2 را فشار دهیم LED ها به صورت یک نقطه نورانی از سمت چپ به راست حرکت خواهند کرد . در این حالت مراحل حرکت نقطه نورانی به صورت شکل 2-22 خواهد بود .

	LED1	LED2	LED3	LED4	LED5	LED6	LED7	LED8
مرحله ۱	●	○	○	○	○	○	○	○
مرحله ۲	○	●	○	○	○	○	○	○
مرحله ۳	○	○	●	○	○	○	○	○
مرحله ۴	○	○	○	●	○	○	○	○
مرحله ۵	○	○	○	○	●	○	○	○
مرحله ۶	○	○	○	○	○	●	○	○
مرحله ۷	○	○	○	○	○	○	●	○
مرحله ۸	○	○	○	○	○	○	○	●

شکل 2-22 مراحل حرکت نقطه نورانی پس از فشار کلید K2

در این حالت اگر SW1 را از حالت HIGH به حالت LOW ببریم ، LED ها به صورت یک حفره از سمت چپ به راست حرکت خواهند کرد . به طور کلی عملکرد کلید SW1 به این صورت خواهد بود که با LOW بودن آن خروجی یک بار معکوس شده سپس روی LED ها قرار می گیرد و با HIGH بودن آن خروجی بدون تغییر روی پورت مربوط به LED ها قرار می گیرد .

اگر SW1 را در حالت HIGH قرار داده و کلید K3 را فشار دهیم مراحل حرکت نقطه نورانی به صورت شکل 2-23 خواهد بود .

	LED1	LED2	LED3	LED4	LED5	LED6	LED7	LED8
مرحله ۱	○	○	○	●	●	○	○	○
مرحله ۲	○	○	●	○	○	●	○	○
مرحله ۳	○	●	○	○	○	○	●	○
مرحله ۴	●	○	○	○	○	○	○	●
مرحله ۵	○	●	○	○	○	○	●	○
مرحله ۶	○	○	●	○	○	●	○	○
مرحله ۷	○	○	○	●	●	○	○	○

شکل 23-2 مراحل حرکت نقطه نورانی پس از فشار کلید K3

پس از اجرا ی آخرین مرحله دو باره مرحله (1) اجرا خواهد شد البته اگر یکی از کلید ها ی K1 ، K2 فشار داده نشود که در این صورت خروجی مربوط به کلید فشار داده شده را خواهیم داشت . میزان تاخیر بین هر مرحله توسط کلید K5,K4 تنظیم می شود ، در حالت پیش فرض مقدار تاخیربین هر دو مرحله 200ms می باشد با هر بار فشار کلید K4 ، 100ms به زمان تاخیر اضافه شده و با هر بار فشار کلید K5 ، 100MS از زمان تاخیر کم می شود مینیمم زمان تاخیر 100ms و ماکزیمم آن 2S خواهد بود .

برنامه نوشته شده برای این پروژه به صورت زیر است :

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Recive_data As Byte , Hossein As Byte , Out_data As Byte , B As Byte
Dim Delay_time As Word , Toggle_pin As Bit , A As Byte , Chek As Byte
Dim D As Byte , C As Byte
Declare Sub Left_rotate
Declare Sub Right_rotate
Declare Sub Open_close
Declare Sub Out_sub
Declare Sub In_sub
Config Portd = Input
Config Portb = Output
Config Pinc.0 = Input
Config Pinc.1 = Output
Toggle_display Alias Portc.1
'START OF LEVEL1-----------------------------------------------------------------------------------------------------
Delay_time = 2
If Pinc.0 = 1 Then
  Toggle_pin = 0
  Set Toggle_display
Else
  Toggle_pin = 1
  Reset Toggle_display
End If
'END OF LEVEL1-------------------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------------------
Do
Start_program:
Recive_data = Pind
Select Case Recive_data
  Case Is = &B00000001
  Hossein = 1
```

```
  Case Is = &B00000010
  Hossein = 2
  Case Is = &B00000100
  Hossein = 3
  Case Is = &B00001000
  If Delay_time < 20 Then Incr Delay_time
  Case Is = &B00010000
  If Delay_time > 1 Then Decr Delay_time
End Select
If Pinc.0 = 1 Then
  Toggle_pin = 0
  Set Toggle_display
Else
  Toggle_pin = 1
  Reset Toggle_display
End If
If Hossein = 1 Then Call Left_rotate
If Hossein = 2 Then Call Right_rotate
If Hossein = 3 Then Call Open_close
Waitms 50
Loop
'END OF LEVEL2-----------------------------------------------------------------------------------------------------
'START OF LEFT_ROTATE SUB-------------------------------------------------------------------------------------------
Sub Left_rotate
Out_data = &B00000001
If Toggle_pin = 1 Then
  Out_data = Not Out_data
  Portb = Out_data
Else
  Portb = Out_data
End If
Chek = Hossein
'START OF DELAY TIME-----------------------------------------------
For D = 1 To Delay_time Step 1
Waitms 100
Call In_sub
Next D
'END OF DELAY TIME-----------------------------------------------
For A = 1 To 7 Step 1
Rotate Out_data , Left
Portb = Out_data
'START OF DELAY TIME-----------------------------------------------
For D = 1 To Delay_time Step 1
Waitms 100
Call In_sub
Next D
'END OF DELAY TIME-----------------------------------------------
Next A
Return
End Sub
'END OF LEFT_ROTATE SUB-------------------------------------------------------------------------------------------
'START OF RIGHT_ROTATE SUB------------------------------------------------------------------------------------------
Sub Right_rotate
Out_data = &B10000000
If Toggle_pin = 1 Then
  Out_data = Not Out_data
  Portb = Out_data
Else
  Portb = Out_data
End If
Chek = Hossein
```

```
'START OF DELAY TIME-----------------------------------------------
For D = 1 To Delay_time Step 1
Waitms 100
Call In_sub
Next D
'END OF DELAY TIME-----------------------------------------------
For A = 1 To 7 Step 1
Rotate Out_data , Right
Portb = Out_data
'START OF DELAY TIME-----------------------------------------------
For D = 1 To Delay_time Step 1
Waitms 100
Call In_sub
Next D
'END OF DELAY TIME-----------------------------------------------
Next A
Return
End Sub
'END OF RIGHT_ROTATE SUB-----------------------------------------------------------------------------------
'START OF OPEN_CLOSE SUB-----------------------------------------------------------------------------------
Sub Open_close
Out_data = &B00011000
Chek = Hossein
Call Out_sub
Out_data = &B00100100
Call Out_sub
Out_data = &B01000010
Call Out_sub
Out_data = &B10000001
Call Out_sub
Out_data = &B01000010
Call Out_sub
Out_data = &B00100100
Call Out_sub
Return
End Sub
'END OF OPEN_CLOSE SUB-----------------------------------------------------------------------------------
'START OF OUT_SUB-----------------------------------------------------------------------------------
Sub Out_sub
If Toggle_pin = 1 Then Out_data = Not Out_data
Portb = Out_data
'START OF DELAY TIME-----------------------------------------------
For D = 1 To Delay_time Step 1
Waitms 100
Call In_sub
Next D
'END OF DELAY TIME-----------------------------------------------
Return
End Sub
'END OF OUT_SUB-----------------------------------------------------------------------------------
'START OF IN_SUB-----------------------------------------------------------------------------------
Sub In_sub
Recive_data = Pind
Select Case Recive_data
  Case Is = &B00000001
  Chek = 1
  Case Is = &B00000010
  Chek = 2
  Case Is = &B00000100
  Chek = 3
  Case Is = &B00001000
```

```
  If Delay_time < 20 Then Incr Delay_time
  Case Is = &B00010000
  If Delay_time > 1 Then Decr Delay_time
End Select
If Pinc.0 = 1 Then
  Toggle_pin = 0
  Set Toggle_display
Else
  Toggle_pin = 1
  Reset Toggle_display
End If
If Chek <> Hossein Then Goto Start_program
Return
End Sub
'END OF IN_SUB-----------------------------------------------------------------------------------------------
```

دستورات جدید :

در برنامه فوق ا از دستور SELECT CASE استفاده شده است

زمانی که بخواهیم چندین شرط را روی یک مورد برسی کنیم از دستور CASE استفاده خواهیم کرد . شکل کلی دستور CASE به صورت زیر است .

```
SELECT CASE VAR
CASE TEAST1
STATEMENT1 OR STATEMENTS1
CASE TEST2
STATEMENT2 OR STATEMENTS2
.
.
CASE TESTN
STATEMENTN OR STATEMENTSN
END SELECT
```

اگر متغیر VAR با مقدار TEST1 برابر باشد STATEMENT1 یا STATEMENTS1 اجرا می شود و سپس اجرای برنامه بعد از END SELECT ادامه پیدا می کند . در غیر این صورت اگر متغیر VAR با مقدار TEST1 برابر نباشد ولی با مقدار TEST2 برابر باشد STATEMENT2 یاSTATEMENTS2 اجرا می شود و سپس اجرای برنامه بعد از END SELECT ادامه می یابد و نهایتا اگر متغیر VAR با هیچ کدام از مقادیر TEST برابر نباشد اجرای برنامه بعد از END SELECT ادامه خواهد یافت .

به جای عبارت CASE TEST می توانید از حالت های زیر استفاده کنید .

برای مثال :

```
CASE IS > 2
```

در این حالت اگر متغیر VAR بزرگتر از 2 باشد دستور یا دستورات مربوطه اجرا می شود .

```
CASE 3 TO 6
```

در این حالت اگر متغیر VAR بین 3 تا 6 باشد دستور یا دستورات مربوطه اجرا می شود .

```
CASE IS = 20
```

در این حالت اگر متغیر VAR برابر با 20 باشد دستور یا دستورات مربوطه اجرا می شود .

همچنین در برنامه فوق از زیر برنامه استفاده شده است زیر برنامه ها ابتدا بایستی معرفی شوند ، سپس توسط دستور CALL فراخوانی شوند البته توجه داشته باشید که زیر برنامه بعد از اتمام برنامه اصلی نوشته می شود .

برای معرفی زیر برنامه از دستور زیر در ابتدای برنامه استفاده می کنیم

DECLARE SUB NAME

NAME نام زیر برنامه استفاده شده می باشد .

برای نوشتن زیر برنامه ابتدا نام آن را توسط دستور زیر تعریف کرده سپس شروع به نوشتن زیر برنامه می کنیم

SUB NAME

NAME نام زیر برنامه می باشد که بایستی توسط دستور DECLARE SUB در ابتدای برنامه معرفی شده باشد . زیر برنامه با دستورات زیر تمام می شود.

RETURN
END SUB

همچنین در این برنامه از عملگر های ریاضی = و < > هم استفاده شده است . به طور کلی می توانید از عملگر های ریاضی زیر در محیط BASCOM استفاده کنید :

نماد	علامت
* ASTERISKS(MULTIPLICATION SYMBOL)	علامت ضرب
+ PLUS SIGN	علامت جمع
- MINUS SIGN	علانت تفریق
. PERIOD (DECIMAL POINT)	علامت ممیز
/ SLASH (DIVISION SYMBOL)	علامت تقسیم
> LESS THAN	علامت کوچکتر از
= EQUAL SIGN	علامت تساوی
< GREATER THAN	علامت بزرگتر از
^ EXPONENT	علامت بتوان
=> LESS THAN OR EQUAL TO	علامت کوچکتر یا مساوی با
>= GREATER THAN OR EQUAL TO	علامت بزرگتر یا مساوی با
<> INEQUALITY	علامت مخالف

جدول عملگر های ریاضی در محیط BASCOM

تشریح نحوه اجرای برنامه :

برنامه بالا توسط دستورات یادداشتی از هم تفکیک شده است در قسمت LEVEL1 متغیر DELAY_TIME برابر 2 قرار داده شده مقدار متغیر DELAY_TIME زمان تاخیر بین هر مرحله را تعیین می کند . برنامه به گونه ای نوشته شده که در حین ایجاد مدت زمان تاخیر ورودی هم چک می شود بدین صورت که پس از هر 100ms تاخیر ورودی خوانده می شود و طبق توضیحاتی که قبلا در رابطه با این تکنیک داده شده مدت زمان تاخیر با تعداد چک کردن ورودی ایجاد می شود برای مثال اگر متغیر DELAY_TIME برابر با 2 باشد ورودی دو بار و با فاصله زمانی 100ms چک می شود که در این حالت مدت زمان تاخیر برابر با 200ms خواهد بود .

مقدار ورودی خوانده شده در حالت های مختلف به صورت زیر خواهد بود .

اگر کلید K1 فشار داده شود مقدار خوانده شده برابر با &B00000001 خواهد بود

اگر کلید K2 فشار داده شود مقدار خوانده شده برابر با &B00000001 خواهد بود

اگر کلید K3 فشار داده شود مقدار خوانده شده برابر با &B00000001 خواهد بود

اگر کلید K4 فشار داده شود مقدار خوانده شده برابر با &B00000001 خواهد بود

اگر کلیدK5 فشار داده شود مقدار خوانده شده برابر با &B00000001 خواهد بود

پس از خواندن کلید های ورودی و اجرای دستور مربوط پین C.0 (کلید SW1)خوانده می شود اگر برابر با یک بود متغیر TAGGLE_PIN برابر صفر می شود و همچنین TAGGLE DISPLAY LED روشن می شود و اگر برابر با صفر بود متغیر TAGGLE_PIN برابر بایک شده و TAGGLE DISPLAY LED خاموش می شود .

در قسمت LEVEL 2 کل پورت D به عنوان ورودی خوانده می شود و در متغیر RECIVE_DATA قرار می گیرد . اگر RECIVE_DATA برابر با مقدار باینری &B0000001 باشد زیر برنامه LEFT_ROTATE فراخوانی می شود و اگر RECIVE-DATA برابر با &B00000010 باشد زیر برنامه RIGHT_ROTATE فراخوانی می شود .

اگر RECIVE_DATA برابر با &B00000100 باشد زیر برنامه OPEN_CLOSE فراخوانی می شود .

اگر RECIVE_DATA برابر با &B00010000 باشد متغیر DELAY_TIME چک می شود و اگرکوچکتر از 20 باشد یک واحد به DELAY_TIME اضافه می شود .

اگر RECIVE_DATA برابر با &B00010000 باشد متغیر DELAY_TIME چک می شود و اگر بزرگتر از یک باشد یک واحد از آن کم می شود .

در قسمت LEFT _ROTATE SUB (زیر برنامه LEFT_ROTATE) ابتدا عدد &B00000001 در متغیر OUT_DATA قرار می گیرد سپس TAGGLE_PIN چک می شود اگر برابر با یک بود متغیر OUT_DATA معکوس شده و روی پورت B قرار می گیرد و اگر برابر با صفر بود متغیر OUT_DATA بدون تغییر روی پورت B قرار می گیرد سپس متغیر CHEK برابر با متغیر HOSSEIN قرار داده می شود با توجه به این که مقدار اولیه متغیر CHEK برابر با صفر می باشد در نتیجه پس از فراخوانی زیر برنامه IN-SUB در قسمت DELAY TIME (چک کردن ورودی در حین ایجاد تاخیر). اگر کلید k1 مجددا فشار داده شود متغیر HOSSEIN با مقدار صفر مقایسه شده ، روند اجرای برنامه دچار مشکل خواهد شد . وقتی که متغیر CHEK برابر با متغیر HOSSEIN قرار داده می شود . تا وقتی ک یکی از کلید های K2 یا K3 فشار داده نشوند زیر برنامه LEFT-ROTATE اجرا خواهد شد . چون با فشار هر یک از این کلید ها محتوای متغیر CHEK تغییر می کند و طبق دستور

```
IF CHEK<> HOSSEIN THEN GOTO START_PROGRAM
```

در زیر برنامه IN_SUB اجرای برنامه به لیبل START_PROGRAM منتقل خواهد شد . پس از دستور CHEK=HOSSEIN تایم تاخیر شروع می شود که مقدار تاخیر نیز توسط متغیرDELAY_TIME تعیین می شود. پس از پایان تاخیر به تعداد 7 بار متغیر OUT_DATA به چپ شیفت داده شده و روی پورت B قرار گرفته و تایم تاخیر اجرا می شود . در این حالت اگر متغیر OUT_DATA توسط کلید SW1 معکوس نشده باشد اعدادی که روی پورت B قرار می گیرند به ترتیب زیر خواهند بود :

```
&B00000001
&B00000010
&B00000100
&B00001000
&B00010000
.
.
&B10000000
```

پس از7 بار چرخش به چپ اگرهیچ کلیدی فشرده نشود اجرای برنامه به خط بعدی دستور CALL LEFT_ROTATE که در این برنامه به صورت زیر می باشد

```
IF HOSSEIN=1 THEN CALL LEFT_ROTATE
```

منتقل می شود که آن هم در قسمت LEVEL2 در داخل حلقه DO_LOOP قرار دارد در این قسمت ورودی باز هم چک می شود اگر هیچ کلیدی فشار داده نشود محتوای متغیر HOSSEIN هیچ تغییری نکرده و زیر برنامه LEFT_ROTATE دوباره فراخوانی می شود . در قسمت RIGHT_ROTATE SUB روند اجرای برنامه شبیه قسمت LEFT_ROTATE SUB می باشد با این تفاوت که مقدار اولیه OUT_DATA برابر با &B10000000 بوده و به سمت راست شیفت چرخشی داده می شود .

در قسمت OPEN_CLOSE SUB ، ابتدا متغیر OUT_DATA برابر با &B00011000 قرار می گیرد سپس زیر برنامه OUT_SUB فراخوانی می شود درزیر برنامهOUT_SUBوضعیت متغیرTAGGLE_PIN مورد برسی قرار می گیرد . بدین صورت که اگر برابر با یک بود ابتدا محتوای متغیر OUT_DATA معکوس می شود سپس روی پورت B قرار می گیرد وگرنه متغیر OUT_DATA بدون تغییر روی پورت B قرار خواهد گرفت .

در این حالت محتوای متغیر OUT_DATA به ترتیب برابر با مقادیر زیر خواهد بود

```
&B00011000
&B00100100
&B01000010
&B10000001
&B01000010
&B00100100
```

پس از این که آخرین مقدار OUT_DATA روی پورت B قرار گرفت اجرای برنامه به خط بعدی دستور CALL OPEN_CLOSE که در این برنامه به صورت زیر می باشد

```
IF HOSSEIN=3 THEN CALL OPEN=CLOSE
```

منتقل می شود که آن هم در قسمت LEVEL2 در داخل حلقه DO_LOOP قرار گرفته است در این قسمت ورودی باز هم چک می شود اگر هیچ کلیدی فشار داده نشود محتوای متغیر HOSSEIN هیچ تغییری نکرده و زیر برنامه OPEN_CLOSE دوباره خوانده می شود . قسمت IN_SUB برای چک کردن ورودی در زمان تاخیر در نظر گرفته شده است .

در این برنامه حرکت نقطه نورانی به 3 صورت می باشد که توسط کلید های K1 ، K2 ، K3 تعیین می شود پس از فشار هر یک از کلید ها زیر برنامه مربوطه اجرا می شود .

در حین اجرای زیر برنامه ورودی هم چک می شود (توسط زیر برنامه IN_SUB) اگر به غیر از کلید مربوط به زیر برنامه کلید دیگری از K2,K1 یا K3 فشار داده شود ، محتوای متغیر CHEK تغییر خواهد کرد و در این صورت اجرای برنامه توسط دستور

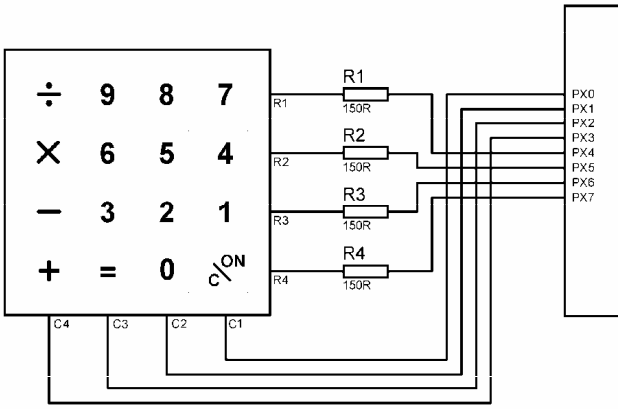
```
IF CHEK<>HOSSEIN THEN GO TO START_PROGRAM
```

به لیبل START_PROGRAM منتقل خواهد شد . البته در زیر برنامه IN_SUB کلید های SW1 ، K4 و K5 نیز چک شده و دستورات مربوط به آن ها هم اجرا می شود .

نحوه پیکره بندی LCD , KEYPAD گاراگتری در BASCOM

پیکره بندی صفحه کلید (KEYPAD) 4*4 :

برای اسکن صفحه کلید کافی است صفحه کلید خود را طبق شکل زیر به یکی از پورت های میکرو وصل نموده و توسط دستور CONFIG KBD=PORTX آن را پیکره بندی کنیم .



شکل 2-24 نحوه اتصال KEYPAD ماتریسی به یکی از پورت های میکرو

اگر پایه های صفحه کلید را بدین صورت به میکرو متصل کنیم ، با فشار هر کدام از کلید ها عدد متناظر باهمان کلید توسط دستور GETKBD( ) برگرداندنده می شود باید توجه داشت که عدد برگردانده شده هیچ ربطی به شکل ظاهری صفحه کلید ندارد . به عنوان مثال اگر صفحه کلید به صورتی که در شکل بالا مشاهده می کنید به میکروکنترلر متصل شود اعداد برگردانده شده توسط دستور GETKBD ( ) به صورت زیر خواهد بود

شکل 2-25 اعداد برگردانده شده از KEYPAD توسط دستور GETKBD

همانطور که در شکل بالا مشاهده می کنید برای مثال با فشار کلید + عدد 12 با فشار کلید X عدد 4 و با فشار کلید 8 عدد 2 برگردانده می شود .

شکل کلی دستور GETKBD ( ) به صورت زیر است .

```
SOURCE=GETKBD( )
```

SOURCE متغیری است که عدد متناظر با کلید فشرده شده در آن قرار می گیرد . توجه داشته باشید اگر هیچ کلیدی فشرده نشود عدد 16 برگردانده خواهد شد .

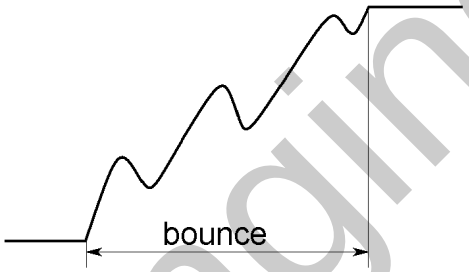
برای برگرداندن نام اصلی کلید ها بایستی از جداول LOOKUP STR و LOOKUP استفاده کنید .

صفحه کلید توسط دستور زیر پیکره بندی می شود .

CONFIG KBD=PORTX و DEBOUNCE=VALHE ، DELAY=VALUE

PORTX : مشخص کننده پورتی است که صفحه کلید به آن متصل می شود .

DEBOUNCE : با توجه به این که هر کلید فشاری دارای BOUNCE بوده و بسته به کیفیت آن ممکن است BOUNCE آن تا 20ms هم طول بکشد . بهتر است میزان BOUNCE کلید در این قسمت به میلی ثانیه تعریف شود . به عنوان مثال برای حذف میزان BOUNCE=20ms بایستی گزینه DEBOUNCE را برابر با 20 انتخاب کنیم . مقدار DEBOUNCE به صورت پیش فرض برابر با 20 و می تواند ماکزیمم مقدار 255 را داشته باشد . شکل 2-26 تصویری تقریبی از BOUNCE یک کلید را نشان می دهد .



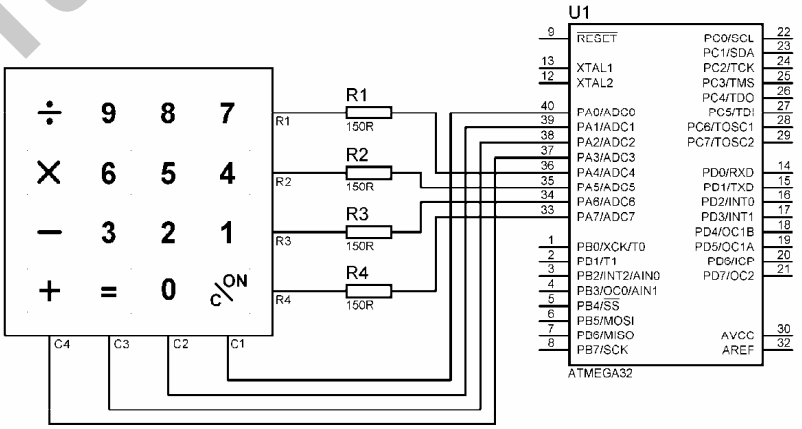
شکل 2-26 تصویری تقریبی از BOUNCE یک کلید

DELAY : مشخص کننده تاخیری بر حسب میلی ثانیه می باشد که در زمان خواندن کلید توسط دستور GETKBD( ) ایجاد می شود ، این گزینه برای کاهش نویز های لحظه ای محیط در نظر گرفته شده است و مقدار DELAY=100 می تواند این مشکل را برطرف سازد .

البته پیکر بندی فوق را می توان به صورت ساده زیر هم نوشت

CONFIG KBD=PORT X

به عنوان مثال در شکل زیر نحوه پیکره بندی بدین صورت خواهد بود



شکل 2-27 نحوه اتصال KEYPAD به پورت A

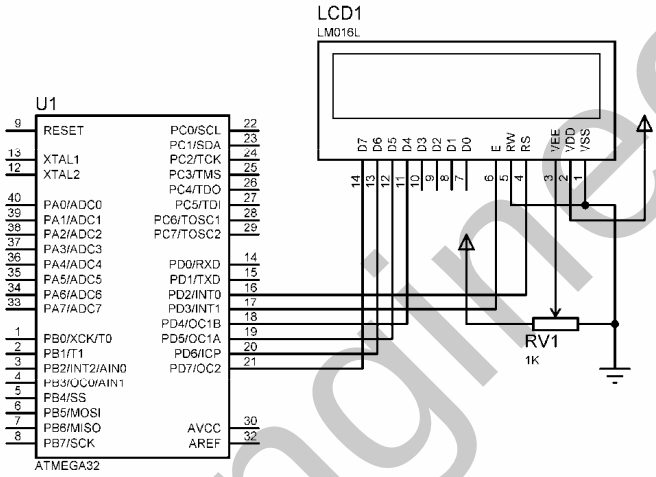
CONFIG KBD=PORT A , DEBOUNCE=50 , DELAY=100

نحوه پیکره بندی LCD کاراکتری :

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

CONFIG LCDPIN=PIN , DB4=PN1 , DB5=PN2 , DB6=PN3 , DB7=PN4 , E=PN5 _ , RS=PN6

PN1 تا PN6 پایه های دلخواه از میکروکنترلر هستند که پایه های LCD به آنها متصل می شود به مثال زیر توجه کنید .

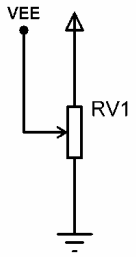


شکل 2-28 نحوه اتصال LCD کاراکتری به میکروکنترلر

برای شکل فوق پیکره بندی به صورت زیر انجام می گیرد

CONFIG LCDPIN=PIN , DB4=PIND.4 , DB5=PIND.5 , DB6=PIND.6 , _ DB7=PIND.7 , RS=PIND.2 , E=PIND.3

با اتصال یک پتانسیو متر به پایه VEE می توان میزان کنتراست صفحه LCD را کنترل کرد .



شکل 2-29 نحوه اتصال یک پتانسیومتر به پایه VEE جهت کنترل روشنایی صفحه LCD

تعین نوع LCD توسط دستور زیر انجام می شود

CONFIG LCD=LCDTYPE

LCD TYPE : نوع LCD بوده وبسته به نیاز کاربر می تواند یکی از انواع زیر باشد

2 * 16 یا 2 * 20 یا 1 * 16 یا 4 * 20

دستورات و توابع مربوط به LCD :

دستور زیر یک یا چند عبارت ثابت یا متغیر را بر روی LCD نمایش می دهد . به عنوان مثال برای نوشتن کلمه HOSSEIN بر روی LCD از دستور زیر استفاده خواهیم کرد .

LCD " HOSSEIN "

برای نوشتن مقدار متغیر B روی LCD از دستور زیر استفاده می شود

LCD B

برای نمایش چند عبارت پشت سر هم بین آن ها علامت (SEMICOLLON) قرار خواهیم داد

LCD A;B; " HOSSEIN "

از دستور CLS (CLEAR SCREEN) برای پاک کردن صفحه نمایش LCD استفاده خواهیم کرد

DISPLAY ON/OFF

برای تنظیم مکان نمای LCD از دستور زیر استفاده می شود

CURSOR ON/OFF BLINK/NOBLINK

ON : مکان نما را روشن می کند.

OFF : مکان نما را خاموش می کند .

BLINK : مکان نما به صورت چشمک زن در می آید.

NOBLINK :مکان نما به صورت چشمک زن نخواهد بود .

توسط دستور زیر مکان نما به ترتیب در اولین ستون سطر اول ، سطر دوم ، سطر سوم یا سطر چهارم قرار می گیرد

HOME UPPER/LOWER/THIRD/FOURTH

اگر دستور HOME به تنهایی نوشته شود مکان نما در سطر و ستون اول قرار می گیرد . با دستور زیر می توان مکان نما را به مکان دلخواه در صفحه LCD منتقل کرد

LOCATE X;Y

X : ثابت یا متغیری است که مشخص کننده سطر می باشد .

Y : ثابت یا متغیری است که مشخص کننده ستون می باشد .

دستور زیر کل مکان نمای LCD را یک واحد به چپ یا راست انتقال می دهد .

SHIFTLCD LEFT/RIGHT

دستور زیر کل صفحه نمایش LCD را یک واحد به چپ یا راست انتقال می دهد .

SHIFTCURSOR LEFT/RIGHT

دستور زیر مکان نما را به خط پایین تر می برد

LOWERLINE

دستور زیر مکان نما را به خط بالا تر می برد

UPPERLINE

با دستور DEFLCDCHAR می توانید حرف یا علامتی را که خودتان در منوی TOOLS و قسمت LCD DESINER در محیط BASCOM طراحی کرده اید بر روی LCD نمایش دهید پس از طراحی علامت یا حرف دلخواه در LCD DESINER 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 بسته به طراحی ، توسط نرم افزار تعیین و نوشته می شوند و شما می توانید به جای علامت ? عدد 0 تا 7 قرار دهید بدین ترتیب شما قادر خواهید بود 8 کاراکتر را به دلخواه خود طراحی کنید .

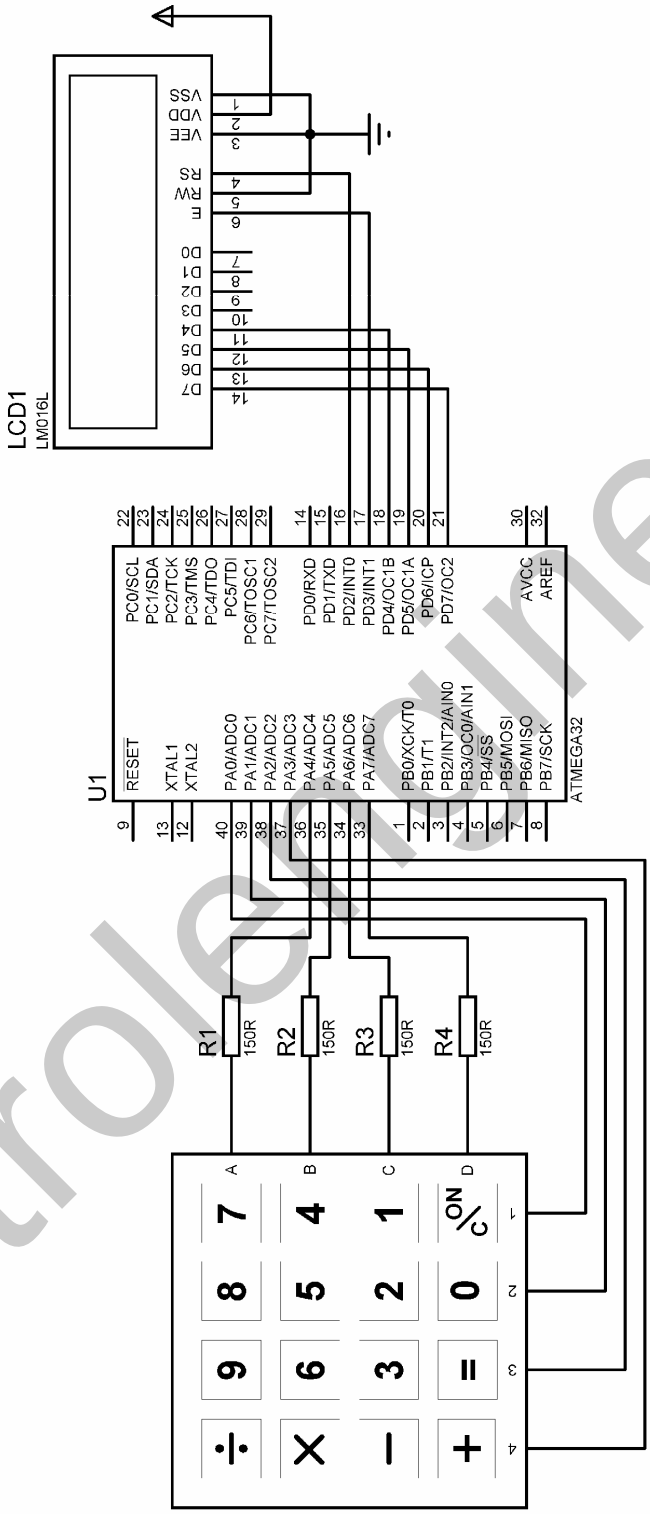
برای نمایش کاراکتر طراحی شده بر روی LCD از دستور زیر استفاده خواهیم کرد

LCDCHR (?)

? : شمارنده کاراکتر طراحی شده است که می تواند بین 0 تا 7 باشد .

پروژه نمایش نام کلید فشرده بر روی LCD :

حالا پروژه ای طراحی می کنیم که نام کلید فشرده شده را روی LCD نمایش دهد .سخت افزار پروژه در شکل 2-30 نشان داده شده است .



شکل 2-30 شماتیک پروژه نمایش نام کلید فشرده شده بر روی LCD

برنامه نوشته شده برای سخت افزار شکل 30-2 به شرح زیر است.

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Recive_data As Byte
Dim Send_data As Byte
Dim Star As String * 4
Declare Sub Main
Config Kbd = Porta , Debounce = 50 , Delay = 100
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
        , E = Pind.3 , Rs = Pind.2
'START OF LEVEL1-----------------------------------------------------------------------------------------------
Start_program:
Cursor Off
Cls : Home
Lcd "IN THE NAME OF"
Locate 2 , 7
Lcd "GOD"
Wait 5
Cls : Home
Lcd "NO INPUT"
Waitms 500
'END OF LEVEL1-------------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------------
A:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto A
Send_data = Lookup(recive_data , Data_code)
If Send_data >= 10 Then Call Main
Cls : Home
Lcd "KEY NAME IS"
Lowerline
Lcd Send_data
Goto A
'END OF LEVEL2-------------------------------------------------------------------------------------------------
'START OF MAIN SUB----------------------------------------------------------------------------------------------
Sub Main:
Send_data = Send_data / 10
Decr Send_data
Star = Lookupstr(send_data , Sdata)
Cls : Home
Lcd "KEY NAME IS"
Locate 2 , 1
Lcd Star
Goto A
Return
End Sub
'END OF MAIN SUB------------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'--------------------------------------------------------------------------------------------------------------
Sdata:
Data "%" , "X" , "-" , "ON/C" , "=" , "+"
'--------------------------------------------------------------------------------------------------------------
```

دستورات جدید :

در برنامه فوق از دستور HOME و CLS در یک خط استفاده شده است ، بطورکلی اگر بخواهید چندین دستور را در یک خط بنویسید بایستی بین آن ها از علامت : استفاده کنید برای مثال

```
CLS : HOME : LCD "GOD"
```

همچنین از جداول LOOKUP استفاده شده که فرم کلی آنها به صورت زیر است

```
VAR=LOOKUP(VALUE , LABLE)
```

LABEL نام برچسب جدول و VALUE شمارنده عدد برگردانده شده از جدول می باشد به مثال زیر توجه کنید

```
VAR=LOOKUP(VALUE,DATA_CODE)
DATA_CODE:
DATA 20,30,40
```

برای این که عدد 20 در متغیر VAR قرار بگیرد بایستی VALUE برابر با صفر باشد به عبارت دیگر عدد صفرام از لیبل DATA_CODE برابر با 20 می باشد و برای این که عدد 40 در متغیر VAR قرار گیرد بایستی VALUE برابر با 2 باشد . در برنامه فوق از جداول LOOKUPSTR نیز استفاده شده که شکل کلی آنها به صورت زیر است

```
STR=LOOKUPSTR(VALUE,LABLE)
```

STR بایستی از توع متغیر STRING باشد .

LABLE نام بر چسب جداول و VALUE شماره رشته برگردانده شده از جدول می باشد . به مثال زیر توجه کنید

```
STR=LOOKUPSTR(VALUE_DATA_CODE)
DATA_CODE:
DATA "HOSSEIN","X","+"
```

برای آن که رشته HOSSEIN در متغیر رشته ای STR قرار گیرد بایستی VALUE برابر با صفر باشد . توجه داشته باشید حداقل طول تعریف شده برای متغیر رشته ای باید بزرگتر ، مساوی طول بزرگترین رشته جدول LOOKUP باشد پس با توجه به این کهHOSSEIN یک رشته 7 حرفی است حداقل تعداد کاراکترهای متغیر STR برابر با 7 می باشد و بایستی به صورت زیر تعریف شود

```
DIM STR AS STRING*7
```

در رابطه با بقیه دستورات استفاده شده در برنامه قبلا توضیح داده شده است .

تشریح نحوه عملکرد برنامه :

در قسمت LEVEL1 ابتدا مکان نمای LCD خاموش شده وسپس صفحه نمایش پاک می شود . سپس عبارت IN THE NAME OF GOD روی صفحه LCD نمایش داده می شود . پس از 5 ثانیه تاخیر صفحه LCD پاک شده و دکمه NO INPUT روی LCD نوشته می شود .

در قسمت LEVEL 2 عدد خوانده شده از KEYPAD در متغیر RECIVE_DATA قرار می گیرد اگر هیچ کلیدی فشرده نشود عدد 16 در متغیر RECIVE_DATA قرار می گیرد که در این صورت ورودی دو باره چک می شود و روند چک کردن ورودی تا وقتی که کلیدی فشرده شود ادامه پیدا خواهد کرد .

توجه داشته باشید که این برنامه در محیط سیمولاتور پروتئوس آزمایش شده است و اعداد برگردانده شده توسط KEYPAD در محیط پروتئوس با اعداد برگردانده شده توسط KEYPAD واقعی متفاوت بوده و به صورت زیر می باشد

شکل 2-31 اعداد برگردانده شده در محیط پرتئوس توسط دستور GETKBD()

اگر کلیدی فشار داده شود اجرای برنامه از حلقه A خارج می شود . برای مثال با زدن کلید 9 عدد 2 در متغیر RECIVE_DATA قرار می گیرد و با استفاده از دستور

```
SEND_DATA=LOOKUP(RECIVE_DATA,DTA_CODE)
```

عدد شماره 2 از جدول DATA_CODE که برابر با 9 است در متغیر SEND_DATA قرار می گیرد .

سپس متغیر SEND_DATA بر روی LCD نمایش داده می شود. اگر یکی از کلید های C\ON, =,+,-,*, / فشار داده شود عددی که توسط جدول LOOKUP در متغیر SEND_DATA قرار می گیرد بزرگتر یا مساوی 10 خواهد بود که در این صورت زیر برنامه MAIN فراخوانی می شود . فرض کنید کلید + را فشار داده ایم که در این صورت عدد 60 توسط جدول LOOKUP در متغیر SEND_DATA قرار می گیرد . با توجه به دستور

```
IF SEND_DATA >= 10 THEN CALL MAIN
```

زیر برنامه MAIN فراخوانی می شود .

در زیر برنامه MAIN ابتدا محتوای متغیر SEND_DATA که برابر با 60 بود بر 10 تقسیم می شود 60/10=6 و حاصل دوباره در متغیر SEND_DATA قرار می گیرد . سپس یک واحد از متغیر SEND_DATA کم می شود که در این صورت محتوای آن برابر 5 خواهد بود و نهایتا طبق دستور زیر

STAR=100KUPSTR(SEND_DATA,SDATA)

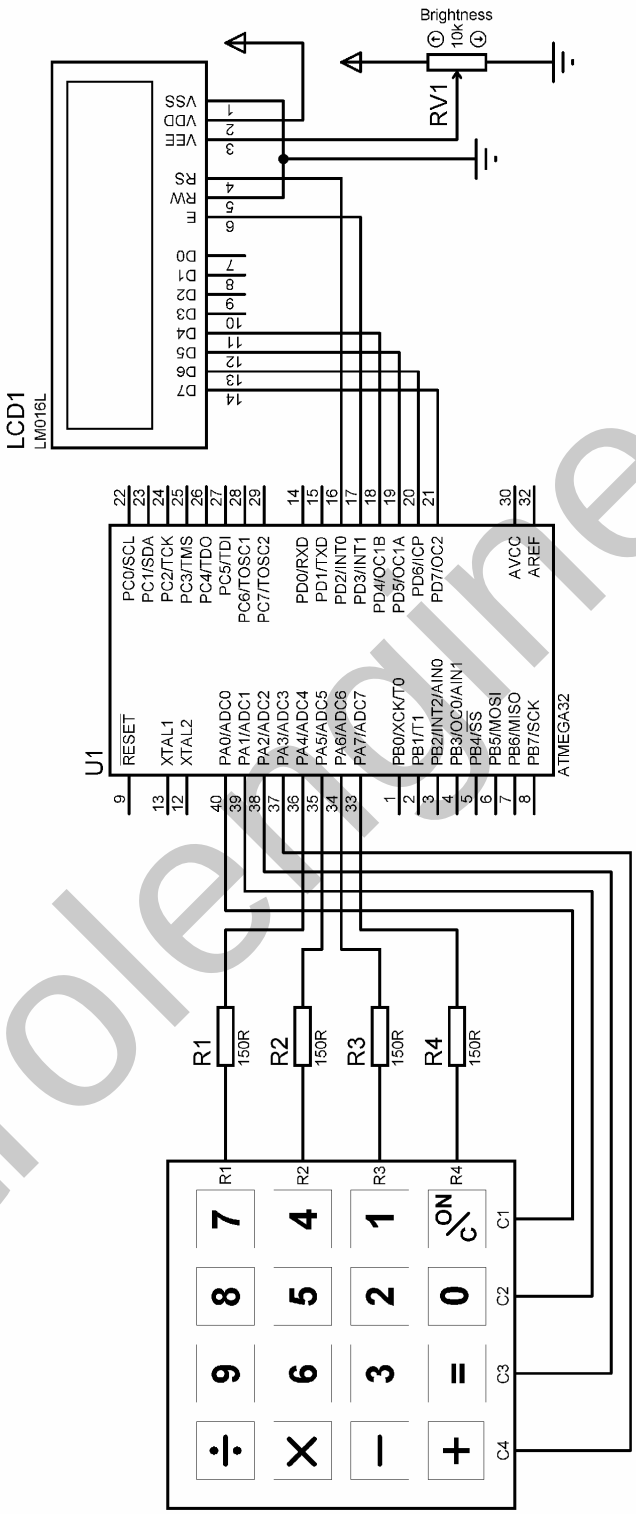
رشته شماره 5 از جدول SDATA که برابر با "+" می باشد در متغیر رشته ای STAR قرار می گیرد و سپس متغیر STAR روی LCD نمایش داده می شود .

نکته مهم :

با توجه به این که شکل ظاهری KEYPAD های موجود در بازار با هم متفاوت است ، اعداد برگردانده شده توسط دستور ()GETKBD نیز متفاوت خواهد بود . همچنین شما بایستی جدول LOOKUP مربوط به KEYPAD را با توجه به اعداد برگردانده شده توسط آن تنظیم کنید . برای اطلاع از اعداد برگردانده شده توسط دستور GETKBD می توانید از مدار شکل 2-32 استفاده کنید . در این مدار با فشار هر کلید عدد برگردانده شده توسط آن بر روی LCD نوشته می شود . شما می توانید جدول LOOKUP مربوط را بر اساس این اعداد تنظیم کنید .

برنامه نوشته شده در محیط BASCOM نیز به صورت زیر می باشد .

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Keypad_data As Byte
Config Kbd = Porta , Debounce = 20 , Delay = 100
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
        , E = Pind.3 , Rs = Pind.2
'-----------------------------------------------------------------------------------------------------------
Cls : Home : Cursor Off
Lcd "NO INPUT"
Start_program:
Keypad_data = Getkbd()
If Keypad_data = 16 Then Goto Start_program
Cls : Home
Lcd "KEY NAME IS"
Lowerline
Lcd Keypad_data
Goto Start_program
'-----------------------------------------------------------------------------------------------------------
```



شکل 2-32 شماتیک مدار نمایش عدد مربوط به کلید فشرده شده

پروژه تشخیص بزرگترین عدد دو رقمی وارد شده و نمایش آن بر روی LCD

حالا پروژه ای طراحی می کنیم که در آن از یک KEYPAD ویک LCD(16*2) استفاده شده است شما توسط KEYPAD ، 3 عدد دو رقمی A,B,C را وارد می کنید و میکرو کنترلر بزرگترین عدد وارد شده را به عنوان MAX VALUE روی LCD نشان می دهد از یک SPEAKER 8 اهمی نیز به عنوان ایجاد کننده صدای سوت کوتاه پس از زدن هر کلید استفاده شده است ، نوع صدای ایجاد شده وابسته به فرکانس است که توسط میکرو تولید می شود . نحوه عملکرد پروژه بدین صورت می باشد که ابتدا عبارت

PLEASE ENTER A

روی LCD نوشته می شود عدد دو رقمی که شما توسط KEYPAD ورودی وارد می کنید به جای علامت سئوال قرار می گیرد. اگر می خواهید عدد یک رقمی وارد کنید اول عدد 0 سپس عدد یک رقمی مورد نظرتان را وارد کنید ، و اگر می خواهید عدد دو رقمی وارد کنید ابتدا رقم دهگان سپس رقم یکان را وارد کنید سپس عبارت

PLEASE ENTER C
C=?

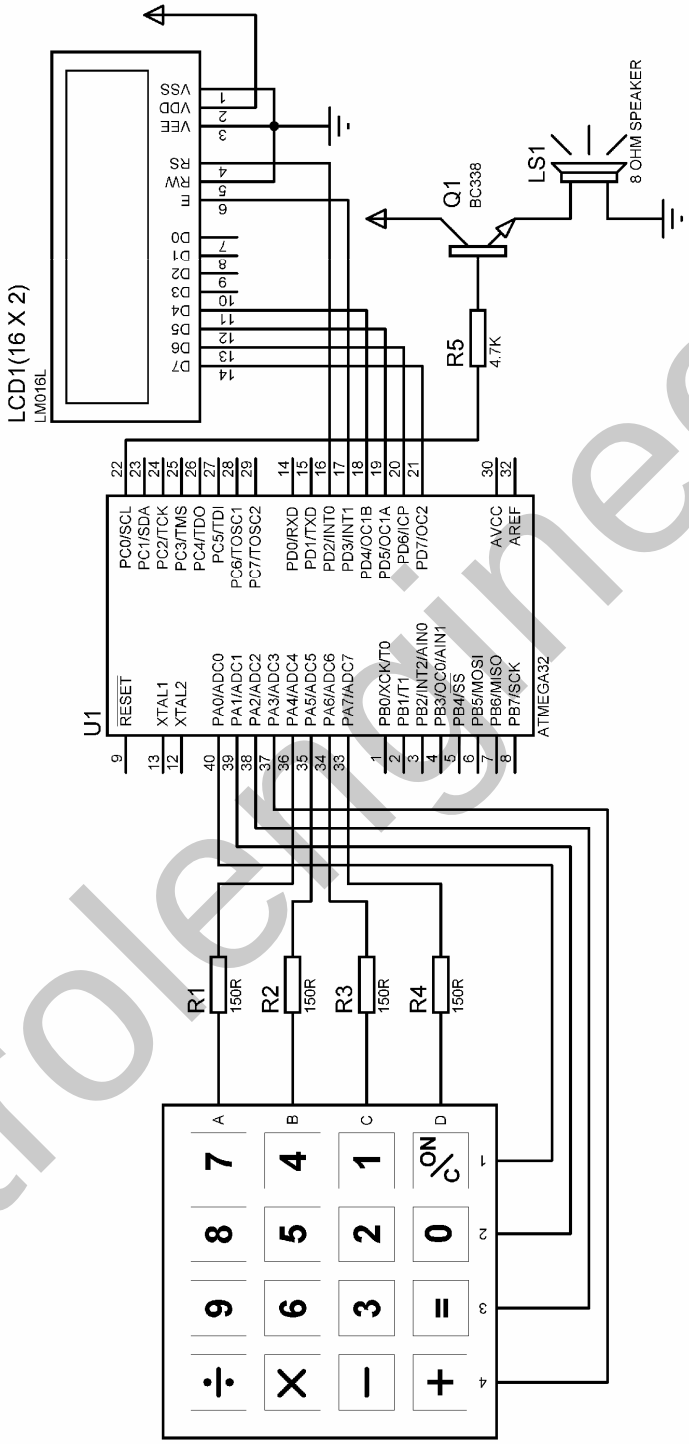
روی LCD نوشته می شود عدد دو رقمی که شما وارد می کنید به جای علامت سئوال قرار می گیرد . پس از این که هر دو رقم را وارد کردید عبارت زیر روی LCD نوشته خواهد شد

PLEASE ENTER C
C=?

عدد دورقمی که شما وارد می کنید به جای علامت سئوال قرار خواهد گرفت . پس از وارد کردن عدد C عبارت زیر روی LCD نوشته خواهد شد

MAX VALUE IS
X

که به جای X بزرگترین عدد داده شده قرار خواهد گرفت و پس از 5 ثانیه تاخیر برنامه RESET شده و دوباره عدد A از شما خواسته خواهد شد . سخت افزار پروژه در شکل 33-2 نشان داده شده است .



شکل 2-33 شماتیک طراحی شده برای تشخیص بزرگترین عدد وارد شده

همان طور که در شماتیک پروژه مشاهده می کنید از یک SPEAKER ، 8 اهمی استفاده شده که توسط یک ترانزیستور BC337,BC338 بافر شده است . برای اطلاع از مشخصات مربوط به این ترنزیستور ها به فصل 3 مراجعه نمایید .

برنامه نوشته شده برای پروژه به صورت زیر است.

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Recive_data As Byte , A1 As Byte , A2 As Byte , A As Byte
Dim Star As Byte , Count As Byte , Max1 As Byte
Dim B1 As Byte , B2 As Byte , B As Byte
Dim C As Byte , C1 As Byte , C2 As Byte
Declare Sub Main
Config Kbd = Porta , Debounce = 50 , Delay = 100
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
        , E = Pind.3 , Rs = Pind.2
Config Pinc.0 = Output

'START OF LEVEL1-----------------------------------------------------------------------------------------
Start_program:
Cursor Off
Cls : Home
Lcd "PLEASE ENTER A"
Locate 2 , 1
Lcd "A = ?"
'END OF LEVEL1-------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Star = Lookup(recive_data , Data_code)
If Star >= 10 Then Goto H1
Sound Portc.0 , 100 , 80
Incr Count
If Count = 1 Then
A1 = Star
A = A1
End If
If Count = 2 Then
A2 = Star
'A=(A1*10)+A2
A = A1 * 10
A = A + A2
End If
Cls : Home
Lcd "PLEASE ENTER A"
Locate 2 , 1
Lcd "A = " ; A
If Count = 2 Then
Count = 0
Wait 2
Goto R1
End If
Waitms 100
Goto H1
'END OF LEVEL2-------------------------------------------------------------------------------------------
```

```
'START OF LEVEL3-----------------------------------------------------------------------------------------
R1:
Cls : Home
Lcd "PLEASE ENTER B"
Locate 2 , 1
Lcd "B =?"
'END OF LEVEL3-------------------------------------------------------------------------------------------
'START OF LEVEL4-----------------------------------------------------------------------------------------
H2:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H2
Star = Lookup(recive_data , Data_code)
If Star >= 10 Then Goto H2
Sound Portc.0 , 100 , 80
Incr Count
If Count = 1 Then
B1 = Star
B = B1
End If
If Count = 2 Then
B2 = Star
'B=(B1*10)+B2
B = B1 * 10
B = B + B2
End If
Cls : Home
Lcd "PLEASE ENTER B"
Locate 2 , 1
Lcd "B = " ; B
If Count = 2 Then
Count = 0
Wait 2
Goto R2
End If
Waitms 100
Goto H2
'END OF LEVEL4-------------------------------------------------------------------------------------------
'START OF LEVEL5-----------------------------------------------------------------------------------------
R2:
Cls : Home
Lcd "PLEASE ENTER C"
Locate 2 , 1
Lcd "C =?"
'END OF LEVEL5-------------------------------------------------------------------------------------------
'START OF LEVEL6-----------------------------------------------------------------------------------------
H3:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H3
Star = Lookup(recive_data , Data_code)
If Star >= 10 Then Goto H3
Sound Portc.0 , 100 , 80
Incr Count
If Count = 1 Then
C1 = Star
C = C1
End If
If Count = 2 Then
C2 = Star
'C=(C1*10)+C2
C = C1 * 10
C = C + C2
```

```
End If
Cls : Home
Lcd "PLEASE ENTER C"
Locate 2 , 1
Lcd "C = " ; C
If Count = 2 Then
Count = 0
Wait 2
Goto H4
End If
Waitms 100
Goto H3
'END OF LEVEL6-----------------------------------------------------------------------------------------
'START OF LEVEL7---------------------------------------------------------------------------------------
H4:
Max1 = A
If B > Max1 Then Max1 = B
If C > Max1 Then Max1 = C
Cls : Home
Lcd "MAX VALUE IS"
Locate 2 , 1
Lcd Max1
Wait 5
Goto Start_program
'END OF LEVEL7-----------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'-----------------------------------------------------------------------------------------------------
```

دستورات جدید :

در برنامه بالا از دستور SOUND استفاده شده است که فرم کلی آن به صورت زیر است

SOUND PIN,DUTATION,PULSES

PIN : نام یکی از پین های است که به صورت خروجی CONFIG شده است .

DUTATION : ثابت یا متغیری است که تعداد پالس های خروجی را مشخص می کند و مقدار آن می تواند ما بین 1 تا 65535 باشد .

PULSES : ثابت یا متغیری است که مدت زمان بالا یا پایین بودن پالس خروجی را به میکرو ثانیه نشان می دهد و می تواند عددی مابین 1 تا 65535 باشد برای درک بهتر به مثال زیر توجه کنید:

```
CONFIG PORTB=OUTPUT
SOUND PORTB.7,100,20
```

تشریح نحوه عملکرد برنامه :

این برنامه توسط متن های یادداشتی به قسمت های مختلف تقسیم شده است در قسمت LEVEL1 مکان نما خاموش شده و عبارت

PLEASE ENTER A
A=?

روی LCD نوشته می شود . در قسمت LEVEL2 عدد A توسط KEYPAD دریافت می شود نحوه دریافت عدد A به صورت زیر است .

ابتدا عدد وارد شده با استفاده از KEYPAD توسط دستور () GETKB در متغیر RECIVE-DATA قرار می گیرد اگر هیچ کلیدی فشرده نشود عدد 16 در متغیر RECIVE-DATA قرار خواهد گرفت که در این صورت تا وقتی که کلیدی فشار داده نشود ورودی چک خواهد شد .

اگر کلیدی فشرده شود عدد معادل آن توسط جدول LOOKUP در متغیر STAR قرار می گیرد برای کلیدهایی که شماره نیستند مانند + , – و غیره اعداد برگردانده شده توسط جدول LOOKUP در متغیر START بزرگتر از 10 خواهد بود . که در این صورت اجرای برنامه دوباره به لیبل H1 منتقل می شود یعنی باز هم ورودی چک می شود .

اگر کلیدی فشار داده شد و محتوای متغیر START نیز کوچکتر از 10 بود یک واحد به متغیر COUNT اضافه می شود ، اگر COUNT برابر یک باشد متغیر STAR در متغیر A1 قرار می گیرد و همچنین محتوای متغیر A1 در متغیر A قرار می کیرد . ورودی LCD به جای علامت سئوال متغیر A نمایش داده می شود پس از نمایش بر روی LCD اجرای برنامه باز هم به لیبل H1 منتقل می شود اگر عددی فشار داده شود یک واحد به متغیر COUNT اضافه می شود که در این صورت محتوای متغیر COUNT برابر با 2 خواهد بود و با توجه به دستور شرطی

```
IF COUNT = 2 THEN
A2=STAR
A=A1*10
A=A+A2
END IF
```

محتوای متغیر A از فرمول زیر بدست آمده و در روی LCD نمایش داده می شود

A=A(A1*10)+A2

پس از نمایش عدد A برروی LCD با توجه به دستور شرطی

```
IF COUNT = 2 THEN
COUNT=0
WAIT 2
GO TO
```

با توجه به اینکه هر دو عدد را وارد کرده ایم متغیر COUNT برابر با دو خواهد بود که در این صورت ابتدا متغیر COUNT را برابر صفر می کنیم چون هنگام گرفتن عدد B هم از این متغیر به عنوان شمارنده عدد وارد شده

استفاده کرده ایم سپس 2 ثانیه تاخیر ایجاد می شود تا کاربرد عدد وارد شده را مشاهده کند سپس اجرای برنامه به لیبل R1 (قسمت LEVEL3) منتقل می شود در این قسمت ابتدا عبارت

PLEASE ENTER B
B=?

روی LCD نمایش داده می شود و در قسمت LEVEL4 عدد B دریافت می شود نحوه دریافت عدد B مانند عدد A می باشد سپس عبارت

PLEASE ENTER C
C=?

روی LCD نمایش داده می شود و در قسمت LEVEL6 عدد C دریافت می شود در قسمت LEVEL7 عدد بزرگتر توسط فرمول زیر شناسائی می شود

```
MAX1=A
IF B>MAX1 THEN MAX1=B
IF C>MAX1 THEN MAX1=C
```

ابتدا عدد A در متغیر MAX1 قرار می گیرد. سپس MAX1 با عدد B مقایسه می شود اگر عدد B بزرگتر بود به جای عدد A که قبلا در متغیر MAX1 قرار گرفته بود عدد B در این متغیر قرار می گیرد سپس متغیر MAX1 با عدد C مقایسه می شود و اگر C بزرگتر ازMAX بود عدد C در متغیر MAX1 قرار می گیرد به این ترتیب آخرین عدد قرار گرفته در متغیر MAX1 بزرگترین عدد خواهد بود .

پس از شناسائی بزرگترین عدد آن را توسط دستور ات

```
LCD " MAX VALUE IS"
LCD MAX1
WAIT 5
```

روی LCD نمایش داده و پس از 5 ثانیه تاخیر برنامه RESET شده و اجرای برنامه به لیبل START-PROGRAM منتقل می شود .

پروژه ماشین حساب ساده :

حالا پروژه ای طراحی می کنیم که دو عدد حداکثر 3 رقمی و یک عملگر را از KEYPAD ورودی گرفته ورودی دو عدد اعمال کند .

از یک SPEAKER ، 8 اهمی نیز به عنوان ایجاد کننده صدای سوت کوتاه پس از زدن هر کلید استفاده شده است . صدای ایجاد شده وابسته به فرکانسی است که توسط دستور SOUND در پایه یd.0 میکرو تولید می

شود . همچنین در این پروژه از یک LCD (2*16) به عنوان نمایشگر استفاده شده است . نحوه عملکرد پروژه بدین صورت می باشد که ابتدا عبارت

ENT INPUT DATA

روی LCD نمایش داده می شود سپس کاربر می تواند اولین عدد خود را وارد کند توجه داشته باشید که برنامه به گونه ای نوشته شده است که عدد وارد شده حداکثر می تواند 3 رقمی باشد برای مثال اگر عدد 25 را وارد کنید عبارت نمایش داده شده بر روی LCD به صورت زیر خواهد بود

ENT INPUT DATA
25

حالا باید عملگر را وارد کنید عملگر می تواند یکی از کلید های + ، – ، / ، * باشد به عنوان مثال اگر شما کلید * رافشار دهید عبارت نمایش داده شده بر روی LCD به صورت زیر خواهد بود

ENT INPUT DATA
25 X

حالا شما باید عدد دوم را وارد کنید به عنوان مثال با وارد کردن عدد 10 عبارت نمایش داده شده برروی LCD به صورت زیر خواهد بود

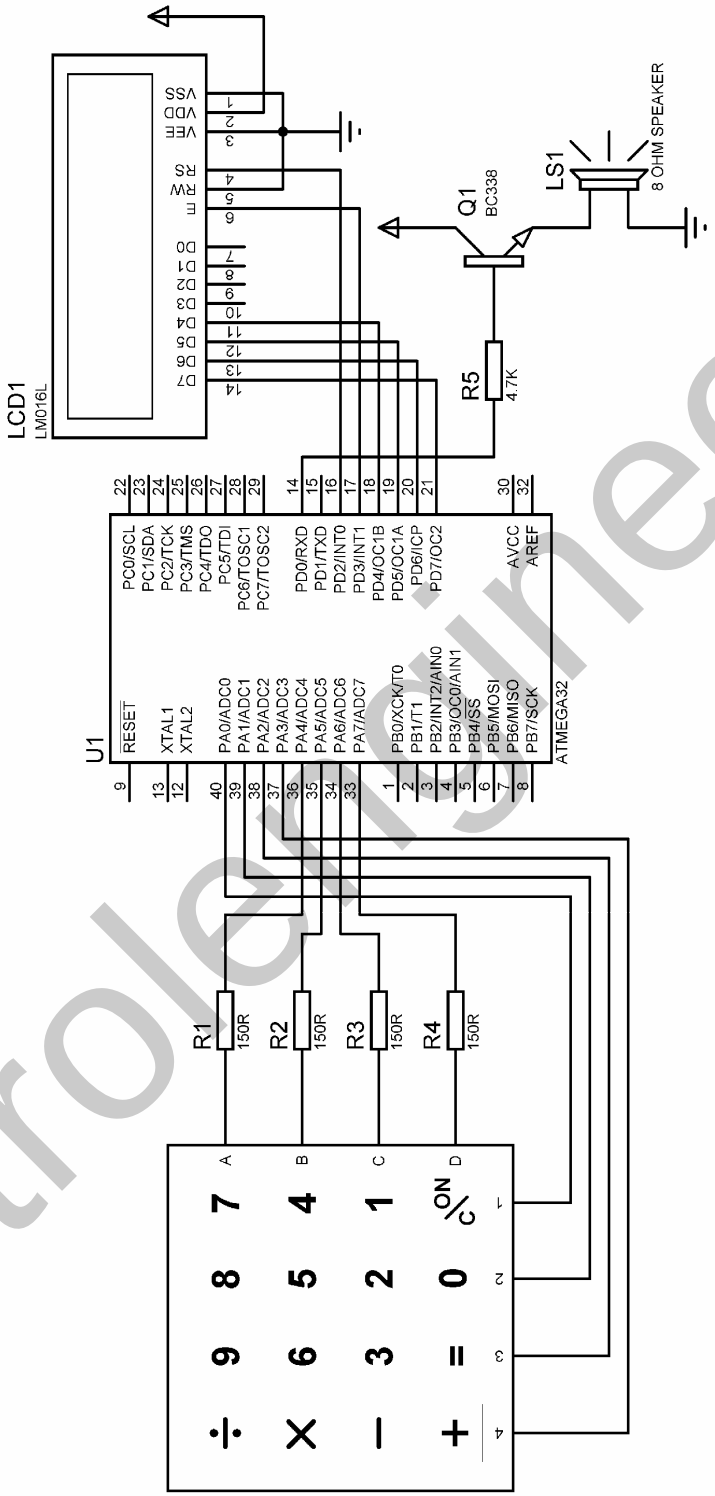
ENT INPUT DATA
25 X 10

حالا با فشار کلید = عبارت نمایش داده شده برروی LCD به صورت زیر خواهد بود

ENT INPUT DATA
25 X 10 = 250.0

و تا وقتی که هیچ کلیدی فشرده نشود عبارت بالا برروی LCD نمایش داده خواهد شد و با فشار هر کدام از کلید ها برنامه RESET شده دوباره شما باید دو عدد و یک عملگر را به ترتیبی که در بالا گفته شد وارد کنید .

سخت افزار پروژه در شکل 2-34 نشان داده شده است .



شکل 2-34 شماتیک طراحی شده برای ماشین حساب ساده

برنامه نوشته شده برای پروژه به صورت زیر است

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Recive_data As Byte , A1 As Word , A2 As Word , A3 As Byte
Dim Star As Byte , Count As Byte , H As Single , A As Word , C As Word
Dim C1 As Word , C2 As Word , C3 As Byte , B As String * 2
Declare Sub Main
Config Kbd = Porta , Debounce = 50 , Delay = 100
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
 , E = Pind.3 , Rs = Pind.2
Config Pind.0 = Output
'START OF LEVEL1-----------------------------------------------------------------------------------------------------
Start_program:
Count = 0
Cursor Off
Cls : Home
Lcd "ENT INPUT DATA"
'END OF LEVEL1-------------------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Star = Lookup(recive_data , Data_code)
If Star >= 10 Then
If Star = 40 Then Goto H1
If Star = 50 Then Goto H1
Count = 0
Goto H2
End If
Sound Portd.0 , 100 , 80
Incr Count
If Count = 1 Then
A1 = Star
A = A1
End If
If Count = 2 Then
A2 = Star
'A=(A1*10)+A2
A = A1 * 10
A = A + A2
End If
If Count = 3 Then
A3 = Star
A1 = A1 * 100
A2 = A2 * 10
A = A1 + A2
A = A + A3
End If
Cls : Home
Lcd "ENT INPUT DATA"
Locate 2 , 1
Lcd A
If Count = 3 Then
Count = 0
H5:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H5
```

```
Star = Lookup(recive_data , Data_code)
If Star < 10 Then Goto H5
If Star = 40 Then Goto H5
If Star = 50 Then Goto H5
Goto H2
End If
Waitms 100
Goto H1
'END OF LEVEL2-----------------------------------------------------------------------------------------------------
'START OF LEVEL3-----------------------------------------------------------------------------------------------------
H2:
Star = Star / 10
Decr Star
B = Lookupstr(star , Sdata)
Sound Portd.0 , 100 , 80
Cls : Home
Lcd "ENT INPUT DATA"
Locate 2 , 1
Lcd A ; B
Wait 1
'END OF LEVEL3-----------------------------------------------------------------------------------------------------
'START OF LEVEL4-----------------------------------------------------------------------------------------------------
H3:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H3
Star = Lookup(recive_data , Data_code)
If Star >= 10 Then
If Star = 40 Then Goto H3
Count = 0
Goto H4
End If
Sound Portd.0 , 100 , 80
Incr Count
If Count = 1 Then
C1 = Star
C = C1
End If
If Count = 2 Then
C2 = Star
'C=(C1*10)+C2
C = C1 * 10
C = C + C2
End If
If Count = 3 Then
C3 = Star
C1 = C1 * 100
C2 = C2 * 10
C = C1 + C2
C = C + C3
End If
Cls : Home
Lcd "ENT INPUT DATA"
Locate 2 , 1
Lcd A ; B ; C
If Count = 3 Then
H7:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H7
Star = Lookup(recive_data , Data_code)
If Star <> 50 Then Goto H7
Count = 0
```

```
Goto H4
End If
Waitms 100
Goto H3
'END OF LEVEL4-----------------------------------------------------------------------------------------------
'START OF LEVEL5-----------------------------------------------------------------------------------------------
H4:
Sound Portd.0 , 100 , 80
Select Case B
Case Is = "/"
H = A / C
Case Is = "X"
H = A * C
Case Is = "-"
H = A - C
Case Is = "+"
H = A + C
End Select
Cls : Home
Lcd "ENT INPUT DATA"
Locate 2 , 1
Lcd A ; B ; C ; "=" ; H
Wait 1
H6:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H6
Goto Start_program
'END OF LEVEL5-----------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'-----------------------------------------------------------------------------------------------------------
Sdata:
Data "/" , "X" , "-" , "ON/C" , "=" , "+"
'-----------------------------------------------------------------------------------------------------------
```

تشریح نحوه عملکرد برنامه :

در قسمت LEVEL1 مکان نما خاموش شده و عبارت ENT INPUT DATA بر روی LCD نمایش داده می شود. در قسمت LEVEL2 اولین عدد گرفته می شود نحوه گرفتن اولین عدد به صورت زیر است .

ابتدا تا وقتی که کلیدی فشرده نشود ورودی خوانده می شود . اگر کلیدی فشار داده شد عدد معادل آن توسط جدول LOOKUP در متغیر STAR برگردانده می شود توجه داشته باشید با فشار کلید ON/C و = به ترتیب اعداد 40 و 50 در STAR قرار می گیرد که در این صورت اجرای برنامه به لیبل H1 منتقل می شود یعنی باز هم ورودی چک می شود . اگر مقدار متغیر STAR بزرگتر از 10 باشد البته به غیر از اعداد (50,40) اجرای برنامه به لیبل H2 منتقل می شود .

در لیبل H2 که در قسمت LEVEL3 قرار دارد با استفاده از جدول LOOKUPSTR با نام SDATA نام عملگر وارد شده در متغیر رشته ای B قرار می گیرد و ورودی LCD به دنبال عدد وارد شده نمایش داده می شود

برای مثال اگر کلید (+) زده شود محتوای متغیر STAR برابر با 60 خواهد بود سپس عملیات زیر در قسمت LEVEL3 انجام می گیرد

```
STAR=STAR/10
DECR STAR
B=LOOKUP STR(STAR,SDATA)
```

یعنی رشته شماره 5 از جدول SDATA در متغیر رشته ای B قرار می گیرد .

در قسمت LEVEL3 عدد دوم دریافت شده ودر متغیر C قرار می گیرد.

پس از دریافت عدد C با زدن کلید (=) اجرای برنامه به لیبل H4 که در قسمت LEVEL5 عملگر وارد شده توسط دستور CASE شناسائی شده و روی دو عدد وارد شده اعمال می شود سپس نتیجه توسط دستورات زیر روی LCD نمایش داده می شود .

```
LCD " ENT INPUT DATA"
LOCATE 2,1
LCD A;B;C;"=";H
```

H متغیری از نوع SINGLE بوده که می تواند عددی ممیزدار ما بین 3.4*10^38 تا 1.5*10^- 45 باشد .

هنگام نمایش نتیجه بر روی LCD ورودی چک می شود و تا وقتی که هیچ کلیدی فشار داده نشود نتیجه نمایش داده خواهد شد . و زمانی که کلیدی فشار داده شود اجرای برنامه به لیبل START_PROGRAM منتقل خواهد شد .

پروژه ای که دو عدد A ، B را گرفته و بزرگترین مقسوم علیه مشترک آن ها را چاپ کند

حالا پروره ای طراحی می کنیم که دو عدد A,B را توسط KEYPAD گرفته و بزرگترین مقسوم علیه مشترک آن ها را روی LCD نمایش دهد ، A,B حداکثر می تواند 3 رقمی باشد ورودی حساس به لبه بالا رونده می باشد از یک SPEAKER ، 8 اهمی نیز به عنوان ایجاد کننده صدای سوت کوتاه پس از فشار هر کدام از کلید ها استفاده شده است . همچنین از یک LCD ، (16*2) به عنوان نمایشگر استفاده شده است . نحوه عملکرد پروژه بدین صورت می باشد که ابتدا عبارت

PLEASE ENTER A
A IS

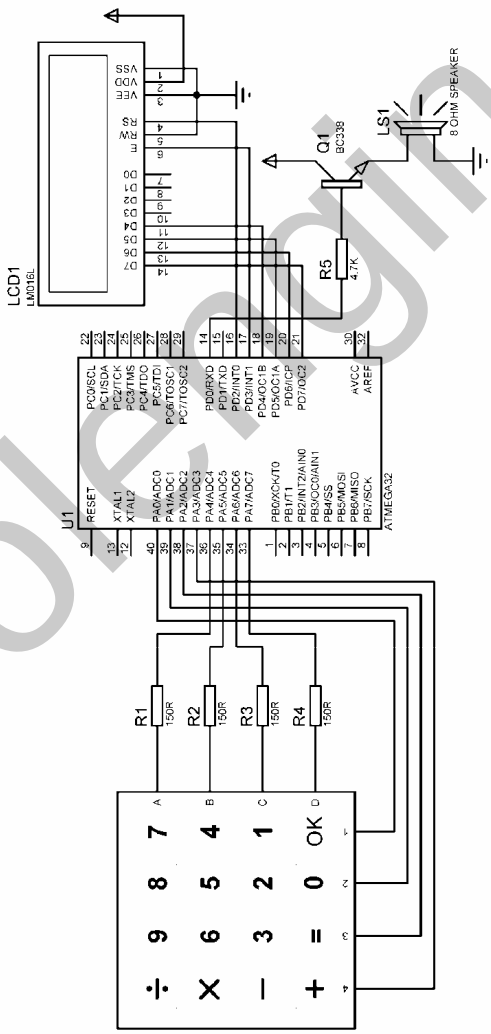
روی LCD نمایش داده می شود سپس باید شما توسط KEYPAD عدد A را وارد کنید پس از وارد کردن عدد A بایستی کلید OK را فشار دهید پس از زدن کلید OK عبارت زیر روی LCD نمایش داده می شود.

PLEASE ENTER B
B IS

حالا شما عدد B را وارد کرده و کلید OK را فشار دهید . پس از فشار کلید OK بزرگترین مقسوم علیه مشترک A,B روی LCD نمایش داده می شود برای مثال اگر عدد A را 12 وارد کرده و عدد B را 6 وارد کنید عبارت زیر روی LCD نمایش داده می شود

IS = 6 ب.م.م

همچنین پس از نمایش عبارت بالا بر روی LCD تا وقتی که هیچ کلیدی فشرده نشده برنامه RESET نمی شود و عبارت نمایش داده شده تغییری نخواهد کرد . در این حالت با فشار هر کدام از کلید ها برنامه RESET شده و دوباره عدد A توسط میکروکنترلر درخواست می شود . سخت افزار پروژه در شکل 35-2 نشان داده شده است .



شکل 35-2 شماتیک طراحی شده برای پروژه نمایش بزرگترین مقسوم علیه مشترک دو عدد وارد شده

برنامه نوشته شده برای پروژه به صورت زیر است

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Recive_data As Byte , A1 As Word , A2 As Word , A3 As Byte , A As Word
Dim Star As Byte , Count As Byte , Min1 As Word , S1 As Word , S2 As Word
Dim B1 As Word , B2 As Word , B3 As Byte , B As Word , Max1 As Word
Dim T As Word
Config Kbd = Porta , Debounce = 50 , Delay = 100
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
            , E = Pind.3 , Rs = Pind.2
Config Pind.0 = Output
Deflcdchar 0 , 32 , 32 , 32 , 17 , 31 , 32 , 4 , 32
Deflcdchar 1 , 32 , 32 , 3 , 3 , 31 , 16 , 16 , 16
'START OF LEVEL1-----------------------------------------------------------------------------------------------------
Start_program:
S1 = 0 : S2 = 0 : Max1 = 0 : Min1 = 0
Cursor Off
Cls : Home
Lcd "PLEASE ENTER A"
Lowerline
Lcd "A IS "
Wait 1
'END OF LEVEL1-------------------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Star = Lookup(recive_data , Data_code)
If Star >= 10 Then
If Star = 40 Then
Count = 0
Recive_data = 16
Goto H2
End If
Goto H1
End If
Sound Portd.0 , 100 , 80
Incr Count
If Count = 1 Then
A1 = Star
A = A1
End If
If Count = 2 Then
A2 = Star
'A=(A2*10)+A1
A = A1 * 10
A = A + A2
End If
If Count = 3 Then
A3 = Star
'A=(A2*10)+A1
A1 = A1 * 100
A2 = A2 * 10
A = A1 + A2
A = A + A3
End If
Cls : Home
Lcd "PLEASE ENTER A"
Lowerline
```

```
Lcd "A IS " ; A
If Count = 3 Then
Count = 0
Goto T2
End If
R1:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto R1
Goto H1
'END OF LEVEL2-----------------------------------------------------------------------------------------------------
'START OF LEVEL3-----------------------------------------------------------------------------------------------------
H2:
Cls : Home
Lcd "PLEASE ENTER B"
Lowerline
Lcd "B IS "
Sound Portd.0 , 100 , 80
T1:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T1
Goto H3
'END OF LEVEL3-----------------------------------------------------------------------------------------------------
'START OF LEVEL4-----------------------------------------------------------------------------------------------------
T2:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T2
T3:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto T3
Star = Lookup(recive_data , Data_code)
If Star <> 40 Then Goto T3
Cls : Home
Lcd "PLEASE ENTER B"
Lowerline
Lcd "B IS "
Sound Portd.0 , 100 , 80
R2:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto R2
'END OF LEVEL4-----------------------------------------------------------------------------------------------------
'START OF LEVEL5-----------------------------------------------------------------------------------------------------
H3:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H3
Star = Lookup(recive_data , Data_code)
If Star >= 10 Then
If Star = 40 Then
Count = 0
Goto H4
End If
Goto H3
End If
Sound Portd.0 , 100 , 80
Incr Count
If Count = 1 Then
B1 = Star
B = B1
End If
If Count = 2 Then
B2 = Star
'B=(B2*10)+B1
```

```
B = B1 * 10
B = B + B2
End If
If Count = 3 Then
B3 = Star
'B=(B2*10)+B1
B1 = B1 * 100
B2 = B2 * 10
B = B1 + B2
B = B + B3
End If
Cls : Home
Lcd "PLEASE ENTER B"
Lowerline
Lcd "B IS " ; B
If Count = 3 Then
Count = 0
Recive_data = 16
Goto T5
End If
R3:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto R3
Goto H3
END OF LEVEL5-----------------------------------------------------------------------------------------
'START OF LEVEL6-----------------------------------------------------------------------------------------
H4:
Sound Portd.0 , 100 , 80
Goto H5
'END OF LEVEL6-----------------------------------------------------------------------------------------
'START OF LEVEL7-----------------------------------------------------------------------------------------
T5:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T5
T6:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto T6
Star = Lookup(recive_data , Data_code)
If Star <> 40 Then Goto T6
Sound Portd.0 , 100 , 80
'END OF LEVEL7-----------------------------------------------------------------------------------------
'START OF LEVEL8-----------------------------------------------------------------------------------------
H5:
If A > B Then
Min1 = B
Else
Min1 = A
End If
For T = 1 To Min1 Step 1
S1 = A Mod T
S2 = B Mod T
If S1 = 0 Then
If S2 = 0 Then Max1 = T
End If
Next T
Cls : Home
Lcd Chr(1) ; Chr(1) ; Chr(0) ; " =" ; Max1
T4:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T4
'END OF LEVEL8-----------------------------------------------------------------------------------------
```

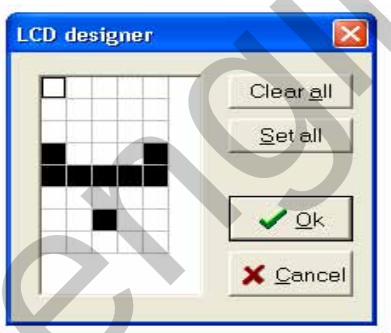
```
'START OF LEVEL9----------------------------------------------------------------------------------------
T8:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto T8
Goto Start_program
'END OF LEVEL9------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'-------------------------------------------------------------------------------------------------------
```

دستورات جدید :

در برنامه بالا از دستور MOD استفاده شده است که فرم کلی آن به صورت زیر است

S=A MOD B

با اجرای این دستور باقیمانده تقسیم A/B در متغیر S قرار می گیرد. همچنین در برنامه بالا از دستور DEFLCDCHR استفاده شده است . با این دستور می توانید حرف یا علامتی را که خودتان در منوی TOOLS در قسمت LCD DESIGNER محیط BASCOM طراحی کرده اید بر روی صفحه LCD نمایش دهید ، پنجره LCD DESIGNER در شکل 2-36 نشان داده شده است .



شکل 2-36 نمائی از پنجره LCD DESIGNER

هر کاراکتر موجود برروی LCD دارای 5*7 پیکسل می باشد ، هنگام طراحی برای انتخاب هر کدام از پیکسل ها روی آن کلیک کنید همچنین می توانید با کلیک دوباره آن را از حالت انتخاب خارج کنید.

در پنجره LCD DESIGNER چهار کلید وجود دارد که نحوه عملکرد آن ها به صورت زیر است

CLEAR ALL : همه پیکسل ها را از حالت انتخاب خارج می کند.

SET ALL : همه پیکسل ها را در حالت انتخاب قرار می دهد .

CANCEL : به منظور انصراف از طراحی از این گزینه استفاده می شود.

OK : پس از طراحی کاراکتر مورد نظر بایستی روی کلید OK کلیک کنید پس از کلیک بر روی این دکمه عبارت زیر به برنامه شما اضافه می شود

DEFLCD CHR?,R1,R2,R3,R4,R5,R6,R7,R8

عبارت های R1 تا R8 توسط نرم افزار و با توجه به کاراکتر طراحی شده تعیین می شوند شما می توانید هشت کاراکتر جدید را برای LCD تعریف کنید پس به جای علامت سئوال می توانید یک عدد مابین 0 تا 7 جایگزین کنید کاراکتر طراحی شده را می توان توسط دستور LCD CHR(?) بر روی LCD نمایش داد .

تشریح نحوه عملکرد برنامه :

در قسمت LEVEL1 ابتدا مکان نما خاموش شده و عبارت

PLEASE ENTER A
A IS

روی LCD نوشته می شود در قسمت LEVEL2 عدد A گرفته می شود عدد A حداکثر می تواند 3 رقمی باشد هنگام وارد کردن عدد پس از زدن هر کدام از کلید های 0 تا 9 برای این که ورودی حساس به لبه بالا رونده باشد بلافاصله دستورات مربوطه انجام شده سپس اجرای برنامه تا غیر فعال شدن کلید توسط دستورات زیر متوقف می شود .

```
R1:
RECIVE_DATA=GETKBD()
IF RECIVE_DATA<> 16 THEN GO TO R1
```

اگر عددی کمتر از 3 رقم را وارد کرده و کلید OK را فشار دهیم اجرای برنامه به لیبل H2 که در قسمت LEVEL 3 قرار دارد منتقل می شود که در این قسمت ابتدا عبارت

PLEASE ENTER B
B IS

روی LCD نوشته می شود سپس ورودی چک می شود و تا وقتی که RECIVE_DATA مخالف 16 باشد عمل چک کردن ورودی تکرار خواهد شد . به محض این که دستمان را از روی کلید OK برداریم اجرای برنامه از حلقه T1 خارج شده و به لیبل H3 (گرفتن عدد B) منتقل خواهد شد این روند به منظور حساس کردن ورودی به لبه بالا رونده انجام می شود .

بدین صورت که پس از فشار کلید بلافاصله دستورات مربوطه انجام می شود سپس تا وقتی که کلید غیر فعال نشده اجرای برنامه متوقف می شود .

اگر عدد وارد شده 3 رقمی باشد پس از وارد کردن رقم سوم بلافاصله اجرای برنامه به لیبل T2 که در قسمت LEVEL4 قرار دارد منتقل می شود در این قسمت تا دستمان را از روی کلید بر نداشتیم اجرای برنامه در حلقه T2 متوقف می شود پس از غیر فعال شدن کلید ، ورودی دوباره چک می شود و تا زمانی که کلید OK را فشار نداده ایم اجرای برنامه در حلقه T3 متوقف خواهد شد . پس از زدن کلید OK عبارت

PLEASE ENTER B
B IS

روی LCD نمایش داده می شود . تا وقتی که دستمان را از روی کلید OK برنداشتیم اجرای برنامه در حلقه R3 متوقف خواهد شد . پس از غیر فعال شدن کلید OK اجرای برنامه دارد LEVEL5 می شود در قسمت LEVEL5 عدد B گرفته می شود نحوه دریافت عدد B نیز شبیه عدد A می باشد . پس از وارد کردن عدد B و زدن کلید OK اجرای برنامه وارد LEVEL8 می شود در این قسمت بزرگترین مقسوم علیه مشترک دو عدد وارد شده تعیین خواهد شد .

بطور کلی برای تشخیص بزرگترین مقسوم علیه مشترک بایستی بصورت زیر عمل کنیم فرض کنید دو عدد 12 و 6 را داریم ، ابتدا مقسوم علیه های هر یک از اعداد حساب می کنیم .

برای این که بدانیم عدد 6 بر کدام یک از اعداد 1 تا 6 بخش پذیر است عدد 6 را یک به یک به هر کدام از این اعداد تقسیم می کنیم اگر باقیمانده صفر آمده 6 بر آن عدد بخش پذیر بوده و آن عدد جز مقسوم علیه های 6 به حساب می آید . که در این حالت مقسوم علیه های 6 عبارتند از

{1,2,3,6}

مقسوم علیه های 12 عبارتند از

{1,2,3,4,6,12}

مقسوم علیه های مشترک بین 6 و 2 عبارتند از :

{1,2,3,6}

سپس بزرگترین مقسوم علیه مشترک برابر با 6 خواهد بود . ملاحظه می شود که بزرگترین مقسوم علیه مشترک نمی تواند از عدد کوچکتر وارد شده یعنی 6 بیشتر باشد . درنتیجه در این برنامه حلقه ، FOR را به اندازه عدد کوچکتر تشکیل خواهیم داد.

از دستور شرطی زیر برای تشخیص عدد کوچکتر استفاده شده است.

```
IF A >B THEN
MIN1=B
ELSE
MIN1=A
END IF
```

سپس طبق دستور زیر اعداد 1 تا MIN1 به ترتیب و به صورت صعودی به هر دو عدد B,A تقسیم می شود اگر باقیمانده هر دو تقسیم صفر بود آن عدد در متغیر MAX1 قرار می گیرد چون ترتیب تقسیم شدن B,A به اعداد 1 تا MIN1 به صورت صعودی است پس آخرین عدد قرار گرفته در MAX1 بزرگترین مقسوم علیه مشترک خواهد بود

```
FOR T=1 TO MIN1 STEP1
ST=A MOD T
S2= B MOD T
IF S1=0 THEN
IF S2=0 THEN MAX1=T
END IF
```

```
NEXT T
```

پس از این که بزرگترین مقسوم علیه مشترک شناسائی شد توسط دستورات زیر روی LCD نمایش داده می شود

```
LCD CHR(1);CHR(1);CHR(0);"IS"
LOWERLINE
LCD MAX1
```

توجه داشته باشید که عمل تشخیص بزرگترین مقسوم علیه مشترک و نوشتن آن بر روی LCD با سرعت بسیار زیاد انجام می شود تا حدی که ممکن است شما هنوز دستتان را از روی کلید OK بر نداشته باشید در این قسمت برای حساس کردن ورودی به لبه بالا رونده ، ورودی چک می شود و تا وقتی که کلید OK غیر فعال نشده عمل چک کردن ورودی توسط حلقه T4 تکرار خواهد شد . سپس اجرای برنامه وارد قسمت LEVEL9 می شود که در این قسمت هم ورودی چک می شود و تا زمانی که هیچ کلیدی فشار داده نشده برنامه RESET نمی شود و عبارت موجود بر روی LCD تغییری نخواهد کرد به محض این که کلیدی فشرده شود اجرای برنامه به لیبل START_PROGRAM منتقل خواهد شد .

پروژه ای که یک معادله درجه 2 را گرفته و ریشه های آن را بر روی LCD نمایش دهد

حالا پروژه ای طراحی می کنیم که یک معادله درجه دو را گرفته و ریشه های آن را بر روی صفحه LCD نمایش دهد . در این پروژه از یک LCD ، 16*4 به عنوان نمایشگر و از یک KEYPAD 4*4 برای واردکردن اطلاعات استفاده شده است ورودی حساس به لبه بالا رونده بوده و عدد وارد شده حداکثر می تواند 3 رقمی باشد برای وارد کردن اعداد منفی ابتدا کلید (-) را زده سپس عدد مورد نظر را وارد کنید از یکSPEAKER ، 8 اهمی به عنوان تولید کننده صدای سوت کوتاه پس از فشار هر کدام از کلید ها استفاده شده است . نحوه عملکرد پروژه بدین صورت می باشد که ابتدا عبارت زیر بر روی LCD نوشته می شود .

ابتدا عبارت زیر روی LCD نوشته می شود .

```
AX^2+BX+C
YOU MUST ENTER
A,B,C
```

پس از 5 ثانیه تاخیر عدد A یعنی ضریب X^2 به صورت زیر از کاربر خواسته می شود.

```
PLEASE ENTER A
A IS
```

برای وارد کردن اعداد منفی قبل از وارد کردن عدد، کلید (-) را فشار دهید تا علامت منفی روی صفحه LCD نمایش داده شود سپس عدد مورد نظر را وارد کنید پس از وارد کردن عدد A و زدن دکمه OK عدد B به صورت زیر از کاربر خواسته می شود.

```
PLEASE ENTER B
```

B IS

پس از وارد کردن عدد B و زدن دکمه OK عدد C به صورت زیر از کاربر خواسته می شود

PLEASE ENTER B
B IS

پس ازوارد کردن عدد C و زدن کلید OKمیکروکنترلر دلتا (b^2- 4 ac) را حساب می کند اگر Δ>0 باشد X2,X1 روی LCD نمایش داده می شوندبرای مثال اگر عدد A را 21 و عدد B را 27- و عدد C را 25 وارد کرده باشید عبارت زیر روی LCD نمایش داده خواهد شد

Δ IS 529.0
DELTA IS > 0
X1= 1.038171885
X2=12.461828229

اگر Δ = 0 باشد معادله یک ریشه مضاعف خواهد داشت یعنی X^2=X1 خواهد بود .

برای مثال اگر C,B,A را صفر وارد کنیم عبارت نمایش داده بر روی LCD به صورت زیر خواهد بود

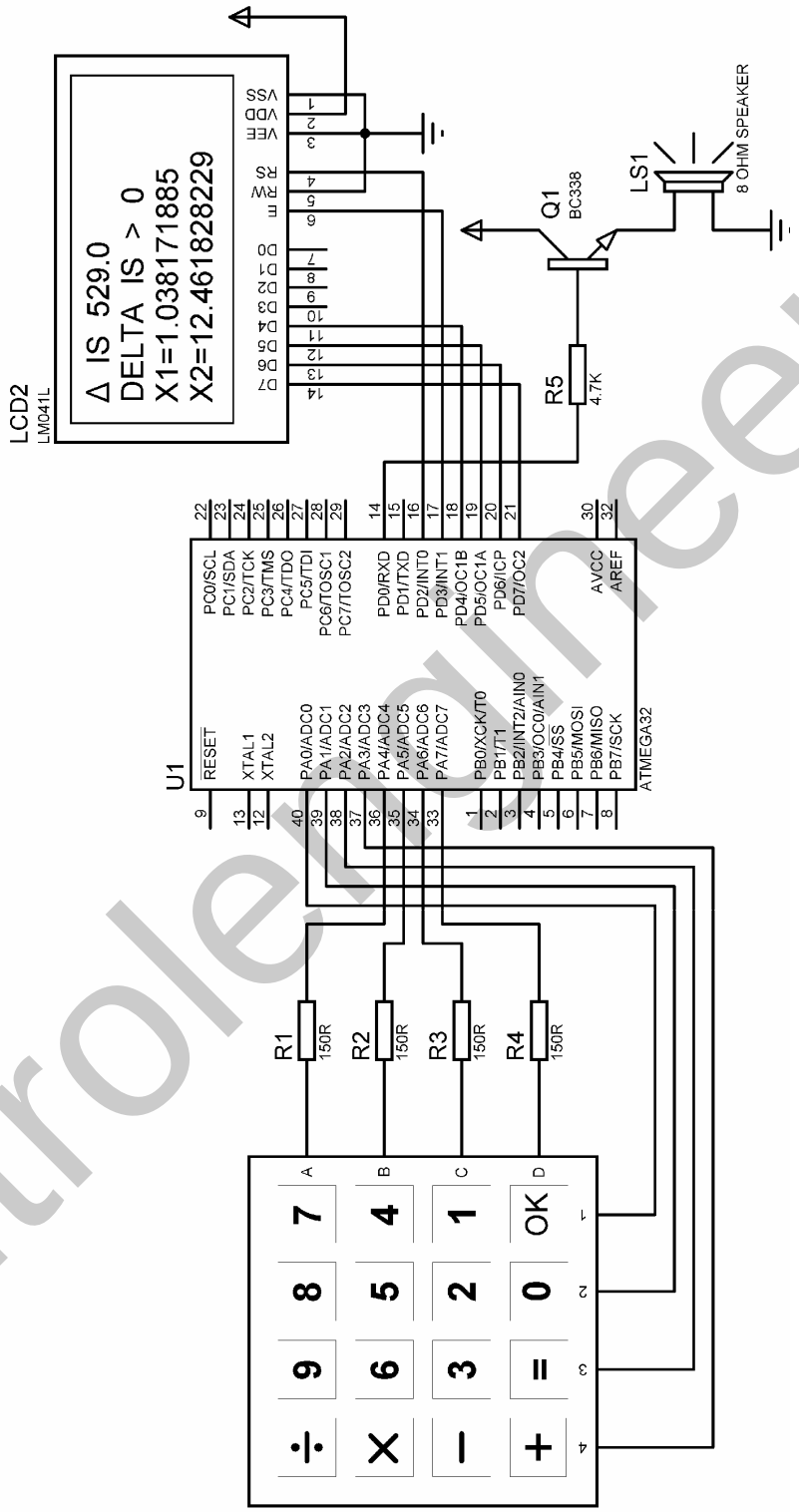
Δ IS 0.0
X1 IS = X2
X1=0.0
X2=0.0

اگر Δ<0 باشد معادله ریشه حقیقی ندارد برای مثل اگر C,B,A را به ترتیب 3-,2-,1- وارد کنیم عبارت نمایش داده شده بر روی LCD به صورت زیر خواهد بود

Δ IS -8.0
DELTA IS < 0
ERROR

پس از نمایش ریشه های معادله بر روی صفحه نمایش LCD ، عبارت نمایش داده شده تا زدن یکی از کلید های ورودی بدون تغییر خواهد بود و با زدن هر کدام از کلید ها برنامه RESET شده و از اول اجرا خواهد شد .

سخت افزار پروژه در شکل 37-2 نشان داده شده است .



شکل 2-37 شماتیک طراحی شده برای حل معادله درجه 2

برنامه نوشته شده برای پروژه به صورت زیر می باشد.

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Recive_data As Byte , A1 As Long , A2 As Long , A3 As Long , A As Long
Dim Star As Byte , Count As Byte , Delta As Single , S1 As Single , S2 As Single
Dim B1 As Long , B2 As Long , B3 As Long , B As Long , X1 As Single
Dim C1 As Long , C2 As Long , C3 As Long , C As Long , X2 As Single
Dim T As Word , Jamp As Bit , O1 As Bit , O2 As Bit , O3 As Bit
Config Kbd = Porta , Debounce = 50 , Delay = 100
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
          , E = Pind.3 , Rs = Pind.2
Config Pind.0 = Output
Deflcdchar 0 , 4 , 4 , 10 , 10 , 17 , 17 , 31 , 32
'START OF LEVEL1-----------------------------------------------------------------------------------------------------------
Start_program:
A1 = 0 : A2 = 0 : A3 = 0 : A = 0
B1 = 0 : B2 = 0 : B3 = 0 : B = 0
C1 = 0 : C2 = 0 : C3 = 0 : C = 0
S1 = 0 : S2 = 0 : X1 = 0 : X2 = 0 : Delta = 0
O1 = 0 : O2 = 0 : O3 = 0
Cursor Off
Cls : Home
Lcd "Ax^2 + Bx + C"
Locate 2 , 1
Lcd "YOU MUST ENTER "
Locate 3 , 1
Lcd "A , B , C "
Wait 5
Cursor Off
Cls : Home
Lcd "PLEASE ENTER A"
Lowerline
Lcd "A IS "
'END OF LEVEL1-------------------------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Star = Lookup(recive_data , Data_code)
'-----------------------------------------------------------------
If Count = 0 Then
If Star = 30 Then
O1 = 1
Cls : Home
Lcd "PLEASE ENTER A"
Lowerline
Lcd "A IS -"
Sound Portd.0 , 100 , 80
K1:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto K1
Goto H1
End If
End If
'-----------------------------------------------------------------
If Star >= 10 Then
If Star = 40 Then
```

```
Count = 0
Recive_data = 16
Goto H2
End If
Goto H1
End If
'-----------------------------------------------------------------
Sound Portd.0 , 100 , 80
Incr Count
If Count = 1 Then
A1 = Star
A = A1
If O1 = 1 Then A = A * -1
End If
If Count = 2 Then
A2 = Star
'A=(A2*10)+A1
A = A1 * 10
A = A + A2
If O1 = 1 Then A = A * -1
End If
If Count = 3 Then
A3 = Star
A1 = A1 * 100
A2 = A2 * 10
A = A1 + A2
A = A + A3
If O1 = 1 Then A = A * -1
End If
'-----------------------------------------------------------------
Cls : Home
Lcd "PLEASE ENTER A"
Lowerline
Lcd "A IS " ; A
'-----------------------------------------------------------------
If Count = 3 Then
Count = 0
Recive_data = 16
Goto T2
End If
'-----------------------------------------------------------------
R1:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto R1
Goto H1
'END OF LEVEL2-----------------------------------------------------------------------------------------------------------
'START OF LEVEL3---------------------------------------------------------------------------------------------------------
H2:
Jamp = 0
Cls : Home
Lcd "PLEASE ENTER B"
Lowerline
Lcd "B IS "
Sound Portd.0 , 100 , 80
T1:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T1
Goto H3
'END OF LEVEL3-----------------------------------------------------------------------------------------------------------
'START OF LEVEL4---------------------------------------------------------------------------------------------------------
T2:
```

```
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T2
'-----------------------------------------------------------------
T3:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto T3
Star = Lookup(recive_data , Data_code)
If Star <> 40 Then Goto T3
'-----------------------------------------------------------------
Cls : Home
Lcd "PLEASE ENTER B"
Lowerline
Lcd "B IS "
Sound Portd.0 , 100 , 80
'-----------------------------------------------------------------
R2:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto R2
'END OF LEVEL4-----------------------------------------------------------------------------------------------------------
'START OF LEVEL5---------------------------------------------------------------------------------------------------------
H3:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H3
Star = Lookup(recive_data , Data_code)
'-----------------------------------------------------------------
If Count = 0 Then
If Star = 30 Then
O2 = 1
Cls : Home
Lcd "PLEASE ENTER B"
Lowerline
Lcd "B IS -"
Sound Portd.0 , 100 , 80
K2:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto K2
Goto H3
End If
End If
'-----------------------------------------------------------------
If Star >= 10 Then
If Star = 40 Then
Count = 0
Goto H4
End If
Goto H3
End If
'-----------------------------------------------------------------
Sound Portd.0 , 100 , 80
Incr Count
If Count = 1 Then
B1 = Star
B = B1
If O2 = 1 Then B = B * -1
End If
If Count = 2 Then
B2 = Star
'B=(B2*10)+B1
B = B1 * 10
B = B + B2
If O2 = 1 Then B = B * -1
```

```
End If
If Count = 3 Then
B3 = Star
B1 = B1 * 100
B2 = B2 * 10
B = B1 + B2
B = B + B3
If O2 = 1 Then B = B * -1
End If
'-----------------------------------------------------------------
Cls : Home
Lcd "PLEASE ENTER B"
Lowerline
Lcd "B IS " ; B
'-----------------------------------------------------------------
If Count = 3 Then
Count = 0
Recive_data = 16
Goto T6
End If
'-----------------------------------------------------------------
R3:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto R3
Goto H3
'END OF LEVEL5-----------------------------------------------------------------------------------------------------
'START OF LEVEL6---------------------------------------------------------------------------------------------------
H4:
Cls : Home
Lcd "PLEASE ENTER C"
Lowerline
Lcd "C IS "
Sound Portd.0 , 100 , 80
T5:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T5
Goto H5
'END OF LEVEL6-----------------------------------------------------------------------------------------------------
'START OF LEVEL7---------------------------------------------------------------------------------------------------
T6:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T6
'-----------------------------------------------------------------
T7:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto T7
Star = Lookup(recive_data , Data_code)
If Star <> 40 Then Goto T7
'-----------------------------------------------------------------
Cls : Home
Lcd "PLEASE ENTER C"
Lowerline
Lcd "C IS "
Sound Portd.0 , 100 , 80
'END OF LEVEL7-----------------------------------------------------------------------------------------------------
'START OF LEVEL8---------------------------------------------------------------------------------------------------
H5:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H5
Star = Lookup(recive_data , Data_code)
'-----------------------------------------------------------------
```

```
If Count = 0 Then
If Star = 30 Then
O3 = 1
Cls : Home
Lcd "PLEASE ENTER C"
Lowerline
Lcd "C IS -"
Sound Portd.0 , 100 , 80
K3:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto K3
Goto H5
End If
End If
'-----------------------------------------------------------------
If Star >= 10 Then
If Star = 40 Then
Count = 0
Goto H6
End If
Goto H5
End If
'-----------------------------------------------------------------
Sound Portd.0 , 100 , 80
Incr Count
If Count = 1 Then
C1 = Star
C = C1
If O3 = 1 Then C = C * -1
End If
If Count = 2 Then
C2 = Star
'C=(C2*10)+C1
C = C1 * 10
C = C + C2
If O3 = 1 Then C = C * -1
End If
If Count = 3 Then
C3 = Star
C1 = C1 * 100
C2 = C2 * 10
C = C1 + C2
C = C + C3
If O3 = 1 Then C = C * -1
End If
'-----------------------------------------------------------------
Cls : Home
Lcd "PLEASE ENTER C"
Lowerline
Lcd "C IS " ; C
'-----------------------------------------------------------------
If Count = 3 Then
Count = 0
Recive_data = 16
Goto T9
End If
'-----------------------------------------------------------------
R4:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto R4
Goto H5
```

```
'END OF LEVEL8----------------------------------------------------------------------------------------------------
'START OF LEVEL9----------------------------------------------------------------------------------------------------
H6:
Sound Portd.0 , 100 , 80
Goto H7
'END OF LEVEL9----------------------------------------------------------------------------------------------------
'START OF LEVEL10----------------------------------------------------------------------------------------------------
T9:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T9
'-----------------------------------------------------------------
T10:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto T10
Star = Lookup(recive_data , Data_code)
If Star <> 40 Then Goto T10
Sound Portd.0 , 100 , 80
'END OF LEVEL10----------------------------------------------------------------------------------------------------
'START OF LEVEL11----------------------------------------------------------------------------------------------------
H7:
S1 = B * B
S2 = A * C
S2 = S2 * 4
Delta = S1 - S2
'END OF LEVEL11----------------------------------------------------------------------------------------------------
'START OF LEVEL12----------------------------------------------------------------------------------------------------
Select Case Delta
Case Is > 0
S1 = B * -1
S2 = Delta ^ 0.5
S1 = S1 + S2
S2 = 2 * A
X1 = S1 / S2
'-----------------------------------------------------------------
S1 = B * -1
S2 = Delta ^ 0.5
S1 = S1 - S2
S2 = 2 * A
X2 = S1 / S2
Cls : Home
Lcd Chr(0) ; " IS " ; Delta
Lowerline
Lcd "DELTA IS > 0"
'-----------------------------------------------------------------
Case Is = 0
S1 = B * -1
S2 = A * 2
X1 = S1 / S2
X2 = X1
Cls : Home
Lcd Chr(0) ; " IS " ; Delta
Lowerline
Lcd "X1 IS = X2"
'-----------------------------------------------------------------
Case Is < 0
Cls : Home
Lcd Chr(0) ; " IS " ; Delta
Locate 2 , 1
Lcd "DELTA IS < 0"
Locate 3 , 1
Lcd "ERROR"
```

```
Goto T11
End Select
'END OF LEVEL12-----------------------------------------------------------------------------------------------------
'START OF LEVEL13---------------------------------------------------------------------------------------------------
Locate 3 , 1
Lcd "X1=" ; X1
Locate 4 , 1
Lcd "X2=" ; X2
'-----------------------------------------------------------------------
T11:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T11
'END OF LEVEL13-----------------------------------------------------------------------------------------------------
'START OF LEVEL14---------------------------------------------------------------------------------------------------
K5:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto K5
Goto Start_program
'END OF LEVEL14-----------------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'-----------------------------------------------------------------------------------------------------------------
```

تشریح نحوه عملکرد برنامه :

توضیحات مربوط به نحوه گرفتن عدد در لبه بالا رونده از KEYPAD در برنامه قبل داده شده نکته برنامه فوق در گرفتن علامت منفی است که قبل از وارد کردن عدد یعنی زمانی که متغیر COUNT برابر با صفر است اگر کلید منفی زده شود متغیر O1 برای عددA و O2 برای عدد B و O3 برای عدد C برابر با یک خواهد شد . برای مثال هنگام وارد کردن عدد A با زدن کلید (-) متغیر O1 برابر با یک می شود و قبل از نمایش عدد A بر روی LCD ابتدا متغیر O1 چک می شود اگر برابر با یک بود عدد A ابتدا در 1- ضرب شده سپس بر روی LCD نمایش داه می شود .

پیس از گرفتن عددC,B,A در قسمت LEVEL11 دلتا محاسبه شده در متغیر DELTA قرار می گیرد در قسمت LEVEL12 با استفاده از دستور CASE حالت های مختلف DELTA مورد بررسی قرار می گیرد ، اگر $\Delta > 0$ باشد K1,K2 طبق فرمول زیر محاسبه می شوند .

$$X1=\frac{b^2+(Delta)^{1/2}}{2a}$$

$$b^2-(Delta)^{1/2}$$

اگر $\Delta=0$ باشد ریشه مضاعف توسط فرمول زیر محاسبه می شود

$$X1=X2=\frac{-b}{2a}$$

اگر $\triangle < 0$ باشد مثلا اگر 6- باشدمعادله ریشه حقیقی نخواهد داشت عبارت زیر بر روی LCD نمایش داده می شود .

△ IS -6
DELTA IS < 0
ERROR

عملیات محاسبه ریشه های معادله درجه 2 در مدت زمان بسیار کوتاهی انجام می شود تا حدی که ممکن است شما هنوز دستتان را از روی کلید OK بر نداشته باشید به منظور حساس کردن ورودی به لبه بالا رونده در لیبل T11 ورودی چک می شود و تا وقتی که کلید OK غیر فعال نشده اجرای برنامه در همین حلقه خواهد بود با غیر فعال شدن کلید OK اجرای برنامه از حلقه T11 خارج شده و وارد حلقه K5 که در قسمت LEVEL14 قرار دارد قرار می گیرد . در این حلقه تا وقتی که هیچ کلیدی فشار داده نشود عبارت نوشته شده بر روی صفحه نمایش LCD بدون تغییر خواهد بود و با فشار یکی از کلید ها برنامه RESET شده و اجرای برنامه به لیبل START_PROGRAM منتقل خواهدشد .

پروژه ای که 10 عدد را گرفته بزرگترین عدد را به عنوان MAX1 و کوچکتر از آن را به عنوان MAX2 رویLCD نشان دهد

حالا پروژه ای طراحی می کنیم که 10 عدد را گرفته و بزرگتر ین عدد را به عنوان MAX1 و عددی که کوچکتر از بزرگترین عدد است را به عنوان MAX2 برروی LCD نمایش دهد . دراین پروژه از یک LCD (20*4) به عنوان نمایشگر و از یک KEYPAD ماتریسی(4*4) برای وارد کردن اطلاعات استفاده شده است و ورودی حساس به لبه بالا رونده بوده و عدد وارد شده حداکثر می تواند 3 رقمی باشد از یک SPEAKER، 8 اهمی نیز به عنوان ایجاد کننده صدای سوت کوتاه پس از فشار هر کدام از کلید ها استفاده شده است . نحوه عملکرد پروژه بدین صورت می باشد که ابتدا عبارت زیر روی LCD نمایش داده می شود .

ENTER INPUT DATA
INPUT VALUE IS
INPUT COUNTER IS 1

که عبارت INPUT COUNTER IS 1 شماره عدد وارد شده را نشان می دهد پس از وارد کردن عدد مورد نظر بایستی کلیدOK را فشار دهید پس از زدن کلید OK عبارت نوشته شده در خط سوم INPUT COUNTER IS 2 خواهد بود و این روند تا دهمین عدد ادامه پیدا خواهد کرد به عنوان مثال اگر شما اعداد 1 تا 10 را به عنوان اعداد ورودی وارد کرده باشید عبارت نمایش داده شده بر روی LCD به صورت زیر خواهد بود

ENTER INPUT DATA
INPUT VALUE IS
INPUT COUNTER IS 10
MAX1= 10 , MAX2= 9

پس از نمایش MAX2,MAX1 بر روی LCD عبارت نوشته شده برروی LCD تا زمانی که هیچ کلیدی فشار داده نشده بدون تغییر خواهد بود . و با فشار هر کدام از کلید ها برنامه ریست شده و از نو اجرا خواهد شد .

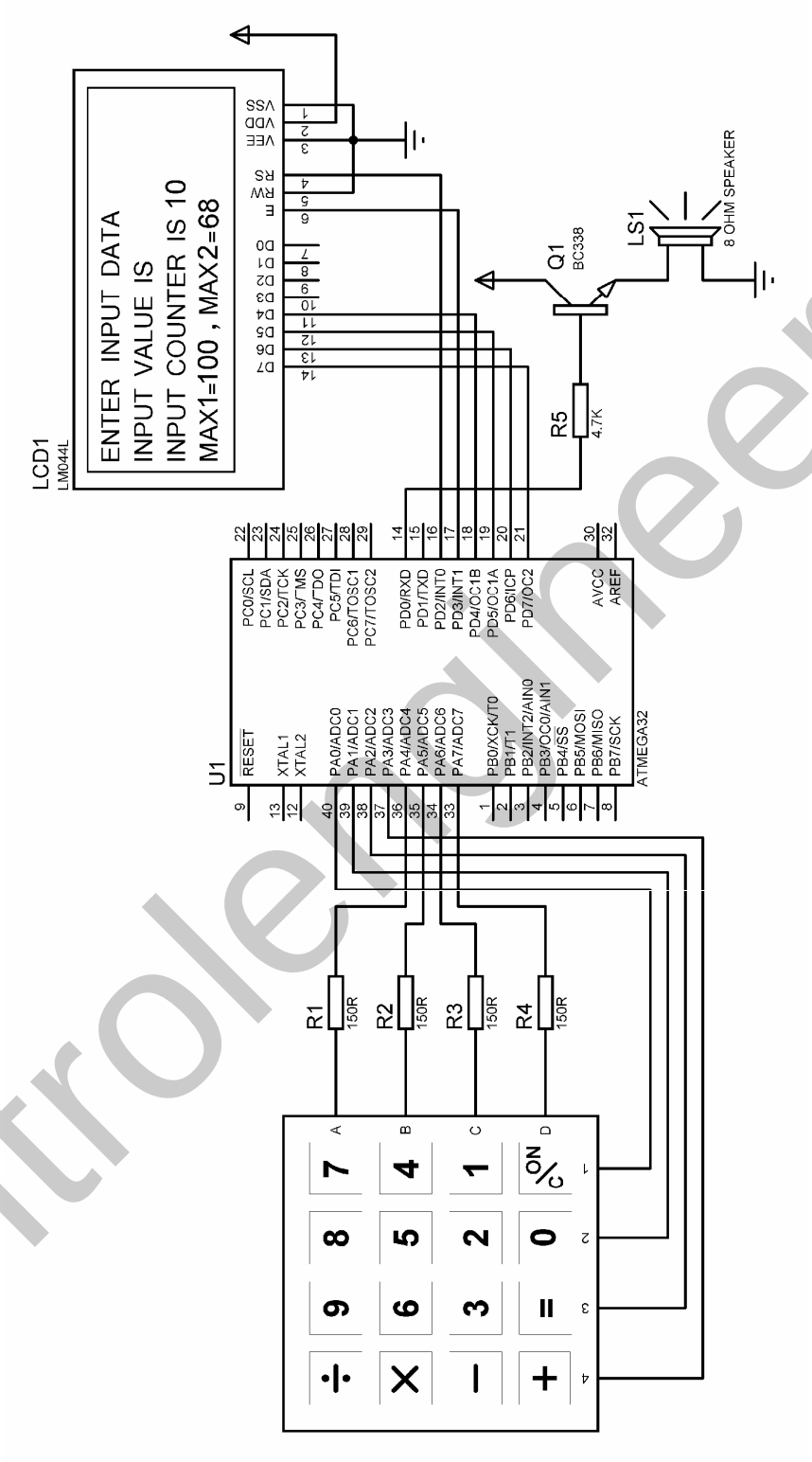
سخت افزار پروژه در شکل 38-2 ارائه شده است.

برنامه نوشته برای پروژه به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Recive_data As Byte , A1 As Long , A2 As Long , A3 As Long
Dim Star As Byte , Count As Byte , A As Long
Dim Hossein As Byte , Max1 As Word , B As Byte , Max2 As Word
Dim T As Byte , S(10) As Long
Config Kbd = Porta , Debounce = 50 , Delay = 100
Config Lcd = 20 * 4
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
            , E = Pind.3 , Rs = Pind.2
Config Pind.0 = Output
'START OF LEVEL1-----------------------------------------------------------------------------------------------
Start_program:
A1 = 0 : A2 = 0 : A3 = 0 : A = 0
B = 0 : Max1 = 0 : Max2 = 0
Hossein = 0
B = Hossein + 1
For T = 0 To 9 Step 1
S(t) = 0
Next T
Cursor Off
Cls : Home
Lcd "ENTER INPUT DATA"
Locate 2 , 1
Lcd "INPUT VALUE IS "
Locate 3 , 1
Lcd "INPUT COUNTER  IS " ; B
'END OF LEVEL1-------------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Star = Lookup(recive_data , Data_code)
'-----------------------------------------------------------------------------------------------
If Star >= 10 Then
If Star = 40 Then
Count = 0
Recive_data = 16
S(hossein) = A
Goto T1
End If
Goto H1
End If
'-----------------------------------------------------------------------------------------------
Sound Portd.0 , 100 , 80
Incr Count
```

```
If Count = 1 Then
A1 = Star
A = A1
S(hossein) = A
End If
If Count = 2 Then
A2 = Star
'A=(A2*10)+A1
A = A1 * 10
A = A + A2
S(hossein) = A
End If
If Count = 3 Then
A3 = Star
A1 = A1 * 100
A2 = A2 * 10
A = A1 + A2
A = A + A3
S(hossein) = A
End If
Cls : Home
Lcd "ENTER INPUT DATA "
Locate 2 , 1
Lcd "INPUT VALUE IS  " ; A
Locate 3 , 1
Lcd "INPUT COUNTER  IS " ; B
'-----------------------------------------------------------------------------------------------
If Count = 3 Then
Count = 0
Goto H2
End If
'-----------------------------------------------------------------------------------------------
R1:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto R1
Goto H1
'END OF LEVEL2-----------------------------------------------------------------------------------------------
'START OF LEVEL3-----------------------------------------------------------------------------------------------
H2:
Cls : Home
Lcd "ENTER INPUT DATA "
Locate 2 , 1
Lcd "INPUT VALUE IS  " ; A
Locate 3 , 1
Lcd "INPUT COUNTER  IS " ; B
T2:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto T2
Star = Lookup(recive_data , Data_code)
If Star <> 40 Then Goto T2
'END OF LEVEL3-----------------------------------------------------------------------------------------------
'START OF LEVEL4-----------------------------------------------------------------------------------------------
T1:
Incr Hossein
If Hossein = 10 Then Goto D3
B = Hossein + 1
D3:
Cls : Home
Lcd "ENTER INPUT DATA"
Locate 2 , 1
Lcd "INPUT VALUE IS "
```

```
Locate 3 , 1
Lcd "INPUT COUNTER  IS " ; B
Sound Portd.0 , 100 , 80
T3:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto T3
If Hossein < 10 Then Goto H1
'END OF LEVEL4-----------------------------------------------------------------------------------------------------
'START OF LEVEL5-----------------------------------------------------------------------------------------------------
Max1 = 0
For T = 0 To 9 Step 1
If S(t) > Max1 Then Max1 = S(t)
Next T
'-----------------------------------------------------------------------------------------------------
For T = 0 To 9 Step 1
If S(t) = Max1 Then S(t) = 0
Next T
'-----------------------------------------------------------------------------------------------------
Max2 = 0
For T = 0 To 9 Step 1
If S(t) > Max2 Then Max2 = S(t)
Next T
'-----------------------------------------------------------------------------------------------------
Locate 4 , 1
Lcd "MAX1= " ; Max1 ; ",MAX2= " ; Max2
'END OF LEVEL5-----------------------------------------------------------------------------------------------------
'START OF LEVEL6-----------------------------------------------------------------------------------------------------
T4:
Recive_data = Getkbd()
If Recive_data <> 16 Then
Goto Start_program
Else
Goto T4
End If
'END OF LEVEL6-----------------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'-----------------------------------------------------------------------------------------------------
```



شکل 2-38 شماتیک طراحی شده برای مدار تشخیص دهنده MAX1 ، MAX2 از 10 عدد ورودی

دستورات جدید :

در برنامه فوق از متغیر های آرایه ای استفاده شده که فرم کلی آن ها به صورت زیر است .

