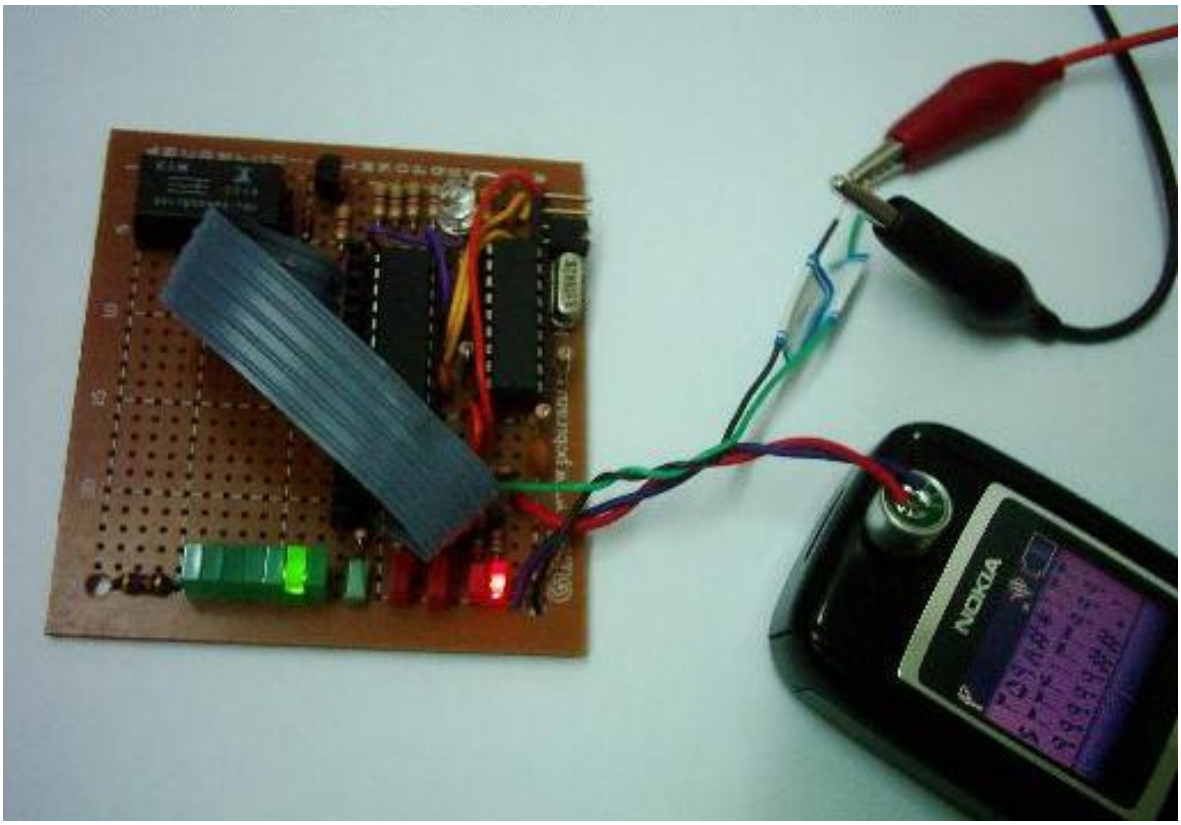
به نام خدا

(آموزش AVR جلسه پنجم)

(Bascom)



مقدمه:

در این جلسه با معرفی تایمرها و شمارنده ها و نیز نحوه مدیریت وقفه ها با Bascom خداحافظی می کنیم. البته این خداحافظی با شروع پژوهش شما همراه می شود. در این ۵ جلسه به خوبی با بسکام آشنا شدیم و از اینجا به بعد برای یادگیری بهتر و تسلط کامل بر این نرم افزار اولین کار مطالعه کامل Help و مشاهده تمام دستورات و امکانات و نیز بررسی تمام مثال های نرم افزار می باشد. قدم بعدی بررسی سورس های موجود در کتاب ها و نیز اینترنت و اجرای آنهاست. دست آخر بهتر است خودمان دست به کار شده و برای تمام ایده هایی

که در ذهن داریم الگوریتمی روی کاغذ پیاده کنیم و با کمی خلاقیت و استفاده از قطعات مناسب و پیاده سازی نقشه اولیه اقدام به طراحی و برنامه نویسی پروژه های دلخواه خود نماییم.

به یاد داشته باشیم که بسکام برای ما محدودیت های زیادی ایجاد می کند؛ اولین محدودیت تجاری بودن آن است و خرید این نرم افزار چند صد دلاری برای نو آموزان غیر ممکن است. محدودیت دوم کامپایل ۴ کیلو کد برای نسخه Demo است که محدودیتی جدی به حساب آمده و نوشتن پروژه های حرفه ای را غیر ممکن می سازد. محدودیت آخر هم Bug های این کامپایلر است که در توافقنامه نصب آن اشاره شده: *"هرگونه استفاده حرفه ای از این نرم افزار در زمینه پروژه های صنعتی از قبیل تاسیسات انرژی هسته ای، ساخت تجهیزات پزشکی، سیستم های هشدار دهنده بلایای طبیعی، سیستم های کنترل ترافیک هوایی و هرگونه دستگاه هایی که کوچکترین اختلال در آنها با مرگ و زندگی افراد ارتباط داشته باشد تنها با رسیک شما خواهد بود."* این بدان معنی است که سازندگان این نرم افزار اطمینان ٪100 به اجرای صحیح کدهای تولیدی ندارند و البته در نسخه های پایین بسکام این نکته به خوبی مشهود است که گاهی اوقات برنامه از لحاظ منطقی درست است ولی مطابق میل ما عمل نمی کند.

در هر صورت بیسکام برای پروژه های معمولی و ساده که تداخل در آنها خطر چندانی ندارد بی رقیب است. در این جلسه برخی از دستورات بسکام برای کنترل تایمرها و شمارنده ها که برای ساخت ادواتی مثل فرکانس متر، شمارنده و... به کار می رود را معرفی می کنیم. همچنین به معرفی امکاناتی از قبیل ارتباط با دنیای خارج توسط پروتکل سریال RS232، شماره گیری تلفن با Tone های DTMF و تولید صدا با PWM که توسط تایمرها کنترل می شوند می پردازیم. ضمناً یکی از ویژگیهای جالب تایمر به نام RTC و توابع آن را که برای ساخت ساعت و تقویم دیجیتال به کار می رود شرح می دهیم. سپس به معرفی وقفه های و مدیریت آنها نیز پرداخته و در نهایت با بررسی چند دستور مهم و کاربردی از قبیل خواندن و نوشتن در حافظه دائمی EEPROM داخلی و نیز توضیحاتی پیرامون سابروتین ها و توابع بحث را خاتمه می دهیم.

(حتماً خطوط ۴ تا ۶ که در جلسه ۴ شرح آن رفت را در ابتدای تمام برنامه ها کپی کنید: تنظیم سایز پشته (Stack) سخت افزاری و نرم افزاری و فریم سایز برای سابروتین ها و متغیرهای محلی)

تایمرها و کانترها (Timer/Counter):

تایمرها و شمارنده ها بخشی از میکرو هستند که وظیفه آنها ایجاد تاخیر و یا ساخت مبنای زمان دقیق و نیز شمارش رویدادها می باشد. شاید این کارها در پایه به نظر کاربردی نباشند چرا که با افـزایش یـک متغیـر هم می توانیم رویدادی را بشماریم و یا با یک حلقه طولانی تاخیر مورد نظر را ایجاد کنـیم. ولـی قابیلـت هـای

خاصی از قبیل تولید DTMF و PWM و نیز ساعت Real Time سبب محبوبیت این بخش از میکرو می شود. این بخش به ۴ بخش تقسیم می شود که به شرح زیر بوده و در هر یک قابلیت مورد نظر را به طور کامل شرح می دهیم:

۱- معرفی تایمرها و کانترها و استفاده بیسیک از آنها؛

۲- شماره گیری تلفن با DTMF؛

۳- تولید PWM برای کنترل موتور، ایجاد صدا و مبدل دیجیتال به آنالوگ؛

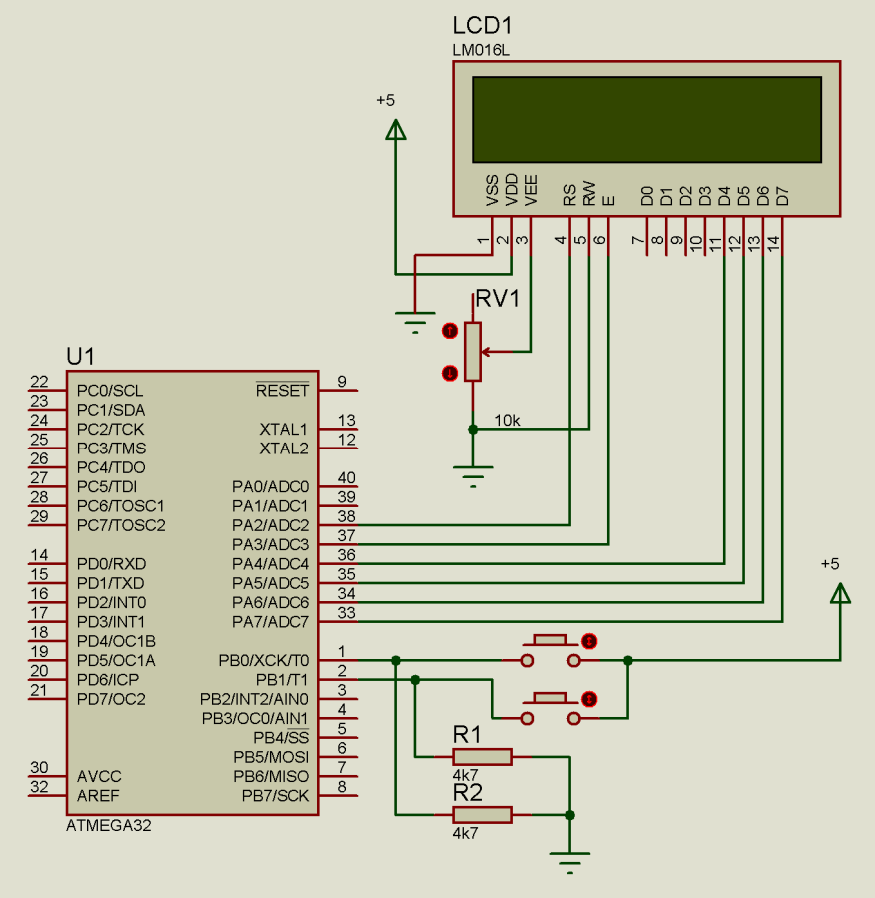
۴- ساعت و تقویم دیجیتال RTC با کریستال ساعت.

معرفی تایمرها و کانترها و استفاده بیسیک از آنها:

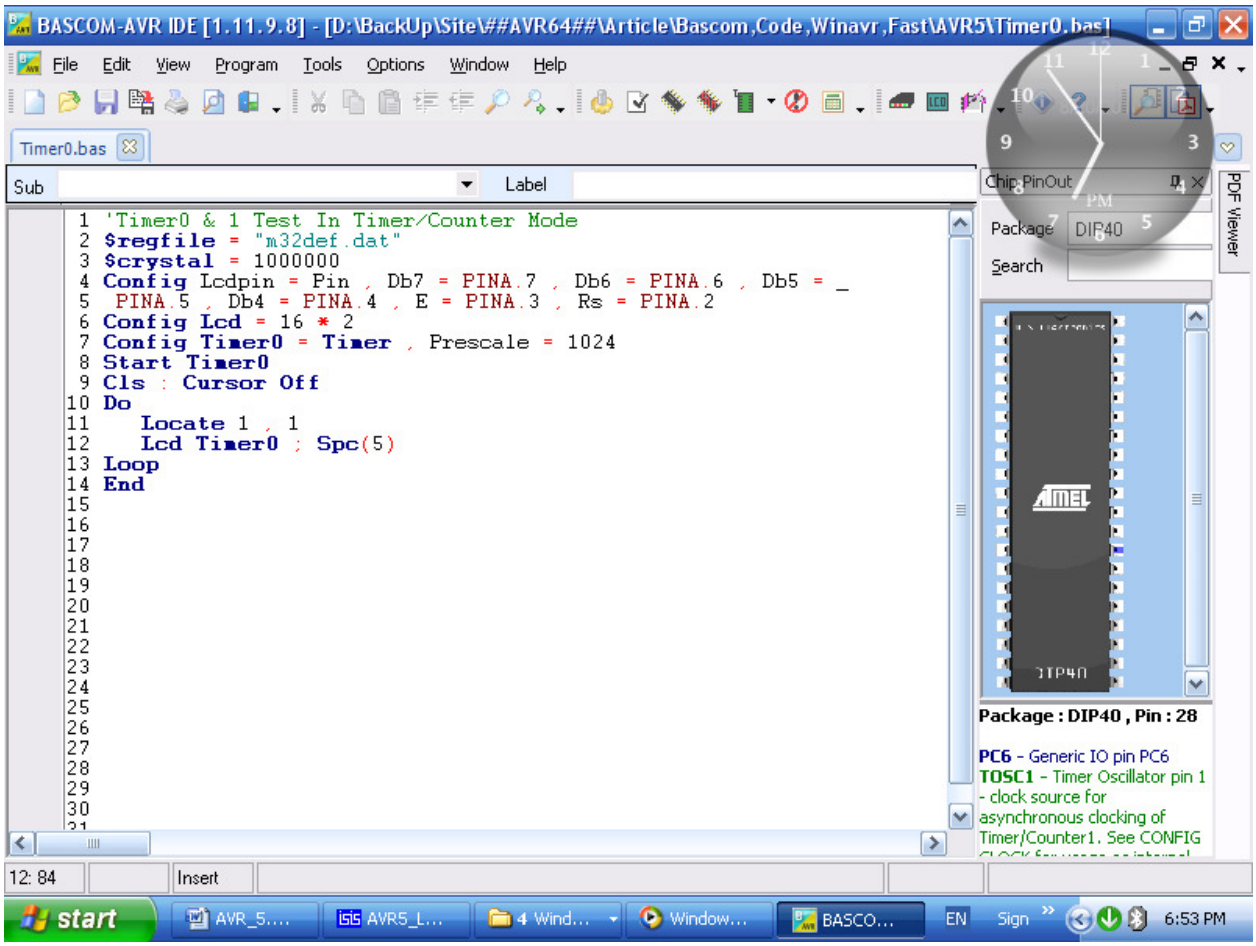
در میکروکنترلر نمونه ATmega32 تعداد ۳ تایمر وجود دارد که آنها را از ۰ تا ۲ شماره گذاری کرده اند. تعداد تایمر ها در میکروهای مختلف متفاوت است و در دیتا شیت آن نوشته شده است. هر تایمر یک رجیستر با پهنای ۸ بیت به به بالا است که قادر به شمارش ۲۵۵ الی ۶۵۵۳۵ رویداد می باشد. معمولاً یکی از تایمرها ۱۶ بیتی بوده و بقیه ۸ بیتی هستند. البته تمام این موارد نسبی است و در مورد هر میکرو تفاوت دارد.

در میکروکنترلر Maga32 سه تایمر وجود دارد؛ تایمر۰ یک تایمر ۸ بیتی بوده و کلاک خود را از CPU یا پین T0 می گیرد. تایمر۱ یک تایمر ۱۶ بیتی بوده و مثل تایمر ۰ کلاک خود را از CPU یا پایه T1 می گیرد. تایمر۲ یک تایمر ۸ بیتی بوده و کلاک خود را از CPU و یا کریستال ساعت بین پایه های TOSC0 و TOSC1 می گیرد و به دلیل مستقل بودن از CPU عموماً برای ساخت ساعت RTC به کار می رود که در مبحث مربوطه شرح داده خواهد شد. در این قسمت به نحوه کار با تایمرهای ۰ و ۱ در Mode معمولی و به عنوان تایمر و شمارنده می پردازیم.

