

فرهنگی روشن در انتظار توست

اصول

کار با میکرو کنترلر های AVR

و

کامپایلر بسکام

ترجمه و تالیف:

1nafar

www.iranmicro.ir

فهرست : ----- شماره صفحه

مقدمه: ----- 7

فصل اول (آشنایی مختصر با محیط بسکام)

آشنایی مختصر با محیط بسکام: ----- 10

معرفی منوها ----- 10

مفاهیم اولیه ----- 14

فصل دوم (اولین برنامه شما ، کار با lcd و پورت ها)

مراحل نوشتن یک برنامه جدید (بدنه ی یک برنامه): ----- 20

دستورات مربوط به پورت ها (کار با پورتهای) ----- 25

دستورات تاخیر ----- 28

دستورات حلقه و پرش ----- 30

Lcd کاراکتری (دستورات مربوط به راه اندازی ، فارسی نویسی و...) ----- 35

پیکربندی منبع کلاک در سری xmega ----- 44

فصل سوم (معرفی سایر دستورات بیسیک)

اعداد و متغیر ها در بسکام ----- 49

دستورات مربوط به کار با رشته ها ----- 60

دستورات حلقه و پرش و شرط ----- 68

دستورات اجرایی ----- 79

زیر برنامه ها و فراخوانی توابع ----- 86

دستورات ریاضی و محاسباتی و تبدیل متغیر های ریاضی ----- 90

توابع تبدیل کدها و متغیر ها به یکدیگر ----- 100

فصل چهارم (راه اندازی امکانات جانبی)

105-----دستور debounce(اتصال کلید به میکرو)

106-----دستور PULSEOUT

107-----دستور PULSEIN

108-----دستور SOUND

108-----دستور ENCODER

110-----دستور DTMFOUT

استفاده از کلید و کیبرد و کی پد و ... در محیط بسکام

112-----استفاده از کلید

114-----اتصال کی پد به AVR

120-----اتصال کیبرد کامپیوتر به AVR

123-----راه اندازی وقفه های داخلی و خارجی

نمایشگر های کریستال مایع و LED

129-----LCD گرافیکی

142-----اتصال lcd گرافیکی رنگی به AVR

147-----نمایشگر های هفت قسمتی

آنالوگ و دیجیتال

158-----مبدل آنالوگ به دیجیتال(ADC)

175-----راه اندازی واحد DAC در سری ATXMEGA

178-----مقایسه کننده آنالوگ

تایمر ها و کانترها

- 188-----راه اندازی تایمر صفر در محیط بسکام
- 191-----راه اندازی تایمر-کانتر یک در محیط بسکام
- 207-----راه اندازی تایمر-کانتر دو در محیط بسکام
- 217-----راه اندازی تایمر/کانتر سه در محیط بسکام
- 233-----تایمر-کانتر ها در سری ATXMEGA
- 238-----RTC (Real Time Counter) (شمارش گر زمان واقعی)

پروتکل های ارتباطی

- 246-----ارتباط سریال RS232
- 268-----پروتکل RS485
- 271-----ارتباط سریال SPI
- 281-----ارتباط سریال i2c یا 2-wire
- 288-----ارتباط سریال 1 WIRE

راه اندازی ماژول و سخت افزار های جانبی مختلف

- 291-----شرحی بر Rfid (Radio Frequency Identification)
- 313-----پروتکل TCP/IP
- 328-----پروتکل x10
- 332-----کار با magnetic card (کارت های مغناطیسی)
- 335-----اتصال avr به عنوان کیبرد به کامپیوتر
- 339-----اتصال avr به عنوان موس به کامپیوتر
- 345-----اندازه گیری یک خازن یا مقاومت
- 347-----راه اندازی فرستنده /گیرنده RC5

- 350----- ساخت کنترلر تلویزیون و سیدی SONY-----
- 355----- راه اندازی انواع موتور های dc و پله ای-----
- 367----- کار با حافظه داخلی میکرو (eeprom)-----
- 370----- اتصال حافظه ی خارجی به میکرو کنترلر-----
- 371----- راه اندازی WATCHDOG:-----
- 374----- بهینه سازی مصرف توان-----

ضمائم

- 378----- ضمیمه 1: طریقه ی نصب بسکام-----
- 379----- ضمیمه 2: آشنایی با محیط شبیه سازی بسکام (simulate)-----
- 385----- ضمیمه 3: شبیه سازی میکرو کنترلر AVR با نرم افزار پروتوس (آشنایی مقدماتی)-----
- 391----- ضمیمه 4: برنامه ریزی میکرو کنترلر و معرفی پروگرامر ها-----
- 403----- ضمیمه 5: طراحی مدار با میکرو کنترلر های AVR-----
- 411----- ضمیمه ی شماره 6: دیتاشیت فارسی میکرو کنترلر های AVR-----
- 456----- منابع و مآخذ-----

مقدمه

میکرو کنترلر های AVR با قیمت پایین و قدرت فوق العاده ی خود ، جایگاه مناسبی در پروژه های میکرو کنترلی بدست آورده اند .

مصرف توان پایین و قدرت بالا در کنار قیمت بسیار ارزان و ابعاد کوچک باعث شده است که از این میکرو کنترلر ها در انواع سیستم های کنترلی ، مانیتورینگ و انتقال داده استفاده شود . از سوی دیگر وجود بلوک های مختلف نظیر ADC ، PWM ، تایمر/کانتر ، SPI ، I2C و.... باعث شده است تا اتصال سایر سنسور ها و دستگاه های جانبی به میکرو کنترلر به سادگی میسر شود . تنوع میکرو کنترلر های avr نیز یکی دیگر از محاسن آن می باشد این تنوع به گونه ای است که شما میتوانید با توجه به نیاز خود بهترین مورد را انتخاب کنید ، خوشبختانه تمامی میکرو کنترلر های AVR در ایران موجود بوده و شما می توانید به سادگی و با کمترین هزینه آن را تهیه نمایید .

برای برنامه نویسی این میکرو کنترلر های کامپایلر های متعددی ارائه شده است که یکی از ساده ترین و در حین حال قدرتمند ترین آنها کامپایلر بسکام میباشد ، در این کامپایلر شما میتوانید به زبان بیسیک برای این خانواده برنامه بنویسید ، وجود کتابخانه های مختلف و پشتیبانی از سری xmega از ویژگی های مطرح این کامپایلر است .

در ادامه با میکرو کنترلر های avr و کامپایلر بسکام بیشتر آشنا خواهیم شد ، در این کتاب به بررسی کلیه موارد مورد نیاز برای کار با این خانواده پرداخته شده است .

امید است این مطالب گامی هر چند کوچک در افزایش دانش شما و سربلندی میهن عزیزمان ایران بردارد .

با تشکر 1nafar

www.1nafar.com

فصل اول

فصل اول : آشنایی با کامپایلر بسکام و میکرو کنترلر های AVR

میکرو کنترلر یک کامپیوتر کوچک است که میتواند برنامه ی موجود در حافظه ی خود را خط به خط اجرا کند . برنامه ها در فرمتی به نام هگز یا باینری در حافظه ی میکرو کنترلر ذخیره میشوند ، در زیر بخشی از یک برنامه در فرمت هگز آورده شده است :

:100000000AC01895189518951895189518956B

:100010001895189518958FED8DBFC8EBE0EA4E2E18

:10002000DD275D2EEEE7F0E0A0E6B0E088278D93B7

:100030003197E9F789E189B988E18AB96624A0E6B0

:10004000EEEE7F2E014D1A8E8B0E0E0E6F0E002D19B

:1000500022D009D010D0A8E8B0E0E0E6F0E0FAD075

در برنامه ی بالا کد 189518951895189518951895189518956 سرعت کاری میکرو کنترلر را مشخص میکند و کد

0E0A0E6B0E088278D93B با کد 8FED8DBFC8EBE0EA4E2E18 پورت A میکرو کنترلر را به عنوان خروجی معرفی میکند .

پورت A روشن شده و کد A8E8B0E0E0E6F0E0FAD07 باعث ایجاد شدن یک تاخیر یک ثانیه بعد از روشن شدن پورت

میشود . با کد A8E8B0E0E0E6F0E002D19B پورت A خاموش میشود . اینها کدهای نامفهومی هستند که ما هیچ گاه نیازی

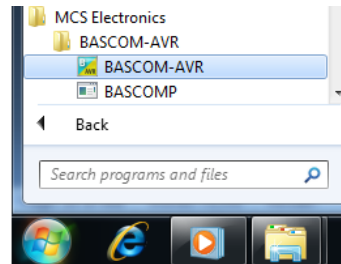
به یاد گیری معنی آنها نخواهیم داشت .

فرض کنید می‌خواهید با یک چینی زبان صحبت کنید، برای این کار به یک مترجم نیاز خواهید داشت، مترجمی که بتواند گفته‌های شما را به فرد چینی زبان انتقال داده و گفته‌های فرد چینی را برای شما ترجمه کند. در دنیای میکرو کنترلرها، کامپایلر دقیقاً کار مترجم را انجام می‌دهد. این نرم افزار کدهای وارد شده در ادیتورش (که توسط شما تایپ شده اند) را به زبان ماشین یا همان کد هگز ترجمه میکند. با توجه به تعدد زبان‌های برنامه نویسی قطعاً برای یک میکرو کنترلر کامپایلرهای مختلفی تولید میشود که شاید بهترین آنها کامپایلر بسکام باشد.

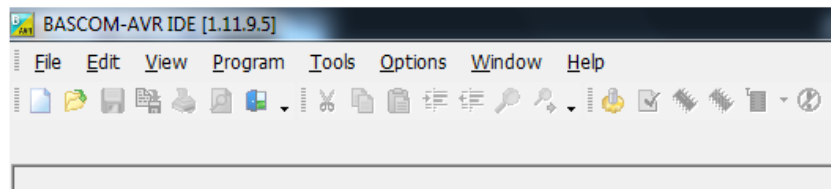
کار با نرم افزار بسکام بسیار ساده است، کافی است کد خود را در ادیتور نرم افزار تایپ کنید و از منوی Program گزینه ی Compile را انتخاب نمایید. در این فصل قصد نداریم تا با توضیحات اضافه شما را خسته کنیم، به همین دلیل فقط موارد اصلی را بررسی نموده و موارد دیگر را به ضmannم انتقال داده ایم.

آشنایی مختصر با محیط بسکام

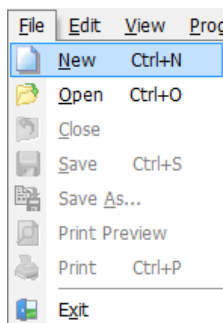
از منوی START و مسیر MCS Electronics گزینه ی BASCOM-AVR را انتخاب نمایید :



به بسکام خوش آمدید ، در این نرم افزار تمامی موارد مورد نیاز شما در منوها جمع آوری شده است :



منوی File :



1- New : با زدن این گزینه یک صفحه جدید برای نوشتن برنامه جدید باز میشود . این صفحه مجهز به ویرایشگر دستورات می باشد ، یعنی در صورتی که دستوری درست وارد شود به رنگ آبی در میاید ولی اگر دستور اشتباه باشد به رنگ معمولی (مشکی) است (این مورد برای تعداد کمی از دستورات اجرا نمیشود) .

2- Open : با زدن این گزینه می توانید برنامه ای را که از قبل ذخیره کرده اید باز کنید (برنامه ها با پسوند .BAS ذخیره میشوند ، شما همچنین میتوانید برنامه خود را در داخل (برنامه های) ویرایشگر های دیگر مانند Notepad بنویسید و از این پنجره آن را باز کنید.

3- Close : با انتخاب این گزینه پنجره ای که برای نوشتن برنامه باز شده ؛ بسته میشود. در صورتی که برنامه ی نوشته شده قبلا ذخیره نشده باشد، از شما در مورد ذخیره کردن برنامه سوال میشود.

4- Save و Save as : این دو گزینه برای ذخیره کردن پروژه به کار میروند.

5- Print و Print Preview : این دو گزینه برای چاپ کردن برنامه استفاده میشوند با زدن گزینه Print Preview می توانید نسخه قابل چاپ را قبل از چاپ مشاهده کنید.

6- Exit : با زدن این گزینه برنامه بسکام به طور کامل بسته می شود ؛ اما اگر برنامه شما ذخیره نشده باشد ؛ در مورد ذخیره برنامه از شما پرسیده میشود.












7- زیر گزینه Exit : چند گزینه دیگر وجود دارد ، این گزینه ها برای دسترسی سریع به آخرین فایل های که باز بوده اند می باشد.

منوی Edit :

1- Undo و Redo : این دو گزینه برای دست یابی به آخرین تغییرات انجام شده می باشد.

2- Copy و Cut و Paste : این سه گزینه برای برداشتن یا کپی کردن قسمتی از متن به جای دیگر میباشد.

3- Find و Find next : این دو گزینه برای پیدا کردن قسمتی از متن در برنامه می باشد. نحوه کار به این صورت است که بعد

	Undo	Ctrl+Z
	Redo	Shift+Ctrl+Z
	Cut	Ctrl+X
	Copy	Ctrl+C
	Paste	Ctrl+V
	Find	Ctrl+F
	Find next	F3
	Replace	Ctrl+R
	Goto	Ctrl+G
	Toggle Bookmark	
	Goto Bookmark	
	Indent Block	Shift+Ctrl+I
	Unindent Block	Shift+Ctrl+U
	Remark/Unremark Block	Ctrl+M
	Insert ASCII	

از انتخاب گزینه Find ، پنجره جدیدی باز می شود که باید در قسمت Text to find مورد نظر را تایپ کنید. بعد روی Ok کلیک کنید تا متن مورد نظر در برنامه انتخاب شود .

Findnext متن های که در خط های بعدی برنامه وجود دارد پیدا می کند .

4- Replace : با این گزینه شما می توانید متنی را جایگزین متن موجود در برنامه

نمایید ، یعنی در قسمت Text To Find متن یا کلمه مورد جستجو که باید توسط متن یا کلمه دیگری جایگزین شود را تایپ کنید و در قسمت Replace Wath متنی را که باید جایگزین شود تایپ می کنیم .

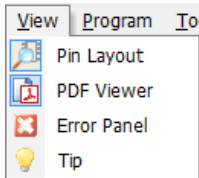
5- دو گزینه بعدی برای گذاشتن علامت در خطوط مختلف و پرش به آنها می باشد .

6- indentblock و unindentblock : این دو گزینه متن انتخاب شده را به اندازه یک Tab به چپ یا راست منتقل میکند .

7- Remark/Unremark Block : با این گزینه میتوانید بخشی از برنامه نوشته شده را علامت گذاری کنید .

8 – insert ascii: با انتخاب این گزینه پنجره ای باز میشود که در آن کد اسکی کارکترهای مختلف وجود دارد، با کلیک کردن روی هر مورد، کد مذکور به برنامه اضافه میشود.

منوی View:



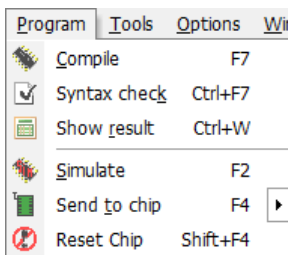
1 – Pin Layout: با انتخاب این گزینه پنجره ای باز میشود، که در آن پایه های میکرو مورد استفاده و شکل آن آورده شده است، با بردن موس روی هر پایه نقش آن در زیر تصویر نوشته میشود.

2 – Pdf Veiwew: با انتخاب این گزینه شما میتوانید مشخصات میکرو مورد استفاده را ببینید.

3 – Error Panel: با انتخاب این گزینه پنجره ای خطاها باز میشود، این پنجره بعد از کامپایل کردن برنامه، در صورت وجود خطا خودکار باز میشود.

4 – Tip: با انتخاب این گزینه پنجره Tip باز میشود، در این پنجره نکاتی در مورد برنامه نویسی و ... وجود دارد.

منوی Program:



1 – Compile: با انتخاب این گزینه برنامه نوشته شده به زبان ماشین ترجمه میشود و فایل های از قبیل هگز و گزارش و... ساخته میشود. اگر در این مرحله برنامه دارای خطا باشد پنجره ای باز میشود که در آن خطاها نمایش داده می شوند؛ با کلیک کردن روی هر خطا، خط مربوط که دارای خطا است قرمز میشود.

2 – Syntax check: با انتخاب این گزینه برنامه از نظر غلط املائی چک میشود (با زدن گزینه Compile دیگر نیازی به زدن این گزینه نمی باشد).

3 – Show result: با انتخاب این گزینه پنجره های باز میشود که در آن گزارش کلی از برنامه وجود دارد.

4 – Simulate: با انتخاب این گزینه پنجره شبیه سازی باز میشود و شما در این پنجره که دارای LCD و کیبرد و مبدل آنالوگ به دیجیتال و... میباشد می توانید برنامه خود را شبیه سازی کنید (برای دریافت اطلاعات بیشتر به ضmann مراجعه کنید).

5 – Send To Chip: با انتخاب این گزینه وارد محیط پروگرام کردن میکرو می شوید که در قسمت ضmann مفصلاً توضیح داده شده است.

Tools و Options :

در این دو منو ابزاری برای کار با برنامه و سایر قطعات جانبی وجود دارد ، با زیر منو های موجود در این منوها در بخش های بعدی آشنا خواهیم شد .

برای هر کدام از گزینه ها کلید های میانبری وجود دارد که در کنار گزینه نوشته شده است .

سوالاتی که ممکن است برای شما ایجاد شده باشد :

1- میکرو کنترلر چیست ؟

✓ به آی سی هایی که قابل برنامه ریزی می باشد و عملکرد آنها توسط برنامه ای که از قبل نوشته ایم تعیین شود ، میکرو کنترلر گویند میکرو کنترلر ها دارای ورودی / خروجی و قدرت پردازش و امکانات جانبی میباشد .

2- میکرو کنترلر چه بخش های دارد ؟

✓ میکرو کنترلر ها از بخشهای زیر تشکیل شده اند:

✓ CPU واحد پردازش : این بخش همچون مغز انسان هست و وظیفه ی کنترل کردن تمامی بخش های میکرو را به عهده دارد ، CPU بر اساس برنامه ای که به آن داده شده کار میکند .

✓ ALU واحد محاسبات : این بخش واحد پردازش و محاسبات میکرو کنترلر است . در این بخش عملیات جمع و تفریق که پایه ی کلیه عملیات ریاضی است ، انجام میشود .

✓ I/O (ورودی ها و خروجی ها) : ورودی ها و خروجی برای ارتباط میکرو با دنیای خارج ایجاد شده اند . یکی از موارد استفاده ی میکرو کنترلرها کنترل کردن یک وضعیت است ، مثلاً شما میتوانید با یک میکرو کنترلر سیستمی را طراحی کنید که به محض ورود شما به ساختمان چراغ های خانه را روشن کرده و دمای شومینه را روی 40 درجه تنظیم کند . برای این سیستم باید دو سنسور را به ورودی میکرو کنترلر متصل کنید ، یکی از سنسور ها برای تشخیص باز شدن درب و دیگری برای تنظیم دمای شومینه به کار میرود ، همچنین شما باید دو عدد رله را به میکرو متصل کنید ، یکی از رله ها ، هنگامی که سنسور تشخیص درب تحریک شد باید وصل شود ، رله ی دیگر نیز گاز مشعل شومینه را هنگامی که سنسور دما تحریک شد قطع میکند ، هر دو رله از میکرو کنترلر فرمان میگیرند و با توجه به برنامه ای که شما برای میکرو نوشته اید ، کار میکنند .

✓ حافظه ها : میکرو کنترلر های دارای حافظه های برای ذخیره دائم و موقت برنامه میباشند ، هنگامی که شما میکرو را برنامه ریزی میکنید ، برنامه در حافظه ای به نام flash ذخیره میشود ، CPU برنامه را خط به خط از این حافظه میخواند و اجرا میکند .

✓ همچنین میکرو به حافظه ای به نام RAM برای ذخیره کردن موقت اطلاعات نیاز دارد .

✓ برای راحتی کاربران ، در داخل این میکرو کنترلر حافظه ای به نام EEPROM در نظر گرفته شده است ، شما

میتوانید اطلاعات تحلیل شده توسط میکرو را در این حافظه ذخیره کنید (اطلاعات موجود در این حافظه با قطع

برق از بین نمیرود) .

✓ امکانات جانبی : در میکرو کنترلر ها امکاناتی برای متصل کردن میکرو به دیگر قطعات و لوازم الکترونیک (واحد

های SPI و USART و I2C و ...) تعدادی تایمر (زمان سنج) و کانتر (شمارنده) ، مبدل های آنالوگ به دیجیتال (

برای ارتباط میکرو با جهان واقعی) و ... در نظر گرفته شده است . در ادامه با این موارد آشنا میشویم .

3- در ایران چه میکرو کنترلر های وجود دارد و بهترین آنها کدام است ؟

✓ میکرو کنترلر های زیادی در بازار وجود دارد ، هر کدام از این میکرو کنترلر ها دارای نام و مشخصات خاص خود

هستند ، اما چهار دسته ی زیر بیشتر استفاده میشوند :

✓ AVR ، 8051 ، PIC ، ARM و

✓ واژه های بالا نام خوانده ی میکرو کنترلر ها میباشد ، مثلاً رضا از گروه انسان ها و از خانواده ی پستان داران

است ، ATMEGA16 نیز از گروه (سری) ATMEGA و از خانواده ی AVR است . در ادامه با اعضای این گروه بیشتر

آشنا میشویم .

✓ تمامی این میکرو کنترلر ها ویژگی خاص خود را دارند و این نیاز شما میباشد که بهترین را تعیین میکند .

4- با میکرو کنترلر چه کارهایی می توان انجام داد ؟

✓ این قطعات همانند یک کامپیوتر کوچک هستند و میتوانند هر چیزی را کنترل و مدیریت کنند ، بدون شک آنها

پاسخ گوی نیاز شما در تمامی زمینه ها خواهند بود .

5- چگونه میتوانیم یک میکرو کنترلر را برنامه ریزی کنیم ؟

✓ برای برنامه ریزی میکرو کنترلر از یک کامپایلر و چند نرم افزار دیگر استفاده میشود .

✓ کامپایلر نرم افزاری است که کدهای نوشته شده در محیطش را به زبان ماشین ترجمه میکند (چون کار با زبان ماشین بسیار سخت است ، کامپایلر ها بوجود آمدند). برای میکرو کنترلر ها کامپایلر های مختلفی وجود دارد که شما میتوانید در آنها به زبان C و بیسیک و اسمبلی و پاسکال و... برنامه بنویسید .

✓ بعد از نوشتن برنامه و کامپایل کردن آن (تبدیل شدن کدهای نوشته شده توسط کاربر به زبان ماشین را کامپایل میگویند) ، کامپایلر در مکان ذخیره ی برنامه ی یک فایل با پسوند HEX. ایجاد میکند ، شما باید این کد را با دستگاهی به نام پروگرامر به میکرو کنترلر منتقل کنید .

✓ همچنین شما میتوانید با استفاده از نرم افزار های شبیه ساز ، برنامه خود را شبیه سازی کنید .

6- فرق میکرو کنترلر ها چیست ؟ و هر کدام چه امکاناتی دارند ؟

✓ امکانات میکرو کنترلرها یکسان نیست و هر کدام امکانات خاصی را دارا می باشند و در قیمت های مختلف عرضه می شود .

✓ شما میتوانید با مراجعه به دیتاشیت میکرو کنترلر ها از امکانات هر کدام اطلاع یابید (دیتاشیت ، یک فایل pdf است که از طرف شرکت سازنده ی میکرو کنترلر ارائه میشود و در آن مشخصات میکرو کنترلر وجود دارد).

7- مزایای میکرو کنترلر نسبت به مدار های منطقی ، رله های هوشمند ، PLC و... چیست ؟

✓ یک میکرو کنترلر را می توان طوری برنامه ریزی کرد که کار چندین گیت منطقی را انجام دهد. تعداد آی سی هایی که در مدار به کار میرود به حداقل میرسد . به راحتی می توان برنامه میکرو کنترلر را تغییر داد و تا هزاران بار میتوان روی میکرو برنامه های جدید نوشت و یا پاک کرد .

✓ به راحتی میتوان از روی یک مدار منطقی کپی کرد و مشابه آن را ساخت ولی در صورتی که از میکرو کنترلر استفاده شود و برنامه میکرو را قفل کرد به هیچ عنوان نمی توان از آن کپی گرفت .

✓ میکرو کنترلرها بسیار ارزان هستند و برای آنها مثال ها و سورس های زیادی وجود دارد ، قطعات جانبی آنها در بازار به سادگی یافت میشود و با کمترین هزینه میتوان مدار طراحی شده با آنها را تغییر داد .

8- مفهوم پورت (PORT) و پین (PIN) چیست ؟

✓ میکرو کنترلر ها دارای چندین پایه میباشند که تعداد دقیق آنها به نوع میکرو بستگی دارد ، در این بین پایه های i/o

(ورودی / خروجی ، inpou/output) به دسته های 8 عددی

تقسیم بندی شده اند ، به هر کدام از این دسته پورت گفته

میشود . پس می توان گفت بین همان پایه های میکرو کنترلر

و پورت مجموع 8 پین است .

✓ معمولا پورت ها را با نام های A یا B یا C یا ... و پین ها را

با نام های 0 تا 7 یا VCC یا گراند یا ... نام گذاری میکنند .

9- کریستال چیست ؟

✓ کریستال قطعه ای است که به میکرو متصل میشود و میکرو توسط پالس های خروجی آن زمان خود را تنظیم

میکند ، در واقع میتوان گفت کریستال ساعت دیواری میکرو کنترلر است و CPU میتواند با استناد به آن فعالیت

های خود را زمان بندی کند .

10- پایه های میکرو چه کاری انجام میدهند ، مثلا مفهوم PB0 (OC0/T0) که در جلوی پایه ی شماره 1 میکرو کنترلر

موجود در تصویر بالا نوشته شده چیست ؟

✓ پایه های میکرو کنترلر معمولا دارای چندین نقش مجزا هستند و میتوانند در حالتی خاص پیکربندی شوند ، در

ادامه و در بخش برنامه نویسی در مورد کاربرد پایه ها توضیح داده ایم ، شما همچنین میتوانید با مراجعه به دیتاشیت

های فارسی میکرو کنترلر های AVR که در بخش ضمائم آورده شده است ، اطلاعات بیشتری در این باره بدست

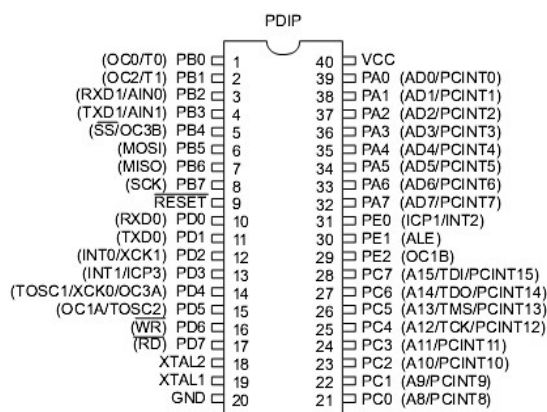
آورید .

11- آیا میکرو کنترلر های AVR در برابر نویز هستند ؟

✓ نویز یکی از عوامل ایجاد اختلال در مدارات الکترونیک (میکرو کنترلری و غیر میکرو کنترلری) میباشد و

یک طراح حرفه ای همیشه میتواند با اضافه کردن بخش های به مدارات الکترونیک خود ، از تاثیر نویز بر

روی آن جلوگیری کند . نویز پذیری میکرو کنترلر های AVR و عدم پایداری آنها تا کنون توسط منابع



رسمی برق و الکترونیک تایید نشده و شاید یکی از دلایل ایجاد این شبهه ، تعداد زیاد کاربران این میکروکنترلر باشد ؛ که اغلب آنها به دلیل تازه کار بودن ، طراحی غیر اصولی مدارات خود و عدم پاسخ مناسب آن را به گردن میکروکنترلر میاندازند .

در بخش ضmann مطالبی پیرو طراحی مناسب مدارات میکروکنترلی آورده شده است .

12- چگونه این کتاب را بخوانم ؟

✓ آموزش مطالب در این کتاب از سطح مبتدی شروع میشود و در سه فصل اول آن نحوه ی نوشتن برنامه و دستورات برنامه نویسی آموزش داده شده است و خواننده باید قبل از مطالعه ی مطالب موجود در فصل چهارم با کلیه مطالب این سه فصل آشنایی کامل داشته باشد .

در فصل چهارم ، در بخش های مختلف به بررسی نحوه ی راه اندازی امکانات داخلی میکروکنترلر و تجهیزات خارجی پرداخته شده است که کاربر میتواند بسته به نیاز و زمینه ی کاری خود مطالب موجود را بدون رعایت ترتیب مطالعه نماید .

در فصل ضmann کتاب ، چندین مطلب آموزشی برای بخش های مختلف کتاب آورده شده است که در فصول مختلف کتاب هر جا که نیاز باشد ، کاربر به ضمیمه ی مورد نظر ارجاع داده میشود .

13- امکانات سخت افزاری و نرم افزاری مورد نیاز برای یادگیری مطالب این کتاب چیست ؟

✓ در این کتاب شما به نرم افزار شبیه ساز پروتوس و کامپایلر بسکام نیاز خواهید داشت که میتوانید آنها را از سایت ایران میکرو دانلود نمایید . همچنین برای اجرا کردن برنامه ها به صورت عملی به یک سخت افزار نیاز خواهید داشت که مطالب مربوط به آن در بخش ضmann آورده شده است .

بدون شک سوالات دیگری نیز در ذهن شما وجود دارد ، که بیان کردن آنها از حوصله ی این کتاب خارج است ، شما میتوانید با مراجعه به سایت ایران میکرو و مطرح کردن سوال خود، در کمترین زمان جواب آن را بیابید :

فصل دوم

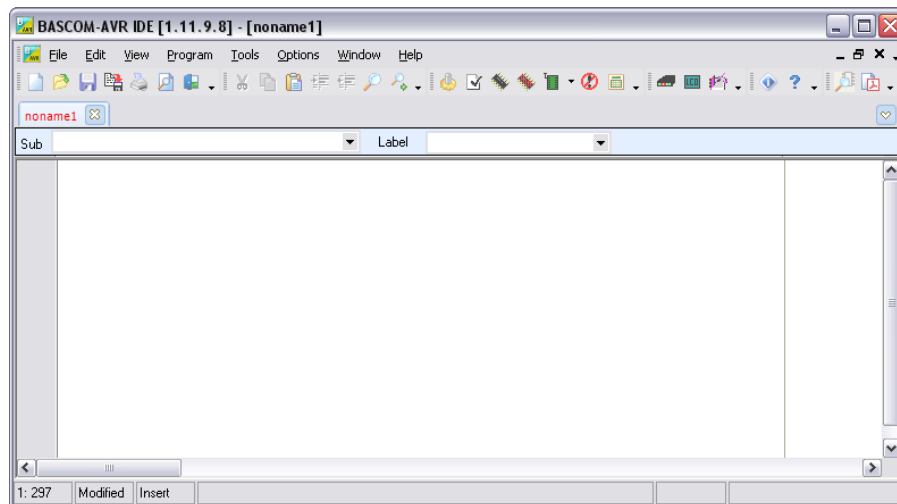
فصل دوم : اولین برنامه ی شما ، نحوه ی کار با lcd کارکتری و پورت های میکرو کنترلر

در این فصل قبل از هر کاری ، اقدام به بررسی نحوه نوشتن برنامه و استفاده از دستورات اولیه نموده ایم . بعد از اینکه با دستورات اولیه آشنا شدید ، نحوه کار با پورت ها و LCD کاراکتری را خواهید آموخت ، یادگیری نحوه کار با پورت ها و نمایشگر LCD شما را در درک ادامه آموزش و تست کردن مثال ها و پروژه ها یاری می دهد .

برای درک بهتر مثال ها و پروژه ها ، آنها را در بسکام کپی کنید و بعد از کامپایل کردن ، عملکردشان را در عمل یا با نرم افزار پروتوس مشاهده کنید ، مشاهده نحوه عملکرد دستورات ، شما را در یادگیری یاری میدهد .

مراحل نوشتن یک برنامه جدید (بدنه یک برنامه) :

بعد از باز کردن برنامه بسکام گزینه New را از منوی File انتخاب کنید صفحه جدیدی که باز می شود محل نوشتن برنامه می باشد .



در زبان بیسیک همیشه اولین خط برنامه مربوط به معرفی میکرو می باشد :

\$regfile = "micro name"

که گزینه micro neme کتابخانه میکرو مورد استفاده را مشخص می کند :

با دستور بالا و معرفی میکرو ، کامپایلر نوع میکرو مورد استفاده را تشخیص داده و کدهای خروجی را برحسب امکانات آن میسازد .مثال :

\$regfile = "1200def.dat" 'for AT90S1200

\$regfile = "2313def.dat" 'for AT90S2313

\$regfile = "2323def.dat" 'for AT90S2323

\$regfile = "2333def.dat" 'for AT90S2333

\$regfile = "2343def.dat" 'for AT90S2343

\$regfile = "4414def.dat" 'for AT90S4414

\$regfile = "4433def.dat" 'for AT90S4433

\$regfile = "4434def.dat" 'for AT90S4434

\$regfile = "8515def.dat" 'for AT90S8515

\$regfile = "8535def.dat" 'for AT90S8535

\$regfile = "86rf401.dat"	'for AT86RF401
\$regfile = "attiny12.dat"	'for attiny12
\$regfile = "attiny13.dat"	'for attiny13
\$regfile = "attiny15.dat"	'for attiny15
\$regfile = "attiny22.dat"	'for attiny22
\$regfile = "attiny24.dat"	'for attiny24
\$regfile = "attiny25.dat"	'for attiny25
\$regfile = "attiny26.dat"	'for attiny26
\$regfile = "attiny44.dat"	'for attiny44
\$regfile = "attiny45.dat"	'for attiny45
\$regfile = "attiny84.dat"	'for attiny84
\$regfile = "attiny85.dat"	'for attiny85
\$regfile = "attiny88.dat"	'for attiny88
\$regfile = "attiny261.dat"	'for attiny261
\$regfile = "attiny461.dat"	'for attiny461
\$regfile = "attiny861.dat"	'for attiny861
\$regfile = "attiny2313.dat"	'for attiny2313
\$regfile = "m103def.dat"	'for atmega103
\$regfile = "m1280def.dat"	'for atmega1280
\$regfile = "m128103.dat"	'for atmega128
\$regfile = "m128can.dat"	'for AT90CAN128
\$regfile = "m128def.dat"	'for atmega128
\$regfile = "m161def.dat"	'for ATMEGA161
\$regfile = "m162def.dat"	'for atmega162
\$regfile = "m163def.dat"	'for ATMEGA163
\$regfile = "m164pdef.dat"	'for ATMEGA164P
\$regfile = "m165def.dat"	'for atmega165
\$regfile = "m168def.dat"	'for atmega168

\$regfile = "m169def.dat"	'for atmega169
\$regfile = "m16def.dat"	'for atmega16
\$regfile = "m2560def.dat"	'for atmega2560
\$regfile = "m2561def.dat"	'for atmega2561
\$regfile = "m323def.dat"	'for ATMEGA323
\$regfile = "m324pdef.dat"	'for ATMEGA324P
\$regfile = "m325def.dat"	'for ATMEGA325
\$regfile = "m329def.dat"	'for atmega329
\$regfile = "m32def.dat"	'for ATMEGA32
\$regfile = "m48def.dat"	'for atmega48
\$regfile = "m406def.dat"	'for atmega406
\$regfile = "m603def.dat"	'for atmega603
\$regfile = "m640def.dat"	'for atmega640
\$regfile = "m644def.dat"	'for ATMEGA644
\$regfile = "m64def.dat"	'for ATMEGA64
\$regfile = "m649def.dat"	'for atmega649
\$regfile = "m645def.dat"	'for ATMEGA645
\$regfile = "m8515.dat"	'for atmega8515
\$regfile = "m8535.dat"	'for atmega8535
\$regfile = "m88def.dat"	'for atmega88
\$regfile = "m8def.dat"	'for atmega8
\$regfile = "m1281def.dat"	'for atmega1281
\$regfile = "usb1287.dat"	'for AT90USB1287
\$regfile = "usb162.dat"	'for AT90USB162
\$regfile = "m644pdef.dat"	'for ATMEGA644P
\$regfile = "m88pdef.dat"	'for atmega88p
\$regfile = "m168pdef.dat"	'for atmega168p
\$regfile = "m328pdef.dat"	'for atmega328p

```
$regfile = "m1284pdef.dat"           'for ATMEGA1284P
$regfile = "usb646.dat"              'for AT90USB646
$regfile = "m32can.dat"              'for AT90CAN32
$regfile = "m32u4def.dat"            'for ATMEGA32U4
$regfile = "xm128a1def.dat"          'for atxmega128a1
....
```

با مراجعه به محل نصب نرم افزار بسکام (مسیر پیش فرض C:\Program Files\MCS Electronics\BASCOM-AVR) میتوانید کتابخانه های میکرو کنترلر های موجود در نرم افزار را مشاهده کنید .

خط بعدی معرفی کریستال می باشد :

```
$crystal=x
```

که x کریستال مورد استفاده بر حسب هرتز است .

مثال : (در اینجا کریستال 8 مگا هرتز است) .

```
$crystal = 8000000
```

شما همچنین میتوانید مقدار کریستال را بر یک عدد تقسیم کنید (این کار برای مواردی که به یک کریستال دسترسی ندارید است می تواند مفید واقع شود)

```
Config Clockdiv = Constant
```

که Constant برابر این مقادیر است : 1 , 2 , 4 , 8 , 16 , 32 , 64 , 128 , 256 .

مثال :

```
$regfile = "m8def.dat"
```

```
$crystal = 8000000
```

```
Config Clockdiv = 8
```

از این به بعد میکرو با فرکانس 1 مگا هرتز کار میکند.

توجه :

✓ در صورتی که قصد دارید برای سری xmega برنامه بنویسید ، بعد از معرفی کریستال باید منابع کلاک میکرو را

پیکربندی نمایید ، برای کسب اطلاعات بیشتر به انتهای این فصل مراجعه کنید .

✓ کریستال درج شده در بالا ، با برنامه ریزی فیوز بیت ها فعال میشود ، توضیحات بیشتر در ادامه آورده شده است

بعد از معرفی کریستال نوبت به معرفی امکانات می باشد (امکانات شامل تایمر ها و ADC (مبدل آنالوگ به دیجیتال) و ورودی یا خروجی قرار دادن پورت ها و می باشد) . معرفی امکانات با دستور زیر شروع می شود:

Config

همچون :

Config Lcd = 16 * 2

بعد از معرفی یا پیکربندی امکانات جانبی نوبت به استفاده از آنها می باشد معمولاً برای استفاده از امکانات آن را درون یک حلقه قرار میدهند . و در نهایت برنامه با end به پایان می رسد .

پس همیشه یک برنامه در بسکام دارای قالب زیر است :

معرفی میکرو

تعیین مقدار کریستال

(پیکربندی منابع کلاک (فقط در سری xmega))

پیکربندی امکانات جانبی

معرفی متغیر ها و (در صورت وجود)

شروع حلقه (در صورت وجود)

برنامه اصلی

پایان حلقه (در صورت وجود)

پایان برنامه

زیر برنامه ها (در صورت وجود)

در ادامه و با مثال های مختلف با نحوه برنامه نویسی آشنا خواهیم شد .

دستورات مربوط به ورودی و خروجی ها :

در بسکام برای استفاده از یک پورت باید آن را به صورت ورودی یا خروجی پیکربندی کرد .

یک پورت هنگامی به عنوان خروجی تعریف میشود که بخواهیم از آن ولتاژ بگیریم و یک پورت هنگامی به عنوان ورودی قرار میگیرد که بخواهیم به آن ولتاژ بدهیم . (مثلا هنگامی که قصد داریم کلیدی را به میکرو متصل کنیم ، باید یک سر کلید را به 5 ولت و سر دیگر آن را به پایه های از میکرو متصل نماییم ، هنگامی که کلید فشرده میشود ولتاژ 5 ولت به پایه ی میکرو اعمال شده و میکرو متوجه تغییر وضعیت میشود ، گرفتن ولتاژ نیز همچون یک LED (چراغ) است که به بین گراند (صفر ولت) و یکی از پایه های میکرو کنترلر متصل شده و میتواند یک وضعیت خروجی را نمایش دهد .)

برای قرار دادن یک پورت به عنوان ورودی از دستور زیر استفاده میکنیم:

Config PORTX = Input

که PORTX یکی از پورت های میکرو می باشد و برای قرار دادن یک پورت به عنوان خروجی از دستور زیر استفاده میکنیم :

Config PORTX = Output

که PORTX یکی از پورت های میکرو می باشد.

مثال:

Config PORTA = Output

پورت A به عنوان خروجی تعریف شده است .

Config PORTB = Input

پورت B به عنوان ورودی تعریف شده است.

همچنین شما می توانید یکی از پایه های پورت (پین) را به عنوان ورودی یا خروجی معرفی کنید .

مثال :

Config PINA.1 = Input

پایه شماره 1 از پورت A (pina.1) به عنوان ورودی تعریف شده است.

Config PINB.7 = Input

پایه شماره 7 از پورت B به عنوان ورودی تعریف شده است .

Config PINC.5 = Output

پایه شماره 5 از پورت C به عنوان خروجی تعریف شده است .

دستورات مربوط به پورت ها :

: Toggle

این دستور یک بایت یا بیت را برعکس میکند، بایت میتواند یک پورت یا هر چیز دیگر باشد.

Toggle porta

: Set

این دستور یک بیت را یک میکند، بیت میتواند یک پین از پورت یا هر چیز دیگر باشد.

Set porta.0

: Reset

این دستور یک بیت را صفر میکند، بیت میتواند یک پین از پورت یا هر چیز دیگر باشد.

Reset portb

: Alias

از این دستور برای تغییر نام متغیر استفاده می شود . مثال:

Config PORTB.1 = Output

Led Alias PORTB.1

Set Led

حال شما می توانید در برنامه، بجای PORTB.1 از نام LED استفاده نمایید .

Bitwait :

با این دستور میکرو (cpu) مدام یک پین را چک می کند و هنگامی برنامه به خط بعدی می رود که پایه 1 یا 0 شد (صفر

یا یک بودن در برنامه مشخص میشود) این دستور به فرم کلی زیر است :

Bitwait X , Set/Reset

X: نام پایه است که قرار است چک شود، مثل PINA.0 یا PINB.7...

SET/RESET: شرط دستور میتواند SET یا ریست شدن پایه مورد نظر باشد، مثلاً با دستور:

Bitwait PORTA.0 , Set

هنگامی که PORTA.0 یک شد، دستورات ادامه اجرا میشود.

مثال:

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config PORTB.7 = Input
```

```
Config PORTB.6 = Output
```

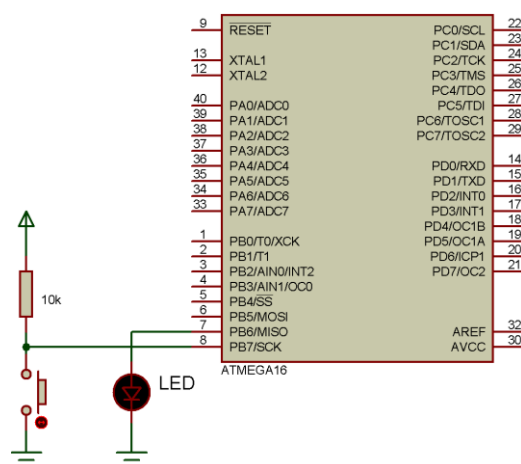
```
Bitwait PINB.7 , Reset
```

```
Set PORTB.6
```

```
Bitwait PINB.7 , Set
```

```
Reset PORTB.6
```

```
End
```



در مثال بالا همانگونه که مشاهده می کنید از میکرو مگا 16 (atmega16)، کلید و LED استفاده شده است. دو خط اول برنامه

معرفی میکرو و کریستال میباشد، میکرو مگا 16 و کریستال 1 مگا هرتز است.

در خط سوم پورت B.7 (پایه 8 میکرو) به عنوان ورودی تعریف شده است و کلید به آن متصل گردیده.

در خط چهارم پایه B.6 (پایه 7 میکرو) به عنوان خروجی تعریف شده است و LED به آن متصل گردیده، در خط پنجم

توسط دستور Bitwait پایه B.7 چک میشود و هنگامی که این پایه ریست شده (به زمین متصل شد یا صفر شد) CPU میکرو به

خط بعدی میرود (توجه داشته باشید که برنامه روی دستور Bitwait قفل میشود و تا زمانی که شرط (0 یا یک شدن پایه) بر

قرار نباشد این قفل شدن ادامه دارد و برنامه به خط بعدی نمی رود)

در خط بعدی یا خط ششم ، پایه B.6 یک می شود و LED روشن می شود ، در خط بعدی دو باره پایه B.7 چک می شود ، این بار شرط برعکس حالت قبل است ، یعنی اگر پایه 1 شود CPU میکرو به خط بعدی پرش میکند و LED را خاموش میکند.

برنامه با دستور End پایان میابد.

وظیفه مقاومت در اینجا بالا نگه داشتن (پول آپ) پایه B.7 میباشد (سطح یک را تصحیح میکند ، مثال را بدون مقاومت اجرا کنید تا ...)

این برنامه را اجرا کنید

همکنون میتوانید با مراجعه به بخش ضmann با استفاده یکی از روش های زیر برنامه را اجرا کرده و نتیجه ی ان را مشاهده کنید :

1- برنامه را به میکروکنترلر منتقل کرده و نتیجه را به صورت عملی مشاهده کنید < برنامه ریزی

میکروکنترلر و معرفی پروگرامر ها

2- از نرم افزار شبیه ساز پروتوس استفاده کنید < آموزش مقدماتی نرم افزار پروتوس

دستورات تاخیر :

برای ایجاد تاخیر در برنامه از دستور wait استفاده میشود.

دستور wait به سه شکل زیر است:

: Waitus X

این دستور برای ایجاد تاخیر میکرو ثانیه ای می باشد. X مقدار تاخیر میباشد که بین 1 تا 65535 میکرو ثانیه می باشد.

مثال :

Waitus 500

تاخیر به مدت 500 میکرو ثانیه .

: Waitms X

این دستور برای ایجاد تاخیر میلی ثانیه ای می باشد . X مقدار تاخیر میباشد که بین 1 تا 65535 میلی ثانیه می باشد.

مثال :

Waitms 720

تاخیر به مدت 720 میلی ثانیه .

: Wait X

این دستور برای ایجاد تاخیر ثانیه ای می باشد . X مقدار تاخیر میباشد که عددی بیشتر از یک ثانیه می باشد.

مثال :

Wait 1000

تاخیر به مدت 1000 ثانیه

: Delay

این دستور در هر جا که استفاده شود یک تاخیر 1 میلی ثانیه ایجاد می شود .

\$regfile = "m16def.dat"

\$crystal = 1000000

Config PORTB.7 = Input

Config PORTB.6 = Output

Bitwait PINB.7 , Reset

Set PORTB.6

Wait 1

Reset PORTB.6

Wait 1

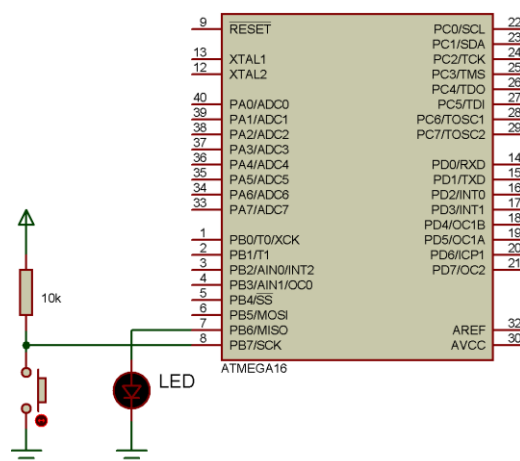
Set PORTB.6

Bitwait PINB.7 , Set

Reset PORTB.6

Wait 1

Toggle PORTB.6



End

دستورات حلقه و پرش :

گاهی اوقات نیاز است برنامه مدام اجرا شود یا در هنگام اجرای برنامه از یک خط به خط دیگری پرش شود.

برای این کار از حلقه ها و دستورات پرش استفاده میشود ، در اینجا به معرفی حلقه Do – Loop پرداخته و بقیه حلقه ها در بخش بعدی موجود است .

شروع این حلقه با Do و پایان آن با Loop است :

Do

برنامه

Loop

در این حلقه ، هنگامی که CPU به دستور Loop می رسد ، به ابتدای حلقه یا دستور DO پرش می کند .

برای پرش از یک قسمت برنامه به قسمت دیگر می توان از دستور Jmp یا Goto استفاده کرد (از دستورات فوق به عنوان حلقه نیز می توان استفاده کرد) .

مثال :

Q :

برنامه نوشته شده

Jmp Q

مثال دوم :

W :

برنامه نوشته شده

Goto W

حال با توجه به توضیحات بالا برنامه یک مدار چشمک زن را با هم می نویسیم :

(میکرو مورد استفاده مگا 16 (atmega16) و کریستال 8 مگا هرتز است و 8 عدد LED با مقاومت 330 اهم به پورت (PORTA)

A متصل است)

مرحله اول معرفی میکرو می باشد

```
$regfile = "m16def.dat"
```

مرحله بعد معرفی کریستال می باشد

```
$crystal = 1000000
```

(دو مرحله بالا در همه برنامه ها ثابت می باشد(وجود دارد))

مرحله بعد قرار دادن پورت A به عنوان خروجی می باشد (چون ما میخواهیم از آن ولتاژ بگیریم باید آن را به عنوان خروجی قرار دهیم)

```
Config PORTA = Output
```

مرحله بعد ایجاد یک حلقه می باشد (در اینجا برای ایجاد حلقه می توانید ، از تمام موارد گفته شده در بالا استفاده کنید)

```
Do
```

مرحله بعد روشن کردن Led ها می باشد(ما در اینجا آنها را یک در میان روشن میکنیم)

```
Set PORTA.0
```

(روشن کردن LED متصل شده به پایه 40 میکرو (PORTA.0))

```
Reset PORTA.1
```

(خاموش کردن LED متصل شده به پایه 39 میکرو (PORTA.1))

```
Set PORTA.2
```

(روشن کردن LED متصل شده به پایه 38 میکرو (PORTA.2))

```
Reset PORTA.3
```

(خاموش کردن LED متصل شده به پایه 37 میکرو (PORTA.3))

```
Set PORTA.4
```

(روشن کردن LED متصل شده به پایه 36 میکرو (PORTA.4))

```
Reset PORTA.5
```

(خاموش کردن LED متصل شده به پایه 35 میکرو (PORTA.5))

Set PORTA.6

(روشن کردن LED متصل شده به پایه 34 میکرو (PORTA.6))

Reset PORTA.7

(خاموش کردن LED متصل شده به پایه 33 میکرو (PORTA.7))

مرحله بعد ایجاد یک تاخیر زمانی است z

Waitms 300

(تاخیر به مدت 300 میلی ثانیه (برای اینکه روشن بودن LED ها دیده شود))

مرحله بعد برعکس کردن وضعیت پایه های موجود است (برای اینکه LED های روشن ، خاموش شود و LED های خاموش ، روشن شود).

Toggle PORTA

(این دستور همانگونه که قبلا گفته شد یک بایت را برعکس میکند که در اینجا بایت مورد نظر 8 پایه پورت A است).

مرحله بعد ایجاد تاخیر زمانی است :

Waitms 300

مرحله بعد نوشتن پایان حلقه می باشد :

Loop

(هنگامی که میکرو این خط را میخواند به خط Do پرش میکند)

و در نهایت برنامه با دستور زیر پایان می یابد :

End

چون برنامه هیچگاه پایان نمی یابد و دستورات موجود در حلقه مدام تکرار می شود پس می توان از دستور فوق چشم پوشی کرد!

برنامه بالا را به دو شکل زیر نیز می توان نوشت :

برنامه اول :

\$regfile = "m16def.dat"

```

$crystal = 1000000

Config PORTA = Output

Do

Set PORTA.0

Reset PORTA.1

Set PORTA.2

Reset PORTA.3

Set PORTA.4

Reset PORTA.5

Set PORTA.6

Reset PORTA.7

Waitms 300

Toggle PORTA

Waitms 300

Loop

End

```

برنامه دوم:

```

$regfile = "m16def.dat"

$crystal = 1000000

Config PORTA = Output

Do

Set PORTA.1

Reset PORTA.0

Set PORTA.3

Reset PORTA.2

Set PORTA.5

Reset PORTA.4

Set PORTA.7

Reset PORTA.6

Waitms 300

Set PORTA.0

Reset PORTA.1

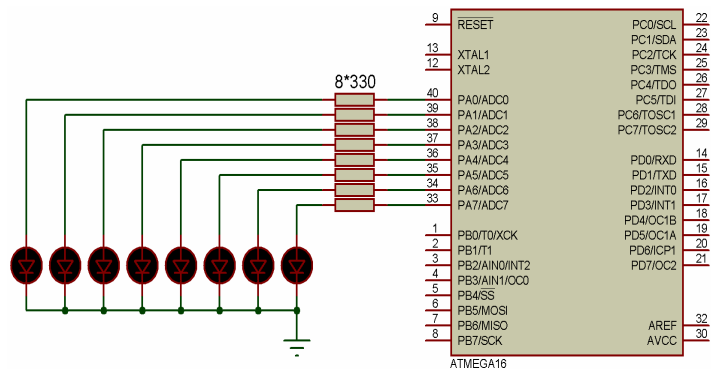
Set PORTA.2

Reset PORTA.3

Set PORTA.4

Reset PORTA.5

```



Set PORTA.6

Reset PORTA.7

Waitms 300

Loop

End

مدار به مورد نیاز برای این دو برنامه را در تصویر بالا مشاهده میکنید .

روش های دیگر هم برای نوشتن برنامه بالا وجود دارد که به عهده شما گذاشته می شود .

راه اندازی LCD کارکتری :



نمایشگر های گرافیکی کاربرد گسترده ای در دستگاه های الکترونیکی دارند، از این نمایشگر ها برای نمایش دادن وضعیت های مختلف در دستگاه استفاده میشود.

راه اندازی ساده، مصرف توان کم، قیمت ارزان و عمر بالا از ویژگی بارز این نمایشگر ها میباشد. اغلب این نمایشگر از کنترل کننده ی hd44780 استفاده میکنند معمولاً سازنده برای کم کردن قیمت نهایی قطعه، تراشه را بر روی برد آن پیدا سازی میکند و رویش را با لایه ای از چسب سخت میپوشاند.

دیتا شیت این lcd را میتوانید از ادرس زیر دانلود کنید (lcd و کنترل کننده ی آن):

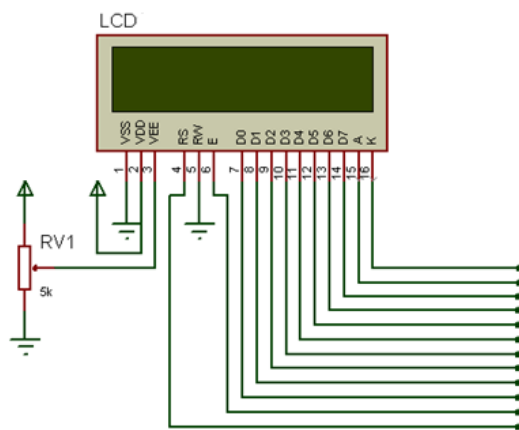
www.alldatasheet.com/q?hd44780

این lcd ها در نمونه های 1x16 و 2x16 و 2x20 ... ساخته میشوند، راه اندازی تمامی آنها مشابه هم است و با کنترل 4 یا 8 بیتی انجام میشود، تمامی این lcd ها دارای 16 پایه به شرح زیر میباشند :

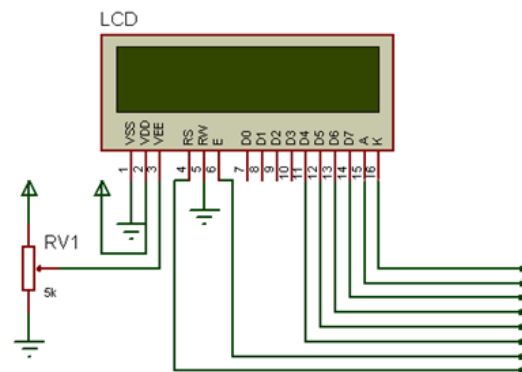
محل اتصال	کاربرد	نام پایه	شماره
گراند مدار	گراند قطعه	vss	1
ولتاژ 5 ولت	تغذیه مثبت قطعه	vdd	2
اتصال به vdd = حداکثر کنتراست، اتصال به گراند = کمترین کنتراست	کنترل کننده ی کنتراست	VO	3
یکی از پایه های میکرو که در برنامه مشخص می شود	انتخاب رجیستر	RS	4
یکی از پایه های میکرو که در برنامه مشخص می شود	نوشتن / خواندن	R/W	5
یکی از پایه های میکرو که در برنامه مشخص می شود	فعال ساز lcd	E	6
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB0	7
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB1	8
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB2	9
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB3	10
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB4	11
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB5	12
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB6	13

14	DB7	پایه دیتا	یکی از پایه های میکرو که در برنامه مشخص می شود
15	LED/A	انند led بکلایت	ولتاژ تغذیه مثبت
16	LED/K	کاتند led بکلایت	گراند مدار

برای ارتباط با این lcd میتوان از مد های 4 و 8 بیتی استفاده نمایید، در مد 4 بیت lcd به صورت شکل الف و در مد 8 بیتی lcd به صورت شکل ب به میکرو متصل میشود:



شکل ب



شکل الف

در مد 8 بیتی علاوه بر نوشتن داده بر روی LCD، توانایی خواندن اطلاعات آن را نیز دارید، برای خواندن LCD پایه RW باید به VCC متصل شود (در این حالت میتوان از LCD به عنوان یک حافظه ی موقت استفاده کرد، خوشبختانه با وجود حافظه های متنوع در میکرو کنترلر ها از این قابلیت استفاده ی چندانی نمیشود). نحوه ی کار lcd:

در صورتی که پشت lcd را مشاهده کنید، بر روی آن دو چیپ وجود دارد. اولین قطعه کنترل کننده ی hd44780 و دومین قطعه حافظه ی جانبی آن میباشد، ممکن است این قطعات به صورت chip یا cob (chip on board) بر روی برد lcd نصب شده باشند.

شرکت سازنده ی اطلاعات مربوط به تمامی کاکتر های اسکی را در حافظه قرار میدهد.... البته چون شما کامپایلر بسکام را انتخاب نموده اید، به این اطلاعات نیازی نخواهید داشت، پس به سراغ اصل مطلب میرویم.

راه اندازی LCD:

LCD می تواند از دو طریق 8 سیمه و 4 سیمه به میکرو متصل شود.

Config Lcdbus = Constant

Constant می تواند 4 به معنای استفاده از مد 4 سیمه یا 8 به معنای مد هشت سیمه باشد (در صورتی که این دستور نوشته نشود ، مد 4 سیمه در نظر گرفته میشود .)

در مد چهار سیمه فقط میتوان روی LCD نوشت ولی در مد هشت سیمه میتوان اطلاعاتی را که قبلا روی LCD نوشته شده است را خواند و به میکرو ارسال کرد .

در حالت نوشتن در LCD باید پایه RW (پایه 5) پایین نگه داشته شود (صفر شود) و در حالت خواندن از LCD باید پایه RW یک شود (5 ولت وصل شود) .

از آنجا که با وجود حافظه میکرو و راحت شدن کار برنامه نویسی نیازی به خواندن از LCD نمی باشد ، از پایه DB0 تا DB3 استفاده نمی شود و پایه RW نیز به GND (صفر ولت) متصل می شود .

دومین مرحله برای راه اندازی LCD معرفی کردن نام آن است :

برای این کار بعد از معرفی میکرو و کریستال با استفاده از دستور زیر می توان LCD را معرفی کرد:

```
Config Lcd = Lcdname
```

که Lcdname یکی از نام های بالا میباشد. مثلا معرفی LCD 2*16 :

```
Config Lcd = 16 * 2
```

مرحله بعد معرفی پایه های از میکرو است که LCD به آنها وصل میشود .

مد 4 سیمه :

```
Config Lcdpin = Pin , Db4 = Pinx.y , Db5 = Pinx.y , Db6 = Pinx.y , Db7 = Pinx.y , Rs = Pinx.y , E = Pinx.y
```

X نام پورت است که یکی از پورت های A یا B یا C یا D یا ... می باشد و y شماره پایه هست که از 0 تا 7 می باشد . (دقت

شود دستور فوق در یک خط نوشته شود یا برای خط دوم از _ استفاده شود).

برای مثال در زیر LCD کاراکتری 16*2 به پورت C (PORTC) متصل است .

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

Config Lcdpin = Pin , Db4 = PINC.0 , Db5 = PINC.1 , Db6 = PINC.2 _

, Db7 = PINC.3 , Rs = PINC.4 , E = PINC.5

در این مثال که از میکرو مگا16 (atmega16) استفاده شده ، اتصال میکرو و LCD به قرار زیر است :

پایه شماره 1: VSS این پایه باید به زمین مدار وصل شود .

پایه شماره 2: VDD این پایه باید به 5 ولت وصل شود .

پایه شماره 3: VEE این پایه با یک مقاومت (مقدار مقاومت بستگی به روشنایی مورد نظر شما دارد) به زمین وصل میشود.

پایه شماره 4: RS این پایه به پورت C پین شماره 4 یا پین C.4 متصل میشود (پایه 26) .

پایه شماره 5: RW این پایه به GND یا زمین متصل میشود .

پایه شماره 6: E این پایه به پورت C پین شماره 5 یا پین C.5 متصل میشود (پایه 27) .

پایه شماره 7: DB0 این پایه به جایی متصل نمی شود. (می توان آن را به زمین وصل کرد) .

پایه شماره 8: DB1 این پایه به جایی متصل نمی شود. (می توان آن را به زمین وصل کرد) .

پایه شماره 9: DB2 این پایه به جایی متصل نمی شود. (می توان آن را به زمین وصل کرد) .

پایه شماره 10: DB3 این پایه به جایی متصل نمی شود. (می توان آن را به زمین وصل کرد) .

پایه شماره 11: DB4 این پایه به پورت C پین شماره 0 یا پین C.0 متصل میشود (پایه 22) .

پایه شماره 12: DB5 این پایه به پورت C پین شماره 1 یا پین C.1 متصل میشود (پایه 23) .

پایه شماره 13: DB6 این پایه به پورت C پین شماره 2 یا پین C.2 متصل میشود (پایه 24) .

پایه شماره 14: DB7 این پایه به پورت C پین شماره 3 یا پین C.3 متصل میشود (پایه 25) .

پایه شماره 15: این پایه تغذیه مثبت LED پشت LCD می باشد که به 5 ولت متصل میشود .

پایه شماره 16: این پایه تغذیه زمین LED پشت LCD می باشد که به زمین متصل میشود .

توجه کنید که اگر پایه VEE به زمین متصل شود پیکسل های LCD دارای بیشترین کنتراست و اگر به 5 ولت وصل شود دارای

کمترین کنتراست می باشند .

اتصال پایه های 15 و 16 اختیاری است . (در پروتئوس این پایه ها وجود ندارند)

مد 8 سیمه :

در مثال زیر پایه DB0 به پورت D.0 و پایه DB1 به پورت D.1 و ... پایه DB7 به پایه D.7 و پایه Rs به پایه C.5 و پایه E به پایه

C.6 و پایه Rw به پایه C.7 متصل می گردد ، در این حالت یازده پایه از LCD اشغال می شود .

Config Lcdpin = PORTD , Rs = PINC.5 , E = PINC.6 , Rw = PINC.7

دستورات LCD :

بعد از راه اندازی LCD نوبت کار کردن با آن است . برای نوشتن روی LCD از دستور زیر استفاده می شود :

Lcd "x"

که X می تواند هر چیزی باشد (البته در محدوده کارکتر های اسکی) .

مثال :

Lcd "12345"

Lcd "1nafar"

برای پاک کردن LCD از دستور Cls استفاده می شود .

مثال :

Lcd "DSA@GJK!?"

Wait 1

Cls

Lcd "asdfgfhk"

با دستور زیر می توان در سطرها و ستون های دیگر LCD نوشت .

Locate x,y

که x آدرس سطر و y آدرس ستون می باشد .

مثال :

Locate 1 , 2

Lcd "qwert"

Locate 2 , 8

Lcd "mnbv"

توجه کنید برای یک LCD ، مثلا 16×2 حداکثر مقدار x برابر 2 و حداکثر مقدار y برابر 16 است .

البته این مدل LCD دارای حافظه 40×2 می باشد که فقط 16×2 تای اول آن نمایش داده می شود و می توان در بقیه حافظه نوشت و با دستورات انتقال متن فوق را نمایش داد .

با دستورات زیر میتوان به سطر های مختلف LCD پرش کرد و در آنجا متن را نوشت :

Upperline

با این دستور به خط بالای پرش میشود

Lowerline

با این دستور به خط پایینی پرش میشود

Home

با این دستور به سطر اول ، ستون اول پرش میشود

Thirdline

Fourthline

با دو دستور بالا می توان به ترتیب به خط سوم و چهارم پرش کرد (این دستور برای LCD های است که 4 سطر دارند) .

مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 4
```

```
Config Lcdpin = Pin , Db4 = PINC.0 , Db5 = PINC.1 , Db6 = PINC.2 , Db7 = PINC.3 , Rs = PINC.4 , E = PINC.5
```

Lowerline

```
Lcd "qwert"
```

Thirdline

```
Lcd "vcxz"
```

Fourthline

```
Lcd "erff"
```

```
Wait 2
```

Home

Lcd "123654"

End

با دستور زیر می توان تعداد فضای خالی دلخواه را بر روی LCD ایجاد کرد :

Lcd Spc(X)

تعداد X ستون خالی بوده ، و بعد از ستون X نوشتن ادامه می یابد.

با استفاده از دستور زیر میتوانید LCD را روشن یا خاموش کنید:

Display On / Off

On : LCD روشن میشود

Off : LCD خاموش میشود.

با استفاده از دستور زیر میتوانید کنتراست (میزان نور (کم رنگی و پر رنگی) متن) متن را تغییر دهید :

Lcdcontrast x

X میزان کنتراست است که میتوانید بین 0 تا 3 باشد ، به ازای 0 کمترین کنتراست و به ازای 3 بیشترین کنتراست مشاهده میشود. (توجه شود فقط بعضی از LCD ها دارای قابلیت فوق می باشند)

LCD دارای یک مکان نما می باشد که با دستور زیر میتوان آن را روشن یا خاموش یا چشمک زن یا ثابت قرارداد .

Cursor On

با این دستور مکان نما روشن می شود (در حالت عادی مکان نما روشن است).

Cursor Off

با این دستور مکان نما خاموش می شود.

Cursor Blink

با این دستور مکان نما چشمک می زند .

Cursor Noblink

با این دستور مکان نما دیگر چشمک نمی زند .

با دستور زیر می توانید کاراکتر های روی lcd را به چپ یا راست شیفت دهید.

ShiftLcd Left

این دستور کارکترها را به اندازه یک ستون به چپ منتقل میکند.

ShiftLcd Right

این دستور کارکترها را به اندازه یک ستون به راست منتقل میکند.

با دستور زیر میتوانید مکان نما را به راست یا چپ منتقل کنید:

Shiftcursor Left | Right

Left این دستور مکان نما را به اندازه یک ستون به چپ منتقل می کند .

Right این دستور مکان نما را به اندازه یک ستون به راست منتقل می کند.

مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 4
```

```
Config Lcdpin = Pin , Db4 = PIND.2 , Db5 = PIND.3 , Db6 = PIND.4 , _ Db7 = PIND.3 , Rs = PIND.0 , E = PIND.1
```

```
Lcdcontrast 1
```

```
Locate 2 , 1
```

```
Lcd "12356"
```

```
Shiftcursor Right
```

```
Wait 1
```

```
Display Off
```

```
Wait 1
```

```
Display On
```

```
Lcdcontrast 2
```

```
Locate 4 , 1
```

```
Lcd Spc (5)
```

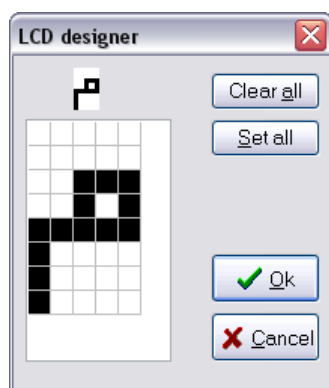
```
Lcd "qwer"
```

```
Shiftcursor Left
```

```
End
```

نمایش کارکتر دلخواه روی LCD :

Lcd های کاراکتری دارای یک حافظه دائم می باشد که درون آن فقط کد ، کارکترهای اسکی وجود دارد (کد کارکترهای فارسی در آن وجود ندارد). در LCD حافظه موقتی وجود دارد که در آن می توان تا 8 کاراکتر دلخواه را قرار داد .



برای ساخت کاراکتر دلخواه مراحل زیر را دنبال کنید:

از منوی Tools گزینه Lcd Designer را انتخاب کنید ، پنجره جدیدی باز می شود که شما

می توانید در آن کاراکتر دلخواه خود را ایجاد کنید.

بعد از ایجاد کاراکتر دلخواه روی Ok کلیک کنید ، پنجره بسته می شود و یک خط به

برنامه شما اضافه می شود. مانند زیر :

```
Deflcdchar ?,32,32,7,5,31,16,16,16
```

به جای علامت سوال ؟ باید یکی از اعداد 0 تا 7 گذاشته شود .

بعد از ساخت کاراکتر جدید بادرستور زیر می توانید آن را روی LCD نشان دهید:

```
Lcd chr (?)
```

به جای علامت سوال باید شماره کاراکتر که یکی از اعداد 0 تا 7 می باشد گذاشته شود .

مثال:

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 4
```

```
Config Lcdpin = Pin , Db4 = PIND.2 , Db5 = PIND.3 , Db6 = PIND.4 , Db7 = PIND.3 , Rs = PIND.0 , E = PIND.1
```

```
Deflcdchar 0 , 32 , 32 , 7 , 5 , 31 , 16 , 16 , 16
```

```
Lcd Chr(0)
```

```
End
```


در صفحه ی 20 نحوه ی مقدار تعیین فرکانس کاری میکرو را توضیح دادیم ، اما در سری xmega تنظیم کریستال مقداری متفاوت است !

پیکربندی منبع کلاک در سری xmega :

یکی از عمده مشکلاتی که کاربران در هنگام کار با سری های atmega و tany و at91s با آن رو برو میشوند برنامه ریزی فیوز بیت های مربوط به کریستال است . در صورتی که شما این فیوز بیت ها را به درستی برنامه ریزی نکنید ، عمل کرد میکرو غیر قابل پیش بینی و در برخی از موارد میکرو غیر قابل برنامه ریزی میشود .

در سری atmega فیوز بیت های مربوط به کریستال حذف شده و راه اندازی این بخش (منابع تامین کلاک) بر عهده واحد oscillators گذاشته شده است .

سری ایکس مگا دارای یک واحد مجزا برای تامین کلاک میباشد که این واحد میتواند کلاک مورد نیاز میکرو کنترلر را از oscillators داخلی ، یا کریستال خارجی در حالت های powermode ، یا ضرب در pll تامین کند ، حداقل و حداکثر فرکانس ایجاد شده در این بخش به ترتیب بین 2 تا 32 مگاهرتز میباشد . (با استفاده از pll میتوان این مقدار را تا 100 مگاهرتز افزایش داد (حال آور کلاک)) .

پیکره بندی کلاک سیستم در بسکام :

CONFIG OSC=ENABLED|DISABLED , PLLOSC=ENABLED|DISABLED, EXTOSC=ENABLED|DISABLED,

32KHZOSC=ENABLED|DISABLED, 32MHZOSC=ENABLED|DISABLED, RANGE=range,

32KHZPOWERMODE=powermode, STARTUP=startup

CONFIG OSC=ENABLED|DISABLED ✓

با انتخاب ENABLED یا DISABLED برای CONFIG OSC ، اسیلاتور داخلی میکرو به ترتیب فعال یا غیر فعال میشود ، اسیلاتور داخلی میکرو کنترلر به صورت پیش فرض فعال است و پالس کلاکی برابر 2 مگاهرتز تولید میکند .

PLLOSC=ENABLED|DISABLED ✓

با انتخاب ENABLED یا DISABLED برای PLLOSC ، واحد PLL داخلی میکرو به ترتیب فعال یا غیر فعال میشود و شما میتوانید با دستوراتی که در ادامه آورده میشود فرکانس کلاک میکرو را به مقادیر دلخواه تقسیم کنید .

EXTOSC=ENABLED|DISABLED ✓

با انتخاب ENABLED یا DISABLED برای EXTOSC، میکرو میتواند کلاک خود را از کریستال خارجی تامین کند، در این حالت باید اسیلاتور داخلی غیر فعال باشد.

32KHZOSC=ENABLED|DISABLED ✓

در سری XMEGA یک نوسان سازی داخلی 32KHZ برای تامین کلاک میکرو در مد های کم مصرفی در نظر گرفته شده است، با انتخاب ENABLED یا DISABLED برای 32KHZOSC میتوانید این نوسان ساز را فعال یا غیر فعال کنید.

32MHZOSC=ENABLED|DISABLED ✓

با انتخاب ENABLED یا DISABLED برای 32MHZOSC، نوسان ساز 32 مگاهرتزی میکرو کنترلر به ترتیب فعال یا غیر فعال میشود.

RANGE=range ✓

در صورتی که قصد دارید کلاک میکرو را از نوسان ساز خارجی تامین کنید، باید بعد از EXTOSC=ENABLED، دستور RANGE را با یکی از ارقام زیر مقدار دهی کنید:

- 400KHZ_2MHZ

- 2MHZ_9MHZ

- 9MHZ_12MHZ

- 12MHZ_16MHZ

32KHZPOWERMODE=powermode ✓

از دستور 32KHZPOWERMODE=powermode در حالتی استفاده میشود، که قصد داشته باشید مصرف توان میکرو کنترلر را کم کنید و به اصطلاح آن را به مد کم مصرفی ببرید، در فصل های بعد در مورد این مد ها بیشتر توضیح داده ایم.

STARTUP=startup ✓

دستور STARTUP میتواند یکی از مقادیر زیر را داشته باشد:

- EXTCLK (6 CLK)

- 32KHZ (for 16 CLK)

- XTAL_256CLK (for 256 CLK)

- XTAL_1KCLK (for 1K CLK)

- XTAL_16CLK (for 16K CLK)

با انتخاب زمان STARTUP، cpu تا n سیکل کلاک برای پایدار شدن شکل موج فرکانس کلاک منتظر میماند و سپس شروع به انجام وظایف می نماید.

بعد از اینکه منبع تامین کننده ی کلاک میکرو کنترلر مشخص شد، نوبت به تنظیمات تقسیمات کلاک میرسد:

CONFIG SYSCLOCK=sysclock, PRESCALEA=prescaleA, PRESCALEBC=prescaleBC

CONFIG SYSCLOCK=sysclock ✓

Sysclock نام منبع تامین کننده ی کلاک میکرو کنترلر که توسط دستورات قبلی پیکربندی شده میباشد و شامل موارد زیر است:

- 2MHZ

- 32MHZ

- EXTERNAL

- PLL

PRESCALEA=prescaleA ✓

با prescaleA کلاک تعیین شده میتواند به یکی از مقادیر 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 تقسیم شود.

PRESCALEBC=prescaleBC ✓

برای دستیابی به مقادیر دقیق تر کلاک prescaleB و C در نظر گرفته شده است، prescaleBC میتواند یکی از مقادیر زیر را داشته باشد:

- 1_1 (1 + 1 division)

- 1_2 (1+2 division)

- 4_1 (4 + 1 division)

- 2_2 (2 + 2 division)

مثال :

```
$regfile = "xm128a1def.dat"
```

```
Config Osc = Enabled , Pllosc = Enabled , Startup = Xtal_256clk
```

```
Config Sysclock = 2mhz , Prescalea = 2 , Prescalebc = 1_2
```

```
Config Portk = Output
```

```
Do
```

```
Portk = 255
```

```
Wait 1
```

```
Portk = 0
```

```
Wait 1
```

```
Loop
```

```
End
```

در این مثال بعد از معرفی میکرو کنترلر ATxmega128A1 توسط دستور regfile = "xm128a1def.dat" به پیکربندی منابع کلاک میکرو کنترلر پرداخته ایم :

```
Config Osc = Enabled , Pllosc = Enabled , Startup = Xtal_256clk
```

با دستور بالا اسیلاتور 2 مگاهرتزی داخلی و واحد PLL فعال شد است ، همچنین زمان Startup برای برابر 256 پالس کلاک در نظر گرفته شده است ، در این حالت CPU منتظر می ماند تا 256 پالس توسط منبع تولید کننده ی کلاک ایجاد و سپری شود و سپس به برنامه را اجرا میکند .

```
Config Sysclock = 2mhz , Prescalea = 2 , Prescalebc = 1_2
```

با دستور بالا ابتدا فرکانس کلاک به 2 و سپس به یک و مجدداً به 2 تقسیم میشود ، در این حالت فرکانس کاری نهایی برابر با 8 مگاهرتز خواهد بود .

در ادامه برنامه پورت K به عنوان خروجی تعریف شده و پایه های آن در هر یک ثانیه خاموش و روشن می شوند .

در بخش های بعدی مثال های بیشتری از XMEGA وجود دارد که دیدی بهتری از این موضوع به شما خواهد داد.

فصل سوم

فصل سوم: معرفی سایر دستورات بسکام

در این فصل دستورات به 7 دسته زیر تقسیم شده اند:

1- اعداد و متغیرها در بسکام

2- دستورات مربوط به کار با رشته ها

3- دستورات حلقه و پرش و شرط

4- دستورات اجرایی

5- زیر برنامه ها و فراخوانی توابع

6- توابع ریاضی و محاسباتی

7- توابع تبدیل کدها و متغیرها به یکدیگر

در این فصل نیازی به حفظ کردن دستورات نیست، برای یادگیری آنها کافیت آنها را اجرا کنید و چند بار ورودی را تغییر

دهید. در ادامه و در بخش راه اندازی سخت افزارهای جانبی، با کاربردهای عملی این دستورات بیشتر آشنا خواهیم شد.

اعداد و متغیر ها در بسکام :

دستور زیر معرفی یک متغیر را نشان میدهد . با این دستور می توانید متغیرهایی که در برنامه به کار برده می شوند تعریف کنید .

Dim X As Data Type

X نام متغیری است، که در برنامه بکار برده میشود و Data Type نوع داده است که می تواند طبق موارد WORD یا STRING

یا LONG یا INTEGER یا BYTE یا BIT یا SINGLE یا Double باشد . (X همچنین میتواند XRAM یا SRAM یا ERAM یا OVERLAY

یا location باشد که همگی متغیر های از انواع حافظه ها هستند ، که در بخشهای بعدی توضیح داده میشود.)

در صورت استفاده از متغیر STRING بیشترین طول آن نیز باید نوشته شود .

نام متغیر	توصیف
Bit	این متغیر میتواند صفر یا یک باشد
Byte	این متغیر می تواند از 0 تا 255 تغییر کند و فقط شامل اعداد صحیح مثبت می شود
Word	این متغیر می تواند از 0 تا 65535 تغییر کند و فقط شامل اعداد صحیح مثبت می شود.
Integer	این متغیر می تواند از -32767 تا +32767 تغییر کند و فقط شامل اعداد صحیح مثبت و منفی می شود.
Long	این متغیر می تواند از -214783648 تا +214783647 تغییر کند و فقط شامل اعداد صحیح مثبت و منفی می شود.
Single	این متغیر میتواند از $1/5 \times 10^{-45}$ تا $3/4 \times 10^{38}$ تغییر کند و فقط شامل اعداد صحیح و اعشاری صحیح مثبت و منفی می شود.
Double	این متغیر می تواند از 5×10^{-324} تا $1/7 \times 10^{308}$ تغییر کند و فقط شامل اعداد صحیح و اعشاری مثبت و منفی می شود.
String	این متغیر می تواند از 1 تا 255 بایت تغییر کند تغییر کند و برای حروف و علائم استفاده می شود. در صورت استفاده از متغیر String بیشترین طول آن نیز باید نوشته شود .

مثال :

Dim B As Bit

Dim A As Byte

Dim K As Integer

Dim Micro As Word

Dim Hasan As String * 16

شما همچنین میتوانید یک متغیر آرایه ای (با یک نام چندین متغیر) بسازید .

مثال:

Dim A(10) As Word

در این حالت شما میتوانید از 10 متغیر A یعنی A(1) تا A(10) در برنامه استفاده کنید .

در معرفی متغیر ها ، استفاده از متغیر صفر غیر مجاز میباشد (به عنوان مثال A(0) غیر مجاز است) ولی شما در کامپایلر بسکام ، در صورت نیاز میتوانید با دستور زیر از متغیر شماره ی صفر نیز استفاده کنید :

CONFIG BASE= value

در این دستور value عددی مثبت است که شماره ی متغیر های آرایه ای از آن شروع میشود . به عنوان مثال اگر value برابر با 0 باشد و دستور Dim A(10) As Word در برنامه استفاده شده باشد ، ما 10 متغیر A یعنی A(0) تا A(9) در برنامه خواهیم داشت . دستور CONFIG BASE فقط یک بار باید در برنامه استفاده شود .

مثال :

\$regfile = "m16def.dat"

\$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

Dim A(4) As Byte

Dim Ali As Word

Dim Wqew As Byte

A(1) = 10

A(2) = 11

Wqew = 5

Locate 1 , 1

A(2) = A(3) + A(4)

Ali = A(2) + Wqew

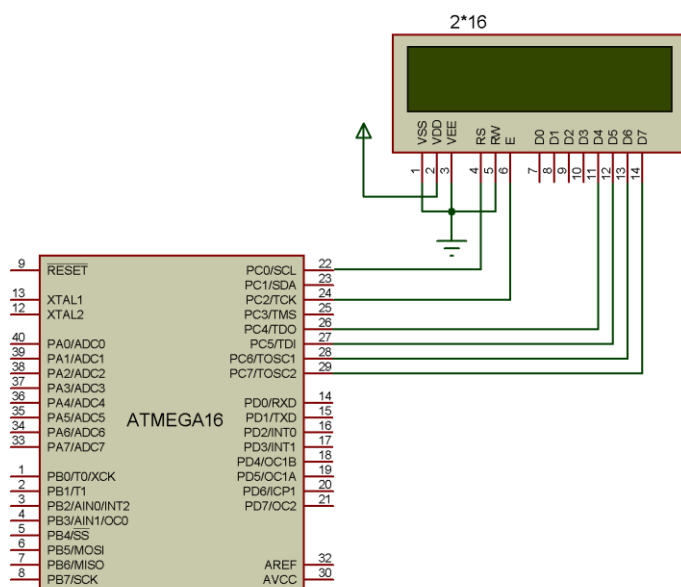
Lcd Ali

Locate 2 , 1

Ali = A(1) * Wqew

Lcd Ali

End



این تصویر ، سخت افزار استفاده شده در کلیه مثال های این فصل است .

توجه داشته باشید که برای استفاده از پورت C (برای راه اندازی LCD یا مقاصد دیگر) ابتدا باید فیوز بیت JTAG را از کار بیندازید تا پایه های این پورت به I/O عمومی تبدیل شوند ، برای کسب اطلاعات بیشتر به بخش ضمائم مراجعه کنید .

نکته : در صورتی که در یک متغیر بیشتر از اندازه اش مقدار قرار دهید با خطا مواجه میشوید.

مثال:

Dim A As Byte

A = 300

مورد بالا غلط می باشد ، چون بایت می تواند از 0 تا 255 تغییر کند و مقدار 300 بیشتر از اندازه (بعد) بایت است .

فرم دیگر تعریف متغیرها به شکل زیر است :

DEFBIT x	Define BIT
DEFBYTE x	Define BYTE
DEFINT x	Define INTEGER
DEFWORD x	Define WORD
DEFLNG x	Define LONG
DEFSNG x	Define SINGLE
DEFDBL x	Define DOUBLE

X : نام متغیر است که میتواند یکی از حروف انگلیسی باشد و حدود تغییر متغیر مانند موارد قبل بوده و فقط شکل نوشتن دستور عوض شده است.

Const:

برای تعریف یک ثبات از این دستور استفاده می شود :

Const Symbol= Numconst

Const Symbol= Stringconst

Const Symbol= Expression

Symbol نام ثابت و Numconst مقدار عددی انتساب یافته به Symbol و Stringconst رشته انتساب یافته به Symbol و

Expression می تواند عبارتی باشد که نتیجه آن به Symbol انتساب یابد .

مثال:

Const S = "TEST"

Const A = 5

Const B1 = &B1001

Const X =(b1 * 3) + 2

Incr و Decr :

دستور Incr یک واحد به متغیر عددی X می افزاید و دستور Decr یک واحد از آن کم می کند .

Incr X

Decr X

مثال :

\$regfile = "m16def.dat"

\$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

Dim A As Byte

Dim B As Long

Dim Bp As Byte

Do

Incr A

Decr B

Locate 1 , 1

Lcd B

Locate 2 , 1

```
Lcd A
```

```
Waitms 500
```

```
Loop
```

```
End
```

: Swap

با اجرای این دستور محتوای متغیر Var1 در متغیر Var2 و محتوای متغیر Var2 در متغیر Var1 قرار می گیرد .

```
Swap Var1 , Var2
```

دو متغیر Var1 و Var2 بایستی از یک نوع باشند . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim A As Byte
```

```
Dim B As Byte
```

```
Cls
```

```
A = 10
```

```
B = 20
```

```
Swap A , B 'swap them
```

```
Locate 1 , 1
```

```
Lcd A 'A=20
```

```
Locate 2 , 1
```

```
Lcd B 'B=10
```

```
End
```

:Config Single

با این دستور میتوان تعداد رقم اعشار متغیر از جنس Single را معین کرد ، این دستور به فرم کلی زیر است:

```
Config Single = Scientific , Digits = x
```

X: عددی بین 1 تا 7 است که تعداد رقم اعشار را نشان میدهد ، در صورتی که از این دستور استفاده کنید ، کلیه متغیر های

Single موجود در برنامه تحت پوشش قرار می گیرند . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Config Single = Scientific , Digits = 1
```

```
Cls
```

```
Dim A As Single
```

```
A = 10
```

```
A = A / 9
```

```
Lcd A
```

```
End
```

: Format

این دستور یک متغیر عددی را شکل دهی می کند .

```
X = Format (Var , "Form")
```

Var رشته ای است که شکل دهی شود و نتایج در X قرار می گیرد. Form نوع شکل دهی است . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim S As String * 10 , I As Integer
```

```
S = " 123"
```

```
S = Format (s , " " ) '5 Space
```

```
Locate 1 , 1
```

```
Lcd S ' S = " 123"Two Space First , Then 123
```

```
S = "12345"
```

```
S = Format (s , "000.000")
```

```
Locate 1 , 8
```

```
Lcd S ' S = "012.345"
```

```
S = Format (s , "+")
```

```
Locate 2 , 1
```

```
Lcd S ' S = "+12345"
```

```
End
```

: Fusing

از این دستور برای روند کردن یک متغیر عددی استفاده می شود .

```
Target = Fusing (Source , "Mask")
```

Source رشته موردنظر برای شکل دهی و نتایج در Target قرار می گیرد. Mask نوع شکل دهی است. عمل Mask حتما باید با علامت # شروع شود و حداقل باید یکی از علامات # یا & را بعد از ممیز داشته باشد. با استفاده از # عدد روند می شود و در صورت استفاده از & روندی صورت نمی گیرد.

مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 4
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim S As Single
Dim A As Byte
Cls
S = 10
A = 6
S = S / A
Locate 1 , 1
Lcd S 'lcd 1.6666666666666666
Locate 2 , 1
Lcd Fusing (s , "#.##") 'lcd 1.67
Locate 3 , 1
Lcd Fusing (s , "#.####") 'lcd 1.6667
Locate 4 , 1
Lcd Fusing (s , "#.&&&") 'lcd 1.666
End
```

: Shift Var

با این دستور میتوان تمام بیت ها را یک بیت به سمت راست یا چپ منتقل کرد ، این دستور به فرم کلی زیر است :

Shift Var , Left/Right

Var : نام متغیر یا عدد ثابتی است که میخواهیم آن را شیفت دهیم (منتقل کنیم) Left/Right جهت شیفت را مشخص میکند ،

که میتواند راست یا چپ باشد . مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
```

```
Config PORTA = Output
```

```
PORTA = 128
```

```
Do
```

```
Shift PORTA , Right
```

```
Wait 1
```

```
Loop
```

```
End
```

مثالی دیگر :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim A As Byte
```

```
A = 5
```

```
Locate 1 , 1
```

```
Lcd a
```

```
Shift A , Left
```

```
Locate 2 , 1
```

```
Lcd A
```

```
End
```

در مثال بالا مقدار اولیه 5 (0101 باینری) برای A در نظر گرفته شده است ، این مقدار بر روی سطر و ستون اول LCD به نمایش

در میاید ، سپس با دستور Shift A , Left تمام بیت های متغیر A به سمت چپ منتقل میشوند ، پس مقدار A برابر با 10 (1010

باینری) می شود .

: Rotate

این دستور تقریباً مشابه دستور Shift میباشد و تمام بیت های یک متغیر را به سمت راست یا چپ جابجا می کند با این تفاوت

که بیتی بیرون نمی رود و چرخش داده می شود ، شما همچنین با این دستور میتوانید پین های روشن یک پورت را جابجا

کنید و به فرم کلی زیر است:

```
Rotate Var , Left/Right
```

Var : نام متغیر یا عدد ثابتی یا پورتهی است که میخواهیم آن را شیفت دهیم (منتقل کنیم)

Left/Right : جهت انتقال را مشخص می کند ، که میتواند راست یا چپ باشد .

مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
Config PORTA = Output
Dim B As Byte
B = 1
Do
Rotate B , Right
PORTA = B
Waitms 250
Loop
End
```

: Shiftin

با این دستور میتوان تعداد بیت را درون یک متغیر شیفت داد ، بیت ها بصورت سریال به یکی از پایه ای میکرو اعمال میشوند . این دستور به فرم کلی زیر است :

Shiftin Pin , Pclock , Var , Option [, Bits , Delay]

Pin : نام پایه ای است که اطلاعات سریال به آن وارد می شوند .

Pclock : نام پایه ای است که خط کلاک دستگاه دیگر به آن متصل میشود .

Var : نام متغیری است که اطلاعات در آن ذخیره میشوند .

Option : نوع شیفت دادن و کلاک را معین میکند و یکی از اعداد زیر است :

1 : هنگامی که فرکانس کلاک کم باشد ابتدا MSB (بیت بازشر) شیفت داده می شود.

2 : هنگامی که فرکانس کلاک زیاد باشد ابتدا MSB (بیت بازشر) شیفت داده می شود.

3 : هنگامی که فرکانس کلاک کم باشد ابتدا LSB (بیت کم ارزش) شیفت داده می شود.

4 : هنگامی که فرکانس کلاک زیاد باشد ابتدا LSB (بیت کم ارزش) شیفت داده می شود.

Bits : مشخص کننده تعداد بیت است که وارد متغیر می شود و نهایتا می تواند 255 باشد (این گزینه اختیاری است)

Delay : تاخیر زمانی برحسب میکرو ثانیه می باشد که در بین دریافت هر بیت رخ می دهد (استفاده از این گزینه اختیاری است) (در صورت بالا بودن کلاک از این گزینه استفاده نکنید).

مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config PORTA = input
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim A As Word
Shiftin PINA.0 , PINA.1 , A , 0
End
```

در مثال بالا پایه کلاک پین A.1 و پایه دیتا پایه A.0 میباشد ، کلاک از میکرو دیگر که برنامه آن را در دستور بعدی مشاهده می فرماید ، گرفته شده است .

: Shiftout

با این دستور میتوان یک متغیر را بصورت سریال از یک پایه به بیرون داد . این دستور به فرم کلی زیر است:

Shiftout Pin , Pclock , Var , Option [, Bits , Delay]

Pin : نام پایه ای است که اطلاعات سریال از آن خارج میشوند.

Pclock : نام پایه ای است که خط کلاک دستگاه دیگر به آن متصل میشود.(خروجی کلاک است)

Var : نام متغیری است که اطلاعات در آن وجود دارد و باید ارسال شود.

Option : نوع شیفت دادن و کلاک را معین میکند و یکی از اعداد زیر است:

- 1 : هنگامی که فرکانس کلاک کم باشد ابتدا MSB (بیت بازشر) شیفت داده میشود.
- 2 : هنگامی که فرکانس کلاک زیاد باشد ابتدا MSB (بیت بازشر) شیفت داده میشود.
- 3 : هنگامی که فرکانس کلاک کم باشد ابتدا LSB (بیت کم ارزش) شیفت داده میشود.
- 4 : هنگامی که فرکانس کلاک زیاد باشد ابتدا LSB (بیت کم ارزش) شیفت داده میشود.

Bits : مشخص کننده تعداد بیت از متغیر است که به بیرون شیفت داده میشود و نهایتاً می تواند 255 باشد (این گزینه اختیاری است)

Delay : تاخیر زمانی برحسب میکرو ثانیه میباشد که در بین ارسال هر بیت رخ میدهد (استفاده از این گزینه اختیاری است) (در صورت بالا بودن کلاک از این گزینه استفاده نکنید).

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config PORTA = Output
Dim A As Word
A = &B11001000 '200
Shiftout PINA.0 , PORTA.1 , A , 0
End
```

نکته :

- در بعضی از کتاب ها از دستورات SHIFT in/out ، با نام spi نرم افزاری یاد شده است .
- برای نشان دادن اعداد به فرم باینری از &B و برای نشان دادن اعداد به فرم هگز از &H استفاده می شود.

&B01010111

&Hf6

دستورات مربوط به کار با رشته ها :

: Asc

```
Var = Asc (String)
```

این دستور اولین کاراکتر یک متغیر از نوع داده String را به مقدار اسکی آن تبدیل می کند .

برای دیدن کارکترهای اسکی و کد متناظر با آنها از منوی Edit گزینه ی Insrt Ascii را انتخاب کنید .

: Instr

این دستور محل و موقعیت یک زیر رشته را در رشته دیگر مشخص می کند .

```
Var =Instr (start , String ,Subset)
```

```
Var =Instr (String ,Subset)
```

Var عددی است که مشخص کننده محل SUBSTR در رشته اصلی STRING می باشد و زمانیکه زیر رشته مشخص شده در

رشته اصلی نباشد صفر برگردانده می شود . START نیز عددی دلخواه است که مکان شروع جستجو زیر رشته در رشته اصلی

را مشخص می کند . در صورتیکه START قید نشود تمام رشته از ابتدا جستجو می شود . رشته اصلی تنها باید از نوع رشته

باشد ولی زیر رشته (SUBSTR) می تواند رشته و عدد ثابت هم باشد .

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim S As String * 15
```

```
Dim Z As String * 5
```

```
Dim Bp As Byte
```

```
Cls
```

```
S = "This is a test"
```

```
Z = "is"
```

```
Bp = Instr (s , Z)
```

```
Lcd Bp
```

```
Bp = Instr (4 , S , Z)
```

```
Lcd Bp
```

```
End
```

: CHECKSUM

این دستور مجموع کد دسیمال اسکی رشته X را برمی گرداند که البته اگر مجموع کد اسکی رشته از عدد 255 بیشتر شود مقدار 256 از مجموع کم می شود.

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim S As String * 10 ' Dim Variable
S = "test" Locate 1 , 1 ' Assign Variable
Lcd Checksum (s) ' print value (192)
S = "testNext"
Locate 2 , 1 ' assign variable
Lcd Checksum (s) ' lcd value 127 (127=383 - 256)
End
```

: Highw و Low و High

Low : این دستور LSB (least significant byte) یک متغیر را برمی گرداند . (LSB باید کمتر از 8 بیت باشد)

High : این دستور MSB (most significant byte) یک متغیر را برمی گرداند . (MSB باید کمتر از 8 بیت باشد)

High : این دستور MSB (most significant byte) یک متغیر را برمی گرداند . (MSB باید کمتر از 16 بیت باشد)

Var = High (s)

MSB متغیر S در Var قرار می گیرد . (var باید از جنس byte باشد)

Var = Low (s)

LSB متغیر S در Var قرار می گیرد . (var باید از جنس byte باشد)

X = Highw (s)

MSB متغیر S در Var قرار می گیرد . (var باید از جنس word باشد)

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim I As Integer
```

```
Dim Z As Byte
```

```
Dim Q As Byte
```

```
Cls
```

```
I = &h1001
```

```
Z = Low (i) ' is 1
```

```
Locate 1 , 1
```

```
Lcd Z
```

```
Q = High (i) 'IS 16
```

```
Locate 2 , 1
```

```
Lcd Q
```

```
End
```

مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim X As Word , L As Long
```

```
L = &H12345678
```

```
X = Highw (l) ' 4660(des)=1238(hex)
```

```
Locate 2 , 1
```

```
Lcd X
```

```
End
```

: Ucase و Lcase

دستور Lcase: این دستور تمام حروف رشته مورد نظر را تبدیل به حروف کوچک می کند .

```
Target = Lcase (source)
```

تمام حروف رشته source کوچک شده و در رشته Target جای داده می شود .

دستور Ucase: این دستور تمام حروف رشته مورد نظر را تبدیل به حروف بزرگ می کند .

```
Target = Ucase (source)
```

تمام حروف رشته source بزرگ شده و در رشته Target جای داده می شود .

```
$regfile = "m16def.dat"
```

```

$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim S As String * 12
Dim Z As String * 12
Dim Q As String * 12
S = "Hello World"
Q = "QWERTGFDD"
Z = Ucase (s) 'Z = HELLO WORLD
Locate 1 , 1
Lcd Z
Z = Lcase (q)
Locate 2 , 1
Lcd Z
End

```

: Left و Right

دستور RIGHT: با این دستور قسمتی از یک رشته را جدا می کنیم .

Var = RIGHT (var1 , n)

از سمت راست رشته var1 به تعداد کاراکتر n , رشته ای جدا شده و در رشته var قرار می گیرد .

دستور LEFT: با این دستور کاراکترهای سمت چپ یک رشته را به تعداد تعیین شده جدا می کند .

Var = LEFT(var1 , n)

از سمت چپ رشته var1 به تعداد کاراکتر n , رشته ای جدا شده و در رشته var قرار می گیرد .

مثال :

```

$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim S As String * 15 , Z As String * 15
Cls
S = "abcdefg"
Z = Left (s , 5) 'Z = abcde
Locate 1 , 1

```

```

Lcd Z
Z = Left (s , 1) 'Z = a
Locate 1 , 8
Lcd Z
Z = Right (s , 5) 'Z = CDEFG
Locate 2 , 1
Lcd Z
Z = Right (s , 2) 'Z = FG
Locate 2 , 8
Lcd Z
End

```

: Peek

با این دستور میتوان بیت n در یک متغیر یا رجیستر را در متغیری همچون var قرار داد (صفر یا یک بودن بیت را مشخص کرد) :

```
Var= peek( address )
```

دستوراتی دیگری همچون INP و POKE و OUT نیز مشابه دستور بالا برای خواندن یا نوشتن مقادیر دلخواه در مکان های مختلف یک متغیر یا رجیستر استفاده میشوند، این دستورات به فرم کلی زیر هستند :

```
OUT address, value
```

توسط دستور بالا میتوان متغیر value را به ادرس address در سخت افزار خارجی یا حافظه ی داخلی / خارجی ارسال کرد .

```
POKE address , value
```

با دستور POKE میتوان متغیر value (0-255) را در رجیستر شماره ی address نوشت (مثلاً 1 به عنوان R1)

```
var = INP(address)
```

دستور INP یک بایت از حافظه ی داخلی / خارجی یا سخت افزار خارجی را در متغیر VAR قرار می دهد .

مثال :

```

$regfile = "m162def.dat"      ' specify the used micro
$crystal = 4000000             ' used crystal frequency
$baud = 19200                  ' use baud rate
$hwstack = 32                  ' default use 32 for the hardware stack
$swstack = 10                  ' default use 10 for the SW stack
$framesize = 40                ' default use 40 for the frame space

Dim I As Integer , B1 As Byte

```

```
'dump internal memory
```

```
For I = 0 To 31 'only 32 registers in AVR
```

```
B1 = Peek(i) 'get byte from internal memory
```

```
Print Hex(b1); " ";
```

```
'Poke I , 1 'write a value into memory
```

```
Next
```

```
Print 'new line
```

```
'be careful when writing into internal memory !!
```

```
'now dump a part of the code-memory(program)
```

```
For I = 0 To 255
```

```
B1 = Cpeek(i) 'get byte from internal memory
```

```
Print Hex(b1); " ";
```

```
Next
```

```
'note that you can not write into code memory!!
```

```
Out &H8000 , 1 'write 1 into XRAM at address 8000
```

```
B1 = Inp(&H8000) 'return value from XRAM
```

```
Print B1
```

```
End
```

: Len

این دستور طول، یا عبارتی تعداد کاراکترهای یک رشته را برمیگرداند.

```
Var = Len (string)
```

طول رشته String در متغیر عددی VAR قرار می گیرد. رشته String نهایتاً می تواند 255 بایت طول داشته باشد. توجه داشته باشید که فضای خالی (SPACE BAR) خود یک کاراکتر به حساب می آید.

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim S As String * 12
```

```
Dim A As Byte
```

```
Cls
```

```
S = "test"
```

```
A = Len (s)
```

```
Locate 1 , 1 ' 4
```

```
Lcd Len (s)
```

```
S = "test "
```

```
A = Len (s)
```

```
Locate 2 , 1
```

```
Lcd A '6
```

```
End
```

: Ltrim

این دستور فضای خالی یک رشته را حذف می کند .

```
Var = Ltrim (Q)
```

فضای خالی رشته Q برداشته می شود و رشته بدون فضای خالی در متغیر رشته ای var قرار می گیرد .

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim S As String * 10
```

```
Dim A As String * 10
```

```
Cls
```

```
S = "Q Q 1"
```

```
Locate 1 , 1
```

```
A = Ltrim (s)
```

```
Lcd A 'QQ1
```

```
S = "Q Q Q"
```

```
Locate 2 , 1
```

```
Lcd Ltrim (s) 'QQQ
```

```
End
```

: Mid

با این دستور می توان قسمتی از یک رشته را برداشت و یا قسمتی از یک رشته را با قسمتی از یک رشته دیگر عوض کرد .

```
Var = Mid (Var1 , St [,L])
```

1- قسمتی از رشته Var1 با شروع از کاراکتر St ام و طول L برداشته شده و در متغیر Var قرار می گیرد.

```
Mid (Var , St [,L]) = Var1
```

2- رشته Var1 در رشته Var با شروع از کاراکتر St ام و طول L قرار می گیرد .

در صورت قید نکردن گزینه اختیاری L بیشترین طول در نظر گرفته می شود .

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim S As String * 10
Dim Z As String * 10
Cls S = "adswer"
Z = Mid (s , 2 , 3)
Locate 1 , 1
Lcd Z 'lcd "dsw"'
Z = "5685"
Z =Mid (s , 2 , 3)
Locate 2 , 1
Lcd S 'lcd "a568er"'
End
```

: Space

برای ایجاد فضای خالی در میان یک رشته ، از این دستور استفاده می شود .

Var = space (X)

X تعداد فضای خالیست که بعنوان رشته در متغیر رشته ای var جای می گیرد.

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim S As String * 10
Dim Z As String * 10
Cls S = Space (5)
Z = "qwer"
Locate 1 , 1
Lcd "(" ; S ; Z ; ")" 'lcd qwer
End
```


دیگر دستورات حلقه و پرش و شرط :

گاهی نیاز است که یک قسمت از برنامه چندین بار اجرا شود یا در حین اجرای برنامه در یک خط به خط دیگری رجوع شود، برای این کار از دستورات حلقه و پرش که چندین نوع است ، استفاده میشود .

انواع دستورات حلقه و پرش:

: Goto

Label:

برنامه

Goto Label

با این دستورات می توان به برچسب Label پرش کرد .برچسب Label باید با علامت : (colon) پایان یابد و می تواند تا 32 کارکتر طول داشته باشد .

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config PORTA = Output
```

```
Q:
```

```
Set PORTA.0
```

```
WAitms 600
```

```
Reset PORTA.0
```

```
Waitms 600
```

```
Goto Q
```

```
End
```

: Gosub

با این دستور میتوان به یک زیربرنامه پرش کرد ، بازگشت از زیربرنامه با دستور Return انجام میشود ، این دستور به فرم کلی زیر است:

Gosub Label

Label:

برنامه

Return

مثال:

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config PORTA = Output
```

```
Do
```

```
Gosub Q
```

```
Set PORTA.0
```

```
WAitms 600
```

```
Loop
```

```
End
```

```
Q:
```

```
Reset PORTA.0
```

```
Waitms 600
```

```
Return
```

: On Var

این دستور به فرم کلی زیر است:

```
On Var [Goto] [Gosub] Label1 [, Label2 ] [,Check]
```

در این دستور به ازای متغیر Var به برچسب n^m پرش میشود ، مثلا اگر Var=2 باشد به برچسب سوم پرش میشود (تعداد

برچسب ها نامحدود است) بستگی به حافظه میکرو دارد . در صورتی که از دستور Gosub استفاده کنید باز گشت از زیر

برنامه (برچسب) با دستور Return انجام می شود و در صورت استفاده از Goto باید به حلقه اصلی پرش کرد که شما میتوانید

از دستور Goto یا دیگر دستورات استفاده کنید .

مثال با دستور Gosub :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config PORTA = Output
```

```
Dim S As Byte
```

```
Do
```

```
On S Gosub Q , W , E , R , T , Y , U , I
```

```
Incr S
```

Wait 1

Loop

End

Q:

Set PORTA.0

Return

W:

Set PORTA.1

Return

E:

Set PORTA.2

Return

R:

Set PORTA.3

Return

T:

Set PORTA.4

Return

Y:

Set PORTA.5

Return

U:

Set PORTA.6

Return

I:

Set PORTA.7

Return

مثال با دستور Goto :

\$regfile = "m16def.dat"

\$crystal = 1000000

Config PORTA = Output

Dim S As Byte

A:

Wait 1

On S Goto Q , W

End

Q:

Set PORTA.0

Incr S

Goto A

W:

Set PORTA.1

Goto A

در مثال های بالا ، در اول کار مقدار S صفر است پس به پرچسب اول پرش شده و پورت PORTA.0 یک میشود میشود و به S یک واحد افزوده میشود ، بعد از این عمل دوباره به حلقه اصلی پرش ، S برابر با 1 میشود ، پس به پرچسب دوم پرش میشود در آنجا پورت ...

: Do - Loop

فرم کلی دستورات DO ... LOOP بصورت زیر می باشد .

Do

برنامه

Loop

این حلقه یک حلقه بینهایت است ، که با EXIT DO می توان از درون حلقه خارج شد و اجرای برنامه در خط بعد از Loop ادامه یابد.

همچنان می توان شرطی تعریف کرد که با برقرار شدن شرط از حلقه خارج شویم :

Do

برنامه

Loop Until A = X

هر گاه متغیر A برابر X شد از حلقه خارج می شویم ، تست شرط در آخر حلقه انجام می شود پس حلقه فوق حداقل یک بار اجرا می شود :

در مثال زیر در هر بار تکرار حلقه یک واحد به A اضافه می گردد و هرگاه مقدار A به 10 رسید خط بعد از حلقه اجرا می گردد. مثال :

```

$regfile = "m16def.dat"

$crystal = 1000000

Config PORTA = Output

Dim A As Byte

Do

Incr A

Set PORTA.0

Waitms 600

Reset PORTA.0

Waitms 600

Loop

Until A = 10

Toggle PORTA

End

```

: For - Next

فرم کلی دستورات FOR .. NEXT بصورت زیر می باشد .

For Var = Start To End [Step Value]

برنامه

Next Var

Var بعنوان یک کانتر عمل می کند که Start مقدار اولیه آن و End مقدار پایانی است و هر دو می توانند یک ثابت عددی یا

متغیر عددی باشند . Value مقدار عددی Step را نشان می دهد که می تواند مثبت یا منفی باشد . وجود نام Var بعد از Next

الزامی نیست . مثال :

```

$regfile = "m16def.dat"

$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

Dim A As Byte

Dim B As Byte

Dim C As Integer

For A = 1 To 10 Step 2

Locate 1 , 1

Lcd A

```

Next A

For C = 9 To -5 Step -1

Locate 1 , 6

Lcd C

Next For B = 1 To 10

Locate 2 , 1

Lcd B

Next

End

: While - Wend

فرم کلی این دستور به شکل زیر است:

WHILE Condition

برنامه

WEND

دستورالعمل While - Wend تشکیل یک حلقه تکرار می دهد که تکرار این حلقه تا زمانی ادامه می یابد که عبارت بکاربرده

شده نتیجه را FALSE کند و یا مقدار صفر بگیرد . دستورالعمل While بصورت ورود به حلقه به شرط می باشد , بنابراین While

ممکن است در حالتیابی اصلا اجرا نشود .

برنامه تا وقتی که حاصل Condition صفر یا False نشده است تکرار خواهد شد . مثال :

\$regfile = "m16def.dat"

\$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

Dim A As Byte

A = 1

While A < 10

Locate 1 , 1

Lcd A

Incr A

Waitms 600

Wend

End

: if

فرم کلی این دستور به شکل زیر است:

If A = 1 Then :

...

Elseif A = 2 Then :

..

Elseif B1 > A Then :

...

Else :

...

End If

در صورتی که شرط اول برقرار باشد (A=1) دستورات زیر If A = 1 Then تا Elseif A = 2 Then اجرا میشود ، در صورتی که

شرط اول برقرار نباشد و شرط دوم برقرار باشد (A=2) دستورات بین Elseif A = 2 Then تا Elseif B1 > A Then اجرا میشود ، در

صورتی که و اگر هیچ کدام از شرط ها برقرار نباشد دستورات بین Else تا End If اجرا میشود. این دستور میتواند هر پیزی

را چک کند (متغیر ها ، واحد های حافظه ، پین ها ، پورت ها و...) . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim A As Byte
```

```
Do
```

```
Incr A : Wait 1
```

```
If A = 1 Then :
```

```
Locate 1 , 1 : Lcd "CLAUSE1 true" : Locate 2 , 1 : Lcd "a=1"
```

```
Elseif A = 2 Then :
```

```
Locate 1 , 1 : Lcd "CLAUSE2 true" : Locate 2 , 1 : Lcd "a=2"
```

```
Elseif A = 3 Then :
```

```
Locate 1 , 1 : Lcd "CLAUSE3 true" : Locate 2 , 1 : Lcd "a=3"
```

```
Elseif A = 4 Then :
```

```
Locate 1 , 1 : Lcd "CLAUSE4 true" : Locate 2 , 1 : Lcd "a=4"
```

```
Elseif A = 5 Then :
```

```
Locate 1 , 1 : Lcd "CLAUSE5 true" : Locate 2 , 1 : Lcd "a=5"
```

```
Elseif A = 6 Then :
```

```
Locate 1 , 1 : Lcd "CLAUSE6 true" : Locate 2 , 1: Lcd "a=6"
```

```
Else :
```

```
Locate 1 , 1 : Lcd "CLAUSE4 false" : Locate 2 , 1 : Lcd "a>6"
```

```
End If
```

```
Loop
```

```
End
```

در مثال بالا ، هر یک ثانیه یک واحد به متغیر A افزوده میشود ، هنگامیکه مقدار آن 1 است روی سطر اول LCD عبارت "CLAUSE1 true" و روی سطر دوم LCD مقدار A نوشته میشود ، هنگامیکه مقدار آن 2 است روی سطر اول LCD عبارت "CLAUSE2 true" و روی سطر دوم LCD مقدار A نوشته میشود ، هنگامی مقدار آن بزرگتر از 6 شد ، چون مقدار A در هیچ کدام از شرط ها صدق نمی کند ، دستور بین Else و End If اجرا میشود . شما میتوانید از این دستور به فرم های مختلف ، یک شرطی یا چند شرطی استفاده کنید با این دستور در ادامه بیشتر آشنا میشویم.

```
: #if
```

این دستور شبیه دستور If است با این تفاوت که در با این دستور میتوان شروط مربوط به کارکترها را اجرا کرد ، این دستور به فرم کلی زیر است:

```
#if Condition
```

```
case 1
```

```
#else
```

```
Case 2
```

```
#ENDIF
```

در صورتی که شرط درست باشد case 1 اجرا میشود و در صورت نا درست بودن case 2 اجرا میشود.

مثال:

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
# if Varexist("S")
```

```
Dim A As Byte
```

```
# else
```



```
Dim S As Byte
```

```
#endif
```

```
End
```

: Select Case

این دستور تقریباً مشابه دستور if است و به فرم زیر می باشد :

```
Select Case Var
```

```
Case Test1 : Statements1
```

```
[Case Test2 : Statements2 ]
```

```
....
```

```
Case Else : Statements3
```

```
End Select
```

در این دستور یک متغیر چک میشود ، در صورتی که مقدار آن با شرط Test1 برابر باشد Statements1 اجرا میشود، در

صورتیکه مقدار آن با Test2 برابر باشد Statements2 اجرا میشود و در صورتی که مقدار آن با هیچ یک از شروط برابر نباشد

Statements3 اجرا میشود . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim A As Byte
```

```
Do
```

```
Incr A : Wait 1
```

```
Select Case A
```

```
Case 1:
```

```
Locate 1 , 1 : Lcd "CLAUSE true" : Locate 2 , 1 : Lcd "a=1"
```

```
Case 2:
```

```
Locate 1 , 1 : Lcd "CLAUSE true" : Locate 2 , 1 : Lcd "a=2"
```

```
Case 3 To 5:
```

```
Locate 1 , 1 : Lcd "CLAUSE true" : Locate 2 , 1 : Lcd "a=3-5"
```

```
Case Else :
```

```
Locate 1 , 1 : Lcd "CLAUSE false" : Locate 2 , 1 : Lcd "a>6"
```

```
End Select
```

Loop

End

در مثال بالا هر یک ثانیه یک واحد به متغیر A افزوده میشود، در صورتی که مقدار آن یک باشد شرط اول اجرا شده و بر روی lcd عبارات "CLAUSE true" و "A=1" نمایش داده میشود، در صورتی که مقدار آن دو باشد عبارت های "CLAUSE true" و "A=2" بر روی LCD نمایش داده میشود، در صورتی که مقدار آن بین 3 تا 5 (3 و 4 و 5) باشد عبارت های "CLAUSE true" و "A=3-5" بر روی LCD نمایش داده میشود،... در صورتی که هیچ کدام از شرط ها برقرار نباشد دستورات بین Case Else و End Select اجرا میشود.

خروج از حلقه :

با دستور Exit میتوان از تمام حلقه های بالا خارج شد، فرم دستور برای هر حلقه در زیر آورده شده است:

Exit For

Exit Do

Exit While

Exit Sub

Exit Function

مورد اول برای خروج از حلقه ی For - Next و مورد دوم برای خروج از حلقه ی Do - Loop و مورد سوم برای خروج از حلقه ی While- Wend و مورد چهارم برای خروج از زیر برنامه و مورد آخر برای خروج از تابع است (دو مورد آخر در بخش های بعدی مورد بررسی قرار میگیرند). مثال :

```
$regfile = "m16def.dat" : $crystal = 1000000 : Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Dim A As Byte
```

```
For A = 1 To 100
```

```
Locate 1 , 1 : Lcd "Exit FOR-NEXT" : Locate 2 , 1 : Lcd "when A=4 a is " : Lcd A
```

```
If A = 4 Then :
```

```
Exit For
```

```
End If
```

```
Wait 1
```

```
Next A = 1
```

```
Do
```

```
Locate 1 , 1 : Lcd "Exit do-loop" : Locate 2 , 1 : Lcd "when A=5 a is " : Lcd A : Wait 1
```

```
If A = 5 Then :
```

Exit Do

End If

Incr A

Loop A = 1

While A < 6

Locate 1 , 1 : Lcd "Exit While-Wend" : Locate 2 , 1 : Lcd "when A=6 a is " : Lcd A: Incr A : Wait 1

If A = 6 Then :

Exit While

End If

Wend

End

دستورات اجرایی :

برای اینکه بتوانید با کامپایلر بسکام راحت تر کار کنید تعدادی دستور به دستورات بیسیک برای AVR اضافه شده است ، این دستورات کدی را برای ریختن روی میکرو تولید نمیکند و فقط جایگزین یک دستور میشود یا آن را خلاصه میکند ، استفاده از این دستورات کاملاً اختیاری است .

مثلاً شما میتوانید با دستور \$prog که در زیر توضیح داده میشود ، فیوز بیت دلخواه خود را پروگرام کنید و.... این دستورات با علامت \$ شروع میشوند در زیر آنها را توضیح میدهیم: (بعضی از دستورات همچون دستور \$baud یا .. در بخش مربوطه گفته میشود).

دستورات : (دونقطه) (شیفت + ک)، (کما) (شیفت + و) ; (ویرگول)(حرف ک) _ (خط فاصله) (شیفت + منفی) ' (نقل قول تکی)

: (شیفت + گ) (البته در حالت کیبرد انگیزی) : (دونقطه) (شیفت + ک) : با استفاده از این دستور شما میتوانید چندین دستور را پشت سر هم بنویسید .

مثال:

```
Incr A : Wait 1 : Locate 1 , 1 : Lcd A : Locate 2 , 1 : Lcd B : Wait 2 : Cls : Incr B
```

, (کما) (شیفت + و) : با این دستور میتوانید چندین متغیر را در یک خط معرفی کنید (dim فقط برای متغیر اول آورده میشود) .

```
Dim A As Byte , B As Integer , C As Word , Q As Bit
```

: (ویرگول) (حرف ک) : با این دستور میتوانید چندین متغیر یا کارکتر را در یک خط بنویسید ، کارکتری بعدی از اولین فضای خالی بعد از کارکتر قبلی نمایش داده میشود .

```
Lcd "a" ; A ; "b" ; B
```

_ (خط فاصله) (شیفت + منفی) با این دستور میتوان یک خط طولانی را نصف کرد .

```
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 _
```

```
, Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
```

' (نقل قول تکی) (شیفت + گ) : با گذاشتن این دستور در برنامه ، کلیه حروف واعدادی که بعد از این دستور میایند، کامپایل نمیشوند ، شما میتوانید بعد از این دستور توضیحاتی را به برنامه اضافه کنید ، شما همچنین میتوانید به جای این علامت کلمه Rem را به کار ببرید.

مثال: (یادشتهای به رنگ سبز در میاید.)

```
Start Adc          'roshan shodan adc
```

برای درک مطالب مثال زیر را شبیه سازی کنید:

```
$regfile = "m16def.dat" : $crystal = 1000000 : Config Lcd = 16 * 2
```

```
_ , Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6
```

Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

' rah andazi lcd

Dim A As Byte , B As Integer , C As Word , D As Byte

A = 1 : B = 2 : C = 5 : D = 7

Locate 1 , 1 : Lcd A ; " " ; B ; " " ; C ; " " ; D Locate 2 , 1 : Lcd "a" ; A ; "b" ; B

End 'payan brname

: \$asm

با این دستور شما میتوانید در بین دستورات بیسیک ، دستورات اسمبلی به کار ببرید دستورات اسمبلی بین \$asm و \$end Asm

قرار میگیرند . (برای دیدن دستورات اسمبلی که برای AVR مورد استفاده قرار میگیرد در HELP بسکام گزینه Assembler

mnemonics را جستجو کنید).

مثال:

\$regfile = "m16def.dat"

\$crystal = 1000000

\$asm

LDI R24,1 ; load register R24 with the constant 1

ST X,R24 ; store 1 into variable c

\$end Asm

End

<دستور \$dbg :

با نوشتن این دستور و کامپایل و پروگرام کردن آن در میکرو ، شما میتوانید توسط پروگرامر سریال و دستور بعدی برنامه

داخل میکرو را اشکال زدایی کنید ، تنها کافی است این دستور را در برنامه خود به کار ببرید.

\$dbg

<دستور Debug :

این دستور به فرم کلی زیر است:

DEBUG ON | OFF | var

با این دستور میتوان تغییرات متغیر var را مشاهده کرد ، هنگامی که از این دستور استفاده میکنید ، متغیر var توسط پروگرامر سریال به پورت کام فرستاده میشود و شما می توانید آن را در محیط شبیه ساز مشاهده کنید ، On و Off برای شروع و پایان ارسال متغیر هستند .

مثال:

```
$regfile = "m16def.dat" : $crystal = 1000000 : Config Lcd = 16 * 2 : $dbg
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim A As Byte
Do
Incr A : Wait 1 : Locate 1 , 1 : Lcd A
Debug On A
Loop
End
```

در مثال بالا متغیر A بهد از هر باز افزوده شدن به پورت کام فرستاده میشود .

: \$default

با این دستور میتوانید متغیر های تعریف شده را در حافظه های مختلف به صورت خود کار ذخیره کنید (ذخیره سازی ار اولین خانه خالی حافظه شروع میشود) ، این دستور به فرم کلی زیر است:

\$default = Var

Var : میتواند یکی از گزینه های Eram , Xram , Sram باشد.

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
$default Eram
Dim A As Byte , B As Byte
$default Sram
Dim D As Byte
A = 1 : B = 1
D = A * B
End
```

در مثال بالا متغیر های A و B در حافظه ی Eeprom و متغیر D در حافظه Sram ذخیره میشود (توجه داشته باشید که میکرو شما باید دارای حافظه های انتخاب شده باشد)

: \$external

با این دستور شما میتوانید از لایبریهای اسمبلی در بسکام استفاده کنید ، با دستور \$asm شما به تعدادی از دستورات دسترسی دارید ، با این دستور میتوانید لایبری های زبان اسمبلی را در پوشه lib موجود در محل نصب بسکام کپی کرده و در برنامه فراخوانی کنید.

مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
$external Test
Declare Sub Test( byval X Asbyte , Y Asbyte)
Dim Z As Byte
Call Test(1 , Z)
End
```

دستور \$EEPLEAVE :

با قرار دادن این دستور در برنامه کامپایلر فایل های EEP موجود در محل ذخیره ی برنامه را تغییر نمیدهد . فایل های EEP هنگام ذخیره ی متغیر در حافظه ی EEPROM ایجاد میشود . این دستور برای زمانی مناسب است که فایل EEP قبلا یا توسط نرم افزاری دیگر ایجاد شده باشد .

دستور \$map :

با نوشتن این دستور در برنامه ، گزارشی به نام map به انتهای فایل report افزوده میشود ، این متن شماره خط های که در آنها کد تولید شده است را نمایش میدهد (برای دیدن فایل گزارش به آدرس : show result > Program بروید) .

: \$include

با این دستور شما میتوانید برنامه دیگری را وارد برنامه خود کنید ، برنامه دیگر باید در محل ذخیره برنامه اصلی موجود باشد ، این دستور به فرم کلی زیر است:


```
$include "file"
```

File : نام برنامه ای است که میخواهید آن را وارد کنید ، نام باید کامل و با پسوند باشد

مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
$include "123.bas"
```

```
End
```

برنامه ای که به نام 123 ذخیره شده در زیر آورده شده است :

```
Locate 1 , 1
```

```
Lcd "12365"
```

```
Locate 2 , 1
```

```
Lcd "qwert"
```

: \$programmer

با این دستور شما میتوانید نوع پروگرامر مورد استفاده را ، در برنامه تعیین کنید ، این دستور به فرم کلی زیر است:

```
$programmer = Number
```

Number : یکی از اعداد جدول زیر است :

Value	Programmer
0	AVR-ISP programmer(old AN 910)
1	STK200/STK300
2	PG302
3	External programmer
4	Sample Electronics
5	Eddie Mc Mullen
6	KITSRUS K122
7	STK500
8	Universal MCS Interface
9	STK500 extended
10	Lawicel Bootloader
11	MCS USB
12	USB-ISP I
13	MCS Bootloader

همچنین میتوانید از مسیر زیر در برنامه بسکام نوع پروگرامر را تغییر دهید :

Options > Programmer

برای اطلاعات بیشتر در مورد پروگرامر ها به قسمت ضمیمه ها مراجعه کنید .

: \$sim

هنگامی که این دستور را در برنامه به کار ببرید، کلیه دستورات تاخیر غیر فعال میشوند ، این دستور هنگامی که از شبیه ساز داخلی بسکام استفاده میکنید کار برد دارد ، هنگامی که میخواهید برنامه را روی میکرو بریزید یا آن را با پروتوس شبیه سازی کنید این دستور را پاک کنید.

: \$lib

با این دستور شما میتوانید از دیگر لایبری های که برای بسکام نوشته شده است استفاده کنید (مثلا لایبری lcd گرافیکی) این دستور به فرم کلی زیر است :

\$LIB "libname"

libname1: نام لایبری میباشد که در محل ذخیره برنامه وجود دارد ، شما همچنین میتوانید آن را در پوشه LIB موجود در محل نصب بسکام کپی کنید ، در این صورت دیگر به این دستور نیازی نیست .

\$lib"mylib.libx"

: \$nocompile

با نوشتن این دستور در برنامه ، برنامه کامپایل نمیشود (کد هگزی تولید نمیشود ، برنامه چک نمیشود ، برنامه اتوماتیک ذخیره نمیشود) :

\$nocompile

: Popall و Pushall

دستور Popall باعث میشود که بعد از ریست شدن میکرو همه رجیستر های حافظه به حالت پیشفرض برگردند و استفاده از دستور Pushall باعث میشود تا رجیستر ها ذخیره شود . (در حالت عادی بعد از ریست شدن میکرو کلیه رجیستر ها به حالت پیش فرض برمیگردند.) برای استفاده از این دستورات آن ها را در برنامه خود تایپ کنید.

زیر برنامه ها و فراخوانی توابع :

: Declare Function

از این دستور برای معرفی تابع در ابتدای برنامه استفاده می شود . زمانی که بخواهیم تابعی را معرفی کنیم بایستی تابع معرفی شده باشد . در صورت استفاده از تابع می بایستی یک داده برگردانده شود .

Declare Function Test [([Byref / Byval] Var As Type1)] As Type2

Test نام تابع موردنظر است . انتقال داده بصورت Byval باعث می شود که یک کپی از متغیر به تابع فرستاده شود و در محتوای آن هیچ تغییری ایجاد نشود . ولی در حالت Byref آدرس متغیر ارسال و تغییرات در آن اثر می گذارد و داده برگشتی در صورت انجام عملیات بر روی آن با مقدار اولیه خود برابر نخواهد بود . در صورت عدم استفاده از گزینه [Byref/Byval] بصورت پیش فرض داده بصورت Byref فرستاده می شود . Type1 نوع داده ارسال شده و Type2 نوع داده برگشتی است . که هر دو می توانند داده نوع String , Long , word , Integer , Byte باشند .

مثال :

در مثال زیر بصورت Byval فرستاده شده است بنابراین یک کپی از مقدار ا به زیر تابع فرستاده می شود و هیچ تغییری در محتوای آن ایجاد نمی شود . S بصورت Byref فرستاده می شود و تغییر در آن صورت می گیرد . فراخوانی تابع Myfunction با K و Z از نوع داده Integer و String است و مقدار برگشتی از نوع Integer است که در متغیر T قرار می گیرد . شما می توانید در محدوده تابع یک متغیر محلی تعریف کنید .

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Declare Function Myfunction ( Byval I As Integer , S As String ) As Integer
Dim K As Integer , Z As String *10, T As Integer
K = 5 : Z = "123 " : T = Myfunction(k , Z)
LocAte 1 , 1
Lcd T '25
Locate 1 , 7
Lcd Z 'Bascom
```

Locate 2 , 1

Lcd K '5

End

Function Myfunction (Byval I As Integer , S As String) As Integer

local P As Integer

Functions:

P = I * 5

I = 5

S = "Bascom"

T = P

Myfunction = T

End Function

معرفی زیر برنامه DECLARE SUB :

از این دستور برای معرفی زیر برنامه استفاده می شود . زیر برنامه ای که قصد فراخوانی آن را داریم بایستی در ابتدای برنامه یا حداقل قبل از فراخوانی آن معرفی شده باشد .

DECLARE SUB TEST[([BYREF/BYVAL] var as type)]

زیر برنامه برخلاف تابع مقداری بر نمی گرداند . در زمان ارسال داده بصورت BYREF آدرس داده به زیر برنامه فرستاده می شود و در محتوای آن تغییر ایجاد می شود . ولی در حالت BYVAL یک کپی از داده فرستاده می شود و به هیچ وجه در محتوای آن تغییری ایجاد نمی شود . TEST نام زیربرنامه و VAR نام متغیر ارسالی به زیر برنامه و TYPE نوع آن است که می تواند داده نوع BYTE , INTEGER, WORD ,STRING باشند .

برای نوشتن زیر برنامه ابتدا نام آنرا توسط دستور زیر تعریف کرده و سپس شروع به نوشتن زیربرنامه می کنیم .

SUB Name [(var1)]

NAME نام زیربرنامه که باید توسط دستور Declare معرفی شده باشد و با دستور End Sub پایان می یابد .

\$regfile = "m16def.dat"

\$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

Dim A As Byte , B1 As Byte , C As Byte

Declare Sub Test (A As Byte)

```

A = 1 : B1 = 2 : C = 3

Lcd A ; B1 ; C '123 will print

Call Test (B1)

End

Sub Test(a As Byte )

Locate 2 , 1 Lcd A ; B1 ; C '123 will print

End Sub

```

تابع فراخوانی CALL :

توسط این دستور زیر برنامه یا تابعی را فراخوانی می کنیم .

```
CALL TEST( VAR1 , VAR2,...)
```

VAR1 , VAR2 متغیرهایی که به زیر برنامه انتقال می یابند , هستند . می توان زیر برنامه را بصورت زیر نیز فراخوانی کرد .

```
TEST VAR1 , VAR2
```

لازم بتذکر است که نام زیر برنامه قبل از فراخوانی آن , باید توسط دستور Declare فراخوانی شود. اگر بخواهیم عدد ثابت را

به زیر برنامه انتقال دهیم بایستی حتما با آرگومان BYVAL آن را انتقال دهیم .

```

$regfile = "m16def.dat"

$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

Dim A As Byte , B As Byte

Declare Sub Test ( B1 As Byte , Byval B2 As Byte )

A = 65

Call Test ( A , 5) Test A , 5

Locate 1 , 1

Lcd A ' will print A = 10

End Sub

Test(b1 As Byte , Byval B2 As Byte )

B1 = 10

B2 = 15

Locate 1 , 8

Lcd B1

Locate 2 , 1

Lcd B2

```

End Sub

بکارگیری متغیر محلی یا LOCAL :

از این دستور برای تعریف متغیر محلی در زیر برنامه استفاده می کنیم .

LOCAL VAR As Type

VAR نام متغیر و type نوع داده است که می توانند STRING , WORD , INTEGER , BYTE , SINGLE , LONG باشند

نوع داده های ERAM , SRAM , XRAM و آرایه ها نمی توانند محلی تعریف شوند.

یک متغیر محلی یک متغیر موقت است که فقط در هنگام فراخوانی زیر برنامه مربوطه برای آن فضا در نظر گرفته می شود و

با برگشت از زیر برنامه عمر متغیر (LIFE TIME) به اتمام می رسد .

تذکر متغیرهای بیتی نمی توانند بصورت محلی تعریف شوند .

مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
```

```
Declare Sub Test2
```

```
Do
```

```
Call test2
```

```
Loop
```

```
End
```

```
Sub Test2
```

```
Local A As Byte
```

```
Incr A
```

```
Lcd A
```

```
End Sub
```

دستورات ریاضی و محاسباتی و تبدیل متغیر های ریاضی

از عملگرهای ریاضی زیر می توان در محیط BASCOM استفاده کرد و عملیات ریاضی را انجام داد.

دستور	نام دستور	دستور	نام دستور
*	ضرب	=>	کوچکتر یا مساوی
/	تقسیم	<=	بزرگتر یا مساوی
+	جمع	<>	مخالف
-	تفریق	NOT	نات منطقی
.	ممیز	AND	اند منطقی
<	بزرگتر از	OR	ار منطقی
>	کوچکتر از	XOR	ایکس ار منطقی
=	مساوی	^	توان

مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim Q As Byte , W As Byte , E As Byte , R As Byte , T As Byte
Q = 2
W = Q + 3
E = W * Q
R = E ^ E
T = E - r
Lcd T 't=10
End
```

دستور SQR:

```
var = SQR( source )
```

این دستور جدر (ریشه دوم) متغیر یا عدد ثابت source را محاسبه کرده و نتیجه را در متغیر VAR قرار میدهد .

مثال:

```
$regfile = "m16def.dat"
```

```

$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single , A As Byte
Cls X = Sqr (9)
Locate 1 , 1 'x=3 : Lcd X
X = X * 27
A = Sqr (x)
Locate 2 , 1 : Lcd A ' A=9
End

```

دستور SGN :

```
var = SGN( x )
```

این دستور علامت یک متغیر شناور را نشان میدهد ، در صورتی که متغیر منفی باشد ($X < 0$) مقدار VAR برابر با -1 و در

صورت مثبت بودن متغیر ($X > 1$) مقدار VAR برابر با 1 میشود .

مثال:

```

$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single , A As Single
Cls X = -9
X = Sgn (x)
Locate 1 , 1 'x=0 : Lcd X
X = 27
A = Sgn (x)
Locate 2 , 1 : Lcd A ' A=1
End 'End

```

دستور POWER :

```
var = POWER( source, raise )
```


این دستور فرم دیگر دستور $^{\wedge}$ (توان) است و حاصل source به توان raise را محاسبه کرده و حاصل را در متغیر var قرار میدهد .

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single , A As Single , B As Single
Cls
A = 2.3 : X = 10
B = A ^ X
Locate 1 , 1 : Lcd B
B = Power (a , X )
Locate 2 , 1 : Lcd B
End
```

دستور ABS :

VAR =Abs (VAR2)

این دستور به معنای ریاضی $VAR = |VAR2|$ (قدر مطلق) است (اعداد منفی را به مثبت تبدیل میکند).

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim A As Integer , C As Integer
A = -100
C = Abs (a) 'c=|a|
Lcd C 'C= 100
End
```

دستور EXP :

Target = Exp (source)

Target برابر با e بتوان source است . Target متغیری از نوع داده SINGLE است .

مثال:

```
$ regfile = "m16def.dat"
$crystal = 12000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single
X = Exp ( 1.1)
Locate 1 , 1 : Lcd X 'x = 3.004166124
X = 1.1
X = Exp (x)
Locate 2 , 1 : Lcd X 'x = 3.004166124
End
```

دستور LOG :

این دستور لگاریتم طبیعی یک داده از نوع SINGLE را برمی گرداند .

Target = Log (source)

لگاریتم متغیر یا ثابت source از نوع داده single گرفته می شود . و در متغیر target قرار می گیرد .

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single
X = Log (100)
Locate 1 , 1 : Lcd X 'x = 4.6051
X = Log (1000)
Locate 2 , 1 : Lcd X 'x = 6.9077
End
```

<دستور LOG10 :

Target = LOG10(source)

این دستور لگاریتم source که از نوع single یا double می باشد را در مبنای 10 محاسبه کرده و نتیجه را در متغیر Target که از نوع جنس single یا double می باشد قرار می دهد.

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single
X = Log10 (100)
Locate 1 , 1 : Lcd X 'x = 2
X = Log10 (1000)
Locate 2 , 1 : Lcd X 'x = 3
End
```

دستور RND :

این دستور یک عدد را تصادفی برمی گرداند .

VAR= RND (limit)

عدد تصادفی بین 0 و limit بدست آمده و در متغیر var قرار می گیرد . با هربار استفاده از این دستور عدد مثبت تصادفی دیگری بدست خواهد آمد .

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Byte
Do
X = Rnd (100)
Lcd X
Waitms 500
Loop
End
```

دستور Frac:

var = FRAC(single)

این دستور مقدار اعشاری یک متغیر از جنس single را جدا میکند و در یک متغیر single دیگر (VAR) میریزد .