NAME(VAR)

NAME : نام متغیر آرایه ای می باشد .

VAR : نام متغیری که شماره یا اندیس مربوط به آرایه را نشان می دهد .

به طور کلی آرا یه شامل تعداد زیادی عنصر از یک نوع می باشد که در فضای پیوسته و پشت سر هم قرار می گیرند به عناصر آرایه از طریق نام آرایه واندیس می توان دسترسی پیدا کرد . پس هنگامی که بخواهیم از یک نوع داده تعداد زیادی تعریف کتیم از متغیر آرایه ای استفاده خواهیم کرد . به عنوان مثال اگر بخواهیم 5 متغیر بایتی با نام A تعریف کنیم به صورت زیر عمل می کنیم .

```
CONFIG A(5) AS BYTE
```

که در این صورت متغیر های تعریف شده عبارتند از A(0),A(1),A(2),A(3),A(4) توجه داشته باشید A(5) جزء متغیر های تعریف شده نمی باشد . متغیر های آرایه ای موارد استفاده زیادی دارند به عنوان مثال اگر بخواهیم 5 متغیر هم نام از نوع BYTE تعریف کرده و محتوای همه آن ها را برابر با 100 قرار دهیم به صورت زیر عمل می کنیم .

```
DIM H(5) AS BYTE
DIM T AS BYTE
FOR T= 0 TO 4 STEP1
H(T)=100
NEXT T
STOP
```

تشریح نحوه عملکرد برنامه :

متغیر آرایه ای تعریف شده برای 10 عدد گرفته شده متغیر S(10) می باشد همان طور که قبلا هم گفته شد طبق این تعریف ما می توانیم از متغیر های S(0) تا S(9) در برنامه استفاده کنیم و خود S(10) جزء متغیر آرایه ای تعریف شده نمی با شد . از متغیر HOSSEIN به عنوان اندیس آرایه یا نشان دهنده شماره آرایه استفاده شده است . از متغیر B نیز به عنوان شمارنده ورودی (INPUT COUNTER) استفاده شده است ، به طوری که هنگام وارد کردن اولین عدد مقدار این متغیر 1 و هنگام وارد کردن آخرین عدد مقدار این متغیر 10 خواهد بود .

در قسمت LEVEL1 ابتدا متغیر ها RESET شده سپس اولین عدد ورودی درخواست می شود .

در قسمت LEVEL2 عدد وارد شده ابتدا در متغیر A قرار گرفته سپس با زدن دکمه OK در متغیر S(HOSSEIN) قرار می گیرد مقدار متغیرHOSSEIN ابتدا صفر بوده اولین عدد وارد شده در متغیر S(0) قرار می گیرد پس از هر بار زدن دکمه OK یک واحد به HOSSEIN اضافه می شود . در ضمن به متغیر B که

شمارنده اعداد وارد شده می باشد توسط دستور B= HOSSEIN +1 نیز یک واحد اضافه خواهد شد .در رابطه با نحوه دریافت ورودی در پروژه های قبل به صورت مفصل توضیح داده شده است به همین دلیل از توضیح آن صرف نظر می شود . پس از این که 10 تا عدد وارد شد ، برنامه وارد قسمت LEVEL 5 می شود . در این قسمت ابتدا توسط دستورات زیر MAX1 شناسایی می شود .

```
MAX1=0
FOR T= 0 TO 9 STEP 1
IF S(T) > MAX1 THEN MAX1 = S(T)
NEXT T
```

سپس برای آن که عدد ماکسیمم را از اعداد وارد شده حذف کنیم توسط دستورات زیر این عدد را صفر خواهیم کرد.

```
FOR T= 0 TO 9 STEP 1
IF S(T) = MAX1 THEN S(T)=0
NEXT T
```

در ضمن این احتمال وجود دارد که بزرگترین عدد چند بار وارد شده باشد که دراین صورت نیز توسط حلقه FOR_NEXT فوق حذف خواهد شد .

سپس عدد MAX2 توسط دستورات زیر شناسائی می شود .

```
MAX2=0
FOR T=0 TO 9 STEP1
IF S(T) > MAX2 THEN MAX2 = S(T)
NEXT T
```

پس از شناسایی MAX2 مقدار متغیر های MAX2,MAX1 روی LCD نمایش داده می شود . سپس برنامه وارد قسمت LEVEL6 می شود که در این قسمت تا زمانی که هیچ کلیدی فشار داده نشده اجرای برنامه در حلقه T4 متوقف خواهد شد . پس از فشار هر کدام از کلید ها اجرای برنامه به لیبل START_PROGRAM منتقل می شود .

نحوه پیکره بندی LCD گرافیکی :

توجه داشته باشید قبل از شروع پیکره بندی بایستی 2 فایل (GLCDKSLOD.LBX) ، (FONT8X8.FONT) ، (FONT16X16.FONT) را در مسیری که فایل هگزBASCOM را ذخیره می کنید ، کپی کنید ، این فایل ها در CD پیوست کتاب ارائه شده است .

در ضمن برای پروژه هایی که در آن ها از LCD گرافیکی استفاده شده از BASCOM1.11.8.7 به عنوان کامپایلرو از پرتئوس 6.2 به عنوان سیمولاتور استفاده شده است و در پروژه های این کتاب از LCD گرافیکی نوع 128*64 WITH KS108 CONTROLLERS استفاده شده است که نحوه پیکره بندی آن به صورت زیر می باشد .

```
$LIB "GLCDKS108.LIB"
```

```
CONFIG GRAPHLCD= 128*64SED,DATAPORT=PORT,CONTROLPORT=PORT,CE
=PIN,CEZ=PIN,CD=PIN,RD=PIN,RESET=PIN,ENABLE=PIN
```

قبل از پیکره بندی بایستی فایل کنترلر مربوطه توسط دستور $LIB معرفی شود .

DATAPORT : پورتی است که به عنوان ورودی داده برای LCD مورد استفاده قرار می گیرد به عنوان مثال DATAPORT= PORT D که در این صورت پایه های PORT D.0_PORTD.7 به پایه های DB0_DB7 از LCD متصل می شوند .

DATAPORT :مشخص کننده پورتی است که از پایه های آن برای کنترل LCD استفاده می شود .

CE : شماره پایه ای از CONTROLPORT که برای فعال کردن پایه CS1 در LCD در نظر گرفته شده است .

CE2 : شماره پایه ای از CONTROLPORT که برای فعال کردن پایه CS2 در LCD در نظر گرفته شده است .

RD : شماره پایه ای از CONTROLPORT که برای کنترل کردن پایه R/W موجود در LCD استفاده می شود .

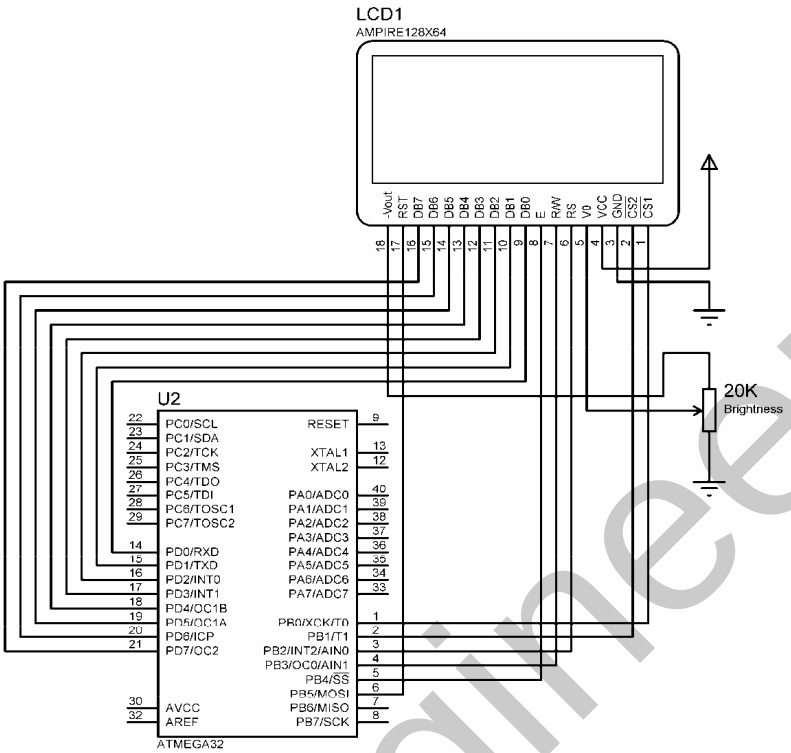
RESET : شماره پایه ای از CONTROLPORT که برای کنترل پایه RST موجود در LCD استفاده شده است .

CD : شماره پایه ای ازCONTROLPORT که برای کنترل کردن پایه RS موجود در LCD استفاده شده است.

ENABLE : شماره پایه ای از CONTROLPORT که برای کنترل کردن پایه E موجود بر روی LCD استفاده شده است .

به طور مثال اگر LCD گرافیکی را مانند شکل 2-39 میکرو متصل کنیم . نحوه پیکره بندی بدین صورت خواهد بود.

```
$LIB"GLCDKS108.LIB"
CONFIG GRAPHLCD=128*64SED,DATAPORT=PORTD,CONTROLPORT=PORTB,CE=0_
,CE2=1,CD=2,RD=3,RESET=5,ENABLE=4
```



شکل 2-39 نحوه اتصال LCD گرافیکی 64 * 128 به میکروکنترلر

نحوه نمایش تصویر بر روی LCD گرافیکی:

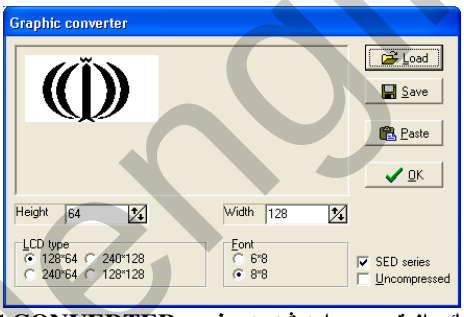
ابتدا وارد محیط PAINT ، ویندوز شده و اندازه تصویر را به 64*128 تغییر می دهیم سپس تصویر مورد نظر را LOAD کرده یا آن رادر محیط PAINT طراحی کنید . توجه داشته باشید اندازه تصویر در قسمت پایین گوشه سمت چپ محیط PAINT بوده و هنگامی که اندازه تصویر را توسط موس تغییر می دهیم مشاهده می شود . به عنوان مثال برای نمایش آرم پرچم کشورمان بر روی LCD گرافیکی ابتدا آن را در محیط PAINT و با اندازه 64 * 128 مانند شکل 2-40 طراحی می کنیم .

شکل 2-40 طراحی شکل موردنظر در محیط PAINT

سپس فایل مورد نظر را در آدرسی که فایل هگز برنامه را SAVE خواهید کرد با پسوند *.BMP ، SAVE کنید سپس واردمحیطBASCOM1.11.8.7شدهوازمنوی TOOLS گزینهGRAPHICCONVERTOR را انتخاب کنید با کلیک بر روی این گزینه پنجره شکل 2-41 باز می شود .

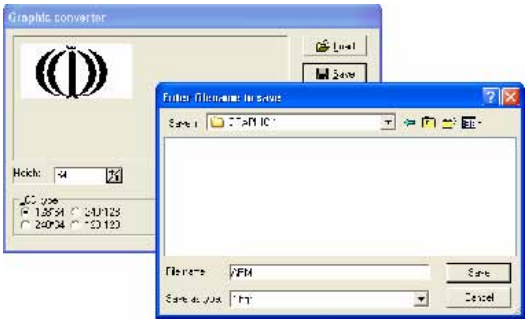
شکل 2-41 نمائی از پنجره GRAPHIC CONVERTER

در قسمت LCD TYPE گزینه 64*128 در قسمت FONT گزینه 8*8 را انتخاب کنید سپس گزینه SED SERIES را فعال کرده و تصویری را که با پسوند *.BMP ذخیره کرده اید توسط کلید LOAD وارد کنید .



شکل 2-42 نمائی از تصویر وارد شده در پنجره GRAPHIC CONVERTER

پس از وارد کردن تصویر می توانید اندازه آن را نیز توسط قسمت های WIDTH و HEIGHT تغییر دهید سپس با زدن کلید SAVE فایل*.BAF(BASCOM GRAPHIC FILE) تصویر را در آدرسی که فایل هگز برنامه را ذخیره خواهید کرد SAVE کنید.



شکل 2-43 نمائی از SAVE تصویر BGF ایجاد شده

پس از ذخیره فایل با زدن دکمه OKپنجره GRAPHIC CONVERTER را ببندید .

حالا می توانید در محیط BASCOM توسط دستور SHOWPIC تصویر موردنظر را بر روی LCD گرافیکی نمایش دهید برنامه نوشته شده در محیط BASCOM به صورت زیر است

```
$regfile = "m32Def.dat"
$crystal = 8000000
$lib "gLcdKS108.lib"
Config Graphlcd = 128 * 64sed , Dataport = Portd , Controlport = Portb_
, Ce = 0 , Ce2 = 1 , Cd = 2 , Rd = 3 , Reset = 5 , Enable = 4
'-----------------------------------------------------------------------------------------------------------
Program_start1:
Cls
Showpic 0 , 0 , Pic1
Stop
'-----------------------------------------------------------------------------------------------------------
    Pic1:
      $bgf "arm.bgf"
'-----------------------------------------------------------------------------------------------------------
```

از شماتیک معرفی شده برای LCD در شکل 2-39 به عنوان سخت افزار استفاده کنید . توجه داشته باشید که قبل از کامپایل کردن برنامه بایستی فایل های (FONT8*8.FONT) ، (FONT16*16.FONT) و GLCDK108.LBX را در آدرس برنامه ذخیره شده کپی نمایید .

در برنامه فوق از دستور SHOWPIC استفاده شده که شکل کلی آن به صورت زیر است

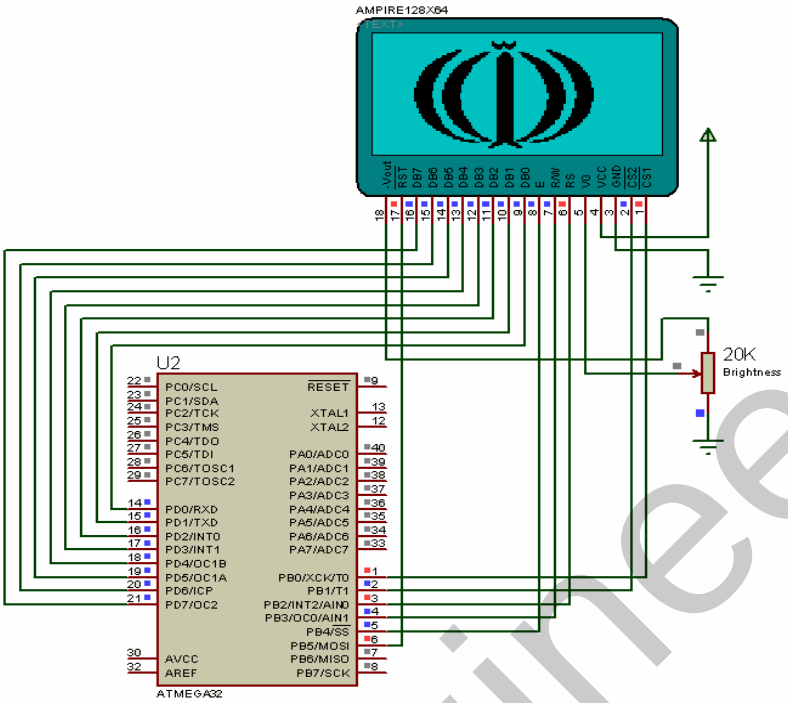
SHOWPIC X,Y,LABLE

X مکان قرار گیری افقی و Y مکان قرار گیری عمودی است. LABLE نام برچسبی است که فایل *.BGF مربوط به تصویر در آن قرار دارد . توجه داشته باشید که پیکسل (0,0) از گوشه سمت چپ بالای صفحه LCD شروع می شود .

همچنین از دستور $BGF استفاده شده که فرم کلی آن به صورت زیر است

$BGF “ FILE.BGF ”

از این دستور برای معرفی فایل BGF تصویر مورد نظر استفاده می شود . در برنامه به جای FILE.BGF نام فایل ذخیره شده توسط GRAPHIC CONVERTER نوشته می شود . از دستور CLS نیز استفاده شده است که کل صفحه LCD گرافیکی را پاک می کند . نتیجه مشاهده شده در سیمولاتور PROTEUS 6.2 در شکل 2-44 نشان داده شده است .

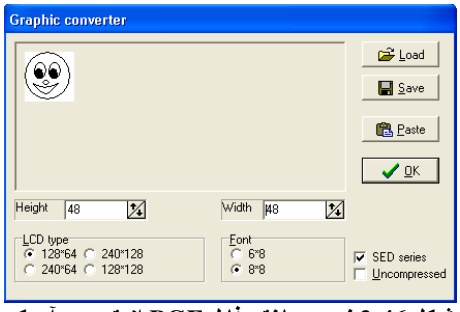


شکل 2-44 نتیجه نشان داده شده در سیمولاتور پروتئوس 6.2 برای تصویر آرم

در LCD گرافیکی شما می توانید یک تصویرکوچکتر را بر روی یک تصویر بزرگتر نمایش دهید برای مثال می توان ابتدا تصویر یک کادر یا حاشیه را در روی LCD نمایش داده سپس تصویر یک آدمک را روی آن نمایش دهید . برای این کار ابتدا تصویر کادر یا حاشیه را در محیط PAINT طراحی و با پسوند *.BMP ذخیره کرده و سپس توسط GRAPHIC CONVERTER با تعداد پیکسل 64*128 به فایل *.BGF تبدیل نموده و در آدرس برنامه ذخیره می کنیم .

شکل 2-45 نحوه ساختن فایل BGF از تصویر کادر یا حاشیه

سپس تصویر آدمک را در محیط PAINT طراحی کرده و با پسوند *.BMP ذخیره می کنیم سپس توسط GRAPHIC CONVERTER تصویر مورد نظر را LOAD کرده و به صورت شکل زیر با تعداد پیکسل 48*48 به فایل *.BGF تبدیل نموده و در آدرس برنامه ذخیره می کنیم .



شکل 2-46 نحوه ساختن فایل BGF از تصویر آدمک

سپس برنامه زیر را در محیط BASCOM می نویسیم

```
$regfile = "m32Def.dat"
$crystal = 8000000
$lib "gLcdKS108.lib"
Config Graphlcd = 128 * 64sed , Dataport = Portd , Controlport = Portb_
 , Ce = 0 , Ce2 = 1 , Cd = 2 , Rd = 3 , Reset = 5 , Enable = 4
'-----------------------------------------------------------------
Program_start1:
Cls
Showpic 0 , 0 , Pic1
Showpic 15 , 10 , Pic2
Showpic 70 , 10 , Pic2
Stop
'-----------------------------------------------------------------
    Pic1:
    $bgf "KADR.bgf"
    Pic2:
    $bgf "ADAMAK.bgf"
```

در این حالت نتیجه نمایش داده شده بر روی LCD گرافیکی در پروتئوس به صورت شکل 2-47 خواهد بود .

برا ی نوشتن به صورت کاراکتری بر روی صفحه LCD گرافیکی به صورت زیر عمل می کنیم.

ابتدا فایل مربوط به تعداد پیکسل های مر بوط به کاراکتر های نوشتاری را که قبلا در آدرس برنامه کپی کرده ایم معرفی می کنیم . این فایل ها عبارتند از (FONT8X8.FONT) ، (FONT16X16.FONT) ، نحوه معرفی فونت 8*8 به صورت زیر است

$ INCLUDE “FONT8X8.FONT”

اگر تعداد پیکسل های LCD 128*64 بوده و نوع فونت معرفی شده 8*8 باشد LCD گرافیکی دارای 16 ستون و 8 سطر کاراکتری خواهد بود . نحوه معرفی فونت 16*16 به صورت زیر است .

$ INCLUDE “FONT16X16.FONT”

اگر تعداد پیکسل های 128*64برابر با LCD بوده و نوع فونت معرفی شده 16*16 باشد LCD گرافیکی دارای 8 ستون و4 سطر خواهد بود . برای نوشتن بر روی LCD گرافیکی پس از معرفی فایل مربوط به فونت مورد نظر به صورت زیر عمل می کنیم . برای مثال می خواهیم با فونت (8*8) روی LCD بنویسیم .

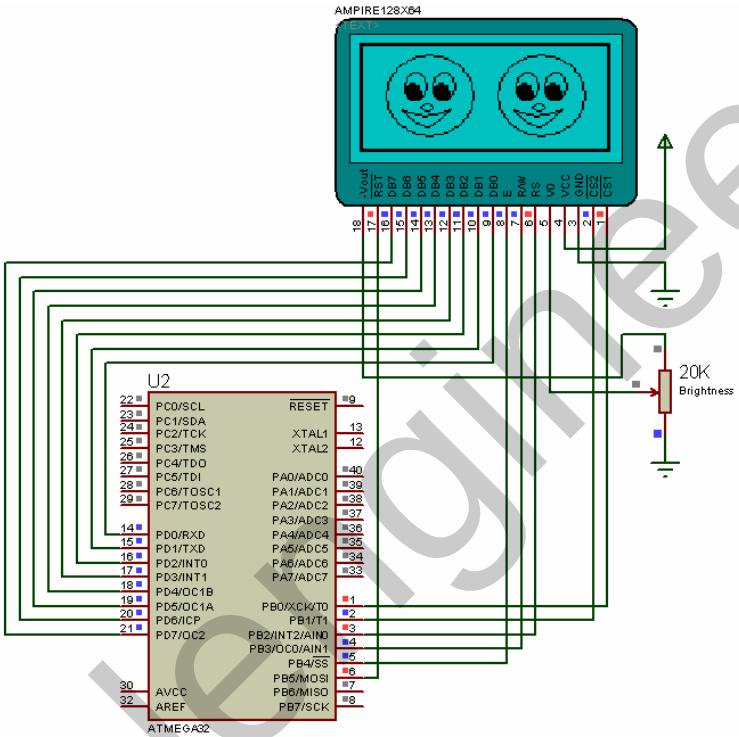
SETFONT FONT8X8

LCDAT X,Y,VAR

X : شماره سطر مورد نظر می باشد .

Y : شماره ستون مورد نظر می باشد .

VAR : می تواند یک متغیر رشته ای (STRING) یا یک ثابت نوشتاری باشد .



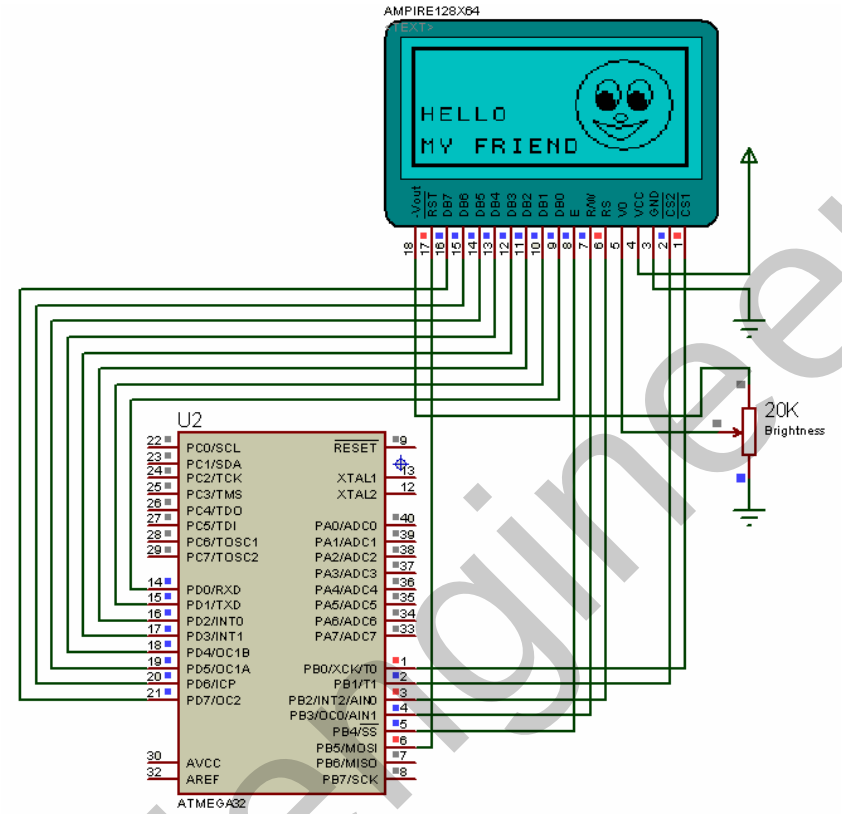
شکل 2-47 نتیجه نمایش داده شده در محیط پروتئوس 6.2 برای نمایش تصویر آدمک بر روی تصویر کادر

از شماتیک معرفی شده در شکل 2-39 استفاده کرده و برنامه زیر را برای آن بنویسید .

```
$regfile = "m32Def.dat"
$crystal = 8000000
$lib "gLcdKS108.lib"
$include "font8x8.font"
Config Graphlcd = 128 * 64sed , Dataport = Portd , Controlport = Portb_
  , Ce = 0 , Ce2 = 1 , Cd = 2 , Rd = 3 , Reset = 5 , Enable = 4
'-----------------------------------------------------------------
Program_start1:
Cls
Showpic 0 , 0 , Pic1
Showpic 75 , 10 , Pic2
Setfont Font8x8
Lcdat 5 , 7 , "HELLO"
Lcdat 7 , 7 , "MY FRIEND"
Wait 2
Stop
'-----------------------------------------------------------------
    Pic1:
```

```
$bgf "KADR.bgf"
Pic2:
$bgf "ADAMAK.bgf"
```

نتیجه نمایش داده شده در محیط پروتئوس برای برنامه فوق در شکل 2-48 نشان داده شده است.



شکل 2-48 نتیجه نمایش داده شده در محیط پروتئوس 6.2 برای نمایش همزمان TEXT و تصویر بر روی LCD گرافیکی

نحوه پیکره بندی و استفاده از مبدل آنالوگ به دیجیتال داخلی

در را بطه با مشخصات ADC داخلی میکروکنترلرهای AVR و نحوه کاهش نویز در ان در فصل یک به طور مفصل بحث شده است که در این جا از پرداختن به این موضوع صرفه نظر می کنیم ، دقت ADC داخلی AVR 10 بیتی بوده و می تواند ولتاژ مرجع (REFERENCE) اعمالی را به 1024 قسمت تقسیم کند .

برای مثال اگر ولتاژ مرجع اعمالی برابر با 5 ولت باشد.

5/1024 = 0.005

یعنی به ازای هر 5mv اعمالی به پایه ورودی ADC عدد مربوط به آن یک شماره صعود خواهد کرد . توجه داشته باشید که در این حالت ADC برای صفرولت ورودی عدد صفر و برای 5 ولت عدد 1023 را در رجیستر مربوط قرار می دهد . به عنوان مثال اگر ورودی آنالوگ برابر با 3.25 ولت باشد .

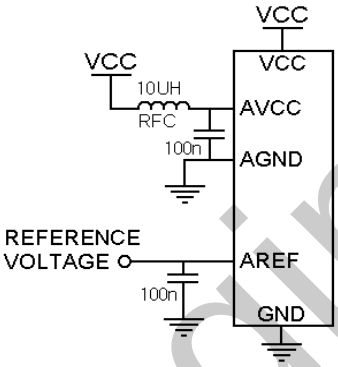
3.25/0.005=650

ADC داخلی میکروکنترلرهای AVR توسط دستور زیر پیکره بندی می شود .

CONFIG ADC = SINGLE/FREE,PRESCALER=AUTO

SINGLE/FREE : برای انجام عمل ADC می توانید دو مد SINGLE,FREE استفاده کنید . اگر از دستور GETADC( ) در برنامه استفاده کنید بایستی مد SINGLE را انتخاب کنید .

PRESCALER : کلاک مربوط به ADC را مشخص می کند با قرار دادن PRESCALER=AUTO کامپایلر بهترین کلاک را برای ADC تعیین می کند . نحوه استفاده از پایه های AREF,AGND,AVCC به صورت زیر خواهد بود .



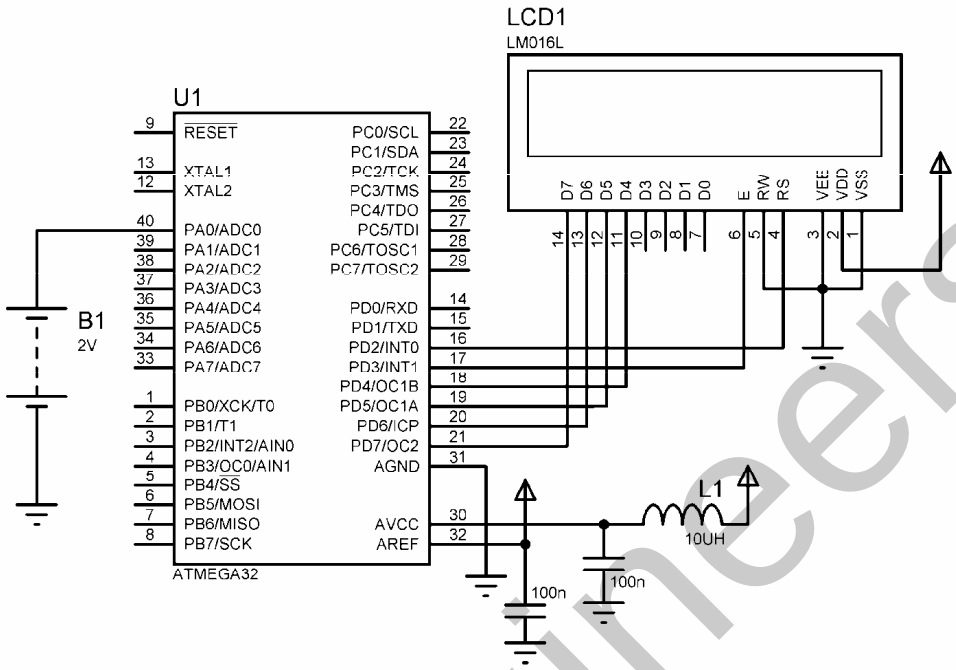
شکل 2-49 نحوه استفاده از پایه های AVCC ، AGND و AREF

با استفاده از دستور زیر می توانید سیگنال آنالوگ وارد شده به کانال های (0-7) را به مقدار دیجیتال تبدیل کنید .

VAR=GETADC(CHANEL)

VAR : متغیر ی از نوع WORD که مقدار دیجیتال در آن قرار می گیرد.

CHANEL : شماره کانال مربوط به ADC را نشان می دهد و می تواند یک متغیر عددی یا یک ثابت عددی باشد . در ضمن توسط دستور START ADC ، ADC شروع به نمونه برداری کرده و توسط دستور STOP ADC ، تغذیه ADC قطع می شود و عملیات تبدیل پایان می پذیرد . این دستورات برای شروع و توقف ADC حتما بایستی نوشته شود .در مثال شکل 2-50 ولتاژ وارد شده به کانال صفر ADC تبدیل به دیجیتال شده وبر روی LCD نوشته می شود.



شکل 2-50 شماتیک مربوط به نمایش عدد دیجیتال شده بر روی LCD

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Adc_input As Word , Save As Word
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
            , E = Pind.3 , Rs = Pind.2
'-----------------------------------------------------------------------------------------------------------
Start Adc
Cursor Off
Start_program:
Adc_input = Getadc(0)
If Adc_input = Save Then Goto Start_program
Cls : Home
Lcd "ADC INPUT IS "
Lowerline
Lcd Adc_input
Waitms 100
Save = Adc_input
Goto Start_program
```

نحوه کار با وقفه داخلی ADC :

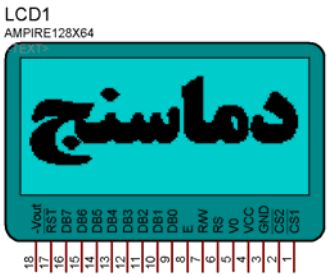
پس از پایان نمونه برداری از سیگنال آنالوگ پرچم وقفه مربوطه یک می شود و در شرایطی که وقفه سراسری توسط دستور ENABLE ENTERRUPTS و وقفه ADC توسط دستور ENABLE ADC و ISR وقفه توسط ON ADC LABLE فعال شده باشند می توان به زیر برنامه وقفه پرش کرد که LABLE نام ISR یا زیر برنامه وقفه می باشد . می توان برای کاهش نویز سیستم میکروکنترلر را در زمان نمونه برداری در مد IDLE قرار داد این مد اسلیپ در قسمت مربوطه به طور کامل معرفی شده است سپس میکرو به صورت خودکار

با بالا رفتن پرچم وقفه اتمام تبدیل ADC از این مد بیدار شده و مقدار دیجیتال شده را در متغیر نوع WORD قرارمی دهد . برای استفاده از وقفه ADC برنامه نوشته شده در مثال قبل را بصورت زیر تغییر دهید.

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Dim Adc_input As Word , Save As Word
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
            , E = Pind.3 , Rs = Pind.2
'-----------------------------------------------------------------------------------------------------------
Enable Interrupts
Enable Adc
On Adc Adc_isr
'-----------------------------------------------------------------------------------------------------------
Start Adc
Cursor Off
Do
Adc_input = Getadc(0)
Idle
Loop
'-----------------------------------------------------------------------------------------------------------
Adc_isr:
If Adc_input = Save Then Goto H1
Cls : Home
Lcd "ADC INPUT IS "
Lowerline
Lcd Adc_input
Save = Adc_input
H1:
Return
'-----------------------------------------------------------------------------------------------------------
```

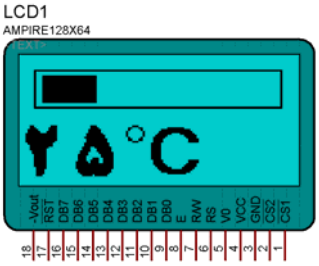
پروژه نمایش دما بر روی LCD گرافیکی با فونت فارسی

حالا پروژه ای طراحی می کنیم که دمای محیط را توسط سنسور دمای LM35 گرفته و برروی LCD گرافیکی به صورت مانیتوریگ نشان دهد . در این پروژه از LCD گرافیکی 128*64 به عنوان نمایشگر استفاده شده است نحوه علکرد پروژه به صورت زیر می باشد . ابتدا تصویر شکل 2-51 بر روی LCD نشان داده می شود .



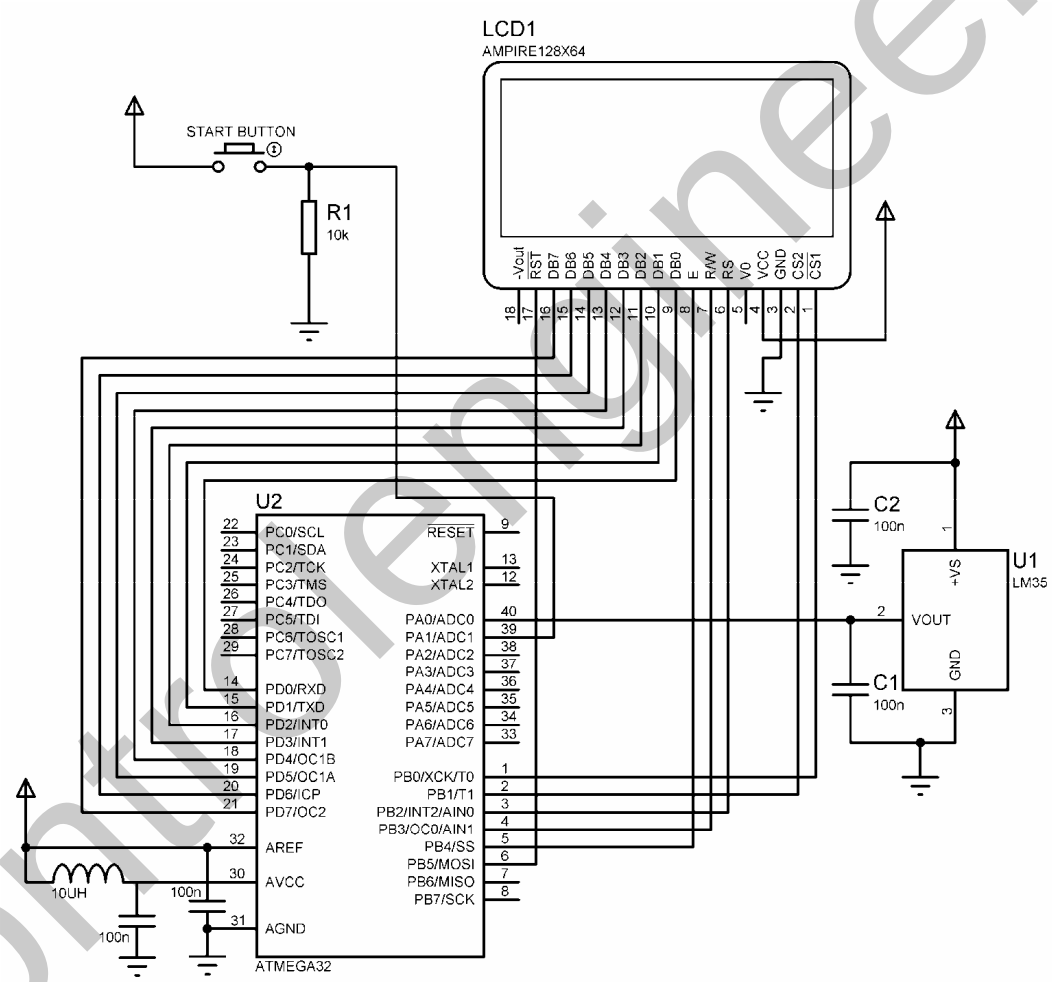
شکل 2-51 عبارت نمایش داده شده بر روی LCD گرافیکی پروژه قبل از فشار کلید START BUTTON

و تا زمانی که کلید START BUTTON فشار داده نشود تصویر فوق بر روی LCD گرافیکی ثابت خواهد بود . پس از فشار این کلید دمای محیط به صورت شکل 2-52 بر روی LCD گرافیکی نمایش داده خواهد شد .



شکل 2-52 نحوه نمایش دما بر روی LCD گرافیکی پروژه

سخت افزار پروژه در شکل 2-53 ارائه شده است .



شکل 2-53 شماتیک طراحی شده برای دماسنج دیجیتالی با فونت فارسی

در این پروژه از سنسوردمای LM35 به عنوان حسگر دما استفاده شده است. ولتاژ خروجی این سنسور به صورت خطی و متناسب با دمای محیط تغییر می کند به طوری که به ازای هر درجه سانتی گراد ولتاژ خروجی

سنسور LM35 ، 10 میلی ولت افزایش پیدا می کند . برای مثال در دمای 25 درجه سانتی گراد ولتاژ خروجی سنسور به صورت زیر خواهد بود .

25*10mV=0.25V

از طرفی دقت ADC داخلی AVR ، 10 بیتی بوده و می تواند ولتاژ REFERNCE اعمالی را به 1024 قسمت تقسیم کند در این پروژه مقدار ولتاژ REFERNCE ، 5 ولت انتخاب شده است که در این صورت ولتاژ اعمالی به ورودی ADC برای افزایش یک شماره مقدار دیجیتال شده به صورت زیر خواهد بود .

5/1204=0.005V

پس با اعمال 0.25 ولت به ورودی کانال ADC دیجیتال شده به صورت زیر خواهد بود .

0.25/0.005=50

که در این شرایط برای به دست آوردن دمای واقعی بایستی مقدار دیجیتال شده را بر عدد 2 تقسیم کنیم .

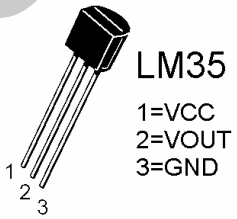
50/2=25

مشخصات انواع مختلف سنسور دمای LM35 در جدول زیر داده شده است .

DEVICE	TEMP RANGE	ACCURECY	OUTPUT SCALE
LM35A	-55C TO +150C	+1.0C	10MV/C
LM35	-55C TO +150C	+1.5C	10MV/C
LM35CA	-40C TO +110C	+1.0C	10MV/C
LM35C	-40C TO +110C	+1.5C	10MV/C
LM35D	0C TO +100C	+2.0C	10MV/C

جدول مشخصات انواع مختلف سنسور دمای LM35

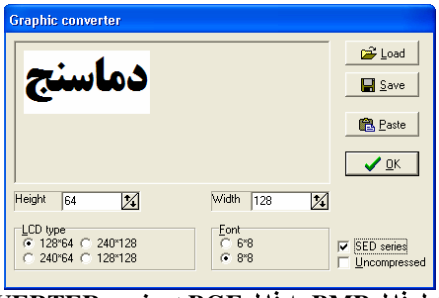
آرایش پایه های سنسور دمای LM35 در شکل 2-54 ارائه شده است .



شکل 2-54 شکل ظاهری و ترتیب پایه های LM35

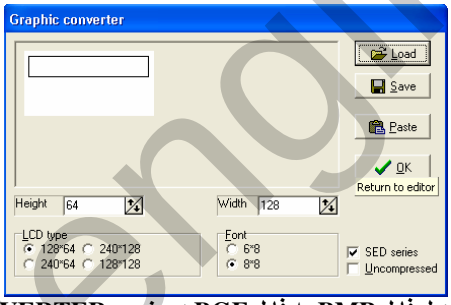
محدوده دمایی در نظر گرفته شده برای این پروژه دمای صفر تا 99 درجه سانتی گراد می باشد . توجه داشته باشید که برای تست کردن پروژه در محیط سیمولاتور PROTEUS بایستی ولتاژ ورودی ADC را با استفاده از یک باطری به میکروکنترلر اعمال کنیم. برای ساختن فایل BGF.* مربوط به تصاویری که بر روی LCD گرافیکی نمایش داده خواهند شد به صورت زیر عمل می کنیم .

ابتدا تصویردماسنج را در محیط PAINT طراحی کرده و با پسوند BMP در آدرسی که برنامه BASCOM را ذخیره خواهیم کرد SAVE می کنیم ، سپس توسط پنجره GRAPHIC CONVERTER آن را با اندازه 128*64 و نام TERMOMETER1.BGF در آدرسی که برنامه اصلی را ذخیره خواهیم کرد SAVE می کنیم . شکل 2-55 نحوه انجام این عملکرد را نشان می دهد .



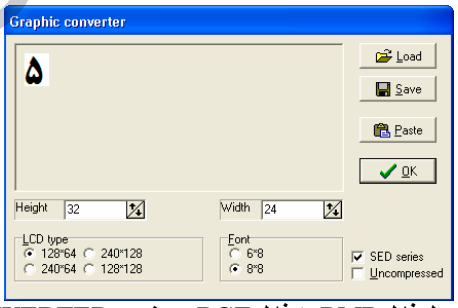
شکل 2-55 نمائی از تبدیل فایل BMP به فایل BGF در پنجره GRAPHIC CONVERTER

سپس این عملیات را روی تصویر کادر مربوط به حالت نموداری نمایش دما انجام می دهیم و آن را با نام CADR.BGF و با اندازه 128*64 در آدرس برنامه اصلی ذخیره می کنیم .



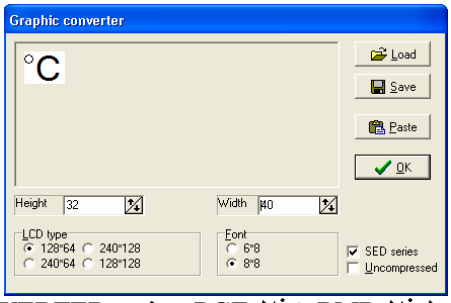
شکل 2-56 نمائی از تبدیل فایل BMP به فایل BGF در پنجره GRAPHIC CONVERTER

سپس عملیات فوق را بر روی تصاویر اعداد 0 تا 9 انجام داده و آنها را با نام های 0.BGF تا 9.BGF با اندازه 32*24 در آدرس برنامه اصلی ذخیره می کنیم.



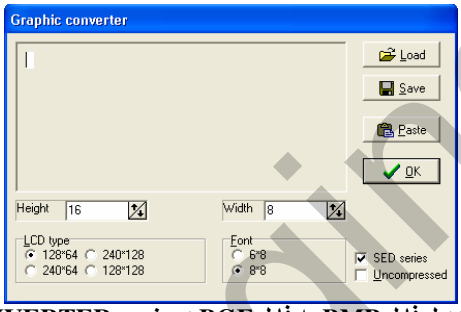
شکل 2-57 نمائی از تبدیل فایل BMP به فایل BGF در پنجره GRAPHIC CONVERTER

سپس تصویر مر بوط به درجه سانتی گراد را با نام CANTIGRAD.BGF و با اندازه 32*48 در آدرس برنامه ذخیره می کنیم .



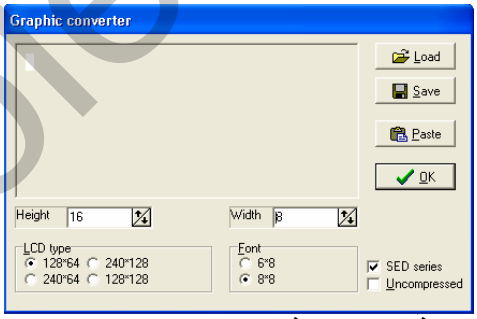
شکل 2-58 نمائی از تبدیل فایل BMP به فایل BGF در پنجره GRAPHIC CONVERTER

سپس تصویر یک خط عمودی را برای پر کردن داخل کادر مربوط به حالت نمایش نموداری دما با نام LINE.BGF و با اندازه 16*8 در آدرس برنامه ذخیره می کنیم .



شکل 2-59 نمائی از تبدیل فایل BMP به فایل BGF در پنجره GRAPHIC CONVERTER

سپس برای پاک کردن قسمت داخلی کادر مربوطه به حالت نمایش نموداری دما یک تصویر سفید را با اندازه 16*8 و با نام CLS_TEMP.BGF در آدرس برنامه اصلی ذخیره می کنیم.



شکل 2-60 نمائی از تبدیل فایل BMP به فایل BGF در پنجره GRAPHIC CONVERTER

برنامه نوشته شده برای پروژه در محیط BASCOM به صورت زیر می باشد .

```
$regfile = "m32Def.dat"
$crystal = 8000000
$lib "gLcdKS108.lib"
Config Graphlcd = 128 * 64sed , Dataport = Portd , Controlport = Portb , Ce = 0_
    , Ce2 = 1 , Cd = 2 , Rd = 3 , Reset = 5 , Enable = 4
Dim X As Byte , Y As Byte , Temp As Word , Hossein As String * 3 , A As Byte
Dim O1 As String * 1 , O2 As String * 1 , O3 As String * 1 , Lcd_data As Byte
Dim Save As Byte , H As Byte
```

```
Config Adc = Single , Prescaler = Auto
Config Pina.1 = Input
'START OF LEVEL1-----------------------------------------------------------------------------------------
Showpic 0 , 0 , Pic11
Do
If Pina.1 = 1 Then Exit Do
Loop
'END OF LEVEL1-------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------
Start Adc
Program_start1:
Temp = Getadc(0)
Temp = Temp / 2
If Temp = Save Then Goto Program_start1
Save = Temp
'END OF LEVEL2-------------------------------------------------------------------------------------------
'START OF LEVEL3-----------------------------------------------------------------------------------------
Showpic 0 , 0 , Pic10
For X = 8 To 110 Step 1
Showpic X , 8 , Pic15
Next X
H = Temp + 8
For X = 8 To H Step 1
Showpic X , 8 , Pic12
Next X
'END OF LEVEL3-------------------------------------------------------------------------------------------
'START OF LEVEL4-----------------------------------------------------------------------------------------
Hossein = Str(temp)
A = Len(hossein)
'END OF LEVEL4-------------------------------------------------------------------------------------------
'START OF LEVEL5-----------------------------------------------------------------------------------------
Select Case A
  Case Is = 1
  Lcd_data = Val(hossein)
  Select Case Lcd_data
    Case Is = 0
    Showpic 0 , 32 , Pic0
    Case Is = 1
    Showpic 0 , 32 , Pic1
    Case Is = 2
    Showpic 0 , 32 , Pic2
    Case Is = 3
    Showpic 0 , 32 , Pic3
    Case Is = 1
    Showpic 0 , 32 , Pic1
    Case Is = 4
    Showpic 0 , 32 , Pic4
    Case Is = 5
    Showpic 0 , 32 , Pic5
    Case Is = 6
    Showpic 0 , 32 , Pic6
    Case Is = 7
    Showpic 0 , 32 , Pic7
    Case Is = 8
    Showpic 0 , 32 , Pic8
    Case Is = 9
    Showpic 0 , 32 , Pic9
    End Select
  Showpic 24 , 32 , Pic14
'-----------------------------------------------------------------------------------------------------
  Case Is = 2
```

```
O1 = Mid(hossein , 1 , 1)
Lcd_data = Val(o1)
Select Case Lcd_data
  Case Is = 0
  Showpic 0 , 32 , Pic0
  Case Is = 1
  Showpic 0 , 32 , Pic1
  Case Is = 2
  Showpic 0 , 32 , Pic2
  Case Is = 3
  Showpic 0 , 32 , Pic3
  Case Is = 1
  Showpic 0 , 32 , Pic1
  Case Is = 4
  Showpic 0 , 32 , Pic4
  Case Is = 5
  Showpic 0 , 32 , Pic5
  Case Is = 6
  Showpic 0 , 32 , Pic6
  Case Is = 7
  Showpic 0 , 32 , Pic7
  Case Is = 8
  Showpic 0 , 32 , Pic8
  Case Is = 9
  Showpic 0 , 32 , Pic9
  End Select
O2 = Mid(hossein , 2 , 1)
Lcd_data = Val(o2)
Select Case Lcd_data
  Case Is = 0
  Showpic 24 , 32 , Pic0
  Case Is = 1
  Showpic 24 , 32 , Pic1
  Case Is = 2
  Showpic 24 , 32 , Pic2
  Case Is = 3
  Showpic 24 , 32 , Pic3
  Case Is = 1
  Showpic 24 , 32 , Pic1
  Case Is = 4
  Showpic 24 , 32 , Pic4
  Case Is = 5
  Showpic 24 , 32 , Pic5
  Case Is = 6
  Showpic 24 , 32 , Pic6
  Case Is = 7
  Showpic 24 , 32 , Pic7
  Case Is = 8
  Showpic 24 , 32 , Pic8
  Case Is = 9
  Showpic 24 , 32 , Pic9
  End Select
Showpic 48 , 32 , Pic14
'-----------------------------------------------------------------------------------------------------------
End Select
Waitms 100
Goto Program_start1
'END OF LEVEL5-----------------------------------------------------------------------------------------------

  Pic0:
  $bgf "0.bgf"
```

```
Pic1:
$bgf "1.bgf"
Pic2:
$bgf "2.bgf"
Pic3:
$bgf "3.bgf"
Pic4:
$bgf "4.bgf"
Pic5:
$bgf "5.bgf"
Pic6:
$bgf "6.bgf"
Pic7:
$bgf "7.bgf"
Pic8:
$bgf "8.bgf"
Pic9:
$bgf "9.bgf"
Pic10:
$bgf "CADR.bgf"
Pic11:
$bgf "TERMOMETER1.bgf"
Pic12:
$bgf "line.bgf"
Pic14:
$bgf "CANTIGRAD.bgf"
Pic15:
$bgf "CLS_TEMP.bgf"
'-----------------------------------------------------------------------------------------------------------------
```

دستورات جدید :

در برنامه فوق از دستور EXIT استفاده شده که فرم کلی آن به صورت زیر است.

```
EXIT FOR
EXIT DO
EXIT WHILE
EXIT SUB
```

با این دستور می توانید از یک حلقه یا یک زیر برنامه خارج شده و ادامه برنامه را بعد از حلقه یا زیر برنامه ادامه دهید . در برنامه فوق از دستورSTR استفاده شده که توسط این دستور می توان یک متغیر عددی را به یک متغیر رشته ای تبدیل کرد .

```
S=STR(VAR)
```

که در دستور بالا S یک متغیر رشته ای (STRING) و VAR یک متغیر عددی می باشد . در برنامه فوق از دستور LEN استفاده شده که این دستور برای برگرداندن تعداد کاراکتر های یک متغیر رشته ای مورد استفاده قرار می گیرد . و فرم کلی آن به صورت زیر است .

```
VAR=LEN(S)
```

S یک متغیر از نوع STRING بوده که تعداد کاراکتر های آن در متغیر عددی VAR قرار می گیرد . در برنامه فوق از دستور MID استفاده شده که با این دستور می توان قسمتی از یک رشته را برداشت و در یک متغیر رشته ای دیگر قرار داد . فرم کلی دستور MID به صورت زیر می باشد .

VAR =MID (S1,ST,L)

قسمتی از رشته S1 با شروع از کاراکتر ST ام از چپ به راست و طول L برداشته شده و در متغیر رشته ای VAR قرار داده می شود .

در برنامه فوق از دستور VAL استفاده شده است که این دستور برای تبدیل یک رشته به یک متغیر عددی کاربرد دارد . و فرم کلی آن به صورت زیر است .

VAR=VAL(S)

متغیر رشته ای S تبدیل به متغیر عددی شده و در متغیر عددی VAR قرار می گیرد .

تشریح نحوه عملکرد برنامه :

در قسمت LEVEL1 ابتدا تصویر TERMOMETER.BGF بر روی LCD گرافیکی نمایش داده شده سپس PIN مربوط به کلید START BUTTON چک می شود و تا زمانی که این کلید فشار داده نشده تصویر نمایش داده شده بر روی LCD ثابت خواهد بود پس از زدن این کلید اجرای برنامه وارد قسمت LEVEL2 می شود در این قسمت ابتدا توسط دستور ADC,START ADC را روشن می کنیم سپس توسط دستور TEMP=GETADC(0) ولتاژ آنالوگ ورودی به کانال شماره صفر ADC را تبدیل به مقدار دیجیتال کرده و در متغیر TEMP که از نوع WORD می باشد قرار می دهیم . ورودی ADC(0) در واقع همان ولتاژ خروجی سنسورLM35 می باشد . سپس محتوای متغیر TEMP را بر 2 تقسیم کرده و دوباره در متغیر TEMP قرار می دهیم . هر بار پس از اندازه گیری دما اگر دمای اندازه گیری شده با مقدار قبلی متفاوت باشد آن را در متغیر SAVE قرار داده سپس بر روی LCD نمایش می دهیم بنابر این پس از این که مقدار دما را در متغیر TEMP قرار دادیم ابتدا با استفاده از دستور

```
IF TEMP=SAVE THEN GOTO PROGRAM_START1
```

بررسی می کنیم که آیا دمای گرفته شده با نمونه قبلی نمایش داده شده بر روی LCD متفاوت است یا نه ، اگر متفاوت نبود تصویر نمایش داده شده تغییری نمی کند و اگر متفاوت بود مقدار جدید بر روی LCD نمایش داده می شود ، پس از دستور SAVE=TEMP برنامه وارد قسمت LEVEL3 می شود . در این قسمت ابتدا تصویر CADR.BGF بر روی LCD قرار می گیرد سپس با استفاده از دستورات

```
FOR X=8 TO 110 STEP1
SHOWPIC X,8,PIC15
NEXT X
```

با توجه به این که PIC15 نام برچسب مر بوط به تصویر CLS_TEMP.BGF می باشد که آن نیز یک تصویر سفید با تعداد پیکسل 8*16 می باشد با نمایش پشت سر هم تصویر سفید در داخل کادر قسمت داخلی کادر پاک می شود .

سپس با نمایش پشت سر هم این تصویر یک خط عمودی به اندازه داخل کادر به صورت متناسب با دما ، دما به صورت نموداری در داخل کادر نمایش داده می شود .

برای نمایش عددی دما بر روی LCD ابتدا بایستی بررسی کنیم که دما یک رقمی است یا دو رقمی که این کار در قسمت LEVEL4 انجام می گیرد ، بدین صورت که ابتدا محتوای متغیر TEMP تبدیل به متغیر رشته ای شده و در متغیر رشته ای HOSSEIN قرار می گیرد سپس تعداد کاراکتر های متغیر رشته ای HOSSEIN در متغیر عددی A قرار می گیرد . سپس در قسمت LEVEL5 با استفاده از دستور SELECT CASE A دستورات لازم برای یک رقمی یا دو رقمی بودن دما اجرا می شود . در قسمت CASE IS =1 دستورات مربوط به یک رقمی بودن دما نوشته شده است که در این صورت ابتدا متغیر رشته ای HOSSEIN تبدیل به متغیر عددی شده و در متغیر عددی LCD_DATA قرار می گیرد سپس تصویر مربوط به عددی که در LCD_DATA قرار دارد در مختصات (0,32) روی LCD نمایش داده می شود . پس از آن تصویر مر بوط به درجه سانتی گراد با نام CANTIGRAD.BGF در مختصات (24,32) یعنی بعد از عدد مربوط به دما بر روی LCD گرافیکی قرار می گیرد .

در قسمت CASE IS=2 دستورات مربوط به دو رقمی بودن دما نوشته شده است . که در این صورت ابتدا اولین کاراکتر از سمت چپ جدا شده وتبدیل به متغیر عددی می شود و در متغیر LCD_DATA قرار می گیرد و تصویر مر بوط به عددی که در متغیر LCD_DATA قرار دارد در مختصات (0,32) از LCD گرافیکی روی آن قرار می گیرد .

سپس رقم دوم از سمت چپ به راست جدا شده و تبدیل به متغیر عددی می شود و در متغیر عددی LCD_DATA قرار می گیرد سپس تصویر عدد متناسب با آن در مختصات (24,32) از LCD گرافیکی قرار می گیرد و تصویر مر بوط به درجه سانتی گراد (CANTIGRAD.BGF) در ادامه یعنی پس از رقم دوم که رقم یکان می باشد در مختصات (48,32) بر روی LCD نمایش داده می شود . سپس بعد از 100 میلی ثانیه تاخیر اجرای برنامه به لیبل PROGRAM_START منتقل می شود .

تایمر/کانترها (TIMER/COUNTERS)

میکروکنترلرهای استفاده شده در این کتاب نهایتا دارای 3 تایمر / کانتر هستند . که هر سه نوع در میکروکنترلر نمونه ATMEGA32 وجود دارد .

تایمر کانتر صفر یک تایمر/کانتر 8 بیتی است که حداکثر می تواند تا مقدار $FF ، (255) را شمارش کند این تایمر /کانتر می تواند کلاک خود را از سیستم یا تقسیمی از کلاک سیستم تامین کند که در این صورت عملکرد آن به صورت تایمر خواهد بود ، همچنین می تواند کلاک خود را از پایه خروجی TO دریافت کند که دراین صورت عملکرد آن به صورت کانتر خواهد بود . زمانی که تایمر/کانتر صفر در مد کانتر عمل کند سیگنال وارد شده به پایه TO با فرکانس CPU سنکرون (SYINCHRONIZE) می شود ، بنابراین برای اطمینان از نمونه برداری مناسب ، بایستی زمان بین دو کلاک خروجی حداقل برابر یک دوره تناسب کلاک CPU باشد . برای مثال اگر فرکانس کاری میکرو برابر با یک مگا هرتز باشد مدت زمان یک دوره تناوب CPU بدین صورت خواهد بود .

T=1/F=1/1MHZ=1US

یعنی حداقل زمان بین دو پالس تریگر ورودی بایستی برابر 1US باشد .

پیکره بندی تایمر/کانتر صفر به صورت تایمر در محیط BASCOM

CONFIG TIMER0=TIMER,PRESCALE=1/8/64//256/1024

در این حالت تایمر/کانتر صفر در مد تایمر با فرکانس سیستم تقسیم بر 1024,256,64,8,1 کار می کند ، با دستور START TIMER شمارش تایمر شروع شده و با دستور STOP TIMER شمارش آن متوقف می شود محتوای تایمر/کانترصفررامی توان با دستورVAR=TIMER0خواند و با دستور TIMER0=VALUE می توان مقدار اولیه تایمر صفر را تعیین نمود .

تایمر 0 پس از شمردن (255+1) شماره پرچم سرریزی خود را با نام OVF0 یک می کند در این شرایط در صورتی که وقفه سراسری با دستور ENABLE INTERUPTS و وقفه سرریزی با دستور ENABLE OVF0 فعال شده باشند می توان در زمان سرریزی تایمر با دستور ON OVF0 به لیبل ISR مر بوطه پرش کرده و آن را اجرا نمود . برگشت از وقفه سرریزی نیز با دستور RETURN انجام می گیرد . برای مثال برای ساختن مدت زمان 1 ثانیه توسط تایمر صفر بایستی به صورت زیر عمل کنیم ، در برنامه زیر مقدار منطقی PORTA.0 در هر یک ثانیه معکوس می شود .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Count1 As Word
Config Pina.0 = Output
Config Timer0 = Timer , Prescale = 8
Enable Interrupts
Enable Timer0
On Ovf0 Tim0_isr
```

```
Timer0 = 56
Start Timer0
Porta.0 = 1
'-----------------------------------------------------------------------------------------------------------
Do
'your program goes here
Loop
'-----------------------------------------------------------------------------------------------------------
Tim0_isr:
Stop Timer0
Incr Count1
If Count1 = 5000 Then
Toggle Porta.0
Count1 = 0
End If
Timer0 = 56
Start Timer0
Return
'-----------------------------------------------------------------------------------------------------------
```

نحوه ساختن مدت زمان یک ثانیه در برنامه فوق به صورت زیر است .

فرکانس کاری تایمر از فرمول f=8MHz/8=1MS بدست می آید که 8MHz فرکانس کاری میکرو و 8 نیز PRESCALE تایمر می باشد . پس مدت زمانی که طول می کشد تا تایمر یک شماره صعود کند بدین صورت خواهد بود .

T=1/F=1/1MHz

با 56 قرار دادن مقدار اولیه تایمر/کانتر، تایمر/کانترپس از 200 شماره صعود سرریز خواهد شد ، مدت زمان سرریز شدن تایمر برابر خواهد بود با

200*1US=0.0002S

پس برای بدست آوردن زمان یک ثانیه تایمر باید 5000 بار سرریز شود.

1S/0.0002S=5000

مدت زمان یک ثانیه طراحی شده در مثال قبل چندان هم دقیق نیست به خاطر این که اگر خوشبینانه قضاوت کرده و تعداد دستورات اجرا شده هنگام سریزی تایمر 0 را 5 دستور در نظر بگیریم با توجه به این که در میکروکنترلر های AVR اکثر دستورات در یک سیکل ساعت انجام می شوند و مدت زمان اجرای یک دستور و همچنین واکشی دستور بعدی برابر با

T=1 / (فرکانس کاری میکرو) = 1 /8000000 = 0.125us

می باشد ، مدت زمان اجرای دستورات سریزی برابر با

0.125us * 5 = 0.625us

خواهد بود ، همچنین با توجه به این که تایمر 0 ، 5000 بار سرریز می شود مدت زمان کل برای اجرای دستورات سرریزی برابر با

5000*0.625us=3.12ms

خواهد بود ، همچنین دستور IF نیز در هر 5000 بار سرریزی یک بار اجرا می شود بنابراین مدت زمان 3*0.125us=0.375us هم بایستی به کل زمان اجرای دستورات سرریزی اضافه شود . بنابراین مدت زمان ایجاد شده توسط تایمر برابر با 1.00312s خواهد بود .

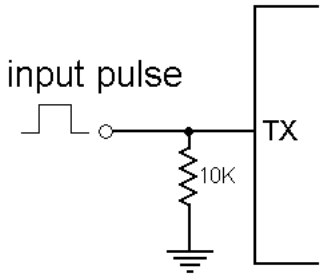
با توجه به این موضوع برای ایجاد زمان های واقعی بهتر است از مد آسنکرون تایمرهای 8 بیتی ، 0 و2 در حالتی که کلاک خود را از کریستال ساعت 32.768KHz دریافت می کنند استفاده کنیم ، نحوه استفاده از این مد در پروژه ساعت فصل AVR در الکترونیک نوری توضیح داده شده است .

پیکره بندی تایمر/کانتر صفر به صورت کانتر در محیط BASCOM

CONFIG TIMER0 =COUNTER,EDGE=RISING/FALLING

وقتی از تایمر/کانتر به عنوان کانتر یا شمارنده استفاده می شود می توان شمارش آن را در لبه بالا رونده یا پایین رونده پالس ورودی فعال کرد . با انتخاب COUNTER EDGE=RISING با اعمال یک لبه بالا رونده به پایه T0 یک شماره صعود خواهد کرد . و با انتخاب EDGE=FALLING با اعمال یک لبه پایین رونده به پایه T0 , COUNTER یک شماره صعود خواهد کرد .

COUNTER پس از شمردن (255+1) پالس پرچم سرریزی خود را با نام OVF0 یک می کند در این شرایط در صورتی که وقفه سراسری با دستور ENABLE INTERRUPTS و وقفه سرریزی با دستور ENABLE OVF0 فعال شده باشند می توان در زمان سرریزی COUNTER با دستور ON OVF0 LABLE به لیبل ISR مربوط پرش کرده و آن را اجرا نمود . برگشت از سرریزی باز با دستور RETURN انجام می گیرد . توجه داشته باشید هنگام استفاده از COUNTER پالس ورودی بایستی مانند شکل 2-61 به پایه مربوطه اعمال شود .



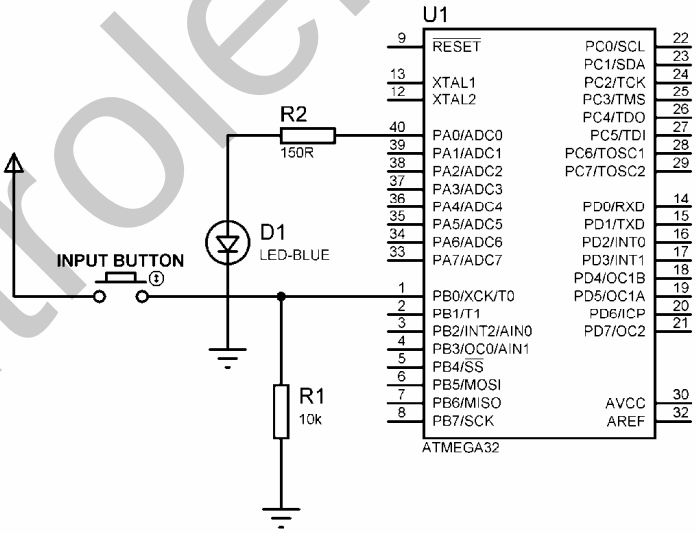
شکل 2-61 نحوه اعمال پالس به پایه ورودی کانتر

در این حالت زمانی که هیچ پالسی اعمال نشود پایه ورودی COUNTER زمین خواهد بود .

برای مثال برنامه ای می نوسیم که پس از اعمال 5 پالس تریگر به ورودی COUNTER ، (در لبه بالا رونده) مقدار منطقی PORTA.0 معکوس شود .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Count1 As Word
Config Pina.0 = Output
Config Timer0 = Counter , Edge = Rising
Enable Interrupts
Enable Counter0
On Ovf0 Counter0_isr
Counter0 = 251
Reset Porta.0
'-----------------------------------------------------------------------------------------------------------
Do
'your program goes here
Loop
'-----------------------------------------------------------------------------------------------------------
Counter0_isr:
Toggle Porta.0
Counter0 = 251
Return
'-----------------------------------------------------------------------------------------------------------
```

سخت افزارمربوط به برنامه بالا در شکل 2-61 نشان داده شده است . در این سخت افزار برای اعمال پالس تریگر به ورودی COUNTER از یک کلید فشاری استفاده شده است .



شکل 2-61 شماتیک مربوط به استفاده از تایمر 0 به عنوان شمارنده پالس تریگر ورودی

دستورات جدید :

در برنامه فوق از دستور TOGGLE PORTX.Y استفاده شده است که این دستور مقدار منطقی موجود بر روی یک پین را معکوس می کند .

با توجه به این که COUNTER پس از اعمال 256 پالس به پایه مربوطه سرریزمی شود مقدار اولیه آن 251 قرار داده شده است تا پس از اعمال 5 پالس تریگر سرریز شده و PORTA.0 در ISR مربوط به COUNTER0 معکوس شود . توجه داشته باشید اگر مقدار اولیه برای TIMER یا COUNTER تعریف نشود کامپایلر مقدار اولیه را صفر در نظر می گیرد .

نحوه پیکره بندی تایمر/کانترهای 1و2 نیز در حالت تایمر و حالت COUNTER مانند تایمر/کانتر 0 می باشد. تایمر/کانتر 2 نیز مانند تایمر/کانتر 0 حداکثر می تواند تا مقدار $FF (255) را شمارش کند و پس از صعود (255+1) شماره سریزمی شود .

با توجه به این که تایمر کانتر 2 در میکروکنترلر های معرفی شده دارای پایه ورودی T2 نمی باشد نمی توان از آن به عنوان کانتر استفاده کرد تایمر/کانتر 1نیز حداکثر می تواند تا مقدار $ffff (65535) را شمارش کند و پس از صعود (65535+1) شماره سرریزمی شود .

طراحی پروژه فرکانس متر دیجیتال

یاد آور می شوم که این پروژه جهت کار با مد سنکرون تایمرها طراحی شده و برای ساختن مدت زمان یک ثانیه بهتر است از مد آسنکرون تایمر صفر یا دو استفاده شود .

این پروژه فرکانس پالس داده شده به پایه T1 را اندازه گیری کرده و روی LCD نشان دهد . در این پروژه از یک LCD (16*2) ، یک SPEAKER یا BUZZER به عنوان خروجی و از دو کلید K1 وOk و ورودی کانتر 1 به عنوان ورودی استفاده شده است .

نحوه عملکرد پروژه به صورت زیر می باشد که ابتدا عبارت زیر بر روی LCD نوشته می شود .

PLEASE ENTER
FREQUENCY RANGE

پس از 4 ثانیه تاخیر رنج فرکانسی به صورت زیر از کاربر درخواست می شود .

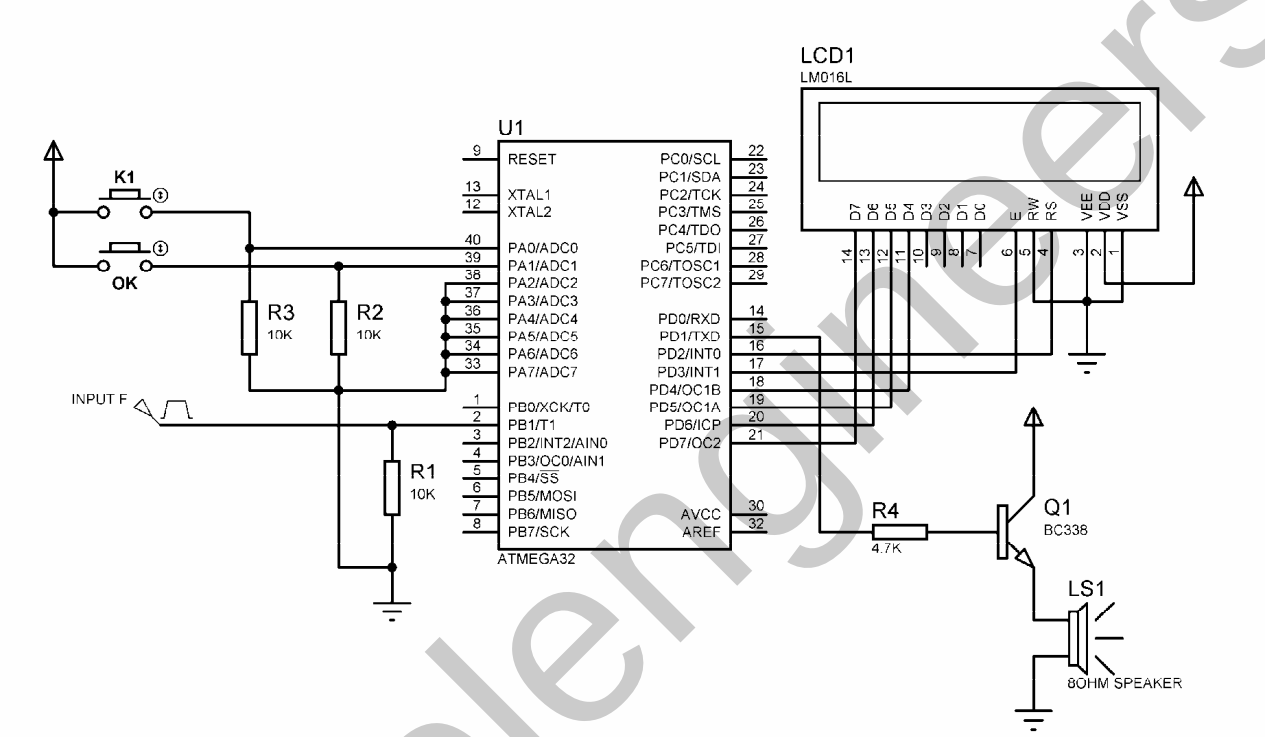
FREQUENCY RANGE
IS Hz

کیلد K1 حساس به لبه بالا رونده بوده و با هر بار فشار آن رنج فرکانس به صورت Hz وKHz و MHz تغییر می کند یعنی ابتدا رنج فرکانسی برابر Hz می باشد با یک بار فشار K1 رنج فرکانس برابر KHz و با فشار مجدد آن برابر با MHz خواهد شد و اگر یک باردیگر نیز K1 را فشار دهیم رنج فرکانسی برابر با Hz می شود برای

انتخاب رنج فرکانسی مورد نظر بایستی روی دکمه OK کلیک کنید . پس از انتخاب رنج فرکانسی سیگنال ورودی به صورت زیر بر روی LCD نمایش داده خواهد شد .

FREQUENCY IS
100.0Hz

سخت افزار طراحی شده برای پروژه در شکل 63-2 نشان داده شده است .



شکل 63-2 شماتیک طراحی شده برای فرکانس متر دیجیتال

برنامه نوشته شده برای پروژه به صورت زیر می باشد .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Count1 As Word
Config Timer1 = Counter , Edge = Rising
Config Timer2 = Timer , Prescale = 8
Config Porta = Input
Config Pind.1 = Output
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.4 , Db5 = Pind.5 , Db6 = Pind.6 , Db7 = Pind.7_
        , E = Pind.3 , Rs = Pind.2
Dim F As Single , Count As Word , A As Long , Recive_data As Byte
Dim Hossein As Byte
'-----------------------------------------------
Enable Counter1
On Ovf1 Counter1_isr
'-----------------------------------------------
Enable Timer2
```

```
On Ovf2 Timer2_isr
'START OF LEVEL1-----------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "PLEASE ENTER "
Lowerline
Lcd "FREQUENY RANGE"
Wait 4
'END OF LEVEL1-------------------------------------------------------------------------------------------------
'START OF LEVEL2-----------------------------------------------------------------------------------------------
Cls : Home
Lcd "FREQUENY RANGE"
Lowerline
Lcd "IS Hz"
H1:
Recive_data = Pina
If Recive_data = &B00000001 Then
  Incr Hossein
  If Hossein > 2 Then Hossein = 0
  Select Case Hossein
    Case Is = 0
    Cls : Home
    Lcd "FREQUENY RANGE"
    Lowerline
    Lcd "IS Hz"

    Case Is = 1
    Cls : Home
    Lcd "FREQUENY RANGE"
    Lowerline
    Lcd "IS KHz"

    Case Is = 2
    Cls : Home
    Lcd "FREQUENY RANGE"
    Lowerline
    Lcd "IS MHz"
  End Select
  Sound Portd.1 , 100 , 80
'-----------------------------------------------
  H2:
  Recive_data = Pina
  If Recive_data <> 0 Then Goto H2
'-----------------------------------------------
End If
If Recive_data = &B00000010 Then
Sound Portd.1 , 100 , 80
Goto H3
End If
Waitms 100
Goto H1
'END OF LEVEL2-------------------------------------------------------------------------------------------------
'START OF LEVEL3-----------------------------------------------------------------------------------------------
H3:
Enable Interrupts
Counter1 = 0
Timer2 = 56
Start Timer2
'-----------------------------------------------------------------------------------------------------------
Do
Loop
```

```
'END OF LEVEL3-----------------------------------------------------------------------------------------
Timer2_isr:
Stop Timer2
Incr Count
If Count = 5000 Then
Cursor Off
Cls : Home
F = A * 65536
F = F + Counter1
Select Case Hossein
  Case Is = 0
  Cls : Home
  Lcd "FREQUENCY IS"
  Lowerline
  Lcd F ; "Hz"

  Case Is = 1
  F = F / 1000
  Cls : Home
  Lcd "FREQUENCY IS"
  Lowerline
  Lcd F ; "KHz"

  Case Is = 2
  F = F / 1000000
  Cls : Home
  Lcd "FREQUENCY IS"
  Lowerline
  Lcd F ; "MHz"
End Select

A = 0 : Counter1 = 0 : Count = 0
End If
Timer2 = 56
Start Timer2
Return
'-----------------------------------------------------------------------------------------
Counter1_isr:
Incr A
Counter1 = 0
Return
'-----------------------------------------------------------------------------------------
```

در این پروژه برای ساختن مدت زمان 1S ، از تایمر 2 استفاده شده است نحوه ساختن مدت زمان یک ثانیه به صورت زیر است .

(فرکانس کاری تایمر) F=8MHz/8=1MHz

(مدت زمان صعود یک شماره) T=1/F=1/1MHz=1MS

با 56 قرار دادن مقدار اولیه تایمر/کانتر پس از 200 شماره صعود سرریز خواهد شد .

(مدت زمان سرریز شده تایمر) =200*1MS=0.0002S

پس برای بدست آوردن زمان یک ثانیه تایمر باید 5000 بار سرریزشود .

1S/0.0002=5000

کانتر 1 نیز پس از شمارش 65536 پالس سرریزی شود اگر قبل از سرریز شدن تایمر /کانتر1 سرریز شود یک واحد به متغیر A اضافه می شود و کانتر1 نیز دوباره از 0 شروع به شمردن پالس ورودی می کند .

و در آخر پس از سرریزشدن تایمر 2 تعداد پالس شمارش شده از فرمول

F=65536*A
F=F+COUNTER 1

بدست می آید . و با توجه به رنج فرکانس انتخاب شده بر روی LCD نمایش داده می شود . رنج فرکانسی توسط کلید های OK,K1 تعیین می شود که کلید K1 حساس به لبه بالا رونده می باشد .

پیکره بندی تایمر/کانتر در مد مقایسه ای (COMPARE)

تایمر/کانتر های 2,1,0 می توانند در مد COMPARE پیکره بندی شوند .

پیکره بندی تایمر/کانتر۱ در مد (COMPARE)

تایمر 1 و مد مقایسه ای :

CONFIG TIMER1=TIMER,COMPARE A = CLEAR/SET/TOGGLE/ DISCONNECT
, COMPAREB=CLEAR/SET/TOGGLE/DISCONNECT,PRESCALE=1/8/64/256/1024

طبق پیکره بندی فوق تایمر/کانتر1 در مد تایمر استفاده شده و فرکانس کاری تایمر با تقسیم کلاک سیستم بر 1، 8 ، 64 ، 256، 1024 تامین می شود ، در هر لحظه محتوای دو رجیستر مقایسه ای B,A با محتوای تایمر مقایسه می شود و هنگام تطابق مقایسه وضعیت منطقی پایه های OC1A و OC1B بنا به تعریف تغییر می کند .

نکته مهم این که پایه های OC1A و OC1B بایستی به صورت خروجی تعریف شده باشند برای مثال در ATMEGA32، (OC1B)PIND.4 ، (OC1A)PIND.5 بایستی خروجی تعریف شود .

COMPARE A=CLEAR/SET/TOGGLE/DISCONNECT :

در زمان تطابق مقایسه (COMPARE MATCH) پایه خروجی OC1A می تواند یک (SET) ،صفر (CLEAR) ، معکوس (TOGGLE) و یا ارتباط پایه با کانتر قطع(DISCONNECT) شود .

COMPARE B =CLEAR/SET/TOGGLE/DISCONNECT :

در زمان تطابق مقایسه پایه خروجی OC1B می تواند یک (SET) ،صفر (CLEAR) ، معکوس (TOGGLE) و یا ارتباط پایه با کانتر قطع (DISCONNECT) شود .

محتوای رجیستر مقایسه ای A یاB را می توان با دستور COMPARE1A/B=VAR تعیین کرد و همچنین برای خواندن محتوای رجیستر های COMPARE A/B می توان از دستور VAR=COMPAREA/B استفاده کرد که در هر دو دستور VAR یک متغیر عددی از نوع WORD می باشد .

نحوه پیکره بندی کانتر1 و مد مقایسه ای (COMPARE) :

CONFIG TIMER1=COUNTER,EDGE=RISING/FALLING,COMPARE A =CLEAR/SET/TOGGLE/DISCONNECT,COMPARE B =CLEAR/SET/TOGGLE/DISCONNECT,PRESCALE=1/8/64/256/1024

طبق پیکره بندی فوق تایمر/کانتر در حالت کانتر استفاده شده است و کلاک خود از پایه T1 با لبه بالا رونده (RISING) یا لبه پایین رونده (FALLING) دریافت میکند . در هر لحظه محتوای دو رجیستر مقایسه ای A,B با محتوای کانتر مقایسه می شود و هنگام تطابق مقایسه وضعیت منطقی پایه های OC1A,OC1B بنا به تعریف می تواند تغییر کند .

محتوای رجیستر مقایسه ای A یا B را می توان با دستور COMPARE1A/B=VAR تعیین نمود و هم چنین برای خواندن محتوای رجیستر های COMPAREA/B می توان از دو دستور VAR=COMPARE A/B استفاده کرد که در هر دو دستور VAR یک متغیر عددی از نوع WORD می باشد.

نکته مهم این که پایه های OC1A و OC1B بایستی به صورت خروجی تعریف شوند برای مثال در ATMEGA32 ، (OC1B)PIND.4 ، (OC1A)PIND.5 بایستی خروجی تعریف شده باشند .

طرز کار با وقفه تطابق مقایسه (COMPARE MATCH) :

هنگام تطابق مقایسه بین محتوای رجیستر های مقایسه ای B,A و محتوای تایمر/کانتر یک ، علاوه بر تغییر وضعیت منطقی پایه های OC1A و OC1B پرچم وقفه تطابق مقایسه نیز یک می شود . که برای هر یک از رجیستر های مقایسه ای متفاوت است و پرچم تطابق مقایسه رجیستر A ، OC1A و پرچم تطابق مقایسه رجیستر B ، OC1B نام دارد.

نکته مهم این که برای مثال اگر محتوای رجیستر مقایسه ای برابر با 100 باشد زمانی که محتوای تایمر کانتر برابر با 100 شود وضعیت پایه خروجی OC1X تغییر می کند و زمانی که محتوای تایمر/کانتر برابر با 101 شود وقفه COMPARE MATCH اتفاق می افتد . در ضمن در تایمر/کانتر یک پس از یک شدن پرچم تطابق مقایسه محتوای تایمر/کانتر صفر می شود که برای رفع این مشکل می توانید تایمر/کانتر را در ISR ، COMPARE MATCH مقدار دهی کنید . برای اجرا شدن وقفه تطابق مقایسه A/B بایستی وقفه سراسری توسط دستور ENABLE INTERRUPTS و وقفه تطابق مقایسه توسط دستور ENABLE OC1A ، ENABLE OC1B فعال شده باشد . برای پرش به ISR تطابق مقایسه می توانید از دستور ON OC1A/B LABLE در ابتدای برنامه استفاده کنید . به عنوان مثال در مدار میکروکنترلری شکل 64-2 زیر از تایمر/کانتر یک به عنوان کانتر و در مد مقایسه ای با وقفه COMPARE MATCH استفاده شده است و در برنامه نوشته شده برای میکرو کنترلر محتوای رجیستر مقایسه ای COMPARE1A برابر با 2 و محتوای رجیستر COMPARE1B

برابر با 4 در نظر گرفته شده است و مقدار اولیه COUNTER نیز صفر می باشد وقفه مربوط به هر دو رجیستر مقایسه ای فعال شده است ، نحوه عملکرد مدار بدین صورت می باشد که ابتدا عبارت زیر بر روی LCD نوشته می شود .

COUNTER IS 0

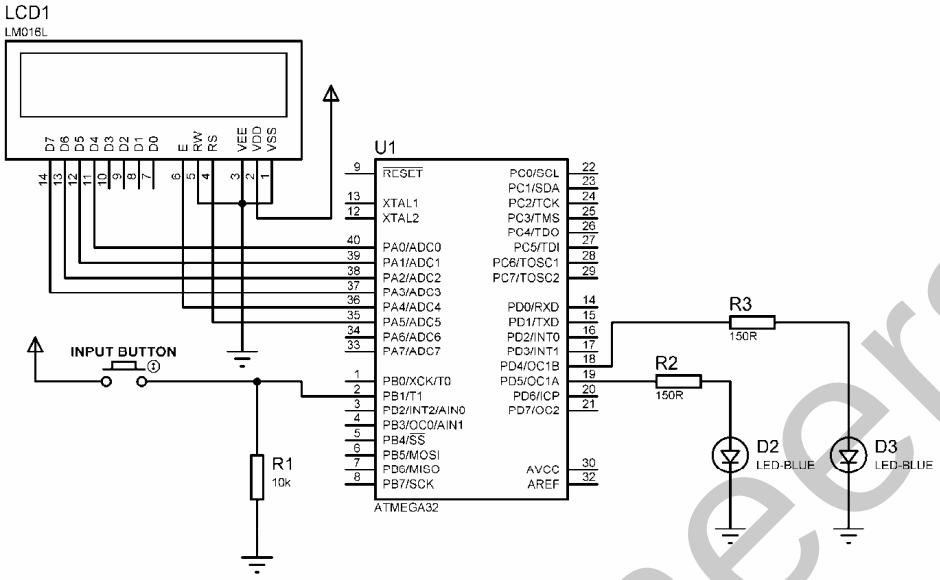
با هر بار فشار کلید INPUT BUTTON یک پالس تریگر به ورودی کانتر 1 اعمال شده و محتوای آن یک شماره افزایش پیدا می کند و محتوای جدید کانتر بر روی LCD نمایش داده می شود .
پس از اعمال دو پالس تریگر به ورودی کانتر وضعیت OC1A معکوس شده و پس از اعمال پالس سوم ISR مربوط به وقفه COMPARE MATCH اجرا می شود و روی LCD عبارت زیر نوشته می شود .

COMPARE A ISR
COUNTER IS 3

با اعمال پالس چهارم وضعیت پایه OC1B معکوس شده و با اعمال پالس پنجم ISR مربوط به وقفه COMPARE 1 B اجرا می شود و عبارت زیر بر روی LCD نوشته می شود .

COMPARE B ISR
COUNTER IS 5

سپس محتوای COUNTER برابر با صفر شده و مراحل گفته شده دو باره اجرا می شوند. عملکرد کلی مدار به این صورت می باشد که پس از اعمال2 پالس تریگر به ورودی کانتر توسط INPUT BUTTON وضعیت منطقی پایه OC1A معکوس می شود و پس از اعمال دو پالس دیگر وضعیت منطقی پایه OC1B معکوس می شود و پس از اعمال پالس پنجم محتوای COUNTER صفر شده و مراحل گفته شده تکرار می شود .
سخت افزار مداردر شکل 64-2 نشان داده شده است .



شکل 2-64 شماتیک طراحی شده برای استفاده از مد COMPARE در کانتر 1

بر نامه نوشته شده برای سخت افزار فوق به صورت زیر می باشد .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Timer1 = Counter , Edge = Rising , Compare A = Toggle _
      , Compare B = Toggle
Dim F As Single , Count As Word , A As Long , Recive_data As Byte
Dim Hossein As Byte
Config Pind.4 = Output
Config Pind.5 = Output
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3_
   , E = Pina.4 , Rs = Pina.5
'-----------------------------------------------------------------------------------------------
Enable Interrupts
Counter1 = 0
'-----------------------------------------------------------------------------------------------
Enable Oc1a
On Oc1a Compparematch_a
Compare1a = 2
'-----------------------------------------------------------------------------------------------
Enable Oc1b
On Oc1b Compparematch_b
Compare1b = 4
'-----------------------------------------------------------------------------------------------
Cursor Off : Cls
Do
Home
Lcd "COUNTER IS"
Lowerline
Lcd Counter1
Waitms 100
Loop
'-----------------------------------------------------------------------------------------------
Compparematch_a:
Stop Counter1
```

```
Cls : Home
Lcd "COPARE A ISR"
Counter1 = 3
Lowerline
Lcd "COUNTER IS " ; Counter1
Wait 4 : Cls
Start Counter1
Return
'-----------------------------------------------------------------------------------------------------------
Compparematch_b:
Stop Counter1
Cls : Home
Lcd "COPARE B ISR"
Lowerline
Counter1 = 5
Lcd "COUNTER IS " ; Counter1
Wait 4
Counter1 = 0 : Cls
Start Counter1
Return
'-----------------------------------------------------------------------------------------------------------
```

پیکره بندی تایمر/کانتر دو در مد مقایسه ای (COMPARE)

تایمر دو و مد مقایسه ای :

CONFIG TIMER=TIMER,COMPARE=CLEAR/SET/TOGGLE/DISCONNECT ,PRESCALE=1/8/32/64/128/256/1024

توسط این پیکره بندی تایمر/کانتر 2 در حالت تایمر استفاده شده است و می تواند فرکانس کلاک خود را از فرکانس سیستم بخش بر 1، 8 ، 32 ، 64 ، 128 ، 1024 تامین کند .

تایمر کانتر های 2 و 0 بر خلاف تایمر/کانتر 1 فقط یک رجیستر مقایسه ای دارند ، در تایمر 2 در هر لحظه محتوای رجیستر مقایسه ای OCR2 با محتوای رجیستر تایمر مقایسه می شود و هنگام تطابق مقایسه وضعیت پایه OC2 بنا به تعریف تغییر خواهد کرد .

نکته مهم این که پایه OC2 بایستی خروجی تعریف شده باشد ، برای مثال در ATMEGA32، PIND.7(OC2) بایستی خروجی تعریف شود .

COMPARE=CLEAR/SET/TOGGLE/DISCONNECT :

در زمان تطابق مقایسه پایه خروجی OC2 می تواند یک (SET) ، صفر (CLEAR) ، معکوس (TOGGLE) و یا ارتباط پایه با تایمر قطع (DISCONNECT) شود. محتوای رجیستر مقایسه ای OCR2 با دستور OCR2=VAR تعیین می شود که در آن VAR می تواند یک ثابت یا یک متغیر عددی از نوع BYTE باشد . محتوای رجیستر مقایسه ای را می توان با دستور VAR=COMPARE خواند که VAR

متغیر نوع BYTE می باشد . به دلیل این که تایمر/کانتر 2 یک تایمر/کانتر 8 بیتی است و حداکثر می تواند مقدار 255 را داشته باشد محتوای رجیستر های مقایسه ای نیز نباید بیشتر از 255 باشد .

به دلیل این که در میکرو کنترلر های معرفی شده از تایمر/کانتر 2 نمی توان به عنوان COUNTER استفاده کرد از پیکره بندی تایمر/کانتر 2 در مد کانتر صرف نظر می شود .

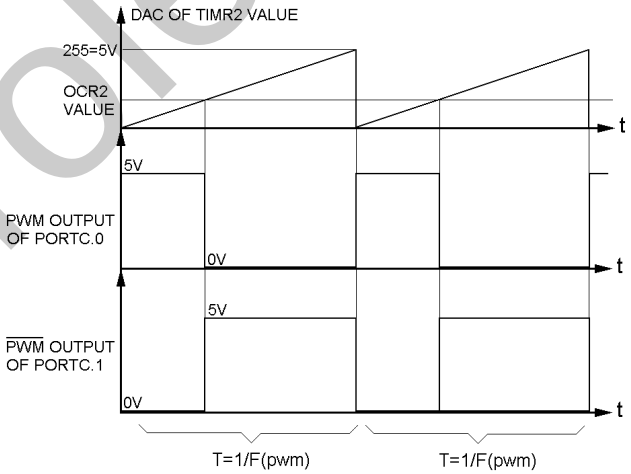
طرز کار با وقفه تطابق مقایسه (COMPARE MATCH) :

هنگام تطابق مقایسه بین محتوای رجیستر تایمر 2 و رجیستر مقایسه ای OCR2 علاوه بر تغییر وضعیت پایه OC2 ، پرچم وقفه COMPARE MATCH نیز یک می شود برای اجرای وقفه تطابق مقایسه بایستی وقفه سراسری توسط دستور ENABLE INTERRUPTS و وقفه تطابق مقایسه توسط دستور ENABLE OC2 فعال شده باشند. برای پرش به ISR تطابق مقایسه می توانید از دستور ON OC2 LABLE استفاده کنید .نکته مهم این که در تایمر/کانتر 2 پس از یک شدن پرچم، تطابق مقایسه محتوای تایمر/کانتر صفر نمی شود .

تولید سیگنال PWM با استفاده از مد مقایسه ای تایمر 2

حالا می خواهیم پروژه ای طراحی کنیم که با استفاده از حالت شمارش صعودی (UP COUNTER) تایمر 2 و وقفه تطابق مقایسه (COMPARE MATCH) سیگنال PWM (مدولاسیون عرض پالس) تولید کند.

برای کسب اطلاعات بیشتر در رابطه با مدولاسیون عرض پالس به قسمت پیکره بندی تایمر/کانتر ها در مد PWM در ادامه همین فصل مراجعه کنید .نحوه تولید پالس PWM در این پروژه در شکل 65-2 نشان داده شده است .



شکل 65-2 نحوه تولید سیگنال PWM با استفاده از مد مقایسه ای تایمر 2

اگر محتوای تایمر/کانتر را در هر لحظه روی یکی از پورت ها قرار داده و آن را توسط یک مبدل دیجیتال به آنالوگ 8 بیتی به مقدار آنالوگ تبدیل کنیم یک شکل موج شیب بوجود می آید. یک مبدل دیجیتال به آنالوگ

(ADC) 8 بیتی فاصله بین +VREF تا -VREF را به 255 قسمت تقسیم کرده و مقدار VOUT را با استفاده از عدد ورودی 8 بیتی تعیین می کند برای مثال مقدار ولتاژ خروجی در شرایطی که عدد 8 بیتی ورودی برابر با 120 وVREF مثبت برابر با 5 ولت و VREFمنفی برابر با صفر ولت باشد به صورت زیر خواهد بود .

5/255 = 0.0196
VOUT=120* 0.0196 = 2.35 V

فرکانس سیگنال PWM خروجی به صورت زیر محاسبه می شود .

فرکانس کاری تایمر 2 = فرکانس کاری میکرو / PRESCALE=8000000 /8=1MHz

زمان صعود یک شماره برای تایمر 2 = 1/ 1MHz = 1US

زمان سرریز شدن تایمر =PWM زمان تناوب = 1US*256=256US

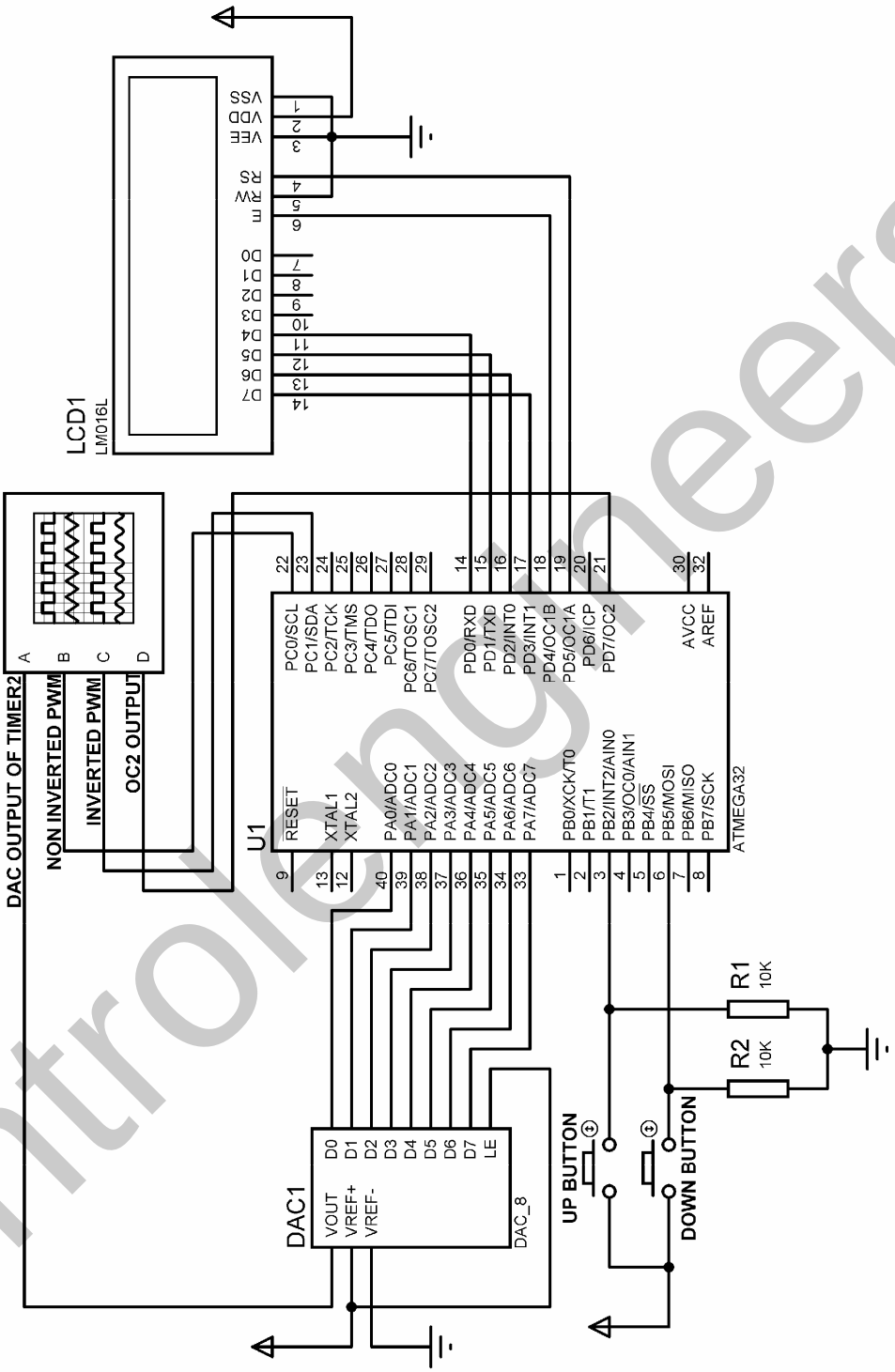
فرکانس PWM = 1/256US=3.906KHz

برای تغییر فرکانس PWM خروجی می توان مقدار PRESCALE تایمر یا مقدار فرکانس کاری میکرو را تغییر داد. البته این پروژه جهت آشنایی با وقفه COMPARE MATCH طراحی شده است شما برای تولید سیگنال هایPWM با دقت بالا تر می توانید تایمر/ کانتر را در مد PWM پیکره بندی کنید .

در این پروژه زمانی که محتوایOCR2 با محتوای تایمر برابر شدPORTC.0، RESET شده و PORTC.1 SET، می شود و هنگام سریز شدن تایمر PORTC.0 ، SET شده و PORTC.1 ، RESET می شود و با هر بار یک شدن پرچم COMPARE MATCH ، مقدار منطقی پایه OC2 معکوس می شود . مقدار اولیه رجیستر OCR2 برابر با 100 بوده و با هر بار فشار کلید UP BUTTON ، 10 واحد به آن اضافه و با هر بار فشار کلید DOWN BUTTON ، 10 واحد از آن کم می شود ، هر دو ورودی حساس به لبه بالا رونده می باشند .

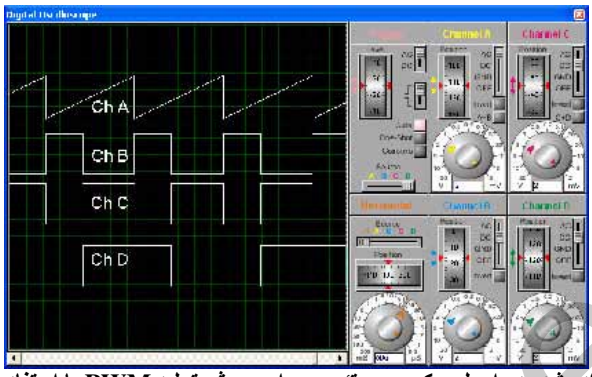
شماتیک پروژه در شکل 2-66 ارائه شده است . در هر لحظه محتوای OCR2 به صورت زیر بر روی LCD نمایش داده می شود .

OCR2 IS 100



شکل 2-66 شماتیک پروژه تولید سیگنال PWM با استفاده از مد COMPARE تایمر2

شما می توانید با تغیر مقدارOCR2 سیگنالهایPWM با ضریب چسبندگی یا زمان وظیفه متفاوت ایجاد کنید. شکل موج های نشان داده شده برای 4 کانال اسیلوسکوپ در سیمولاتور پروتئوس هنگامی که OCR2 برابر 100 می باشد در شکل 67-2 ارائه شده است .



شکل 67-2 نتیجه نشان داده شده در اسیلوسکوپ پروتئوس برای پروژه تولید PWM با استفاده از مد مقایسه ای تایمر 2

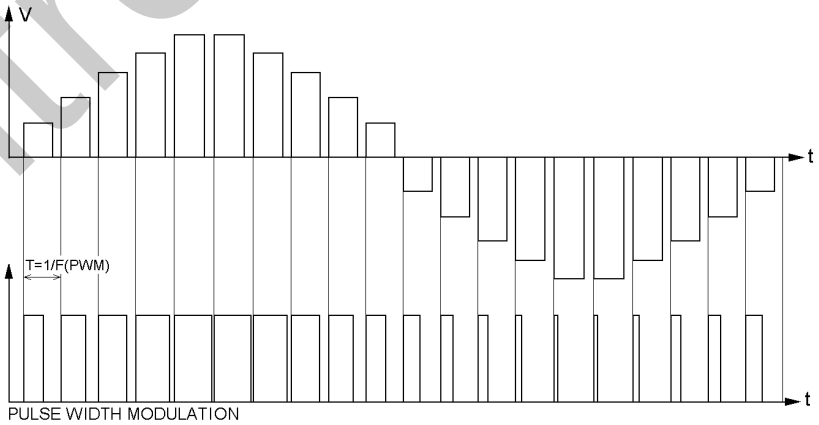
برنامه نوشته شده برای پروژه به صورت زیر می باشد.