در ابتدا مدار شکل زیر روی Bread Board ببندید:



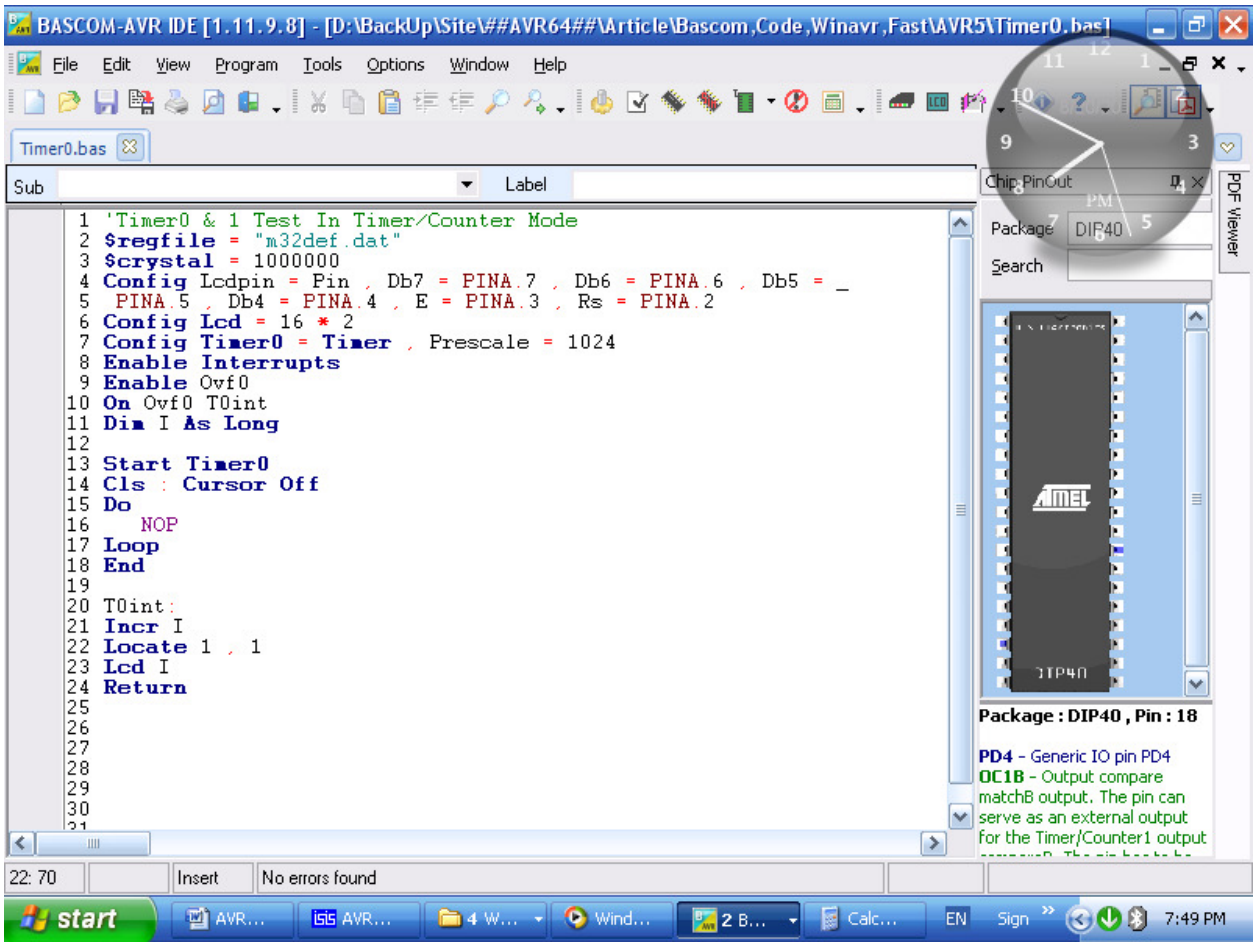
بسکام را باز کرده و کدهای زیر را در آن وارد کنید تا به توضیح تمام دستورات بپردازیم:



خطوط ۱ تا ۶ مربوط به تعریف میکرو، فرکانس CPU و سیم بندی LCD هستند. در خط ۷ تایمر ۰ را به عنوان تایمر (زمانسنج) تعریف کرده و برای کلاک آن کلاک CPU را بر ۱۰۲۴ تقسیم می کنیم. عدد PreScale می تواند مقادیر مجاز: ۱، ۸، ۶۴، ۲۵۶ یا ۱۰۲۴ باشد که برای میکرو های مختلف متفاوت است و برخی از میکروها فقط بعضی از تقسیم کننده ها را دارا می باشند. وقتی که تایمر را به عنوان Timer پیکره بندی می کنیم کلاک آن از داخل سیستم و از CPU دریافت می شود و مجاز به دادن کلاک بر روی پین T0 نیستیم. در این حالت با دستور Start Timer0 تایمر روشن شده و با کلاک 1000000/1024=976 شروع به شمارش می کند. در این آی سی تایمر ۰ یک تایمر ۸ بیتی است که مقادیر قابل شمارش برای آن از ۰ تا ۲۵۵ (یعنی ۲۵۶ رویداد) می باشد. پس از بستن این پروژه مشاهده می کنید که مقدار تایمر با سرعت فوق العاده از ۰ تا ۲۵۵ پر شده و این کار مجدداً ادامه می یابد. با دستور Stop Timer0 میتوانیم تایمر را متوقف کنیم و با دستور Timer0 = 0 مقدار ۰ یا هر عدد دیگری را می توان در تایمر قرار داد. اصولاً از تایمر برای اندازه گیری زمان بین دو رویداد خاص استفاده می شود؛ مثلاً با ۱ شدن یک پین می توان تایمر را روشن کرد و با صفر شدن پین آن را خاموش نمود و با مشاهده مقدار تایمر زمان بالا بودن یک پالس را به دست آورد.

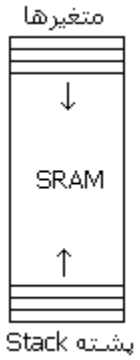
همین کار بالا را می توانید با تایمر ۱ انجام دهید که در این صورت اعداد بین بازه ۰ تـا ۶۵۵۳۵ تغییـر خواهند کرد چرا که تایمر ۱ یک تایمر ۱۶ بیتی است. یکی دیگر از ویژگیهای تایمرها وقفه دار بودن آنهاست. ما در مثال قبل از وقفه استفاده نکردیم. با فعال کردن وقفه های سراسری و نیز وقفه تایمر مـورد نظـر پـس از پـر شدن تایمر که معمولاً ۲۵۵ یا ۶۵۵۳۵ خواهد بود یک وقفه نرم افزاری روی می دهد و برنامه به طور خودکار به یک برچسب خاص که قابل تعریف است پرش می کند. این کار مثل فراخوانی یک تابع یا سابروتین است که در بخش آخر توضیح می دهیم. (در بسکام با دستور Goto می توان به یک بخش خاص از برنامه که با یـک نـام و دونقطه پس از آن مثل :Here مشخص شده است پرش کرد. جدیداً استفاده از دستور Goto در برنامه نویسی های ساخت یافته ممنوع شده است چرا که منطق برنامـه را دچـار سـردرگمی مـی کنـد. همچنـین بـا دسـتور Gosub می توان به قطعه برنامه های کوچکی به نام سابروتین که ابتدای آنها با یک نام و دونقطه شروع شده و پایان آنها با کلمه Return ختم شده است پرش نمود که کمـی منطقـی تـر از Goto اسـت و برنامـه پـس از رسیدن به Return به طور خودکار به همان جایی باز می گردد که از آنجا پرش کرده بود. این اتفاق در وقفه به طور خودکار روی می دهد که در ادامه به توضیح آن می پردازیم)

کدهای برنامه بالا را به صورت زیر تغییر دهید:



تا خط ۷ تغییری ایجاد نشده است. در خط ۸ با دستور Enable Interrupts وقفه های سراسری را فعال می کنیم. در خط ۹ وقفه مربوط به سرریزی تایمر ۰ یعنی Ovf0 که مخفف Over Flow 0 است را فعال می کنیم که البته به جای آن می توان از Enable Timer0 هم استفاده کرد. در خط ۱۰ معین می کنیم که پس از سرریزی و ایجاد وقفه به چه برچسبی پرش شود؛ برچسبی به نام T0int را تعریف می کنیم که این نام اختیاری است. در خط ۱۱ یک متغیر تعریف می کنیم که تعداد وقفه ها را می شمارد و بر روی نمایشگر نشان می دهد.سپس تایمر را روشن کرده، صفحه را پاک می کنیم و مکان نما را خاموش می نماییم. در خطوط ۱۵ تا ۱۷ یک حلقه نامحدود ایجاد کرده ایم که برنامه در آن دور می زند و کاری نمی کند. دستور NOP در داخل حلقه یک دستور اسمبلی است و بودن یا نبودن آن در اینجا تاثیری ندارد و فقط یک سیکل کاری را که معمولاً در فرکانس ۱ مگاهرتز برابر با ۱ میکروثانیه می باشد هدر می دهد. (از این دستور در زبان اسمبلی برای ساخت تاخیر های میکروثانیه ای استفاده می شود و معادل Waitus 1 بسکام است).

برنامه در حلقه دور می زند و مقادیر تایمر زیاد می شوند تا به ۲۵۵ برسند. با رسیدن به ۲۵۵ وقفه اتفاق افتاده و برنامه به طور خودکار وارد خط ۲۰ می شود. در خط ۲۱ یک واحد به متغیر i افزوده شده و در دو خط بعد در مکان Home نمایشگر نشان داده می شود. با رسیدن به دستور Return برنامه به همانجایی که از آن جا پرش کرده بود باز می گردد. آدرس برگشت برنامه در پشته ذخیره می شود و این کار به طور خودکار توسط کامپایلر انجام می گیرد. البته در اسمبلی هم این کار به صورت خودکار انجام می شود و ما نباید نگران ذخیره کردن آدرس برگشت باشیم (فقط رجیسترها باید ذخیره شوند که در جای خود بحث می شود). نکته مهم در بسکام و اسمبلی سایز Stack است که از انتهای Ram به سمت بالا رشد می کند و در صورت تداخل با اطلاعات موجود در SRAM که معمولا در اثر فراخوانی های زیاد به وقع می پیوندد اثرات جبران ناپذیری خواهد داشت. (با دستور $hwstack می توان سایز پشته را تنظیم کرد).

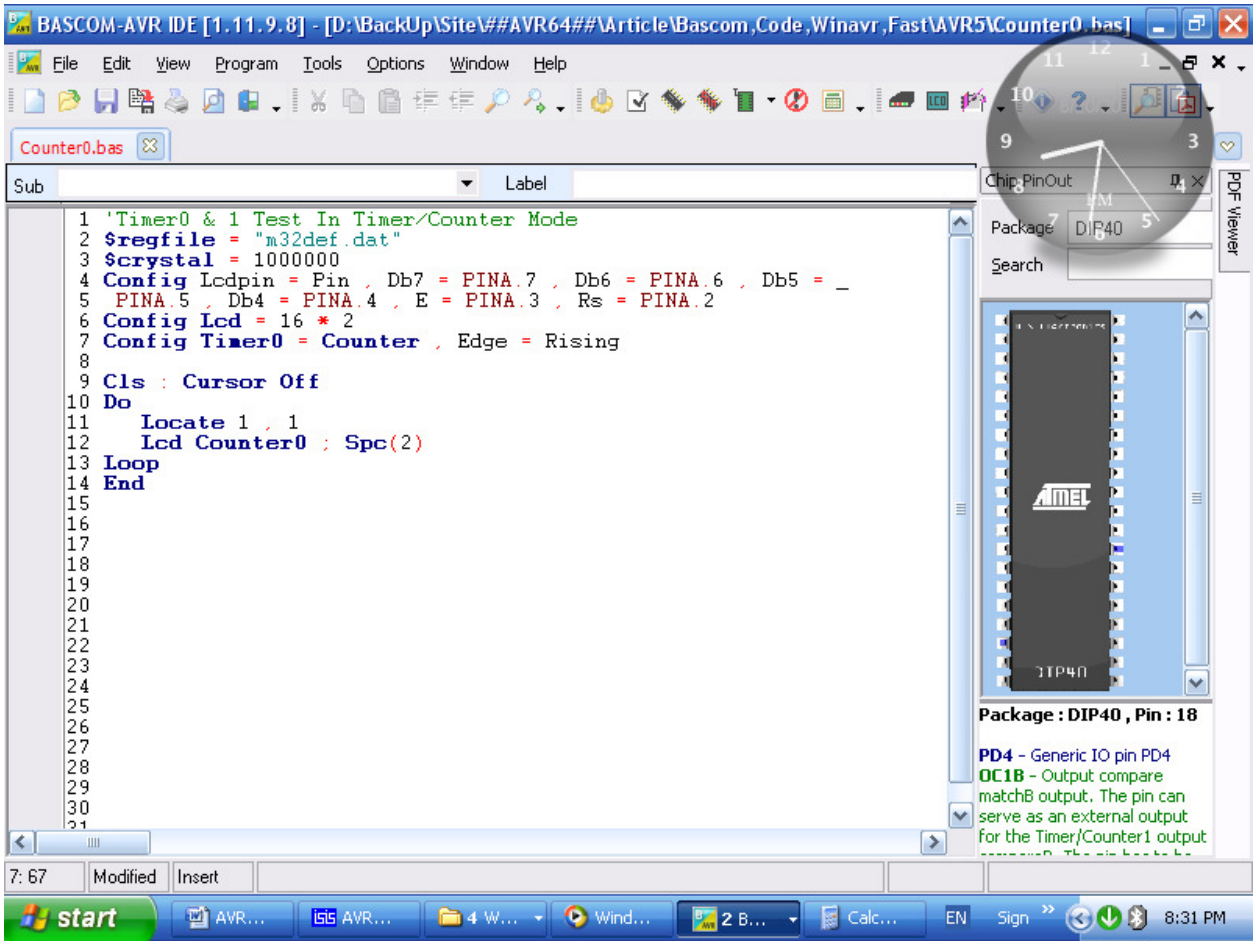


اصولاً آدرس بردار وقفه در ابتدای Flash واقع شده است و با رویداد وقفه برنامه به آن آدرس خاص پرش می کند، در اسمبلی کدهای مورد نظر بایستی در ابتدای فلش و در بردار مربوطه نوشته شوند. ولی در بیسیک به راحتی در هر لیبلی با نام تعریف شده قابل نوشتن است.

در هنگام انجام عملیات مربوط به وقفه، پروسسور به طور خودکار وقفه های سراسری را غیر فعال می کند تا وقفه دیگری اتفاق نیفتد، این معماری AVR سبب می شود که ما مشکلات ایجاد وقفه در وقفه را نداشته باشیم، برای حل مشکلات ناشی از وقفه در وقفه در سیستم عامل ها معمولاً عملیات پیچیده ای انجام می شود که در AVR به سادگی این ویژگی را غیر فعال کرده اند و در واقع بجای حل مساله صورت مساله را پاک کرده اند! توجه داشته باشد که وقفه ها اولویت دارند و اگر دو وقفه به طور همزمان اتفاق بیفتند اولویت با وقفه ای است که عدد کوچکتری دارد. این مبحث را در بخش وقفه های سخت افزاری به صورت عملی توضیح می دهیم. (2011: این بخش نیاز به تست دارد و فرضیه است)

کاربرد دوم تایمر استفاده از آن به عنوان شمارنده با کلاک خارجی روی پین های T0 و T1 است. در این مد نمی توان از تقسیم کننده استفاده کرد و فقط با ۱ پالس تایمر یک واحد افزایش می یابد. در این مد به تایمر شمارنده یا کانتر می گوییم.

کدهای زیر را در بسکام وارد کنید:

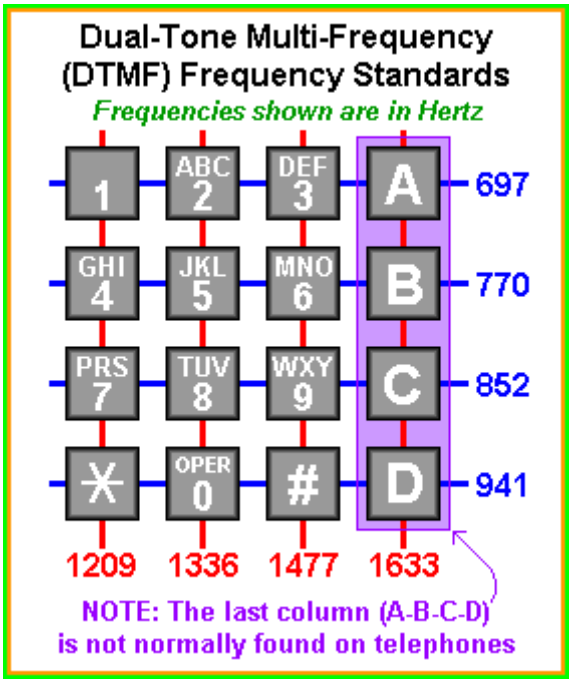


در این برنامه در خط ۷ تغییراتی حاصل شده است. به جای Timer از عبارت Counter استفاده کرده ایم. در ادامه PreScale را حذف کرده و به جای آن از دستور جدید Edge استفاده کرده ایم. لغت Edge به معای لبه است. توجه داشته باشد که ورودی تایمر حساس به لبه بوده و با قرار داده Rising (بالا رونده) و Falling (پایین رونده) می توانیم نحوه حساسیت تحریک پین کلاک ورودی تایمر را تنظیم نماییم. اصولا از کانتر برای کارهای دقیق از قبیل شمارش پالس های کلاک تجهیزات خارجی استفاده می شود و با توجه به نوع کلاک آنها می توان حساسیت لبه را تنظیم نمود. ما در اینجا از لبه بالا رونده استفاده کرده ایم و طبق شماتیک قبل به ازای هر پالسی که به پایه T0 می دهیم مقدار کانتر یک واحد افزایش می یابد. البته به دلیل اینکه کلیدها در هر اتصال چند صد پالس کوچک تولید می کنند تا وصل شوند ممکن است با هر بار فشار کلید چندین پالس بر روی صفحه نمایش ببینیم که طبیعی است. تایمر / کانتر ۱ نیز مانند ۰ است و فقط به دلیل ۱۶ بیتی بودن دامنه شمارش آن بیشتر است. در اینجا بحث ما در مورد تایمرها تمام می شود. در بخش های

بعدی به کاربردهای جالب تایمرها در صنعت می پردازیم. (لازم به ذکر است که برخی از کارهای بالا را با تـایمر ۲ هم می توان انجام داد).