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single , A As Single
X = 57.35456
Locate 1 , 1
Lcd Frac (x) 'x = 35456
X = X * 3
A = Frac (x)
Locate 2 , 1 : Lcd A ' A = 0636749
End 'End
$sim
```

دستور INT:

var = INT(source)

این دستور مقدار صحیح یک متغیر از جنس single یا double را محاسبه میکند و حاصل را در متغیر VAR قرار میدهد (محدود متغیر VAR باید با محدوده جواب مناسب باشد) .

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single , A As Byte
Cls X = 57.35456
Locate 1 , 1
Lcd Int (x) 'x =57
X = X * 3
A = Int (x)
```

```
Locate 2 , 1
Lcd A ' A = 172
End
```

دستورات محاسبه نسبت های مثلثاتی:

```
var=sin\cos\tan(source)
```

این دستور سینوس یا کسینوس یا تانژانت ثابت یا متغیر source را در متغیر var از نوع SINGLE قرار می دهد . تمام دستورات مثلثاتی با رادیان کار می کنند و ورودی این دستور بایستی رادیان باشد .

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single
X = Cos (90)
Locate 1 , 1 : Lcd X
X = Sin (90)
Locate 1 , 8 : Lcd X
X = Tan (90)
Locate 2 , 1 : Lcd X
End
```

دستورات محاسبه نسبتهای مثلثاتی هایپربولیک :

```
Var = COSH\sinh\tanh( source)
```

این دستور کسینوس یا سینوس یا تانژانت هایپربولیک ثابت یا متغیر source را در متغیر var از نوع SINGLE قرار می دهد . تمام دستورات مثلثاتی با رادیان کار می کنند و ورودی این دستور بایستی رادیان باشد .

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single
Dim Q As Single
Q = .512
```

```

X = Cosh (q)

Locate 1 , 1 : Lcd X

X = Sinh (q)

Locate 1 , 8 : Lcd X

X = Tanh (q)

Locate 2 , 1 : Lcd X

End

```

معکوس توابع مثلثاتی (Arc):

Var = A sin\Acos\ATN(source)

این دستور آرک سینوس یا کسینوس یا تانژانت ثابت یا متغیر source را در متغیر var از نوع SINGLE قرار می دهد . تمام دستورات مثلثاتی با رادیان کار می کنند و ورودی این دستور بایستی رادیان باشد .

مثال:

```

$regfile = "m16def.dat"

$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

Dim X As Single

Dim Q As Single

Q = .512

X = Acos (q)

Locate 1 , 1 : Lcd X

X = Asin (q)

Locate 1 , 8 : Lcd X

X = Atn (q)

Locate 2 , 1 : Lcd X

End

```

دستور RAD2DEG:

Var =RAD2DEG(single)

برای تبدیل رادیان به درجه از این دستور استفاده می شود .

رادیان single به درجه تبدیل می شود و در متغیر VAR از نوع داده SINGLE قرار می گیرد

دستور DEG2RAD:

Var =DEG2RAD(single)

برای تبدیل درجه به رادیان از این دستور استفاده می شود .

زاویه single به رادیان تبدیل می شود و در متغیر VAR از نوع داده SINGLE قرار می گیرد .

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single
Dim Q As Single
Q = 180
X = Deg2rad (q)
Locate 1 , 1 : Lcd X
Q = Rad2deg (x)
Locate 2 , 1 : Lcd Q
End
```

دستور ROUND :

Var =ROUND(x)

متغیر یا داده X از نوع SINGLE روند شده و در متغیر VAR از نوع داده SINGLE قرار می گیرد .

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Dim X As Single
Dim Q As Single
X = 2.8 : Q = 5.2
Locate 1 , 1 : Lcd Round (x)
Locate 2 , 1 : Lcd Round (q)
End
```

دستورات min و max :

با این دو دستور میتوان کمترین و بیشتر مقدار یک متغیر آرایه ای از جنس word یا BYTE را بدست آورد ، این دو دستور به فرم کلی زیر هستند:

```
var1 = MIN(var2)
```

```
var1 = MAX(var2)
```

var(2): نام متغیر آرایه است و var یک متغیر دیگری است که نتیجه در آن ریخته میشود.

مثال:

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0
Do
Locate 1 , 1 : Lcd Max (a(1))
B(1) = Min (b(2))
Locate 2 , 1 : Lcd B(1)
Loop
End
```

دستور FIX :

با این دستور کوچکترین عدد صحیح نزدیک به داده اعشاری از نوع بدست میاید (جزء صحیح). این دستور به فرم کلی زیر است:

```
var = FIX( x )
```

x میتواند یک عدد اعشاری یا یک داده از نوع single باشد ، با اجرای دستور کوچکترین عدد نزدیک به داده در متغیر var ریخته میشود ، مثلا کوچکترین عدد صحیح نزدیک به 2.2 ، 2 است .

مثال:

```
$regfile = "m16def.dat" : $crystal = 12000000 : Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 ,
E = PORTC.2 , Rs = PORTC.0
Dim A As Single , S As Single
Cls
A = -45.56
S = Fix (25.9 )
A = Fix (a )
Locate 1 , 1: Lcd S Locate 2 , 1: Lcd A
End
```


توابع تبدیل کدها و متغیر ها به یکدیگر

دستور HEX :

Var = Hex (x)

این دستور یک داده از نوع BYTE,INTEGER , WORD , LONG را به مقدار هگزادسیمال تبدیل می کند .

مقدار HEX متغیر یا ثابت X در متغیر VAR جای می گیرد .

دستور Bin :

Var = Bin(x)

این دستور یک داده از نوع BYTE,INTEGER , WORD , LONG را به مقدار باینری تبدیل می کند .

مقدار باینری متغیر یا ثابت X در متغیر VAR جای می گیرد .

دستور STR :

Var = STR (X)

با این دستور می توان یک متغیر عددی (X) را به رشته (VAR) تبدیل کرد . در صورتی که متغیر عددی در فرمت اعشار

باشد ، علامت اعشار . به ، تبدیل خواهد شد ، با دستور زیر میتوانید تعیین کنید که علاومت اعشار در چه فرمتی باشد :

CONFIG DP= "dp"

در این دستور dp میتواند . (نقطه) یا , (ممیز) باشد . مثال :

CONFIG DP=","

Dim s as single

S=1234.56

print s '1234,56

مثال :

CONFIG DP="."

Dim s as single

S=1234.56

print s '1234.56

دستور VAL :

Var = VAL (S)

با این دستور می توان یک رشته (S) را به متغیر عددی (VAR) تبدیل کرد .

دستور HEXVAL :

Var = HexVal (x)

این دستور یک داده هگزادسیمال را به مقدار عددی تبدیل می کند .

مقدار عددی داده هگزادسیمال X که می تواند LONG , WORD , INTEGER , BYTE باشد در متغیر VAR جای می گیرد

دستور Binval :

var = Binval (x)

این دستور یک داده باینری را به مقدار عددی تبدیل می کند .

مقدار عددی داده باینری X که می تواند LONG , WORD , INTEGER , BYTE باشد در متغیر VAR جای می گیرد

دستور BCD و MAKEBCD :

Var1 = MAKEBCD (Var2)

این دستور متغیر یا ثابت var2 را تبدیل به مقدار BCD اش می کند و در متغیر var1 جای می دهد .

PRINT BCD(var)

LCD BCD(var)

با دو دستور بالا میتوان مقدار bcd یک متغیر را مستقیماً روی lcd نمایش داد یا آن را به پورت سریال فرستاد (کار پورت

سریال در بخش های بعدی آمده است)

دستور MAKEDEC :

Var1 = MAKEDEC (Var2)

برای تبدیل یک داده BCD نوع BYTE , WORD , INTEGER به مقدار DECIMAL از این دستور استفاده می شود . مقدار دسیمال

متغیر یا ثابت var2 در متغیر var1 جای می گیرد .

دستور MAKEINT :

Varn = MAKEINT (LSB , MSB)

این دستور دو بایت را به هم متصل می کند و یک داده نوع WORD یا INTEGER می سازد که LSB بایت کم ارزش و MSB بایت پر ارزش متغیر دو بایتی Varn را تشکیل می دهد .

$Varn = (256 * MSB) + LSB$

دستور STRING :

$Var = STRING (m, n)$

این دستور کد اسکی m را با تعداد تکرار n تبدیل به رشته کرده و در متغیر var قرار می دهد . در صورت قرار دادن $m = 0$ یک رشته بطول 255 کاراکتر تولید می شود و قرار دادن $n = 0$ قابل قبول نیست .

دستور BIN2GREY :

$Var1 = BIN2GREY (Var2)$

متغیر var2 که می تواند داده ای از نوع WORD , INTEGER , BYTE , LONG باشد به کد گری تبدیل شده و در متغیر VAR1 قرار می گیرد .

دستور GREY2BIN :

$Var1 = grey2bin (Var2)$

کد گری var2 به مقدار باینری تبدیل شده و در متغیر var1 که می تواند داده ای از نوع WORD , INTEGER , BYTE , LONG باشد قرار می گیرد .

مثال برای موارد بالا :

```
$regfile = "m16def.dat" : $crystal = 12000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 ,
E = PORTC.2 , Rs = PORTC.0
Dim A As Byte , S As String * 10
S = "ABC"
A = ASC (s)
Locate 1 , 1 : Lcd A
S = Hex (a)
Locate 1 , 8 : Lcd S
A = Hexval (a)
Locate 2 , 1 : Lcd S
A = 50 A = Makebcd (A)
Locate 2 , 8 : Lcd A
```

End

دستور BASE64DEC و BASE64ENC :

Result = BASE64DEC(source)

Result = BASE64ENC(source)

Base-64 روشی برای کدینگ داده است که در وب سرور ها و پروتکل های اینترنتی و ارتباطی جهت مخفی سازی داده میتواند از آن استفاده میشود ، در این روش داده ها در 7 بیت و در فرمت ascii فرستاده می شود و سایر دستگاه های که در شبکه وجود دارند نمیتوانند به صورت مستقیم آن را دیکود کنند .

مثال :

```
$regfile = "m16def.dat" : $crystal = 12000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 ,
E = PORTC.2 , Rs = PORTC.0
Dim S As String * 15 , Z As String * 15
S = "code is:123654 "
Z = Base64enc(s)
Print Z
Locate 1 , 1 : Lcd Z
S = Base64dec(z)
Locate 2 , 1 : Lcd S
End

$sim
```

در مثال بالا رشته ی code is:123654 در صورتی که به یکی از پورت های spi یا rs232 یا i2c یا ... ارسال شود ، با خواندن داده ی روی باس به سادگی میتوان آن را تشخیص داد . اما با استفاده از دستور Base64enc میتوان این رشته را کد کرده و به گیرنده ی مورد نظر ارسال کرد و سپس در گیرنده ان را با دستور Base64dec دیکود نمود ، در این حالت دادهای که بر روی باس قرار میگیرد Y29kZSBpczoxMjM2NTQg است که مفهومی خاصی ندارد .

فصل چهارم

در فصل های قبل با مباحث اولیه برنامه نویسی و دستورات پایه ی زبان بیسیک آشنا شدیم ، در این فصل نحوه ی کاربرد عملی دستورات برای کار با امکانات جانبی داخلی و خارجی را فرا خواهیم گرفت .

در این بخش به دلیل ماهیت زبان بیسیک و عدم نیاز به دانستن نحوه ی کار وسیله ی جانبی ، از تشریح عمل کردن آن صرف نظر شده است . در صورتی که مایلید نحوه ی کار و نوع داده ی پردازشی وسیله ی جانبی را بدانید ، میتوانید به دیتاشیت میکرو کنترلر مراجعه نمایید .

دستور debounce:

فرم کلی این دستور به شکل زیر است:

DEBOUNCE Px.y , state , label [, SUB]

توسط این دستور پین x.y چک میشود و هنگامی که مقدار آن برابر با state شد cpu به label پرش میکند . شما میتوانید با استفاده از گزینه اختیاری SUB ، در هنگام رخ داد رویداد به یک زیر برنامه پرش کنید .

به جای گزینه state میتوانید 0 یا 1 قرار دهید ، در صورتی که state صفر باشد ، هنگامی که پایه مورد نظر به گراند متصل شد به برجسب مورد نظر پرش میشود و هنگامی که state یک باشد ، هنگامی که پایه به ولتاژ 5 ولت متصل شد پرش صورت میگیرد (پینی که کلید به آن متصل است باید به عنوان ورودی تعریف شود).

CONFIG DEBOUNCE = time

توسط دستور بالا میتوانید تاخیری را در خواندن وضعیت پین X.Y بر حسب میلی ثانیه ایجاد کنید . در صورتی که از این دستور استفاده نکنید به صورت پیشفرض تاخیر 25 ثانیه در خواندن کلید ها ایجاد میشود .

مثال:

\$regfile = "m16def.dat"

\$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = PORTC.4 , Db5 = PORTC.5 , Db6 = PORTC.6 , Db7 = PORTC.7 , E = PORTC.2 , Rs = PORTC.0

Config PORTD = Input

W:

Locate 1 , 1

Lcd "pinc.0 ro 1 kon"

Debounce PIND.0 , 1 , Q

Goto W

Q:

Locate 1 , 1

Lcd "pinc.1 ro 1 kon"

Debounce PIND.1 , 1 , W

Goto Q

End

مثال :

\$regfile = "m48def.dat" ' specify the used micro

\$crystal = 4000000 ' used crystal frequency

\$baud = 19200 ' use baud rate

\$hwstack = 32 ' default use 32 for the hardware stack

\$swstack = 10 ' default use 10 for the SW stack

\$framesize = 40 ' default use 40 for the frame space

Config Debounce = 30 'when the config statement is not used a default of 25mS will be used

'Debounce Pind.0 , 1 , Pr 'try this for branching when high(1)

Debounce Pind.0 , 0 , Pr , Sub

Debounce Pind.0 , 0 , Pr , Sub

 ' ^---- label to branch to

 ' ^----- Branch when P1.0 goes low(0)

 ' ^----- Examine P1.0

'When Pind.0 goes low jump to subroutine Pr Pind.0 must go high again before it jumps again to the label Pr when Pind.0 is low

Debounce Pind.0 , 1 , Pr 'no branch

Debounce Pind.0 , 1 , Pr 'will result in a return without gosub

End

Pr:

Print "PIND.0 was/is low"

Return

دستور PULSEOUT:

فرم کلی این دستور به شکل زیر است:

PULSEOUT PORT , PIN , PERIOD

با این دستور میتوان یک پالس بر روی پورت PORT و پایه PIN با زمان تناوب PERIOD (بر حسب میکرو ثانیه) ، ایجاد کرد.

توجه داشته باشید که :

✓ پایه ای که پالس بر روی آن ایجاد میشود باید به عنوان خروجی تعریف شود.

✓ در صورتی که از حلقه استفاده نکنید ، دستور فقط یک بار اجرا میشود .

✓ پالس ایجاد شده مربعی است، در واقع وضعیت پایه از صفر به یک یا بلعکس تغییر میکند.

مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Portc.0 = Output
W:
Pulseout Portc , 0 , 60000
Goto W
End
```

دستور PULSEIN:

توسط این دستور میتوان زمان تناوب یک پالس مربعی را اندازه گرفت ، فرم کلی این دستور به شکل زیر است:

PULSEIN var , PINX , PIN , STATE

این دستور زمان تناوب پالس اعمال شده به پورت PINX و پین PIN دلخواه را در متغیر var که باید از جنس word باشد قرار میدهد.

شما میتوانید مشخص کنید که زمان بین از صفر به یک رفتن یا از یک به صفر رفتن پالس اندازه گرفته شود ، برای حالت اول به جای STATE صفر و برای حالت دوم به جای STATE یک قرار دهید .

مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
Dim X As Word
Config Portc.0 = Input
Do
Pulsein X , Pinc , 0 , 1
Locate 1 , 1
Waitms 500
Lcd X
Loop
End
```


در مثال بالا زمان تناوب پالسی که به پین c.0 اعمال شده در هر 500 میلی ثانیه اندازه گرفته میشود و سپس بر روی lcd نمایش داده میشود. (زمان تناوب نباید از 65.535 میلی ثانیه بیشتر باشد ، این دستور از تایمر های میکرو استفاده نمیکند).

دستور : SOUND

توسط این دستور میتوان پالسی را بر روی یکی از پایه های میکرو ایجاد نمود ، فرم کلی دستور به شکل زیر است:

SOUND pin, duration, pulses

Pin نام پایه دلخواهی است که پالس از آن خارج میشود ، duration مشخص کننده تعداد پالس های خروجی است و pulses

زمان تناوب پالس برحسب میکرو ثانیه است که حداکثر مقدار آن 65535 است . مثال:

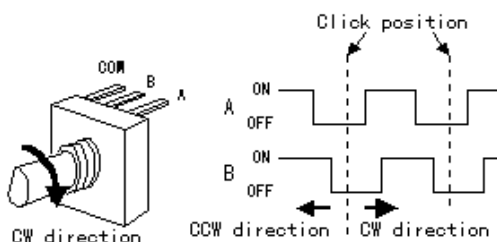
```
$regfile = "m16def.dat"
$crystal = 1000000
Config Portc.0 = Output
Do
Sound Pinc.0 , 10 , 60000
Loop
End
```

در مثال بالا پالس از پین c.0 خارج میشود ، زمان تناوب آن 60 میلی ثانیه است و 10 بار تکرار میشود . از این دستور معمولا برای راه اندازی بازر استفاده میشود.

دستور : ENCODER

ENCODER نوعی کلید دوطرفه میباشد که تصویر آن را در

شکل روبرو مشاهده میکنید:



از این قطعه در کبیردها و لوازم صوتی / تصویری برای کم و

زیاد کردن صدا و نور و ... استفاده میشود . در واقع این قطعه از

دو کلید تشکیل شده است ، هنگامی که شما سری (ولوم) را به

سمت راست میچرخانید ، کلید سمت راست (که یک پایه آن

به پایه وسط و پایه دیگر به پایه سمت راست متصل است) قطع

و وصل میشود ، و هنگامی که سری را به سمت چپ

میچرخانید کلید سمت چپ (که یک پایه آن به سر وسط و پایه دیگر به پین سمت چپ متصل است) قطع و وصل میشود ، با استفاده از دستور زیر میتوان عملیات مناسب با جهت چرخش (قطع و وصل شدن هر کلید) را انجام داد:

Var = ENCODER(pin1, pin2, LeftLabel, RightLabel , wait)

Var: یک متغیر از جنس دلخواه میباشد که به ازای پالس های فرد مقدار آن صفر و به ازای پالس های زوج مقدار آن یک است (مقدار آن حول صفر و یک تغییر میکند).

pin1: نشان دهنده پایه ای است که پین 1 (چپ یا راست) انکدر به آن متصل میشود.

pin2: نشان دهنده پایه ای است که پین 2 (چپ یا راست) انکدر به آن متصل میشود.

LeftLabel: نام برجسی است که در هنگام به چپ چرخیدن انکودر به آن پرش میشود. باز گشت از برجسب با دستور return انجام میشود.

RightLabel: نام برجسی است که در هنگام به راست چرخیدن انکودر به آن پرش میشود. باز گشت از برجسب با دستور return آن جام میشود.

Wait: در صورتی که شما به جای این کلمه 1 بگذارید cpu میکرو بر روی این دستور متوقف میشود ، هنگامی که انکودر چرخید cpu به زیر برنامه میرود و بعد از انجام دستورات موجود در زیر برنامه دوباره روی این دستور متوقف شده و منتظر میماند تا پالس بعدی اعمال شود و در صورتی که 0 قرار دهید cpu مدام در حلقه گردش میکند و اگر دستور انکودر در حلقه باشد آن را چک میکند و اگر نباشد که هیچ .
مثال :

```
$regfile = "m16def.dat"
$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
Config Portb = Input
Dim A As Byte , B As Byte , C As Word : Cls
Do
A = Encoder(pinb.0 , Pinb.1 , Q , W , 0)
Locate 1 , 1
Lcd A
Loop
End
```

Q:

Incr B

Locate 1 , 5 : Lcd B

Return

W:

Incr C

Locate 1 , 11 : Lcd C

Return

در مثال بالا پایه مشترک انکدر به گراند و پایه چپ آن به پین b.0 و پایه راست آن به پین b.1 متصل شده است. شما میتوانید به جای این قطعه از سنسور ها مادون قرمز ، کلید های فشاری و سایر قطعات مشابه استفاده نمایید .

دستور DTMFOUT :

توسط این دستور میتوانید پالس مناسب با یک عدد را تولید کرده و آن را روی خط تلفن سوار کنید. این دستور به دو شکل استفاده میشود :

DTMFOUT number, duration

با این دستور شما میتوانید فقط یک شماره را به خط تلفن ارسال کنید ، در این دستور ، شماره به جای number گذاشته میشود و duration تاخیر زمانی بین ارسال این رقم و ارقام بعدی میباشد ، شما میتوانید به جای شماره یک متغیر قرار دهید ، اما متغیر باید بین 0 تا 15 باشد . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Dim A As Byte
```

```
Do
```

```
A = 2
```

```
Dtmfout A , 50
```

```
A = 6
```

```
Dtmfout A , 50
```

```
A = 8
```

```
Dtmfout A , 50
```

```
A = 8
```

```
Dtmfout A , 50
```

```
A = 0
```

```
Dtmfout A , 50
```

Loop

End

فرم دوم:

در این حالت شما میتوانید شماره خود را در یک متغیر از جنس `string * x` که تعداد شماره است و حداکثر آن 15 است، قرار دهید، با دستور زیر شماره های موجود در متغیر با تاخیر زمانی `duration` پشت سرهم گرفته میشوند.

DTMFOUT string , duration

مثال:

`$regfile = "m16def.dat"``$crystal = 1000000``Dim A As String * 15``A = "2696580"``Do``Dtmfout A , 500``Loop``End`

در این مثال شماره 2695680 مدام به خط تلفن ارسال میشود.

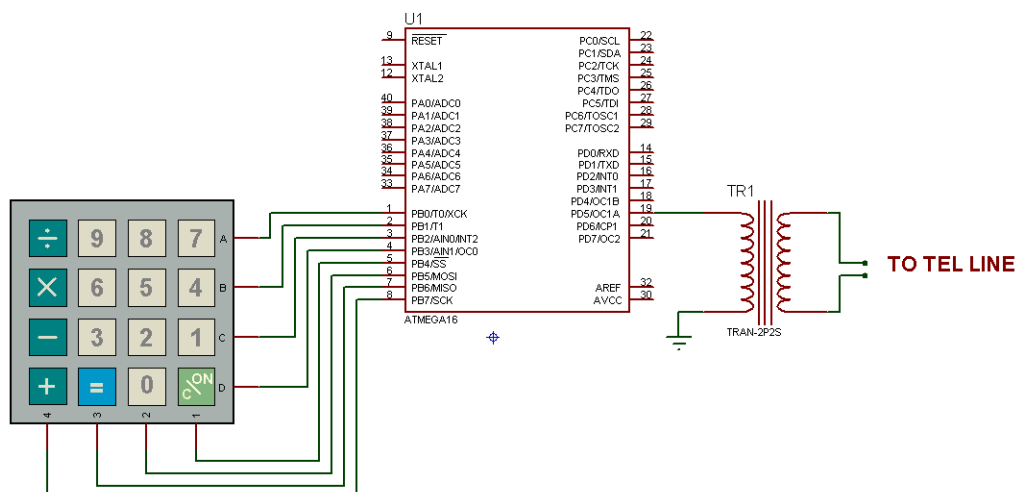
نکته:

✓ این دستور فقط با استفاده از کریستال های 4 تا 10 مگا هرتر جواب میدهد.

✓ خروجی پالس dtmf پایه oc1a و گراند است (پایه شماره 19 مگا 16).

✓ در هنگام کار با این دستور مراقب ولتاژ خط تلفن باشید. (بهتر است از اپتوکوپلر یا ترانس ایزوله استفاده کنید تا

میکرو آسیب نبیند، ترانس ایزوله نوعی ترانس است که ورودی را از خروجی جدا میکند).



استفاده از کلید و کیبرد و کی پد و ... در محیط بسکام:

کلید ها نیز همچون نمایشگر ها کاربرد گسترده ای در مدارات میکرو کنترلری دارند ، ما این قطعات برای وارد کردن وضعیت های مختلف به میکرو کنترلر استفاده میکنیم ، وضعیت های همچون موجود بودن یا نبودن یک شرط ، صفر یا یک بودن یک خروجی ، وضعیت های منطقی مختلف و

کلیه این وضعیت ها بعد از وارد شدن به میکرو میتوانند عمل کرد های مختلف را بسته به برنامه ایجاد کند ، در ادامه با نحوه ی استفاده از این قطعات آشنا خواهیم شد .

استفاده از کلید :

برای اتصال کلید به avr در محیط بسکام روش های مختلفی وجود دارد که در زیر به بیان هر یک میپردازیم .

✓ استفاده از دستور شرطی if:

کلید ها دارای دو پایه می باشد که یک پایه آن به یکی از پایه های میکرو و دیگری به vcc یا gnd متصل میشود با استفاده از دستور شرطی if میتوان فشرده شدن کلید را چک کرد ، هنگامی که کلید فشرده میشود پایه ای که کلید به آن متصل است صفر (به گراند متصل میشود) یا یک (به 5 ولت متصل میشود) میشود.

مثال : در این مثال یک سر کلید به vcc و سر دیگر به پایه 4 پورت c متصل شده است .

```
If Portc.4 = 1 Then
```

```
Set Porta.0
```

```
end IF
```

در مثال بالا ، اگر کلید فشرده شود portc.4 به vcc متصل میگردد (1 میشود) . شرط به این قرار است که اگر portc.4 یک شد porta.0 نیز یک شود در غیر این صورت porta.0 صفر بماند . (توجه داشته باشید که پایه های کلید به آنها متصل میشود باید به عنوان ورودی تعریف شود).

✓ روش دیگر استفاده از دستور Debounce است ، که قبلا آن را بررسی کردیم . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 12000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
```

```
Config Portc = Input
```

```
Debounce Pinc.0 , 1 , A
```

```
If Portc.1 = 1 Then
```

```
Lcd "qwer"
```

```
End If
```

```
End
```

```
A:
```

```
Lcd "12345"
```

```
Return
```

در برنامه بالا اگر پین C.0 یک شود روی LCD عبارت 12345 نشان داده میشود و اگر پین C.1 یک شود روی LCD عبارت QWER نشان داده میشود.

✓ یکی از روش های دیگر برای خواند کلید استفاده از دستور ON var میباشد :

```
ON var [GOTO] [GOSUB] label1 [, label2 ] [,CHECK]
```

توسط این دستور میتوان به ازای مغادیر مختلف VAR که می تواند یک متغیر ، یک عدد یا یک پورت باشد ، به یکی از LABEL های 0 تا X پرش کرد . با استفاده از دستور GOSUB میتوان به حلقه ای که در خارج از برنامه ی قرار داد پرش کرد ، باز گشت از حلقه با دستور Return انجام میشود ، دستور GOTO باعث رفتن به یک حلقه که در داخل برنامه قرار گرفته است میشود .

به جای CHECK میتوانید یک عدد قرار دهید تا در صورت برابر بودن مقدار var با آن به اولین برچسب پرش شود .

مثال :

```
$regfile = "m48def.dat" ' specify the used micro
```

```
$crystal = 4000000 ' used crystal frequency ' default use 40 for the frame spac
```

```
Config Portb = Input
```

```
Do ' Rem Note That The Starting Value Begins With 0
```

```
On Portb Gsub L0 , L1 , L2 , L3 , L4
```

```
Loop
```

```
End
```

L0:

Print "0 entered"

Return

L1:

Print "1 entered"

Return

L2:

Print "2 entered"

Return

L3:

Print "3 entered"

Return

L4:

Print "4 entered"

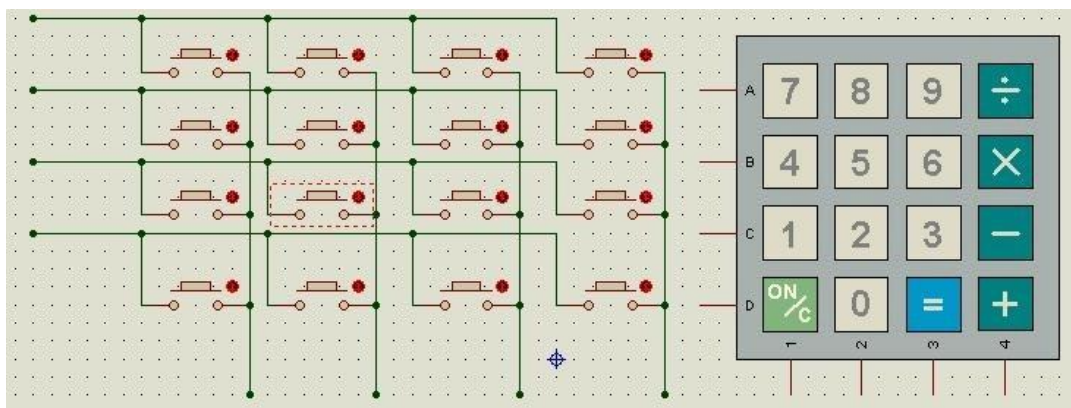
Return

✓ به جز روش های بالا ، راه های دیگری نیز برای اتصال کلید به میکرو وجود دارد که در بخش های بعدی و

در مثال های عملی با آنها آشنا خواهیم شد .

اتصال کی پد به AVR :

در برخی از پروژه ها به تعداد زیادی کلید نیاز داریم ، ما میتوانیم به هر کدام از پایه های میکرو یک کلید متصل کنیم و در برنامه ای که نوشته ایم مدام پایه های ورودی را چک کنیم ، اما این روش به دلیل اشغال شدن تعداد زیادی از پایه های میکرو و همچنین افزایش حجم برنامه ، عاقلانه به نظر نمی رسد ، روشی که برای رفع این نیاز ارائه شده است ، استفاده از صفحه کلید های ماتریسی میباشد .



با استفاده از این صفحه کلید ها میتوانیم تعداد 12 یا 16 یا 24 کلید را به میکرو کنترلر متصل کنیم در حالی که فقط بین 7 تا 10 پایه از میکرو اشغال میشود و کمتر از چند خط به برنامه اضافه می شود .

نحوه ی کار این نوع صفحه کلید به این صورت است که داده ی 0001 با تاخیر زمانی منظم بر روی سطر های صفحه کلید (که به پایه های میکرو متصل شده اند) به سمت چپ شیفت داده میشود (...>0001>1000>0100>0010>0001) در حین انجام شدن این عملیات ، ستون های صفحه کلید مدام چک میشود و اگر کلیدی فشرده شود ، با یک شدن ستون و تعیین سطری که رقم 1 روی آن قرار داده شده است ، میکرو رقمی متناظر با کلید فشرده شده بر میگردداند .

در بسکام برای استفاده از صفحه کلید ماتریسی که از این به بعد به آن کپید می گوئیم از دستور زیر استفاده میشود :

```
CONFIG KBD = PORTx , DEBOUNCE = value
```

portx ، پورتهی است که کپید به آن متصل میشود و DEBOUNCE تاخیر کلید است که بین 20 تا 255 میلی ثانیه است (هنگامی که کلید فشرده میشود ، بر اثر لرزش دست چندین بار دو کنتاکت آن به هم برخورد میکنند و در نهایت ثابت میشوند اگر از دستور DEBOUNCE استفاده نشود لرزش به منزله فشردن کلید است)

بعد از تعریف کپید در برنامه ، با استفاده از دستور زیر میتواند مقدار متناظر با کلید فشرده شده را خواند و در متغیر VAR قرار داد :

```
VAR= Getkbd()
```

VAR یک متغیر از جنس بایت است که عدد گرفته شده از کپید در آن گذاشته میشود ، در صورتی که کلیدی فشرده نشود عدد 16 در متغیر مذکور ریخته میشود . برای درک بهتر مطلب مثال زیر را ببینید :

```
$regfile = "m16def.dat"
```



```

crystal = 12000000$

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5

Config Kbd = Portb,DEBOUNCE = 50

Dim A As Byte

Q:

A = Getkbd()

If A > 15 Then

Goto Q

End If

Locate 1 , 1

Lcd A

Goto Q

End

```

تحلیل برنامه :

دو خط اول برنامه مثل همیشه معرفی میکرو و کریستال است (که در اینجا از میکرو مگا 16 و کریستال 8 مگاهرتز استفاده شده است).

در خط سوم و چهارم و پنجم lcd راه اندازی شده است (که در اینجا از lcd 16*2 استفاده شده و lcd به پورت d متصل است).

در خط ششم کپید معرفی شده(در این برنامه کپید به پورت b متصل شده و DEBOUNCE پنجاه میلی ثانیه گرفته شده است).

در خط هفتم یک متغیر از جنس بایت معرفی گردیده .

در خط هشتم یک برچسب به نام q قرار داده شده است .

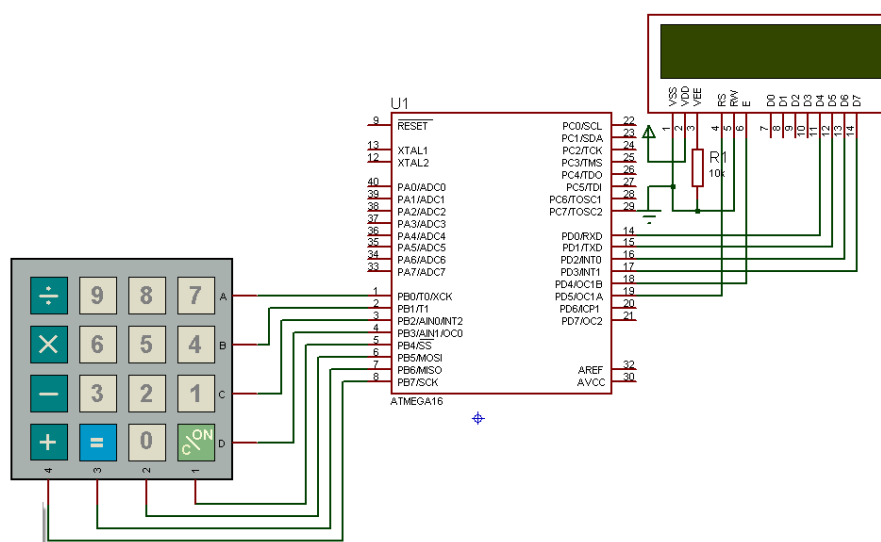
در خط نهم عدد گرفته شده از کپید در متغیر a قرار میگیرد(در صورتی که هیچ کلیدی فشرده نشود مقدار 16 (a=16) در a

ریخته میشود).

در خط ده و یازده و دوازده یک دستور شرطی قرار گرفته ، این دستور شرطی میگوید :اگر a بزرگتر از 15 شد به برجسب q پرش کن (در صورتی که شرط درست باشد دستورات بین if و endif اجرا میشود و اگر شرط درست نباشد برنامه از خط بعد از endif ادامه مییابد).

در خط سیزده و چهارده مقدار a در سطر اول و ستون اول lcd نمایش داده میشود .
و در خط پانزده برنامه به برجسب q پرش میکند و مراحل قبل دوباره تکرار میگردد .
خط آخر برنامه همیشه end است .

مدار مورد استفاده را در تصویر زیر مشاهده میکنید ، شما همچنین میتوانید از شبیه ساز نرم افزار بسکام برای اجرای برنامه بالا و مشاهده ی نتیجه استفاده نماید ، برای آشنایی با نحوه ی عملکرد شبیه ساز به بخش ضمائم مراجعه نمایید .



در این مثال به ازای هر کلید در lcd یک نام نوشته میشود :

```
$regfile = "m16def.dat"
```

```
crystal = 12000000$
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
```

```
Config Kbd = Portb , Debounce = 20
```

```
Dim A As Byte
```

```
Q:
```

```
A = Getkbd()
```

```

If A > 15 Then : Goto Q : End If

If A = 0 Then : Locate 1 , 1 : Lcd "RETURN KEY=0" : End If

If A = 1 Then : Locate 1 , 1 : Lcd "RETURN KEY=1" : End If

If A = 2 Then : Locate 1 , 1 : Lcd "RETURN KEY=2" : End If

If A = 3 Then : Locate 1 , 1 : Lcd "RETURN KEY=3" : End If

If A = 4 Then : Locate 1 , 1 : Lcd "RETURN KEY=4" : End If

If A = 5 Then : Locate 1 , 1 : Lcd "RETURN KEY=5" : End If

If A = 6 Then : Locate 1 , 1 : Lcd "RETURN KEY=6" : End If

If A = 7 Then : Locate 1 , 1 : Lcd "RETURN KEY=7" : End If

If A = 8 Then : Locate 1 , 1 : Lcd "RETURN KEY=8" : End If

If A = 9 Then : Locate 1 , 1 : Lcd "RETURN KEY=9" : End If

If A = 10 Then : Locate 1 , 1 : Lcd "RETURN KEY=10" : End If

If A = 11 Then : Locate 1 , 1 : Lcd "RETURN KEY=11" : End If

If A = 12 Then : Locate 1 , 1 : Lcd "RETURN KEY=12" : End If

If A = 13 Then : Locate 1 , 1 : Lcd "RETURN KEY=13" : End If

If A = 14 Then : Locate 1 , 1 : Lcd "RETURN KEY=14" : End If

If A = 15 Then : Locate 1 , 1 : Lcd " RETURN KEY=15" : End If

Locate 2 , 1 : Lcd A : Goto Q

End

```

در برنامه بالا برای اینکه حجم کمتری اشغال شود دستورات به صورت سطری نوشته شده‌اند ، شما می‌توانید با قرار دادن علامت دونقطه (:) در بین دو دستور آنها را در یک خط بنویسید ، هم چنین با قرار دادن علامت ویرگول (,) چندین علامت را روی lcd در یک خط نمایش دهید ، همچنین با دستور کما (,) چندین متغیر را در یک خط معرفی کنید .

برنامه بالا را با استفاده از جدول lookupstr مینویسیم:

```

$regfile = "m16def.dat"

crystal = 12000000$

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5

Config Kbd = Portb , Debounce = 20

Dim A As Byte

```

```
Dim B As String * 5
```

```
Q:
```

```
A = Getkbd()
```

```
If A > 15 Then
```

```
Goto Q
```

```
End If
```

```
B = Lookupstr(a , W)
```

```
Locate 1 , 1: Lcd B
```

```
Locate 2 , 1: Lcd A
```

```
Goto Q
```

```
End
```

```
W:
```

```
Data " RETURN KEY=0" , " RETURN KEY=1" , " RETURN KEY=2" , " RETURN KEY=3" , " RETURN KEY=4" , " RETURN  
KEY=5" , " RETURN KEY=6" , " RETURN KEY=7" , " RETURN KEY=8" , " RETURN KEY=9" , " RETURN KEY=10" , " RETURN  
KEY=11" , " RETURN KEY=12" , " RETURN KEY=13" , " RETURN KEY=14" , " RETURN KEY=15"
```

جدول lookupstr نیز مانند جدول lookup است اما جدول lookup برای بازگردانی اعداد و جدول lookupstr برای بازگردانی حروف به کار میرود .

شما جای پایه های kbd که به پایه میکرو متصل است را تغییر دهید و نتیجه را ببینید. همیشه نیاز نیست که ورودی یک کلید باشد بعضی وقت ها می توان صفر یا یک شدن یک پین را هم چک کرد.

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Kbd = Portb
```

```
Dim A As Byte
```

```
Q:
```

```
A = Getkbd()
```

```
If A > 15 Then
```

```
Goto Q
```

```
End If
```

```
Dtmfout A , 50
```

Goto Q

End

در این مثال عدد گرفته شده از کیپد به پالس dtmf تبدیل شده و به خط تلفن ارسال میشود ، در صورتی که در بین فشردن کیپد ها تاخیر زیادی رخ دهد خط تلفن اشغال میشود.

بعضی وقت ها به کیپد با تعداد کیپد بیشتر نیاز است شما با دستور زیر 2 سطر دیگر به تعدا سطر های کیپد اضافه کنید (در مجموع 24 کیپد)

Config Kbd = Portx , Debounce = Value , Rows = 6 , Row5 = Pina.b, Row6 = Pina.b

X نام پورتی است که کیپد به آن متصل شده است (4سطر و 4ستون اصلی).

Value مقدار تاخیر در فشردن کیپد برای گرفتن لرزش است.

a.b نام پورت و پینی است که دوسطر دیگر به آن متصل شده اند.

\$regfile = "m16def.dat"

\$crystal = 12000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5

Config Kbd = Portb , Debounce = 20 , Rows = 6 , Row5 = Pina.0 , Row6 = Pina.1

Dim A As Byte

Q:

A = Getkbd()

If A > 16 Then : Goto Q : End If

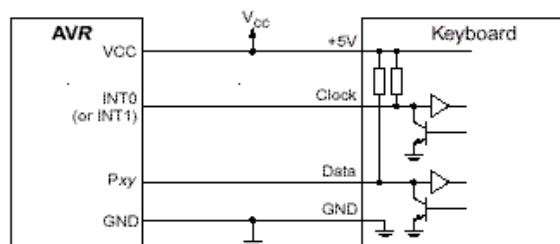
Locate 1 , 1 : Lcd A : Goto Q



End

اتصال کیبرد کامپیوتر به AVR :

در صورتی که کیبرد های ماتریسی پاسخ گوی نیاز شما نیست و باز هم به کیپد های بیشتری نیاز دارید میتوانید از کیبرد های کامپیوتر استفاده کنید ، اتصال کیبرد به avr در کامپایلر بسکام کار ساده ای میباشد ،چون تمامی توابع تعریف شده هستند و نیاز به نوشتن برنامه اضافه نمی باشد . این کیبرد ها تعداد بیش از 80 کیپد را در اختیار شما قرار میدهد و شما میتوانید توسط دستورات شرطی عملیات دلخواه را با توجه به کیپد فشرده شده انجام دهید .

کیبرد کامپیوتر دارای 4 سیم میباشد که دوتا از آنها مربوط به تغذیه ، یکی برای انتقال داده به میکرو کنترلر و دیگری کلاک (پالس همزمانی) میباشد .



AT Computer		
Signals	DIN41524, Female at Computer, 5-pin DIN 180°	6-pin Mini DIN PS2 Style Female at Computer
Clock	1	5
Data	2	1
nc	3	2,6
GND	4	3
+5V	5	4
Shield	Shell	Shell

پیکربندی کیبرد در کامپایلر بسکام به صورت زیر است :

CONFIG KEYBOARD = PINX.y , DATA = PINX.y , KEYDATA = table

در دستور بالا PINX.y یکی از پایه های دلخواه میکرو می باشد و table نام جدول کدهای کیبرد است (از آنجا که کدهای گرفته شده از کیبرد هگز می باشد ، باید به وسیله یک جدول آنها را به کد اسکی برای نمایش تبدیل کرد).

مقدار گرفته شده از کیبرد بعد از تبدیل به کد اسکی باید در یک متغیر ریخته شود که این کار با دستور زیر انجام میشود .

B = Getatkbd()

اگر کلیدی فشرده نشود مقدار صفر در b ریخته می شود .

به همین سبب در برنامه یک دستور شرط (if) قرار داده می شود تا موقعی که کلید فشرده نشود مقدار صفر در b قرار نگیرد.

برای درک بیشتر موضوع به مثال زیر توجه کنید :

\$regfile = "m16def.dat"

\$crystal = 12000000

Config Lcd = 16 * 2

```
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
```

```
Config Keyboard = Pind.2 , Data = Pind.4 , Keydata = Keydata
```

```
Dim B As Byte
```

```
Do
```

```
B = Getatkbd()
```

```
If B > 0 Then
```

```
Loocate 1,1
```

```
Lcd, B
```

```
End If
```

```
Loop
```

```
End
```

```
Keydata:
```

```
'normal keys lower case
```

```
Data 0 , 0 , 0 , 0 , 0 , 200 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , &H5E , 0
```

```
Data 0 , 0 , 0 , 0 , 0 , 113 , 49 , 0 , 0 , 0 , 122 , 115 , 97 , 119 , 50 , 0
```

```
Data 0 , 99 , 120 , 100 , 101 , 52 , 51 , 0 , 0 , 32 , 118 , 102 , 116 , 114 , 53 , 0
```

```
Data 0 , 110 , 98 , 104 , 103 , 121 , 54 , 7 , 8 , 44 , 109 , 106 , 117 , 55 , 56 , 0
```

```
Data 0 , 44 , 107 , 105 , 111 , 48 , 57 , 0 , 0 , 46 , 45 , 108 , 48 , 112 , 43 , 0
```

```
Data 0 , 0 , 0 , 0 , 0 , 92 , 0 , 0 , 0 , 0 , 13 , 0 , 0 , 92 , 0 , 0
```

```
Data 0 , 60 , 0 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0
```

```
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0
```

```
'shifted keys UPPER case
```

```
Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
```

```
Data 0 , 0 , 0 , 0 , 0 , 81 , 33 , 0 , 0 , 0 , 90 , 83 , 65 , 87 , 34 , 0
```

```
Data 0 , 67 , 88 , 68 , 69 , 0 , 35 , 0 , 0 , 32 , 86 , 70 , 84 , 82 , 37 , 0
```

```
Data 0 , 78 , 66 , 72 , 71 , 89 , 38 , 0 , 0 , 76 , 77 , 74 , 85 , 47 , 40 , 0
```

```
Data 0 , 59 , 75 , 73 , 79 , 61 , 41 , 0 , 0 , 58 , 95 , 76 , 48 , 80 , 63 , 0
```

```
Data 0 , 0 , 0 , 0 , 0 , 96 , 0 , 0 , 0 , 0 , 13 , 94 , 0 , 42 , 0 , 0
```

```
Data 0 , 62 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0 , 0
```

```
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0
```

برای کسب اطلاعات بیشتر در مورد کدهای ساخته شده توسط کیبورد و نحوه تولید آنها به [این آدرس](#) مراجعه نمایید .

راه اندازی وقفه های داخلی و خارجی :

به طور کلی ، وقفه ها مکان های از حافظه فلش هستند که با فعال کردن آنها ، cpu میتواند در مواقع ضروری برنامه ی در دست اجرا را رها کرده و به سمت آنها پرش کند ، میکرو کنترلر های Avr دارای بیش از 40 منبع داخلی و 7 منبع وقفه خارجی میباشد . با وقفه های داخلی در هنگام کار با واحد های جانبی (همچون adc و ..) آشنا خواهیم شد ، بحثی که در ادامه آورده شده اطلاعات اولیه را در اختیار شما قرار میدهد .

برای راه اندازی وقفه های داخلی در میکرو کنترلر های سری TINY و AT90S و ATMEGA ابتدا باید با استفاده از دستور زیر وقفه ی کلی را فعال کنیم :

```
ENABLE interrupt
```

در میکرو کنترلر های سری Atxmega علاوه بر دستور بالا باید از مجموعه دستورات زیر نیز برای مشخص کردن بردار وقفه و نحوه ی واکنش به آن استفاده کنیم :

```
CONFIG PRIORITY= prio, VECTOR= vector, HI= hi, LO= lo, MED= med
```

Prio : به جای این واژه میتوانیم از دستور STATIC یا ROUNDROBIN استفاده کنیم ، در میکرو کنترلر های سری Atxmega وقفه ها دارای اولویت بندی بوده و کم ارزش ترین وقفه دارای بیشترین اولویت میباشد در این بخش هنگامی که کاربر از STATIC استفاده میکند ، میکرو کنترلر مانند یک میکرو کنترلر معمولی عمل کرده و اولین وقفه ی رخ داده را اجرا میکند ، با انتخاب دستور ROUNDROBIN وقفه ها به ترتیب اولویت خود اجرا می شوند ، مثلاً اگر وقفه ی adc فعال شود و بعد از یکی از ورودی های وقفه set شود ، cpu ابتدا زیر برنامه ی نوشته شده برای ورودی وقفه (INTX) را اجرا میکند و سپس به زیر برنامه ی وقفه ی ADC می رود .

VECTOR= vector : با این دستور بردار وقفه و محل زیر برنامه ی موجود که می تواند در یکی از حافظه های BOOT یا APPLICATION باشد مشخص می شود ، در حالت عادی برنامه ی های نوشته شده در حافظه ی برنامه (APPLICATION) ذخیره میشود و کاربر باید در این بخش از دستور Vector = Application استفاده کند .

HI= hi, LO= lo, MED= med : با قرار دادن دستورات ENABLED یا DISABLED در مقابل هر یک از این دستورات می توانید اولویت وقفه ی های پیکربندی شده را در سطوح زیاد ، متوسط یا کم تعیین کنید .

بعد از فعال سازی وقفه ی سراسری میتوانیم با دستور enable میتوانیم وقفه ی سایر بخش ها را نیز فعال نماییم :

```
ENABLE interrupt [, prio]
```


در دستور بالا interrupt میتواند یکی از دستورات موجود در جدول زیر برای راه اندازی وقفه ی بخش های مختلف باشد، مادر بخش های مختلف این کتاب با نحوه ی استفاده از وقفه ی واحد های جانبی آشنا خواهیم شد :

Interrupt	Description
INT0	External Interrupt 0
INT1	External Interrupt 1
OVF0,TIMER0, COUNTER0	TIMER0 overflow interrupt
OVF1,TIMER1, COUNTER1	TIMER1 overflow interrupt
CAPTURE1, ICP1	INPUT CAPTURE TIMER1 interrupt
COMPARE1A,OC1A or COMPARE1, OC1	TIMER1 OUTPUT COMPARE A interrupt In case of only one compare interrupt
COMPARE1B,OC1B	TIMER1 OUTPUT COMPARE B interrupt
SPI	SPI interrupt
URXC	Serial RX complete interrupt
UDRE	Serial data register empty interrupt
UTXC	Serial TX complete interrupt
SERIAL	Disables URXC, UDRE and UTXC
ACI	Analog comparator interrupt
ADC	A/D converter interrupt
XMEGA ONLY	
prio	The priority you want to assign to the interrupt. Specify Lo, Hi or Med. In the Xmega you must provide the priority of the interrupts. Lo=Low priority. Hi=High priority and Med=Medium priority. If you do not specify a priority, LO will be used.

در دستور بالا prio میزان اولویت وقفه ی بخش مورد نظر را مشخص میکند، به جای prio میتوانید از دستورات LO (برای وقفه با اولویت کم) یا HI (برای وقفه با اولویت بالا) یا MED (برای وقفه با اولویت میانه) استفاده کنید، مثال :

Config Priority = Static , Vector = Application , Lo = Enabled

On Usartc0_rxc Rxc_isr

Enable Usartc0_rxc , Lo

Enable Interrupts

مثال :

Enable OVF0 'TIMER0 overflow Interrupt

Enable ADC 'A/D converter Interrupt

,.....

و در نهایت با دستور زیر میتوانیم در هنگام رخ دادن وقفه cpu را به برجسب دلخواه هدایت کنیم :

On Interrupt label

با دستور بالا هنگامی که وقفه ای رخ میدهد ، Cpu به برجسب label پرش میکند ، و بعد از انجام دادن دستور در ادامه ی برجسب با دستور Return به برنامه ی اصلی برمیگردد .

مثال :

```
Config Priority = Static , Vector = Application , Lo = Enabled , Med = Enabled , Hi = Enabled
On Aca_acw Ac_window_isr
Enable Aca_acw , Lo
Enable Interrupts
```

در این مثال وقفه ی مقایسه کننده ی آنالوگ در یکی از میکروکنترلر های خانواده ی ATXMEGA فعال شده است ، با رخ دادن وقفه CPU میکروکنترلر به برجسب Ac_window_isr پرش کرده و زیر برنامه ی موجود در آن را اجرا میکند .

دستور Enable Aca_acw , Lo باعث فعال سازی وقفه ی مقایسه کننده با اولیت کم میشود .

مثال :

```
$regfile = "8535def.dat"           ' specify the used micro
$crystal = 4000000                 ' used crystal frequency
$baud = 19200                      ' use baud rate
Const Cmaxchar = 20                'number of characters
Dim B As Bit                       'a flag for signalling a received character
Dim Bc As Byte                     'byte counter
Dim Buf As String * Cmaxchar       'serial buffer
Dim D As Byte
'Buf = Space(20)
'unremark line above for the MID() function in the ISR 'we need to fill the buffer with spaces otherwise it will contain garbage
Print "Start"
On Urxc Rec_isr                    'define serial receive ISR
Enable Urxc                        'enable receive isr
Enable Interrupts                  'enable interrupts to occur
Do
If B = 1 Then                      'we received something
```

```

Disable Serial
Print Buf                                'print buffer
Print Bc                                'print character counter  'now check for buffer full
If Bc = Cmaxchar Then                    'buffer full
    Buf = ""                             'clear
    Bc = 0                               'rest character counter
End If
Reset B                                  'reset receive flag
Enable Serial
End If
Loop
Rec_isr:
Print "*"
If Bc < Cmaxchar Then                    'does it fit into the buffer?
    Incr Bc                              'increase buffer counter
    If Udr = 13 Then                      'return?
        Buf = Buf + Chr(0)
        Bc = Cmaxchar
    Else
        Buf = Buf + Chr(udr)             'add to buffer
    End If ' Mid(buf , Bc , 1) = Udr
    'unremark line above and remark the line with Chr() to place
    'the character into a certain position
    'B = 1                               'set flag
End If
B = 1                                    'set flag
Return

```

در برنامه ی بالا بعد از فعال سازی وقفه ی واحد uart فعال شده است ، در این حالت اگر داده وارد بافر انتقال داده ی واحد uart شود ، وقفه فعال میشود و

هدف از مطرح کردن موارد بالا ، آشنایی کلی شما با وقفه ها بود مادر ادامه بیشتر به بررسی منابع وقفه ی داخلی خواهیم پرداخت .

در این بین موردی که در بخش منابع وقفه نا تمام باقی مانده است ، منابع تحریک خارجی است .

میکروکنترلر های AVR دارای تعدادی پایه میباشند که در نام گذاری پایه ها با نام intx مشخص می شوند ، هنگامی که

این پایه ها را در برنامه راه اندازی میکنیم ، با تحریک شدن آنها میکرو میتواند به برچسب موجود در برنامه پرش کند . با

استفاده از دستور زیر میتوان منابع وقفه خارجی را راه اندازی کرد :

CONFIG INTx = state

INTx : نام پایه وقفه است که در اکثر میکرو ها به نام INT0 و INT1 و در بعضی از میکرو ها به نام INT0 تا INT7 موجود است .

State : نوع پالس اعمالی برای فعال شده وقفه را معین میکند ، State میتواند یکی از موارد زیر باشد:

Falling : با اعمال یک پالس پایین روند (یک به صفر) به پایه مورد نظر وقفه فعال میشود

RISING : با اعمال یک پالس بالا رونده (صفر به یک) به پایه مورد نظر وقفه فعال میشود.

LOW LEVEL : با اعمال سطح صفر به پایه مورد نظر وقفه فعال میشود.

سپس با دستور Enable Interrupts وقفه سراسری و بادرستور Enable Intx وقفه پیکر بندی شده فعال میشود.

و در نهایت شما میتوانید با دستور On Intx lable ، به هنگام اعمال پالس به lable مورد نظر پرش کنید .

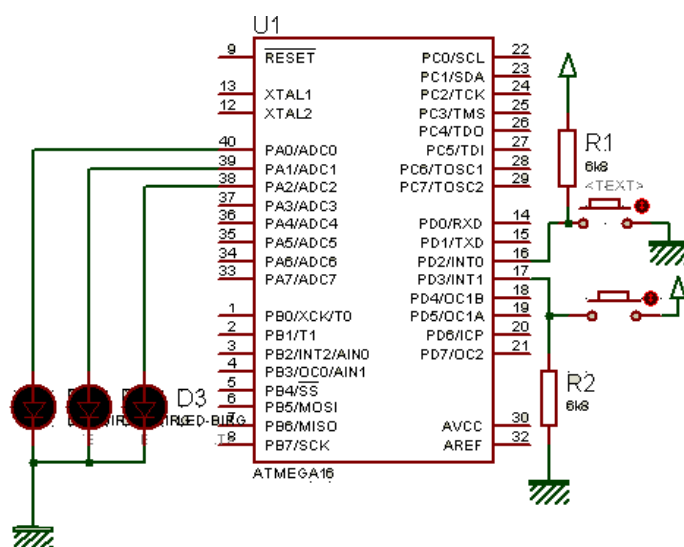
با فعال شدن وقفه ، پایه های ورودی وقفه در همه حالت ها چک میشود و نیازی به آوردن دستورات در حلقه اصلی و دیگر

حلقه ها نیست . باز گشت از زیر برنامه با دستور Return انجام میشود .مانند:

```
$regfile = "m16def.dat" : $crystal = 8000000
Config Porta = Output
Config Int0 = Falling
Config Int1 = Rising
Enable Int0
Enable Int1
Enable Interrupts
On Int0 Q
On Int1 W
Do
Set Porta.2 : Waitms 500 : Reset Porta.2 : Waitms 500
Loop
End
Q:
If Porta.0 = 0 Then : Set Porta.0 : Else : Reset Porta.0 : End If
Return
W:
If Porta.1 = 0 Then : Set Porta.1 : Else : Reset Porta.1 : End If
Return
```

در برنامه بالا از دو منبع وقفه $int0$ و $int1$ استفاده شده است ، همانگونه که می بینید به یکی از پایه ها یک پالس بالا رونده و به دیگری پالس پایین رونده اعمال میشود.

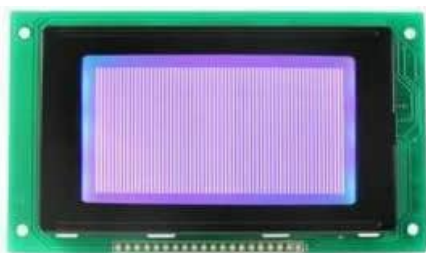
در حلقه ی اصلی مدام یکی از پایه ها خاموش و روشن میشود و منبع وقفه مدام چک میشوند ، هنگامی که پالس مشخص شده به پایه وقفه اعمال شد به زیر برنامه تعریف شده (q برای منبع وقفه صفر و w برای منبع وقفه 1) پرش میشود و وضعیت دو led تغییر میکند . مدار مورد استفاده برای این برنامه در زیر آورده شده است .



نمایشگر های کریستال مایع و LED :

نمایشگر های کریستال مایع یا LCD و نمایشگر های هفت قسمتی یا 7SEGMENT کاربرد گسترده ای در مدارات میکرو کنترلری دارند ، ما از این نمایشگر ها برای نمایش وضعیت ها ، دستورات ، پیغام ها و ... استفاده میکنیم . این نوع نمایشگر ها که به دسته های کارکتری ، گرافیکی تک رنگ و گرافیکی رنگی ، LED تکی ، LED مولتی پلکس ... تقسیم میشوند ، با استفاده از پروتکل های مختلفی با میکرو تبادل داده میکنند .

LCD گرافیکی :



lcd های گرافیکی در نمونه های مختلف در بازار وجود دارد و میتواند تصاویر و متون را در صفحه ی $x \times x$ پیکسلی خود نمایش دهد . این نوع lcd ها معمولا به صورت موازی به میکرو کنترلر متصل میشوند و از طریق 8 خط داده و 6 خط کنترل با آن تبادل داده میکنند . داده ی مبادله شده در این lcd ها به نوع چیپ آنها بستگی دارد ، در ادامه به بررسی نحوه ی راه اندازی lcd های مبتنی بر دو چیپ t6963 و KS108 (که با نام SED نیز شناخته میشود) پرداخته ایم . شما میتوانید اطلاعات مربوط به نوع چیپ lcd را در دیتاشیت lcd مشاهده کرده یا از فروشنده بپرسید .

کلیه lcd ها گرافیکی دارای پایه های زیر میباشد ، البته ممکن است پایه های به جز این نیز وجود داشته باشد که در راه اندازی lcd نقشی ندارد .

1 - vss: پایه تغذیه lcd که به 0 ولت متصل میشود.

2 - vdd: پایه تغذیه lcd که به 5 ولت متصل میشود.

3 - d0 تا d7 دیتا پورت (dataport) این 8 پایه مربوط به دیتای lcd میباشد (lcd اطلاعات را از طریق این 8 پایه رد و بدل میکند) که به یکی از پورت های میکرو که در برنامه مشخص می شود متصل میشود .

4 - controlport: که شامل پایه های زیر است و به یکی از پورت های میکرو که در برنامه مشخص میشود متصل میشود. این پایه ها برای کنترل lcd به کار میروند .

– rst: پایه ریست (باز نشانی) lcd، که به یکی از پایه های میکرو که در برنامه مشخص میشود متصل میگردد.

– CEx یا CSx: این پایه برای فعال کردن چیپ lcd است، که به یکی از پایه های میکرو که در برنامه مشخص میشود متصل

میگردد. ممکن است در برخی lcd ها دو عدد از این پایه ها وجود داشته باشد (در lcd ها بزرگتر از 64*64)

– cd: این پایه مشخص کننده ارسال کد یا دیتا است (بدین صورت که اگر این پایه 1 باشد lcd کد را میگیرد و اگر 0 باشد

lcd دیتا را میگیرد) (دیتا فرمانها می باشد و کد متن ها و اشکال است))، که به یکی از پایه های میکرو که در برنامه مشخص

میشود متصل میگردد.

– wr (در LCD با چیپ T6963 و ENABLE در LCD با چیپ KS108): این پایه برای نوشتن در lcd یا فعال کردن آن است،

که به یکی از پایه های میکرو که در برنامه مشخص میشود متصل میگردد.

– rd: این پایه برای خوانده از lcd است، که به یکی از پایه های میکرو که در برنامه مشخص میشود متصل میگردد.

– fs: این پایه برای مشخص کردن فونت lcd است، که به یکی از پایه های میکرو که در برنامه مشخص میشود متصل

میگردد.

5 – con یا vo: پایه کنترل کنتراست lcd است که با توجه به نوع lcd به vcc یا -vcc هر ولتاژدیگر متصل میشود.

با توجه به مطالب بالا پیکر بندی lcd گرافیکی در بسکام به صورت زیر است:

Config GRAPHLCD = type , DATAPORT = port, CONTROLPORT=port , CE = pin , CD = pin , WR = pin, RD=pin, RESET=

pin, FS=pin, MODE = mode

type: نام lcd است که میتواند یکی از موارد زیر باشد:

128 * 240 , 48 * 160 , 64 * 128 , 128 * 128 , 64 * 240 , 128 * 64 sed , ...

Config GRAPHLCD = 64*240

port: یکی از پورت های دلخواه میکرو است مثال:

DATAPORT = portd, CONTROLPORT=portb

pin: یکی از پایه های دلخواه پورتهی است که در قسمت CONTROLPORT مشخص شده است: مثال برای راه اندازی lcd با

چیپ T6963:

Config Graphlcd = 240 * 128 , Dataport = Porta , Controlport = Portc , Ce = 2 , Cd = 3 , Wr = 0 , Rd = 1 , Reset = 4 , Fs = 5 , Mode = 8

mode: مشخص کننده تعداد ستون متنی lcd است که میتواند 6 یا 8 باشد

شرح پایه ها در مثال بالا مطابق زیر است :

شماره پایه	نام پایه بر روی lcd	محل اتصال
1	GND	GND
2	GND	GND
3	+5V	VCC
4	V-	-9V potmeter
5	WR	PORTC.0
6	RD	PORTC.1
7	CE	PORTC.2
8	C/D	PORTC.3
9	NC	not conneted
10	RESET	PORTC.4
18-11	D0-D7	porta
19	FS	PORTC.5
20	NC	not connected

مثال برای اندازی lcd با چیپ KS108 :

Config Graphlcd = 128 * 64sed , Dataport = Portd , Controlport = Portc , Ce = 3 , Ce2 = 4 , Cd = 0 , Rd = 1 , Reset = 6 , Enable = 2

نحوه ی اتصال LCD به میکرو مطابق مدار زیر است :

توجه داشته باشید که :

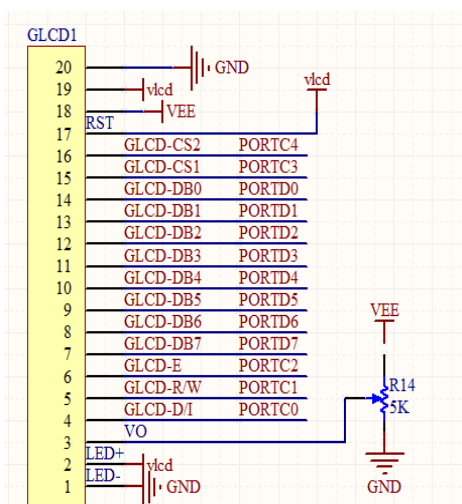
✓ در برخی از LCD ها ممکن است حروف نوشته شده با

مطالب بیان شده متفاوت باشد ، در این حالت پایه های میکرو را به پایه های

متنظر متصل کنید (طبق تصویر مقابل)

✓ تقریباً 90٪ نمایشگرهای موجود در بازار از دو چیپ معرفی شده

استفاده میکنند ، در صورتی که نوع چیپ را نیدانید هر دو نوع برنامه را امتحان کنید .



دستورات مربوط به lcd گرافیکی با چیپ T6963 :

دستور lcd :

با این دستور میتوان متن یا کاراکتری را بر روی lcd نمایش داد .مثال:

Lcd "MCS Electronics"

Lcd "Mdgdgsdsscs"

دستور locate :

با این دستور میتوان متن یا کاراکتری را در مکان دلخواه بر روی lcd گرافیکی نمایش داد . مثال:

Locate 16 , 1

Lcd "write this to the lower line"

Locate 16 , 5

Lcd "fgghfghfhgjhj"

دستورات مربوط به lcd گرافیکی با چیپ ks108 :

دستور lcdat :

Lcdat y , x , var [, inv]

این دستور متغیر یا داده ی var را در مختصات x و y نمایش میدهد ، همچنین inv یک عملگر اختیاری است که میتوانید با صفر کردن آن داده ی var را به صورت نرمال و با دادن دیگر مقادیر داده را به صورت معکوس نمایش دهید .

توجه داشته باشید که حداکثر مقدار x و y به اندازه ی lcd گرافیکی بستگی دارد ، مثلا برای یک lcd با ابعاد 128*64 پیکسل مقدار x میتواند مابین 0 تا 63 و مقدار y میتواند در محدوده ی 0 تا 127 تغییر کند . مثال :

Lcdat 4 , 1 , "0123456789..."

Lcdat 4 , 1 , a , 1

دستور Glcdcmd :

توسط این دستور که به فرم زیر نوشته میشود ، میتوان فرمانی را به lcd گرافیکی ارسال کرد :

Glcdcmd byte , chip

فرمان دلخواه باید به جای byte نوشته شود ، مثلا ما میتوانیم با ارسال 3E در مبنای هگز کنترل کننده ی lcd را خاموش و با ارسال 3f آن را روشن کنیم .

در این دستور به جای chip شماره ی قطعه ی کنترل کننده ی lcd قرار میگیرد ، مثلا اگر شما از یک نمایشگر با ابعاد 64*128 استفاده میکنید ، این نمایشگر دارای دو کنترلر کننده میباشد که هر یک تعداد 64*64 پیکسل را کنترل میکند ، در صورتی که مالید دستور موجود را به کنترلر کننده ی اول ارسال کنید به جای chip رقم یک و اگر مالید دستور را به کنترلر کننده ی دوم ارسال کنید به جای chip رقم 2 را قرار دهید . مثال :

Glcddcmd &H3E , 1 : Glcddcmd &H3E , 2 'Off

Waitms 70

Glcddcmd &H3F , 1 : Glcddcmd &H3F , 2 'On

دستور GLCDDATA :

GLCDDATA byte

با استفاده از این دستور میتوان داده ای را به lcd ارسال کرد ، داده میتواند یک عدد یا یک متغیر باشد که به جای byte نوشته میشود .

دستور SETFONT :

با استفاده از این دستور میتوانید ابعاد (فونت) داده ی ارسالی که بر روی lcd نوشته میشود را تغییر دهید ، برای استفاده از این دستور باید مراحل زیر را انجام دهید :

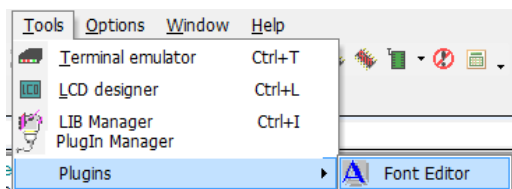
❖ در صورتی که فونت دلخواه شما آماده است آن را با دستور زیر به برنامه وارد کنید ، توجه داشته باشید که فونت باید در محل ذخیره ی برنامه موجود باشد : (این دستور بعد از دستور end (پایان برنامه) قرار میگیرد .