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Timer2 = Timer , Compare = Toggle , Prescale = 8
Config Pind.7 = Output
Config Pinc.0 = Output
Config Pinc.1 = Output
Config Porta = Output
Config Pinb.2 = Input
Config Pinb.5 = Input
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
        , E = Pind.4 , Rs = Pind.5
Dim Pwm As Byte
'-----------------------------------------------------------------------------------------------
Enable Interrupts
Enable Oc2
On Oc2 Compparematch
'-----------------------------------------------------------------------------------------------
Enable Timer2
On Ovf2 Timer2_isr
Pwm = 100
Timer2 = 0
'-----------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "OCR2 IS"
Lowerline
Lcd Pwm
'-----------------------------------------------------------------------------------------------
Do
If Pinb.2 = 1 Then
If Pwm < 250 Then Pwm = Pwm + 10
Cls : Home
Lcd "OCR2 IS"
Lowerline
Lcd Pwm
H1:
```

```
Porta = Timer2
If Pinb.2 = 1 Then Goto H1
End If
'-----------------------------------------------------------------------------------------------------------
If Pinb.5 = 1 Then
If Pwm > 0 Then Pwm = Pwm - 10
Cls : Home
Lcd "OCR2 IS"
Lowerline
Lcd Pwm
H2:
Porta = Timer2
If Pinb.5 = 1 Then Goto H2
End If
'-----------------------------------------------------
Ocr2 = Pwm
Porta = Timer2
'-----------------------------------------------------
Loop
'-----------------------------------------------------------------------------------------------------------
Compparematch:
Reset Portc.0
Set Portc.1
Return
'-----------------------------------------------------------------------------------------------------------
Timer2_isr:
Set Portc.0
Reset Portc.1
Timer2 = 0
Return
'-----------------------------------------------------------------------------------------------------------
```

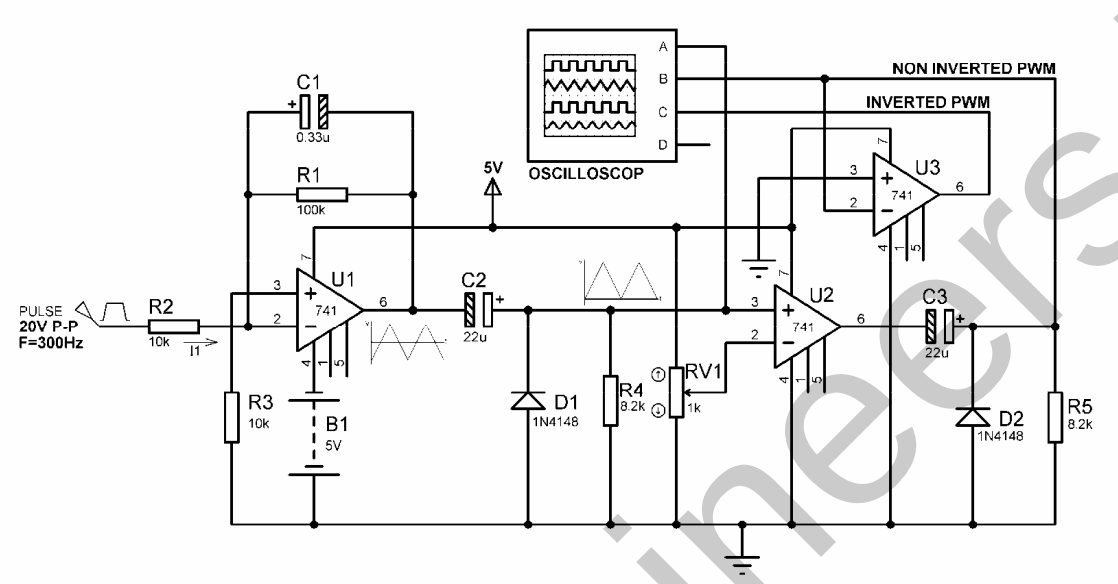
معرفی مد PWM و پیکره بندی تایمر/کانتر ها در این مد

مد PWM برای تولید مدولاسیون عرض پالس (PULSE WIDTH MODULATION) مورد استفاده قرار می گیرد . در مدولاسیون عرض پالس دامنه پالسها ثابت بوده و عرض آن ها متغیر است بدین صورت که باریکترین پالس نشان دهنده کمترین (منفی ترین) مقدار وعریض ترین پالس نشان دهنده مثبت ترین مقدار خواهد بود . شکل 68-2 نمونه ای از انجام مدولاسیون PWM را بر روی یک شکل موج سینوسی نشان می دهد



شکل 68-2 نمونه ای از انجام مدولاسیون PWM بر روی شکل موج سینوسی

برای ایجاد خروجی PWM آنالوگ می توان از مدارشکل 2-69 استفاده کرد .



شکل 2-69 مدار طراحی شده برای ایجاد خروجی PWM آنالوگ

نحوه عملکرد مدار بدین صورت می باشد ابتدا با انتگرال گیری از موج مربعی ورودی یک موج مثلثی توسط U1 ایجاد می شود که دارای نیم سیکل مثبت و منفی است برای حذف نیم سیکل منفی بایستی از یک مدار قفل کننده دیودی مثبت استفاده کنیم . مدار قفل کننده دیودی مثبت ولتاژ ورودی را در VD– (ولتاژ آستانه دیود) قفل کرده و در خروجی ظاهر می کند یعنی ما در خروجی به اندازه ولتاژ آستانه دیود (0.6 یا 0.7) سیکل منفی خواهیم داشت که برای حذف آن هم می توانید از معادل آپ امپی دیود استفاده کنید . قطعات انتگرال گیر عبارتند از C1 ، R1 ، U1 ، B1 ، R3 و R2 نحوه طراحی انتگرال که بدین صورت می باشد . با توجه به این که فرکانس موج مربعی ورودی برابر 300Hz بوده و دامنه آن ±10V می باشد برای داشتن خروجی مثلثی 5VP-P مقادیر قطعات انتگرال گیر به صورت زیر تعیین می شود.

مقدار جریان ورودی به پایه های مثبت و منفی آپ امپ در محاسبات معمولا صفر در نظر گرفته می شود یعنی مقاومت ورودیOP_AMPرا بی نهایت می گیرند ولی در عمل چنین چیزی امکان پذیر نیست با استفاده از DATA SHEET مربوط به آپ امپ مورد نظر می توانیم مقادیر مقاومت ورودی و جریان ورودی را بدست آوریم که در آپ امپ 741 مقدار جریان ورودی IB(MAX) برابر با 500nA می باشد با توجه به این که I1>>IB(MAX) مقدار I1=1mA در نظر گرفته می شود . در این حالت R1 از فرمول زیر بدست می آید .

R2=VI /I1=10 / 1mA = 10 K

از طرفی R1 همواره بایستی برابر با 10R2 باشد.

R1=10R2=100K

ما در محاسبات جریان ورودی را صفر و ولتاژ پایه مثبت و منفی را برابرصفر می گیریم و در عمل جریان ناچیزی در حد 500nA وارد آپ امپ می شود در این صورت برای برابر کردن ولتاژ پایه های ورودی مثبت و ورودی منفی آپ امپ بایستی مقاومت AC این دو پایه با هم برابر باشند .

در نتیجه R3 به صورت زیر تعیین می شود .

R3=R1//R2=10K

طول یک شیب موج مثلثی برابر با نصف تناوب ورودی است یعنی ½ F

T=½300Hz=1.66ms

از طرفی دامنه خروجی شیب بایستی برابر با 5V(P-P) باشد .

C1=I1 /VO(P-P) = 1mA* 1.66ms/5=0.33UF

همان طور که قبلا نیز گفته شد خروجی انتگرال گیر میلر دارای دامنه 5V(P-P) یعنی ±2.5VP می باشد ، برای حذف قسمت منفی موج مثلثی از یک مدار قفل کننده مثبت دیودی استفاده می کنیم . قطعات مدار قفل کننده عبارتند از C2,D1,R4 با توجه به این که ولتاژ ورودی ±2.5V می باشد و مقاومت خروجیOP-AMP را هم با توجه بهDATA SHEET ،741برابر75اهم در نظر گرفته ایم برای این که کجی خروجی بیشتر از یک درصد نباشد مدار قفل کننده به صورت زیر طراحی می شود .

T=1/F = 1/300Hz=3.33ms

(پهنای پالس) PW=T/2= 1666.65US

C2=PW/RO(OP-AMP)=1666.65US/75=22.22UF

با توجه به این که 22.22UF رنج استاندارد نمی باشد از خازن 22UF استفاده می کنیم مقدار R4 نیز برای این که کجی خروجی بیشتر از یک درصد نباشد به صورت زیر تعیین می شود .

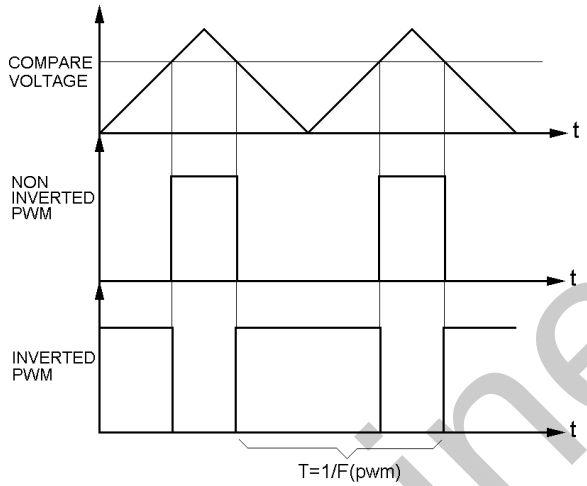
R4= PW/0.01 * C2=7.5K

از رنج استاندارد 8.2K برایR4 استفاده می کنیم.

حال با اعمال یک ولتاژ DC به ورودی منفی U2 و اعمال موج مثلثی به ورودی مثبت آن از حالت مقایسه گر OP-AMP استفاده کرده ایم و خروجی PWM خواهیم داشت که ضریب اتصال آن توسط RV1 تعیین می شود. نحوه عملکردOP_AMP در حالت مقایسه گر به این صورت می باشد که اگر ولتاژ پایه ورودی منفی بیشتر از ولتاژ پایه ورودی مثبت باشد خروجی به اشباع منفی خواهد رفت .

خروجی PWM ،UZ دارای سطح DC می باشد که برای حذف آن دوباره از یک مدار قفل کننده مثبت دیودی با قطعات R5,D2,C3 استفاده شده است .

از U3 نیز برای معکوس کردن خروجی PWM به منظور ایجاد INVERTED PWM استفاده شده است ، شکل موج PWM خروجی در شکل 2-70 نشان داده شده است .

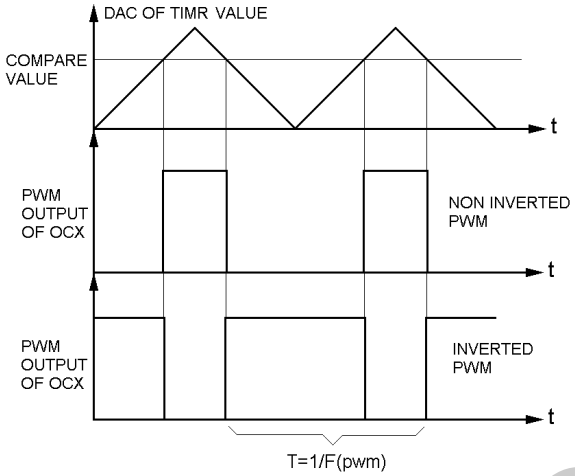


شکل 2-70 شکل موج خروجی مدار تولید کننده PWM آنالوگ

COMPARE VOLTAGE همان ولتاژ DC اعمالی به پایه منفی OP_AMP می باشد که توسط پتانسیو متر RV1 ایجاد می شود با تغییر پتانسیو متر RV1 می توانید ضریب اتصال PWM خروجی راتغییر دهید .

در میکروکنترلر ATMEGA32 از تایمر کانترهای 1و2 می توان برای تولید پالس PWM استفاده کرد ، زمانی که تایمر/ کانتر دو در مد PWM استفاده می شود رجیستر مقایسه OCR2 که یک رجیستر 8 بیتی است برای تولید پالسPWM در پایه OC2 استفاده می شود . تایمر/کانتر ها در مد PWM به صورت UP/DOWN COUNTER عمل می کنند به دلیل این که تایمر/کانتر صفر قابلیت شمارش DOWN COUNTER را ندارد نمی توان از آن در مد PWM استفاده کرد .

هنگامی که از تایمر/کانتر 2 در مد PWM استفاده می شود این تایمر/کانتر در زمان UPCOUNTE از 0 تا 255 و در زمان DOWN COUNTER از 255 تا 0 می شمارد. زمانی که محتوای تایمر/کانتر با محتوای OCR2 برابر شد وضعیت پایه OC2 بنا به تعریف تغییر می کند. نحوه تولید PWM توسط تایمر/کانتر 2 در شکل 2-71 نشان داده شده است .



شکل 2-71 نحوه ایجاد PWM با استفاده از تایمر/کانتر 2 روی پایه OC2

در تایمر/کانتر 2 فرکانس PWM با استفاده از جدول زیر بدست می آید .

PWM RESOLATION	TIMER TOP VALUE	FREQENCY
8-BIT	$FF(255)	FTCK2/510

FTCK2 فرکانس کاری تایمر/کانتر 2 می باشد که از فرمول (TIMER PRESCALE / فرکانس کاری میکرو) بدست می آید .

فرکانس PWM تولید شده با استفاده از فرمول زیر محاسبه می شود .

PWM FREQUNCY = FOSC / (510*PRESCALE)

FOSC : فرکانس کاری میکرو کنترلر می باشد .

PRESCALE : برای تولید PWM با فرکانس های مختلف از این گزینه استفاده می شود و می تواند مقادیر 1 ,8 , 64 ,256 ,1024 را داشته باشد . نحوه پیکره بندی تایمر/کانتر 2 در مد PWM به صورت زیر است.

CONFIG TIMER2=PWM,PRESCALE=1/8/64/256/1024,PWM=ON/OFF
COMPARE PWM=CLEAR UP/CLEAR DOWN/DISCONNECT

PRESCALE : برای تولید PWM با فرکانس های مختلف از این گزینه استفاده می شود .

PWM=ON/OFF : برای این که بتوانیم از تایمر در مد PWM استفاده کنیم بایستی گزینه ON انتخاب شود

CLEAR UP : در صورت استفاده از این گزینه پالس PWM به صورت INVERTED در پایه خروجی OC2 ایجاد می شود .

DISCONNECT : در صورت استفاده از این گزینه پالس PWM در زمان تطابق مقایسه از پایه خروجی OC2 قطع می شود .

برای نوشتن در رجیستر های OCR2 نیز می توانید از دستورOCR2=VAR استفاده کنید که VAR ثابت یا متغیری از نوع BYTE می باشد .

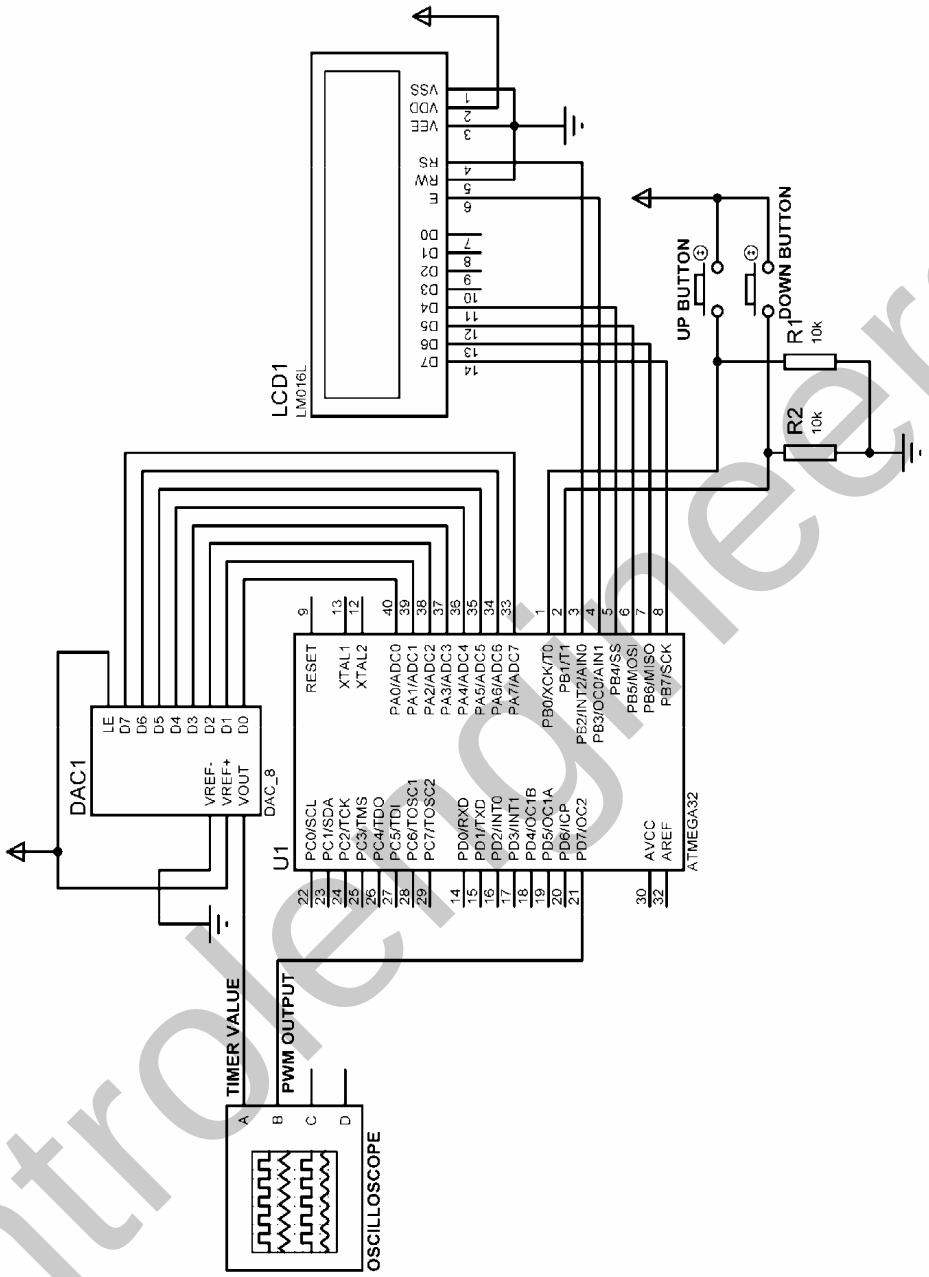
نکته مهم این که پین OC2 حتما بایستی خروجی تعریف شود به عنوان مثال در میکروکنترلر ATMEGA32 پایه OC2 روی پین D.7 قرار گرفته و پین D.7 بایستی خروجی CONFIG شود .

پروژه ایجاد PWM هشت بیتی با ضریب چسبندگی قابل تنظیم

به عنوان مثال پروژه هایی طراحی می کنیم که سیگنال PWM را با استفاده از تایمر/کانتر 2 روی پایه خروجی OC2 تولید کند مقدار رجیستر OCR2 نیز توسط دو کلید UP BUTTON و DON BUTTON قابل تغییر است ، در ضمن محتوای رجیستر OCR2 در هر لحظه به صورت زیر روی LCD نمایش داده می شود و مقدار پیش فرض OCR2 برابر 100 تنظیم شده است .

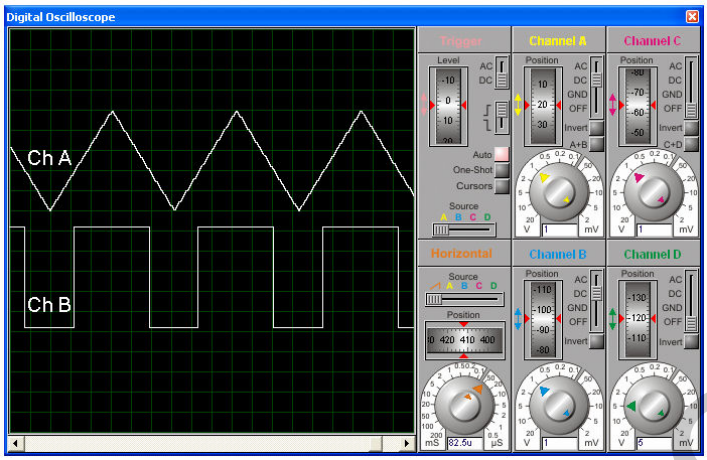
```
COMPARE PWM IS
100
```

شماتیک پروژه در شکل 72-2 نشان داده شده است .



شکل 2-72 شماتیک طراحی شده برای پروژه ایجاد PWM هشت بیتی با ضریب چسبندگی قابل تنظیم

شکل موج های نشان داده شده برای دو کانال A,B اسیلوسکوپ در سیمولاتور پروتئوس به صورت شکل 2-73 است که کانال A مقادیر DAC شده تایمر 2 و کانال B ، شکل موج PWM خروجی را نشان می دهد .



شکل 73-2 شکل موج های نشان داده شده برای تولید PWM ، 8 بیتی توسط تایمر 2

برنامه نوشته شده برای پروژه به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Timer2 = Pwm , Prescale = 8 , Pwm = On , Compare Pwm_
      = Clear Down
Config Pinb.0 = Input
Config Pinb.1 = Input
Config Pind.7 = Output
Config Porta = Output
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinb.4 , Db5 = Pinb.5 , Db6 = Pinb.6 , Db7 = Pinb.7_
                , E = Pinb.3 , Rs = Pinb.2
Dim Pwm As Byte
Pwm = 100
'-----------------------------------------------------------------------------------------------------------
Enable Interrupts
Cursor Off
Cls : Home
Lcd "COMPARE PWM IS"
Lowerline
Lcd Pwm
'START-------------------------------------------------------------------------------------------------------
Do
'-----------------------------------------------------------------------------------------------------------
If Pinb.0 = 1 Then
If Pwm < 255 Then Pwm = Pwm + 5
Cls : Home
Lcd "COMPARE PWM IS"
Lowerline
Lcd Pwm
H1:
Porta = Timer2
If Pinb.0 = 1 Then Goto H1
End If
'-----------------------------------------------------------------------------------------------------------
If Pinb.1 = 1 Then
If Pwm > 0 Then Pwm = Pwm - 5
Cls : Home
Lcd "COMPARE PWM IS"
Lowerline
```

```
Lcd Pwm
H2:
Porta = Timer2
If Pinb.1 = 1 Then Goto H2
End If
'-----------------------------------------------------------------------------------------------------------
Ocr2 = Pwm
Porta = Timer2
Loop
'END-----------------------------------------------------------------------------------------------------------
```

فرکانس PWM در برنامه فوق به صورت زیر محاسبه می شود .

PWM FREQUENCY = FOSC / (510 * PRESCALE)
F=8000000 / (510 * 8) = 1.96KHZ

نحوه طراحی دمدولاتور PWM

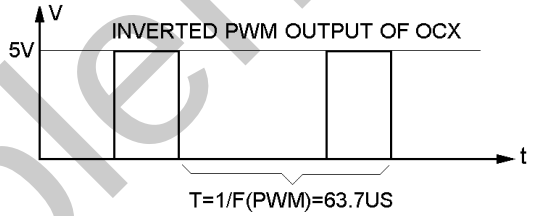
فرض کنید می خواهیم عددی را که در رجیستر مقایسه ای میکروکنترلر مدولاتور قرار دارد را توسط یک میکروکنترلر دمدولاتور دریافت کنیم.

اگر فرکانس PWM تولید شده در مدولاتور به صورت زیر باشد .

F(PWM)=8000000 / (510 * 1) = 15.686KHz

زمان نمونه برداری برابر خواهد بود با

T= 1 / 15.686KH =63.7US

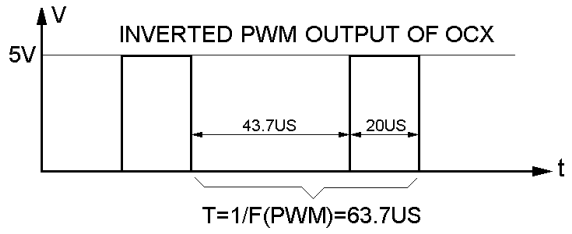


شکل 74-2 شکل موج PWM خروجی

اگر از PWM تایمر/کانتر دو استفاده کرده باشیم دقت PWM هشت بیتی بوده و محتوای رجیستر مقایسه ای OCR2 نیز بایستی 8 بیتی باشد همچنین درPWM ، 8 بیتی یک دوره تناوب (نمونه برداری) به 255 قسمت مساوی تقسیم خواهد شد .

63.7US /255 = 0.2US

پس مدت زمان نمونه برداری به 255 قسمت 0.2US تقسیم می شود در این شرایط اگر محتوای رجیستر مقایسه ای OCR2 برابر با100بوده و خروجی PWM به صورت INVERTED باشد . زمان صفر شدن سیگنال خروجی برابر (100*0.2US=20US) و زمان یک شدن سیگنال خروجی برابر (63.7US-20US =43.7US) خواهد بود .



شکل 2-75 شکل موج PWM خروجی همراه با مدت زمان toff و ton سیگنال PWM

حال اگر در میکروکنترلر دمدولاتور با استفاده از یک تایمر زمان صفر بودن سیگنال ورودی را اندازه بگیریم می توانیم عددی را که در رجیستر مقایسه ای مدولاتور قرار داشته بدست آوریم اگر فرکانس کاری تایمر در دمدولاتور برابر با 32MHz بوده و PRESCALE تایمر برابر با 8 باشد فرکانس کاری تایمر برابر است با

فرکانس کاری تایمر = 32MHz /8 =4MHz

نکته : فرکانس کاری میکروکنترلرهای معرفی شده در این کتاب در حالت عملی نمی تواند بیشتر از 16 مگاهرتز باشد ولی شما می توانید در محیط پروتئوس فرکانس کاری میکرو را هر اندازه که می خواهید بالا ببرید برای همین به منظور یادگیری بهتر مطلب ابتدا فرکانس کاری میکروکنترلر دمدولاتور را روی 32 مگاهرتز تنظیم می کنیم . پس از انجام دمدولاسیون در محیط پروتئوس و درک نحوه انجام آن نحوه کاهش فرکانس کاری دمدولاتور به 16 مگاهرتز نیز آموزش داده خواهد شد .

در این حالت مدت زمان صعود یک شماره برابر خواهد بود با

مدت زمان صعود یک شماره =1 / 4MHZ = 0.2US

اگر از تایمر/کانتر یک استفاده کنیم مدت زمان سرریز شدن تایمر برابر خواهد بود

65536 * 0.2US =13.1MS

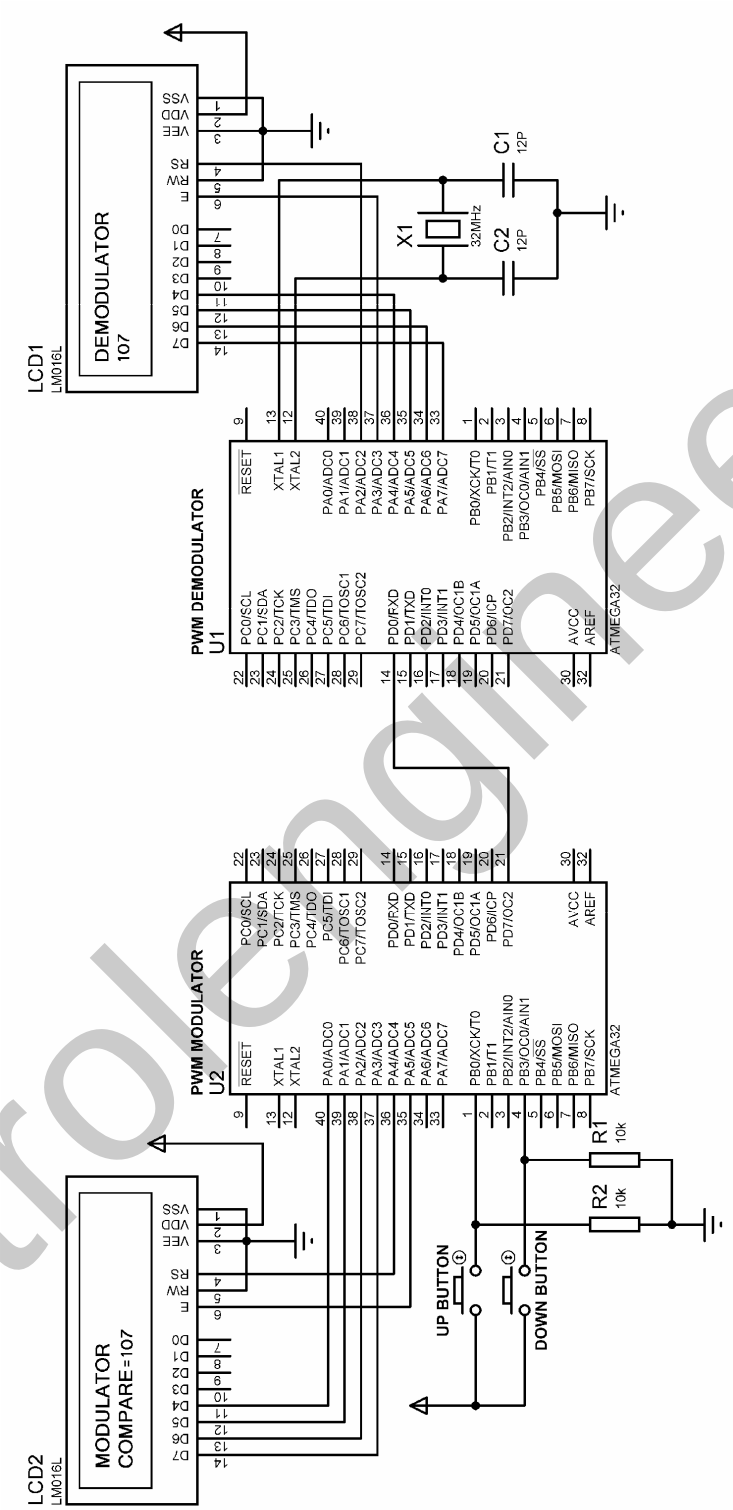
که این زمان بسیار بیشتر از یک دوره نمونه برداری در مدولاتور یعنی 63.7US می باشد ، پس تایمر/کانتر یک برای اندازه گیری مدت زمان صفر بودن سیگنال مناسب است ، با توجه به این که در میکرو کنترلر مدولاتور مدت زمان نمونه برداری به255 قسمت 0.2US تقسیم می شود و مدت زمان صعود یک شماره در تایمر میکرو کنترلر دمدولاتور برابر0.2US می باشد . محتوای تایمر برابر با محتوای رجیستر مقایسه ای OCR2 در مدولاتور خواهد بود .

پروژه ارسال یک عدد با استفاده از مدولاسیون PWM و دریافت آن توسط میکروکنترلر دمدولاتور

به عنوان مثال در پروژه زیر عدد وارد شده برای رجیستر مقایسه ای OCR2 مدولاسیون PWM شده و در پایه OC2 میکروکنترلر مدولاتور قرار گیرد سیگنال PWM وارد یکی از پایه های ورودی میکروکنترلر دمدولاتور شده و میکروکنترلر پس از ازانجام عمل دمدولاسیون PWM عددی را که در رجیستر مقایسه ای میکروکنترلر مدولاتور قرار گرفته بود روی LCD نشان می دهد . سخت افزار پروژه در شکل 2-76 نشان داده شده است.

برنامه نوشته برای مدولاتور به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Timer2 = Pwm , Prescale = 1 , Pwm = On , Compare Pwm_
        = Clear Down
Config Pinb.0 = Input
Config Pinb.3 = Input
Config Pind.7 = Output
Config Porta = Output
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3_
                    , E = Pina.5 , Rs = Pina.4
Dim Pwm As Byte
Pwm = 100
'-----------------------------------------------------------------------------------------------------------
Enable Interrupts
Cursor Off
Cls : Home
Lcd "MODULATOR"
Lowerline
Lcd "COMPARE=" ; Pwm
'START-------------------------------------------------------------------------------------------------------
Do
'-----------------------------------------------------------------------------------------------------------
If Pinb.0 = 1 Then
If Pwm < 255 Then Incr Pwm
Cls : Home
Lcd "MODULATOR"
Lowerline
Lcd "COMPARE=" ; Pwm
End If
'-----------------------------------------------------------------------------------------------------------
If Pinb.3 = 1 Then
If Pwm > 0 Then Decr Pwm
Cls : Home
Lcd "MODULATOR"
Lowerline
Lcd "COMPARE=" ; Pwm
End If
'-----------------------------------------------------------------------------------------------------------
Ocr2 = Pwm
Waitms 50
Loop
'END OF PROGRAM-----------------------------------------------------------------------------------------------
```



شکل 2-76 شماتیک طراحی شده برای ارسال و دریافت یک عدد 8 بیتی با استفاده از مدولاسیونPWM

برنامه نوشته شده برای دمدولاتور به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 32000000
Config Pind.0 = Input
Config Timer1 = Timer , Prescale = 8
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pina.4 , Db5 = Pina.5 , Db6 = Pina.6 , Db7 = Pina.7_
                , E = Pina.3 , Rs = Pina.2
Dim Save As Byte
Stop Timer1
Cursor Off
'-----------------------------------------------------------------------------------------------------------
Do
Timer1 = 0
Bitwait Pind.0 , Reset
Bitwait Pind.0 , Reset
Start Timer1
Bitwait Pind.0 , Set
Stop Timer1
If Timer1 <> Save Then
Cls : Home
Lcd "DEMODULATOR"
Lowerline
Lcd Timer1
End If
Save = Timer1
Bitwait Pind.0 , Set
Loop
'-----------------------------------------------------------------------------------------------------------
```

دستورات جدید :

در برنامه فوق از دستور BIT WAIT استفاده شده که فرم کلی آن به صورت زیر می باشد .

BITWAIT X,SET/RESET

توسط این دستور اجرا ی برنامه تا SET یا RESET شدن پایه X در خط جاری متوقف می شود و پس از SET RESET شدن اجرای برنامه از خط بعدی ادامه پیدا می کند .

تشریح نحوه عملکرد دمدولاتور :

نحوه عملکرد برنامه در دمدولاتور بدین صورت می باشد که ابتدا میکروکنترلر صبر می کند تا پین D.0 ، RESET شود ، به محض RESET شدن پین D.0 تایمر یک شروع به شمارش می کند ، شمارش تایمر یک تا زمان SET شدن پین D.0 ادامه خواهد یافت و بلافاصله پس از SET شدن آن شمارش تایمر متوقف شده و مقدار آن به عنوان عدد دریافت شده بر روی LCD نمایش داده می شود .

پروژه ارسال و دریافت یک سیگنال سینوسی با استفاده از مدولاسیون PWM

حالا می خواهیم یک سیگنال سینوسی با دامنه 2.5V(P-P) را با استفاده از یک ADC ، 8 بیتی تبدیل به دیجیتال کرده و آن را در رجیستر مقایسه ای OCR2 قرار دهیم و سیگنال PWM موجود بر روی پایه OC2

مدولاتور را به ورودی میکروکنترلر دمدولاتور اعمال کرده و در نهایت سیگنال سینوسی با دامنه 2.5V(P-P) آشکار سازی و آن را بر روی اسیلوسکوپ سیمولاتور پروتئوس نمایش دهیم . در قسمت دمدولاتور برای تبدیل مقدار دیجیتال خروجی به مقدار آنالوگ از یک DAC , 8 بیتی استفاده شده است .

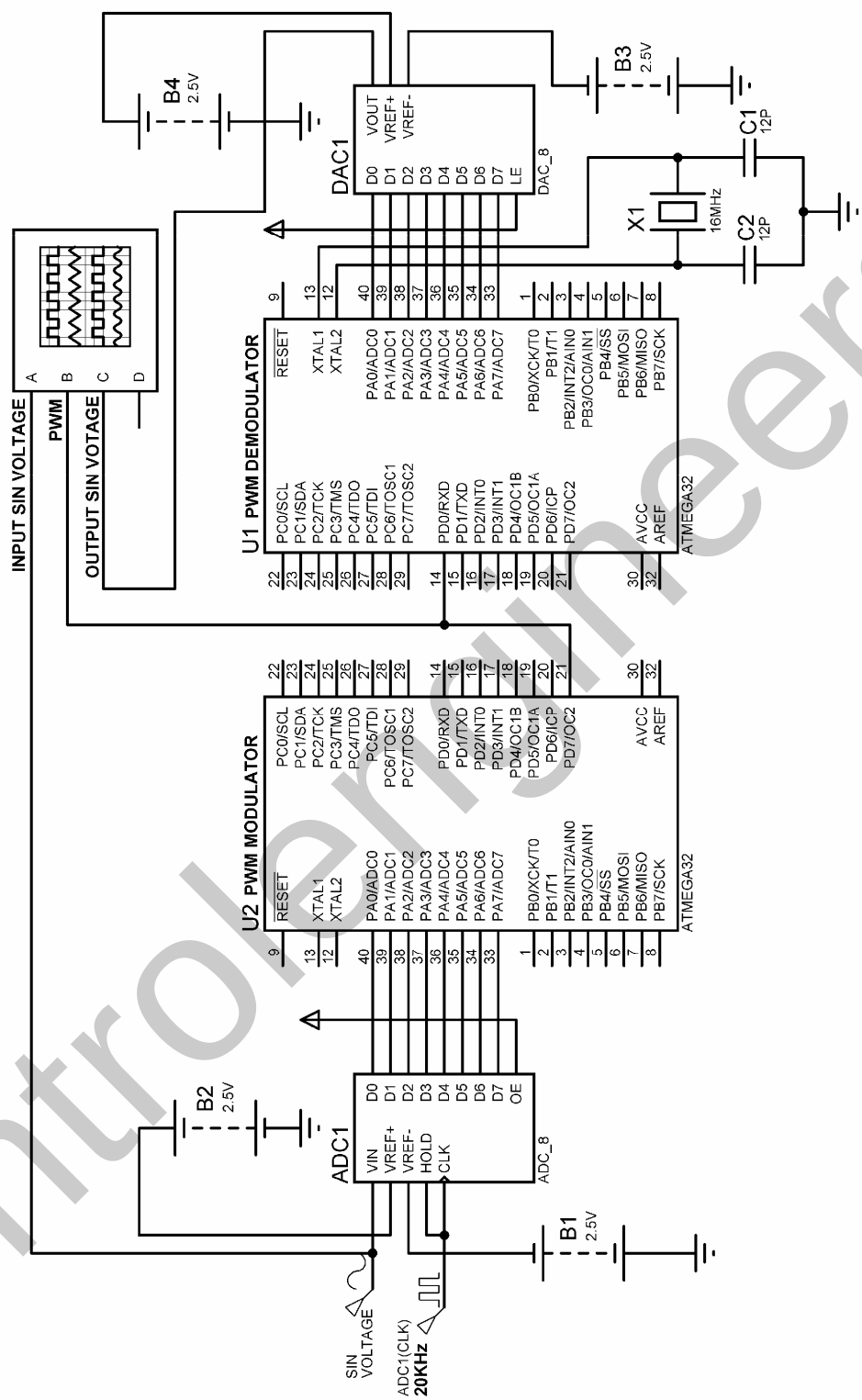
سخت افزار پروژه در شکل 77-2 ارائه شده است .

برنامه نوشته شده برای مدولاتور به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Timer2 = Pwm , Prescale = 1 , Pwm = On , Compare Pwm_
        = Clear Down
Config Pind.7 = Output
Config Porta = Input
'-----------------------------------------------------------------------------------------------------------
Do
Ocr2 = Pina
Loop
'-----------------------------------------------------------------------------------------------------------
```

برنامه نوشته شده برای دمدولاتور به صورت زیر است

```
$regfile = "M32DEF.DAT"
$crystal = 16000000
Config Pind.0 = Input
Config Porta = Output
Config Timer1 = Timer , Prescale = 8
Stop Timer1
Cursor Off
'-----------------------------------------------------------------------------------------------------------
Do
Bitwait Pind.0 , Reset
Start Timer1
Bitwait Pind.0 , Set
Stop Timer1
Timer1= Timer1*2
Porta = Timer1
Timer1 = 0
Loop
'-----------------------------------------------------------------------------------------------------------
```



شکل 2-77 ارسال و دریافت یک سیگنال سینوسی از طریق مدولاسیون PWM

در رابطه با تعیین فرکانس سیگنال سینوسی ورودی بایستی به روابط زیر توجه کنیم همان طور که در پروژه قبل توضیح داده شد فرکانس PWM برابر با 15.686KHz می باشد بنابر این مدت زمان نمونه برداری برابر T= 1/ 15.686K = 63.7US خواهد بود به عنوان مثال تعداد نمونه برداری در یک سیکل برای سیگنال ورودی با فرکانس 100Hzبرابر خواهد بود با

T=1/100Hz = 0.01S

تعداد نمونه برداری از یک سیکل = 0.01S / 63.7US= 156.98

پس شکل سیگنال خروجی قابل قبول خواهد بود ولی اگر فرکانس ورودی برابر 3KHz باشد

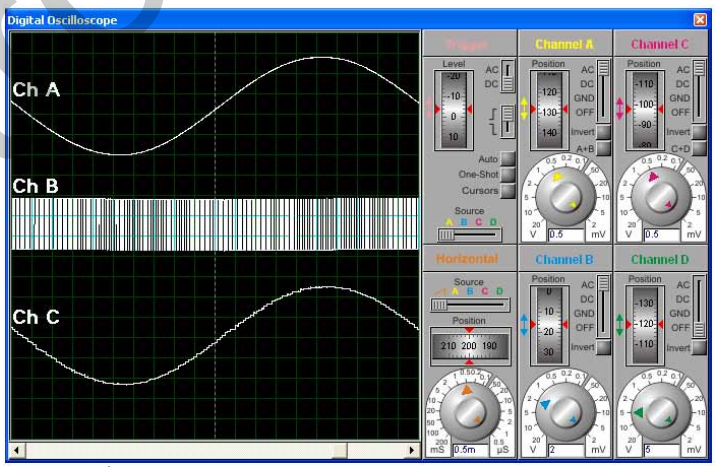
T= 1 / 3KHz = 0.33MS

تعداد نمونه برداری از یک سیکل = 0.33MS / 63.7US =5.23

در نتیجه شکل سیگنال خروجی برای ورودی 3KHz شکل قابل قبولی نخواهد بود. شکل سیگنال خروجی نشان داده شده برای ورودی 100Hz با دامنه 2.5V(P-P) در اسیلوسکوپ سیمولاتور پروتئوس به صورت زیر می باشد . در شکل زیر کانال A سیگنال ورودی ، کانال B سیگنال PWM و کانال C سیگنال خروجی را نشان می دهد .

تشریح نحوه عملکرد برنامه دمدولاتور

نحوه عملکرد میکروکنترلر دمدولاتور بدین صورت می باشد که ابتدا میکروکنترلر صبر می کند تا پین D.0 ، RESET شود ، به محض RESET شدن پین D.0 تایمر یک شروع به شمارش می کند ، شمارش تایمر یک تا زمان SET شدن پین D.0 ادامه خواهد یافت و بلافاصله پس از SET شدن شمارش تایمر متوقف می شود با توجه به این که در این پروژه بر خلاف پروژه قبل فرکانس کاری دمدولاتور برابر با 16MHz است برای دانستن عدد ارسال شده توسط مدولاتور بایستی محتوای تایمر در 2 ضرب شود .



شکل 2-78 نتیجه نمایش داده شده در اسیلوسکوپ پروتئوس برای پروژه ارسال و دریافت سیگنال سینوسی از طریق مدولاسیون PWM

نحوه عملکرد تایمر/کانتر 1 در مد PWM

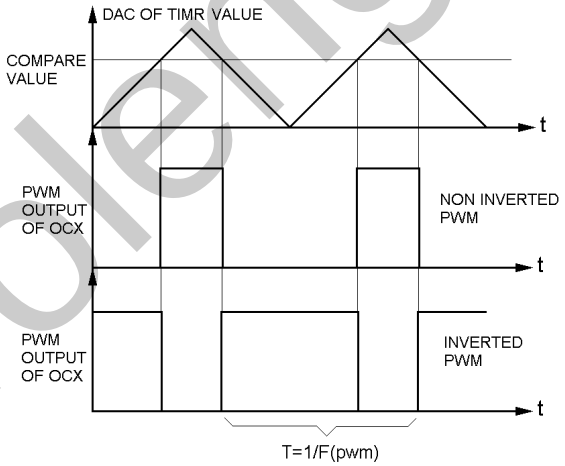
تایمر/کانتر یک دارای دو خروجی PWM در پایه های OC1B،OC1A می باشد زمانی که تایمر/کانتر یک در مد PWM استفاده می شود، رجیستر مقایسه ای A (OCR1A) و رجیستر مقایسه ای B (OCR1B) در حالت 8 ، 9 و 10 بیتی برای تولید پالس PWM در پایه های OC1A ، OC1B استفاده می شوند .

تایمر/کانتر یک در مد PWM به صورت UP/DOWN COUNTER عمل می کند و در زمان UP COUNTER از صفر تا TOP و در زمان DOWN COUNTER از TOP تا صفر می شمارد که عدد TOP توسط جدول زیر و با توجه به این که دقت PWM چند بیتی می باشد تعیین می شود . که FTCK1 در جدول زیر به معنای فرکانس کاری تایمر/کانتر یک است .

PWM RESOLUTION	TIMER TOP VALUE	FREQUENCY
8-BIT	$00FF(255)	FTCK1/510
9-BIT	$01FF(511)	FTCK1/1022
10-BIT	$03FF(1023)	FTCK1/2046

زمانی که محتوای تایمر با محتوای رجیستر های OCR1A یا OCR1B برابر شد و ضعیت پایه های OC1A و OC1B بنا به تعریف تغییر خواهد کرد .

برای درک بهتر نحوه ایجاد PWM به شکل 2-79 توجه کنید .



شکل 2-79 نحوه ایجاد PWM در پایه OCX

فرکانس PWM با توجه به معادله های زیر بدست می آید

PWM FREQUNCY=FOSE/(510*PRESCALE)	8,PWM بیتی :
PWM FREQUNCY=FOSE/(1022*PRESCALE)	9,PWM بیتی :
PWM FREQUNCY=FOSE/(2046*PRESCALE)	10,PWM بیتی :

FOSC : فرکانس کاری میکروکنترلر می باشد .

PRESCALE : برای تولیدPWM با فرکانس های مختلف از این گزینه استفاده می شود و می تواند مقادیر 64,8,1 یا 1024 را داشته باشد .

پیکره بندی تایمر /کانتر یک در مد PWM

CONFIG TIMER1=PWM,PWM=8/9/10,COMPARE A PWM=CLEAR UP/ CLEAR DOWN/DISCONNECT,COMPARE B PWM =CLEAR UP/CLEAR DOWN / DISCONNECT,PRESCALE=1/8/64/256/1024

در TIMER COUNTER یک , PWM می تواند 8 ، 9 و 10 بیتی باشد که در مد های 8 ، 9 و 10 بیتی مقدار بالای تایمر به ترتیب $FF ، $1FF ، $3FF است .

CLEAR UP : در صورت استفاده از این گزینه PWM به صورت INVERTED در پایه خروجی OC1A یا OC1B ظاهر می شود .

CLEAR DOWN : در صورت استفاده از این گزینه PWM به صورت NON-INVERTED در پایه خروجی OC1A یا OC1B ظاهر می شود .

DISCONNECT : در صورت استفاده از این گزینه PWM در زمان تطابق مقایسه از پایه خروجی OC1A یا OC1B قطع می شود .

PRESCALE : برای تولید PWM با فرکانس های مختلف از این گزینه استفاده می شود .

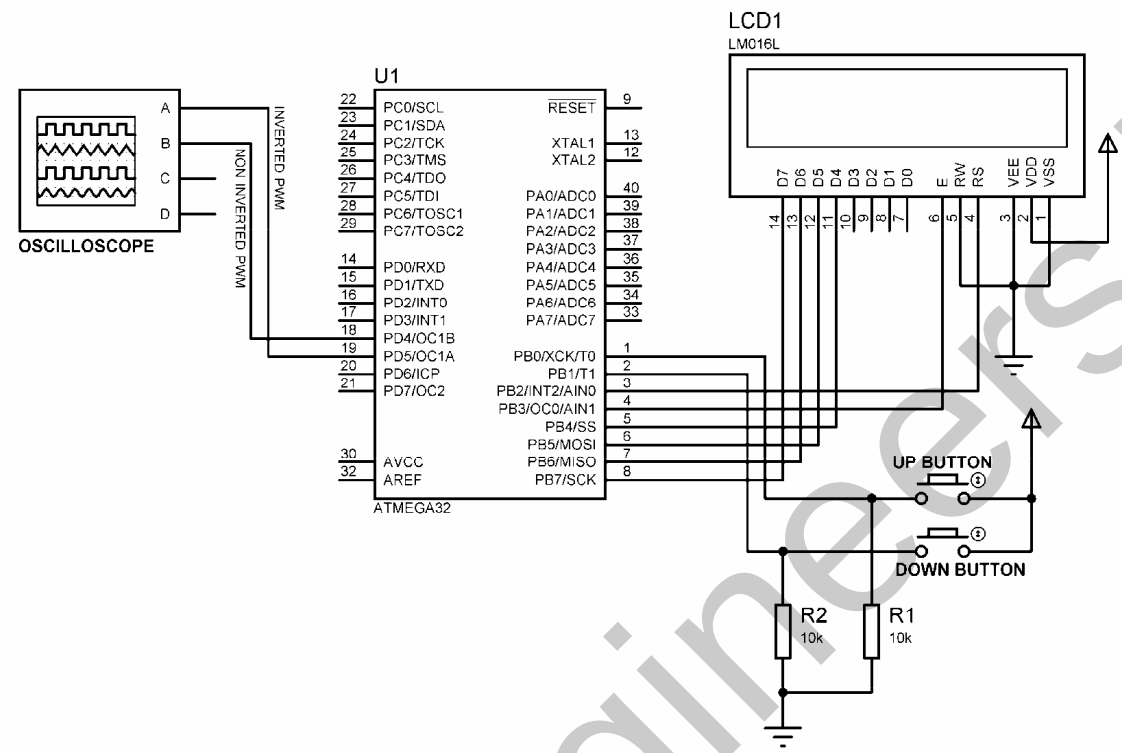
برای نوشتن در رجیستر مقایسه ای B,A می توانید از دستورات PWM1A=VAR , PWM1B=VAR و یا COMPARE1A=VAR , COMPARE1B=VAR استفاده کنید که VAR با توجه به این که دقت PWM چند بیتی است می تواند ثابت یا متغیری 1 یا 2 بایتی باشد.

پروژه تولید PWM با دقت 10 بیتی با خروجی INVERTED و NON NVERTED

به عنوان مثال در پروژه زیر عدد داده شده توسط کلید های ورودی در هر دو رجیستر مقایسه ایA ، B قرار می گیرد و PWM خروجی از پایه OC1A به صورت INVERTED و PWM خروجی از پایه OC1B به صورت NON-INVERTED می باشد . دقت PWM نیز 10 بیتی انتخاب شده است .

یاد آور می شوم پین های مربوط به خروجی PWM حتما بایستی به صورت خروجی CONFIG شده باشند .

سخت افزار طراحی شده برای پروژه در شکل 80-2 ارائه شده است .



شکل 2-80 شماتیک طراحی شده برای پروژه تولید PWM ، 10 بیتی با خروجی INVERTED وNON INVERTED

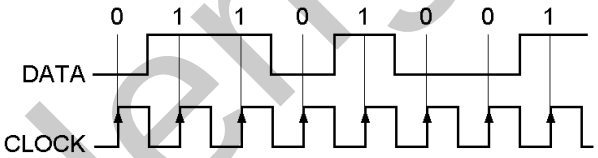
برنامه نوشته شده برای پروژه به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Up , Compare B Pwm_
   = Clear Down , Prescale = 8
Config Pinb.0 = Input
Config Pinb.1 = Input
Config Pind.4 = Output
Config Pind.5 = Output
Config Porta = Output
Cursor Off
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinb.4 , Db5 = Pinb.5 , Db6 = Pinb.6 , Db7 = Pinb.7_
                , E = Pinb.3 , Rs = Pinb.2
Dim Pwm As Word
'-----------------------------------------------------------------------------------------------------------
Pwm = 100
Cls : Home
Lcd "COMPARE PWM IS"
Lowerline
Lcd Pwm
'START-----------------------------------------------------------------------------------------------------
Do
If Pinb.0 = 1 Then
If Pwm < 1020 Then Pwm = Pwm + 10
Cls : Home
Lcd "COMPARE PWM IS"
Lowerline
Lcd Pwm
```

```
End If
'-----------------------------------------------------------------------------------------------------------
If Pinb.1 = 1 Then
If Pwm > 0 Then Pwm = Pwm - 10
Cls : Home
Lcd "COMPARE PWM IS"
Lowerline
Lcd Pwm
End If
'-----------------------------------------------------------------------------------------------------------
Pwm1a = Pwm
Pwm1b = Pwm
Porta = Timer1
'-----------------------------------------------------------------------------------------------------------
Waitms 100
Loop
'END-----------------------------------------------------------------------------------------------------------
```

ارتباط سریال

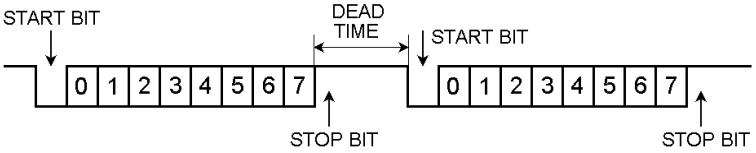
ارتباط سریال می تواند به دو صورت آسنکرون و سنکرون صورت گیرد . در ارتباط سریال سنکرون علاوه بر ارسال اطلاعات پالس ساعت نیز برای MATCH کردن فرستنده و گیرنده از نظر زمانی فرستاده می شود . شکل 2-81 زمان بندی ارسال و دریافت یک بایت به صورت سنکرون را نشان می دهد توجه داشته باشید که وجود لبه بالا رونده در خط CLOCK فرمان دریافت داده را به گیرنده می دهد ، یعنی گیرنده در 8 لبه بالا رونده یک بایت را از خط DATA می گیرد.



شکل2-81 زمان بندی ارسال و دریافت یک بایت به صورت سنکرون

از ارتباط سریال سنکرون می توان به پروتکل ارتباطی SPI در میکروکنترلر های AVR اشاره نمود .

در ارتباط سریال آسنکرون پالس ساعت ارسال نمی شود و به جای پالس ساعت از مفهوم سرعت انتقال بیت(BAUD) استفاده می شود به عبارت دیگر سرعت انتقال در فرستنده و سرعت دریافت در گیرنده برابر خواهد بود و برای تشخیص شروع و پایان انتقال یک بایت داده از START BIT و STOP BIT استفاده می شود ، اگر START BIT را صفر و STOP BIT را یک در نظر گرفته و فاصله بین دو بایت ارسالی را DEAD TIME در نظر بگیریم. شکل 2-82 نحوه ارسال اطلاعات به صورت آسنکرون را نشان می دهد .



شکل 2-82 نحوه ارسال اطلاعات به صورت آسنکرون

BAUD سرعت انتقال اطلاعات بوده و واحد آن BPS (بیت بر ثانیه) می باشد برای مثال با تعیین میزان BAUD برابر 2400 ، اطلاعات با سرعتی برابر 2400 بیت بر ثانیه انتقال می یابند .

ارتباط سریال UART در میکرو های AVR می تواند به صورت سنکرون و آسنکرون پیکره بندی شود که ما در این کتاب فقط از ارتباط UART به صورت آسنکرون استفاده خواهیم کرد .

در BASCOM ارتباط UART می تواند به دو صورت سخت افزاری و نرم افزاری پیکره بندی شود در پیکره بندی به صورت سخت افزاری از پایه های RXD (دریافت داده) و TXD (ارسال داده) به عنوان پایه های ورودی و خروجی داده سریال استفاده می شود . و در پیکره بندی به صورت نرم افزاری پایه های RXD و TXD به صورت دلخواه توسط برنامه نویس تعیین می شود .

پیکره بندی UART سخت افزاری برای ارسال داده

CONFIG SERIALOUT = BUFFERED,SIZE=X

X مشخص کننده تعداد بایت بافر در نظر گرفته شده برای پورت سریال می باشد و حافظه بافر نیز از حافظه $RAM تعیین می شود .

تعیین میزان BAUD نیز توسط دستور زیر انجام می شود .

$BAUD = X

X مقدار باودی است که در ارتباط سریال استفاده می شود.

ارسال داده به کانال سریال سخت افزاری توسط دستور زیر انجام می گیرد .

PRINT VAR

VAR یک مقدار عددی یا رشته ای است که به کانال سریال ارسال می شود .

همچنین چندین متغیر می تواند فرستاده شود که با علامت ; از هم جدا می شوند .

پیکره بندی UART سخت افزاری برای دریافت داده

CONFIG SERIALIN = BUFFERED,SIZE=X

X نشان دهنده تعداد بایت بافر در نظر گرفته شده برای پورت سریال می باشد .

حافظه بافر از حافظه $RAM تعیین می شود . تعیین میزان BAUD توسط دستور زیر انجام می شود.

$BAUD = X

X مقدار باودی است که در ارتباط سریال استفاده می شود .

دریافت داده از کانال سریال توسط دستور زیر انجام می شود .

INPUT VAR

VAR متغیر رشته ای است که داده دریافت شده درآن قرار می گیرد .

دریافت مقدار باینری داده ارسال شده به پورت سریال توسط دستور زیر انجام می شود .

INPUTBIN VAR

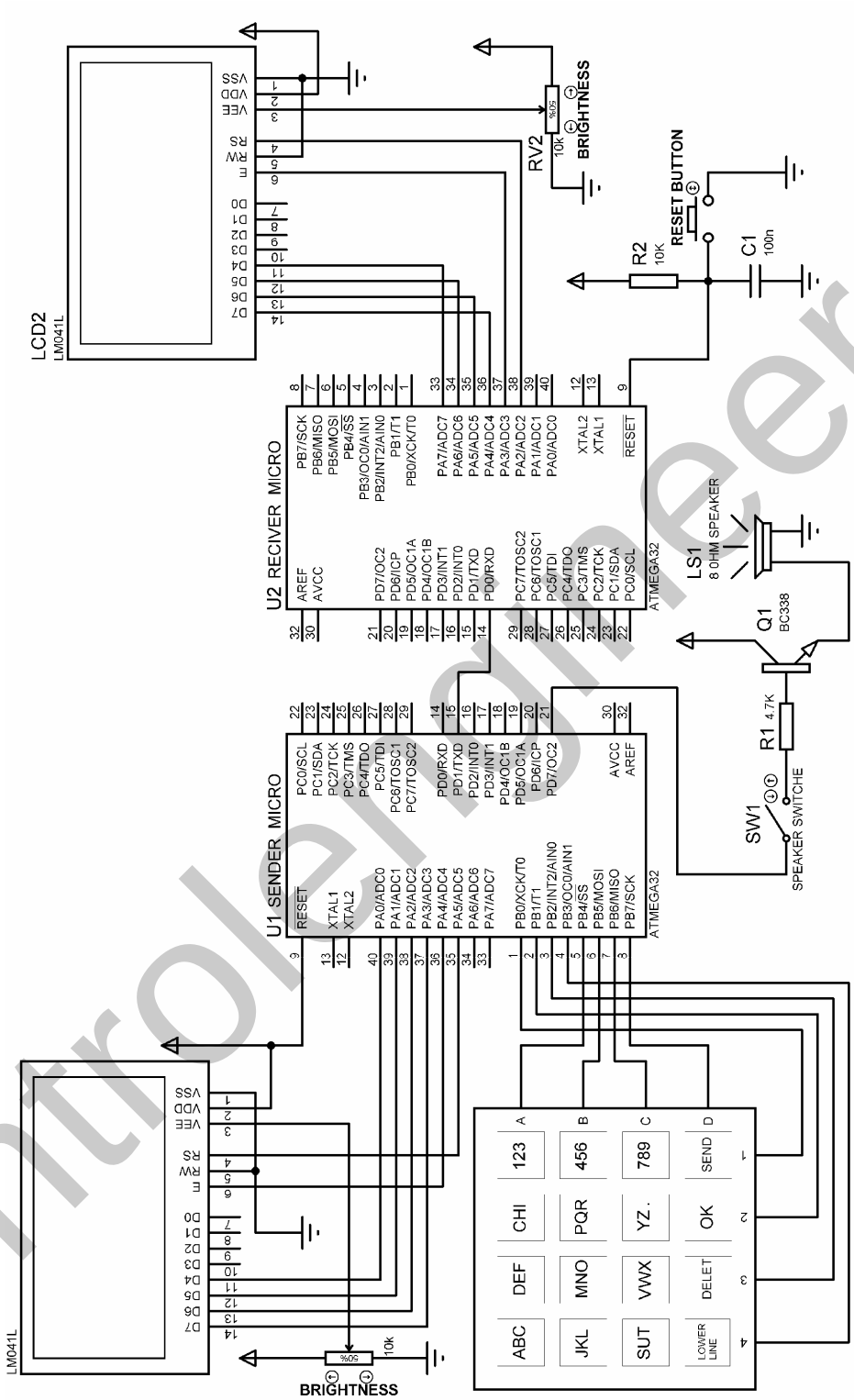
اگر متغیر VAR به عنوان BYTE تعریف شده باشد یک بایت ، به ازاء INTEGER دو بایت و اگر یک آرایه تعریف شود به تعداد آرایه ها کاراکتر دریافت می کند این دستور برای گرفتن تعداد بایت ها در همان خط می ایستد.

پروژه ارسال پیام کوتاه از طریق UART سخت افزاری

در این پروژه ابتدا پیام کوتاه مانند SMS موبایل و توسط KEYPAD ورودی بر روی LCD (16 * 4) نوشته می شود سپس با فشار دکمه SEND پیام نوشته شده از طریق پورت سریال به U2 (گیرنده) ارسال شده و بر روی LCD ، U2 نمایش داده می شود .

توجه داشته باشید که پس از تعیین حرف مورد نظر توسط کلید مربوطه بایستی دکمه OK را فشار دهید . برای رفتن به خط پایین از دکمه LOWER LINE استفاده کنید و برای پاک کردن کاراکتر های نوشته شده دکمه DELET را فشار دهید . برای ایجاد فضای خالی دکمه OK را دو بار فشار دهید .

سخت افزار پروژه در شکل 83-2 ارائه شده است .



شکل 2-83 شماتیک طراحی شده برای ارسال و دریافت پیام کوتاه از طریق پورت سریال

برنامه نوشته شده برای U1 (فرستنده) به صورت زیر می باشد .

```
$regfile = "M32DEF.DAT"
$crystal = 4000000
Config Kbd = Portb , Debounce = 50 , Delay = 100
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3_
                    , E = Pina.4 , Rs = Pina.5
Dim Lcd_data As String * 1 , X As Byte , Y As Byte , Recive_data As Byte
Dim C1 As Byte , C2 As Byte , C3 As Byte , C4 As Byte
Dim C5 As Byte , C6 As Byte , C7 As Byte , C8 As Byte
Dim C9 As Byte , C10 As Byte , C11 As Byte , C0 As Byte
Dim S(81) As String * 1 , Count As Byte , Send As Byte
Dim X1_save As Byte , X2_save As Byte , X3_save As Byte
Config Serialout = Buffered , Size = 200
Config Pind.7 = Output
Config Pind.1 = Output
$baud = 2400
'-----------------------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "PLEASE ENTER"
Locate 2 , 1
Lcd "YOUR MESSAGE "
Wait 1
Cls : Cursor On : Y = 1 : X = 1
'START OF WRITEING MESSAGE ---------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Select Case Recive_data:
'-----------------------------------------------
Case Is = 0
Incr C0
If C0 = 1 Then Lcd_data = "1"
If C0 = 2 Then Lcd_data = "2"
If C0 = 3 Then
C0 = 0 : Lcd_data = "3"
End If
'-----------------------------------------------
Case Is = 4
Incr C4
If C4 = 1 Then Lcd_data = "4"
If C4 = 2 Then Lcd_data = "5"
If C4 = 3 Then
C4 = 0 : Lcd_data = "6"
End If
'-----------------------------------------------
Case Is = 8
Incr C8
If C8 = 1 Then Lcd_data = "7"
If C8 = 2 Then Lcd_data = "8"
If C8 = 3 Then
C8 = 0 : Lcd_data = "9"
End If
'-----------------------------------------------
Case Is = 3
Incr C3
If C3 = 1 Then Lcd_data = "A"
If C3 = 2 Then Lcd_data = "B"
If C3 = 3 Then
```

```
C3 = 0 : Lcd_data = "C"
End If
'-----------------------------------------------
Case Is = 2
Incr C2
If C2 = 1 Then Lcd_data = "D"
If C2 = 2 Then Lcd_data = "E"
If C2 = 3 Then
C2 = 0 : Lcd_data = "F"
End If
'-----------------------------------------------
Case Is = 1
Incr C1
If C1 = 1 Then Lcd_data = "G"
If C1 = 2 Then Lcd_data = "H"
If C1 = 3 Then
C1 = 0 : Lcd_data = "I"
End If
'-----------------------------------------------
Case Is = 7
Incr C7
If C7 = 1 Then Lcd_data = "G"
If C7 = 2 Then Lcd_data = "H"
If C7 = 3 Then
C7 = 0 : Lcd_data = "I"
End If
'-----------------------------------------------
Case Is = 6
Incr C6
If C6 = 1 Then Lcd_data = "M"
If C6 = 2 Then Lcd_data = "N"
If C6 = 3 Then
C6 = 0 : Lcd_data = "O"
End If
'-----------------------------------------------
Case Is = 5
Incr C5
If C5 = 1 Then Lcd_data = "P"
If C5 = 2 Then Lcd_data = "Q"
If C5 = 3 Then
C5 = 0 : Lcd_data = "R"
End If
'-----------------------------------------------
Case Is = 11
Incr C11
If C11 = 1 Then Lcd_data = "S"
If C11 = 2 Then Lcd_data = "T"
If C11 = 3 Then
C11 = 0 : Lcd_data = "U"
End If
'-----------------------------------------------
Case Is = 10
Incr C10
If C10 = 1 Then Lcd_data = "V"
If C10 = 2 Then Lcd_data = "W"
If C10 = 3 Then
C10 = 0 : Lcd_data = "X"
End If
'-----------------------------------------------
Case Is = 9
Incr C9
```

```
If C9 = 1 Then Lcd_data = "Y"
If C9 = 2 Then Lcd_data = "Z"
If C9 = 3 Then
C9 = 0 : Lcd_data = "."
End If
'START OF LOWERLINE BUTTON PROGRAM-----------------------------------------------------------------
Case Is = 15
If S(count) <> Lcd_data Then Lcd_data = " "
Locate Y , X
Lcd Lcd_data
'-------------------------------------------------------------
Incr Count
S(count) = "$"
If Y = 4 Then Goto H2
Lcd_data = " "
If Y = 1 Then X1_save = X
If Y = 2 Then X2_save = X
If Y = 3 Then X3_save = X
Incr Y : X = 1
H2:
'END OF LOWERLINE BUTTON PROGRAM-------------------------------------------------------------------
'START OF DELETE BUTTON PROGRAM---------------------------------------------------------------------
Case Is = 14
S(count) = " "
If Count > 0 Then Decr Count
Decr X
'-------------------------------------------------------------
If Y = 1 Then
If X = 0 Then X = 1
End If
'-------------------------------------------------------------
If Y > 1 Then
If X = 0 Then
If Y = 4 Then X = X3_save
If Y = 3 Then X = X2_save
If Y = 2 Then X = X1_save
Decr Y
End If : End If
'-------------------------------------------------------------
Lcd_data = " "
Locate Y , X
Lcd Lcd_data
'END OF DELETE BUTTON PROGRAM-----------------------------------------------------------------------
'START OF OK BUTTON PROGRAM-------------------------------------------------------------------------
Case Is = 13
Incr Count
S(count) = Lcd_data
Lcd_data = " "
'-------------------------------------------------------------
Incr X
If X > 15 Then
If Y = 1 Then X1_save = 15
If Y = 2 Then X2_save = 15
If Y = 3 Then X3_save = 15
If Y < 4 Then
X = 1 : Incr Y
Else
X = 15
End If : End If
'-------------------------------------------------------------
'END OF OK BUTTON PROGRAM---------------------------------------------------------------------------
```

```
'START OF SEND BUTTON PROGRAM-----------------------------------------------------------------
Case Is = 12
Print "##"
For Send = 1 To Count Step 1
Print S(send)
Next Send
Print "##"
'END OF SEND BUTTON PROGRAM-------------------------------------------------------------------
End Select
'-----------------------------------------------------------------
Locate Y , X
Lcd Lcd_data
Sound Portd.7 , 100 , 80
'-----------------------------------------------------------------
H3:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto H3
'-----------------------------------------------------------------
Goto H1
'END OF WRITEING MESSAGE-----------------------------------------------------------------------
```

تشریح نحوه عملکرد برنامه فرستنده

برای هر کدام از کلید ها یک شمارنده در نظر گرفته شده است که پس از هر بار فشار کلید یک واحد به شمارنده اضافه می شود و اگر بیش از 3 بار کلید را فشار دهیم شمارنده ریست شده و مقدار آن صفر می شود و با توجه به مقدار شمارنده رشته یا حروف مربوطه در متغیر LCD_DATA قرار می گیرد .

رشته مربوطه پس از قرار گرفتن در متغیر LCD_DATA بر روی LCD نمایش داده می شود و با زدن دکمه OK یک واحد به شمارنده COUNT که شمارنده تعداد حروف وارد شده می باشد اضافه می شود . سپس کاراکتر مربوطه در متغیر رشته ای S (COUNT) قرار می گیرد برای مثال اگر مقدار COUNT برابر یک باشد حرف انتخاب شده در متغیر S(1) قرار خواهد گرفت سپس یک واحد به متغیر X که مشخص کننده ستونی است که کرزر یا مکان نما در آن قرار می گیرد اضافه می شود اگر X بزرگتر از 15 بود مقدار Y چک می شود که متغیر Y مشخص کننده سطری است که مکان نما در آن قرار دارد ، اگر Y برابر یک بود X1 _SAVE برابر 15 و اگر Y برابر 2 بود X2_SAVE برابر 15 و اگر Y برابر 3 بود X3_SAVE برابر 15 قرار می گیرد . متغیر های X_SAVE هنگام پاک کردن LCD مورد استفاده قرار می گیرند ، بدین صورت که اگر دکمه DELETE را پشت سر هم فشار دهیم و اولین کاراکتر از یک خط برای مثال خط سوم را هم پاک کنیم با زدن مجدد دکمه DELETE در این حالت مکان نما به X2_SAVE رفته و کاراکتر مر بوط به آن را پاک خواهد کرد .

با زدن دکمه LOWER LINE ابتدا چک می شود که آخرین کاراکتر موجود بر روی LCD توسط دکمه OK پذیرش شده است یا نه ، اگر آخرین کاراکتر OK نشده باشد ابتدا آن را با استفاده از دستورات

```
IF S(COUNT) <> LCD_DATA THE LCD_DATA=" "
LOCATE Y,X
LCD LCD_DATA
```

پاک می کند سپس به منظور رفتن به خط بعدی ابتدا یک واحد به COUNT اضافه کرده و رشته $ را در متغیر S(COUNT) قرار می دهد تا هنگام دریافت رشته $ در گیرنده ، گیرنده متوجه شود که بایستی نوشتن را از خط بعدی ادامه دهد سپس مقدار X در متغیر X_SAVE مربوطه ذخیره می شود تا هنگام پاک کردن صفحه LCD مورد استفاده قرار گیرد . بدین صورت که اگر دکمه DELETE را به صورت پشت سر هم فشار دهیم و اولین کاراکتر از یک خط برای مثال خط سوم را پاک کنیم با زدن مجدد دکمه DELETE مکان نما به X2_SAVE رفته و کاراکتر مربوط به آن را پاک می کند .

عملکرد دکمه DELETE بدین صورت می باشد ابتدا متغیر S(COUNT) برابر با یک رشته خالی می شود ، سپس یک واحد از COUNT کم شده و یک واحد هم از X کم می شود ، اگر X برابر صفر است مقدار Y چک می شود و اگر Y بزرگتر از یک باشد برای مثال برابر با 4 باشد مقدار X برابر با مقدار X2_SAVE قرار خواهد گرفت . سپس یک کاراکتر خالی در مکان X نمایش داده خواهد شد .

نحوه عملکرد دکمه SEND بدین صورت می باشد که ابتدا (##) به عنوان دیتای همزمانی برای گیرنده ارسال می شود که گیرنده از دریافت آن می فهمد که باید شروع کند به دریافت کاراکتر های ارسال شده سپس کاراکتر ها یکی یکی به گیرنده ارسال می شوند پس از ارسال آخرین کاراکتر یک رشته (##) دیگر نیز به عنوان دیتای همزمانی به گیرنده ارسال می شود که گیرنده پس از دریافت آن می فهمد که ارسال کاراکتر ها توسط گیرنده تمام شده و اگر یک بار دیگر علامت (##) را دریافت نماید بایستی صفحه نمایش را پاک کرده و همه کاراکتر ها را دوباره دریافت کند و در قسمت آخر برنامه یعنی بعد از دستور END SELECT ابتدا کاراکتر نوشته شده بر روی LCD نمایش داده شده سپس به منظور حساس به لبه بالا رونده کردن ورودی اجرای برنامه تا زمانی که کلید فشار داده شده غیر فعال نشود در این قسمت متوقف خواهد شد . پس از غیر فعال شدن ورودی اجرای برنامه دوباره به لیبل H1 منتقل می شود .

برنامه نوشته شده برای میکروکنترلر گیرنده به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 4000000
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = Pina.7 , Db5 = Pina.6 , Db6 = Pina.5 , Db7 = Pina.4_
                , E = Pina.3 , Rs = Pina.2
Dim Lcd_data As String * 2 , X As Byte , Y As Byte , Count As Byte
Dim S As String * 1 , Cls_lcd As Bit
Config Serialin = Buffered , Size = 200
Config Pind.0 = Input
$baud = 2400
Enable Interrupts
'-----------------------------------------------------------------------------------------------------------
Y = 1 : X = 1
Cls : Home
Lcd "UART RECIVER"
```

```
'-----------------------------------------------------------------------------------------------------------------
Do
Input Lcd_data
S = Mid(lcd_data , 2 , 1)
If S = "#" Then
Cls : Goto Recive
End If
Loop
'-----------------------------------------------------------------------------------------------------------------
'RECIVEING MESSAGE OF SERIAL PORT-----------------------------------------------------------------------------
Recive:
Input Lcd_data
S = Mid(lcd_data , 2 , 1)
'-----------------------------------------
If Cls_lcd = 1 Then
Cls : Cls_lcd = 0
Y = 1 : X = 1
End If
'-----------------------------------------
Locate Y , X
'-----------------------------------------
If S = "$" Then
Incr Y : X = 1
Goto Recive
End If
'-----------------------------------------
If S = "#" Then
Cls_lcd = 1
Goto Recive
End If
'-----------------------------------------
Lcd S
'-----------------------------------------
Incr X
If X > 15 Then
If Y = 4 Then
Y = 4 : X = 15
Else
X = 1 : Incr Y
End If : End If
'-----------------------------------------
Goto Recive
'END OF RECIVE-------------------------------------------------------------------------------------------------
```

قبل از شروع تشریح نحوه عملکرد برنامه گیرنده به این نکته توجه داشته باشید که درUART سخت افزاری دستور INPUT VAR کاراکتر دریافت شده از پورت سریال را به صورت "H " در متغیر رشته ایVAR قرار می دهید یعنی شما برای دریافت یک کاراکتر بایستی متغیری با طول 2 رشته تعریف کنید و با توجه به این که کاراکتر اول از سمت چپ خالی است کاراکتر دوم را از رشته VAR جدا کرده و به عنوان کاراکتر دریافت شده مورد استفاده قرار می دهیم .

ابتدا عبارت UART RECIVER بر روی LCD نوشته شده سپس اجرای برنامه تا وقتی که رشته گرفته شده برابر با # نباشد متوقف خواهد شد. پس از این که رشته # دریافت شد میکرو کنترلر LCD را پاک کرده و شروع می کند به دریافت کاراکتر های مربوط به MESSAGE ارسال شده .

دستور INPUT تا زمانی که هیچ رشته ای را دریافت نکرده ورودی سریال را چک خواهد کرد و اجرای برنامه به دستور بعدی منتقل نخواهد شد .

پس از دریافت رشته ابتدا بررسی می شود که متغیر CLS_LCD برابر با یک است یا نه اگر برابر یک بود صفحه LCDپاک می شود توجه داشته باشید که متغیر CLS_LCD زمانی یک می شود که دیتای همزمانی پایان ارسال توسط میکروکنترلر گیرنده دریافت شود همان طور که قبلا هم گفته شد دیتای همزمانی ارسال شده برای شروع و پایان برابر "##" بوده و دیتای هم زمانی گرفته شده توسط میکروکنترلر گیرنده برای شروع و پایان برابر "#" می باشد .

میکروی گیرنده با دریافت کاراکتر "$" مکان نما را به خط بعدی منتقل می کند .

پیکره بندی UART نرم افزاری

برای استفاده از UART نرم افزاری ابتدا بایستی یک کانال ار تباطی سریال توسط دستور زیر باز شود .

OPEN "COMPIN:SPEAD,DATA,PARITY,STOPBITS,INVERTED" FOR MOD AS # CHANNEL

PIN : نام پین مورد استفاده برای کانال سریال می باشد .

DATA : تعداد بیت های ارسالی بوده و می تواند هفت یا هشت بیت داده باشد .

PARITY : بیت پریتی می تواند N برای NON ، O برای ODD (فرد) و E برای EVEN (زوج) باشد .

STOPBITS : تعداد بیت های STOP بین دو بایت ارسالی که می تواند یک یا دو بیت باشد .

INVERTED : یک پارامتر اختیاری است و می تواند نوشته نشود ، با نوشتن این دستور خروجی پورت سریال به صورت معکوس خواهد بود .

MOD : می تواند OUTPUT برای (TXD) و INPUT برای (RXD) باشد .

CHANNEL : شماره کانال سریال باز شده می باشد توجه داشته باشید که کاربر می تواند چندین کانال سریال ورودی یا خروجی در یک میکروکنترلر ایجاد کند .

برای بستن هر کدام از کانال ها می توانید از دستور CLOSE #CHANNEL استفاده کنید. CHANNEL شماره کانال باز شده می باشد .

برای ارسال داده به کانال سریال نرم افزاری از دستور زیر استفاده می شود .

PRINT #CHANNEL,VAR

VAR مقدار یا متغیر رشته ای یا عددی است که به کانال سریال شماره CHANNEL ارسال می شود .

برای ارسال مقدار باینری داده به کانال سریال نرم افزاری از دستور زیر استفاده می شود .

PRINTBIN #CHANNEL,VAR

VAR متغیری است که مقدار آن ابتدا به باینری تبدیل شده سپس به کانال شماره CHANNEL سریال ارسال می شود .

برای دریافت داده از کانال سریال نرم افزاری از دستور زیر استفاده می شود .

INPUT #CHANNEL,VAR

VAR متغیر رشته ای است که از کانال سریال نرم افزاری با شماره CHANNEL دریافت می شود .

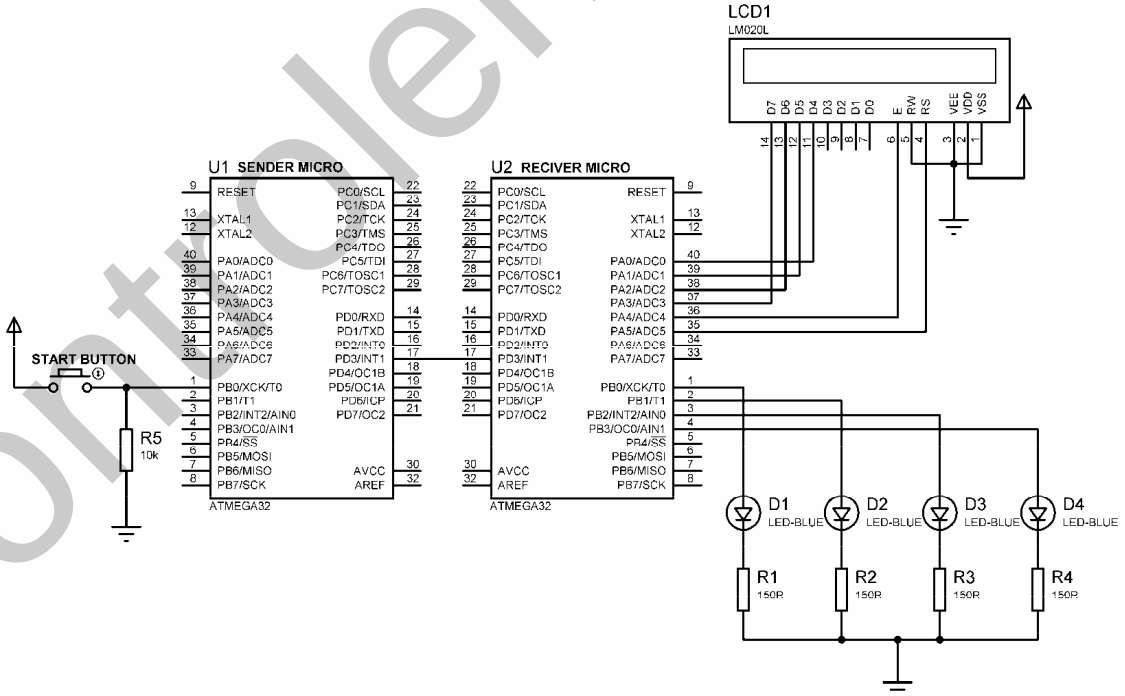
برای در یافت مقدار باینری از کانال سریال از دستور زیر استفاده می شود .

INPUTBIN #CHANNEL,VAR

VAR یک متغیر عددی است که مقدار باینری گرفته شده در آن قرار می گیرد .

نحوه استفاده از UART نرم افزاری

در پروژه ی زیر از ار تباط UART نرم افزاری استفاده شده نحوه عملکرد پروژه بدین صورت می باشد که با زدن کلید START BUTTON در فرستنده میکروکنترلر ، فرستنده شروع به ارسال داده از طریق UART نرم افزاری به میکروکنترلر گیرنده می کند . سخت افزار پروژه در شکل 2-84 ارائه شده است .



شکل 2-84 شماتیک طراحی شده برای استفاده از UART نرم افزاری

برنامه نوشته شده برای میکروکنترلر فرستنده به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 4000000
Config Pinb.0 = Input
Config Pind.3 = Output
Dim Send_data As String * 12 , H As Byte
Dim Start_button As Byte
'-----------------------------------------------------------------------------
Starting_send:
Start_button = Pinb.0
If Start_button = 0 Then Goto Starting_send
'-----------------------------------------------------------------------------
Open "COMD.3:2400,8,N,1,INVERTED" For Output As #1
'-----------------------------------------------------------------------------
Print #1 , "HELLO"
Wait 1
Print #1 , "MY NAME IS"
Wait 1
Send_data = "HOSSEIN"
Print #1 , Send_data
Wait 1
Send_data = "START VALUE"
Print #1 , Send_data
'-----------------------------------------------------------------------------
For H = 0 To 15 Step 1
Print #1 , H
Wait 1
Next H
'-----------------------------------------------------------------------------
Send_data = "STOP"
Print #1 , Send_data
Close #1
Stop
'-----------------------------------------------------------------------------
```

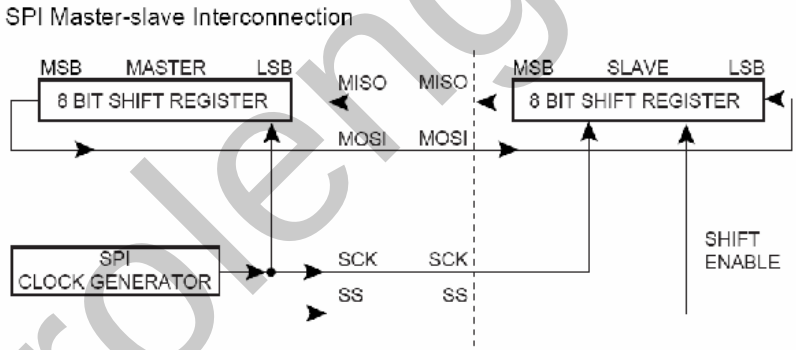
برنامه نوشته شده برای میکروکنترلر گیرنده به صورت زیر می باشد .

```
$regfile = "M32DEF.DAT"
$crystal = 4000000
Config Pind.3 = Input
Config Portb = Output
Config Lcd = 16 * 1
Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3_
  , E = Pina.4 , Rs = Pina.5
Dim Recive_data As String * 12 , H As Byte
Declare Sub Input_value
Open "COMD.3:2400,8,N,1,INVERTED" For Input As #1
Cursor Off
Cls : Home
Lcd "STARTING RECIVE"
'-----------------------------------------------------------------------------------------------
Starting_recive:
Input #1 , Recive_data
If Recive_data = "START VALUE" Then Call Input_value
Cls : Home
Lcd Recive_data
Goto Starting_recive
'-----------------------------------------------------------------------------------------------
Sub Input_value:
```

```
Do
Input #1 , Recive_data
If Recive_data = "STOP" Then
Cls : Home
Lcd "STOP"
Stop
End If
H = Val(recive_data)
Cls : Home
Lcd H
Portb = H
Loop
End Sub Input_value
Return
'-----------------------------------------------------------------------------------------------------------
```

ارتباط سریال SPI

ارتباط سریال SPI یک پروتکل ارتباطی سریال (3_WIRE) با سرعت بالا می باشد که به صورت MASTER/SLAVE پیکره بندی می شود . از خصوصیات بارز این نوع ارتباط سریال می توان به پرچم وقفه اتمام سریال ، بیدار شدن از حالت بیکاری IDLE ، ارتباط به صورت سنکرون ، ارسال هم زمان داده از MASTER به SLAVE و از SLAVE به MASTER اشاره کرد . نحوه برقراری اتصالات بین میکروکنترلرهای MASTER ، SLAVE در شکل 2-85 ارائه شده است .



شکل 2-85 نحوه برقراری اتصالات بین میکروکنترلر های MASTER ، SLAVE

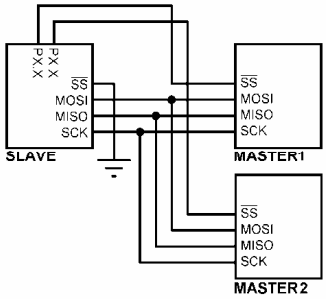
پایه MOSI (MASTER OUT SLAVE IN) خروجی داده برای MASTER و ورودی داده برای SLAVE می باشد .

پایه MISO (MASTER IN SLAVE OUT) و ورودی داده برای MASTER و خروجی داده برای SLAVE می باشد .

پایه SCK خروجی کلاک برای MASTER و ورودی کلاک برای SLAVE می باشد .

پایه SS در MASTER :

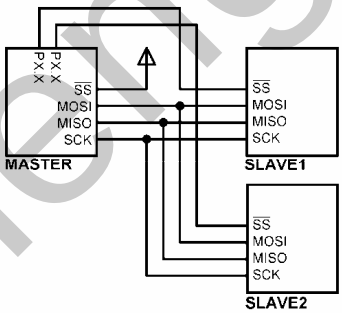
اگر SS خروجی تعریف شده باشد هیچ تاثیری در SPI ندارد ولی اگر ورودی تعریف شده باشد برای فعال شدن میکروکنترلر MASTER حتما بایستی بالا باشد از این پایه می توان برای انتخاب MASTER مورد نظر توسط SLAVE استفاده کرد . شکل2-86 چگونگی این عملکرد را نشان می دهد .



شکل2-86 چگونگی انتخاب MASTER مورد نظر توسط میکروکنترلر SLAVE

پایه SS در SLAVE :

پایه SS در SLAVE همیشه ورودی است هنگامی که پایه SS پایین باشد میکروکنترلر SLAVE فعال خواهد بود . از این حالت می توان برای انتخاب SLAVE مورد نظر توسط MASTER استفاده کرد . شکل2-87 چگونگی این عملکرد را نشان می دهد .



شکل2-87 چگونگی انتخاب SLAVE موردنظر توسط میکروکنترلر MASTER

در حالتی که فقط از یک MASTER و یک SLAVE استفاده می کنید می توانید پایه های SS در MASTER وSLAVE را به هم وصل کنید .

SPI را نیز مانند UART می توان به دو صورت سخت افزاری و نرم افزاری پیکره بندی کرد در حالت سخت افزاری از پایه های پیش فرض برای ارتباط سریال SPI استفاده می شود ولی در حالت نرم افزاری این پایه ها توسط کاربر تعیین می شود . که در این کتاب فقط از حالت SPI سخت افزاری استفاده شده است .

پیکره بندی سخت افزاری SPI

CONFIG SPI = HARD,INTERRUPT=ON/OFF,DATA ORDER=LSB/MSB,

MASTER=YES/NO,POLARITY=HIGH/LOW,PHASE=0/1,CLOCKRATE=4/16/64/128,NOSS=0/1

INTERRUPT=ON/OFF : در صورت استفاده از وقفه در ارتباط SPI از گزینه ON استفاده می شود . پس از ارسال کامل داده کلاک SPI از PIN ، SCK قطع و پرچم وقفه SPI فعال می شود .

DATAORDER=LSB/MSB : در صورت انتخاب LSB ، ابتدا LSB و سپس MSB داده ارسال خواهد شد و در صورت انتخاب MSB ابتدا MSB و سپس LSB داده ارسال خواهد شد .

MASTER=YES/NO : اگر میکرویی که در حال برنامه نویسی برای آن هستیم MASTER باشد گزینه YES و اگر SLAVE باشد گزینه NO را انتخاب می کنم .

PHASE=0/1 : تعداد بیت ارسالی بین دو بایت داده (STOPBIT) را مشخص می کند بهتر است گزینه صفر انتخاب شود .

POLARITY=HIGH/LOW : اگر بخواهیم زمانی که SPI در حالت بیکاری (IDLE) است پایه کلاک بالا باشد، گزینه HIGH انتخاب می شود . انتخاب LOW باعث پایین قرار گرفتن پایه کلاک می شود .

CLOCK RATE : مشخص کننده فرکانس کلاک SPI بوده و می تواند 1/128 ، 1/64 ، 1/16 ، 1/14 فرکانس سیستم باشد .

NOSS=0/1 : زمانی که در حالت MASTER نمی خواهید سیگنال SS ایجاد شود ، 1 را انتخاب کنید . در این حالت کاربر بایستی نرم افزاری پایه SS در SLAVE مورد نظر را پایین کند .

پیکره بندی سخت افزاری را می توان نیز به صورت ساده زیر نیز نوشت .

CONFIG SPI = HARD

که در این حالت به صورت پیش فرض اول MSB فرستاده می شود و POLARITY= HIGH ، MASTER=YES ، PHASE=0 ، CLOCKRATE=4 در نظر گرفته می شود .

دستورات مربوط به ارتباط SPI

دستور SPIINIT برای فعال سازی SPI استفاده می شود .

دستور زیر به تعداد X بایت از باس SPI دریافت می کند و در متغیر عددی VAR قرار می دهد .

SPIIN VAR,X

دستور زیر به تعداد X بایت داده VAR را به باس SPI ارسال می کند .

SPIOUT VAR,X

دستور زیر متغیر یا ثابت VAR را به باس SPI ارسال کرده داده دریافت شده از باس SPI را به صورت هم زمان در متغیر BYTE قرار می دهد .

VAR=SPIMORE(BYTE)

پروژه تبادل اطلاعات تمام دو طرفه (FULL DUPLEX)

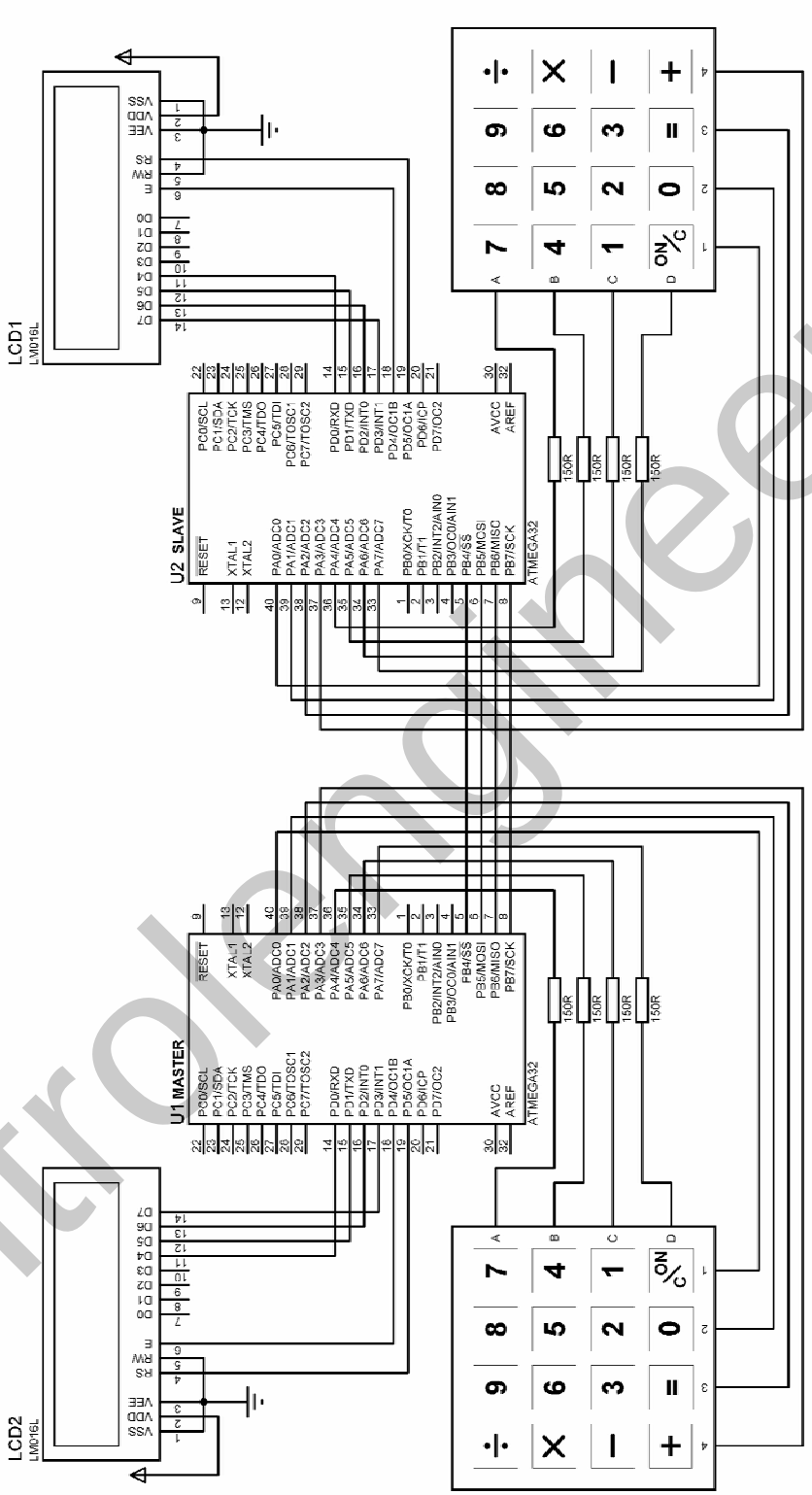
پروژه زیر به منظور آشنایی بیشتر در رابطه با تبادل اطلاعات دو طرفه طرح شده است به طوری که در هر لحظه اطلاعات MASTER به SLAVE و اطلاعات SLAVE به MASTER ارسال می شود به عبارتی دیگر در هر لحظه محتوای رجیستر 8 بیتی MASTER و SLAVE با هم عوض می شود .

عملکرد پروژه بدین صورت است که عدد فشار داده شده در کی پد در MASTER در LCDمیکروکنترلر SLAVE نشان داده می شود و عدد فشار داده شده در کی پد SLAVE در LCD میکروکنترلر MASTER نشان داده می شود .

سخت افزار پروژه در شکل 88-2 ارائه شده است .

برنامه نوشته شده برای میکروکنترلر MASTER به صورت زیر می باشد .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Spi = Hard , Interrupt = Off , Data Order = Lsb , Master = Yes , _
Polarity = High , Phase = 1 , Clockrate = 128
Config Kbd = Porta
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
            , E = Pind.4 , Rs = Pind.5
Dim Keypad_data As Byte , Spi_in As Byte , Spi_send As Byte
'-----------------------------------------------------------------------------------------------------------
Spiinit : Cursor Off
Cls : Home
Lcd "MASTER MICRO"
'-----------------------------------------------------------------------------------------------------------
Starting:
Keypad_data = Getkbd()
If Keypad_data = 16 Then Spi_send = 100
Spi_send = Lookup(keypad_data , Data_code)
If Spi_send >= 10 Then Spi_send = 100
Spi_in = Spimove(spi_send)
If Spi_in = 100 Then Goto Starting
Cls : Home
Lcd "MASTER MICRO"
Lowerline
Lcd Spi_in
H1:
Keypad_data = Getkbd()
If Keypad_data < 16 Then Goto H1
Goto Starting
'-----------------------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'-----------------------------------------------------------------------------------------------------------
```



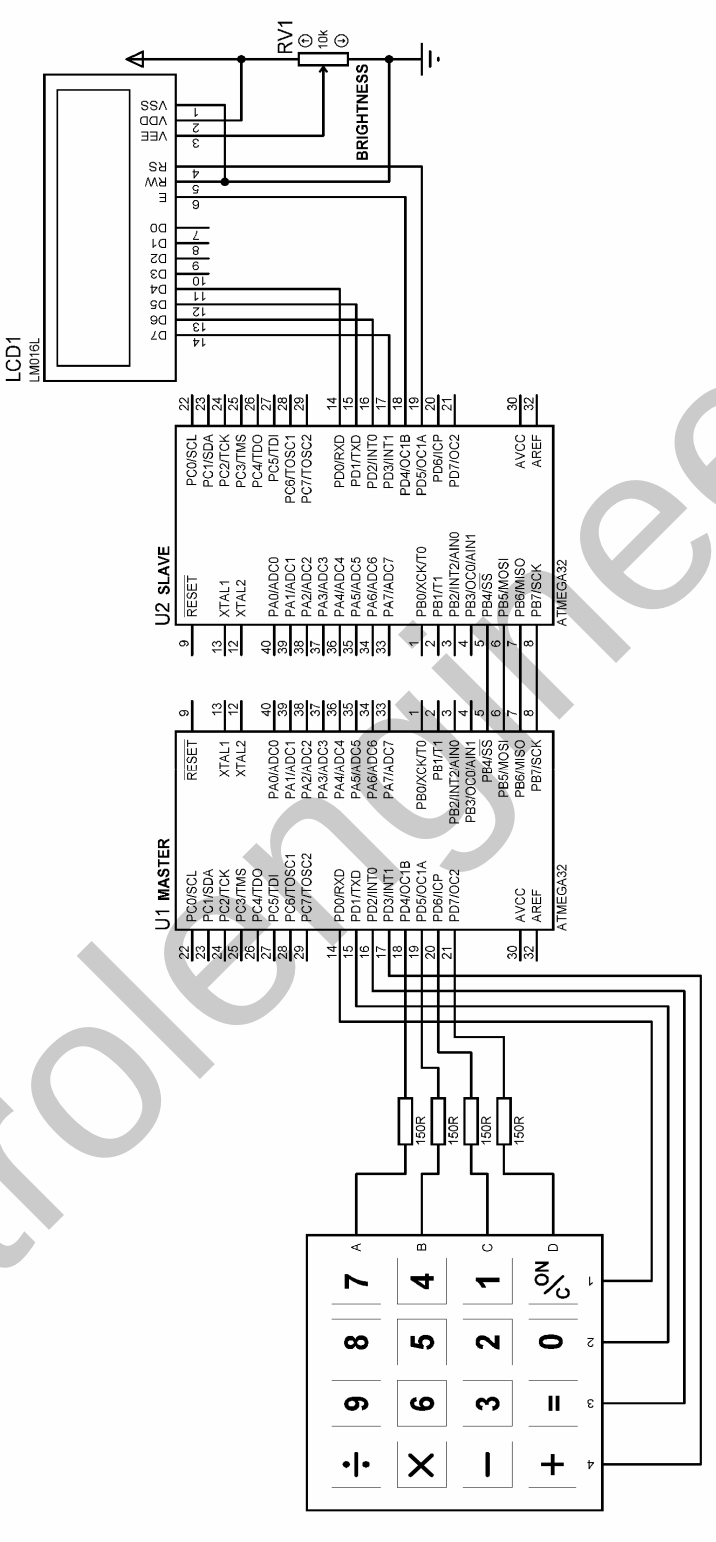
شکل 2-88 شماتیک طراحی شده برای پروژه تبادل اطلاعات تمام دو طرفه (FULL DUPLEX)

برنامه نوشته شده برای میکروکنترلر SLAVE به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Config Spi = Hard , Interrupt = Off , Data Order = Lsb , Master = No , _
Polarity = High , Phase = 0 , Clockrate = 128
Config Kbd = Porta
Dim Keypad_data As Byte , Spi_in As Byte , Spi_send As Byte
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
                , E = Pind.4 , Rs = Pind.5
'-----------------------------------------------------------------------------------------------------------
Spiinit : Cursor Off
Cls : Home
Lcd "SLAVE MICRO"
'-----------------------------------------------------------------------------------------------------------
Starting:
Keypad_data = Getkbd()
If Keypad_data = 16 Then Spi_send = 100
Spi_send = Lookup(keypad_data , Data_code)
If Spi_send >= 10 Then Spi_send = 100
Spi_in = Spimove(spi_send)
If Spi_in = 100 Then Goto Starting
Cls : Home
Lcd "SLAVE MICRO"
Lowerline
Lcd Spi_in
H1:
Keypad_data = Getkbd()
If Keypad_data < 16 Then Goto H1
Goto Starting
'-----------------------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'-----------------------------------------------------------------------------------------------------------
```

پروژه ارسال اطلاعات به صورت یک طرفه از طریق باس SPI

در پروژه زیر کلید فشار داده شده در کی پد MASTER به میکروکنترلر SLAVE ارسال شده و بر روی LCD نمایش داده می شود در این پروژه ارتباط به صورت یک طرفه می باشد .وسخت افزار پروژه در شکل 2-89 ارائه شده است .



شکل 89-2 شماتیک طراحی شده برای ارتباط یک طرفه SPI

برنامه نوشته شده برای میکروکنترلر MASTER به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Config Spi = Hard , Interrupt = Off , Data Order = Lsb , Master = Yes , _
Polarity = High , Phase = 1 , Clockrate = 128
Config Kbd = Porta
Dim Keypad_data As Byte , Spi_send As Byte
Spiinit
'-----------------------------------------------------------------------------------------------------------
Starting:
Keypad_data = Getkbd()
If Keypad_data = 16 Then Goto Starting
Spi_send = Lookup(keypad_data , Data_code)
If Spi_send >= 10 Then Goto Starting
Spiout Spi_send , 1
H1:
Keypad_data = Getkbd()
If Keypad_data < 16 Then Goto H1
Goto Starting
'-----------------------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 30 , 40 , 0 , 50 , 60
'-----------------------------------------------------------------------------------------------------------
```

برنامه نوشته شده برای میکروکنترلر SLAVE به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Config Spi = Hard , Interrupt = Off , Data Order = Lsb , Master = No , _
Polarity = High , Phase = 0 , Clockrate = 128
Dim Spi_recive As Byte
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
                , E = Pind.4 , Rs = Pind.5
'-----------------------------------------------------------------------------------------------------------
Spiinit : Cursor Off
Cls : Home
Lcd "SLAVE MICRO"
'-----------------------------------------------------------------------------------------------------------
Starting:
Spiin Spi_recive , 1
Cls : Home
Lcd "SLAVE MICRO"
Lowerline
Lcd Spi_recive
Goto Starting
'-----------------------------------------------------------------------------------------------------------
```

هنگام استفاده از وقفه SPI زمانی که انتقال داده کامل شود وقفه SPI روی می دهد برای مثال در پروژه قبل برنامه SLAVE را می توان به منظور استفاده از وقفه SPI به شکل زیر تغییر داد در این حالت اجرای برنامه نوشته شده در داخل DO_LOOP روال عادی خود را دارد و زمانی که MASTER داده ای به SLAVE ارسال می کند SLAVE برنامه را موقتا قطع کرده و به وقفه پاسخ می دهد و داده را از باس SPI می گیرد ، سپس با بازگشت از وقفه اجرای برنامه ادامه پیدا می کند.

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
```

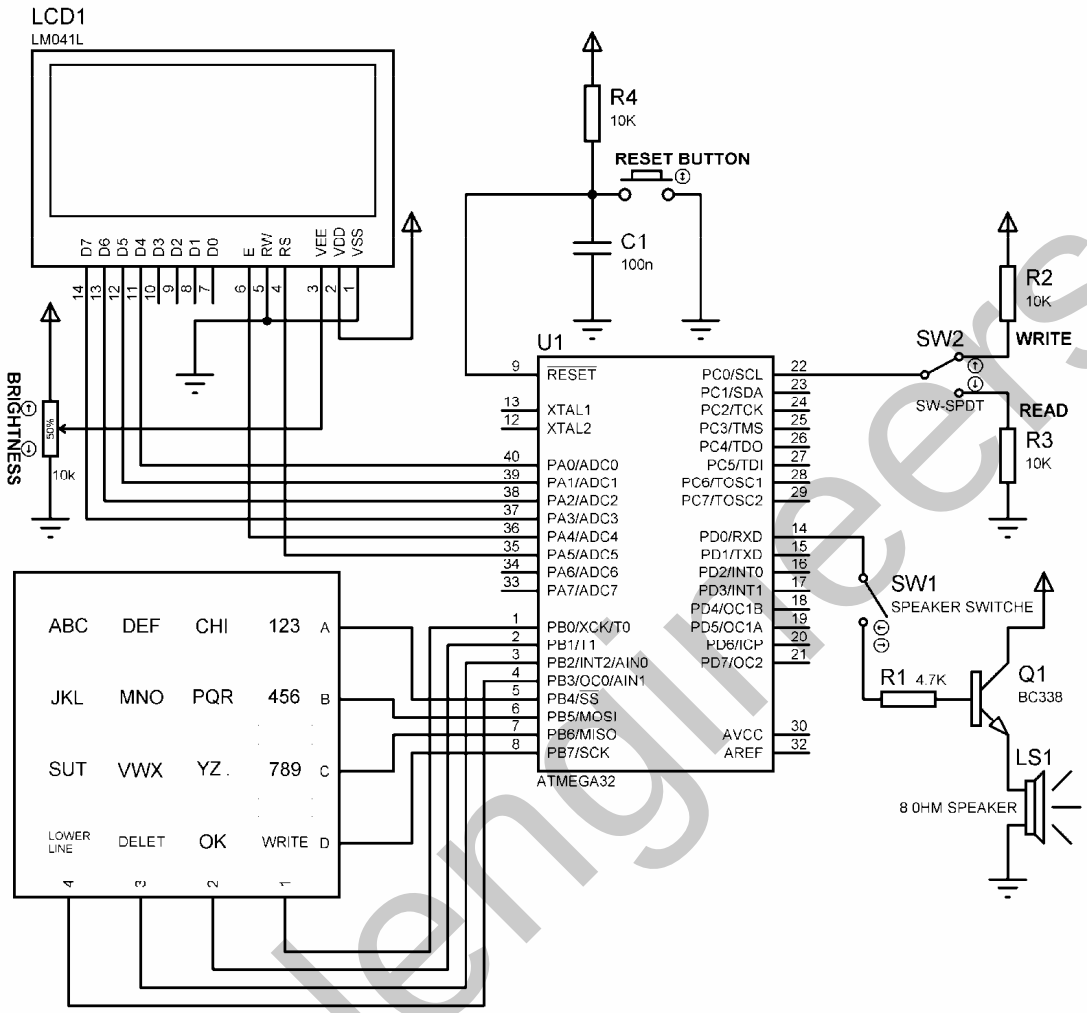
```
Config Spi = Hard , Interrupt = On , Data Order = Lsb , Master = No , _
Polarity = High , Phase = 0 , Clockrate = 128
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
                , E = Pind.4 , Rs = Pind.5
Dim Spi_recive As Byte
Enable Interrupts
Enable Spi
On Spi Spi_isr
Config Pind.6 = Output
'-----------------------------------------------------------------------------------------------------------
Spiinit : Cursor Off
Cls : Home
Lcd "SLAVE MICRO"
Do
'YOU CAN WRITE YOUR PROGRAM HERE.
Loop
'-----------------------------------------------------------------------------------------------------------
Spi_isr:
Disable Interrupts
Spiin Spi_recive , 1
Cls : Home
Lcd "SLAVE MICRO"
Lowerline
Lcd Spi_recive
Enable Interrupts
Return
'-----------------------------------------------------------------------------------------------------------
```

کار با حافظه EEPROM داخلی میکرو AVR

محتویات ذخیره شده در حافظه EEPROM با قطع تغذیه میکرو از بین نمی رود . به همین دلیل می توان برای ذخیره اطلاعات از آن ها استفاده نمود برای نوشتن و خواندن از حافظه ها EEPROM کافی است متغیری که داده را در آن ذخیره می کنیم از نوع EPROM تعریف شده باشد برای درک بهتر این موضوع پروژه زیر را در نظر بگیرید .

پروژه ذخیره پیام کوتاه در EEPROM داخلی AVR

دراین پروژه برای نوشتن پیام بر روی LCD بایستی ابتدا کلید SW2 را در حالت WRITE قرار داده سپس پیام مورد نظر را توسط KEYPAD وارد کنیم پس از وارد کردن پیام با زدن دکمه WRITE پیام نوشته شده در EEPROM داخلی میکرو ذخیره می شود . پس از ذخیره پیام کلید SW2 را در حالت READ قرار داده و دکمه RESET را فشار دهید در این حالت پیام ذخیره شده بر روی LCD نمایش داده می شود . سخت افزار پروژه در شکل 90-2 ارائه شده است .



شکل 90-2 شماتیک طراحی شده برای ذخیره پیام کوتاه در EEPROM داخلی AVR

برنامه نوشته شده برای پروژه به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Config Kbd = Portb , Debounce = 50 , Delay = 100
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3_
                , E = Pina.4 , Rs = Pina.5
Dim Lcd_data As String * 1 , X As Byte , Y As Byte , Recive_data As Byte
Dim C1 As Byte , C2 As Byte , C3 As Byte , C4 As Byte
Dim C5 As Byte , C6 As Byte , C7 As Byte , C8 As Byte
Dim C9 As Byte , C10 As Byte , C11 As Byte , C0 As Byte
Dim S(81) As String * 1 , Count As Byte , Write_e2 As Byte
Dim X1_save As Byte , X2_save As Byte , X3_save As Byte
Dim Save(81) As Eram String * 1 , Lcd_write As String * 1
Dim E2_count As Eram Byte , Read_e2 As Byte
Config Pind.0 = Output
Config Pinc.0 = Input
```

```
'READ OR WRITE?-----------------------------------------------------------------------------------------
If Pinc.0 = 0 Then Goto Read_message
'-------------------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "PLEASE ENTER"
Locate 2 , 1
Lcd "YOUR MESSAGE "
Wait 4
Cls : Cursor On : Y = 1 : X = 1
'-------------------------------------------------------------------------------------------------------
'START OF WRITEING MESSAGE-------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Select Case Recive_data:
'-----------------------------------------------------
Case Is = 0
Incr C0
If C0 = 1 Then Lcd_data = "1"
If C0 = 2 Then Lcd_data = "2"
If C0 = 3 Then
C0 = 0 : Lcd_data = "3"
End If
'-----------------------------------------------------
Case Is = 4
Incr C4
If C4 = 1 Then Lcd_data = "4"
If C4 = 2 Then Lcd_data = "5"
If C4 = 3 Then
C4 = 0 : Lcd_data = "6"
End If
'-----------------------------------------------------
Case Is = 8
Incr C8
If C8 = 1 Then Lcd_data = "7"
If C8 = 2 Then Lcd_data = "8"
If C8 = 3 Then
C8 = 0 : Lcd_data = "9"
End If
'-----------------------------------------------------
Case Is = 3
Incr C3
If C3 = 1 Then Lcd_data = "A"
If C3 = 2 Then Lcd_data = "B"
If C3 = 3 Then
C3 = 0 : Lcd_data = "C"
End If
'-----------------------------------------------------
Case Is = 2
Incr C2
If C2 = 1 Then Lcd_data = "D"
If C2 = 2 Then Lcd_data = "E"
If C2 = 3 Then
C2 = 0 : Lcd_data = "F"
End If
'-----------------------------------------------------
Case Is = 1
Incr C1
If C1 = 1 Then Lcd_data = "G"
If C1 = 2 Then Lcd_data = "H"
```

```
If C1 = 3 Then
C1 = 0 : Lcd_data = "I"
End If
'-----------------------------------------------
Case Is = 7
Incr C7
If C7 = 1 Then Lcd_data = "G"
If C7 = 2 Then Lcd_data = "H"
If C7 = 3 Then
C7 = 0 : Lcd_data = "I"
End If
'-----------------------------------------------
Case Is = 6
Incr C6
If C6 = 1 Then Lcd_data = "M"
If C6 = 2 Then Lcd_data = "N"
If C6 = 3 Then
C6 = 0 : Lcd_data = "O"
End If
'-----------------------------------------------
Case Is = 5
Incr C5
If C5 = 1 Then Lcd_data = "P"
If C5 = 2 Then Lcd_data = "Q"
If C5 = 3 Then
C5 = 0 : Lcd_data = "R"
End If
'-----------------------------------------------
Case Is = 11
Incr C11
If C11 = 1 Then Lcd_data = "S"
If C11 = 2 Then Lcd_data = "T"
If C11 = 3 Then
C11 = 0 : Lcd_data = "U"
End If
'-----------------------------------------------
Case Is = 10
Incr C10
If C10 = 1 Then Lcd_data = "V"
If C10 = 2 Then Lcd_data = "W"
If C10 = 3 Then
C10 = 0 : Lcd_data = "X"
End If
'-----------------------------------------------
Case Is = 9
Incr C9
If C9 = 1 Then Lcd_data = "Y"
If C9 = 2 Then Lcd_data = "Z"
If C9 = 3 Then
C9 = 0 : Lcd_data = "."
End If
'START OF LOWERLINE BUTTON PROGRAM-----------------------------------------------------------------
Case Is = 15
If S(count) <> Lcd_data Then Lcd_data = " "
Locate Y , X
Lcd Lcd_data
'-------------------------------------------------------------
Incr Count
S(count) = "$"
If Y = 4 Then Goto H2
Lcd_data = " "
```

```
If Y = 1 Then X1_save = X
If Y = 2 Then X2_save = X
If Y = 3 Then X3_save = X
Incr Y : X = 1
H2:
'END OF LOWERLINE BUTTON PROGRAM-----------------------------------------------------------------
'START OF DELETE BUTTON PROGRAM-------------------------------------------------------------------
Case Is = 14
S(count) = " "
Lcd_data = " "
Locate Y , X
Lcd Lcd_data
If Count > 0 Then Decr Count
Decr X
'-----------------------------------------
If Y = 1 Then
If X = 0 Then X = 1
End If
'-----------------------------------------
If Y > 1 Then
If X = 0 Then
If Y = 4 Then X = X3_save
If Y = 3 Then X = X2_save
If Y = 2 Then X = X1_save
Decr Y
End If : End If
'---------------------------------------
Lcd_data = " "
Locate Y , X
Lcd Lcd_data
'END OF DELET BUTTON PROGRAM-----------------------------------------------------------------------
'START OF OK BUTTON PROGRAM------------------------------------------------------------------------
Case Is = 13
Incr Count
S(count) = Lcd_data
Lcd_data = " "
Incr X
If X > 15 Then
If Y = 1 Then X1_save = 15
If Y = 2 Then X2_save = 15
If Y = 3 Then X3_save = 15
If Y < 4 Then
X = 1 : Incr Y
Else
X = 15
End If : End If
'END OF OK BUTTON PROGRAM--------------------------------------------------------------------------
'START OF  WRITEIG IN EEPROM-----------------------------------------------------------------------
Case Is = 12
Cls : Home
Lcd "WRITEING YOUR"
Locate 2 , 1
Lcd "MESSAGE IN"
Locate 3 , 1
Lcd "EEPROM"
E2_count = Count
Waitms 4
For Write_e2 = 1 To Count Step 1
Save(write_e2) = S(write_e2)
Waitms 4
Next Write_e2
```

```
Stop
'END OF WRITEING IN EEPROM-----------------------------------------------------------------------------
End Select
Locate Y , X
Lcd Lcd_data
Sound Portd.0 , 100 , 80
'-----------------------------------------------------
H3:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto H3
'-----------------------------------------------------
Goto H1
'END OF WRITEING MESSAGE -------------------------------------------------------------------------------
'READING MESSAGE OF INTERNAL EEPROM-----------------------------------------------------------------
Read_message:
Cls : Cursor On : Y = 1 : X = 1
Count = E2_count
For Read_e2 = 1 To Count Step 1
Lcd_write = Save(read_e2)
'-----------------------------------------------------
Locate Y , X
'-----------------------------------------------------
If Lcd_write = "$" Then
Incr Y : X = 1
Goto H4
End If
'-----------------------------------------------------
Lcd Lcd_write
'-----------------------------------------------------
Incr X
If X > 15 Then
If Y = 4 Then
Y = 4 : X = 15
Else
X = 1 : Incr Y
End If : End If
'-----------------------------------------------------
H4:
Next Read_e2
Stop
'END OF READING-----------------------------------------------------------------------------------------
```

نحوه نوشتن پیام بر روی LCD قبلا توضیح داده شده است به همین دلیل در این قسمت از توضیح آن صرفه نظر می شود . نکته مهم این است که پس از هر بار نوشتن در متغیری که محتوای آن در EEPROM ذخیره می شود بایستی برای تکمیل عملیات نوشتن 4ms تاخیر ایجاد شود .

وقفه های خارجی و نحوه پیکره بندی آنها در محیط BASCOM

میکروکنترلر های استفاده شده در این کتاب حداکثر دارای 3 وقفه EXTERNAL (خارجی) می باشند که وقفه INT0 و INT1 می توانند حساس به لبه یا سطح پیکره بندی شوند ولی وقفه INT2 فقط می تواند در لبه بالا رونده یا پایین رونده پالس تریگر ورودی تحریک شود . نحوه تریک شدن وقفه های خارجی در BASCOM توسط دستور زیر تعیین می شود .

CONFIG INTX=LOWLEVEL/FALLING/RISING

LOW LEVEL : حساس به سطح پایین یا صفر پالس ورودی

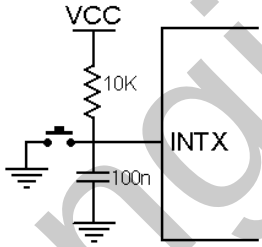
FALLING : حساس به لبه پایین رونده پالس ورودی

RISSING : حساس به لبه بالا رونده پالس ورودی

توجه داشته باشید که پس از نوشتن دستور فوق وقفه مورد نظر بایستی توسط دستورات

```
ENABLE INTERRUPTS
ENABLE INTX
ON INTX LABLE
```

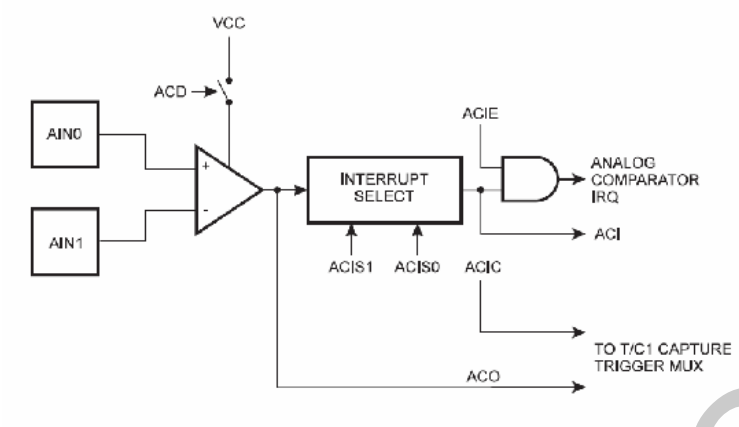
فعال شود ، در این حالت با اعمال یک پالس با لبه یا سطح تعیین شده به پایه وقفه مربوطه ، وقفه فعال شده و به برچسب LABLE پرش می کند . لازم به ذکر است که بازگشت از ISR وقفه نیز توسط دستور RETURN انجام می گیرد اگر از کلید برای اعمال پالس تحریک وقفه استفاده می کنید می توانید از سخت افزارشکل 2-91 استفاده کنید .



شکل 2-91 نحوه استفاده از کلید فشاری برای اعمال پالس تریگر وقفه

مقایسه کننده آنالوگ (ANALOG COMPARATOR)

مقایسه کننده آنالوگ مقدار پایه ورودی مثبت (AIN0) را با مقدار پایه ورودی منفی (AIN1) مقایسه می کند در صورتی که مقدار ولتاژ پایه مثبت ، بیشتر از پایه منفی باشد خروجی مقایسه کننده HIGH و در غیر این صورت خروجی LOW می شود خروجی این مقایسه کننده می تواند برای تحریک ورودی CAPTURE یا واحد دریافت ورودی تایمر/کانتر یک به کار رود ، مقایسه کننده هم چنین دارای یک پرچم وقفه مجزاست کاربر می تواند نحوه تریگر شدن وقفه خروجی مقایسه کننده را در لبه بالا رونده ، پایین رونده یا TOGGLE انتخاب کند. شکل 2-92 بلوک دیاگرام مقایسه کننده آنالوگ را نشان را می دهد.



شکل 92-2 بلوک دیاگرام داخلی مقایسه کننده آنالوگ

پیکره بندی مقایسه کننده آنالوگ در BASCOM

CONFIG ACI=ON/OFF,COMPARE=ON/OFF,TRIGGER=TOGGLE/RISING/FALLING

ON/OFF : هنگام استفاده از مقایسه کننده آنالوگ گزینه ON را انتخاب کنید .