شماره گیری تلفن با DTMF

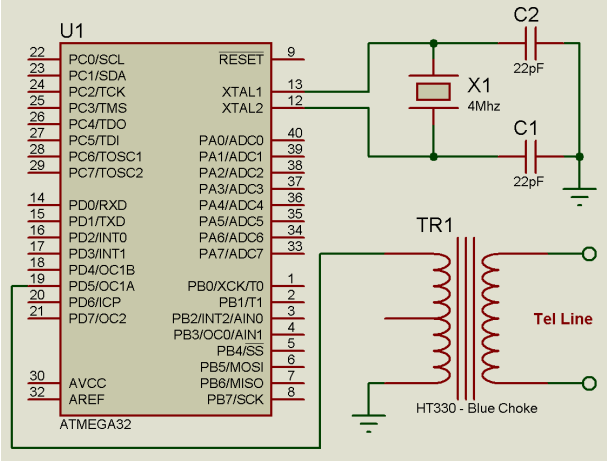
یکی از دستورات جالب بسکام که از تایمر استفاده می کند توانایی ایجاد تن های صوتی DTMF است. لغت DTMF که مخفف Dual Tone Multi Frequency است در صنعت تلفن استفاده زیادی دارد. یکی از کاربردهای اصلی DTMF شماره گیری تلفن است. اصولاً دو راه برای شماره گیری وجود دارد؛ تـن و پـالس. در سیستم های قدیم از روش شماره گیری پالسی استفاده می شد، بدین صورت که برای هر عددی به تعداد همان عدد دو سر سیم های تلفن در فواصل زمانی خاصی قطع و وصل می شد. در سیسـتم هـای جدیـد از تـن هـای صوتی DTMF برای شماره گیری استفاده می شود. همانطوری که در تصویر زیر مشاهده می کنیـد بـا ترکیـب



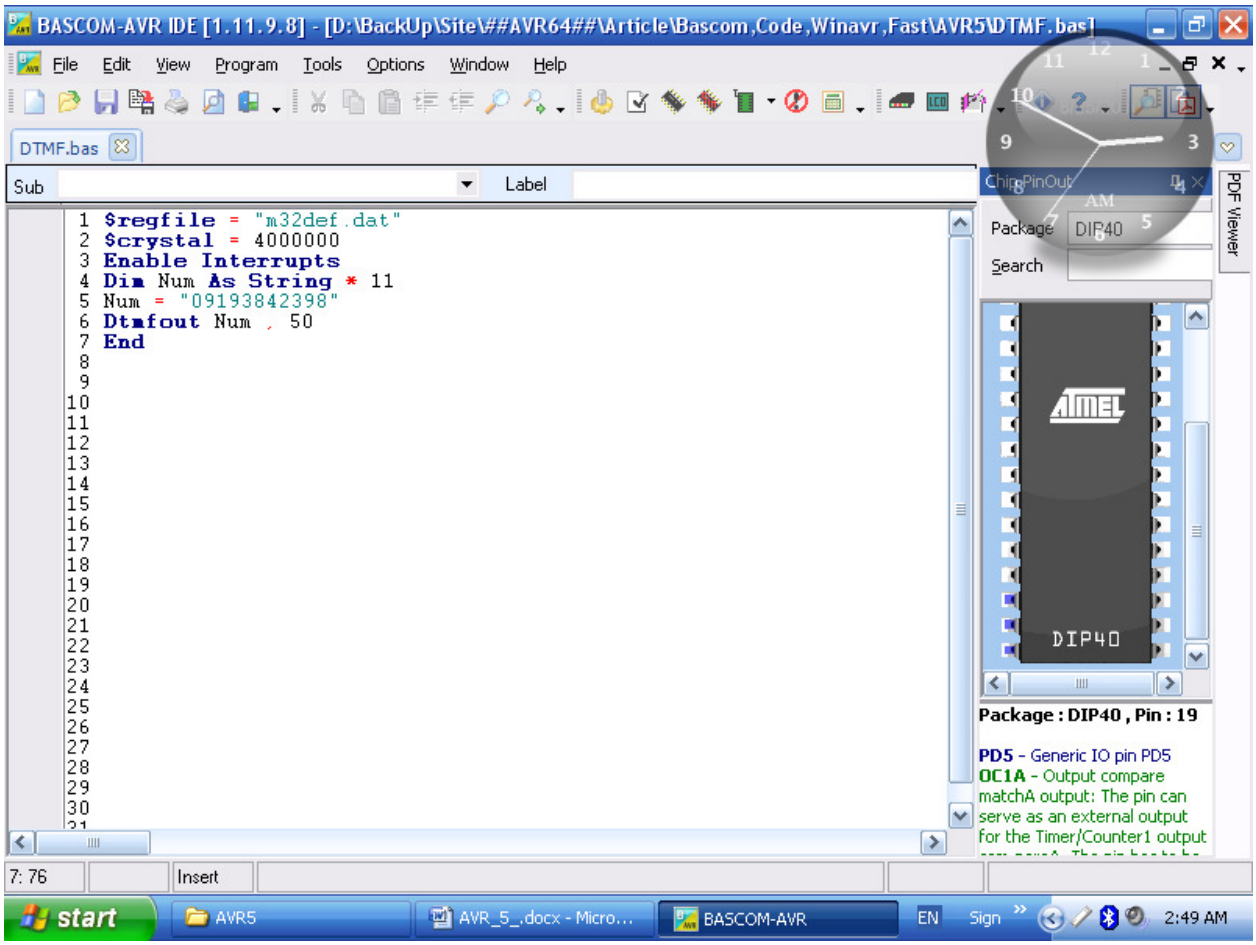
هشت تن صوتی مختلف می توان ۱۶ تـن صـوتی ترکیبـی ایجاد کرد که هر کدام مخـتص یـک کلیـد و یـک شـماره خاص می باشد. معمولاً فقط از ۱۲ تن صوتی بـرای شـماره گیری استفاده می شود و از بین آنها فقط ۱۰ تن مربوط به اعداد ۰ تا ۹ از بقیه مهم ترنـد. همانطوریکـه ملاحظـه مـی فرمایید برای تولید تن صوتی عدد ۱ بایـد هـر دو فرکـانس ۶۹۷ هرتز و ۱۲۰۹ هرتز به طور همزمان با هم تولید شـده شده به بـه طریقـی روی خـط تلفـن تزریـق شـوند. تولیـد همزمان دو تن صوتی توسط عملیات پیچیده ای بـا کمـک وقفه ها و قابلیت های تایمر صورت می گیـرد کـه در ایـن زمینه سورس های فراوانی بـه زبـان C و اسـمبلی موجـود است. در هر صورت این ویژگی توسط بیسکام با دستور ساده Dtmfout انجام می شود. توجه داشته باشید کـه این دستور از تایمر ۱ استفاده می کند و موقع استفاده از آن نباید همزمان از تایمر۱ استفاده کنید. همچنین در ابتدای برنامه حتماً وقفه های سراسری را با دستور Enable Interrupts فعال کنید. به عنوان نکته آخر بـرای عملکرد صحیح این دستور حتماً از کریستال خارجی با فرکانس بین ۴ تـا ۱۰ مگـاهرتز بهـره ببریـد، بـه دلیـل حساسیت فرکانس ها نوسان ساز داخلی جواب نمی دهد. تن صوتی بر روی پایه OC1A آی سی ظاهر می شود و برای تزریق آن به تلفن بهتر است از یک چوک آبی استفاده کنید. ضمناً موقع کار با تلفن توجه داشته باشـید

که قبل از آزاد کردن خط انجام عمل شماره گیری امکان ندارد. برای آزاد کردن خط بایستی دو سر خط توسط یک مقاومت 330Ω الی ۱ کیلو اهم اتصال کوتاه شود که این مقاومت می تواند همان خروجی چوک باشد. البته کمتر شدن مقدار مقاومت ممکن است باعث عدم شماره گیری شود. همچنین قبل از آزاد کردن خط در صورتی که تماسی دریافت کنید ولتاژ متناوب خطرناکی بر روی سیم تلفن جاری می شود که باید نکات ایمنی را رعایت کنید. ما هنگام کار با خط تلفن برای جلوگیری از تماس افراد و جاری شدن ولتاژ Ring، تلفن را با دستور #045*21* روی شماره غیر فعال دایورت می کنیم. (برای خارج کردن از این حالت کد #21# را شماره گیری نمایید). انجام این کار به شدت توصیه می شود، چرا که ولتاژ زنگ تلفن مانند برق شهر مرگبار است. ☠

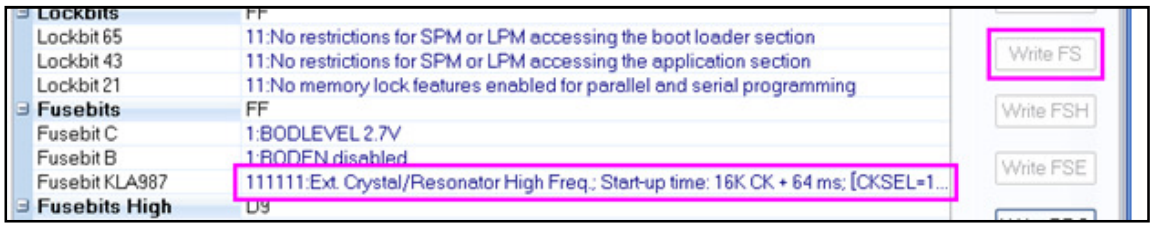
یکی از دلایل استفاده از چوک، ایزوله کردن خط تلفن از مدار دیجیتال است. چوک های آبی رنگ HT330 معمولا در یک طرف ۳ سر و در طرف دیگر دو سر دارند. اهم قسمت سه سر (بدون در نظر گرفتن سر وسط) در حدود ۲۰۰ اهم و مقاومت طرف دو سر آن ۳۳۰ اهم است. البته این اختلاف اهم چندان اهمیتی ندارد و می توانید از چوک یک به یک هم استفاده کنید مشروط بر اینکه مقاومت هر دو طرف حدود ۱۰۰ الی ۵۰۰ اهم باشد، البته این مقادیر امتحان نشده اند و حتماً بایستی روی خطوط مختلف تلفن تست کنید تا از صحت عملکرد آن مطمئن شوید. سمت ۳۳۰ اهم را مستقیماً به خط تلفن وصل کرده و سمت ۲۰۰ اهم را نیز به پایه OC1A و زمین میکرو وصل می کنیم. این نکته مهم را در نظر داشته باشید که با آزاد کردن خط تلفن ۱۵ ثانیه فرصت دارید تا شماره گیری را آغاز نمایید چرا که در غیر این صورت خط به حالت غیر فعال رفته و پس از مدت مشخصی دستگاه شما در مخابرات به طور کامل به حالت Standby می رود. بنابر این توصیه می شود که در پروژه خود سمتی را که به تلفن وصل می شود با خروجی یک رله سری کنید و قبل از شماره گیری با دستوری رله را وصل کنید و به مدت ۱ ثانیه صبر کنید (Wait 1) تا خط آزاد شود سپس با دستور Dtmfout اقدام به شماره گیری نمایید و پس از اتمام شماره گیری ۳۰ ثانیه صبر کنید تا در سمت مقابل ارتباط برقرار شود و پس از ارسال دیتا و یا ... رله را قطع کنید. دستور Wait 1 فوق العاده حیاتی بوده و حتماً قبل از شماره گیری باید برای تثبیت خطوط و آمادگی دستگاه های سمت مخابرات این کار را انجام دهید. البته ما در این پروژه ساده از رله استفاه نمی کنیم و چند ثانیه پس از وصل میکرو به خط تلفن، تغذیه را وصل می نماییم. برای شروع کد نویسی شماتیک زیر را ببندید: (به دلیل دقت فرکانس *بدون کریستال خارجی جواب نمی دهد*).



بسکام را باز کرده و کدهای زیر را در آن وارد کنید:

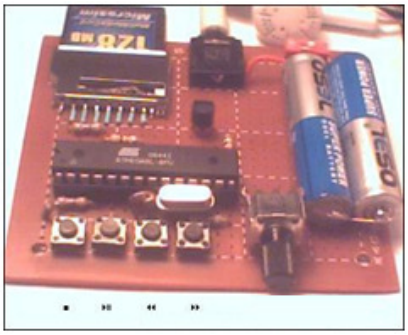


در این پروژه حتماً از کریستال خارجی بین ۴ تا ۱۰ مگاهرتز و خازن های ۲۲ پیکوفاراد بین پایه هـای Xtal1,2 و زمین استفاده کنید، فیوز بیت را نیز مطابق زیر بر روی کریستال خارجی قـرار دهیـد و روی دکمـه Write FS کلیک نمایید.



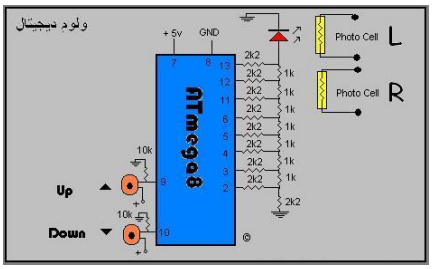
با فعال سازی وقفه ها و دستور Dtmf string, delay می توانید شماره رشته string را روی خـط OC1A بفرسـتید. مقدار Delay زمان حضور هر تن است و مقدار ۵۰ میلی ثانیه کفایت می کند. ضمناً بجای رشته می توانید از یک متغیر Byte بـا مقادیر معتبر ۰ تا ۱۵ برای ارسال تک تن استفاده نمایید. (تن های ۱۲ تا ۱۵ کاربردی ندارند).

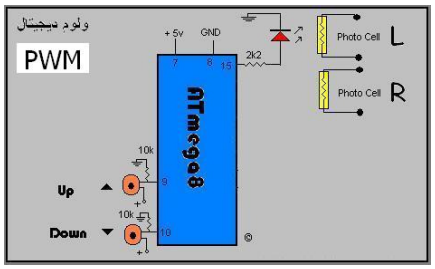
تولید PWM برای کنترل موتور، ایجاد صدا و مبدل دیجیتال به آنالوگ



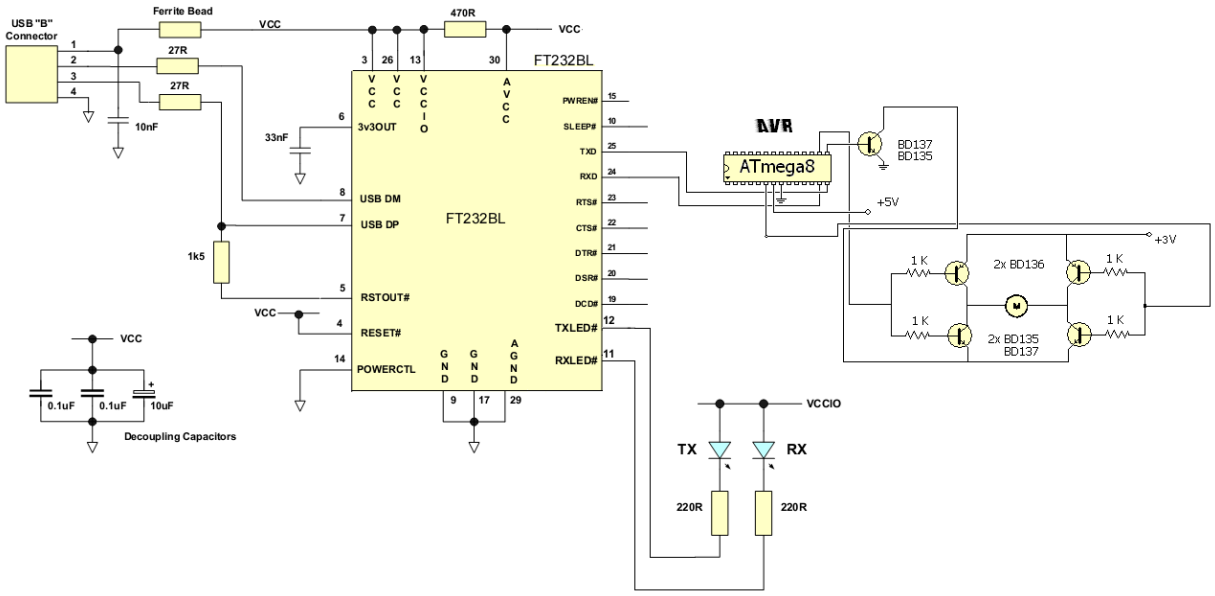
یکی دیگر از کاربردهای تایمر تولید موج پیوسته PWM می باشـد. مـوج PWM را می توان توسط تایمر با یک فرکانس خاص و ثابت تولیـد کـرد و بـا قـرار دادن اعداد بین ۰ تا ۲۵۵ در رجیسـتر PWM مقـدار Duty Cycle یـک مـوج را تغییر داد. تغییر Duty Cycle به منزله تغییر زمان بالا بودن یـک پـالس بـه زمـان پایین بودن آن است. چنین خروجی را می توان با تغییر اندکی بـه عنـوان خروجـی آنالوگ با دامنه متغییر مورد استفاده قرار داد. همچنین از این روش برای اولـین بـار در سال ۸۷ برای پخش بایت های فایل صـوتی Wave کـه از MMC خوانـده مـی شد استفاده کردیم که این پروژه بعداً تکمیل و سورس آن آزاد شد و به عنوان Wave Player در اینترنـت بـرای اسـتفاده عمـوم قرار گرفت.