\$include "font8x8.font"

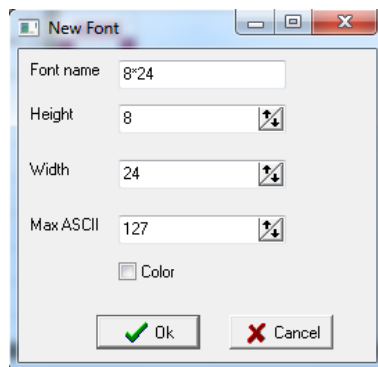
✓ با استفاده از دستور زیر میتوانید فونت نوشته ی روی lcd را تغییر دهید :

SetFont Font8x8

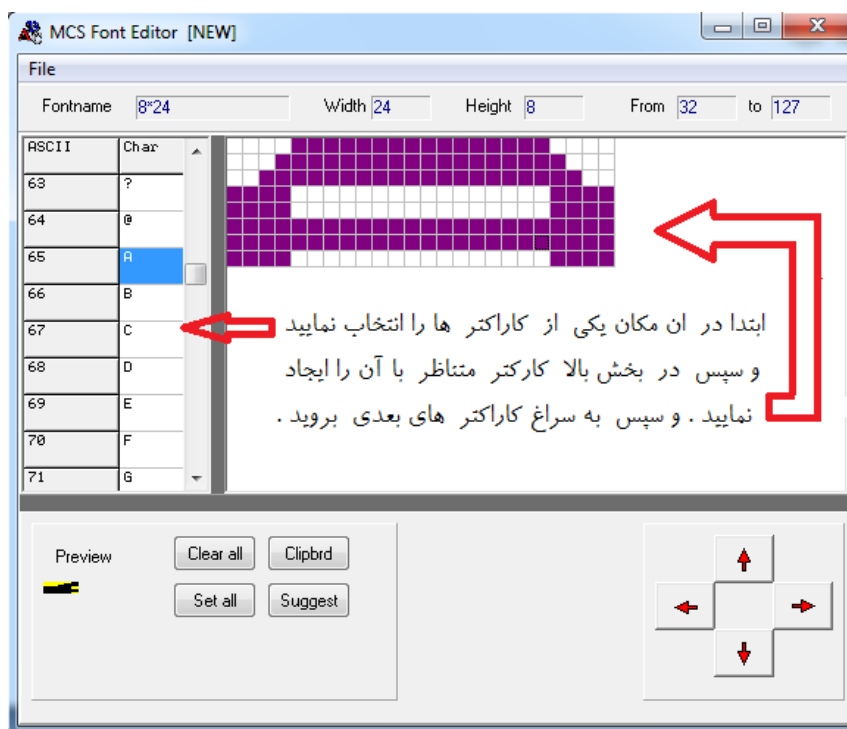
در صورتی که مالید فونت دلخواه خود را ایجاد کنید ، از مسیر tools>plugins گزینه ی font editor را انتخاب نمایید .



در پنجره ی باز شده، از منوی file گزینه ی new را انتخاب کنید و در پنجره ی باز شده، بعد از وارد کردن نام دلخواه و اندازه ی فونت بر روی ok کلیک کنید.



در پنجره ی font editor کردن کاراکتر دلخواه را انتخاب کرده و فونت مربوط به آن را در جدول موجود ایجاد نمایید و این کار را برای تمامی اعداد و حروف موجود تکرار کنید و در با انتخاب گزینه ی save از منوی file فونت ایجاد شده را در کنار برنامه ذخیره نمایید، شما میتوانید از فونت ایجاد شده در سایر برنامه نیز استفاده کنید.



در صورتی که نمیتوانید گزینه ی font editor را در منوی tools بیابید، از منوی tools گزینه ی plugin maneger را انتخاب نمایید و تیک کلیه گزینه های موجود در پنجره ی باز شده را بگذارید .

در بالا دستورات مربوط به دو نوع lcd گرافیکی موجود در بازار را بررسی کرده ایم ، سایر دستوراتی که در ادامه آورده شده است برای هر دو نوع lcd مشترک بوده و شما میتوانید بعد از پیکربندی lcd گرافیکی از آنها استفاده نمایید ، برای درک بهتر دستورات به دو مثالی که در پایان این بخش آورده شده است مراجعه نمایید .

دستور cls :

با این دستور تمام lcd پاک میشود . با استفاده از دستور Cls Text می توان قسمت متنی lcd را پاک کرد و با دستور Cls graph می توان قسمت گرافیکی را پاک کرد .

مثال :

```
$regfile = "m16def.dat"
$crystal = 8000000
Config Graphlcd = 240 * 128 , Dataport = Porta , Controlport = Portc , Ce = 2 , Cd = 3 , Wr = 0 , Rd = 1 , Reset = 4 , Fs = 5 ,
Mode = 8
Cls
Wait 1
Locate 1 , 1
Lcd "1nafar"
Locate 2 , 1
Lcd "/*-+234#$$%^&*"
Locate 3 , 1
Lcd "1234567890123456789012345678901234567890"
Locate 16 , 1
Lcd "www.iranmicro.ir"
Wait 2
Lcd "for ever"
Locate 2 , 20
Lcd "in the ..."
Locate 3 , 13
Lcd "avr basic book"
Locate 30 , 1
```

```
Lcd "qwertyuiop"
Wait 2
Cls Text
End
```

دستور $\text{pset } X, Y, \text{value}$:

این دستور یک پیکسل را در مختصات x, y به ازای $\text{value} = 255$ روشن و به ازای $\text{value} = 0$ خاموش میکند ، مثال:

```
Pset 10 , 20 , 255
```

```
Pset 5, 127 , 255
```

```
Pset 10 , 20 , 0
```

```
Pset 5, 127 , 0
```

حداکثر مقدار x, y بستگی به تعداد پیکسل lcd گرافیکی دارد برای مثال در lcd 240×128 حداکثر مقدار $x=239, y=127$ است .

دستور $\text{CURSOR ON / OFF BLINK / NOBLINK}$:

Lcd گرافیکی همچون lcd کاراکتری دارای یک مکان نما می باشد که با دستور زیر میتوان آن را روشن یا خاموش یا

چشمک زن یا ثابت قرارداد :

Cursor On با این دستور مکان نما روشن می شود (در حالت عادی مکان نما روشن است).

Cursor off با این دستور مکان نما خاموش می شود.

Cursor blink با این دستور مکان نما چشمک می زند .

Cursor noblink با این دستور مکان نما دیگر چشمک نمی زند .

دستور $\text{LINE}(x0,y0) - (x1,y1), \text{color}$

با این دستور میتوان در lcd یک خط کشید ، که $(x0,y0)$ پیکسل شروع خط و $(x1,y1)$ پیکسل پایان خط است و $\text{color}=255$

خط با رنگ مشکی و $\text{color}=0$ خطی با رنگ سفید رسم خواهد کرد .

مثال :

```
$regfile = "m165def.dat"
```

```
$crystal = 8000000
```

```
Config Graphlcd = 240 * 128 , Dataport = Porta , Controlport = Portc , Ce = 2 , Cd = 3 , Wr = 0 , Rd = 1 , Reset = 4 , Fs = 5 ,
```

```

Mode = 8
Cls
Cursor Blink
Wait 1
Cursor On
Wait 1
Cursor Off
Locate 1 , 1
Lcd "MCS Electronics"
Locate 2 , 1 : Lcd "T6963c support"
Locate 3 , 1 : Lcd "123456789012345678901234567890"
Locate 16 , 1 : Lcd "write this to the lower line"
Wait 2
Cls Text
Line(0 , 0) -(239 , 127) , 255
Line(0 , 127) -(239 , 0) , 255
Line(0 , 0) -(240 , 0) , 255
Line(0 , 127) -(239 , 127) , 255
Line(0 , 0) -(0 , 127) , 255
Line(239 , 0) -(239 , 127) , 255
Wait 3
Cls Graph
End

```

دستور : CIRCLE(x0,y0) , radius, color

این دستور یک دایره بر روی lcd میکشد، (x0,y0) مرکز دایره و radius شعاع دایره می باشد و color=255 دایره با رنگ مشکی و color=0 دایره با رنگ سفید (دایره را پاک میکند) رسم خواهد کرد .

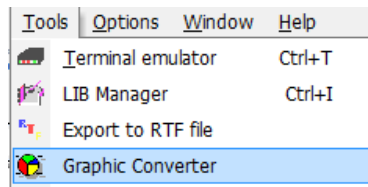
دستور SHOWPIC x, y , label

این دستور یک عکس را بر روی lcd گرافیکی نمایش میدهد .

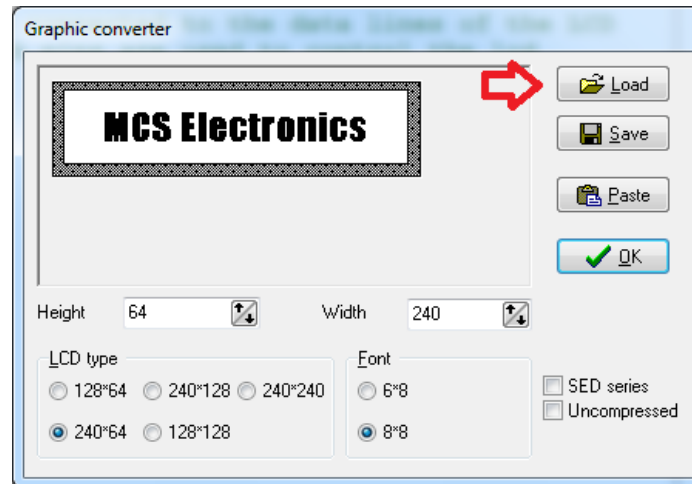
مراحل نمایش عکس بر روی lcd گرافیکی:

اگر عکس مورد نظر رنگی است آن را به محیط فتوشاپ برده و در آنجا آن را به عکس سیاه و سفید تبدیل کنید سپس آن را با برنامه point و با پسوند BMP و در اندازه استاندارد ذخیره کنید (اندازه صفحه نمایش LCD).

سپس از منوی TOOLS گزینه Graphic Converter را انتخاب کنید :



در پنجره باز شده گزینه load را بزنید و در پنجره باز شده عکس مورد نظر که با پسوند BMP ذخیره کردید، باز کنید.



بعد از مشخص کردن نوع lcd و ابعاد عکس، گزینه save را بزنید و فایل را با نام دلخواه و با پسوند BGF در کنار برنامه ذخیره کنید.

با استفاده از دستور SHOWPICE x, y, label عکس را در مختصات x, y نمایش دهید. label نام برجسیبی است که عکس مورد نظر در آن قرار میگیرد. برجسب "\$mcs.bgf" اشاره به عکس مورد نظر که در کنار برنامه اصلی قرار گرفته است.

مثال :

در مثال زیر یک lcd گرافیکی 240 * 128 با چیپ راه انداز T6963c، پیکر بندی شده است، نام تصویری که روی lcd نمایش داده میشود qwe است که در محل ذخیره برنامه ذخیره شده است. در زیر نام پایه های lcd و محل اتصال آنها در میکرو مشاهده میکنید. در صورتی که قصد دارید این مثال را در عمل اجرا کنید باید فیوز بیت jtag را از کار ببندازید. برای کسب اطلاعات بیشتر در مورد فیوز بیت ها به بخش ضمائم و دیتاشیت فارسی ATMEGA16 مراجعه کنید :

شماره پایه	نام پایه بر روی lcd	محل اتصال
1	GND	GND
2	GND	GND
3	+5V	VCC
4	V-	-9V potmeter

PORTC.0	WR	5
PORTC.1	RD	6
PORTC.2	CE	7
PORTC.3	C/D	8
not conneted	NC	9
PORTC.4	RESET	10
porta	D0-D7	18-11
PORTC.5	FS	19
not connected	NC	20

برنامه:

```
$regfile = "m16def.dat"
```

```
$crystal = 12000000
```

```
Config Graphlcd = 240 * 128 , Dataport = Porta , Controlport = Portc , Ce = 2 , Cd = 3 , Wr = 0 , Rd = 1 , Reset = 4 , Fs = 5 , Mode = 8
```

```
Dim X As Byte , Y As Byte
```

```
Cls
```

```
Cursor Off
```

```
Wait 1
```

```
Locate 1 , 1
```

```
Lcd "HELLO WORLD"
```

```
Locate 2 , 1 : Lcd "T6963c support"
```

```
Locate 3 , 1 : Lcd "123456789"
```

```
Wait 2
```

```
Cls Text
```

```
For X = 0 To 140
```

```
Pset X , 20 , 255 ' set the pixel
```

```
Next
```

```
Wait 2
```

```
Showpic 0 , 0 , Plaatje
```

```
Wait 2
```

```
Cls Text ' clear the text
```

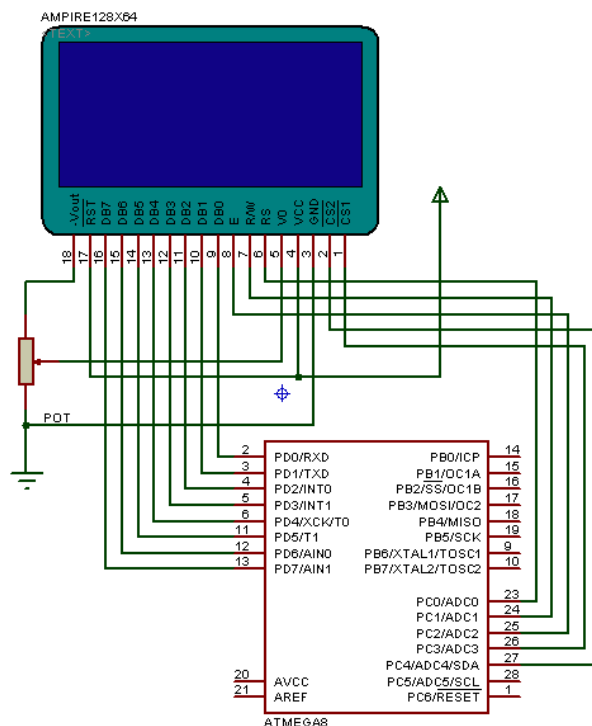
```
End
```

```
Plaatje:
```

```
$bgf "qwe.bgf"
```

مثال :

در زیر برنامه ی مربوط به راه اندازی نمایشگر 64*128 با چیپ ks108 را مشاهده میکنید :



```
$regfile = "m8def.dat"
```

' specify the used micro

```
$crystal = 8000000
```

' used crystal frequency

'some routines to control the display are in the glcdKS108.lib file

```
$lib "glcdKS108.lib"
```

'First we define that we use a graphic LCD

```
Config Graphlcd = 128 * 64sed , Dataport = Portd , Controlport = Portc , Ce = 3 , Ce2 = 4 , Cd = 0 , Rd = 1 , Reset = 6 ,
```

```
Enable = 2
```

'The dataport is the portname that is connected to the data lines of the LCD

'The controlport is the portname which pins are used to control the lcd

'CE =CS1 Chip select

'CE2=CS2 Chip select second chip

'CD=Data/instruction

'RD=Read

'RESET = reset

'ENABLE= Chip Enable

'specify the font we want to use

```

SetFont Font8x8

Lcdat 4 , 1 , "HELLO WORLD"

'end

Dim I As Byte , I2 As Byte

Showpic 0 , 0 , Pic1

Wait 4

Showpic 0 , 0 , Pic2

Wait 4

Showpic 0 , 0 , Pic3

For I = 1 To 4

For I2 = 1 To 3

Waitms 70

Glcdcmd &H3E , 1 : Glcdcmd &H3E , 2          'Off

Waitms 70

Glcdcmd &H3F , 1 : Glcdcmd &H3F , 2          'On

Next I2

Waitms 120

Next I

Wait 6

Cls Graph

'LCDAT Y , COL, value

Lcdat 4 , 1 , "0123456789..."

Wait 1

Circle(59 , 27) , 7 , 1

Wait 1

Line(0 , 0) -(127 , 63) , 1                  'make line

Wait 1

Pset 50 , 18 , 1

Pset 50 , 36 , 1

Pset 68 , 18 , 1

Pset 68 , 36 , 1

Wait 1

SetFont Font8x8

```

```

Lcdat 1 , 16 , "/*-+.><MN"

Lcdat 8 , 1 , "!@#$$%^&*()_:= "

End                                     'end program

$include "font8x8.font"

Pic1:

$bgf "1.bgf"

Pic2:

$bgf "2.bgf"

Pic3:

$bgf "3.bgf"

```

اتصال lcd گرافیکی رنگی به AVR :

اکثر lcd های گرافیکی رنگی یا lcd موبایل دارای یک چیپ داخلی هستند . این چیپ ها میتوانند همچون lcd های عادی ، داده ها و فرمان ها را به صورت سریال دریافت کرده و سپس بر روی lcd نمایش بدهند . این نوع lcd ها دارای چیپ های به شماره زیر میباشند :

چیپ pcf8833 یا 8533 : این پردازنده معمولاً در lcd گوشی های نوکیا استفاده میشود . چیپ pcf8833 برای lcd های بزرگ و چیپ دیگر برای lcd های کوچک تر استفاده میشود .

چیپ sed15xx : این چیپ معمولاً در lcd گوشی های سامسونگ و ... استفاده میشود .

چیپ های معرفی شده از پروتکل spi برای تبادل داده با میکرو یا پردازنده اصلی استفاده میکنند . در بسکام توابعی برای راه اندازی lcd نوع اول در نظر گرفته شده است . شما با استفاده از پروتکل spi میتوانید نوع دوم را نیز راه اندازی کنید . در ادامه

به بررسی PCF8533 و نحوه راه اندازی آن در بسکام خواهیم پرداخت :

این lcd دارای 10 پایه میباشد ، در زیر نام پایه ها آورده شده است :

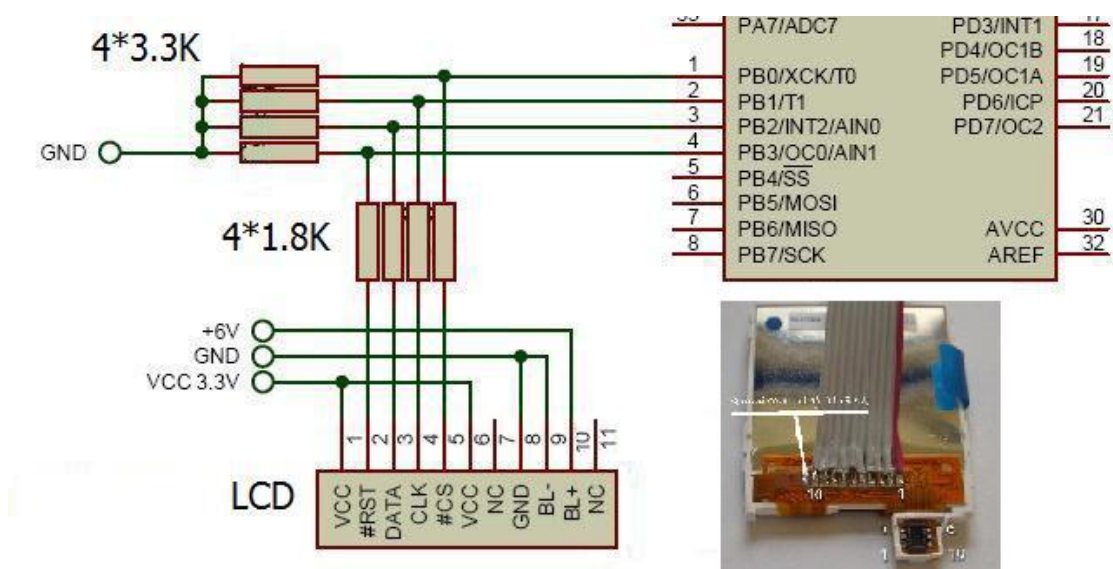
شماره پایه	نام	عملکرد	محل اتصال
1	VCC	پایه تغذیه چیپ LCD	3.3 ولت
2	Reset	پایه ریست LCD	پایه میکرو که در پیکربندی lcd تعیین میشود
3	data	پایه انتقال داده مابین lcd و ایسی پردازنده (میکرو)	پایه میکرو که در پیکربندی lcd تعیین میشود
4	clack	پالس کلاک ورودی برای lcd	پایه میکرو که در پیکربندی lcd تعیین میشود

5	Chip select (cs)	پایه انتخاب lcd (با پالس یک lcd انتخاب میشود)	پایه میکرو که در پیکربندی lcd تعیین میشود
6	vcc	پایه تغذیه سلول های lcd	3.3 ولت
7	nc	پایه بدون اتصال	
8	gnd	پایه گراند	گراند مدار
9	Backlight-	پایه تغذیه led پس زمینه (ولتاژ صفر)	گراند مدار
10	Backlight+	پایه تغذیه led پس زمینه (ولتاژ مثبت)	6 ولت
11	nc	پایه بدون اتصال	

ولتاژی که برای تحریک lcd مورد نیاز است 3.3 ولت میباشد ، همان طور که میدانید ولتاژ خروجی پایه های میکرو 5 ولت است .

در صورتی که ما ولتاژ 5 ولت را مستقیماً به lcd اعمال کنیم ، امکان آسیب رسیدن به چیپ آن وجود دارد به همین دلیل از شبکه مقاومتی زیر استفاده میشود :

توجه داشته باشید که ولتاژ 3.3 ولت برای راه اندازی lcd را باید توسط رگولاتور های ولتاژ تامین کنید (شبکه مقاومتی توانایی تامین جریان لازم را ندارد) . بدین ترتیب برای اتصال lcd به میکرو به مدار زیر نیاز است :



راه اندازی این lcd در بسکام با دستور زیر انجام میشود :

Config Graphlcd = Color , Controlport = Portx , Cs = a , Rs = b , Scl = c , Sda = d

Controlport = Portx: نام پورتهی است که lcd به آن متصل میشود ، شما میتوانید از پورت های a , b, ,c, d و... استفاده کنید

Cs = a

A نام پایه ای از میکرو است که پین chip select نمایشگر به آن متصل میشود

Rs = b

b نام پایه ای از میکرو است که پین reset نمایشگر به آن متصل میشود

Scl = c

c نام پایه ای از میکرو است که پین کلاک نمایشگر به آن متصل میشود

Sda = d

d نام پایه ای از میکرو است که پین داده نمایشگر به آن متصل میشود

مثال :

Config Graphlcd = Color , Controlport = Portc , Cs = 1 , Rs = 0 , Scl = 3 , Sda = 2

رنگ ها : هر رنگ داری یک کد خاص میباشد . شما برای نمایش دادن واژه یا اشکال هندسی (مانند خط یا دایره یا ...) بر

روی lcd باید رنگ آن را مشخص کنید ، در زیر کد رنگه های مختلف را مشاهده میکنید :

Blue	&B00000011	آبی
Yellow	&B11111100	زرد
Red	&B11100000	قرمز
Green	&B00011100	سبز
Black	&B00000000	مشکی
White	&B11111111	سفید
Brightgreen	&B00111110	سبز روشن
Darkgreen	&B00010100	سبز تیره
Darkred	&B10100000	قرمز تیره
Darkblue	&B00000010	آبی تیره
Brightblue	&B00011111	آبی روشن
Orange	&B11111000	نارنجی

دستور LINE :

این دستور به فرم زیر است ، توسط این دستور میتوانید یک خط از مختصات x1 و y1 تا مختصات x2, y2 رسم کنید

،همچنین color مشخص کننده رنگ خط میباشد :

Line(x1 , y1) -(x2 , y2) , color

دستور CIRCLE :

این دستور به فرم زیر است ، توسط این دستور میتوانید یک دایره به شعاع a و در مختصات x,y و به رنگ color رسم کنید :

Circle(x , y) , a, color

دستور PSET :

این دستور به فرم زیر است ، توسط این دستور میتوانید یک نقطه در مختصات x,y و به رنگ color ایجاد کنید :

Pset x , y , color

دستور BOX :

این دستور به فرم زیر است ، توسط این دستور میتوانید یک جعبه به طولش از x1 تا x2 عرضش از y1 تا y2 ادامه دارد و به رنگ color رسم کنید :

BOX (x1 , y1) -(x2 , y2) , color

دستور LCDAT :

این دستور به فرم زیر است ، توسط این دستور میتوانید متن خود را در مختصات x , y به رنگ color نمایش دهید :

Lcdat x , y , "your v" , color

دیگر دستورات مشابه lcd های گرافیکی و کارکتری میباشد ، مثلا دستور cls برای پاک کردن lcd به کار میرود و....

برای ایجاد رنگ های دیگر میتوانید دو رنگ را با هم ترکیب کنید ، مثلا :

Lcdat 100 , 0 , "12345678" , &B00000011 , &B11111100

نوشته ما داری رنگی بین زرد و ابی خواهد بود .

نمایش تصویر بر روی lcd گرافیکی رنگی :

برای نمایش تصویر بر روی lcd رنگی نیاز به برنامه LCD RGB-8 Converter دارید ، این برنامه را میتوانید از طریق

لینک زیر دانلود کنید :

http://www.mcselec.com/index.php?option=com_docman&task=doc_download&gid=168&Itemid=54

توجه داشته باشید که در هنگام دانلود برنامه بسکام باز باشد .

تصویر خود را بوسیله برنامه فتوشاب یا point به 256 color bmt تبدیل کنید (تصویر را توسط نرم افزار باز کنید و سپس با این پسوند ذخیره نمایید).

اکنون نرم افزار LCD RGB-8 Converter را اجرا کنید و از منوی فایل و مسیر open تصویر خود را باز کنید . مشاهده میکنید که در نرم افزار ابزار های برای رسم و تغییر تصویر وجود دارد .

بعد از ایجاد تغییرات از منوی فایل گزینه ی Save, Binary را انتخاب کنید و فایل را با نام دلخواه ذخیره کنید . تصویر شما با فرمت BGC ذخیره شد ، شما میتوانید با دستور زیر تصویر را بر روی lcd نمایش دهید :

Showpic x , y , lable

X,y مختصاتی هستند که تصویر در آنجا نمایش داده میشود . lable نام برجسی است که نام تصویر در آنجا وجود دارد .

lable:

\$bgf "name.bgc"

Lable نامی است که در بخش قبل معرفی شد و name نام تصویر میباشد . تصویر بعد از تبدیل باید در محل ذخیره برنامه ذخیره شود .

مثال :

```
$lib "lcd-pcf8833.lbx"           'special color display support
$regfile = "m8def.dat"
$crystal = 8000000             '8 MHz
'First we define that we use a graphic LCD
Config Graphlcd = Color , Controlport = Portc , Cs = 1 , Rs = 0 , Scl = 3 , Sda = 2
'here we define the colors
Const Blue = &B00000011      'predefined contants are making programming easier
Const Yellow = &B11111100
Const Red = &B11100000
Const Green = &B00011100
Const Black = &B00000000
Const White = &B11111111
Const Brightgreen = &B00111110
Const Darkgreen = &B00010100
```

```

Const Darkred = &B10100000
Const Darkblue = &B00000010
Const Brightblue = &B00011111
Const Orange = &B11111000
Cls'clear the display
Line(0 , 0) -(130 , 130) , Blue'create a cross
Line(130 , 0) -(0 , 130) , Red
Waitms 1000
Showpic 0 , 0 , Plaatje'show an RLE encoded picture
Showpic 40 , 40 , Plaatje
Waitms 1000
Lcdat 100 , 0 , "12345678" , Blue , Yellow'and show some text
Waitms 1000
Circle(30 , 30) , 10 , Blue
Waitms 1000
Box(10 , 30) -(60 , 100) , Red 'make a box
Pset 32 , 110 , Black'set some pixels
Pset 38 , 110 , Black
Pset 35 , 112 , Black
End
Plaatje:
$bgf "a.bgc"

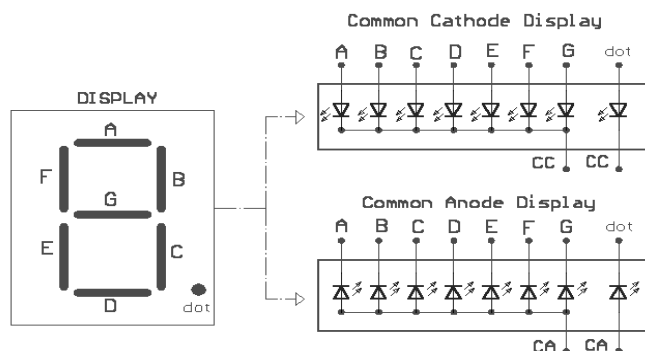
```

نمایشگر های هفت قسمتی

7 سگمنت (سون سگمنت یا نمایشگر هفت قسمتی)

یکی از قطعات پر کاربرد در مدارات میکروکنترلی است که از آن برای نمایش داده های مختلف استفاده می شود .

این قطعات از 8 عدد led که پایه های کاتد یا آند



آنها در داخل قطعه هم متصل است تشکیل می شود ، در بازار به نوع اول کاتد مشترک و به نوع دوم آنند مشترک گفته می شود . در این قطعات 7 عدد از LED ها نمایشگر ویکی ممیز است .

کار با 7 سگمنت بسیار ساده است و برای راه اندازی آن کافی است پورت متصل شده به آن را مقدار دهی کنیم . در ادامه به بررسی روش های مختلف راه اندازی این قطعات پرداخته ایم .

اصول کار 7 سگمنت : در زیر می خواهیم عدد 8 را روی 7 سگمنت کاتد نمایش دهیم:

برای نمایش عدد 8 باید همه led ها روشن شوند پس برای 7 سگمنت کاتد مشترک باید آنها را 1 کنیم (5ولت بدهیم) و برای 7 سگمنت اند مشترک باید آنها را 0 کنیم (زمین) .

مثل همیشه اولین خط برنامه معرفی کریستال و میکرو است :

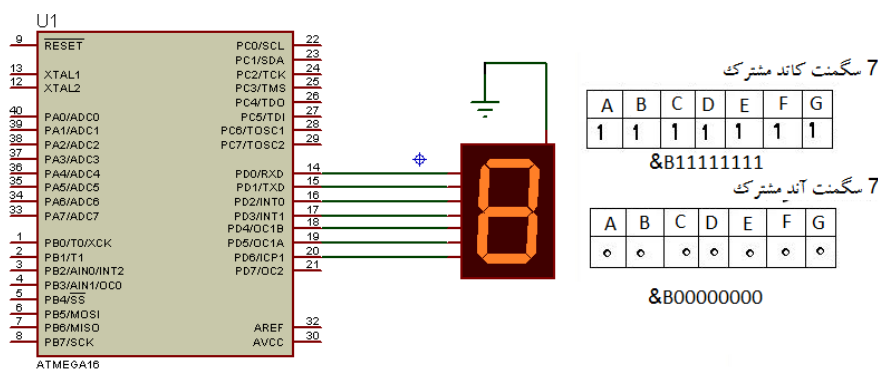
```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

در مرحله بعد پورتهی که 7 سگمنت به آن وصل میشود به عنوان خروجی تعریف میگردد (در اینجا 7 سگمنت به پورت d وصل میشود) .

```
config PORTD= OUTPUT
```

مرحله بعد قراردادن کد 7 سگمنت در پورت d است در شکل زیر نحوه ی بدست آوردن کد برای عدد 8 وجود دارد :



از آنجا که 7 سگمنت مورد استفاده کاتد مشترک است ، پس :

```
Portd = &B11111111
```

و آخرین خط برنامه دستور END است :

```
END
```

میخواهیم اعداد 0 تا 2 را با تاخیر یک ثانیه روی 7 سگمنت نمایش دهیم ، مانند برنامه قبلی کریستال و میکرو را معرفی میکنیم و پورت D را به عنوان خروجی قرار میدهیم :

```
$regfile = "m16def.dAt"
```

```
$crystal = 8000000
```

```
Config Portd = Output
```

مرحله بعد قرار دادن کد اعداد برای 7 سگمنت است ، اولین عدد 0 است برای عدد صفر تمام LED ها به جز G روشن هستند پس کد 0 برای 7 سگمنت می شود: &B1111110 پس:

```
Portd = &B1111110
```

مرحله بعد قرار دادن تاخیر زمانی است تا عدد صفر دیده شود .قرار دادن تاخیر زمانی به مدت 1 ثانیه :

```
Wait 1
```

عدد بعدی 1 است برای عدد 1، LED ها B و C روشن هستند وبقیه خاموش ،.پس کد 1 برای 7 سگمنت می شود:

```
&B1111110
```

```
Portd = &B0110000
```

مانند دو مرحله قبل ،قرار دادن تاخیر زمانی به مدت 1 ثانیه:

```
Wait 1
```

عدد بعدی 2 است، برای عدد 2، LED ها A ,B , G , E , D روشن هستند و بقیه خاموش ، پس کد 2 برای 7 سگمنت می شود

```
&B0110000
```

```
Portd = &B0110000
```

قرار دادن تاخیر زمانی به مدت 1 ثانیه :

```
Wait 1
```

و در نهایت برنامه با دستور END به پایان میرسد :

```
End
```

این برنامه فقط یک بار تکرار میشود در صورتی که شما میخواهید برنامه مدام تکرار شود کافی است یک حلقه در ابتدای برنامه (بعد از دستور Config Portd = Output دستور DO یا دیگر حلقه ها را بنویسید) و پایان حلقه را در انتهای برنامه (قبل از دستور END دستور LOOP یا دیگر حلقه ها را بنویسید) قرار دهید. شما همچنین میتوانید کد اعداد دیگر را نیز به برنامه اضافه کنید.

شما میتونید کد های باینری را به هگز تبدیل کنید و کد هگز را روی پورت قرار بدهید :

```
&b1010101=&h55
```

```
portd=&H55
```

7 سگمنت و جدول lookup

در روش اول برنامه طولانی می شود ، اما روشهای دیگری نیز وجود دارد که علاوه بر کاهش حجم برنامه آن را ساده تر میکند، یکی از این روش ها استفاده از جدول lookup هست .
توسط این جدول می توان مقدار دلخواهی را از جدولی برگرداند.

```
var = LOOKUP(value , label)
```

Label برچسب جدول و value اندیس داده دلخواه است . داده برگشتی از جدول در متغیر var قرار می گیرد . value اولین داده در جدول را برمی گرداند و value = n داده nام را از جدول برمیگرداند . تعداد اندیس ها و مقدار داده برگشتی به ترتیب نهایتاً می تواند 255 و 65535 باشد .

میخواهیم عدد 8 را با استفاده از این جدول روی 7 سگمنت نمایش دهیم :

همانند برنامه های قبلی ابتدا میکرو و کریستال معرفی میگردد و پورتی که 7 سگمنت به آن متصل میشود به عنوان خروجی تعریف می شود .

```
$regfile = "m16def.dAt"  
$crystal = 8000000  
Config Portd = Output
```

بعد از مراحل اولیه با دستور زیر مقدار جدول را در پورت d قرار میدهم :

```
Portd = Lookup(0 , Q  
End
```

Q:

Data &B1111111

برنامه شمارنده 0 تا 9 با جدول lookup و چکونکی کار آن

```
$regfile = "m16def.dat"
$crystal = 8000000
Config Portd = Output
Dim Q As Byte
Do
Waitms 1
Portd = Lookup(q , W)
Incr Q
Loop Until Q = 9
End
W:
Data &B11111110 , &B0110000 , &B1101101 , &B1111001 , &B0110011 , &B1011011 , &B1011111
, &B1110000 , &B1111111 , &B1111011
```

سه خط اول پیکربندی امکانات است.

در خط چهارم یک متغیر از جنس بایت معرفی میشود .

خط پنجم شروع حلقه do-loop است.

در خط ششم یک تاخیر یک ثانیه ایجاد میشود .

در خط هفتم مقدار q ام خوانده شده از جدول w در پورت d قرار میگیرید ، در اینجا چون مقدار q=0 است پس اولین عدد

جدول w در پورت d قرار میگیرد (&b11111110) (که بر روی 7 سگمت عدد صفر را نشان میدهد) .

در خط هشتم یک واحد به متغیر معرفی شده اضافه میگردد (0+1=1) (q=q+1) .

خط نهم پایان حلقه do-loop میباشد که یک شرط نیز در آن به کار برده شده ، این شرط میگوید هرگاه مقدار q=9 شد حلقه

تمام شود و برنامه از خط بعد از حلقه شروع شود (در اینجا بعد از حلقه پایان برنامه است) .

اما در حالتی که q مخالف 9 است برنامه به خط do پرش میکند در آنجا یک واحد دیگر به q افزوده میشود (q=2) و بعد

از تاخیر یک ثانیه دومین عدد از جدول lookup در پورت d گذاشته میشود (&b0110000) (که عدد یک را روی 7 سگمت

نمایش میدهد) .

برنامه تا آنجا اجرا میشود که مقدار q=9 شده و برنامه پایان یابد .

خط دهم ، خط پایان برنامه است .

خط های یازده و دوازده جدول lookup هستند که مقادیر در آنها قرار میگیرد.

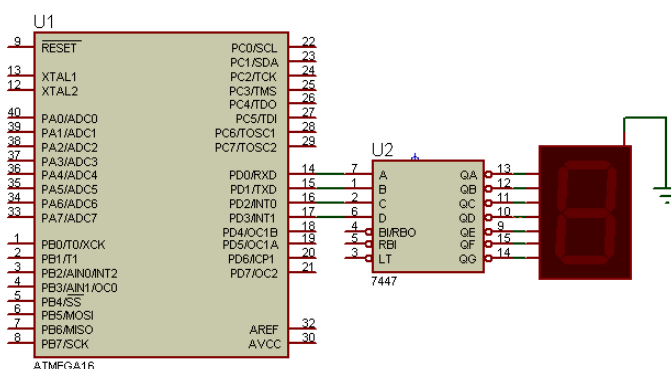
استفاده از مبدل BCD به کد 7 سگمنت:

برای راه اندازی 7 سگمنت راه دیگری هم وجود دارد که نسبت به دو روش قبلی ساده تر است . آی سی 7447 یک ایسی

مبدل کد bcd به کد 7 سگمنت است . برای مثال اگر به وردی این آیسی کد bcd عدد 5 را بدهید، در خروجی کد 7

سگمنت مربوط به عدد 5 ظاهر میشود(در این روش مقدار عدد مستقیما روی پورت قرار میگیرد) .

```
$regfile = "m16def.dAt"
$crystal = 8000000
Config Portd = Output
Dim Q As Byte
W:
Do
Portd = Q
Incr Q
Waitms 500
Loop Until Q = 9
Q = 0
jmp w
End
```



7 سگمنت چند تایی (مولتی پلکس):

در بخش قبلی با روش های راه اندازی 7 سگمنت تکی آشنا شدید ، در این قسمت می خواهیم چند 7 سگمنت را به میکرو

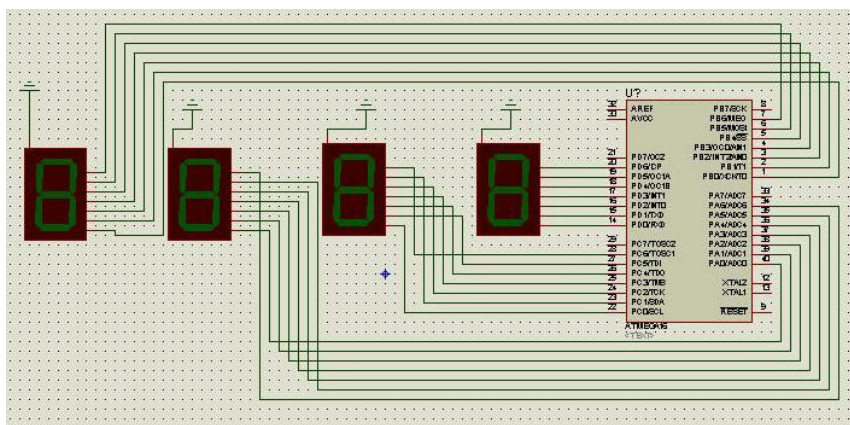
متصل کنیم . فرض کنید می خواهیم عدد 8321 را روی چهار عدد 7 سگمنت نمایش دهیم برای این کار چندین راه وجود

دارد:

1- چهار 7 سگمنت به طور جداگانه به پورت های میکرو وصل شود و هر عدد روی هر کدام از آنها نمایش داده شود :

مانند برنامه و مدار زیر :

```
$regfile = "m16def.dAt"
$crystal = 8000000
Config Porta = Output
Config Portb = Output
Config Portc = Output
Config Portd = Output
Portd = &B0110000
```

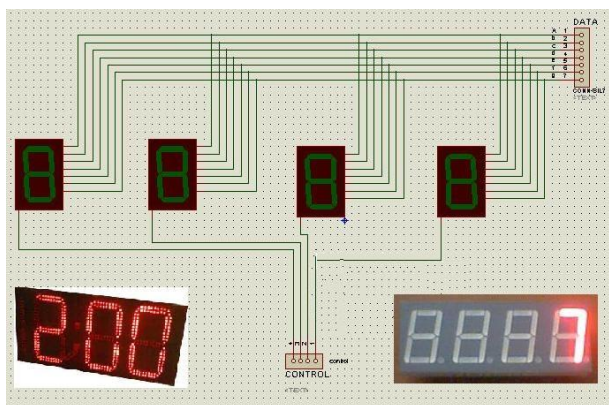


```
Portb = &B1111111
Porta = &B1111001
Portc = &B1101101
End
```

در روش بالا علاوه بر اینکه برای تقویت جریان 7 سگمنت ها نیاز به تقویت کننده میباشد ، کلیه پورت های میکرو اشغال میشود و بیشتر از چهار 7 سگمنت نمیتوان به میکرو متصل کرد ، بنابراین از آن به ندرت استفاده میشود .

2- روش رفرشی یا تازه سازی:

در این روش خطوط دیتای 7 سگمنتها به هم متصل گردیده فقط یک پورت برای دیتای 7 سگمنتها (در صورت استفاده از 7447 ، 4 پایه) و n پایه برای کنترل n سون سگمنت مورد استفاده قرار میگیرد . مانند شکل زیر : (مدار داخلی کلیه ی 7 سگمنت های چند تایی مانند زیر است)



این روش براساس خطای چشم انسان کار میکند (در صورتی که 25 تصویر (یا بیشتر) پشت سرهم ، در یک ثانیه پخش شود انسان آنها را پیوسته می بیند)، بدین صورت که در هر واحد زمانی فقط یکی از 7 سگمنت ها روشن است و کد مربوط به آن ارسال می شود ، این کار با سرعت زیادی انجام می شود ، بطوری که افراد متوجه چشمک زدن اعداد نمیشوند .

برای مثال می خواهیم عدد 8321 رو روی 7 سگمنت مالتی پلکس نشان دهیم :

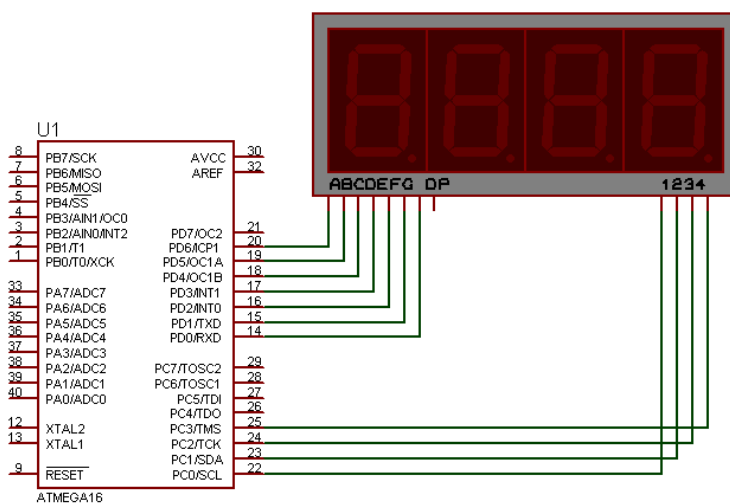
اول عدد 8 را روی 7 سگمنت اول نشان می دهیم ، برای این کار باید پایه گراند آنرا (که به میکرو متصل است) 0 کنیم و کد مربوط به عدد 8 برای 7 سگمنت را بفرستیم (پایه گراند بقیه 7 سگمنتها 1 می گردند) .

بعد عدد 3 را روی 7 سگمنت دوم نشان می دهیم ، برای این کار پایه گراند آنرا (که به میکرو متصل است) 0 میکنیم و کد مربوط به عدد 3 برای 7 سگمنت را می فرستیم (پایه گراند بقیه 7 سگمنتها 1 می گردند) .

سپس عدد 2 را روی 7 سگمنت سوم نشان می دهیم ، برای این کار پایه گراند آنرا (که به میکرو متصل است) 0 میکنیم و کد مربوط به عدد 2 برای 7 سگمنت را می فرستیم (پایه گراند بقیه 7 سگمنتها 1 می گردند).

و در آخرین مرحله عدد 1 را روی 7 سگمنت چهارم نشان می دهیم ، برای این کار پایه گراند آنرا (که به میکرو متصل است) 0 میکنیم و کد مربوط به عدد 1 برای 7 سگمنت را می فرستیم (پایه گراند بقیه 7 سگمنتها 1 می گردند). و کلیه موارد بالا مدام تکرار می گردند ، مانند برنامه و مدار زیر :

```
$regfile = "m16def.dAt"
$crystal = 8000000
Config Portc = Output
Config Portd = Output
Do
Portd = &B00000000
Waitms 1
Reset Portc.3 : Reset Portc.2 : Reset Portc.0 : Set Portc.1
Portd = &B00001110
Waitms 1
Reset Portc.0 : Reset Portc.3 : Reset Portc.1 : Set Portc.2
Portd = &B0010010
Waitms 1
Reset Portc.1 : Reset Portc.0 : Reset Portc.2
Set Portc.3 : Portd = &B1001111
Waitms 1
Reset Portc.1 : Reset Portc.2 : Reset Portc.3 : Set Portc.0
Loop
End
```



مانند 7 سگمنت تکی ، میتوانید با استفاده از جدول LOOKUP برنامه را ساده و کمتر کنید:

برای مثال می خواهیم عدد 8321 رو روی 4 عدد 7 سگمنت نشان دهیم ، مانند همه برنامه های خطوط اول معرفی میکرو و کریستال پیکر بندی امکانات و معرفی مغیر ها است

```
$regfile = "m16def.dAt"
$crystal = 8000000
Config Portc = Output
Config Portd = Output
```

و بعد برنامه:

```
Do
Reset Portc.1 : Reset Portc.2 : Reset Portc.3 : Set Portc.0
Portd = Lookup(0 , W)
Waitms 300
Reset Portc.1 : Reset Portc.3 : Reset Portc.0 : Set Portc.1
Portd = Lookup(1 , W)
Waitms 300
Reset Portc.3 : Reset Portc.2 : Reset Portc.1 : Set Portc.2
Portd = Lookup(2 , W)
Waitms 300
Reset Portc.1 : Reset Portc.3 : Reset Portc.2 : Set Portc.3
Portd = Lookup(3 , W)
Waitms 300
Loop
```

و پایان برنامه و جدول lookup :

```
End
W:
Data &B00000000 , &B00001110 , &B0010010 , &B1001111
```

(مدار این برنامه مانند مدار مثال قبلی میا شد) .

تحلیل برنامه :

بعد از خط do پین های 0,1,2,3,c.1,c.2,c.3 صفر میشوند و پین 0,c.0 یک می شود تا 7 سگمنت اول روشن شود وبقیه خاموش شوند(در این پروژه لز 7 سگمنت اند مشترک استفاده شده است) .

بعد از روشن شدن 7 سگمنت اول کد مربوط به عدد هشت (&B00000000) توسط جدول lookup در پورت d گذاشته می شود ،بعد یک تاخیر زمانی کوتاه (1 میلی ثانیه)(در اینجا برای دیده شدن عمل رفرش مقدار تاخیر 300 میلی ثانیه گرفته شده است) پین های 0,c.1,c.2,c.3 صفر میشوند و پین 1,c.1 یک می شود تا 7 سگمنت دوم روشن شود وبقیه خاموش شوند .

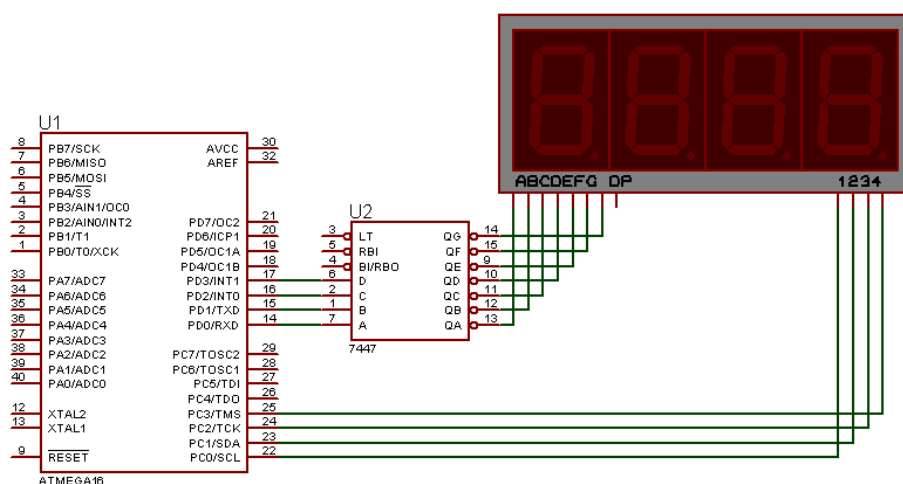
بعد از روشن شدن 7 سگمنت دوم کد مربوط به عدد سه (&B0000110) توسط جدول lookup در پورت d گذاشته می شود و این کار برای اعداد 2 و 1 نیز تکرار میگردد. وقتی برنامه به خط loop رسید به do پرش میکند این مراحل مدام تکرار میگردد.

❖ استفاده از ای سی 7447 برای راه انداز 7 سگمنت چندتایی :

استفاده از ای سی 7447 ساده ترین روش برای راه اندازی 7 سگمنتها می باشد، چون علاوه بر این که تعداد پایه های کمتری از میکرو را اشغال میکند، باعث ساده و کمتر شدن برنامه میگردد.

می خواهیم عدد 8321 رو با استفاده از 7447 ری 7 سگمنت نمایش بدیم

```
$regfile = "m16def.dAt"
$crystal = 8000000
Config Portc = Output
Config Portd = Output
Do
Reset Portc.1 : Reset Portc.2 : Reset Portc.3 : Set Portc.0
Portd = 8
Waitms 1
Reset Portc.1 : Reset Portc.3 : Reset Portc.0 : Set Portc.1
Portd = 3
Waitms 1
Reset Portc.3 : Reset Portc.2 : Reset Portc.1 : Set Portc.2
Portd = 2
Waitms 1
Reset Portc.1 : Reset Portc.3 : Reset Portc.2 : Set Portc.3
Portd = 1
Waitms 1
Loop End
```



همانطور که مشاهده میکنید استفاده از آیسی 7447 باعث سادگی برنامه شد. شما میتوانید 7 سگمنت های آنند مشترک را با

آیسی 7448 به سادگی راه اندازی کنید .

مبدل آنالوگ به دیجیتال (ADC):

گاهی نیاز است که یک کمیت بیرونی (مانند دما و شدت صدا و شدت نور و...) اندازه گیری شود، برای اینکار از وسیله ای به نام سنسور استفاده میشود. سنسور ها مقدار یک کمیت آنالوگ را به ولتاژ یا جریان تبدیل میکند، سپس این ولتاژ آنالوگ به مبدل آنالوگ به دیجیتال میکرو داده میشود و مبدل مقدار ولتاژ را به کمیت دیجیتال متناظر تبدیل میکند و در نهایت این مقدار دیجیتال با اعمال ریاضی به مقدار عددی متناظر تبدیل شده و روی lcd یا 7 سگمنت نمایش داده میشود.

حداکثر ولتاژی که مبدل آنالوگ به دیجیتال، که از این به بعد به آن ADC میگوئیم میتواند اندازه بگیرد برابر با V_{CC} است و اگر ولتاژ اعمالی از V_{CC} بیشتر شود ADC آسیب می بیند (معمولا بیشترین ولتاژ ورودی که به adc اعمال میکنند $V_{CC}-0.5$ ولت و کمترین ولتاژ اعمالی برابر با 0 است). adc به ازای ولتاژ 5 ولت عدد 1023 و به ازای صفر ولت عدد صفر را در متغیر مربوطه قرار می دهد.

Adc با دستور زیر راه اندازی میشود:

Config adc = single/free, PRESCALER = AUTO, REFERENCE = opt

گزینه های single/free: در حالتی که single انتخاب شود مقدار دیجیتال سیگنال آنالوگ توسط دستور getadc در یک متغیر از جنس word ریخته میشود و در حالتی که free انتخاب شود مقدار دیجیتال سیگنال آنالوگ در رجیستر مربوط به adc ریخته میشود.

PRESCALER: این گزینه کلاک adc را مشخص میکند و در حالتی که AUTO انتخاب شود کامپایلر با توجه به کریستال انتخاب شده بهترین کلاک را در نظر میگیرد، موارد دیگر برای کلاک عبارتند از 2و4و8و15و32و64و128 است که به جای گزینه AUTO نوشته میشود.

REFERENCE: در صورتی که بخواهید از یک ولتاژ مرجع استفاده کنید این گزینه را بنویسید (در صورت عدم استفاده از این دستور ولتاژ مرجع برابر با ولتاژ اعمال شده به پایه AREF است و نیازی به نوشتن این دستور نیست).
opt میتواند یکی از موارد زیر باشد:

Off: در این حالت ولتاژ مرجع داخلی خاموش شده و از ولتاژ مرجع بر روی پایه aref استفاده میشود.

AVCC: در این حالت ولتاژ پایه avcc به عنوان ولتاژ مرجع در نظر گرفته میشود.

Internal: در این حالت از ولتاژ مرجع داخلی 2.65 ولت استفاده میشود .

برای حالت های AVCC و Internal پایه ی AREF باید با یک خازن 100 نانو فاراد به زمین متصل شود .

ولتاژ مرجع مبنای تبدیل داده ی ADC خواهد بود ، به عنوان مثال هنگامی که ولتاژ مرجع برابر با 5 ولت است ، ADC به

ازای ولتاژ اعمال شده به کانال دلخواه عدد 1023 و به ازای ولتاژ 0 ولت عدد 0 را برمیگرداند ، اگر در همین برنامه از

ولتاژ اعمال شده به پایه ی AREF به عنوان ولتاژ مرجع استفاده شود ، ADC به ازای ولتاژ برابر با AREF عدد 1023 و به ازای

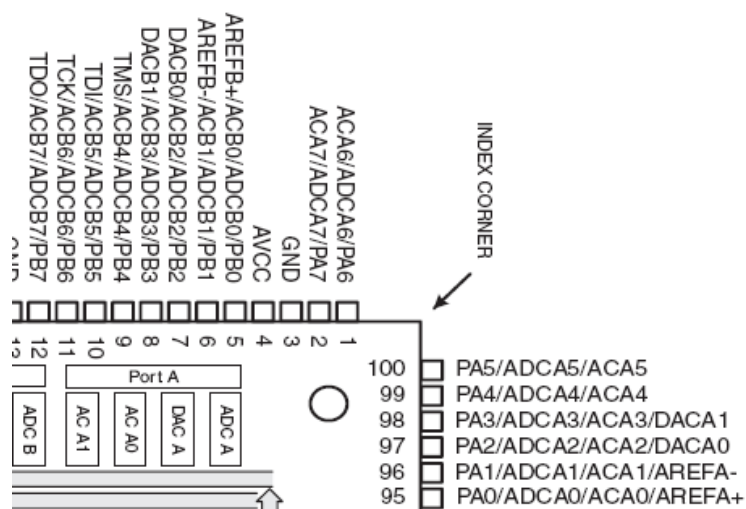
ولتاژ 0 ولت عدد 0 را برمیگرداند .

در سری ATXMEGA راه اندازی ADC مقداری متفاوت میباشد . در این خانواده تعداد 16 کانال ADC وجود دارد که در دو

گروه A و B دسته بندی شده اند :

- Two ADCs with 12-bit resolution
- Signed and Unsigned conversions
- 8 single ended inputs for each ADC
- 4 internal inputs:
 - Integrated Temperature Sensor
 - DAC Output
 - VCC voltage divided by 10
 - Bandgap voltage
- Software selectable gain of 2, 4, 8, 16, 32 or 64
- Software selectable resolution of 8- or 12-bit.

در این خانواده دو واحد ADC با دقت 12 بیت وجود دارد ، این دو واحد دارای 8 کانال ورودی میباشد :



ADC در این خانواده با مجموعه دستورات زیر راه اندازی میشود :

CONFIG ADCA | ADCB = mode, CONVMODE=sign, RESOLUTION=res, DMA=dma,

REFERENCE=ref, EVENT_MODE=evt, EVENT_CHANNEL=evtchan, PRESCALER=pre, BANDGAP=gap, TEMPREF=tref,

SWEEP=sweep, CH0_GAIN=gain, CH0_INP= inp, MUX0=mux, CH1_GAIN=gain, CH1_INP= inp, MUX1=mux ,

CH2_GAIN=gain, CH2_INP= inp, MUX2=mux, CH3_GAIN=gain, CH3_INP= inp, MUX3=mux

Mode: مد کاری adc میباشد که میتوانید به جای آن یکی از موارد SINGLE یا FREE را قرار دهید .

CONVMODE=sign: با استفاده از این دستور میتوانید علامت داده ی تبدیل شده را تعیین کنید، با قرار دادن دستور SIGNED

به جای sign باید داده ی تبدیل شده را در یک متغیر از جنس integer ذخیره گردد، در این حالت اگر ولتاژ ورودی

منفی باشد علامت آن در مقدار موجود در متغیر لحاظ میشود . با استفاده از دستور UNSIGNED داده ی تبدیل شده همیشه

مثبت خواهد بود و شما میتوانید آن را در یک متغیر از جنس WORD ذخیره کنید .

RESOLUTION=res: مقدار دقت ADC که میتواند یکی از مقادیر 8BIT یا 12BIT یا LEFT12BIT باشد به جایی res قرار داده

میشود . دقت ADC با سرعت آن رابطه عکس دارد، با بالا رفتن دقت سرعت نمونه برداری کم میشود . (3.5 us برای 12-

bit و 2.5 us برای 8-bit).

DMA=dma : Direct Memory Access یا DMA واحدی (یا مسیری) در سری xmega میباشد که اجازه ی دست یابی مستقیم

امکانات جانبی به CPU را میدهد . ADC یا DAC در این خانواده میتوانند از طریق این باس به صورت مستقیم با cpu ارتباط

برقرار کرده و داده ی تبدیل شده را برای آن ارسال کنند، بدین ترتیب سرعت تبدیل در adc به 2 Msps خواهد رسید و

قطعا هر چقدر تعداد این مسیر ها بیشتر باشد، داده سریع تر منتقل خواهد شد . با استفاده از این دستور میتوانید مشخص

کنید adc از چند کانال dma استفاده کند :

- OFF (no DMA)

- CH01 (channel 0 + 1)

- CH012 (channel 0 + 1 + 2)

- CH0123 (channel 0 + 1 + 2 + 3)

REFERENCE=ref : مقدار ولتاژ مرجع استفاده شده در adc با این دستور مشخص میشود :

- INT1V. For internal 1V reference

- INTVCC. For internal voltage divided by 1.6

- AREFA. External reference from AREF pin on PORT A.

- AREFB. External reference from AREF pin on PORT B.

EVENT_MODE=evt : با استفاده از این دستور در بخش "پیکربندی Event System" آشنا خواهیم شد .

EVENT_CHANNEL=evtchan : با استفاده از این دستور در بخش "پیکربندی Event System" آشنا خواهیم شد .

PRESCALER=pre : این گزینه کلاک adc را مشخص میکند ، موارد مجاز برای کلاک عبارتند از 2 و 4 و 8 و 15 و 32 و 64 و 128

است که به جای گزینه pre نوشته میشود ، فرکانس کلاک ADC برابر با کلاک اصلی سیستم تقسیم بر عدد انتخاب شده است .

BANDGAP=gap : BANDGAP یک ولتاژ مرجع داخلی با مقدار 1 ولت است که میتوان از آن به عنوان ولتاژ مرجع مبدل

آنالوگ به دیجیتال ، مبدل دیجیتال به آنالوگ ، مقایسه کننده ی آنالوگ و... استفاده کرد . با قرار دادن دستور ENABLED

به جای gap میتوانید این ولتاژ را فعال یا غیر فعال کنید .

TEMPREF=tref : میکرو کنترلر های سری atxmega دارای یک واحد کالیراتور داخلی هستند ، این واحد میتواند با اندازه

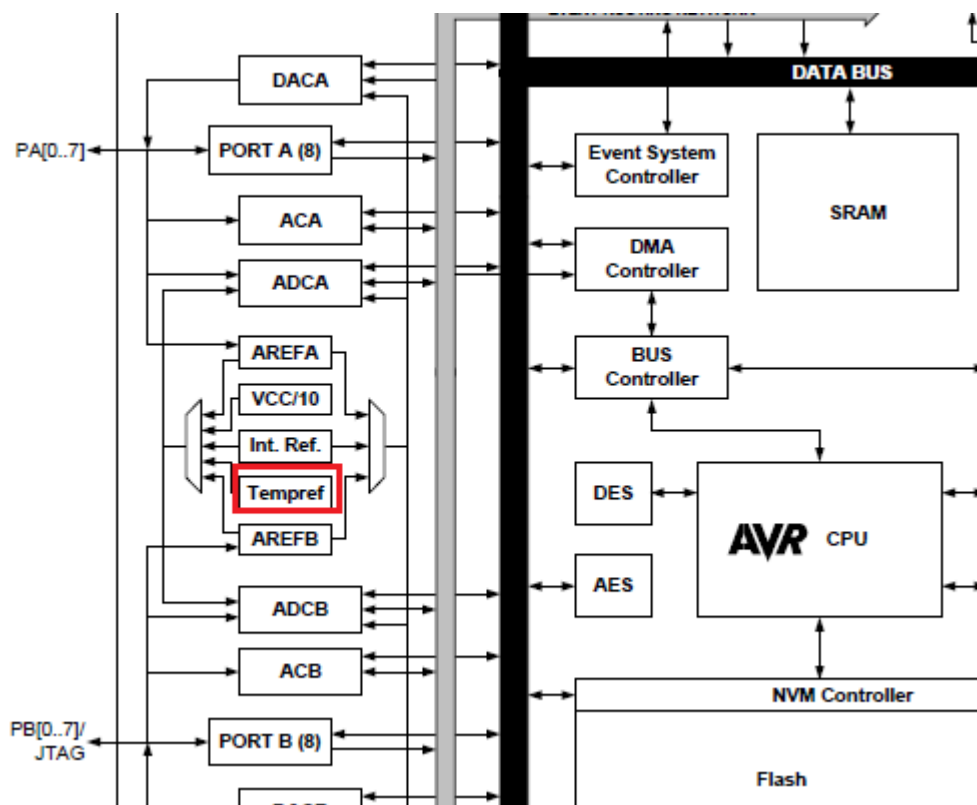
گیری مقدار دما و ولتاژ تغذیه ، ظرابی را در داده های تبدیل شده توسط واحد های مختلف (نظیر ADC یا DAC یا ...

) وارد کند .

با دستور قرار دادن دستور ENABLED یا DISABLED به جای tref میتوانید داده تبدیل شده توسط واحد ADC را در هنگام

افزایش دمای میکرو کنترلر کالیره کنید ، افزایش دما ممکن است باعث ایجاد تغییر جزئی در داده ی تبدیل شده شود

که با این دستور میتوان این تغییرات را حذف کرد .



SWEEP=sweep : با استفاده از این دستور در بخش "پیکربندی Event System" آشنا خواهیم شد.

CH0_GAIN=gain و CH1_GAIN=gain و CH2_GAIN=gain و CH3_GAIN=gain

CH0_INP= inp و CH1_INP= inp و CH2_INP= inp و CH3_INP= inp

MUX0=mux, و MUX1=mux, و MUX2=mux, و MUX3=mux,

همان طور که قبلا نیز اشاره شد، میکرو کنترلر های سری ATXMEGA دارای یک یا دو عدد مبدل آنالوگ به دیجیتال 12 بیتی میباشند که این کانال ها به صورت مالتی پلکس شده از طریق 8 عدد ورودی که معمولا بر روی پورت A (برای ADCA) و پورت B (برای ADCB) قرار گرفته اند، در دسترس کاربر می باشد. این مبدل ها میتواند به صورت single-ended و differential (یک جهت و تفاضلی) مورد استفاده قرار گیرد.

بعد از راه اندازی adc نوبت به استفاده از آن است برای اینکار با دستور `adc_start adc` روشن شده و شروع به نمونه برداری از سیگنال آنالوگ موجود بر روی پایه اش میکند و آن را به مقدار دیجیتال تغییر میدهد، این مقدار دیجیتال با دستور زیر در یک متغیر از جنس `word` ریخته میشود

```
var = GETADC(channel)
```

`var` یک متغیر از جنس `word` میباشد

```
var = Getadc(adcx , v)
```

برای خواندن adc در سری `xmega` از دستور بالا استفاده میشود .

`Var` یک متغیر از جنس `word` یا `integer` میباشد ، `adcx` میتواند `adca` یا `adcb` باشد .

`Channel` : شماره adc است که سیگنال آنالوگ به آن اعمال شده است .

مثال:

```
$regfile = "m8def.dat"
```

```
$crystal = 8000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Pinb.2 , Db5 = Pinb.3 , Db6 = Pinb.4 , Db7 = Pinb.5 , Rs = Pinb.0 , E = Pinb.1
```

```
Config Adc = Single , Prescaler = Auto
```

```
Dim A As Word
```

```
Start adc
```

```
Q:
```

```
A = Getadc(1)
```

```
Locate 1 , 1
```

```
Lcd A
```

```
Goto Q
```

```
End
```

مثالی دیگر:

```
$regfile = "xm128a1def.dat"
```



```
Config Osc = Enabled , Pllosc = Enabled , , Startup = Xtal_256clk
```

```
Config Sysclock = 2mhz , Prescalea = 2 , Prescalebc = 1_2
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Pine.2 , Db5 = Pine.3 , Db6 = Pine.4 , Db7 = Pine.5 , Rs = Pine.0 , E = Pine.1
```

```
Config Adca = Single , Convmode = Unsigned , Resolution = 12bit , Dma = Off , Reference = Int1v , Event_mode = None
, Prescaler = 32 , Ch0_gain = 1 , Ch0_inp = Single_ended , Mux0 = 0 'you can setup other channels as well
```

```
Dim A As Word
```

```
Start adc
```

```
Q:
```

```
a = Getadc(adca , 0)
```

```
Locate 1 , 1
```

```
Lcd A
```

```
Goto Q
```

```
End
```

در خط های اول lcd و adc پیکر بندی شده است در خط دهم مقدار انالوگ داده شده به پایه 24 میکرو (portc.1) بعد از

تبدیل به مقدار دیجیتال در متغیر a ریخته میشود و سپس این متغیر در سطر اول و ستون اول lcd به نمایش در می آید .

```
$regfile = "m16def.dat"
```

```
$crystal = 12000000
```

```
Config Lcd = 16 * 4
```

```
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
```

```
Config Adc = Single , Prescaler = Auto
```

```
Dim A As Word , B As Word , C As Word , D As Word , E As Word , F As Word , G As Word , H As Word
```

```
Cls
```

```
Q:
```

```
A = Getadc(0) : Locate 1 , 1 : Lcd A
```

```
B = Getadc(1) : Locate 1 , 8 : Lcd B
```

```
C = Getadc(2) : Locate 2 , 1 : Lcd C
```

```
D = Getadc(3) : Locate 2 , 8 : Lcd D
```

```
E = Getadc(4) : Locate 3 , 1 : Lcd E
```

```
F = Getadc(5) : Locate 3 , 8 : Lcd F
```

```
G = Getadc(6) : Locate 4 , 1 : Lcd G
```

```
H = Getadc(7) : Locate 4 , 8 : Lcd H
```

Goto Q

End

در این مثال از میکرو مگا 16 و lcd 16*4 استفاده شده است ، میکرو مگا 16 دارای 8 کانال adc میباشد ، که در این مثال از همه adc های این میکرو استفاده شده است .

مثال : در این مثال از ولتاژ مرجع داخلی 2.56 ولت استفاده شده است ، در این حالت ADC به ازای ولتاژ 2.56 ولت عدد 1023 و به ازای ولتاژ 0 ولت عدد 0 را برمیگرداند . به ازای ولتاژ های بیشتر از 2.56 ولت عدد 1023 همچنان ثابت خواهد بود .