COMPARE=ON/OFF : زمانی که بخواهید خروجی مقایسه کننده ورودی CAPTURE تایمر/کانتر یک را تریگ کند . گزینه ON و در غیر این صورت گزینه OFF را انتخاب کنید .

TRIGGER = TOGGLE/RISING/FALLING

مقایسه کننده آنالوگ دارای پرچم وقفه (ANALOG COMPARATOR) مجزاست که توسط گزینه فوق تحریک می شود .در صورت انتخاب گزینه FALLING یک لبه پایین رونده در خروجی مقایسه کننده باعث یک شدن پرچم وقفه می شود .

در صورت انتخاب گزینه RISING یک لبه بالا رونده در خروجی مقایسه کننده باعث یک شدن پرچم وقفه می شود .

در صورت انتخاب گزینه TOGGLE معکوس شدن خروجی مقایسه کننده باعث یک شدن پرچم وقفه می شود .

مثال زیر نحوه کار با وقفه مقایسه کننده آنالوگ را نشان می دهد.

در این مثال یک لبه بالارونده در خروجی مقایسه کننده آنالوگ باعث رخ دادن وقفه مربوطه شده و با هر بار رخ دادن وقفه وضعیت منطقی PORTB.0 در ISR وقفه معکوس می شود .

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
Config Aci = On , Compare = Off , Trigger = Rising
Config Pinb.0 = Output
Enable Interrupts
Enable Aci
On Aci Comparator_isr
```

```
Do
'YOU CAN WRITE YOUR PROGRAM HERE.
Loop
'-----------------------------------------------------------------------------------------------------------
Comparator_isr:
Toggle Portb.0
Waitms 100
Return
'-----------------------------------------------------------------------------------------------------------
```

مدهای Sleep :

فعال شدن قسمت های مختلف میکروکنترلر AVR با استفاده از منابع مختلف پالس صورت می گیرد با توجه به این که وارد شدن قسمت های مختلف به مد های SLEEP ، با غیر فعال شدن این پالس ها صورت می گیرد در ادامه مد های SLEEP را مورد بررسی قرار خواهیم داد. به طور کلی مد های SLEEP به منظور متوقف کردن قسمت های غیر فعال میکروکنترلر به کار می روند تا توان تلفاتی کاهش یابد .

میکروکنترلر های AVR دارای 6 مد SLEEP می باشند که تمام این حالات در میکرو کنترلر نمونه ATMEGA32 وجود دارد در ادامه به معرفی 6 نوع مد SLEEP می پردازیم .

مد IDLE :

در این حالت CPU متوقف می شود ولی ماژول های SPI،USART، مقایسه کننده آنالوگ ، WATCHDOG ، I2C ، شمارنده ها ، مبدل ADC و سیستم وقفه به کار خود ادامه می دهد در این حالت فقط پالس های کلاک FLASH و CPU متوقف می شوند .

وقوع هر گونه وقفه خارجی و داخلی نظیر وقفه های (TIMER/COUNTER) یا وقفه اتمام انتقال توسط فرستنده UART یا SPI منجربه WAKE UP (بیدار شدن از مد اسلیپ) شدن میکروکنترلر می شود .

مد ADC NOISE REDACTION :

این مد با کاهش اثر نویز مدار های دیجیتال بر روی مبدل ADC باعث افزایش دقت مبدل ADC می شود. در صورتی که ADC فعال باشد به محض وارد شدن به این مد کار خود را شروع می کند در این مد CPU متوقف می شود ولی ADC وقفه های خارجی ، WATCHDOG ، COUNTER2 ، TIMER و I2C فعال هستند در این مد فقط پالس های کلاک FLASH ، CPU ، I/O غیر فعال هستند .

وقفه اتمام عملیات تبدیل ADC ، RESET خارجی ، RESET مربوطه WATCHDOG وقفه I2C ، وقفه های تایمر/کانتر 2 و وقفه های خارجی ، میکروکنترلر را از این مد به حالت WAKE UP می برد.

مد POWER_DOWN :

در این مد اسیلاتور خارجی متوقف می شود ولی وقفه های خارجی ، WATCHDOG ، I2C در صورت فعال بودن به کار خود ادامه می دهند در این مد RESET خارجی ، WATCHDOG RESET ، BROUN OUT ، وقفه های تایمر/کانتر2 و وقفه های خارجی میکروکنترلر را به حالت WAKE UP می برند وقفه های خارجی حساس به سطح INT2,INT1,INT0 نیز می توانند میکروکنترلر را از این مد WAKE UP کنند .

WAKE UP شدن در این مد ، با یک تاخیر همراه است ، این تاخیر فعال شدن مجدد پالس و پایدار شدن آن را تضمین می کند به این تاخیر مدت زمان START_UP سیستم می گویند که مدت زمان آن با استفاده از فیوز بیت های CKSEL3..0 و SUT1..0 تعیین می شود .

مد POWER_SAVE :

این مد مشابه مد POWER_DOWN است به جز این که اگرتایمر/کانتر 2 به صورت آسنکرون کار کند ، در این مد فعال خواهد بود . زمانی که از تایمر/کانتر 2 به عنوان شمارنده تایم واقعی (REAL TIME COUNTER) استفاده می شود یعنی کریستال ساعت 32.768KHz بین پایه ها TOSC2,TOSC1 قرار می گیرد تایمر/کانتر 2 کلاک خود را به صورت آسنکرون از کلاک سیستم و سنکرون با اسیلاتور ساعت خارجی دریافت می کند . این مد فقط با وقفه های مربوط به تایمر/کانتر 2 به حالت WAKE UP می رود به طور کلی در این مد تمامی پالسها به جز پالس های کلاک آسنکرون متوقف هستند در نتیجه فقط تجهیزات که به صورت آسنکرون کار می کنند در این مد فعال خواهند بود .

توجه داشته باشید زمانی که از تایمر 2 در مد آسنکرون استفاده نمی شود مد POWER_DOWN به مد POWER_SAVE ترجیح داده می شود .

مد STAND BY :

این مد زمانی مورد استفاده قرار می گیرد که کلاک سیستم از اسیلاتور خارجی تامین می شود . این مد مشابه مد POWER_SAVE است بااین تفاوت که در این مد اسیلاتور خارجی قطع نمی شود WAKE UP شدن میکروکنترلر از این مد 6 کلاک سیکل طول می کشد .

مد EXTENDED STANDBY :

این مد زمانی مورد استفاده قرار می گیرد که کلاک سیستم از اسیلاتور خارجی تامین می شود . این مد مشابه مد POWER_SAVE است با این تفاوت که در این مد اسیلاتور خارجی قطع نمی شود WAKE UP شدن میکروکنترلر از این مد 6 کلاک سیکل طول می کشد .

دستورات اجرای مد های SLEEP در BASCOM

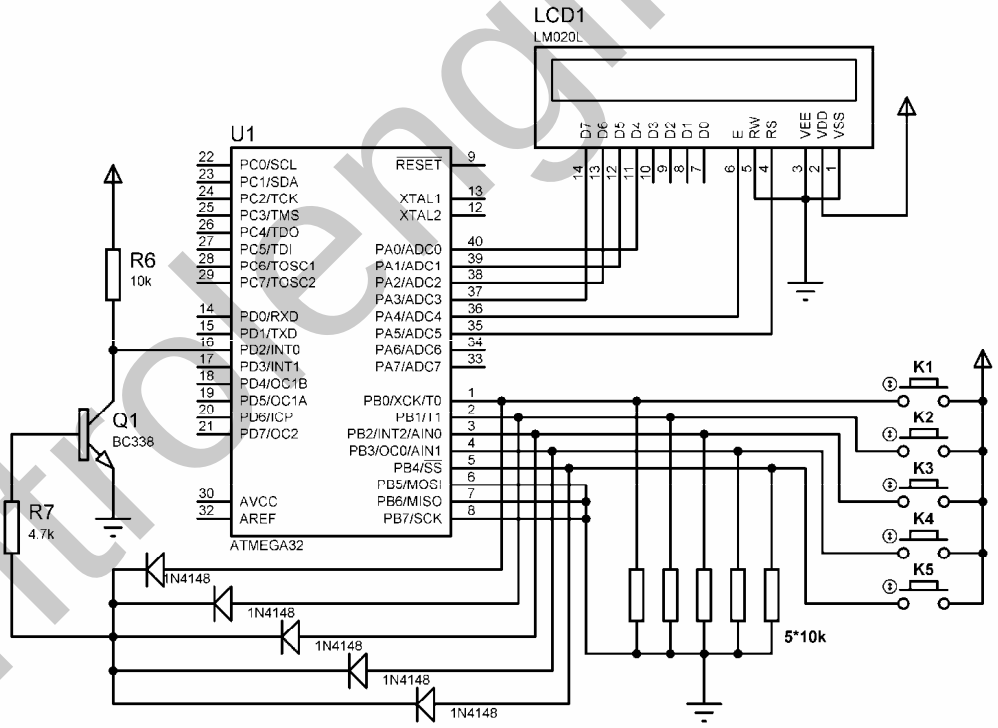
توسط دستور IDLE میکروکنترلر وارد مد اسلیپ IDLE می شود .

توسط دستور POWER_DOWN میکروکنترلر وارد مد اسلیپ POWER_DOWN می شود .

توسط دستور POWER_SAVE میکروکنترلر وارد مد اسلیپ POWER_SAVE می شود .

موارد استفاده

یکی از موارد استفاده مد های اسلیپ ریموت کنترل هاست. در ریموت ها به دلیل استفاده از منابع تغذیه قابل حمل (باطری) ، حداقل توان مصرفی مورد نیاز است، برای درک بهتر موضوع پروژه زیر را در نظر بگیرید . دراین پروژه زیر در حالت عادی و زمانی که هیچ یک از کلید های K1 تا K5 فشار داده نشده میکروکنترلر به منظور منیمم کردن توان مصرفی وارد مد اسلیپ IDLE می شود و عبارت IN IDLE روی LCD نوشته می شود پس از فشار هر کدام از کلید ها وقفه خارجی INT0 رخ داده و میکروکنترلر از این مد WAKE UP شده و در ISR وقفه INT0 به کلید ورودی پاسخ می دهد و نام کلید فشرده شده بر روی LCD نوشته می شود . سخت افزار پروژه در شکل 2-93 ارائه شده است .



شکل 2-93 شماتیک طراحی شده برای استفاده از مد اسلیپ IDLE برای به حداقل رساندن توان مصرفی

برنامه نوشته شده برای پروژه فوق به صورت زیر است .

```
$regfile = "M32DEF.DAT"
$crystal = 1000000
```

```
Config Lcd = 16 * 1
Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3_
                    , E = Pina.4 , Rs = Pina.5
Config Portb = Input
Dim Key_data As Byte
'-----------------------------------------------------------------------------------------------------------
Enable Interrupts
Enable Int0
On Int0 Int0_isr
'-----------------------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "IN IDLE"
Do
Idle
Loop
'-----------------------------------------------------------------------------------------------------------
Int0_isr:
Key_data = Pinb
Select Case Key_data:
Case Is = &B00000001
Cls : Home
Lcd "INPUT IS K1"
Case Is = &B00000010
Cls : Home
Lcd "INPUT IS K2"
Case Is = &B00000100
Cls : Home
Lcd "INPUT IS K3"
Case Is = &B00001000
Cls : Home
Lcd "INPUT IS K4"
Case Is = &B00010000
Cls : Home
Lcd "INPUT IS K5"
End Select
Wait 2
Cls : Home
Lcd "IN IDLE"
Return
'-----------------------------------------------------------------------------------------------------------
```

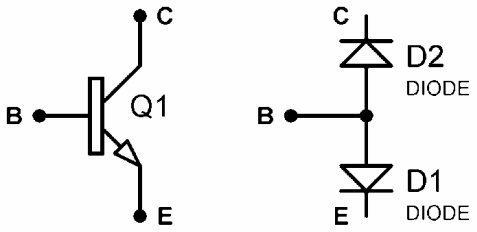
اگر از هر چیزی بهترینش را نداری ، از هر چیزی که داری بهترین استفاده را بکن زیرا خوشبختی داشتن همه چیز نیست خوشبختی لذت بردن از چیز هاییست که داریم در واقع با ناراحت بودن به خاطر آن چه ندارید داشته هایتان را نابود خواهید کرد و با لذت بردن از آن چه که دارید نداشته هایتان را .

فصل سوم

اطلاعات کاربردی

سوییچینگ با ترانزیستور

از یک ترانزیستور دو قطبی می توان به عنوان یک کلید وصل (در ناحیه اشباع) و یک کلید قطع (در ناحیه قطع) استفاده نمود برای این منظور مشخصات نواحی قطع و اشباع را مورد بررسی قرار می دهیم . در ناحیه اشباع ترانزیستور مانند یک کلید وصل بین کلکتور و امیتر و در ناحیه قطع ترانزیستور مانند یک کلید قطع بین کلکتور و امیتر عمل می کند . یک ترانزیستور دو قطبی را می توان مانند شکل 1-3 توسط دو دیود مدل کرد .



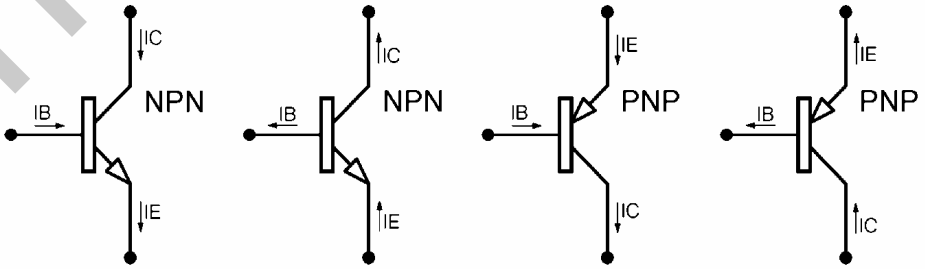
شکل 1-3 معادل دیودی یک ترانزیستور دو قطبی (BJT)

در ناحیه اشباع دیود BE , BC در بایاس مستقیم قرار دارند (برای این که دیود در بایاس مستقیم باشد بایستی ولتاژ آنود آن نسبت به کاتود مثبت تر باشد.) در ناحیه قطع دیود BE , BC در بایاس معکوس قرار دارند .

نکته مهم در رابطه با جهت جریانها در یک ترانزیستور BJT :

توجه داشته باشید در یک ترانزیستور BJT از نوع NPN همواره جهت جریان کلکتر و امیتر تابعی از جهت جریان بیس می باشد به عنوان مثال اگر جریان بیس وارد ترانزیستور شود ، کلکتور هم وارد ترانزیستور می شود در این حالت جریان امیتر که نسبت عکس با جهت جریان بیس دارد به سمت خارج ترانزیستور خواهد بود.

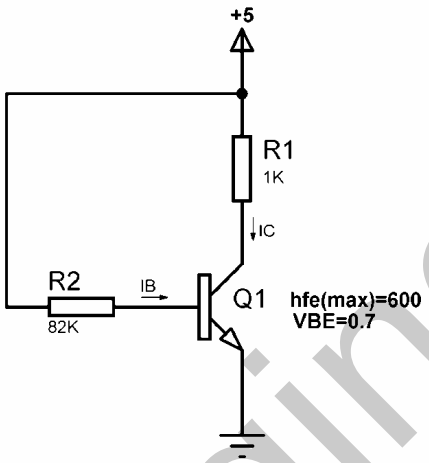
همچنین در تزانزیستور BJT از نوع PNP اگر جریان بیس وارد ترانزیستور شود جریان امیتر هم وارد ترانزیستور خواهد شد و جهت جریان کلکتور که نسبت عکس با جهت جریان بیس دارد به سمت خارج ترانزیستور خواهد بود.



شکل 2-3 جهت جریان ها در ترانزیستور BJT

نحوه تشخیص ناحیه اشباع

فرض کنید مدار زیر را داریم می خواهیم بررسی کنیم که ترانزیستور در ناحیه اشباع قرار دارد یا نه برای این منظور ابتدا بایستی ترانزیستور را در ناحیه فعال فرض کرده و VCE را بدست آوریم اگر VCE منفی باشد ترانزیستور در ناحیه اشباع خواهد بود .



IB=(5-0.7) / 82k=52.4UA
IC= hfe (max) * IB = 600*52.4UA = 31.44mA
VCE=5 – (1k * 31.44mA) = - 26.44V

ملاحظه می شود که ترانزیستور فوق اشباع شده است .

به صورت کلی برای قرار گرفتن در ناحیه اشباع hfe ترانزیستور بایستی مقدار مینیمم را داشته باشد این مقدار به شرایط مدار بستگی دارد به عنوان مثال اگر جریان بیس یک ترانزیستور 50UA و جریان کلکتور آن در حالت اشباع IC (SAT) را 1mA فرض کنیم hfe (min) از رابطه

Hfe(min) = IC (SAT) / IB = 1mA/50 UA=20

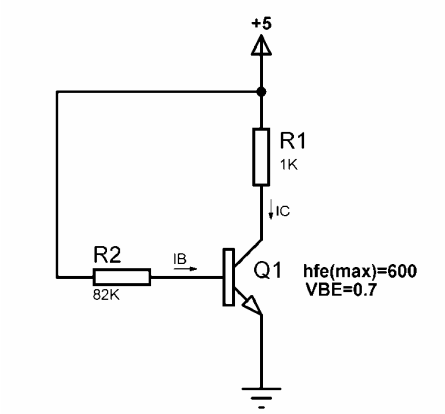
بدست می آید . که در این حالت اگر مقدار hfe از 20 کمتر باشد IC کمتر از 1 mA بوده و ترانزیستور اشباع نخواهد بود ولی اگر hfe از 20 بزرگتر باشد جریان IC می خواهد از1mA بیشتر شود و ترانزیستور را اشباع کند.

اگر در ترانزیستوری hfe(min)=20 ، IC (SAT)=1 mA باشد اگر IB از مقدار

IB=IC (SAT) / hfe (min) =50UA

کمتر شود ترانزیستور وارد ناحیه تقویت کننده گی یا فعال خواهد شد علاوه بر این با توجه به این که اگر جریان IC (SAT) مقدار ماکزیمم IC می باشد و IC دیگر نمی تواند از این مقدار زیاد تر شود با زیاد شدن جریان IB از 50UA به ناچار hfe(min) تغییر خواهد کرد . با توجه به فرمول IB=IC(SAT)/hfe(min) ، hfe(min) رابطه عکس با جریان IB دارد .

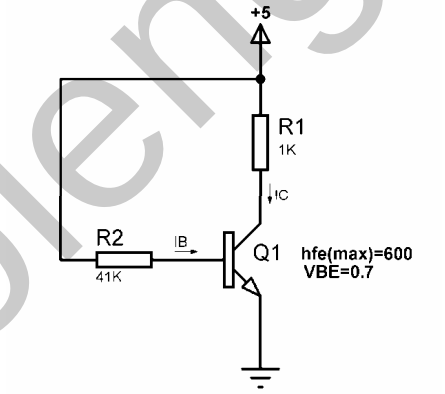
برای درک بهتر این موضوع ابتدا hfe(min) را در شکل زیر بدست آورید .



نکته : زمانی که ترانزیستور در ناحیه اشباع قرار دارد ولتاژ ی حدود 0.2 ولت روی کلکتر امیتر افت می کند که در اینجا از آن صرفه نظر شده است .

IC (SAT) = 5 /1K=5mA
IB = (5-0.7) / 82K = 52.4UA
hfe (min) =IC (SAT) / IB = 5mA/52.4UA =95.4

حالا IB را دو برابر کرده و دوباره hfe(min) را بدست می آوریم ، برای این که IB دو برابر شود بایستی مقدار RB را نصف کنیم .

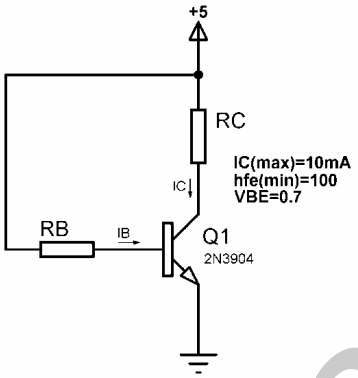


IB = (5-0.7) / 41K = 104.8UA
IC (SAT) = 5 /1K=5mA
hfe (min) =IC (SAT) / IB = 5mA/104.8UA =47

ملاحظه می شود که مقدار hfe(min) وقتی که IB = 52.4UA می باشد برابر با 95.4 بوده و با دو برابر شدن IB یعنی IB = 104.8UA مقدار hfe(min) به ½ مقدار قبلی کاهش پیدا می کند .

برای بدست آوردن مشخصات ترانزیستور از DATA SHEET (برگه های اطلاعاتی) مربوط به ترانزیستور استفاده می کنیم ، به عنوان مثال ترانزیستور 2N3904 را در نظر بگیرید با توجه به DATA SHEET این

ترانزیستور IC (MAX)=10mA و hfe(min)=100 وhfe(max)=300 می باشد یعنی در این ترانزیستور به ازای IB=0.2mA ، IC ، می تواند بین (20mA تا 60mA) باشد. ولی در حقیقت جریان کلکتر نمی تواند از مقدار IC=VCC/RC=10mA بیشتر شود حالا با استفاده از ترانزیستور 2N3904 ، مدار زیر را طوری طراحی کنید که ترانزیستور اشباع شود .



با توجه به برگه های اطلاعاتی ترانزیستور فوق در می یابیم که IC نمی تواند بیشتر از 10mA شود پس حداقل مقدار RC در حالتی که از VCE صرفه نظر شود می تواند برابر با RC=5/10mA=500 اهم باشد . برای این که ترانزیستور اشباع شود بایستی IB=IC (SAT)/hfe(min)=0.1mA باشد .

که در این حالت مقدار RB به صورت زیر محاسبه می شود .

RB = (5-0.7) / 0.1mA=43K

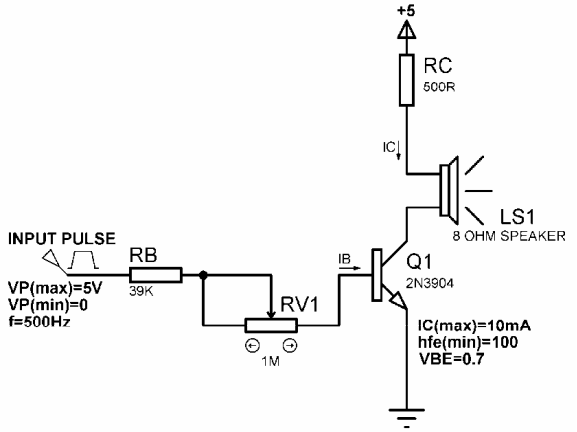
اگر مقاومت RB بدست آمده جزء مقاومت های استاندارد نباشد ، بایستی رنج استاندارد پایین تر را برگزینیم پس RB را برابر 39K می گیریم.

برای اطلاع از مقادیر استاندارد مقاومت و خازن به ضمیمه آخر کتاب مراجعه نمایید.

دو نکته مهم در رابطه با IB این مدار :

۱- برای این که ترانزیستور اشباع شود IB بایستی برابر با 0.1mA و یا بیشتر از آن باشد .

۲- در حالتی که IB کمتر از 0.1mA باشد ترانزیستور وارد ناحیه فعال شده و مانند یک تقویت کننده کلاس B عمل خواهد کرد . در این نوع تقویت کننده به دلیل این که BE ترانزیستور بایاس نمی شود ، قسمتی از ورودی صرف بایاس کردن ترانزیستور خواهد شد . پس در این حالت می توان با تغییر جریان IB (کمتر از مقدار 0.1mA) مقدار IC (کمتر از 10mA) را تعیین نمود که در شکل 3-3 از این حالت برای طراحی ولوم به منظور تعیین دامنه صوتی SPEAKER استفاده شده است ، در این مدار اگر ورودی یک شکل موج مربعی با دامنه 5 ولت باشد صدای تولید شده توسط SPEAKER صدای سوت خواهد بود که دامنه آن توسط ولوم RV1 تنظیم می شود.



شکل 3-3 نحوه طراحی ولوم برای تغییر دامنه صوتی SPEAKER

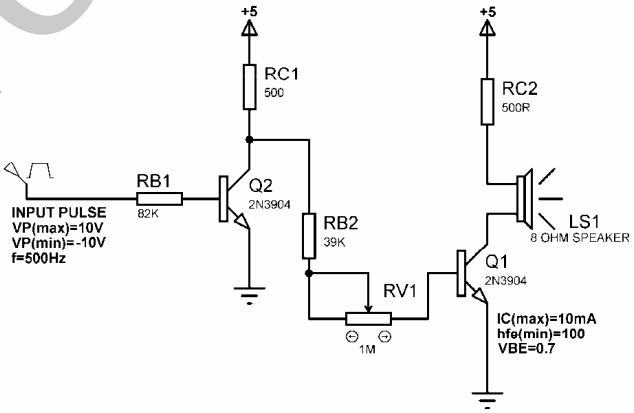
در مدار فوق در حالتی که مقدار مقاومت پتانسیومتر 1M اهم برابر صفر باشد IB توسط مقاومت RB=39K بیشتر از 0.1mA خواهد بود ، پس ترانزیستور در نواحی قطع و اشباع عمل کرده و صدای بوق شنیده شده از SPEAKER ماکزیمم خواهد بود .

اگر مقدار مقاومت پتانسیومتر را کمی زیاد تر کنیم مقدار مقاومت RB از رابطه زیر بدست خواهد آمد .

RBT=RB + R (پتانسیومتر)

اگر RBT بیشتر از 43K شود ترانزیستور مانند یک تقویت کننده توان کلاس B عمل کرده و با افزایش مقدار پتانسیومتر دامنه صدای تولید شده توسط SPEAKER کم خواهد شد .

برای این که ترانزیستور با اعمال ولتاژ صفر به بیس سوئیچ کند و با اعمال ولتاژ 5 ولت ، قطع شود می توانیم از یک ترانزیستور NPN دیگر به عنوان معکوس کننده در ورودی استفاده کنیم که در این حالت مدار به صورت شکل 3-4 تغییر داده می شود .



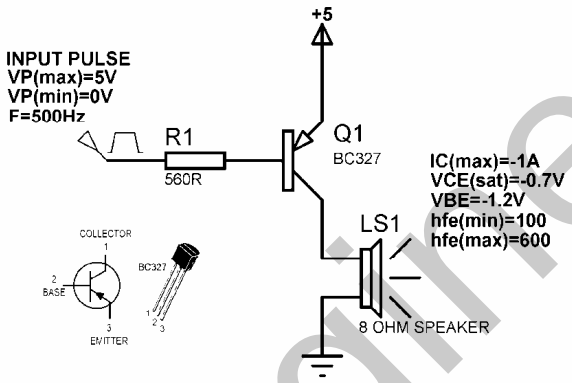
شکل 3-4 نحوه استفاده از یک ترانزیستور NPN به عنوان معکوس کننده در ورودی

در مدار فوق با توجه به این که مقدار IC =10mA بوده و hfe(min) =100 می باشد ib در هر دو ترانزیستور برابر با 0.1mA خواهد بود در نتیجه RB1 به صورت زیر تعیین می شود .

RB1 = (10 – 0.7)/0.1mA =93k

نزدیک ترین رنج استاندارد پایین تر از 93K برابر است با 82K .

بهترین راه حل برای این که ترانزیستور با اعمال ولتاژ صفر به بیس سوئیچ کرده و با اعمال ولتاژ 5 ولت قطع شود استفاده از یک ترانزیستور PNP است به عنوان مثال مدار شکل 3-5 را طوری طراحی کنید که ترانزیستور BC327 اشباع شود .



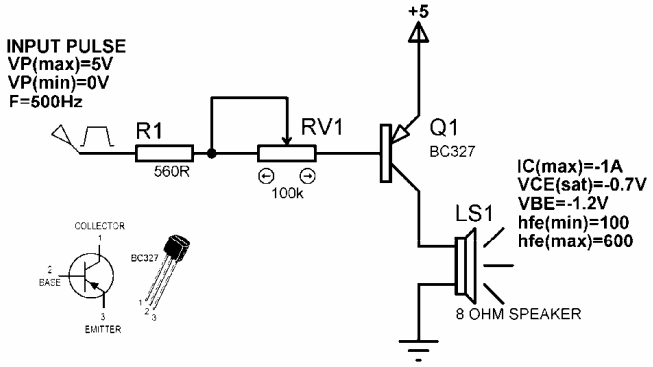
شکل 3-5 نمونه مدار ارائه شده برای این که ترانزیستور با اعمال ولتاژ صفر به بیس سوئیچ کند

در ترانزیستور های قبلی به دلیل کم بودن VCE(SAT) (حدودا 0.2) از آن در محاسبات صرفه نظر می کردیم ولی ترانزیستور BC327 به دلیل این که مقدار VCE(SAT) برابر با 0.7 ولت می باشد از آن در محاسبات استفاده می کنیم .

IC (SAT) = (5 – VCE(SAT)) / 8 = (5 – 0.7) / 8 = 0.53A
IB = IC (SAT) / hfe (min) = 0.53/100 =5.37mA
RB = (5 – VBE) / IB =(5 – 1.2) / IB =707.63R

چون 707.63 رنج استاندارد نمی باشد نزدیک ترین رنج کمتر از آن یعنی 560R را برای RB انتخاب می کنیم .

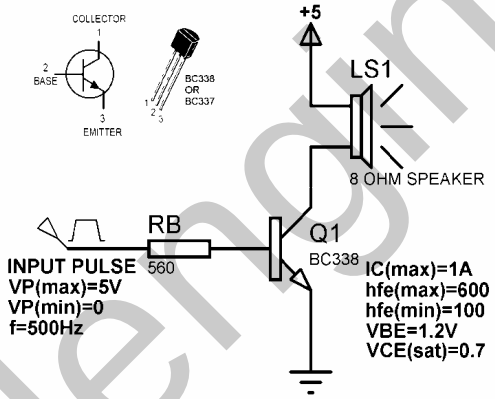
در مدار فوق مقدار دامنه صدای تولید شده توسط SPEAKER ماکزیمم خواهد بود برای تغییر دامنه صدا به صورت دستی می توانیم از مدارشکل 3-6 استفاده کنیم .



شکل 3-6 نحوه طراحی ولوم برای کنترل دامنه صوتی SPEAKER هنگام استفاده از BC327 به عنوان درایور

در پروژه های این کتاب از ترانزیستور BC338 یا BC337 به عنوان درایور SPEAKER استفاده شده است .

نحوه تعیین RB در ترانزیستور BC338 به منظور سوئیچ کردن بار 8 اهمی به صورت زیر است .

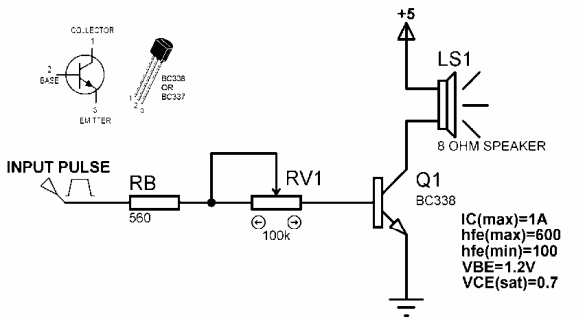


شکل 3-7 مدار مربوط به تعیین RB در ترانزیستور BC338 برای این که ترانزیستور در نواحی قطع و اشباع کار کند

IC (SAT) = (5 – VCE(SAT))/8 =0.53A
IB = IC(SAT)/hfe(min) = 5.37mA
RB=(5 – 1.2)/ IB = 707.63

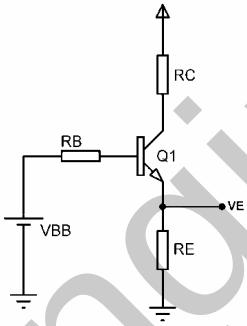
به دلیل این که 707.63 رنج استاندارد نمی باشد ، نزدیکترین رنج کمتر از آن یعنی 560 اهم را برای RB انتخاب می کنیم .

در مدار فوق دامنه صدای تولید شده توسط SPEAKER ماکزیمم خواهد بود برای تغییر دامنه صدا به صورت دستی می توانیم از مدار شکل 3-8 استفاده کنیم .



شکل 3-8 نحوه طراحی ولوم برای تغییر دامنه صوتی SPEAKER

توجه داشته باشید برای این که ترانزیستور اشباع شود دامنه ولتاژ ورودی نبایستی از حد مشخص کمتر شود برای درک بهتر این موضوع مدار شکل 3-9 را در نظر بگیرید .

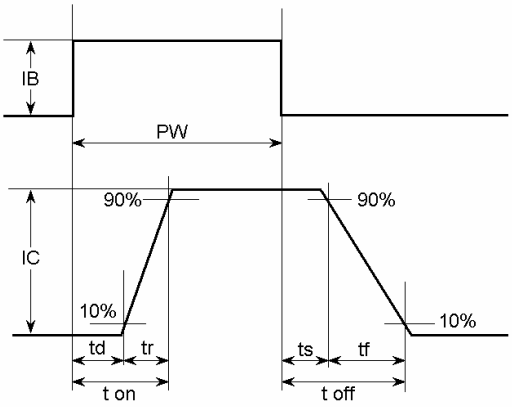


شکل 3-9 نحوه تعیین مقدار VBB

در چنین شرایطی برای این که ترانزیستور اشباع شود مقدار VBB به صورت زیر تعیین می شود .

VBB > (VBE + VE)
VE = IC (SAT) * RE

عامل مهمی که هنگام سوئیچینگ با فرکانس های بالا بایستی مورد توجه قرار گیرد مدت زمان Ton و Toff ترانزیستور می باشد .

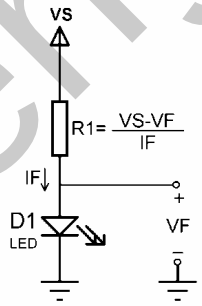


شکل 3-10 مدت زمان های ton و toff ترانزیستور

با اعمال جریان بیس IB ترانزیستور فورا روشن (وصل) نمی شود . زمان بین اعمال جریان بیس و شروع افزایش جریان کلکتور زمان تاخیر td نام دارد. زمان تاخیر ، زمان بین شروع IB تا 10 درصد مقدار نهایی خود است . بعد از شروع روشن شدن ترانزیستور هم مقداری زمان لازم است تا IC به مقدار نهایی خود برسد . زمان صعود T2 زمان لازم برای رسیدن IC ، از 10 درصد مقدار نهایی به 90 درصد مقدار نهایی خود است . زمان روشن شدن ton جمع td و tr است. ترانزیستور نمی تواند به طور آنی خاموش شود. زمان خاموش شدنtoff مجموع زمان ذخیره ts و زمان نزول tf است . برای بدست آوردن زمان های مزبور می توانید به DATA SHEET ترانزیستور مراجعه کنید .

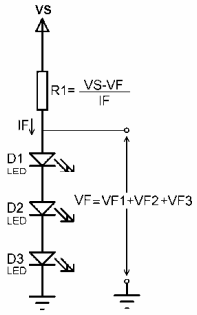
مدار های پایه مربوطه به LED

با اتصال سری یک LED به یک مقاومت محدود کننده جریان و یک منبع تغذیه DC مناسب می توان از LED به عنوان مولد نور استفاده کرد . به عنوان مثال اگر یک LED در هنگام بایاس مستقیم و در جریان 20mA افت ولتاژ 2 ولت داشته باشد و از یک منبع تغذیه 10 ولتی و 2 آمپری تغذیه کند در این حالت R بایستی مقدار (10 – 2)/ 20mA=400R را داشته باشد که R می تواند به آنود یا کاتود LED وصل شود شکل 3-11 حالت کلی این روند را نشان می دهد . در مدار های معرفی شده در رابطه با LED می توانید VF را برابر 2 ولت و HF را برابر 20mA جایگزین کنید .



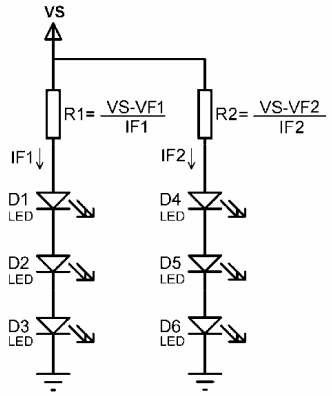
شکل 3-11 نحوه سری کردن مقاومت محدود کننده جریان با LED

اگر لازم شود که چندین LED از یک منبع تغذیه استفاده کنند می توان از مدارات زیر استفاده کرد .



شکل 3-12 نحوه سری کردن چندین LED

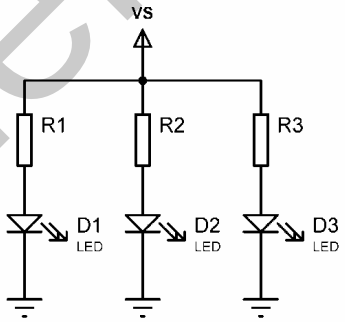
در مدار فوق LED ها به صورت سری به هم وصل شده اند و از طریق یک مقاومت محدود کننده جریان واحد جریان دهی می شوند در حالت کلی می توانید مقدار VF هر LED را برابر با 2V و مقدار IF را برابر 20mA بگیرید . شکل 3-13 نمونه دیگری از مدارات راه اندازی چندین LED توسط یک منبع می باشد .



شکل 3-13 نمونه دیگری از راه اندازی چندین LED با استفاده از منبع تغذیه منفرد

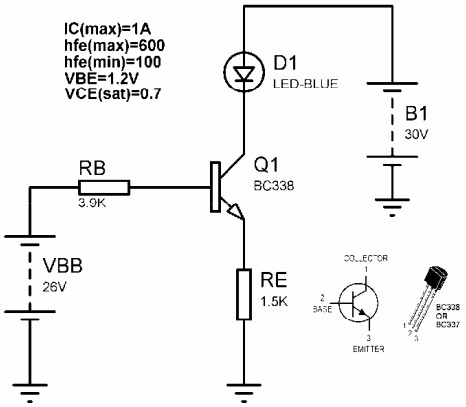
هر تعداد از مدارات شکل قبل می تواند مانند شکل فوق به صورت موازی به هم متصل شوند تا تعداد دلخواهی از LED ها را جریان دهی نمایند . البته میزان جریان دهی ماکزیمم منبع نیز بایستی مورد توجه قرار گیرند .

مدار شکل 3-14 نیز قادر به جریان دهی به هر تعداد دلخواهی از LED ها می باشد در این حالت اتلاف جریان تغذیه بسیار زیاد است (برابر با مجموع جریان های هر LED)



شکل 3-14 نمونه دیگری از راه اندازی چندین LED با استفاده از منبع تغذیه منفرد

برای سوئیچ کردن یک LED از طریق ترانزیستور BC338 می توانید از مدار شکل 3-15 استفاده کنید .



شکل 3-15 نحوه راه اندازی LED توسط ترانزیستور BC338

IC (SAT) =ILED =20mA
RE=(30 – (VCE (SAT) +2)) / 20mA = 27.3/ 20mA = 1365R

برای RE از رنج استاندارد 1.2K استفاده می کنیم .

برای تعیین VBB همواره بایستی شرط زیر برقرار باشد .

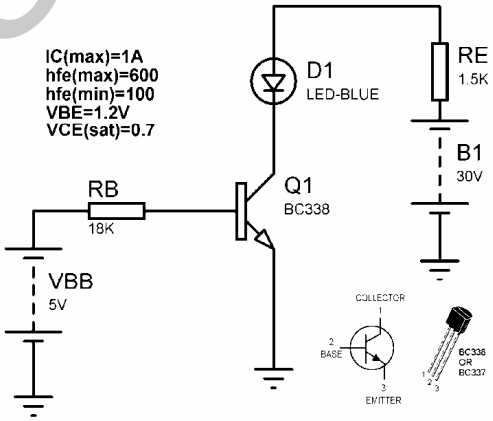
VBB > (VBE + VE)
VE = 1.2K * 20mA =24V
VBB > 25.2 V

در نتیجه مقدار VBB را 26 ولت در نظر می گیریم که در این حالت مقدار RB به صورت زیر تعیین می شود .

IB = IC (SAT) / hfe(min) =20mA / 100 = 0.2mA
RB = (VBB – (VBB + VE)) / IB = 4K

نزدیک ترین رنج استاندارد کمتر از 4K برابر با 3.9K می باشد .

برای این که مقدار VBB را کاهش می دهیم می توانیم مدار را به صورت شکل 3-16 تغییر دهیم .



شکل 3-16 مثال دیگری از نحوه راه اندازی LED توسط ترانزیستور BC338

IC (SAT) = I LED =20mA
RE = 1.2K

برای تعیین VBB همواره بایستی شرط زیر برقرار باشد .

VBB > (VBB + VE)
VE= 0 ⟶ VBB >1.2V

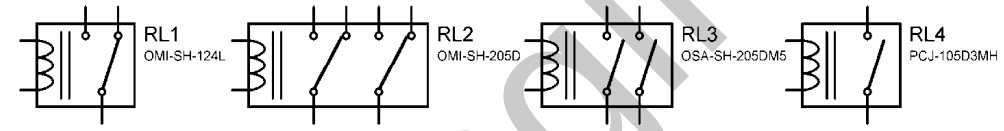
برای آنکه بتوانیم از خروجی مدارات دیجیتالی برای سوئیچ کردن ترانزیستور استفاده کنیم مقدار VBB را برابر با 5 ولت در نظر می گیریم .

IB = IC (SAT) / hfe(min) = 20mA / 100 = 0.2mA
RB = (VBB – VBE) / IB = (5 – 1.2) / 0.2mA

برای RB مقدار استاندارد 18K را انتخاب می کنیم.

مدار های پایه مربوطه به رله

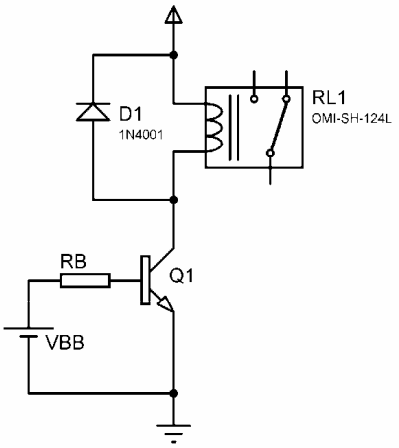
انواع مختلفی از رله های الکترو مغناطیسی وجود دارد شکل 3-17 شماتیک داخلی چند نمونه از این رله ها را نشان می دهد .



شکل 3-17 شماتیک داخلی چندین نوع رله الکترومغناطیسی

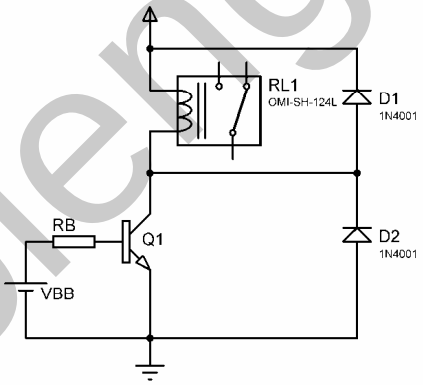
نحوه عملکرد رله بدین صورت می باشد که در صورتی که سیم پیچ رله تحریک شود کلید یا کلید های داخلی آن تغییر وضعیت می دهند . در پروژه های این کتاب از رله نوع RL1 استفاده شده است .

سیم پیچ داخلی رله دارای بارهای القائی است و در صورتی که جریان سیم پیچ آن به صورت ناگهان قطع شود می تواند ولتاژ موثر معکوس محدود در حد چند صد ولت ایجاد کند ، این ولتاژ معکوس به آسانی می تواند به کنتاکت های رله یا قطعات نیمه هادی متصل شده و به سیم پیچ آن آسیب برساند ، بنابراین در اغلب موارد لازم است با استفاده از دیود محافظ ولتاژ موثر معکوس تخفیف داده شود . شکل زیر مدار تخفیف دهنده سیگنال های سیم پیچ را نشان می دهد .



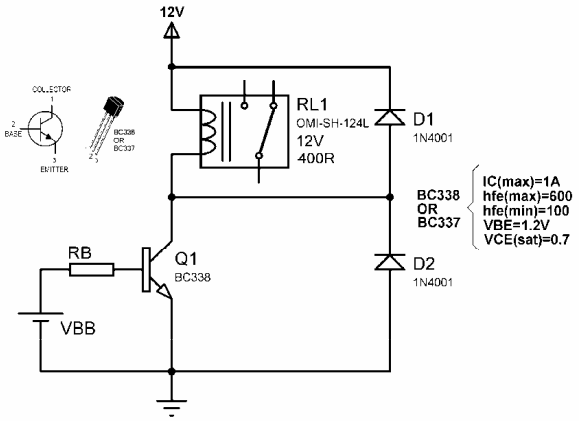
شکل 3-18 مدار تخفیف دهنده سیگنال های سیم پیچ رله

در شکل فوق در صورتی که ولتاژ موثر معکوس ناشی از قطع جریان بیشتر از VD یعنی 0.6 یا 0.7 ولت باشد دیود D1 از افزایش بیشتر آن جلوگیری خواهد کرد برای مدارات حساس تر می توانید از مدار شکل 3-19 استفاده کنید که در این حالت نوسانات ولتاژ به 0.6 ولت بیشتر از ولتاژ تغذیه و 0.6 ولت کمتر از ولتاژ تغذیه محدود می شود .



شکل 3-19 نوع دیگری از مدارات تخفیف دهنده سیگنال های سیم پیچ رله

برای تحریک رله با استفاده از ترانزیستور می توانید از مدار زیر استفاده کنید .



شکل 20-3 نحوه طراحی مقدار RB، VBB برای سوئیچ کردن یک رله با ترانزیستور BC338

اگر مقاومت سیم پیچ رله را 400R در نظر بگیریم محاسبات به صورت زیر انجام می شود .

IC (SAT) = (12 – 0.7) / 400 = 28.25mA
IB = IC (SAT) /hfe(min) = 28.25mA /100 =0.282mA
VBB > (VBB + VE)
VBB > 1.2V

برای آنکه بتوانیم از خروجی مدارات دیجیتالی برای تحریک رله استفاده کنیم مقدار VBB را برابر با 5 ولت در نظر می گیریم که در این حالت مقدار RB به صورت زیر تعیین می شود .

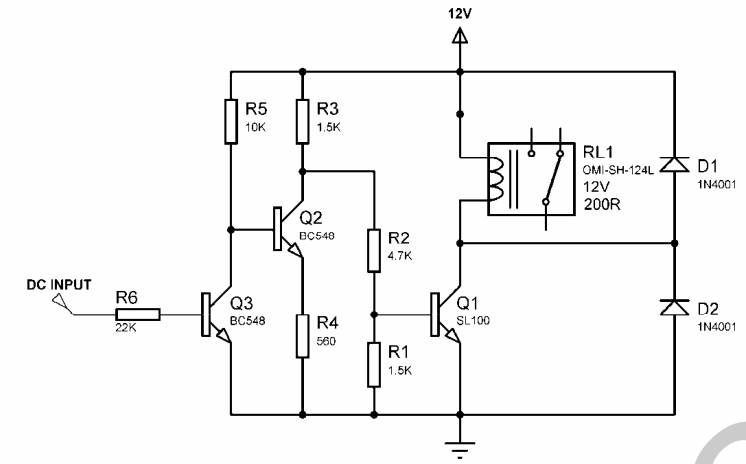
RB = (VBB – VBE) / IE = (5- 1.2) / 0.282mA

نزدیکترین رنج استاندارد کمتر از این مقدار برابر است با 12k پس RB برابر با 12K خواهد بود .

رابط تحریک رله برای مدار های دارای جریان خروجی پایین

معمولا برای تحریک کردن رله ها با استفاده از خروجی تراشه های TTL یا مدار های دارای جریان خروجی پایین به یک مدار واسط نیاز داریم در این بخش یک مدار تحریک معرفی شده است که از 3 ترانزیستور استفاده می کند و به ولتاژ و جریان تحریک بسیار کوچکی نیاز دارد (750mV و 30 تا 50 میکروآمپر).

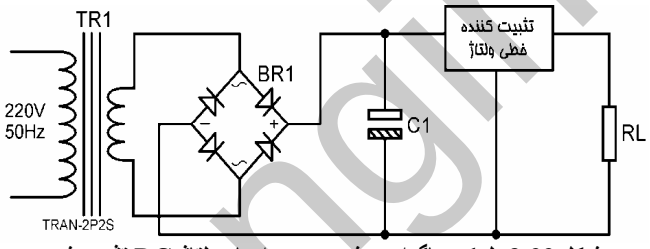
برای دو طبقه اول از ترانزیستورهای قدرت متوسط BC548 استفاده شده است در حالی که برای طبقه آخر می توان از ترانزیسور های BEL187 ، SL100 یا BD139 استفاده کرد . در همه ترانزیستورها از آرایش امیتر مشترک استفاده شده ، ضمن این که طبقه دوم دارای مقاومت امیتر R4 (فیدبک منفی) می باشد تا پایداری حرارتی مدار افزایش یابد حساسیت مدار را می توان با اتصال انگشت دست بین مثبت تغذیه +12V و ورودی DC INPUT آزمایش نمود جریان DC ناچیزی که از پیوست دست میگذرد برای تحریک مدار و انرژی دار کردن رله کافی خواهد بود .



شکل 3-21 رابط تحریک رله برای مدار های دارای جریان خروجی پایین

منابع تغذیه DC

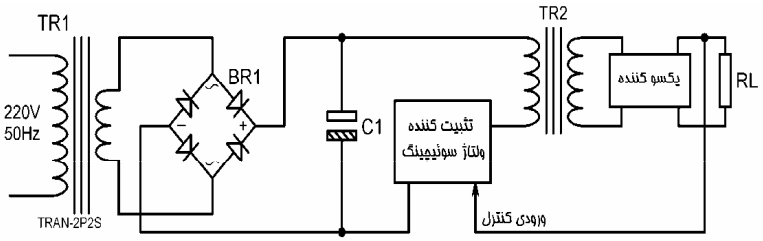
دو روش اصلی برای دستیابی به ولتاژ DC اندک تا متوسط از طریق خط تغذیه اصلی وجود دارد که بلوک دیاگرام آنها در شکل های زیر نشان داده شده است.



شکل 3-22 بلوک دیاگرام روش مرسوم ایجاد ولتاژ DC تثبیت شده

روش فوق روش مرسوم ایجاد ولتاژ DC تثبیت شده می باشد که کارائی معمول آن 85 درصد است ، در این روش ابتدا با استفاده از ترانسفورماتور کاهنده ولتاژ AC ورودی به ولتاژ AC حدودا 3 ولت بیشتر از ولتاژ نامی تثبیت کننده خطی کاهش داده می شود ، سپس ولتاژ AC خروجی ترانسفورماتور با استفاده از مدار یکسو کننده که شامل پل دیودBR1 و خازن C1 می باشد یکسو سازی شده و به ورودی تثبیت کننده خطی اعمال می شود. در خروجی تثبیت کننده ، ولتاژ DC تثبیت شده به مقدار ولتاژ نامی تثبیت کننده خطی خواهیم داشت .

روش دیگر تبدیل سیگنال AC به DC تثبیت شده استفاده از روش سوئیچینگ می باشد که بلوک دیاگرام آن در شکل 3-23 نشان داده شده است .



شکل 3-23 روش تبدیل ولتاژ AC به ولتاژ DC تثبیت شده به روش سوئیچینگ

نحوه عملکرد مدار بدین صورت می باشد که ابتدا ولتاژ AC ورودی توسط طبقه یکسوکننده BR1 و خازن حذف ریپل C1 به ولتاژ DC برای تغذیه تثبیت کننده ولتاژ سوئیچینگ تبدیل می شود ، تثبیت کننده ولتاژ سوئیچینگ نوعی مدولاتور پهنای پالس (PWM) می باشد . خروجی مدار مزبور که فرکانس معمول آن 20KHz بوده و یک موج مربعی پر قدرت با ضریب اتصال متغیر می باشد ، با استفاده از ترانسفورماتور ایزوله 20KHz (TR3) و مدار یکسوساز بعد از TR2 معمولا از یک شبکه فیلتر LC و یک دیود شاتکی تشکیل شده است که با انتگرال گیری از شکل موج مربعی با ضریب اتصال متغیر ولتاژ DC همواری در خروجی ایجاد می کند .

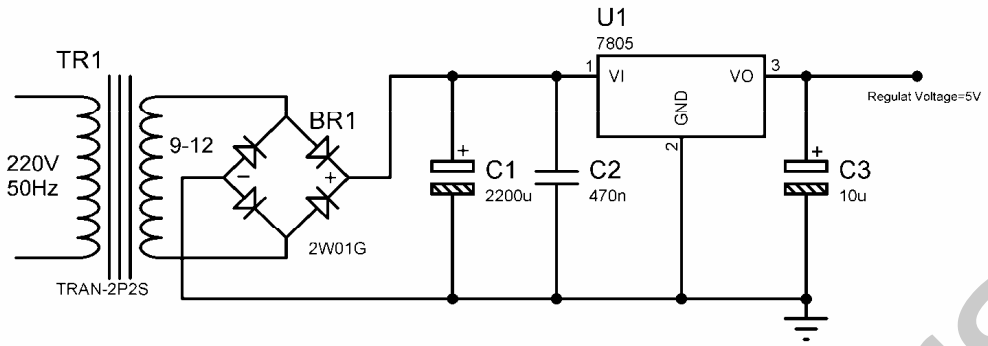
سطح ولتاژ DC خروجی با ضریب اتصال مرجع مربعی تولید شده توسط تثبیت کننده سوئیچینگ نسبت مستقیم دارد ، با اعمال یک فیدبک از ولتاژ خروجی تثبیت شده به تثبیت کننده سوئیچینگ می توان درصد تثبیت ولتاژ خروجی را افزایش داد به این صورت که تثبیت کننده ضریب اتصال PWM تولید شده را با مقدار ولتاژ خروجی MATCH می کند .

کارائی چنین سیستمی معمولا 80 درصد می باشد ولی در این روش سیگنال های رادیوئی که تاثیر نامطلوبی در مدارات دیجیتال حساس به نویز می گذارد ، به میزان زیادی ایجاد می شود، همچنین حذف ریپل و تثبیت کننده گی ولتاژ نیز در آن کمتر است ، به همین دلیل چنین سیستم هایی امروزه کمتر مورد استفاده قرار می گیرند .

مدار های تثبیت کننده ولتاژ 3 پایه

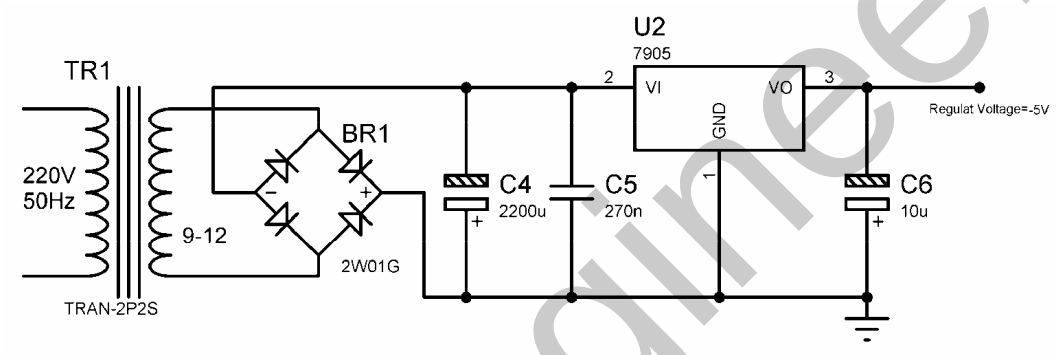
امروزه در اکثر موارد برای تثبیت کردن ولتاژ خروجی از IC های 3 پایه تثبیت کننده ولتاژ مانند IC های سری 78XXX برای ولتاژ های مثبت و سری 79XXX برای ولتاژ های منفی استفاده می شوند ، این نوع IC ها با مقادیر متنوعی از نظر ولتاژ جریان ارائه شده اند که این مقادیر با پسوند XXX در آن ها مشخص می شود اولین حرف از این پسوند نشان دهنده جریان می باشد حرف L به معنای حداکثر جریان خروجی 100mA و جای خالی به معنای جریان خروجی 1A و حرف S به معنای حداکثر جریان خروجی 2 آمپر می باشد همچنین دو رقم آخر نشان دهنده مقدار ولتاژ خروجی است که مقادیر استاندارد عبارتند از (24,15,12,9,8,6,5) ، بنابر این IC ، 7805 ولتاژ مثبت 5 ولت با جریان یک آمپر را ارائه می کند و IC ، 79L15 ولتاژ منفی 15 ولت با حداکثر جریان خروجی 100 میلی آمپر را ارائه می دهد .

شکل3-24 نحوه برقراری اتصالات برای IC، 7805 را نشان می دهد.



شکل3-24 نحوه برقراری اتصالات برای 7805

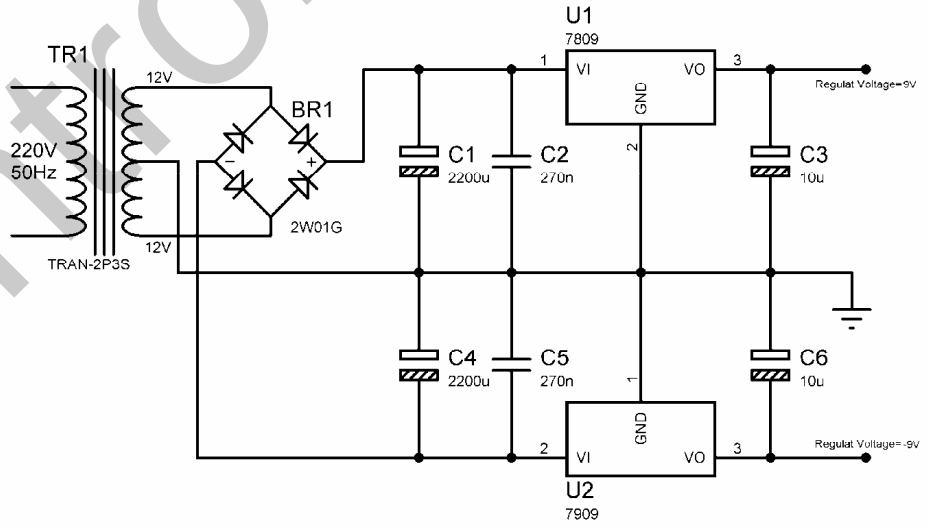
شکل 3-25 نحوه برقراری اتصالات برای IC ، 7905 را نشان می دهد.



شکل3-25 نحوه برقراری اتصالات برای 7905

توجه داشته باشید که همواره ولتاژ پایه مثبت خازن بایستی از پایه منفی آن بیشتر باشد . بنابراین در تعیین پلاریته خازن در مدار دقت کنید .

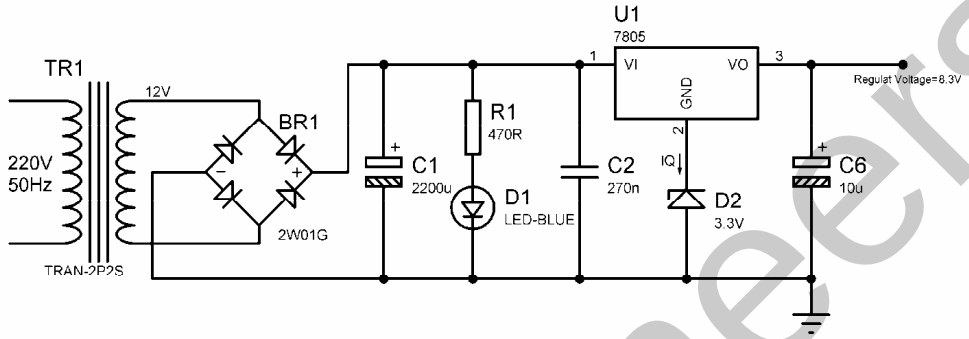
شکل 3-26 مدار کامل تثبیت کننده ولتاژ مضاعف 9 ولت و یک آمپر رانشان می دهد.



شکل 3-26 مدار کامل تثبیت کننده ولتاژ مضاعف 9 ولتی

نحوه ساختن ولتاژ خروجی متغیر با استفاده از IC های سری 78 , 79 :

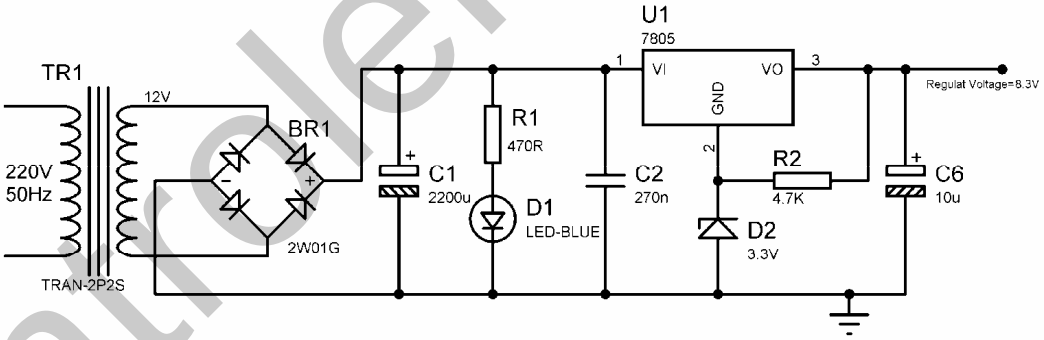
ولتاژ خروجی IC های 3 پایه نسبت به پایه مشترک (COMMON) آنها تنظیم می شود . این پایه معمولا به نقطه زمین متصل می شود اما اگر آن را با ولتاژ مناسبی بایاس کنیم می توانیم ولتاژ تثبیت شده خروجی را تا میزان مورد نظر افزایش دهیم. شکل 27-3 نحوه افزایش ولتاژ خروجی در این نوع IC ها را نشان می دهد .



شکل 27-3 نحوه افزایش ولتاژ خروجی در آی سی 7805

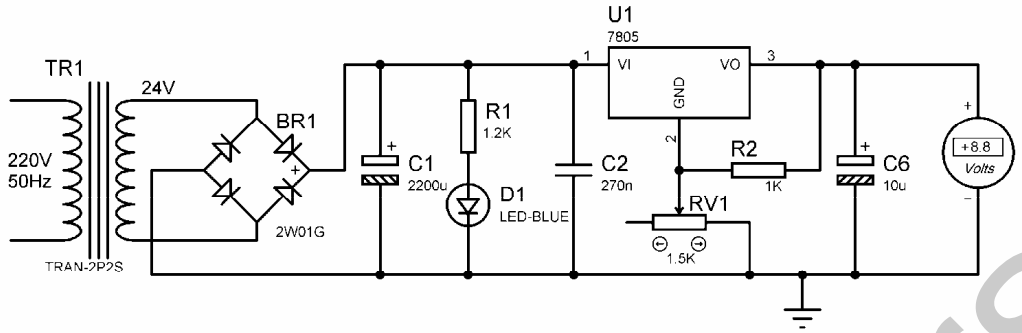
در شکل فوق ولتاژ تثبیت شده خروجی آی سی7805 که برابر با 5 می باشد با بایاس کردن پایه COMMOND آی سی با استفاده از دیود زینر 3.3V به ولتاژ تثبیت شده 8.31 تغییر می کند .

البته ولتاژ خروجی نسبت به تغییرات جریان سکون کمی متغیر است که با استفاده از از مدار شکل 28-3 می توان تاثیرات چنین تغییری را به حداقل رساند .



شکل 28-3 نحوه حذف تغییرات ولتاژ خروجی نسبت به تغییرات جریان سکون

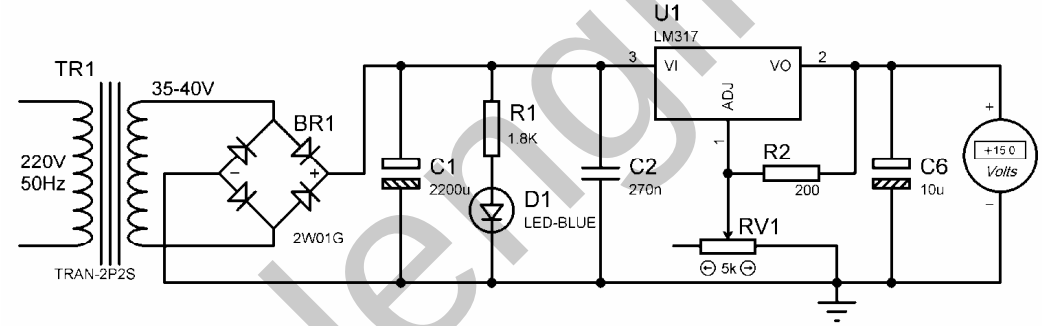
برای ایجاد ولتاژ خروجی قابل تنظیم نیز می توانید از مدار شکل 29-3 استفاده کنید ، در این مدار می توانید ولتاژ خروجی را با استفاده از RV1 در محدوده 5 تا 19 ولت تغییر دهید .



شکل 3-29 نحوه طراحی رگلاتور ولتاژ متغیر در محدوده 5 تا 19 ولت

اگر چه می توان IC های سری 79,78 را برای ایجاد ولتاژ متغیر (با دامنه تغییرات محدود) مورد استفاده قرار داد اما این ICها برای ارائه ولتاژ خروجی ثابت طراحی شده اند در صورتی که به ولتاژ های خروجی تثبیت شده متغیری در محدوده وسیع نیاز داشته باشید می توانید از IC های LM317 استفاده کنید . شکل 3-30 مدار پایه IC ، LM317 را برای داشتن ولتاژ تثبیت شده متغیر نشان می دهد . در مدار زیر ولتاژ خروجی از فرمول

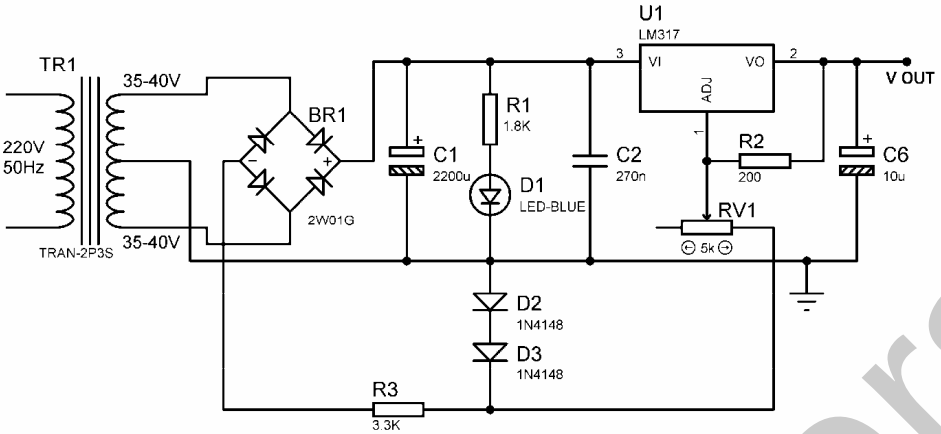
Vout = 1.25 (1 + RV1 / R1)



شکل 3-30 رگلاتور ولتاژ با خروجی قابل تنظیم از1.25V تا 30 ولت

محدوده تغییرات ولتاژ خروجی در مدار فوق 1.25 ولت تا 30 ولت می باشد .

شکل 3-31 نشان می دهد که چگونه می توان محدوده تغییرات ولتاژ خروجی را از صفر تا مقدار ماکزیمم ولتاژ ورودی افزایش داد برای این کار از خط تغذیه منفی و دو عدد دیود که با یکدیگر سری شده و ولتاژ RV1 را به 1.25 ولت می رسانند استفاده شده است .



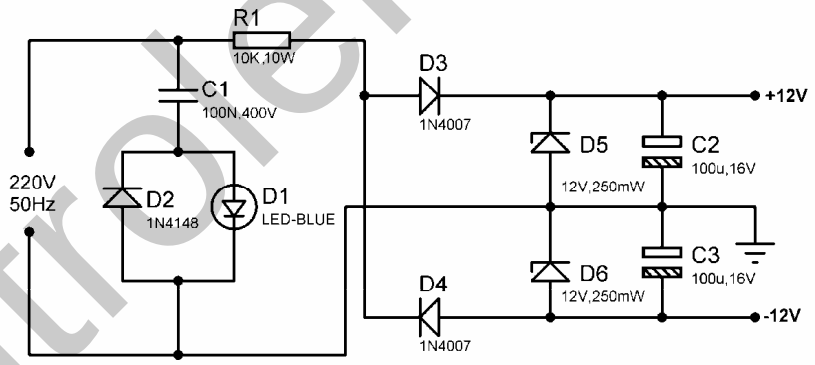
شکل 3-31 رگلاتور ولتاژ با خروجی قابل تنظیم از صفر تا 30 ولت

منبع تغذیه متقارن ±12V ساده ، بدون استفاده از ترانسفورماتور

مدار معرفی شده در این بخش یک منبع تغذیه مضاعف 12 ولت با جریان خروجی 200mA می باشد .

از LED برای نشان دادن وجود برق شهر استفاده شده است . یاد آور می شوم که خازن C1 بایستی ولتاژ 400 ولت را تحمل کند و هم چنین هنگام استفاده از مدار نبایستی به بخشهایی از مدار که دارای ولتاژ برق شهری باشد دست زد و برای دست زدن به خازن C1 بایستی ابتدا آن را تخلیه کرد .

ولتاژ ورودی پس از افت به وسیله R1 توسط دیود های D3 ، D4 یکسو سازی شده و توسط دیود های زنر و خازن های C3 ، C2 صاف می گردد.

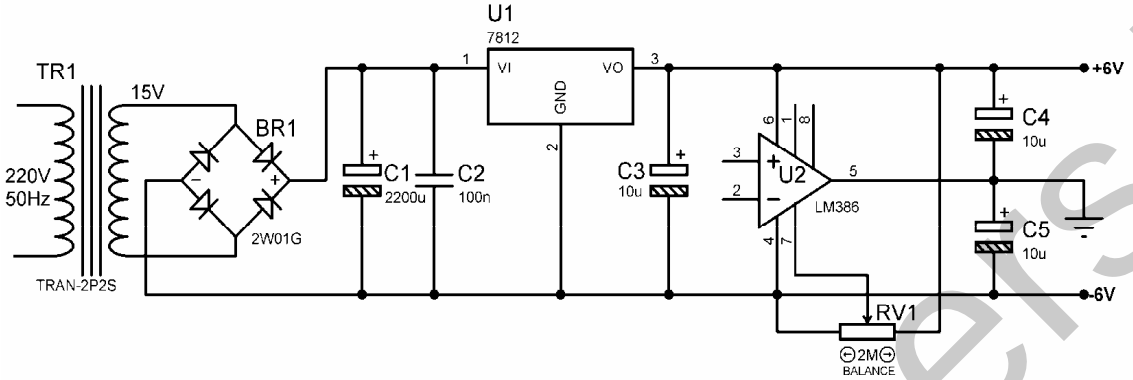


شکل 3-32 منبع تغذیه متقارن 12 ولتی ساده بودن استفاده از ترانسفور ماتور

تولید ولتاژ مثبت و منفی توسط IC , LM386

با استفاده از یک IC ، LM386 که یک تقویت کننده صوتی است و دو عدد خازن می توان ولتاژ مثبت و منفی بوجود آورد . پتانسیو متر RV1 برای تنظیم توازن (BALANCE) تغذیه دوبل به کار می رود ، توجه داشته باشید که ولتاژ های مثبت ومنفی نسبت به پایه خروجی LM386 که یک زمین مجازی است سنجیده می شود .

به عنوان مثال با اتصال IC مزبور به خط تغذیه 12 ولت می توان خروجی های 6+ و 6- ولت رانسبت به پایه خروجی LM386 بوجود آورد.

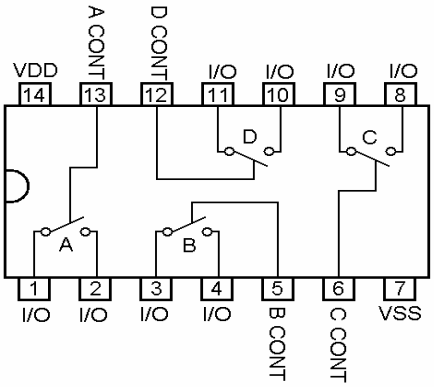


شکل 3-33 نحوه تولید ولتاژ های 6+ و 6- ولت از منبع تغذیه 12 ولتی با استفاده از LM386

کلید ها و سلکتور های انتخاب گر دو طرفه

کلید های دو طرفه CMOS را می توان به عنوان نوعی کلید تک پل دو طرفه در نظر گرفت عملکرد این نوع کلید ها مشابه کلید های تک پل مکانیکی بوده و قادر هستند سیگنال های دیجیتال و آنالوگ را در دو جهت انتقال دهند ، قطع و وصل این کلید ها از طریق اعمال سطح منطقی صفر یا یک به پایه کنترل صورت می گیرد . هم چنین امپدانس آن ها در حالت قطع نزدیک به بی نهایت و در حالت روشن در حد چند صد اهم است . فرکانس کاری کلید های دو طرفه CMOS بسیار بالا بوده و به 120 مگاهرتزمی رسد . در نتیجه عملکرد این نوع کلید ها چه در طراحی سیستم های آنالوگ و چه در طراحی سیستم های دیجیتال بسیار مفید خواهد بود به عنوان مثال می توان آن ها را در مداراتی که حامل سیگنال می باشند به کار برد در نتیجه مشکلات ناشی از تشعشع سیگنال و تداخل های ناخواسته و نویز های لحظه ای که معمولا از عوارض قطع و وصل کلید های مکانیک به شمار می روند بر طرف خواهد شد .

شکل 3-34 شماتیک داخلی مشترک برای IC های 4016B ، 4066B ، 74HC4016 ، 74HC4066 که در واقع از چهار کلید دو طرفه مستقل تشکیل شده اند را نشان می دهند.



Schematic Diagram Of 4066B,4016B
74HC4016,74HC4066

شکل 34-3 شماتیک داخلی مشترک برای آی سی های 4016B ، 4066B ، 74HC4016 ، 74HC4066

نحوه بایاسینگ کلید های دو طرفه

همانطور که گفته شد می توان کلید های دو طرفه CMOS را به منظور قطع و وصل سیگنال های دیجیتال و آنالوگ مورد استفاده قرار داد ولی در نظر داشته باشد این کلید ها بایستی نسبت به نوع سیگنالی که کنترل می کنند به خوبی بایاس شده باشند . هنگام استفاده از این کلید ها در مدارات دیجیتالی نبایستی ولتاژ سیگنال کنترل از VDD بیشتر و از VSS کمتر باشد .

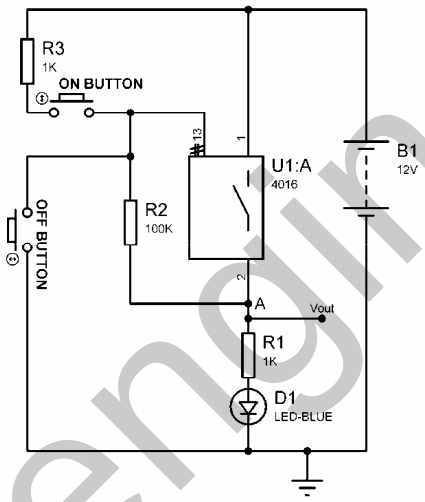
- خط تغذیه مثبت (VDD) بایستی به VCC متصل شود و مقدار آن باید از پیک مثبت سیگنالی که توسط کلید کنترل می شود بیشتر باشد .
- خط تغذیه منفی (VSS) بایستی به VCC- متصل شود و مقدار آن باید از پیک منفی سیگنالی که توسط کلید کنترل می شود بیشتر باشد .
- با توجه به این که این IC ها به مدار داخلی کنترل کننده سطوح منطقی می باشند اعمال سیگنال با ولتاژ 5Vبه پایه کنترل باعث وصل شدن کلید و اعمال ولتاژ صفر به پایه کنترل باعث قطع شدن آن خواهد شد.
- ولتاژ تغذیه در این نوع IC ها به 7.5 ± ولت محدود می شود.
- کلید های استفاده نشده در یک آی سی بایستی با اتصال پایه کنترل و یکی از پایه های I/O کلید به GND غیر فعال شوند.

جدول زیر پارامتر های مهم چهار گروه اصلی IC های خانواده 4066 ، 4016 را نشان می دهد.

پارامتر	4016B	4066B	74HC4016	74HC4066
ولتاژ تغذیه(VS) [VDD تا VSS]	3–15 ولت	3–15 ولت	2–12 ولت	2 – 12 ولت
مقاومت حالت روشن(VS= 5ولت)	520 اهم	250 اهم	70 اهم	40 اهم

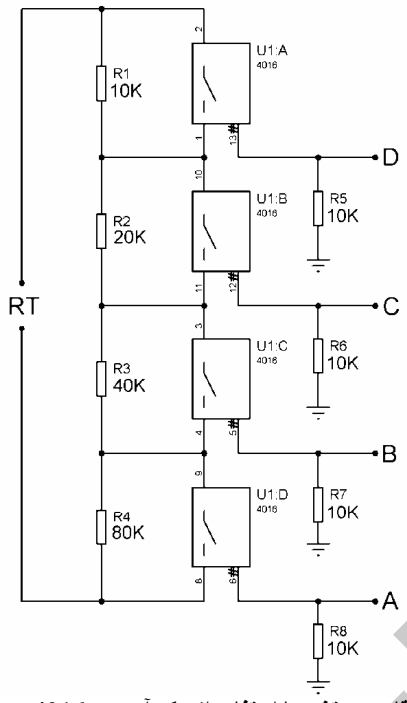
مقاومت حالت روشن(VS = 10ولت)	250 اهم	120 اهم	30 اهم	18 اهم
مقاومت حالت روشن (VS =15ولت)	200 اهم	80 اهم	–	–
پهنای باند (VS = 10ولت)	54 مگاهرتز	65 مگاهرتز	120 مگاهرتز	120 مگاهرتز
تاثیر متقابل سیگنال ها بین 2 کلید در فرکانس یک مگاهرتز(VS = 10ولت)	80 – دسی بل	50 – دسی بل	50 – دسی بل	50 – دسی بل
حداکثر تاخیر در روشن / خاموش شدن کلید (**VS = 10**ولت)	20 نانوثانیه	35 نانو ثانیه	10 نانو ثانیه	10 نانو ثانیه

شکل3-35 نحوه ساختن شاسی فشاری با قابلیت لچ کردن توسط IC های مزبور را نشان می دهد .



شکل3-35 نحوه ساختن شاسی فشاری با قابلیت لچ کردن توسط کلید های دو طرفه

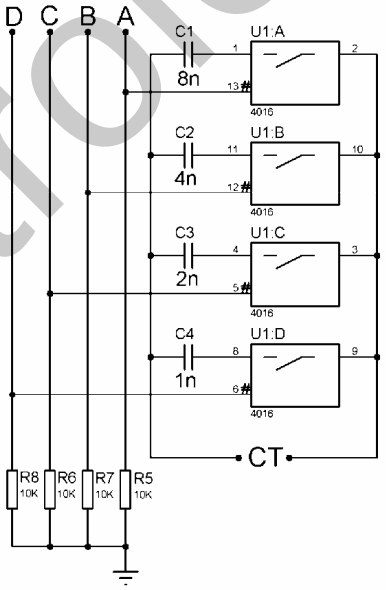
در شکل فوق ابتدا کلید قطع می باشد با فشار شاسی ON BUTTON ولتاژ مثبت تغذیه به پایه کنترل اعمال می شود. در این حالت ولتاژ نقطه A نیز برابر با VCC خواهد بود پس هنگامی که کلید وصل می شود پایه کنترل از طریق نقطه A برابر با VCC خواهد بود. در این حالت تا وقتی که شاسی OFF BUTTON را فشار نداده ایم کلید در حالت وصل خواهد بود . از کلید های دو طرفه می توان برای کنترل دیجیتالی کمیت های مختلفی مانند مقاومت ، ظرفیت خازنی ، امپدانس ، ضریب تقویت وغیره استفاده کرد شکل زیر نشان می دهد که چگونه می توان با استفاده از یک IC ، 4016 یک مقاومت متغیر ساخت .



A	B	C	D	RT
0	0	0	0	150K
0	0	0	1	140K
0	0	1	0	130K
0	0	1	1	120K
0	1	0	0	110K
0	1	0	1	100K
0	1	1	0	90K
0	1	1	1	80K
1	0	0	0	70K
1	0	0	1	60K
1	0	1	0	50K
1	0	1	1	40K
1	1	0	0	30K
1	1	0	1	20K
1	1	1	0	10K
1	1	1	1	0K

شکل 3-36 نحوه ساختن یک مقاومت متغیر با استفاده از یک آی سی 4016

همان طور که مشاهده می شود هر یک از مقاومت ها از دو برابر کردن مقاومت قبلی به دست می آید ، مقدار مقاومت متغیر نیز توسط اتصال کوتاه شدن یا نشدن هر یک از مقاومت ها تغییر پیدا می کند ، شما می توانید با عوض کردن مقدار متغیر های R1 ، R2، R3 ، R4 محدوده تغییرات مقاومت را تغییر دهید . برای ساختن خازن متغیر نیز می توانید از مدار شکل 3-37 استفاده کنید .



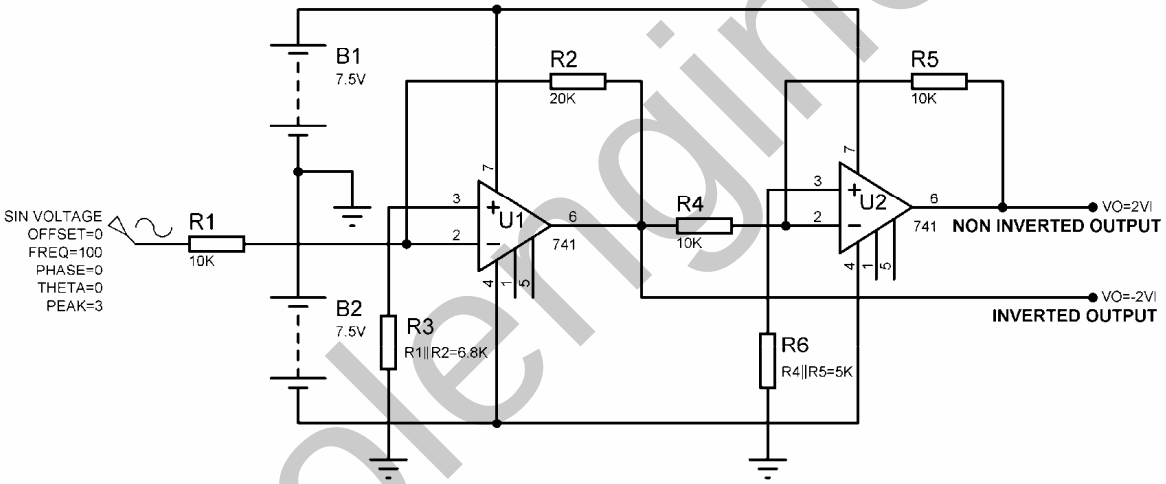
A	B	C	D	CT
0	0	0	0	0n
0	0	0	1	1n
0	0	1	0	2n
0	0	1	1	3n
0	1	0	0	4n
0	1	0	1	5n
0	1	1	0	6n
0	1	1	1	7n
1	0	0	0	8n
1	0	0	1	9n
1	0	1	0	10n
1	0	1	1	11n
1	1	0	0	12n
1	1	0	1	13n
1	1	1	0	14n
1	1	1	1	15n

شکل 3-37 نحوه ساختن خازن متغیر توسط IC های دوطرفه CMOS

نحوه استفاده از کلید های دو طرفه برای کنترل دیجیتالی ضریب بهره

شکل 3-38 را در نظر بگیرید آپ امپ U1 به عنوان یک تقویت کننده وارونساز مورد استفاده قرار گرفته است . ضریب تقویت سیگنال ورودی AV=-R2/R1=2 می باشد از آنجایی که جریان ورودی در پایانه مثبت و منفی آپ امپ 741 صفر نیست ، برای آنکه ولتاژ پایانه های مثبت و منفی با هم برابر شود بایستی مقاومت AC این دو پایانه نسبت به زمین با هم برابر باشد در نتیجه مقدار R3 از رابطه R1// R2 = 6.8K بدست می آید .

با توجه به این که سیگنال خروجی تقویت شده نسبت به سیگنال ورودی دارای اختلاف فاز 180 درجه می باشد، به منظور هم فاز کردن ورودی و خروجی می توان از یک تقویت کننده وارونساز دیگر با ضریب بهره یک در خروجی استفاده کرد . در شکل 3-38 از آپ امپ U2 به عنوان تقویت کننده وارونساز با ضریب بهره یک استفاده شده است . توجه داشته باشید که برای تقویت یک سیگنال مربعی بایستی مقدار مقاومتها را در حد مگا اهم افزایش دهید و مدار شکل 3-38 برای تقویت سیگنال سینوسی طراحی شده است .



شکل 3-38 نحوه طراحی یک تقویت کننده معکوس کننده و غیر معکوس کننده

پروژه میکروکنترلری کنترل دیجیتالی ضریب تقویت ولتاژ

حال با استفاده از کلید های دو طرفه و مدار شکل 3-38 یک تقویت کننده طراحی می کنیم که ضریب بهره آن به صورت دیجیتالی توسط میکروکنترلر AT90S2313 قابل کنترل می باشد . همچنین شما می توانید خروجی سیگنال تقویت شده را با اختلاف فاز 180 درجه (INVERTED) یا با اختلاف فاز صفر درجه (NON INVERTED) به صورت دیجیتالی تعیین کرده و به کانال A اسیلوسکوپ اعمال کنید . ضریب بهره در مدار زیر می تواند ½ ،1 ،2 باشد که توسط شاسی فشاری BUTTON2 تعیین می شود ، با هر بار فشار کلید

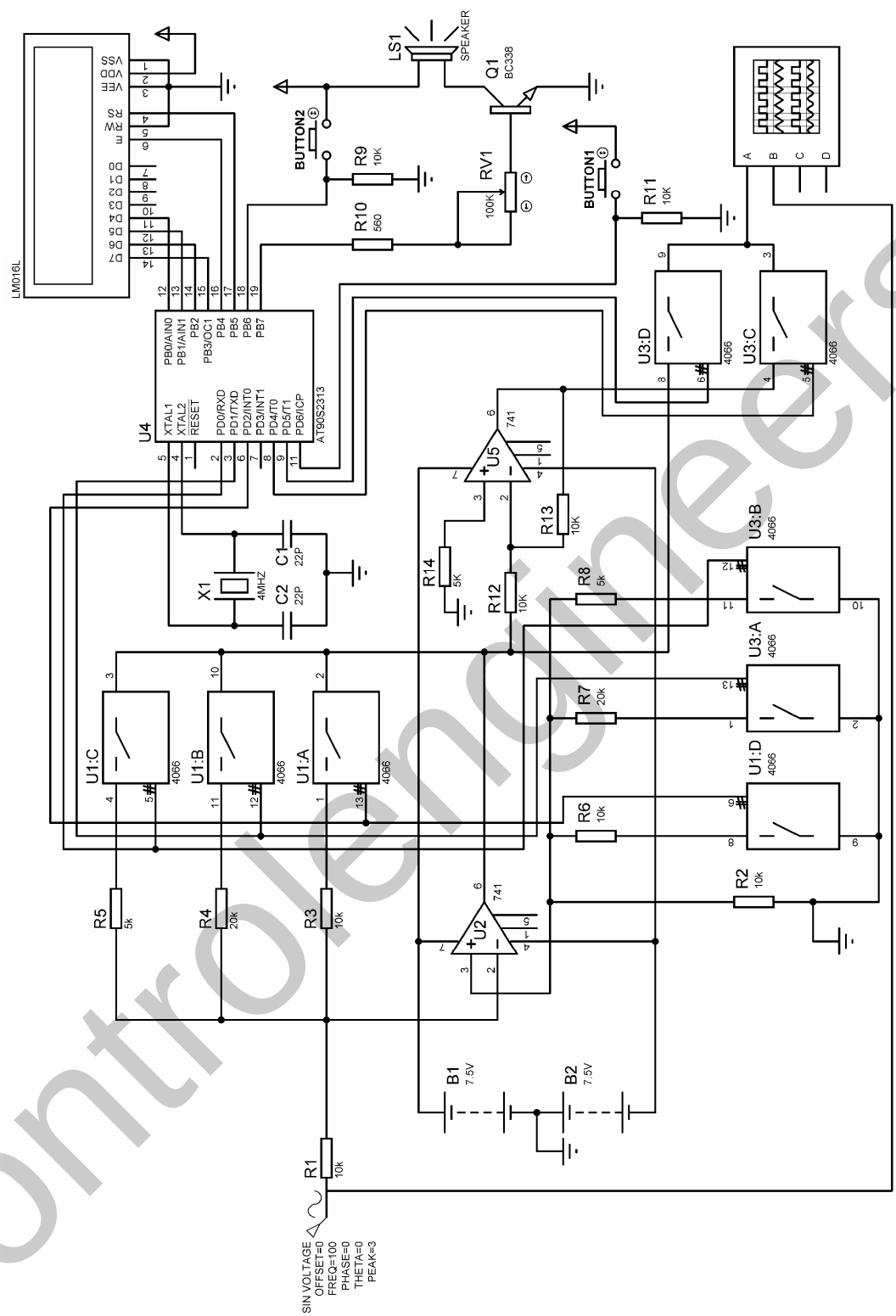
BUTTON2 عبارت نوشته شده در خط دوم LCD تغییر می کند . این عبارت می تواند AV=1/2 ، AV=1 ، AV=2 باشد .

همچنین اختلاف فاز سیگنال خروجی نسبت به سیگنال ورودی می تواند صفر درجه یا 180 درجه باشد که توسط BUTTON1 انتخاب می شود ، با هر بار فشار کلید BUTTON1 عبارت نوشته شده در خط اول LCD تغییر می کند این عبارت می تواند VO IS INVERTED یا VO=NON INVERTED باشد ، اختلاف فاز سیگنال خروجی برابر با مقدار نوشته شده بر روی LCD خواهد بود . سخت افزار پروژه در شکل 3-39 ارائه شده است .

برنامه نوشته شده برای پروژه به صورت زیرمی باشد .

```
'COMPILER:1.11.7.4
$regfile = "2313DEF.DAT"
$crystal = 4000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinb.0 , Db5 = Pinb.1 , Db6 = Pinb.2 , Db7 = Pinb.3_
                    , E = Pinb.4 , Rs = Pinb.5
Config Pinb.6 = Input
Config Pinb.7 = Output
Config Pind.0 = Output : X1_2 Alias Portd.0
Config Pind.1 = Output : X2 Alias Portd.1
Config Pind.2 = Output : X1 Alias Portd.2
Config Pind.4 = Output : Non_invert Alias Portd.4
Config Pind.5 = Output : Invert Alias Portd.5
Config Pind.6 = Input
Dim H As Byte , T As Bit
'-----------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "VO IS INVERTED "
Lowerline
Lcd "AV=1"
Set X1 : Reset X2 : Reset X1_2
Set Invert : Reset Non_invert
'-----------------------------------------------------------------------------------------------------------
Start_program:
If Pinb.6 = 1 Then
Incr H : If H > 2 Then H = 0
Sound Portb.7 , 50 , 150
End If
If Pind.6 = 1 Then
T = Not T : Sound Portb.7 , 50 , 150
End If
Select Case H:
'-----------------------------------------------------------------
Case Is = 0
Set X1 : Reset X2 : Reset X1_2
Locate 2 , 1 : Lcd "      "
Locate 2 , 1 : Lcd "AV=1"
'-----------------------------------------------------------------
Case Is = 1
Reset X1 : Set X2 : Reset X1_2
Locate 2 , 1 : Lcd "      "
Locate 2 , 1 : Lcd "AV=2"
'-----------------------------------------------------------------
```

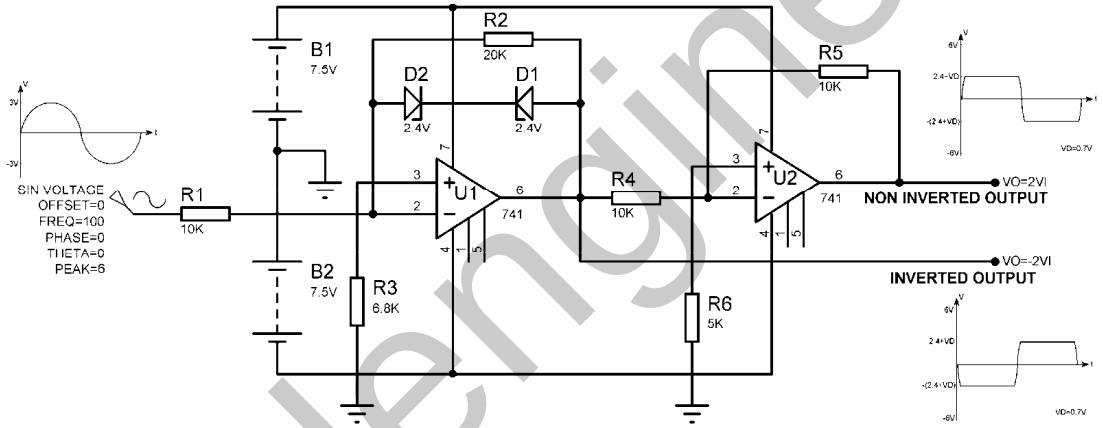
```
Case Is = 2
Reset X1 : Reset X2 : Set X1_2
Locate 2 , 1 : Lcd "      "
Locate 2 , 1 : Lcd "AV=1/2"
'-----------------------------------------------------------------
End Select
'-----------------------------------------------------------------
If T = 0 Then
Set Invert : Reset Non_invert
Locate 1 , 1
Lcd "VO IS INVERTED "
Else
Reset Invert : Set Non_invert
Locate 1 , 1
Lcd "VO=NON INVERTED"
End If
'-----------------------------------------------------------------
Waitms 100
Goto Start_program
'-----------------------------------------------------------------------------------------------------------------
```



شکل 39-3 شماتیک طراحی شده برای پروژه کنترل دیجیتال ضریب تقویت ولتاژ

نحوه استفاده از کلید های دو طرفه برای کنترل ولتاژ برش در مدار برشگر قله

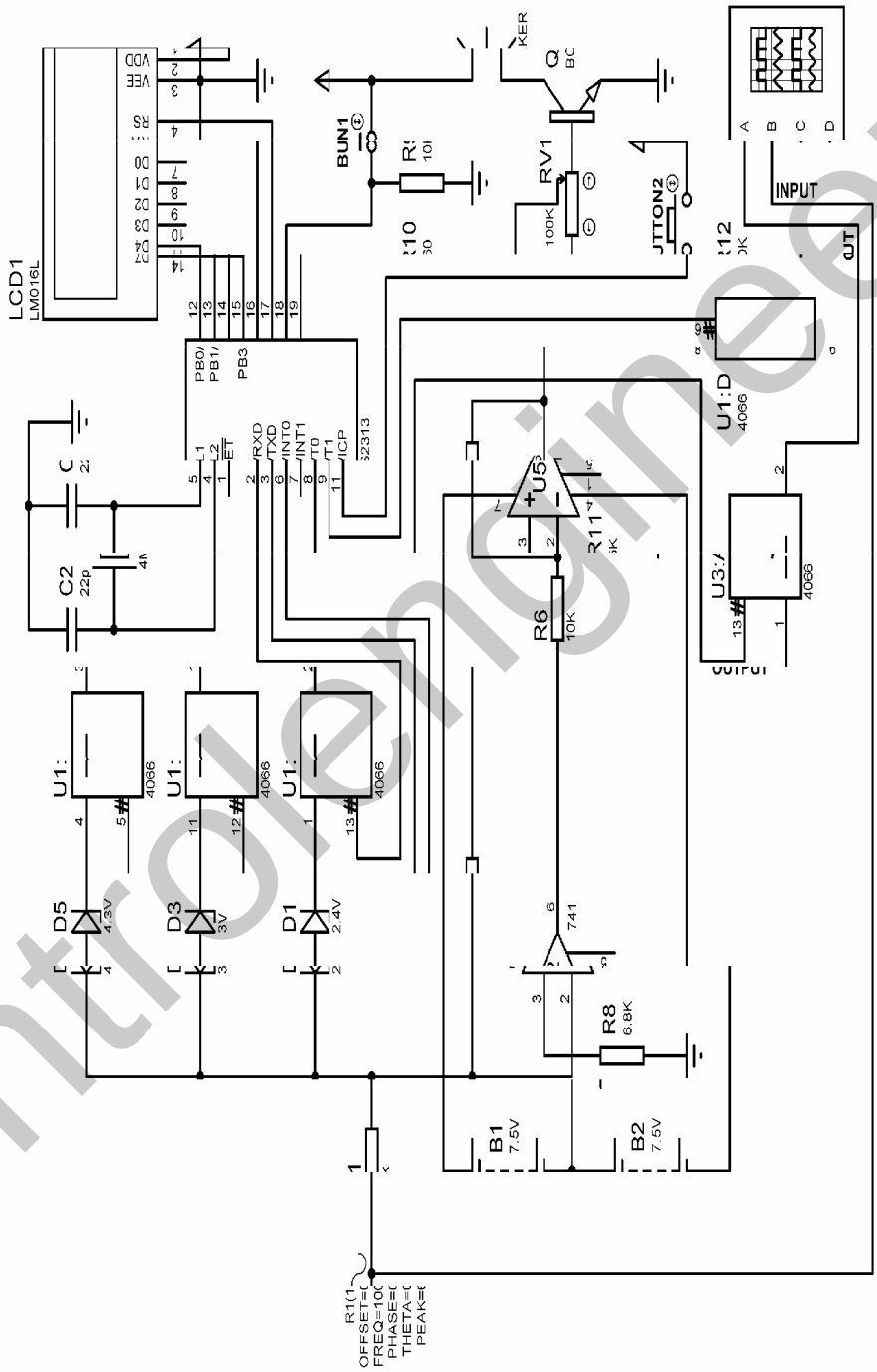
شکل3-40 قله آپ امپی را با خروجی INVERTED ، NON INVERTED نشان می دهد . در مدار زیر U1 به عنوان تقویت کننده وارونساز با ضریب بهره 2 و همچنین برش دهنده ولتاژ (2.4V+VD) برای نیم سیکل مثبت و منفی استفاده شده است . با توجه به این که ولتاژ خروجی ولتاژی است که روی R2 افت می کند . می توان با محدود کردن افت ولتاژ R2 توسط دیود زنر مقداری از نیم سیکل منفی را برش داد. همانگونه که می دانید دیود زنر در بایاس معکوس مورد استفاده قرار می گیرد و زمانی که در بایاس مستقیم باشد مانند یک دیود معمولی عمل خواهد کرد. بنابراین اگرVD(افت ولتاژ مستقیم دیود)را برابر با0.7 ولت فرض کنیم ولتاژ برش برابر با (0.7+2.4) ولت خواهد بود آمپ امپ U2 نیز به عنوان تقویت کننده معکوس کننده با ضریب بهره یک به منظور ایجاد خروجی NON INVERTED مورد استفاده قرار گرفته است .



شکل 3-40 مدار برشگر قله با خروجی INVERTED و NON INVERTED

حال با استفاده از کلید های دو طرفه و مدار فوق یک برشگر قله طراحی می کنیم که ولتاژ برش آن به صورت دیجیتالی توسط میکروکنترلر AT90S2313 قابل کنترل می باشد ، همچنین شما می توانید خروجی سیگنال برش داده شده را با اختلاف فاز 180 درجه (INVERTED) یا با اختلاف فاز صفر درجه (NON INVERTED) به صورت دیجیتالی تعیین کرده و به کانالA اسیلوسکوپ اعمال کنید ولتاژ برش در مدار زیر می تواند 2.4V ، 3V و 4.3V باشد که توسط شاسی فشاری BUTTON1 تعیین می شود. با هر بار فشار کلید BUTTON1 ، عبارت نوشته شده در خط دوم LCD تغییر پیدا می کند ، این عبارت می تواند CUT VOLTAGE=2.4 ، CUT VOLTAGE= 4.3 و همچنین CUT VOLTAGE=3 باشد. ولتاژ برش مدار برابر با مقدار نوشته شده بر روی LCD خواهد بود .هم چنین اختلاف فاز سیگنال خروجی نسبت به سیگنال ورودی می تواند صفر درجه یا 180 درجه باشد که توسط BUTTON1 انتخاب می شود . با هر بار فشار کلید BUTTON2 عبارت نوشته شده در

خط اول LCD تغییر می کند این عبارت می تواند VO IS INVERTED، VO=NON INVERTED باشد . اختلاف فاز سیگنال خروجی برابر با مقدار نوشته شده بر روی LCD خواهد بود. شکل 3-41 شماتیک طراحی شده برای پروژه را نشان می دهد .



شکل 3-41 شماتیک مدار برشگر دیجیتالی قله با استفاده از AT90S2313 و کلید های دوطرفه CMOS

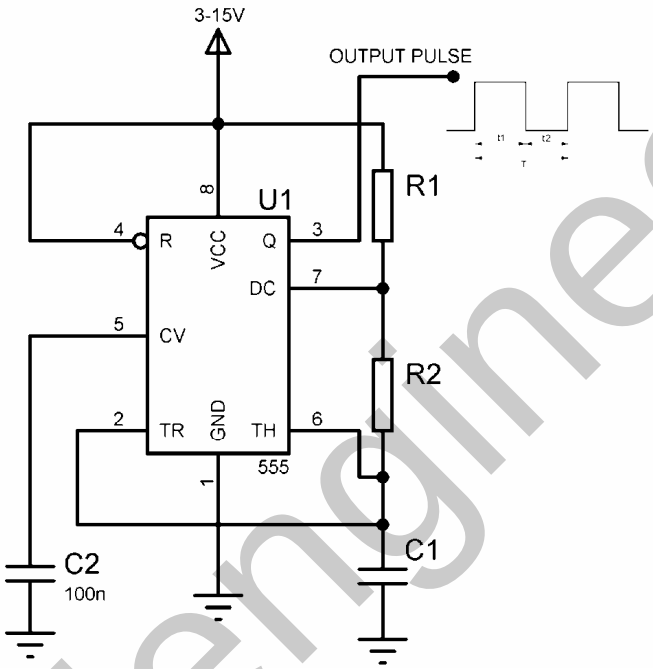
برنامه نوشته شده برای پروژه به صورت زیر است .

```
'COMPILER:1.11.7.4
$regfile = "2313DEF.DAT"
$crystal = 4000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinb.0 , Db5 = Pinb.1 , Db6 = Pinb.2 , Db7 = Pinb.3_
                    , E = Pinb.4 , Rs = Pinb.5
Config Pinb.6 = Input
Config Pinb.7 = Output
Config Pind.0 = Output : 2_4v Alias Portd.0
Config Pind.1 = Output : 3v Alias Portd.1
Config Pind.2 = Output : 4_3v Alias Portd.2
Config Pind.4 = Output : Invert Alias Portd.4
Config Pind.5 = Output : Non_invert Alias Portd.5
Config Pind.6 = Input
Dim H As Byte , T As Bit
'-----------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "VO IS INVERTED"
Lowerline
Lcd "CUT VOLTAGE=2.4"
Set 2_4v : Reset 3v : Reset 4_3v
Set Invert : Reset Non_invert
'-----------------------------------------------------------------
Start_program:
If Pinb.6 = 1 Then
Incr H : If H > 2 Then H = 0
Sound Portb.7 , 50 , 150
End If
If Pind.6 = 1 Then
T = Not T : Sound Portb.7 , 50 , 150
End If
Select Case H:
'-----------------------------------------------------------------
Case Is = 0
Set 2_4v : Reset 3v : Reset 4_3v
Locate 2 , 1 : Lcd "CUT VOLTAGE=2.4"
'-----------------------------------------------------------------
Case Is = 1
Reset 2_4v : Set 3v : Reset 4_3v
Locate 2 , 1 : Lcd "CUT VOLTAGE=3  "
'-----------------------------------------------------------------
Case Is = 2
Reset 2_4v : Set 3v : Reset 4_3v
Locate 2 , 1 : Lcd "CUT VOLTAGE=4.3"
'-----------------------------------------------------------------
End Select
'-----------------------------------------------------------------
If T = 0 Then
Set Invert : Reset Non_invert
Locate 1 , 1
Lcd "VO IS INVERTED"
Else
Reset Invert : Set Non_invert
Locate 1 , 1
Lcd "VO=NON INVERTED"
End If
'-----------------------------------------------------------------
Waitms 50
Goto Start_program
```

نحوه طراحی مولد موج مربعی :

به طور معمول در مدارات دیجیتالی از آی سی 555 به عنوان مولد موج مربعی یا مولتی ویبراتور آاستابل استفاده می شود ، مدار پایه طراحی مولتی ویبراتور آاستابل توسط IC تایمر 555 در شکل 3-42 نشان داده شده است .

زمانی که از مدار شکل 3-42 به عنوان مولتی ویبراتور استفاده می کنید مقدار ماکزیمم موج مربعی خروجی برابر با VCC و مقدار مینیمم آن برابر با صفر خواهد بود .



شکل 3-42 ، نحوه استفاده از آی سی NE555 به عنوان مولد موج مربعی

به عنوان مثال برای تولید یک سیگنال مربعی با دامنه 5 ولت و فرکانس 5 کیلوهرتز و زمان وظیفه 65 درصد در پایه خروجی به صورت زیر عمل می کنیم .

T=1/FOUT=1/5KHz=200US

T1=(زمان وظیفه) * T=(65/100)*200US=130US

t2=T-t1=200US-130US=70US

RT=R1+R2=VCC/0.03=5/0.03=1666.66R

برای این که ماکزیمم موج مربعی خروجی برابر با 5 ولت باشد بایستی VCC را برابر 5 ولت انتخاب کنیم .

C1=T1/(0.693RT)=0.12UF

R2=T2/(0.693C1)=912.88R

رنج استاندارد 820 اهم را برای R2 انتخاب می کنیم.

R1=RT-R2=753.78

رنج استاندارد 820 اهم را برای R1 در نظر می گیریم .

نکته : در تعیین مقادیر استاندارد مقاومت ها نزدیک ترین رنج استاندارد را انتخاب کنید .

در این حالت خروجی تایمر 555 یک سیگنال مربعی با مقدار ماکزیمم 5 ولت و مقدار منیمم صفر ولت می باشد برای ایجاد تقارن بین پیک مثبت و منفی بایستی مقدار ماکزیمم خروجی در ولتاژ 2.5 ولت قفل شود .

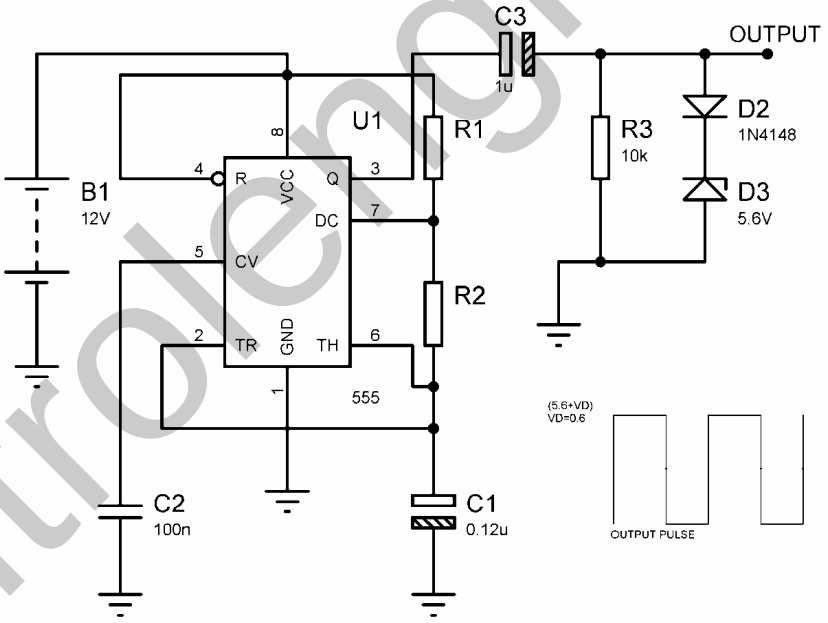
به عنوان مثال یک مولد موج مربعی 5KHz با آی سی 555 طراحی کرده ایم که ماکزیمم ولتاژ خروجی آن +12V می باشد ، به منظور ایجاد تقارن بین پیک مثبت و پیک منفی می خواهیم ماکزیمم ولتاژ خروجی را در +6V قفل کنیم اگر مقاومت خروجی آی سی 555 را 100 اهم در نظر بگیریم نحوه طراحی مدار قفل کننده به صورت زیر خواهد بود .

T=1/F=1/5K=0.2mS
PW= T/2=0.1mS
C3=PW/R0(555)=0.1mS/100=1UF

برای این که کجی خروجی بیش از یک درصد نباشد مقدار R3 به صورت زیر تعیین می شود .

R3=PW/(0.01*C3)=10K

شماتیک مدار طراحی در شکل 3-43 ارائه شده است .

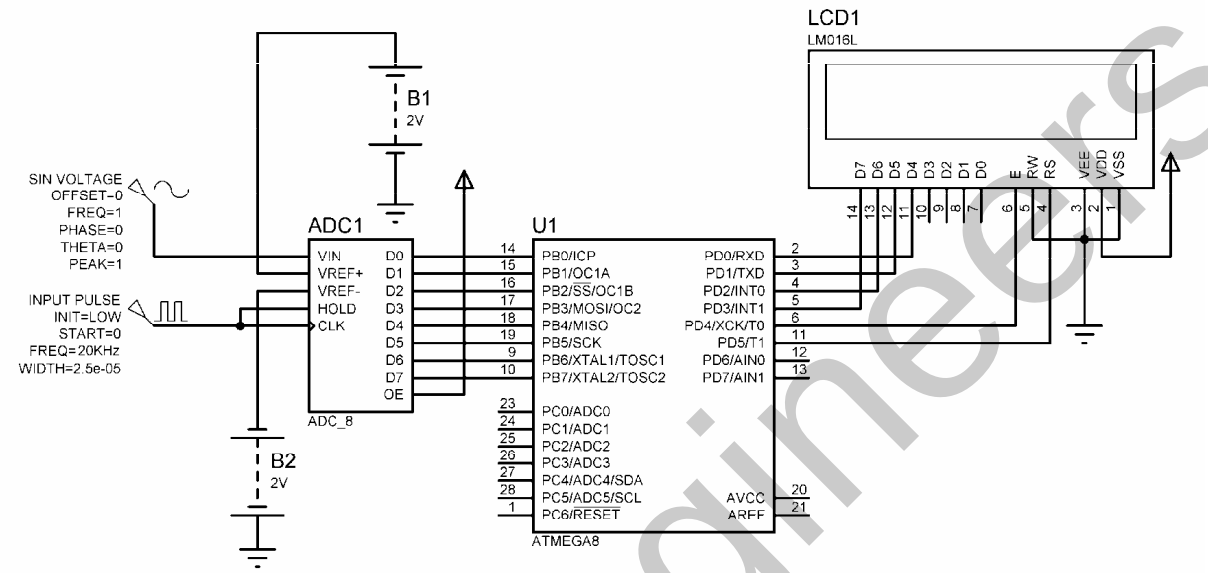


شکل 3-43 نحوه ایجاد تقارن بین پیک مثبت و منفی

نحوه طراحی مولد موج سینوسی با میکروکنترلر AVR

برای این کار ابتدا از یک سیگنال سینوسی توسط ADC (مبدل آنالوگ به دیجیتال) نمونه برداری کرده سپس نمونه های گرفته شده را به ورودی یک DAC (مبدل دیجیتال به آنالوگ) اعمال می کنیم در این حالت در خروجی

DAC سیگنال سینوسی خواهیم داشت . برای نمونه برداری از سیگنال سینوسی می توانید از مدار شکل 3-44 استفاده کنید در این مدار از یک سیگنال سینوسی ورودی 500 نمونه گرفته می شود و پس از نمونه برداری نمونه ها یک به یک بر روی LCD نمایش داده می شوند و شما می توانید آن ها را یادداشت کنید . سخت افزار مدار به صورت زیر می باشد .



شکل 3-44 نمونه برداری از سیگنال سینوسی ورودی

برنامه نوشته شده برای سخت افزار شکل 3-44 به صورت زیر می باشد .

```
'COMPILER:1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
                        , E = Pind.4 , Rs = Pind.5
Config Portb = Input
Dim Adc_of_sin(501) As Byte , H As Word
'-----------------------------------------------------------------------------------------
For H = 1 To 500 Step 1
Adc_of_sin(h) = Pinb
Waitms 2
Next H
'-----------------------------------------------------------------------------------------
Start_writeing:
For H = 1 To 500 Step 1
Cls : Home
Lcd H ; "=" ; Adc_of_sin(h)
Wait 1.5
Next H
Goto Start_writeing
'-----------------------------------------------------------------------------------------
```

پروژه تولید موج سینوسی با دامنه متغیر با استفاده از میکروکنترلر AVR

حال می توانیم با اعمال اعداد نمونه برداری شده به ورودی یک DAC (مبدل دیجیتال به آنالوگ) در خروجی آن سیگنال سینوسی داشته باشیم ، اگر به خاطر داشته باشید در مثال قبل از سیگنال سینوسی ورودی 500 نمونه گرفتیم از آن جا که یادداشت کردن 500 نمونه و نوشتن آن در برنامه کمی مشکل است در پروژه زیر از 250 نمونه برای تولید موج سینوسی استفاده شده است . نحوه ایجاد موج سینوسی به این صورت می باشد که ابتدا نمونه های گرفته شده از اولین نمونه تا نمونه 250 ام ، یک به یک و به صورت صعودی به ورودی DAC اعمال شده ، سپس از نمونه 250 ام تا نمونه اول به صورت نزولی به ورودی DAC اعمال می شود خروجی می تواند دارای دامنه 1VP-P یا 5VP-P باشد . شما می توانید مقدار VO(P-P) را با استفاده از کلید INPUT BUTTON بین دو مقدار ذکر شده تغییر داده و با فشار کلید OK آن را در خروجی داشته باشید . سخت افزار طراحی شده برای پروژه در شکل 3-45 ارائه شده است .

برنامه نوشته شده برای پروژه به صورت زیر می باشد .