از جمله ایده های جالب دیگر نیز پروژه ولوم دیجیتال بود که با استفاده از PWM طراحی شده بود که در همـان اوان ارائـه شـد و نمونـه ای از کـاربرد PWM به عنوان D2A را نشان مـی دهد (شکل سمت چپ همان سیستم را بدون PWM و با استفاده از شبکه تقسیم مقاومتی نشان می دهد).

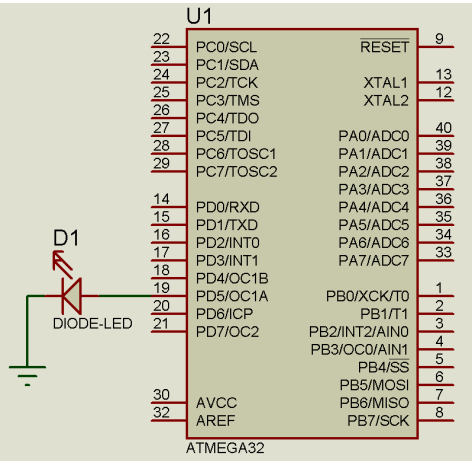




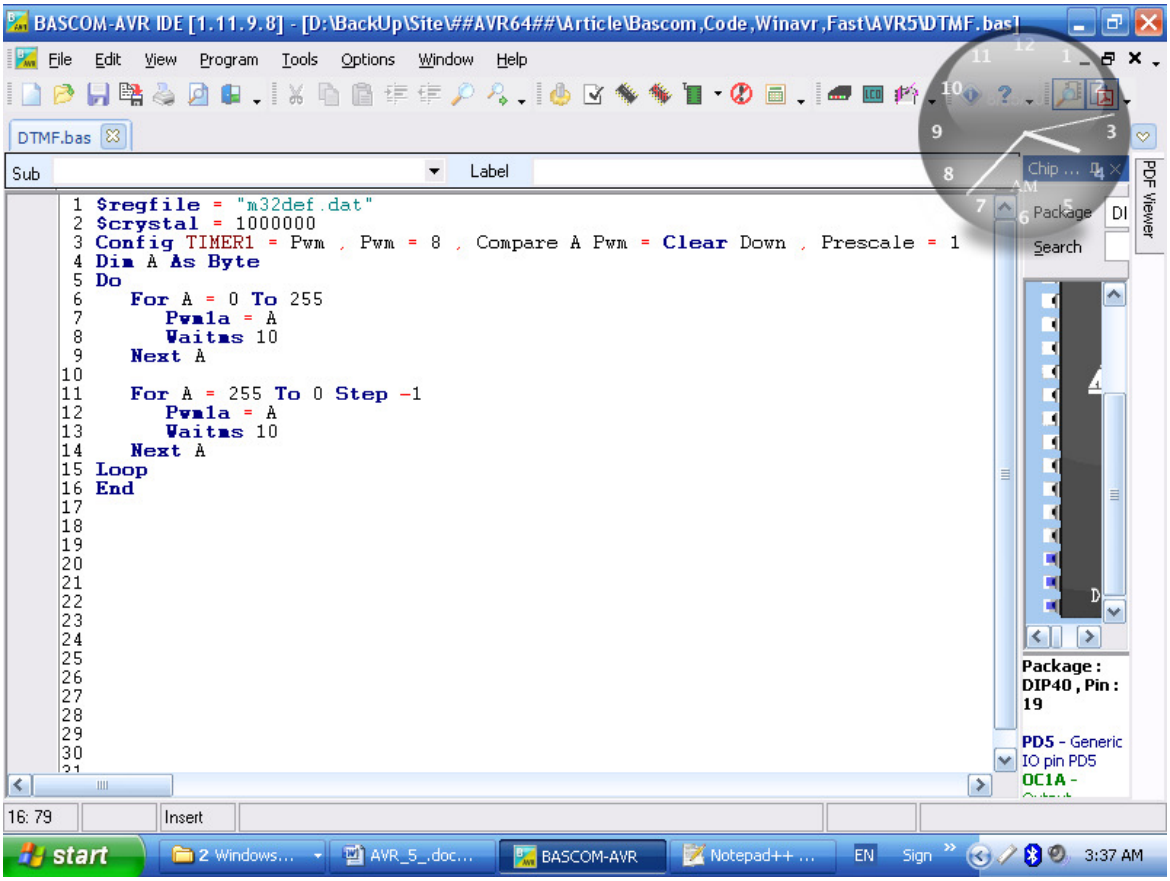
کاربرد سوم این موج کنترل موتور DC است که با استفاده از آن پروژه هـای مختلفـی از قبیـل کنتـرل موتور با میکرو، کنترل موتور با پورت سریال PC و کنترل موتور با پورت USB طراحی و اجرا شده است کـه در شکل زیر بعد شماتیک پروژه کنترل موتور با پـورت USB را بـرای نشـان دادن نحـوه اسـتفاده از ایـن سیسـتم مشاهده می نمایید:



در کل با استفاده از PWM کارهای متفاوتی را می توانید انجام دهید، ما برای سادگی کار و تست ایـن بخش از میکرو تنها به کم و زیاد کردن نور یک LED بسنده می کنـیم. توجـه داشـته باشـید کـه در برخـی از میکروها دو خروجی OC1A و OC1B داریم که نشان دهنده دو کانال مجزای PWM اسـت. مـا فعـلاً فقـط از کانال a استفاده می کنیم. در طی برنامه نویسی شرح می دهیم که چگونه به کانال b سوئیچ نماییم. برای تست PWM شماتیک زیـر را بر روی برد بورد پیاده کنید:



بسکام را باز کرده و کدهای زیر را در آن وارد نمایید:



نظر به اینکه در این پروژه مقدار دقیق فرکانس برای ما مهم نیست از نوسان ساز RC داخلی ۱ مگاهرتز استفاده می کنیم (فیوز بیت را روی حالت Int RC 1Mhz قرار داده و کریستال و دو خازن ۲۲ پیکوفاراد را از روی برد حذف نمایید). با توجه به ۱۶ بیتی بودن تایمر ۱ می توانیم با تغییر عدد مقابل PWM به ۸، ۹ و یا ۱۱ از PWM با تعداد بیت دلخواه استفاده کنیم (فقط همین ۳ مقدار مجاز هستند)، ضمناً با تغییر عبارت Compare A PWM به Compare B PWM می توانیم خروجی را بر روی پایه OC1B بفرستیم مشروط بر اینکه در برنامه مقدار مورد نظر را در رجیستر Pwm1b قرار دهیم. توجه داشته باشید که فرکانس PWM ثابت است و فقط دامنه آن است که تغییر می کند. برای به دست آوردن فرکانس PWM بایستی مقادیر PWM را که می تواند ۸، ۹ یا ۱۰ بیت باشد با مقادیر Prescale که می تواند ۱، ۸، ۶۴، ۲۵۶ یا ۱۰۲۴ باشد ترکیب نمایید. با توجه به فرکانس CPUی سیستم فرکانس PWM در مدهای مختلف از فرمول های زیر حاصل می شود:

۸ بیتی: فرکانس سیستم تقسیم بر (510*Prescale)

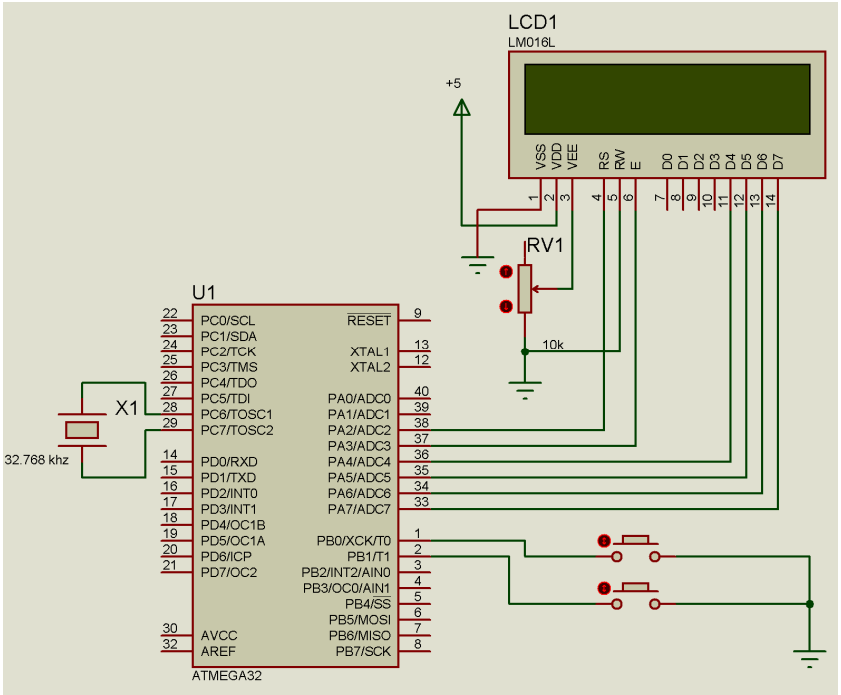
۹ بیتی: فرکانس سیستم تقسیم بر (1022*Prescale)

۱۰ بیتی: فرکانس سیستم تقسیم بر (2046*Prescale)

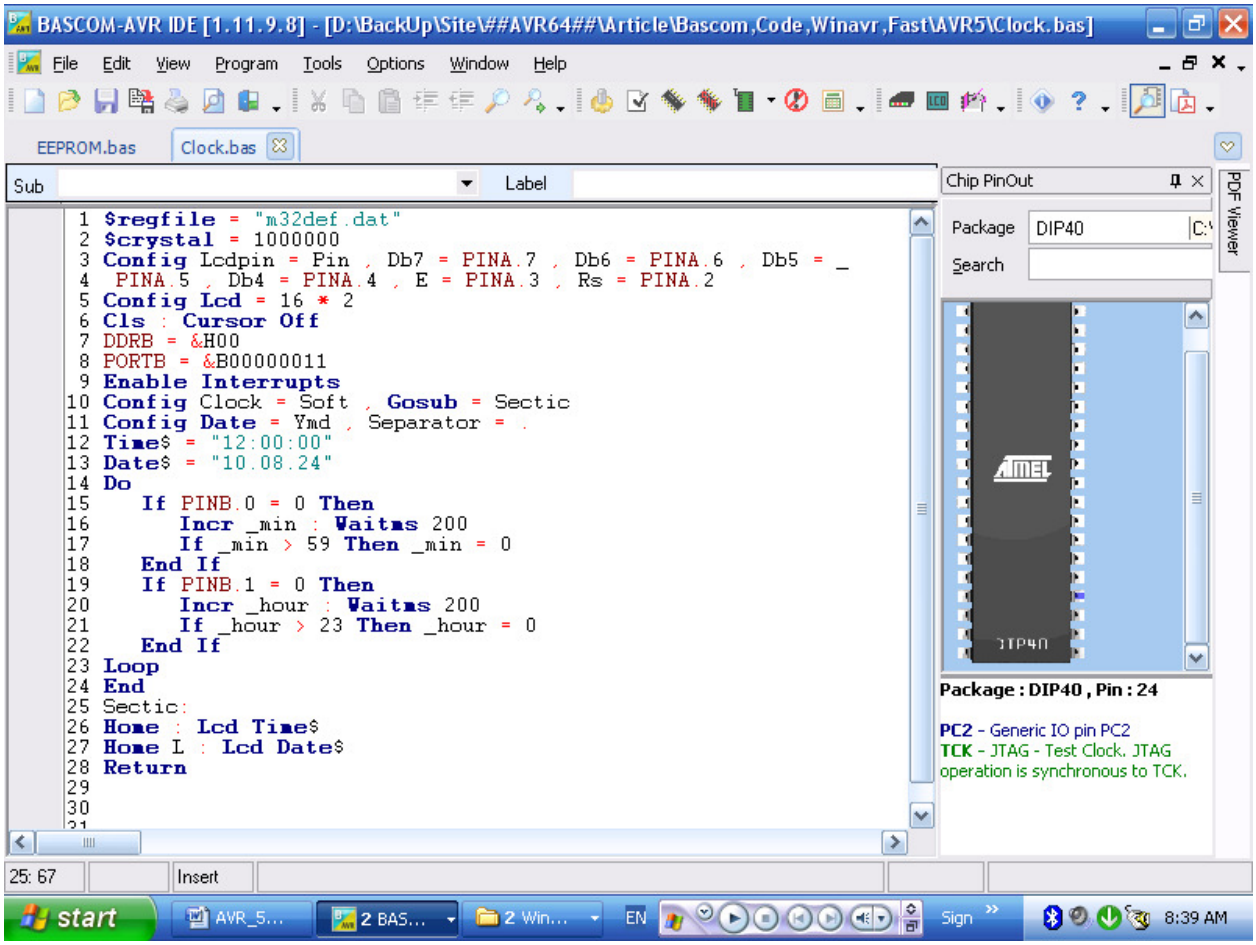
پروژه فوق نور یک LED را کم و زیاد می کند و فرکانس آن ۱۹۶۰ هرتز است.

ساعت و تقویم دیجیتال RTC با کریستال ساعت

یکی از ویژگیهای تایمر شماره ۲ قابلیت دریافت کلاک از یک کریستال مجزا در بین پایه های TOSC1 و TOSC2 است. این ویژگی سبب می شود که تایمر بتواند جدا از CPU به فعالیت بپردازد و میکروکنترلر را به یک پردازنده دو هسته ای تبدیل کند. البته هسته دوم قابلیت زیادی نداشته و تنها استفاده مفید آن اینست که می تواند به عنوان یک مولد پالس دقیق پیکره بندی شود. برای این منظور از یک کریستال ساعت با فرکانس 32.768Khz در بین پایه های TOSC1,2 استفاده می شود. می توانیم تایمر را به صورت دستی به گونه ای پیکره بندی کنیم که هر ثانیه یک بار به روتین وقفه پرش کرده و در آنجا متغیری را که مربوط به ثانیه است یک واحد افزایش دهیم و سپس برای مشاهده دقیقه عدد حاصل را ۶۰ تقسیم کنیم و... اما بسکام کارها راحت تر کرده و با قرار دادن پیکره بندی مجازی خاصی به نام Clock نوعی ساعت و تاریخ زیبا طراحی کرده است که البته تاریخ آن میلادی بوده و برای ما کاربرد چندانی ندارد. اما ساعت آن فوق العاده کارها را آسان می کند. برای نشان دادن سادگی دستورات این ساعت چیزی بهتر از ارائه یک مثال واقعی نیست، پس مدار زیر را روی برد بورد ببندید تا برنامه نویسی را آغاز کنیم:



بسکام را باز کرده و کدهای زیر را در آن وارد نمایید:



دو خط ۷ و ۸ دستورات اسمبلی هستند که پورت B را ورودی تعریف کرده و مقاومت Pull-up پین های B0 و B1 را فعال می کنند، این دو پین به دو کلید برای تنظیم دقیقه و ساعت می روند. در خط بعدی وقفه های سراسری را فعال کرده ایم چرا که تایمر در مد وقفه کار می کند. در خط بعدی ساعت نرم افزاری بسکام را پیکره بندی می کنیم، نام لیبیل Sectic نباید چیز دیگری باشد. برنامه سر هر ثانیه به لیبل Sectic پرش می کند. در خط بعدی نحوه نمایش سال y، ماه m و روز d را در رشته Date$ مشخص کرده و نوع جداکننده را که می تواند یکی از انواع /، - یا . باشد مشخص می کنیم. در دو خط بعدی مقادیر پیش فرضی را برای ساعت و تاریخ معین می نماییم. توجه داشته باشید که ساعت بسکام نظامی و ۲۴ ساعتی است. ضمناً اعداد Date$ را نیز به ترتیب تعریف بالا و با رعایت نوع جدا کننده که در اینجا نقطه (.) در نظر گرفته شده است تعیین نمایید. در حلقه به دستوراتی برای تنظیم ساعت و دقیقه پرداخته ایم. با تعریف ساعت به طور خودکار متغیر های _sec، _min، _hour، _day، _month و _year تعریف می شوند که با تغییر آنها در محدوده

مجاز می توانید مقادیر درون رشته های Time$ و Date$ را تنظیم کنید. ضمناً برای افزودن قابلیت آلارم نیز متغیرهای یاد شده را با مقادیر دلخواه مقایسه نمایید. متغیرهای سیستمی همیشه با آندرلاین شروع می شوند. در این برنامه در لیبل Sectic که هر ثانیه اتفاق می افتد مقادیر ساعت و تاریخ در خطوط بالا و پایین نمایشگر نشان داده می شوند. شایان ذکر است که Home مکان نما را به خط اول و سطر اول می برد و Home L مکان نما را به سطر دوم و ستون اول میل می دهد. به جای آن می توانید از lowerline هم استفاده کنید و یا دستور Locate 2,1 که مکان نما را به هر سطر و ستونی منتقل می نماید.