```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
```

```
Config Adc = Single , Prescaler = Auto , Reference = Internal
```

```
Dim A As Word , B As Word ,
```

```
Cls
```

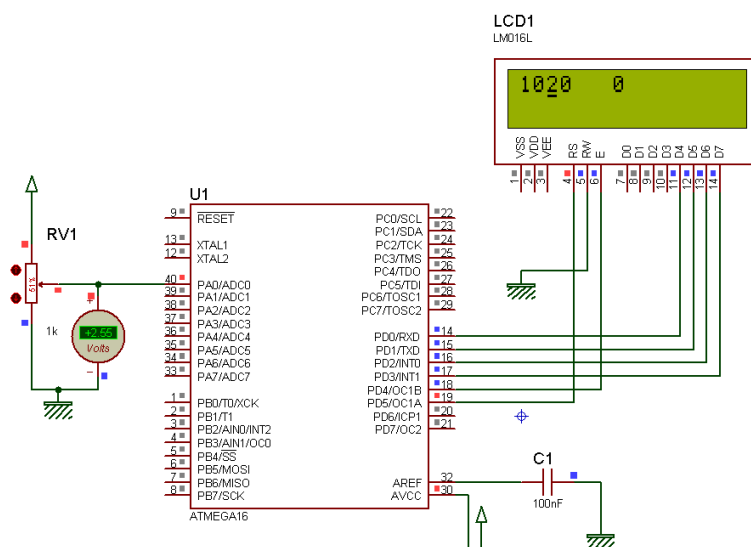
```
Q:
```

```
A = Getadc(0) : Locate 1 , 1 : Lcd A
```

```
B = Getadc(1) : Locate 1 , 8 : Lcd B
```

```
Goto Q
```

```
End
```



مقدار خازن C1 ممکن است با توجه به سرعت نمونه برداری و مقدار ولتاژ تغذیه تغییر کند ، با مراجعه به دیتاشیت میکرو کنترلر میتوانید اطلاعات بیشتری در این باره بدست آورید .

اندازه گیری ولتاژ های منفی زیاد مثبت و... با ADC (کار با OP-AMP):

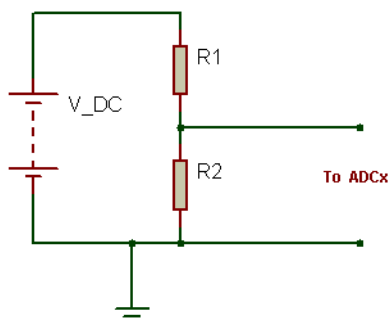
مبدل آنالوگ به دیجیتال میکرو توانای اندازه گیری ولتاژ 0 تا Vref (ولتاژ مرجع) را دارد ، این مبدل این محدوده ولتاژ را با دقت بالا اندازه میگیرد اما توانایی اندازه گیری ولتاژ های بیشتر از این مقدار را ندارد ، همچنین دقت در اندازه گیری ولتاژ های کمتر از 1 ولت (ولتاژ های اعشاری و خیلی کم) بسیار پایین است .

در میکرو کنترلر های سری tiny و 90s و atmega حداکثر ولتاژ مرجع (Vref) برابر با 5 ولت در سری ATXMEGA برابر با 3.3 ولت است .

در زیر نحوه ی اندازه گیری رنج های مختلفی ولتاژی با ADC میکرو کنترلر بیان شده است :

اندازه گیری ولتاژ های بیشتر از 5 ولت:

برای اندازه گیری ولتاژ های بیشتر از 5 ولت ، ابتدا باید با استفاده از تقسیم ولتاژ مقاومتی ، آن را به رنج استاندارد ورودی



adc (0 تا VREF ولت) کاهش داد . در این روش ولتاژ سر مشترک مقاومت ها باید به adc اعمال گردد ، این ولتاژ از رابطه ی زیر محاسبه میشود :

$$Va = Vin * \frac{R2}{R1 + R2}$$

در تبدیل آنالوگ به دیجیتال Va (ولتاژ نقطه ی مشترک) برابر با vref ولت و Vin بیشترین ولتاژ ورودی میباشد ، بنابراین با

داشتن یکی از مقاومت ها مقاومت دیگر به سادگی قابل محاسبه است :

با داشتن R2:

$$R1 = \frac{R2(Vin - Va)}{Va}$$

با داشتن R1:

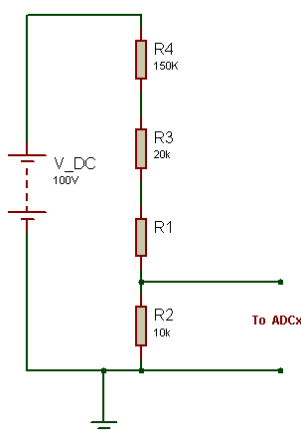
$$R2 = \frac{Va * R1}{(Vin - Va)}$$

مثال:

میخواهیم ولتاژ 100 ولت را با مبدل آنالوگ به دیجیتال اندازه بگیریم ، اولین قدم محاسبه دو مقاومت برای تبدیل 100 ولت به 5 ولت است (با فرض اینکه ولتاژ مرجع برابر با 5 ولت است)، ابتدا با استفاده از دو فرمول بالا مقدار مقاومت مجهول را بدست میاوریم (به یکی از مقاومت ها مقدار داده و مقاومت دیگر را حساب میکنیم ، مقدار R2 را 10 کیلو در نظر میگیریم و مقدار R1 را طبق فرمول بدست میاوریم)

$$R1 = \frac{10(100 - 5)}{5} = 190K$$

از آنجا که 190 کیلو اهم در رنج استاندارد نیست ، میتوان با سری کردن چند مقاومت آن را ایجاد کرد :



حداکثر مقدار مجاز برای ADC میکرو کنترلر 5 ولت است ، افزایش ولتاژ ممکن است باعث سوختن ADC و میکرو کنترلر شود.

نوشتن برنامه:

برنامه زیر مقدار آنالوگ موجود بر روی پایه ADC0 را به دیجیتال تبدیل میکند ، همانطور که در عمل یا شبیه سازی مشاهده می کنید ، به ازای ولتاژ صفر ولت ، بر روی LCD عدد 0 و به ازای ولتاژ 5 ولت عدد 1023 نمایش داده می شود.

(با نوشتن دستور \$sim در انتهای برنامه میتوانید آن را توسط شبیه ساز بسکام شبیه سازی کنید ، نحوه شبیه سازی در ضمیمه

های قبلی گفته شد):

```
$regfile = "m16def.dat" : $crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.1 , Db5 = Pinc.2 , Db6 = Pinc.3 , Db7 = Pinc.4 , Rs =
```

```
Pind.3 , E = Pind.2
Config Adc = Single , Prescaler = Auto
Dim A As Word
Do
A = Getadc(0)
Locate 1 , 1 : Lcd A
Loop
End
```

در مدار ولتاژ ورودی برابر با 100 ولت است و به ازای 100 ولت ورودی که آن را با مقاومت به 5 ولت کاهش دادیم باید عدد 100 روی lcd نمایش داده شود . با استفاده از فرمول زیر میتوان مقدار مناسب با ولتاژ ورودی را بدست آورد (adc به ازای ولتاژ بین صفر تا 5 ولت مقدار 0 تا 1023 را در متغیر مورد نظر (متغیری که با دستور getadc مقدار دیجیتال در آن ریخته میشود) میریزد):

$$Vin = \frac{1023}{x} \Rightarrow x = 1023/Vin$$

با تقسیم کردن x به 1023 خروجی مورد نیاز به دست می آید ، برنامه اصلاح شده را در زیر آورده شده است:

```
$regfile = "m16def.dat" : $crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.1 , Db5 = Pinc.2 , Db6 = Pinc.3 , Db7 = Pinc.4 , Rs =
Pind.3 , E = Pind.2
Config Adc = Single , Prescaler = Auto
Dim A As Word
Do
A = Getadc(0)
Locate 1 , 1 : Lcd A
A = A / 10.23
Locate 2 , 1 : Lcd A
Loop
End
```

در برنامه بالا ولتاژ ورودی 100 ولت است ، با تقسیم کردن 1023 به 100 ، و تقسیم کردن 1023 ر در برنامه (A = A / 10.23)، در شبیه سازی مشاهده می کنید که به ازای ولتاژ 5 ولت بر روی پایه ی adc مقدار 102 روی lcd نمایش داده میشود. اختلاف موجود به این دلیل است که متغیر a از جنس word است ، word اعداد صحیح بین 0 تا 65535 را شامل میشود و فقط میتواند در اعداد صحیح ضرب یا به اعداد صحیح تقسیم شود ، در برنامه عدد 10.23 به 10 گرد شده و سپس متغیر a به 10 تقسیم میشود ، در برنامه زیر این مشکل نیز رفع شده است:

```

$regfile = "m16def.dat" : $crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.1 , Db5 = Pinc.2 , Db6 = Pinc.3 , Db7 = Pinc.4 , Rs = Pind.3 , E = Pind.2
Config Adc = Single , Prescaler = Auto
Dim A As Word , B As Single
Config Single = Scientific , Digits = 1
Do
A = Getadc(0)
Locate 1 , 1 : Lcd A
B = A : B = B / 10.23
Locate 2 , 1 : Lcd B
Loop
End

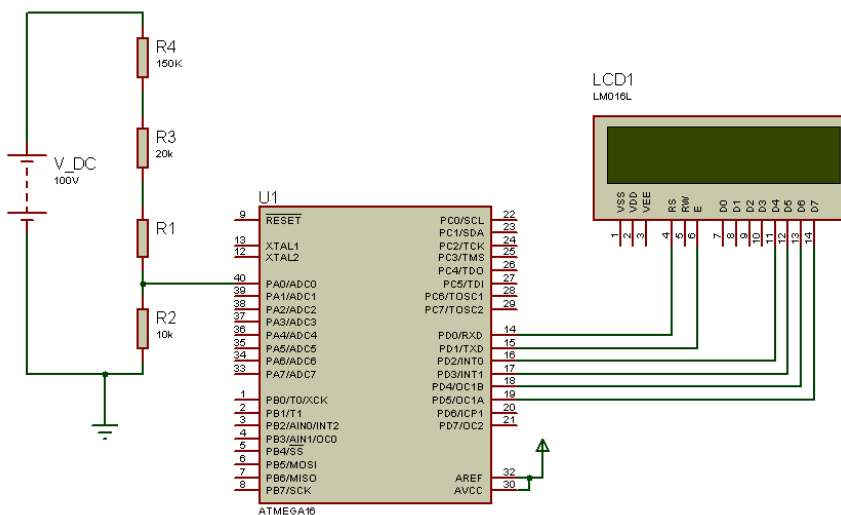
```

در برنامه بالا مقدار متغیر a در متغیر b که از جنس single است و اعداد اعشاری را نیز شامل می شود ریخته شده است، سپس

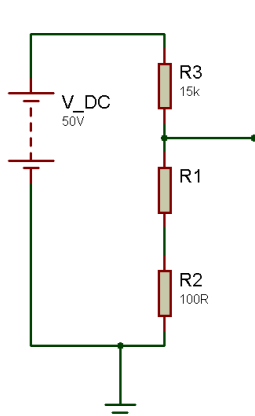
این متغیر بر عدد 10.23 تقسیم شده و جواب روی lcd نمایش داده شده است. دستور 1 = Digits , Config Single = Scientific ،

مقدار رقم اعشار متغیر سینگل را تعیین میکند که در اینجا 1 است.

مدار مربوط به مثال های بالا را مشاهده میکنید :



مثال : مداری طراحی کنید که ولتاژ خروجی یک منبع تغذیه ی 0 تا 50 ولت را تا دو رقم اعشار اندازه گیری کند .



اولین قدم محاسبه ی R1 و R2 است ، مقدار R1 را 15 کیلو در نظر میگیریم و R2 را

محاسبه میکنیم:

$$R2 = \frac{Va * R1}{(Vin - Va)} \Rightarrow R2 = \frac{5 * 15}{(50 - 5)} = 1.66K$$

بیشترین ولتاژ ورودی 50 ولت است ، بنابراین به جای 1023 باید عدد 50.00 نمایش داده شود ، در این حالت در برنامه عدد

1023 باید به 20.46 تقسیم شود ، همچنین برای دو رقم اعشار باید یک متغیر از جنس SINGLE با حداکثر رقم اعشار 2 پیش

بینی شود ، برنامه مذکور را در زیر مشاهده میکنید :

```
$regfile = "m16def.dat" : $crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.1 , Db5 = Pinc.2 , Db6 = Pinc.3 , Db7 = Pinc.4 , Rs = Pind.3 , E
= Pind.2
Config Adc = Single , Prescaler = Auto
Dim A As Word , B As Single
Config Single = Scientific , Digits = 2
Do
A = Getadc(0)
Locate 1 , 1 : Lcd A ; " "
B = A : B = B / 20.46
Locate 2 , 1 : Lcd B ; " "
Loop
End
```

با شبیه سازی برنامه مشاهده خواهد کرد که ولتاژ خروجی شبکه ی مقسم مقاومتی برابر با 4.81 ولت خواهد بود ، این

کمبود ولتاژ باعث ایجاد خطا در محاسبات و مقادیر نمایش داده شده میگردد . برای حل این مشکل باید دید که به ازای

4.8 ولت چه مقداری در متغیر A ریخته می شود (5ولت = 1024 ، 4.81 = ؟) این مقدار را میتوانید با رابطه ی ساده زیر

بدست آورید:

$$OUT = \frac{1023}{5} * X$$

X ولتاژ خروجی تقسیم ولتاژ است و OUT مقدار دیجیتال ریخته شده در متغیر میباشد ، حال بقیه محاسبات را با OUT انجام

دهید:

$$Vin = \frac{OUT}{x} \Rightarrow x = OUT / Vin$$

در برنامه بالا اگر به جای 20.46 عدد 19.68252 را قرار دهید مقدار نمایش داده شده روی LCD دقیقاً برابر با ورودی میشود.

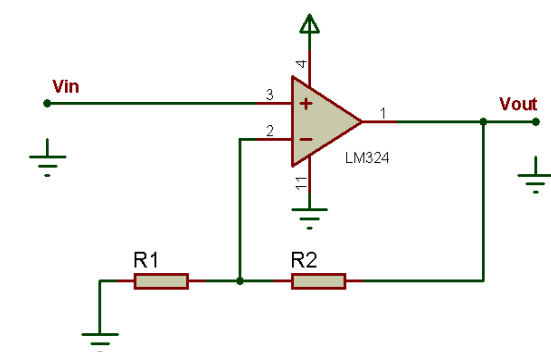
مثال : ولتاژ ورودی بین 0 تا 3.5 ولت را با دقت یک رقم اعشار اندازه بگیرید:

چون ولتاژ ورودی کمتر از 5 ولت است نیازی به تقسیم ولتاژ نیست و آن را مستقیم به ADC اعمال میکنیم . در ولتاژ های زیر 5 ولت ، بیشترین ولتاژ را 5 ولت در نظر بگیرید و محاسبات را انجام دهید (فرض کنید ولتاژ ورودی 0 تا 5 ولت است و شما آن را تا 3.5 زیاد میکنید).

در برنامه بالا به جای $B = B / 20.46$ مقدار $B = B / 204.6$ را قرار دهید و به ورودی ADC ولتاژ 3.5 ولت اعمال کنید .

اندازه گیری ولتاژ های زیر 1 ولت :

برای اندازه گیری این ولتاژ ها ، شما باید ابتدا آن را تقویت کرده و به 5 ولت برسانید ، برای این کار میتوانید از تقویت کننده های عملیاتی (OP-AMP) استفاده کنید :



مدار روبرو یک تقویت کننده غیر معکوس است ، که در زیر

روابط آن آورده شده است :

$$V_o = R1 * \frac{V_{in}}{R2} + V_{in}$$

$$I_{in} = \frac{V_i}{R2}$$

$$A_v = 1 + \frac{R1}{R2}$$

$$A_v = \frac{V_o}{V_{in}}$$

در این مدار ولتاژ خروجی باید 5 ولت باشد بنابراین با داشتن ولتاژ ورودی و خروجی و یکی از مقاومت ها میتوانید به سادگی مقاومت دیگر را محاسبه کنید :

بدست آوردن R1 با معلوم بودن R2:

$$R1 = (AV * R2) - 1$$

مثال ، میخواهیم ولتاژ 0 تا 100 میلی ولت را اندازه بگیریم ، مدار مناسب را طراحی کنید:

با فرض اینکه R2 یک کیلو اهم باشد (آن را 1 کیلو در نظر میگیریم) مقدار R1 را محاسبه میکنیم ، ابتدا بهره ولتاژ تقویت کننده را بدست میآوریم :

$$A_v = \frac{5}{.1} = 50$$

$$R1 = (Av * R2) - 1 \Rightarrow R1 = (50 * 1) - 1 = 49K$$

سپس ولتاژ خروجی op-amp را به ADC میکرو اعمال میکنیم ، همانطور که مشاهده میکنید ، به ازای ولتاژ ورودی ADC (5

ولت) ، روی LCD عدد 1023 نمایش داده میشود، در حالی که باید روی lcd عدد 100 نمایش داده شود .

از طریق فرمول زیر که مانند حالت قبل است به سادگی میتوانید عدد 1023 را به 100 تبدیل کنید:

$$OUT = \frac{1023}{5} * X$$

X ولتاژ خروجی تقسیم ولتاژ و OUT مقدار دیجتالی ریخته شده در متغیر میباشد ، حال بقیه محاسبات را با OUT انجام دهید:

$$Vin = \frac{OUT}{x} \Rightarrow x = OUT/Vin$$

در صورتی که vin را 100 میلی ولت در نظر بگیرید ، عدد اعشاری 1. روی lcd نمایش داده میشود که واحد آن ولت است اما

اگر vin را 100 در نظر بگیرید ، باید واحد mv را به آن بیافزاید :

```
$regfile = "m16def.dat" : $crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.1 , Db5 = Pinc.2 , Db6 = Pinc.3 , Db7 = Pinc.4 , Rs = Pind.3 , E = Pind.2
Config Adc = Single , Prescaler = Auto
Dim A As Word , B As Single
Config Single = Scientific , Digits = 1
Do
A = Getadc(0)
Locate 1 , 1 : Lcd A ; " "
B = A : B = B / 10.23
Locate 2 , 1 : Lcd B ; "mv"
Loop
End
```

مثال:مداری طراحی کنید که ولتاژ 5 میلی ولت را اندازه بگیرد ؟

ابتدا مدار تقویت کننده را طراحی میکنیم ، مقدار مقاومت R2 را 6.8 کیلو در نظر میگیریم و R2 را بدست میاوریم:

$$Av = \frac{VO}{Vin} \Rightarrow Av = \frac{5}{5m} = 1000$$

$$R1 = (Av * R2) - 1 \Rightarrow R1 = (1000 * 6.8) - 1 = 6799K$$

مشاهده میکنید که مقدار مقاومت خارج از رنج مجاز شد ، مقدار 1 کیلو اهم را انتخاب میکنیم:

$$R1 = (Av * R2) - 1 \Rightarrow R1 = (1000 * 1) - 1 = 999K$$

در اینجا به علت زیاد بودن بهره مقدار یکی از مقاومت ها زیاد بدست میاید ، در این حالت شما میتوانید از دو تقویت کننده پشت سرهم استفاده کنید ...

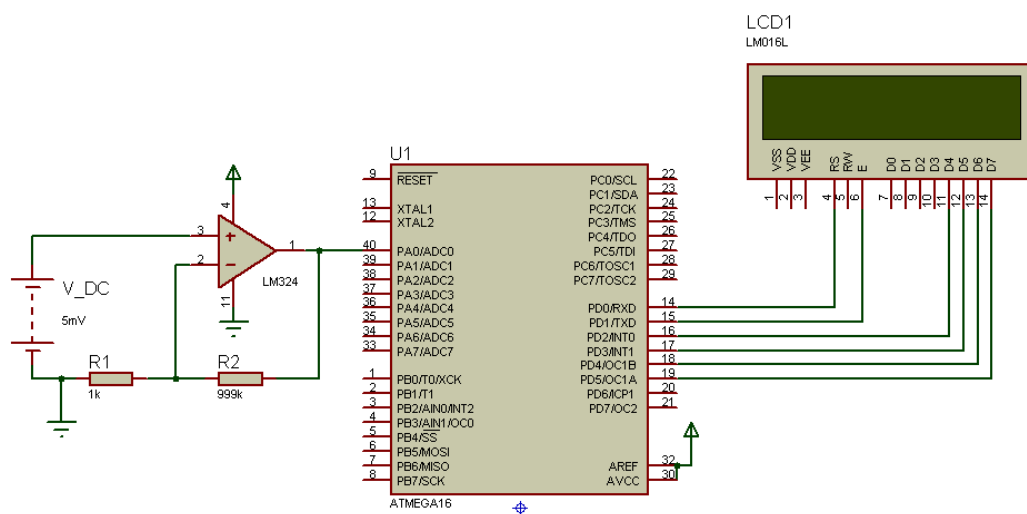
حالا به ازای ولتاژ 5 میلی ولت که به 5 ولت تبدیل شد و به ADC اعمال گردید ، روی LCD عدد 1023 نمایش داده میشود . ما باید 1023 را به 5 میلی ولت تبدیل کنیم (به جای 1023 5 میلی ولت را نمایش دهیم) ، طبق فرمول:

$$OUT = \frac{1023}{5} * X$$

$$Vin = \frac{OUT}{x} \Rightarrow x = OUT / Vin$$

برنامه و مدار مربوطه :

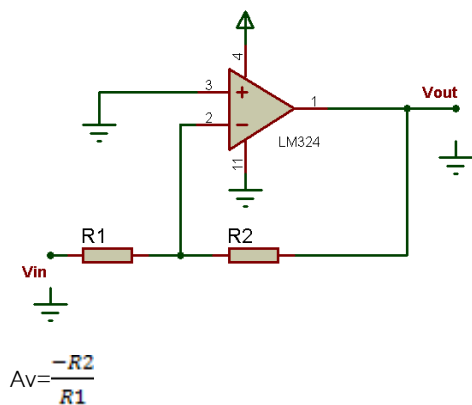
```
$regfile = "m16def.dat" : $crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Pinc.1 , Db5 = Pinc.2 , Db6 = Pinc.3 , Db7 = Pinc.4 , Rs =
Pind.3 , E = Pind.2
Config Adc = Single , Prescaler = Auto
Dim A As Word , B As Single
Config Single = Scientific , Digits = 5
Do
A = Getadc(0)
B = A : B = B / 204600
Locate 2 , 1 : Lcd B ; "v"
B = A : B = B / 204.600
Locate 1 , 1 : Lcd B ; "mv"
Loop
End
```



اندازه گیری ولتاژ های منفی :

برای اندازه گیری ولتاژ منفی باید آن را به مقدار متناظر مثبت تبدیل کرده و اندازه گیری کنید ، یکی از راه های تبدیل ولتاژ منفی به مثبت استفاده از تقویت کننده واران ساز (معکوس کننده) است .

در زیر مدار و محاسبات مربوطه را مشاهده میکنید:



در صورتی که ولتاژ منفی شما از 5 ولت بیشتر است ابتدا آن را با مقاومت (در بالا توضیح داده شد) تضعیف کنید و بعد از

مثبت کردن به میکرو اعمال کنید . کلیه محاسبات مربوط به این مدار مشابه مدار قبلی میباشد .

چند نکته ی ضروری:

- ولتاژ ورودی adc میتواند نهایتا 5 باشد (بیشتر از این ولتاژ سوختن adc حتمی است) .
- در هنگام کار اتصال پایه avcc به تغذیه و vref به ولتاژ مرجع فراموش نشود.

راه اندازی واحد DAC در سری ATXMEGA :

میکرو کنترلر های سری Atxmega دارای دو واحد DAC با مشخصات زیر میباشند :

- Two DACs with 12-bit resolution
- Up to 1 Msps conversion rate for each DAC
- Flexible conversion range
- Multiple trigger sources
- 1 continuous output or 2 Sample and Hold (S/H) outputs for each DAC
- Built-in offset and gain calibration
- High drive capabilities
- Low Power Mode

مبدل های دیجیتال به آنالوگ یا DAC برای تبدیل کردن داده ی دیجیتال به ولتاژ آنالوگ به کار میروند ، در سری ایکس مگا با اعمال رقم 0 در مبنای دیجیتال به واحد DAC میتوان ولتاژ 0 و با اعمال رقم 4096 میتوان ولتاژ 3.3 ولت را تولید کرد ، سایر مقادیر بین 0 تا 4096 ولتاژ ی بین 0 تا 3.3 ولت را به وجود می آورد . برای آگاهی در مورد نحوه ی کارکرد dac میتوانید به دیتاشیت یکی از قطعات پر کاربرد در این زمینه یعنی چیپ dac0808 مراجعه نمایید .

در کامپایلر بسکام میتوانیم با استفاده از دستورات زیر واحد dac را راه اندازی نماییم :

```
CONFIG DACx=dac, IO0=IO0, IO1=IO1, INTERNAL_OUTPUT =INTOTP, CHANNEL=channel, TRIGGER_CH0=trig0,
TRIGGER_CH1=trig1, REFERENCE=ref, LEFT_ADJUSTED=adjusted, EVENT_CHANNEL=event, INTERVAL=interval, R
EFRESH=refresh
```

DACx=dac : همانطور که در تصویر مقابل مشاهده میکنید ، یکی از واحد

های dac بر روی پورت a و دیگری بر روی پورت b قرار گرفته است :

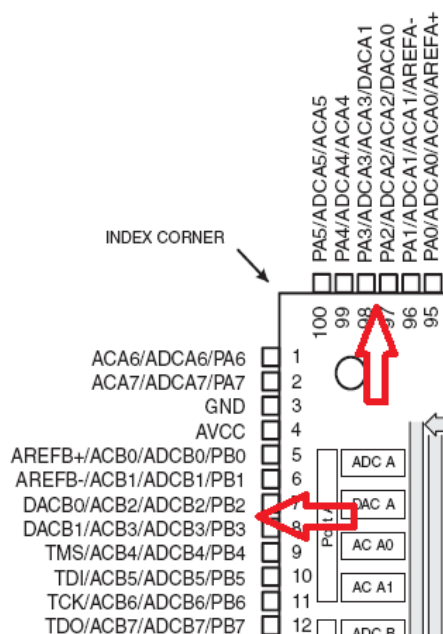
هر یک از واحد های dac دارای دو خروجی میباشد که با استفاده از

دستورات بعدی میتوانیم آنها را پیکربندی نماییم .

در دستور بالا به جای واژه ی dac میتوانیم یکی از دستورات ENABLED یا

DISABLED را برای فعال یا غیر فعال کردن DACB یا DACX که به جای DACX

نوشته میشود ، استفاده کنیم .



IO0=IO0 : در این دستور به جای واژه ی IO0 میتوانیم یکی از دستورات ENABLED یا DISABLED را برای فعال یا غیر فعال کردن خروجی 0 واحد DACx قرار دهیم .

IO1=IO1 : در این دستور به جای واژه ی IO1 میتوانیم یکی از دستورات ENABLED یا DISABLED را برای فعال یا غیر فعال کردن خروجی 1 واحد DACx قرار دهیم .

INTERNAL_OUTPUT =INTOTP : با استفاده از این دستور میتوانید خروجی داخلی را بر روی پایه ی DACX ایجاد نمایید ، با قرار دادن یکی از دستورات ENABLED یا DISABLED به جای INTOTP میتوانید این حالت را فعال یا غیر فعال نمایید ، این حالت بیشتر برای تست کردن DAC استفاده میشود و در حالت پیشفرض غیر فعال است .

CHANNEL=channel : در صورتی که قصد دارید از هر خروجی DACX استفاده نمایید ، به جای channel از DUAL و در صورتی که فقط از یکی از کانال ها استفاده کرده اید به جای واژه ی مذکور از SINGLE استفاده کنید . توجه داشته باشید که این دستور در صورتی معتبر است که هر دو خروجی Dac در دستورات قبلی فعال شده باشد .

TRIGGER_CH1=trig1 و TRIGGER_CH0=trig0 : با استفاده از این دو دستور میتوان تحریک کانال صفر یا یک dacx را فعال کرد ، در این حالت اگر مقدار موجود در رجیستر Dac به مقداری خاص برسد cpu میتواند یکی از المان های داخلی (نظیر تایمر ها ، پورت ها ، Dma یا ...) را فعال کند ، تحریک با قرار دادن ENABLED به جای trigx فعال و با قرار دادن DISABLED به جای آن غیر فعال میشود .

REFERENCE=ref : واحد dac برای عملکرد صحیح به یک ولتاژ مرجع داخلی نیاز دارد ، شما میتوانید یکی از ولتاژ های مرجع زیر را برای این واحد انتخاب نمایید :

INT1V : با نوشتن این گزینه به جای واژه ی ref واحد Dac از ولتاژ ولت داخلی به عنوان مرجع استفاده میکند .

AVCC : با نوشتن این گزینه به جای واژه ی ref واحد Dac از ولتاژ تغذیه ی میکرو به عنوان مرجع استفاده میکند .

AREFA : با نوشتن این گزینه به جای واژه ی ref واحد Dac از ولتاژ موجود بر روی پایه ی AREFA به عنوان مرجع استفاده میکند .

AREFB : با نوشتن این گزینه به جای واژه ی ref واحد Dac از ولتاژ موجود بر روی پایه ی AREFB به عنوان مرجع استفاده میکند .

توجه داشته باشید که تعریفی که در مبدل های دیجیتال به آنالوگ برای ولتاژ مرجع ارائه می شود، با تعریف ولتاژ مرجع در مبدل های آنالوگ به دیجیتال متفاوت است.

در این دستور منظور از ولتاژ مرجع حداکثر سطح ولتاژ خروجی آن است، مثلاً اگر ولتاژ مرجع برابر با V_{CC} انتخاب شود، ولتاژ خروجی $DACX$ میتواند از V_{CC} با 4096 پله تغییر کند (یا اگر ولتاژ مرجع برابر ولتاژ ولت داخلی انتخاب شود، خروجی از صفر تا یک ولت با 4096 پله خواهد بود).

LEFT_ADJUSTED=adjusted: در حالتی عادی واحد $DACX$ از سمت راست تنظیم می شود، شما میتوانید با قرار دادن ENABLED یا DISABLED به جای adjusted، تنظیم شدن را بر روی چپ یا راست تغییر دهید.

EVENT_CHANNEL=event: با نحوه ی کار این دستور در بخش های بعدی آشنا خواهیم شد.

INTERVAL=interval: به جای واژه ی interval یکی از مقادیر 1,2,4,8,16,32,64 یا 128 قرار داده میشود، تا تاخیر زمانی میان تبدیل های $DACX$ مشخص شود، مثلاً اگر شما از 64 استفاده نمایید، در میان هر تبدیل 64 سیکل کلاک تاخیر خواهیم داشت.

REFRESH=refresh: در صورتی که از هر دو کانال $DACX$ استفاده نمایید، میتوانید با قرار دادن یکی از مقادیر 16, 32, 128، refresh تاخیر زمانی میان تبدیل دو کانال 0 و 1 را معین نمایید.

قرار دادن خروجی :

با قرار دادن مقادیر عددی یا متغیر دلخواه در یکی از رجیستر های DACB و DACA میتوانید $DACB$ و $DACA$ را مقدار دهی کنید. مثال :

```
$regfile = "xm128A1def.dat"
```

```
Config Osc = Disabled , 32mhzosc = Enabled
```

```
Config Sysclock = 32mhz , Prescalea = 2 , Prescalebc = 1_2
```

```
Config Dac = Enabled , Io0 = Enabled , Channel = Single , Reference = Int1v , Interval = 64 , Refresh = 64
```

```
Dim A As Word
```

```
Start Dac
```

Do

Daca = A

Incr A

Waitms 1

If A > 4096 Then

A = 0

End If

Loop

End

در مثال بالا یک پالس دندان اره ای با حداکثر دامنه ی 1 ولت و زمان تناوب 4.096 ثانیه ایجاد شده است ، همانطور که مشاهده میکنید مقدار A از صفر شروع به افزایش کرده و هنگامی که مقدار آن به 4096 رسید ، با دستور شرطی موجود ، مقدارش صفر میشود . از آنجا که ولتاژ مرجع dac برابر با 1 ولت داخلی است ، مقدار 4096 باعث ایجاد ولتاژ 1 ولت در خروجی میشود .

مقایسه کننده آنالوگ :

مقایسه کننده آنالوگ مقادیر ولتاژ آنالوگ موجود بر روی دو پایه مثبت و منفی خود را با هم مقایسه می کند . (مانند op - amp) . زمانی که ولتاژ موجود در ورودی مثبت بیشتر از ولتاژ موجود در ورودی منفی باشد ، خروجی مقایسه کننده (AC0) یک می شود و کاربر با توجه به آن میتواند عملیات های بعدی را انجام دهد . مقایسه کننده دارای یک پرچم وقفه مجزا نیز میباشد . خروجی مقایسه کننده می تواند به عنوان تریگر ورودی CAPTURE تایمر / کانتر یک نیز استفاده شود . در خانواده ی AVR ، سری های TINY و AT90S معمولا دارای یک مقایسه کننده ی آنالوگ بوده و پیکربندی آن به شرح زیر است :

CONFIG ACI =ON/OFF, COMPARE = ON/OFF, TRIGGLE=TOGGLE,RISING/FALLING

CONFIG ACI = ON/OFF : در زمان استفاده از مقایسه کننده باید یک باشد.(در صورت استفاده از صفر مقایسه کننده کار نخواهد کرد).

COMPARE = ON/OFF: در صورت انتخاب ON ، AC0 مستقیما به ورودی CAPTURE تایمر/کانتر یک وصل می شود.

TRIGGLE=TOGGLE,RISING/FALLING: نحوه روی دادن وقفه مقایسه کننده را نشان می دهد.

FALLING: یک لبه پایین رونده در خروجی مقایسه کننده باعث یک شدن پرچم وقفه مقایسه کننده و اجرا شدن برنامه وقفه

خواهد شد.

RISING: یک لبه بالا رونده در خروجی مقایسه کننده باعث یک شدن پرچم وقفه مقایسه کننده و اجرا شدن برنامه وقفه

خواهد شد.

TOGGLE: یک به صفر یا یک صفر به یک در خروجی مقایسه کننده باعث یک شدن پرچم وقفه مقایسه کننده و اجرا شدن

برنامه وقفه خواهد شد.

مثال :

```
$regfile = "m8def.dat"
$crystal = 16000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portb.2 , Db5 = Portb.3 , Db6 = Portb.4 , Db7 = Portb.5 , E = Portb.1 , Rs = Portc.0
Config Portd = Input
Config Aci = On , Compare = On , Trigger = Falling
Config Timer1 = Timer , Capture Edge = Falling , Noise Cancel = 1 , Prescale = 1024
Enable Interrupts
Enable Icp1
Enable Aci
On Aci Q
Dim A As Byte
Do
Locate 1 , 1
Lcd Timer1
Loop
End
Q:
Locate 2 , 1
Lcd Capture1
Stop Timer1
Return
```

در مثال بالا تایمر یک در مد Capture راه اندازی شده است ، در این مد در صورتی که یک لبه به پایه ICP (پایه 14 مگا 8)

اعمال شود ، مقدار شمرده شده توسط تایمر 1 در رجیستر Capture1 ریخته میشود و شما میتوانید از این مقدار استفاده کنید .

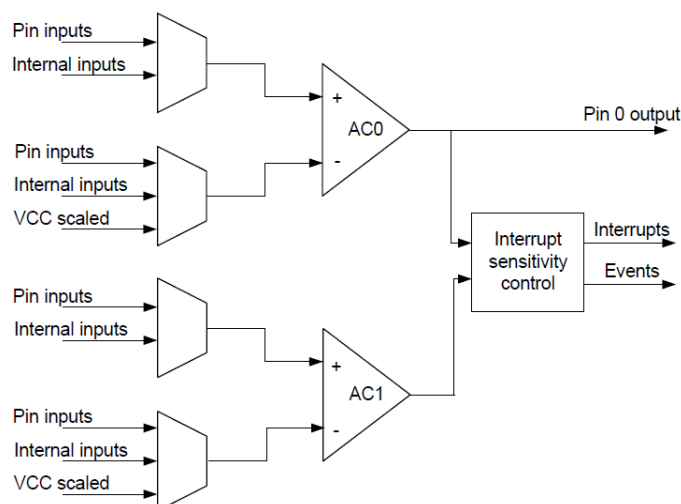
در مثال به جای لبه بالا رونده از وقفه مقایسه گر استفاده شده است ، هنگامی مقدار ولتاژ پایه مثبت مقایسه گر (AIN0) از ولتاژ پایه منفی (AIN1) بیشتر شود وقفه فعال شده و مقدار شمرده شده توسط تایمر در رجیستر Capture1 ریخته میشود و سپس در موقعیت 1 و 2 روی LCD به نمایش در میاید ، شما میتوانید در زیر روال وقفه از هر دستور دیگری نیز استفاده کنید ، با استفاده از این روش میتوان زمان تناوب پالس را اندازه گرفت.

در میکروکنترلر های سری ATXMEGA تعداد مقایسه کننده ی آنالوگ تا 4 عدد افزایش پیدا کرده است ، ورودی این مقایسه کننده ها که بر روی پورت های آنالوگ (پورت a و پورت b قرار دارد) میتواند تا چهار سیگنال آنالوگ را از ورودی های تعیین شده توسط کاربر دریافت کرده و بعد از مقایسه آنها رویداد تعیین شده را اجرا کند .

ورودی های مقایسه کننده ی آنالوگ در سری ATXMEGA پایه های زیر هستند :

- ✓ پایه های 0, 1, 2, 3, 4, 5, 6 برای ورودی مثبت
- ✓ پایه های 0, 1, 3, 5, 7 برای ورودی منفی
- ✓ در پیکربندی پایه های مثبت کاربر میتواند از داده ی قرار گرفته در DAC نیز استفاده کند .
- ✓ در پیکربندی پایه منفی کاربر میتواند علاوه بر پایه های ذکر شده ، از موارد زیر نیز به عنوان ورودی استفاده کند :

- مقیاس دلخواه از VCC (حداکثر تا 64)
- ولتاژ مرجع Bandgap (Bandgap یک ولتاژ مرجع داخلی با مقدار دقیق 1 ولت است)
- داده ی قرار گرفته در خروجی DAC



راه اندازی این واحد با دستورات زیر انجام میشود :

CONFIG ACXX = state, TRIGGER=trigger, HISPEED=speed, HYSMODE=hys , MUXPLUS=mp , MUXMIN=mm ,
 OUTPUT=otp , SCALE=scale , WINDOW=w , WINTMODE = wint

CONFIG ACXX = state : در این بخش ACXX شماره ی مقایسه ی کننده ای است که قصد پیکربندی آن را داریم (ACA0 ، ACA1 ، ACB0 یا ACB1) ، همچنین به جای واژه ی state از دستور ON یا OFF برای روشن یا خاموش کردن مقایسه کننده استفاده می شود . در سری ATXMEGA بر روی هر پورت آنالوگ (پورت های A و B) دو عدد مقایسه کننده ی آنالوگ وجود دارد .

TRIGGER=trigger : این دستور نحوه ی ایجاد کردن پالس تحریک در هنگام برابر شدن ورودی های مقایسه کننده را تعیین میکند ، به جای واژه ی trigger میتوانید از دستورات RISING, FALLING or BOTH / TOGGLE برای ایجاد پالس بالا رونده ، پایین رونده یا تغییر وضعیت لبه ی فعلی استفاده کنید . در صورت استفاده از این دستور در برنامه بسته به مقایسه کننده ی پیکربندی شده (A یا B) ، پایه ی 7 پورت A یا B تغییر وضعیت می دهد . در صورتی که این دستور در برنامه نوشته نشود ، وقفه ی مقایسه کننده ی فعال نمیشود ، اما با فعال شدن خروجی مقایسه کننده پایه ی 7 یک میشود .
 HISPEED=speed : با قرار دادن دستور Enabled یا Disabled به جای واژه ی speed میتوانید قابلیت نمونه برداری با سرعت بالا را برای مقایسه کننده فعال یا غیر کنید ، در صورتی که این قابلیت فعال باشد مقایسه کننده تمامی تغییرات ولتاژ لحظه ای یا غیر لحظه ی ورودی را در محاسبات خود لحاظ میکند .

HYSMODE=hys : در واحد ACI قابلیت ایجاد پسماند ولتاژ برای کاربر تعبیه شده است ، به نوعی میتوان گفت که در ورودی مقایسه کننده با استفاده از یک خازن داخلی میتوان میزان پسماند ولتاژ اعمال شده را کم ، زیاد یا صفر کرد . این کارها با قرار دادن دستورات Off ، Small ، Large به جای واژه ی hys قابل انجام است . در این حالت میزان پسماند Large باعث واکنش کند مقایسه کننده به ورودی خواهد شد .

MUXPLUS=mp : با این دستور میتوانید ورودی مثبت مقایسه کننده آنالوگ را با قرار دادن اعداد 0 تا 7 به جای واژه ی mp تعیین کنید . اعداد صفر تا شش برای پایه های 0, 1, 2, 3, 4, 5, 6 و عدد 7 برای استفاده از خروجی DAC به عنوان ورودی مثبت است .

MUXMIN=mm : در این دستور پایه ی ورودی منفی مقایسه کننده تعیین میشود ، به جای mm میتوانید یکی از

مقادیر زیر را قرار دهید :

✓ 0 - اتصال به پین صفر (ACB0 یا ACA0)

✓ 1 - اتصال به پین یک

✓ 2 - اتصال به پین سه

✓ 3 - اتصال به پین پنج

✓ 4 - اتصال به پین هفت

✓ 5 - اتصال به خروجی DAC

✓ 6 - اتصال به ولتاژ BANDGAP

✓ 7 - تنظیم در دستور SCALE

OUTPUT=otp : با استفاده از این دستور میتوانید پایه ی خروجی مقایسه ی کننده ی آنالوگ را فعال یا غیر فعال کنید . با

قرار دادن دستور Enabled به جای Otp خروجی مقایسه ی کننده ی آنالوگ فعال شده و پایه ی 7 پورت آنالوگ

پیکربندی شده در حالت خروجی پیکربندی میشود . پایه ی مذکور متناسب با پیکربندی انجام شده در دستور

TRIGGER در هنگام ایجاد روی داده تغییر وضعیت میدهد . دستور Disabled باعث غیر فعال شدن خروجی میشود . این

دستور فقط برای ACA0 و ACB0 قابل استفاده است .

SCALE=scale : در صورتی که در دستور MUXMIN=mm به جای mm عدد 7 را قرار دهید میتوانید از مقیاس دلخواه ولتاژ

تغذیه VCC به عنوان ورودی منفی مقایسه کننده استفاده کنید . در این دستور به جای scale عددی بین 0 تا 63 قرار داده

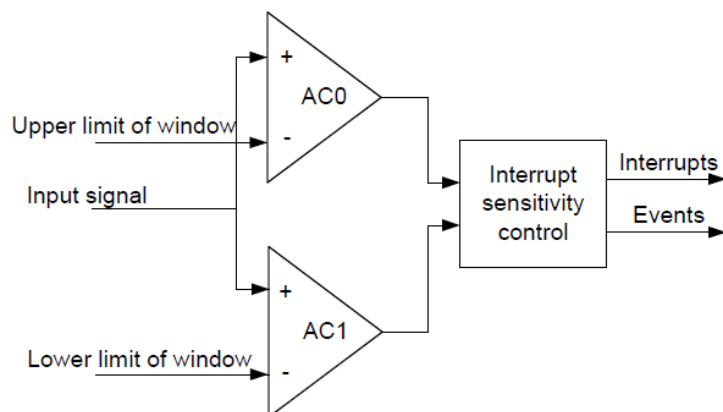
میشود ، در این حالت عدد 63 ولتاژی برابر با VCC و عدد 0 ولتاژ صفر ولت را به ورودی منفی مقایسه کننده اعمال

میکند $(vcc * (scale + 1)) / 64$.

Window Function : WINDOW=w یکی دیگر از قابلیت های جدیدی است که به سری ATXMEGA اضافه شده است ، با قرار

دادن دستور Enabled یا Disabled به جای W میتوانید این قابلیت را فعال یا غیر فعال کنید . با فعال شدن این قابلیت مقایسه

کننده ی صفر و یک در پورت آنالوگ X مطابق با تصویر زیر پیکربندی میشوند :



در این حالت میتوانید یک سیگنال ورودی را در دو بازه ی مینیمم و ماکزیمم مقایسه کنید .

WINTMODE = wint مقایسه ی کننده ی آنالوگ در حالت Window دارای چهار پرچم برای اعلام وضعیت است :

ABOVE : سیگنال ورودی بیشتر از بازه ی تعیین شده است .

INSIDE : سیگنال ورودی در بازه ی تعیین شده است.

BELOW : سیگنال ورودی کمتر از بازه ی تعیین شده است.

OUTSIDE : سیگنال ورودی بیرون از بازه ی تعیین شده است.

با قرار دادن هر یک از دستورات بالا به جای wint میتوانید وقفه ی مقایسه کننده را برای آن وضعیت فعال کنید . این

وضعیت ها در بیت 6 و 7 رجیستر aca_status ذخیره میشود .

مثال :

```
$regfile = "xm128a1def.dat"
$crystal = 32000000
$hwstack = 64
$swstack = 64
$framesize = 64

'include the following lib and code, the routines will be replaced since they are a workaround
$lib "xmega.lib"
$external _xmegafix_clear
$external _xmegafix_rol_r1014

'First Enable The Osc Of Your Choice , make sure to enable 32 KHz clock or use an external 32 KHz clock
Config Osc = Enabled , 32mhzosc = Enabled

'configure the systemclock
Config Sysclock = 32mhz , Prescalea = 1 , Prescalebc = 1_1

Config Com1 = 19200 , Mode = Asynchronous , Parity = None , Stopbits = 1 , Databits = 8

'setup comparator pin 0 and pin 1 are the input of portA. Pin 7 is an output in this sample
Config Aca0 = On , Hysmode = Small , Muxplus = 0 , Muxmin = 1 , Output = Enabled
```

```

Do
Print Bin(aca_status)
Print Aca_status.4 'output ac0
Waitms 1000
Loop

```

در مثال بالا مقایسه کننده ی آنالوگ 0 پورت آنالوگ A با دستور Config Aca0 = On راه اندازی شده و PORTA.0 به عنوان ورودی مثبت و PORTA.1 به عنوان ورودی منفی آن تعیین شده است. هنگامی که ولتاژ اعمال شده به پایه ی مثبت از ولتاژ اعمال شده به پایه ی منفی بیشتر شود PORTA.7 یک میشود. سایر وضعیت های مقایسه ی کننده آنالوگ از طریق رجیستر Aca_status در دسترس است. برای کسب اطلاعات بیشتر در مورد این رجیستر دیتاشیت میکروکنترلی که با آن کار میکنید را مطالعه نمایید.

مثال:

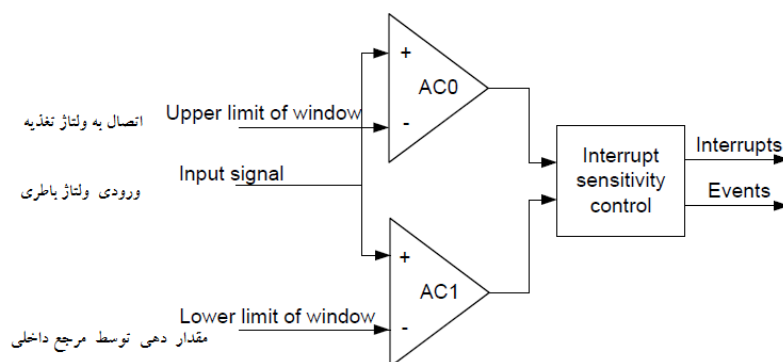
```

'$regfile = "xm128A1def.dat"
$regfile = "xm32A4def.dat"
$crystal = 32000000
$hwstack = 64
$swstack = 64
$framesize = 64
'include the following lib and code, the routines will be replaced since they are a workaround
$lib "xmega.lib"
$external _xmegafix_clear
$external _xmegafix_rol_r1014
'First Enable The Osc Of Your Choice , make sure to enable 32 KHz clock or use an external 32 KHz
clock
Config Osc = Enabled , 32mhzosc = Enabled
'configure the systemclock
Config Sysclock = 32mhz , Prescalea = 1 , Prescalebc = 1_1
Config Com1 = 19200 , Mode = Asynchronous , Parity = None , Stopbits = 1 , Databits = 8
Config Priority = Static , Vector = Application , Lo = Enabled , Med = Enabled , Hi = Enabled
'setup comparator in window mode on Port A
'MUXPLUS = 2 on both Comparators --> This is the INPUT SIGNAL in Window Mode (PIN 2)
'MUXMIN for ACA0 = 1 (which is PIN 1) = UPPER LIMIT OF SINGAL (For example connect 5 Volt)
'MUXMIN for ACA1 = 6 (For example connect 2.5 Volt or SCALE from VCC) = LOWER LIMIT OF SIGNAL
'So the Window is fom 2.5 Volt to 5 Volt
Config Aca0 = On , Hispeed = Disabled , Window = Enabled , Hysmode = Small , Muxplus = 2 , Muxmin
= 1
Config Aca1 = On , Hispeed = Disabled , Window = Enabled , Hysmode = Small , Muxplus = 2 , Muxmin
= 7 , SCALE=32 , Wintmode = Outside ' specify WINTMODE once
On Aca_acw Ac_window_isr
Enable Aca_acw , Lo
Enable Interrupts
'Now connect a variable 0-5 volt power supply alternately GND with PIN A.2 (INPUT SIGNAL)
Do
Print "ACA_ac0ctrl = " ; Bin(aca_ac0ctrl)
Print "ACA_ac1ctrl = " ; Bin(aca_ac1ctrl)
Print "ACA_status = " ; Bin(aca_status)
If Aca_status.6 = 1 And Aca_status.7 = 0 Then
Print "INSIDE"
Else
Print "OUTSIDE"
End If
If Aca_status.2 = 1 Then
Print "Window INT Flag is set"
End If
Waitms 1000
Loop
End
'end program

```

```
Ac_window_isr:
    Print "BELOW Window"
    Waitms 5
    Return
```

برنامه بالا مربوط به یک سیستم مانیتور ولتاژ باتری است، این سیستم باید ولتاژ یک باتری 4.2 ولتی را در محدوده مجاز 2.5 تا 5 ولت نگه دارد و در صورتی که ولتاژ باتری از این محدوده کم تر یا بیشتر شود (خالی شدن باتری یا خراب شدن شارژر)، آن را از سیستم خارج کند. در این حالت ورودی منفی AC0 که در اینجا PORTA.1 است، به ولتاژ تغذیه متصل شده است همچنین با مقدار دهی دستور MUXMIN با عدد 7 و مقدار دهی دستور SCALE با عدد 32 ولتاژی برابر با $VCC/2$ به ورودی منفی AC1 اعمال میشود. همانطور که در تصویر زیر مشاهده میکنید، ورودی مثبت هر دو مقایسه کننده باید به یک پایه متصل شده و سپس در پیکربندی سخت افزاری به باتری وصل گردد که در برنامه بالا ورودی های مثبت با دستور $Muxplus = 2$ به PORTA.2 متصل شده اند.



با دستور $Wintmode = Outside$ وقفه ی مقایسه کننده ی آنالوگ برای حالتی که مقدار اندازه گیری شده از بازه ی تعیین شده خارج شده باشد، فعال میشود، در این حالت اگر ولتاژ PORTA.2 از 2.5 ولت کمتر شده یا از 5 ولت بیشتر شود، CPU میکرو کنترلر با برچسب Ac_window_isr پرش کرده و زیر برنامه ی موجود در آن را اجرا میکند، همچنین در حلقه ی اصلی برنامه:

```
Do
    Print "ACA_ac0ctrl = " ; Bin(aca_ac0ctrl)
    Print "ACA_ac1ctrl = " ; Bin(aca_ac1ctrl)
    Print "ACA_status = " ; Bin(aca_status)
    If Aca_status.6 = 1 And Aca_status.7 = 0 Then
        Print "INSIDE"
    Else
        Print "OUTSIDE"
```

```

End If
If Aca_status.2 = 1 Then
    Print "Window INT Flag is set"
End If
Waitms 1000
Loop

```

مقادیر رجیستر های `aca_ac0ctrl` و `aca_ac1ctrl` و `aca_status` به پورت سریال ارسال میشوند.

فعال سازی وقفه ی مقایسه کننده ی آنالوگ با دستورات زیر انجام شده است:

```

Config Priority = Static , Vector = Application , Lo = Enabled , Med = Enabled , Hi = Enabled
On Aca_acw Ac_window_isr
Enable Aca_acw , Lo
Enable Interrupts

```

برای کسب اطلاعات بیشتر در مورد دستورات بالا به بخش "راه اندازی وقفه" مراجعه کنید.

تایمر ها و کانترها در میکرو کنترلر های AVR:

میکرو کنترلر های سری TINY ، 90S و ATMEGA در خانواده ی AVR نهایتا دارای سه تایمر / کانتر هستند (به جز MEGA128, MEGA64, MEGA162 و تعداد محدود دیگر که 4 تایمر دارند). این تایمر / کانتر ها به نام های تایمر-کانتر 0 و تایمر-کانتر 1 و تایمر-کانتر 2 و تایمر-کانتر سه ، نام گذاری میشوند. کار تایمر ها شمردن تا یک عدد خاص و کار کانتر ها شمردن یک پالس ، که به پایه مخصوص اعمال میشود است ، از تایمر و کانتر استفاده های دیگری همچون ساخت پالس PWM ، ایجاد پالس تحریک و ... نیز میشود که در ادامه آنها را معرفی میکنیم.

در میکرو کنترلر های سری ATXMEGA نحوه ی راه اندازی و کار با واحد تایمر - کانتر ، اندکی با سری های قبلی متفاوت میباشد ، در این بخش ما ابتدا این واحد را در سری های TINY ، 90S و ATMEGA بررسی نموده ایم و سپس در بخشی مجرا نحوه ی کار با تایمر-کانتر های سری ATXMEGA را شرح داده ایم .

اولین تایمر/کانتری که معرفی میشود تایمر/کانتر صفر است ، این تایمر/کانتر در تمامی میکرو کنترلر های AVR به جز سری ATXMEGA وجود دارد . در زیر مشخصات این بخش را مشاهده می فرمایید:

✓ این تایمر/کانتر 8 بیتی است ، و نهایتا میتواند تا 2^8 (255) بشمارد.

✓ کلاک این تایمر میتواند توسط نوسان ساز داخلی یا نوسان ساز خارجی تامین شود (مقدار فرکانس نوسان ساز بر

عدد PRESCALE تقسیم میشود).

✓ این تایمر دارای چندین منبع وقفه میباشد که شما میتوانید آنها را در هر قسمت از برنامه فعال یا غیر فعال کنید.

✓ از این منابع وقفه میتوان ، وقفه سرریزی را نام برد.

✓ این تایمر/کانتر میتواند در مد تایمر و کانتر راه اندازی شود.

✓ ورودی کانتر پایه T0 (در میکرو مگا 16 پورت B.0 ، پایه شماره 1) میباشد

راه اندازی تایمر صفر در محیط بسکام:

تایمر صفر با دستور زیر پیکر بندی میشود:

```
CONFIG TIMER0 = TIMER ,PRESCALE = 1|8|64|256|1024
```

عدد PRESCALE فرکانس (دقت) تایمر را معین میکند. فرکانس و زمانی که تایمر میشمارد از فرمولهای زیر محاسبه میشوند:

$$\text{زمان} = \frac{\text{PRESCALE} * \text{تایمر بیت}}{\text{مقدار کریستال}}$$

$$\text{فرکانس} = \frac{\text{مقدار کریستال}}{\text{PRESCALE}}$$

بعد از دستور بالا، تایمر با دستور START TIMER شروع به شمارش میکند و با دستور STOP TIMER متوقف می شود.

تایمر پس از شمردن تا 255 سر ریز میشود ، شما با استفاده از دستور ENABLE OVFO یا ENABLE INTRRUPTS میتوانید وقفه سر ریزی تایمر را راه اندازی کنید . در صورتی که وقفه سر ریزی تایمر فعال باشد ، تایمر پس از سر ریزی به برجسیبی که با یکی از دستورات ON OVFO LABEL و یا ON TIMERO LABEL مشخص شده پرش میکند ، باز گشت از وقفه با دستور RETURN انجام میشود.

شما همچنین میتوانید با استفاده از دستور VAR = TIMERO ، مقدار تایمر را در یک متغیر از جنس بایت قرار دهید و یا با دستور TIMERO = VALUE ، مقدار اولیه ای در تایمر قرار داد تا تایمر از آن شروع به شمارش کند. مثال :

در این مثال میخواهیم توسط تایمر صفر زمان 1 ثانیه بسازیم ، طبق فرمول اگر شما از کریستال 1 مگا هرتز (که کمترین مقدار کریستال مورد استفاده برای AVR میباشد) به جز کریستال ساعت (و PRESCALE ، 1024 استفاده نمایید ، بیشترین زمانی که میتوانید ایجاد کنید { $\text{زمان} = \frac{256 * 1024}{1000000} = 265 \text{ ms}$ } ، است، بنابراین ما باید از وقفه سر ریزی تایمر استفاده کنیم ، و در هر بار وقفه به یک متغیر عددی یک واحد بیفزاییم ، هنگامی که متغیر برابر 4 شد ، متغیر صفر شود و یک واحد به متغیر اصلی افزوده شود. بنابراین با ضرب کردن زمان بدست آمده در 4 زمان اصلی بدست میاید که برابر با 1048 میلی ثانیه است.

```
$regfile = "m16def.dat"
```

```
crystal = 12000000$
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Dim B As Byte , C As Byte
```

```
Config Timer0 = Timer , Prescale = 1024
```

```
Enable Interrupts
```

```
Enable Timer0
```

```
On Timer0 P
```

```
Start Timer0
```

```
Do
```

Loop

End

P:

Locate 1, 8 : Lcd B

Incr B

If B > 3 Then : Incr C : Locate 1, 1 : Lcd C : B = 0 : End If

return

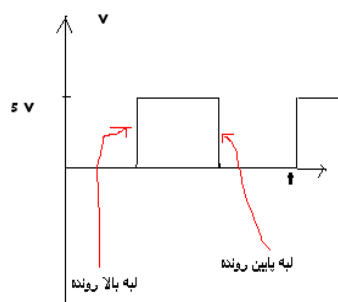
در این برنامه ، هنگامی که تایمر سرریز میشود (تا 255 میشمارد) به زیر برنامه p پرش میشود و یک واحد به متغیر b افزوده میشود ، شرط if میگوید که اگر مقدار متغیر b برابر با 4 شد یک واحد به متغیر c اضافه کن و آن را در سطر اول ، ستون اول lcd نمایش بده ، بعد متغیر b را صفر کن ، از آنجا که مقدار b به 4 نرسیده (در اولین رجوع مقدار b برابر 1 میشود) این شرط اجرا نمیشود .

در این حالت CPU با دستور return به برنامه تایمر برمیگردد . مقدار تایمر صفر شده و دوباره تا 255 شمرده میشود ... و این عمل مدام تکرار میشود. با چند دستور ساده میتوان از برنامه بالا یک ساعت ساخت (برنامه ساعت را در پروژه ها ببینید)

راه اندازی کانتر صفر در محیط بسکام:

کانتر صفر در بسکام با دستور زیر پیکربندی میشود:

CONFIG TIMER0 = COUNTER , EDGE = RISING / FALLING



✓ با انتخاب EDGE = RISING کانتر نسبت به لبه ی بالا رونده حساس است

✓ با انتخاب EDGE = FALLING کانتر نسبت به لبه ی پایین رونده حساس است

✓ (لبه های بالا رونده یا پایین رونده را میشمارد) (بالا رونده < صفر به یک و پایین

رونده > یک به صفر))

شما همچنین میتوانید با استفاده از دستور VAR = COUNTER 0 ، مقدار کانتر را در یک متغیر از جنس بایت قرار دهید و یا با دستور COUNTER 0 = VALUE ، مقدار اولیه ای در کانتر قرار دهید تا کانتر از آن شروع به شمارش کند. کانتر نیز مانند تایمر پس از شمردن تا 255 سر ریز میشود ، شما با استفاده از دستور ENABLE OVFO میتوانید و قفه سر ریزی کانتر را راه اندازی

کنید. در صورتی که وقفه سرریزی کانتر فعال باشد، کانتر پس از سرریزی به برجسبی که با یکی از دستورات ON OVFO

LABLE و یا ON TIMER0 LABLE مشخص شده پرش میکند، باز گشت از وقفه با دستور RETURN انجام میشود. مثال:

در این مثال کانتر صفر تعداد پالسهای اعمالی به پایه t0 (پایه شماره 1 مگا 16) را می‌شمارد و پس از شمردن تا مقدار 255، به زیر برنامه وقفه میرود، در آنجا یک واحد به متغیر b افزوده میشود، متغیر b تعداد دفعات سرریزی کانتر را نشان میدهد که ما آن را روی سطر دوم lcd به نمایش میگذاریم، برنامه با دستور return از زیر برنامه خارج میشود و به حلقه اصلی برمیگردد.

```
$regfile = "m16def.dat"
```

```
crystal = 12000000$
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Dim B As Byte
```

```
Config Timer0 = Counter , Edge = Rising
```

```
Enable Interrupts
```

```
On Counter0 P
```

```
Do
```

```
Locate 1 , 1 : Lcd Counter0
```

```
Loop
```

```
End
```

```
P:
```

```
Incr B : Locate 2 , 1 : Lcd "tedad dafeat sarrazi" ; B
```

```
Return
```

نکته: عملکرد تایمر و کانتر از عملکرد سایر قسمت ها جدا بوده و دیگر اعمال (مانند دستورات تاخیر مثل wait و...) بر

عملکرد آن اثری ندارد.

تایمر-کانتر یک:

در زیر مشخصات این مورد را مشاهده میکنید:

✓ این تایمر-کانتر 16 بیتی است، و نهایتاً میتواند تا 2^{16} (65535) بشمارد.

✓ کلاک تایمر میتواند توسط نوسان ساز داخلی یا نوسان ساز خارجی یا از پایه t1 (در مگا 16 پایه شماره 2 (portb.1))

توسط پالس خارجی تامین شود (مقدار فرکانس نوسان ساز بر عدد PRESCALE تقسیم میشود).

✓ تایمر - کانتر یک دارای دو خروجی مقایسه ای است که دو رجیستر OCR1A و OCR1B دو مقدار مقایسه ای را در خود جای می دهند و با محتوای تایمر/کانتر مقایسه می شوند .

✓ در زمان تساوی محتوای رجیستر مقایسه و محتوای تایمر/کانتر ، وضعیت پایه های خروجی مد مقایسه ای OC1A و OC1B می تواند تغییر کند.

✓ تایمر / کانتر در مد CAPTURE نیز می تواند به کار رود . با تحریک پایه ICP می توان محتوای تایمر/ کانتر را در رجیستر ورودی CAPTURE (ICR1) قرار داد.

✓ خروجی مقایسه کننده آنالوگ نیز می تواند به عنوان تریگر ورودی CAPTURE استفاده شود .

✓ این تایمر دارای چندین منبع وقفه میباشد که شما میتوانید آنها را در هر قسمت از برنامه فعال یا غیر فعال کنید از این منابع وقفه میتوان ، وقفه سرریزی را نام برد.

✓ این تایمر کانتر میتواند در مد تایمر و کانتر و pwm راه اندازی شود.

✓ ورودی کانتر پایه T1 (در میکرو مگا 16 پورت B.0 ، پایه شماره 1) میباشد و خروجی پالس pwm پایه های OC1A و

✓ OC1B (در میکرو مگا 16 به ترتیب پایه های 18 و 19 (portd.4 و d.5)) است همچنین دو پایه ی فوق میتوانند به عنوان

خروجی مد مقایسه ای تایمر مورد استفاده قرار بگیرند.

راه اندازی تایمر یک در محیط بسکام:

تایمر 1 با دستور زیر پیکربندی میشود:

Config Timer1 = Timer , PRESCALE = 1 | 8 | 64 | 256 | 1024

عدد PRESCALE فرکانس (دقت) تایمر را معین میکند. فرکانس و زمانی که تایمر می شمارد از فرمولهای زیر محاسبه میشوند:

$$\text{زمان} = \frac{\text{PRESCALE} \times \text{تایمر بیت}}{\text{مقدار کریستال}}$$

$$\text{فرکانس} = \frac{\text{مقدار کریستال}}{\text{PRESCALE}}$$

بعد از دستور بالا، تایمر با دستور START TIMER شروع به شمارش میکند و با دستور STOP TIMER متوقف می شود. تایمر پس از شمردن تا 65536 (2^{16}) سر ریز میشود ، شما با استفاده از دستور ENABLE OVFI یا ENABLE INTRRUPTS میتوانید وقفه سر ریزی تایمر را راه اندازی کنید . در صورتی که وقفه سر ریزی تایمر فعال باشد ، تایمر پس از سر ریزی به برجسی که با یکی از دستورات ON OVFI LABEL و یا ON TIMER1 LABEL مشخص شده پرش میکند ، باز گشت از وقفه با دستور RETURN انجام میشود.

شما همچنین میتوانید با استفاده از دستور VAR = TIMER1 ، مقدار تایمر را در یک متغیر از جنس word قرار دهید و یا با دستور TIMER1 = VALUE ، مقدار اولیه ای در تایمر قرار دهید تا تایمر از آن شروع به شمارش کند . مثال :

در این مثال میخواهیم توسط تایمر 1 یک ساعت بسازیم ، برای این کار ابتدا باید یک زمان 1 ثانیه ای ایجاد کنیم طبق فرمول اگر شما از کریستال 4 مگا هرتز و PRESCALE ، 64 استفاده نمایید ، میتوانید زمانی برابر با 1.04 ثانیه ایجاد کنید(با مقادیر دیگر نیز میتوانید زمان های دقیق تر بسازید .(بیشترین زمانی که با این تایمر میتوانید بسازید برابر با 67.108864 ثانیه میباشد)).

```
$regfile = "m16def.dat"
```

```
crystal = 12000000$
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Dim San As Byte , Da As Byte , Saat As Byte
```

```
San = 0 : Da = 0 : Saat = 1
```

```
Config Timer1 = Timer , Prescale = 64
```

```
Enable Interrupts
```

```
Enable Timer1
```

```
On Timer1 P
```

```
Start Timer1
```

```
Do
```

```
Locate 1 , 1 : Lcd Saat ; ":" ; Da ; ":" ; San
```

```
Loop
```

```
End
```

```
P:
```

Incr San

If San > 59 Then : Incr Da : San = 0 : End If

If Da > 59 Then : Incr Saat : Da = 0 : End If

If Saat > 12 Then : San = 1 : End If

Return

در برنامه بالا هنگامی که تایمر سرریز میشود به زیر برنامه p پرش میشود در آنجا متغیر ثانیه 1 واحد افزایش می یابد و...

راه اندازی تایمر 1 در مد مقایسه ای (Compare):

در این مد شما میتوانید مقدار شمرده شده توسط تایمر 1 را با دو عدد دلخواه مقایسه کنید و در صورت برابری یا نابرابری

مقدار تایمر با عدد دلخواه ، وضعیت پایه های oc1a و oc1b را تغییر دهید. راه اندازی تایمر یک در مد مقایسه ای با

دستورات زیر انجام میشود:

CONFIG TIMER1= TIMER,COMPARE A = CLEAR |SET|TOGGLE|DISCONNECT,COMPARE B = CLEAR

|SET|TOGGLE|DISCONNECT, PRESCALE=1|8|64|256|1024,CLEAR TIMER =1|0

، COMPARE A = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط تایمر 1 با مقدار COMPARE A ،

که بعدا معرفی میشود برابر شد ، پایه خروجی OC1A می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط

پایه با مد مقایسه ای قطع شود (پایه oc1a به یک ورودی خروجی عادی تبدیل شود).

، COMPARE B = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط تایمر 1 با مقدار COMPARE B ،

که بعدا معرفی میشود برابر شد ، پایه خروجی OC1B می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط

پایه با مد مقایسه ای قطع شود (پایه oc1b به یک ورودی خروجی عادی تبدیل شود).

PRESCALE=1|8|64|256|1024 : عدد PRESCALE فرکانس (دقت) تایمر را معین میکند.

CLEAR TIMER = 1|0 : با انتخاب گزینه 1، محتوای تایمر/کانتر در زمان تطابق مقایسه ای RESET (\$0000) می شود و در

صورت انتخاب 0 مقدار شمرده شده تغییری نمیکند.

با دستورات زیر میتوان عددی را که محتوای تایمر باید با آن مقایسه شود را تعیین کرد

Compare1a = x

Compare1b = x

به جای x یک عدد ثابت یا یک متغیر قرار میگیرد ، هنگامی که عدد شمرده شده توسط تایمر یک با اعداد گذاشته شده برابر شد میکرو وضعیت پایه های مربوطه را همانگونه که در پیکربندی مشخص کردید تغییر میدهد. مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Config Timer1 = Timer , Compare A = Set , Compare B = Toggle , Prescale = 1024 , Clear Timer = 0
```

```
Compare1a = 1000
```

```
Compare1b = 5000
```

```
Do
```

```
Locate 1 , 1 : Lcd Timer1
```

```
Loop
```

```
End
```

در مثال بالا هنگامی که تایمر 1 تا عدد 1000 شمرده پایه oc1a یک میشود و هنگامی که مقدار شمرده شده به 5000 رسید پایه oc1b یک میشود. (oc1a پایه 19 و oc1b پایه 18 مگا 16 میباشد).