```
'COMPILER:1.11.7.4
$regfile = "M16DEF.DAT"
$crystal = 16000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3_
                                , E = Pina.4 , Rs = Pina.5
Config Portd = Output
Config Pinb.2 = Input
Config Pinb.5 = Input
Config Pinb.0 = Output
Config Pinc.0 = Output : 5v Alias Portc.0
Config Pinc.1 = Output : 1v Alias Portc.1
Dim V As Byte , T As Bit , H As Word
'-----------------------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "ENTER VOLTAGE"
Lowerline
Lcd "VO IS = 1Vp-p"
'-----------------------------------------------------------------------------------------------------------
Start_program:
If Pinb.2 = 1 Then
T = Not T
Sound Portb.0 , 50 , 150
Bitwait Pinb.2 , Reset
End If
'-----------------------------------------------------------
If T = 0 Then
Set 1v : Reset 5v
Locate 2 , 1
Lcd "VO IS = 1Vp-p"
Else
Reset 1v : Set 5v
Locate 2 , 1
Lcd "VO IS = 5Vp-p"
End If
'-----------------------------------------------------------
```

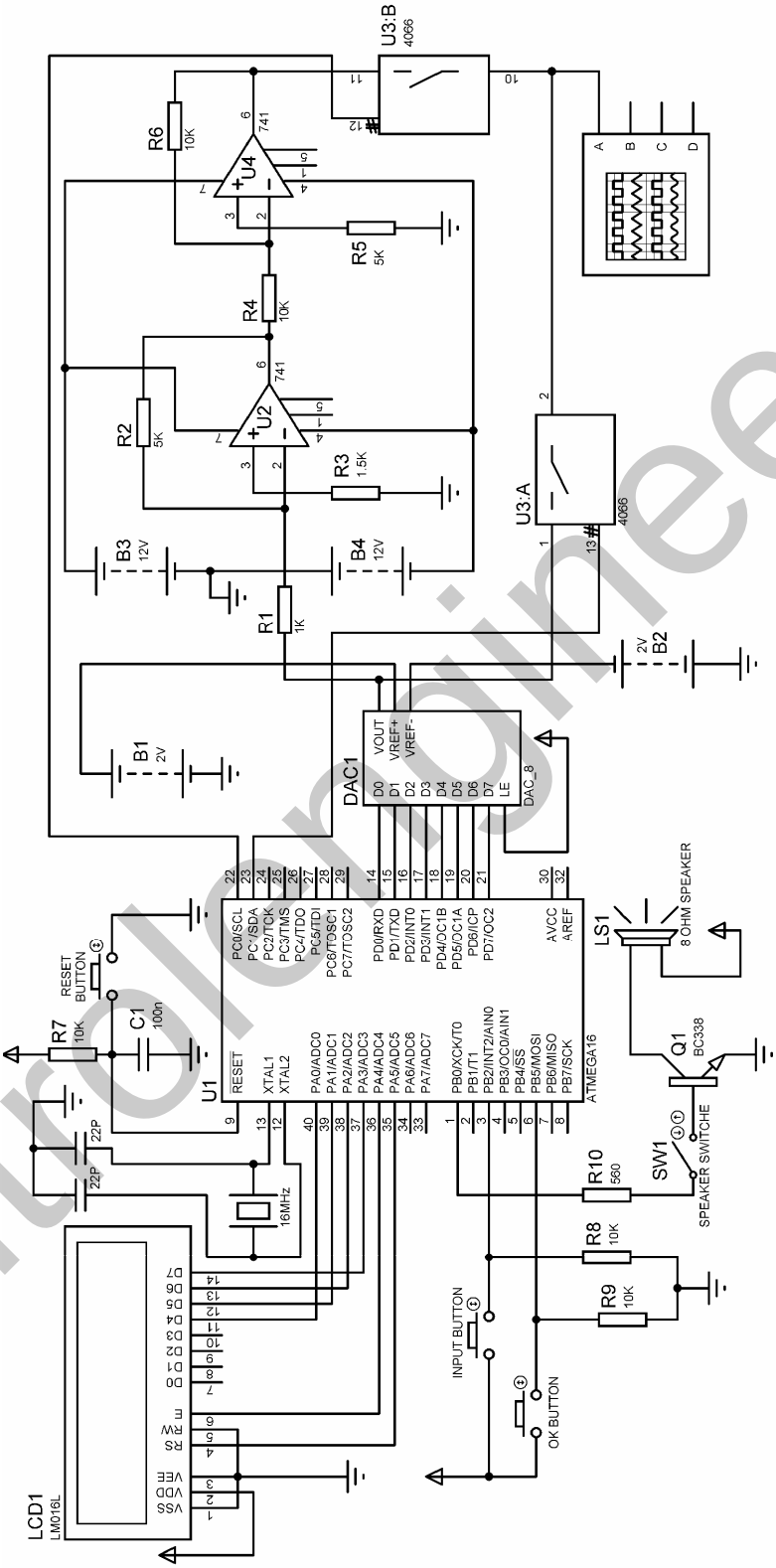
```
If Pinb.5 = 1 Then
Sound Portb.0 , 50 , 150
Goto H1 : End If
Waitms 50
Goto Start_program
'PROGRAM OF SIN VOLTAGE-----------------------------------------------------------------------
H1:
Do
For H = 0 To 250 Step 1
Portd = Lookup(h , Adc_value)
Waitus 20
Next H
'-------------------------------------------------------------
For H = 250 To 0 Step -1
Portd = Lookup(h , Adc_value)
Waitus 20
Next H
Loop
'END OF PROGRAM-----------------------------------------------------------------------------------------
'ADC VALUE OF SIN VOLTAGE
Adc_value:
Data 160 , 160 , 160 , 160 , 160 , 160 , 160 , 159
Data 159 , 159 , 159 , 159 , 159 , 159 , 159 , 158
Data 158 , 158 , 158 , 158 , 158 , 158 , 158 , 157
Data 157 , 157 , 157 , 157 , 157 , 157 , 156 , 156
Data 156 , 156 , 156 , 155 , 155 , 155 , 155 , 155
Data 154 , 154 , 154 , 154 , 154 , 153 , 153 , 153
Data 153 , 152 , 152 , 152 , 152 , 151 , 151 , 151
Data 151 , 150 , 150 , 150 , 149 , 149 , 149 , 148
Data 148 , 148 , 148 , 147 , 147 , 147 , 146 , 146
Data 146 , 145 , 145 , 145 , 144 , 144 , 144 , 143
Data 143 , 143 , 142 , 142 , 141 , 141 , 141 , 140
Data 140 , 140 , 139 , 139 , 138 , 138 , 138 , 137
Data 137 , 137 , 136 , 136 , 135 , 135 , 135 , 134
Data 134 , 133 , 133 , 133 , 132 , 132 , 131 , 131
Data 131 , 130 , 130 , 129 , 129 , 129 , 128 , 128
Data 127 , 127 , 127 , 126 , 126 , 125 , 125 , 125
Data 124 , 124 , 123 , 123 , 123 , 122 , 122 , 121
Data 121 , 121 , 120 , 120 , 120 , 119 , 119 , 118
Data 118 , 118 , 117 , 117 , 116 , 116 , 116 , 115
Data 115 , 115 , 114 , 114 , 113 , 113 , 113 , 112
Data 112 , 112 , 111 , 111 , 111 , 110 , 110 , 110
Data 109 , 109 , 109 , 108 , 108 , 108 , 107 , 107
Data 107 , 106 , 106 , 106 , 106 , 105 , 105 , 105
Data 104 , 104 , 104 , 104 , 103 , 103 , 103 , 103
Data 102 , 102 , 102 , 102 , 101 , 101 , 101 , 101
Data 100 , 100 , 100 , 100 , 100 , 99 , 99 , 99
Data 99 , 99 , 99 , 98 , 98 , 98 , 98 , 98 , 98 , 98 , 97 , 97 , 97 , 97 , 97 , 97
Data 97 , 97 , 97 , 97 , 96 , 96 , 96 , 96 , 96 , 96 , 95 , 95 , 95 , 95 , 95 , 95
Data 95 , 95 , 95 , 95 , 95 , 95 , 94 , 94 , 94 , 94 , 94 , 94 , 94 , 94 , 94 , 94
'END OF ADC VALUE
'-------------------------------------------------------------------------------------------------------------
```

فرکانس تغریبی سیگنال سینوسی تولید شده با استفاده از فرمول زیر محاسبه می شود .

T= (تعداد نمونه ها) * (تاخیر بین ارسال دو نمونه) =500*20US=0.01S

F=1/T=1/0.01=100Hz

سخت افزار پروژه به صورت زیر است .

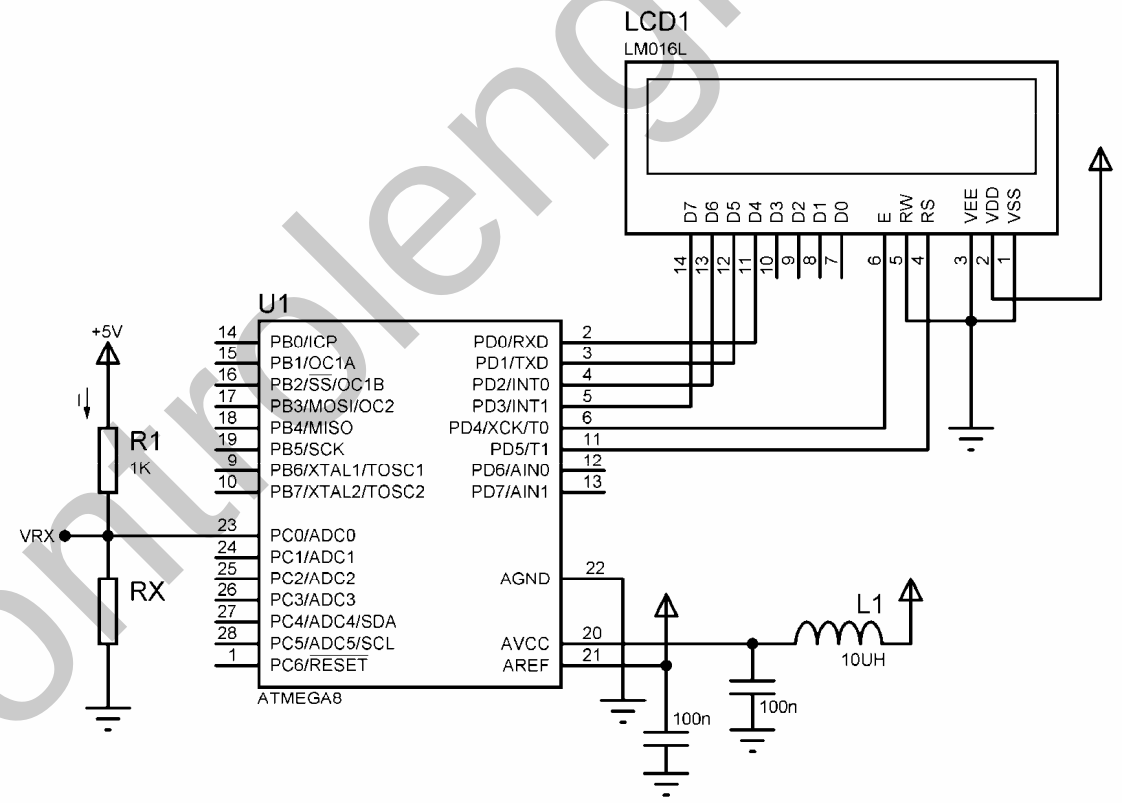


شکل 3-45 شماتیک طراحی شده برای پروژه مولد موج سینوسی با دامنه خروجی 1V(P-P) و 5V(P-P)

طراحی اهمتر دیجیتال با استفاده از میکروکنترلر AVR

برای طراحی اهمتر دیجیتالی می توانید از مدار ساده شکل 3-46 استفاده کنید نحوه عملکرد مدار بدین صورت می باشد که ابتدا مقدار افت ولتاژ روی مقاومت RX توسط میکرو کنترلر اندازه گیری می شود . سپس با استفاده از فرمول I=(5-VRX)/1K مقدار جریان محاسبه می شود ، با توجه به این که جریان RX و R1 با هم برابر است مقدار RX از رابطه RX=VRX/I بدست می آید.

با توجه به این که در این پروژه ازADC داخلی میکروAVR به عنوان مبدل آنالوگ به دیجیتال استفاده شده است. و دقتADC داخلی میکروکنترلرهایAVR در حد 10 بیتی است فاصله بین ولتاژ مرجع و زمین به 1024 قسمت مساوی تقسیم خواهد شد. در این حالت مقدار ولتاژ اعمالی برای یک شماره صعود عدد دیجیتال شده برابر خواهد بود با 5/1024=0.005V به عنوان مثال اگر VRX برابر با 2.5 ولت باشد مقدار عدد دیجیتال شده توسط ADC برابر با 2.5/0.005=500 خواهد بود. برای داشتن مقدار ولتاژ اعمالی به ورودی ADC بایستی مقدار بدست آمده را در 0.005 ضرب کنیم .



شکل 3-46 شماتیک طراحی شده برای اهمتر دیجیتال

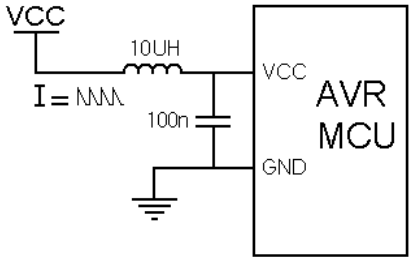
برنامه نوشته شده برای پروژه به صورت است.

```
'COMPILER:1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
                        , E = Pind.4 , Rs = Pind.5
Config Adc = Single , Prescaler = Auto
Dim Vrx As Single , Adc_value As Word , Vr1 As Single , I As Single
Dim Rx As Single
'-----------------------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "OHMMETER "
Lowerline
Lcd "R IS "
Start Adc
'-----------------------------------------------------------------------------------------------------------
Start_program:
Adc_value = Getadc(0)
Vrx = Adc_value * 0.005
'-----------------------------------------------
If Vrx = 5 Then
Cls : Home
Lcd "ERROR"
Goto Start_program
End If
'-----------------------------------------------
Vr1 = 5 - vrx
I = Vr1 / 1000
Rx = Vrx / I
Cls : Home
Lcd "OHMMETER "
Lowerline
Lcd "R IS " ; Rx
Waitms 100
Goto Start_program
'-----------------------------------------------------------------------------------------------------------
```

حذف پرش های ناخواسته جریان در مدارات میکروکنترلری

با نگاهی به DATASHIT میکروکنترلر های AVR به نظر می رسد که این میکروکنترلرها در مقابل تغییرات وسیع تغذیه حساس نیستند و جریان کمی در حد میلی آمپر مصرف می کنند اما با توجه به سوئیچینگ مدارات دیجیتالی ممکن است پرش های جریان شدیدی بالاتر از مقدار مشخص شده در خطوط تغذیه ایجاد شود.

برای حل این مشکل منابع تغذیه فیلترهایی به پایه های میکروکنترلر متصل می شود . فیلتر حفاظت مناسب تغذیه از پرش های جریان در شکل 3-47 نشان داده شده است .



شکل 3-47 فیلتر حفاظت از پرش های جریان

سلف سری قرار داده شده در این مدار بایستی تا حد امکان کوچک باشد تا منجر به افت ولتاژ تغذیه نگردد همچنین خازن به کار برده شده باید حدالامکان به پایه های میکروکنترلر نزدیک باشد .

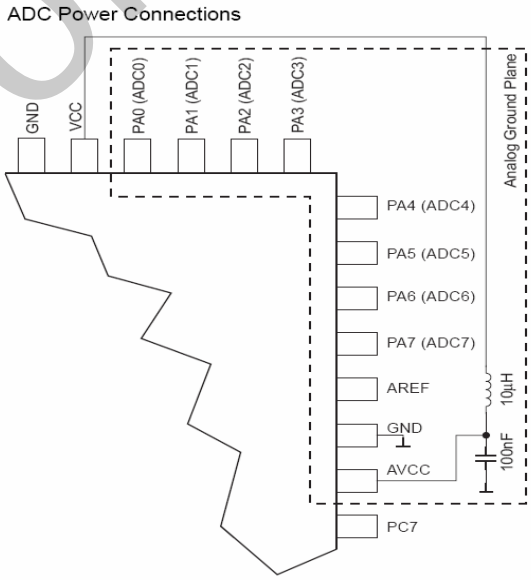
مقادیر پیشنهادی برای سلف (RFC) 10UH وبرای خازن 100nF می باشد

تکنیک های کاهش نویز ADC

مدارات دیجیتالی داخلی و خارجی میکروکنترلر تولید نویز می کنند که بر روی دقت اندازه گیری آنالوگ تاثیر می گذارند . در صورت نیاز به دقت تبدیل بالا ، سطح نویز آنالوگ با به کار بردن روش های زیر کاهش می یابد .

۱- مسیر سیگنال آنالوگ تا حد امکان باید کوتاه باشد . این مسیر ها نزدیک مسیر زمین آنالوگ و دور از مسیر های دیجیتال و سوئیچینگ فرکانس بالا قرار داده می شوند .

۲- پایه AVCC از طریق شبکه LC (یک سلف ده میکروهانری و خازن سری صد نانو فاراد که به زمین متصل می شود ، به طوریکه سلف به پایه VCC و محل اتصال سلف و خازن به AVCC متصل باشد) به پایه VCC متصل می شود .



۳- از قابلیت ADC NOISE CANCELLER به منظور کاهش نویز CPU بر روی مبدل ADC استفاده شود .

۴- اگر هر کدام از بیت های PORTA به عنوان خروجی استفاده شده است ، هنگام عمل تبدیل نباید روشن و خاموش شده یا به عبارت دیگر با فرکانس بالا سوئیچ شوند .

۵- بخش آنالوگ چیپ و تمام قسمت های آنالوگ باید دارای زمین جداگانه باشند این زمین ها با زمین دیجیتال به وسیله یک مسیر به هم متصل می شوند.

حفاظت اسیلاتور مدارات میکروکنترلری در مقابل نویز

حساس ترین پایه نسبت به الکتریسیته ساکن و نویز ، پایه های مرتبط با اسیلاتور است راه حل رفع این مشکل بسیار ساده بوده و حتما بایستی رعایت شود . برای رفع این مشکل بایستی کریستال یا رزوناتور تا حد امکان نزدیک پایه ها و چسپیده آن ها قرار داده شود و خازن های مربوطه به طور مستقیم به زمین متصل شوند ، همچنین هنگام استفاده از منبع پالس خارجی ، این منبع نباید در فاصله زیادی از میکرو کنترلر قرار گیرد ، زیرا مسیر پالس منجر به تولید و دریافت نویز شده ، بر روی عملکرد مدار تاثیر نامطلوب می گذارد در این حالت می توان از بافر و در ورودی آن از فیلتر استفاده نمود .

در محیط های صنعتی ، ممکن است اسیلاتور تحت تاثیر نویز قرار گرفته ودر صورت بالا بودن توان نویز اسیلاتور از نوسان افتاده و متوقف شود . برا ی جلوگیری از اثر نویز بر روی اسیلاتور علاوه بر آن که کریستال باید نزدیک پایه های میکروکنترلر قرار داده شود باید اطراف آن با یک لایه زمین SHIELD شود .

مدیریت توان (صرفه جویی در توان مصرفی)

به منظور مینیمم کردن توان تلفاتی میکروکنترلرهای AVR نکات خاصی را باید در نظر گرفت . در حالت کلی برای کاهش توان تلفاتی باید از یکی از مد های SLEEP استفاده نمود. هنگام انتخاب مدSLEEP نیز باید مدی را انتخاب نمود که در آن حداقل ماژول های جانبی فعال باشد. به عبارت دیگر ماژول های جانبی که مورد استفاده قرار نمی گیرد باید غیر فعال باشند. در این مورد نکات خاصی را در مورد بعضی از ما ژول ها باید در نظر گرفت که در ادامه بررسی می شود.

با توجه به این که ADC در بعضی از مد های SLEEP فعال است ، در صورت عدم نیاز ، به منظور کاهش تلفات توان بهتر است قبل از ورود به این مد ها آن را غیر فعال نمود.

قبل از وارد شدن به مد IDLE و ADC NOISE REDUCTION ، در صورت عدم استفاده از مقایسه کننده آنالوگ ، بایدآن را غیر فعال نمود. در دیگر مد های SLEEP مقایسه کننده آنالوگ به طور خود کار غیر

فعال است. با این وجود در صورتی که ولتاژ مرجع داخلی ADC فعال شده باشد ، مقایسه کننده آنالوگ باید در تمامی مدها غیر فعال شود. زیرا ولتاژ مرجع داخلی به مدهای SLEEP وابسته است. در صورت عدم نیاز به قابلیت BROWN OUT DETECTOR , باید قبل از ورود به هر مد SLEEP آن را غیر فعال نمود. در صورتی که BROWN OUT DETECTOR با استفاده از فیوز بیتBODEN فعال شود در تمام مد ها فعال بوده و منجر به افزایش توان تلفاتی می شود.

تعریف BROWN OUT DETECTOR :

میکروکنترلرMEGA32 دارای یک مدار آشکار ساز BROWN–OUT است این مدار در هر لحظه ولتاژ VCC را برسی می کند و همواره آن را با یک سطح ولتاژ مرجع مقایسه می نماید سطح ولتاژ مرجع توسط فیوز بیت های BODLEVEL انتخاب می شود. (توضیحات مربوط به این فیوز بیت ها در قسمت فیوز بیت های ATMEGA32 ارائه شده است .)

در صورتی که VCC از ولتاژ مرجع کمتر شود خروجی این مدار عمل RESET را انجام می دهد.

ولتاژ مرجع داخلی در صورت فعال بودن مبدل ADC , BROWN OUT DETECTOR و مقایسه کننده آنالوگ فعال می شود. در صورتی که این ماژول ها مطابق آن چه که قبلا گفته شد ، غیر فعال شوند ، مرجع ولتاژ داخلی نیز غیر فعال شده و منجر به کاهش توان تلفاتی می شود.

با توجه به این که شمارنده WATCHDOG در تمامی مد ها فعال است، در صورت عدم نیاز به آن ، برای کاهش توان تلفاتی ، می توان آن را غیر فعال نمود.

تمامی پایه های PORT ها ، در هنگام وارد شدن به مد های SLEEP باید طوری پیکره بندی شوند که توان تلفاتی توسط آن ها حداقل باشد. مهمترین موضوع در این مورد ، اطمینان از عدم بار گذاری پایه ها توسط یک بار مقاومتی است. در مد های SLEEP که پالس های سا عت مربوط به ماژول های I/O وADC غیر فعال هستند ، بافرهای ورودی میکروکنترلر غیر فعال بوده و تلف نشدن توان را تضمین می کند. در بعضی از شرایط پایه های ورودی برای تشخیص شرایط WAKE UP فعال هستند. در این حالت که بافر ورودی فعال است در صورتیکه پایه ها به صورت شناور رها شوند یا ولتاژی نزدیک VCC/2 داشته باشند می توانند توان زیادی را تلف نمایند.

در صورتیکه رابط JTAG و سیستم عیب یاب درونی توسط فیوزبیت ON CHIP DEBUG ENABLE فعال شده باشد و میکروکنترلر وارد یکی از مد های SLEEP , POWER–DOWN و POWER– SAVE شود ، پالس سیستم هنوز فعال بوده و منجر به تلفات توان می شود. برای جلوگیری از اتلاف توان در این شرایط یا باید یکی از فیوز های OCDEN و JTAGEN غیر فعال شود یا بیت JTD در رجیستر MCUCSR یک شود.

گاهی اوقات مسیر زندگی انسان را به سمت نابودی می کشاند. در این هنگام است که باید بی درنگ از جا برخیزید. بنابراین هیچ گاه خود را مانند یک قایق بدون موتور در رودخانه زندگی رها نکنید موتوری برای قایق خود بسازید که حتی در مواقع لزوم بتوانید در خلاف جهت رودخانه نیز حرکت کنید. چرا که برای شنا کردن به سمت مخالف رودخانه قدرت و جرات لازم است و گرنه هر ماهی مرده ای هم می تواند در جهت موافق جریان آب حرکت کند.

فصل چهارم

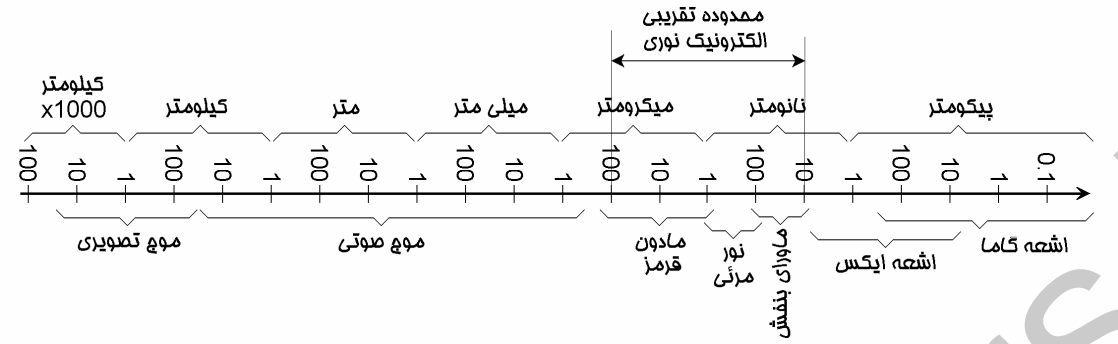
AVR در الکترونیک نوری (OPTOELECTRONICS)

الکترونیک نوری (OPTOELECTRONICS)

نور یکی از پدیده های مهم طبیعت است که نه تنها در زندگی روزمره و حیات انسان اهمیت دارد ، بلکه در پیشرفت و توسعه تکنولوژی هم نقش بسیار مهمی ایفا می کند. نور و خواص آن از ابتدای پیدایش بشر مهم و مورد توجه بوده است. به طوریکه که انسان های اولیه آن را مظهر قدرت می دانستند ، خورشید پرستی و آتش پرستی نمونه های بارزی از توجه انسان به نور بوده است . در مورد نور و ماهیت آن نظریه های گوناگونی ساخته و پرداخته شده است. انسان های اولیه چنین می پنداشتند که نور از چشم خارج و به اجسام برخورد می کند تا آن ها قابل روئیت شوند. بعد ها این نظریه کنار گذاشته شد و انسان فهمید که منبع نور خارج از چشم او و در محیط اطراف است. نیوتن در قرن هفدهم نور را به صورت ذره تعریف کرده و بر اساس آن برخی از خواص نور را توجیه کرد. ولی دانشمندان دیگری مثل هویگنس با آزمایش های گوناگونی ثابت کردند نور موج الکترومغناطیسی است که از منبع خارج و به صورت کروی در محیط انتشار پیدا می کند .طبق این نظریه نور از دو مولفه عمود بر هم مغناطیسی و الکتریکی تشکیل شده است که با سرعت 3*10^8 متر بر ثانیه در محیط انتشار پیدا می کند. بسیاری از خواص نور بر اساس تئوری موجی بودن نور توجیه و ثابت شدند و نظریه نیوتن به فراموشی سپرده شد تا این که در اواخر قرن نوزدهم و شروع قرن بیستم انیشتن یک بار دیگر تعریف ذره ای بودن نور را بیان داشت و در اثبات آن مطالبی را بیان نمود. به هر حال در مورد این که نور ذره و یا موج الکترو مغناطیسی است هنوز هم بحث و جدل هایی وجود دارد و ما ناچاریم تلفیقی از این دو نظریه و در بعضی موارد متناسب با شرایط یکی از این دو نظریه را به کار ببریم . در نظریه جدیدی که پیرامون نور وجود دارد گفته می شود که نور از ذرات ریزی بنام فوتون تشکیل شده است که یک ویژگی دوگانه دارد یعنی ضمن موجی بودن متمرکز هم می باشد. نور هر چه که باشد اهمیت و جایگاه ویژه ای برای ما در الکترونیک نوری دارد.

و اما الکترونیک نوری یا OPTO ELECTRONICS.

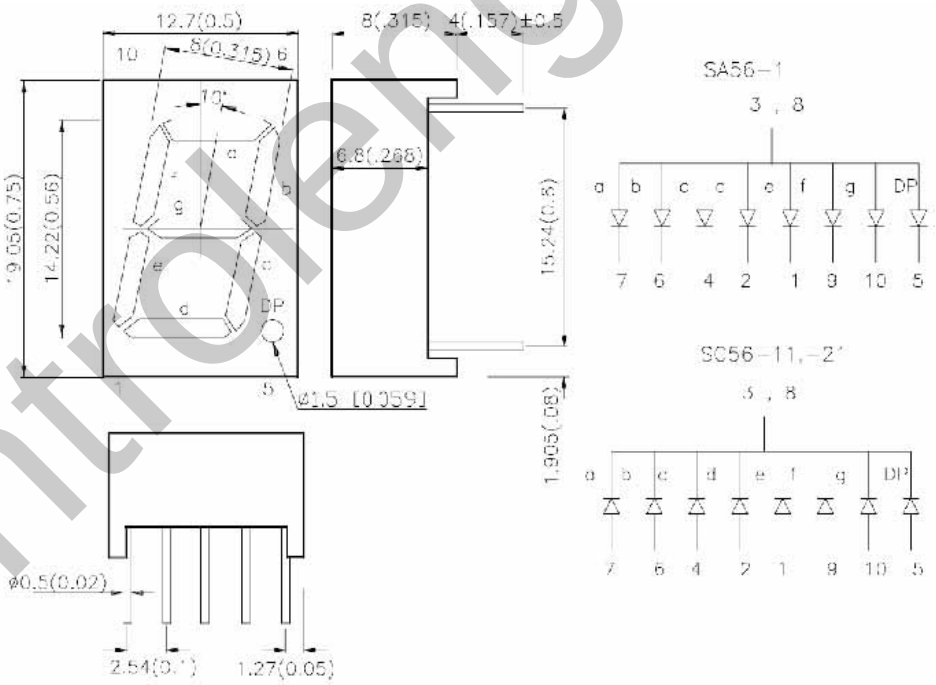
این نام نخستین بار به طور عام در طول دهه 1970 مورد استفاده قرار گرفت که به طور کلی شاخه ای از الکترونیک را در رابطه با کاربردهای عملی ابزار های نوری جدید توصیف می کند. در این مبحث یک ابزار نوری عمدتا می تواند به عنوان وسیله ای برای عمل کردن در محلی با وجود بخشهای نوری مرئی یا نامرئی از طیف الکترومغناطیس مورد بحث قرار گیرد به عبارتی علمی تر این ابزار در توابع بین محدوده طول موج های 10 نانومتر یا 100 میکرومتر از طیف الکترومغناطیس معمولی عمل می کنند. شکل 1-4 جزئیات طیف الکترومغناطیس وهمچنین محدوده تقریبی الکترونیک نوری را نشان می دهد.



شکل 4-1 جزئیات طیف الکترو مغناطیس

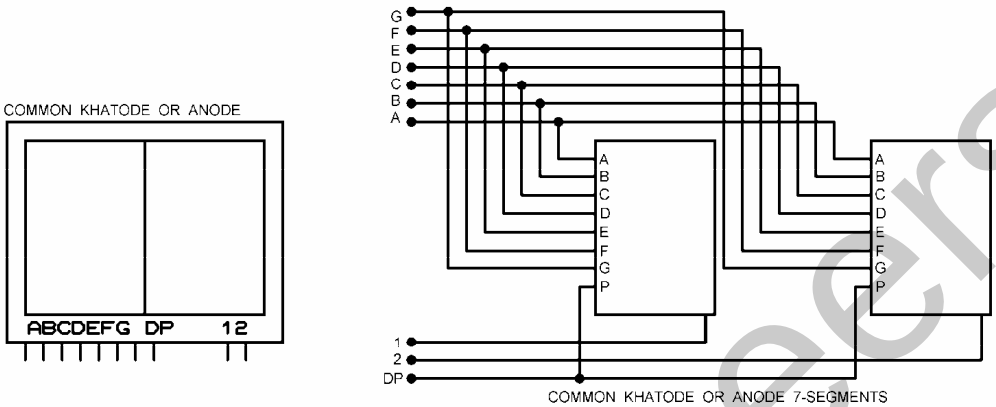
نمایشگر های هفت قسمتی یا 7-SEGMENT

نمایشگر های 7-SEGMENT نوعی ابزار OPTOELECTRONIC هستند که معمولا برای نمایش الفبای عددی مورد استفاده قرار می گیرند. پکیج های مختلفی از این نوع نمایشگر ها ساخته شده است ولی به طور کلی می توان آنها را به دو نوع آنود مشترک (COMMON ANODE) و کاتد مشترک (COMMON KHATODE) تقسیم بندی کرد . به عنوان مثال یک 7-SEGMENT آنود مشترک از 7 عدد LED که آنود های آن ها به هم متصل شده تشکیل شده است ، شکل 4-2 شماتیک داخلی و PACKAGE DIMENSION یک نوع سون سگمنت کاتود مشترک و یک نوع سون سگمنت آنود مشترک را نشان می دهد .



شکل 4-2 شماتیک داخلی و PACKAGE DIMENSION یک نوع 7-SEGMENT کاتود مشترک و یک نوع 7-SEGMENT آنود مشترک

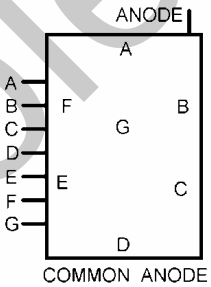
با قرار دادن چندین 7-SEGMENT در کنار هم می توان پکیج های مختلفی از 7-SEGMENT ها را بدست آورد. شکل 3-4 شماتیک داخلی یک 7-SEGMENT دورقمی را نشان می دهد لازم به ذکر است که برای راه اندازی این نوع نمایشگر ها بایستی از روش جاروب یا اسکن استفاده شود .



شکل 3-4 شماتیک داخلی یک نوع 7-SEGMENT دو تائی کاتود یا آنود مشترک

نحوه راه اندازی 7-SEGMENT

برای راه اندازی 7-SEGMENT ابتدا بایستی پایه COMMOND را با توجه به این که 7-SEGMENT مورد نظر آنود مشترک است یا کاتود مشترک VCC یا زمین کنیم ، سپس با توجه به الفبای عددی که قرار است روی نمایشگر نشان داده شود ، اطلاعات مربوطه به پایه های A تا G را به نمایشگر اعمال کنیم نحوه راه اندازی یک نمایشگر (COMMOND ANODE) در شکل 4-4 نشان داده شده است .

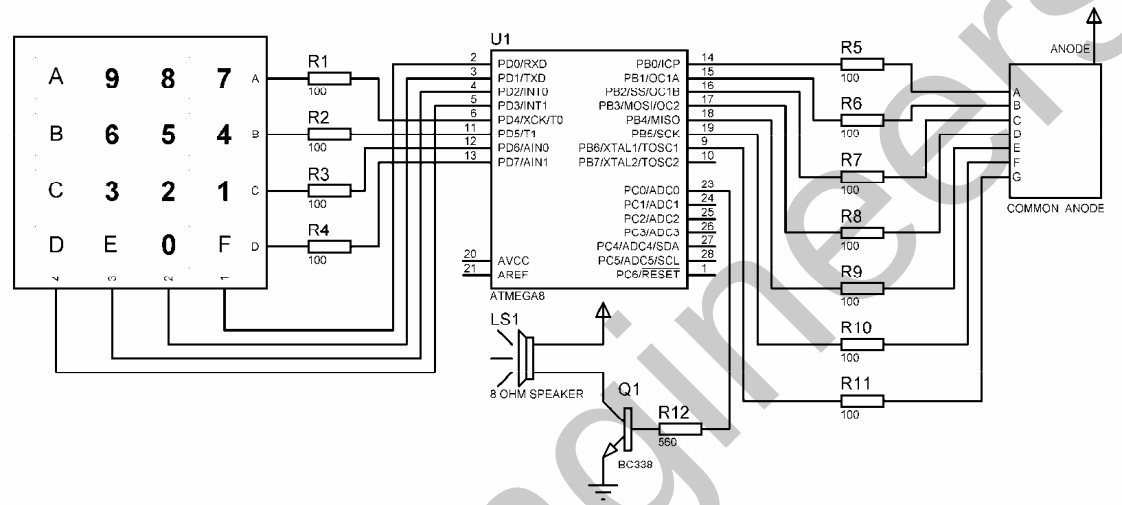


G	F	E	D	C	B	A	
1	0	0	0	0	0	0	0
1	1	1	1	0	0	1	1
0	1	0	0	1	0	0	2
0	1	1	0	0	0	0	3
0	0	1	1	0	0	1	4
0	0	1	0	0	1	0	5
0	0	0	0	0	1	0	6
1	1	1	1	0	0	0	7
0	0	0	0	0	0	0	8
0	0	1	0	0	0	0	9
0	0	0	1	0	0	0	A
0	0	0	0	0	1	1	b
1	0	0	0	1	1	0	C
0	1	0	0	0	0	1	d
0	0	0	0	1	1	0	E
0	0	0	1	1	1	0	F

شکل 4-4 نحوه نمایش الفبای عددی بر روی یک 7-SEGMENT آنود مشترک

طراحی پروژه نمایش نام کلید فشرده شده توسط KEYPAD بر روی 7-SEGMENT

در این پروژه حرف یا کلید فشرده شده توسط KEYPAD بر روی 7-SEGMENT نمایش داده شده و تا فشار کلید بعدی بدون تغییر با قی می ماند . با فشار کلید جدید حرف یا عدد قبلی پاک شده و اطلاعات جدید بر روی 7-SEGMENT نمایش داده می شود. 7-SEGMENT استفاده شده یک نمایشگر COMMON ANODE می باشد سخت افزار پروژه در شکل 5-4 ارائه شده است .



شکل 5-4 شماتیک طراحی شده برای پروژه نمایش نام کلید فشرده شده بر روی 7-SEGMENT

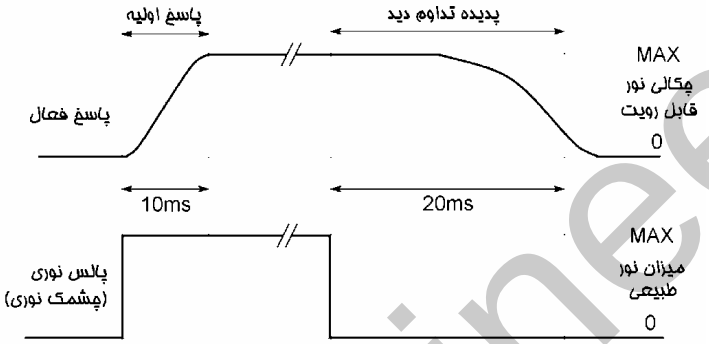
برنامه نوشته شده برای پروژه به صورت زیر می باشد .

```
'COMPILER:1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 1000000
Config Portb = Output
Config Pinc.0 = Output
Config Kbd = Portd , Debounce = 10 , Delay = 100
Dim Keypad_data As Byte , Recive_data As Byte
'-----------------------------------------------------------------------------------------------------------
Portb = 255
'START OF PROGRAM-----------------------------------------------------------------------------------------
Start_program:
Keypad_data = Getkbd()
If Keypad_data = 16 Then Goto Start_program
Sound Portc.0 , 100 , 80
Recive_data = Lookup(keypad_data , Data_code)
Portb = Lookup(recive_data , 7seg_data)
H1:
Keypad_data = Getkbd()
If Keypad_data <> 16 Then Goto H1
Goto Start_program
'END OF PROGRAM-------------------------------------------------------------------------------------------
7seg_data:
Data &B01000000 , &B01111001 , &B00100100 , &B00110000 , &B00011001
Data &B00010010 , &B00000010 , &B01111000 , &B00000000 , &B00010000
Data &B00001000 , &B00000011 , &B01000110 , &B00100001 , &B00000110
Data &B00001110
```

```
'-----------------------------------------------------------------------------------------------------------
Data_code:
Data 7 , 8 , 9 , 10 , 4 , 5 , 6 , 11 , 1 , 2 , 3 , 12 , 15 , 0 , 14 , 13
'-----------------------------------------------------------------------------------------------------------
```

راه اندازی 7-SEGMENT های چند رقمی

برای راه اندازی 7-SEGMENT های چند رقمی بایستی از روش اسکن یا جاروب استفاده شود ، برای درک نحوه عملکرد جاروب یا اسکن ابتدا بایستی برخی واقعیات اساسی را که مربوط به حس بینائی شما هستند فرا بگیرید . شکل 6-4 پاسخ معمول ترکیب مغز انسان با چشم اورا به چشمک نوری نشان می دهد .



شکل 6-4 پاسخ معمول مغز انسان به چشمک نوری

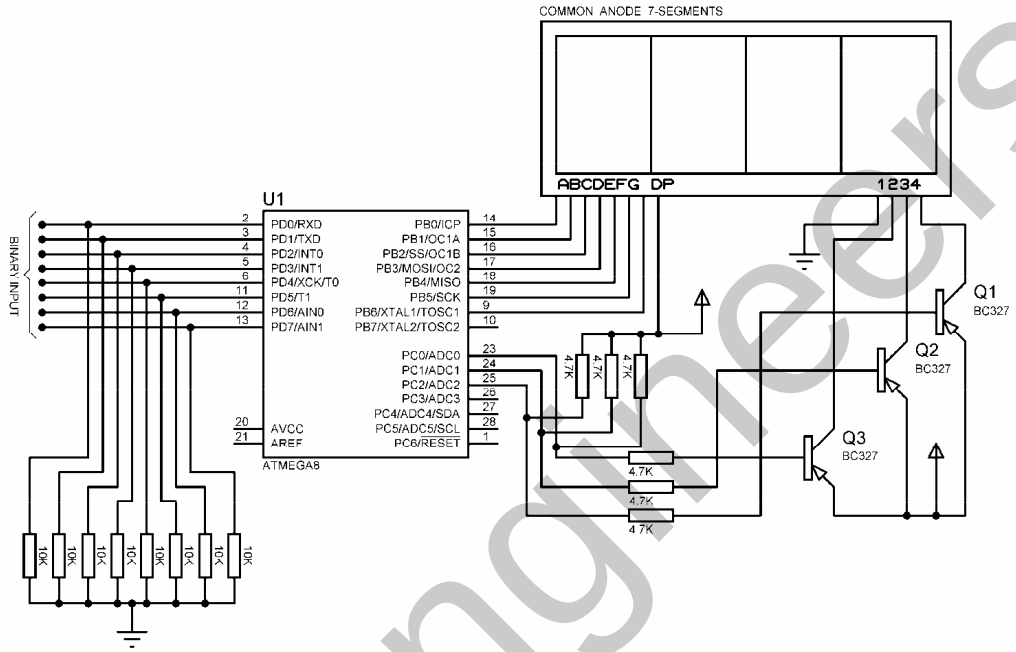
نکاتی در رابطه با شکل فوق

با توجه به شکل فوق پالس نوری حداقل بایستی10ms تداوم داشته باشد تا به وضوح و به صورت کاملا روشن دیده شود همچنین پس از این که روشنائی به صورت کامل توسط مغز انسان تشخیص داده شد مدت زمان 20ms طول می کشد تا قطع شدن پالس نوری توسط مغز تشخیص داده شود با توجه به مطلب فوق اگر چشمک های نوری حداقل 20ms از هم فاصله داشته باشند و مدت زمان تداوم آن ها بیشتر از 10ms باشد مغز انسان به وضوح قادر به تشخیص چشمک های نوری خواهد بود ، ولی اگر مدت زمان تداوم آن ها کمتر از 10ms باشد از روشنایی و وضوح آن ها کاسته می شود حال اگر چشمک های نوری 10ms از هم فاصله داشته باشند و مدت زمان تداوم آن ها نیز 10ms باشد مغز انسان قادر به تشخیص چشمک ها نبوده و منبع نوری به صورت دائم روشن دیده می شود. در حالت کلی اگر فرکانس چشمک نوری بیشتر از 100Hz باشد چشم انسان قادر به تشخیص چشمک ها نخواهد بود.

طراحی یک مبدل باینری به 7-SEGMENT (8 بیتی)

در این پروژه ابتدا 7-SEGMENT شماره 4 توسط پایه مشترک آن یعنی پایه شماره 4 انتخاب شده و اطلاعات مربوطه به آن اعمال می شود، پس از 100US تاخیر7-SEGMENT شماره 4 خاموش شده و7-SEGMENT شماره 3 روشن می شود پس از اعمال اطلاعات مربوط به 7-SEGMENT شماره 3 و 100US تاخیر عملیات

مزبور روی 7-SEGMENT شماره 2 انجام شده و این روند ادامه خواهد یافت. البته رقم نمایش داده شده بر روی 7-SEGMENT ها متناسب با مقدار باینری ورودی است . با توجه به این که مقدار عدد باینری ورودی 8 بیتی است ، بیشترین رقم نشان داده شده بر روی 7-SEGMENT ها 255 خواهد بود. سخت افزار پروژه در شکل7-4 ارائه شده است .



شکل7-4 شماتیک طراحی شده برای مبدل باینری به 7-SEGMENT ، 8 بیتی

برنامه نوشته شده برای سخت افزار فوق به صورت زیر می باشد.

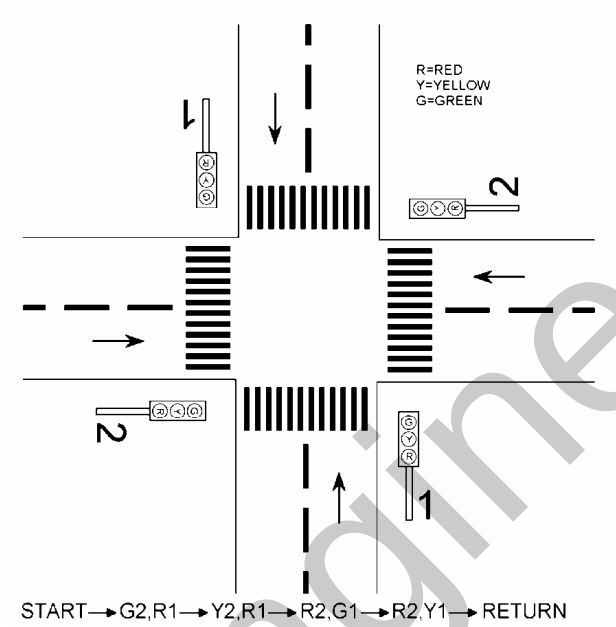
```
'COMPILER:1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Portb = Output
Config Portd = Input
Declare Sub Binary_input
Config Pinc.0 = Output : 7seg2 Alias Portc.0
Config Pinc.1 = Output : 7seg3 Alias Portc.1
Config Pinc.2 = Output : 7seg4 Alias Portc.2
Dim Recive_data As Byte , S1 As String * 4 , H As Byte , S2 As String * 1
Dim 7seg2_data As Byte , 7seg3_data As Byte , 7seg4_data As Byte
'START OF PROGRAM-----------------------------------------------------------------------------------------
Start_program:
PORTB=255
Reset 7seg2 : Set 7seg3 : Set 7seg4
Portb = 7seg2_data : Waitus 100
Call Binary_input
PORTB=255
Set 7seg2 : Reset 7seg3 : Set 7seg4
Portb = 7seg3_data : Waitus 100
Call Binary_input
PORTB=255
Set 7seg2 : Set 7seg3 : Reset 7seg4
```

```
Portb = 7seg4_data : Waitus 100
Call Binary_input
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------------
7seg_code:
Data &B01000000 , &B01111001 , &B00100100 , &B00110000 , &B00011001
Data &B00010010 , &B00000010 , &B01111000 , &B00000000 , &B00010000
'-----------------------------------------------------------------------------------------------------------
Sub Binary_input:
Recive_data = Pind
S1 = Str(recive_data)
H = Len(s1)
Select Case H:
'-----------------------------------------------------------------
Case Is = 1
7seg4_data = Val(s1)
7seg4_data = Lookup(7seg4_data , 7seg_code)
7seg3_data = Lookup(0 , 7seg_code)
7seg2_data = Lookup(0 , 7seg_code)
'-----------------------------------------------------------------
Case Is = 2
S2 = Mid(s1 , 2 , 1)
7seg4_data = Val(s2)
7seg4_data = Lookup(7seg4_data , 7seg_code)

S2 = Mid(s1 , 1 , 1)
7seg3_data = Val(s2)
7seg3_data = Lookup(7seg3_data , 7seg_code)
7seg2_data = Lookup(0 , 7seg_code)
'-----------------------------------------------------------------
Case Is = 3
S2 = Mid(s1 , 3 , 1)
7seg4_data = Val(s2)
7seg4_data = Lookup(7seg4_data , 7seg_code)

S2 = Mid(s1 , 2 , 1)
7seg3_data = Val(s2)
7seg3_data = Lookup(7seg3_data , 7seg_code)

S2 = Mid(s1 , 1 , 1)
7seg2_data = Val(s2)
7seg2_data = Lookup(7seg2_data , 7seg_code)
End Select
Return
End Sub
'-----------------------------------------------------------------------------------------------------------
```

توجه داشته باشید در برنامه فوق به منظور جلوگیری از ایجاد پس ماند تصویر قبل از روشن کردن هر کدام 7-SEGMENT ها عدد 255 به پورت B اعمال می شود ، در صورت عدم انجام این عمل زمانی که اطلاعات مربوط به یکی از 7-SEGMENT ها را روی پورت B قرار داده و پس از تاخیر بدون این که عدد 255 یا &B11111111 را روی پورت B قرار دهیم 7-SEGMENT دیگر را روشن کنیم اطلاعات مربوط به 7-SEGMENT قبل بر روی 7-SEGMENT ای که تازه روشن کرده ایم به مدت بسیار کوتاهی نمایش داده می شود که این عمل باعث ایجاد پس ماند تصویر و کاهش کیفیت آن خواهد شد. روشن کردن هر کدام از 7-SEGMENT ها نیز با استفاده از اعمال صفر به بیس ترانزیستور PNP مربوطه صورت می گیرد .

طراحی چراغ راهنمایی توسط میکروکنترلر AVR

شکل 8-4 نحوه عملکرد چراغ راهنمایی را در یک چهارراه نشان می دهد توجه داشته باشید که عملکرد چراغ های هم نام دقیقا مشابه هم باشد .



شکل 8-4 نحوه عملکرد یک چراغ راهنمایی در یک چهارراه

دراین پروژه از مبدل باینری به7-SEGMENT که قبلا طراحی کردیم به عنوان نمایشگر زمان استفاده شده است . با توجه به این که زمان نشان داده شده در نمایشگر هر چهار چراغ راهنمایی یکی است در شماتیک پروژه فقط یک نمایشگر استفاده شده است لازم به ذکر است که شما می توانید با توجه به مطالبی که در فصل 3 در رابطه با راه اندازی LED گفته شده است 7-SEGMENT هایی با اندازه های بزرگتر طراحی کنید . رله های مدار از نوع رله های 12 ولتی است که می تواند 220 ولت یا بیشتر از آن را سوئیچ کند .

این پروژه دارای دو حالت برنامه ریزی و اجرا می باشد.کاربر بایستی ابتدا با قرار دادن کلید دوحالته PROGRAM SW در حالت ، HIGH یا 5 ولت و ریست کردن میکرو وارد قسمت برنامه ریزی دستگاه شده و با استفاده از کلید های UP TIME ، DOWN TIME ، OK BUTTON و همچنین LCD کاراکتری تایم تاخیر مورد نظر را برای سبز یا قرمز شدن چراغ راهنمایی وارد کند. حداکثر مقدار تایم تاخیر یا DELAY TIME ، 255 ثانیه می باشد .

پس از وارد کردن DELAY TIME و زدن کلید OK BUTTON تایم تاخیر مورد نظر در EEPROM داخلی میکروکنترلر نوشته می شود با این کار مقدار تایم تاخیر پس از قطع تغذیه میکروکنترلر نیز پاک نخواهد شد .

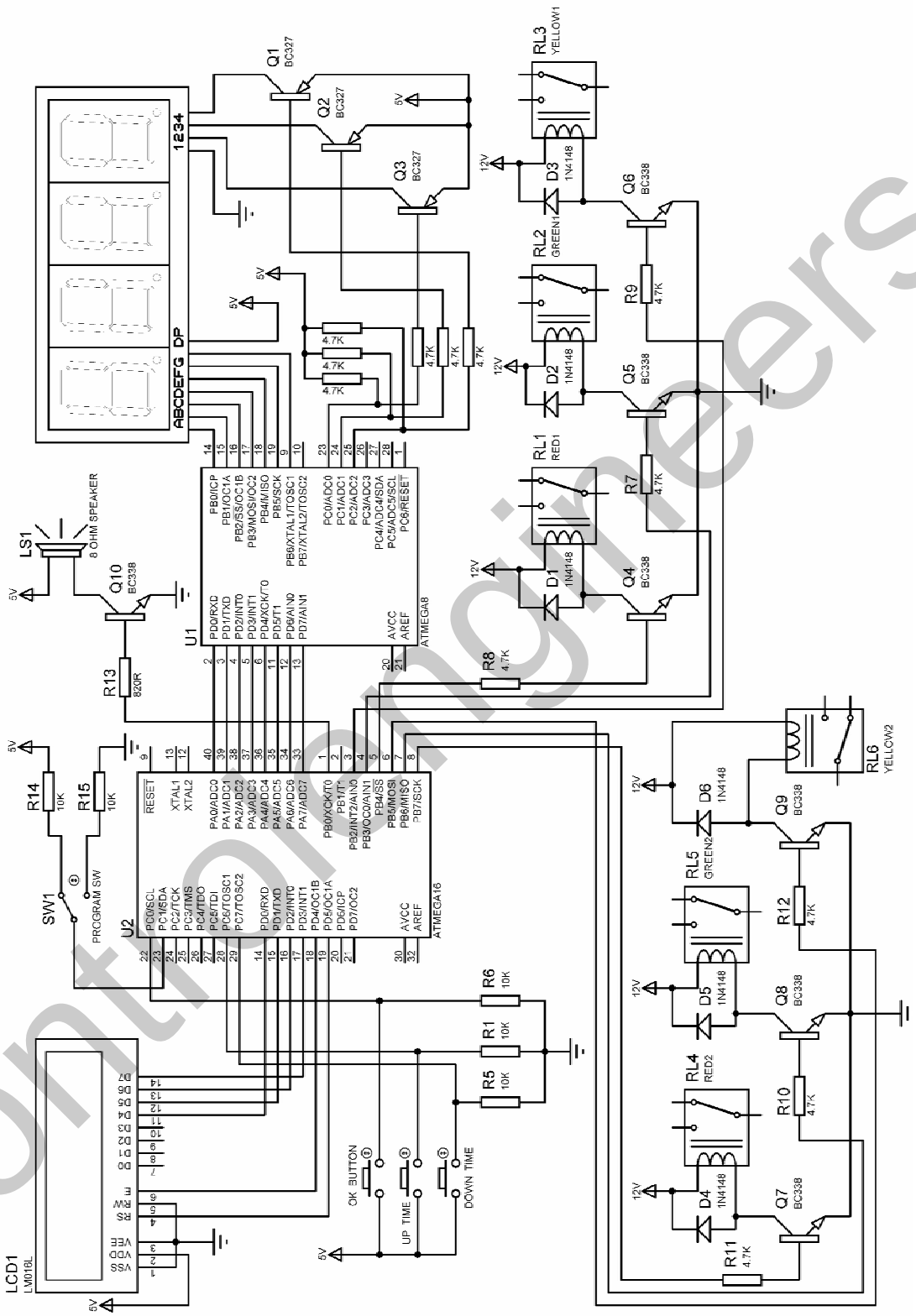
سپس برای این که میکروکنترلر پس از RESET شدن دوباره وارد قسمت برنامه ریزی نشود کلید PROGRAM SW را در حالت LOW یا زمین قرار دهید . در هر کدام از چراغ های راهنمایی که چراغ ، سبز می باشد پس از این که تایم تاخیر به صورت نزولی از مقدار تعیین شده تا عدد 5 شمرده شد چراغ سبز خاموش شده و چراغ زرد به مدت5 ثانیه باقیمانده ازDELAY TIME روشن خواهد بود.سخت افزار پروژه در شکل9-4 ارائه شده است.

برنامه نوشته شده برای پروژه به صورت زیر می باشد .

```
'COMPILER:1.11.7.4
$regfile = "M16DEF.DAT"
$crystal = 1000000
Dim Delay_time As Byte , T As Byte , Save As Eram Byte
Dim Program As Eram Byte
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
            , E = Pind.4 , Rs = Pind.5
Config Porta = Output : Config Portb = Output
Config Pinc.6 = Input : Config Pinc.7 = Input : Config Pinc.0 = Input
Red1 Alias Portb.4 : Red2 Alias Portb.7
Yellow1 Alias Portb.2 : Yellow2 Alias Portb.5
Green1 Alias Portb.3 : Green2 Alias Portb.6
'-----------------------------------------------------------------------------------------------------------
Reset Red1 : Reset Green1 : Reset Yellow1
Reset Red2 : Reset Green2 : Reset Yellow2
Porta = 0
'-----------------------------------------------------------------
If Program = 0 Then Save = 20
If Pinc.1 = 0 Then Goto Start_program
'-----------------------------------------------------------------
Cursor Off : Cls : Home
Lcd "ENTER DELAY TIME"
Lowerline
Lcd "TIME IS 20 S"
'-----------------------------------------------------------------------------------------------------------
Delay_time = 20
H1:
If Pinc.6 = 1 Then
If Delay_time < 255 Then Incr Delay_time
Cls : Home
Lcd "ENTER DELAY TIME"
Lowerline
Lcd "TIME IS " ; Delay_time ; " S"
Sound PortB.0 , 100 , 80
End If
'-----------------------------------------------------------------
If Pinc.7 = 1 Then
If Delay_time > 6 Then Decr Delay_time
Cls : Home
Lcd "ENTER DELAY TIME"
Lowerline
Lcd "TIME IS " ; Delay_time ; " S"
Sound Portb.0 , 100 , 80
End If
'-----------------------------------------------------------------
```

```
If Pinc.0 = 1 Then
Cls : Home
Lcd "OK"
Lowerline
Lcd "TIME IS " ; Delay_time ; " S"
Sound Portb.0 , 100 , 80
Save = Delay_time
Program = 1
Goto Start_program
End If : Waitms 10
Goto H1
'-----------------------------------------------------------------------------------------------------------------
Start_program:
Delay_time = Save
Cursor Off
'R1=SET,G2=SET-----------------------------------------
Set Red1 : Reset Green1 : Reset Yellow1
Reset Red2 : Set Green2 : Reset Yellow2
For T = Delay_time To 6 Step -1
Porta = T
Cls : Home
Lcd "DELAY TIME IS"
Lowerline : Lcd T
Wait 1 : Next T
'R1=SET,Y2=SET-----------------------------------------
Set Red1 : Reset Green1 : Reset Yellow1
Reset Red2 : Reset Green2 : Set Yellow2
For T = 5 To 0 Step -1
Porta = T
Cls : Home
Lcd "DELAY TIME IS"
Lowerline : Lcd T
Wait 1 : Next T
'G1=SET,R2=SET-----------------------------------------
Reset Red1 : Set Green1 : Reset Yellow1
Set Red2 : Reset Green2 : Reset Yellow2
For T = Delay_time To 6 Step -1
Porta = T
Cls : Home
Lcd "DELAY TIME IS"
Lowerline : Lcd T
Wait 1 : Next T
'Y1=SET,R2=SET-----------------------------------------
Reset Red1 : Reset Green1 : Set Yellow1
Set Red2 : Reset Green2 : Reset Yellow2
For T = 5 To 0 Step -1
Porta = T
Cls : Home
Lcd "DELAY TIME IS"
Lowerline : Lcd T
Wait 1 : Next T
Goto Start_program
'-----------------------------------------------------------------------------------------------------------------
```

در برنامه فوق متغیر های SAVE وPROGRAM از نوع ERAM BYTE تعریف شده اند ، یعنی مقدار این متغیر ها در حافظه EEPROM ذخیره خواهد شد. زمانی که این متغیرها مقدار دهی نشده باشند مقدار اولیه آن ها صفر خواهد بود.



شکل 4-9 سخت افزار طراحی شده برای پروژه چراغ راهنمایی

با توجه به این موضوع مقدار اولیه متغیرPROGRAM صفر بوده و اگر وارد قسمت برنامه ریزی نشده و مقدار تاخیر را وارد نکنیم مقدار اولیه تاخیر توسط دستور

IF PROGRAM=0 THEN SAVE=20

روی 20 ثانیه تنظیم می شود با وارد شدن به حالت برنامه ریزی و واردکردن DELAY TIME علاوه بر این که تایم تاخیر وارد شده در متغیر SAVE قرار می گیرد. مقدار متغیر PROGRAM نیز 1 می شود ودستور فوق نیز اجرا نخواهد شد.

طراحی ساعت الکترونیکی با استفاده از 7-SEGMENT و کریستال ساعت

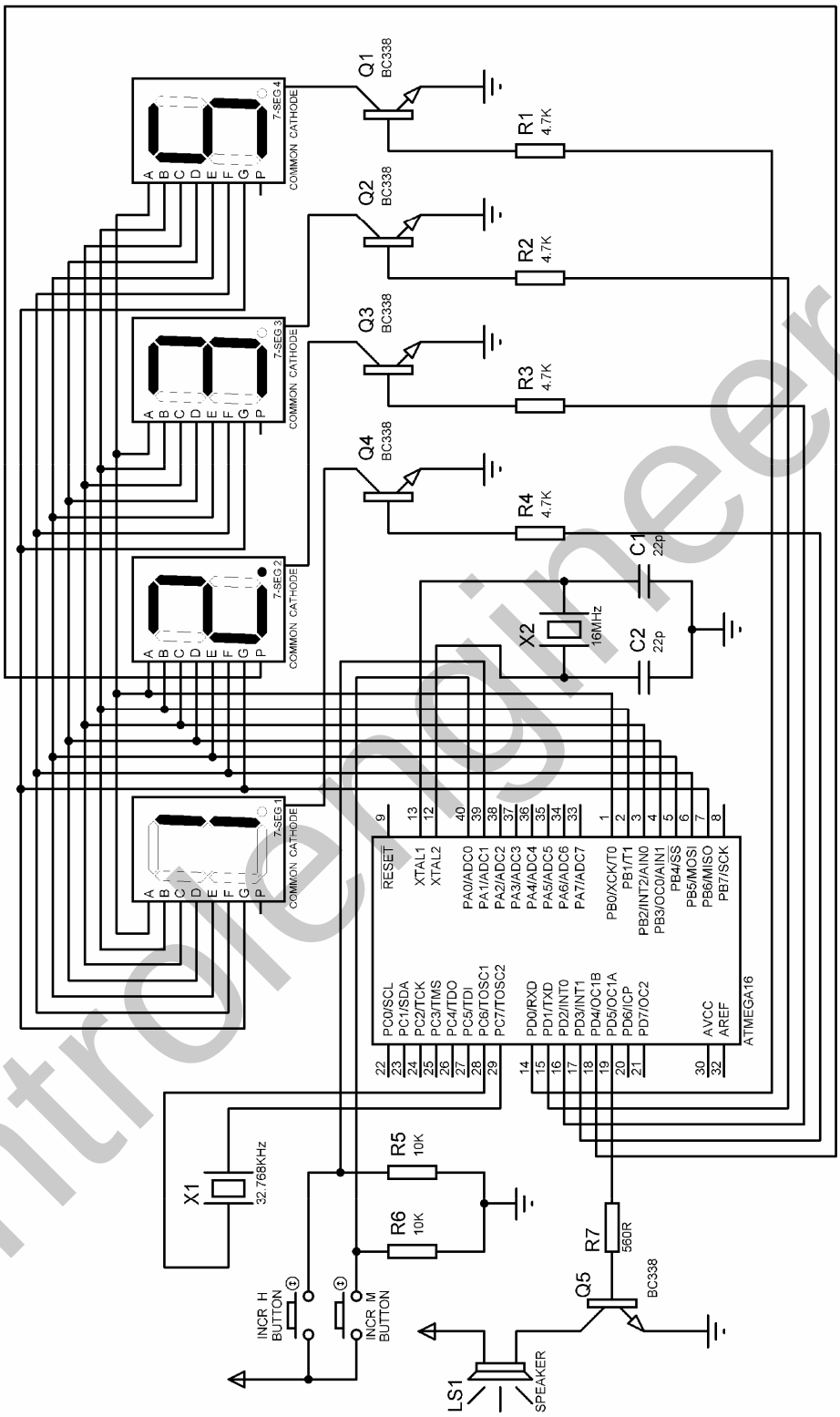
در این پروژه برای درست کردن تایم واقعی (REAL TIME) از تایمر 2 در حالت آسنکرون استفاده شده است، در این حالت تایمر2 کلاک خود را به صورت آسنکرون از کلاک میکرو و سنکرون با اسیلاتور ساعت 32.768KHz دریافت می کند. به صورت کلی کلاک تایمر 2 از کلاک میکرو قطع شده و از اسیلاتور خارجی تامین می شود. توجه داشته باشید که فقط تایمر های 8 بیتی 2 و0 می توانند به صورت آسنکرون پیکره بندی شوند. زمان های مختلفی که می توان با کریستال 32.768KHz توسط تایمر های صفر یا دو ساخت در جدول زیر ارائه شده است .

PRESCALE	OVERFLOW PERIOD
STOP,TIMER/COUNTER IS STOPED	-
1	1/64 S
8	1/8 S
32	1/4 S
64	1/2 S
128	1 S
256	2 S
1024	8 S

جدول انتخاب کلاک تایمر/کانتر **0** یا **2** به ازاء **CK2,0=32,768Hz** در مد آسنکرون

در این پروژه از 7-SEGMENT کاتود مشترک به عنوان نمایشگر استفاده شده است. شما می توانید با توجه به مطالبی که در فصل 3 در رابطه با راه اندازی LED گفته شده است 7-SEGMENT هایی بااندازه های بزرگتر طراحی کنید. برای تنظیم زمان می توانید از کلید INCR M BUTTON برای تنظیم دقیقه و از کلید INCR H BUTTON برای تنظیم ساعت استفاده کنید. برای تامین کلاک میکرو از کریستال خارجی16MHz استفاده شده است .

شماتیک پروژه در شکل 10-4 ارائه شده است .



شکل 4-10 شماتیک طراحی شده برای پروژه ساعت

برنامه نوشته شده برای پروژه به صورت زیر است.

```
'COMPILER:1.11.8.7
$regfile = "M16DEF.DAT"
$crystal = 16000000
Config Portb = Output
Config Pina.0 = Input
Config Pina.1 = Input
Config Pind.5 = Output
Config Pina.3 = Output
Config Timer2 = Timer , Async = On , Prescale = 128
Enable Timer2
Enable Ovf2
Enable Interrupts
On Ovf2 Overflow
Declare Sub Min_scane
Declare Sub H_scane
Declare Sub Incr_m
Declare Sub Incr_h
Config Pind.3 = Output : 7seg1 Alias Portd.3
Config Pind.2 = Output : 7seg2 Alias Portd.2
Config Pind.1 = Output : 7seg3 Alias Portd.1
Config Pind.0 = Output : 7seg4 Alias Portd.0
Config Pind.4 = Output : Point Alias Portd.4
Dim Scane_data As Word , S1 As String * 4 , T As Byte , S2 As String * 1
Dim 7seg2_data As Byte , 7seg3_data As Byte , 7seg4_data As Byte
Dim 7seg1_data As Byte , M As Byte , S As Byte , H As Byte , A As Byte
'START OF PROGRAM-----------------------------------------------------------------------------------------
S = 0 : M = 0 : H = 1
'-----------------------------------------------------------------------------
Start_program:
Portb = 0
If Pina.0 = 1 Then Call Incr_m
If Pina.1 = 1 Then Call Incr_h
Reset 7seg1 : Reset 7seg2 : Reset 7seg3 : Set 7seg4
Call Min_scane
7seg4_data = Not 7seg4_data
Portb = 7seg4_data : Waitus 100
'-----------------------------------------------------------------------------
Portb = 0
If Pina.0 = 1 Then Call Incr_m
If Pina.1 = 1 Then Call Incr_h
Reset 7seg1 : Reset 7seg2 : Set 7seg3 : Reset 7seg4
Call Min_scane
7seg3_data = Not 7seg3_data
Portb = 7seg3_data : Waitus 100
'-----------------------------------------------------------------------------
Portb = 0
If Pina.0 = 1 Then Call Incr_m
If Pina.1 = 1 Then Call Incr_h
Reset 7seg1 : Set 7seg2 : Reset 7seg3 : Reset 7seg4
Call H_scane
7seg2_data = Not 7seg2_data
Portb = 7seg2_data : Waitus 100
'-----------------------------------------------------------------------------
Portb = 0
If Pina.0 = 1 Then Call Incr_m
If Pina.1 = 1 Then Call Incr_h
Set 7seg1 : Reset 7seg2 : Reset 7seg3 : Reset 7seg4
Call H_scane
7seg1_data = Not 7seg1_data
Portb = 7seg1_data : Waitus 100
```

```
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------------
7seg_code:
Data &B01000000 , &B01111001 , &B00100100 , &B00110000 , &B00011001
Data &B00010010 , &B00000010 , &B01111000 , &B00000000 , &B00010000
'-------------------------------------------------------------------------------------------------------------
Sub Min_scane:
Scane_data = M
S1 = Str(scane_data)
T = Len(s1)
Select Case T:
'-----------------------------------------------------------------
Case Is = 1
7seg4_data = Val(s1)
7seg4_data = Lookup(7seg4_data , 7seg_code)
7seg3_data = Lookup(0 , 7seg_code)
'-----------------------------------------------------------------
Case Is = 2
S2 = Mid(s1 , 2 , 1)
7seg4_data = Val(s2)
7seg4_data = Lookup(7seg4_data , 7seg_code)

S2 = Mid(s1 , 1 , 1)
7seg3_data = Val(s2)
7seg3_data = Lookup(7seg3_data , 7seg_code)
'-----------------------------------------------------------------
End Select
Return
End Sub
'-------------------------------------------------------------------------------------------------------------
Sub H_scane:
Scane_data = H
S1 = Str(scane_data)
T = Len(s1)
Select Case T:
'-----------------------------------------------------------------
Case Is = 1
7seg2_data = Val(s1)
7seg2_data = Lookup(7seg2_data , 7seg_code)
7seg1_data = Lookup(0 , 7seg_code)
'-----------------------------------------------------------------
Case Is = 2
S2 = Mid(s1 , 2 , 1)
7seg2_data = Val(s2)
7seg2_data = Lookup(7seg2_data , 7seg_code)

S2 = Mid(s1 , 1 , 1)
7seg1_data = Val(s2)
7seg1_data = Lookup(7seg1_data , 7seg_code)
'-----------------------------------------------------------------
End Select
Return
End Sub
'-------------------------------------------------------------------------------------------------------------
Overflow:
Disable Interrupts
Stop Timer2
Toggle Point
Incr S
If S > 59 Then
S = 0 : Incr M
End If
```

```
'-----------------------------------------------------------------
If M > 59 Then
Incr H : M = 0
End If
'-----------------------------------------------------------------
If H > 12 Then
S = 0 : M = 0 : H = 1
End If
'-----------------------------------------------------------------
Enable Interrupts
Start Timer2
Return
'-----------------------------------------------------------------
Sub Incr_m
Incr M
If M > 59 Then M = 0
Sound Portd.5 , 100 , 150
Bitwait Pina.0 , Reset
Return
End Sub Incr_m
'-----------------------------------------------------------------
Sub Incr_h
Incr H
If H > 12 Then H = 1
Sound Portd.5 , 100 , 150
Bitwait Pina.1 , Reset
Return
End Sub Incr_m
'-----------------------------------------------------------------
```

دربرنامه فوق به دلیل این که جدول LOOKUP ، 7SEG_CODE مربوط به نمایشگرهای آنود مشترک می باشد و 7-SEGMENT های استفاده شده در این پروژه از نوع کاتود مشترک است. کد های موجود در جدول لوک آپ 7SEG_CODE ابتدا معکوس شده سپس به نمایشگرها اعمال می شود .

طراحی تابلو روان با استفاده از میکروکنترلر AVR

تابلو روان یکی از پروژه های پر طرفدار OPTOELECTRONIC می باشد که دارای تکنولوژی نسبتا پیچیده ای است ، متاسفانه در کشور ما آموزش تابلوهای روان با قیمت های بسیار بالا توسط اشخاصی که اکثریت آن ها صلاحیت کامل برای آموزش این نوع پروژه ها را ندارند آموزش داده می شود. به طور کلی تابلو های روان در دونوع خطی و مانیتورینگ طراحی می شوند که نوع مانیتورینگ سازگار با سیگنال ویدئو بوده و مانند یک مانیتور بزرگ عمل می کند. شکل 11-4 نمونه ای از یک تابلو خطی تک رنگ را نشان می دهد. هم چنین تابلو های روان از نظر رنگ به 3 نوع تک رنگ ، RG و RGB تقسیم بندی می شوند . همانطور که می دانید همه رنگ های موجود در طبیعت را می توان از 3 رنگ اصلی RED ، GREEN ، BLUE وتغییر میزان هر کدام از این رنگ ها بدست آورد ، بنابراین در تابلو های تمام رنگی (FULL COLOR) هر پیکسل دارای 4 عدد LED می باشد که یکی از آن ها سبز ، یکی آبی و دو عدد از آن ها به رنگ قرمز می باشد .

شکل 4-11 نمونه ای از تابلوی خطی تک رنگ

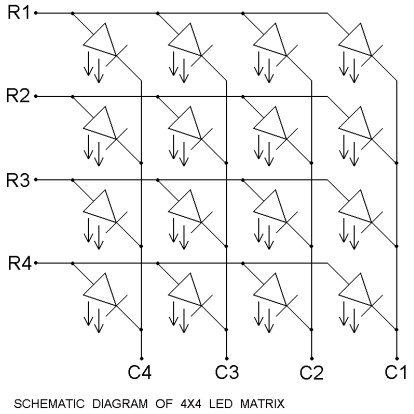
علت استفاده از 2 عدد LED قرمز در هر پیکسل این است که رنگ قرمز بیشترین استفاده را در طبیعت دارد .

در تابلو های RG معمولا هر پیکسل از 3 عدد LED تشکیل می شود که دو تا از آن ها به رنگ قرمز و یکی به رنگ سبز می باشد در این نوع تابلو ها به دلیل استفاده نکردن از رنگ آبی کیفیت تصویر کمتر از تابلو های RGB خواهد بود .

فاصله بین پیکسل ها بستگی به فاصله تابلو تا چشم ناظر دارد ، هر چقدر فاصله چشم ناظر با تابلو کمتر باشد ، فاصله بین پیکسل ها نیز کمتر خواهد بود . تابلو های تک رنگ نسبت به تابلو های RG و RGB دارای تکنولوژی ساده تری هستند و شما می توانید با استفاده از میکروکنترلر آن ها را طراحی کنید. از این رو در این قسمت به آموزش کامل تابلو های تک رنگ می پردازیم .

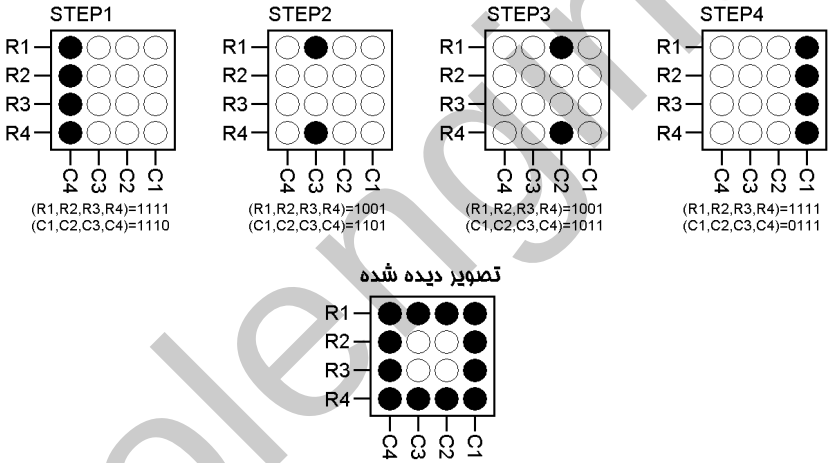
در تابلو های روان از پدیده تداوم دید که قبلا در رابطه با آن بحث شده است برای ایجاد تصویر استفاده می شود ، با توجه به مطالب گفته شده در این رابطه اگر یکLED با فرکانس بیشتر از 100Hz روشن و خاموش شود چشم انسان قادر به تشخیص چشمک زدن LED نبوده و LED از نظر او به صورت دائم روشن دیده می شود برای آن که بتوان از LED به عنوان یک مولد نور استفاده کرد بایستی از یک مقاومت محدود کننده جریان استفاده کنیم ولی اگر برای روشن و خاموش کردن LED از یک ورودی مربعی با فرکانس بالا استفاده شود به علت این که LED مدت زمان بسیار کمی روشن خواهد بود می توان مقاومت محدود کننده R را از مدار حذف نمود و یا مقدار آن راتا حد قابل ملاحظه ای کاهش داد .

برای ساختن صفحه نمایشگر LED می توانید از ساختار ماتریسی شکل 12-4 استفاده کنید .



شکل 4-12 ساختار ماتریسی معمول برای طراحی نمایشگر LED در تابلو روان

در ماتریس شکل 4-12 برای انتخاب کردن سطر ها بایستی به آن ها یک و برای انتخاب کردن ستون ها بایستی به آن ها صفر بدهیم. شکل 4-13 نحوه ایجاد تصویر مربع را با استفاده از ماتریس معرفی شده را نشان می دهد .



شکل 4-13 نحوه ایجاد تصویر مربع با استفاده از ماتریس LED شکل 4-12

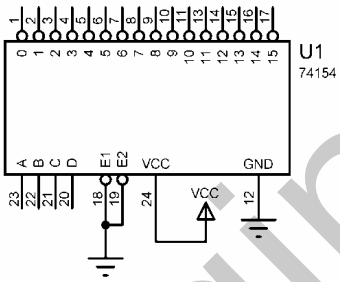
طراحی یک تابلو روان 8*16

برای شروع یک تابلوی ساده وکوچک 8*16 را با استفاده از میکروکنترلر ATMEGA32 طراحی می کنیم . برای انتخاب ستون ها از یک IC انتخاب گر(4 به16) استفاده شده است ، در این حالت با توجه به این که خروجی انتخاب گر74154 به صورت ACTIVE LOW می باشد. هنگام طراحی ماتریس LED کاتود LED ها را به ستون ها متصل کرده و آنود آن ها را به سطر ها متصل می کنیم .

اگر از IC انتخاب کننده 4 مسیری 74154 برای انتخاب کردن سطر ها استفاده نکنیم بالاجبار بایستی 16 عدد از پین های میکرو کنترلر را برای انتخاب ستون ها در نظر بگیریم ، در صورتی که هنگام استفاده از IC مزبور تعداد پین های استفاده شده برای کنترل ستون ها به 4 عدد کاهش می یابد .

نحوه عملکرد 74154 بدین صورت می باشد که در حالت عادی هنگامی که چهار بیت ورودی صفر می باشد وضعیت منطقی خروجی های Q1 تا Q15 به صورت HIGH یا یک بوده و وضعیت خروجیQ0 به صورت LOW یا صفر خواهد بود. برای LOW کردن هر کدام از خروجی مورد نظر بایستی شماره خروجی به صورت باینری به 4 پین ورودی ABCD اعمال شود. به طور مثال برای LOW کردن خروجی شماره 1 بایستی ورودی A=1 ، B=0 ، C=0 ، D=0 باشد.

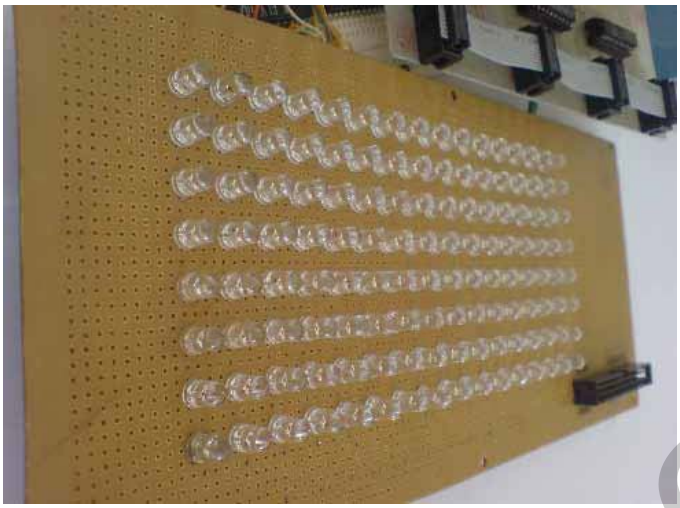
شکل 4-14 روش معمول برقراری اتصالات آی سی 74HC154 را به صورت انتخاب گر (4 به 16) نشان می دهد .

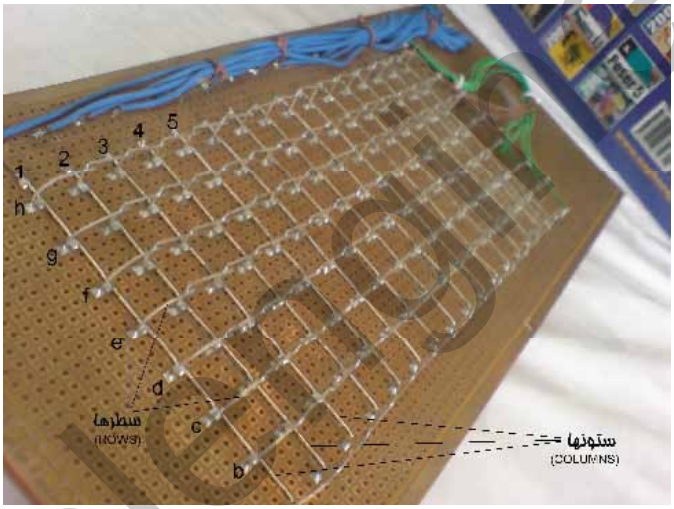


شکل 4-14 روش معمول برقراری اتصالات در آی سی 74HC154

برای ساختن ماتریس LED می توانید از یک برد سوراخ دار استفاده کنید تعداد LED های به کار رفته در این ماتریس (8*16) عدد می باشد شما می توانید سطر ها و ستون ها را بسته به سلیقه خودتان آنود یا کاتود کنید ولی در این پروژه از ستون ها به عنوان کاتود و از سطر ها به عنوان آنود استفاده شده است .

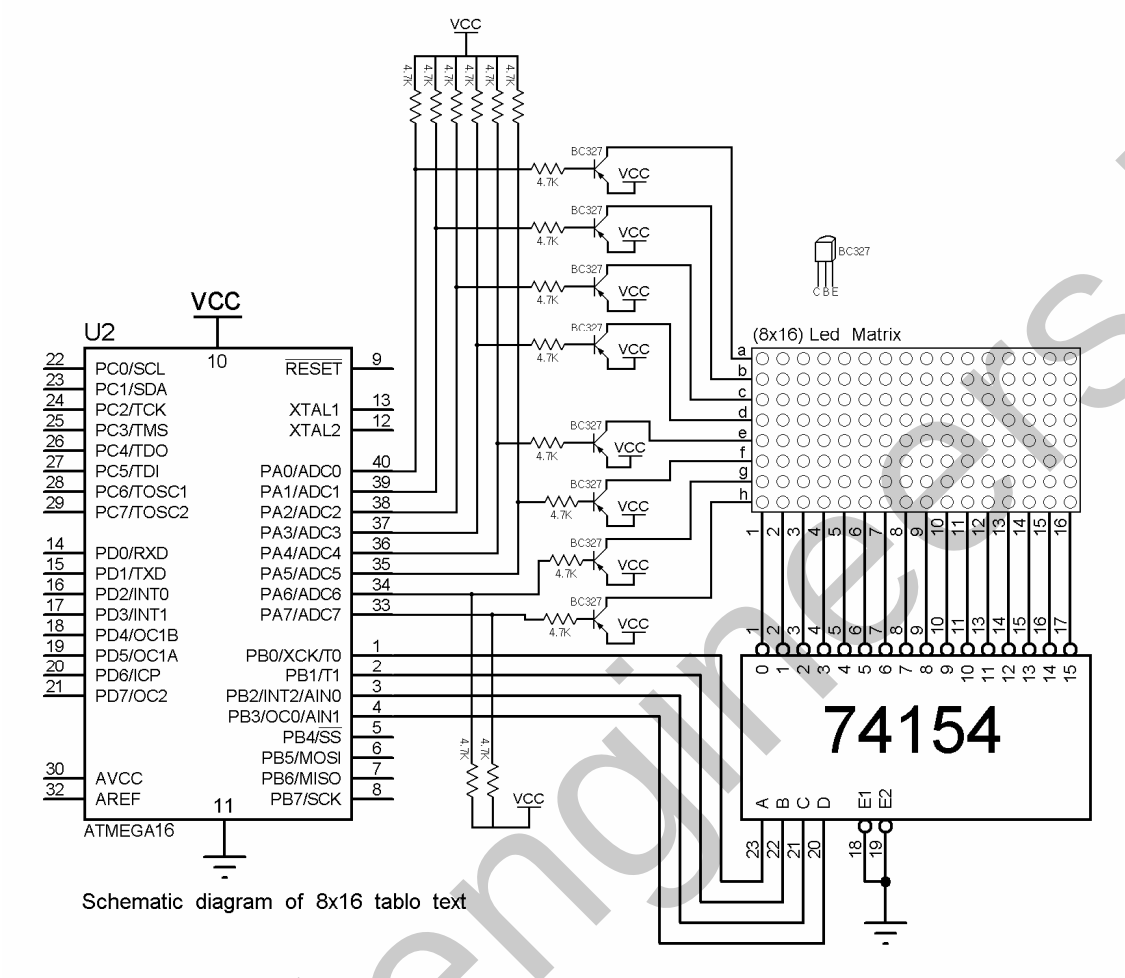
یعنی برای انتخاب هر کدام از ستون ها بایستی آن ستون را صفر و برای انتخاب هر کدام از سطر ها بایستی آن سطر را یک کنیم. شکل 4-15 روش معمول ساختن ماتریس LED را نشان می دهد .





شکل 4-15 تصویری از نمای پشت و روبروی ماتریس LED ساخته شده

توجه داشته باشید به منظور ساده کردن سخت افزار ابتدا فقط سطر ها را توسط ترانزیستور های PNP، BC327 بافر کرده و از بافر کردن ستون ها توسط IC های ULN2803 صرف نظر می کنیم در این حالت شماتیک پروژه به صورت شکل 4-16 خواهد بود .



شکل 4-16 شماتیک طراحی شده برای تابلو روان (8*16)

برنامه اجرای متن فارسی بر روی ماتریس (8*16) :

برنامه نوشته شده برای اجرای متن فارسی که حرکت آن از سمت چپ به راست می باشد به صورت زیر است .

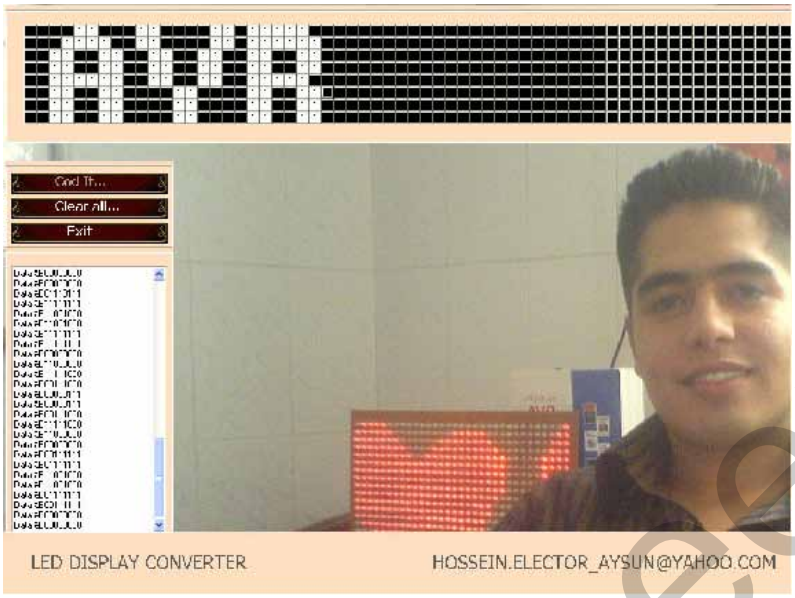
```
'PROGRAM OF PERSIAN TEXT FOR 8X16 MATRIX-----------------------------------------------
'COMPILER:1.11.8.7
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Column As Word , Scan_column As Byte
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
'-----------------------------------------------------------------------------------------
Start_program:
Control_lookup_number = 0 : Column = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-----------------------------------------------------------------------
Start_scan:
'LEVEL 1--------------------------------------------------------------------------------
Lookup_number = Control_lookup_number
Scan_column = 0
Portb = Scan_column
```

```
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 500
Porta = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 500
Porta = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan
Rotate_scan = 0
Incr Control_lookup_number
Incr Column
If Column =< 115 Then Goto Start_scan
'115=LOOKUP ROWS-LED MATRIX CULOMNS
'115=131-16
'-----------------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------
'START OF PERSIAN TEXT LOOKUP-----------------------------------------------------------------------
Persian_text:
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B010000000
Data &B010000000
Data &B010111110
Data &B010000000
Data &B001000000
Data &B000011100
Data &B000010100
Data &B000010100
Data &B000011100
Data &B000000100
Data &B000000100
Data &B000000100
Data &B000011111
Data &B000010101
Data &B000011101
Data &B000000001
Data &B000000000
Data &B000000000
Data &B001011111
Data &B000000001
Data &B000000001
Data &B000011100
Data &B001000100
```

```
Data &B011011100
Data &B001000100
Data &B000000100
Data &B000011110
Data &B000000001
Data &B000000001
Data &B000000001
Data &B000011110
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000011100
Data &B000010100
Data &B000010100
Data &B000010100
Data &B000010100
Data &B011111100
Data &B000000100
Data &B000000100
Data &B000000100
Data &B000000100
Data &B000011111
Data &B000000001
Data &B000000001
Data &B011111100
Data &B000000000
Data &B000000000
Data &B000011100
Data &B000010100
Data &B000010100
Data &B000010100
Data &B000010100
Data &B000000100
Data &B000000110
Data &B000000101
Data &B000000101
Data &B000000001
Data &B000000001
Data &B000011110
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B001011100
Data &B001000100
Data &B000000100
Data &B000000100
Data &B011111100
Data &B000000000
Data &B000000000
Data &B000011101
Data &B000000100
Data &B000000100
Data &B000000100
Data &B011111100
Data &B000000100
Data &B000000100
Data &B000000100
Data &B000011111
Data &B000010101
```

```
Data &B000011101
Data &B000000001
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000011111
Data &B000000001
Data &B000000000
Data &B000011110
Data &B000010101
Data &B000010101
Data &B000011101
Data &B000000000
Data &B000000000
Data &B011111100
Data &B000000000
Data &B000000000
Data &B000011110
Data &B000000001
Data &B001000001
Data &B000000001
Data &B000000001
Data &B000011110
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
'END OF PERSIAN TEXT LOOKUP-----------------------------------------------------------------------------
```

در برنامه فوق کدهای مربوط به متن فارسی که قرار است بر روی ماتریس LED نمایش داده شود در جدول PERSIAN_TEXT قرار دارد . کدهای باینری موجود در این جدول با استفاده از نرم افزار LED DISPLAY CONVERTER که در CD پیوست کتاب ارائه شده است طراحی می شود. شکل 17-4 نمائی از محیط اجرائی این نرم افزار را نشان می دهد .



شکل4-17 نمایی از نرم افزار LED Display converter

با توجه به شکل فوق با کلیک کردن بر روی هر کدام از خانه ها ، خانه مربوطه به حالت انتخاب رفته و عدد یک بر روی آن نوشته می شود و با کلیک مجدد بر روی آن از حالت انتخاب خارج خواهد شد . با زدن کلید CODE IT ، کد باینری مربوط به شکل طراحی شده در سمت چپ نرم افزار در قسمت مربوطه ارائه خواهد شد ، شما می توانید با کپی کردن آن در داخل برنامه BASCOM و در جدول LOOK UP مربوطه متن طراحی شده را بر روی ماتریس LED نمایش دهید .

تشریح نحوه عملکرد برنامه

در این برنامه ابتدا 16 عدد از کد های موجود در جدول PERSIAN_TEXT به صورت یک به یک بر روی ماتریس LED جاروب می شود ، سپس برای این که چشم انسان قادر به تشخیص تصاویر باشد این عمل 10 بار تکرار می شود ، یعنی عمل جاروب کردن 16 کد اول PERSIAN_TEXT ، 10 بار تکرار شده سپس یک کد از اول حذف شده و یک کد از آخر وارد کدهای جاروب ماتریس LED می شود .

به عبارت دیگر، نوشته موجود بر روی ماتریس LED یک ستون به سمت راست شیفت داده می شود ، پس از انجام عمل شیفت ، تصویر جدید نیز 10 بار بر روی ماتریس LED جاروب می شود و این روند تا زمانی که کدهای جدول LOOK UP تمام نشده ادامه خواهد یافت .

در قسمت LEVEL1 ابتدا مقدار متغیر CONTROL_LOOKUP_NUMBER صفر می باشد پس مقدار LOOKUP_NUMBER نیز صفر خواهد شد ، سپس متغیر SCAN_COLUMN برابر با صفر شده و بر روی پورت B قرار می گیرد ، 4 پین اول پورت B به 4 پایه ورودی IC , 74154 به منظور انتخاب کردن ستون مورد نظر مورد استفاده قرار می گیرد ، پس بدین ترتیب ستون شماره یک ماتریس LED انتخاب خواهد شد ، پس با

توجه به این که LOOKUP_NUMBER صفر است ، که شماره صفر از جدول PERSIAN_TEXT در متغیر ROW_DATA قرار می گیرد ، با توجه به این که برای انتخاب سطر ها از ترانزیستورهای PNP ، BC327 استفاده شده است ، اطلاعات اعمالی به بیس ترانزیستور بایستی ابتدا معکوس شده سپس به آن ها اعمال شود ، زیرا هر یک از ترانزیستور ها با اعمال صفر به بیس سوئیچ کرده و سطر مورد نظر را یک می کند ، بنابراین قبل از این که متغیر ROW_DATA را بر روی پورت A قرار دهیم آن را معکوس می کنیم. پس از این که اطلاعات مربوط به ستون شماره یک به آن اعمال شد ، اجرای برنامه به مدت 500 میکرو ثانیه متوقف می شود. سپس برای جلوگیری از پدیده پس ماند تصویر ابتدا با قرار دادن عدد255=&B11111111 بر روی پورت A همه ترانزیستور ها را به حالت قطع برده سپس عملیات مربوط به ستون بعدی را انجام می دهیم .

اگر قبل از این که ترانزیستور ها را به حالت قطع ببریم ستون بعدی را انتخاب کنیم اطلاعات مربوط به ستون قبلی برای مدت بسیار کوتاهی در ستون جدید انتخاب شده نمایش داده خواهد شد که این عمل باعث ایجاد پس ماند تصویر شده و کیفیت تصویر اصلی را کاهش خواهد داد.

در قسمت LEVEL2 عملیات انجام شده برای ستون اول بر روی 15 ستون بعدی نیز انجام می شود ، پس از پایان LEVEL2 ، 16 کد اول جدول PERSIAN_TEXT بر روی ماتریس LED جاروب شده است ولی برای این که چشم انسان قادر به تشخیص تصویر جاروب شده باشد بایستی عمل جاروب کامل ماتریس LED برای یک تصویر چندین بار تکرار شود ، عمل تکرار جاروب یک فریم کامل در قسمت CONTROL با استفاده از متغیر ROTATE_SCAN کنترل می شود .

در قسمت کنترل، پس از این که عمل جاروب یک فریم کامل 10 بار تکرار شد مقدار متغیر ROTATE_SCAN صفر شده و یک واحد به CONTROL_LOOKUP_NUMBER اضافه می شود. اضافه کردن یک واحد به این متغیر باعث می شود که اولین کد موجود در جدول PERSIAN_TEXT از کد های جاروب حذف شده و کد های جاروب یک واحد به جلو شیفت داده شود .

پس از اضافه کردن یک واحد به متغیر CONTROL_LOOKUP_NUMBER یک واحد به متغیر COLUMN اضافه می شود و تا زمانی که متغیر COLUMN کوچکتر یا مساوی عدد 115 باشد عمل جاروب تصاویر جدید ادامه خواهد یافت . عدد 115 از فرمول زیر محاسبه شده است .

115= (PERSIAN_TEXT در جدول تعداد کد های جاروب)-(LED تعداد ستون های ماتریس)

اگر COLUMN بزرگتر از 115 شود برنامه RESET شده و از نو اجرا خواهد شد. برای بدست آوردن حداکثر مقدار زمان تاخیر برای هر ستون می توانید از فرمول زیر استفاده کنید .

WAIT=20ms/(LED تعداد ستون های ماتریس)=20ms/16=1.25ms

در این برنامه تاخیر برای هر ستون برابر با 500 میکروثانیه در نظر گرفته شده است. شما می توانید سرعت حرکت را با تغییر دادن میزان تاخیر و تعداد تکرار هر فریم تغییر دهید.

برای تسلط بیشتر تعداد تکرار هر فریم و میزان تاخیر در هر ستون را تغییر داده و نتایج بدست آمده را مشاهده کنید .

با کمی دقت در کد های جدول LOOK UP ، می توانید متنی را که بر روی ماتریس LED نمایش داده خواهد شد ، تشخیص دهید .

برنامه اجرای انیمیشن بر روی ماتریس LED (8*16)

برنامه نوشته شده برای اجرای انیمیشن دو بعدی بر روی ماتریس LED (8*16) به صورت زیر می باشد .

```
'PROGRAM OF ANIMATION FOR 8X16 MATRIX-----------------------------------------------
'COMPILER:1.11.8.7
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Figure_number As Word , Scan_column As Byte
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
'-----------------------------------------------------------------------------------------
Start_program:
Control_lookup_number = 0 : Figure_number = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-------------------------------------------------------------------------
Start_scan:
'LEVEL 1----------------------------------------------------------------------------------
Lookup_number = Control_lookup_number
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Animat)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 500
Porta = 255
'LEVEL 2----------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Animat)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 500
Porta = 255
Next Scan
'CONTROL----------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 20 Then Goto Start_scan
Rotate_scan = 0
Control_lookup_number = Control_lookup_number + 16
'16=MATRIX COLUMNS
Incr Figure_number
If Figure_number < 5 Then Goto Start_scan
```

```
'5=ANIMATION FIGURES
'-----------------------------------------------------------------------------------------------------------------
Figure_number = 0
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------------
'START OF ANIMAT LOOKUP---------------------------------------------------------------------------------------
Animat:
'FIGURE1
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
'FIGURE2
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00011000
Data &B00011000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
'FIGURE3
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00011000
Data &B00100100
Data &B00100100
Data &B00011000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
'FIGURE4
Data &B00000000
Data &B00000000
Data &B00000000
```

```
Data &B00000000
Data &B00000000
Data &B00011000
Data &B00100100
Data &B01000010
Data &B01000010
Data &B00100100
Data &B00011000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
'FIGURE5
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00111100
Data &B01000010
Data &B10000001
Data &B10000001
Data &B10000001
Data &B10000001
Data &B01000010
Data &B00111100
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
'END OF ANIMAT LOOKUP-----------------------------------------------------------------------------
```

با توجه به این که با نشان دادن تصاویر به صورت پشت سر هم ، می توانیم یک تصویر متحرک ایجاد کنیم ، در برنامه فوق ابتدا یک فریم کامل جاروب شده و پس از 20 بار تکرار به جای این که کد های جاروب موجود در جدول ANIMAT یک عدد شیفت داده شوند به اندازه ستون های یک فریم کامل یعنی 16 عدد شیفت داده می شوند ، با استفاده از این روش شما می توانید انیمیشن 2 بعدی بر روی ماتریس LED نمایش دهید ، انیمیشن ایجاد شده توسط برنامه فوق بدین صورت می باشد که ابتدا یک نقطه بر روی ماتریس LED ایجاد شده و در طول 4 فریم به یک دایره بزرگ تبدیل می شود .

برنامه نوشتن متن انگلیسی بر روی ماتریس 8*16

برنامه نوشته شده برای اجرای متن انگلیسی که حرکت آن از راست به چپ می باشد به صورت زیر است .

```
'PROGRAM OF ENGLISH TEXT FOR 8X16 TABLO TEXT-----------------------------------------
'COMPILER:1.11.8.7
$regfile = "M16DEF.DAT"
$crystal = 8000000
Dim Column As Word , Scan_column As Byte
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
'-------------------------------------------------------------------------------------------
Start_program:
Column = 0 : Scan_column = 15
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
```

```
'START OF PROGRAM-----------------------------------------------------------------------------------------
Control_lookup_number = 144
'144=LOOKUP ROWS
Start_scan:
'LEVEL 1-------------------------------------------------------------------------------------------------
Lookup_number = Control_lookup_number
Scan_column = 15
Portb = Scan_column
Row_data = Lookup(lookup_number , Engish_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Waitus 500
Porta = 255
'LEVEL 2-------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Decr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Engish_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Waitus 500
Porta = 255
Next Scan
'CONTROL-------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan
Rotate_scan = 0
Decr Control_lookup_number
Incr Column
If Column =< 128 Then Goto Start_scan
'128=(LOOKUP ROWS-LED MATRIX COLUMNS)
'128=(144-16)
'-----------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program
'END OF PROGRAM-------------------------------------------------------------------------------------------
'START OF ENGLISH TEXT LOOKUP-----------------------------------------------------------------------------
Engish_text:
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B01110001
Data &B11011011
Data &B10001100
Data &B10001000
Data &B10001000
Data &B10001000
Data &B11111111
Data &B11111111
Data &B00000000
Data &B00000000
```

```
Data &B00111100
Data &B01000010
Data &B10000001
Data &B10000001
Data &B10000001
Data &B10000001
Data &B01000010
Data &B00111100
Data &B00000000
Data &B00000000
Data &B10000000
Data &B10000000
Data &B11111111
Data &B11111111
Data &B10000000
Data &B10000000
Data &B00000000
Data &B00000000
Data &B01100010
Data &B11000001
Data &B10000001
Data &B10000001
Data &B10000001
Data &B01000010
Data &B00111100
Data &B00000000
Data &B00000000
Data &B10010001
Data &B10010001
Data &B10010001
Data &B10010001
Data &B10010001
Data &B11111111
Data &B11111111
Data &B00000000
Data &B00000000
Data &B00000001
Data &B00000001
Data &B00000001
Data &B00000001
Data &B11111111
Data &B11111111
Data &B00000000
Data &B00000000
Data &B10010001
Data &B10010001
Data &B10010001
Data &B10010001
Data &B10010001
Data &B11111111
Data &B11111111
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B11111111
Data &B00000111
Data &B00000110
Data &B00001100
Data &B00011000
Data &B01110000
Data &B11111111
```

```
Data &B11111111
Data &B00000000
Data &B00000000
Data &B11111111
Data &B11111111
Data &B00000000
Data &B00000000
Data &B10010001
Data &B10010001
Data &B10010001
Data &B10010001
Data &B10010001
Data &B11111111
Data &B11111111
Data &B00000000
Data &B00000000
Data &B01100110
Data &B11001101
Data &B10001001
Data &B10001001
Data &B10011001
Data &B11010001
Data &B01110010
Data &B00000000
Data &B00000000
Data &B01100110
Data &B11001101
Data &B10001001
Data &B10001001
Data &B10011001
Data &B11010001
Data &B01110010
Data &B00000000
Data &B00000000
Data &B00111100
Data &B01000010
Data &B10000001
Data &B10000001
Data &B10000001
Data &B10000001
Data &B01000010
Data &B00111100
Data &B00000000
Data &B00000000
Data &B11111111
Data &B00001000
Data &B00001000
Data &B00001000
Data &B00001000
Data &B00001000
Data &B11111111
Data &B11111111
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
```

```
'END OF ENGLISH TEXT LOOKUP-----------------------------------------------------------------------
```

با توجه به این که حرکت متن انگلیسی از راست به چپ می باشد ، جاروب ستون ها از ستون آخر به ستون اول و همچنین نحوه خواندن اطلاعات موجود در جدول ENGLISH_TEXT نیز از آخر به اول خواهد بود ، بنابراین این مقدار اولیه CONTROL_LOOKUP_NUMBER بایستی برابر با تعداد کد های موجود در جدول ENGLISH_TEXT و مقدار اولیه متغیر SCAN_ COLUMN برابر با 15 باشد ، با کمی دقت بر روی کد های جدول ENGLISH_TEXT می توانید متن نمایش داده شده بر روی ماتریس LED را تشخیص دهید.

برنامه اجرای متن فارسی ، انیمیشن و متن انگلیسی به صورت پشت سر هم در تابلوی (8*16)

با قرار دادن برنامه های مربوط به متن فارسی ، انیمیشن و متن انگلیسی به صورت پشت سر هم می توانید هر 3 حالت را بر روی ماتریس LED داشته باشید .

برنامه نوشته شده به صورت زیر می باشد.

```
'PROGRAM OF PERSIAN TEXT,ANIMATION AND ENGLISH TEXT---------------------------
'COMPILER:1.11.8.7
$regfile = "M16DEF.DAT"
$crystal = 8000000
Dim Column As Word , Scan_column As Byte , Figure_number As Word
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
'-----------------------------------------------------------------------------------------------------
Start_program:
Control_lookup_number = 0 : Column = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-----------------------------------------------------------------------------------
Start_scan:
'LEVEL 1--------------------------------------------------------------------------------------------
Lookup_number = Control_lookup_number
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 500
Porta = 255
'LEVEL 2--------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 500
```

```
Porta = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan
Rotate_scan = 0
Incr Control_lookup_number
Incr Column
If Column =< 115 Then Goto Start_scan
'115=LOOKUP CULUMNS-LED MATRIX CULOMNS
'115=131-16
'-------------------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program2
'END OF PROGRAM-----------------------------------------------------------------------------------------
'START OF PERSIAN TEXT LOOKUP---------------------------------------------------------------------------
Persian_text:
Data &B000000000
Data &B010000000
Data &B010000000
Data &B010111110
Data &B010000000
Data &B001000000
    .
    .
    .
'END OF PERSIAN TEXT LOOKUP-----------------------------------------------------------------------------
Start_program2:
Control_lookup_number = 0 : Figure_number = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM---------------------------------------------------------------------------------------
Start_scan2:
'LEVEL 1------------------------------------------------------------------------------------------------
Lookup_number = Control_lookup_number
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Animat)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 500
Porta = 255
'LEVEL 2------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Animat)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 500
Porta = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 20 Then Goto Start_scan2
Rotate_scan = 0
Control_lookup_number = Control_lookup_number + 16
'16=MATRIX COLUMNS
Incr Figure_number
If Figure_number < 5 Then Goto Start_scan2
'5=ANIMATION FIGURES
```

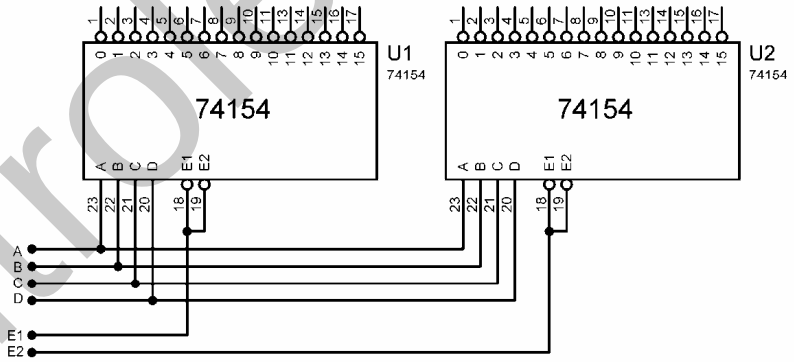
```
Figure_number = 0
Goto Start_program3
'END OF PROGRAM-----------------------------------------------------------------------------------------
'START OF ANIMAT LOOKUP---------------------------------------------------------------------------------
Animat:
'FIGURE1
    .
    .
    .
'FIGURE5
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00111100
Data &B01000010
Data &B10000001
Data &B10000001
Data &B10000001
Data &B10000001
Data &B01000010
Data &B00111100
Data &B00000000
Data &B00000000
Data &B00000000
Data &B00000000
'END OF ANIMAT LOOKUP----------------------------------------------------------------------------------
Start_program3:
Column = 0 : Scan_column = 15
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM---------------------------------------------------------------------------------------
Control_lookup_number = 144
'144=LOOKUP COLUMNS
Start_scan3:
'LEVEL 1------------------------------------------------------------------------------------------------
Lookup_number = Control_lookup_number
Scan_column = 15
Portb = Scan_column
Row_data = Lookup(lookup_number , Engish_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Waitus 500
Porta = 255
'LEVEL 2------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Decr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Engish_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Waitus 500
Porta = 255
Next Scan
'CONTROL------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan3
Rotate_scan = 0
Decr Control_lookup_number
Incr Column
If Column =< 128 Then Goto Start_scan3
```

```
'128=(LOOKUP COLUMNS-LED MATRIX COLUMNS)
'128=(144-16)
'-----------------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------
'START OF ENGLISH TEXT LOOKUP-----------------------------------------------------------------------
Engish_text:
Data &B00000000
Data &B01110001
Data &B11011011
Data &B10001100
Data &B10001000
Data &B10001000
Data &B10001000
Data &B11111111
Data &B11111111
Data &B00000000
Data &B00000000
    .
    .
    .
'END OF ENGLISH TEXT LOOKUP-----------------------------------------------------------------------
```

در برنامه فوق دقیقا از جدول های ، LOOKUP برنامه های قبلی استفاده شده است ، ولی به علت طولانی بودن این جداول تعدادی از کد های آنها را حذف کرده ایم البته برنامه کامل درCD پیوست کتاب ارائه شده است .

نحوه افزایش ستون های تابلو روان

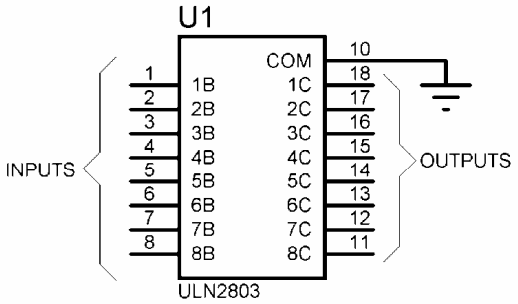
برای افزایش ستون های تابلو روان بایستی از چندین آی سی ، انتخاب گر 74154 استفاده کنیم به عنوان مثال برای طراحی یک تابلوی 32*8 بایستی از دو عدد 74154 برای انتخاب ستونها استفاده کنیم. شکل 18-4 نحوه انجام این عملکرد را نشان می دهد .



شکل 18-4 نحوه افزایش ستون های تابلو با استفاده از آی سی 74154

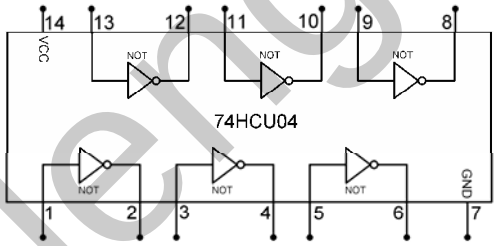
هنگامی که بخواهیم یکی از ستون های 1 تا 16 را انتخاب کنیم U2 غیر فعال بوده وU1 فعال است و هنگامی که بخواهیم یکی از ستون های 17 تا 32 را انتخاب کنیم U1 غیر فعال بوده و U2 فعال است ، با استفاده از روش فوق شما قادر خواهید بود طول تابلو را تا هر اندازه که بخواهید افزایش دهید. برای افزایش نور LED ها می

توانید از IC های بافر ULN2803 استفاده کنید. نحوه برقراری اتصالات در آی سی ULN2803 در شکل 4-19 ارائه شده است .



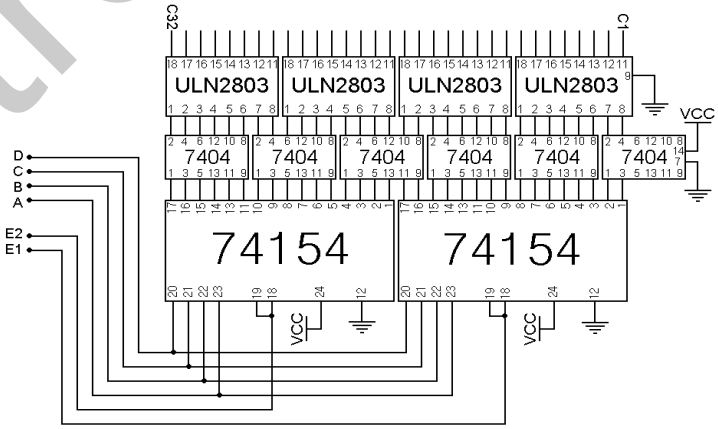
شکل 4-19 نحوه برقراری اتصالات در آی سی ULN2803

توجه داشته باشید که آی سی ULN2803 فقط دارای پایه GND بوده و پایه VCC ندارد و همچنین اطلاعات بافر شده در خروجی IC به صورت معکوس قرار خواهند گرفت . به همین دلیل قبل از اعمال اطلاعات به ورودی آی سی ULN2803 ابتدا آن ها را توسط IC های 7404 معکوس کرده ، سپس به ورودی ULN2803 اعمال می کنیم ، شکل 4-20 شماتیک داخلی IC ، 7404 را نشان می دهد.



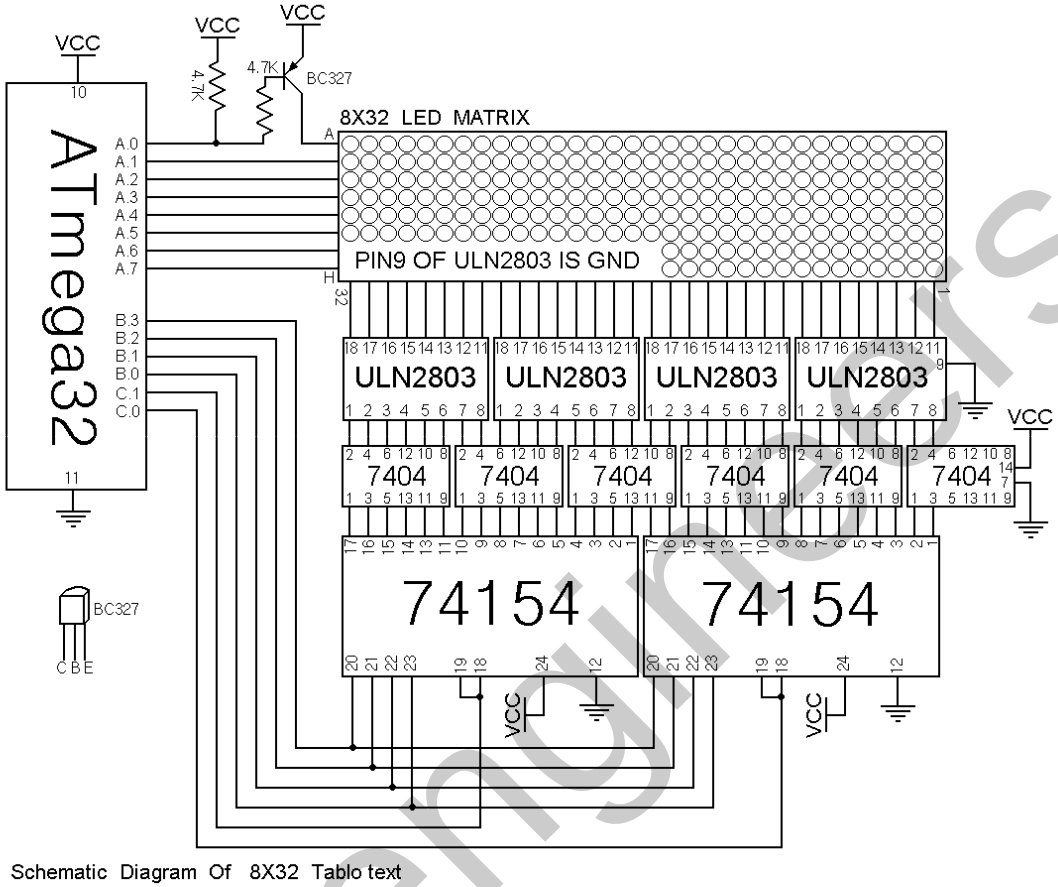
شکل 4-20 شماتیک داخلی آی سی 74HC04

بدین ترتیب مدار انتخاب گر ستون ها را به صورت شکل 4-21 تغییر می دهیم .



شکل 4-21 شماتیک مربوط به مدار بافر و انتخاب گر ستون ها

شماتیک تابلوی 32*8 در شکل 22-4 ارائه شده است .



شکل 22-4 شماتیک طراحی شده برای تابلو ی 32*8

توجه داشته باشید در شکل 22-4 مدار راه انداز سطر A که شامل ترانزیستور BC327 و دو عدد مقاومت 4.7K می باشد ، بایستی برای همه سطر ها بسته شود ، در شماتیک پروژه به علت این که کشیدن مدار درایور تمام سطر ها امکان پذیر نبود فقط درایور مربوط به سطر A کشیده شده است .

برنامه نوشته شده برای اجرای متن فارسی در تابلوی 32*8 به صورت زیر است .