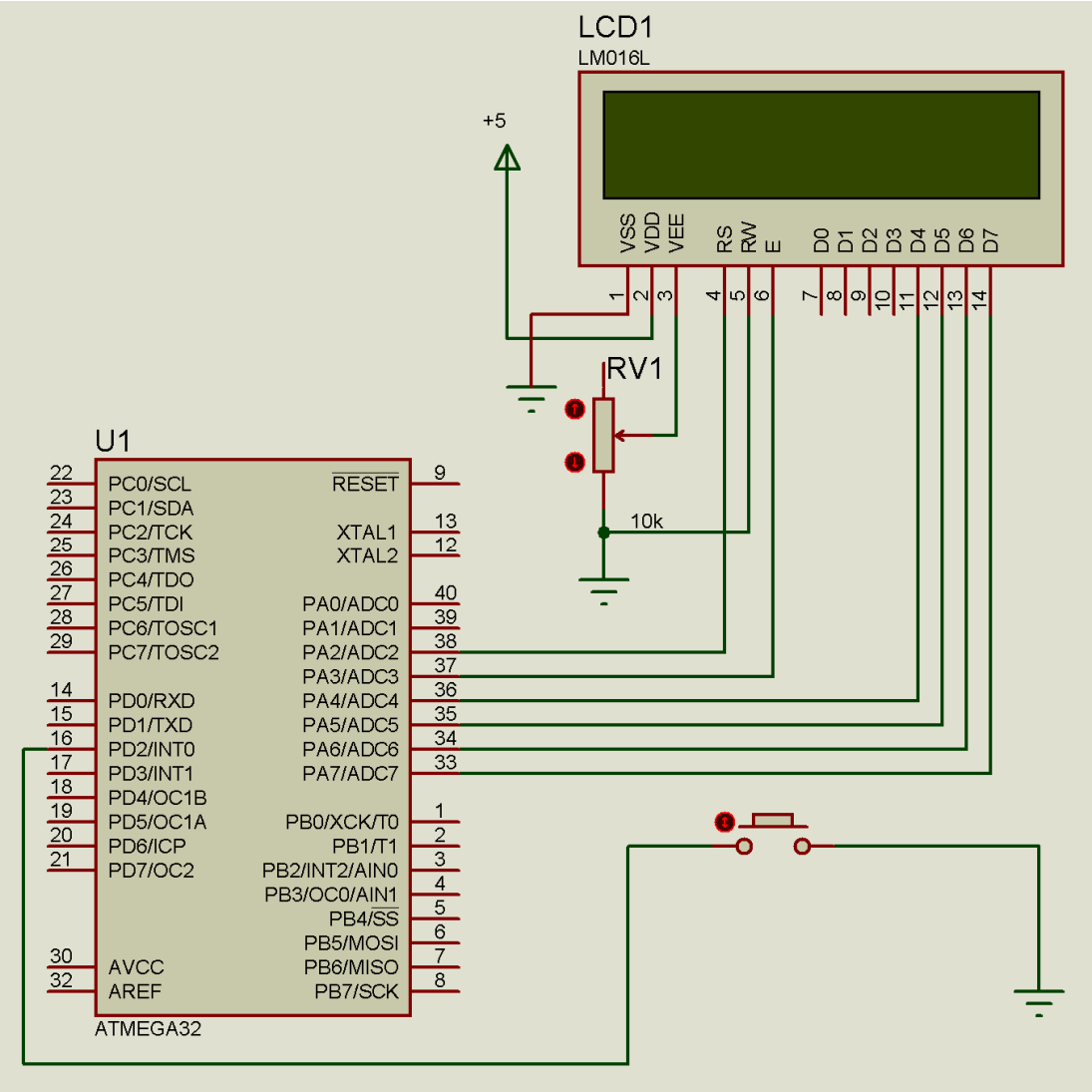
(برای ساخت ساعت دقیقتر می توانید از آی سی ساعت DS1307 استفاده کنید، همراه با باتری بکاپ. این پروژه با تاریخ شمسی و میلادی در بخش پروژه های AVR64.com قرار داده شده است)

وقفه ها (Interrupts):

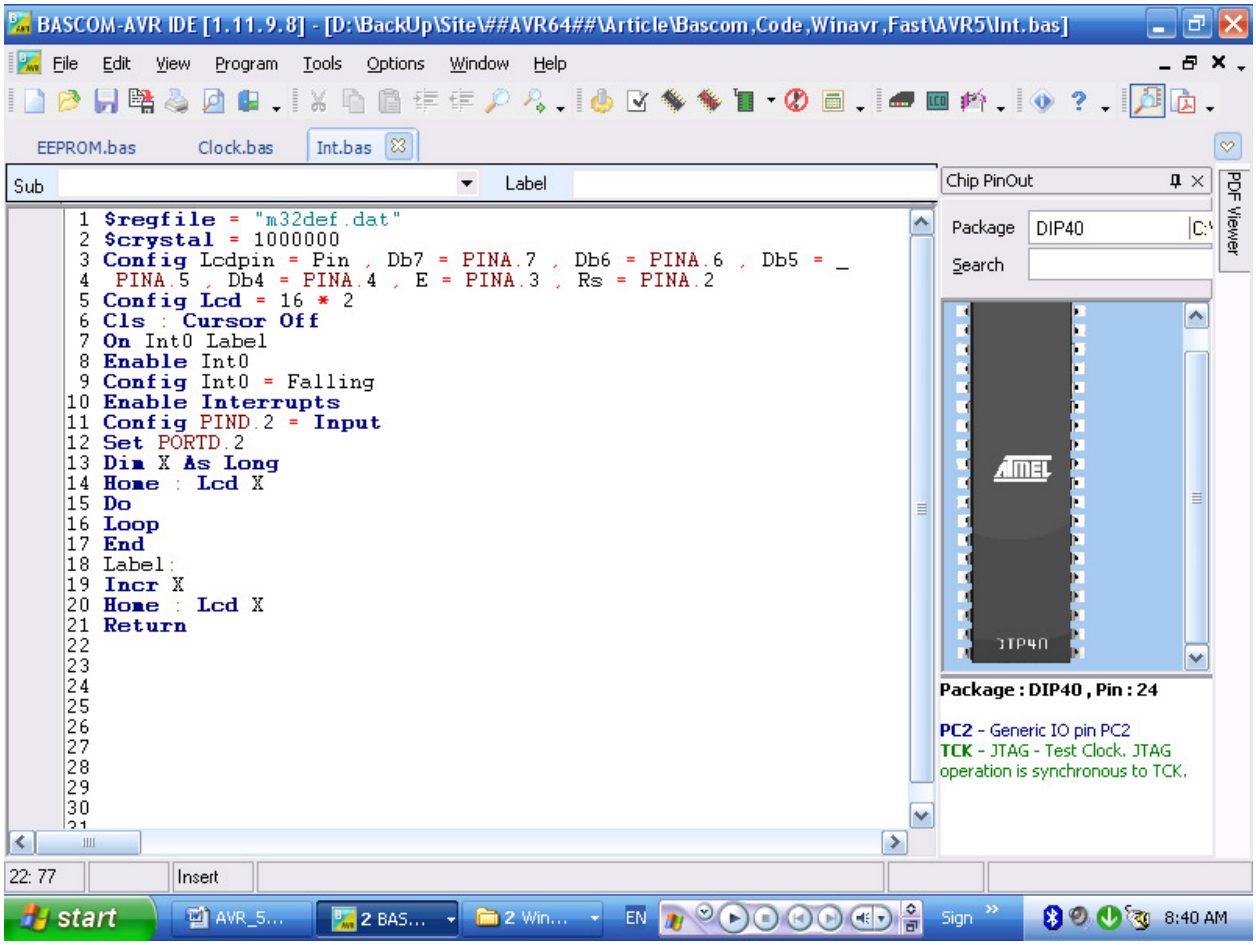
در این قسمت به نحوه کار با وقفه های سخت افزاری می پردازیم، اصولاً در سیستم های کامپیوتری دو نوع منبع وقفه وجود دارد: سخت افزاری و نرم افزاری. منابع وقفه نرم افزاری مانند وقفه در اثر پر شدن تایمر و یا وقفه ساعت در هر ثانیه را مشاهده نمودیم، وقفه های دیگری مانند وقفه تطابق مقایسه و پورت سریال و SPI هم وجود دارند که در پروژه های ساده کاربردی ندارند. اینک به وقفه های سخت افزاری می پردازیم. وقفه های سخت افزاری به پین های خاصی از میکرو میکرو ختم می شوند که معمولاً با عباراتی نظیر INT0، INT1، INT2 و... برچسب خورده اند. وقفه ها می توانند حساس یه سطح و یا لبه باشند. با دستور `Config Int0 = Falling` اینتراپت شماره ۰ حساس به لبه پاین رونده می شود و با پایین رفتن از + به زمین فقط یک بار روتین وقفه اتفاق می افتد. به جای Falling می توانیم از Rising استفاده کنیم که در این صورت وقفه حساس به لبه بالا رونده می شود. در برخی از چیپ ها می توان از Change استفاده کرد که در این صورت اینتراپت در هر دو لبه اتفاق می افتد و شبیه RAM های DDR2 می شود که از یک کلاک ساده برای سرعت

دو برابر استفاده می کنند. دستور چهارمی هم وجود دارد و آن Low level است؛ در این حالت هر گاه پین اینتراپت در حالت پایین قرار گیرد روتین وقفه به طور مرتب اجرا می شود (برخلاف Falling و Rising که یک بار اجرا می شدند). نکته دوم اولویت وقفه ها و عدم ایجاد چند وقفه به طور همزمان است. اگر دو پالس به پین های وقفه های شماره ۰ و ۱ به طور همزمان برسد وقفه شماره ۰ اجرا می شود و اولویت با عدد کوچکتر است. ضمناً در AVR وقتی که یک وقفه اتفاق می افتد وقفه های سراسری غیر فعال شده و تا پایان روتین وقفه جاری هیچ وقفه دیگری اجازه روی دادن ندارد. همانطوریکه در مبحث تایمر ذکر شد در سیستم عامل کامپیوتر اینگونه نیست و با تکنیکهای ویژه ای امکان ایجاد وقفه در وقفه را داریم.

برای تست وقفه، مداری را مطابق شکل زیر ببندید:



بسکام را باز کرده و کدهای زیر را در آن وارد کنید:



در خط ۷ نام لیبلی را که وقفه شماره صفر پس از روی دادن به آنجا پرش می کند آورده ایم. در خط ۸ وقفه ۰ را فعال کرده و در خط ۹ آن را به صورت حساس به لبه پایین رونده تنظیم می کنیم. در خط ۱۰ هم وقفه های سراسری را فعال می کنیم. با توجه به اینکه وقفه ۰ روی پین D.2 قرار دارد و از حالت حساس به لبه پایین رونده استفاده کرده ایم این پین را ورودی تعریف کرده و مقاومت Pull up آن را فعال می کنیم. وجود این دو خط در صورتی که از یک مقاومت پول آپ خارجی استفاده کنیم مهم نیست و برای حالت Rising حتماً باید حذف شود. در این برنامه در حالت کد بالا یعنی Falling با هر بار فشار کلید یک واحد به مقدار X افزوده شده و بر روی نمایشگر نشان داده می شود (به دلیل وجود نویزکلید در هر بار فشار چندین عدد به X افزوده می شود که طبیعی است). حال اگر در خط ۹ از Low level به جای Falling استفاده کنیم با نگه داشتن کلید، وقفه با طور مرتب اتفاق می افتد و عدد با سرعت بالایی زیاد می شود. با به کار بردن Change با فشار دادن کلید یک واحد افزوده شده و با رها کردن آن هم یک واحد افزوده می شود؛ مانند رویداد ماوس در

VB. با حذف دو خط ۱۱ و ۱۲ و قرار دادن یک مقاومت بین پایه INT0 و زمین و اتصال کلید به + و با تغییر عبارت جلوی خط ۹ به Rising با هر بار فشار کلید یک واحد به x افزوده می شود و این حالت حساس به لبه بالا رونده است.

حافظه دائمی (EEPROM):

یکی از ویژگیهای اکثر میکروهای AVR وجود حافظه EEPROM در داخل چیپ است. اصولا میکروها از حافظه SRAM برای نگهدای موقتی متغیرها استفاده می کنند که با قطع برق پاک می شود. ولی گاهی اوقات لازم است که اعدادی پس از قطع برق نیز در حافظه سیستم باقی بمانند؛ مثل مدارات قفل رمز دیجیتالی برای نگهداری رمز و یا کنترل کننده های آسانسور برای نگهداری شماره طبقه جاری و بسیاری از پروژه هایی که باید به نوعی "حافظه" داشته باشند و تمام Setting ها را در خود نگه دارند. معمولاً این اعداد بسیار کوچک هستند به همین دلیل حافظه EEPROM میکروها نیز اغلب در محدوده Byte است. مثلاً M32 در حدود 1024 بایت حافظه EEPROM داخلی دارد. برای نوشتن در حافظه EEPROM دو راه وجود دارد که یکی از آنها نوشتن در آدرسی خاص است که کمتر استفاده می شود و Syntax آن به صورت WRITEEEPROM var , addr و READEEPROM var, addr است که آن بجای Var متغیر و به جای addr آدرس آن را قرار می دهند.

ولی روش دوم که ساده تر و پر کاربرد تر است بدین صورت می باشد که در همان موقع تعریف متغیر آن را از نوع EEPROM انتخاب می کنیم مثل: Dim A as Eram Byte با استفاده از کلمه Eram قبل از نوع متغیر آن را به عنوان EEPROM تعریف می کنیم. نکته فوق العاده مهم اینست که متغیر های EEPROM را قبل از هر گونه استفاده را در همان ابتدای برنامه باید در یک متغیر معمولی SRAM ذخیره کنید و پس از هر دستور انتقال، ذخیره سازی و یا خواندن بایستی از دستور Waitms 5 بلافاصله پس از دستور مربوطه استفاده کنید تا عملیات نوشتن و خواندن تکمیل گردد چرا که خواندن و نوشتن در EEPROM اندکی زمان می برد. نکته دیگر محدودیت خواندن و نوشتن در سلول های EEPROM است؛ هر سلول EEPROM قابلیت ۱۰۰ هزار بار خواندن و نوشتن دارد پس هرگز متغیرهای EEPROM را در حلقه قرار ندهید. البته اگر شرط اول را رعایت کرده و قبل از استفاده از متغیر های EEPROM آنها را یک متغیر SRAM کپی کنید امکان استفاده اشتباهی از متغیر EEPROM در حلقه از بین می رود.

در نمونه کد زیر نحوه استفاده از متغیر EEPROM را نشان داده ایم:

```
$regfile = "m32def.dat"
$crystal = 1000000
Dim A As Byte
Dim B As Eram Byte
A = B دانلود
Waitms 5
A = A + 1بدلیل اینکه کار کردن با متغیرهای ای ای پرام به طور مستقیم امکان پذیر نیست
B = A آپلود
Waitms 5
End
```

معرفی پورت سریال UART و پروتکل RS232

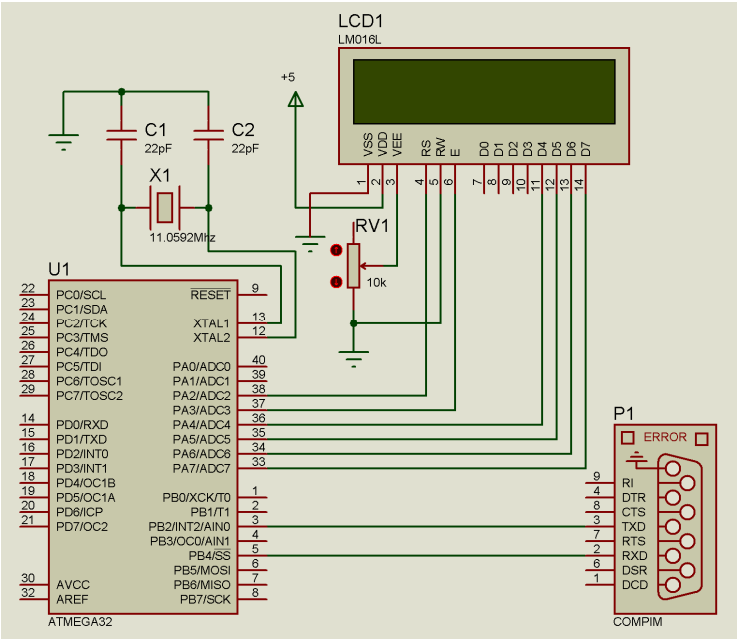
اصولاً هر سیستم کامپیوتری کوچک و بزرگ بدون داشتن قابلیت ارتباط با سایر دستگاه ها کارآیی زیادی نخواهد داشت. از همان آغاز اختراع کامپیوترها استفاده از مودم و کابلهای شبکه و در کل انتقال اطلاعات از جذابیت و محبوبیت زیادی برخوردار بوده است. تلفن های همراه نیز هرچند در اساس برای ایجاد ارتباط بیان اشخاص ساخته شده اند و Base آنها بر پایه مخابرات می باشد ولی در صورتی که فاقد پورت های ارتباطی از قبیل Infrared، Bluetooth و... باشند کمتر مورد توجه قرار می گیرند.