استفاده از وقفه مد مقایسه ای تایمر 1 :

مد مقایسه ای دارای 2 منبع وقفه میباشد که با دستورات زیر فعال میشوند:

```
Enable Interrupts
```

```
Enable Oc1a
```

```
Enable Oc1b
```

با دستور فعال سازی وقفه مقایسه ، هنگامی که مقدار شمرده شده توسط میکرو با Compare1a برابر شد cpu میکرو با دستور on oc1a lable به برچسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد، هنگامی که مقدار شمرده شده توسط میکرو با Compare1b برابر شد cpu میکرو با دستور on oc1b lable به برچسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد. در صورتی که در پایان برچسب دستور return گذاشته شود cpu به حلقه ی اصلی پرش میکند . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2

Config Timer1 = Timer , Compare A = Set , Compare B = Toggle , Prescale = 1024 , Clear Timer = 0

Compare1a = 2120

Compare1b = 63000

Enable Interrupts

Enable Oc1a

Enable Oc1b

On Oc1a Q

On Oc1b W

Do

Locate 1 , 1 : Lcd Timer1

Loop

End

Q:

Locate 2 , 1 : Lcd Timer1

Return

W:

Locate 2 , 8 : Lcd Timer1

Return

در مثال بالا هنگامی که رقم شمرده شده توسط تایمر با مقدار Compare1a (2120) برابر شد cpu به برچسب q پرش میکند و

در آنجا مقدار شمرده شده توسط تایمر را بر روی lcd نمایش داده و با دستور return به حلقه ی اصلی برمیگردد ، هنگامی

که رقم شمرده شده توسط تایمر با مقدار Compare1b (63000) برابر شد cpu به برچسب W پرش میکند و در آنجا مقدار

شمرده شده توسط تایمر را بر روی lcd نمایش داده و با دستور return به حلقه ی اصلی برمیگردد .همچنین در مورد اول پایه

oc1a یک میشود و در مورد دوم وضعیت پایه oc1b تغییر میکند (در صورت 0 بودن یک میشود و بلعکس).

راه اندازی تایمر 1 در مد CAPTURE :

در صورتی که تایمر یک را در این مد پیکربندی کنید ، با اعمال یک پالس بالا رونده یا پایین رونده (که نوع آن در هنگام

پیکربندی مشخص میشود) به پایه icp (پایه 20 میکرو کنترلر مگا 16 (ATMEGA16)) ، در همان لحظه مقدار شمرده شده

توسط تایمر 1 در ریجیستر CAPTURE قرار میگیرد ، محتوای ریجیستر CAPTURE را می توان با دستور VAR = CAPTURE در یک متغیر از جنس word قرار داد .

راه اندازی تایمر یک در مد CAPTURE با دستورات زیر انجام میشود:

Config Timer1 = Timer, Capture Edge= Falling | Rising ,Noise Cancel=1|0 , Prescale =1|8|64|256|1024

Capture Edge= Falling | Rising : این گزینه مشخص میکند ، پالس اعمالی به پایه icp بالا رونده (Falling) است یا پایین رونده (Rising). (CAPTURE) به کدام پالس حساسیت نشان میدهد).

Noise Cancel=1|0 : در صورت انتخاب یک از نویز های موجود بر روی پایه icp چشم پوشی میشود و هر پالس با لبه ی تعیین شده میتواند CAPTURE را راه اندازی کند ، در صورت انتخاب صفر فقط پالس های با دامنه ی 5 ولت قادر به راه اندازی CAPTURE خواهند بود.

Prescale =1|8|64|256|1024: دقت تایمر را نشان میدهد (با استفاده از این مقدار و مقدار کریستال میتوانید زمان شمرده شده

توسط تایمر تا هنگام سرریزی را با فرمولی که در بالا گفته شد محاسبه کنید)

با دستور Enable Icp1 این مد فعال میشود و پایه icp آماده دریافت پالس میگردد . مثال:

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Config Timer1 = Timer , Capture Edge = Falling , Noise Cancel = 1 , Prescale = 1024
```

```
Do
```

```
Locate 1 , 1 : Lcd Timer1
```

```
Locate 2 , 1 : Lcd Capture1
```

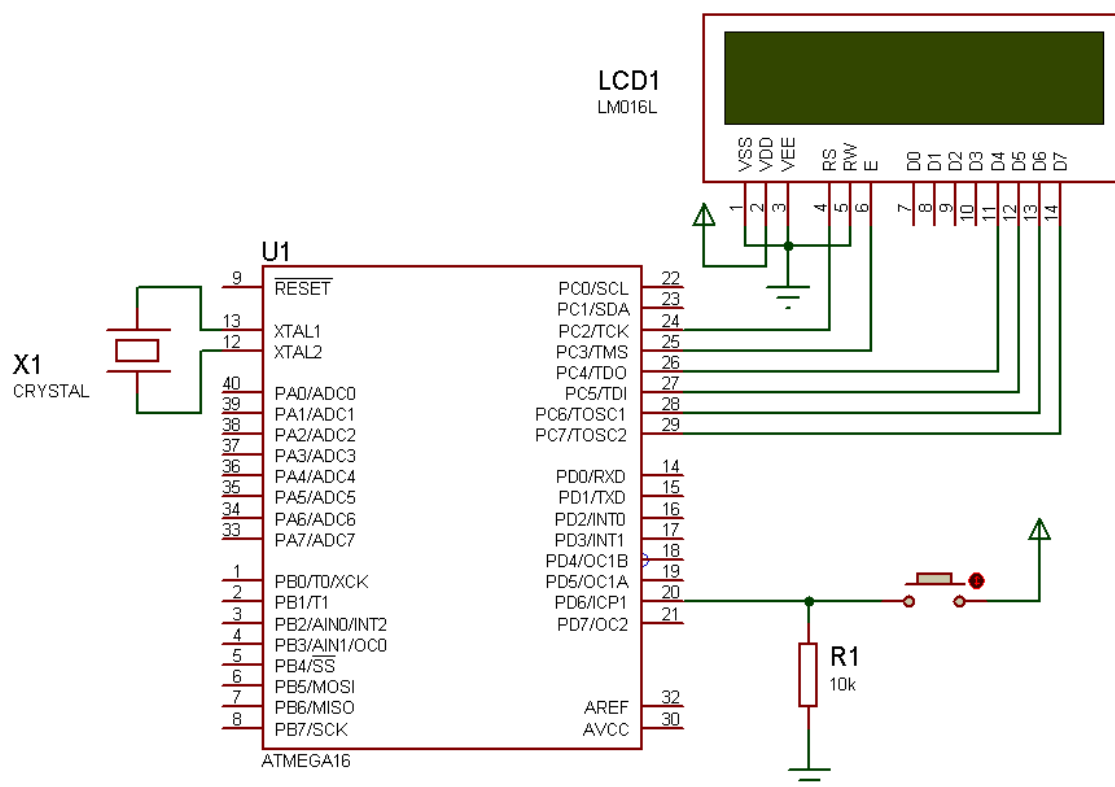
```
Loop
```

```
End
```

در مثال بالا هنگامی که یک پالس بالا رونده به پایه icp اعمال میشود مقدار شمرده شده توسط تایمر 1 درون ریجیستر

Capture قرار میگیرد و سپس بر روی lcd نمایش داده میشود. مدار استفاده شده برای مثال بالا و سایر مثال های این بخش را

مشاهده می فرمایید:



استفاده از وقفه مد Capture تایمر 1:

مد Capture تایمر یک نیز مانند دیگر مد ها دارای یک منبع وقفه است که شما میتوانید با دستور `Enable Icp1` آن را فعال

کنید تا با دستور `On icp1 lable`، هنگامی که پالسی به پایه icp اعمال شد ، cpu به برچسب مورد نظر پرش کند و در آنجا

عملیات دلخواه را انجام دهد . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Config Timer1 = Timer , Capture Edge = Falling , Noise Cancel = 1 , Prescale = 1024
```

```
Enable Interrupts
```

```
Enable Icp1
```

```
On Icp1 Q
```

```
Dim A As Byte
```

```
Do
```

Locate 1 , 1

Lcd Timer1

Loop

Q:

Locate 2 , A : Lcd Capture1

A = A + 4

Return

End

در مثال بالا ، هنگامی که پالسی به پایه icp اعمال شود مقدار شمرده شده توسط تایمر 1 در ریجیستر Capture1 ریخته میشود و cpu به برچسب q برش میکند و در آنجا مقدار Capture1 را در ستون های مختلف lcd نمایش میدهد.

راه اندازی کانتر 1 در محیط بسکام: (راه اندازی تایمر-کانتر 1 در مد کانتر در محیط بسکام)

کانتر 1 در بسکام با دستور زیر پیکربندی میشود:

CONFIG TIMER1 = COUNTER , EDGE = RISING / FALLING

با انتخاب EDGE = RISING کانتر نصب به لبه ی بالا رونده حساس است

با انتخاب EDGE = FALLING کانتر نصب به لبه ی پایین رونده حساس است

(لبه های بالا رونده یا پایین رونده را میشمارد (بالا رونده < صفر به یک و پایین رونده > یک به صفر))

شما همچنین میتوانید با استفاده از دستور VAR = COUNTER1 ، مقدار کانتر را در یک متغیر از جنس word قرار دهید و یا با

دستور COUNTER1 = VALUE ، مقدار اولیه ای را در کانتر قرار دهید تا کانتر از آن شروع به شمارش کند. کانتر نیز مانند تایمر

پس از شمردن تا 65536 سر ریز میشود ، شما با استفاده از دستور ENABLE OV1F1 میتوانید وقفه سر ریزی کانتر را راه اندازی

کنید . در صورتی که وقفه سر ریزی کانتر فعال باشد ، کانتر پس از سر ریزی به برچسبی که با یکی از دستورات ON OV1F1

LABLE مشخص شده پرش میکند ، باز گشت از وقفه با دستور RETURN انجام میشود. . مثال :

در این مثال کانتر یک تعداد پالسهای اعمالی به پایه t1 (پایه شماره 2 مگا 16) را میشمارد.

\$regfile = "m16def.dat"

crystal = 12000000\$

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5

Config Timer1 = Counter , Edge = Falling

Do

Locate 1 , 1 : Lcd Counter1

Loop

End

کانتر یک را نیز میتوان مانند تایمر در دو مد مقایسه ای و Capture راه اندازی کرد که طریقه راه اندازی در ادامه آورده شده است .

راه اندازی کانتر 1 در مد مقایسه ای (Compare):

در این مد شما میتوانید مقدار کانتر 1 را با دو عدد دلخواه مقایسه کنید و در صورت برابری یا نابرابری مقدار شمرده شده توسط کانتر با مقدار دلخواه وضعیت پایه های OC1A و OC1B را تغییر دهید. راه اندازی کانتر یک در مد مقایسه ای با دستورات زیر انجام میشود:

CONFIG TIMER1 = COUNTER , EDGE = RISING / FALLING, COMPARE A = CLEAR |SET|TOGGLE|

DISCONNECT,COMPARE B = CLEAR |SET|TOGGLE|DISCONNECT,CLEAR TIMER = 1|0

EDGE = RISING / FALLING : با انتخاب EDGE = RISING کانتر نصب به لبه ی بالا رونده حساس است و با انتخاب

FALLING کانتر نصب به لبه ی پایین رونده حساس است.

COMPARE A = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط کانتر 1 با مقدار COMPARE A ،

که بعدا معرفی میشود ، برابر شد ، پایه خروجی OC1A می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط

پایه با مد مقایسه ای قطع شود (پایه OC1A به یک ورودی خروجی عادی تبدیل شود).

COMPARE B = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط کانتر 1 با مقدار COMPARE B ،

که بعدا معرفی میشود ، برابر شد ، پایه خروجی OC1B می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط

پایه با مد مقایسه ای قطع شود (پایه OC1B به یک ورودی خروجی عادی تبدیل شود).

CLEAR TIMER = 1|0 : با انتخاب گزینه 1 ، محتوای کانتر 1 در زمان تطابق مقایسه ای RESET (\$0000) می شود و در صورت

انتخاب 0 مقدار شمرده شده تغییری نمیکند.

با دستورات زیر میتوان عددی که محتوای تایمر باید با آن مقایسه شود را تعیین کرد :

Compare1a = x

Compare1b = x

به جای x یک عدد ثابت یا یک متغیر قرار میگیرد ، هنگامی که عدد شمرده شده توسط کانتر یک با اعداد گذاشته شده برابر شد میکرو وضعیت پایه های مربوطه را همانگونه که در پیکربندی مشخص کردید تغییر میدهد. مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Config Timer1 = Counter , Edge = Falling , Compare A = Set , Compare B = Set , Clear Timer = 0
```

```
Compare1a = 300
```

```
Compare1b = 400
```

```
Do
```

```
Locate 1 , 1
```

```
Lcd Counter1
```

```
Loop
```

```
End
```

در مثال بالا هنگامی که پالس های شمرده شده توسط کانتر 1 به 300 رسید ، پایه oc1a یک میشود و هنگامی که مقدار شمرده شده به 400 رسید پایه oc1b یک میشود (oc1a پایه 19 و oc1b پایه 18 مگا 16 میباشد). (ورودی کانتر 1 پایه t1 پایه 2 مگا 16 است) .

استفاده از وقفه مد مقایسه ای کانتر 1 :

مد مقایسه ای دارای 2 منبع وقفه میباشد که با دستورات زیر فعال میشوند:

```
Enable Interrupts
```

```
Enable Oc1a
```

```
Enable Oc1b
```

با دستور فعال سازی وقفه مقایسه ، هنگامی که تعداد پالس شمرده شده توسط کانتر با Compare1a برابر شد cpu میکرو با دستور on oc1a lable به برچسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد، هنگامی که تعداد پالس

شمردن شده توسط کانتر با Compare1b برابر شد cpu میکرو با دستور on oc1b lable به برچسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد. در صورتی که در پایان برچسب دستور return گذاشته شود cpu به حلقه ی اصلی پرش میکند . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Config Timer1 = Counter , Edge = Falling , Compare A = Set , Compare B = Set , Clear Timer = 0
```

```
Compare1a = 30
```

```
Compare1b = 40
```

```
Enable Interrupts
```

```
Enable Oc1a
```

```
Enable Oc1b
```

```
On Oc1a Q
```

```
On Oc1b W
```

```
Do
```

```
Locate 1 , 1
```

```
Lcd Counter1
```

```
Loop
```

```
End
```

```
Q:
```

```
Cls
```

```
Locate 2 , 1
```

```
Lcd " Counter1=30"
```

```
Return
```

```
W:
```

```
Cls
```

```
Locate 2 , 1
```

```
Lcd " Counter1=40"
```

```
Return
```

در مثال بالا هنگامی که تعداد پالس شمرده شده توسط کانتر با مقدار Compare1a (30) برابر شد cpu به برچسب q پرش میکند و در آنجا عبارت "Counter1=30" را بر روی lcd نمایش داده و با دستور return به حلقه ی اصلی برمیگردد ، هنگامی که تعداد پالس شمرده شده توسط کانتر با مقدار Compare1b (40) برابر شد cpu به برچسب W پرش میکند و در آنجا عبارت "Counter1=40" را بر روی lcd نمایش داده و با دستور return به حلقه ی اصلی برمیگردد .همچنین در مورد اول پایه oc1a یک میشود و در مورد دوم وضعیت پایه oc1b .

راه اندازی کانتر 1 در مد CAPTURE :

در صورتی که کانتر یک را در این مد پیکربندی کنید ، با اعمال یک پالس بالا رونده یا پایین رونده (که نوع آن در هنگام پیکربندی مشخص میشود) به پایه icp (پایه 20 مگا 16) ، در همان لحظه تعداد پالس شمرده شده توسط کانتر 1 در رجیستر CAPTURE قرار میگیرد ، محتوای رجیستر CAPTURE را می توان با دستور VAR = CAPTURE در یک متغیر از جنس word قرار داد .راه اندازی کانتر یک در مد CAPTURE با دستورات زیر انجام میشود:

Config Timer1 = Counter , Edge= Falling|Rising , Capture Edge=_ Falling | Rising , Noise Cancel=1|0 ,

EDGE = Edge= Falling|Rising : با انتخاب EDGE = RISING کانتر نصب به لبه ی بالا رونده حساس است و با انتخاب FALLING کانتر نصب به لبه ی پایین رونده حساس است.

Capture Edge= Falling | Rising : این گزینه مشخص میکند ، پالس اعمالی به پایه icp بالا رونده (Falling) است یا پایین رونده (Rising). (CAPTURE به کدام پالس حساسیت نشان میدهد).

Noise Cancel=1|0 : در صورت انتخاب یک از نویز های موجود بر روی پایه icp چشم پوشی میشود و هر پالس با لبه ی تعین شده میتواند CAPTURE را راه اندازی کند ،در صورت انتخاب صفر فقط پالس های با دامنه ی 5 ولت قادر به راه اندازی CAPTURE خواهند بود. مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```

```
Config Timer1 = Counter , Edge = Falling , Capture Edge = Rising , Noise Cancel = 0
```

```
Do
```

Locate 1 , 1 : Lcd Counter1

Locate 2 , 1 : Lcd Capture1

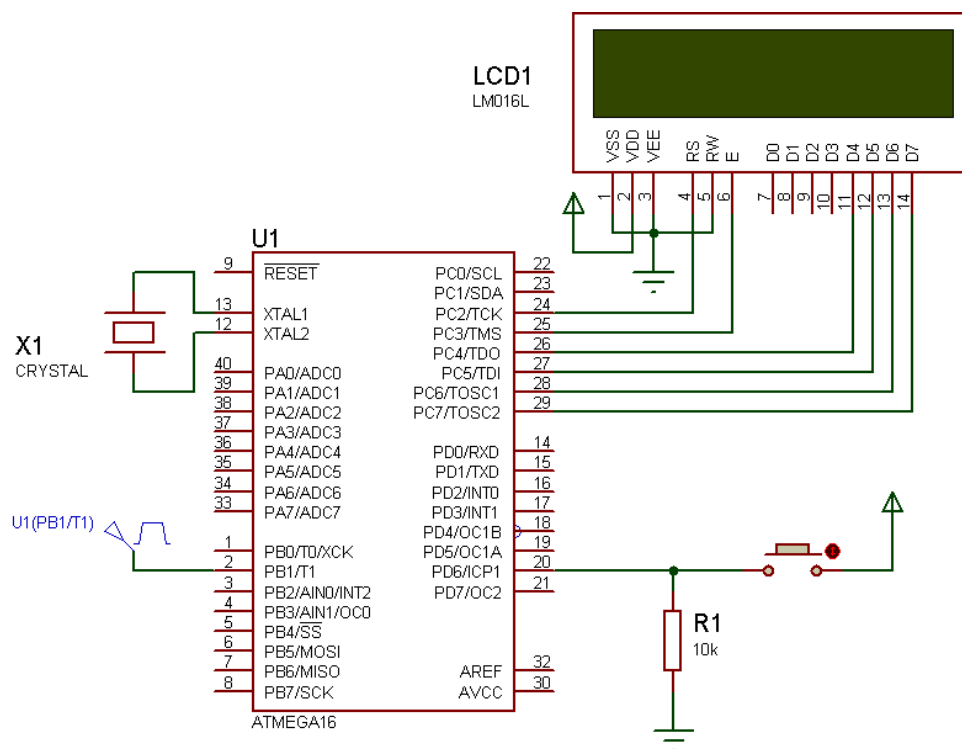
Loop

End

در مثال بالا هنگامی که یک پالس بالا رونده به پایه icp اعمال میشود مقدار شمرده شده توسط تایمر 1 درون رجیستر

Capture قرار میگیرد و سپس بر روی lcd نمایش داده میشود. مدار استفاده شده برای مثال بالا و سایر مثال های این بخش را

مشاهده می فرمایید:



استفاده از وقفه مد Capture کانتر 1:

مد Capture کانتر یک نیز مانند دیگر مد ها دارای یک منبع وقفه است که شما میتوانید با دستور Icp1 Enable آن را فعال

کنید تا با دستور On icp1 lable ، هنگامی که پالسی به پایه icp اعمال شد ، cpu به برجسب مورد نظر پرش کند و در آنجا

عملیات دلخواه را انجام دهد . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2
```


Config Timer1 = Counter , Edge = Falling , Capture Edge = Rising , Noise Cancel = 0

Enable Interrupts

Enable Icp1

On Icp1 Q

Do

Locate 1 , 1

Lcd Counter1

Loop

End

Q:

Locate 2 , 1

Lcd Capture1

Return

در مثال بالا ، هنگامی که پالسی به پایه icp اعمال شود ، تعداد پالس شمرده شده توسط کانتر 1 در ریجیستر Capture1 ریخته میشود و cpu به برچسب q برش میکند و در آنجا مقدار Capture1 را بر روی lcd نمایش میدهد.

پیکر بندی تایمر-کانتر یک در مد PWM :

pwm یا مدولاسیون پهنای پالس یکی از موارد پر کاربرد در میکرو کنترلر ها میباشد ، با استفاده از این بخش میتوانیم یک موج مربعی با دیوتی سایکل و فرکانس قابل کنترل برای کنترل دور موتورهای مختلف ، ساخت پالس مربعی و دیگر پالس ها ، درایو کردن قطعات الکترونیک صنعتی ، ایجاد ولتاژ متغیر DC و... ایجاد کنیم ، در این نوع مدولاسیون ، دامنه پالس ثابت بوده و کنترل نسبت زمان صفر به یک (دیوتی سایکل) و فرکانس به عهده ی کاربر گذاشته میشود . برای درک بیشتر موضوع بهتر است کلیه مثالها را اجرا کنید . راه اندازی تایمر/کانتر 1 در مد pwm با دستورات زیر انجام میشود :

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

pwm میتواند 8 یا 9 یا 10 بیتی باشد که مقدار بیت هرچه بیشتر باشد دقت موج بیشتر است (تعداد پله بیشتر است

(pwm 8 بیتی تا 256 سرریز میشود (شما میتوانید 256 واحد آن را کم یا زیاد کنید) pwm 9 بیتی تا 512 و pwm 10 بیتی تا

1024 سرریز میشود.

Compare A Pwm=Clear Up |Clear Down |Disconnect : در صورت استفاده از گزینه Clear Up ، موج pwm از سطح 1 شروع

میشود و در صورت انتخاب Clear Down ، موج pwm از سطح صفر شروع میشود و در صورت انتخاب Disconnect ،

هنگامی که مقدار pwm با pwm1a که در برنامه مشخص میشود برابر شد ، ارتباط پالس با پایه ی oc1a قطع میشود .

Compare b Pwm=Clear Up |Clear Down |Disconnect : در صورت استفاده از گزینه Clear Up ، موج pwm از سطح 1 شروع

میشود و در صورت انتخاب Clear Down ، موج pwm از سطح صفر شروع میشود و در صورت انتخاب Disconnect ،

هنگامی که مقدار pwm با pwm1b که در برنامه مشخص میشود برابر شد ، ارتباط پالس با پایه ی oc1b قطع میشود .

Prescale : این گزینه و مقدار کریستال در تعیین فرکانس pwm نقش دارند . برای تولید PWM با فرکانس های متفاوت از این

گزینه ها استفاده می شود.

با استفاده از دو دستور زیر میتوان یک عدد ثابت یا متغیر را در رجیستر pwm قرار داد تا مقدار pwm با آنها مقایسه شود:

Pwm1a=x

Pwm1b=x

یا

COMPARE1A = x

COMPARE1B = x

مثال:

\$regfile = "m16def.dat"

\$crystal = 8000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3 , Rs = Portc.2

Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Up , Compare B Pwm = Clear Down , Prescale = 1

Dim A As Byte

Dim B As Byte

Config portd = output

Do

Pwm1a = A

Pwm1b = B

Incr A

Incr B

Waitms 500

Loop

End

در مثال بالا مقدار دو متغیر a و b در رجیستر pwm قرار داده شده اند ، مقدار آنها هر یک میلی ثانیه افزایش میابد . در این حالت زمان تناوب پالس pwm از رابطه ی زیر بدست می آید:

$$\text{زمان} = \frac{PRESCALE * 2 * \text{تایمر بیت}}{\text{مقدار کریستال}}$$

زمان / 1 = فرکانس

راه اندازی تایمر-کانتر دو در محیط بسکام:

تایمر - کانتر 8 بیتی شماره ی 2 که در اکثر میکرو کنترلر های AVR وجود دارد دارای قابلیت های متفاوتی نسبت به تایمر-کانتر 1 و 0 میباشد ، از این تایمر-کانتر که ویژگی های آن را در زیر مشاهده میکنید بیشتر برای ساختن زمان های دقیق استفاده میشود .

✓ این تایمر / کانتر 8 بیتی است ، و نهایتا میتواند تا 2^8 (255) بشمارد.

✓ کلاک این تایمر میتواند توسط نوسان ساز داخلی یا نوسان ساز خارجی **یا از دو پایه $toc1$ و $toc2$** (در مگا 16 پایه

شماره 28 و 29 ($portc.6$ و $portc.7$) **توسط کریستال 32768 هرتز، تامین شود** (مقدار فرکانس نوسان ساز بر عدد PRESCALE تقسیم میشود).

✓ تایمر - کانتر دو دارای یک خروجی مقایسه ای است که جیستر ocr2 مقدار مقایسه ای را در خود جای می دهد و با محتوای تایمر/کانتر مقایسه می کند .

✓ در زمان تساوی محتوای رجیستر مقایسه و محتوای تایمر/کانتر ، وضعیت پایه های خروجی مد مقایسه ای ocr2 همان گونه که در برنامه مشخص میشود ، می تواند تغییر کند.

✓ این تایمر دارای چندین منبع وقفه میباشد که شما میتوانید آنها را در هر قسمت از برنامه فعال یا غیر فعال کنید . از این منابع وقفه میتوان ، وقفه سرریزی را نام برد.

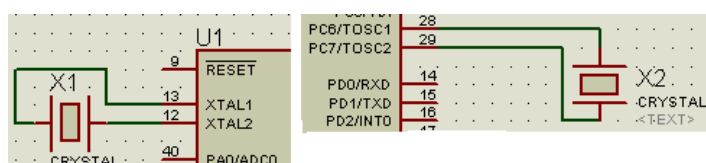
- ✓ این تایمر/ کانتر میتواند در مد تایمر و کانتر و pwm راه اندازی شود. (در بعضی از میکروها مانند 8535 و مگا 32 و... مد کانتر وجود ندارد، برای کسب اطلاعات بیشتر در مورد امکانات avr به دیتاشیت آنها مراجعه کنید).
- ✓ ورودی کانتر پایه T2 (در میکرو مگا 103 پورت d.7، پایه شماره 32) میباشد و خروجی پالس pwm پایه oc2 (در میکرو مگا 16 پایه 21 (portd.7)) است همچنین پایه ی فوق میتواند به عنوان خروجی مد مقایسه ای تایمر نیز مورد استفاده قرار گیرد.

راه اندازی تایمر دو در محیط بسکام:

تایمر دو با دستور زیر پیکر بندی میشود:

```
CONFIG TIMER2 = TIMER , ASYNC = ON|OFF , PRESCALE=1|8|32|64|128|256|1024
```

ASYNC = ON|OFF : زمانی که ON انتخاب شود، تایمر کلاک مورد نیاز را از پایه های TOSC1 و TOSC2 با کریستال 32768HZ در یافت می کند. در این حالت با PRESCALE=128 دقیقاً تایمر بعد از یک ثانیه سرریز میشود، شما با استفاده از وقفه ها میتوانید به یک برجسب پرش کرده و یک متغیر را بیافزاید و یک ساعت دقیق بسازید، در صورت انتخاب off تایمر کلاک خود را از کریستال داخلی یا کریستال متصل به دو پایه xtal1 و xtal2 تامین میکند.



عدد PRESCALE فرکانس (دقت) تایمر را معین میکند. فرکانس و زمانی که تایمر می شمارد از فرمولهای زیر محاسبه میشوند: (ممکن است در بعضی از میکروها اعداد 32 و 128 موجود نباشد)

$$\text{زمان} = \frac{\text{PRESCALE} * \text{تایمر بیت}}{\text{مقدار کریستال}}$$

$$\text{فرکانس} = \frac{\text{مقدار کریستال}}{\text{PRESCALE}}$$

مثالاً با استفاده از کریستال 32768HZ و PRESCALE=128 زمانی که تایمر می شمارد برابر است با:

$$\text{زمان} = \frac{2^8 * 128}{32768} = 1s$$

بعد از دستور بالا، تایمر با دستور START TIMER شروع به شمارش میکند و با دستور STOP TIMER متوقف می شود. تایمر پس از شمردن تا 255 سر ریز میشود. شما با استفاده از دستور ENABLE OV2 و ENABLE INTRRUPTS میتوانید وقفه سر ریزی تایمر را راه اندازی کنید. در صورتی که وقفه سر ریزی تایمر فعال باشد، تایمر پس از سر ریزی به برجسی که با یکی از دستورات ON OV2 LABEL و یا ON TIMER2 LABEL مشخص شده پرش میکند، باز گشت از وقفه با دستور RETURN انجام میشود.

شما همچنین میتوانید با استفاده از دستور VAR = TIMER2، مقدار تایمر را در یک متغیر از جنس بایت قرار دهید و یا با دستور TIMER2 = VALUE، مقدار اولیه ای در تایمر قرار داد تا تایمر از آن شروع به شمارش کند. مثال:

```
$regfile = "M16DEF.DAT"
```

```
$crystal = 1000000
```

```
Config Lcdpin = Pin , Db4 = Pinc.2 , Db5 = Pinc.3 , Db6 = Pinc.4 , Db7 = Pinc.5 , E = Pinc.1 , Rs = Pinc.0
```

```
Config Lcd = 16 * 2
```

```
Dim A As Byte , B As Byte
```

```
Config Timer2 = Timer , Prescale = 1024
```

```
Enable Interrupts
```

```
Enable Ovf2
```

```
On Ovf2 R
```

```
Timer2 = 12
```

```
Start Timer2
```

```
Do
```

```
Loop
```

```
End
```

```
R:
```

```
Incr A
```

```
If A > 3 Then : Incr B : Locate 1 , 1 : Lcd B : A = 0 : End If
```

```
Return
```

در مثال بالا توسط تایمر 2 زمان 0.999424 ثانیه ایجاد شده است (مقدار اولیه 12 در تایمر قرار داده شده ، پس تایمر از 12 تا 256 می‌شمارد ، یعنی 244 ، پس در فرمول به جای بیت تایمر (256) عدد 244 قرار می‌گیرد و زمان فوق بدست می‌آید ، شما می‌توانید با این روش زمان های دیگری را نیز بدست آورید.)

راه اندازی تایمر 2 در مد مقایسه ای (Compare):

تایمر دو با دستور زیر در مد مقایسه ای پیکر بندی میشود (در این مد مقدار شمرده شده توسط تایمر با عددی که شما در برنامه تعیین میکنید ، مقایسه میشود و در صورت اختلاف مقدار شمرده شده با عدد شما وضعیت پایه oc2 (پایه 21 مگا 16) مطابق آنچه که در برنامه تعیین کردید تغییر میکند)

```
CONFIG TIMER2 = TIMER ,COMPARE = CLEAR |SET|TOGGLE|DISCONNECT,PRESCALE = 1|8 |32|64 |256|1024,CLEAR
TIMER=_ 1|0
```

COMPARE = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط تایمر 2 با مقدار COMPARE ، که بعدا معرفی میشود ، برابر شد ، پایه خروجی oc2 می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط پایه با مد مقایسه ای قطع شود (پایه oc2 به یک ورودی خروجی عادی تبدیل شود).

PRESCALE = 1|8 |32|64 |256|1024: عدد PRESCALE فرکانس (دقت) تایمر را معین میکند.(ممکن است در بعضی از میکرو

ها اعداد 32 و 128 موجود نباشد)

CLEAR TIMER = 1|0 : با انتخاب گزینه 1 ، محتوای تایمر/کانتر در زمان تطابق مقایسه ای RESET (\$0000) می شود و در صورت انتخاب 0 مقدار شمرده شده تغییری نمیکند.

با دستورات زیر میتوان عددی را که محتوای تایمر باید با آن مقایسه شود را تعیین کرد :

Compare1 = x

یا

Ocr2=x

به جای x یک عدد ثابت یا یک متغیر قرار می‌گیرد ، هنگامی که عدد شمرده شده توسط تایمر یک با اعداد گذاشته شده برابر شد میکرو وضعیت پایه های مربوطه را همانگونه که در پیکربندی مشخص کردید تغییر میدهد . مثال:

```

$regfile = "M16DEF.DAT"

$crystal = 1000000

Config Lcdpin = Pin , Db4 = Pinc.2 , Db5 = Pinc.3 , Db6 = Pinc.4 , Db7 = Pinc.5 , E = Pinc.1 , Rs = Pinc.0

Config Lcd = 16 * 2

Config Timer2 = Timer , Compare = Toggle , Prescale = 64 , Clear Timer = 0

Ocr2 = 100

Start Timer2

Config Portd = Output

Do

Locate 1 , 1

Lcd Timer2

Loop

End

```

در مثال بالا هنگامی که مقدار شمرده شده با مقدار Compare (100) برابر شد وضعیت پایه oc2 تغییر میکند.

استفاده از وقفه مد مقایسه ای تایمر 2 :

مد مقایسه ای تایمر دو دارای 1 منبع وقفه میباشد که با دستورات زیر فعال میشوند:

Enable Interrupts

Enable Oc2

با دستور فعال سازی وقفه مقایسه ، هنگامی که مقدار شمرده شده توسط میکرو با Compare (یا ocr2) برابر شد cpu میکرو با

دستور on oc2 lable به برچسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد، در صورتی که در پایان

برچسب دستور return گذاشته شود cpu به حلقه ی اصلی پرش میکند . مثال :

```

$regfile = "M16DEF.DAT" : $crystal = 4000000

Config Lcdpin = Pin , Db4 = Pinc.2 , Db5 = Pinc.3 , Db6 = Pinc.4 , Db7 = Pinc.5 , E = Pinc.1 , Rs = Pinc.0

Config Lcd = 16 * 2

Config Timer2 = Timer , Compare = Toggle , Prescale = 128 , Clear Timer = 1

Ocr2 = 125 : Cursor Off

Dim A As Byte , B As Byte , C As Byte , D As Byte

Enable Interrupts

Enable Oc2

```

```

On Oc2 Q
Start Timer2
Config Portd = Output : D = 1 : C = 0 : B = 0
Do
Locate 1 , 1
Lcd D ; " " ; C ; " " ; B ; " "
Loop
End
Q:
Incr A
If A > 249 Then : Incr B : A = 0 : End If
If B > 59 Then : Incr C : B = 0 : End If
If C > 49 Then : Incr D : C = 0 : End If
If D > 12 Then : D = 1 : End If
Return

```

در مثال بالا با استفاده از مد مقایسه ای برنامه یک ساعت نوشته شده است ، این ساعت دقیق است و زمان ساخته شده توسط تایمر دقیقا یک ثانیه میباشد . در برنامه بالا هنگامی که تایمر تا 125 می‌شمارد (مقدار شمرده شده (بیت تایمر= 125) مقدار آن با مقدار مقایسه ای (Ocr2) برابر میشود ، بنابراین cpu میکرو به برچسب q پرش کرده و در آنجا یک واحد به متغیر a افزوده میشود ، هنگامی که a برابر با 250 شد، یک واحد به متغیر ثانیه افزوده میشود و مقدار a صفر شده و موارد گفته شده تکرار میگردد ، هنگامی که b=60 شد (ثانیه 60 شد) به متغیر دقیقه یک واحد افزوده میشود و مقدار b صفر میشود ، این کار برای متغیر ساعت نیز تکرار میگردد ...

مقدار کریستال / (مقداری که توسط تایمر شمرده میشود (بیت تایمر * تعداد حلقه) * پریسکیلر) = زمان

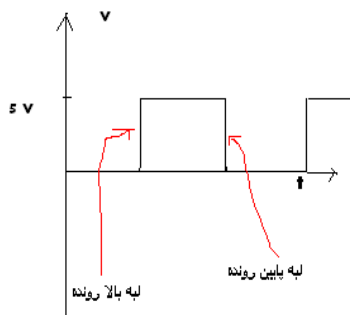
$$T = (128 (250 * 125)) / 4000000 = 1s$$

شما میتوانید زمانهای گوناگون را با روش های مختلف بسازید .

< راه اندازی کانتر دو در محیط بسکام:

کانتر دو نیز عملکردی مانند کانتر های 0 و 1 دارد ، این کانتر در بسکام با دستور زیر پیکربندی میشود:

CONFIG TIMER2 = COUNTER , EDGE = RISING / FALLING



با انتخاب EDGE = RISING کانتر نصب به لبه ی بالا رونده حساس است.

با انتخاب EDGE = FALLING کانتر نصب به لبه ی پایین رونده حساس است.

(لبه های بالا رونده یا پایین رونده را می‌شمارد (بالا رونده < صفر به یک و پایین رونده >

یک به صفر))

شما همچنین می‌توانید با استفاده از دستور VAR = COUNTER2 ، مقدار کانتر را در یک متغیر از جنس بایت قرار دهید و یا با دستور COUNTER2 = VALUE ، مقدار اولیه ای در کانتر قرار دهید تا کانتر از آن شروع به شمارش کند.

کانتر نیز مانند تایمر پس از شمردن تا 255 سر ریز میشود ، شما با استفاده از دستور ENABLE_OVF2 و Enable Interrupts می‌توانید وقفه سر ریزی کانتر را راه اندازی کنید . در صورتی که وقفه سر ریزی کانتر فعال باشد ، کانتر پس از سر ریزی به برجسی که با یکی از دستورات ON_OVF2_LABEL و یا ON_COUNTER2_LABEL مشخص شده پرش میکند ، باز گشت از وقفه با دستور RETURN انجام میشود.(این حالت در مواردی که رقم شمرده شده توسط کانتر از 255 بیشتر میشود ، کاربرد دارد) .

مثال :

در این مثال کانتر دو تعداد پالسهای اعمالی به پایه t2 (32 میکرو مگا 103) را می‌شمارد و بر روی lcd نمایش میدهد:

```
$regfile = "M103DEF.DAT" : $crystal = 4000000
```

```
Config Lcdpin = Pin , Db4 = Pinb.2 , Db5 = Pinb.3 , Db6 = Pinb.4 , Db7 = Pinb.5 , E = Pinb.1 , Rs = Pinb.0
```

```
Config Lcd = 16 * 2
```

```
Config Timer2 = Counter , Edge = Rising
```

```
ENABLE_TIMER2
```

```
Do
```

```
Locate 1 , 1
```

```
Lcd Counter2
```

```
Loop
```

```
End
```

مثال:

```
$regfile = "M103DEF.DAT" : $crystal = 4000000
```

```
Config Lcdpin = Pin , Db4 = Pinb.2 , Db5 = Pinb.3 , Db6 = Pinb.4 , Db7 = Pinb.5 , E = Pinb.1 , Rs = Pinb.0
```

```
Config Lcd = 16 * 2
```

```
Dim A As Word
```

```
Config Timer2 = Counter , Edge = Rising
```

```
Enable Timer2
```

```
Enable Interrupts
```

```
Enable Ovf2
```

```
On Counter2 Q
```

```
Do
```

```
Locate 1 , 1
```

```
Lcd Counter2
```

```
Loop
```

```
End
```

```
Q:
```

```
A = A + Counter2 : Locate 2 , 1 : Lcd A
```

```
Return
```

در مثال بالا از وقفه کانتر استفاده شده است ، هنگامی که کانتر سرریز شد (255 پالس اعمای را شمرد) به زیر برنامه q پرش میشود و در آنجا مقدار شمرده شده توسط کانتر با متغیر a جمع میشود و متغیر a در سطر اول ستون دوم lcd نمایش داده میشود و... ، بدین ترتیب کانتر میتواند تا 65536 پالس را بشمارد.

راه اندازی کانتر 2 در مد مقایسه ای (Compare):

در این مد شما میتوانید مقدار کانتر 2 را با یک عدد دلخواه مقایسه کنید و در صورت برابری یا نابرابری مقدار شمرده شده توسط کانتر با مقدار دلخواه وضعیت پایه oc2 را تغییر دهید. راه اندازی کانتر دو در مد مقایسه ای با دستورات زیر انجام میشود:

```
CONFIG TIMER2 = COUNTER , EDGE = RISING / FALLING, COMPARE = CLEAR |SET|TOGGLE| DISCONNECT, CLEAR
```

```
TIMER = 1|0
```

```
EDGE = RISING / FALLING : با انتخاب EDGE = RISING کانتر نصب به لبه ی بالا رونده حساس است و با انتخاب
```

FALLING کانتر نصب به لبه ی پایین رونده حساس است.

COMPARE = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط کانتر دو با مقدار COMPARE ، که

بعداً معرفی میشود، برابر شد، پایه خروجی OC2 می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط پایه با مد مقایسه ای قطع شود (پایه OC2 به یک ورودی خروجی عادی تبدیل شود).

CLEAR TIMER = 1|0 : با انتخاب گزینه 1، محتوای کانتر دو در زمان تطابق مقایسه ای RESET (\$0000) می شود و در صورت انتخاب 0 مقدار شمرده شده تغییری نمیکند.

با دستورات زیر میتوان عددی که محتوای تایمر باید با آن مقایسه شود را تعیین کرد

Ocr2=x

به جای x یک عدد ثابت یا یک متغیر قرار میگیرد، هنگامی که عدد شمرده شده توسط کانتر دو با اعداد گذاشته شده برابر شد میکرو وضعیت پایه مربوطه را همانگونه که در پیکربندی مشخص کردید تغییر میدهد. مثال:

\$regfile = "M103DEF.DAT" : \$crystal = 4000000

Config Lcdpin = Pin , Db4 = Pinb.2 , Db5 = Pinb.3 , Db6 = Pinb.4 , Db7 = Pinb.5 , E = Pinb.1 , Rs = Pinb.0

Config Lcd = 16 * 2

Dim A As Word

Config Timer2 = Counter , Edge = Rising , Compare = Toggle , Clear Timer = 1

Enable Oc2

Ocr2 = 100

Config Portb.7 = Output

Do

Locate 1 , 1 : Lcd Counter2

Loop

End

هنگامی که پالس های شمرده شده توسط کانتر 1 به 100 رسید، پایه OC2 تغییر وضعیت میدهد (اگر صفر باشد یک میشود و بلعکس)

استفاده از وقفه مد مقایسه ای کانتر دو :

مد مقایسه ای دارای یک منبع وقفه میباشد که با دستور زیر فعال میشوند:

Enable Interrupts

Enable Oc2

با دستور فعال سازی وقفه مقایسه ، هنگامی که تعداد پالس شمرده شده توسط کانتر با Compare برابر شد cpu میکرو با دستور on oc2 lable به برجسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد. در صورتی که در پایان برجسب دستور return گذاشته شود cpu به حلقه ی اصلی پرش میکند. مثال :

```
$regfile = "M103DEF.DAT" : $crystal = 4000000
Config Lcdpin = Pin , Db4 = Pinb.2 , Db5 = Pinb.3 , Db6 = Pinb.4 , Db7 = Pinb.5 , E = Pinb.1 , Rs = Pinb.0
Config Lcd = 16 * 2
Dim A As Word
Config Timer2 = Counter , Edge = Rising , Compare = Toggle , Clear Timer = 1
Enable Interrupts
Enable Oc2
Ocr2 = 100
On Oc2 B
Config Portb.7 = Output
Config Portb.6 = Output
Do
Locate 1 , 1 : Lcd Counter2
Loop
End
B:
Toggle Portb.6
return
```

در مثال بالا هنگامی که تعداد پالس شمرده شده توسط کانتر با مقدار Compare (100) برابر شد cpu به برجسب bپرش میکند و در آنجا پایه b.6 تغییر وضعیت میدهد و با دستور return به حلقه ی اصلی برمیگردد.

راه اندازی تایمر / کانتر دو در مد PWM :

همانگونه که اشاره شد تایمر کانتر دو دارای یک خروجی PWM 8 بیتی میباشد ، این خروجی پایه OC2 میکرو است . در زیر نحوه راه اندازی تایمر کانتر دو در مد مدولاسیون عرض پالس آورده شده است:

```
Config Timer2 = Pwm ,Pwm =ON|OFF , Compare Pwm=Clear Up|Clear Down|Disconnect ,
```

```
Prescale=1|8|32|64|128|256|1024
```

PWM=ON|OFF : این گزینه برای روشن یا خاموش کردن PWM است ، برای راه اندازی PWM از ON استفاده میکنیم.

Compare A Pwm=Clear Up |Clear Down |Disconnect : در صورت استفاده از گزینه Clear Up ، موج pwm از سطح 1 شروع

میشود و در صورت انتخاب Clear Down ، موج pwm از سطح صفر شروع میشود و در صورت انتخاب Disconnect ،

هنگامی که مقدار pwm با pwm1a که در برنامه مشخص میشود برابر شد ، ارتباط پالس با پایه ی OC2 قطع میشود .(OC2 به

ورودی خروجی عادی تبدیل میشود)

Prescale : این گزینه و مقدار کریستال در تعیین فرکانس pwm نقش دارند . برای تولید PWM با فرکانس های متفاوت از این

گزینه ها استفاده می شود.

با استفاده از دستور زیر میتوان یک عدد ثابت یا متغیر را در رجیستر pwm قرار داد تا مقدار pwm با آنها مقایسه شود:

OCR2=X

مثال:

\$regfile = "M8DEF.DAT" : \$crystal = 8000000

Config Portb.3 = Output

Config Timer2 = Pwm , Pwm = On , Compare Pwm = Clear Up , Prescale = 64

Dim A As Byte

Do

Ocr2 = A

Waitms 500 : Incr A

Loop

End

راه اندازی تایمر/کانتر سه در محیط بسکام:

در برخی از میکرو کنترلر های AVR تایمر-کانتری چهارمی نیز وجود دارد که ویژگی های آن تقریباً مشابه با تایمر-

کانتر یک میباشد ، در زیر مشخصات این تایمر-کانتر را مشاهده میکنید :

✓ این تایمر کانتر 16 بیتی است ، و نهایتاً میتواند تا 2^{16} (65535) بشمارد.

✓ کلاک این تایمر میتواند توسط نوسان ساز داخلی یا نوسان ساز خارجی یا از پایه t3 توسط پالس خارجی، تامین

شود (مقدار فرکانس نوسان ساز بر عدد PRESCALE تقسیم میشود).

✓ تایمر / کانتر یک دارای دو خروجی مقایسه ای است که دو رجیستر OC3A و OC3B مقدار مقایسه ای را در خود

جای می دهند و با محتوای تایمر/کانتر مقایسه می شوند .

✓ در زمان تساوی محتوای رجیستر مقایسه و محتوای تایمر/کانتر ، وضعیت پایه های خروجی مد مقایسه ای OC3A و OC3B می تواند تغییر کند.

✓ تایمر /کانتر در مد CAPTURE نیز می تواند به کار رود . با تحریک پایه ICp3 می توان محتوای تایمر / کانتر را در رجیستر ورودی CAPTURE قرار داد.

✓ این تایمر دارای چندین منبع وقفه میباشد که شما میتوانید آنها را در هر قسمت از برنامه فعال یا غیر فعال کنید. از این منابع وقفه میتوان ،وقفه سرریزی را نام برد.

✓ این تایمر کانتر میتواند در مد تایمر و کانتر و pwm راه اندازی شود.

✓ ورودی کانتر پایه T3 میباشد و خروجی پالس pwm پایه های OC3a و OC3b است همچنین دوپایه ی فوق میتواند به عنوان خروجی مد مقایسه ای تایمر مورد استفاده قرار بگیرند.

✓ این تایمر / کانتر فقط در میکرو های مگا 64 و 128 و 162 موجود است.

راه اندازی تایمر سه در محیط بسکام:

تایمر 3 با دستور زیر پیکربندی میشود:

Config Timer3 = Timer , PRESCALE = 1| 8 | 64 | 256 | 1024

عدد PRESCALE فرکانس (دقت) تایمر را معین میکند .فرکانس و زمانی که تایمر می شمارد از فرمولهای زیر محاسبه میشوند:

$$\text{زمان} = \frac{\text{PRESCALE} * \text{تایمر بیت}}{\text{مقدار کریستال}}$$

$$\text{فرکانس} = \frac{\text{مقدار کریستال}}{\text{PRESCALE}}$$

بعد از دستور بالا، تایمر با دستور START TIMER شروع به شمارش میکند و با دستور STOP TIMER متوقف می شود. تایمر

پس از شمردن تا 65536 (2^{16}) سر ریز میشود ، شما با استفاده از دستور ENABLE OVF3 و ENABLE INTRRUPTS میتوانید

در این مثال میخواهیم توسط تایمر 3 یک شمارنده معکوس بسازیم، برای این کار ابتدا باید یک زمان 1 میلی ثانیه ای ایجاد کنیم طبق فرمول اگر شما از کریستال 4 مگاهرتز و PRESCALE، 64 استفاده نمایید، میتوانید زمانی برابر با 1.04 ثانیه ایجاد کنید، با مقادیر دیگر نیز میتوانید زمان های دقیق تر بسازید. (بیشترین زمانی که با این تایمر میتوانید بسازید برابر با 67.108864 ثانیه میباشد)، شما ابتدا باید یک مقدار را تعیین کنید، میکرو بعد از گذشت 1 ثانیه از مقدار تعیین شده 1 واحد کم میکند:

```
$regfile = "m64def.dat"

$crystal = 4000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0

Config Timer3 = Timer , Prescale = 64

Dim A As Byte : A = 15

Enable Timer3

Start Timer3

Do

If Timer3 > 1 Then : A = A - 1 : End If

Locate 1 , 1 : Lcd A

Loop

End
```

راه اندازی تایمر 3 در مد مقایسه ای (Compare):

در این مد شما میتوانید مقدار شمرده شده توسط تایمر 3 را با دو عدد دلخواه مقایسه کنید و در صورت برابری یا نابرابری مقدار تایمر با مقدار دلخواه وضعیت پایه های oc3a و oc3b (پایه های 5 و 6 میکرو مگا 64) را تغییر دهید. راه اندازی تایمر سه در مد مقایسه ای با دستورات زیر انجام میشود:

```
CONFIG TIMER3= TIMER,COMPARE A = CLEAR |SET|TOGGLE|DISCONNECT,COMPARE B = CLEAR
```

```
|SET|TOGGLE|DISCONNECT, PRESCALE=1|8|64|256|1024,CLEAR TIMER =1|0
```

COMPARE A = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط تایمر 3 با مقدار COMPARE A ،

که بعدا معرفی میشود ، برابر شد ، پایه خروجی OC3A می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط پایه با مد مقایسه ای قطع شود (پایه oc3a به یک ورودی خروجی عادی تبدیل شود).

COMPARE B = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط تایمر 3 با مقدار COMPARE B ،

که بعدا معرفی میشود ، برابر شد ، پایه خروجی OC3B می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط پایه با مد مقایسه ای قطع شود (پایه oc3b به یک ورودی خروجی عادی تبدیل شود).

PRESCALE=1|8|64|256|1024 : عدد PRESCALE فرکانس (دقت) تایمر را معین میکند.

CLEAR TIMER = 1|0 : با انتخاب گزینه 1 ، محتوای تایمر/کانتر در زمان تطابق مقایسه ای RESET (\$0000) می شود و در صورت انتخاب 0 مقدار شمرده شده تغییری نمیکند.

با دستورات زیر میتوان عددی را که محتوای تایمر باید با آن مقایسه شود را تعیین کرد

```
Compare3a = x
```

```
Compare3b =x
```

به جای x یک عدد ثابت یا یک متغیر قرار میگیرد ، هنگامی که عدد شمرده شده توسط تایمر یک با اعداد گذاشته شده برابر شد میکرو وضعیت پایه های مربوطه را همانگونه که در پیکربندی مشخص کردید تغییر میدهد . مثال:

```
$regfile = "m64def.dat"
```

```
$crystal = 4000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0
```

```
Config Timer3 = Timer , Prescale = 64
```



```
Compare3a = 1000
```

```
Compare3b = 5000
```

```
Do
```

```
Locate 1 , 1
```

```
Lcd Timer3
```

```
Loop
```

```
End
```

هنگامی که تایمر 3 تا 1000 شمرد پایه oc3a یک میشود و هنگامی که مقدار شمرده شده به 5000 رسید پایه oc3b یک میشود (oc3a پایه 5 و oc3b پایه 6 مگا 64 میباشد).

استفاده از وقفه مد مقایسه ای تایمر 3:

مد مقایسه ای دارای 2 منبع وقفه میباشد که با دستورات زیر فعال میشوند:

```
Enable Interrupts
```

```
Enable Oc3a
```

```
Enable Oc3b
```

با دستور فعال سازی وقفه مقایسه ، هنگامی که مقدار شمرده شده توسط میکرو با Compare3a برابر شد cpu میکرو با دستور on oc3a lable به برچسب مورد نظر پرش میکند و در آنجا عملیات دلخواه را انجام میدهد، هنگامی که مقدار شمرده شده توسط میکرو با Compare3b برابر شد cpu میکرو با دستور on oc3b lable به برچسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد. در صورتی که در پایان برچسب دستور return گذاشته شود cpu به حلقه ی اصلی پرش میکند . مثال :

```
$regfile = "m64def.dat"
```

```
$crystal = 4000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0
```

```
Config Timer3 = Timer , Prescale = 64
```

```
Compare3a = 2120
```

```
Compare3b = 63000
```

```
Enable Interrupts
```

```
Enable Oc3a
```

Enable Oc3b

On Oc3a Q

On Oc3b W

Do

Locate 1 , 1 : Lcd Timer3

Loop

End

Q:

Locate 2 , 1 : Lcd Timer3

Return

W:

Locate 2 , 8 : Lcd Timer3

Return

در مثال بالا هنگامی که رقم شمرده شده توسط تایمر با مقدار Compare3a (2120) برابر شد cpu به برچسب q پرش میکند و در آنجا مقدار شمرده شده توسط تایمر را بر روی lcd نمایش داده و با دستور return به حلقه ی اصلی برمیگردد ، هنگامی که رقم شمرده شده توسط تایمر با مقدار Compare3b (63000) برابر شد cpu به برچسب w پرش میکند و در آنجا مقدار شمرده شده توسط تایمر را بر روی lcd نمایش داده و با دستور return به حلقه ی اصلی برمیگردد. همچنین در مورد اول پایه oc3a یک میشود و در مورد دوم وضعیت پایه oc3b تغییر میکند (در صورت 0 بودن یک میشود و بلعکس).

راه اندازی تایمر 3 در مد CAPTURE :

در صورتی که تایمر 3 را در این مد پیکربندی کنید ، با اعمال یک پالس بالا رونده یا پایین رونده (که نوع آن در هنگام پیکربندی مشخص میشود) به پایه ic3 (پایه 9 مگا 64) ، در همان لحظه مقدار شمرده شده توسط تایمر 3 در رجیستر CAPTURE قرار میگیرد ، محتوای رجیستر CAPTURE را می توان با دستور VAR = CAPTURE در یک متغیر از جنس word قرار داد. راه اندازی تایمر 3 در مد CAPTURE با دستورات زیر انجام میشود:

Config Timer3 = Timer, Capture Edge= Falling | Rising ,Noise Cancel=1|0 , Prescale =1|8|64|256|1024

Capture Edge= Falling | Rising : این گزینه مشخص میکند ، پالس اعمالی به پایه icp بالا رونده (Falling) است یا پایین

رونده (Rising). (CAPTURE به کدام پالس حساسیت نشان میدهد).

Noise Cancel=1|0: در صورت انتخاب یک از نویز های موجود بر روی پایه icp چشم پوشی میشود و هر پالس با لبه ی تعیین شده میتواند CAPTURE را راه اندازی کند، در صورت انتخاب صفر فقط پالس های با دامنه ی 5 ولت قادر به راه اندازی CAPTURE خواهند بود.

Prescale = 1|8|64|256|1024: دقت تایمر را نشان میدهد (با استفاده از این مقدار و مقدار کریستال میتوانید زمان شمرده شده توسط تایمر تا هنگام سرریزی را با فرمولی که در بالا گفته شد محاسبه کنید)

با دستور Enable Icp3 این مد فعال میشود و پایه icp آماده دریافت پالس میگردد. مثال:

```
$regfile = "m64def.dat"
$crystal = 4000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0
Config Timer3 = Timer , Capture Edge = Falling , Noise Cancel = 1 , Prescale = 1024
Do
Locate 1 , 1
Lcd Timer3
Locate 2 , 1
Lcd Capture3
Loop
End
```

در مثال بالا هنگامی که یک پالس بالا رونده به پایه icp اعمال میشود مقدار شمرده شده توسط تایمر3 درون رجیستر Capture قرار میگیرد و سپس بر روی lcd نمایش داده میشود.

استفاده از وقفه مد Capture تایمر 3:

مد Capture تایمر 3 نیز مانند دیگر مد ها دارای یک منبع وقفه است که شما میتوانید با دستور Enable Icp3 آن را فعال کنید تا با دستور On icp3 lable، هنگامی که پالسی به پایه icp اعمال شد ، cpu به برجسب مورد نظر پرش کند و در آنجا عملیات دلخواه را انجام دهد .

راه اندازی کانتر 3 در محیط بسکام :

کانتر 3 در بسکام با دستور زیر پیکربندی میشود:

CONFIG TIMER3 = COUNTER , EDGE = RISING / FALLING

✓ با انتخاب EDGE = RISING کانتر نسبت به لبه ی بالا رونده حساس است .

✓ با انتخاب EDGE = FALLING کانتر نسبت به لبه ی پایین رونده حساس است .

(لبه های بالا رونده یا پایین رونده را میشمارد (بالا رونده < صفر به یک و پایین رونده > یک به صفر))

شما همچنین میتوانید با استفاده از دستور VAR = COUNTER3 ، مقدار کانتر را در یک متغیر از جنس word قرار دهید و یا با

دستور COUNTER3 = VALUE ، مقدار اولیه ای را در کانتر قرار دهید تا کانتر از آن شروع به شمارش کند. کانتر نیز مانند تایمر

پس از شمردن تا 65536 سر ریز میشود ، شما با استفاده از دستور ENABLE OV3F3 میتوانید وقفه سر ریزی کانتر را راه اندازی

کنید . در صورتی که وقفه سر ریزی کانتر فعال باشد ، کانتر پس از سر ریزی به برچسبی که با یکی از دستورات ON OV3F3

LABLE مشخص شده پرش میکند ، باز گشت از وقفه با دستور RETURN انجام میشود. در این مثال کانتر سه تعداد پالسهای

اعمالی به پایه T3 (پایه شماره 8 مگا 64) را میشمارد. مثال :

```
"$regfile = "m64def.dat"
```

```
$crystal = 4000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0
```

```
Config Timer3 = Counter , Edge = Falling
```

```
Do
```

```
Locate 1 , 1 : Lcd Counter3
```

```
Loop
```

```
End
```

کانتر سه را نیز میتوان مانند تایمر 3 در دو مد مقایسه ای و Capture راه اندازی کرد که طریقه راه اندازی در زیر آورده شده

است:

راه اندازی کانتر 3 در مد مقایسه ای (Compare):

در این مد شما میتوانید مقدار کانتر 3 را با دو عدد دلخواه مقایسه کنید و در صورت برابری یا نابرابری مقدار شمرده شده توسط کانتر با مقدار دلخواه وضعیت پایه های oc3a و oc3b را تغییر دهید. راه اندازی کانتر 3 در مد مقایسه ای با دستورات زیر انجام میشود:

CONFIG TIMER3 = COUNTER , EDGE = RISING / FALLING, COMPARE A = CLEAR |SET|TOGGLE|

DISCONNECT,COMPARE B = CLEAR |SET|TOGGLE|DISCONNECT,CLEAR TIMER = 1|0

EDGE = با انتخاب EDGE = RISING / FALLING کانتر نصب به لبه ی بالا رونده حساس است و با انتخاب

FALLING کانتر نصب به لبه ی پایین رونده حساس است.

COMPARE A = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط کانتر 3 با مقدار COMPARE A ،

که بعدا معرفی میشود ، برابر شد ،پایه خروجی OC3A می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط

پایه با مد مقایسه ای قطع شود (پایه oc3a به یک ورودی خروجی عادی تبدیل شود).

COMPARE B = CLEAR |SET|TOGGLE|DISCONNECT : زمانی که مقدار شمرده شده توسط کانتر 3 با مقدار COMPARE B ،

که بعدا معرفی میشود ، برابر شد ،پایه خروجی OC3B می تواند SET (یک)، (صفر)، CLEAR (برعکس) TOGGLE و یا ارتباط

پایه با مد مقایسه ای قطع شود (پایه oc3b به یک ورودی خروجی عادی تبدیل شود).

CLEAR TIMER = 1|0 : با انتخاب گزینه 1 ،محتوای کانتر 3 در زمان تطابق مقایسه ای RESET (0000) می شود و در صورت

انتخاب 0 مقدار شمرده شده تغییری نمیکند.

با دستورات زیر میتوان عددی که محتوای تایمر باید با آن مقایسه شود را تعیین کرد

Compare3a = x

Compare3b =x

به جای x یک عدد ثابت یا یک متغیر قرار میگیرد ، هنگامی که عدد شمرده شده توسط کانتر 3 با اعداد گذاشته شده برابر

شد میکرو وضعیت پایه های مربوطه را همانگونه که در پیکربندی مشخص کردید تغییر میدهد. مثال :

\$regfile = "m64def.dat"

\$crystal = 4000000

Config Lcd = 16 * 2

```
Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0
```

```
Config Timer3 = Counter , Edge = Falling , Compare A = Set , Compare B = Set , Clear Timer = 0
```

```
Compare3a = 300
```

```
Compare3b = 400
```

```
Do
```

```
Locate 1 , 1
```

```
Lcd Counter3
```

```
Loop
```

```
End
```

هنگامی که پالس های شمرده شده توسط کانتر 3 به 300 رسید ، پایه oc3a یک میشود و هنگامی که مقدار شمرده شده به 400 رسید پایه oc3b یک میشود (oc1a پایه 15 و oc1b پایه 16 مگا 64 میباشد).

استفاده از وقفه مد مقایسه ای کانتر 3 :

مد مقایسه ای دارای 2 منبع وقفه میباشد که با دستورات زیر فعال میشوند:

```
Enable Interrupts
```

```
Enable Oc3a
```

```
Enable Oc3b
```

با دستور فعال سازی وقفه مقایسه ، هنگامی که تعداد پالس شمرده شده توسط کانتر با Compare3a برابر شد cpu میکرو با دستور on oc3a lable به برچسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد، هنگامی که تعداد پالس شمرده شده توسط کانتر با Compare3b برابر شد cpu میکرو با دستور on oc3b lable به برچسب مورد نظر برش میکند و در آنجا عملیات دلخواه را انجام میدهد. در صورتی که در پایان برچسب دستور return گذاشته شود cpu به حلقه ی اصلی پرش میکند . مثال :

```
: $regfile = "m64def.dat" : $crystal = 4000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0
```

```
Config Timer3 = Counter , Edge = Falling , Compare A = Set , Compare B = Set , Clear Timer = 0
```

```
Compare3a = 30
```

```
Compare3b = 40
```

```
Enable Interrupts
```

Enable Oc3a

Enable Oc3b

On Oc3a Q

On Oc3b W

Do

Locate 1 , 1 : Lcd Counter3

Loop

End

Q:

Cls : Locate 2 , 1 : Lcd " Counter3=30"

Return

W:

Cls : Locate 2 , 1 : Lcd " Counter3=40"

Return

در مثال بالا هنگامی که تعداد پالس شمرده شده توسط کانتر با مقدار Compare3a (30) برابر شد cpu به برچسب q پرش میکند و در آنجا عبارت "Counter3=30" را بر روی lcd نمایش داده و با دستور return به حلقه ی اصلی برمیگردد ، هنگامی که تعداد پالس شمرده شده توسط کانتر با مقدار Compare3b (40) برابر شد cpu به برچسب w پرش میکند و در آنجا عبارت "Counter3=40" را بر روی lcd نمایش داده و با دستور return به حلقه ی اصلی برمیگردد .همچنین در مورد اول پایه oc3a یک میشود و در مورد دوم وضعیت پایه oc3b یک میشود .

راه اندازی کانتر 3 در مد CAPTURE :

در صورتی که کانتر 3 را در این مد پیکربندی کنید ، با اعمال یک پالس بالا رونده یا پایین رونده (که نوع آن در هنگام پیکربندی مشخص میشود) به پایه ic3 (پایه 9 مگا 64) ، در همان لحظه تعداد پالس شمرده شده توسط کانتر 3 در رجیستر CAPTURE قرار میگیرد ، محتوای رجیستر CAPTURE را می توان با دستور VAR = CAPTURE در یک متغیر از جنس word قرار داد .راه اندازی کانتر یک در مد CAPTURE با دستورات زیر انجام میشود:

Config Timer3 = Counter , Edge= Falling|Rising , Capture Edge=_ Falling | Rising , Noise Cancel= 1|0 ,

EDGE = Edge= Falling|Rising : با انتخاب EDGE = RISING کانتر نصبت به لبه ی بالا رونده حساس است و با انتخاب

FALLING کانتر نصبت به ی پایین رونده حساس است.

Capture Edge= Falling | Rising : این گزینه مشخص میکند ، پالس اعمالی به پایه ic3 بالا رونده (Falling) است یا پایین رونده (Rising). (CAPTURE به کدام پالس حساسیت نشان میدهد)

Noise Cancel=1|0 : در صورت انتخاب یک از نویز های موجود بر روی پایه ic3 چشم پوشی میشود و هر پالس با لبه ی تعیین شده میتواند CAPTURE را راه اندازی کند ، در صورت انتخاب صفر فقط پالس های با دامنه ی 5 ولت قادر به راه اندازی CAPTURE خواهند بود. مثال :

```
$regfile = "m64def.dat" : $crystal = 4000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0
```

```
Config Timer3 = Counter , Edge = Falling , Capture Edge = Rising , Noise Cancel = 0
```

```
Do
```

```
Locate 1 , 1
```

```
Lcd Counter3
```

```
Locate 2 , 1
```

```
Lcd Capture3
```

```
Loop
```

```
End
```

در مثال بالا هنگامی که یک پالس بالا رونده به پایه ic3 اعمال میشود مقدار شمرده شده توسط تایمر 3 درون رجیستر Capture قرار میگیرد و سپس بر روی lcd نمایش داده میشود.

استفاده از وقفه مد Capture کانتر 3 :

مد Capture کانتر 3 نیز مانند دیگر مد ها دارای یک منبع وقفه است که شما میتوانید با دستور Enable lcp3 آن را فعال کنید تا با دستور On icp3 lable ، هنگامی که پالسی به پایه icp اعمال شد ، cpu به برجسب مورد نظر پرش کند و در آنجا عملیات دلخواه را انجام دهد . مثال :