```
'PROGRAM OF 8X32 TABLO TEXT-----------------------------------------------------------------------
'COMPILER:1.11.8.7
$regfile = "M16DEF.DAT"
$crystal = 8000000
Dim Column As Word , Scan_column As Byte
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
Config Pinc.0 = Output : Enable_74154_1 Alias Portc.0
Config Pinc.1 = Output : Enable_74154_2 Alias Portc.1
'-----------------------------------------------------------------------------------------------------
Start_program:
```

```
Control_lookup_number = 0 : Column = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-----------------------------------------------------------------------------------------
Start_scan:
Lookup_number = Control_lookup_number
Reset Enable_74154_1 : Set Enable_74154_2
'NOW 74154_1 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255
'LEVEL 2--------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255
Next Scan
'--------------------------------------------------------------------------------------------------------
Set Enable_74154_1 : Reset Enable_74154_2
'NOW 74154_2 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255
'LEVEL 2--------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255
Next Scan
'CONTROL--------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan
Rotate_scan = 0
Incr Control_lookup_number
Incr Column
If Column =< 99 Then Goto Start_scan
'99=LOOKUP ROWS-LED MATRIX CULOMNS
'99=131-32
'--------------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program
```

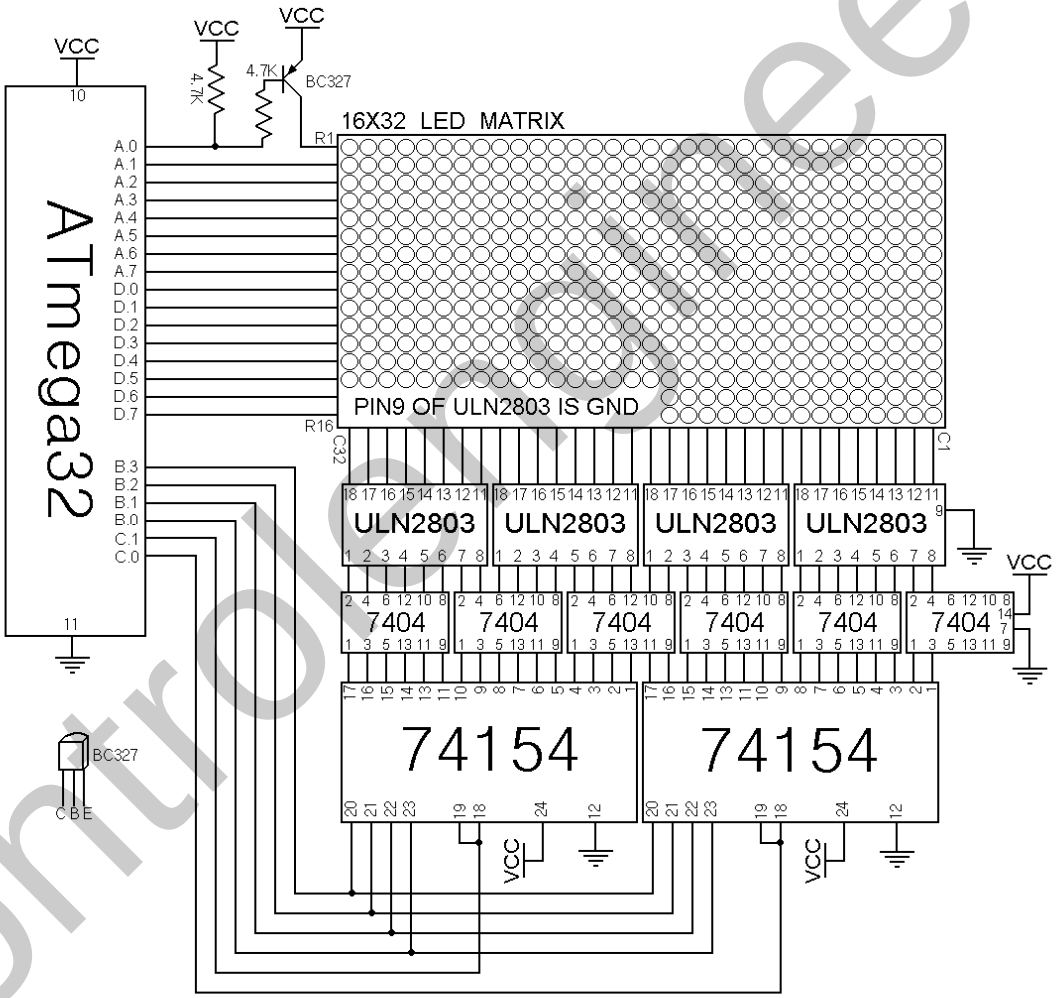
```
'END OF PROGRAM-----------------------------------------------------------------------------------------
'START OF PERSIAN TEXT LOOKUP-----------------------------------------------------------------------
Persian_text:
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B010000000
Data &B010000000
Data &B010111110
Data &B010000000
Data &B001000000
Data &B000011100
Data &B000010100
Data &B000010100
Data &B000011100
Data &B000000100
Data &B000000100
Data &B000000100
Data &B000011111
Data &B000010101
Data &B000011101
Data &B000000001
Data &B000000000
Data &B000000000
Data &B001011111
Data &B000000001
Data &B000000001
Data &B000011100
Data &B001000100
Data &B011011100
Data &B001000100
Data &B000000100
Data &B000011110
Data &B000000001
Data &B000000001
Data &B000011110
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000011100
Data &B000010100
Data &B000010100
Data &B000010100
Data &B000010100
Data &B011111100
Data &B000000100
Data &B000000100
Data &B000000100
Data &B000000100
Data &B000011111
Data &B000000001
Data &B000000001
Data &B011111100
Data &B000000000
Data &B000000000
Data &B000011100
```

```
Data &B000010100
Data &B000010100
Data &B000010100
Data &B000010100
Data &B000000100
Data &B000000110
Data &B000000101
Data &B000000101
Data &B000000001
Data &B000000001
Data &B000011110
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B001011100
Data &B001000100
Data &B000000100
Data &B000000100
Data &B011111100
Data &B000000000
Data &B000000000
Data &B000011101
Data &B000000100
Data &B000000100
Data &B000000100
Data &B011111100
Data &B000000100
Data &B000000100
Data &B000000100
Data &B000011111
Data &B000010101
Data &B000011101
Data &B000000001
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000011111
Data &B000000001
Data &B000000000
Data &B000011110
Data &B000010101
Data &B000010101
Data &B000011101
Data &B000000000
Data &B000000000
Data &B011111100
Data &B000000000
Data &B000000000
Data &B000011110
Data &B000000001
Data &B001000001
Data &B000000001
Data &B000000001
Data &B000011110
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
```

```
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
Data &B000000000
'END OF PERSIAN TEXT LOOKUP----------------------------------------------------------------------
```

طراحی تابلوی 16*32

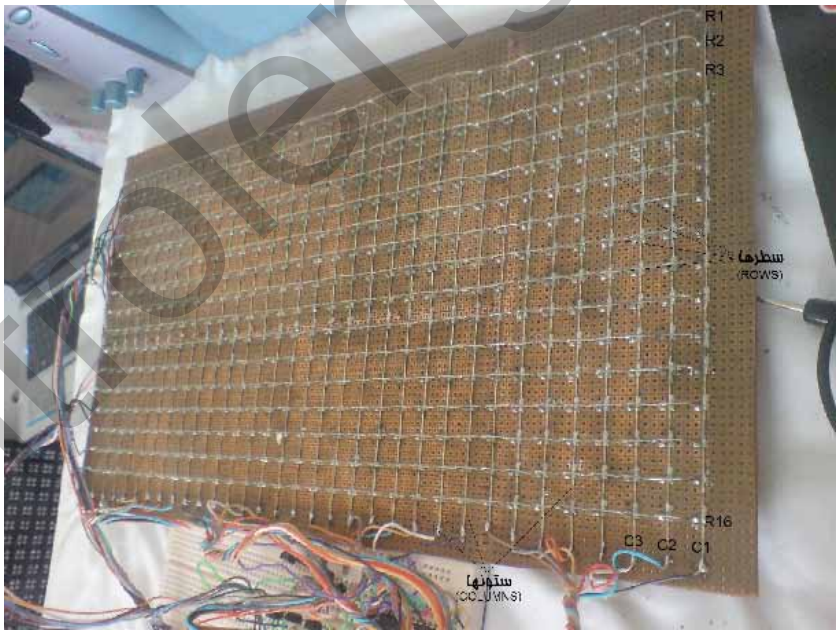
برای طراحی یک تابلوی 16*32 می توانید از شماتیک شکل 23-4 استفاده کنید .



Schematic Diagram Of 16x32 Tablo text

شکل 23-4 شماتیک طراحی شده برای تابلوی 16*32

در این پروژه از پورت A برای کنترل سطر های R1 تا R8 و از پورت D برای کنترل سطرهای R9 تا R16 استفاده شده است. نحوه انتخاب ستون ها نیز مانند پروژه های قبلی می باشد شکل های 24-4 نمائی از ماتریس LED (16*32) و سطر و ستون های آن را نشان می دهند.



شکل 24-4 نمائی از ماتریس LED طراحی شده برای پروژه و سطر و ستون های آن

برنامه اجرای متن فارسی در تابلوی (16*32)

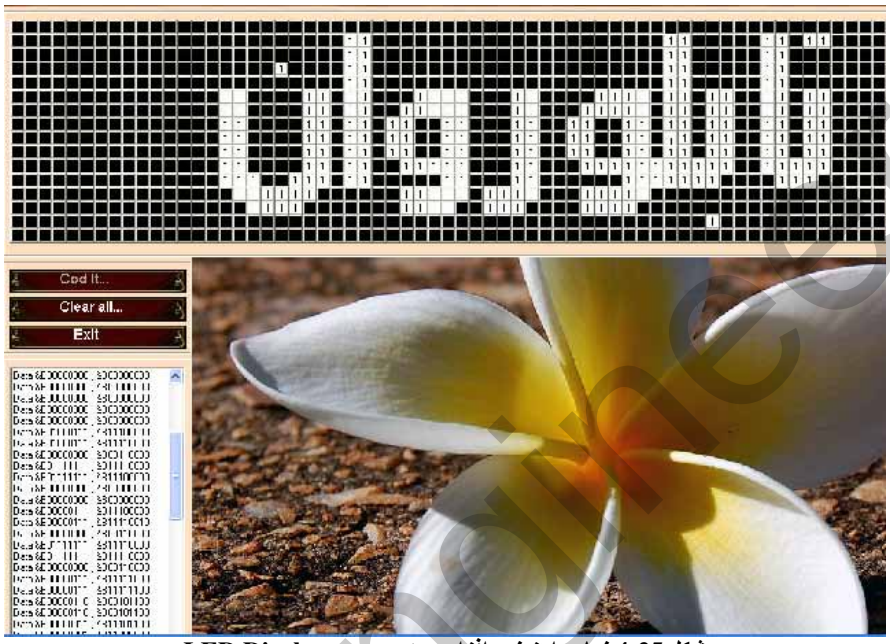
برنامه نوشته شده برای اجرای متن فارسی در تابلوی 32*16 به صورت زیر است.

```
'PROGRAM OF PERSIAN TEXT FOR 16X32 TABLO TEXT-----------------------------------------
'COMPILER:1.11.8.7
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Column As Word , Scan_column As Byte
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
Config Portd = Output
Config Pinc.0 = Output : Enable_74154_1 Alias Portc.0
Config Pinc.1 = Output : Enable_74154_2 Alias Portc.1
'-----------------------------------------------------------------------------------------
Start_program:
Control_lookup_number = 0 : Column = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM----------------------------------------------------------------------
Start_scan:
Lookup_number = Control_lookup_number
Reset Enable_74154_1 : Set Enable_74154_2
'NOW 74154_1 IS ACTIVE
'LEVEL 1-------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2-------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'-----------------------------------------------------------------------------------------
Set Enable_74154_1 : Reset Enable_74154_2
'NOW 74154_2 IS ACTIVE
'LEVEL 1-------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
```

```
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan
Rotate_scan = 0
Control_lookup_number = Control_lookup_number + 2
Incr Column
If Column =< 52 Then Goto Start_scan
'52=LOOKUP ROWS-LED MATRIX CULOMNS
'52=84-32
'-----------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------
'START OF PERSIAN TEXT LOOKUP---------------------------------------------------------------------------
Persian_text:
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B11011111 , &B11100000
Data &B11011111 , &B11110000
Data &B11000000 , &B00110000
Data &B00000000 , &B00110000
Data &B00000000 , &B00110000
Data &B11111111 , &B11110000
Data &B11111111 , &B11100000
Data &B00000000 , &B00000000
Data &B00111111 , &B11100000
Data &B00111111 , &B11110110
Data &B00000000 , &B00110110
Data &B00000000 , &B00110000
Data &B11111111 , &B11110000
```

```
Data &B11111111 , &B11110000
Data &B00000000 , &B00110000
Data &B00000000 , &B00110000
Data &B00111111 , &B11111100
Data &B00111111 , &B11111110
Data &B00110000 , &B00110110
Data &B00110000 , &B00110110
Data &B00110000 , &B01110110
Data &B00011100 , &B11100000
Data &B00001111 , &B11000000
Data &B00000011 , &B10000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00111111 , &B11111100
Data &B00111111 , &B11111110
Data &B00000000 , &B00000110
Data &B00000000 , &B00000110
Data &B00000000 , &B00000000
Data &B00111111 , &B11111100
Data &B00111111 , &B11111110
Data &B00110000 , &B00110110
Data &B00110000 , &B00110110
Data &B00110000 , &B00110110
Data &B00011000 , &B11100110
Data &B00001111 , &B11000000
Data &B00000111 , &B10000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B11111111 , &B11110000
Data &B11111111 , &B11110000
Data &B00000000 , &B00000000
Data &B00111111 , &B11000000
Data &B00111111 , &B11110000
Data &B00000000 , &B00111000
Data &B00000000 , &B00011100
Data &B11000000 , &B00001100
Data &B11000000 , &B00001100
Data &B00000000 , &B00001100
Data &B00000000 , &B00011100
Data &B00000000 , &B00111000
Data &B00111111 , &B11110000
Data &B00111111 , &B11100000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'END OF PERSIAN TEXT LOOKUP----------
```

در برنامه فوق کد های مربوط به متن فارسی که قرار است بر روی ماتریس LED نمایش داده شود ، در جدول PERSIAN_TEXT قرار دارد. کد های باینری موجود در این جدول با استفاده از نرم افزار LED DISPLAY CONVERTER که در CD پیوست کتاب ارائه شده طراحی شده است. شکل 25-4 نمائی از محیط اجرائی این نرم افزار را نشان می دهد .



شکل 25-4 نمایی ارز نرم افزار LED Display converter

با توجه به شکل فوق با کلیک کردن بر روی هر کدام از خانه ها ، خانه مربوطه به حالت انتخاب رفته و عدد یک بر روی آن نوشته می شود و با کلیک مجدد بر روی آن از حالت انتخاب خارج شده و با زدن کلید CODE IT کد مربوط به شکل طراحی شده در سمت چپ نرم افزار در قسمت مربوطه ارائه خواهد شد، شما می توانید با کپی کردن آن در داخل برنامه BASCOM و در جدول LOOKUP مربوطه متن طراحی شده را بر روی ماتریس LED نمایش دهید .

توجه داشته باشید که در قسمت CONTROL برنامه فوق ، بعد از این که تصویر 10 بار جاروب شد بایستی به مقدار CONTROL_LOOKUP_NUMBER، 2 واحد اضافه شود ، زیرا تعداد کد های مر بوط به یک ستون دو عدد می باشد. همچنین عدد 52 نیز از فرمول زیر محاسبه می شود .

52= (LOOKUP تعداد سطر های جدول)-(LED تعداد ستون های ماتریس)

52=84-32

برنامه اجرای انیمیشن در تابلوی (16*32)

برنامه نوشته شده برای اجرای انیمیشن به صورت زیر است .

```
'PROGRAM OF ANIMATION FOR 16X32 TABLO TEXT-----------------------------------------
'COMPILER:1.11.8.7
```

```
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Figure_number As Word , Scan_column As Byte
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
Config Portd = Output
Config Pinc.0 = Output : Enable_74154_1 Alias Portc.0
Config Pinc.1 = Output : Enable_74154_2 Alias Portc.1
'-----------------------------------------------------------------------------------------------------------
Start_program:
Control_lookup_number = 0 : Figure_number = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-----------------------------------------------------------------------------------------
Start_scan:
Lookup_number = Control_lookup_number
Reset Enable_74154_1 : Set Enable_74154_2
'NOW 74154_1 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2--------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'-----------------------------------------------------------------------------------------------------------
Set Enable_74154_1 : Reset Enable_74154_2
'NOW 74154_2 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
```

```
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 20 Then Goto Start_scan
Rotate_scan = 0
Control_lookup_number = Control_lookup_number + 64
'64=ONE FIGURE DATAS
Incr Figure_number
If Figure_number < 5 Then Goto Start_scan
'5=ANIMATION FIGURES
'-----------------------------------------------------------------------------------------------------
Figure_number = 0
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------
'START OF Animation LOOKUP-----------------------------------------------------------------------------
Animation:
'FIGURE1
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000001 , &B10000000
Data &B00000001 , &B10000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
```

```
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'FIGURE2
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000001 , &B10000000
Data &B00000010 , &B01000000
Data &B00000010 , &B01000000
Data &B00000001 , &B10000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'FIGURE3
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000011 , &B11000000
Data &B00000100 , &B00100000
Data &B00000100 , &B00100000
Data &B00000100 , &B00100000
Data &B00000100 , &B00100000
Data &B00000011 , &B11000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
```

```
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'FIGURE4
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000111 , &B11100000
Data &B00001000 , &B00010000
Data &B00001000 , &B00010000
Data &B00001000 , &B00010000
Data &B00001000 , &B00010000
Data &B00001000 , &B00010000
Data &B00001000 , &B00010000
Data &B00000111 , &B11100000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'FIGURE5
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00001111 , &B11110000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00001111 , &B11110000
Data &B00000000 , &B00000000
```

```
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'END OF Animation LOOKUP-----------------------------------------------------------------------------
```

توجه داشته باشید که در قسمت CONTROL برنامه فوق پس از این که هر تصویر 20 بار جاروب شد به اندازه تعداد دیتاهای یک فریم یا تصویر کامل یعنی64 واحد بایستی به متغیرCONTROL_LOOKUP_NUMBER اضافه شود. برای تغییر سرعت انیمیشن می توانید تعداد تکرار جاروب هر فریم را که در این برنامه برابر با 20 است تغییر دهید. با اجرای برنامه بالا ابتدا یک نقطه بر روی ماتریس LED ظاهر شده سپس در طول 5 فریم تبدیل به یک مربع بزرگ می شود .

برنامه اجرای متن انگلیسی در تابلوی 16*32

برنامه نوشته شده برای اجرای متن انگلیسی به صورت زیر است .

```
'PROGRAM OF ENGLISH TEXT FOR 16X32 TABLO TEXT-----------------------------------------
'COMPILER:1.11.8.7
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Column As Word , Scan_column As Byte
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
Config Portd = Output
Config Pinc.0 = Output : Enable_74154_1 Alias Portc.0
Config Pinc.1 = Output : Enable_74154_2 Alias Portc.1
'-----------------------------------------------------------------------------------------
Start_program:
Control_lookup_number = 263 : Column = 0 : Scan_column = 0
'263=LOOKUP DATAS-1
'263=264-1
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-------------------------------------------------------------------------
Start_scan:
Lookup_number = Control_lookup_number
Set Enable_74154_1 : Reset Enable_74154_2
'NOW 74154_2 IS ACTIVE
'LEVEL 1----------------------------------------------------------------------------------
Scan_column = 15
Portb = Scan_column
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Portd = Row_data
DECR Lookup_number
```

```
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Decr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Portd = Row_data
Decr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'-------------------------------------------------------------------------------------------------------------
Reset Enable_74154_1 : Set Enable_74154_2
'NOW 74154_1 IS ACTIVE
'LEVEL 1-----------------------------------------------------------------------------------------------------
Scan_column = 15
Portb = Scan_column
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Portd = Row_data
Decr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Decr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Portd = Row_data
Decr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan
Rotate_scan = 0
Control_lookup_number = Control_lookup_number - 2
Incr Column
If Column =< 100 Then Goto Start_scan
'100=LOOKUP ROWS-LED MATRIX CULOMNS
'100=132-32
'-------------------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------------
'START OF ENGLISH TEXT LOOKUP---------------------------------------------------------------------------------
```

```
English_text:
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000100 , &B00010000
Data &B00001100 , &B00011000
Data &B00011000 , &B00001100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00011000 , &B00001100
Data &B00001111 , &B11111000
Data &B00000111 , &B11110000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00011111 , &B11111100
Data &B00011111 , &B11111100
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00011111 , &B11111100
Data &B00011111 , &B11111100
Data &B00000000 , &B00111000
Data &B00000000 , &B01110000
Data &B00000000 , &B11100000
Data &B00000011 , &B10000000
Data &B00000111 , &B00000000
Data &B00001110 , &B00000000
Data &B00011111 , &B11111100
Data &B00011111 , &B11111100
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000011 , &B11100000
Data &B00001111 , &B11111000
Data &B00001000 , &B00001000
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00011000 , &B00001100
Data &B00001000 , &B00001000
Data &B00001111 , &B11111000
Data &B00000011 , &B11100000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000110 , &B00001100
Data &B00001111 , &B00111100
Data &B00010000 , &B11100000
Data &B00010000 , &B11000000
Data &B00010000 , &B10000000
Data &B00010000 , &B10000000
Data &B00010000 , &B10000000
Data &B00010000 , &B10000000
Data &B00011111 , &B11111100
```

```
Data &B00011111 , &B11111100
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00010000 , &B00000000
Data &B00010000 , &B00000000
Data &B00010000 , &B00000000
Data &B00011111 , &B11111100
Data &B00011111 , &B11111100
Data &B00010000 , &B00000000
Data &B00010000 , &B00000000
Data &B00010000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000110 , &B00110000
Data &B00001110 , &B00111000
Data &B00011000 , &B00001100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00001000 , &B00001000
Data &B00001111 , &B01111000
Data &B00000011 , &B11100000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00010000 , &B00000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00011111 , &B11111100
Data &B00011111 , &B11111100
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000100
Data &B00000000 , &B00000100
Data &B00000000 , &B00000100
Data &B00000000 , &B00000100
Data &B00000000 , &B00000100
Data &B00011111 , &B11111100
Data &B00011111 , &B11111100
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00010000 , &B00000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00010000 , &B10000100
Data &B00011111 , &B11111100
Data &B00011111 , &B11111100
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
```

```
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'END OF ENGLISH TEXT LOOKUP-----------------------------------------------------------------
```

با توجه به این که حرکت متن انگلیسی از چپ به راست می باشد جاروب ستون ها از ستون آخر به ستون اول و همچنین نحوه خواندن اطلاعات موجود در جدول ENGLISH_TEXT نیز از آخر به اول خواهد بود . بنابر این مقدار اولیه CONTROL_LOOKUP_NUMBER از فرمول زیر محاسبه می شود .

CONTROL_LOOKUP_NUMBER = (ENGLSH_TEXT تعداد کدهای جدول) - 1

برنامه اجرای متن فارسی ، متن انگلیسی و انیمیشن در تابلوی 16*32

با قرار دادن برنامه های مربوط به متن فارسی ، متن انگلیسی و انیمیشن به صورت پشت سر هم می توانید هر 3 حالت را بر روی ماتریس LED داشته باشید ، برنامه نوشته شده برای عملکرد فوق به صورت زیر می باشد .

```
'PROGRAM OF PERSIAN TEXT,ANIMATION AND ENGLISH TEXT-------------------------
'COMPILER:1.11.8.7
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Column As Word , Scan_column As Byte , Figure_number As Word
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
Config Portd = Output
Config Pinc.0 = Output : Enable_74154_1 Alias Portc.0
Config Pinc.1 = Output : Enable_74154_2 Alias Portc.1
'-----------------------------------------------------------------------------------------------
Start_program:
Control_lookup_number = 0 : Column = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-----------------------------------------------------------------------------
Start_scan:
Lookup_number = Control_lookup_number
Reset Enable_74154_1 : Set Enable_74154_2
'NOW 74154_1 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2--------------------------------------------------------------------------------------
```

```
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'-----------------------------------------------------------------------------------------------------------------
Set Enable_74154_1 : Reset Enable_74154_2
'NOW 74154_2 IS ACTIVE
'LEVEL 1-----------------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan
Rotate_scan = 0
Control_lookup_number = Control_lookup_number + 2
Incr Column
If Column =< 52 Then Goto Start_scan
'52=LOOKUP ROWS-LED MATRIX CULOMNS
'52=84-32
'-----------------------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program2
'END OF PROGRAM-----------------------------------------------------------------------------------------------
'START OF PERSIAN TEXT LOOKUP---------------------------------------------------------------------------------
Persian_text:
Data &B00000000 , &B00000000
Data &B11011111 , &B11100000
```

```
Data &B11011111 , &B11110000
Data &B11000000 , &B00110000
Data &B00000000 , &B00110000
Data &B00000000 , &B00110000
Data &B11111111 , &B11110000
Data &B11111111 , &B11100000
Data &B00000000 , &B00000000
    .
    .
    .
    .
'END OF PERSIAN TEXT LOOKUP-----------------------------------------------------------------------
Start_program2:
Control_lookup_number = 263 : Column = 0 : Scan_column = 0
'263=LOOKUP DATAS-1
'263=264-1
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-----------------------------------------------------------------------------------
Start_scan2:
Lookup_number = Control_lookup_number
Set Enable_74154_1 : Reset Enable_74154_2
'NOW 74154_2 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------------
Scan_column = 15
Portb = Scan_column
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Portd = Row_data
Decr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2--------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Decr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Portd = Row_data
Decr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'--------------------------------------------------------------------------------------------------
Reset Enable_74154_1 : Set Enable_74154_2
'NOW 74154_1 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------------
Scan_column = 15
Portb = Scan_column
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
```

```
Portd = Row_data
Decr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Decr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Porta = Row_data
Decr Lookup_number
Row_data = Lookup(lookup_number , English_text)
Row_data = Not Row_data
Portd = Row_data
Decr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan2
Rotate_scan = 0
Control_lookup_number = Control_lookup_number - 2
Incr Column
If Column =< 100 Then Goto Start_scan2
'100=LOOKUP ROWS-LED MATRIX CULOMNS
'100=132-32
'-------------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program3
'END OF PROGRAM-----------------------------------------------------------------------------------------
'START OF ENGLISH TEXT LOOKUP---------------------------------------------------------------------------
English_text:
Data &B00000000 , &B00000000
Data &B00000100 , &B00010000
Data &B00001100 , &B00011000
Data &B00011000 , &B00001100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00010000 , &B00000100
Data &B00011000 , &B00001100
Data &B00001111 , &B11111000
Data &B00000111 , &B11110000
Data &B00000000 , &B00000000
    .
    .
    .
    .
'END OF ENGLISH TEXT LOOKUP----------------------------------------------------------------------------
Start_program3:
Control_lookup_number = 0 : Figure_number = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM---------------------------------------------------------------------------------------
Start_scan3:
Lookup_number = Control_lookup_number
Reset Enable_74154_1 : Set Enable_74154_2
'NOW 74154_1 IS ACTIVE
'LEVEL 1-----------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
```

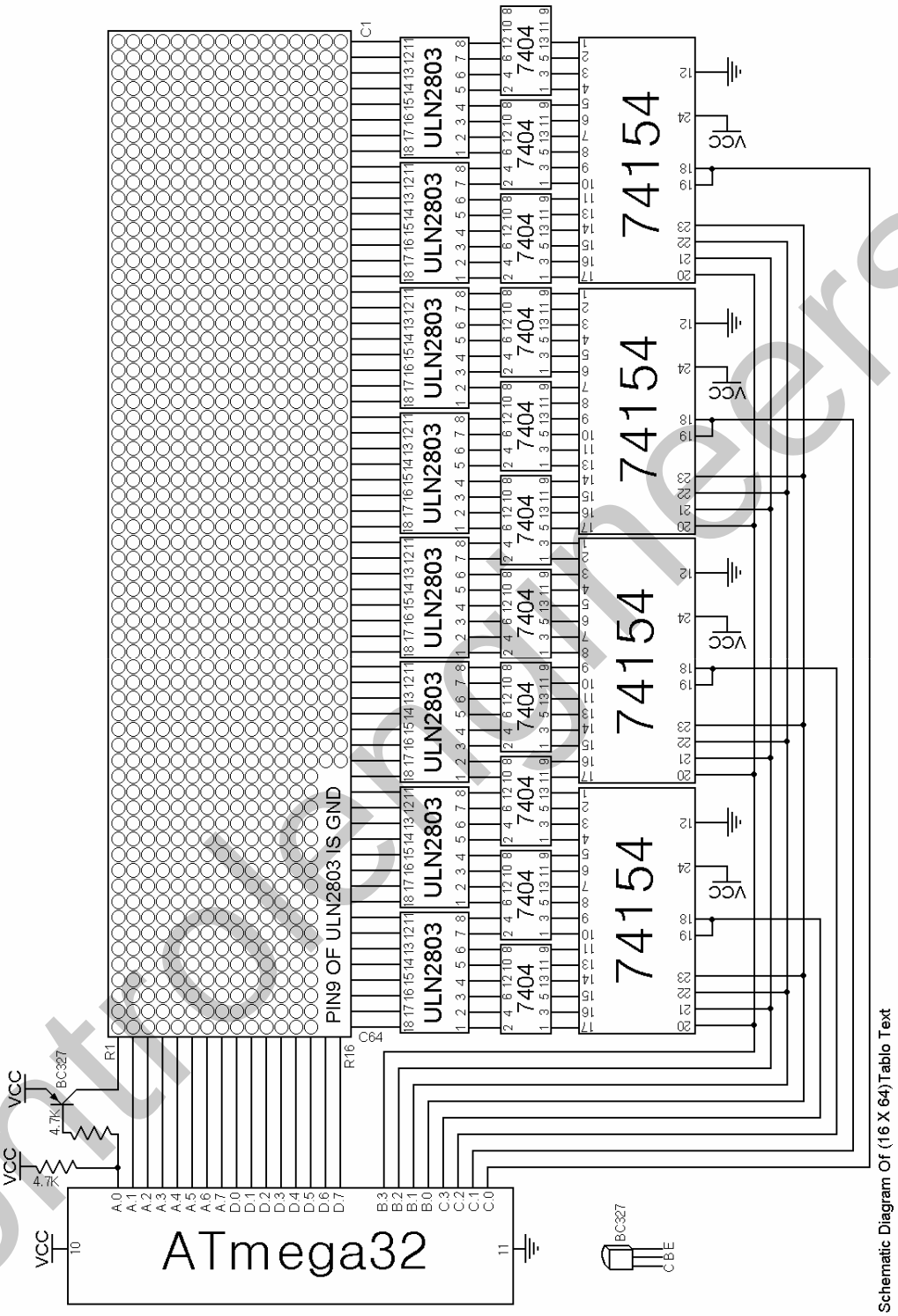
```
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'-----------------------------------------------------------------------------------------------------------
Set Enable_74154_1 : Reset Enable_74154_2
'NOW 74154_2 IS ACTIVE
'LEVEL 1-----------------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Animation)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 200
Porta = 255 : Portd = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 20 Then Goto Start_scan3
Rotate_scan = 0
Control_lookup_number = Control_lookup_number + 64
'64=ONE FIGURE DATAS
```

```
Incr Figure_number
If Figure_number < 5 Then Goto Start_scan3
'5=ANIMATION FIGURES
'-----------------------------------------------------------------------------------------------------------------------
Figure_number = 0
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------------------
'START OF Animation LOOKUP-------------------------------------------------------------------------------------------
Animation:
'FIGURE1
    .
    .
    .
    .
'FIGURE5
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00001111 , &B11110000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00010000 , &B00001000
Data &B00001111 , &B11110000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'END OF Animation LOOKUP-------------------------------------------------------------------------------------------
```

در برنامه فوق دقیقا از جدول های LOOKUP برنامه های قبلی استفاده شده است ، ولی به علت طولانی بودن این جداول تعدادی از کد های آن ها را حذف کرده ایم ، البته برنامه کامل در CD پیوست کتاب ارائه شده است.

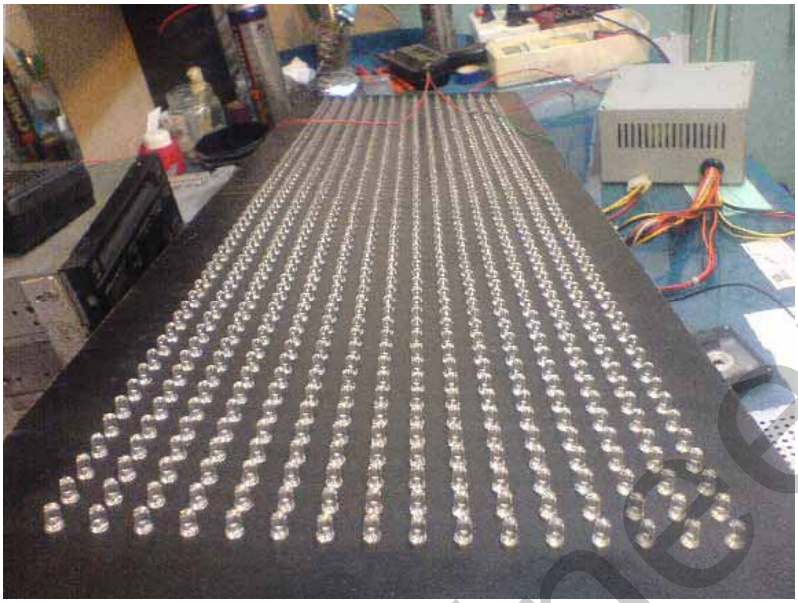
طراحی یک تابلوی 16*64

برای طراحی تابلوی 16*64 می توانید از شماتیک شکل 26-4 استفاده کنید .



شکل 26-4 شماتیک طراحی شده برای تابلوی 64*16

شکل 27-4 نمائی از یک ماتریس LED طراحی شده برای یک تابلوی 64*16 را نشان می دهد .



شکل 27-4 نمونه ای از ماتریس LED طراحی شده برای ماتریس 64*16

شکل زیر نمائی از جعبه طراحی شده برای ماتریس LED فوق را نشان می دهد .

شکل 28-4 نمونه ای از جمعه طراحی شده برای تابلوی 64*16

برنامه اجرای متن فارسی در تابلوی 64*16

برنامه نوشته شده برای اجرای متن فارسی در تابلوی 64*16 به صورت زیر می باشد.

```
'PROGRAM OF 16X64 TABLO TEXT-----------------------------------------------------------------
'COMPILER:1.11.8.7
$regfile = "M32DEF.DAT"
$crystal = 8000000
Dim Column As Word , Scan_column As Byte
Dim Control_lookup_number As Word , Rotate_scan As Byte
Dim Lookup_number As Word , Row_data As Byte , Scan As Byte
Config Porta = Output
Config Portb = Output
Config Portd = Output
Config Pinc.0 = Output : Enable_74154_1 Alias Portc.0
Config Pinc.1 = Output : Enable_74154_2 Alias Portc.1
Config Pinc.2 = Output : Enable_74154_3 Alias Portc.2
Config Pinc.3 = Output : Enable_74154_4 Alias Portc.3
'-----------------------------------------------------------------------------------------------
Start_program:
Control_lookup_number = 0 : Column = 0 : Scan_column = 0
Rotate_scan = 0 : Row_data = 0 : Lookup_number = 0
'START OF PROGRAM-----------------------------------------------------------------------------
Start_scan:
Lookup_number = Control_lookup_number
Reset Enable_74154_1 : Set Enable_74154_2
Set Enable_74154_3 : Set Enable_74154_4
'NOW 74154_1 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 70
Porta = 255 : Portd = 255
'LEVEL 2--------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 70
Porta = 255 : Portd = 255
Next Scan
'-----------------------------------------------------------------------------------------------
Set Enable_74154_1 : Reset Enable_74154_2
Set Enable_74154_3 : Set Enable_74154_4
'NOW 74154_2 IS ACTIVE
'LEVEL 1--------------------------------------------------------------------------------------
Scan_column = 0
```

```
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 70
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 70
Porta = 255 : Portd = 255
Next Scan
'-----------------------------------------------------------------------------------------------------------
Set Enable_74154_1 : Set Enable_74154_2
Reset Enable_74154_3 : Set Enable_74154_4
'NOW 74154_3 IS ACTIVE
'LEVEL 1-----------------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 70
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 70
Porta = 255 : Portd = 255
Next Scan
'-----------------------------------------------------------------------------------------------------------
Set Enable_74154_1 : Set Enable_74154_2
Set Enable_74154_3 : Reset Enable_74154_4
'NOW 74154_4 IS ACTIVE
```

```
'LEVEL 1-----------------------------------------------------------------------------------------------------
Scan_column = 0
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 70
Porta = 255 : Portd = 255
'LEVEL 2-----------------------------------------------------------------------------------------------------
For Scan = 1 To 15 Step 1
Incr Scan_column
Portb = Scan_column
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Portd = Row_data
Incr Lookup_number
Row_data = Lookup(lookup_number , Persian_text)
Row_data = Not Row_data
Porta = Row_data
Incr Lookup_number
Waitus 70
Porta = 255 : Portd = 255
Next Scan
'CONTROL-----------------------------------------------------------------------------------------------------
Incr Rotate_scan
If Rotate_scan < 10 Then Goto Start_scan
Rotate_scan = 0
Control_lookup_number = Control_lookup_number + 2
Incr Column
If Column =< 20 Then Goto Start_scan
'20=LOOKUP ROWS-LED MATRIX CULOMNS
'20=84-64
'-------------------------------------------------------------------------------------------------------------
Column = 0
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------------
'START OF PERSIAN TEXT LOOKUP---------------------------------------------------------------------------
Persian_text:
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B11011111 , &B11100000
Data &B11011111 , &B11110000
Data &B11000000 , &B00110000
Data &B00000000 , &B00110000
Data &B00000000 , &B00110000
Data &B11111111 , &B11110000
Data &B11111111 , &B11100000
```

```
Data &B00000000 , &B00000000
Data &B00111111 , &B11100000
Data &B00111111 , &B11110110
Data &B00000000 , &B00110110
Data &B00000000 , &B00110000
Data &B11111111 , &B11110000
Data &B11111111 , &B11110000
Data &B00000000 , &B00110000
Data &B00000000 , &B00110000
Data &B00111111 , &B11111100
Data &B00111111 , &B11111110
Data &B00110000 , &B00110110
Data &B00110000 , &B00110110
Data &B00110000 , &B01110110
Data &B00011100 , &B11100000
Data &B00001111 , &B11000000
Data &B00000011 , &B10000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00111111 , &B11111100
Data &B00111111 , &B11111110
Data &B00000000 , &B00000110
Data &B00000000 , &B00000110
Data &B00000000 , &B00000000
Data &B00111111 , &B11111100
Data &B00111111 , &B11111110
Data &B00110000 , &B00110110
Data &B00110000 , &B00110110
Data &B00110000 , &B00110110
Data &B00011000 , &B11100110
Data &B00001111 , &B11000000
Data &B00000111 , &B10000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B11111111 , &B11110000
Data &B11111111 , &B11110000
Data &B00000000 , &B00000000
Data &B00111111 , &B11000000
Data &B00111111 , &B11110000
Data &B00000000 , &B00111000
Data &B00000000 , &B00011100
Data &B11000000 , &B00001100
Data &B11000000 , &B00001100
Data &B00000000 , &B00001100
Data &B00000000 , &B00011100
Data &B00000000 , &B00111000
Data &B00111111 , &B11110000
Data &B00111111 , &B11100000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
```

```
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
Data &B00000000 , &B00000000
'END OF PERSIAN TEXT LOOKUP-----------------------------------------------------------------------
```

طراحی تابلو 32*16 با قابلیت وارد کردن متن با کی پد

در این قسمت به آموزش نحوه ساخت یک تابلوی 32*16 با قابلیت وارد کردن متن با استفاده از کی پد (4*4) می پردازیم. تابلو هایی که تا این قسمت آموزش داده شده همگی تابلوهای پیام ثابت بودند یعنی شما یک پیام ثابت را در میکروکنترلر پروگرم می کردید و تابلو فقط قادر به اجرای پیام پروگرم شده بود .

برا ی تغییر پیام اجرائی بایستی آی سی میکروکنترلر را پس از نوشتن برنامه جدید دوباره پروگرم می کردید . ولی عملکرد تابلوئی که در این قسمت قصد بررسی آن را داریم ، کاملا با تابلو های قبلی متفاوت بوده و به صورت زیر می باشد.

ابتدا با قرار دادن کلید دو حالته PROGRAM SW در حالت ENTER TEXT و فشار شاستی RESET BUTTON وارد قسمت برنامه ریزی دستگاه می شویم ، در صورت وارد شدن دستگاه به قسمت برنامه ریزی عبارت زیر بر روی ماتریس LED نوشته می شود .

(کاربر گرامی اکنون شما می توانید متن فارسی مورد نظر خود را وارد کنید .)

در این قسمت شما می توانید با استفاده از کی پد4*4 مانند SMS موبایل متن فارسی مورد نظر خود را وارد کنید ، شکل 29-4 نمائی از کی پد دستگاه را نشان می دهد .

شکل 29-4 نمائی از کی پد طراحی شده برای پروژه

توجه داشته باشید که برای انتخاب شدن حرف نوشته شده بر روی ماتریسLED بایستی دکمه OK را فشار دهید اگر پس از زدن دکمه OK از انتخاب حرف مورد نظر پشیمان شدید می توانید با زدن کلید DEL یک مرحله به عقب برگردید. با استفاده از کلید (شکل) می توانید ، از چندین شکل زیبای طراحی شده نیز در متن مورد

نظرخودتان استفاده کنید پس از وارد کردن متن مورد نظر بایستی کلید ثبت را فشار دهید. با زدن کلید ثبت کدهای مر بوط به متن وارد شده در EEPROM داخلی میکروکنترلر نوشته خواهد شد .

بدین ترتیب شما قادر خواهید بود که حتی پس از RESET شدن و قطع تغذیه مدار نیز از متن وارد شده محافظت کنید .پس از زدن کلید ثبت متن زیر بر روی ماتریس LED نمایش داده می شود .

(کاربر گرامی دستگاه در حال بار گذاری اطلاعات وارد شده می باشد ، لطفا کمی صبر کنید.)

سپس متن وارد شده بر روی ماتریس LED نمایش داده می شود، در این حالت برای این که میکروکنترلر پس از RESET شدن یا قطع و وصل تغذیه دوباره وارد قسمت برنامه ریزی نشود کلید دو حالته PROGRAM SW را در حالت PLAY قرار دهید ، هنگامی که متن وارد شده در حال اجرا باشد ، می توانید سرعت حرکت آن را توسط کلید های سرعت تنظیم کنید . توجه داشته باشید که توسط این پروژه قادر هستید یک متن 1000 حرفی را وارد کنید .

نکته جالب :

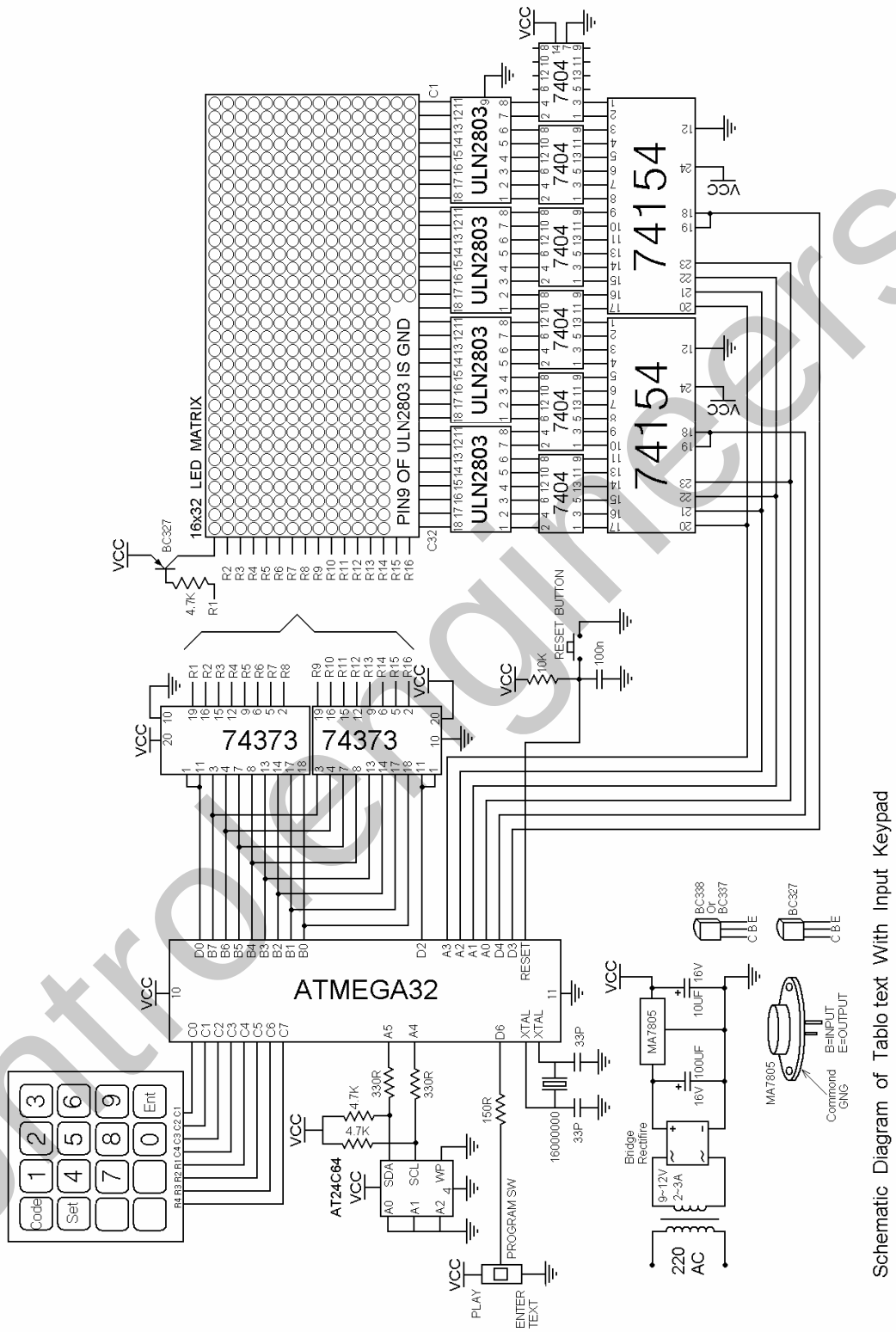
پس از ساخت و راه اندازی دستگاه وارد قسمت برنامه ریزی شده و متن زیر را وارد کنید .

اسم طراحتو بنویس

چه اتفاقی می افتد ؟ نتیجه را بیان کنید .

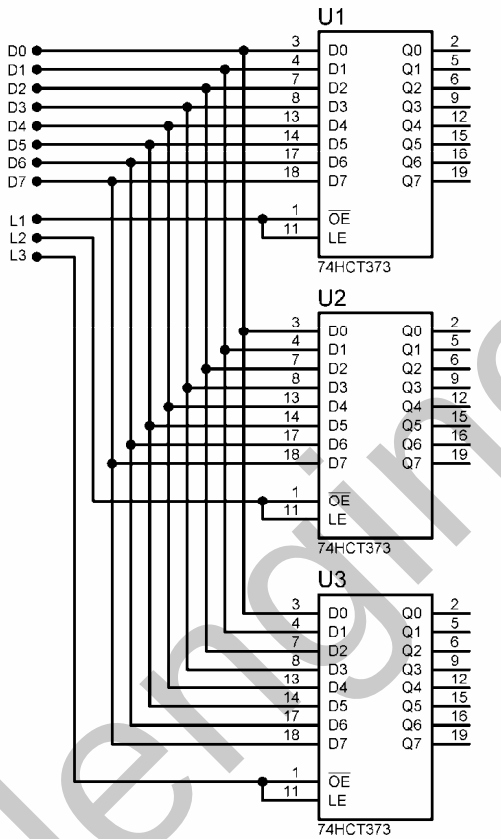
شماتیک طراحی شده برای پروژه در شکل 30-4 ارائه شده است . همان طور که مشاهده می کنید در شماتیک پروژه از IC های قفل کننده (LATCHER) 74373 استفاده شده است . این IC یک قفل کننده 8 بیتی 20 پایه می باشد. این نوع IC ها دارای خروجی 3 وضعیتی هستند ، یعنی هر خروجی می تواند صفر یا یک یا (در حالت امپدانس بالا) باشد ، نحوه عملکرد آی سی 74373 به صورت زیر است.

وقتی که ولتاژ پایه EN بالا باشد،این لچ 8 بیتی به صورت شفاف عمل کرده و با پایین رفتن ولتاژ مزبور داده ها را لچ خواهد شد با بالا بردن ولتاژ پایه یک ، می توان امپدانس خروجی را بسیار بالا برده و آی سی را در وضعیت سوم قرار داد. برای عملکرد معمولی مدار بایستی با اتصال این پایه به ولتاژ پایین آن را غیر فعال کرد.



شکل 30-4 شماتیک طراحی شده برای پروژه تابلوی 32*16 با قابلیت وارد کردن متن

IC های LATCH به منظور افزایش سطر های تابلو مورد استفاده قرار می گیرند به عنوان مثال برای کنترل کردن 24 سطر ماتریس LED با استفاده از 11 پین میکروکنترلر می توانیم از شماتیک شکل 31-4 استفاده کنیم .



شکل 31-4 نحوه کنترل 24 سطر با استفاده از 11 پین میکروکنترلر توسط آی سی های LATCH

فرض کنید می خواهیم 8 سطر اول را یک و 8 سطر دوم را صفر و 8 سطر سوم را هم یک کنیم. ابتدا هر 3 پایه L1 ، L2 ،L3 را در حالت HIGH قرار می دهیم ، سپس عدد 255 را به ورودی اعمال کرده و L1 را به حالت LOW می بریم ، سپس عدد صفر را به ورودی اعمال کرده و L2 را به حالت LOW می بریم ، پس از آن دو باره عدد 255 را به ورودی اعمال نموده و L3 را به حالت LOW می بریم .

نقشه های PCB طراحی شده برای پروژه

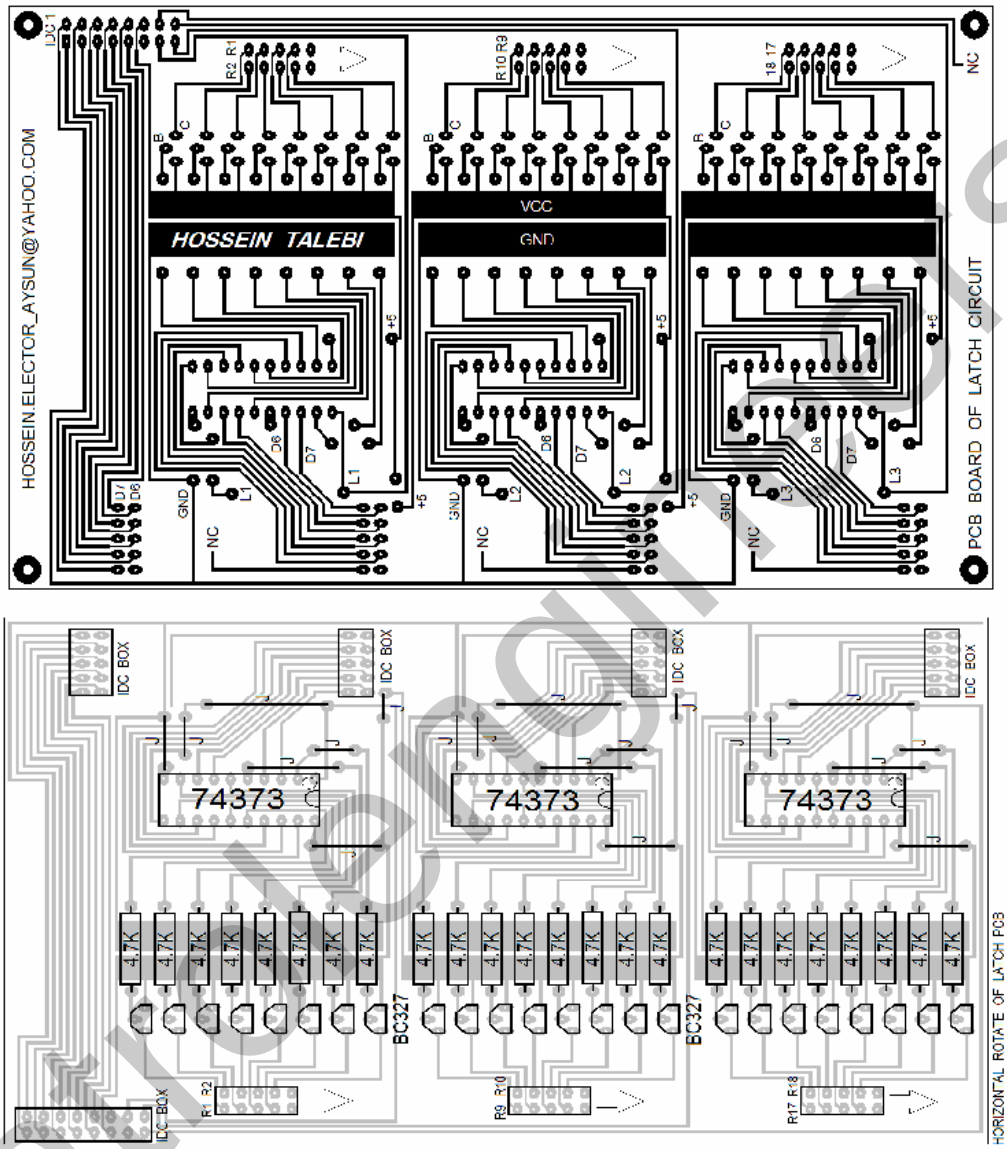
نقشه های PCB مربوط به پروژه در 3 قسمت طراحی شده اند.

۱- برد مربوط به مدار LACH

۲- برد راه انداز ستون ها

۳- برد کنترلر

شکل 4-32 نقشه PCB برد LACH و آرایش قطعات بر روی آن را نشان می دهد .



شکل 4-32 نقشه PCB و آرایش قطعات بر روی برد LATCH

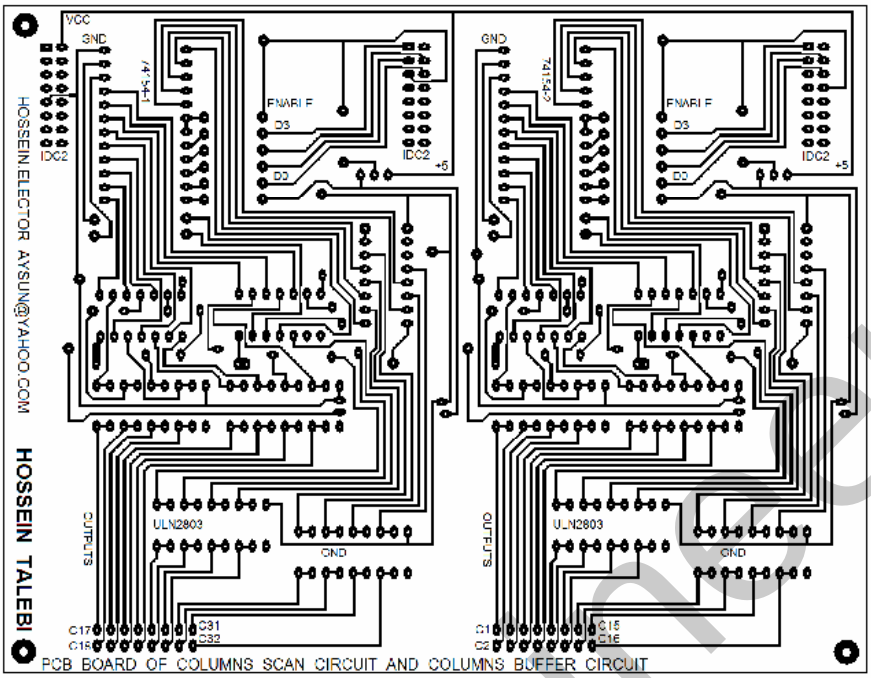
توجه داشته باشید که نقشه ها در اندازه واقعی نیستند برای بدست آورن اندازه واقعی نقشه ها می توانید با پرینت گرفتن از فایل های BMP مربوط به نقشه ها که در CD پیوست کتاب ارائه شده است اندازه واقعی را بدست آورید. برای این منظور ابتدا یک پرینت از نقشه PCB فوق بگیرید و آی سی74373 را در جای خود بر روی نقشه قرار دهید و اگر پایه های IC با فاصله بین پین ها و اندازه آن ها در نقشه PCB مطابقت داشت ، نقشه PCB شما در اندازه واقعی می باشد و گرنه با تغییر اندازه نقشه PCB و پرینت گرفتن از آن می توانید اندازه واقعی

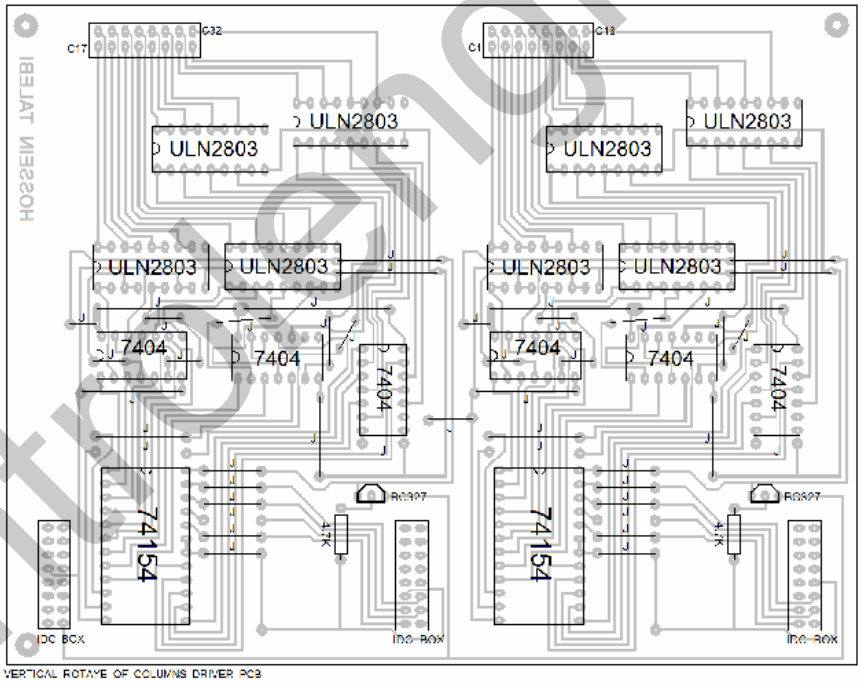
را بدست آورید . توجه داشته باشید که در این پروژه از 2 عدد IC ، 74373 استفاده شده است تصویر برد LACH منتاژ شده در شکل 33-4 ارائه شده است .

شکل 33-4 تصویری از برد LATCH منتاژ شده

در واقع این برد برای تابلوی 24 سطری طراحی شده است ولی شما می توانید از آن برای طراحی یک تابلوی 16 سطری نیز استفاده کنید .

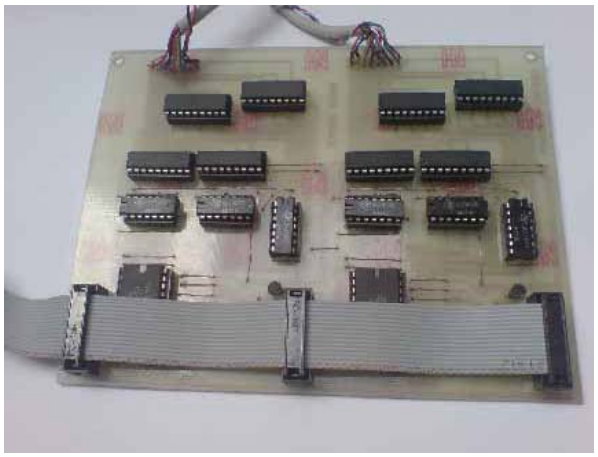
شکل 34-4 نقشه PCB برد درایور ستون ها و نحوه قرار گیری قطعات بر روی آن را نشان می دهد. توجه داشته باشید که در نقشه شماتیک مربوط به پروژه هر ستون فقط یک بار بافر شده است ولی هنگام طراحی نقشه PCB برای افزایش نور LED ها آن ها را دوبار به صورت موازی کرده ایم .





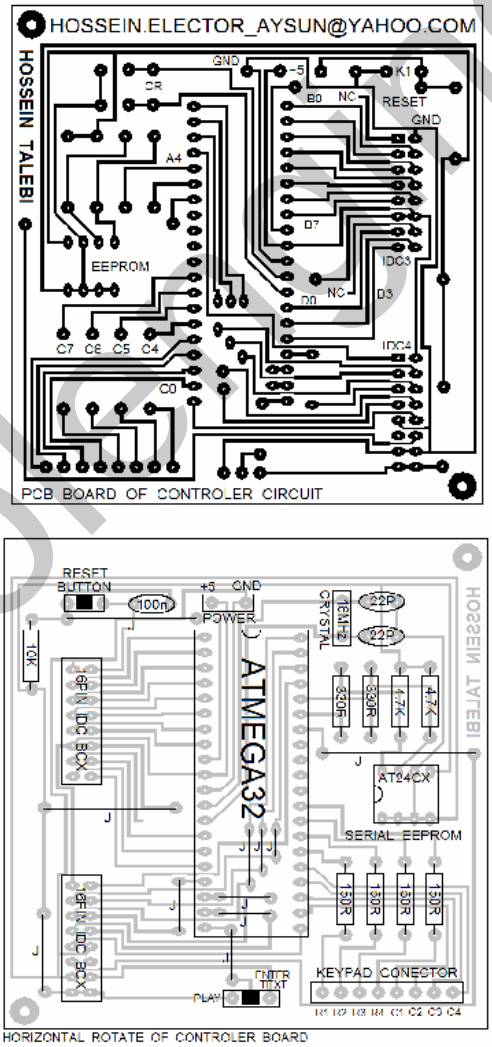
شکل 4-34 نقشه PCB و نحوه قرار گیری قطعات در برد درایور ستون ها

شکل 4-35 تصویری از برد درایور منتاژ شده را نشان می دهد.



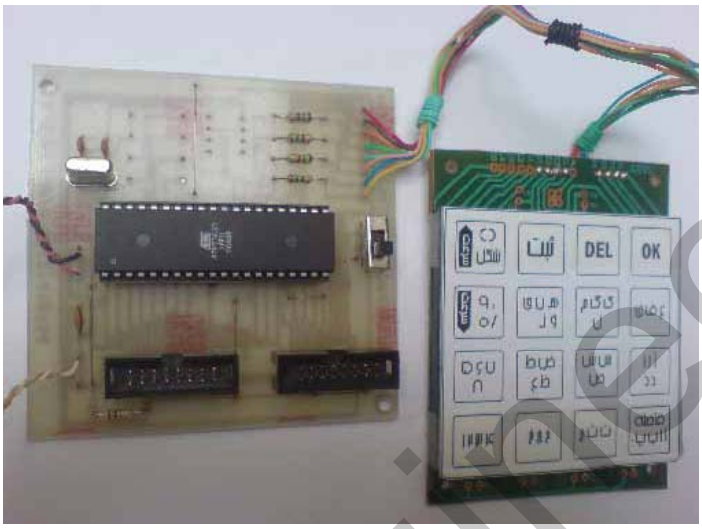
شکل 4-35 نمائی از برد درایور منتاژ شده

شکل 4-36 نقشه PCB مربوط به مدار کنترلر و نحوه قرار گیری قطعات بر روی آن را نشان می دهد.



شکل 4-36 نقشه PCB و نحوه قرار گیری قطعات بر روی آن در برد کنترلر

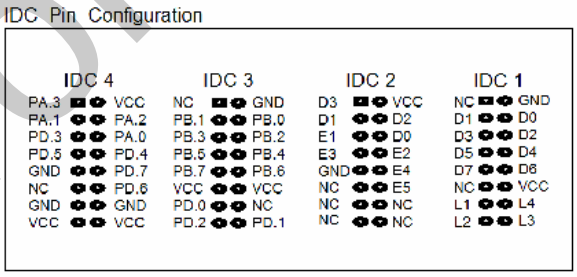
با توجه به این که در این پروژه از EEPROM داخلی میکروکنترلر استفاده شده است می توانید از منتاژ EEPROM سریال خارجی AT24 و مقاومت های مربوط به آن صرفه نظر کنید. در این حالت تصویر برد کنترلر منتاژ شده به صورت شکل 4-37 خواهد بود .



شکل 4-37 تصویری از برد کنترلر منتاژ شده

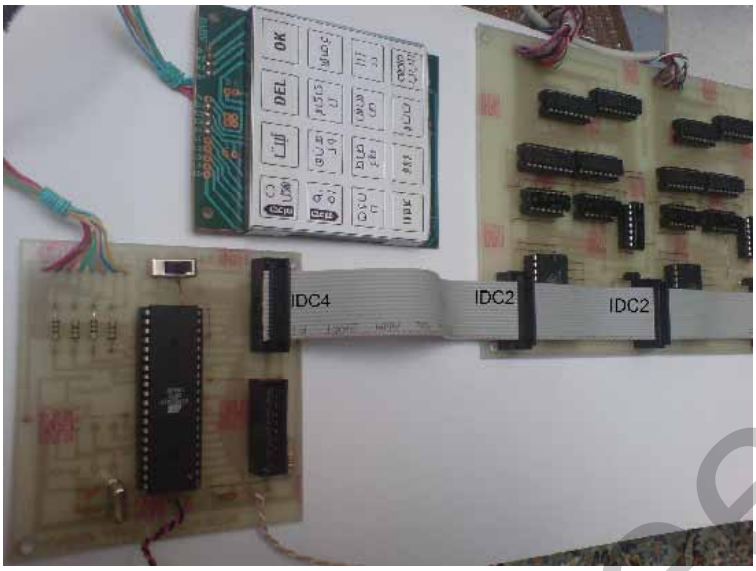
از تصویر طراحی شده برای KEYPAD که در CD پیوست کتاب ارائه شده است در اندازه واقعی پرینت گرفته و آن را مانند شکل فوق بر روی KEYPAD بچسبانید .

شکل 4-38 پیکره بندی پایه های IDC های مربوطه به هر 3 برد را نشان می دهد .



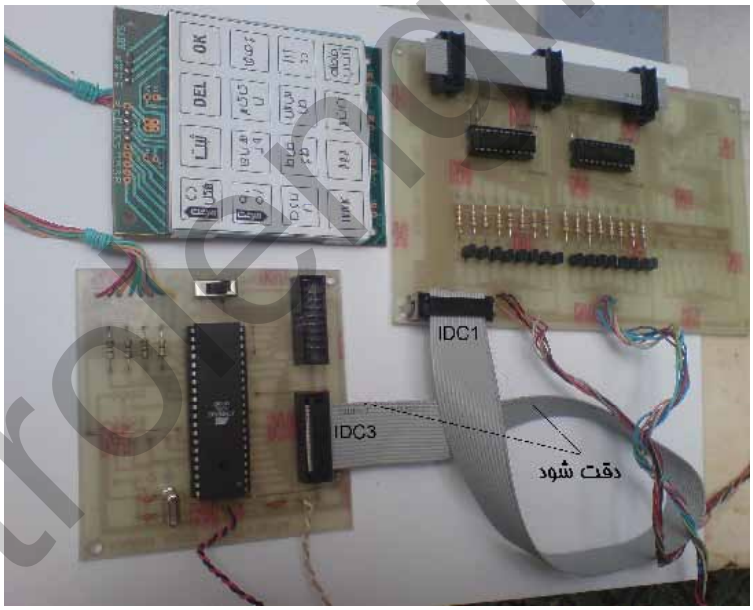
شکل 4-38 PIN CONFIGUARTION ، IDC های مربوط به پروژه

شکل 4-39 نحوه ارتباط دهی برد کنترلر با برد درایو را نشان می دهد .



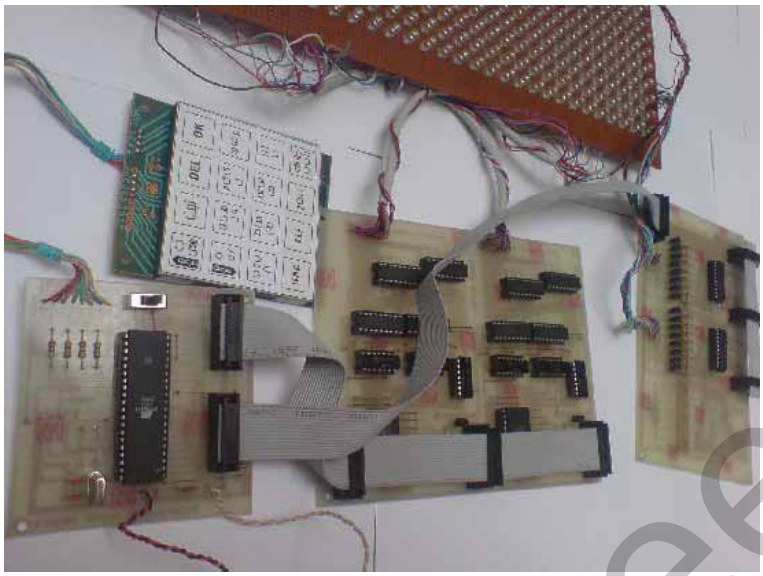
شکل 4-39 نحوه ارتباط دهی برد کنترلر با برد درایور ستون ها

شکل4-40 نحوه ارتباط دهی برد کنترلر با برد LATCH را نشان می دهد.



شکل4-40 نحوه ارتباط دهی برد کنترلر با برد LATCH

شکل 4-41 یک نمای کلی از برد های منتاژ شده پروژه را نشان می دهد.



شکل 4-41 نمائی کلی از برد های منتاژ شده برای پروژه

هنگام منتاژ قطعات حتما به جای IC ها از سوکت استفاده کنید .

پس از منتاژ و ارتباط دهی برد های PCB پروژه ابتدا فایل هگز برنامه نوشته شده برای میکروکنترلر را که در CD پیوست کتاب ارائه شده است توسط پروگرمر LOAD کرده ودر میکروکنترلر ATMEGA32 پروگرم کنید ، توجه داشته باشید که به علت استفاده از کریستال خارجی 16 مگاهرتز فیوزبیت های مربوطه به صورت زیر پروگرم خواهد شد .

CKOPT=0(PROGRAMED)
CKSEL0=0(PROGRAMED)
CKSEL1=1(UNPROGRAMED)
CKSEL2=1(UNPROGRAMED)
CKSEL3=1(UNPROGRAMED)

همچنین به دلیل استفاده از پایه های TAB به عنوان I/O بایستی فیوز بیت JTAG را غیر فعال کنید.

برای مدار تغذیه نیز می توانید منبع تغذیه معرفی شده در شماتیک پروژه را بر روی برد سوراخ دار ببندید. یاد آور می شوم برای IC رگلاتور MA7805 حتما از گرما گیر(هیت سینگ) استفاده کنید .

لیست قطعات پروژه :

نام قطعه (COMPONENT)	تعداد (NUMBER)	شرح (DESCRIPTION)
10PIN IDC & BOX	4	سوکت و BOX
BC327	18	ترانزیستور PNP
4.7K	18	مقاومت ¼ W
150R	4	مقاومت ¼ W
74HC373	2	آی سی قفل کننده (LATCHER)
10WIRE FLAT CABLE	-	کابل فلت 10 رشته ای 50 سانتی متر

16WIRE FLAT CABLE	-	کابل فلت 16 رشته ای 2 متر طول
16PIN IDC & BOX	6	سوکت و BOX
74154	2	انتخابگر 16 مسیری
ULN2803	8	آی سی BUFFER
ATMEGA32	1	آی سی میکروکنترلر
16MHz CRYSTAL	1	کریستال 16MHz
KEYPAD	1	کی پد ماتریسی 4*4
کلید دو حالته	1	-
22P	2	خازن عدسی
100n	3	خازن عدسی
کلید فشاری	1	-
MA7805	1	رگلاتور 5 ولتی با جریان خروجی بالا
ترانسفورماتور	1	ترانسفورماتور 220 به 12 یا 9 ، 2 یا 3 آمپر
RC203	1	پل دیود (BRIDGE RECTIFIRE)
گرماگیر (هیت سینگ)	1	برای رگلاتور ولتاژ
7404	6	آی سی معکوس کننده (INVERTER)
100UF-16V	1	خازن الکترولیتی 16 ولتی
10UF-16V	1	خازن الکترولیتی 16 ولتی
LED (SE)	512	برای ساختن ماتریس LED
برد سوراخ دار	1	برای ساختن ماتریس LED که اندازه آن هم بستگی به فاصله پیکسل ها دارد ، اندازه پیشنهادی (20cm*35cm)
سیم آیفون	-	5 متر طول برای برقراری اتصالات بین ماتریس و PCB
بردسوراخ دار	-	با اندازه کوچک برای ساختن مدار منبع تغذیه
بردهای PCB پروژه		

(زندگی سرشار از گزینه هاست ، سرنوشت شانس و تصادف نیست بلکه موضوع انتخاب است ، سرگذشت چیزی نیست که منتظر آن بمانیم بلکه چیزی است که باید آن را بدست بیاوریم) ویلیام جینگز برایان

نیروی لایزال الهی که در وجود شماست زمانی به کار می افتد و شما را به کار وتلاش وا می دارد که سرانجام در میابید اعتقاد داشتن به بخت و قسمت بد ترین نوع بردگی است و سرنوشت در دست خود شماست و شما در قبال زندگی خویش مسئولید در این هنگام است که زندگی تان دگرگون می شود و کم کم چیز هایی را که آرزو دارید به چنگ می آورید در واقع عمر شما از زمانی آغاز می شود که اختیار سرنوشت خود را بدست می گیرید پس آن زمان را به خاطر بسپارید و از آن به عنوان تولدی دوباره یاد کنید .

فصل پنجم

AVR در ارتباطات WIRELESS

Wireless چیست ؟

Wireless تکنولوژی است که در آن انتقال سیگنالهای اطلاعاتی میان دو دستگاه از طریق امواج رادیوئی ، امواج مادون قرمز، امواج مایکروویو و غیره بدون استفاده از کابل و سیم صورت می گیرد ، در حالت کلی این سیستم ها به سه دسته تقسیم بندی می شوند .

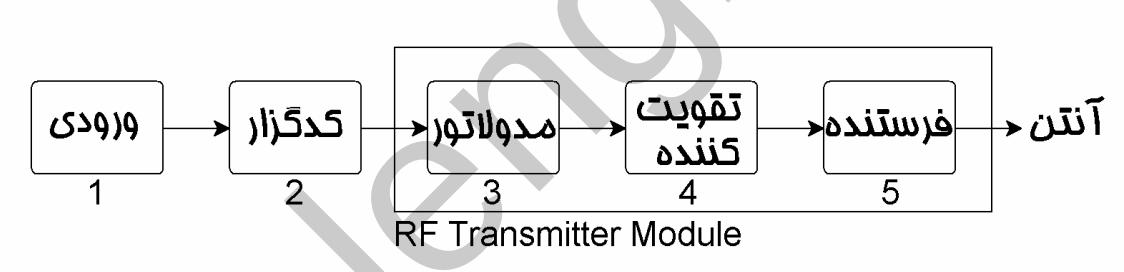
۱- سیستم های wireless ثابت که در آن امواج رادیوئی با خط دید مستقیم برای برقراری ارتباط استفاده می شوند. از نمونه های برقراری این ارتباط می توان به اتصال پر سرعت به اینترنت اشاره کرد.

۲- سیستم های wireless قابل حمل که در آن از امواج رادیوئی و مایکروویو برای انتقال سیگنال های اطلاعاتی استفاده می شود و نمونه های آن در لپ تاپ ها و گوشی های موبایل دیده می شود .

۳- سیستم های wireless مادون قرمز که بهترین نمونه آن دستگاههای ریموت (کنترل از راه دور) تلویزیونهاست و از اموج رادیوئی مادون قرمز در آن استفاده می شود. نمونه دیگری از این سیستم با نام infrared در گوشی های موبایل استفاده می شود.

تشریح یک ریموت رادیوئی

به طور کلی بلوک دیاگرام سیستم های ریموت (فرستنده) به صورت زیر است .

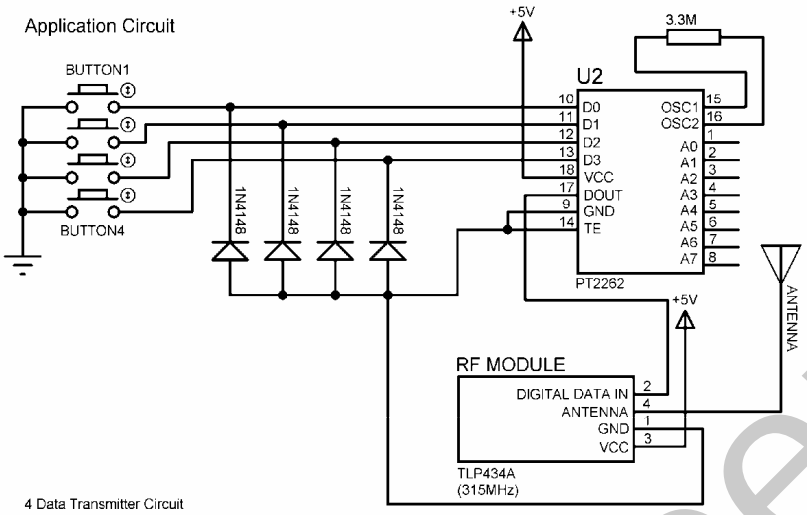


شکل1-5 بلوک دیاگرام کلی سیستم های فرستنده ریموت

بلوک شماره1 : شامل کلید های ورودی ریموت می باشد که بسته به نیاز طراح و همچنین تعداد کانال های آی سی ENCODER تعداد شان متفاوت است به طور معمول از میکرو سوئیچ های مخصوص ریموت به عنوان کلید ورودی استفاده می شود .

بلوک شماره2 : بلوک شماره 2 شامل آی سی انکدر یا رمزگذار می باشد این آی سی اطلاعات ورودی را تبدیل به اطلاعاتی می کند که توسط بلوک های بعدی قابل شناسائی و ارسال می باشد در پروژه های این کتاب از آی سی PT2262 به عنوان ENCODER استفاده شده است .

شکل 1-5 نحوه طراحی یک ریموت 4 کاناله را با استفاده از انکدر PT2262 و ماژول 315 مگاهرتز TLP434A نشان می دهد .



شکل 5-1 ریموت کنترل RF ، 4 کاناله

طراحی مدار فوق به گونه ای است که تا یکی از کلید های BUTTON1 تا BUTTON4 فشرده نشود جریان مدار توسط باطری برقرار نشده و مدار کار نمی کند به عبارت دیگر در حالت بی کاری ترس از خالی شدن باطری وجود ندارد و به علت مصرف بسیار کم فرستنده باطری آن می تواند برای مدت بسیار طولانی دوام بیاورد .

همچنین در مدار فوق با فشار هر کدام از کلید های ورودی کد پالسی متفاوتی در خروجی PT2262 روی پایه شماره 17 آن ایجاد می شود. که این کدها پس از دریافت توسط گیرنده قابل شناسائی و DECODE کردن خواهد بود.

پایه های 1 تا8 انکدر PT2262 مربوط به کدبندی دستگاه می باشد هریک از این پایه ها می توانند 3 وضعیت اتصال باز ، GND و VCC را داشته باشند. از پایه های کدبندی برای حذف ایجاد تداخل بین فرستنده گیرنده های مختلف استفاده می شود بدین صورت که نحوه کدبندی آی سی ENCODER در فرستنده و آی سی DECODER در گیرنده بایستی با هم برابر باشند ، در غیر این صورت تبادل اطلاعات غیر ممکن خواهد بود . با توجه به این که هر پایه کدبندی می تواند 3 حالت متفاوت داشته باشد. می توان 8^3=512 فرستنده گیرنده متفاوت ساخت که هر گیرنده ای فقط با فرستنده هم کد خودش کار کند.

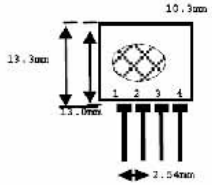
بلوک های 5,4,3 :

با توجه به این که در پروژه های این کتاب از ماژول های فرستنده RF (315MHz) استفاده شده است. بلوک های مدولاتور ، تقویت کننده و فرستنده در داخل ماژول فرستنده قرار گرفته اند بلوک مدولاتور وظیفه انجام مدولاسیون را بر عهده دارد و کد پالسی خروجی از آی سی ENCODER را روی فرکانس کاری ماژول سوار

می کند که پالس مدوله شده توسط بلوک تقویت کننده ، تقویت شده و توسط بلوک فرستنده و آنتن به امواج الکترو مغناطیسی تبدیل شده و در فضا منتشر می شود .

مشخصات ماژول های فرستنده استفاده شده در پروژه های این کتاب در ادامه ارائه شده است .

TLP434A Ultra Small Transmitter



pin 1 : GND
pin 2 : Data In
pin 3 : Vcc
pin 4 : Antenna (RF output)

Frequency 315, 418 and 433.92 Mhz
Modulation : ASK
Operation Voltage : 2 - 12 VDC

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
VCC	OPERATING SUPPLY VOLTAGE		2.0	-	12.0	V
ICC1	PEAK CURRENT(2V)		-	-	1.64	mA
ICC2	PEAK CURRENT(12V)		-	-	19.4	mA
VH	INPUT HIGH VOLTAGE	IDATA=100UA(HIGH)	VCC-0.5	VCC	VCC+0.5	V
V1	INPUT LOW VOLTAGE	IDATA=0 UA (LOW)	-	-	0.3	V
FO	ABSOLUTE FREQUENCY	315MHz MODULE	314.8	315	315.2	MHz
PO	RF OUTPUT POWER – 50 OHM	VCC=9V-12V	-	16	-	dBm
		VCC=5V-6V	-	14	-	dBm
DR	DATA RATE	EXTERNAL ENCODING	512	4.8K	200K	bps

Notes:(case temperature = 25c+-2c,test load impedance =50 ohm)

ویژگی های خاص این ماژولها عبارتند از :

۱- اندازه کوچک ، حساسیت بالا ، توان مصرفی کم و توان ارسالی بالا

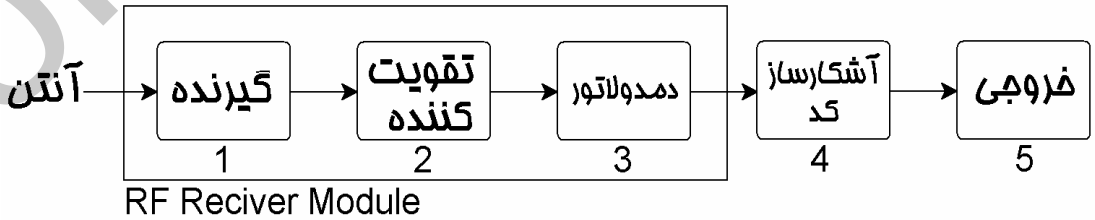
۲- مصونیت بالا در برابر نویز

۳- برد بالا حدود 300 متر در ناحیه بازو و در ناحیه بسته وابسته به محیط

۴- عدم نیاز به تنظیمات خاص جهت استفاده

۵- محدوده وسیع دمائی از 35- تا 80 درجه سانتی گراد

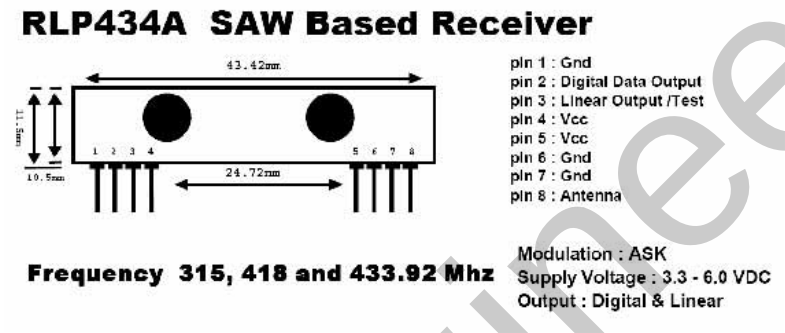
بلوک دیاگرام کلی سیستم های گیرنده ریموت در شکل 3-5 نشان داده شده است .



شکل3-5 بلوک دیاگرام کلی سیستم های گیرنده ریموت

بلوک های شماره 3,2,1 :

بلوک گیرنده امواج الکترومغناطیسی ارسال شده با استفاده از فرستنده را توسط آنتن دریافت و به امواج الکترونیکی تبدیل می کند. سپس این امواج توسط بلوک بعدی تقویت شده و سیگنال تقویت شده به دمدولاتور اعمال می شود دمدولاتور کد پالسی را از سیگنال حامل جدا کرده و به خروجی ماژول گیرنده اعمال می کند لازم به توضیح است که هر 3 بلوک معرفی شده در داخل ماژول گیرنده RLP434A قرار گرفته اند. مشخصات ماژول های گیرنده استفاده شده در این کتاب به همراه شکل ظاهری و ترتیب پایه ها در ادامه ارائه شده است .



SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	
VCC	OPERATING SUPPLY VOLTAGE		3.3	5.0V	6.0	V
ITOT	OPERATING CURRENT		-	4.5		mA
VDATA	DATA OUT	IDATA=+200UA(HIGH)	VCC-0.5	-	VCC	V
		IDATA=-10UA(LOW)	-	-	0.3	V

ELECTRICAL CHARACTERISTICS					
CHARACTERISTICS	SYM	MIN	TYP	MAX	UNIT
OPERATION RADIO FREQUENCY	FC	315,418 AND 433.92			MHz
SENSITIVITY	PREF		-110		dBm
CHANNEL WIDTH			+ -500		KHz
NOISE EQUIVALENT BW			4		KHz
RECEVER TURN ON TIME			5		mS
OPERATION TEMPERATURE	TOP	-20	-	80	C
BASEBOARD DATA RATE			4.8		KHz

بلوک شماره 4 :

بلوک DECODER یا آشکار ساز رمز وظیفه آشکار سازی یا DECODE کردن کد پالسی ایجاد شده توسط آی سی ENCODER در فرستنده را بر عهده دارد ، آی سی استفاده شده در پروژه های این کتاب PT2272-M4 نام دارد در صورتی کد بندی این IC مشابه کد بندی آی سی ENCODER در فرستنده باشد کد ارسالی توسط 4 کلید فرستنده را به طور مجزا در 4 پایه 13,12,11,10 خود آشکار خواهد کرد.

بلوک شماره 5 :

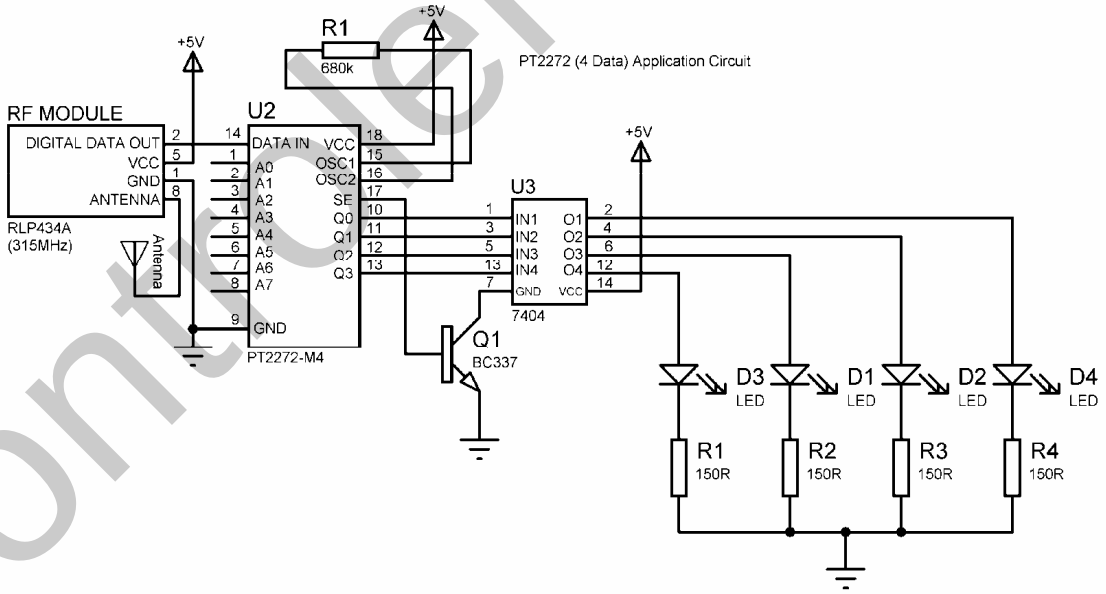
بلوک خروجی شامل DEVICE های خروجی مانند رله ، LED ، و غیره می باشد . شکل 5-4 نحوه طراحی یک گیرنده ریموت 4 کاناله را با استفاده از PT2272-M4 و ماژول 315 مگاهرتز RLP434A نشان می دهد .

با توجه به شماتیک 5-4 به دلیل این که در مدار فرستنده از پالس منفی برای فعال کردن مدار استفاده کرده ایم خروجی مدار گیرنده نیز به صورت ACTIVE LOW خواهد بود برای همین هم از یک IC معکوس کننده در خروجی استفاده شده است ، هر زمان که IC ، PT2272-M4 سیگنالی دریافت کند علاوه بر تغییر وضعیت خروجی ها ، پایه 17 آی سی نیز در حالت HIGH قرار می گیرد ، از این پایه برای سوئیچ کردن تغذیه آی سی معکوس کننده 7404 استفاده شده است .

نکته در رابطه با انواع مختلف آی سی های PT2272 :

آی سی PT2272-M4 دارای 4 خروجی لحظه ای می باشد یعنی خروجی تا زمانی فعال خواهد بود که انگشت شما بر روی کلید مربوط در فرستنده قرار دارد و با غیر فعال شدن کلید خروجی مربوطه در گیرنده نیز غیر فعال خواهد شد.

آی سی PT2272-L4 دارای 4 خروجی LATCH می باشد یعنی با فشار هر کدام از کلید ها در فرستنده خروجی مر بوطه در گیرنده به حالت فعال رفته و با فشار آن برای بار دوم خروجی غیر فعال می شود.

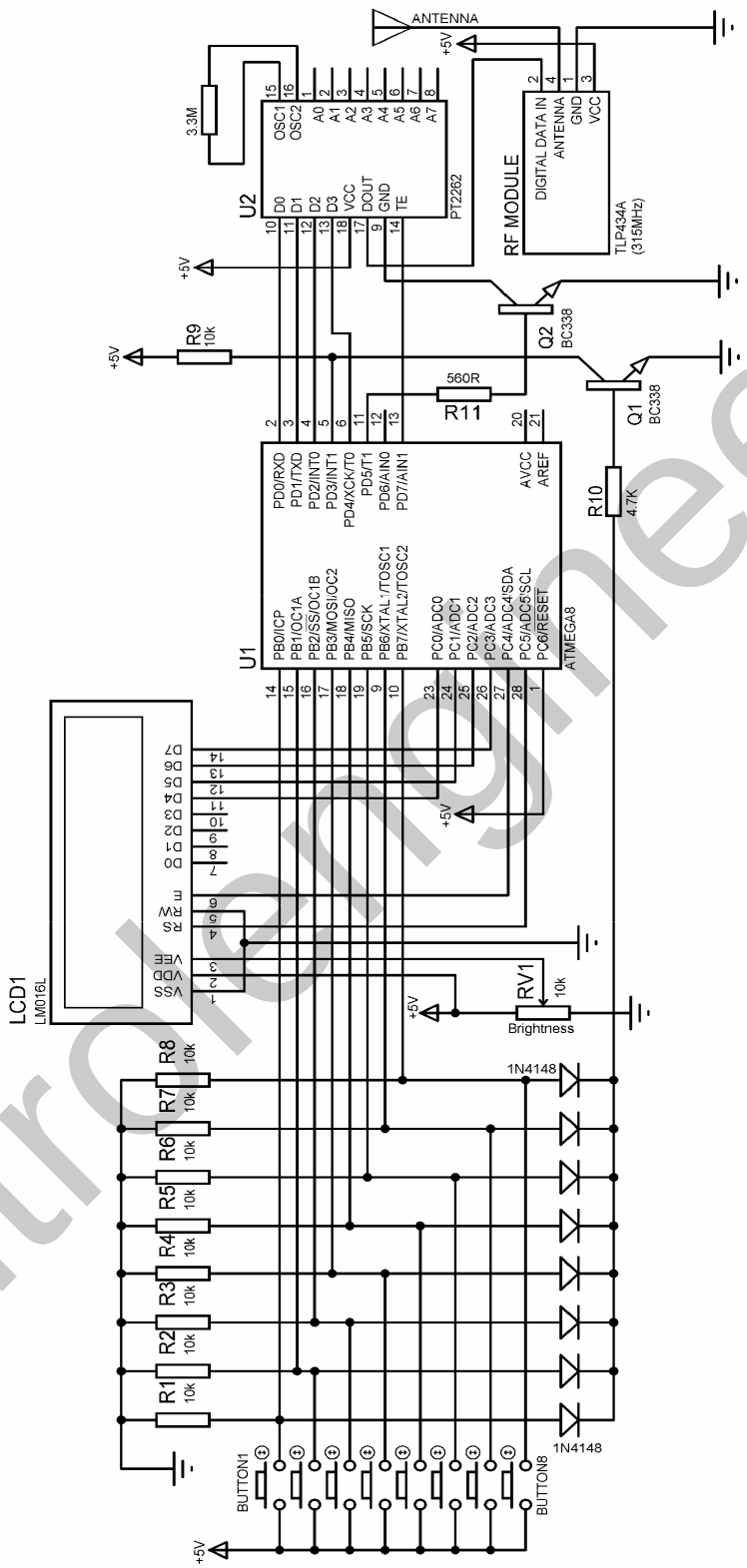


شکل 5-4 شماتیک گیرنده ریموت RF ، 4 کاناله

طراحی ریموت کنترل RF ، 8 کاناله میکروکنترلری

با توجه به این که دراین کتاب از IC های ENCODER,DECODER ، 4 کاناله استفاده شده و همچنین می توان با داشتن 4 بیت داده ، 16 حالت مختلف باینری ایجاد کرد . می توانیم کانالهای ریموت کنترل را تا 16 عدد افزایش دهیم و به جای ارسال یک بیت، یک داده باینری 4 بیتی را ارسال کنیم. در این پروژه از میکروکنترلر ATMEGA8 در مدار فرستنده استفاده شده است ، مدار به گونه ای طراحی شده است که ابتدا بر روی LCD عبارت RF REMOTE نوشته شده و میکروکنترلر وارد مد اسلیپ POWERDOWN می شود سپس با فشار هر کدام از کلید هایBUTTON1 تاBUTTON8 یک پالس تریگر نیز توسط دیود های 1N4148 و ترانزیستور Q1 به پایه ورودی INT1 اعمال شده و میکروکنترلر را از حالت SLEEP خارج می کند. پس از خارج شدن میکروکنترلر از حالت SLEEP بلافاصله در ISR وقفه INT1 کلید ورودی خوانده شده و نام آن بر روی LCD نوشته می شود سپس عدد باینری متناسب با آن به ورودی آی سی PT2262 اعمال می شود سخت افزار مدار فرستنده در شکل 5-5 نشان داده شده است.

همانطور که در شماتیک پروژه مشاهده می شود از ترانزیستور Q2 برای سوئیچ کردن تغذیه PT2262 استفاده شده است به طوری که هنگام استفاده نکردن از آی سی PT2262 و یا زمانی که میکروکنترلر در حالت SLEEP، POWERDOWN قرار دارد تغذیه آی سی مزبور قطع شده وموجب کاهش توان مصرفی سیستم می شود ، شما می توانید این کار را برای ماژول فرستنده TLP434A نیز انجام دهید .



شکل 5-5 شماتیک فرستنده ریموت 8 کاناله

برنامه نوشته شده برای میکروکنترلر مدار فرستنده به صورت زیر است .

```
'COMPILER:BASCOM 1.11.8.7
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Input
Config Pind.0 = Output
Config Pind.1 = Output
Config Pind.2 = Output
Config Pind.4 = Output
Config Pind.5 = Output : Pt_power Alias Portd.5
Config Pind.7 = Output : Pt_enable Alias Portd.7
Config Lcdpin = Pin , Db4 = Pinc.0 , Db5 = Pinc.1 , Db6 = Pinc.2 , Db7 = Pinc.3_
      , E = Pinc.4 , Rs = Pinc.5
Config Lcd = 16 * 2
Dim Input_buttons As Byte , Send_data As Byte
Declare Sub Send_to_pt
Enable Interrupts
Enable Int1
On Int1 Exit_of_sleep
Config Int1 = Low Level
'-----------------------------------------------------------------------------------------------------------------
Reset Pt_power
Set Pt_enable
Cls : Home : Cursor Off
Lcd "RF REMOTE"
'GOTO SLEEP-------------------------------
Do
Powerdown
Loop
'-----------------------------------------------------------------------------------------------------------------
Exit_of_sleep:
Disable Interrupts
Input_buttons = Pinb
'-----------------------------------------
Select Case Input_buttons:
Case Is = &B00000001
Locate 2 , 1
Lcd "INPUT IS BUTTON0"
Send_data = 1
Call Send_to_pt
'-----------------------------------------
Case Is = &B00000010
Locate 2 , 1
Lcd "INPUT IS BUTTON1"
Send_data = 2
Call Send_to_pt
'-----------------------------------------
Case Is = &B00000100
Locate 2 , 1
Lcd "INPUT IS BUTTON2"
Send_data = 3
Call Send_to_pt
'-----------------------------------------
Case Is = &B00001000
Locate 2 , 1
Lcd "INPUT IS BUTTON3"
Send_data = 4
Call Send_to_pt
'-----------------------------------------
Case Is = &B00010000
Locate 2 , 1
```

```
Lcd "INPUT IS BUTTON4"
Send_data = 5
Call Send_to_pt
'-----------------------------------------
Case Is = &B00100000
Locate 2 , 1
Lcd "INPUT IS BUTTON5"
Send_data = 6
Call Send_to_pt
'-----------------------------------------
Case Is = &B01000000
Locate 2 , 1
Lcd "INPUT IS BUTTON6"
Send_data = 7
Call Send_to_pt
'-----------------------------------------
Case Is = &B10000000
Locate 2 , 1
Lcd "INPUT IS BUTTON7"
Send_data = 8
Call Send_to_pt
'-----------------------------------------
End Select
Enable Interrupts
Return
'-----------------------------------------------------------------------------------------
Sub Send_to_pt:
Select Case Send_data:
Case Is = 1
Set Portd.0 : Reset Portd.1 : Reset Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 2
Reset Portd.0 : Set Portd.1 : Reset Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 3
Set Portd.0 : Set Portd.1 : Reset Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 4
Reset Portd.0 : Reset Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 5
Set Portd.0 : Reset Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 6
Reset Portd.0 : Set Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 7
Set Portd.0 : Set Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 8
Reset Portd.0 : Reset Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
End Select
Set Pt_power
Reset Pt_enable
Waitms 500
Reset Pt_power
Set Pt_enable
Cls : Home : Lcd "RF REMOTE"
Return
End Sub Send_to_pt
'-------------------------------------------------------------------------------------------------------------------
```

تشریح نحوه عملکرد برنامه :

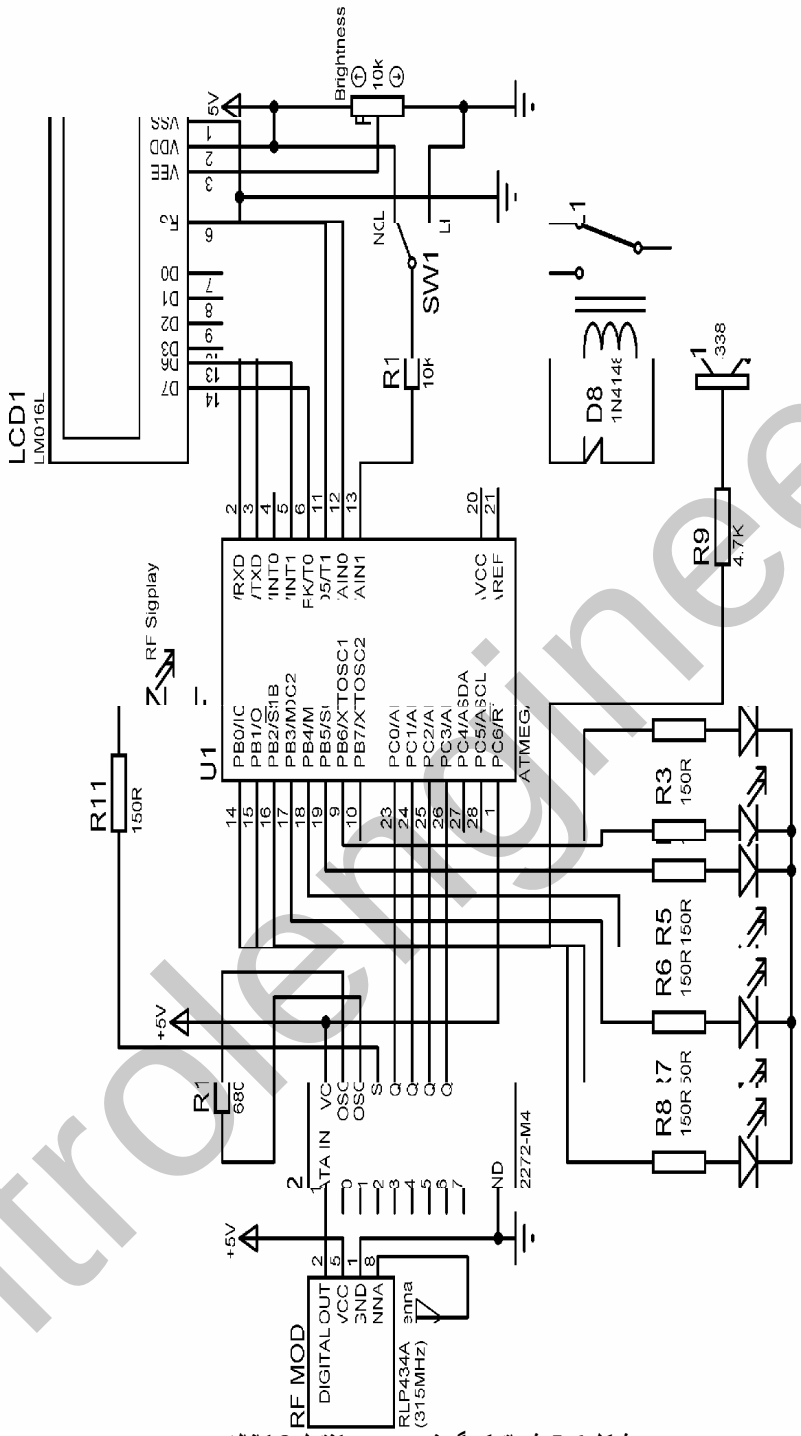
در برنامه فوق وقفه خارجی INT1 حساس به سطح پایین (صفر منطقی) پیکره بندی شده است و ISR مربوط به آن EXIT_OF_SLEEP نام دارد . که در ISR ، EXIT_OF_SLEEP ابتدا کلید ورودی خوانده شده و عدد متناسب با آن در متغیر SEND_DATA قرار می گیرد ، سپس زیر برنامهSEND_TO_PT فراخوانی می شود . در این زیر برنامه مقدار باینری عدد قرار گرفته در متغیر SEND_DATA به ورودی PT2262 اعمال شده سپس با استفاده از دستور SET PT_POWER تغذیه آی سی مزبور برقرارمی شود ، پس از آن با اجرای دستور RESET PT_ENABLE یک سیگنال با سطح منطقی صفر به پایه ENABLE انکدر اعمال شده و آن را فعال سازی می کند. لازم به توضیح است که آی سی انکدر PT2262 زمانی فعال می شود که علاوه بر برقراری تغذیه آن پایه ENABLE آن نیز زمین شده باشد.

پس از فعال سازی آی سی انکدر کد پالسی عدد باینری اعمال شده به ورودی آن در خروجی IC ظاهر شده و به ورودی ماژول TLP434A اعمال می شود. پس از 500 میلی ثانیه تاخیر برای انجام عمل ارسال DATA توسط انکدر PT2262 و ماژول TLP434A ، آی سی انکدر دوباره توسط دستورات SET PT_ENABLE , RESET PT_POWERغیر فعال می شود . سپس توسط دستور RETURN اجرای برنامه به ISR EXIT_OF_SLEEP، برگشته و از آنجا نیز به حلقه DO_LOOP برگشت داده می شود که به این ترتیب دوباره میکروکنترلر وارد حالت SLEEP ، POWERDOWN خواهد شد. توجه داشته باشید که اعمال عدد باینری صفر به ورودی آی سی انکدر PT2262 به منزله غیر فعال کردن هر چهار ورودی آن می باشد و در این حالت هیچ کدی در خروجی IC ظاهر نخواهد شد پس شما نمی توانید توسط آی سی فوق عدد صفر را بفرستید.

شماتیک گیرنده طراحی شده برای ریموت کنترل RF ، 8 کاناله در شکل 5-6 ارائه شده است .

با توجه به شکل 5-6 مشاهده می شود که از 8 پایه پورت B به عنوان خروجی استفاده شده است ، شما می توانید هر یک از خروجی ها را برای سوئیچ کردن یک رله 5 ولتی یا 12 ولتی و یا برای روشن کردن یک LED مورد استفاده قرار دهید ، نحوه عملکرد مدار بدین صورت می باشد که ابتدا عبارت RF REMOTE بر روی LCD نوشته شده و یا دریافت ورودی از PT2272_M4 خروجی مربوطه فعال می شود . وضعیت خروجی ها با استفاده از کلید SW1 به صورت لحظه ای یا LATCH قابل تنظیم است ، LED RF SIGNAL DISPLAY نیز هنگام دریافت اطلاعات توسط مدار گیرنده روشن می شود با استفاده از این LED شما می توانید از دریافت سیگنال RF توسط مدار گیرنده مطمئن شوید.

لازم به توضیح است که کدبندی پایه های A0 تاA7 در گیرنده و فرستنده بایستی برابر باشد و گرنه تبادل اطلاعات غیر ممکن خواهد بود.



شکل 5-6 شماتیک گیرنده ریموت کنترل 8 کاناله

برنامه نوشته شده برای میکروکنترلر گیرنده RF ، 8 کاناله به صورت زیر است.

```
'COMPILER:BASCOM 1.11.8.7
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Output
```

```
Config Pinc.0 = Input
Config Pinc.1 = Input
Config Pinc.2 = Input
Config Pinc.3 = Input
Config Pind.7 = Input
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.3 , Db7 = Pind.4_
    , E = Pind.5 , Rs = Pind.6
Config Lcd = 16 * 2
Dim Recive_data As Byte
Declare Sub Recive_of_pt
'-----------------------------------------------------------------------------------------------------------------
Cls : Home : Cursor Off
Lcd "RF RECIVER"
'-----------------------------------------------------------
Start_program:
If Pind.7 = 0 Then Goto Start_program2
Call Recive_of_pt
'-----------------------------------------------------------
Select Case Recive_data:
Case Is = 1
Locate 2 , 1
Lcd "INPUT IS CHANEL0"
Portb = &B00000001
'-----------------------------------------------------------
Case Is = 2
Locate 2 , 1
Lcd "INPUT IS CHANEL1"
Portb = &B00000010
'-----------------------------------------------------------
Case Is = 3
Locate 2 , 1
Lcd "INPUT IS CHANEL2"
Portb = &B00000100
'-----------------------------------------------------------
Case Is = 4
Locate 2 , 1
Lcd "INPUT IS CHANEL3"
Portb = &B00001000
'-----------------------------------------------------------
Case Is = 5
Locate 2 , 1
Lcd "INPUT IS CHANEL4"
Portb = &B00010000
'-----------------------------------------------------------
Case Is = 6
Locate 2 , 1
Lcd "INPUT IS CHANEL5"
Portb = &B00100000
'-----------------------------------------------------------
Case Is = 7
Locate 2 , 1
Lcd "INPUT IS CHANEL6"
Portb = &B01000000
'-----------------------------------------------------------
Case Is = 8
Locate 2 , 1
Lcd "INPUT IS CHANEL7"
Portb = &B10000000
'-----------------------------------------------------------
End Select
Waitms 300
Portb = 0
```

```
Cls : Home
Lcd "RF RECIVER"
Goto Start_program
'-----------------------------------------------------------------------------------------------
Start_program2:
If Pind.7 = 1 Then
Portb = 0
Goto Start_program
End If
'-----------------------------------------------------------
Call Recive_of_pt
'-----------------------------------------------------------
Select Case Recive_data:
Case Is = 1
Locate 2 , 1
Lcd "INPUT IS CHANEL0"
Toggle Portb.0
'-----------------------------------------------------------
Case Is = 2
Locate 2 , 1
Lcd "INPUT IS CHANEL1"
Toggle Portb.1
'-----------------------------------------------------------
Case Is = 3
Locate 2 , 1
Lcd "INPUT IS CHANEL2"
Toggle Portb.2
'-----------------------------------------------------------
Case Is = 4
Locate 2 , 1
Lcd "INPUT IS CHANEL3"
Toggle Portb.3
'-----------------------------------------------------------
Case Is = 5
Locate 2 , 1
Lcd "INPUT IS CHANEL4"
Toggle Portb.4
'-----------------------------------------------------------
Case Is = 6
Locate 2 , 1
Lcd "INPUT IS CHANEL5"
Toggle Portb.5
'-----------------------------------------------------------
Case Is = 7
Locate 2 , 1
Lcd "INPUT IS CHANEL6"
Toggle Portb.6
'-----------------------------------------------------------
Case Is = 8
Locate 2 , 1
Lcd "INPUT IS CHANEL6"
Toggle Portb.7
'-----------------------------------------------------------
End Select
Waitms 300
Cls : Home
Lcd "RF RECIVER"
Goto Start_program2
'-----------------------------------------------------------------------------------------------
Sub Recive_of_pt:
Recive_data = &B00001111
If Pinc.0 = 0 Then Recive_data = Recive_data And &B00001110
```

```
If Pinc.1 = 0 Then Recive_data = Recive_data And &B00001101
If Pinc.2 = 0 Then Recive_data = Recive_data And &B00001011
If Pinc.3 = 0 Then Recive_data = Recive_data And &B00000111
End Sub
Return
'-----------------------------------------------------------------------------------------------
```

تشریح نحوه عملکرد برنامه :

عملکرد برنامه فوق بسیار ساده بوده و براحتی قابل تحلیل است تنها نکته مربوط به این برنامه نحوه دریافت ورودی است ، روش دریافت ورودی در این برنامه زمانی مورد استفاده قرار می گیرد که از یک پورت کامل برای دریافت ورودی استفاده نکرده باشیم به عنوان مثال اگر از 4 پین یک پورت به عنوان ورودی و از 4پین دیگر آن به عنوان خروجی استفاده کنیم ، برای دریافت ورودی بایستی این روش را مورد استفاده قرار دهیم در برنامه فوق دریافت ورودی توسط زیر برنامه RECIVE_OF_PT انجام می شود .