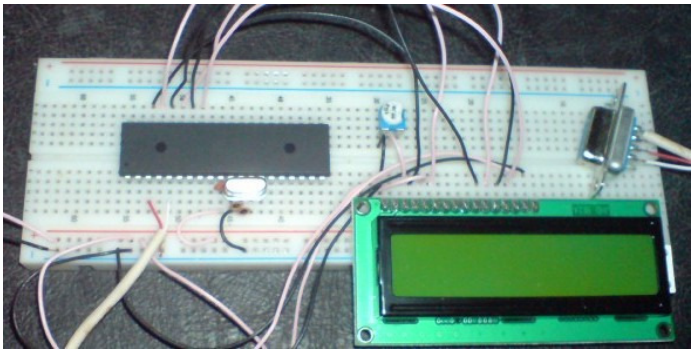
پروتکل ارتباطی که در اینجا ار آن سخن می گوییم یک پروتکل استاندارد و بین المللی می باشد که امروزه بیشتر دستگاه های الکترونیکی از قبیل GSM MODEM، Tagهای RFID، ماژول های بیسیم، مودم های کامپیوتر و... از آن برای ارتباط بین بخش های داخلی خود و سایر دستگاه ها استفاده می کنند. (پروتکل های I2C، TWI، SPI، 1Wire و (JTAG) IEEE 1149.1 نیز به نوبه خود کاربرد های زیادی دارند ولی معرفی هر کدام از آنها نیاز به مقاله ای جدا دارد!)

ما در این بخش به ساده ترین و پر کاربرد ترین روش ارتباط بین دو میکرو یا میکرو و کامپیوتر می پردازیم. در این روش از ۳ سیم برای ارتباط بین دو دستگاه استفاده می شود. یک سیم زمین بوده و دو سیم دیگر ارسال (TX) و دریافت (RX) می باشند. این پروتکل در مد آسنکرون کار می کند و نیازی به پین کلاک ندارد. فقط باید به این نکته توجه داشته باشیم که برای کار با این پورت ارتباطی حتماً باید از کریستال خارجی با فرکانس 11.0592Mhz و خازن های ۲۲ پیکوفاراد استفاده نماییم. (نوسان سازهای داخلی و سایر فرکانس های داخلی و خارجی جواب نمی دهند و درصد خطا را بالا خواهند برد).

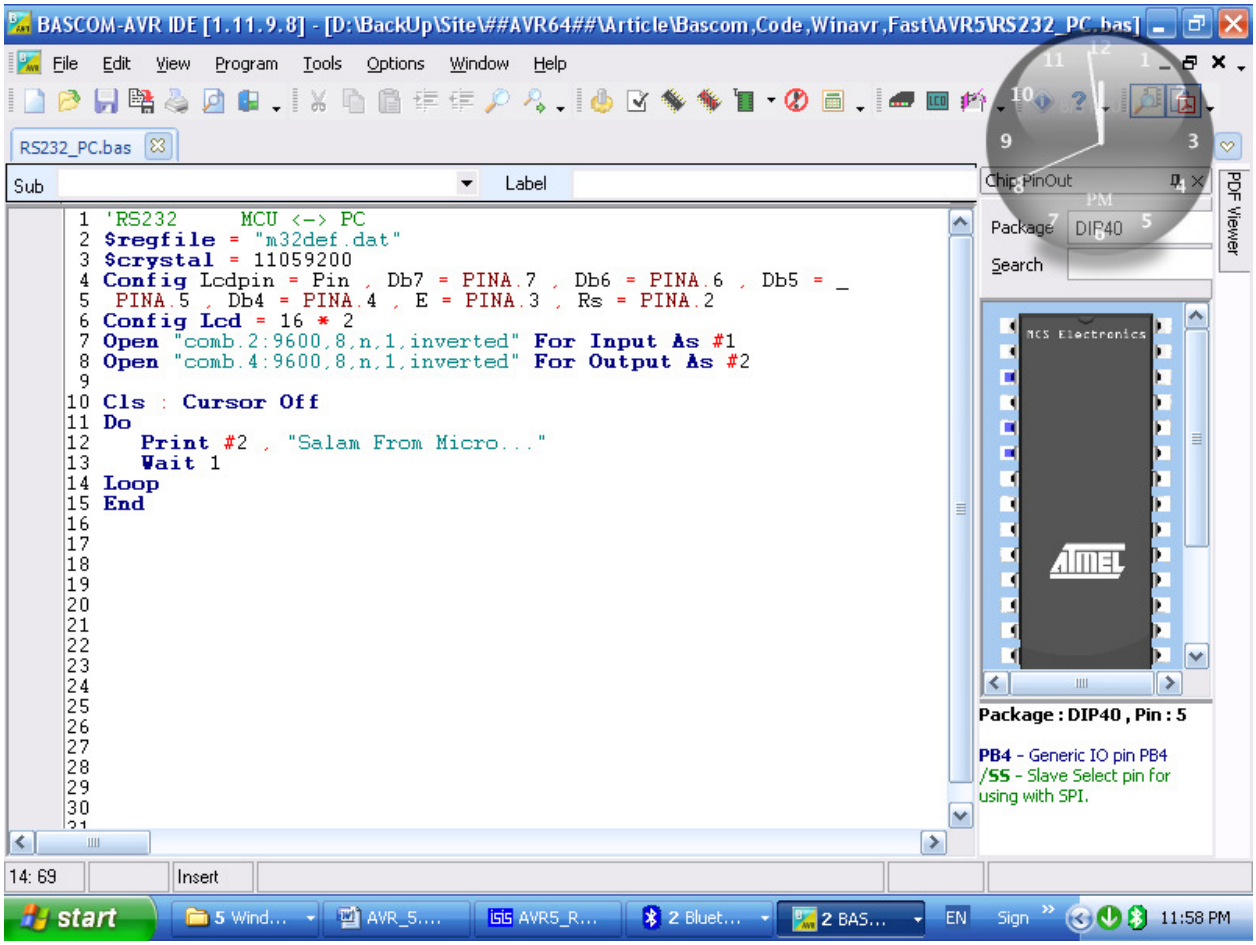
در پروتکل RS232 اطلاعات به صورت متن ASCII جابجا مـی شـوند و بـرای ارتبـاط بـین PC و میکرو به یک محیط متنی خاص نیاز داریم. می تـوانیم از Hyper Terminal ویـنـدوز اسـتفاده کنـیم یـا از ترمینال امولیتور بسکام که با فشار کلید های Ctrl+T باز می شود. در اولین پروژه قصـد داریـم بـین میکـرو و کامپیوتر ارتباط دو طرفه برقرار سازیم. برای این منظور بایستی با پایه های پورت سریال کامپیوتر و میکرو آشنا شویم. تمام کامپیوتر های دسکتاپ مجهز به یک یا دو پورت سریال هستند که با نام های COM1 و COM2 شناخته می شوند. این پورت به صورت ۹ پین و نری بوده و در PC های جدیـد در پشـت Case واقـع شـده است. از بین ۹ پایه این پورت پایه های ۲ و ۳ و ۵ برای کار ما کفایت می کنند. پایه ۲ (RX)، پایـه ۳ (TX) و پایه ۵ زمین است. برای ارتباط با این پـورت یـک کـانکتور مـادگی DB9 تهیـه کـرده و سـه سـیم کوتـاه ۸۰ سانتیمتری به این سه پایه لحیم می کنیم و پایه ۵ را به زمین میکرو وصل می نماییم. پایه های ۲ و ۳ را نیز به دو پایه دلخواه میکرو وصل می کنیم. (نظر به اینکه در میکرو پایه های سریال را به صورت نرم افـزاری بـاز مـی کنیم از هر پایه ای می توان برای ارسال و دریافت استفاده کرد). معمولاً در ارتباط بین PC و میکرو از آی سی MAX232 استفاده می کنند، با استفاده از این آی سی طول کابل را می توان تا ۱۵ متر افزایش داد ولی ما از این چیپ استفاده نمی کنیم و با دستور Inverted سطح منطقی ۰ و ۱ را جابجـا مـی کنـیم. (بـرای فواصـل کمتر از ۱ متر).

مداری مطابق شکل زیر ببندید تا اولین ارتباط فیزیکی بین میکرو و PC را آغاز کنیم. در شـکل هـای زیر شماتیک، پورت سریال PC و چیدمان قطعات بر روی برد بورد مشاهده می شوند:

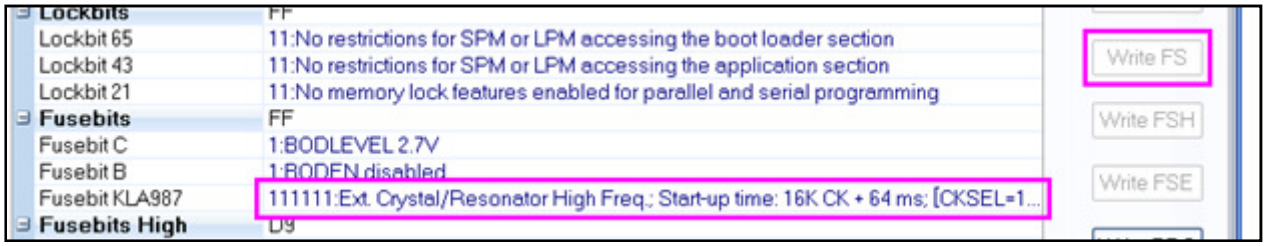




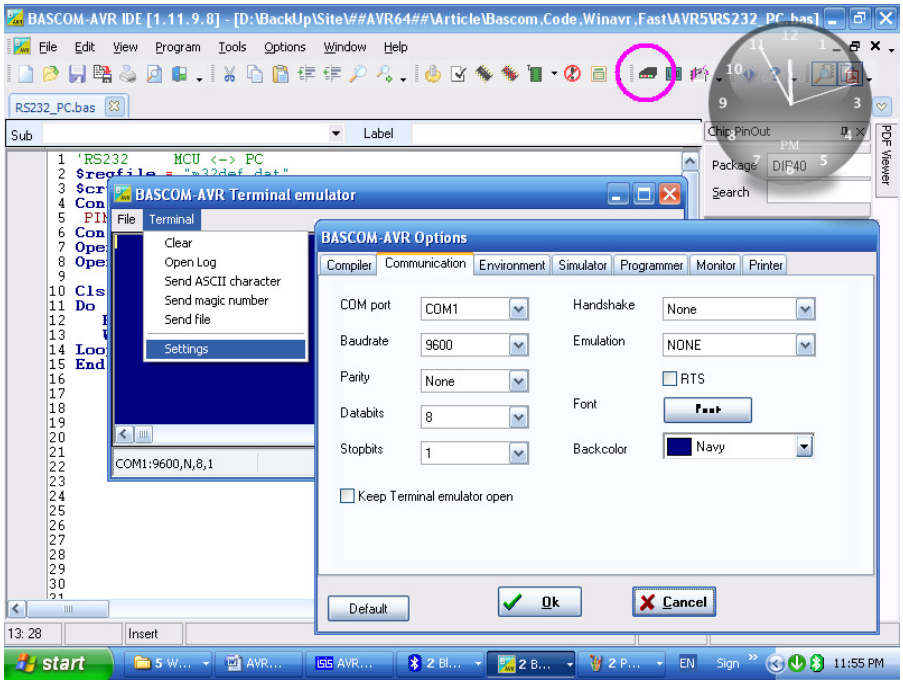
بسکام را باز کرده و کدهای زیر را در آن وارد نمایید:



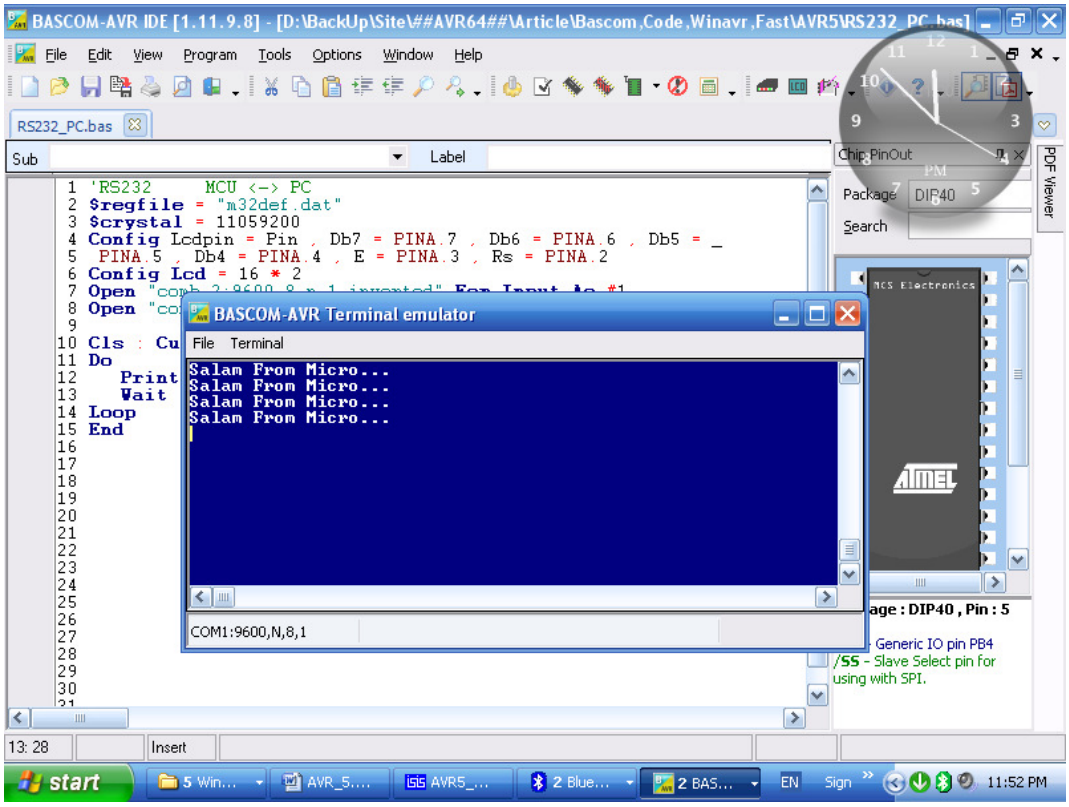
نکته فوق العاده مهم در ارتباط سریال فرکانس کاری میکرو مـی باشـد کـه حتمـاً بایسـتی 11.0592 مگاهرتز بوده و در اینجا در خط سوم فرکانس را بر حسب هرتـز بـه صـورت 11059200 نوشـته ایـم. در ایـن صورت نصب کریستال 11.0592MHz و خازن های مربوطه و نیز تنظیم فیوز بیت کلاک بر روی گزینـه آخـر 111111 لازم به نظر می رسد.