```
$regfile = "m64def.dat" : $crystal = 4000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0
```

```
Config Timer3 = Counter , Edge = Falling , Capture Edge = Rising , Noise Cancel = 0
```

```
Enable Interrupts
```

```
Enable lcp3
```


On Icp3 Q

Do

Locate 1 , 1

Lcd Counter3

Loop

End

Q:

Locate 2 , 1

Lcd Capture3

Return

در مثال بالا ، هنگامی که پالسی به پایه icp اعمال شود ، تعداد پالس شمرده شده توسط کانتر 3 در ریجیستر Capture3 ریخته

میشود و cpu به برچسب q برش میکند و در آنجا مقدار Capture3 را بر روی lcd نمایش میدهد.

پیکر بندی تایمر/کانتر 3 در مد PWM

راه اندازی تایمر/کانتر 3 در مد pwm با دستورات زیر انجام میشود :

Config Timer3 = 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

Pwm = 8|9|10 :pwm میتواند 8 یا 9 یا 10 بیتی باشد که مقدار بیت هرچه بیشتر باشد دقت موج بیشتر است (تعداد پله بیشتر است

(

pwm 8 بیتی تا 256 سرریز میشود (شما میتوانید 256 واحد آن را کم یا زیاد کنید) pwm 9 بیتی تا 512 و pwm 10 بیتی تا

1024 سرریز میشود.

Compare A Pwm=Clear Up |Clear Down |Disconnect : در صورت استفاده از گزینه Clear Up ، موج pwm از سطح 1 شروع

میشود و در صورت انتخاب Clear Down ، موج pwm از سطح صفر شروع میشود و در صورت انتخاب Disconnect ،

هنگامی که مقدار pwm با pwm1a که در برنامه مشخص میشود برابر شد ، ارتباط پالس با پایه ی oc1a قطع میشود .

Compare b Pwm=Clear Up |Clear Down |Disconnect : در صورت استفاده از گزینه Clear Up ، موج pwm از سطح 1 شروع

میشود و در صورت انتخاب Clear Down ، موج pwm از سطح صفر شروع میشود و در صورت انتخاب Disconnect ،

هنگامی که مقدار pwm با pwm1b که در برنامه مشخص میشود برابر شد ، ارتباط پالس با پایه ی oc1b قطع میشود .

Prescale : این گزینه و مقدار کریستال در تعیین فرکانس pwm نقش دارند . برای تولید PWM با فرکانس های متفاوت از این گزینه ها استفاده می شود.

با استفاده از دو دستور زیر میتوان یک عدد ثابت یا متغیر را در رجیستر pwm قرار داد تا مقدار pwm با آنها مقایسه شود:

Pwm3a=x

Pwm3b=x

یا

COMPARE3A = x

COMPARE3B = x

مثال:

\$regfile = "m64def.dat" : \$crystal = 4000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portf.2 , Db5 = Portf.3 , Db6 = Portf.4 , Db7 = Portf.5 , E = Portf.1 , Rs = Portf.0

Config Timer3 = Pwm , Pwm = 8 , Compare A Pwm = Clear Up , Compare B Pwm = Clear Down , Prescale = 1

Dim A As Word

Dim B As Word

Config Portb = Output

Do

Pwm3a = A

Pwm3b = B

Incr A

Incr B

Waitms 500

Loop

End

در مثال بالا مقدار دو متغیر a و b در رجیستر pwm قرار داده شده اند ، مقدار آنها هر 500 میلی ثانیه افزایش میابد . زمان تناوب پالس pwm از رابطه ی زیر بدست میاید:

$$\text{زمان} = \frac{PRESCALE * 2 * \text{تایمر بیت}}{\text{مقدار کریستال}}$$

زمان / 1 = فرکانس

چند مثال عملی در رابطه با pwm :

pwm یا Pulse Width Modulator یا مدولاسیون پهنای پالس نوعی از مدولاسیون (ایجاد ولتاژ) است که در آن دامنه ی موج

ثابت و نسبت صفر به یک یا Duty cycle یا چرخه ی کار متغیر هست .

در قسمت های قبلی با نحوه راه اندازی تایمر 1 و 2 و 3 در مد pwm آشنا شدید ، در زیر اندکی بیشتر در این مورد میخوانیم:

فرکانس pwm از روابط زیر محاسبه میشود :

➤ در صورتی که بیت 8 pwm باشد :

(پرسکال * 510) / فرکانس کریستال = فرکانس pwm

➤ در صورتی که بیت 9 pwm باشد :

(پرسکال * 1022) / فرکانس کریستال = فرکانس pwm

➤ در صورتی که بیت 10 pwm باشد :

(پرسکال * 2046) / فرکانس کریستال = فرکانس pwm

Duty cycle یا چرخه ی کار چیست ؟

فرض کنید شما از pwm ده بیتی استفاده کرده اید ، در این صورت pwm شما میتواند 1023 پله داشته باشد ، با چند مثال ادامه

می دهیم :

یک موج مربعی با زمان تناوب 20 میلی ثانیه و Duty cycle نود درصد تولید کنید؟

جواب :

اولین مرحله ساخت یک موج pwm با فرکانس 50 هرتز هست، ما میتوانیم از بین فرکانس ها و بیت ها موجود یک مورد را

به دلخواه انتخاب کنیم ، من کریستال 6528000 هرتز و pwm هشت بیتی و مقدار Prescale = 256 انتخاب کرده بودم :

\$regfile = "M8DEF.dat"

\$crystal = 6528000

Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Up , Prescale = 256

Do

Pwm1a = 10

Loop

End

در برنامه بالا ، ابتدا پایه oc1a در سطح صفر می باشد . هنگامی که مقدار شمرده شده توسط تایمر به رقم 10 رسید ، وضعیت پایه معکوس میشود ، از آنجا که pwm هشت بیتی است ، موج 256 (دو بتوان 8) پله دارد . تایمر تا 255 می شمرد (از صفر تا 255 برابر 256 پله است) و صفر می شود ، با صفر شدن تایمر دو باره وضعیت پایه oc1a معکوس میشود ، دو باره تایمر تا 10 می شمارد ، دو باره وضعیت پایه معکوس میشود و این روند مدام تکرار میشود .

با این حساب ، از 255 پله ، تعداد 10 عدد در سطح صفر و تعداد 245 عدد در سطح یک می باشد . در صورتی که 255 را صد درصد بگیریم ، میتوان بگوییم که 96.07 درصد موج را سطح یک آن تشکیل میدهد $((245 * 100) / 255)$ و در واقع ضریب مدولاسیون یا چرخه ی کار یا Duty cycle برابر با 96.07 درصد است . با تغییر دادن رقم 10 به 25 میتوان به ضریب مدولاسیون 90٪ دست پیدا کرد .

مثال 2 :

توسط تایمر یک ، موجی با فرکانس 2 کیلو هرتز و با مشخات زیر ایجاد کنید . 60 میکرو ثانیه از این موج را سطح صفر و الباقی را سطح یک در برگیرد :

جواب :

همانند مثال قبل ابتدا مقدار کریستال و بیت تایمر و پرسیکال را محاسبه میکنیم . (من معمولاً مقدار پرسیکال و بیت تایمر را ثابت در نظر میگیرم و مقدار فرکانس را برای آن محاسبه میکنم)

من از pwm ده بیتی و پرسیکال 1 استفاده میکنم و مقدار فرکانس کریستال را بدست میآورم :

Prescale x بیت pwm x فرکانس pwm = مقدار کریستال

$$2000 \times 2046 \times 1 = 4092000$$

در صورتی که از کریستال 4092000 هرتز استفاده کنیم ، مقدار فرکانس pwm برابر با 2 کیلو هرتز خواهد شد .

نوشتن برنامه :

\$regfile = "M8DEF.dat"

\$crystal = 4092000

راه اندازی تایمر در مد pwm :

Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Up , Prescale = 1

می‌توانستیم pwm را در حالت زیر نیز راه اندازی کنیم ، در این حالت پایه oc1b خروجی بود (همچنین می‌شد از Clear down به جای Clear up استفاده کرد)

\$regfile = "M8DEF.dat"

\$crystal = 4092000

Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear down , Prescale = 1

Do

Pwm1a = 10

Loop

End

شما حتما باید یک مقدار در ریجستر مقایسه ای pwm (در هنگام استفاده از oc1a ب Pwm1a = 10 و در موقع استفاده از oc1b

Pwm1b = 10) قرار دهید تا موج pwm ایجاد شود .

تا کنون موجی با فرکانس 2 کیلو ساختیم ، در زیر می‌خواهیم شرط را اجرا کنیم:

مقدار زمان تناوب کل ، 500 میکرو ثانیه است ، موج باید 60 میکرو ثانیه در سطح صفر و 440 میکرو ثانیه در سطح یک باشد

، در این صورت Duty cycle برابر با 88 درصد است ، چون pwm ده بیتی است ، یعنی 1024 پله دارد (دو بتوان 10) از این

1024 پله باید 88٪ در سطح یک منطقی باشد و الباقی در سطح صفر ، با یک حساب ساده میشود فهمید که 901 پله در سطح

یک هستند ، بنابراین ماباید به جای 10 عدد 901 را بگذاریم .

در این مثال اگر به جای Clear Down از Clear up استفاده می‌کردیم ، باید به جای 901 رقم 1023-901 یا 121 را قرار میدادیم .

اگر می‌خواهید موارد بالا را در عمل ببینید حتما از کریستال خارجی استفاده نمایید ، مقادیر فرکانس را می‌توانید با استفاده از

نوسان ساز rc بسازید .

در بخش کنترل دور موتور های DC در مورد PWM توضیحات بیشتری داده شده است .

تایمر- کانترها در سری ATXMEGA :

میکرو کنترلر های سری ATXMEGA دارای یک تا 8 تایمر / کانتر میباشند که کاربر میتواند آنها را در مد های تایمر ، کانتر یا PWM راه اندازی و استفاده کند . این واحدها که با نام های TCC0, TCC1, TCD0, TCD1, TCE0, TCE1, TCF0 و TCF1 نام گذاری می شوند با مجموعه دستورات زیر پیکربندی می شوند :

CONFIG TCxx = mode , PRESCALE=pre, COMPAREx= setup , WGMODE=wg, EVENT_SOURCE= event, EVENT_ACTION=act, EVENT_DELAY=ed, RESOLUTION=res

نام TCxx : CONFIG TCxx = mode تایمر-کانتری که قصد پیکربندی آن را داریم میباشد ، همچنین به جای mode یکی از دستورات NORMAL ، PWM یا COUNTER به ترتیب برای راه اندازی واحد تایمر-کانتر xx در مد تایمر ، مدولاسیون پهنای پالس یا کانتر قرار میگیرد .

PRESCALE=pre : در این بخش به جای pre یکی از دستورات زیر قرار داده میشود :

➤ مقادیر 1, 2, 4, 8, 64, 256, 1024 که فرکانس کاری واحد تایمر - کانتر از تقسیم شدن کلاک اصلی سیستم به آنها تعیین میشود .

➤ OFF : در این حالت واحد تایمر-کانتر خاموش می شود .

➤ بخش های E0, E1, E2, E3, E4, E5, E6, E7 در واحد رویداد (Event channel) که در ادامه به بررسی آنها خواهیم پرداخت .

➤ مقادیر 0 تا 15 که در رجیستر CTRLA نوشته میشود ، برای کسب اطلاعات بیشتر در این مورد میتوانید به دیتا شیت میکرو و بخش منابع کلاک مراجعه کنید . (در انتهای کتاب دیتاشیت فارسی برخی از میکرو کنترلر های AVR آورده شده است) .

COMPAREx= setup : تایمر - کانتر های سری Xmega دارای 4 پین (پایه های ورودی و خروجی که با نام OCXX معرفی میشوند) برای مد های COMPARE و CAPTURE میباشند که توسط این دستور میتوانید این پایه ها را در حالت دلخواه پیکربندی کنید . در این دستور x میتواند حرف A یا B یا C یا D برای انجام پیکربندی پایه های COMPARE یا CAPTURE تایمر - کانتر پیکربندی شده باشد . همچنین به جای setup یکی از دستورات زیر استفاده میشود :

ENABLED : با این دستور رجیستر های capture/compare فعال میشود .

DISABLED : با این دستور رجیستر های capture/compare غیر فعال میشود

0 : در این حالت پایه های خروجی مد compare در سطح منطقی 0 قرار میگیرد .

1 : در این حالت پایه های خروجی مد compare در سطح منطقی 1 قرار میگیرد .

RTC (Real Time Counter) (شمارش گر زمان واقعی):

RTC یا شمارش گر زمان واقعی ، واحدی در میکروکنترلر ها میباشد که میتواند با عملکردی کاملا مستقل از سایر بخش ها ، زمان را اندازه گیری کند ، در بعضی از میکروکنترلر های خانواده ی AVR تایمر / کانتر دو یا صفر میتواند به صورت آسنکرون کار کند و زمان و تاریخ را بشمارد ، این شمارش در همه حالتها حتی زمانی که میکرو در حالت power-save است نیز وجود دارد . (در حالت آسنکرون کلاک تایمر توسط کریستال ساعت Hz32768 از دو پایه toc1,toc2 تامین میشود) .

Rtc داخلی میکرو :

. Rtc با دستور زیر پیکربندی میشود: (این قابلیت در بعضی از میکروها وجود دارد ، که شما میتوانید با مراجعه به دیتا شیت میکرو از وجود یا عدم وجود آن مطلع شوید)

CONFIG CLOCK = soft | USER [, GOSUB = SECTIC] [,RTC=rtc]

Soft: هنگامی استفاده میشود که میخواهید از rtc داخلی میکرو استفاده کنید.

USER: هنگامی استفاده میشود که بخواهید از rtc خارجی (مانند Ds1307 یا...این ایسی از پروتکل I2C استفاده میکند ، کار با این ایسی در بخش "روشهای ارتباطی در "avr گفته شده است)) استفاده کنید.

GOSUB = SECTIC: این گزینه اختیاری است ، زمانی که تایمر سرریز شد (تا یک ثانیه شمرد) به برجسب SECTIC پرش میشود ، بازگشت از برجسب با دستور return صورت میگیرد .

RTC=rtc: این گزینه فقط در میکروکنترلر های خانواده ی ATXMEGA کاربرد دارد و توسط آن کاربر میتواند منبع کلاک واحد RTC را مشخص نماید ، کاربر باید به جای واژه ی rtc یکی از دستورات زیر استفاده کند :

1KHZ_INT32KHZ_ULP ' 1 kHz from internal 32 kHz ULP

1KHZ_32KHZ_CRYSTOSC ' 1 kHz from 32 kHz Crystal Oscillator on TOSC

1KHZ_INT32KHZ_RCOSC ' 1 kHz from internal 32 kHz RC Oscillator

32KHZ_32KHZ_CRYSTOSC ' 32 kHz from 32 kHz Crystal Oscillator on TOSC

با دستورات زیر مقدار اولیه زمان و تاریخ مشخص می شوند(روش اول):

```
DATE$ = "mm/dd/yy"
```

```
TIME$ = "hh:mm:ss"
```

به جای حروف اعداد اولیه قرار داده میشوند .

مثال :

```
$regfile = "M16DEF.DAT"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , E = Pind.1 , Rs = Pind.0
Config Lcd = 16 * 2
Enable Interrupts
Config Clock = Soft
Date$ = "11/11/00"
Time$ = "02:20:00"
Do
  Locate 1 , 1
  Lcd Date$
  Locate 2 , 1
  Lcd Time$
Loop
End
```

مثال برای سری ATXMEGA :

```
$regfile = "xm128a1def.dat"
$crystal = 32000000
'include the following lib and code, the routines will be replaced since they are a workaround
$lib "xmega.lib"
$external _xmegafix_clear
$external _xmegafix_rol_r1014
Config Portb = Output
'First Enable The Osc Of Your Choice , make sure to enable 32 KHz clock or use an external 32 KHz clock
Config Osc = Enabled , 32mhzosc = Enabled , 32khzosc = Enabled
```

' For the CLOCK we use the RTC so make sure the 32 KHZ osc is enabled!!!

'configure the systemclock

Config Sysclock = 32mhz , Prescalea = 1 , Prescalebc = 1_1

Config Com1 = 19200 , **Mode** = Asynchronous , Parity = None , Stopbits = 1 , Databits = 8

Open "COM1:" For Binary As #1

Config Clock = Soft , Rtc = 1khz_int32khz_ulp ' we select the internal 1 KHz clock from the 32KHz internal oscillator

'the following clocks can be used to clock the RTC

' 1KHZ_INT32KHZ_ULP 1 kHz from internal 32 kHz ULP

' 1KHZ_32KHZ_CRYSTOSC 1 kHz from 32 kHz Crystal Oscillator on TOSC

' 1KHZ_INT32KHZ_RCOSC 1 kHz from internal 32 kHz RC Oscillator

' 32KHZ_32KHZ_CRYSTOSC 32 kHz from 32 kHz Crystal Oscillator on TOSC

Config Priority = Static , Vector = Application , Lo = Enabled ' the RTC uses LO priority interrupts so these must be enabled !!!

Enable Interrupts ' as usual interrupts must be enabled

Do

Print Time\$ ' print the time

Waitms 1000

Loop

Sectic:

Toggle Portb 'optional toggle some leds when using the gosub=sectic option

Return

شما میتوانید با دستورات زیر مقدار اولیه ای را برای زمان و تاریخ مشخص کنید:

__sec = X ' بین 0 تا 59 است

__min = X ' بین 0 تا 59 است

__hour = X ' بین 0 تا 23 است

__day = X ' بین 1 تا 31 است

__month = X ' بین 1 تا 12 است

__year = x ' بین 0 تا 99 است

شما میتوانید کلیه اعمال جمع و تفریق (کاستن و افزودن) را مستقیم روی متغیر های بالا انجام دهید. مثال:

```
$regfile = "M16DEF.DAT"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , E = Pind.1 , Rs = Pind.0
Config Lcd = 16 * 2
Config Portb = Input
Enable Interrupts
Config Date = YMD , Separator = .
Config Clock = Soft , Gosub = Sectic
_sec = 57 : _min = 59 : _hour = 23 : _day = 32 : _month = 11 : _year = 99
Goto W
Set_ok:
Locate 1 , 10 : Lcd "set ok "
Wait 2 : Locate 1 , 10 : Lcd "      "
W:
Do
Debounce Pinb.5 , 0 , Incr_sec
Loop
Incr_sec:
Do
Locate 1 , 10 : Lcd "set sec"
Debounce Pinb.5 , 0 , Incr_min : Waitms 100
If Pinb.6 = 0 Then : _sec = 0 : Waitms 400 : End If
If Pinb.7 = 0 Then : _sec = 30 : Waitms 400 : End If
Loop
Incr_min:
Do
Locate 1 , 10 : Lcd "set min" : Waitms 100
Debounce Pinb.5 , 0 , Incr_hour
If Pinb.6 = 0 Then : Incr_min : Waitms 400 : End If
If Pinb.7 = 0 Then : Decr_min : Waitms 400 : End If
Loop
Incr_hour:
Do
Locate 1 , 10 : Lcd "set hou"
Debounce Pinb.5 , 0 , Incr_day:Waitms 100
If Pinb.6 = 0 Then : Incr_hour : Waitms 400 : End If
If Pinb.7 = 0 Then : Decr_hour: Waitms 400 : End If
Loop
Incr_day:
Do
```

```

Locate 1 , 10 : Lcd "set day"
Debounce Pinb.5 , 0 , Incr_month:Waitms 100
If Pinb.6 = 0 Then : Incr _day : Waitms 400 : End If
If Pinb.7 = 0 Then : Decr _day : Waitms 400 : End If
Loop
Incr_month:
Do
Locate 1 , 10 : Lcd "set mon"
Debounce Pinb.5 , 0 , Incr_year :Waitms 100
If Pinb.6 = 0 Then : Incr _month : Waitms 400 : End If
If Pinb.7 = 0 Then : Decr _month : Waitms 400 : End If
Loop
Incr_year:
Do
Locate 1 , 10 : Lcd "set yea"
Debounce Pinb.5 , 0 , Set_ok : Waitms 100
If Pinb.6 = 0 Then : Incr _year : Waitms 400 : End If
If Pinb.7 = 0 Then : Decr _year : Waitms 400 : End If
Loop
End
Sectic:
Locate 1 , 1 : Lcd Date$ : Locate 2 , 1 : Lcd Time$
Return

```

مثال:

```

$regfile = "M16DEF.DAT"
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , E = Pind.1 , Rs = Pind.0
Config Lcd = 16 * 2
Enable Interrupts
Cursor Off
Config Clock = Soft , Gosub = Sectic
Config Adc = Single , Prescaler = Auto
Dim Q(2) As Word
Date$ = "11/11/00"
Time$ = "02:20:00"
Start Adc
Do
Q(1) = Getadc(0)
Q(2) = Getadc(1)
Q(1) = Q(1) / 2

```

Q(2) = Q(2) / 2

Locate 1 , 11

Lcd "t1:" ; Q(1)

Locate 2 , 11

Lcd "t2:" ; Q(2)

Loop

End

Sectic:

Locate 1 , 1

Lcd Date\$

Locate 2 , 1

Lcd Time\$

Return

برنامه بالا علاوه بر نمایش ساعت و تقویم دمای دو نقطه ی دلخواه را نیز نمایش میدهد ، همانگونه که مشاهده میفرمایید برنامه اسکن کانال های adc در حلقه ی do-loop قرار داده شده ، هنگامی که تایمر سرریز میشود به برجسب Sectic پرش میشود و زمان و تاریخ بر روی lcd نمایش داده شده و دوباره با دستور return به حلقه ی do-loop رجوع می شود و این کار مدام تکرار میگردد. شما میتوانید در حلقه از هر دستوری استفاده کنید .

شما همچنین میتوانید با دستور زیر فرمت نمایش تاریخ را معین کنید (در هر کشوری تاریخ به شکل خاص نشان داده می شود ، مثلا در ایران به صورت : yy/mm/dd و در ایالات متحده امریکا به صورت: mm-dd-yy است)

CONFIG DATE = DMY/MDY/YMD , Separator = char

DMY/MDY/YMD: نشان دهنده مکان نمایش روز ، ماه و سال است ، d نمایشگر روز (Day) ، m نمایشگر ماه (month) و y نمایشگر سال (year) است .

Char: علامت بین روز و ماه و سال میباشد که میتواند نقطه "." ، ممیز "/" یا خط فاصله "-" باشد.

جدول زیر اطلاعات بیشتری را در اختیار شما میگذارد:

Country	Format	Statement
American	mm/dd/yy	Config Date = MDY, Separator = /
ANSI	yy.mm.dd	Config Date = YMD, Separator = .
Britisch/French	dd/mm/yy	Config Date = DMY, Separator = /
German	dd.mm.yy	Config Date = DMY, Separator = .
Italian	dd-mm-yy	Config Date = DMY, Separator = -
Japan/Taiwan	yy/mm/dd	Config Date = YMD, Separator = /
USA	mm-dd-yy	Config Date = MDY, Separator = -

مثال :

```

$regfile = "M16DEF.DAT"
$crystal = 8000000

Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , E = Pind.1 , Rs = Pind.0

Config Lcd = 16 * 2

Enable Interrupts

Config Date = Dmy , Separator = .

Config Clock = Soft

Date$ = "10/11/02" : Time$ = "02:20:00"

Do

Locate 1 , 1 : Lcd Date$ : Locate 2 , 1 : Lcd Time$

Loop

End

```

مثال :

```

$regfile = "M16DEF.DAT"
$crystal = 8000000

Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , E = Pind.1 , Rs = Pind.0

Config Lcd = 16 * 2

Enable Interrupts

Config Date = Ymd , Separator = /

Config Clock = Soft

Date$ = "10/11/02" : Time$ = "02:20:00"

Do

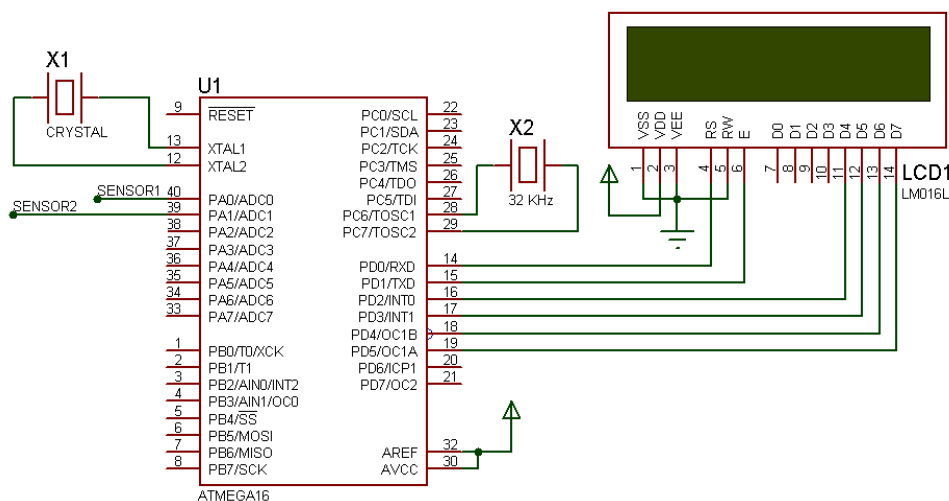
Locate 1 , 1 : Lcd Date$ : Locate 2 , 1 : Lcd Time$

Loop

End

```

مدار استفاده شده برای مثال بالا را در زیر مشاهده می کنید:



با استفاده از دستور زیر میتوانید روزهای هفته را مشخص کنید:

Target = DayOfWeek()

Target = DayOfWeek(): با این دستور ، به ازای هر روزی که از هفته میگذرد یک واحد به متغیر Target افزوده میشود .

این متغیر از صفر تا شش تغییر میکند ، شما با استفاده از جدول Lookupstr میتوانید زور های هفته را به راحتی جایگزین اعداد کنید. مثال :

```
$regfile = "M16DEF.DAT" : $crystal = 8000000
```

```
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , E = Pind.1 , Rs = Pind.0
```

```
Config Lcd = 16 * 2 : Cursor Off : Enable Interrupts
```

```
Dim Bweekday As Byte , Strweekday As String * 10
```

```
Config Date = YMD , Separator = .
```

```
Config Clock = Soft
```

```
_sec = 59 : _min = 59 : _hour = 23 : _day = 26 : _month = 11 : _year = 2
```

```
Do
```

```
Locate 1 , 1 : Lcd Date$ : Locate 2 , 1 : Lcd Time$
```

```
Bweekday = Dayofweek()
```

```
Strweekday = Lookupstr(bweekday , Weekdays)
```

```
Locate 1 , 10 : Lcd Bweekday
```

```
Locate 2 , 10 : Lcd Strweekday
```

```
Loop
```

```
End
```

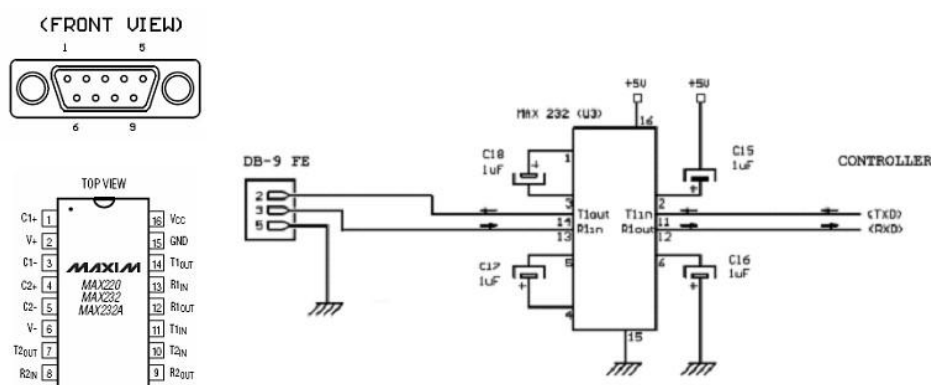
Weekdays:

```
Data "Monday" , "Tuesday" , "Wednesday" , "Thursday" , "Friday" , "Saturday" , "Sunday"
```

دستورات دیگری نیز مانند دستور DayOfWeek وجود دارد که با آنها میتوان روزهای گذشته از ماه ، سال ، ثانیه گذشته از ماه ، سال ، دقیقه گذشته از ماه سال و.. را مشخص کرد . شما میتوانید این دستورات را در help نرم افزار بسکام بیابید ، استفاده از این دستورات مانند دستور DayOfWeek است.

ارتباط سریال RS232 :

RS232 یکی از پروتکل های ارتباطی مهم در دنیای میکرو کنترلر ها میباشد . از این پروتکل برای ارتباط میکرو با میکرو ، میکرو با کامپیوتر و کامپیوتر با کامپیوتر استفاده میشود ، برای ارتباط میکرو با میکرو شما نیاز به دو برنامه برای دو میکرو دارید ، برای ارتباط میکرو با کامپیوتر شما باید علاوه بر نوشتن برنامه برای میکرو یک برنامه نیز برای کامپیوتر بنویسید و در نهایت برای ارتباط دو کامپیوتر با هم شما باید با یکی از زبان های برنامه نویسی برای هر دو کامپیوتر برنامه بنویسید ، در این بخش ما نحوه ی پیداسازی کلیه موارد بالا را به شما خواهیم آموخت .



دستورات این پروتکل در بسکام به شرح زیر است:

تعیین نرخ انتقال دیتا:

\$BAUD=VAR

این دستور میزان انتقال دیتا در ثانیه را مشخص میکند و باید در هر دو وسیله ای که به هم متصل میشوند یکی باشد ، نرخ انتقال داده مقدار داده ای که در یک ثانیه بین دو وسیله ی متصل شده جابجا میشود ، مشخص میکند این مقدار با توجه به فرکانس کاری میکرو و فاصله ی دو وسیله مشخص میشود ، با مراجعه به فایل conn در پوشه ی پیوست میتوانید مقادیر استاندارد نرخ انتقال داده به ازای کریستال های مختلف و مقدار خطای حاصله را مشاهده کنید

بعضی از میکرو کنترلر های avr مانند ATXMEGA128A1 یا ATMEGA2560 یا ... دارای چند پورت UART میباشند ، شما میتوانید با دستورات زیر پورت مورد استفاده را تعیین و پیکربندی نمایید :

سری ATMEGA :

CONFIG COMx = baud , synchrone=0|1,parity=none|disabled,stopbits=1|2,databits=4|6|7|8|9,clockpol=0|1

COMx : شماره ی پورتی است که قصد دارید آن را راه اندازی کنید ، X میتواند از 1 تا چهار باشد ، چون Mega2560 میتواند 4 واحد UART را پشتیبانی کند .

Baud : مقدار نرخ انتقال داده است که در صفحه ی قبل به بررسی آن پرداختیم .

Synchrone : رقم 1 برای کارکرد میکرو در مد سنکرون و رقم صفر برای کارکرد میکرو در مد آسنکرون قرار دهید .

Parity : این بخش ، وجود یا عدم وجود بیت تشخیص خطا را مشخص میکند ، None باعث میشود که کامپایلر با توجه به

برنامه بهترین گزینه را انتخاب نماید ، disabled بیت تشخیص خطا را غیر فعال میکند .

Stopbits : تعداد بیت های توقف ارسال با این دستور مشخص میشود .

Databits : تعداد بیت های داده در هر فریم با این دستور مشخص میگردد

Clockpol : پلاریته ی اولین پالس کلاک با این دستور مشخص میگردد ،

سری ATXMEGA :

CONFIG COMx = baud , Mode=mode, Parity=parity, Stopbits=stopbits, Databits=databits,clockpol=Clockpol

: Mode

1 یا SYNCHRONEOUS برای کارکرد میکرو در مد سنکرون

0 یا ASYNCHRONEOUS برای کارکرد میکرو در مد آسنکرون

IRDA یا IRCOM : برای کارکرد میکرو در مد IRDA

SPI یا MSPI : برای کارکرد میکرو در مد SPI

سایر موارد مطابق با موارد گفته شده برای سری ای.تی.مگا میباشد .

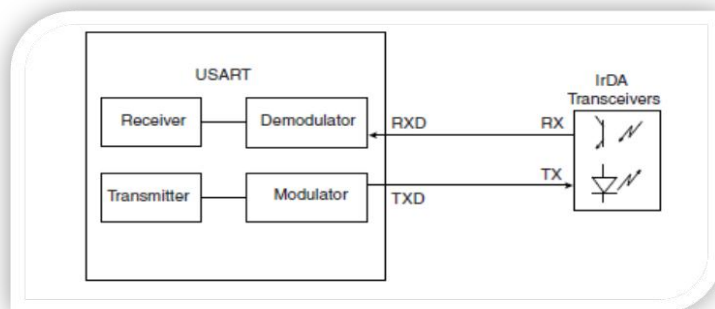
توضیحات بیشتر :

احتمالا تا کنون پورت Infrared یا به اصطلاح مادون قرمز را بر روی گوشی موبایل خود دیده اید ، این بخش از یک

فرستنده / گیرنده مادون قرمز تشکیل شده است و شما میتوانید توسط آن هر چیزی را از گوشی خود به گوشی دیگر

منتقل کنید .

Led های فرستنده / گیرنده موجود در این بخش ، در یک بسته بندی قرار گرفته اند (یک قطعه مجزا) و اتصال آنها به میکرو کنترلر طبق شکل زیر است (یک نمونه مشهور از این قطعات سری tfd4xxx میباشند ، توجه داشته باشید که استفاده از led های مادون قرمز معمولی به جایی این قطعه باعث کاهش کیفیت داده ی ارسالی و بالا رفتن خطا میشود).



این قطعه داده ی خروجی از باس uart میکرو کنترلر فرستنده (که متناسب با ورودی قطعه پیکرپندی شده است) را دریافت کرده (داده ی خروجی میکرو بر حسب استاندارد IrDA version 1.1 با یک پالس دیگر مدوله میشود) و آن را از طریق فرستنده ی مادون قرمز در محیط اطراف منتشر میکند .

در آن طرف گیرنده داده منتشر شده را دریافت کرده و آن را به میکرو تحویل میدهد و میکرو بعد از کدگشایی داده ، آن را به بخش های بعدی که در برنامه معین میشوند ، ارسال میکند .

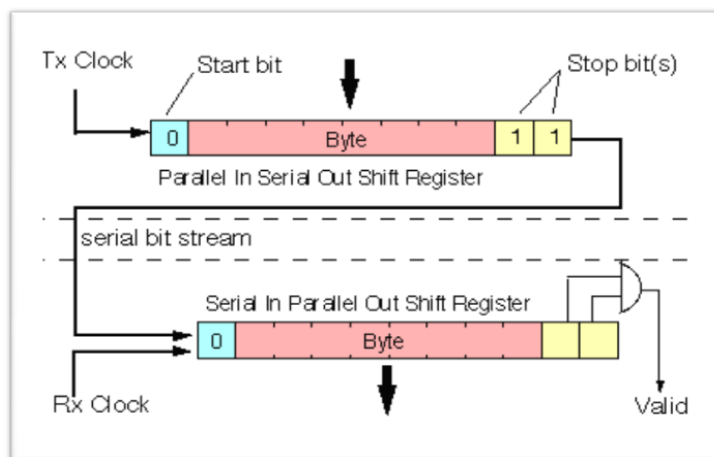
✓ مبادله آسنکرون (Asynchronous Transmission)

در این روش ، انتقال داده به صورت غیر همزمان انجام می شود ، در واقع بین فرستنده و گیرنده همزمانی وجود ندارد . فرستنده داده موجود را ارسال می کند و گیرنده نیز آن را دریافت خواهد نمود . در این میان بیت های استارت و همزمانی ، زمان ارسال داده و صحیح بودن آن را مشخص می کنند .

در مبادله آسنکرون هر داده کاراکتری، یک بیت شروع (start bit) که مبین شروع آن کاراکتر است و همچنین یک یا دو بیت پایان (Stopbits) که تعیین کننده پایان آن کاراکتر می باشد.

در این رابط ، داده ارسالی معمولاً یک بایت (8 بیت) می باشد . به دنبال بیت های داده، یک بیت توازن (parity bit) وجود دارد که دریافت کننده آنرا به منظور خطایابی مورد استفاده قرار می دهد. پس از اینکه بیت Parity ارسال شد سیگنال باید حداقل برای مدت زمان یک بیت دارای ارزش 1 شود تا پایان کاراکتر را معین کند. بیت جدید شروع (Start bit) موجب می

شود که وسیله دریافت کننده از آمدن یک کاراکتر داده، آگاه شود و ساعت خود را سنکرون سازد. در صورتی که بیت Parity داده قبلی صحیح باشد، گیرنده برای دریافت ادامه اطلاعات آماه می شود، در غیر این صورت مجددا اطلاعات قبلی را درخواست می کند:



پورت های RS232 (UART) و RS458 و... نمونه های از Asynchronous Transmission هستند.

✓ مبادله سنکرون (Synchronous Transmission)

در این روش نیازی به بیت های شروع و پایان و... نیست و پالس کلاک وظیفه ایجاد همزمانی میان دو دستگاه را برعهده دارد. در این روش در هر پالس کلاک یک بیت از اطلاعات ارسال می شود، بنابراین ما می توانیم با تغییر دادن فرکانس کلاک سرعت انتقال را عملاً تغییر دهیم، پروتکل SPI و i2s و... نمونه های از Synchronous Transmission هستند. در این روش علاوه بر ورودی و خروجی های داده به یک پایه ی COLCK نیز نیاز خواهیم داشت.

✓ SPI یا MSPI :

در ادامه با پروتکل SPI بیشتر آشنا خواهیم شد، این پروتکل از سه پایه برای ارتباط با قطعه ی جانبی استفاده میکند، در صورتی که ما از واحد UART میکروکنترلر در مدی شبیه مد سنکرون استفاده نماییم (UART) در این حالت انتقال داده ای شبیه به SPI خواهیم داشت، با بیشتر کردن سرعت انتقال داده و افزودن یک پایه ی CS، این پروتکل دقیقاً شبیه به SPI عمل خواهد کرد.

پس در سری XMEGA میتوانید از پورت COM به عنوان باس SPI نیز استفاده نمایید.

توجه داشته باشید که دستور \$BAUD=VAR به صورت پیش فرض پورت COM1 (RX0 & TX0) را در میکرو کنترلر با تنظیمات No parity, 1 stopbit, 8 data bits پیکربندی میکند. شما مجاز به استفاده از این دستور در کنار دستور CONFIG COMx نیستید.

در سری XMEGA، پایه های زیر برای پورت COM به کار میروند :

UART	TX pin	RX pin
COM1	PORTC.3	PORTC.2
COM2	PORTC.7	PORTC.6
COM3	PORTD.3	PORTD.2
COM4	PORTD.7	PORTD.6
COM5	PORTE.3	PORTE.2
COM6	PORTE.7	PORTE.6
COM7	PORTF.3	PORTF.2
COM8	PORTF.7	PORTF.6

تعیین ساینز بافر ارسال و دریافت داده :

در میکرو کنترلر های که دارای یک پورت سریال هستند، ارسال داده بدون هیچ مشکلی انجام میشود، در این میکرو کنترلر ها CPU داده ی ارسالی را به واحد UART میکرو کنترلر ارسال کرده و در انجا داده با توجه به نرخ انتقال داده و حجم بافر ارسال (حافظه ی SRAM) در یک یا چند بخش ارسال میگردد.

در میکرو کنترلر های که دارای چندین پورت سریال هستند، عدم وجود فضای کافی در حافظه ی SRAM همیشه داده ی ارسالی یا دریافتی را تهدید خواهد کرد، برای مثال فرض کنید در برنامه ای یک رشته ی 100 بایتی را توسط پورت سریال شماره ی یک ارسال میشود، و سپس داده ی بزرگی دیگری توسط پورت سریال دو ارسال میشود. در این برنامه داده ی پورت سریال دو به جای داده ی سریال پورت یک در حافظه ی SRAM نشسته و باعث پاک شدن آن میشود. برای جلوگیری از بروز چنین مشکلی میتوان از دستورات زیر استفاده نمود :

CONFIG SERIALOUT | SERIALOUT1 | SERIALOUT2 | SERIALOUT3 = BUFFERED , SIZE = size

در این دستور SERIALOUT یک تا 3 نام بافر های مورد استفاده برای پورت های سریال یک تا سه هستند.

همچنین ساینز عددی بین 1 تا 255 برحسب بایت برای مشخص کردن حجم بافر ارسال داده است.

```
CONFIG SERIALIN | SERIALIN1 | SERIALIN2 | SERIALIN3 = BUFFERED , SIZE = size [, BYTEMATCH=ALL|BYTE|NONE]
[,CTS=pin, RTS=pin , Threshold_full=num , Threshold_empty=num ]
```

در این دستور SERIALIN یک تا 3 نام بافر های مورد استفاده برای پورت های سریال یک تا سه هستند . همچنین ساینز عددی بین 1 تا 255 برحسب بایت برای مشخص کردن حجم بافر دریافت داده است .

از دستور ALL با وارد شدن هر کارکتر اسکی به بافر داده زیر برنامه تعیین شده مطابق دستور زیر پرش میکند :

- Serial0ByteReceived (for SERIALIN or the first UART/UART0)
- Serial1ByteReceived (for SERIALIN1 or the second UART/UART1)
- Serial2ByteReceived (for SERIALIN2 or the third UART/UART2)
- Serial3ByteReceived (for SERIALIN3 or the fourth UART/UART3)

در صورتی که در این دستور به جای بایت کد اسکی کارکتر مورد نظر قرار گیرد ، با وارد شدن آن کارکتر به بافر دریافت داده CPU به برچسب معرفی شده پرش میکند ، استفاده از دستور NONE باعث غیر فعال شدن این قابلیت خواهد شد .

با این دستور میتوان پایه های را برای ارسال وضعیت CTS=pin, RTS=pin , Threshold_full=num , Threshold_empty=num : های Clear to send و Ready to send پیکربندی کرد . دستور Threshold_full=num تعداد بایت های مورد نیاز برای یک شدن پایه ی RTS و دستور Threshold_empty=num مشخص کننده ی تعداد بایت های خالی برای صفر شدن CTS است . دستور PRINT :

```
PRINT VAR
```

توسط این دستور میتوان داده یا متغیری را به پورت سریال ارسال کرد. VAR یک متغیر از جنس متغیرهای گفته شده میباشد. این دستور متغیر VAR به COM1 ارسال میکند ، در صورتی که قصد دارید متغیر موجود را به پورت های دیگر استفاده کنید ، قبل از هر چیز باید پورت مورد نظر را باز نمایید :

```
Open "comX:" For Binary As #X
```

X شماره ی پورت مورد نظر میباشد و #1 کارکتر شناسایی پورت است :


```

$regfile = "m2560def.dat" ' specify the used micro
$crystal = 8000000 '
'The M128 has an extended UART.
'when CO'NFIG COMx is not used, the default N,8,1 will be used
Config Com1 = 19200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0
Config Com2 = 19200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0
Config Com3 = 19200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0
Config Com4 = 19200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0 'Open all UARTS
Open "com2:" For Binary As #1
Open "Com3:" For Binary As #2
Open "Com4:" For Binary As #3
Print "Hello" 'first uart
Dim B As Byte
Dim Tel As Word
Do
  Incr Tel
  Print Tel ; " test serial port 1"
  Print #1 , Tel ; " test serial port 2"
  Print #2 , Tel ; " test serial port 3"
  Print #3 , Tel ; " test serial port 4"
  B = Inkey(#3)
  If B <> 0 Then
    Print #3 , B ; " from port 4"
  End If
  Waitms 500
Loop
Close #1
Close #2
Close #3
End

```

دستور CLOSE #X پورت COMx را مینماید .

مثال :

```

$regfile = "m128def.dat" ' specify the used micro
$crystal = 4000000 ' used crystal frequency
$baud = 9600 ' use baud rate
$hwstack = 40 ' default use 32 for the hardware stack
$swstack = 40 ' default use 10 for the SW stack
$framesize = 40 ' default use 40 for the frame space

```

Config Com1 = Dummy , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0

Config Com2 = Dummy , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0

Config Serialout = Buffered , Size = 20'setup to use a serial output buffer and reserve 20 bytes for the buffer

Enable Interrupts 'It is important since UDRE interrupt is used that you enable the interrupts

Print "Hello world"

Print "test1"

Do

Wait 1'notice that using the UDRE interrupt will slow down execution of waiting loops like waitms

Print "test"

Loop

End

دستور PRINTBIN :

PRINTBIN var [; varn]

PRINTBIN #channel, var [; varn]

توسط این دستور متغیر VAR به باینر تبدیل شده سپس به پورت سریال ارسال میشود. در این دستور امکان ارسال تعداد varn

متغیر از متغیر آرایه ای var وجود دارد. مثال :

```
Printbin ar(1) ; 3 ' will send 3 bytes from array ar().
Printbin ar(1) ; 2 ; ar(2) ; 4 ' will send 2 bytes from array ar() starting at index 1, then 4
bytes from array ar() starting at index 4.
```

در صورتی که مقدار varn از 255 بیشتر شود، باید دستور زیر قبل از دستور PRINTBIN آورده شود :

CONFIG PRINTBIN = extended

مثال :

\$regfile = "m103def.dat" ' specify the used micro

\$crystal = 8000000 ' used crystal frequency

\$baud = 19200 ' use baud rate

\$hwstack = 32 ' default use 32 for the hardware stack

\$swstack = 10 ' default use 10 for the SW stack

\$framesize = 40 ' default use 40 for the frame space

Config Com1 = Dummy , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0

Config Printbin = Extended

Dim A(1000)

Printbin A(1) ; 1000

دستور WAITKEY :

VAR=WAITKEY()

این دستور تا زمانی که متغیر توسط دستگاه دیگر به پورت سریال ارسال شود منتظر میماند و پس از دریافت متغیر برنامه از خط بعد ادامه می یابد.

دستور INKEY :

VAR=INKEY()

این دستور مقدار اسکی کاراکتر دریافت شده از پورت سریال را برمیگرداند.

مثال :

\$regfile = "m161def.dat" ' specify the used micro

\$crystal = 4000000 ' used crystal frequency

\$baud = 9600 ' use baud rate

\$hwstack = 32 ' default use 32 for the hardware stack

\$swstack = 10 ' default use 10 for the SW stack

\$framesize = 40 ' default use 40 for the frame space

'first compile and run this program with the line below remarked

Config Serialin = Buffered , Size = 20

Dim Na As String * 10

'the enabling of interrupts is not needed for the normal serial mode

'So the line below must be remarked to for the first test

Enable Interrupts

Print "Start"

Do

'get a char from the UART

If Ischarwaiting() = 1 Then 'was there a char?

Input Na

Print Na 'print it

End If

Wait 1 'wait 1 second

Loop

'You will see that when you slowly enter characters in the terminal emulator they will be received/displayed. When you enter them

fast you will see that you loose some chars NOW remove the remarks from line 11 and 18 and compile and program and run again

This time the chars are received by an interrupt routine and are stored in a buffer. This way you will not loose characters providing

that you empty the buffer So when you fast type abcdefg, they will be printed after each other with the 1 second delay Using the

'CONFIG SERIAL=BUFFERED, SIZE = 10 for example will use some SRAM memory' The following internal variables will be generated

:'_Rs_head_ptr0 BYTE , a pointer to the location of the start of the buffer

'_Rs_tail_ptr0 BYTE , a pointer to the location of tail of the buffer

'_RS232INBUF0 BYTE ARRAY , the actual buffer with the size of SIZE

مثال :

```
$regfile = "m2560def.dat" ' specify the used micro
$crystal = 8000000 ' used crystal frequency
$hwstack = 40 ' default use 32 for the hardware stack
$swstack = 40 ' default use 10 for the SW stack
$framesize = 40 ' default use 40 for the frame space
'$timeout = 1000000
'The M128 has an extended UART.
'when CO'NFIG COMx is not used, the default N,8,1 will be used
Config Com1 = 19200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0
Config Com2 = 19200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0
Config Com3 = 19200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0
Config Com4 = 19200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0
Enable Interrupts
Config Serialin = Buffered , Size = 20
Config Serialin1 = Buffered , Size = 20 , Bytematch = 65
Config Serialin2 = Buffered , Size = 20 , Bytematch = 66
Config Serialin3 = Buffered , Size = 20 , Bytematch = All
'Open all UARTS
Open "COM2:" For Binary As #2
Open "COM3:" For Binary As #3
Open "COM4:" For Binary As #4
Print "Hello" 'first uart
Dim B1 As Byte , B2 As Byte , B3 As Byte , B4 As Byte
Dim Tel As Word , Nm As String * 16
'unremark to test second UART
'Input #2 , "Name ?" , Nm
'Print #2 , "Hello " ; Nm
Do
Incr Tel
Print Tel ; " test serial port 1"
Print #2 , Tel ; " test serial port 2"
Print #3 , Tel ; " test serial port 3"
Print #4 , Tel ; " test serial port 4"
B1 = Inkey() 'first uart
B2 = Inkey(#2)
B3 = Inkey(#3)
```

```

B4 = Inkey(#4)
If B1 <> 0 Then
Print B1 ; " from port 1"
End If
If B2 <> 0 Then
Print #2 , B2 ; " from port 2"
End If
If B3 <> 0 Then
Print #3 , B3 ; " from port 3"
End If
If B4 <> 0 Then
Print #4 , B4 ; " from port 4"
End If
Waitms 500
Loop
'Label called when UART2 received an A
Serial1charmatch:
Print #2 , "we got an A"
Return
'Label called when UART2 received a B
Serial2charmatch:
Print #3 , "we got a B"
Return
'Label called when UART3 receives a char
Serial3bytereceived:
Print #4 , "we got a char"
Return
End
Close #2
Close #3
Close #4
$eeprom
Data 1 , 2

```

در مثال بالا با دستور Config Serialin2 = Buffered , Size = 20 , Bytematch = 66 در صورتی که کد اسکی داده وارد شده به بافر دریافت داده برابر با 66 باشد به برچسب Serial2charmatch: پرش شده و در آنجا عبارت we got a B از طریق پورت سریال شماره ی سه به دستگاه جانبی متصل شده ارسال میشود . عدد 66 کد اسکی کارکتر B است . دستور INPUTBIN :

INPUTBIN VAR

این دستور داده باینری را از پورت سریال میگیرد و در متغیر VAR قرار میدهد.

دستور INPUTHEX :

INPUTHEX VAR

این دستور داده هگز را از پورت سریال دریافت میکند و در متغیر VAR قرار میدهد.

مثال:

در مثال زیر با استفاده از ارتباط سریال ، یک ارتباط دوطرفه بین دو میکرو برقرار کرده ایم ، در زیر برنامه مدار مربوطه و

توضیحات برنامه را مشاهده میفرمایید.

```
$regfile = "m32def.dat" : $crystal = 1000000
```

```
$baud = 9600
```

```
Config Portb = Input : Config Porta = Output
```

```
Dim A As Byte , Q As Byte
```

```
W:
```

```
Q = Pinb : Printbin Q
```

```
A = Inkey() : Porta = A
```

```
Goto W
```

```
End
```

میکرو 2:

```
$regfile = "xm128a1def.dat"
```

```
Config Osc = Enabled , Pllosc = Enabled , Startup = Xtal_256clk
```

```
Config Sysclock = 2mhz , Prescalea = 2 , Prescalebc = 1_2
```

```
Config Com1 = 9600 , Mode = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0
```

```
Config Portb = Input : Config Porta = Output
```

```
Dim A As Byte : Dim Q As Byte
```

```
W:
```

```
Q = Pinb : Printbin Q
```

```
A = Inkey() : Porta = A
```

```
Goto W
```

```
End
```

در این برنامه یک ارتباط دو طرفه میان دو میکرو کنترلر ATMEGA32 و ATXMEGA128A1 ایجاد شده است ، همان طور که در برنامه مشاهده میکنید داده ی موجود بر روی پورت B میکرو کنترلر ها در یک متغیر قرار داده میشود و سپس به میکرو کنترلر دیگر ارسال میشود ، در میکرو کنترلر دیگر داده ی دریافت شده در یک متغیر قرار گرفته و سپس بر روی پورت A ریخته میشود ، این یعنی :

به ازای پایه ی یک شده از پورت B ، پایه ی متناظر در پورت A میکرو کنترلر دیگر روشن میشود .

هر چند در این برنامه به دلیل استفاده از کریستال های متفاوت ، ممکن است مقدار خطای ارسالی/دریافتی بالا رفته و شما نتوانید داده ی مناسب را در تست عملی مشاهده نمایید ، اما در بخش پروژه های کاربردی چندیدن مثال عملی برای این بخش آورده شده است .

انتقال پایه های انتقال داده به پین های دلخواه (ایجاد پورت سریال نرم افزاری)

در میکرو کنترلر های Avr واحدی مجزا برای ارتباط سریال uart تعییه شده است . هنگامی که در برنامه از دستور پرینت استفاده میکنید ، CPU داده ی مورد نظر شما را به واحد UART ارسال کرده و این واحد عملیات انتقال داده را بر عهده میگیرد . کنترل کردن خطا ، آمادگی برای دریافت داده ، ارسال همزمان و صحیح داده و... از جمله قابلیت های این واحد است . در کامپایلر بسکام این امکان برای کاربران وجود دارد تا با استفاده از دستورات زیر اقدام به ایجاد پورت سریال نرم افزاری نمایند ، در پورت سریال نرم افزار خبری از بافر ارسال و دریافت داده و کنترل نرخ ارسال و... نیست و داده شما به صورت آنی بعد از اجرا شدن دستور پرینت (یا سایر دستورات مرتبط) به صورت سریال از طریق پایه ی تعیین شده خارج میشود .

عدم وجود کنترل سخت افزاری بر روی داده ارسالی / دریافتی باعث بالا رفتن احتمال وقوع خطا خواهد شد ، از این رو توصیه میشود برای کار های حرفه ای ، در صورت نیاز به پورت های سریال بیشتر از میکرو کنترلر های نظیر ATXMEGA یا Atmega 2560 یا سایر میکرو کنترلر های که دارای تعداد بیشتری پورت سریال سخت افزاری هستند استفاده کنید .

با دستور زیر شما میتوانید پایه دلخواه را به عنوان ورودی /خروجی سریال تعیین کنید :

Open "comx.y:\$baud,8,n,1" For Output/input As #q

comx.y: نام پورت و پایه ای است که باید به عنوان txd یا rxd جدید عمل کند.

\$baud: نرخ داده عبوری از پایه را نشان می‌دهد، این مقدار باید با نرخ انتقال در دستگاه جانبی برابر باشد.

Output/input: پایه می‌تواند وردی داده (rxd) یا خروجی داده (txd) باشد.

Q: شماره کانال را مشخص می‌کند.

CONFIG WAITSUART = value

در دستور بالا value عددی بین 0 تا 255 می‌باشد. value تاخیری بر حسب میلی ثانیه را بعد از ارسال هر بایت ایجاد می‌کند.

مثال:

Open "comd.1:19200,8,n,1" For Output As #1

Open "comd.0:19200,8,n,1" For Input As #2

CONFIG WAITSUART = 100

در مورد بالا portd.1 به عنوان txd و portd.0 به عنوان rxd نرم افزاری در نظر گرفته شده است، همچنین نرخ انتقال داده برابر

با 19200 است. دستور Close باعث تغییر وضعیت پایه های بالا به I/O عادی می‌شود. مثال:

\$regfile = "m48def.dat" ' specify the used micro

\$crystal = 10000000 ' used crystal frequency

\$baud = 19200 ' use baud rate

Dim B As Byte 'Optional you can fine tune the calculated bit delay

'Why would you want to do that? Because chips that have an internal oscillator may not run at the speed specified. This depends on the voltage, temp etc. You can either change \$CRYSTAL or you can use BAUD #1,9610

'In this example file we use the DT006 from www.simmstick.com This allows easy testing with the existing serial port The MAX232 is fitted for this example.

'Because we use the hardware UART pins we MAY NOT use the hardware UART The hardware UART is used when you use PRINT, INPUT or other related statements We will use the software UART.

Waitms 100

open "comd.1:19200,8,n,1" For Output As #1 'open channel for output

Print #1 , "serial output"

open "comd.0:19200,8,n,1" For Input As #2 'NoW open a pin for input

'since there is no relation between the input and output pin there is NO ECHO while keys are typed

Print #1 , "Number"

'get a number

Input #2 , B'print the number

Print #1 , B

'now loop until ESC is pressed With INKEY() we can check if there is data available To use it with the software UART you must

provide the channel

Do 'store in byte

B = Inkey(#2) 'when the value > 0 we got something

If B > 0 Then

Print #1 , Chr(b) 'print the character

End If

Loop Until B = 27

Close #2

Close #1

'OPTIONAL you may use the HARDWARE UART The software UART will not work on the hardware UART pins so you must

choose other pins . use normal hardware UART for printing Print B

'When you dont want to use a level inverter such as the MAX-232 . You can specify ,INVERTED : Open

"comd.0:300,8,n,1,inverted" For Input As #2 . Now the logic is inverted and there is no need for a level converter But the distance

of the wires must be shorter with this

End

بررسی محیط :TERMINAL EMULATOR

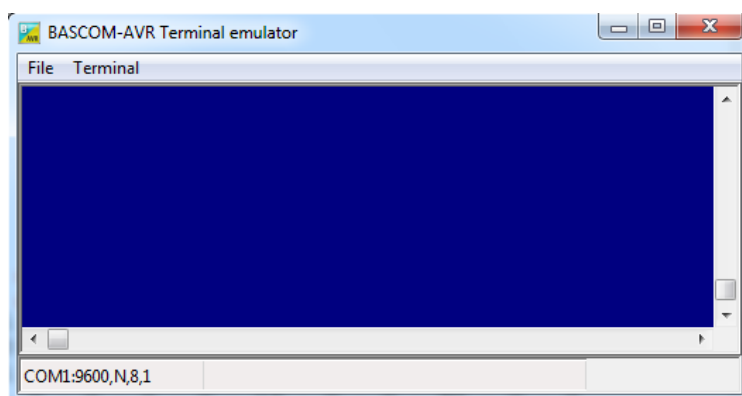
از این محیط برای نمایش داده ارسالی و دریافتی در ارتباط سریال RS-232 بین میکرو و کامپیوتر استفاده میشود. اطلاعاتی

که در این محیط تایپ می شود به میکرو ارسال و اطلاعاتی که از پورت سریال کامپیوتر (COM) دریافت می شود در این

پنجره نمایش داده می شود . هنگامیکه در برنامه از SERIAL IN و یا SERIAL OUT استفاده می شود , پس از PROGRAM

کردن برنامه درون میکرو و اتصال آن به پورت سریال PC, می توان داده های ارسالی توسط UART میکرو را دریافت کرده و نمایش داد و از صحت آنها اطلاع یافت. همچنین اگر از دستوری مانند INKEY در برنامه استفاده شود, میتوان داده خود را از طریق پنجره TERMINAL EMULATOR به میکرو ارسال نمود.

برای مشاهده ی این پنجره از منوی tools گزینه ی Terminal Emulator را انتخاب نمایید:

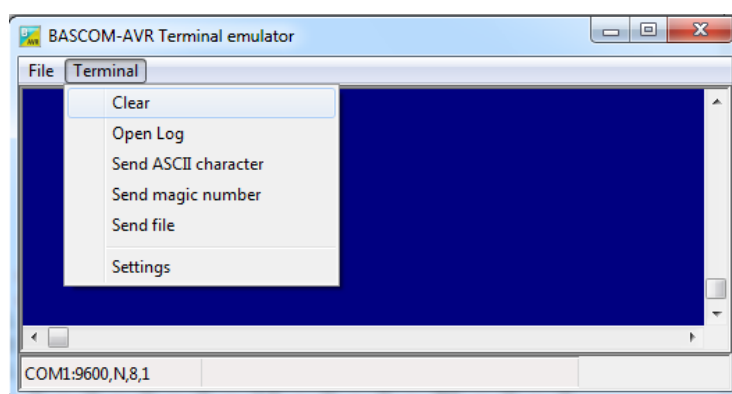


در زیر به بررسی این محیط و گزینه های موجود در دو منوی FILE و Terminal پرداخته ایم:

FILE UPLOAD: برنامه جاری در فرمت HEX را UPLOAD میکند

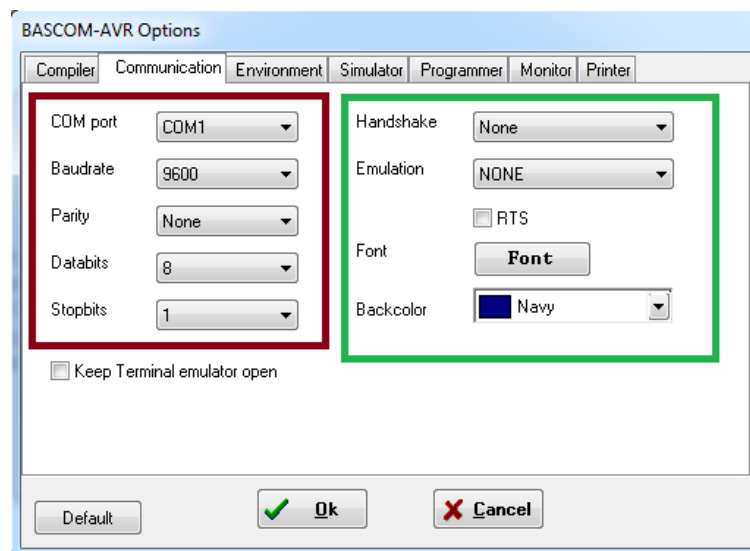
FILE ESCAPE: صرفنظر کردن از UPLOAD کردن فایل.

FILE EXIT: خروج از برنامه EMULATOR.



TERMINAL CLEAR: پنجره ترمینال را پاک می کند.

SETTING: تنظیمات پورت COM و دیگر OPTION ها توسط این منو صورت می گیرد:



در این پنجره باید، مقادیر نرخ انتقال داده، بیت توازن و تعداد بیت ارسالی و... با مقادیر درج شده در برنامه ی میکرو یکسان باشد تا انتقال اطلاعات به درستی انجام شود.

LOG : TERMINAL OPEN LOG فایل را باز یا بسته می کند. هنگامیکه فایل LOG وجود نداشته باشد درخواست نامی برای فایل گزارش می کند. تمام اطلاعاتی که در پنجره TERMINAL پرینت می شود داخل فایل LOG ثبت می شود.

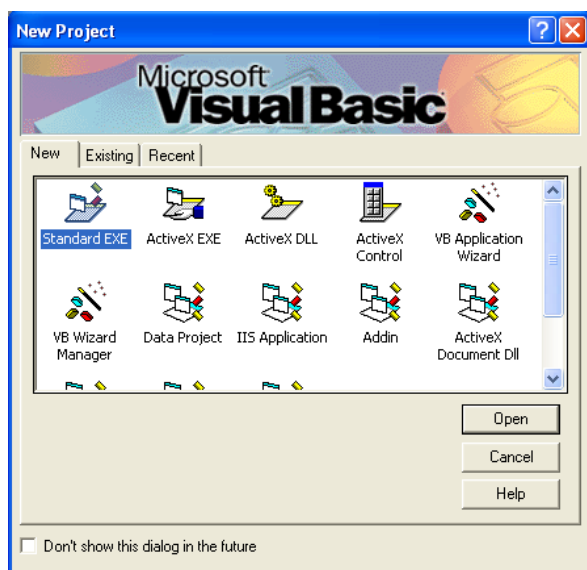
بعد از انجام دادن تنظیمات و اتصال میکرو و کنترلر به پورت سریال، داده ی ارسال شده از طرف میکرو کنترلر در محیط آبی رنگ نمایش داده میشود. شما همچنین میتوانید با تایپ کردن حروف و اعداد مورد نیاز و زدن کلید enter، آنها را به سمت میکرو کنترلر ارسال کنید.

ارتباط میکرو AVR با پورت سریال کامپیوتر :

برای برقراری ارتباط میان میکرو کنترلر و کامپیوتر نیاز به یک برنامه کامپوتری برای باز کردن، و خواندن / نوشتن اطلاعات در پورت سریال (پورت COM) کامپیوتر داریم. برنامه ی مورد نیاز میتواند با یکی از زبان های برنامه نویسی همچون ویژوال بیسیک نوشته شود.

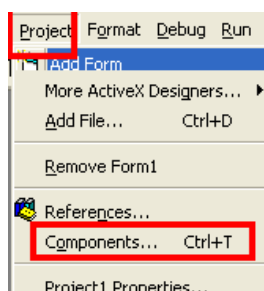
نوشتن برنامه ویژوال بیسیک :

بعد از نصب نرم افزار ویژوال بیسیک 6، آن را باز کنید، در اولین صفحه ای که نمایان می شود، گزینه ی standard EXE را انتخاب نمایید :



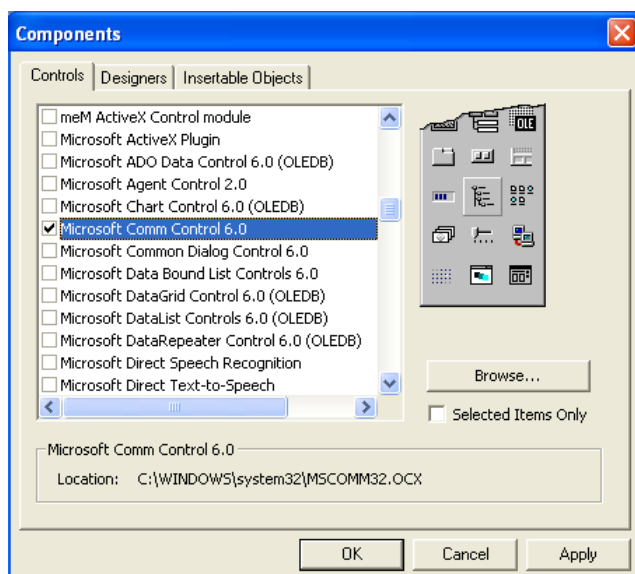
مشاهده میکنید که پنجره ای برای ایجاد ظاهر نرم افزار نمایش داده میشود . به منوی Project بویید و در آنجا گزینه ی

Components را انتخاب کنید :

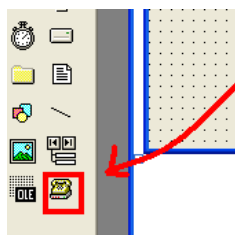


پنجره ای مطابق شکل باز می شود ، در این پنجره گزینه ی MICROSOFT COMM CONTROL 6.0 را پیدا کنید و تیک کنار

آن را بگذارید و سپس بر روی OK کلیک نمایید .

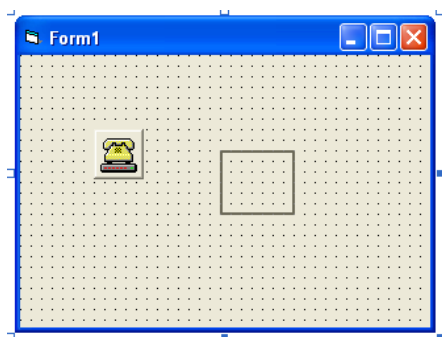


مشاهده میکنید که رویداد انتخاب شده به نوار ابزار اضافه میشود :

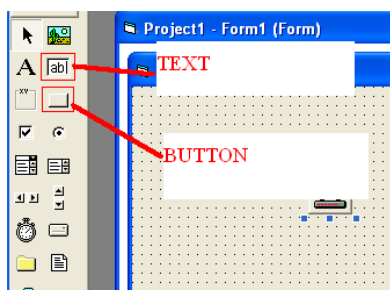


بر روی رویداد کلیک کنید و سپس به FORM بروید و آن را در فرم ایجاد کنید ، برای ایجاد یک گزینه در فرم ، بعد از

انتخاب آن از نوار ابزار در یک مکان مناسب کلیک کنید ، بکشید و رها کنید:



از نوار ابزار دو عدد TEXT و یک کلیک انتخاب کنید و آنها را در صفحه FORM ایجاد کنید :



مجموعه دستورات VB برای پورت COM :

```
MSComm1.CommPort = X
```

با این دستور آدرس پورت مشخص میشود ، X میتواند 0 یا 1 یا دو باشد (در کامپیوتر های امروزی معمولا دو عدد پورت

کام وجود دارد ، در صورت وجود پورت های بیشتر X بیشتر می شود)

```
MSComm1.Settings = "BAUD,N,8,1"
```

BAUD نرخ انتقال اطلاعات در ثانیه میباشد و باید در اینجا و در برنامه نوشته شده برای میکرو یکی باشد ، نرخ انتقال به شدت ، به مقدار کریستال وابسته است ، در ضمیمه ها اطلاعات بیشتر آورده شده است :

```
MSComm1.PortOpen = True / false
```

بوسیله ی این دستور میتوان پورت را باز کرد یا بست ، پورت با True باز و با false بسته می شود . بعد از باز کردن پورت ، در صورتی که پورت توسط دستگاه دیگری استفاده نشود می توانید داده خود را ارسال یا دریافت کنید .