```
Sub Recive_of_pt:
Recive_data = &B00001111
If Pinc.0 = 0 Then Recive_data = Recive_data And &B00001110
If Pinc.1 = 0 Then Recive_data = Recive_data And &B00001101
If Pinc.2 = 0 Then Recive_data = Recive_data And &B00001011
If Pinc.3 = 0 Then Recive_data = Recive_data And &B00000111
End Sub
Return
```

در زیر برنامه فوق ابتدا متغیر RECIVE_DATA برابر با &B00001111 قرار داده می شود . سپس هر کدام از ورودی ها به صورت مجزا چک شده و اگر ورودی صفر باشد عدد یک مربوطه توسط AND شدن با صفر به عدد صفر تغییر پیدا می کند ، بدین ترتیب در صورتی که سطح منطقی هر4 ورودی صفر باشد عدد قرار گرفته در متغیر RECIVE_DATA برابر با &B00000000 خواهد بود.

پروژه ارسال و دریافت اطلاعات توسط ماژول های RF

با توجه به این که در پروژه قبل توانستیم اطلاعات را به صورت باینری SEND و RECIVE کنیم . می توانیم اعداد چند رقمی را نیز به باینری تبدیل کرده و آن را ارسال کنیم به عنوان مثال برای فرستادن عدد 3 رقمی 364 می توانیم ابتدا عدد 3 سپس عدد 6 و پس از آن عدد 4 را بفرستیم البته با توجه به این که ارسال و دریافت اطلاعات به صورت آسنکرون انجام می شود یعنی پالس ساعت برای match کردن فرستنده و گیرنده از نظر زمانی فرستاده نمی شود بایستی از یک دیتای همزمانی (synchronus data) برای همزمان کردن فرستنده و گیرنده استفاده کنیم. در این پروژه یک عدد 4 رقمی توسط کاربر وارد شده و بر روی LCD ، (2*16) نوشته می شود ، پس از وارد کردن عدد چهارم ، عدد چهار رقمی وارد شده در 5 مرحله ارسال خواهد شد ، ارسال 4 مرحله مربوط به 4 رقم عدد وارد شده بوده و یک مرحله ، مربوط به ارسال دیتای همزمانی می باشد . به عنوان مثال برای ارسال 1364 ابتدا عدد 12 به منظور همزمان کردن فرستنده و گیرنده ارسال شده سپس به ترتیب اعداد 1 ،2 ،3 ،6 و 4

ارسال می شود گیرنده با دریافت دیتای همزمانی(synchronus data) متوجه می شود که بایستی یک به یک رقم های هزار گان ، صدگان ، دهگان و یکان را دریافت کند .

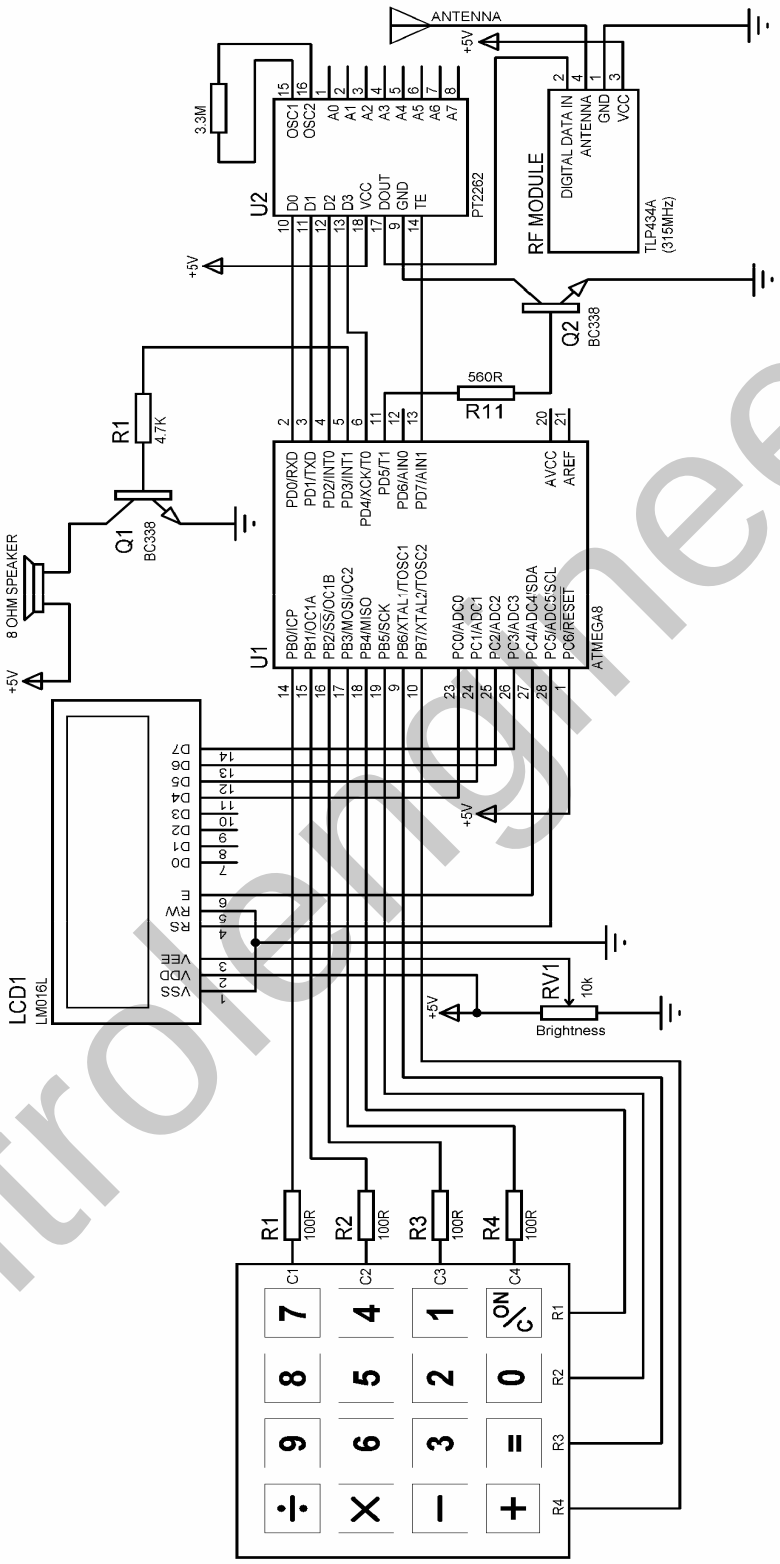
شکل 7-5 شماتیک مدار فرستنده دیتا را نشان می دهد.

برنامه نوشته شده برای سخت افزار فوق به صورت زیر است.

```
'PROGRAM OF TRANSMITTER DATA-----------------------------------------------------------------
'COMPILER:BASCOM 1.11.8.7
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Input
Config Pind.0 = Output
Config Pind.1 = Output
Config Pind.2 = Output
Config Pind.4 = Output
Config Pind.3 = Output
Config Pind.5 = Output : Pt_power Alias Portd.5
Config Pind.7 = Output : Pt_enable Alias Portd.7
Config Lcdpin = Pin , Db4 = Pinc.0 , Db5 = Pinc.1 , Db6 = Pinc.2 , Db7 = Pinc.3_
      , E = Pinc.4 , Rs = Pinc.5
Config Lcd = 16 * 2
Config Kbd = Portb
Dim Input_data As Byte , Send_data As Byte , Keypad_data As Byte
Dim Count As Byte , S1 As Byte , S2 As Byte , S3 As Byte , S4 As Byte
Dim Synchronous_data As Byte
Declare Sub Wireless_send
'-----------------------------------------------------------------------------------------------
Rst:
Reset Pt_power : Set Pt_enable
Cls : Home : Cursor Off
Lcd "ENTER SEND DATA"
S1 = 0 : S2 = 0 : S3 = 0 : S4 = 0 : Count = 0
'-----------------------------------------------------------------------------------------------
Start_program:
Keypad_data = Getkbd()
If Keypad_data > 15 Then Goto Start_program
Input_data = Lookup(keypad_data , Data_code)
If Input_data = 20 Then Goto Start_program
Incr Count
Select Case Count:
Case Is = 1
S1 = Input_data
Locate 2 , 1 : Lcd S1
Sound Portd.3 , 500 , 150

Case Is = 2
S2 = Input_data
Locate 2 , 2 : Lcd S2
Sound Portd.3 , 500 , 150

Case Is = 3
S3 = Input_data
Locate 2 , 3 : Lcd S3
Sound Portd.3 , 500 , 150

Case Is = 4
S4 = Input_data
Locate 2 , 4 : Lcd S4
```

```
Sound Portd.3 , 500 , 150
Waitms 500
Goto Sending
End Select
H1:
Keypad_data = Getkbd()
If Keypad_data <> 16 Then Goto H1
Goto Start_program
'-----------------------------------------------------------------------------------------------------------------
Sending:
Cls : Home : Lcd "SENDIG....."
Synchronous_data = 12
Send_data = Synchronous_data
Locate 2 , 1
Lcd "LEVEL 1"
Call Wireless_send
'-----------------------------------------------------
Send_data = S1
Locate 2 , 1
Lcd "LEVEL 2"
Call Wireless_send
'-----------------------------------------------------
Send_data = S2
Locate 2 , 1
Lcd "LEVEL 3"
Call Wireless_send
'-----------------------------------------------------
Send_data = S3
Locate 2 , 1
Lcd "LEVEL 4"
Call Wireless_send
'-----------------------------------------------------
Send_data = S4
Locate 2 , 1
Lcd "LEVEL 5"
Call Wireless_send
'-----------------------------------------------------
Goto Rst
'END OF PROGRAM-----------------------------------------------------------------------------------------------
'START OF WIRELESS_SEND SUB-----------------------------------------------------------------------------------
Sub Wireless_send:
Reset Pt_power : Set Pt_enable
Incr Send_data
Select Case Send_data:
Case Is = 1
Set Portd.0 : Reset Portd.1 : Reset Portd.2 : Reset Portd.4
'-------------------------------------------------------------------------------------------
Case Is = 2
Reset Portd.0 : Set Portd.1 : Reset Portd.2 : Reset Portd.4
'-------------------------------------------------------------------------------------------
Case Is = 3
Set Portd.0 : Set Portd.1 : Reset Portd.2 : Reset Portd.4
'-------------------------------------------------------------------------------------------
Case Is = 4
Reset Portd.0 : Reset Portd.1 : Set Portd.2 : Reset Portd.4
'-------------------------------------------------------------------------------------------
Case Is = 5
Set Portd.0 : Reset Portd.1 : Set Portd.2 : Reset Portd.4
'-------------------------------------------------------------------------------------------
Case Is = 6
Reset Portd.0 : Set Portd.1 : Set Portd.2 : Reset Portd.4
'-------------------------------------------------------------------------------------------
```

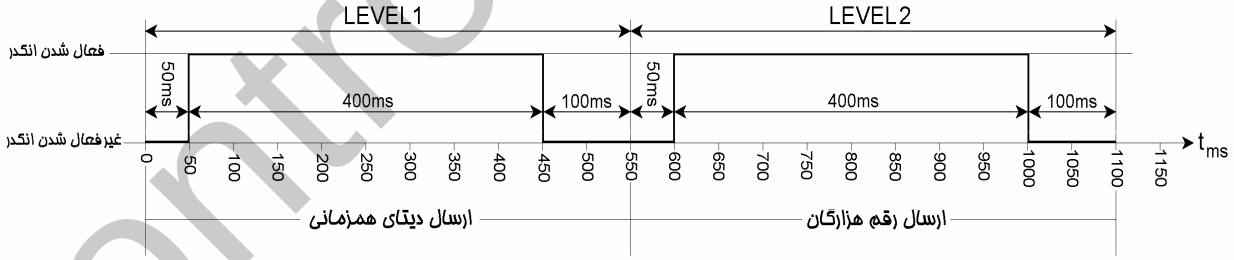
```
Case Is = 7
Set Portd.0 : Set Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 8
Reset Portd.0 : Reset Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 9
Set Portd.0 : Reset Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 10
Reset Portd.0 : Set Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 11
Set Portd.0 : Set Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 12
Reset Portd.0 : Reset Portd.1 : Set Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 13
Set Portd.0 : Reset Portd.1 : Set Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
End Select
Waitms 50
Set Pt_power
Reset Pt_enable
Waitms 400
Reset Pt_power
Set Pt_enable
Waitms 100
Return
End Sub Wireless_send
'END OF WIRELESS_SEND SUB-----------------------------------------------------------------------
Data_code:
Data 1 , 2 , 3 , 20 , 4 , 5 , 6 , 20 , 7 , 8 , 9 , 20 , 20 , 0 , 20 , 20
'-----------------------------------------------------------------------------------------------
```



شکل 7-5 شماتیک مدار فرستنده دیتا

تشریح نحوه عملکرد برنامه :

در برنامه فوق پس از وارد شدن عدد چهارم اجرای برنامه به بر چسب SENDING منتقل می شود که در این بر چسب ابتدا عبارت ...SENDING بر روی LCD نوشته شده سپس متغیر SEND_DATA برابر با 12 قرار می گیرد سپس در خط دوم LCD ، LEVEL1 نوشته شده و زیر برنامه WIRELESS_SEND فراخوانی می شود در زیر برنامه WIRELESS_SEND ابتدا یک واحد به متغیر SEND_DATA اضافه می شود ، همان طور که قبلا نیز توضیح داده شده ، اعمال عدد باینری صفر به ورودی انکدر PT2262 به منزله غیر فعال کردن همه ورودی های آن بوده و در این حالت هیچ کدی فرستاده نخواهد شد. از طرفی امکان دارد که یکی از ارقام وارد شده برابر با صفر باشد از این رو به همه ارقام وارد شده یک واحد اضافه کرده و سپس به ورودی PT2262 اعمال می کنیم در گیرنده نیز پس از دریافت عدد ابتدا یک واحد از آن کم کرده سپس بر روی LCD نمایش خواهیم داد. در برنامه فوق پس از اضافه کردن یک واحد به SEND_DATA آن را به ورودی PT2262 اعمال کرده و پس از 50 میلی ثانیه تاخیر آی سی مزبور را فعال می کنیم ، پس از فعال سازی آی سی انکدر ، 400 میلی ثانیه تاخیر ایجاد شده سپس آی سی مزبور غیر فعال می شود و پس از 100 میلی ثانیه تاخیر با اجرای دستور RETURN اجرای برنامه به قسمت LEVEL2 منتقل می شود. در قسمت LEVEL2 متغیر SEND_DATA برابر با S1 که همان رقم هزارگان وارد شده است قرار می گیرد سپس زیر برنامه WIRELESS_SEND فراخوانی می شود این مراحل تا LEVEL5 ادامه پیدا کرده سپس عدد 4 رقمی جدید درخواست می شود. نمودار زمانی فعال و غیر فعال شدن IC انکدر PT2262 در شکل 5-8 نشان داده شده است . توجه داشته باشید که فعال شدن آی سی مزبور به معنای ارسال DATA و غیر فعال شدن آن به معنای ارسال نشدن دیتا می باشد.



شکل 5-8 نمودار زمانی فعال و غیر فعال شدن انکدرPT2262

با توجه به نمودار شکل 5-8 اجرای هر LEVEL ، 550 میلی ثانیه طول می کشد ، بنا بر این برای ارسال کامل یک عدد 4 رقمی و یک دیتای همزمانی 2.75 ثانیه زمان لازم است البته این روش یک روش پایه برای ارسال و

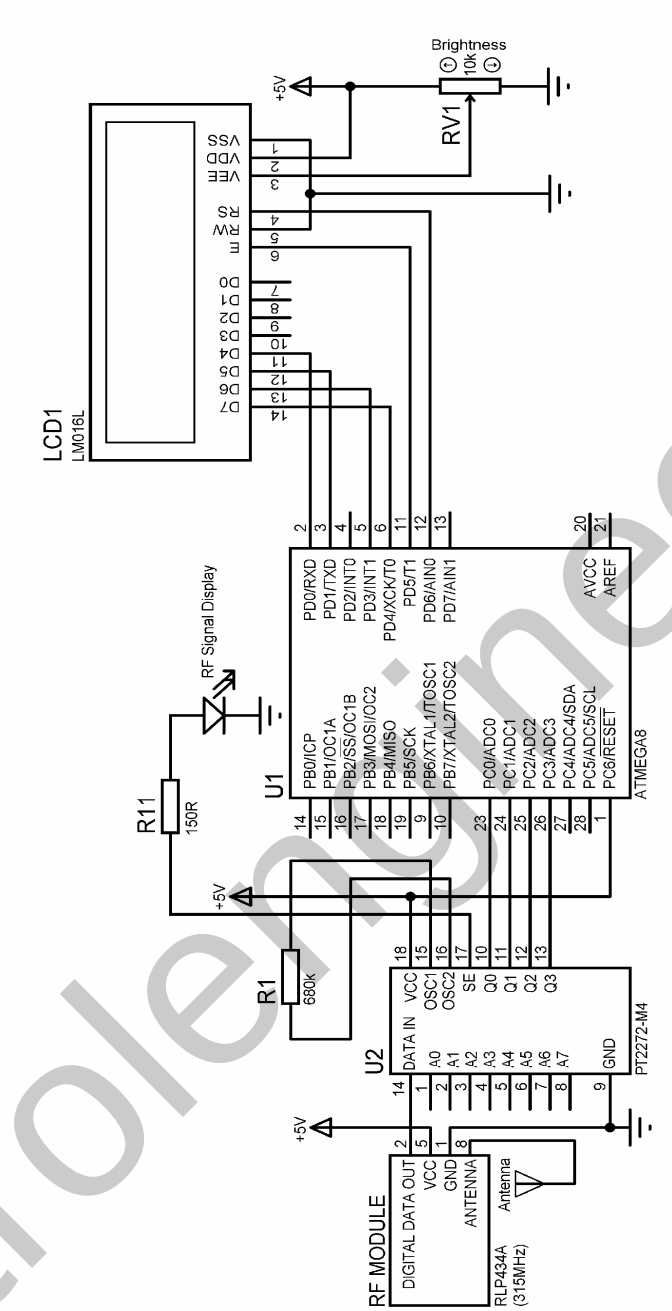
دریافت اطلاعات می باشد شما می توانید با کار کردن بر روی برنامه فوق و همچنین برنامه میکروکنترلر گیرنده مدت زمان ارسال و دریافت را کاهش دهید .

نکته دیگر در رابطه با برنامه فوق مربوط به جدول DATA_CODE می باشد با توجه به این که شکل ظاهری KEYPAD های موجود در بازار باهم متفاوت است شما بایستی ابتدا اعداد برگردانده شده توسط دستور GETKBD() را با استفاده از روشی که در فصل دوم کتاب توضیح داده شده است شناسائی کرده و جدول DATA_CODE را بر اساس آن تنظیم کنید . شماتیک طراحی شده برای گیرنده دیتا در شکل 9-5 نشان داده شده است .

برنامه نوشته شده برای میکروکنترلر مدار گیرنده دیتا به صورت زیر است .

```
'PROGRAM OF RECIVER DATA-----------------------------------------------------------------------
'COMPILER:BASCOM 1.11.8.7
$regfile = "m8def.dat"
$crystal = 8000000
Config Portb = Output
Config Pinc.0 = Input
Config Pinc.1 = Input
Config Pinc.2 = Input
Config Pinc.3 = Input
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.3 , Db7 = Pind.4_
    , E = Pind.5 , Rs = Pind.6
Config Lcd = 16 * 2
Dim Recive_data As Byte
Dim R1 As Byte , R2 As Byte , R3 As Byte , R4 As Byte
Declare Sub Wireless_recive
'-----------------------------------------------------------------------------------------------
Start_program:
'-----------------------------------------------------------
Cls : Home : Cursor Off
Lcd "RF RECIVER DATA"
Lowerline
Lcd "DATA IS " ; R1 ; R2 ; R3 ; R4
'-----------------------------------------------------------
Do
Call Wireless_recive
Waitms 50
Decr Recive_data
If Recive_data = 12 Then Goto Level1
Loop
'-----------------------------------------------------------
Level1:
Locate 2 , 16 : Lcd "1"
Do
Call Wireless_recive
If Recive_data = 0 Then Goto Level2
Loop
'-----------------------------------------------------------
Level2:
Locate 2 , 16 : Lcd "2"
Waitms 200
Call Wireless_recive
Decr Recive_data
R1 = Recive_data
```

```
Do
Call Wireless_recive
If Recive_data = 0 Then Goto Level3
Loop
'-----------------------------------------------------------------
Level3:
Locate 2 , 16 : Lcd "3"
Waitms 200
Call Wireless_recive
Decr Recive_data
R2 = Recive_data
Do
Call Wireless_recive
If Recive_data = 0 Then Goto Level4
Loop
'-----------------------------------------------------------------
Level4:
Locate 2 , 16 : Lcd "4"
Waitms 200
Call Wireless_recive
Decr Recive_data
R3 = Recive_data
Do
Call Wireless_recive
If Recive_data = 0 Then Goto Level5
Loop
'-----------------------------------------------------------------
Level5:
Locate 2 , 16 : Lcd "5"
Waitms 200
Call Wireless_recive
Decr Recive_data
R4 = Recive_data
Do
Call Wireless_recive
If Recive_data = 0 Then Goto Start_program
Loop
'-----------------------------------------------------------------------------------------------------------
Sub Wireless_recive:
Recive_data = &B00001111
If Pinc.0 = 0 Then Recive_data = Recive_data And &B00001110
If Pinc.1 = 0 Then Recive_data = Recive_data And &B00001101
If Pinc.2 = 0 Then Recive_data = Recive_data And &B00001011
If Pinc.3 = 0 Then Recive_data = Recive_data And &B00000111
End Sub Wireless_recive
Return
'-----------------------------------------------------------------------------------------------------------
```

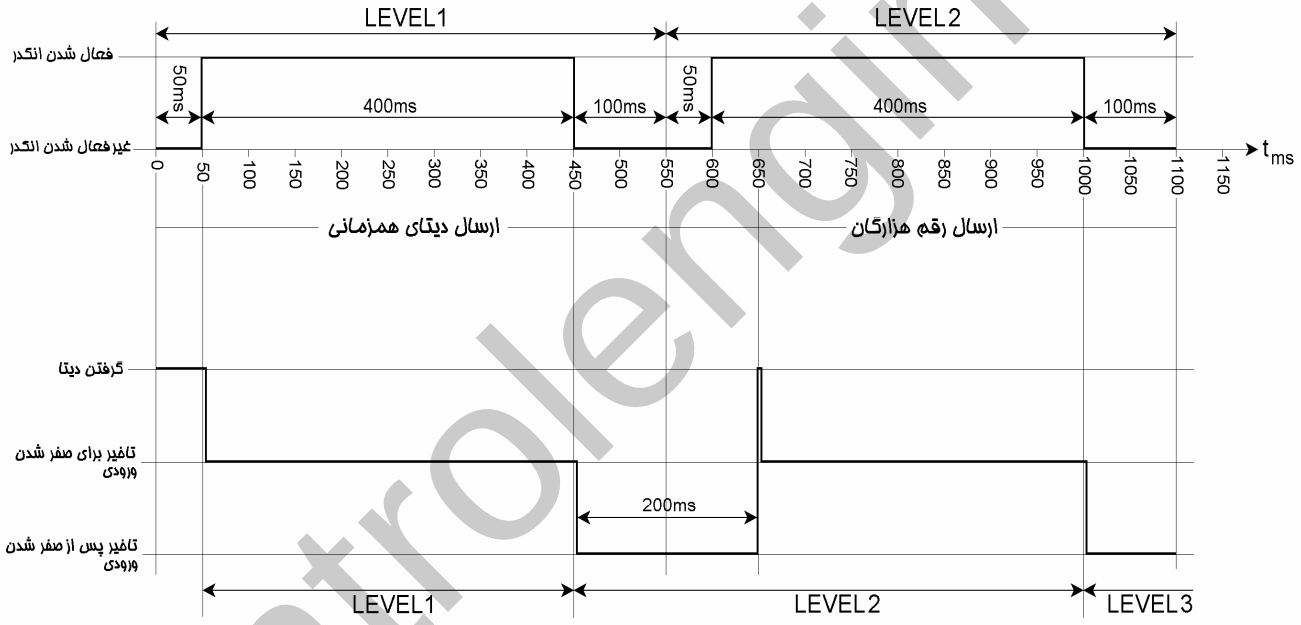


شکل 9-5 شماتیک مدار گیرنده دیتا

تشریح نحوه عملکرد برنامه :

در برنامه فوق ابتدا عبارت RF RECIVER DATA بر روی خط اول LCD نوشته شده و عبارت DTA IS 0000 در خط دوم آن نوشته می شود ، علت نوشته شدن عدد 0000 به عنوان دیتای ورودی این است که متغیر های R4,R3,R2,R1 هنوز مقدار دهی نشده اند که در این حالت مقدار اولیه آن ها صفر خواهد بود سپس ورودی

چک شده و اجرای برنامه تا زمانی که عدد گرفته شده برابر با 12 نباشد در این قسمت متوقف خواهد شد به محض این که دیتای همزمانی یعنی عدد 12 توسط فرستنده ارسال شده و توسط گیرنده دریافت شد اجرای برنامه به برچسب LEVEL1 منتقل می شود. در قسمت LEVEL1 ابتدا عدد 1 در آخرین کاراکتر خط دوم نوشته می شود ، سپس ورودی چک می شود و تا زمانی که ورودی برابر با صفر نشده اجرای برنامه در این قسمت متوقف خواهد شد ، توجه داشته باشید صفر شدن ورودی در گیرنده به معنی غیر فعال شدن IC انکدر در فرستنده می باشد. به محض این که ورودی برابر با صفر شد اجرای برنامه به برچسب LEVEL2 منتقل می شود در این برچسب ابتدا عدد 2 در آخرین کاراکتر خط دوم نوشته شده و پس از 200 میلی ثانیه تاخیر عدد ورودی گرفته می شود سپس یک واحد از این عدد کم شده و این عدد در متغیر R1 قرار می گیرد ، سپس ورودی چک می شود و تا زمانی که ورودی صفر نشده اجرای برنامه در این قسمت متوقف خواهد شد. شکل 9-5 نمودار همزمانی بین فرستنده و گیرنده را نشان می دهد .



شکل 9-5 نمودار همزمانی بین فرستنده و گیرنده

به محض این که ورودی صفر شد اجرای برنامه به LEVEL3 منتقل می شود ، در این قسمت نیز پس از 200 میلی ثانیه تاخیر مقدار R2 گرفته می شود و این عملیات تا LEVEL5 تکرار خواهد شد ، پس از این که پنجمین عدد گرفته شده در متغیر R4 قرار داده شد اجرای برنامه به لیبل START_PROGRAM منتقل شده و عدد چهار رقمی گرفته شده بر روی LCD نمایش داده می شود .

پروژه ارسال و دریافت پیام کوتاه با استفاده از ماژول های RF

اگر برای هر کدام از حروف یک کد عددی دو رقمی تعریف کرده و کد عددی آن ها را توسط ماژول های فرستنده ارسال کنیم ، پس از دریافت این کد ها توسط گیرنده و آشکار سازی آن ها قادر به شناسائی حروف خواهیم بود . می توان از این روش برای ارسال و دریافت پیام کوتاه استفاده کرد. شکل 10-5 شماتیک مدار فرستنده پیام کوتاه را نشان می دهد .

در این پروژه ابتدا پیام کوتاه مانند SMS موبایل توسط KEYPAD ورودی بر روی LCD (4*16) نوشته می شود ، سپس با فشار کلید SEND کد حروف نوشته شده بر روی LCD از طریق ماژول های RF به مدار گیرنده ارسال می شود .

هنگام نوشتن پیام کوتاه توجه داشته باشید که پس از تعیین حرف مورد نظر توسط کلید مربوطه بایستی کلید OK را فشار دهید برای رفتن به خط پایین از کلید LOWERLINE استفاده کنید و برای پاک کردن کاراکتر های نوشته شده کلید DELETE را فشار دهید برای ایجاد فضای خالی یا فاصله دکمه OK را دو بار فشار دهید .

برنامه نوشته شده برای میکروکنترلر در مدار فرستنده پیام کوتاه به صورت زیر است .

```
'COMPILER:BASCOM 1.11.8.7
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Kbd = Portb , Debounce = 50 , Delay = 100
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = Pinc.0 , Db5 = Pinc.1 , Db6 = Pinc.2 , Db7 = Pinc.3_
                                , E = Pinc.4 , Rs = Pinc.5
Dim Lcd_data As String * 1 , X As Byte , Y As Byte , Recive_data As Byte
Dim C1 As Byte , C2 As Byte , C3 As Byte , C4 As Byte , Code As String * 1
Dim C5 As Byte , C6 As Byte , C7 As Byte , C8 As Byte , S1 As String * 1
Dim C9 As Byte , C10 As Byte , C11 As Byte , C0 As Byte
Dim S(81) As String * 1 , Count As Byte , Send As Byte , S2 As String * 1
Dim X1_save As Byte , X2_save As Byte , X3_save As Byte , Send_data As Byte
Dim Synchronous_data As Byte , String_of_send_data As String * 2 , H As Byte
Config Portd = Output
Declare Sub Wireless_send
Declare Sub Message_code
Pt_power Alias Portd.5 : Pt_enable Alias Portd.7
'-------------------------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "PLEASE ENTER"
Locate 2 , 1
Lcd "YOUR MESSAGE"
Wait 1
Cls : Cursor On : Y = 1 : X = 1
'START OF WRITEING MESSAGE-----------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Select Case Recive_data:
'-------------------------------------------------------------
Case Is = 0
```

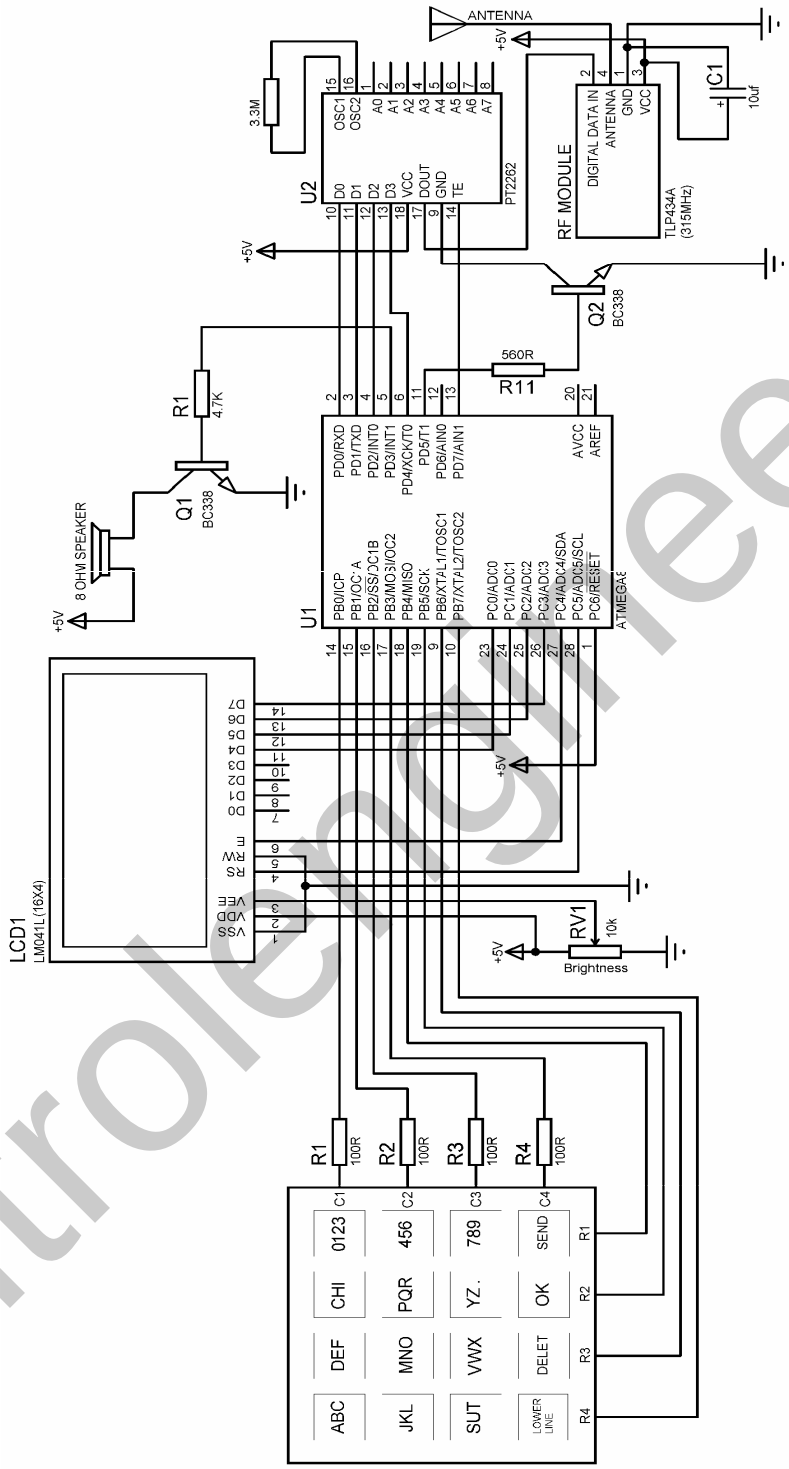
```
Incr C0
If C0 = 1 Then Lcd_data = "0"
If C0 = 2 Then Lcd_data = "1"
If C0 = 3 Then Lcd_data = "2"
If C0 = 4 Then
C0 = 0 : Lcd_data = "3"
End If
'-----------------------------------------------------------------
Case Is = 4
Incr C4
If C4 = 1 Then Lcd_data = "4"
If C4 = 2 Then Lcd_data = "5"
If C4 = 3 Then
C4 = 0 : Lcd_data = "6"
End If
'-----------------------------------------------------------------
Case Is = 8
Incr C8
If C8 = 1 Then Lcd_data = "7"
If C8 = 2 Then Lcd_data = "8"
If C8 = 3 Then
C8 = 0 : Lcd_data = "9"
End If
'-----------------------------------------------------------------
Case Is = 3
Incr C3
If C3 = 1 Then Lcd_data = "A"
If C3 = 2 Then Lcd_data = "B"
If C3 = 3 Then
C3 = 0 : Lcd_data = "C"
End If
'-----------------------------------------------------------------
Case Is = 2
Incr C2
If C2 = 1 Then Lcd_data = "D"
If C2 = 2 Then Lcd_data = "E"
If C2 = 3 Then
C2 = 0 : Lcd_data = "F"
End If
'-----------------------------------------------------------------
Case Is = 1
Incr C1
If C1 = 1 Then Lcd_data = "G"
If C1 = 2 Then Lcd_data = "H"
If C1 = 3 Then
C1 = 0 : Lcd_data = "I"
End If
'-----------------------------------------------------------------
Case Is = 7
Incr C7
If C7 = 1 Then Lcd_data = "G"
If C7 = 2 Then Lcd_data = "H"
If C7 = 3 Then
C7 = 0 : Lcd_data = "I"
End If
'-----------------------------------------------------------------
Case Is = 6
Incr C6
If C6 = 1 Then Lcd_data = "M"
If C6 = 2 Then Lcd_data = "N"
If C6 = 3 Then
C6 = 0 : Lcd_data = "O"
```

```
End If
'-----------------------------------------------------------------
Case Is = 5
Incr C5
If C5 = 1 Then Lcd_data = "P"
If C5 = 2 Then Lcd_data = "Q"
If C5 = 3 Then
C5 = 0 : Lcd_data = "R"
End If
'-----------------------------------------------------------------
Case Is = 11
Incr C11
If C11 = 1 Then Lcd_data = "S"
If C11 = 2 Then Lcd_data = "T"
If C11 = 3 Then
C11 = 0 : Lcd_data = "U"
End If
'-----------------------------------------------------------------
Case Is = 10
Incr C10
If C10 = 1 Then Lcd_data = "V"
If C10 = 2 Then Lcd_data = "W"
If C10 = 3 Then
C10 = 0 : Lcd_data = "X"
End If
'-----------------------------------------------------------------
Case Is = 9
Incr C9
If C9 = 1 Then Lcd_data = "Y"
If C9 = 2 Then Lcd_data = "Z"
If C9 = 3 Then
C9 = 0 : Lcd_data = "."
End If
'START OF LOWERLINE BUTTON PROGRAM-----------------------------------------------------------
Case Is = 15
If S(count) <> Lcd_data Then Lcd_data = " "
Locate Y , X
Lcd Lcd_data
'-----------------------------------------------------------------
Incr Count
S(count) = "$"
If Y = 4 Then Goto H2
Lcd_data = " "
If Y = 1 Then X1_save = X
If Y = 2 Then X2_save = X
If Y = 3 Then X3_save = X
Incr Y : X = 1
H2:
'END OF LOWERLINE BUTTON PROGRAM-------------------------------------------------------
'START OF DELETE BUTTON PROGRAM---------------------------------------------------------
Case Is = 14
S(count) = " "
If Count > 0 Then Decr Count
Decr X
'-----------------------------------------------------------------
If Y = 1 Then
If X = 0 Then X = 1
End If
'-----------------------------------------------------------------
If Y > 1 Then
If X = 0 Then
If Y = 4 Then X = X3_save
```

```
If Y = 3 Then X = X2_save
If Y = 2 Then X = X1_save
Decr Y
End If : End If
'-----------------------------------------------------------
Lcd_data = " "
Locate Y , X
Lcd Lcd_data
'END OF DELETE BUTTON PROGRAM-----------------------------------------------------------
'START OF OK BUTTON PROGRAM-------------------------------------------------------------
Case Is = 13
Incr Count
S(count) = Lcd_data
Lcd_data = " "
'-----------------------------------------------------------
Incr X
If X > 15 Then
If Y = 1 Then X1_save = 15
If Y = 2 Then X2_save = 15
If Y = 3 Then X3_save = 15
If Y < 4 Then
X = 1 : Incr Y
Else
X = 15
End If : End If
'END OF OK BUTTON PROGRAM---------------------------------------------------------------
'START OF SEND BUTTON PROGRAM-----------------------------------------------------------
Case Is = 12
Sending:
'-----------------------------------------------------------
Synchronous_data = 12
Send_data = Synchronous_data
Call Wireless_send
'-----------------------------------------------------------
For Send = 1 To Count Step 1
Code = S(send)
Call Message_code

String_of_send_data = Str(send_data)
H = Len(string_of_send_data)

If H = 1 Then
Send_data = Val(string_of_send_data)
Call Wireless_send
Send_data = 0
Call Wireless_send
End If

If H = 2 Then
S1 = Mid(string_of_send_data , 2 , 1)
Send_data = Val(s1)
Call Wireless_send
S2 = Mid(string_of_send_data , 1 , 1)
Send_data = Val(s2)
Call Wireless_send
End If
Next Send
'-----------------------------------------------------------
Synchronous_data = 12
Send_data = Synchronous_data
Call Wireless_send
'END OF SEND BUTTON PROGRAM-------------------------------------------------------------
```

```
End Select
'-----------------------------------------------------------
Locate Y , X
Lcd Lcd_data
Sound Portd.3 , 100 , 80
'-----------------------------------------------------------
H3:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto H3
'-----------------------------------------------------------
Goto H1
'END OF WRITEING MESSAGE-----------------------------------------------------------------------------
'START OF WIRELESS_SEND SUB---------------------------------------------------------------------------
Sub Wireless_send:
Reset Pt_power : Set Pt_enable
Incr Send_data
Select Case Send_data:
Case Is = 1
Set Portd.0 : Reset Portd.1 : Reset Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 2
Reset Portd.0 : Set Portd.1 : Reset Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 3
Set Portd.0 : Set Portd.1 : Reset Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 4
Reset Portd.0 : Reset Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 5
Set Portd.0 : Reset Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 6
Reset Portd.0 : Set Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 7
Set Portd.0 : Set Portd.1 : Set Portd.2 : Reset Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 8
Reset Portd.0 : Reset Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 9
Set Portd.0 : Reset Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 10
Reset Portd.0 : Set Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 11
Set Portd.0 : Set Portd.1 : Reset Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 12
Reset Portd.0 : Reset Portd.1 : Set Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
Case Is = 13
Set Portd.0 : Reset Portd.1 : Set Portd.2 : Set Portd.4
'-----------------------------------------------------------------------------------------
End Select
Waitms 50
Set Pt_power
Reset Pt_enable
Waitms 400
Reset Pt_power
```

```
Set Pt_enable
Waitms 100
Return
End Sub Wireless_send
'END OF WIRELESS_SEND SUB-----------------------------------------------------------------------
'START OF MESSAGE_CODE SUB-----------------------------------------------------------------------
Sub Message_code:
Select Case Code:
Case Is = "0" : Send_data = 0
Case Is = "1" : Send_data = 1
Case Is = "2" : Send_data = 2
Case Is = "3" : Send_data = 3
Case Is = "4" : Send_data = 4
Case Is = "5" : Send_data = 5
Case Is = "6" : Send_data = 6
Case Is = "7" : Send_data = 7
Case Is = "8" : Send_data = 8
Case Is = "9" : Send_data = 9
Case Is = "A" : Send_data = 10
Case Is = "B" : Send_data = 11
Case Is = "C" : Send_data = 12
Case Is = "D" : Send_data = 13
Case Is = "E" : Send_data = 14
Case Is = "F" : Send_data = 15
Case Is = "G" : Send_data = 16
Case Is = "H" : Send_data = 17
Case Is = "I" : Send_data = 18
Case Is = "J" : Send_data = 19
Case Is = "K" : Send_data = 20
Case Is = "L" : Send_data = 21
Case Is = "M" : Send_data = 22
Case Is = "N" : Send_data = 23
Case Is = "O" : Send_data = 24
Case Is = "P" : Send_data = 25
Case Is = "Q" : Send_data = 26
Case Is = "R" : Send_data = 27
Case Is = "S" : Send_data = 28
Case Is = "T" : Send_data = 29
Case Is = "U" : Send_data = 30
Case Is = "V" : Send_data = 31
Case Is = "W" : Send_data = 32
Case Is = "X" : Send_data = 33
Case Is = "Y" : Send_data = 34
Case Is = "Z" : Send_data = 35
Case Is = "$" : Send_data = 36
Case Is = "." : Send_data = 37
Case Is = " " : Send_data = 38
End Select
Return
End Sub Message_code
'END OF MESSAGE_CODE SUB-----------------------------------------------------------------------
```



شکل 10-5 شماتیک مدار فرستنده پیام کوتاه

تشریح نحوه عملکرد برنامه :

برای هر کدام از کلید ها یک شمارنده در نظر گرفته شده است که پس از هر بار فشار کلید یک واحد به شمارنده اضافه می شود به عنوان مثال در کلید ABC اگر بیش از 3 بار کلید را فشار دهیم شمارنده RESET شده و مقدار آن صفر می شود و با توجه به مقدار شمارنده رشته یا حرف مربوطه در متغیر LCD_DATA قرار می گیرد و رشته مربوطه پس از قرار گرفتن در متغیر LCD_DATA بر روی LCD نمایش داده می شود و با زدن کلید OK یک واحد به شمارنده COUNT که شمارنده تعداد حروف وارد شده می باشد اضافه می شود ، سپس کاراکتر مربوطه در متغیر رشته ای S(COUNT) قرار می گیرد ، برای مثال اگر مقدار COUNT برابر با یک باشد حرف انتخاب شده در متغیر S1 قرار خواهد گرفت سپس یک واحد به متغیر X که مشخص کننده ستونی است که کرزر یا مکان نما در آن قرار می گیرد اضافه می شود .

اگر X بزرگتر از 15 بود مقدار Y چک می شود که متغیر Y مشخص کننده سطری است که مکان نما در آن قرار می گیرد ، اگر Y برابر یک بود X1_SAVE برابر 15 و اگر Y برابر 2 بود X2_SAVE برابر 15 و اگر Y برابر 3 بود X3_SAVE برابر با 15 قرار می گیرد ، متغیر های X_SAVE هنگام پاک کردن حروف نوشته شده توسط کلید DELETE مورد استفاده قرار گیرد بدین صورت که اگر دکمه DELETE را پشت سر هم فشار دهیم و اولین کاراکتر از یک خط برای مثال خط دوم را پاک کنیم ، در این حالت با زدن مجدد دکمه DELETE مکان نما به X2_SAVE رفته و کاراکتر مربوط به آن را پاک خواهد کرد .

با زدن دکمه LOWER LINE ابتدا چک می شود که آخرن کاراکتر موجود بر روی LCD توسط دکمه OK پذیرش شده است یا نه . اگر آخرین کاراکتر OK نشده باشد ابتدا آن را با استفاده از دستورات

```
If S(count) <> Lcd_data Then Lcd_data = " "
Locate Y , X
Lcd Lcd_data
```

پاک می کند ، سپس به منظور رفتن به خط بعدی ابتدا یک واحد به COUNT اضافه کرده و رشته $ را در متغیر S(COUNT) قرار می دهد ، تا هنگام دریافت رشته $ در گیرنده ، گیرنده متوجه شود که بایستی نوشتن را از خط بعدی ادامه دهد، سپس مقدار X در متغیر X_SAVE مربوطه ذخیره می شود تا هنگام پاک کردن صفحه LCD مورد استفاده قرار گیرد. بدین صورت که اگر دکمه DELETE را به صورت پشت سر هم فشار دهیم و اولین کاراکتر از یک خط برای مثال خط سوم را پاک کنیم با زدن مجدد دکمه DELETE مکان نما به X2_SAVE رفته و کاراکتر مربوط به آن را پاک می کند.

عملکرد دکمه DELETE بدین صورت می باشد که ابتدا متغیر S(COUNT) برابر با یک رشته خالی می شود ، سپس یک واحد از COUNT کم می شود ، اگر X برابر با صفر است مقدار y چک می شود و اگر y بزرگتر از یک باشد برای مثال برابر با 4 باشد مقدار X برابر با مقدار X3_SAVE قرار خواهد گرفت ، سپس یک کاراکتر

خالی در مکان X نمایش داده خواهد شد. نحوه عملکرد دکمه SEND بدین صورت می باشد ابتدا متغیر SEN_DATA برابر با دیتای همزمانی یا همان عدد 12 قرار گرفته و زیر برنامه WIRELESS_SEND فراخوانی می شود ، در زیر برنامه WITELESS_SEND یک واحد به SEND_DATA اضافه شده و مقدار باینری آن توسط مدار فرستنده ارسال می شود . نحوه انجام این عملکرد در زیر برنامه WIRELESS_SEND در پروژه های قبل توضیح داده شده است .

سپس اجرای برنامه وارد حلقه FOR_NEXT ، SEND می شود در این حلقه حروف وارد شده یک به یک در متغیر CODE قرار گرفته و زیر برنامه MESSAGE_CODE فراخوانی می شود ، در زیر برنامه MESSAGE_CODE کد مربوط به حرف مورد نظر در متغیر SEND_DATA قرار می گیرد .

کدهای انتخاب شده برای حروف در زیر برنامه MESSAGE_CODE به شرح زیر می باشد.

کاراکتر	کد ارسالی	کاراکتر	کد ارسالی	کاراکتر	کد ارسالی	کاراکتر	کد ارسالی	کاراکتر	کد ارسالی
A	10	**J**	19	**S**	28	**0**	0	**9**	9
B	11	**K**	20	**T**	29	**1**	1	فاصله	37
C	12	**L**	21	**U**	30	**2**	2		
D	13	**M**	22	**V**	31	**3**	3		
E	14	**N**	23	**W**	32	**4**	4		
F	15	**O**	24	**X**	33	**5**	5		
G	16	**P**	25	**Y**	34	**6**	6		
H	17	**Q**	26	**Z**	35	**7**	7		
I	18	**R**	27	**$**	36	**8**	8		

پس از قرار گرفتن کد مربوط به حرف موردنظر در متغیر SEND_DATA و برگشت از زیر برنامه MESSAGE_CODE ، متغیر SEND_DATA توسط دستور

STRING_OF_SEND_DATA=STR (SEND_DATA)

تبدیل به یک متغیر رشته ای شده و در متغیر STRING_OF_SEND_DATA قرار می گیرد سپس تعداد رشته های متغیر STRING_OF_SEND_DATA توسط دستور

H=LEN (STRING_OF_SEND_DATA)

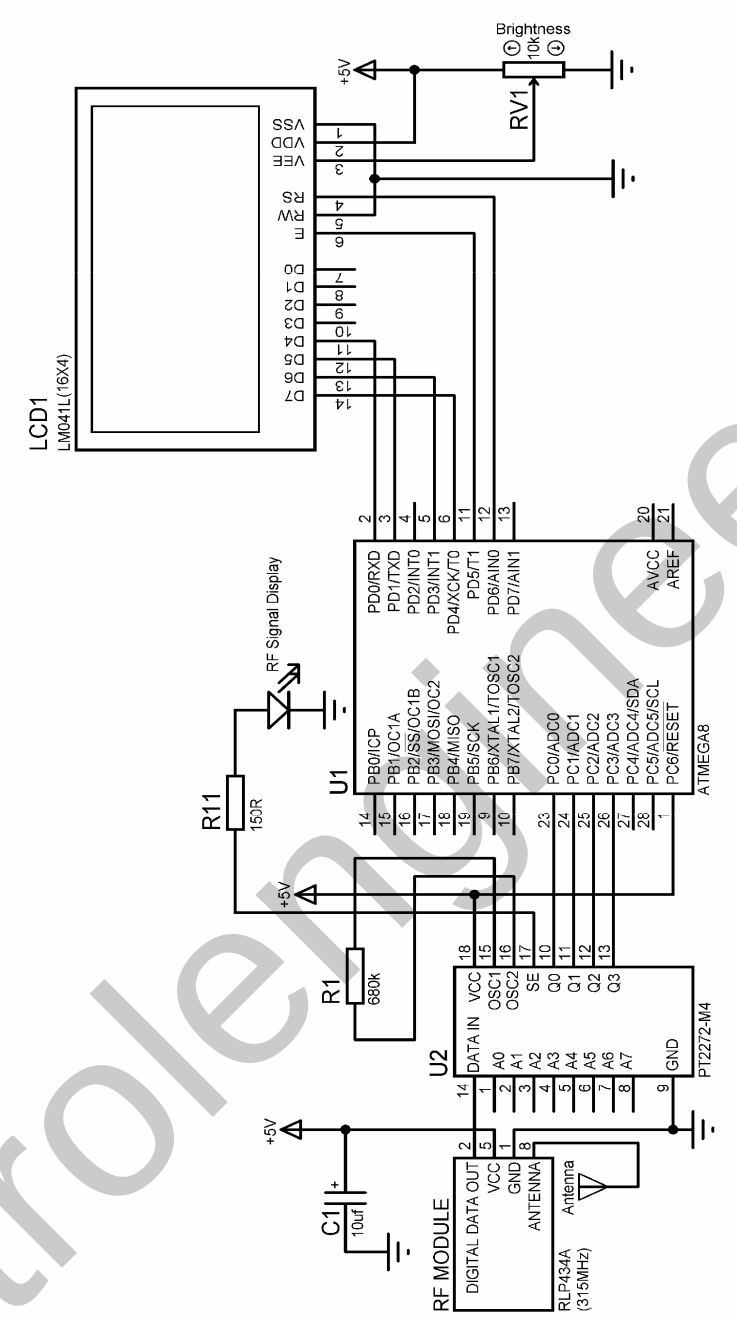
در متغیر H قرار می گیرد. این عمل به منظور شناسائی تعداد رقم های متغیر SEND_DATA انجام می شود برای این که شما تعداد رقم های یک متغیر عددی را داشته باشید یا بتوانید یک یا چند رقم از آن را جدا کنید بایستی ابتدا آن را تبدیل به یک متغیر رشته ای کرده و پس از بدست آوردن تعداد رشته ها (رقم ها) با جداسازی

چند رشته (چند رقم) دوباره رشته یا رشته های جدا شده از آن را تبدیل به متغیر عددی کرده و از آنها استفاده کنید.

پس از این که تعداد رقم ها شناسائی شده و در متغیر H قرار گرفت مقدار متغیر H بررسی می شود ، اگر برابر با یک بود ، ابتدا متغیر رشته ای STRING_OF_SEND_DATA تبدیل به یک متغیر عددی شده و در متغیرSEND_DATA قرار می گیرد ، سپس زیر برنامه WIRELESS_SEND به منظور ارسال آن فراخوانی می شود .

همان طور که قبلا هم گفته شد تعداد ارقام کد های حروف نمی تواند بیشتر از دو رقم باشد توجه داشته باشید که در این برنامه ابتدا رقم یکان فرستاده شده سپس رقم دهگان فرستاده می شود ، اگر کد مورد نظر یک رقمی باشد ابتدا آن را به عنوان رقم یکان فرستاده و به جای رقم دهگان عدد صفر فرستاده می شود و اگر هم کد مربوطه دو رقمی باشد ، ابتدا رقم یکان جدا شده و ارسال می شود ، سپس رقم دهگان جدا شده و فرستاده می شود . پس بدین ترتیب گیرنده پس از دریافت هر دو عدد آن ها را به یک عدد دو رقمی تبدیل کرده و به عنوان کد یک حرف مورد استفاده قرار خواهد داد . در فرستنده پس از ارسال کدهای مربوط به نام حروف نوشته شده بر روی LCD دوباره عدد 12 به عنوان دیتای همزمانی فرستاده می شود ، در مدار گیرنده میکروکنترلر پس از دریافت اولین عدد 12 شروع به دریافت کد های مربوط به کاراکتر های ارسال شده کرده و پس از دریافت دومین عدد 12 به این کار خاتمه می دهد.

شکل 11-5 شماتیک گیرنده طراحی شده برای فرستنده RF پیام کوتاه را نشان می دهد.



شکل 5-11 شماتیک گیرنده RF پیام کوتاه

برنامه نوشته شده برای میکروکنترلر گیرنده RF پیام کوتاه به صورت زیر است.

```
'COMPILER:BASCOM 1.11.8.7
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.3 , Db7 = Pind.4 , _
                E = Pind.5 , Rs = Pind.6
```

```
Dim Lcd_data As String * 1 , X As Byte , Y As Byte , Rs As Byte
Dim Cls_lcd As Bit , Recive_data As Byte , R1 As Byte , R2 As Byte , R As Byte
Config Pinc.0 = Input
Config Pinc.1 = Input
Config Pinc.2 = Input
Config Pinc.3 = Input
Declare Sub Wireless_recive
Declare Sub Message_code
Declare Sub Synchronous_program
'-----------------------------------------------------------------------------------------------------------
Y = 1 : X = 1
Cls : Home : Cursor Off
Lcd "RF MESSAGE"
Locate 2 , 1 : Lcd "RECIVER"
'-----------------------------------------------------------------------------------------------------------
Do
Call Wireless_recive
Waitms 50
Decr Recive_data
If Recive_data = 12 Then
Cls : Home
Goto Recive
End If
Loop
'START OF RECIVE MESSAGE-----------------------------------------------------------------------------------
Recive:
Do
Call Wireless_recive
If Recive_data = 0 Then Goto Level1
Loop
Level1:
Waitms 200
Call Wireless_recive
Decr Recive_data
R1 = Recive_data
Rs = R1 : Call Synchronous_program
Do
Call Wireless_recive
If Recive_data = 0 Then Goto Level2
Loop
'-----------------------------------------------------------------------------------------------------------
Level2:
Waitms 200
Call Wireless_recive
Decr Recive_data
R2 = Recive_data
Rs = R2 : Call Synchronous_program
Do
Call Wireless_recive
If Recive_data = 0 Then Goto Level3
Loop
'-----------------------------------------------------------------------------------------------------------
Level3:
R2 = R2 * 10
R = R1 + R2
Call Message_code
'-----------------------------------------------------------------------------------------------------------
If Cls_lcd = 1 Then
Cls : Cls_lcd = 0
Y = 1 : X = 1
End If
'-----------------------------------------------------------------------------------------------------------
```

```
Locate Y , X
'-----------------------------------------------------------------------------------------
If Lcd_data = "$" Then
Incr Y : X = 1
Goto Recive
End If
'-----------------------------------------------------------------------------------------
Lcd Lcd_data
'-----------------------------------------------------------------------------------------
Incr X
If X > 15 Then
If Y = 4 Then
Y = 4 : X = 15
Else
X = 1 : Incr Y
End If : End If
'-----------------------------------------------------------------------------------------
Goto Recive
'END OF RECIVE MESSAGE-----------------------------------------------------------------------------------------
'START OF WIRELESS_RECIVE SUB-----------------------------------------------------------------------------------
Sub Wireless_recive:
Recive_data = &B00001111
If Pinc.0 = 0 Then Recive_data = Recive_data And &B00001110
If Pinc.1 = 0 Then Recive_data = Recive_data And &B00001101
If Pinc.2 = 0 Then Recive_data = Recive_data And &B00001011
If Pinc.3 = 0 Then Recive_data = Recive_data And &B00000111
End Sub Wireless_recive
Return
'END OF WIRELESS_RECIVE SUB-----------------------------------------------------------------------------------
'START OF MESSAGE_CODE SUB-------------------------------------------------------------------------------------
Sub Message_code:
Select Case R:
Case Is = 0 : Lcd_data = "0"
Case Is = 1 : Lcd_data = "1"
Case Is = 2 : Lcd_data = "2"
Case Is = 3 : Lcd_data = "3"
Case Is = 4 : Lcd_data = "4"
Case Is = 5 : Lcd_data = "5"
Case Is = 6 : Lcd_data = "6"
Case Is = 7 : Lcd_data = "7"
Case Is = 8 : Lcd_data = "8"
Case Is = 9 : Lcd_data = "9"
Case Is = 10 : Lcd_data = "A"
Case Is = 11 : Lcd_data = "B"
Case Is = 12 : Lcd_data = "C"
Case Is = 13 : Lcd_data = "D"
Case Is = 14 : Lcd_data = "E"
Case Is = 15 : Lcd_data = "F"
Case Is = 16 : Lcd_data = "G"
Case Is = 17 : Lcd_data = "H"
Case Is = 18 : Lcd_data = "I"
Case Is = 19 : Lcd_data = "J"
Case Is = 20 : Lcd_data = "K"
Case Is = 21 : Lcd_data = "L"
Case Is = 22 : Lcd_data = "M"
Case Is = 23 : Lcd_data = "N"
Case Is = 24 : Lcd_data = "O"
Case Is = 25 : Lcd_data = "P"
Case Is = 26 : Lcd_data = "Q"
Case Is = 27 : Lcd_data = "R"
Case Is = 28 : Lcd_data = "S"
Case Is = 29 : Lcd_data = "T"
```

```
Case Is = 30 : Lcd_data = "U"
Case Is = 31 : Lcd_data = "V"
Case Is = 32 : Lcd_data = "W"
Case Is = 33 : Lcd_data = "X"
Case Is = 34 : Lcd_data = "Y"
Case Is = 35 : Lcd_data = "Z"
Case Is = 36 : Lcd_data = "$"
Case Is = 37 : Lcd_data = "."
Case Is = 38 : Lcd_data = " "
End Select
Return
End Sub Message_code
'END OF MESSAGE_CODE SUB-----------------------------------------------------------------------------
'START OF Synchronous_program SUB---------------------------------------------------------------------
Sub Synchronous_program:
If Rs = 12 Then
Cls_lcd = 1
Do
Call Wireless_recive
If Recive_data = 0 Then Goto S1
Loop
S1:
Do
Call Wireless_recive
Waitms 50
Decr Recive_data
If Recive_data = 12 Then
Cls : Home
Goto Recive
End If
Loop
End If
Return
End Sub Synchronous_program
'END OF Synchronous_program SUB-----------------------------------------------------------------------
```

تشریح نحوه عملکرد برنامه گیرنده پیام کوتاه :

ابتدا عبارت RF REMOTE RECIVER بر روی LCD نوشته می شود سپس ورودی چک می شود به محض این که دیتای ورودی برابر با 12 شد ، LCD پاک شده و اجرای برنامه به برچسب RECIVE منتقل می شود ،در بر چسب RECIVE ورودی چک می شود و تا زمانی که ورودی برابر صفر نشده اجرای برنامه در این قسمت متوقف خواهد شد ، پس از صفر شدن ورودی 200 میلی ثانیه تاخیر ایجاد شده سپس دیتای ورودی در متغیر RECIVE_DATA قرار داده می شود ، سپس یک واحد از متغیر RECIVE_DATA کم شده و متغیر R1 برابر با RECIVE_DATA قرار داده می شود . محتوای متغیر R1 در واقع همان رقم یکان کد فرستاده شده می باشد ، پس از گرفتن رقم یکان ابتدا متغیر RS را برابرR1 قرار داده و زیر برنامه SYNCHRONOUS_PROGRAM فراخوانی می شود در این زیر برنامه ابتدا متغیر RS چک می شود اگر برابر با 12 بود ، متغیر CLS_LCD برابر با یک شده و اجرا ی برنامه تا زمانی که عدد 12 بعدی گرفته شود (یعنی تا شروع ارسال پیام بعدی) در این قسمت متوقف خواهد شد. توجه داشته باشید زمانی که میکروکنترلر می

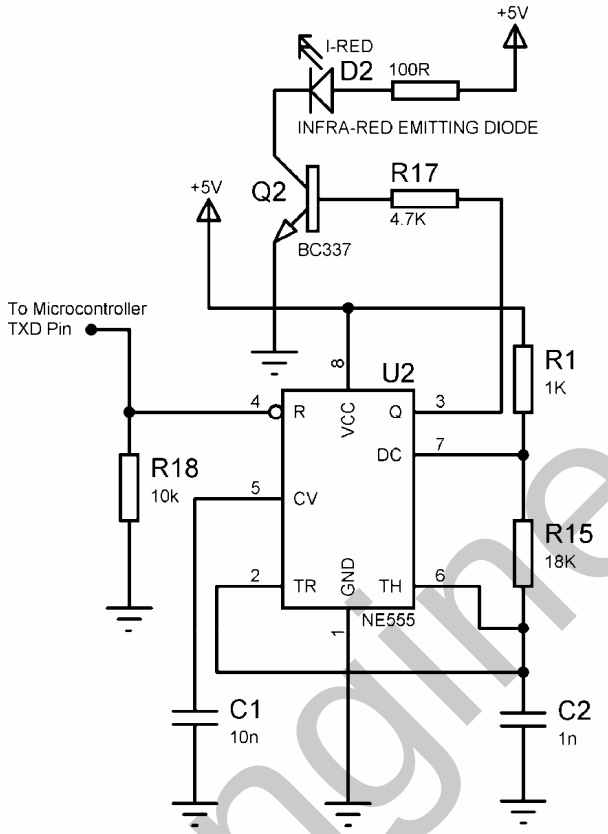
خواهد ، کاراکتر گرفته شده را بر روی LCD بنویسید ابتدا متغیر CLS_LCD را چک می کند ، زمانی که این متغیر برابر با یک باشد صفحه LCD پاک شده و مکان نما به کاراکتر اول خط اول می رود ، به عبارتی دیگر پیام جدید بر روی LCD نوشته می شود .

پس از فراخوانی زیر برنامه SYNCHRONOUS_PROGRAM ، اگر عدد گرفته شده برابر با 12 نباشد اجرای برنامه به محل قبلی خود برگشت داده می شود. که در آنجا نیز ورودی چک می شود و تا زمانی که ورودی برابر صفر نشده اجرای برنامه در این قسمت متوقف خواهد شد . به محض صفر شدن ورودی اجرای برنامه به بر چسب LEVEL2 منتقل می شود ، در این بر چسب ابتدا 200 میلی ثانیه تاخیر ایجاد شده ، سپس ورودی خوانده شده و در متغیر RECIVE_DATA قرار می گیرد ، سپس یک واحد از متغیر RECIVE_DATA کم شده و R2 برابر با RECIVE_DATA قرار داده می شود ، پس از گرفتن رقم دهگان ابتدا RS برابر با R2 قرار داده شده و زیر برنامه SYNCHRONOUS_PROGRAM برای چک کردن این که عدد گرفته شده برابر با 12 است یا نه فراخوانی می شود ، پس از فراخوانی زیر برنامه و برگشت از آن ورودی چک می شود و تا زمانی که ورودی برابر با صفر نشده اجرای برنامه در این قسمت متوقف خواهد شد ، در این برچسب ابتدا کد فرستاده شده آشکار سازی شده و در متغیر R قرار می گیرد ، سپس زیر برنامه MESSAGE_CODE فراخوانی می شود در این زیربرنامه کاراکتر مربوط به کد دریافت شده در متغیر SEND_DATA قرار می گیرد. پس از برگشت از زیر برنامه MESSAGE_CODE ابتدا متغیر CLS_LCD چک می شود اگر برابر با یک بود ، صفحه LCD پاک شده و مکان نما در ابتدای خط اول قرار می گیرد . سپس متغیر LCD_DATA چک می شود اگر برابر با $ بود مکان نما به خط بعدی منتقل می شود ، پس از آن متغیر LCD_DATA بر روی LCD نوشته می شود . سپس مکان نما یک واحد به جلو تغییر مکان می دهد . پیشنهاد می کنم برای درک بهتر نحوه ارسال و دریافت پیام کوتاه برنامه های مربوطه به فرستنده و گیرنده پیام کوتاه را چندین بار به دقت مطالعه کنید .

طراحی یک ریموت کنترل مادون قرمز میکروکنترلری

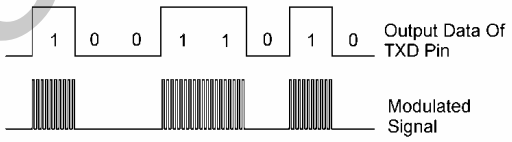
طراحی یک ریموت کنترل مادون قرمز توسط میکروکنترلر بسیار ساده می باشد برای این کار کافی است از خروجی سریال میکروکنترلر برای ارسال اطلاعات استفاده کنیم البته خروجی پورت سریال بایستی با توجه به فرکانس کاری چشمی 3 پایه گیرنده مدولاسیون شود . به عنوان مثال اگر فرکانس کاری چشمی 40KHz باشد ، کافی است اطلاعات خروجی از پایه TXD را روی فرکانس 40KHz سوار کرده و توسط یک دیود مادون قرمز (INFRA-RED EMITTING DIODE) در فضا منتشر کنیم . در این کتاب از چشمی مادون قرمز HS0038 به عنوان گیرنده مادون قرمز استفاده شده است ، فرکانس کاری این چشمی ها 38 کیلو هرتز می باشد . به طور معمول دو رقم آخر نام این چشمی ها نشان دهنده فرکانس کاری آنها بر حسب کیلو هرتز می باشد.

شکل 5-12 یک مدولاتور 38KHz را برای اطلاعات خروجی از پایه TXD میکروکنترلر نشان می دهد .



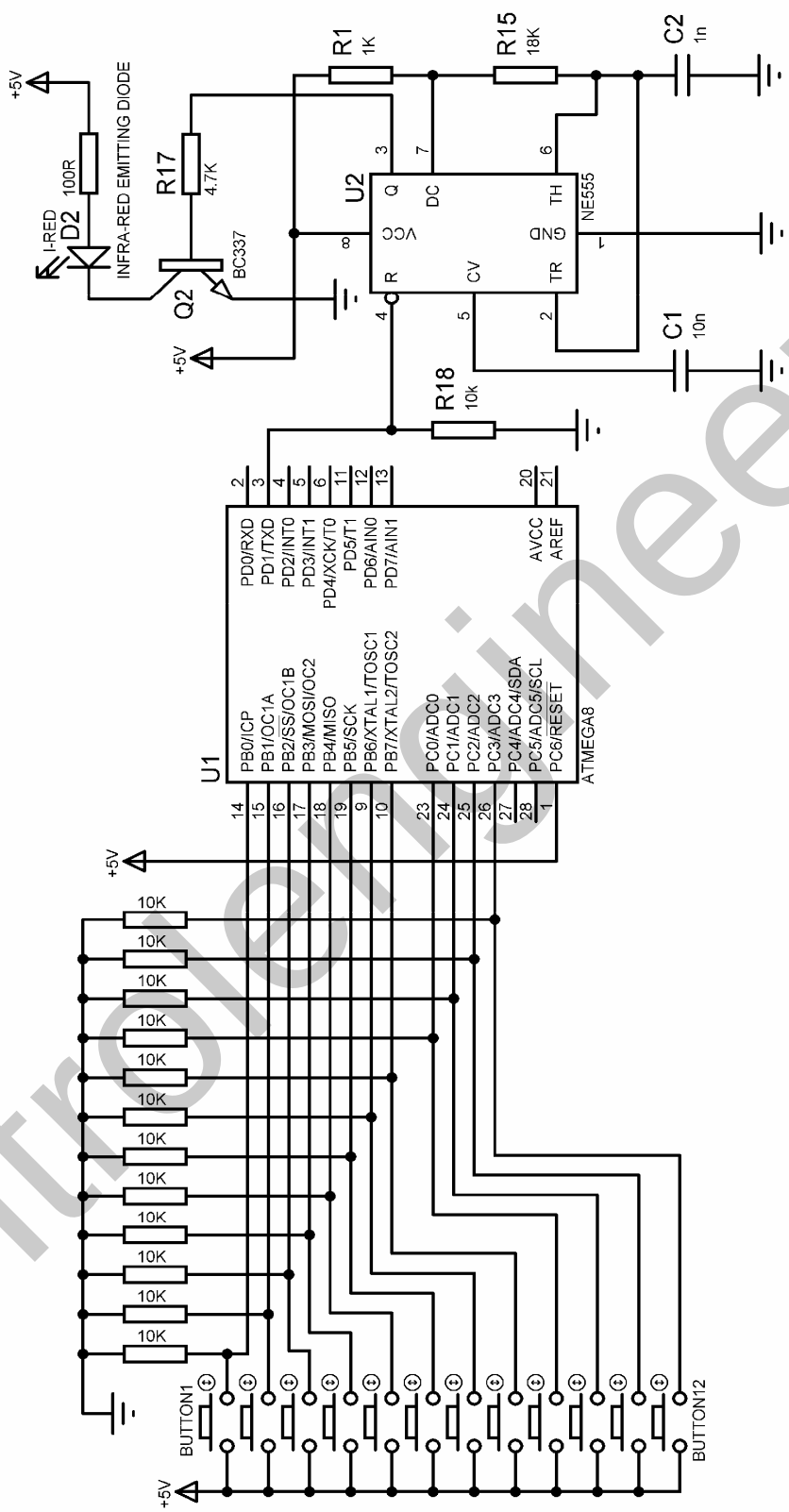
شکل 5-12 مدولاتور 38KHz برای ریموت مادون قرمز

با توجه به این که آی سی NE555 زمانی فعال می شود که پایه شماره 4 (ENABLE) آن در حالت HIGH قرار گیرد ، نحوه انجام مدولاسیون به صورت زیر خواهد بود .



شکل 5-13 نحوه انجام مدولاسیون برای خروجی سریال

شکل 5-14 شماتیک مدار فرستنده مادون قرمز 12 کاناله را نشان می دهد.



شکل 5-14 شماتیک مدار فرستنده مادون قرمز 12 کاناله

برنامه نوشته شده برای میکروکنترلر مدار فرستنده مادون قرمز به صورت زیر می باشد.

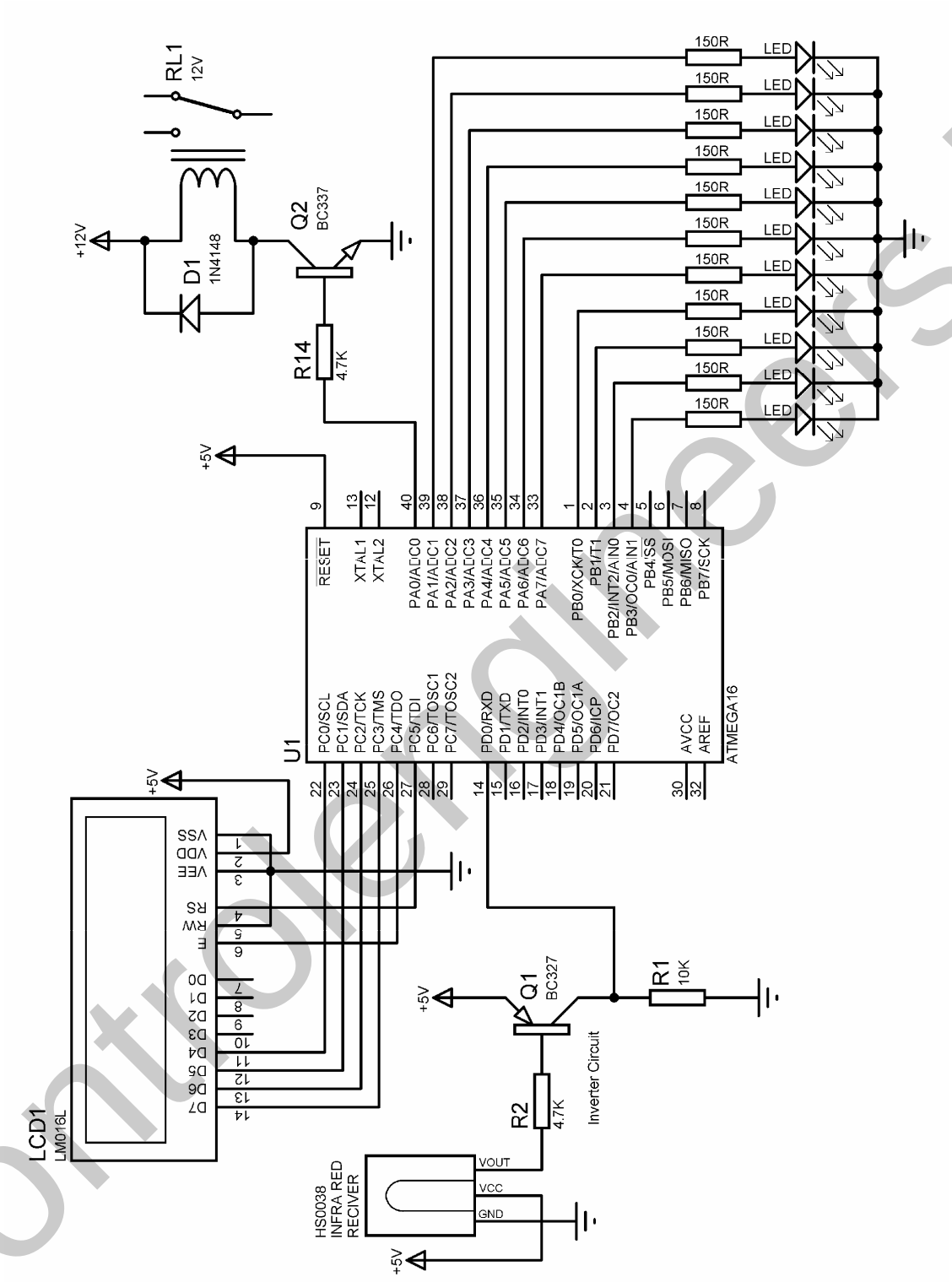
```
'COMPILER:BASCOM 1.11.8.7
$regfile = "M8DEF.DAT"
$crystal = 4000000
Config Serialout = Buffered , Size = 20
$baud = 2400
Enable Interrupts
Config Portb = Input
Config Pinc.0 = Input
Config Pinc.1 = Input
Config Pinc.2 = Input
Config Pinc.3 = Input
Config Pind.1 = Output
'START OF PROGRAM-----------------------------------------------------------------------------------------------
Start_program:
If Pinb.0 = 1 Then
Print "CHANNEL0 "
Bitwait Pinb.0 , Reset
End If
'-----------------------------------------------------------------
If Pinb.1 = 1 Then
Print "CHANNEL1 "
Bitwait Pinb.1 , Reset
End If
'-----------------------------------------------------------------
If Pinb.2 = 1 Then
Print "CHANNEL2 "
Bitwait Pinb.2 , Reset
End If
'-----------------------------------------------------------------
If Pinb.3 = 1 Then
Print "CHANNEL3 "
Bitwait Pinb.3 , Reset
End If
'-----------------------------------------------------------------
If Pinb.4 = 1 Then
Print "CHANNEL4 "
Bitwait Pinb.4 , Reset
End If
'-----------------------------------------------------------------
If Pinb.5 = 1 Then
Print "CHANNEL5 "
Bitwait Pinb.5 , Reset
End If
'-----------------------------------------------------------------
If Pinb.6 = 1 Then
Print "CHANNEL6 "
Bitwait Pinb.6 , Reset
End If
'-----------------------------------------------------------------
If Pinb.7 = 1 Then
Print "CHANNEL7 "
Bitwait Pinb.7 , Reset
End If
'-----------------------------------------------------------------
If Pinc.0 = 1 Then
Print "CHANNEL8 "
Bitwait Pinc.0 , Reset
```

```
End If
'-----------------------------------------------------------------
If Pinc.1 = 1 Then
Print "CHANNEL9 "
Bitwait Pinc.0 , Reset
End If
'-----------------------------------------------------------------
If Pinc.2 = 1 Then
Print "CHANNEL10"
Bitwait Pinc.0 , Reset
End If
'-----------------------------------------------------------------
If Pinc.3 = 1 Then
Print "CHANNEL11"
Bitwait Pinc.0 , Reset
End If
'-----------------------------------------------------------------
Waitms 300
Goto Start_program
'END OF PROGRAM-----------------------------------------------------------------------------------------------
```

تشریح نحوه عملکرد برنامه :

برنامه فوق بسیار ساده بوده و براحتی قابل تحلیل است. اما توجه داشته باشید برای این که عمل گرفتن دیتا توسط چشمی 3 پایه HS0038 بدرستی انجام گیرد بایستی بیت ارسال شده حداقل 395 میکرو ثانیه وجود داشته باشد و گرنه تبادل اطلاعات امکان پذیر نخواهد بود. با تعیین میزان BAUD=2400 مدت زمان ارسال هر بیت تقریبا برابر با 416 میکرو ثانیه خواهد بود بنابراین این میزان باود (BAUD RATE) برای ارسال اطلاعات به چشمی HS0038 مناسب می باشد توجه داشته باشید که شما می توانید برنامه وسخت افزار مدار فرستنده را برای استفاده از مد اسلیپ POWERDOWN به منظور کاهش توان مصرفی سیستم تغییر دهید .

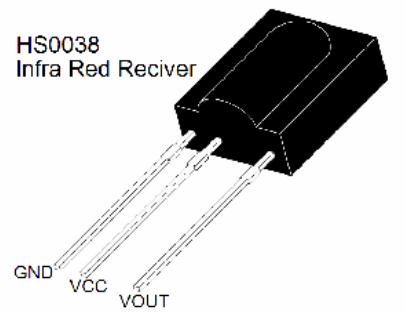
شکل 15-5 شماتیک مدار گیرنده طراحی شده برای ریموت کنترل مادون قرمز 12 کاناله را نشان می دهد .



شکل 5-15 شماتیک گیرنده ریموت کنترل مادون قرمز 12 کاناله

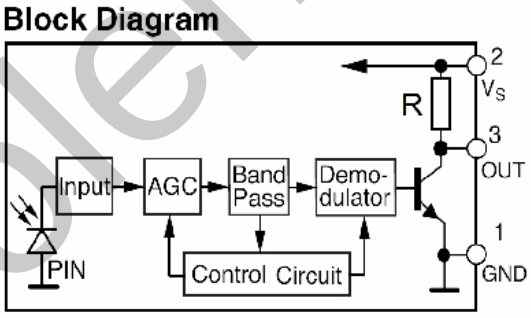
در شماتیک فوق می توانید برحسب نیاز خود از خروجی ها برای راه اندازی رله یا روشن ، خاموش کردن LED و غیره استفاده کنید .

شکل 5-16 آرایش پایه ها ی چشمی مادون قرمز HS0038 و همچنین شکل ظاهری آن را نشان می دهد .



شکل 5-16 شکل ظاهری HS0038 و ترتیب پایه ها

چشمی فوق شامل گیرنده مادون قرمز ، تقویت کننده AGC ،فیلتر،دمودلاتور و مدار کنترل می باشد و لزومی به طراحی هیچکدام از موارد نامبرده نیست فقط با توجه به این که خروجی چشمی به صورت INVERTED می باشد، از یک ترانزیستور PNP (BC327) برای معکوس کردن خروجی استفاده شده است. بلوک دیاگرام چشمی گیرنده مادون قرمز در شکل 5-17 نشان داده شده است .



شکل 5-17 بلوک دیاگرام داخلی گیرنده مادون قرمز HS0038

برنامه نوشته شده برای گیرنده ریموت کنترل مادون قرمز به صورت زیر می باشد.

```
'COMPILER:BASCOM 1.11.8.7
$regfile = "M16DEF.DAT"
$crystal = 4000000
Config Lcd = 16 * 2
Dim Recive_data As String * 10
Config Lcdpin = Pin , Db4 = Pinc.0 , Db5 = Pinc.1 , Db6 = Pinc.2 , Db7 = Pinc.3_
                        , E = Pinc.4 , Rs = Pinc.5
Config Serialin = Buffered , Size = 20
Config Pind.0 = Input
```

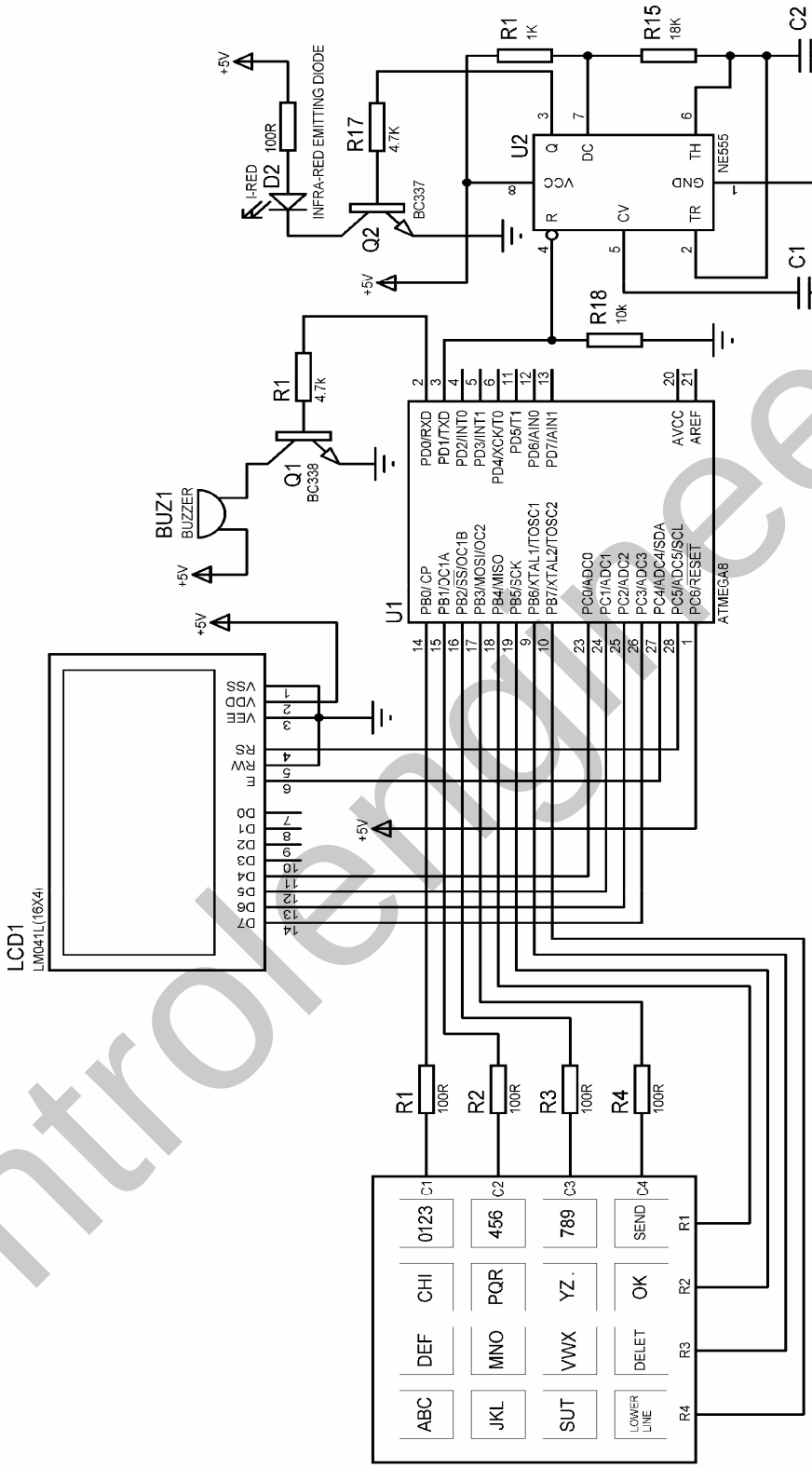
```
$baud = 2400
Config Porta = Output
Config Portb = Output
Enable Interrupts
'START OF PROGRAM-----------------------------------------------------------------------------------------
Cls : Home : Cursor Off
Lcd "I-R RECIVER"
Start_program:
Input Recive_data
Recive_data = Mid(recive_data , 2 , 9)
Locate 2 , 1
Lcd Recive_data
Select Case Recive_data:
Case Is = "CHANNEL0 " : Toggle Porta.0
Case Is = "CHANNEL1 " : Toggle Porta.1
Case Is = "CHANNEL2 " : Toggle Porta.2
Case Is = "CHANNEL3 " : Toggle Porta.3
Case Is = "CHANNEL4 " : Toggle Porta.4
Case Is = "CHANNEL5 " : Toggle Porta.5
Case Is = "CHANNEL6 " : Toggle Porta.6
Case Is = "CHANNEL7 " : Toggle Porta.7
Case Is = "CHANNEL8 " : Toggle Portb.0
Case Is = "CHANNEL9 " : Toggle Portb.1
Case Is = "CHANNEL10" : Toggle Portb.2
Case Is = "CHANNEL11" : Toggle Portb.3
End Select
Goto Start_program
'END OF PROGRAM-------------------------------------------------------------------------------------------
```

برنامه فوق بسیار ساده بوده و براحتی قابل تحلیل است تنها نکته برنامه فوق این است که در UART سخت افزاری و دستور INPUT VAR کاراکتر دریافت شده از پورت را به صورت "H" در متغیر رشته ای VAR قرار می دهد یعنی شما برای دریافت یک کاراکتر بایستی متغیری با طول دو رشته تعریف کنید و با توجه به این که کاراکتر اول از سمت چپ خالی است ، کاراکتر دوم را از رشته VAR جدا کرده و به عنوان کاراکتر دریافت شده مورد استفاده قرار دهید .

پروژه ارسال و دریافت پیام کوتاه از طریق کانال ارتباطی مادون قرمز (INFRA RED)

ارسال و دریافت پیام کوتاه از طریق امواج نوری مادون قرمز کار بسیار راحتی است کافی است بتوانیم پیام کوتاه را ازطریق پورت سریال بین دو میکروکنترلر SEND و RECIVE کنیم ، پس از انجام این کار کافی است به جای کانال ارتباطی سیم بین میکروکنترلر های MASTER و SLAVE از کانال ارتباطی مادون قرمز استفاده کنیم .

شما می توانید از مدار شکل 18-5 به عنوان فرستنده پیام کوتاه استفاده کنید .



شکل 5-18 شماتیک فرستنده پیام کوتاه از طریق کانال ارتباطی مادون قرمز

برنامه نوشته شده برای فرستنده پیام کوتاه از طریق کانال ارتباطی مادون قرمز به صورت زیر می باشد .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 4000000
Config Kbd = Portb , Debounce = 50 , Delay = 100
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = Pinc.0 , Db5 = Pinc.1 , Db6 = Pinc.2 , Db7 = Pinc.3_
                    , E = Pinc.4 , Rs = Pinc.5
Dim Lcd_data As String * 1 , X As Byte , Y As Byte , Recive_data As Byte
Dim C1 As Byte , C2 As Byte , C3 As Byte , C4 As Byte
Dim C5 As Byte , C6 As Byte , C7 As Byte , C8 As Byte
Dim C9 As Byte , C10 As Byte , C11 As Byte , C0 As Byte
Dim S(81) As String * 1 , Count As Byte , Send As Byte
Dim X1_save As Byte , X2_save As Byte , X3_save As Byte
Config Serialout = Buffered , Size = 200
Config Pind.0 = Output
Config Pind.1 = Output
Enable interrupts
$baud = 2400
'-----------------------------------------------------------------------------------------------------------------
Cursor Off
Cls : Home
Lcd "PLEASE ENTER"
Locate 2 , 1
Lcd "YOUR MESSAGE "
Wait 1
Cls : Cursor On : Y = 1 : X = 1
'-----------------------------------------------------------------------------------------------------------------
'START OF WRITEING MESSAGE-----------------------------------------------------------------------------------
H1:
Recive_data = Getkbd()
If Recive_data = 16 Then Goto H1
Select Case Recive_data:
'-----------------------------------------------------
Case Is = 0
Incr C0
If C0 = 1 Then Lcd_data = "1"
If C0 = 2 Then Lcd_data = "2"
If C0 = 3 Then
C0 = 0 : Lcd_data = "3"
End If
'-----------------------------------------------------
Case Is = 4
Incr C4
If C4 = 1 Then Lcd_data = "4"
If C4 = 2 Then Lcd_data = "5"
If C4 = 3 Then
C4 = 0 : Lcd_data = "6"
End If
'-----------------------------------------------------
Case Is = 8
Incr C8
If C8 = 1 Then Lcd_data = "7"
If C8 = 2 Then Lcd_data = "8"
If C8 = 3 Then
C8 = 0 : Lcd_data = "9"
End If
'-----------------------------------------------------
Case Is = 3
Incr C3
```

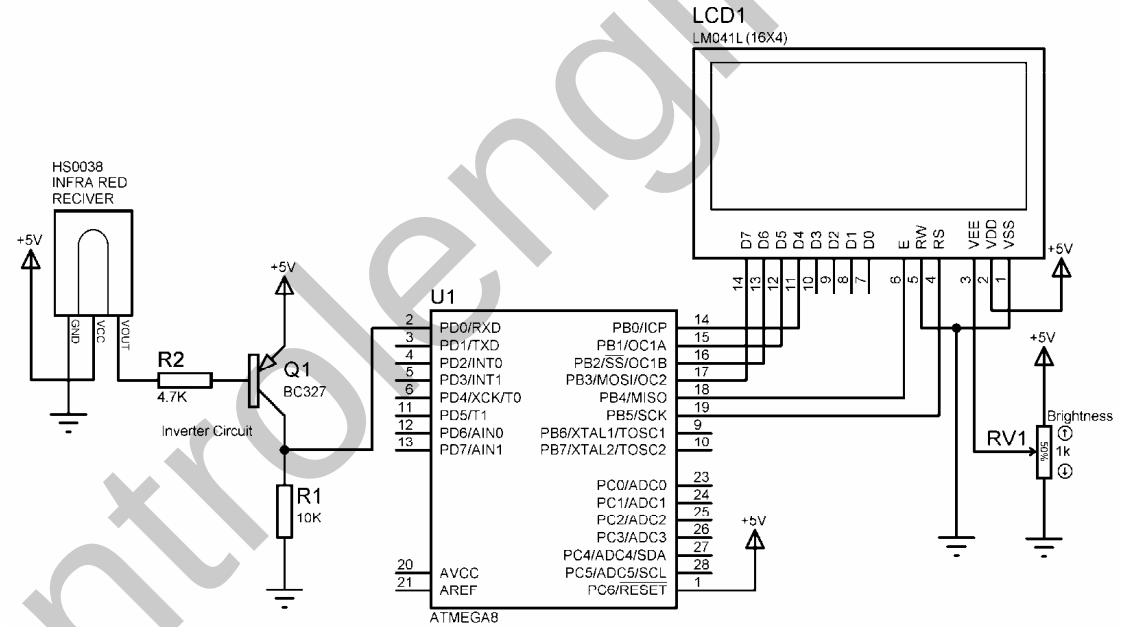
```
If C3 = 1 Then Lcd_data = "A"
If C3 = 2 Then Lcd_data = "B"
If C3 = 3 Then
C3 = 0 : Lcd_data = "C"
End If
'-----------------------------------------------
Case Is = 2
Incr C2
If C2 = 1 Then Lcd_data = "D"
If C2 = 2 Then Lcd_data = "E"
If C2 = 3 Then
C2 = 0 : Lcd_data = "F"
End If
'-----------------------------------------------
Case Is = 1
Incr C1
If C1 = 1 Then Lcd_data = "G"
If C1 = 2 Then Lcd_data = "H"
If C1 = 3 Then
C1 = 0 : Lcd_data = "I"
End If
'-----------------------------------------------
Case Is = 7
Incr C7
If C7 = 1 Then Lcd_data = "G"
If C7 = 2 Then Lcd_data = "H"
If C7 = 3 Then
C7 = 0 : Lcd_data = "I"
End If
'-----------------------------------------------
Case Is = 6
Incr C6
If C6 = 1 Then Lcd_data = "M"
If C6 = 2 Then Lcd_data = "N"
If C6 = 3 Then
C6 = 0 : Lcd_data = "O"
End If
'-----------------------------------------------
Case Is = 5
Incr C5
If C5 = 1 Then Lcd_data = "P"
If C5 = 2 Then Lcd_data = "Q"
If C5 = 3 Then
C5 = 0 : Lcd_data = "R"
End If
'-----------------------------------------------
Case Is = 11
Incr C11
If C11 = 1 Then Lcd_data = "S"
If C11 = 2 Then Lcd_data = "T"
If C11 = 3 Then
C11 = 0 : Lcd_data = "U"
End If
'-----------------------------------------------
Case Is = 10
Incr C10
If C10 = 1 Then Lcd_data = "V"
If C10 = 2 Then Lcd_data = "W"
If C10 = 3 Then
C10 = 0 : Lcd_data = "X"
End If
'-----------------------------------------------
```

```
Case Is = 9
Incr C9
If C9 = 1 Then Lcd_data = "Y"
If C9 = 2 Then Lcd_data = "Z"
If C9 = 3 Then
C9 = 0 : Lcd_data = "."
End If
'START OF LOWERLINE BUTTON PROGRAM-----------------------------------------------------------
Case Is = 15
If S(count) <> Lcd_data Then Lcd_data = " "
Locate Y , X
Lcd Lcd_data
'-------------------------------------------------------------
Incr Count
S(count) = "$"
If Y = 4 Then Goto H2
Lcd_data = " "
If Y = 1 Then X1_save = X
If Y = 2 Then X2_save = X
If Y = 3 Then X3_save = X
Incr Y : X = 1
H2:
'END OF LOWERLINE BUTTON PROGRAM-------------------------------------------------------------
'START OF DELET BUTTON PROGRAM----------------------------------------------------------------
Case Is = 14
S(count) = " "
If Count > 0 Then Decr Count
Decr X
'-------------------------------------------------------------
If Y = 1 Then
If X = 0 Then X = 1
End If
'-------------------------------------------------------------
If Y > 1 Then
If X = 0 Then
If Y = 4 Then X = X3_save
If Y = 3 Then X = X2_save
If Y = 2 Then X = X1_save
Decr Y
End If : End If
'-------------------------------------------------------------
Lcd_data = " "
Locate Y , X
Lcd Lcd_data
'END OF DELET BUTTON PROGRAM------------------------------------------------------------------
'START OF OK BUTTON PROGRAM-------------------------------------------------------------------
Case Is = 13
Incr Count
S(count) = Lcd_data
Lcd_data = " "
'-------------------------------------------------------------
Incr X
If X > 15 Then
If Y = 1 Then X1_save = 15
If Y = 2 Then X2_save = 15
If Y = 3 Then X3_save = 15
If Y < 4 Then
X = 1 : Incr Y
Else
X = 15
End If : End If
'END OF OK BUTTON PROGRAM---------------------------------------------------------------------
```

```
'START OF SEND BUTTON PROGRAM-----------------------------------------------------------------
Case Is = 12
Print "##"
For Send = 1 To Count Step 1
Print S(send)
Next Send
Print "##"
'END OF SEND BUTTON PROGRAM-------------------------------------------------------------------
End Select
'-----------------------------------------------------------------
Locate Y , X
Lcd Lcd_data
Sound Portd.0 , 100 , 80
'-----------------------------------------------------------------
H3:
Recive_data = Getkbd()
If Recive_data <> 16 Then Goto H3
'-----------------------------------------------------------------
Goto H1
' END OF  PROGRAM-------------------------------------------------------------------------------
```

برای اطلاع از نحوه عملکرد برنامه به فصل دوم کتاب مراجعه نمایید .

شکل 18-5 شماتیک مدار گیرنده پیام کوتاه از طریق کانال ارتباطی مادون قرمز را نشان می دهد .



شکل 19-5 شماتیک مدار گیرنده پیام کوتاه

برنامه نوشته شده برای گیرنده پیام کوتاه از طریق کانال ارتباطی مادون قرمز به صورت زیر می باشد .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 4000000
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = Pinb.0, Db5 = Pinb.1 , Db6 = Pinb.2 , Db7 = Pinb.3_
```

```
                    , E = Pinb.4 , Rs = Pinb.5
Dim Lcd_data As String * 2 , X As Byte , Y As Byte , Count As Byte
Dim S As String * 1 , Cls_lcd As Bit
Config Serialin = Buffered , Size = 200
Config Pind.0 = Input
$baud = 2400
Enable Interrupts
'-----------------------------------------------------------------------------------------------------------
Y = 1 : X = 1
Cls : Home
Lcd "UART RECIVER"
'-----------------------------------------------------------------------------------------------------------
Do
Input Lcd_data
S = Mid(lcd_data , 2 , 1)
If S = "#" Then
Cls : Goto Recive
End If
Loop
'-----------------------------------------------------------------------------------------------------------
'RECIVEING MESSAGE OF SERIAL PORT-----------------------------------------------------------------------
Recive:
Input Lcd_data
S = Mid(lcd_data , 2 , 1)
'-----------------------------------------
If Cls_lcd = 1 Then
Cls : Cls_lcd = 0
Y = 1 : X = 1
End If
'-----------------------------------------
Locate Y , X
'-----------------------------------------
If S = "$" Then
Incr Y : X = 1
Goto Recive
End If
'-----------------------------------------
If S = "#" Then
Cls_lcd = 1
Goto Recive
End If
'-----------------------------------------
Lcd S
'-----------------------------------------
Incr X
If X > 15 Then
If Y = 4 Then
Y = 4 : X = 15
Else
X = 1 : Incr Y
End If : End If
'-----------------------------------------
Goto Recive
'END RECIVE-----------------------------------------------------------------------------------------------
```

برای اطلاع از نحوه عملکرد برنامه فوق به فصل دوم کتاب مراجعه نمایید .

(افتخار من این نیست که هرگز شکست نخورده ام افتخار من این است که پس از هر شکست انگیزه ام برای پیروزی دو چندان شده .)

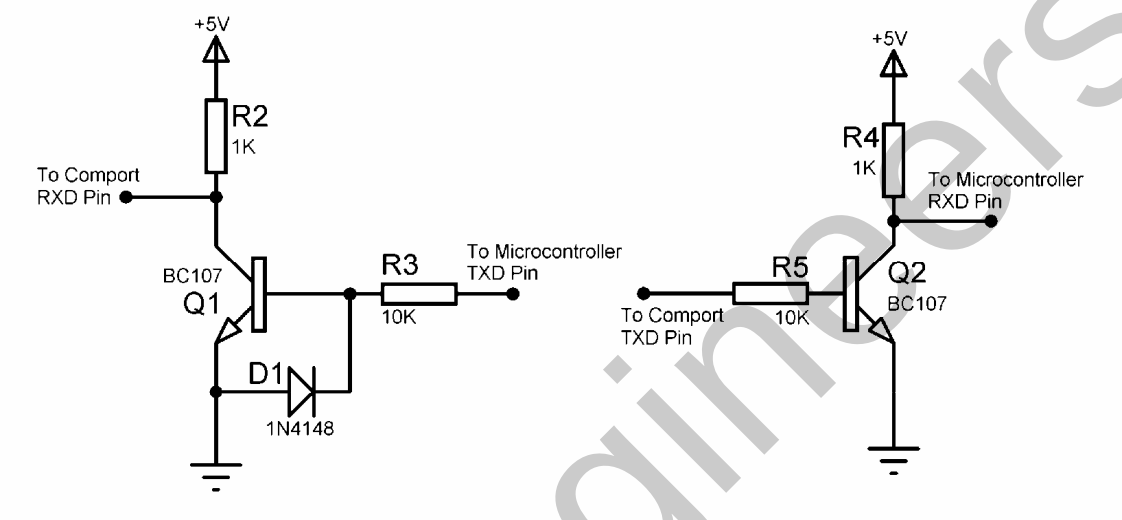
تنها کسانی به موفقیت های بزرگ دست می یابند که توانایی مواجه شدن با شکست های بزرگ را داشته باشند. توماس ادیسون قبل از اختراع لامپ ده هزار بار شکست خورد ولی استدلال او چنین بود که ده هزار بار شکست نخورده بلکه ده هزار راه مختلف یاد گرفته که بوسیله آن ها نمی توان لامپ را اختراع کرد . به خاطر داشته باشید مهم نیست چند بار به زمین می خورید مهم این است که چند بار بر می خیزید .

فصل ششم

AVR در پروژه های حفاظتی و کنترل

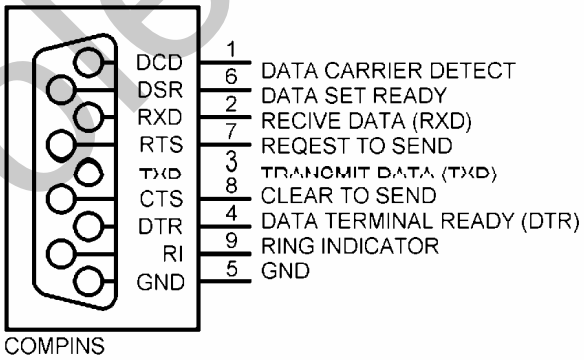
پروژه کنترل وسایل برقی توسط کامپیوتر

با استفاده از پورت سریال میکروکنترلر AVR و پایه های RXD,TXD پورت COM کامپیوتر می توان اطلاعات را به صورت سریال توسط کامپیوتر و میکروکنترلر های AVR رد و بدل کرد ، برای این کار کافی است استاندارد RS-232 را با استاندارد TTL سازگار کنیم ، شما می توانید از مبدل های ترانزیستوری شکل 1-6 برای تبدیل این دو استاندارد به یکدیگر استفاده کنید .



شکل 1-6 مبدل منطق های TTL , RS-232 به یکدیگر

شکل 2-6 آرایش پین های پورت COM کامپیوتر را نشان می دهد .



شکل 2-6 آرایش PIN های پورت COM

برای ارسال و دریافت داده های سریال بین میکروکنترلر و کامپیوتر بایستی از محیط TERMINAL EMULATOR در داخل کامپایلر BASCOM استفاده کنید. برای این منظور پس از برقراری ارتباط سخت افزاری بین میکروکنترلر AVR و پورت COM کامپیوتر محیط TERMINAL EMULATOR را با

CTRL+T بالا بیاورید . برای ارسال اطلاعات از میکروکنترلر به این محیط می توانید از دستور PRINT و برای دریافت اطلاعات از این محیط می توانید از دستور INPUT در برنامه میکروکنترلر استفاده کنید. به عنوان مثال با اجرای دستور

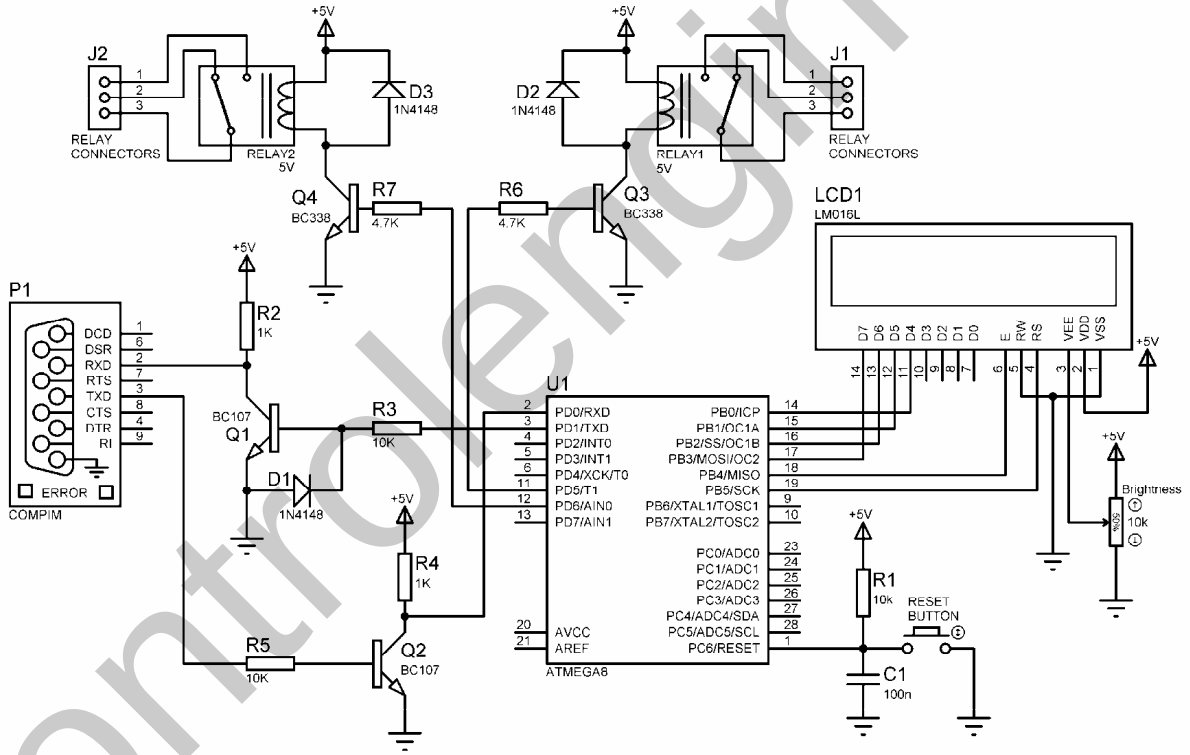
```
PRINT " ENTER YOUR NAME "
```

عبارت ENTER YOUR NAME در محیط TERMINAL نمایش داده می شود و با اجرای دستورات

```
DIM VAR AS STRING * 10
INPUT " NAME IS:";VAR
```

عبارت : NAME IS در محیط TERMINAL نمایش داده می شده و تا زمانی که شما یک نام وارد نکرده و دکمه ENTER را فشار ندهید اجرای برنامه میکروکنترلر در این خط متوقف خواهد شد. به عبارت دیگر دستور فوق تا زمان دریافت شدن یک رشته منتظر می ماند .

فرض کنید می خواهیم دورله 12 ولتی را با استفاده از کامپیوتر SET و RESET کنیم برای این کار ابتدا بایستی مدار شکل 3-6 را ببندیم.



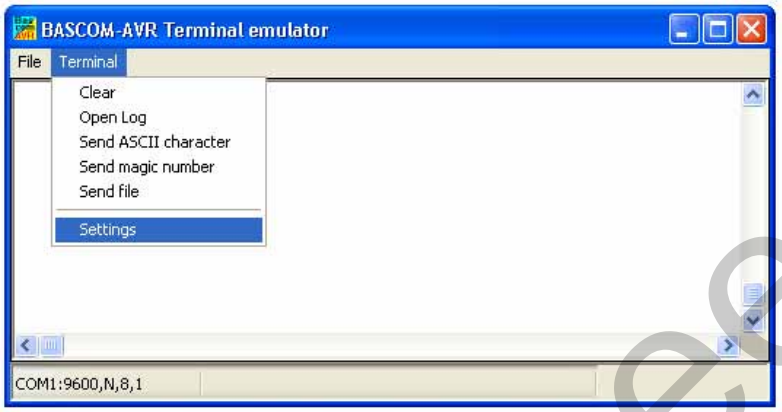
شکل 3-6 شماتیک مدار کنترل رله توسط پورت COM کامپیوتر

سپس برنامه زیر را برای میکروکنترلر نوشته و در آن PROGRAM کنیم .