در خط ۷ و ۸ دو پایه دلخواه از میکرو که پایه های B2 و B4 هستند و به پایه های ارسال و دریافت PC رفته انـد را بـه صورت نرم افزاری باز میکنیم. (چنین قابلیتی سبب می شود تا از هر پایه ای برای هر کاری بتـوان اسـتفاده کـرد و باعـث سـادگی طراحی PCB می شود. این ویژگی یکی از امتیازات Bascom محسوب شده اما سبب تولید کدهای بیشتر می شود). توجـه کنیـد که پایه B.2 میکرو به پایه TXD (ارسال) PC رفته است پس باید ورودی تعریف شود . پایه B.4 هـم کـه بـه RXD (دریافـت) PC وصل شده بایستی خروجی تعریف شود. در جلوی هر عبارت یک عدد صحیح نوشته می شود که مشخص کننده شـماره کانـال بـاز شده است. اصولاً می توان کانال های زیادی را به صورت نرم افزاری باز کرد و برای تشخص کانال مورد نظر یک عدد خـاص بـه آن نسبت می دهند که باید بزگتر از ۰ و مثبت باشد. در داخل این دستورات Open یک رشته وجود دارد که در آن در ابتدا نـام پایـه مورد نظر آورده می شود، سپس Baud یا سرعت انتقال اطلاعات که می تواند مقادیر معتبر خاصی باشد که اسـتاندارد تـرین آنهـا 9600 بیت در ثانیه اسـت. عـدد بعدی دیتا بیت بوده که معمولاً ۸ است، عدد بعدی پریتـی یـا بیـت توازن اسـت کـه بـرای تشـخیص خطا به کار مـی رود و نوشـتن N یعنی None نمایـانگر صـرفنظر کردن از آن اسـت. عـدد بعـدی Stop Bit یا بیت توقف است کـه مقدار آن را ۱ در نظر گرفته ایـم. دسـتور بعـدی یعنـی inverted منطـق اطلاعـات را بـرعکس مـی کند و موقع عدم استفاده از چیپ Max232 باید نوشته شود. ایـن



مقادیر بایستی در دو دستگاه به طور یکسان تظیم شوند. با توجه به اینکه کانال ارسال ما کانال شماره 2 است در خط 12 با دستور Print #2 رشته نمایش داده شده را با فواصل زمانی ۲ ثانیه به طور مرتب بر روی پایه B.4 میکرو ارسال می کنیم. این پایه به پایه دریافت PC رفته است پس برای دیدن نتیجه باید به نحوی اطلاعات دریافت شده توسط کامپیوتر را شنود کنیم. برای شنود پورت سریال می توانیم از نرم افزار Hyper Terminal ویندوز و یا Terminal Emulator بسکام بهره ببریم. برای این منظور کلیدهای Ctrl + T را در محیط فعال بسکام فشار دهید و یا مانند شکل زیر روی دکمه ای که دور آن دایره کشیده شده است کلیک کنید. در پنجره باز شده از منوی Terminal روی Setting کلیک کرده و تنظیمات را مطابق زیر انجام دهید. توجه داشته باشید که اگر کامپیوتر شما دو پورت Com داشته باشد در قسمت اول باید شماره آن را درست تنظیم کنید. (ممکن است تعداد بیشتری مثلاً تا ۳۰ پورت در اینجا نمایش داده شوند که بیشتر آنها مجازی بوده و نرم افزاری هستند. مثل پروت سریال مجازی بلوتوث یا مودم) در قسمت باودریت اعداد زیادی را می بینید که می توانید بالا ترین آنها را هم در اینجا و هم در میکرو تنظیم کنید که بهترین آنها 9600 است. پس از OK کردن پنجره پورت سریال میکرو را به PC وصل کرده و میکرو را روشن کنید. اگر همه چیز به خوبی پیش رفته باشد شکل صفحه بعد را روی مانیتور خود می بینید.

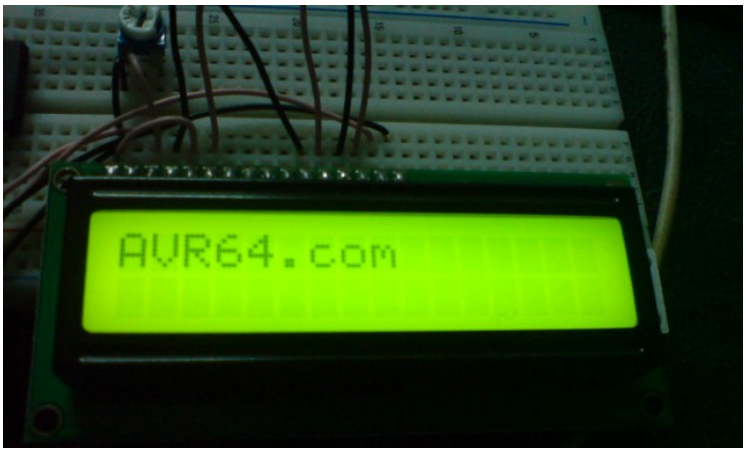


تا اینجا نحوه ارسال اطلاعات از میکرو به PC را آموختیم. در بخش بعد می آموزیم که چگونه اطلاعات را از PC به میکرو بفرستیم. پس کد های زیر را در بسکام وارد کنید و بدون دستکاری سخت افزار خروجی را در

آی سی پروگرم نموده و محیط ترمینال را باز کنید. (ابتدا ترمینال را باز کنید و بعد از آن تغذیه میکرو را متصل کنید تا پیام اولیه مشاهده شود)

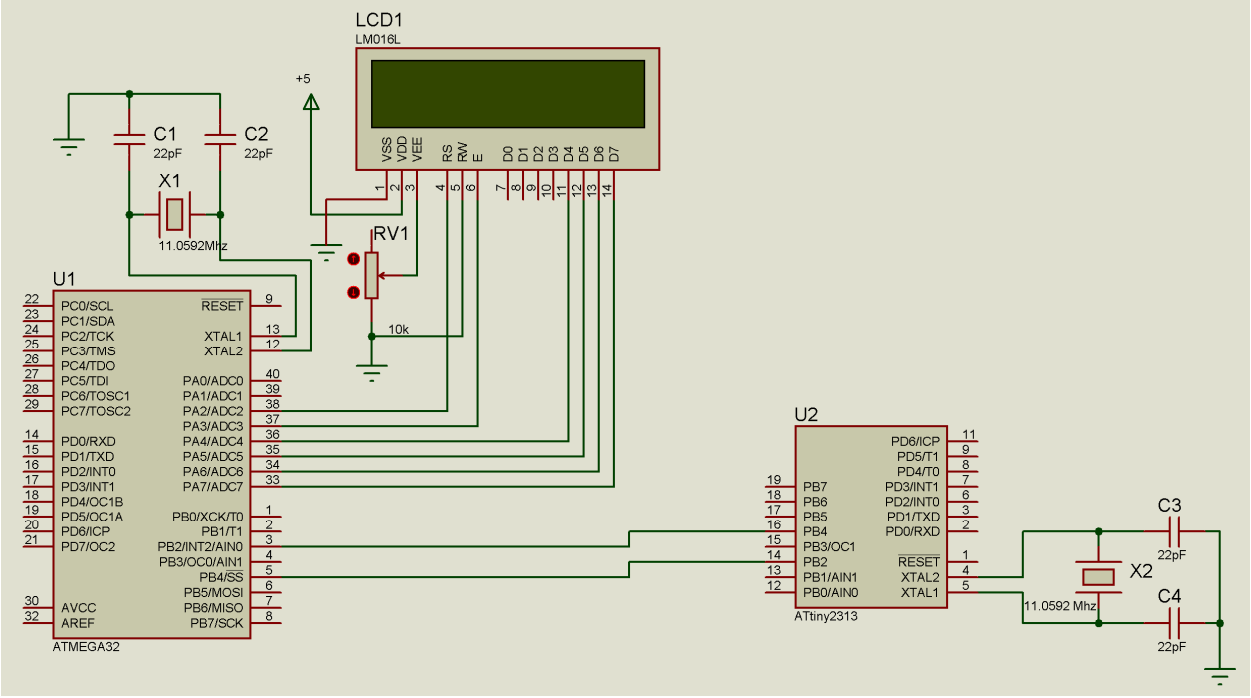
```
'RS232      MCU <-> PC
$regfile = "m32def.dat"
$crystal = 11059200
Config Lcdpin = Pin , Db7 = Pina.7 , Db6 = Pina.6 , Db5 = _
 Pina.5 , Db4 = Pina.4 , E = Pina.3 , Rs = Pina.2
Config Lcd = 16 * 2
Open "comb.2:9600,8,n,1,inverted" For Input As #1
Open "comb.4:9600,8,n,1,inverted" For Output As #2

Cls : Cursor Off
Dim A As Byte
Dim X As String * 10
Print #2 , "you?"
Input #1 , X
Print #2 , "Salam " ; X
Do
   A = Inkey(#1)
   If A <> 0 Then Lcd Chr(a)
Loop
End
```

در برنامه بالا پس از تعریف متغیرها با دستور Print نام کاربر خواسته می شود. در این حالت کلمه you? را بر روی مانیتور می بینید. نام خود را که کمتر از ۱۰ رشته است در پاسخ تایپ کنید. در حین تایپ چیزی در صفحه مشاهده نمی کنید چرا که این دستور No Echo بوده و فقط منتظر دریافت کاراکتر Enter می ماند. بعد از تایپ نام خود Enter کنید. کار دستور Input تمام شده و نام شما را در متغیر رشته ای x ذخیره می کند و بلافاصله با اجرای دستور بعدی به شما سلام می کند. در حلقه انتهایی دستور دیگری به نام inkey() رابه کار برده ایم که منتظر تایپ Enter نمی ماند و با فشار هر کلیدی از کیبرد کد اسکی آن را بر می گرداند. ما با دستور Chr() کد را به کاراکتر مربوطه تبدیل کرده و بر روی صفحه نمایش نشان می دهیم. با کمک همین سه دستور به راحتی می توانید دستگاه هایی بسازید که قابلیت ارتباط با PC را داشته باشند. البته برای ارسال فایل هم باید خلاقیت به خرج داده و با تبدیل اطلاعات به کاراکترهای اسکی و نوشتن برنامه های خاص تحت PC به تبادل اطلاعات بین میکرو و کامپیوتر بپردازید.



دستور Print پس از ارسال رشته، کاراکتر Enter را هم ارسال می کند. این کار باعث می شود که با هر بار ارسال رشته در محیط ترمینال به خط بعدی برویم. موقعی که بین دو میکرو ارتباط برقرار می کنیم صحبتی از محیط ترمینال به میان نیامده و تمام ارسال و دریافت ها فقط بین دو میکرو انجام می شود. اصولاً برای ارسال فقط یک دستور وجود دارد و آن Print است که همیشه کاراکتر شماره ۱۳ اسکی یعنی Enter را هم در انتهای رشته ارسال می کند. برای دریافت می توانیم از سه دستور Input، Inkey() و Waitkey() استفاده نماییم. دستور Input آنقدر منتظر می ماند تا کاراکتر ۱۳ را دریافت کند و سپس نتیجه را بر می گرداند. دستور Inkey() به طور مداوم ورودی را اسکن می کند و اگر کاراکتری نبود ۰ را بر می گرداند؛ پس باید در حلقه و با شرط مورد استفاده قرار گیرد. این دستور کاراکتر ۱۳ را هم نشان می دهد. دستور Waitkey() منتظر می ماند تا کاراکتری دریافت شود و شبیه Input عمل می کند با این تفاوت که کاری به دریافت Enter ندارد و با رسیدن هر کاراکتری مقدار آن را بر می گرداند. ما در پروژه زیر فقط از Print و Input بهره می گیریم. (در مورد کارکترهای انتهای خط CR و LF و روش نمایش فایلهای متنی در ویندوز و لینوکس تحقیق کنید).

مداری مطابق شکل زیر بر روی برد بورد ببندید:



در این پروژه از یک میکروی کوچک ATtiny2313 همراه با کریستال مخصوص به عنوان میکروی طرف مقابل استفده کرده ایم. در این سیستم ارتباطی بر طبق برنامه ای که برای هر دو IC نوشته شده است یک مکالمه کوتاه بین دو میکرو صورت می گیرد و نتایج بر روی LCD نشان داده می شود. در این سیستم همه چیز به صورت ایده آل فرض شده و تغذیه هر دو میکرو به یکدیگر مرتبط بوده و عملاً با هم شروع به کار می کنند. با توجه به اینکه میکروی کوچکتر آغازگر بحث می باشد بایستی مطمئن شویم که میکروی بزرگتر (M32) آماده شنیدن شده است؛ برای این منظور در میکروی کوچکتر از دستور Wait 1 قبل از آغاز سخن استفاده کرده ایم تا به مدت ۱ ثانیه صبر کند تا میکروی M32 بالا آمده و منتظر دریافت شود. همچنین قبل از تمام دستورات Print به مدت ۱ ثانیه صبر کرده ایم تا طرف مقابل وارد مد دریافت شود. مسلماً این روش ها در سیستم های ارتباطی حرفه ای کارآمد نخواهد بود. در سیستم های ارتباطی مثل فکس از روشی به نام دست تکانی استفاده می کنند؛ بدین ترتیب که فرستنده در ابتدا با ارسال پالسی به گیرنده از آمادگی آن خبردار می شود و پس از OK گرفتن از گیرنده اقدام به ارسال دیتا می کند، اگر گیرنده را در وحله اول پیدا نکرد چند بار تلاش می کند و سپس پیغام Failed! را نشان می دهد. از این روش در این نمونه پروژه ساده استفاده نشده است. برای ایجاد یک ارتباط مناسب علاوه بر اینکه پروتکل های دو طرف در سطح فیزیکی (مثل Baud و ...) باید یکسان باشند نیاز به یکسان سازی پروتکل ها در سطح بالاتر نیز احساس می شود. در صنعت پروتکل های پیچیده ارتباطی با لایه های گوناگون برای کشف خطا و... طراحی شده است؛ یکی از این پروتکل ها پروتکل TCP/IP است که یک ارتباط مطمئن بین دو دستگاه ایجاد می کند. ما نیز باید برای میکروها پروتکل های ارتباطی کوچک و کارآمدی را طراحی کنیم تا ایجاد ارتباط در شرایط مختلف را تضمین کند.

بسکام را باز کرده و برای هر میکرو کد مربوطه اش را تایپ کرده و پروگرم نمایید: (هر دو میکرو در مد کریستال خارجی هستند)

کد ATmega32:

```
'RS232      Master
$regfile = "m32def.dat"
$crystal = 11059200
Config Lcdpin = Pin , Db7 = Pina.7 , Db6 = Pina.6 , Db5 = _
 Pina.5 , Db4 = Pina.4 , E = Pina.3 , Rs = Pina.2
Config Lcd = 16 * 2
Open "comb.2:9600,8,n,1,inverted" For Input As #1
Open "comb.4:9600,8,n,1,inverted" For Output As #2
Cls : Cursor Off
Dim X As String * 10
Lcd "Receiving..."
Input #1 , X
Cls : Home : Lcd "Tiny Say: " ; X
Wait 1
Home L : Lcd "We Say: Mega32"
Print #2 , "Mega32"
Input #1 , X
```

```
Home : Lcd "Tiny: " ; X
End
```

کد ATtiny2313:

```
'RS232      Slave
$regfile = "attiny2313.dat"
$crystal = 11059200
Open "comb.2:9600,8,n,1,inverted" For Input As #1
Open "comb.4:9600,8,n,1,inverted" For Output As #2
Dim X As String * 10
Wait 1
Print #2 , "You?"
Input #1 , X
Wait 1
Print #2 , "Hi, " ; X
End
```

پس از اتصال هر دو میکرو مطابق نقشه مذور و اتصال تغذیه، مکالمه زیر بین دو میکرو صورت می گیرد که آغاز گر آن میکروی Tiny2313 است:

Tiny2313: *You?*

M32: *Mega32*

Tiny2313: *Hi, Mega32*

در این بخش آموختیم که چگونه یک شبکه بین میکروها و کامپیوتر ایجاد کنیم. انتقال اطلاعات یکی از جذاب ترین ویژگیهای پروژه های دیجیتال است. پروتکلی که در این مختصر معرفی کردیم یک پروتکل استاندارد می باشد. امروزه ماژول های بیسیم RS232 به وفور یافت می شوند. می توانید با اتصال این ماژول ها به میکرو سیستم های بیسیم قابل حمل مثل موبایل طراحی کنید که قابلیت ارسال و دریافت پیام و اطلاعات مختلف را داشته باشند. سایر پروتکل های ارتباطی از قبیل SPI و I2C پیچیدگی های خاص خود را دارند. با اندکی مطالعه در مورد نحوه انتقال اطلاعات می توانید پروتکل های ارتباطی سنکرون خاص با پایه کلاک طراحی کنید و برای ارتباط بین دو میکرو به کار ببرید یا از یک پورت برای انتقال اطلاعات به صورت موازی و ۸ بیتی استفاده کنید. در کل همیشه مجبور به استفاده از استانداردها نیستید و می توانید مطابق میل خود از

ساده ترین روش استفاده کنید. در اینجا به مبحث ارتباطات پایان مـی دهـیم و در بخـش آخـر کمـی در مـورد سابروتین ها و نحوه برنامه نویسی ساخت یافته صحبت می کنیم.

زیر برنامه ها و توابع (Subroutines & Functions):

برنامه هایی که تا اینجا نوشتیم همگی در حد ساده و چند خطی بوده و فقط یک کار مفرد را انجام می دادند. البته همیشه وضع به همین منوال نیست. در پروژه های سنگین تر معمولاً کارهای پیچیده تری صورت می پذیرد که کل کار از کارهای کوچکتری تشکیل شده است. مثلاً در یک برنامه قفل امنیتی پیشرفته کارهایی از قبیل دریافت رمز از کاربر، بررسی صحیح بودن رمز، دریافت پین کد مدیریت، بررسی صحیح بودن پین، انجام عمل خاص در هنگام درست بودن رمز، انجام عمل خاص در هنگام غلط بودن رمز (از قبیل خاموش کردن دستگاه یا به صدا در آوردن آلارم)، امکان تغییر رمز و غیره هرکدام یک زیر برنامه کوچک و مستقل هستند که کار خود را انجام می دهند و کار برنامه اصلی فقط فراخوانی این زیر برنامه هاست. به این زیر برنامه ها سابروتین گفته می شود. سابروتین ها می توانند چندین متغیر را نیز به عنوان ورودی بپذیرند. در مقابل توابع قرار دارند که معمولاً یک یا چند عدد را گرفته و یک خروجی می دهند. به طور مثال تابع Sin() یک تابع ساده است که سینوس زاویه داده شده را در خروجی می دهد.

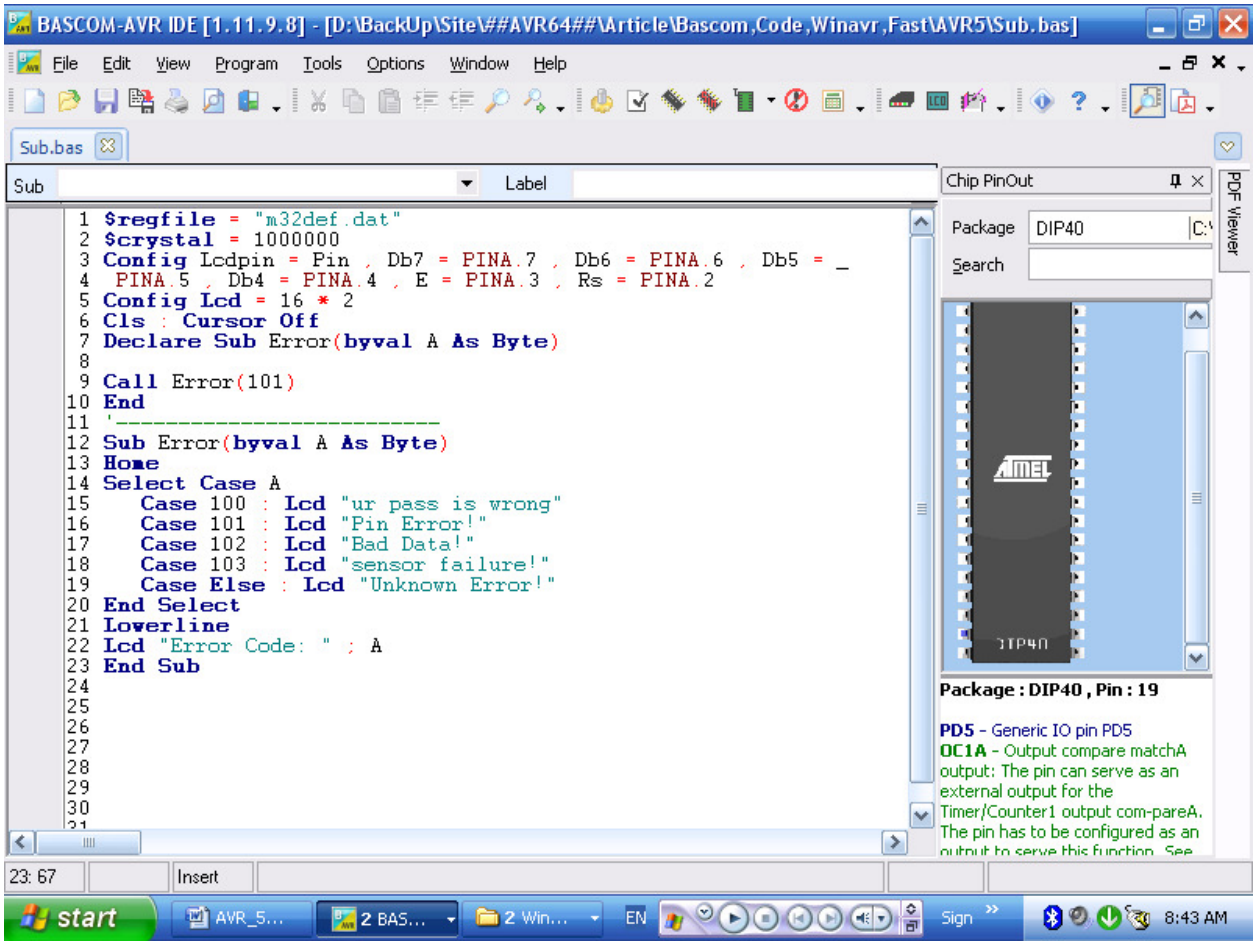
عموماً به دو دلیل از زیربرنامه ها استفاده می شود: ۱- نظم و انسجام بخشیدن به برنامه و افزایش قابلیت استفاده مجدد از قطعه برنامه های کاربردی در سایر برنامه ها (مثلاً برنامه اسکن کی پد) ۲- کم کردن حجم کد نهایی با فراخوانی برنامه های تکراری به جای دوباره نویسی آنها. دلیل اول مهم ترین دلیل برای برنامه نویسی ساخت یافته با سابروتین محسوب می شود ما همیشه از برنامه های کاربردی در سایر برنامه های استفاده می کنیم و تمام برنامه ها اکثراً ظاهری مشابه با هم دارند. دلیل دوم نیز بیشتر در برنامه هایی استفاده می شود که یک بخش تکراری بایستی بارها فراخوانی شود و به جای تایپ مجدد، یک بار آن را می نویسیم و در هر قسمت از برنامه که نیاز بود فراخوانی می کنیم. برای آغاز کار از سابروتین ها شروع می کنیم یک نمونه سابروتین با یک ورودی را مثال می زنیم. با کلید های Byref و Byval می توان یکی کپی از متغیر را فرستاد و یا آدرس آنرا ارسال کرد. اگر از byref استفاده کنیم و یا چیزی ننویسم کامپایلر به طور پیش فرض از byref استفاده می کند و آدرس متغیر را به زیربرنامه می فرستد، در این صورت سرعت و کارآیی برنامه بالا می رود و البته هر تغییری که در زیر برنامه روی ورودی اعمال شود روی متغیر اصلی هم اعمال می شود. در صورتی که از byval استفاده کنیم یکی کپی از متغیر به سابروتین ارسال می شود که مستلزم گرفتن وقت پردازنده است و

نیز تغییرات اعمال شده روی متغیر در زیربرنامه بر روی متغیر اصلی ارسال شده تاثیری نخواهد داشت. توجه: موقع استفاده از عدد در فراخوانی حتماً باید از byval استفاده کنید در غیر این صورت باید ابتدا عدد را در یک متغیر ریخته و آن متغیر را در فراخوانی به کار ببرید. قبل از شروع برنامه نویسی به عنوان آخرین نکته یادآور می شویم که در داخل سابروتین ها می توانیم به تعریف متغیر های محلی و موقتی بپردازیم برای این کار بجای Dim a as ... از Local a as ... استفاده می شود. متغیرهای محلی موقع فراخوانی سابروتین ایجاد شده و تا پایان آن باقی می مانند و فقط در محدوده همان سابروتین قابل کاربردند. از این روش زمانی استفاده می شود که فضای SRAM محدود باشد.

به هنگام استفاده از سابروتین ها و توابع حتماً سه دستور زیر را در ابتدای برنامه قرار دهید. با دستور اول اندازه فریم را بر حسب بایت تعیین می کنید. فریم همان فضایی است که وقتی یک پروسیجر را فراخوانی می کنید و متغیری را به صورت byval به آن ارسال می کنید کپی آن در آنجا ذخیره می شود. نیز موقع اعلان متغیرهای محلی Local از این فضا استفاده می شود. همچنین وقتی در مقابل LCD از تابعی استفاده می کنید ایتدا حاصل تابع محاسبه شده و در frame قرار می گیرد و سپس در LCD نمایش داده می شود. دستور دوم سایز پشته سخت افزاری میکرو را که همان Stack می باشد تعیین می کند و در نهایت دستور سوم یک پشته نرم افزاری خاص را برای نگه داشتن آدرس های سابروتین ها و توابع ایجاد می کند. Stack سخت افزاری در لایه پایین تر برای فراخوانی های سیستمی و پرش های وقفه ها در اسمبلی لازم است و فضای این پشته از انتهای SRAM رشد می کند ولی در یک سطح بالاتر برای نگه داری آدرس های برگشت از توابع و سابروتین ها هیچ تدبیری در داخل میکرو اندیشیده نشده. به همین روی بسکام بخشی از فضای SRAM را با دستور $swstack برای این منظور جدا می کند. برنامه نویسی با توابع و سابروتین ها به خاطر مساله Stack Overflow کار واقعی مشکلی است و در علوم کامپیوتر مبحث خاصی دارد. پس اگر برنامه های Bascom هنگ می کنند مشکل را به گردن کامپایلر نیندازید چرا که مدیریت پشته ها بر عهده برنامه نویس است، چیزی که برنامه نویس های غیر متخصص حتی از آن اطلاع هم ندارند چه برسد به رعایت آن!

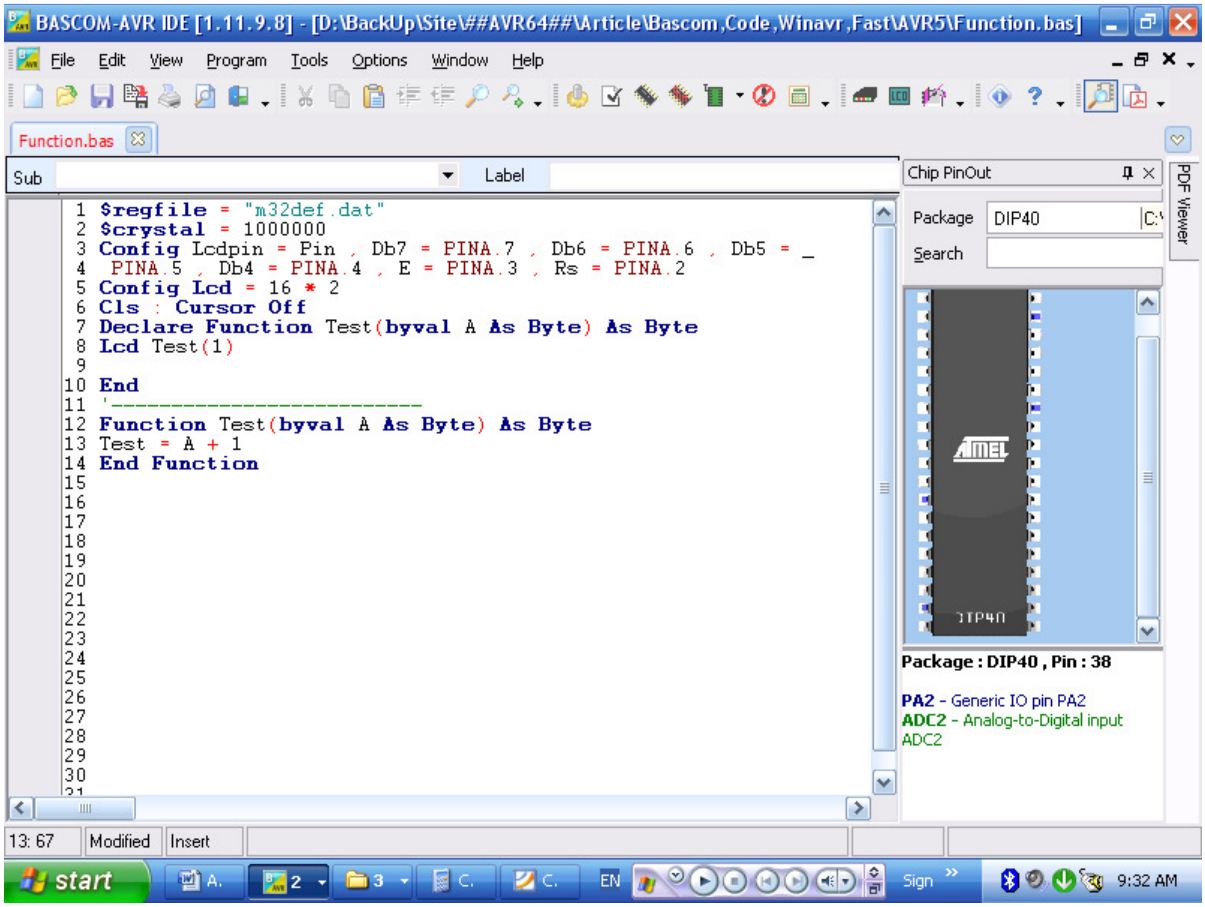
```
$framesize = 128
$hwstack = 128
$swstack = 128
```

برای شروع، نمونه کد زیر را روی سخت افزار قبل اجرا کنید: (ما در دو برنامه زیر دستورات تنظیم پشته و فریم را در بخش Options مقدار دهی کرده و در متن برنامه ننوشته ایم).



در این پروژه در خط ۷ نام سابروتین که Error می باشد به همراه یک متغیر از نوع بایت تعریف شده است. در خط ۹ سابروتین را با مقدار ۱۰۱ فراخوانی می کنیم. برنامه سابروتین پس از دستور End و در خطوط ۱۲ تا ۲۳ قرار گرفته است. این برنامه با توجه به کد خطای اتفاق افتاده در برنامه پیغام مناسب و نیز شماره کد را نشان می هد. توجه داشته باشید که سابروتین با کلمات کلیدی Sub و End Sub محصور می شود و حتماً باید پس از دستور End نوشته شود. استفاده از چنین سابروتینی هم برنامه را گویا تر می کند، هم نحوه تغییر کد نویسی آن را برای اشخاص نا آشنا را راحت تر می کند و هم باعث صرفه جویی در نوشتن چندین کد می شود. پس از اجرای سابروتین اشاره گر برنامه دستورات پس از خط ۹ را اجرا می کند که در اینجا دستور End است.

توابع نیز تقریباً شبیه به سابروتین تعریف می شوند ولی یک عدد را بر می گردانند. کد زیر را در بسکام وارد نموده و روی سخت افزار پروژه قبل Run کنید:



تنها تفاوت آن وجود یک تعریف متغیر بعد از پرانتز است که نوع داده برگشتی را مشخص می کند. اسم تابع همان متغیر برگشتی است که در داخل بدنه این تابع به عدد دریافت شده یک واحد افزوده شده و برگشت داده می شود. دستور برگشت مقدار به نام تابع باید آخرین دستور تابع باشد و بعد از آن هیچ کد دیگری غیر از End Function نوشته نشود.

در این جلسه با دستورات مهم بسکام آشنا شدیم. با ترکیب این دستورات و نیز مطالعه کامل Help بسکام در زمینه دستورات ریاضیاتی و کار با رشته ها تقریباً می توانیم هر پروژه ای را طراحی کنیم. بسکام برای ساپورت تجهیزات جانبی مثل GSM MODEM، RFID TAG، TCP/IP و سایر چیپ ها و LCD های رنگی و... نیاز به کتابخانه های مجزا دارد. هر چند می توان با ترکیب دستورات اسمبلی و بیسیک تا حدی با اشیاء مذکور ارتباط برقرار کرد ولی Base کار از نظر پنهان بوده و ایجاد کدهای فراوان و Hex های طولانی

همیشه برای کاربر به صورت پنهانی ونامفهوم باقی خواهد ماند و او را در هاله ای از ابهام قرار می دهد. ممکن است در بسکام کتابخانه ای برای راه اندازی LCD مارک Philips قرار داده شده باشد ولی با تهیه یک LCD گوشی NOKIA حتی در وصل کردن آن به میکرو و پیدا کردن پایه های آن دچار ابهام خواهید شد. همچنین در مورد کار کردن با تگ های RFID که امروزه با مارک های مختلف به چشم می خورد. تمام این قضایا از مشکلات زبان های سطح بالاست که در ابتدا با ظاهری دلفریب وارد می شوند و موقع کارهای حرفه ای ضعف ها و ناتوانی های خود را نشان می دهند. هیچکدام از این مشکلات در سطح صفر سیستم وجود ندارند. جایی که فقط ۰ و ۱ ها را می شناسیم. در سطح صفر هیچ خطایی را نمی توان به اسمبلر نسبت داد چرا که وظیفه آن از همان ابتدا معلوم است و با هر دستور رمزی، فقط یک عمل ساده را در میکرو به انجام می رساند، هیچ System Failure و Errorی وجود ندارد، وظیفه برنامه نویس است که به دور از هرگونه مشغله ذهنی با دقت هرچه تمام تر توابع را روی کاغذ پیاده سازی کرده و پس از تایپ و اسمبل کردن آنها جواب ۱۰۰٪ درست خود را بگیرد. زبان اسمبلی ورود به دنیای واقعی تلفیق الکترونیک و کامپیوتر است، مباحث اسمبلی سخت تر، طاقت فرساتر و مسلماً طولانی تر از سایر زبان ها هستند، تنها افراد کمی از جلسه بعد همراه ما باقی می مانند و آنها فقط حرفه ای ها و علاقه مندان واقعی به دنیای دیجیتال هستند.

آخرین جلسه بسکام #

ادامه دارد ... (شروع اسمبلی از جلسه بعد)

(برای تولید سورس های بسکام / بیسکام از نسخه دموی Bascom 1.11.9.8 استفاده شده است)

(برای رسم شماتیک ها از نسخه دموی Proteus 6.0 Lite استفاده شده است)

مولف: بهنام زکی زاده
www.avr64.com
۵ شهریور ۱۳۸۹

آخرین ویرایش در تاریخ ۱۳ مهر ۱۳۹۲ ✓