```
'COMPILER:BASCOM 1.11.8.7
$regfile = "m8def.dat"
$crystal = 8000000
```

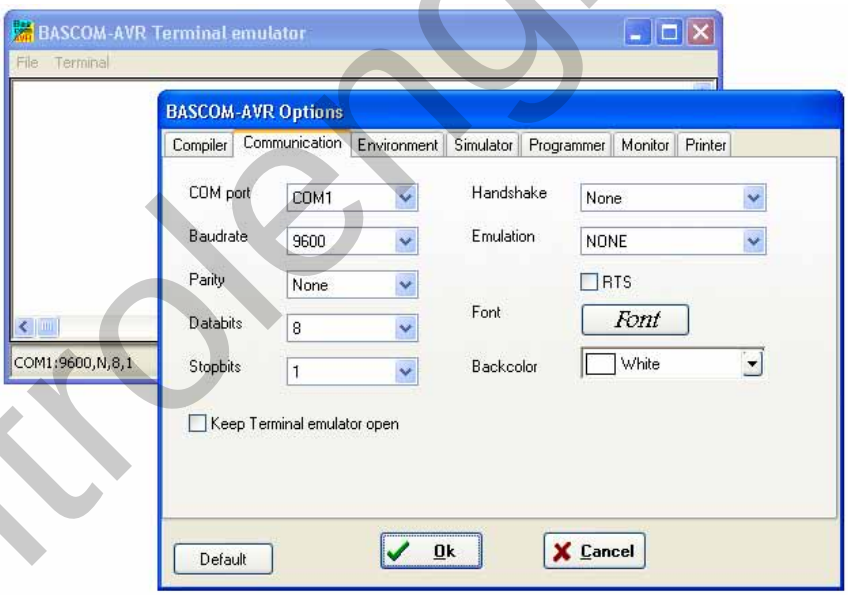
```
$baud = 9600
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.0 , Db5 = Pinc.1 , Db6 = Pinc.2 , Db7 = Pinc.3_
  , E = Pinc.4 , Rs = Pinc.5
Dim Password As String * 10 , Input_value As String * 10
Dim Input_name As String * 10
Config Portd.5 = Output : Relay1 Alias Portd.5
Config Portd.6 = Output : Relay2 Alias Portd.6
Cursor Off
'-----------------------------------------------------------------------------------------------------------
Recive_password:
Cls : Home : Lcd "ENTER PASSWORD"
Input "PLEASE ENTER PASSWORD:" , Password
'-----------------------------------------------------------------------------------------------------------
If Password <> "HOSSEIN.T" Then
Print "INVALID PASSWORD"
Cls : Home : Lcd "PASSWORD WRONG"
Lowerline : Lcd "PLEASE TRY AGAIN"
Print "PLEASE TRY AGAIN"
Goto Recive_password : End If
'-----------------------------------------------------------------------------------------------------------
If Password = "HOSSEIN.T" Then
'-----------------------------------------------------------------------------------------------------------
Cls : Home : Lcd "OK!!"
Lowerline : Lcd "PASSWORD IS TRUE"
'-----------------------------------------------------------------------------------------------------------
Print "VALID PASSWORD"
Input "PLEASE ENTER YOUR NAME:" , Input_name
Print "HELLO " ; Input_name
Print "NOW ENTER RELAY NUMBER"
Do
Input "RELAY NAME IS " , Input_value
'-----------------------------------------------------------------------------------------------------------
Select Case Input_value:
Case Is = "RL1 SET"
Set Relay1
Cls : Home : Lcd "RELAY1"
Lowerline : Lcd "IS SET"

Case Is = "RL1 RESET"
Reset Relay1
Cls : Home : Lcd "RELAY1"
Lowerline : Lcd "IS RESET"

Case Is = "RL2 SET"
Set Relay2
Cls : Home : Lcd "RELAY2"
Lowerline : Lcd "IS SET"

Case Is = "RL2 RESET"
Reset Relay2
Cls : Home : Lcd "RELAY2"
Lowerline : Lcd "IS RESET"
End Select
'-----------------------------------------------------------------------------------------------------------
Loop
End If
'END OF PROGRAM-----------------------------------------------------------------------------------------
```

پس از PROGRAM کردن برنامه فوق در میکروکنترلر ATMEGA8 وارد محیط BASCOM شده و TERMINAL EMULATOR را با CTRL+T بالا بیاورید . با انجام این کار پنجره شکل 6-4 در محیط BASCOM باز می شود .



شکل 6-4 محیط TERMINAL EMULATOR

در پنجره فوق از منوی TERMINAL گزینه SETTING را انتخاب کنید در این حالت شکل 6-5 جهت انجام تنظیمات باز می شود .



شکل 6-5 پنجره AVR OPTIONS

در قسمت COMMUNICATION پنجره فوق تنظیمات لازم همانند شکل 6-5 انجام داده سپس دکمه OK را کلیک کنید ، پس از کلیک کردن بر روی دکمه OK ابتدا با فشار RESET BUTTON در مدار میکروکنترلر ، ATMEGA8 را RESET کنید . پس از RESET میکرو عبارت PLEASE ENTER PASSWORD در

محیط TERMINAL نوشته می شود و کلمه PASSWORD از شما درخواست می شود شما بایستی کلمه ای با طول 10 کاراکتر را توسط کی بورد کامپیوتر وارد کرده و دکمه ENTER را فشار دهید PASSWORD تعریف شده برای پروژه فوق HOSSEIN.T می باشد شما می توانید هر PASSWORD دیگری را در برنامه تعریف کنید پس از وارد کردن PASSWORD و زدن کلید ENTER اگر PASSWORD صحیح نباشد عبارت

INVALID PASSWORD
PLEASE TRY AGAIN

در محیط TERMINAL ظاهر شده و همچنین عبارت

PASSWORD WRONG
PLEASE TRY AGAIN

بر روی LCD (16*2) نمایش داده می شود و سپس با نوشته شدن عبارت

PLEASE ENTER PASSWORD :

در محیط TERMINAL ، دوباره PASSWORD جدید از شما درخواست می شود ، اما اگر PASSWORD وارد شده صحیح باشد عبارت

OK
PASSWORD IS TRUE

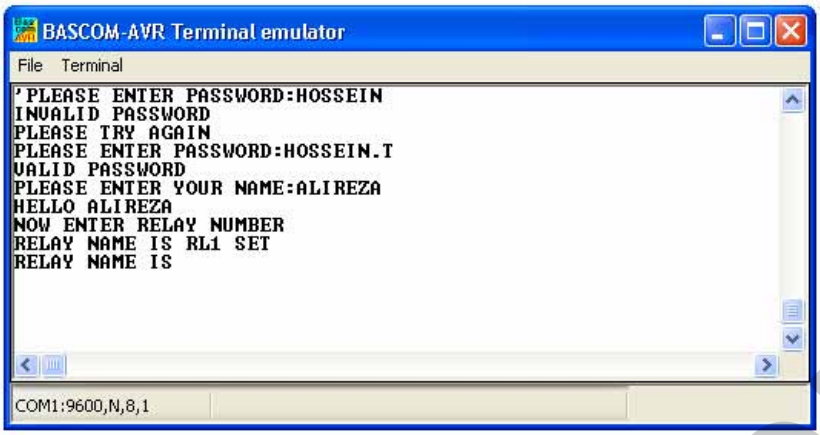
بر روی LCD مدار نوشته شده و عبارت

VALID PASSWORD
PLEASE ENTER YOUR NAME :

در محیط TERMINAL نوشته می شود . در این حالت شما بایستی نام خود را وارد کرده و کلید ENTER را فشار دهید. به عنوان مثال اگر نام وارد شده ALIREZA باشد عبارت زیر در محیط TERMINAL نوشته می شود .

HELLO ALIREZA
NOW ENTER RELAY NUMBER
REALY NAME IS :

در این حالت شما می توانید با وارد کردن RL1 SET/RESET و یا RL2 SET/RESET رله های 1,2 را کنترل کنید . به عنوان مثال برای روشن کردن رله شماره یک بایستی عبارت RL1 SET را وارد کنید . شکل 6-6 انجام عملیات فوق را در محیط TERMINAL EMULATOR نشان می دهد .



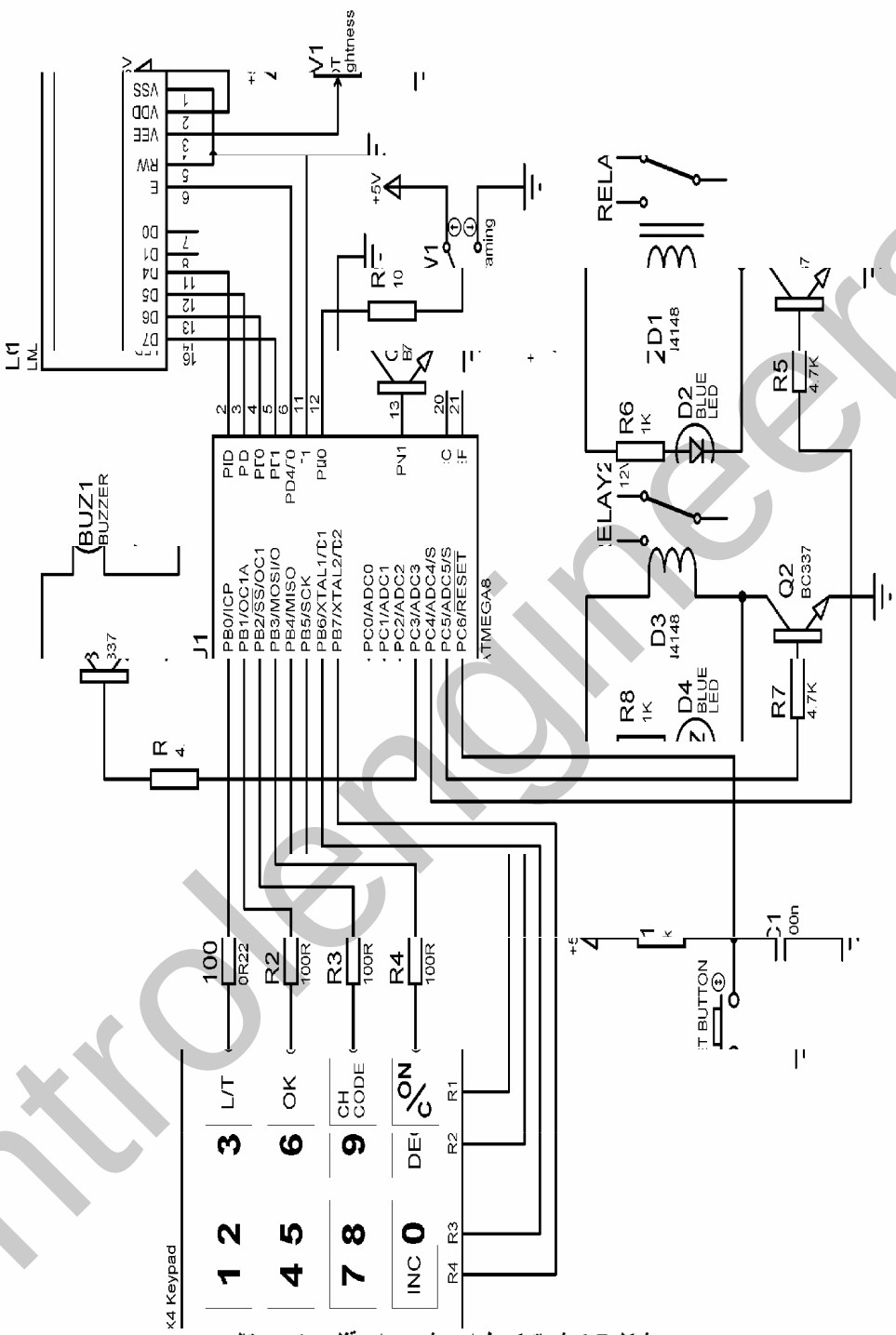
شکل 6-6 نمونه ای از کار با محیط TERMINAL EMULATOR

توجه داشته باشید که برای پاک کردن صفحه TERMINAL EMULATOR بایستی گزینه CLEAR را از منوی TERMINAL انتخاب کنید .

طراحی یک قفل رمز دیجیتالی توسط میکروکنترلر ATMEGA8

در این قسمت قصد داریم یک قفل رمز دیجیتال 4 رقمی با قابلیت تعویض کد طراحی کنیم این پروژه دارای دو رله خروجی می باشد که با استفاده از کد 4 رقمی تعریف شده برای دستگاه کنترل می شود . رله های مدار می توانند در دو حالت لحظه ای و LATCH توسط کد تعریف شده کنترل شوند ، برای تنظیم حالت تحریک رله ها بایستی ابتدا کلید SW1 را در حالت PROGRAMING قرار داده سپس میکروکنترلر را RESET کنیم در این حالت دستگاه وارد قسمت برنامه ریزی می شود در این قسمت شما می توانید با استفاده از کلید های OK,L/T، LATCH یا لحظه ای (TIMER) بدون حالت تحریک رله های خروجی را تنظیم کنید ، بدین صورت که با هر بار فشار کلید L/T عبارت نوشته شده بر روی LCD به صورت های RELAYS ARE TIMER ، RELAYS ARE LATCHER تغییر می کند و شما می توانید با فشار کلید OK یکی از گزینه ها را انتخاب کنید . اگر حالت تایمر را انتخاب کنید DELAY TIME نیز از شما درخواست می شود . برای وارد کردن DELAY TIME می توانید از کلید های DEC,INC و OK استفاده کنید . برای مثال در حالت تایمر اگر DELAY TIME را برابر با 500ms انتخاب کنید ، پس از وارد کردن کد صحیح هر دو رله دستگاه به مدت 500ms روشن شده سپس خاموش می شوند ولی در حالت LATCHER با یک بار وارد کردن PASSWORD صحیح رله ها وصل شده و با وارد کردن آن برای بار دوم هر دو رله خاموش می شوند . توجه داشته باشید که پس از انجام تنظیمات لازم در قسمت برنامه ریزی برای جلوگیری از وارد شدن مجدد دستگاه به حالت برنامه ریزی پس از قطع تغذیه یا RESET شدن میکروکنترلر، کلید SW1 را از حالت PROGRAMING خارج کنید .

شکل 7-6 سخت افزار طراحی شده برای پروژه را نشان می دهد .



شکل 7-6 شماتیک طراحی شده برای قفل رمز دیجیتال

نکته جدید استفاده شده در شماتیک فوق استفاده از LED داخلیLCD می باشد که برای سوئیچ کردن آن از ترانزیستور Q4 استفاده شده است .

کد پیش فرض تعریف شده برای دستگاه 2222 می باشد برای تعویض کردن آن بایستی کلید CHANGE CODE را فشار دهید . پس از فشار این کلید عبارت

ARE YOU SURE?
YES=9 NO=8

بر روی LCD نوشته می شود در این قسمت شما بایستی برای تعویض کد کلید 9 را فشار دهید اگر از تعویض کد پشیمان شدید کلید 8 را فشار دهید ، پس از فشار کلید 9 عبارت

REPEAT ACCESS
CODE

بر روی LCD نوشته می شود در این قسمت شما بایستی کد فعلی را وارد کنید در صورتی که کد فعلی وارد شده صحیح نباشد تعویض کد دستگاه امکان پذیر نخواهد بود . اگر برای اولین بار می خواهید کد را تعویض کنید . با توجه به این که کد پیش فرض تعریف شده برای دستگاه 2222 می باشد در این قسمت کد 2222 را وارد کنید ، پس از این که کد وارد شده مورد تایید قرار گرفت کد جدید به صورت زیر از کاربر درخواست می شود .

ENTER NEW
CODE

در این قسمت شما بایستی کد جدید را وارد کنید ، پس از وارد کردن کد جدید ، این کد به عنوان کد دستگاه در EEPROM میکروکنترلر نوشته شده و شما برای کنترل رله های مدار بایستی کد جدید را وارد کنید .

برنامه نوشته شده برای پروژه به صورت زیر است .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
                    , E = Pind.4 , Rs = Pind.5
Config Pind.6 = Input : Program_sw Alias Pind.6
Config Pinc.3 = Output : Speaker Alias Portc.3
Config Pind.7 = Output : Lcd_light Alias Portd.7
Config Pinc.4 = Output : Relay1 Alias Portc.4
Config Pinc.5 = Output : Relay2 Alias Portc.5
Dim Code1 As Eram Byte , Code2 As Eram Byte , Code3 As Eram Byte
Dim Code4 As Eram Byte , Latch As Eram Byte , Delay_time As Word
Dim Password1 As Byte , Password2 As Byte , Password3 As Byte
Dim Password4 As Byte , Write_to_e2 As Eram Byte , Keypad_data As Byte
Dim Input_counter As Byte , H As Bit , Ovf_counter As Byte
Dim Delay_time_save As Eram Word
Config Kbd = Portb , Debounce = 20 , Delay = 100
'TIMER0 CONFIGURATION-----------------------------------------------------------------------------
Config Timer0 = Timer , Prescale = 1024
'OVERFLOW TIME=150/(8000000*256*1024)
Enable Interrupts : Enable Timer0
On Ovf0 Owerflow
```

```
'-----------------------------------------------------------------------------------------------------------------------
Start_program:
If Write_to_e2 = 1 Then Goto Programing
Code1 = 2 : Waitms 10
Code2 = 2 : Waitms 10
Code3 = 2 : Waitms 10
Code4 = 2 : Waitms 10
'START OF LATCHER/TIMER PROGRAM-----------------------------------------------------------------------
Programing:
If Program_sw = 1 Then Goto Input_password
Set Lcd_light
Cursor Off : Cls : Home
Lcd "IN THE NAME OF"
Locate 2 , 7 : Lcd "GOD"
Wait 4 : Cls : Home
Lcd "RELAYS ARE"
Lowerline : Lcd "LATCHER"
'-----------------------------------------------------------------------------------------------------------------------
Input_keypad1:
Keypad_data = Getkbd()
If Keypad_data > 15 Then Goto Input_keypad1
Keypad_data = Lookup(keypad_data , Data_code)
'-----------------------------------------------------------------------------
Select Case Keypad_data:
'-----------------------------------------------------------------------------
Case Is = 10 : H = Not H
If H = 0 Then
Cls : Home : Lcd "RELAYS ARE"
Lowerline : Lcd "LATCHER"
Latch = 0 : Waitms 10 : End If

If H = 1 Then
Cls : Home : Lcd "RELAYS ARE"
Lowerline : Lcd "TIMER"
Latch = 1 : Waitms 10 : End If
'-----------------------------------------------------------------------------
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
Sound Speaker , 100 , 150
'-----------------------------------------------------------------------------
Case Is = 20
If Latch = 0 Then
Cls : Home : Lcd "OK!! RELAYS"
Lowerline : Lcd "ARE LATCHER"
'-----------------------------------------------------------------------------
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
Sound Speaker , 1000 , 150 : Wait 4
'-----------------------------------------------------------------------------
Goto Input_password : End If

If Latch = 1 Then
Cls : Home : Lcd "OK!! RELAYS"
Lowerline : Lcd "ARE TIMER"
'-----------------------------------------------------------------------------
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
Sound Speaker , 1000 , 150 : Wait 4
'-----------------------------------------------------------------------------
Goto Input_time : End If
'-----------------------------------------------------------------------------
End Select
```

```
'-----------------------------------------------------------------------
Waitms 200
Goto Input_keypad1
'END OF LATCHER/TIMER PROGRAM-----------------------------------------------------------------------
'START OF DELAY TIME PROGRAM-------------------------------------------------------------------------
Input_time:
Cls : Home : Lcd "ENTER DELAY TIME"
Lowerline : Lcd "TIME IS " ; Delay_time ; "mS"
'-----------------------------------------------------------------------
Input_keypad2:
Keypad_data = Getkbd()
If Keypad_data > 15 Then Goto Input_keypad2
Keypad_data = Lookup(keypad_data , Data_code)
'-----------------------------------------------------------------------
Select Case Keypad_data:
'-----------------------------------------------------------------------
Case Is = 40
If Delay_time < 65530 Then Delay_time = Delay_time + 10
Waitms 10
'-----------------------------------------------------------------------
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
'-----------------------------------------------------------------------
Sound Speaker , 100 , 150
Case Is = 50
If Delay_time >= 10 Then Delay_time = Delay_time - 10
Waitms 10
'-----------------------------------------------------------------------
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
'-----------------------------------------------------------------------
Sound Speaker , 100 , 150
Case Is = 20
Cls : Home : Lcd "OK DELAY TIME"
Lowerline : Lcd "IS " ; Delay_time ; "mS"
'-----------------------------------------------------------------------
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
'-----------------------------------------------------------------------
Sound Speaker , 1000 , 150
Delay_time_save = Delay_time
Wait 4 : Goto Input_password
'-----------------------------------------------------------------------
End Select
'-----------------------------------------------------------------------
'Waitms 200
Goto Input_time


'END OF DELAY TIME PROGRAM---------------------------------------------------------------------------
'START OF INPUT PASSWORD PROGRAM---------------------------------------------------------------------
Input_password:
Cursor Off : Cls
Home : Lcd "ENTER ACCESS"
Lowerline : Lcd "CODE"
'-----------------------------------------------------------------------
Input_keypad3:
Keypad_data = Getkbd()
If Keypad_data > 15 Then Goto Input_keypad3
Keypad_data = Lookup(keypad_data , Data_code)
If Keypad_data = 30 Then Goto Change_code
If Keypad_data >= 10 Then Goto Input_keypad3
```

```
'-----------------------------------------------------------------------
Incr Input_counter
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
'-----------------------------------------------------------------------
Select Case Input_counter:
'-----------------------------------------------------------------------
Case Is = 1 : Password1 = Keypad_data
Cls : Home : Lcd "ENTER ACCESS"
Lowerline : Lcd "CODE  " ; "*"

Case Is = 2 : Password2 = Keypad_data
Cls : Home : Lcd "ENTER ACCESS"
Lowerline : Lcd "CODE  " ; "*" ; "*"

Case Is = 3 : Password3 = Keypad_data
Cls : Home : Lcd "ENTER ACCESS"
Lowerline : Lcd "CODE  " ; "*" ; "*" ; "*"

Case Is = 4 : Password4 = Keypad_data
Cls : Home : Lcd "ENTER ACCESS"
Lowerline : Lcd "CODE  " ; "*" ; "*" ; "*" ; "*"
Input_counter = 0
Goto Compare_match
'-----------------------------------------------------------------------
End Select
'-----------------------------------------------------------------------
Sound Speaker , 100 , 150
Waitms 200
Goto Input_keypad3
'END OF INPUT PASSWORD PROGRAM-----------------------------------------------------------------
'START OF COMPARE MATCH PROGRAM-----------------------------------------------------------------
Compare_match:
If Password1 <> Code1 Then Goto Error
If Password2 <> Code2 Then Goto Error
If Password3 <> Code3 Then Goto Error
If Password4 <> Code4 Then Goto Error
Sound Speaker , 1000 , 150
Cls : Home : Lcd "OK!! PASSWORD"
Lowerline : Lcd "IS TRUE"
'-----------------------------------------------------------------------
If Latch = 0 Then
Toggle Relay1 : Toggle Relay2 : End If
'-----------------------------------------------------------------------
If Latch = 1 Then
Set Relay1 : Set Relay2
Delay_time = Delay_time_save : Waitms 10
Waitms Delay_time
Reset Relay1 : Reset Relay2 : End If
'-----------------------------------------------------------------------
Wait 4
Goto Input_password
'END OF COMPARE MATCH PROGRAM-----------------------------------------------------------------
'START OF ERROR PROGRAM-----------------------------------------------------------------
Error:
Cls : Home : Lcd "PASSWORD WRONG"
Lowerline : Lcd "PLEASE TRY AGAIN"
Sound Speaker , 1000 , 150
Wait 4 : Goto Input_password
'END OF ERROR PROGRAM-----------------------------------------------------------------
'START OF CHANGE CODE PROGRAM-----------------------------------------------------------------
Change_code:
```

```
Cls : Home : Lcd "ARE YOU SURE"
Lowerline : Lcd "YES=9 NO=8"
'-----------------------------------------------------------------------
Input_keypad4:
Keypad_data = Getkbd()
If Keypad_data > 15 Then Goto Input_keypad4
Keypad_data = Lookup(keypad_data , Data_code)
If Keypad_data > 10 Then Goto Input_keypad4
'-----------------------------------------------------------------------
Select Case Keypad_data:
'-----------------------------------------------------------------------
Case Is = 9
'-----------------------------------------------------------------------
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
'-----------------------------------------------------------------------
Sound Speaker , 1000 , 150
Goto H1

Case Is = 8
'-----------------------------------------------------------------------
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
'-----------------------------------------------------------------------
Input_counter = 0
Sound Speaker , 1000 , 150
Goto Input_password
'-----------------------------------------------------------------------
End Select
'-----------------------------------------------------------------------
Waitms 200
Goto Input_keypad4
'-----------------------------------------------------------------------
H1:
Cls : Home : Lcd "REPEAT ACCESS"
Lowerline : Lcd "CODE"
Input_keypad5:
Keypad_data = Getkbd()
If Keypad_data > 15 Then Goto Input_keypad5
Keypad_data = Lookup(keypad_data , Data_code)
If Keypad_data > 10 Then Goto Input_keypad5
'-----------------------------------------------------------------------
Incr Input_counter
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
'-----------------------------------------------------------------------
Select Case Input_counter:
'-----------------------------------------------------------------------
Case Is = 1 : Password1 = Keypad_data
Cls : Home : Lcd "ENTER ACCESS"
Lowerline : Lcd "CODE  " ; "*"

Case Is = 2 : Password2 = Keypad_data
Cls : Home : Lcd "ENTER ACCESS"
Lowerline : Lcd "CODE  " ; "*" ; "*"

Case Is = 3 : Password3 = Keypad_data
Cls : Home : Lcd "ENTER ACCESS"
Lowerline : Lcd "CODE  " ; "*" ; "*" ; "*"

Case Is = 4 : Password4 = Keypad_data
Cls : Home : Lcd "ENTER ACCESS"
```

```
Lowerline : Lcd "CODE  " ; "*" ; "*" ; "*" ; "*"
Input_counter = 0
If Password1 <> Code1 Then Goto Error2
If Password2 <> Code2 Then Goto Error2
If Password3 <> Code3 Then Goto Error2
If Password4 <> Code4 Then Goto Error2
Sound Speaker , 1000 , 150
Cls : Home : Lcd "OK!! PASSWORD"
Lowerline : Lcd "IS TRUE"
Wait 4 : Goto Enter_new_code
'-----------------------------------------------------------------------
End Select
'-----------------------------------------------------------------------
Sound Speaker , 100 , 150
Waitms 200
Goto Input_keypad5
'START OF ERROR2 PROGRAM-----------------------------------------------------------------------------------------
Error2:
Sound Speaker , 1000 , 150
Cls : Home : Lcd "SURRY.YOU CAN"
Lowerline : Lcd "NOT CHANGE CODE"
Wait 4 : Goto Input_password
'END OF ERROR2 PROGRAM-------------------------------------------------------------------------------------------
Enter_new_code:
Cls : Home : Lcd "ENTER NEW"
Lowerline : Lcd "CODE"
Input_keypad6:
Keypad_data = Getkbd()
If Keypad_data > 15 Then Goto Input_keypad6
Keypad_data = Lookup(keypad_data , Data_code)
If Keypad_data > 10 Then Goto Input_keypad6
'-----------------------------------------------------------------------
Incr Input_counter
Stop Timer0 : Timer0 = 0 : Ovf_counter = 0
Start Timer0 : Set Lcd_light
'-----------------------------------------------------------------------
Select Case Input_counter:
'-----------------------------------------------------------------------
Case Is = 1 : Password1 = Keypad_data
Cls : Home : Lcd "ENTER NEW"
Lowerline : Lcd "CODE  " ; Password1

Case Is = 2 : Password2 = Keypad_data
Cls : Home : Lcd "ENTER NEW"
Lowerline : Lcd "CODE  " ; Password1 ; Password2

Case Is = 3 : Password3 = Keypad_data
Cls : Home : Lcd "ENTER NEW"
Lowerline : Lcd "CODE  " ; Password1 ; Password2 ; Password3

Case Is = 4 : Password4 = Keypad_data
Cls : Home : Lcd "ENTER NEW"
Lowerline : Lcd "CODE  " ; Password1 ; Password2 ; Password3 ; Password4
Input_counter = 0
Code1 = Password1 : Waitms 10
Code2 = Password2 : Waitms 10
Code3 = Password3 : Waitms 10
Code4 = Password4 : Waitms 10
Write_to_e2 = 1
Sound Speaker , 1000 , 150
Wait 4 : Goto Input_password
'-----------------------------------------------------------------------
```

```
End Select
'-----------------------------------------------------------------------------
Sound Speaker , 100 , 150
Waitms 200
Goto Input_keypad6
'END OF CHANGE CODE PROGRAM-----------------------------------------------------------
'START OF DATA CODE LOOKUP-------------------------------------------------------------
Data_code:
Data 40 , 0 , 50 , 60 , 7 , 8 , 9 , 30 , 4 , 5 , 6 , 20 , 1 , 2 , 3 , 10
'END OF DATA CODE LOOKUP--------------------------------------------------------------
'START OF OWERFLOW PROGRAM------------------------------------------------------------
Owerflow:
Incr Ovf_counter
If Ovf_counter > 150 Then
Ovf_counter = 0 : Stop Timer0
Reset Lcd_light : Timer0 = 0
End If
Return
'END OF OWERFLOW PROGRAM---------------------------------------------------------------
```

تشریح نحوه عملکرد برنامه :

از تایمر 2 برای کنترل مدت زمان روشن بودن LED داخلی LCD استفاده شده است . بدین صورت که با فشار هر کلید LED داخلی روشن شده پس از این که تایمر 2 ، 150 بار سرریز شد ، خاموش می شود . اگر قبل از پایان مدت زمان تنظیم شده توسط تایمر 2 برای روشن ماندن LED داخلی LCD کلید دیگری فشار داده شود مدت زمان شمارش شده صفر شده و از نو شروع به شمارش می کند و پس از 150 بار سریز شدن LED داخلی را خاموش می کند .

در آغاز برنامه فوق ابتدا متغیر WRITE_TO_E2 چک می شود اگر برابر با 1 نباشد عدد 2 در متغیر های CODE4,CODE3,CODE2,CODE1 نوشته می شود این متغیر ها از نوع ERAM BYTE بوده و محتوای آن ها در EEPROM داخلی ATMEGA8 ذخیره خواهد شد . با توجه به این که مقدار اولیه متغیر WRITE_TO_E2 (قبل از مقدار دهی در برنامه) صفر می باشد پس اجرای برنامه به لیبل PROGRAMING منتقل نشده و عدد2222 به عنوان کد دستگاه در EEPROM ذخیره خواهد شد . توجه داشته باشید که متغیر WRITE_TO_E2 نیز ازنوع ERAM BYTE بوده و محتوای آن در EEPROM ذخیره خواهد شد . اگر کد دستگاه تعویض شود محتوای WRITE_TO_E2 نیز 1 خواهد شد بنابراین دیگر عدد2222 در EEPROM نوشته نخواهد شد در برچسب PROGRAMING ابتدا کلید SW1 توسط دستور

```
IF PROGRAM_SW=1 THEN GOTO INPUT_PASSWORD
```

خوانده شده و اگر برابر با یک بود یعنی در حالت PROGRAMING قرار نداشت اجرای برنامه به بر چسب INPUT_PASSWORD منتقل می شود ، که در این بر چسب کد دستگاه از کاربر درخواست خواهد شد اگر کلید SW1 در حالت PROGRAMING بوده و برابر با صفر منطقی باشد ، دستگاه وارد قسمت برنامه ریزی می

شود. در این قسمت شما بایستی LATCHER یا TIMER بودن حالت تحریک رله ها را تعیین کرده و اگر گزینه TIMERرا انتخاب کرده باشید بایستی مدت زمان تاخیر (DELAY TIME) را نیز وارد کنید ، پس از وارد کردن اطلاعات فوق کلید SW1 را از حالت PROGRAMING خارج کنید . پس از انجام تنظیمات لازم در قسمت برنامه ریزی اجرای برنامه به بر چسب INPUT_PASSWORD منتقل می شود . در این بر چسب ابتدا عبارت

ENTER ACCESS
CODE

بر روی LCD نوشته شده، سپس اجرای برنامه در حلقه INPUT_KEYPAD قرار می گیرد . و تا زمانی که شما 4 عدد را وارد نکرده اید یا کلید CHENGE_CODE را فشار نداده اید اجرای برنامه از این حلقه خارج نخواهد شد پس از وارد کردن 4 عدد اجرای برنامه به برچسب COMPARE_MATCH منتقل می شود. در این بر چسب ابتدا 4 عدد وارد شده با کد دستگاه مقایسه می شود اگر تطابق مقایسه صورت نگیرد اجرای برنامه به لیبل ERROR منتقل خواهد شد که در این لیبل ابتدا عبارت

PASSWORD WRONG
PLEASE TRY AGAIN

بر روی LCD نوشته شده سپس اجرای برنامه به برچسب INPUT_PASSWORD منتقل می شود که در آن جا با نوشتن عبارت

ENTER ACCESS
CODE

کد دستگاه از کاربر درخواست می شود ، اما اگر 4 عدد وارد شده با کد دستگاه مطابقت داشته باشد ، اجرای برنامه در برچسب COMPARE_MATCH باقی مانده و عبارت

OK PASSWORD
IS TRUE

بر روی LCD نوشته می شود ، سپس مقدار متغیر LATCH بررسی می شود .توجه داشته باشید که اگر در قسمت برنامه ریزی حالت تحریک رله ها را TIMER تعریف کرده باشید محتوای متغیر LATCH برابر با 1 و اگر LATCHER تعریف کرده باشید مقدار متغیر LATCH برابر با صفر خواهد بود ، در ضمن مقدار متغیر LATCH از نوع ERAM BYTE بوده و محتوای آن در EEPROM داخلی ATMEGA 8 ذخیره خواهد شد .

پس از بررسی محتوای متغیر LATCH اگر محتوای آن برابر با صفر باشد وضعیت هر دو رله معکوس می شود ولی اگر محتوای متغیر LATCH یک باشد ابتدا هر دو رله روشن شده و پس از تاخیر ی که مدت زمان آن در قسمت برنامه ریزی تعیین شده است هر دو رله خاموش می شوند. سپس اجرای برنامه به بر چسب

INPUT_PASSWORD منتقل می شود. یعنی باز هم کد دستگاه از کاربر خواسته می شود . هنگامی که کد دستگاه از کاربر درخواست می شود اگر به جای وارد کردن کد کلید CHANGECODE را فشار دهید اجرای برنامه به برچسب CHANGE_CODE منتقل می شود . در این قسمت عبارت

ARE YOU SURE
YES=9 NO=8

بر روی LCD نوشته می شود ، در این حالت با فشار کلید 9 اجرای برنامه به برچسب H1 و با فشار کلید 8 اجرای برنامه به برچسب INPUT_PASSWORD منتقل می شود و در لیبل H1 ابتدا عبارت

REPEAT ACCESS
CODE

بر روی LCD نوشته می شود که در این حالت کاربر بایستی کد فعلی دستگاه را وارد کند پس از وارد کردن چهارمین عدد ، کد وارد شده با کد دستگاه مقایسه می شود ، اگر تطابق مقایسه صورت نگیرد اجرای برنامه به برچسب ERROR2 منتقل می شود که در این بر چسب پس از نوشته شدن عبارت

SURRY YOU CAN
NOT CHANGE CODE

بر روی LCD ، اجرای برنامه به بر چسب INPUT_PASSWORD منتقل می شود و اگر تطابق مقایسه صورت بگیرد اجرای برنامه به برچسب ENTER_NEW_CODE منتقل می شود در این برچسب عبارت

ENTER NEW
CODE

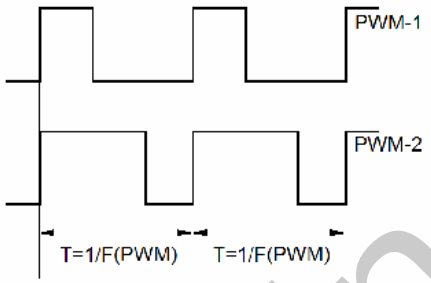
بر روی LCD نمایش داده می شود که در این حالت کد وارد شده به عنوان کد دستگاه در نظر گرفته خواهد شد ، پس از وارد کردن چهارمین عدد اجرای برنامه به لیبل INPUT_PASSWORD متقل خواهد شد .

نکته مهمی که در این برنامه بایستی مورد توجه قرار گیرد ، جدول DATA_CODE برنامه می باشد با توجه به این که شکل ظاهری KEYPAD های موجود در بازار با هم متفاوت است شما بایستی ابتدا اعداد برگردانده شده توسط دستور GETKBD() را با استفاده از روشی که در فصل دوم کتاب توضیح داده شده است شناسائی کرده و جدول DATA_CODE را بر اساس آن تنظیم کنید توجه داشته باشید جدول DATA_CODE بایستی طوری تنظیم شود که با زدن کلید L/T عدد10 ، با زدن کلید OK عدد20 ، با زدن کلید INC عدد 40 ، بازدن کلید DEC عدد 50 و با زدن کلید CHANGE CODE عدد 30 برگردانده شود .

پروژه کنترل سرعت موتور DC با استفاده از پالس PWM

PWM مخفف کلمه لاتین PULSE WITH MODULATION است در این روش هدف کنترل سرعت موتور با استفاده از دریافت پالس یا سیگنال است با استفاده از این روش می توان سرعت موتور در حال حرکت را کم یا

زیاد کرد. موتور های DC در اشکال ، اندازه و مشخصات مختلفی در بازار یافت می شوند که به تبع آن درایور مربوط به سرعت آن ها نیز متفاوت است ، به طور کلی سرعت دور یا چرخش یک موتور DC وابسته به تغذیه آن می باشد به عنوان مثال اگر موتوری با ولتاژ نامی 12 ولت را به تغذیه 12 ولتی متصل کرده سپس ولتاژ تغذیه را تا مقدار 6 ولت پائین بیاورید. سرعت چرخشی آن نصف حالت قبلی خواهد بود. در حالت PWM میانگین ولتاژ های فرستاده شده توسط مدار درایور ، سرعت موتور را تنظیم می کنند. شکل 8-6 را در نظر بگیرید سرعت موتور DC هنگام اعمالPWM-2 دو برابر بیشتر از زمانی است که سیگنال PWM_1 به آن اعمال می شود.

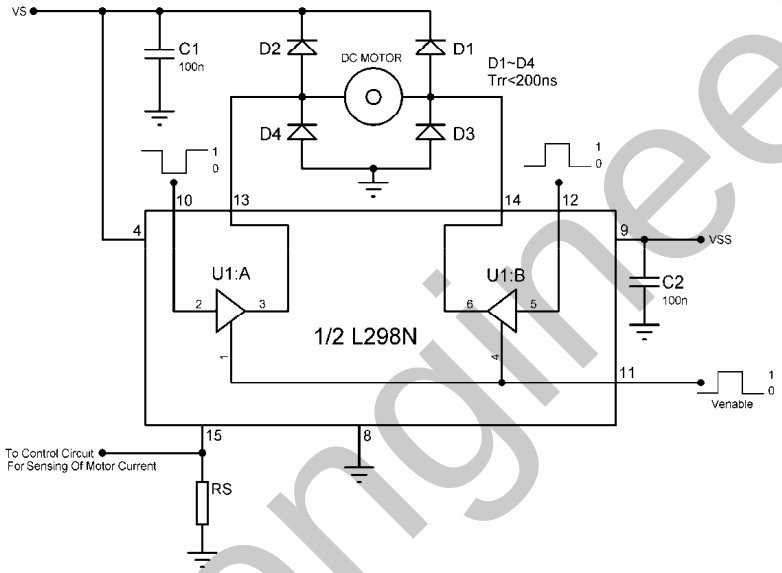


شکل 8-6 دو سیگنال PWM با ضریب چسبندگی متفاوت

برای درک بهتر موضوع به این مثال توجه کنید. هنگامی که یک فیلم را مشاهده می کنید در واقع شما شاهد هزاران عکس ثابت هستید که با یک فرکانس بالا به شما نشان داده می شوند ، سرعت پخش شدن عکس ها آنقدر زیاد است که مغز شما قادر به تشخیص فواصل زمانی بین پخش شدن و عدم پخشی شدن آن ها نمی باشد در واقع مغز شما میانگین این عکس ها را مشاهده می کند .

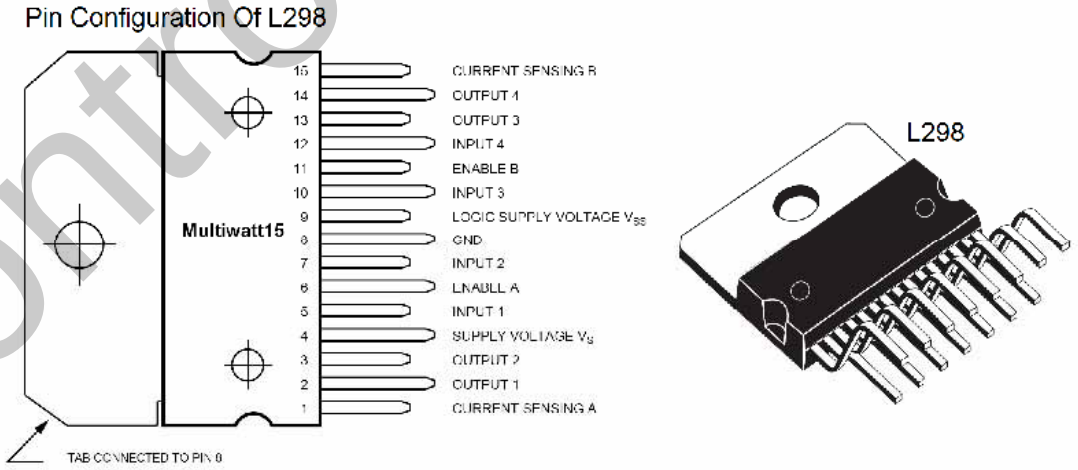
در کنترل PWM نیز همین وضعیت وجود دارد سرعت روشن و خاموش شدن آنقدر زیاد است (حدودا 250Hz) که شما متوجه آن نمی شوید ، هرچه فرکانس PWM بالاتر باشد موتور سریعتر روشن و خاموش می شود و هر چه فرکانس PWM کمتر باشد فواصل زمانی بین روشن وخاموش شدن موتور کمتر می شود ولی به طور معمول فرکانس سیگنال PWM را برای کنترل موتور های DC در بازه 200 تا 300 هرتز انتخاب می کنند. البته برای کنترل یک موتور DC علاوه بر مدار تولید کننده PWM به یک IC راه انداز موتور نیز نیازمندیم. در این پروژه از IC راه انداز موتور L298N استفاده شده است. از این IC می توان برای راه اندازی دو عدد موتور DC در دو جهت استفاده نمود ولتاژ کاری موتور های DC می تواند حداکثر 50 ولت باشد همچنین این IC قادر است تا 2 آمپرجریان را برای بار تامین کند از مزایای دیگر این IC وجود پایه حسگر جریان (CURRENT SENSING) در آن می باشد ، شما می توانید این پایه را با استفاده از مقاومت RS به زمین وصل نموده و با اندازه گیری افت ولتاژ روی آن از مقدار جریانی که موتور DC مصرف می کند مطلع شوید . شکل 8-6 نحوه راه اندازی یک موتور DC در دو جهت را با استفاده از IC مزبور نشان می دهد .

توجه داشته باشید که در شکل 6-8 سیگنال PWM به پایه VENABLE اعمال می شود تغییر جهت موتور نیز با اعمال سطح منطقی صفر یا یک به پایه های 12 و 10 انجام می گیرد. البته توجه داشته باشید که سطح منطقی اعمالی به این دو پایه بایستی به صورت عکس هم باشند یعنی اگر به پایه 12 سطح منطقی یک اعمال شود ، سطح منطقی پایه 10 حتما بایستی صفر باشد در غیر این صورت موتور DC حرکت نخواهد کرد حال برای تغییر جهت حرکت موتور می توانید پایه 12 را صفر و پایه 10 را یک کنید. مقدار مقاومت RS بایستی کم باشد به طور معمول مقدار این مقاومت برابر 1 اهم انتخاب می شود.



شکل 6-8 نحوه راه اندازی یک موتور DC توسط راه انداز L298N

شکل 6-9 آرایش پایه ها و همچنین شکل ظاهری L298 را نشان می دهد .



شکل 6-9 آرایش پایه ها و شکل ظاهری L298

برای تولید پالس PWM می توانید از خروجی PWM میکروکنترلر AVR استفاده کنید نحوه تولید سیگنال PWM در فصل دوم کتاب مفصلا توضیح داده شده است .

شکل 10-6 مدار میکروکنترلری کنترل سرعت موتور 12 ولتی را با استفاده از راه انداز L298 نشان می دهد .

در این پروژه با استفاده از کلید های SPEED DOWN,SPEED UP می توانید سرعت حرکت موتور DC را تغییر دهید. همچنین جهت حرکت موتور با استفاده از کلید های LEFT TO RIGHT و RIGHT TO LEFT قابل تغییر است .

برای متوقف کردن موتور نیز می توانید از کلید PLAY/PAUSE استفاده کنید .برنامه نوشته شده به صورت زیر است .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Timer2 = Pwm , Prescale = 64 , Pwm = On , Compare Pwm = Clear Up
Config Pind.0 = Input : Speed_up Alias Pind.0
Config Pind.1 = Input : Speed_down Alias Pind.1
Config Pind.2 = Input : Left_to_right Alias Pind.2
Config Pind.3 = Input : Right_to_left Alias Pind.3
Config Pind.4 = Input : Play_pause Alias Pind.4
Config Pinb.0 = Output
Config Pinb.1 = Output
Config Pinb.3 = Output
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.0 , Db5 = Pinc.1 , Db6 = Pinc.2 , Db7 = Pinc.3_
                , E = Pinc.4 , Rs = Pinc.5
Dim Speed As Byte , Play As Bit , Left_or_right As Bit
Speed = 50 : Play = 0 : Left_or_right = 0 : Cursor Off
'-----------------------------------------------------------------------------------------------------------
Enable Interrupts
Cls : Home:Lcd "SPEED CONTROL"
Lowerline : Lcd "SPEED=" ; Speed
'START OF SPEED CONTROL PROFRAM-----------------------------------------------------------------
Do
'START OF LEFT TO RIGHT BUTTON PROGRAM-----------------------------------------------------------
If Left_to_right = 1 Then
Left_or_right = 0 : Reset Portb.0 : Set Portb.1
Bitwait Left_to_right , Reset : End If
'END OF LEFT TO RIGHT BUTTON PROGRAM-------------------------------------------------------------

'START OF RIGHT TO LEFT BUTTON PROGRAM-----------------------------------------------------------
If Right_to_left = 1 Then
Left_or_right = 1 : Set Portb.0 : Reset Portb.1
Bitwait Right_to_left , Reset : End If
'END OF RIGHT TO LEFT BUTTON PROGRAM-------------------------------------------------------------

'START OF PLAY PAUSE BUTTON PROGRAM--------------------------------------------------------------
If Play_pause = 1 Then
Play = Not Play

If Play = 0 Then
If Left_or_right = 1 Then
Set Portb.0 : Reset Portb.1 : End If
If Left_or_right = 0 Then
```

```
Reset Portb.0 : Set Portb.1 : End If
End If

If Play = 1 Then
Reset Portb.0 : Reset Portb.1 : End If

Bitwait Play_pause , Reset : End If
'END OF PLAY PAUSE BUTTON PROGRAM-----------------------------------------------------------

'START OF SPEED UP BUTTON PROGRAM-----------------------------------------------------------
If Speed_up = 1 Then
If Speed < 255 Then Speed = Speed + 5
Cls : Home : Lcd "SPEED CONTROL"
Lowerline : Lcd "SPEED=" ; Speed
Bitwait Speed_up , Reset
End If
'END OF SPEED UP BUTTON PROGRAM-----------------------------------------------------------

'START OF SPEED DOWN BUTTON PROGRAM-----------------------------------------------------------
If Speed_down = 1 Then
If Speed > 0 Then Speed = Speed - 5
Cls : Home : Lcd "SPEED CONTROL"
Lowerline : Lcd "SPEED=" ; Speed
Bitwait Speed_down , Reset
End If
'END OF SPEED DOWN BUTTON PROGRAM-----------------------------------------------------------
Ocr2 = Speed
Loop
'END OF SPEED MOTOR CONTROL PROFRAM-----------------------------------------------------------
```

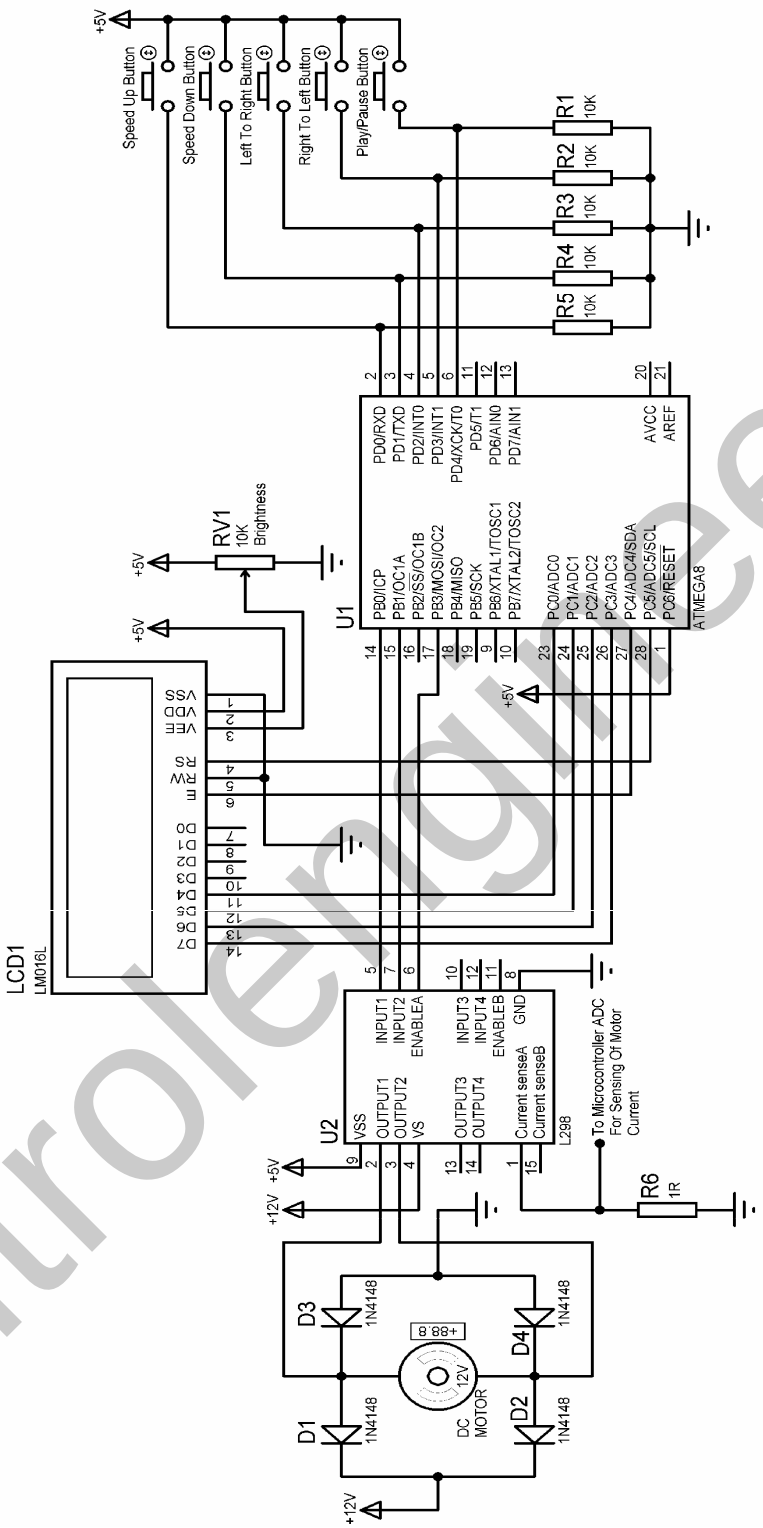
برنامه فوق بسیار ساده بود و براحتی قابل تحلیل است اگر در تحلیل برنامه فوق مشکل داشتید به فصل دوم کتاب مراجعه کنید.

تنها نکته ای که در رابطه با برنامه فوق بایستی مورد توجه قرار گیرد فرکانس سیگنال PWM خروجی می باشد. فرکانس PWM برای کنترل موتور های DC معمولا در بازه 200 تا 300 هرتز انتخاب می شود.

فرکانس PWM خروجی در این پروژه از فرمول زیر محاسبه می شود .

PWM FREQUNCY = فرکانس کاری میکرو / (510 * PRESCALE)

PWM FREQUNCY=8000000 / (510 * 64) = 245Hz



شکل 6-10 شماتیک پروژه کنترل سرعت و جهت موتور DC

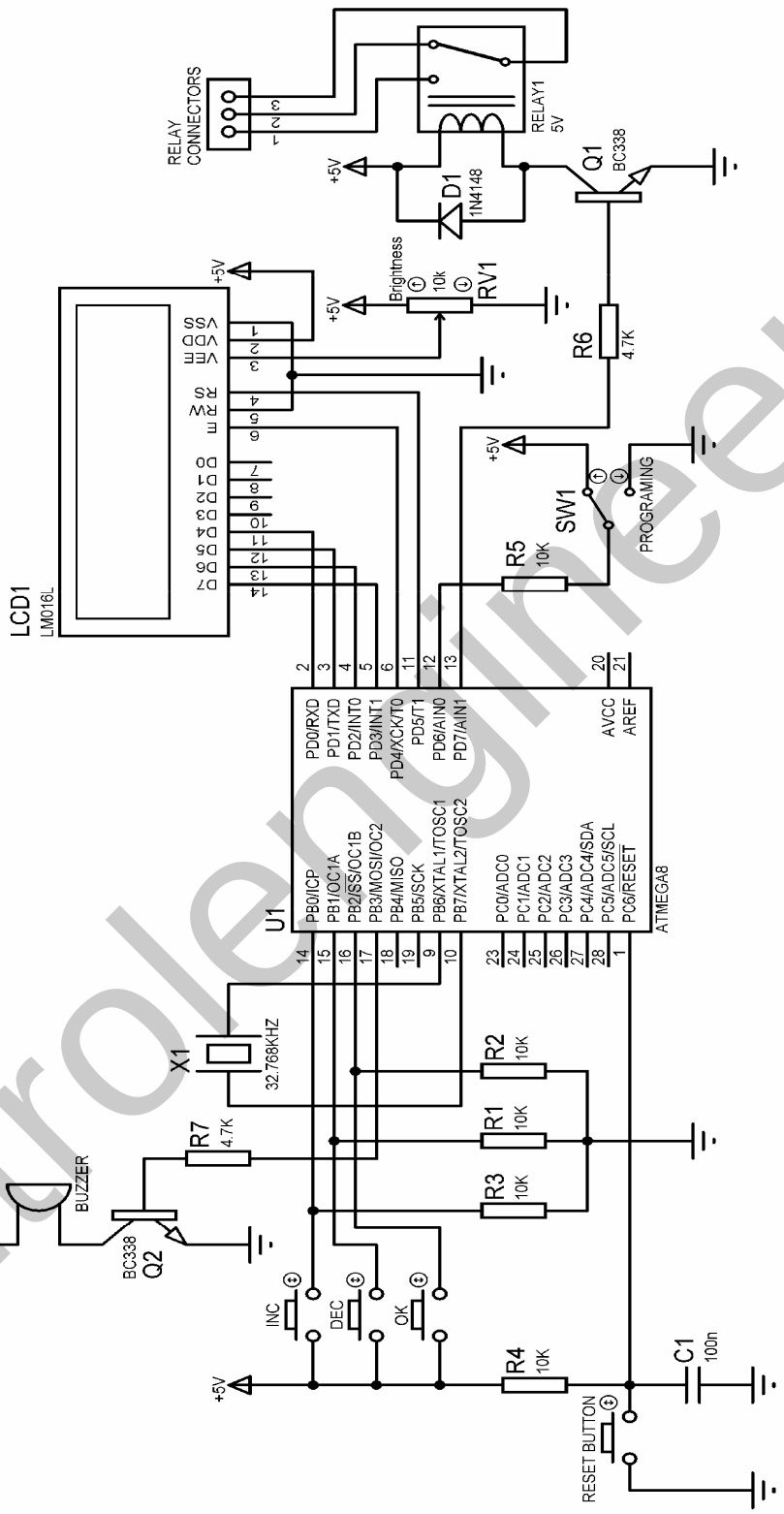
تایمر میکروکنترلری دقیق با مدت زمان قابل تنظیم از یک دقیقه تا 250 ساعت

در این پروژه از کریستال ساعت 32.768KHz برای ساختن زمان واقعی یک ثانیه استفاده شده است. زمان های قابل تنظیم با استفاده از مد آسنکرون تایمر های 8 بیتی 2,0 در حالتی که کلاک خود را از کریستال خارجی 32.768KHz دریافت کنند. در جدول زیر ارائه شده است .

PRESCALE	OVERFLOW PERIOD
STOP,TIMER/COUNTER IS STOPED	-
1	1/64 S
8	1/8 S
32	1/4 S
64	1/2 S
128	1 S
256	2 S
1024	8S

شکل 11-6 شماتیک طراحی شده برای پروژه تایمردیجیتال را نشان می دهد .

برای را ه اندازی پروژه ابتدا بایستی کلید دو حالته SW1 را در حالت PROGRAMING قرار داده سپس با استفاده از کلید های INC ، DEC و OK ساعت و دقیقه را وارد کرده و کلید SW1 را از حالت PROGRAMING خارج کنید . در این حالت تایمر از مدت زمان صفر شروع به شمارش کرده و زمانی که مدت زمان شمارش شده توسط تایمر با مدت زمان تعیین شده توسط کاربر برابر شد. رله خروجی در حالت روشن قرار گرفته و صدای بوق از SPEAKER پخش می شود برای قطع کردن صدای بوق می توانید یک کلید در مسیر SPEAKER قرار داده و آن را با استفاده از کلید روشن و خاموش کنید .



شکل 6-11 شماتیک طراحی شده برای تایمر دیجیتال

برنامه نوشته شده برای پروژه به صورت زیر است .

```
'COMPILER:BASCOM 1.11.8.7
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Pinb.0 = Input : Inc_button Alias Pinb.0
Config Pinb.1 = Input : Dec_button Alias Pinb.1
Config Pinb.2 = Input : Ok_button Alias Pinb.2
Config Pinb.3 = Output : Speaker Alias Portb.3
Config Pind.7 = Output : Relay Alias Portd.7
Config Pind.6 = Input : Sw1 Alias Pind.6
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
 , E = Pind.4 , Rs = Pind.5
Config Timer2 = Timer , Async = On , Prescale = 128
Enable Timer2
Enable Ovf2
On Ovf2 Timer2_overflow
Enable Interrupts
Dim M As Byte , S As Byte , H As Byte , A As Byte , H_save As Eram Byte
Dim M_save As Eram Byte , M_compare As Byte , H_compare As Byte
'START OF PROGRAM-----------------------------------------------------------------------------------------------
S = 0 : M = 0 : H = 0
If Sw1 = 1 Then Goto Time_counter
'-----------------------------------------------------------------------------
Program_start:
Cursor Off : Cls : Home
Lcd "IN THE NAME OF"
Locate 2 , 7 : Lcd "GOD"
Wait 4
Cls : Home
Lcd "PLEASE ENTER"
Lowerline : Lcd "OVERFLOW TIME"
Wait 4
'-----------------------------------------------------------------------------------------------------------------
Input_h:
Cls : Home : Lcd "PLEASE ENTER H"
Lowerline : Lcd "H=" ; H
'-----------------------------------------------------------------------------
If Inc_button = 1 Then
If H < 255 Then Incr H
Sound Speaker , 100 , 150
End If
'-----------------------------------------------------------------------------
If Dec_button = 1 Then
If H > 0 Then Decr H
Sound Speaker , 100 , 150
End If
'-----------------------------------------------------------------------------
If Ok_button = 1 Then
Cls : Home : Lcd "OK!!"
Lowerline : Lcd "H IS=" ; H
Sound Speaker , 1000 , 150
Wait 4 : H_save = H : Waitms 10
Goto Enter_m : End If
'-----------------------------------------------------------------------------
Waitms 200
Goto Input_h
'-----------------------------------------------------------------------------------------------------------------
Enter_m:
Cls : Home : Lcd "PLEASE ENTER M"
```

```
Lowerline : Lcd "M=" ; M
'-----------------------------------------------------------------------------
If Inc_button = 1 Then
If M < 59 Then Incr M
Sound Speaker , 100 , 150
End If
'-----------------------------------------------------------------------------
If Dec_button = 1 Then
If M > 0 Then Decr M
Sound Speaker , 100 , 150
End If
'-----------------------------------------------------------------------------
If Ok_button = 1 Then
Cls : Home : Lcd "OK!!"
Lowerline : Lcd "M IS=" ; M
Sound Speaker , 1000 , 150
Wait 4 : M_save = M : Waitms 10
Goto Time_counter : End If
'-----------------------------------------------------------------------------
Waitms 200
Goto Enter_m
'-----------------------------------------------------------------------------------------------------------------
Time_counter:
S = 0 : M = 0 : H = 0 : Cursor Off
M_compare = M_save : H_compare = H_save
'-----------------------------------------------------------------
Do
Home
Lcd "OVF TIME= " ; H_compare ; ":" ; M_compare
Lowerline : Lcd "TIME IS " ; H ; ":" ; M ; ":" ; S
'-----------------------------------------------------------------
If H = H_compare Then
If M = M_compare Then
Set Relay
Cls : Home
Lcd "TIME IS"
Lowerline : Lcd "OWERFLOW"
Disable Interrupts
Do
Sound Speaker , 1000 , 150
Waitms 100
Loop
End If : End If
'-----------------------------------------------------------------
Waitms 300
Loop
'-----------------------------------------------------------------
'END OF PROGRAM-----------------------------------------------------------------------------------------------
Timer2_overflow:
Disable Interrupts
Stop Timer2
Incr S
If S > 59 Then
S = 0 : Incr M : Cls
End If
'-----------------------------------------------------------------
If M > 59 Then
Incr H : M = 0 : Cls
End If
'-----------------------------------------------------------------
If H > 250 Then
S = 0 : M = 0 : H = 1 : Cls
```

```
End If
'-----------------------------------------------------------------
Enable Interrupts
Start Timer2
Return
'-----------------------------------------------------------------------------------------------------------
```

برنامه فوق ساده بوده و براحتی قابل تحلیل است اگر در تحلیل برنامه فوق مشکل دارید به پروژه ساعت در فصل AVR در الکترونیک نوری مراجعه کنید .

دماسنج دیجیتال با قابلیت تحریک رله هنگام خارج شدن از محدوده دمائی تعیین شده

در این پروژه کاربر قادر است یک محدوده دمائی برای دستگاه تعریف کند به طوری که اگر دمای محیط بالاتر از محدوده دمائی تعیین شده باشد رله شماره یک و اگر دمای محیط کمتر از محدوده دمائی تعیین شده باشد رله شماره 2 روشن خواهد شد. کاربر می تواند توسط رله شماره 1 سیستم سرمایشی و توسط رله شماره 2 سیستم گرمایشی را سوئیچ کند ، در این پروژه از سنسور دمای LM35 ، به عنوان حسگر دما استفاده شده است ولتاژ خروجی این سنسور به صورت خطی و متناسب با دمای محیط تغییر می کند به طوری که به ازای افزایش هر درجه سانتی گراد ولتاژ خروجی سنسور LM35 ، 10 میلی ولت افزایش می یابد ، به عنوان مثال در دمای 25 درجه سانتی گراد ولتاژ خروجی سنسور به صورت زیر خواهد بود .

25 * 10mV = 0.25v

از طرفی دقت ADC داخلی AVR ، 10 بیتی بوده و می تواند ولتاژ REFERENCE اعمالی را به 1024 قسمت تقسیم کند با توجه به این که در این پروژه مقدار ولتاژ REFERENCE برابر 5 ولت انتخاب شده است ولتاژ اعمالی به ورودی ADC برای افزایش یک شماره به صورت زیر خواهد بود.

5 / 1024 =0.005V

پس با اعمال 0.25 ولت به ورودی ADC عدد دیجیتال شده به صورت زیر خواهد بود .

0.25 / 0.005 = 50

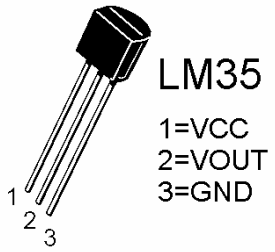
که در این شرایط برای بدست آوردن دمای واقعی بایستی مقدار دیجیتال شده را بر عدد 2 تقسیم کنیم .

50 / 2 = 25

مشخصات انواع مختلف سنسور دمای LM35 در جدول زیر نشان داده شده است .

DEVICE	TEMP RANGE	ACCURENCY	OUTPUT SCALE
LM35A	-55 C TO +150C	+1.0C	10mv/c
LM35	-55 C TO +150C	+1.5C	10mv/c
LM35CA	-40 C TO +110C	+1.0C	10mv/c
LM35C	-40 C TO +110C	+1.5C	10mv/c
LM35D	0 C TO +100C	+2.0C	10mv/c

شکل ظاهری و ترتیب پایه های سنسور دمای LM35 ، در شکل 6-12 نشان داده شده است .



شکل 6-12 شکل ظاهری و ترتیب پایه های LM35

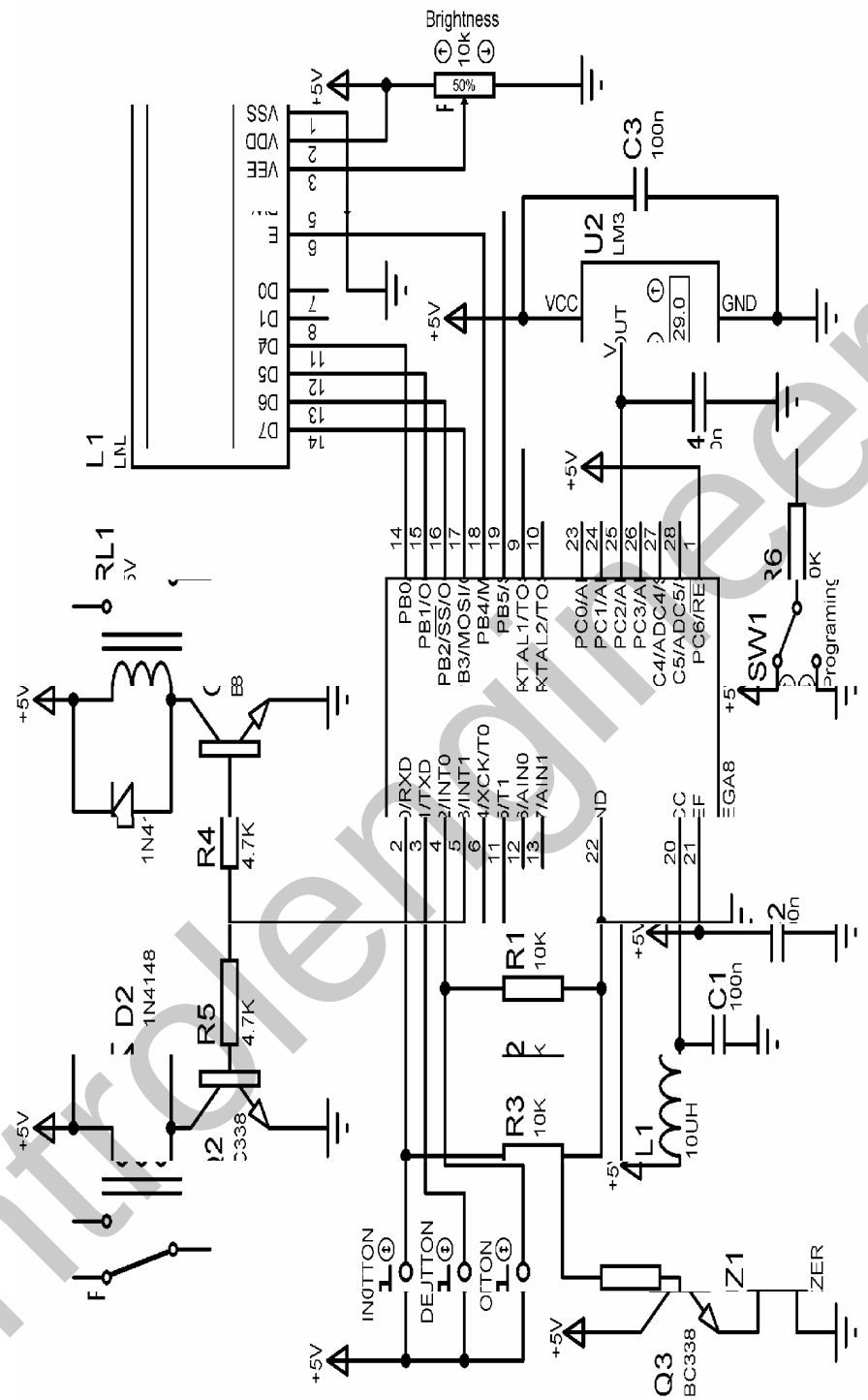
شکل 6-13 شماتیک طراحی شده برای دماسنج دیجیتال را نشان می دهد .

در این پروژه ، کاربر بایستی ابتدا کلید SW1 را در حالت PROGRAMING قرار داده سپس تغذیه مدار را به آن اعمال کند در این حالت دستگاه وارد قسمت برنامه ریزی می شود و عبارت

```
IF A>TEMP THEN
RELAY1 IS ON
```

بر روی LCD نوشته شده سپس عدد A از کاربر خواسته می شود در این حالت کاربر بایستی با استفاده از کلید های INCR BUTTON , DECR BUTTON ، OK BUTTON عدد A را وارد کند. پس از وارد کردن عدد A عبارت

```
IF B<TEMP THEN
RELAY2 IS ON
```



شکل 13-6 شماتیک طراحی شده برای دماسنج دیجیتال

بر روی LCD نوشته شده سپس عدد B از کاربر درخواست می شود در این حالت کاربر بایستی عدد B را وارد کرده سپس کلید SW1 را از حالت PROGRAMING خارج کند .

برنامه نوشته شده برای پروژه به صورت زیر است .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinb.0 , Db5 = Pinb.1 , Db6 = Pinb.2 , Db7 = Pinb.3_
            , E = Pinb.4 , Rs = Pinb.5
Config Adc = Single , Prescaler = Auto
Deflcdchar 0 , 24 , 24 , 32 , 32 , 32 , 32 , 32 , 32
Dim Temp As Byte , A As Byte , B As Byte
Dim A_save As Eram Byte , B_save As Eram Byte
Config Pind.0 = Input : Incr_button Alias Pind.0
Config Pind.1 = Input : Decr_button Alias Pind.1
Config Pind.2 = Input : Ok_button Alias Pind.2
Config Pinb.6 = Input : Program_sw Alias Pinb.6
Config Pind.3 = Output : Relay1 Alias Portd.3
Config Pind.4 = Output : Relay2 Alias Portd.4
Config Pind.5 = Output : Speaker Alias Portd.5
'-----------------------------------------------------------------------------------------------------------
Program_start:
If Program_sw = 1 Then Goto Starting_adc
Cursor Off : Cls : Home
Lcd "IN THE NAME OF"
Locate 2 , 7 : Lcd "GOD"
Wait 4
Cls : Home
Lcd "IF A>TEMP THEN"
Lowerline : Lcd "RELAY1 IS ON"
Wait 4
'-----------------------------------------------------------------------------------------------------------
Input_a:
Cls : Home : Lcd "PLEASE ENTER A"
Lowerline : Lcd "A=" ; A
'-----------------------------------------------------------------------------
If Incr_button = 1 Then
If A < 255 Then Incr A
Sound Speaker , 100 , 150
End If
'-----------------------------------------------------------------------------
If Decr_button = 1 Then
If A > 0 Then Decr A
Sound Speaker , 100 , 150
End If
'-----------------------------------------------------------------------------
If Ok_button = 1 Then
Cls : Home : Lcd "OK!!"
Lowerline : Lcd "A IS=" ; A
Sound Speaker , 1000 , 150
Wait 4 : A_save = A : Waitms 10
Goto Input_b : End If
'-----------------------------------------------------------------------------
Waitms 200
Goto Input_a
'-----------------------------------------------------------------------------------------------------------
Input_b:
Cls : Home
Lcd "IF B<TEMP THEN"
Lowerline : Lcd "RELAY2 IS ON"
Wait 4
```

```
'-----------------------------------------------------------------------------
Enter_b:
Cls : Home : Lcd "PLEASE ENTER B"
Lowerline : Lcd "B=" ; B
'-----------------------------------------------------------------------------
If Incr_button = 1 Then
If B < 255 Then Incr B
Sound Speaker , 100 , 150
End If
'-----------------------------------------------------------------------------
If Decr_button = 1 Then
If B > 0 Then Decr B
Sound Speaker , 100 , 150
End If
'-----------------------------------------------------------------------------
If Ok_button = 1 Then
Cls : Home : Lcd "OK!!"
Lowerline : Lcd "B IS=" ; B
Sound Speaker , 1000 , 150
Wait 4 : B_save = B : Waitms 10
Goto Starting_adc : End If
'-----------------------------------------------------------------------------
Waitms 200
Goto Enter_b
'-----------------------------------------------------------------------------------------------------------------
Starting_adc:
Start Adc
A = A_save : B = B_save
Do
Temp = Getadc(2)
Temp = Temp / 2
Cls : Home : Lcd "TEMP IS:" ; Temp ; Chr(0) ; "C"
Locate 2 , 1 : Lcd "TERMOMETER"
Waitms 200
'-----------------------------------------------------------------------------
If A > Temp Then
Set Relay1 : Else
Reset Relay1 : End If
'-----------------------------------------------------------------------------
If B < Temp Then
Set Relay2 : Else
Reset Relay2 : End If
'-----------------------------------------------------------------------------
Loop
'END OF PROGRAM-----------------------------------------------------------------------------------------------
```

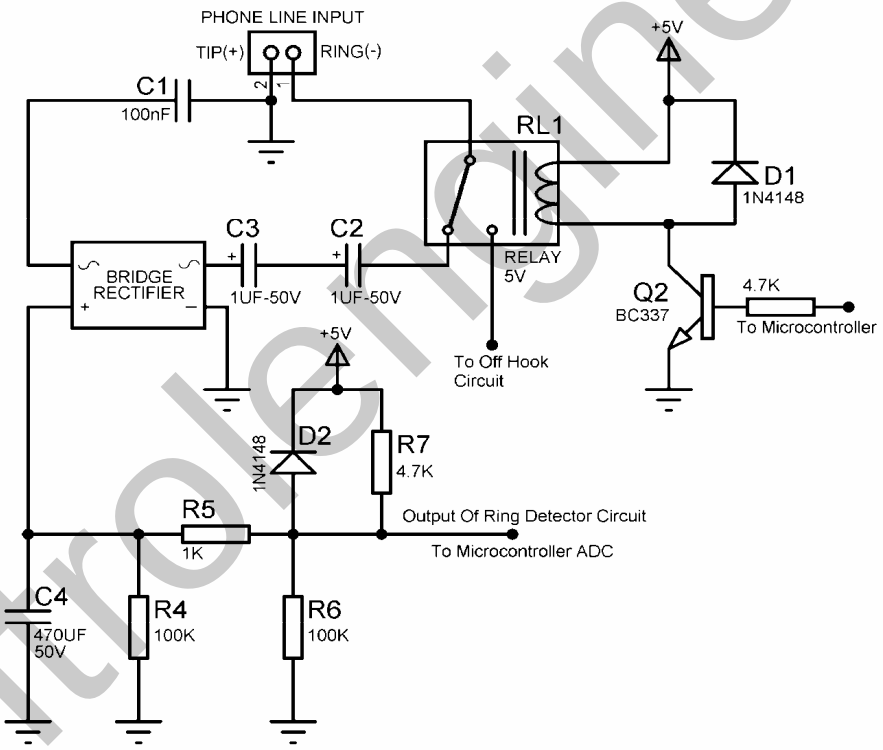
کنترل وسایل برقی با استفاده ازخط تلفن (TELE REMOTE CONTROL)

نحوه کار با پروژه TELE REMOTE CONTROL بدین صورت می باشد که کاربر با استفاده گوشی موبایل یا گوشی تلفن ثابت شماره خطی که مدار آشکار ساز DTMF به آن متصل است را گرفته و منتظر می شود تا مدار گیرنده سیگنال زنگ را تشخیص داده و خط تلفن (PHONE LINE) را از حالت سکون(ON HOOK) به حالت اشغال (OFF HOOK) برده و یک صدای بوق از طریق خط تلفن به گوشی کاربر ارسال کند ، کاربر می تواند پس از شنیدن صدای بوق کلید مورد نظر خود را فشار دهد در این حالت مدار گیرنده کلید فشرده شده را

تشخیص داده و میکروکنترلر خروجی مورد نظر را فعال می کند ، برای طراحی این پروژه ابتدا بایستی اطلاعاتی در رابطه با خط تلفن داشته باشیم .

توجه داشته باشید که خط تلفن شهری PSTN (PUPLIC SWITCHED TELEPHONE NETWORK) در حالت سکون دارای ولتاژی در حدود 50 ولت (DC) می باشد. این ولتاژ در زمان زنگ خوردن تلفن به 150 ولت (AC) افزایش می یابد . زمانی که خط از حالت سکون به حالت اشغال برده می شود ولتاژ خط به 10 ولت DC کاهش پیدا می کند البته ولتاژ های فوق بسته به استاندارد محلی مرکز مخابرات (LEC) ممکن است کمی تغییر کند .

ما بایستی با توجه به مطالب فوق یک مدار (RING DETECTOR CIRCUIT) طراحی کنیم مدار آشکار ساز زنگ طراحی شده برای این پروژه در شکل 6-14 نشان داده شده است .



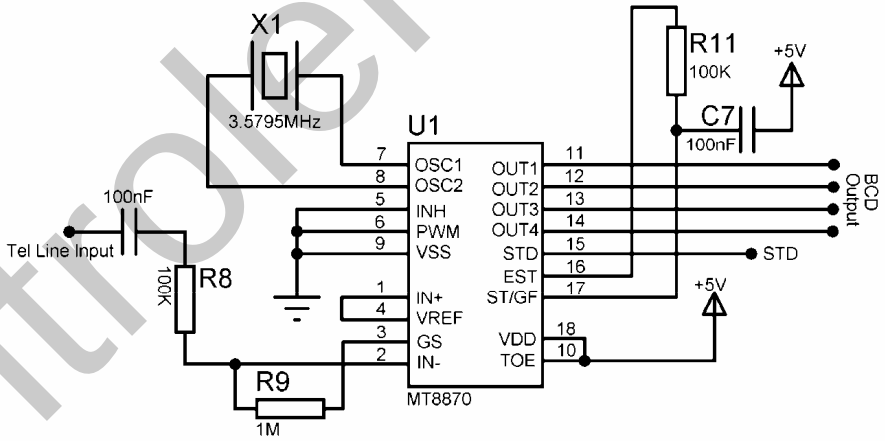
شکل 6-14 شماتیک مدارات تشخیص زنگ

خروجی مدار تشخیص زنگ به ADC میکروکنترلر متصل می شود نحوه عملکرد مدار به گونه ای است که هنگامی که خط تلفن در حالت سکون می باشد مقدار عدد دیجیتال شده توسط ADC کمتر از 800 بوده و هنگامی که سیگنال زنگ توسط خط تلفن به مدار فوق اعمال شود مقدار دیجیتال شده بیشتر از 800 خواهد بود البته این اعداد ممکن است با توجه به مرکز مخابرات محلی کمی تغییر کند ولی عملکرد کلی مدار تشخیص زنگ

بسیار ساده بوده و کافی است شما مقدار دیجیتال شده را در حالت سکون و زمانی که سیگنال زنگ وجود دارد بررسی کنید میکروکنترلر پس از تشخیص سیگنال زنگ و تغییر وضعیت رله 5 ولتی خط تلفن را به حالت اشغال (OFF HOOK) می برد . برای آنکه خط تلفن در حالت OFF HOOK قرار بگیرد بایستی مقاومتی در حد 200 تا 300 اهم بین دو سیم خط تلفن به صورت موازی قرار داده شود . در واقع هنگامی که شما گوشی تلفن ثابت را بر می دارید یک مقاومت 300 اهمی با خط تلفن موازی می شود در این پروژه پس از اشغال شدن خط تلفن یک صدای بوق با استفاده از قطع و وصل سریع خط تلفن با تحریک بیس ترانزیستور D880 به کاربر ارسال می شود ، کاربر می تواند پس از شنیدن صدای بوق کلید مورد نظر خود را برای تحریک خروجی های مدار گیرنده فشار دهد. البته توجه داشته باشید در گوشی هایی که مجهز به دو حالت شماره گیری TONE وPULSE می باشند بایستی شماره گیری در حالت TONE تنظیم شود و گرنه پالس ارسالی توسط مدار گیرنده قابل تشخیص نخواهد بود .

با فشار هر کلید در حالت TONE یک سیگنال DTMF (DUAL-TONE MULTI FREQENCY) به معنی دو تون چند فرکانسی به گیرنده ارسال می شود . برای تشخیص این سیگنال ها از یک آی سی DTMF DECODER استفاده شده است . آی سی DTMF DECODER استفاده شده در این پروژه MT8870D نام دارد.

مدار راه انداز پیشنهادی شرکت سازنده برای این IC در شکل 15-6 نشان داده شده است .



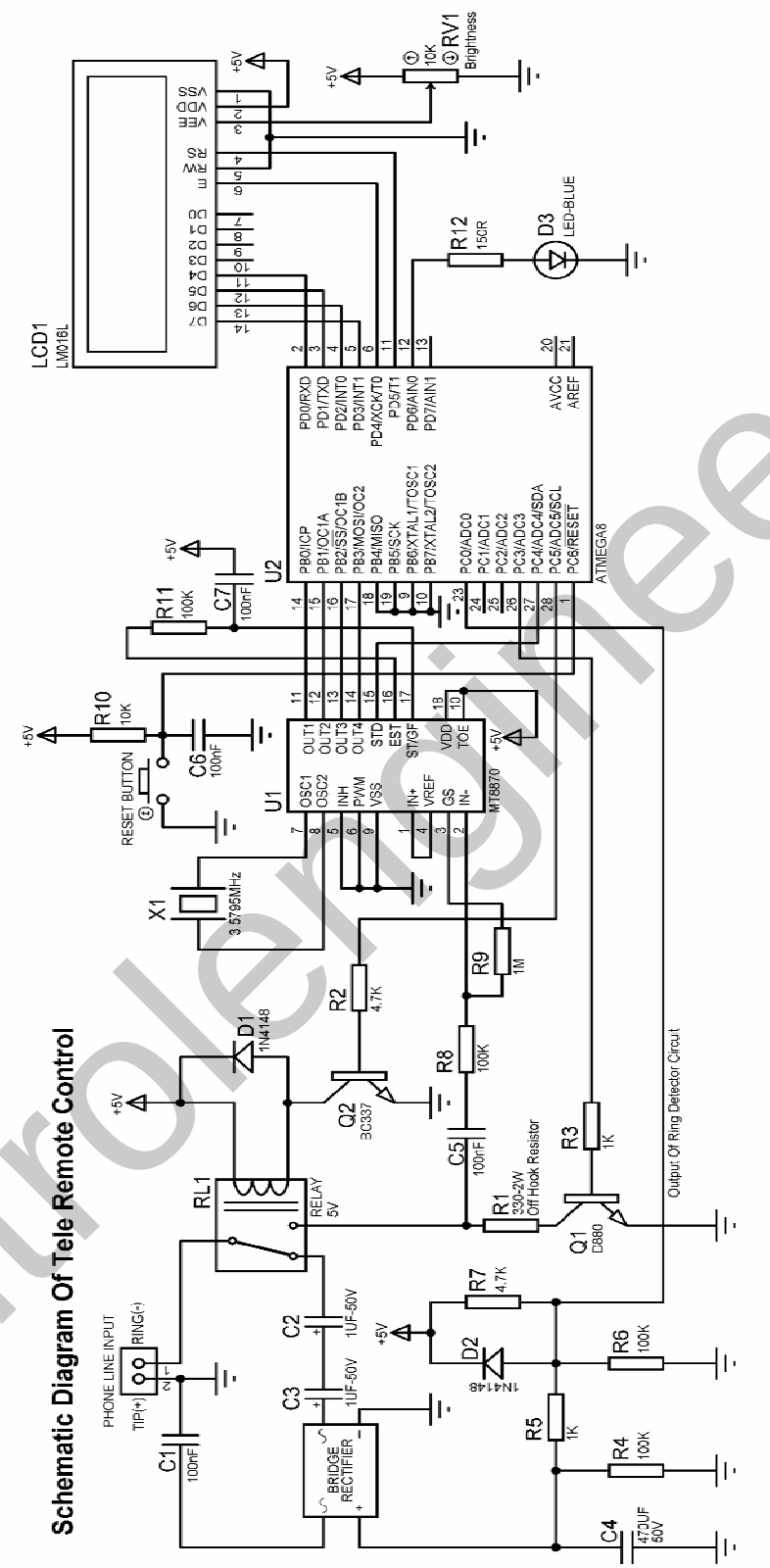
شکل 15-6 شماتیک مدار راه انداز MT8870D

وضعیت خروجی های MT8870D به ازای دریافت سیگنال DTMF مربوط به کلید های مختلف در جدول زیر ارائه شده است .

DIGIT	TOE	INH	EST	OUT4	OUT3	OUT2	OUT1
ANY	L	X	H	Z	Z	Z	Z
1	H	X	H	0	0	0	1
2	H	X	H	0	0	1	0
3	H	X	H	0	0	1	1
4	H	X	H	0	1	0	0
5	H	X	H	0	1	0	1
6	H	X	H	0	1	1	0
7	H	X	H	0	1	1	1
8	H	X	H	1	0	0	0
9	H	X	H	1	0	0	1
0	H	X	H	1	0	1	0
*	H	X	H	1	0	1	1
#	H	X	H	1	1	0	0
A	H	L	H	1	1	0	1
B	H	L	H	1	1	1	0
C	H	L	H	1	1	1	1
D	H	L	H	0	0	0	0
A	H	H	L	UNDETEDTED, THE OUTPUT CODE WILL REMAIN THE SAME AS THE PREVIOUS DETECTED CODE			
B	H	H	L				
C	H	H	L				
D	H	H	L				

خروجی باینری مربوط به کلید فشرده شده توسط گوشی موبایل یا گوشی تلفن ثابت در چهار پایه 11، 12، 13 و 14 قرار می گیرد به عنوان مثال با دریافت سیگنال DTMF مربوط به کلید 1 عدد 1 باینری در خروجی MT8870D ، LATCH شده و همچنین برای مدت کوتاهی وضعیت پایه STD به صورت HIGH خواهد بود با توجه به این که خروجی IC مزبور به صورت LATCH می باشد از پایه STD برای تشخیص کلید جدید زده شده استفاده می شود .

در پروژه فوق پس از تشخیص سیگنال زنگ (RING SIGNAL) توسط میکروکنترلر ، رله در حالت روشن قرار گرفته و همچنین ترانزیستور D880 روشن می شود با انجام عملیات فوق خط تلفن در حالت اشغال قرار می گیرد. سپس با استفاده از ترانزیستور D880 سیگنال بوق از طریق خط تلفن به کاربر ارسال می شود، کاربر پس از شنیدن صدای بوق می تواند با فشار کلید 9 ، LED خروجی گیرنده را روشن و با فشار کلید 8 ، LED را خاموش کند. همچنین با فشار کلید 5 خط تلفن از حالت اشغال به حالت سکون می رود ، شما می توانید بسته به نیاز خودتان خروجی ها را افزایش داده و برنامه را تغییر دهید .



شکل 6-16 شماتیک مدار TELE REMOTE CONTROL

برنامه نوشته شده برای پروژه به صورت زیر است .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pind.0 , Db5 = Pind.1 , Db6 = Pind.2 , Db7 = Pind.3_
                , Rs = Pind.5 , E = Pind.4
Config Adc = Single , Prescaler = Auto
Dim Adc_value As Word , Input_data As Byte , Ring_signal_counter As Byte
Config Pinc.4 = Input : Dtmf_detector Alias Pinc.4
Config Pinc.5 = Output : Off_hook_relay Alias Portc.5
Config Pinc.3 = Output : Off_hook_resistor Alias Portc.3
Config Pind.6 = Output : Led_display Alias Portc.6
Config Portb = Input
'-----------------------------------------------------------------------------------------------------------
Program_start:
Cursor Off : Cls : Home
Lcd "TELE REMOTE "
Lowerline : Lcd "CONTROL"
Wait 4
'-----------------------------------------------------------------------------------------------------------
Start Adc
Adc_of_ring_detector:
Adc_value = Getadc(0)
Waitms 100
Adc_value = Adc_value / 2
Cls : Home : Lcd "TELE REMOTE"
Lowerline : Lcd "ADC VALUE IS " ; Adc_value
If Adc_value < 400 Then Goto Adc_of_ring_detector
'-----------------------------------------------------------------------------------------------------------
Incr Ring_signal_counter
If Ring_signal_counter > 5 Then Goto Off_hook_program
Ht:
Adc_value = Getadc(0)
Waitms 100
Adc_value = Adc_value / 2
If Adc_value > 400 Then Goto Ht
Goto Adc_of_ring_detector
'-----------------------------------------------------------------------------------------------------------
Off_hook_program:
Cls : Home : Lcd "NOW LINE IS"
Lowerline : Lcd "OFF HOOK"
Set Off_hook_resistor
Set Off_hook_relay
Wait 1
    Sound Off_hook_resistor , 100 , 130
    Sound Off_hook_resistor , 100 , 140
    Sound Off_hook_resistor , 100 , 150
    Sound Off_hook_resistor , 100 , 160
    Sound Off_hook_resistor , 100 , 170
    Sound Off_hook_resistor , 100 , 180
    Sound Off_hook_resistor , 100 , 170
    Sound Off_hook_resistor , 100 , 160
    Sound Off_hook_resistor , 100 , 150
    Sound Off_hook_resistor , 100 , 140
    Sound Off_hook_resistor , 100 , 130
    Sound Off_hook_resistor , 100 , 130
Set Off_hook_resistor
Hossein:
```

```
If Dtmf_detector = 0 Then Goto Hossein
Input_data = Pinb
Cls : Home : Lcd "DATA IS" ; Input_data
'-----------------------------------------------------------------------------------------------------------
If Input_data = 9 Then
Set Led_display
Locate 2 , 1 : Lcd "BLUE LED IS ON"
End If
'-----------------------------------------------------------------------------------------------------------
If Input_data = 8 Then
Reset Led_display
Locate 2 , 1 : Lcd "BLUE LED IS OFF"
End If
'-----------------------------------------------------------------------------------------------------------
If Input_data = 5 Then
Reset Off_hook_resistor
Reset Off_hook_relay
Ring_signal_counter = 0
Cls : Home : Lcd "NOW LINE IS"
Lowerline : Lcd "ON HOOK"
Wait 5
Goto Adc_of_ring_detector
End If
Waitms 200
Goto Hossein
'-----------------------------------------------------------------------------------------------------------
```

در برنامه فوق پس از ارسال صدای بوق به خط تلفن برای اطمینان از اشباع بودن ترانزیستور D880 ، پین تحریک کننده بیس این ترانزیستور SET می شود همچنین پس از تشخیص زنگ و اشغال کردن خط تلفن برای تشخیص دریافت سیگنال DTMF جدید از پایه STD ، MT8870D استفاده شده است .

کیم وو چونگ (بنیان گذار شرکت اتوموبیل سازی دوو) در کتاب سنگ فرش هر خیابان از طلاست می نویسد :

(تاریخ متعلق به کسانی است که رویاهای بزرگی در سر داشته اند.)

آغاز هر موفقیت بزرگ یک رویاست. اگر هرگز رویایی نداشته اید پس هیچ وقت هم رویایی که به حقیقت بپیوندد نخواهید داشت .

فصل هفتم

AVR در مدارات و پروژه های صوتی

ارگ الکترونیکی با استفاده از AVR

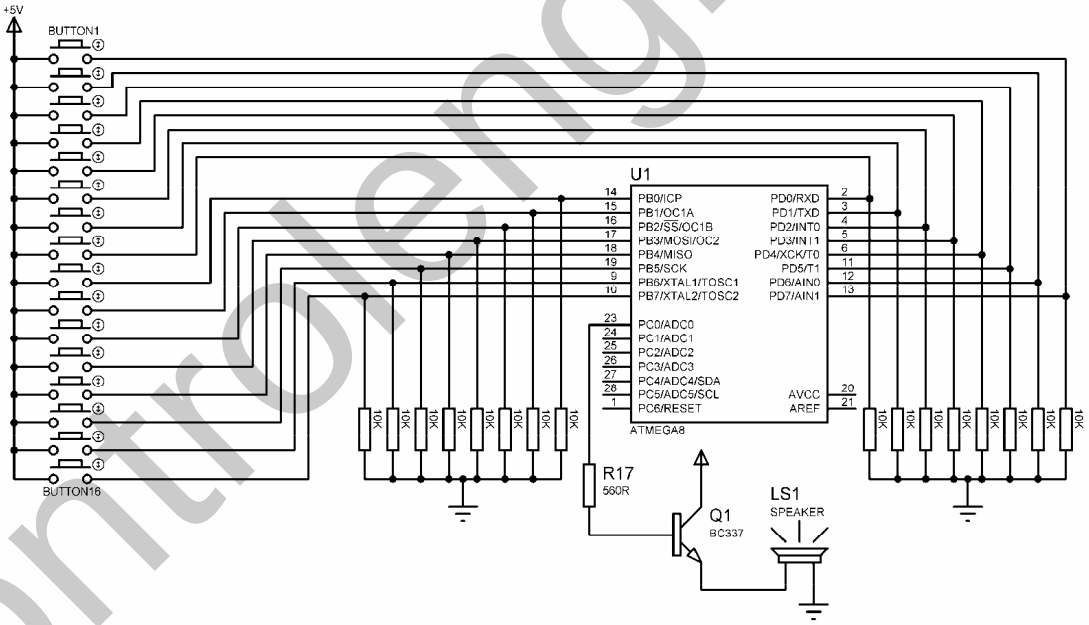
برای ایجاد صدا با استفاده از مدارات دیجیتالی می توانید یک سیگنال مربعی ایجاد کرده و آن را از طریق مدار درایور به SPEAKER اعمال کنید . برای تغییر صوت ایجاد شده می توانید فرکانس سیگنال مربعی را تغییر دهید. در صورتی که سیگنال مربعی اعمالی به SPEAKER دارای فرکانس ثابت باشد صدای سوت شنیده خواهد شد . توسط دستور SOUND می توان پالسهایی با فرکانس دلخواه را به یکی از پین های میکروکنترلر ارسال کرد . فرم کلی دستور SOUND به صورت زیر است .

SOUND PIN,DURATION,PULSES

PIN : نام یکی از پایه های خروجی به عنوان مثال PORTC.1 می باشد .

DUTATION : ثابت یا متغیری است که تعداد پالس های ارسالی به پایه PIN را مشخص می کند و میتواند مقادیر بین 1 تا 65535 را داشته باشد .

PULSES : ثابت یا متغیری است که مدت زمان بالا یا پایین بودن یک سیکل سیگنال ارسالی را به میکرو ثانیه نشان می دهد و می تواند مقادیر بین 1 تا 65535 را داشته باشد حال با استفاده از دستور فوق می توانید یک ارگ الکترونیکی ساده طراحی کنید. شکل 1-7 شماتیک طراحی شده برای ارگ الکترونیکی را نشان می دهد .



شکل 1-7 شماتیک یک ارگ الکترونیکی ساده

برنامه نوشته شده برای سخت افزار فوق به صورت زیر است .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 1000000
Config Portb = Input
```

```
Config Portd = Input
Config Pinc.0 = Output : Speaker Alias Portc.0
Dim Input_button As Byte
Declare Sub Sound1
Declare Sub Sound2
'-----------------------------------------------------------------------------------------------------------
Start_program:
Input_button = Pinb
Call Sound1
Input_button = Pind
Call Sound2
Goto Start_program
'-----------------------------------------------------------------------------------------------------------
Sub Sound1:
Select Case Input_button:
Case Is = &B00000001 : Sound Speaker , 100 , 40
Case Is = &B00000010 : Sound Speaker , 100 , 50
Case Is = &B00000100 : Sound Speaker , 100 , 60
Case Is = &B00001000 : Sound Speaker , 100 , 70
Case Is = &B00010000 : Sound Speaker , 100 , 80
Case Is = &B00100000 : Sound Speaker , 100 , 90
Case Is = &B01000000 : Sound Speaker , 100 , 100
Case Is = &B10000000 : Sound Speaker , 100 , 110
End Select
Return
End Sub Sound1
'-----------------------------------------------------------------------------------------------------------
Sub Sound2:
Select Case Input_button:
Case Is = &B00000001 : Sound Speaker , 100 , 120
Case Is = &B00000010 : Sound Speaker , 100 , 130
Case Is = &B00000100 : Sound Speaker , 100 , 140
Case Is = &B00001000 : Sound Speaker , 100 , 150
Case Is = &B00010000 : Sound Speaker , 100 , 160
Case Is = &B00100000 : Sound Speaker , 100 , 170
Case Is = &B01000000 : Sound Speaker , 100 , 180
Case Is = &B10000000 : Sound Speaker , 100 , 190
End Select
Return
End Sub Sound2
'-----------------------------------------------------------------------------------------------------------
```

برنامه فوق بسیار ساده بوده و به راحتی قابل تحلیل است .

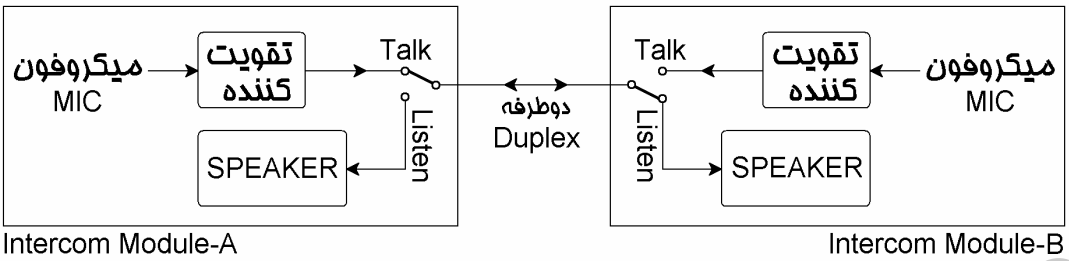
طراحی تلفن داخلی (INTERCOM) دو طرفه

با توجه با این که در پروژه بعدی در رابطه با ارسال و دریافت سیگنال صوتی با استفاده از میکروکنترلرهای AVR بحث خواهیم کرد ابتدا یک پروژه تبادل صوت پایه را بررسی می کنیم .

مدار معرفی شده در این قسمت یک سیستم تلفن داخلی دو طرفه است که فقط از دو رشته سیم برای تبادل صوت بین دو واحد ارتباطی استفاده می کند.

در این پروژه هر یک از واحد ها دارای مدار تقویت کننده ، میکروفن ، SPEAKER و تغذیه مستقل می باشد.

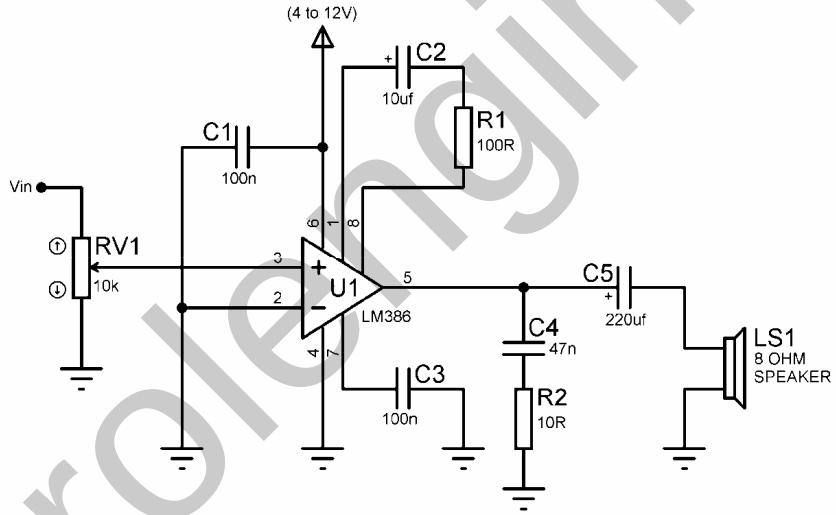
شکل 2-7 بلوک دیاگرام کلی یک سیستم تلفن INTERCOM دو طرفه را نشان می دهد .



شکل 7-2 بلوک دیاگرام یک تلفن INTERCOM

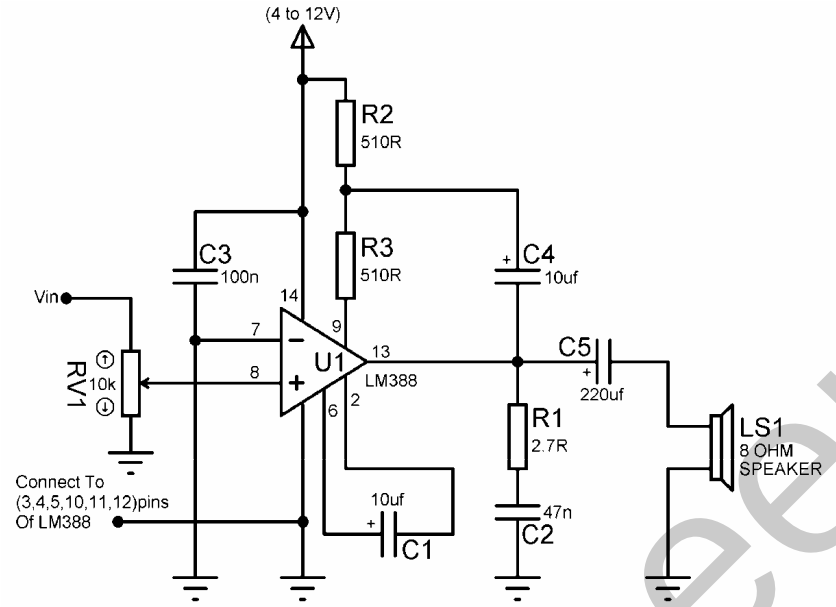
به منظور استفاده از مدار بایستی کلید را در وضعیت TALK قرار داده و صحبت نمود قرار دادن کلید مزبور در حالت LISTEN به طور اتوماتیک واحد را خاموش کرده و آنرا آماده دریافت پیام واحد دیگر می نماید .

برای بلوک تقویت کننده می توانید یکی از مدار های تقویت کننده معرفی شده در این فصل را مورد استفاده قرار دهید. شکل 7-3 یک تقویت صوتی 325 میلی وات را نشان می دهد در این مدار زمانی که خازن C2 در مدار قرار داشته باشد بهره ولتاژ برابر با 200 و اگر خازن C2 حذف شود بهره ولتاژ برابر با 20 خواهد بود.



شکل 7-3 مدار تقویت کننده مبتنی بر آی سی LM386 با خروجی 325 میلی وات

شکل 7-4 یک تقویت کننده صوتی 1.5 وات را نشان می دهد در این مدار زمانی که خازن C2 در مدار قرار داشته باشد بهره ولتاژ برابر با 200 و اگر خازن C2 حذف شود بهره ولتاژ برابر با 20 خواهد بود. در طراحی تلفن INTERCOM از تقویت کننده شکل 7-3 استفاده شده است. شکل 7-5 شماتیک طراحی شده برای تلفن INTERCOM دو طرفه را نشان می دهد .

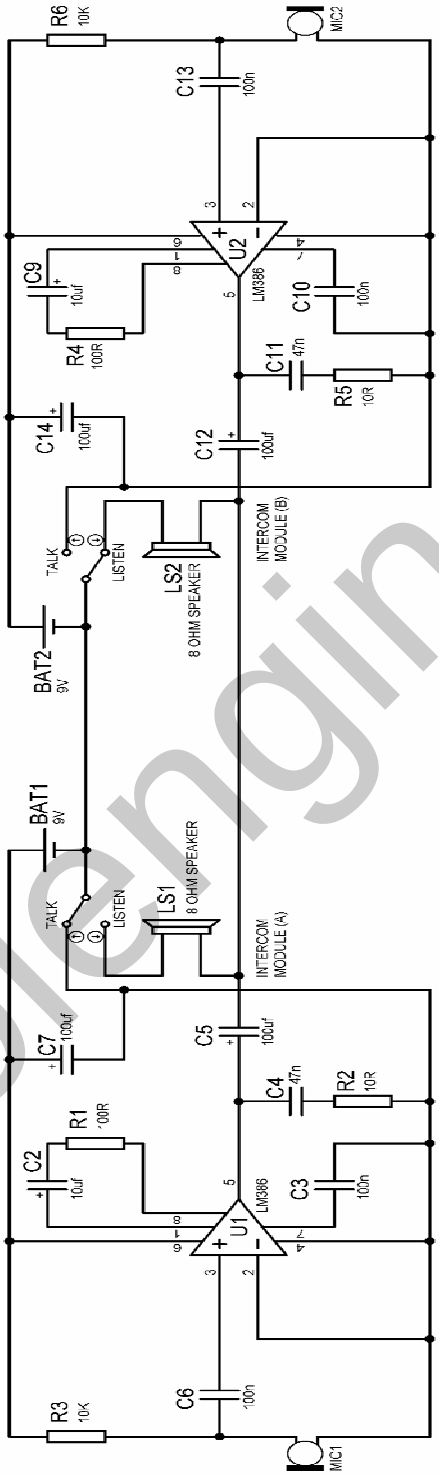


شکل 7-4 مدار تقویت کننده مبتنی بر آی سی LM388 با خروجی 1.5 وات

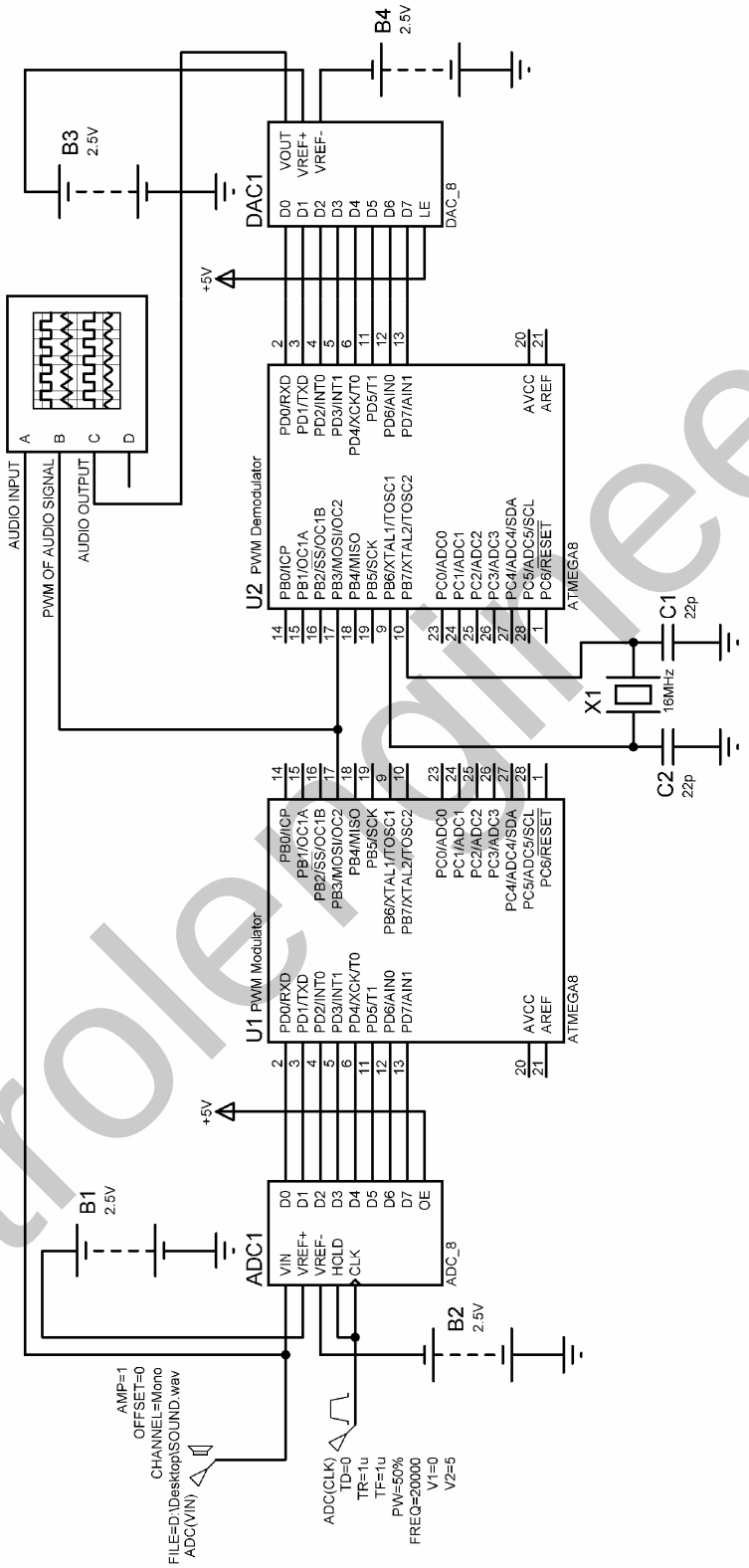
با توجه به شکل 7-5 در حالت عادی بهتر است هر دو کلید در حالت LISTEN قرار گیرد زیرا در این حالت SPEAKER ها با هم موازی شده و هیچ جریانی از باطری ها کشیده نمی شود ، با قرار دادن هر یک از کلید ها در وضعیت TALK تغذیه واحد مربوط به آن برقرار شده و خروجی آنSPEAKER واحد دیگر را تغذیه می کند.

ارسال دیجیتالی یک سیگنال صوتی با استفاده از مدولاسیون PWM

برای ارسال دیجیتالی یک سیگنال صوتی بایستی ابتدا آن را توسط یک ADC تبدیل به دیجیتال کرده ، سپس روی مقادیر دیجیتال شده سیگنال صوتی ورودی مدولاسیون PWM انجام داده و مقادیر PWM را از طریق کانال ارتباطی ارسال کنیم . کانال ارتباطی می تواند ، کانال RF ،کانال مادون قرمز ، دورشته سیم و غیره باشد مساله مهم دیگری که در نمونه برداری (SAMPLING) از سیگنال صوتی بایستی مورد توجه قرار دهیم اصل نمونه برداری است ، طبق این اصل فرکانس نمونه برداری از سیگنال صوتی در کمترین حالت بایستی دو برابر فرکانس خود سیگنال صوتی باشد با توجه به این که فرکانس سیگنال های صوتی در محدوده 300Hz تا 4KHz قرار دارد حداقل میزان نمونه برداری بایستی برابر با 8KHz باشد و گرنه سیگنال صوتی گرفته شده توسط گیرنده با کیفیت پایین تری باز سازی (REPORDUCTION) خواهد شد. البته اگر فرکانس نمونه برداری بیش تر از 8KHz باشد. سیگنال گرفته شده توسط گیرنده با کیفیت بالاتری REPORDUCTION خواهد شد فرکانس نمونه برداری در این پروژه همان فرکانس PWM بوده و مقدار آن برابر با 15.686KHz می باشد. شکل 7-5 شماتیک طراحی شده برای پروژه را نشان می دهد .



شکل 7-5 شماتیک طراحی شده برای تلفن INTERCOM دو طرفه



شکل 5-7 شماتیک طراحی شده برای پروژه ارسال و دریافت دیجیتالی سیگنال صوتی

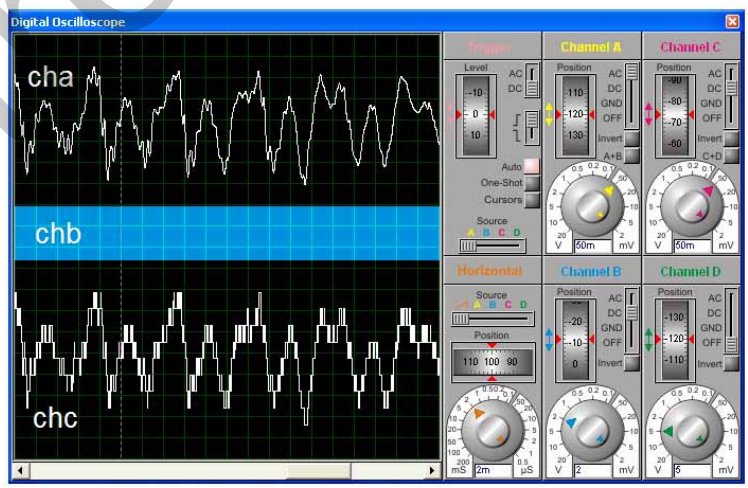
برنامه نوشته شده برای میکروکنترلر مدولاتور به صورت زیر است .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Timer2 = Pwm , Prescale = 1 , Pwm = On , Compare Pwm_
          = Clear Down
Config Pinb.3 = Output
Config Portd = Input
'-----------------------------------------------------------------------------------------------------------
Do
Ocr2 = Pind
Loop
'-----------------------------------------------------------------------------------------------------------
```

برنامه نوشته شده برای میکروکنترلردمدولاتور به صورت زیر است .

```
'COMPILER:BASCOM 1.11.7.4
$regfile = "M8DEF.DAT"
$crystal = 16000000
Config Pinb.3 = Input
Config Portd = Output
Config Timer1 = Timer , Prescale = 8
Stop Timer1
Cursor Off
'-----------------------------------------------------------------------------------------------------------
Do
Bitwait Pinb.3 , Reset
Start Timer1
Bitwait Pinb.3 , Set
Stop Timer1
Timer1 = Timer1 * 2
Portd = Timer1
Timer1 = 0
Loop
'-----------------------------------------------------------------------------------------------------------
```

توجه داشته باشید که این پروژه در سیمولاتور پروتئوس تست شده است . نتیجه نمایش داده شده در اسیلوسکوپ پروتئوس برای کانال های C,B,A در شکل 7-6 نشان داده شده است .



شکل 7-6 شکل موج نقاط متصل شده به کانال های C ، B ، A

تشریح نحوه انجام دمدولاسیون PWM در میکروکنترلر گیرنده

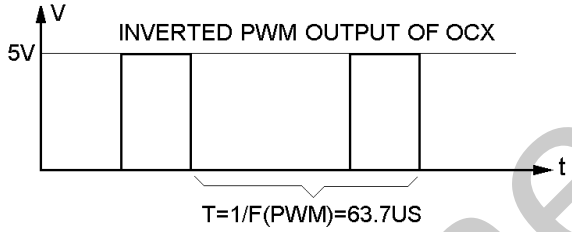
فرض کنید می خواهیم عدد قرار گرفته در رجیستر مقایسه ای میکروکنترلر مدولاتور را توسط یک میکروکنترلر دمدولاتور آشکار سازی کنیم .

با توجه به این که فرکانس PWM تولید شده در مدولاتور به صورت زیر است .

F(PWM) = 8000000 / (510 * 1) = 15.686KHz

زمان نمونه برداری (sampling) برابر خواهد بود با

T = 1 / 15.686khz



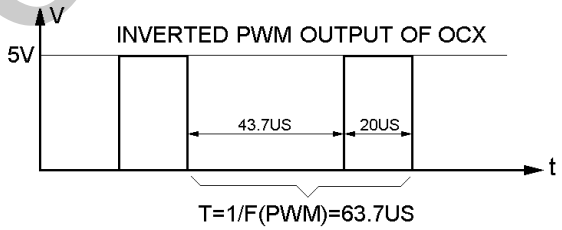
شکل 7-7 مدت زمان نمونه برداری یا زمان تناوب PWM خروجی

اگر از PWM تایمر/کانتر دو استفاده کرده باشیم ، دقت 8,PWM بیتی بوده و محتوای رجیستر مقایسه ای OCR2 نیز بایستی 8 بیتی باشد. با توجه به این که در PWM ، 8 بیتی یک دوره تناوب (نمونه برداری) به 255 قسمت مساوی تقسیم می شود. مدت زمان 63.7US به 255 قسمت تقسیم خواهد شد .

63.7US /255 =0.2US

در این شرایط به عنوان مثال اگر محتوای رجیستر مقایسه ای OCR2 در میکروکنترلر مدولاتور برابر با 100 بوده و خروجی PWM به صورت INVERTED باشد مدت زمان LOW شدن یک سیکل از سیگنال خروجی برابر (100 * 2US=20US) و زمان HIGH شدن آن برابر (63.7US-20US=43.7US) خواهد بود .

شکل 7-8 مدت زمان HIGH و LOW بودن سیگنال PWM خروجی را نشان می دهد .



شکل 7-8 مدت زمان HIGH و LOW بودن سیگنال PWM خروجی

حال اگر در میکروکنترلر دمدولاتور مدت زمان صفر بودن سیگنال ورودی را اندازه بگیریم می توانیم عددی را که در رجیستر مقایسه ای مدولاتور قرار داشته بدست آوریم ، اگر فرکانس کاری تایمر در دمدولاتور برابر با 16MHz بوده و PRESCALE تایمر برابر با 8 باشد فرکانس کاری تایمر برابر خواهد بود با

فرکانس کاری تایمر = 16MHz/8= 2MHz

مدت زمان صعود یک شماره =1 /2MHz = 0.5US

اگر از تایمر یک استفاده کنیم مدت زمان سرریز شدن تایمر برابر خواهد بود با

65536 * 0.5US = 32.76 ms

که این زمان بسیار بیشتر از مدت زمان یک دوره نمونه برداری در مدولاتور یعنی 63.7US است ، پس تایمر یک برای اندازه گیری مدت زمان صفر بودن سیگنال مناسب است. باتوجه به این که در میکروکنترلر مدولاتور مدت زمان نمونه برداری به255 قسمت 0.2US تقسیم شده است و مدت زمان صعود یک شماره در میکروکنترلر دمدولاتور 0.5US می باشد. برای بدست آوردن عدد موجود در رجیستر مقایسه ای مدولاتور بایستی محتوای تایمر یک در دمدولاتور را در 2.5 ضرب کنیم. در این حالت دامنه شکل موج خروجی دقیقا برابر با دامنه شکل موج ورودی خواهد بود ولی در این پروژه عدد موجود در رجیستر مقایسه ای مدولاتور در عدد 2 ضرب شده است .

گاهی اوقات نقاط ضعف انسان به عنوان بزرگترین نقاط قوت برای پیروزی او به شمار می روند. به عبارتی انسان های موفق کسانی هستند که یاد می گیرند چگونه منفی را به مثبت، نقطه ضعف را به نقطه قوت و سنگ های مانع را به سنگ های پله تبدیل کنند. چگونه می شود که یک جودوکار تک دست تمامی حریفان خود را فقط با دانستن یک فن شکست می دهد زیرا بدل فنی که جودوکار از آن استفاده می کند گرفتن دست راست اوست در حالی که او دست راست ندارد. تعدادی از بهترین موسیقی های جهان توسط بتهوون ساخته شده است در حالی که او ناشنوا بود. تعدادی از بهترین اشعار درباره طبیعت توسط میلتون نوشته شده است. فکر می کنید میلتون چه امتیازی داشت؟ او نابینا بود.

با آرزوی موفقیت روز افزون و بیش از پیش برای یکایک شما عزیزان.