```
MSComm1.Output = x
```

با این دستور متغیر X به بیرون فرستاده می شود ، متغیر X میتواند یکی از متغیر های تعریف شده در VB باشد . توجه داشته باشد که در VB تعداد زیادی متغیر وجود دارد ، در صوتی که شما کارکتری را در متغیر می ریزید و ارسال میکنید ، در میکرو باید با متغیر مشابه کارکتر را دریافت کنید تا اطلاعات خراب نشود .

```
X = MSComm1.Input
```

توسط این دستور میتوان مقدار موجود بر روی پورت COM را خواند ، مقدار خوانده شد در متغیر X ریخته میشود .

مثال :

در پروژه ای که در بالا ایجاد کردید ، بر روی FORM دوبار کلیک کنید و برنامه زیر را در آن بنویسید :

```
Private Sub Form_Load()
```

```
MSComm1.CommPort = 1 ' Set the port number
```

```
MSComm1.Settings = "9600,N,8,1" ' Set UART parameters
```

```
MSComm1.PortOpen = True ' Required, might lock port
```

```
Dim s As String * 2
```

```
Dim x As String * 2
```

```
End Sub
```

```
Private Sub Command1_Click()
```

```
x = "fd"
```

```
MSComm1.Output = x
```

```
Text1.Text = x
```

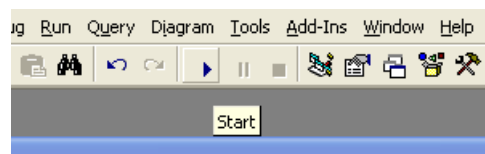
```
s = MSComm1.Input
```

```
Text2.Text = s
```

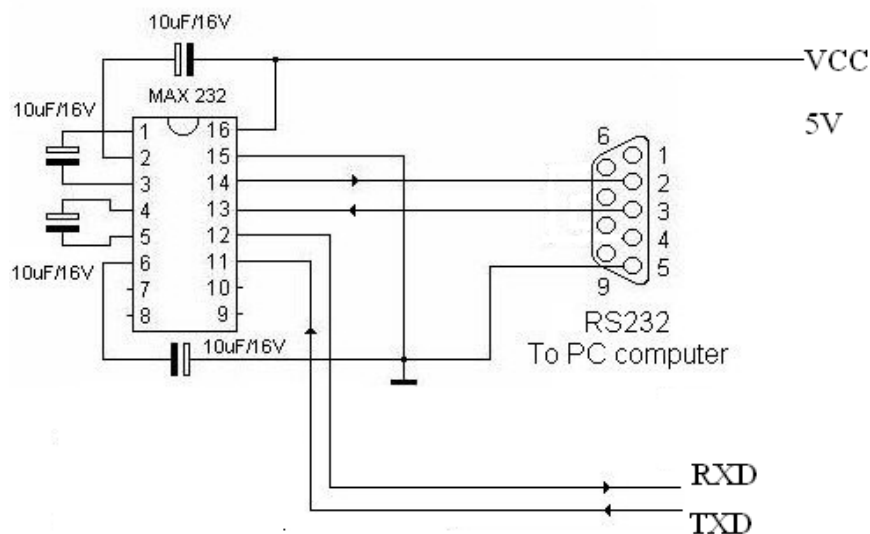
```
End Sub
```

در این برنامه کارکتر " FD " به پورت سریال ارسال میشود ، همچنین در این برنامه مقدار موجود بر روی پورت در متغیر S ریخته میشود .

برای شبیه سازی برنامه از گزینه منو ابزار گزینه ی START را انتخاب کنید :



مدار مورد نیاز برای راه اندازی پورت COM :



در مدار همه چیز واضح است . شما با متصل کردن پایه های TXD و RXD به هم میتوانید از صحت برنامه نوشته برای PC و مدار بالا مطمئن شوید . برنامه بالا را اجرا کنید ، پایه های 2 و 3 پورت COM را به هم متصل نمایید بر روی BUTTON دوبار کلیک کنید ، ببینید که کارکتر ارسال شده در TEXT 2 نمایش داده میشود .

مثال دیگر :

```
Private Sub Form_Load()
```

```

MSComm1.CommPort = 1 ' Set the port number

MSComm1.Settings = "9600,N,8,1" ' Set UART parameters

MSComm1.PortOpen = True ' Required, might lock port

Dim s As String * 2

Dim x As String * 2

End Sub

Private Sub Command1_Click()

x = Text1.Text

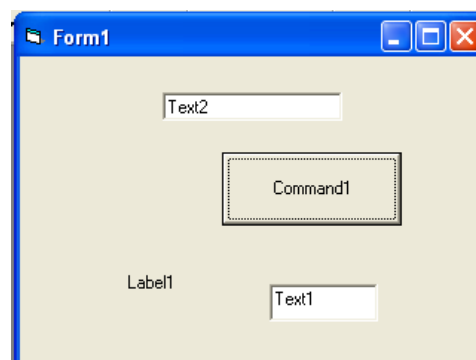
MSComm1.Output = x

s = MSComm1.Input

Text2.Text = s

End Sub

```

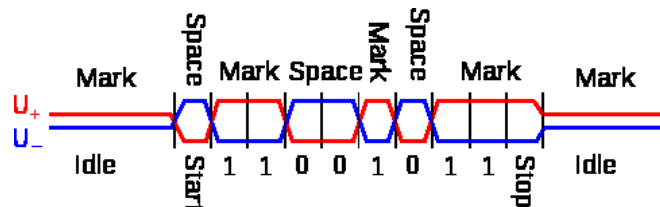


در مثال بالا از COM1 استفاده شده است ، نرخ انتقال داده 9600 است ، هنگامی شما کارتری را در TEXT1 وارد میکنید ، با فشار دادن کلید ، مقدار آن در داخل متغیر X ریخته میشود ، و متغیر X به بیرون فرستاده میشود در صورتی که شما پایه های TXD و RXD را به هم متصل کنید ، مقدار خارج شده دوباره به PC برمیگردد و با کلیک دوباره ، بر وی TEXT2 نمایش داده میشود .

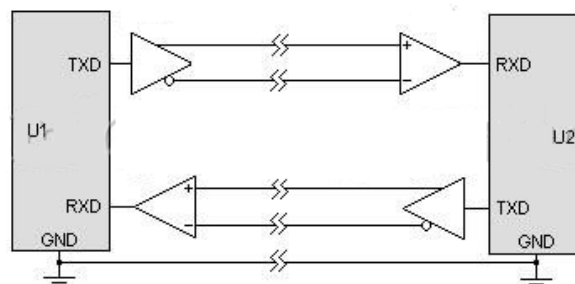
در ضمن خطای قابل قبول در ارتباط سریال دو دهم درصد میباشد ، در صورتی که خطای بیشتر شود ، کارکتر دریافتی نمایش داده نمیشود و ارتباط تا کم شدن خطا متوقف می شود.

پروتکل RS485:

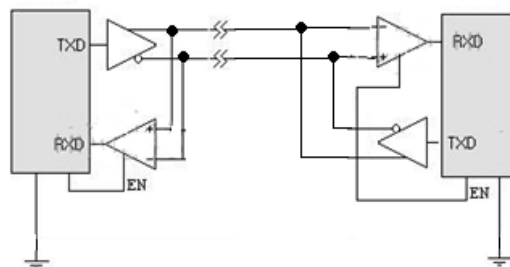
RS485 یک پروتکل سریال برای انتقال داده می باشد، این پروتکل که ارتفاع یافته ی RS232 است بیشتر در مقاصد صنعتی استفاده شده و تشابه زیادی با RS232 دارد. در پروتکل RS485 از خطوط دیفرانسیلی استفاده می شود. در حالت دیفرانسیلی، داده ی موجود از طریق دو خط ارسال می شود و در صورتی که نویزی در محیط وجود داشته باشد، بر روی داده موجود در هر دو خط تاثیر گذاشته و در عمل نمیتواند آن را تغییر دهد:



در این حالت ما به 4 سیم برای انتقال داده نیاز خواهیم داشت، دو سیم برای ارسال و دو سیم برای دریافت (پروتکل RS 433):



با حذف کردن دو سیم و سوییچ کردن ارسال و دریافت بر روی دو سیم دیگر پروتکل RS485 بوجود میاید.



در RS485 به دلیل استفاده از خطوط دیفرانسیلی میتوان فاصله ی دو وسیله را تا 1200 متر افزایش داد، در این حالت حداکثر سرعت انتقال داده برابر با 100 کیلو بیت بر ثانیه است (در فاصله های کم (مثلا 10 متر) سرعت تا 3.5 مگابیت بر ثانیه افزایش می باید.

در کامپایلر بسکام با استفاده از دستورات زیر میتوان پورت RS232 میکرو را در مد RS485 پیکربندی کرد:

Config Print0 = pin , Mode = mode

در دستور بالا pin نام پایه ای است که به پایه ی سویچ (فعال ساز ارسال/دریافت) چیپ مبدل RS485 متصل میشود، همچنین mode میتواند با توجه به نوع مبدل انتخاب شده، set یا reset باشد. (مثلا اگر قطعه با سطح منطقی یک فعال میشود، باید از set استفاده نمایید). مثال:

```
$regfile = "m48def.dat"           ' we use the M48

$crystal = 8000000

$baud = 19200

Config Print0 = Portb.0 , Mode = Set

Config Pinb.0 = Output             'set the direction yourself

Dim Resp As String * 10

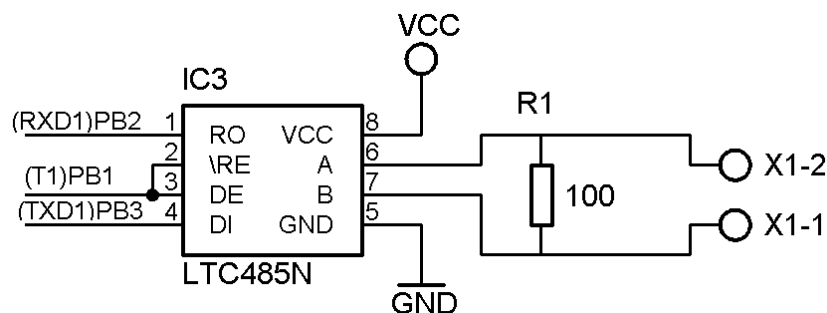
Do

    Print "test message"

    Input Resp                     ' get response

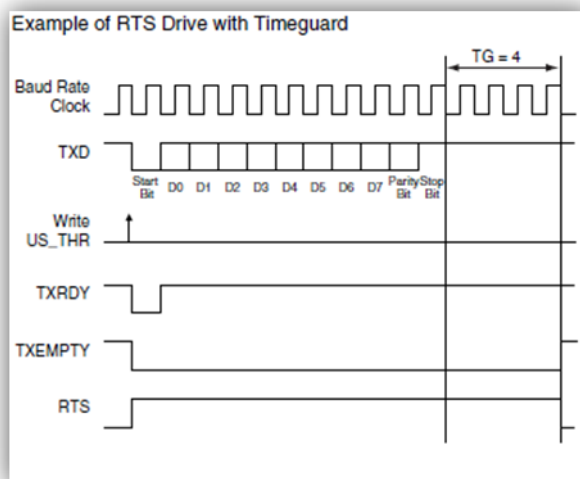
Loop
```

در برنامه ی بالا از پایه های txd و rxt برای انتقال داده و از پایه ی b.0 برای کنترلر کردن سویچ استفاده شده است، مدار مورد استفاده برای برنامه بالا را در زیر مشاهده میکنید:

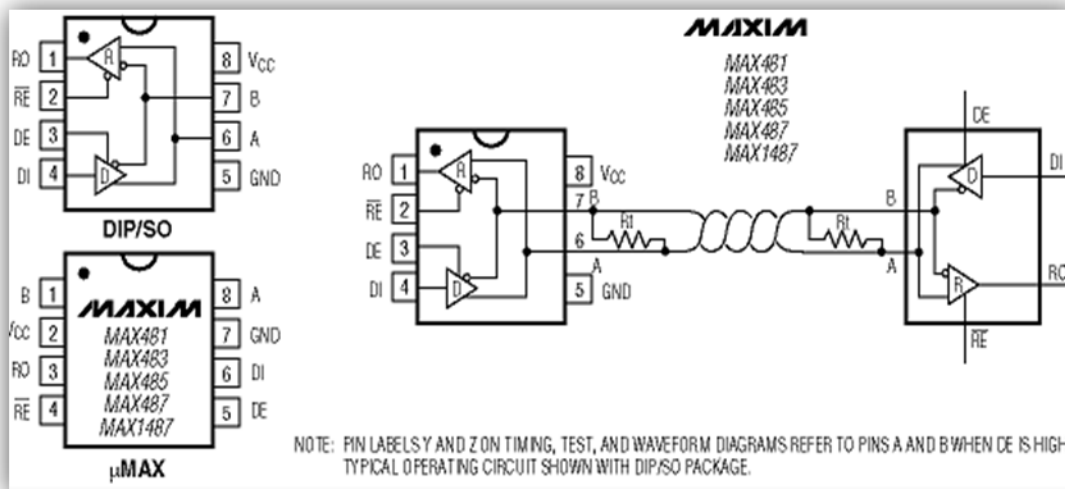


یکی از کاربردهای عمده ی پروتکل RS485 اجرای پروتکل MODBUS است، که در ادامه و در بخش شبکه های میکروکنترلی به بررسی مفصل آن خواهیم پرداخت.

در پروتکل RS485 نیز مانند پروتکل RS232، بیت اول مربوط به شروع کار، 8 بیت بعدی داده ی موجود روی باس و بیت آخر، بیت تشخیص خطا میباشد، تنها تفاوت عمده ی میان این دو پروتکل نحوه ی انتقال داده است:



برای درایو کردن پورت RS485 قطعات مختلفی ارائه شده است که در اینجا میتوانیم به تراشه های MAX481, MAX483, MAX485, MAX487, MAX1487 اشاره کنیم. (توجه داشته باشید که فقط xx485 مخصوص کار با این پورت است و سایر قطعات برای پروتکل مشابه طراحی شده اند و ممکن است در این باس به درستی کار نکنند)



ارتباط سریال SPI :

Serial Peripheral Interface یا spi یک رابط سریال همزمان میباشد که توسط آن میتوانید انواع لوازم جانبی نظیر حافظه های eeprom ، mmc/sd ، پر سرعت ، مبدل های دیجیتال به آنالوگ و آنالوگ به دیجیتال و... را به میکرو کنترلر متصل کنید . از این باس برای ایجاد شبکه های میکرو کنترلری نیز استفاده میشود .

میکرو کنترلر های AVR معمولاً دارای 1 تا 4 باس spi با مشخصات زیر هستند :

✓ ارتباط سریال سنکرون با سرعت بالا

✓ از این ارتباط میتوان برای اتصال میکرو های avr به یکدیگر یا اتصال میکرو به هر وسیله ای که این ارتباط را پشتیبانی میکند استفاده کرد

✓ ارسال و دریافت داده هم زمان و قابلیت تنظیم سرعت انتقال دیتا

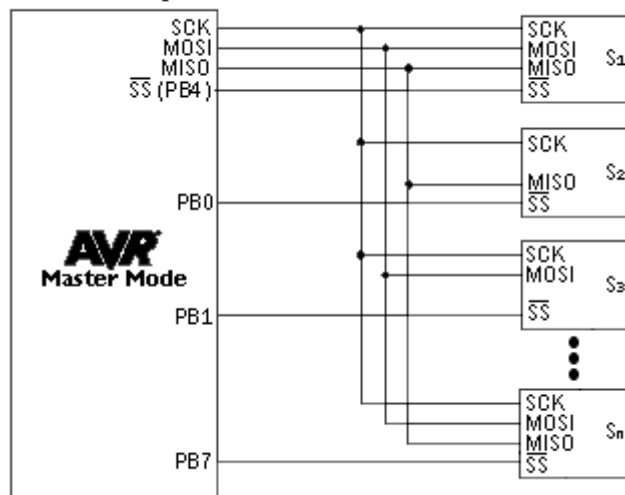
✓ استفاده از 4 سیم برای انتقال اطلاعات

✓ دارای منبع وقفه پایان ارسال

✓ ارتباط به صورت های مستر یا اسلیو

در تصویر زیر نحوه ی اتصال چند وسیله از طریق باس spi آورده شده است ، همانطور که قبلاً نیز گفته شد در این باس خطوط mosi و miso و sck بین تمامی قطعات مشترک هستند و این خطوط CS هستند که تعیین میکنند کدام میکرو اطلاعات را از مستر بگیرد یا برای آن ارسال کند :

Multi Slave system



در این باس میکرو کنترلر master میتواند به صورت همزمان به چند وسیله متصل شده و اطلاعاتی را برای آنها ارسال می نماید، این کار با یک کردن پایه های cs (chip select) دستگاه های مذکور انجام می شود. برای خواندن اطلاعات خروجی یک قطعه، master باید پایه ی cs آن را یک کرده و اطلاعات موجود در بافر خروجی آن را بخواند، در هنگام خواندن اطلاعات master می تواند فقط به یک وسیله متصل شود.

در این ارتباط از چهار پایه mosi و miso و sck و ss استفاده میشود که در میکرو مگا 16 به ترتیب پایه 5 تا 8 میباشد (از portb.4 تا portb.7)، در صورتی که نیاز دارید میکرو کنترلر های بیشتری را به هم متصل کنید، باید تعدادی پایه ی دلخواه را در حالت خروجی پیکربندی کرده و به عنوان پایه ی CS انتخاب نمایید، پایه ی CS بسته به نوع قطعه ی جانبی ممکن است در حالت صفر یا یک ارتباط را برقرار کند.

در زیر نحوه پیکربندی spi در بسکام آمده است:

▪ دستور پیکره بندی SPI در محیط BASCOM: (برای میکرو کنترلر های سری تاینی و اتمگا و AT90S).

CONFIG SPI = HARD, INTERRUPT=ON/OFF, DATA ORDER = LSB/MSB, MASTER =

YES/NO, POLARITY=HIGH/LOW, PHASE=0/1, CLOCK RATE=4/16/64/128, NOSS=0/1

INTERRUPT=ON/OFF: در صورت استفاده از وقفه در ارتباط سریال از گزینه ON استفاده میشود، در این حالت تنها زمانی

که داده ای میخواهد منتقل شود میکرو کار میکند.

DATA ORDER = LSB/MSB: در صورت انتخاب LSB، ابتدا LSB (بیت کم ارزش) و سپس MSB (بیت پر ارزش) داده ارسال

می شود و بالعکس.

MASTER = YES/NO: این گزینه مشخص میکند میکرو master (فرمانده) است یا slave (فرمانبردار) که گزینه yes مشخص

کننده میکرو مستر و گزینه no مشخص کننده میکرو اسلیو است.

POLARITY=HIGH/LOW: این گزینه وضعیت پایه کلاک را در زمان بیکاری میکرو مشخص میکند که میتواند صفر (low) یا

یک (high) باشد

CLOCK RATE=4/16/64/128: مشخص کننده فرکانس کلاک SPI است.

NOSS=0/1: زمانی که در حالت MASTER نمی خواهید سیگنال SS/ ایجاد شود، یک انتخاب می شود و در این حالت کاربر به صورت نرم افزاری باید پایه SLAVE مورد نظر را پایین نگه دارد.

در حالت بالا از پایه های پیشفرض برای انتقال دیتا استفاده میشود ، در صورتی که میخواهید آنها را به پایه های دیگر تغییر دهید باید از دستور زیر استفاده کنید:

CONFIG SPI=SOFT, DIN=PIN, DOUT = PIN , SS = PIN|NONE, CLOCK = PIN

DIN=PIN: نشانگر پایه MISO است و پین نام پایه دل خواه میباشد.

DOUT = PIN: نشانگر پایه MOSI است و پین نام پایه دل خواه میباشد.

SS = PIN|NONE: نشانگر پایه SS است و پین نام پایه دلخواه میباشد. (در صورتی از گزینه NONE استفاده کنید پایه تغییر نمیکند

CLOCK = PIN: نشانگر پایه SCK است و پین نام پایه دلخواه میباشد. برای مثال :

CONFIG SPI=SOFT, DIN=PIND.0, DOUT = PIND.1 , SS = PIND.2, CLOCK = PIND.3

دیگر دستورات مربوط به SPI :

دستور SPIINIT :

SPIINIT

توسط این دستور پایه های که برای SPI (mosi و miso و sck و ss (در میکرو مگا 16 به ترتیب پایه 5 تا 8 میباشد (از portb.4 تا portb.7)) استفاده میشوند ، برای این مد فعال میگردند و دیگر نمیتوان از آنها به عنوان ورودی یا خروجی استفاده کرد.

دستور SPIIN :

SPIIN VAR,BAYT

توسط این دستور به تعداد BAYT از درگاه SPI اطلاعات دریافت میشود و در متغیر VAR قرار میگیرد ، در صورتی که متغیر شما از جنس WORD یا دیگر متغیر ها است ، شما باید تعداد بایت متغیر را به جای BAYT بنویسید ، مثال :

'MASTER

\$regfile = "m16def.dat"

\$crystal = 8000000

Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , Rs = Pind.0 , E = Pind.1

Config Lcd = 16 * 2

Dim A As Byte

Config Spi = Hard , Interrupt = On , Data Order = Lsb , Master = Yes , Polarity = High , Phase = 0 , Clockrate = 128

```

Spiinit
Do
Spiin A , 1
Locate 1 , 1
Lcd A
Loop
End

```

دستور SPIOUT :

SPIOUT VAR , BAYT

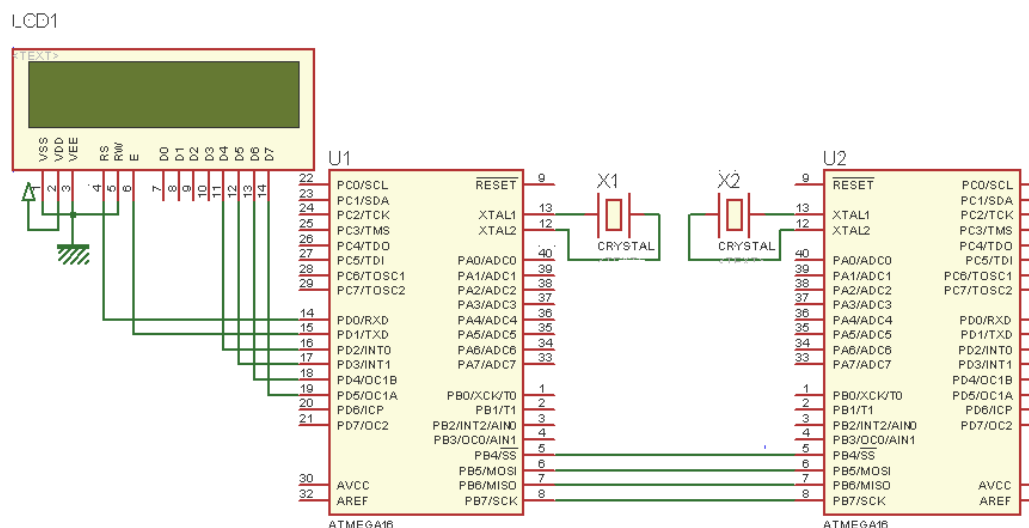
توسط این دستور به تعداد BAYT، داده VAR به درگاه SPI اطلاعات ارسال میشود ، در صورتی که متغیر شما از جنس WORD یا دیگر متغیرها است ، شما باید تعداد بایت متغیر را به جای BAYT بنویسید ، مثال :

```

'SLAVE
$regfile = "m16def.dat"
$crystal = 8000000
Dim A As Word
Config Spi = Hard , Interrupt = On , Data Order = Lsb , Master = No , Polarity = High , Phase = 0 , Clockrate = 128
Spiinit
Do
Incr A
Waitms 300
Spiout A , 1
Loop
End

```

هر دو مثال بالا مربوط به ارتباط SPI بین دو میکرو مگا 16 است که مدار آن را در زیر مشاهده میفرمایید:



دستور SPIMOVE :

VAR=SPIMOVE(BAYT)

از این دستور در زمان ارتباط دوطرفه استفاده میشود ، توسط این دستور متغیر BAYT به باس SPI ارسال شده و همزمان داده

دریافت شده از باس در متغیر VAR قرار میگیرد . مثال:

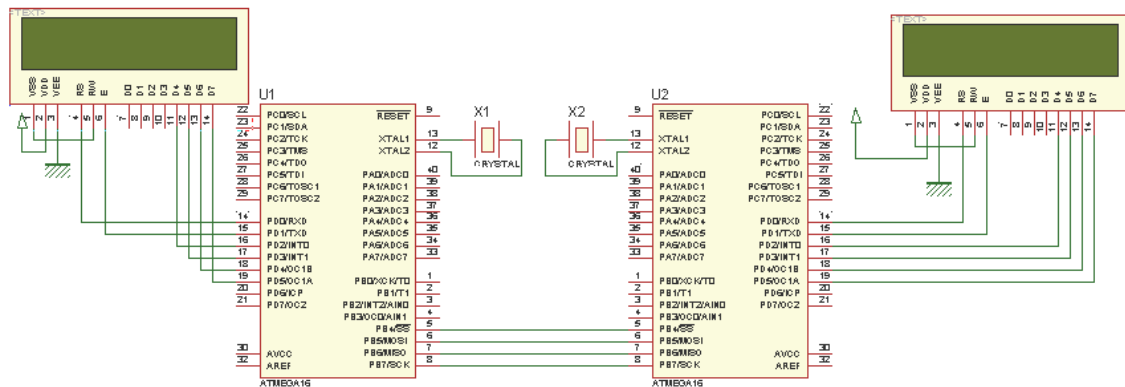
برنامه SLAVE :

```
$regfile = "m16def.dat": $crystal = 8000000
Dim A As Byte,B As Byte
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , Rs = Pind.0 , E = Pind.1
Config Lcd = 16 * 2
Config Spi = Hard , Interrupt = On , Data Order = Lsb , Master = No , Polarity = High , Phase = 0 , Clockrate = 128
Spiinit
Do
B = Spimove(a)
A = A + 2
Locate 1 , 1 :Lcd "Spiout: "; A
Locate 2 , 1 :Lcd "SPIIN: "; B
Waitms 300
Loop
End
```

برنامه MASTER :

```
$regfile = "m16def.dat": $crystal = 8000000
Dim A As Byte,B As Byte
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , Rs = Pind.0 , E = Pind.1
Config Lcd = 16 * 2
Config Spi = Hard , Interrupt = On , Data Order = Lsb , Master = Yes , Polarity = High , Phase = 0 , Clockrate = 128
Spiinit
Do
B = Spimove(a)
Incr A
Locate 1 , 1 :Lcd "Spiout: "; A
Locate 2 , 1 :Lcd "SPIIN: "; B
Waitms 300
Loop
End
```


مدار مورد استفاده:



مثال:

در این مثال پایه های مربوط به ارتباط SPI تغییر کرده اند ، شما میتوانید آنها را به حالت پیش فرض برگردانید:

برنامه SLAVE :

```

$regfile = "m16def.dat":$crystal = 8000000
Dim A As Byte,B As Byte
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , Rs = Pind.0 , E = Pind.1
Config Lcd = 16 * 2
Config Spi = Soft , Din = Pina.0 , Dout = Pina.1 , Ss = Pina.2 , Clock = Pina.3
Spiinit
Do
  B = Spimove(a)
  A = A + 2
  Locate 1 , 1: Lcd "Spiout:" , A
  Locate 2 , 1: Lcd "SPIIN:" , B
  Waitms 300
Loop
End

```

برنامه MASTER :

```

$regfile = "m16def.dat":$crystal = 8000000
Dim A As Byte,B As Byte
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , Rs = Pind.0 , E = Pind.1
Config Lcd = 16 * 2
Config Spi = Soft , Din = Pina.0 , Dout = Pina.1 , Ss = Pina.2 , Clock = Pina.3
Spiinit
Do

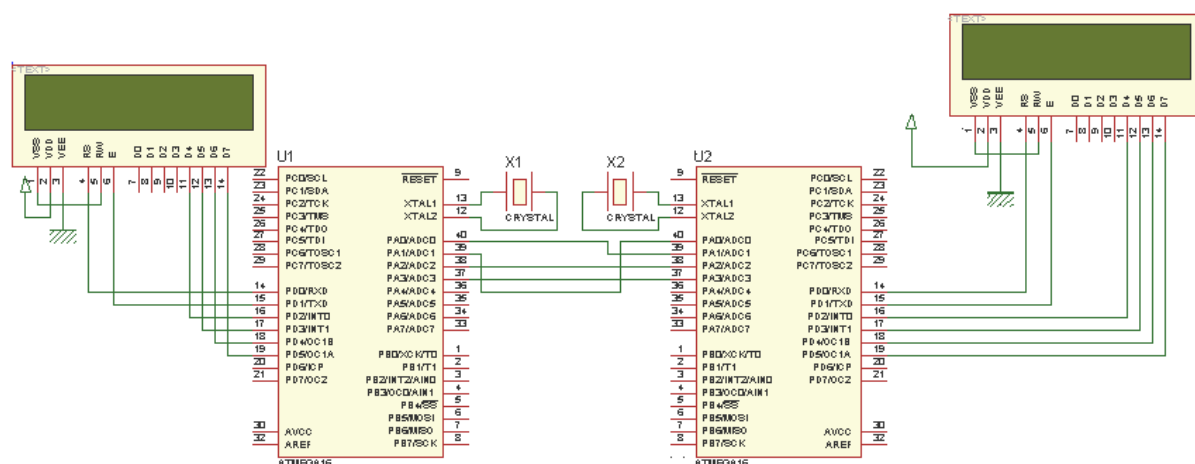
```

```

B = Spimove(a)
Incr A
Locate 1, 1: Lcd "Spiout:"; A
Locate 2, 1: Lcd "SPIIN:"; B
Waitms 300
Loop
End

```

مدار مورد استفاده:



■ دستوری که بندی SPI در محیط BASCOM: (برای میکرو کنترلر های سری ATXMEGA):

CONFIG SPIx = HARD, MASTER = YES|NO, MODE=0-3, CLOCKDIV=div, DATA_ORDER = LSB|MSB

SPIx: میکرو کنترلر های سری ATXMEGA معمولاً دارای 1 تا 4 باس SPI میباشند که شما باید عملیات پیکربندی هر باس

را به صورت مجزا انجام دهید (در صورتی که قصد دارید از بیش از یک باس استفاده کنید) برای اینکار باید به جای

SPIX نام پاس مورد نظر خود که یکی از موارد SPIE, SPID, SPIC یا SPIF میباشد را قرار دهید. همانطور که متوجه

شدید هر باس SPI بر روی پورت متناظر خود قرار دارد، برای مثال میتوانید پایه های باس SPIE را بر روی پورت E

مشاهده کنید.

MASTER = YES|NO: با انتخاب YES میکرو در حالت MASTER و با انتخاب NO میکرو در حالت SLAVE پیکربندی میشود.

MODE=0-3: شماره های 0 تا 3 مدل کاری باس SPI را مطابق زیر تعیین میکنند:

Mode	Leading	Edge	Trailing	Edge
0	Rising	Sample	Falling	Setup

1	Rising	Setup	Falling	Sample
2	Falling	Sample	Rising	Setup
3	Falling	Setup	Rising	Sample

در جدول بالا گزینه ی Falling (پایین رونده) و Rising (بالا رونده) مربوط به پالس کلاک موجود بر روی پایه ی SCK می باشد، با این تنظیمات شما میتوانید مشخص کنید که اولین پالس کلاک از چه سطحی شروع شود (به ترتیب یک به صفر یا صفر به یک).

همچنین گزینه های موجود در بخش Edge (Sample و Setup) به کاربر این اجازه را میدهد تا با انتخاب Sample داده ی ارسالی را یک بار چک کرده و از صحت آن اطمینان یابد، انتخاب گزینه ی Setup باعث ارسال داده به صورت پیش فرض خواهد شد.

CLOCKDIV=div : به جای div باید یکی از دستورات CLK2, CLK4, CLK8, CLK16, CLK32, CLK64 and CLK128 قرار گیرد.

در میکرو کنترلر های سری Atxmega کلاک مورد نیاز باس spi از تقسیم شدن کلاک اصلی سیستم به مقادیر بالا حاصل میشود. توجه داشته باشید که در حالت slave فقط دستورات CLK2, CLK4 معتبر هستند.

DATA_ORDER = LSB|MSB : در صورت انتخاب LSB، ابتدا LSB (بیت کم ارزش) و سپس MSB (بیت پر ارزش) داده ارسال می شود و بالعکس.

ارسال داده از طریق باس spi در میکرو کنترلر های سری atxmega با استفاده از دستورات زیر انجام میشود:

```
OPEN "device" for MODE As #channel
```

با مجموعه دستورات بالا ابتدا باید باس مورد استفاده (پیکربندی شده) را باز کنیم، در این بخش به جای device نام باس spi پیکربندی شده (SPIC, SPID, SPIE یا SPIF) و به جای MODE نحوه ی ارسال داده (BINARY یا RANDOM) و به جای channel یک عدد دسیمال درج میشود، عدد دسیمال مذکور شماره ی کانال بوده و کاربر باید با دستورات زیر داده را به آن ارسال کند:

```
PRINT [#channel , ] var ; " constant"
```

در دستور بالا channel ، عدد دسیمال درج شده برای باس مورد نظر است ، همچنین var یک عدد ثابت یا یک متغیر است که به باس ارسال میشود ، در این بخش کاربر میتواند در صورت نیاز رشته ی عددی constant نیز به فرمت رشته به باس ارسال کند .

INPUT #ch, var[, varn]

دریافت داده از طریق باس spi در میکرو کنترلر های سری atxmega با یکی از دستور بالا انجام می شود ، در این دستور نیز ch شماره ی کانال مورد نظر و var و varn یک متغیر است که داده ی دریافت شده از باس در آن ذخیره میشود ، در صورتی که داده ی ارسالی از میکرو کنترلر دیگر فقط یک متغیر عددی باشد ، نیازی به استفاده از متغیر varn نیست .

مثال :

\$regfile = "xm128a1def.dat" : \$crystal = 32000000

\$hwstack = 32 ' default use 32 for the hardware stack

\$swstack = 32 'default use 10 for the SW stack

\$framesize = 32 'default use 40 for the frame space

'First Enable The Osc Of Your Choice

Config Osc = Enabled , 32mhzosc = Enabled

'configure the systemclock

Config Sysclock = 32mhz , Prescalea = 1 , Prescalebc = 1_1

Dim Bspivar As Byte , Ar(4) As Byte , W As Word

Bspivar = 1

Config Spic = Hard , Master = Yes , Mode = 0 , Clockdiv = Clk2 , Data_order = Msb

Config Spid = Hard , Master = Yes , Mode = 1 , Clockdiv = Clk8 , Data_order = Lsb

Config Spie = Hard , Master = Yes , Mode = 2 , Clockdiv = Clk4 , Data_order = Msb

Config Spif = Hard , Master = Yes , Mode = 3 , Clockdiv = Clk32 , Data_order = Msb

Open "SPIC" For Binary As #10

Open "SPID" For Binary As #11

```
Open "SPI" For Binary As #12
```

```
Open "SPI" For Binary As #bspivar
```

```
'SPI channel only support PRINT and INPUT
```

```
Print #10, "to spi"; W
```

```
Input #11, Ar(1), W
```

```
Print #bspivar, W
```

```
Input #bspivar, W
```

```
End
```

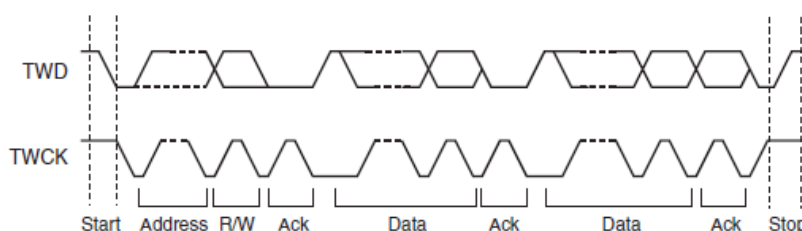
ارتباط سریال I2C یا 2-wire

یکی از پروتکل های پر کاربرد در میکرو کنترلر ها پروتکل یا باس I2C میباشد، توسط این باس میتوانيد تمامی قطعات I2C را، فقط با استفاده از دو سیم به میکرو کنترلر متصل کنید، در این حالت میکرو کنترلر در مد مستر و سایر قطعات متصل شده به باس در مد اسلیو پیکربندی میشوند.

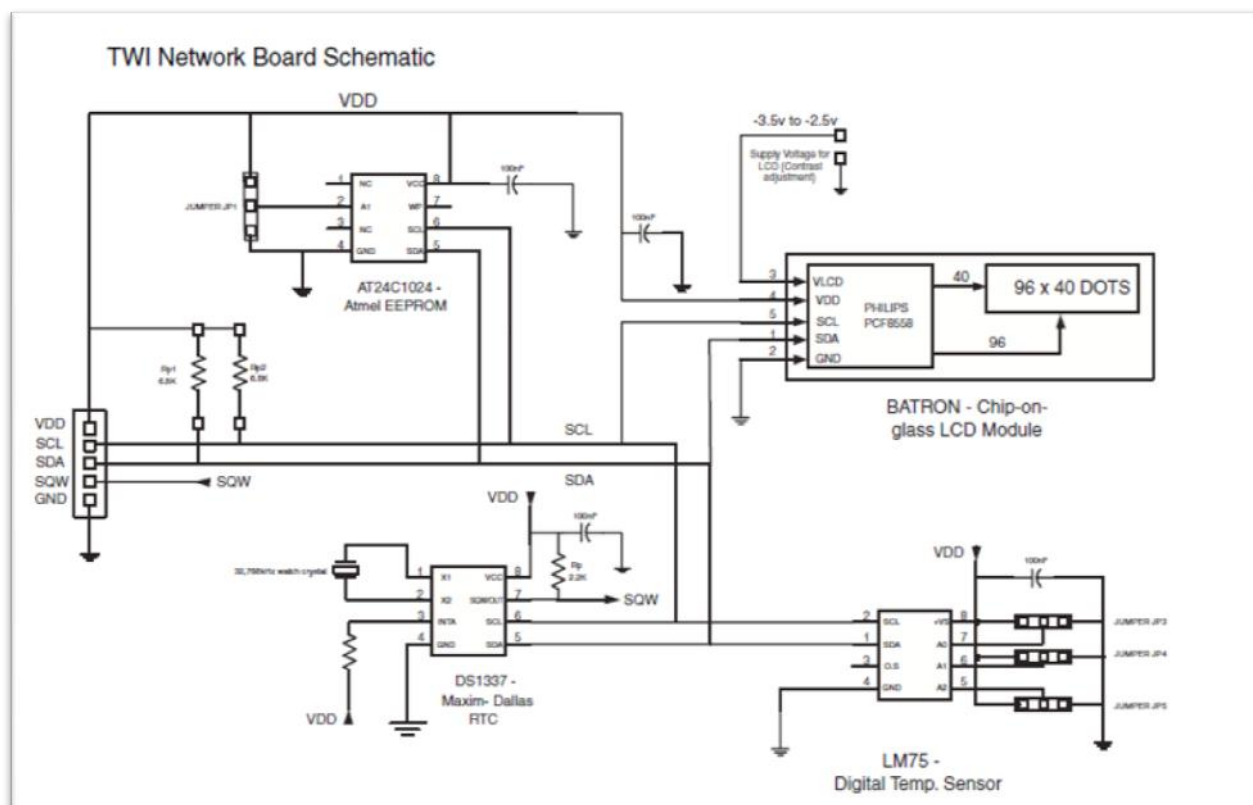
در پروتکل I2C به هر وسیله ی جانبی که باس متصل می شود یک آدرس تعلق میگیرد، معمولاً آدرس توسط سه پایه ی موجود بر روی قطعه که با نام های a0 تا a2 مشخص میشوند، معین می شود. (البته بعضی از قطعات دارای آدرس داخلی هستند، که در دیتاشیت آنها مشخص گردیده است).

آدرس قطعه میتواند از صفر تا 255 دسیمال یا 0 تا ff هگز باشد. هنگامی که میکرو کنترلر قصد دارد تا با یکی از دستگاه های جانبی ارتباط برقرار کند، ابتدا پایه TWD (data) را به یک منطقی میرد و بعد از گذشت یک پالس زمانی آدرس دستگاه جانبی مورد نظر را بر روی باس میفرستد، کلیه دستگاه های جانبی آدرس را از روی باس میخوانند و فقط دستگاهی که آدرسش با آدرس موجود بر روی خط یکی است، داده های بعدی را دریافت میکند.

Transfer Format



به این ترتیب برای ارتباط با دستگاه های جانبی به دو خط TWD (انتقال داده) و TWCK (خط کلاک) نیاز خواهد بود. از خط اول برای ارسال پالس همزمانی (clock) و از خط دوم برای انتقال داده استفاده میشود، همچنین گراند کلیه لوازم متصل شده به باس باید مشترک باشد.



تصویر بالا مربوط به اتصال چهار دستگاه جانبی به باس i2c است. همانطور که مشاهده میکنید کلیه خطوط داده و کلاک دستگاه های موجود به یکدیگر متصل شده و بعد از pullup شدن توسط مقاومت های 10 یا 2.2 کیلو به میکرو کنترلر (ترمینال) متصل شده اند.

میکرو کنترلر های خانواده ی avr دارای 1 تا 4 باس i2c میباشند ویژگی های این پروتکل در این خانواده به شرح زیر است:

1- در این پروتکل از دوسیم همراه با گراند و VCC، که درمجموع چهار سیم میشود، برای انتقال دیتا استفاده میشود.

2- بالا ترین فرکانس کلاک 400 کیلو هرتز است.

3- در این ارتباط میتوان تعداد نامحدود وسیله جانبی با آدرس سخت افزاری مختلف را به هم متصل کرد .

4- حداکثر طول کابل ارتباطی باسیم شیلد 80 سانتی متر است .

5- کلاک ارتباط I2C به شدت به کلاک سیستم (فرکانس کریستال) وابسته است .

برای ارتباط I2C از دو پایه SCL و SDA (PORTC.0 و PORTC.1 میکرو مگا 16) استفاده میشود که پایه SDA پایه داده و پایه

SCL پایه کلاک میباشد. (دو پایه مذکور پایه های پیش فرض میباشد ، شما میتوانید با دستوراتی که گفته میشود آنها را به

پایه های دلخواه خود تغییر دهید)

دستورات مربوط به راه اندازی I2C در محیط بسکام:

▪ تعیین کلاک I2C :

Config I2cdelay = X

X میتواند از 1 تا 255 باشد ، رابطه ای بین کلاک و عدد وجود ندارد ، مثلاً برای عدد 10 کلاک 100 کیلو و برای عدد 5

کلاک 200 کیلو و برای عدد 1 کلاک 400 کیلو هرتز است (کلاک I2C به فرکانس کریستال وابسته است ، در این ارتباط

باید کریستال نوشته شده در برنامه با کریستال استفاده شده یکی باشد ، همچنین کلاک هر دو دستگاه باید مساوی باشد)

▪ تعیین پایه های داده و کلاک I2C :

با دستور زیر پایه SCL (پایه کلاک) تعیین میشود :

CONFIG SCL = pin

Pin نام یکی از پایه های دلخواه میکرو میباشد .

با دستور زیر پایه SDA (پایه داده) تعیین میشود :

CONFIG SDA = pin

Pin نام یکی از پایه های دلخواه میکرو میباشد

بعد از اینکه I2C پیکر بندی شد با استفاده از دستور زیر میتوان ارتباط را آغاز کرد:

I2CSTART

با این دستور ارسال و دریافت داده شروع میشود شما همچنین میتوانید با دستور زیر به ارسال و دریافت داده خاتمه دهید:

I2CSTOP

با استفاده از دستور زیر میتوان داده ای را به باس I2C فرستاد:

I2CSEND slave, var

I2CSEND slave, var , bytes

Slave: آدرس گیرنده اطلاعات است که میتواند به فرم یک عدد ثابت یا متغیر باشد

Var: عدد ثابت یا متغیری است که میخواهیم آن را ارسال کنیم

bytes: با این گزینه شما میتوانید تعداد بایت دلخواه را به باس ارسال کنید (این گزینه اختیاری است)

فرم خلاصه شده این دستور به شکل زیر است:

I2CWRITE val

Val: عدد ثابت یا متغیری است که کد دریافت شده در آن قرار میگیرد. مثال:

برنامه فرستنده:

\$regfile = "m32def.dat"

\$crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Pinb.2 , Db5 = Pinb.3 , Db6 = Pinb.4 , Db7 = Pinb.5 , Rs = Pinb.0 , E = Pinb.1

Config Kbd = Porta

Config I2cdelay = 5

Config Sda = Portc.1

Config Scl = Portc.0

Dim A As Byte

I2cstart

W:

A = Getkbd()

If A > 15 Then

Goto W

End If

I2csend &H40 , A

Locate 2 , 1:Lcd A

Goto W

End

در مثال با کلاک I2C ، 200 کیلو هرتز انتخاب شده است همچنین پایه SCL (پایه کلاک) به پورت c.0 (پایه 22 مگا 16) و پایه SDA (پایه داده) به پین c.1 (پایه 23 مگا 16) متصل شده است. با دستور I2cstart پروتکل I2C شروع به کار کرده و عدد گرفته شده از کپد را بخ باس I2C میفرستد ، برای گیرنده آدرس &h40 در نظر گرفته شده است ، برای درک بیشتر موضوع متغیر ارسالی بر روی یک lcd به نمایش در میاید.

با استفاده از دستور زیر میتوان داده ای را از باس I2C دریافت کرد:

I2CRECEIVE slave, var

I2CRECEIVE slave, var , b2W, b2R

Slave: آدرس فرستنده اطلاعات است که میتواند به فرم یک عدد ثابت یا متغیر باشد.

Var: عدد ثابت یا متغیری است که کد دریافت شده در آن قرار میگیرد

bytes: با این گزینه شما میتوانید تعداد بایت دلخواه را از باس دریافت کنید (این گزینه اختیاری است). (توجه داشته باشید که

تعداد بایت دریافتی و ارسالی باید با هم برابر باشند در غیر اینصورت اطلاعات دریافتی ناقص خواهد بود).

فرم خلاصه شده این دستور به شکل زیر است:

I2CRBYTE var, ack/nack

Var: عدد ثابت یا متغیری است که کد دریافت شده در آن قرار میگیرد

ack/nack: زمانی که بخواهیم بیشتر از یک بایت را از باس بخوانیم باید از ack استفاده کنیم و زمانی که میخواهیم آخرین

بایت را از باس بخوانیم از nack استفاده میکنیم . مثال:

این برنامه مربوط به گیرنده مداری است که برنامه آن را در بالا مشاهده فرمودید:

```
$regfile = "m32def.dat" : $crystal = 1000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3 , Rs = Pina.4 , E = Pina.5
```

```
Config I2cdelay = 5
```

```
Config Sda = Portc.1
```

```
Config Scl = Portc.0
```

```
Dim A As Byte
```

```
I2cstart
```

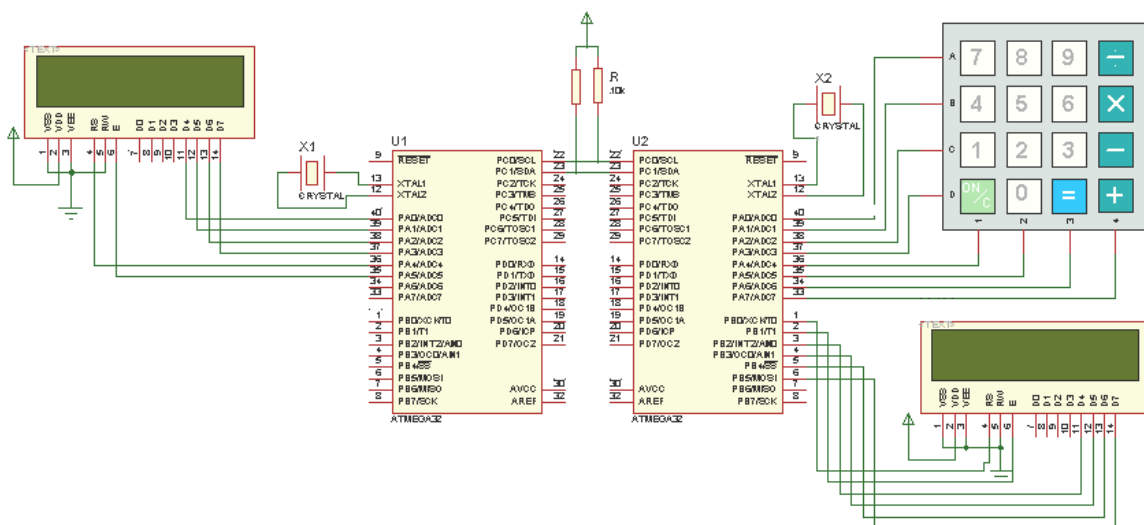
```
W:
```

```
I2creceive &H40 , A
```

```
If A < 16 Then : Locate 1 , 1 : Lcd A : End If
```

```
Goto W
```

```
End
```



در مثال بالا کلاک I2C ، 200 کیلو هرتز (Config I2cdelay = 5) انتخاب شده است همچنین پایه SCL (پایه کلاک) به پورت

c.0 (پایه 22 مگا) و پایه SDA (پایه داده) به پین c.1 (پایه 23 مگا) متصل شده است. با دستور I2cstart پروتکل I2C شروع

به کار کرده و عدد گرفته شده از باس را بر روی lcd نمایش میدهد ، برای فرستاده آدرس &h40 در نظر گرفته شده است .

در قسمت های قبل گفته شده بود که ایسی به شماره ds1307 وجود دارد که میتواند زمان را دقیق بشمارد. ویژگی های این ایسی به شرح زیر است:

- 1- ای سی DS1307 یک RTC می باشد rtc مخفف (Real Time Counter) (شمارش گر زمان واقعی) است.
- 2- این ای سی با یک باتری بک آپ 3 ولتی می تواند تا 10 سال زمان را در خود بشمارد. (در صورتی که باتری دوام آورد ، منظور کم مصرفی ایسی است).
- 3- نمایش ساعت (شامل ثانیه - دقیقه و ساعت) به صورت 24 ساعت.
- 4- نمایش تاریخ (شامل روز-ماه - سال) به صورت میلادی
- 5- شمارش روز هفته (شنبه - 1 شنبه تا جمعه)
- 6- شمارش روز های طی شده از اول سال (به صورت میلادی)
- 7- دارای پایه جدا برای اتصال باتری بک آپ
- 8- دارای دو پایه برای اتصال نوسان ساز مستقل
- 9- عملکرد کاملاً ثابت در تمامی محیط ها
- 10- قیمت و پشتیبانی خوب
- 11-

شما میتوانید با مراجعه با دیتاشیت این قطعه اطلاعات بیشتری را در مورد آن بدست آورید .

مثال :

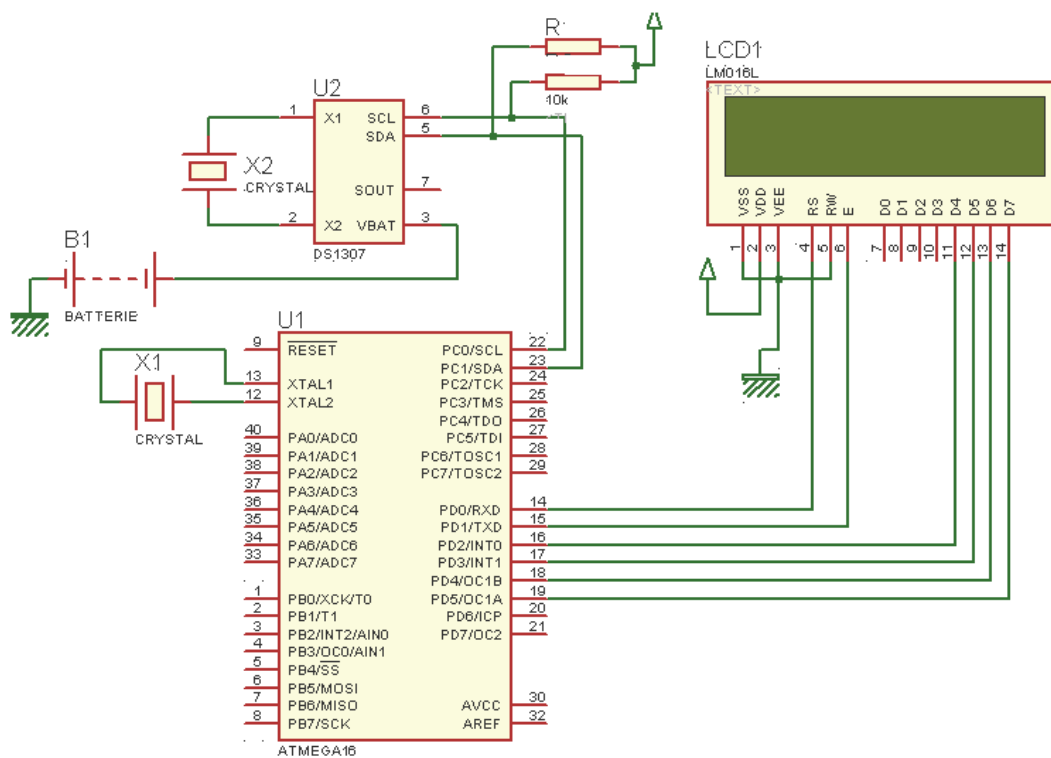
```
$regfile = "m16def.dat":$crystal = 1000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Rs = Pind.0 , E = Pind.1 , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5
Config Clock = User:Config Sda = Portc.1:Config Scl = Portc.0
Const Ds1307w = &HD0:Const Ds1307r = &HD1:Cursor Off
Do
Locate 1 , 1:Lcd Time$
Loop
End
Getdatetime:
I2cstart
```

```

I2cwrite Ds1307w
I2cwrite 0
I2cstart
I2cwrite Ds1307r
I2cwrite _sec , Ack
I2cwrite _min , Ack
I2cwrite _hour , Ack
I2cstop
_sec = Makedec(_sec) : _min = Makedec(_min) : _hour = Makedec(_hour)
Return

```

مدار برنامه بالا را در زیر مشاهده میکنید:



در برنامه ی بالا RTC در میکروکنترلر به گونه ای پیکربندی است تا بتواند اطلاعات مربوط به زمان را از چیپ DS1307

بخواند .

ارتباط سریال 1 WIRE :

1 WIRE یا ارتباط سریال یک سیمه ، یک پروتکل ارتباطی نرم افزاری برای میکروکنترلر های AVR است ، این پروتکل

که توسط بسکام برای میکروکنترلر های AVR پیاده سازی شده دارای ویژگی های به شرح می باشد :

1- در این ارتباط از یک پایه ی ورودی خروجی ، همراه با گراند و VCC ، که در مجموع سه سیم میشود ، برای

انتقال دیتا استفاده میشود.

2- بالاترین فرکانس کلاک در این پروتکل 2 کیلو هرتز است.

3- در این ارتباط میتوان تعداد دو وسیله ی اصلی و تعداد نامحدود وسیله جانبی را به هم متصل کرد.

4- کلیه خطوط باید با مقاومت 4.7 کیلو به VCC متصل شوند.

با دستور زیر باس 1 WIRE مشخص میشود:

CONFIG 1WIRE = pin

Pin : نام پایه ی دلخواه میکرو است که به عنوان ورودی و خروجی داده 1 WIRE استفاده میشود.

(این پایه باید با پایه ای که در قسمت compiler setting بسکام مشخص شده یکی باشد) پایه ای که در بالا مشخص میشود ،

باس اصلی میباشد که به دستگاه دیگر متصل است)

با دستور زیر میتوان داده را از باس 1 WIRE خواند:

var2 = 1WREAD([bytes])

var2 = 1WREAD(bytes , port , pin)

دستور اول داده را از باس اصلی و دستور دوم داده را از دیگر دستگاه های جانبی میخواند

var2 : یک متغیر است که داده خوانده شده از باس در آن ریخته میشود ، شما همچنین میتوانید معین کنید چند بایت از باس

خوانده شود ([bytes]).

Port : نام پورتهی است که دستگاه جانبی به آن متصل است .

Pin : نام پایه ی پورتهی است که دستگاه به آن متصل شده است:

A = 1wread(8 , Pinb , 2)

با دستور زیر میتوان داده را در باس 1 WIRE نوشت:

```
1WWRITE var1, bytes
```

```
1WWRITE var1 , bytes , port , pin
```

دستور اول داده را در باس اصلی و دستور دوم داده را در باس دیگر دستگاه های جانبی میریزد.

Var1: متغیر یا عدد ثابتی است که در باس نوشته میشود ، شما همچنین میتوانید معین کنید چند بایت در باس نوشته شود

([bytes]). (تعداد بایت خوانده شده و نوشته در گذر گاه باید با هم برابر باشد ، در غیر اینصورت خطا بوجود میاید)

Port: نام پورتی است که دستگاه جانبی به آن متصل است .

Pin: نام پایه ی پورتی است که دستگاه به آن متصل شده است:

```
1wwrite b , 8,pinb,2
```

با دستور زیر باس 1 WIRE ریست میشود (داده های موجود در آن پاک میشود):

```
1WRESET
```

```
1WRESET , PORT , PIN
```

دستور اول باس اصلی و دستور دوم دیگر باس ها را ریست میکند

Port: نام پورتی است که دستگاه جانبی به آن متصل است .

Pin: نام پایه ی پورتی است که دستگاه به آن متصل شده است و باید ریست شود:

```
1wreset Pinb , 2
```

با دستور زیر شماره دستگاه متصل شده به باس 1 WIRE خوانده میشود :

```
var2 = 1WIRECOUNT()
```

```
var2 = 1WIRECOUNT( port , pin)
```

var2: یک متغیر از جنس word یا integer است که داده خوانده شده از باس در آن ریخته میشود .

Port: نام پورتی است که دستگاه جانبی به آن متصل است .

Pin: نام پایه ی پورتی است که دستگاه به آن متصل شده است:

```
W = 1wirecount(pinb , 2)
```

با دستور زیر میتوان داده را از دستگاه های که به صورت سریال به یک باس متصل شده اند را خواند:

```
var2 = 1WSEARCHFIRST()
```

```
var2 = 1WSEARCHFIRST( port , pin)
```

```
var2 = 1WSEARCHNEXT()
```

```
var2 = 1WSEARCHNEXT( port , pin)
```

var2: یک متغیر از جنس long است که داده خوانده شده از باس در آن ریخته میشود.

Port: نام پورتی است که دستگاه جانبی به آن متصل است.

Pin: نام پایه ی پورتی است که دستگاه به آن متصل شده است.

مثال:

```
$regfile = "m16def.dat" : $crystal = 8000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Pind.2 , Db5 = Pind.3 , Db6 = Pind.4 , Db7 = Pind.5 , Rs = Pind.0 , E = Pind.1
```

```
Config 1wire = Portb.0 'use this pin
```

```
Dim A As Byte , C As Byte
```

```
Wait 1
```

```
1wreset
```

```
1wwrite &H33
```

```
Do
```

```
A = 1wread(8 , Pinb , 0)
```

```
Locate 1 , 1 : Lcd Hex(a)
```

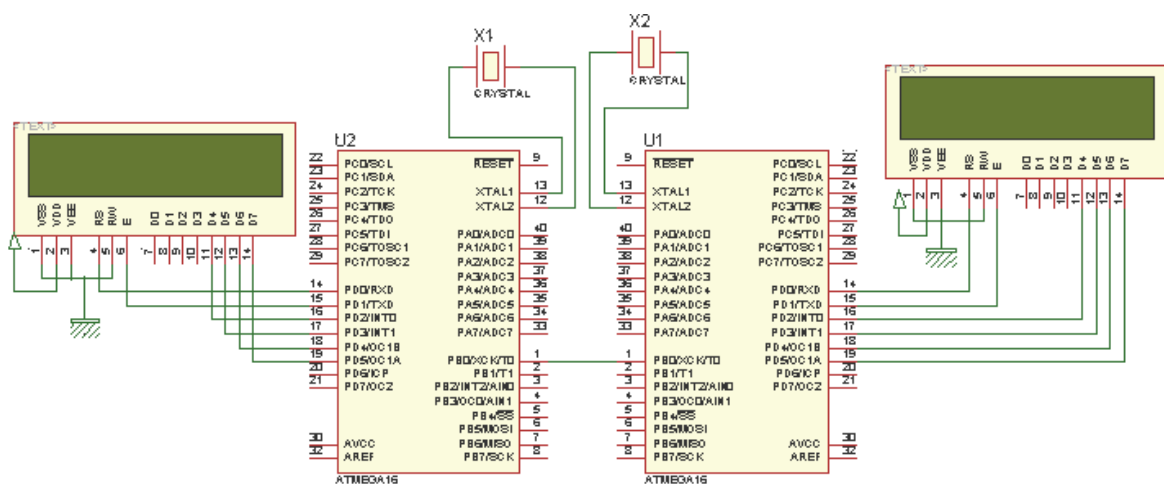
```
Wait 1 : Incr C
```

```
1wwrite C , 8 , Pinb , 0
```

```
Locate 2 , 1 : Lcd C
```

```
Loop
```

```
End
```



شرحی بر Rfid (Radio Frequency Identification)

اصولاً به هر سیستمی که قادر به خواندن و تشخیص اطلاعات افراد یا کالاها باشد سیستم شناسایی یا System Identification گفته می شود. بطور کلی شناسایی خودکار و نگهداری داده ها (AIDC) روشی است که طی آن تجهیزات خواه سخت افزاری یا نرم افزاری قادر به خواندن و تشخیص داده ها بدون کمک گرفتن از یک فرد هستند. بارکدها، کدهای دو بعدی، سیستم های انگشت نگاری، سیستم شناسایی با استفاده از فرکانس رادیویی، سیستم شناسایی با استفاده از قرنیه چشم و صدا و ... از جمله این راهکارها میباشند.

یکی از جدید ترین مباحث مورد توجه دانشمندان جهت شناسایی افراد یا کالاها استفاده از سیستم شناسایی با استفاده از فرکانس رادیویی یا RFID میباشد. RFID که مخفف سه واژه Radio Frequency Identification است؛ در دوره های متوالی توسط فروشگاه های زنجیره ای بزرگی چون "وال مارت" و "مک دونالد" و نیز سازمانهای مهمی چون "وزارت دفاع ایالت متحده آمریکا" استفاده شده و امتحان خود را به خوبی پس داده است. امروزه این تکنولوژی در تمامی مکانها، حتی سوپر مارکت های محلی نیز استفاده میشود.

■ RFID چیست ؟

تصور کنید که وارد یک فروشگاه زنجیره ای شده اید و اقلام مورد نیاز خود را داخل چرخ دستی (trolley) قرارداده اید. صندوق دار با استفاده از بار کد میبایستی تک تک اقلام داخل سبد را برداشته و اطلاعات آن را توسط بارکد خوان (Reader Barcode) یکی یکی به داخل رایانه وارد کند تا فاکتور اقلام انتخابی شما صادر گردد. بسیاری از اوقات بدلیل آنکه تعداد کالاهای خریداری شده بسیار زیاد میباشند؛ صفهای طولانی ای در فروشگاه های زنجیره ای تشکیل میشود. گاهی اوقات نیز مخدوش شدن علائم بار کد، از خواندن اطلاعات جلوگیری میکند، که این خود موجب مشکلات بیشتری میشود.

با فن آوری RFID شما سبد کالای خود را بر میدارید و بدون اینکه مجبور به ایستادن در صفهای طولانی شوید و یا حتی بدون اینکه مجبور باشید اقلام خریداری شده را به صندوقدار یا نگهبان نشان دهید، از در خارج میشوید. در سیستم RFID اطلاعات مربوط به کالا به جای چاپ شدن بر روی آن در حافظه ی تگ RFID قرار گرفته است. با

نزدیک شدن کارت RFID به دستگاه خواننده ی تگ ، با استفاده از علائم رادیویی کلیه اطلاعات جاری از قبیل تعداد، قیمت، وزن، ... به کامپیوترهای موجود در درهای خروجی مخابره میشود .

این برچسبها دارای دو بخش تراشه و آنتن هستند و دارای عملکرد بسیار ساده ای می باشند؛ تراشه اطلاعات را از طریق آنتن منتشر میکند و حسگرهایی در اطراف قرار دارند، این اطلاعات را دریافت میکنند. از جمله مهمترین محاسن RFID کاهش سرقت یا دزدی و آمار گیری سریع از تعداد اقلام بدون نیاز به نیروهای انسانی است.

بطور کلی RFID یا سیستم شناسایی با استفاده از فرکانس رادیویی سامانه ی شناسایی بی سیمی است که قادر به تبادل داده ها بوسیله برقراری اطلاعات بین یک Tag که به یک کالا ، شیء یا .. متصل شده است و یک بازخوان (Reader) می باشد. اصولاً سامانه های RFID از سیگنالهای الکترونیکی و الکترو مغناطیسی برای خواندن و نوشتن داده ها بدون تماس بهره گیری می کنند.

Tag ها وسیله شناسایی متصل شده به کالایی است که ما میخواهیم آن را رد یابی کنیم و بازخوان ها (Reader) ها وسایلی هستند که حضور برچسب ها را در محیط تشخیص داده و اطلاعات ذخیره شده در آنها را بازیابی میکنند. با توجه به اینکه این سیستمها بر مبنای تغییرات امواج مغناطیسی و یا فرکانس های رادیویی کار میکنند، جهت تقویت سیگنالهای موجود در محیط گاهی اوقات از آنتن (تقویت کننده سیگنال) نیز استفاده میشود.

بطور کلی فن آوری RFID از تجهیزات ذیل جهت پیاده سازی بهیه خود کمک میگیرد:

- ✓ انواع برچسب Tag
- ✓ انواع خواننده برچسب Reader
- ✓ انواع نویسنده اطلاعات Printer
- ✓ آنتن - تقویت کننده سیگنال
- ✓ نرم افزار مدیریت اطلاعات
- ✓ بانک اطلاعاتی، ساختار شبکه اطلاعاتی

TAG چیست؟

همانطور که گفته شد Tag ها وسیله شناسایی متصل شده به کالا، شیء یا فردی هستند که ما میخواهیم آنرا رد یابی کنیم. اما اینکه هر یک از کالاها دارای اشکال و ظواهر گوناگون و نیز دارای محیطهای فیزیکی گوناگونی است، این ضرورت را ایجاد میکند تا Tag ها را با توجه به ویژگیهای فیزیکی (ظاهریشان) دسته بندی کنیم.

بطور کلی بعضی از ویژگیهای ظاهری Tag ها بصورت زیر میباشد:

الف - Tag هایی که دارای کفه پلاستیکی از جنس PVC میباشد و معمولاً در وسط آنها یک سوراخ دیده میشود که بسیار با دوام بوده و میتوان از آنها بارها و بارها استفاده کرد.

ب - Tag هایی که شبیه کارتهای اعتباری هستند و معمولاً به آنها کارتهای هوشمند بدون تماس (Cards Contact less Smart) گفته میشود. (کارت غذا، در اکثر دانشگاههای کشور)

ج - Tag هایی که بصورت لایه های کاغذی بر روی برچسب ساخته میشوند که به آنها برچسب های هوشمند (Smart Labels) گفته میشود.

د - Tag هایی که در محیطهای قابل فرسایش (مثلاً آب یا مایع) به خوبی کار میکنند. اینگونه Tag ها در کپسولهای شیشه ای قرار دارند.

ه - Tag های کوچک که در داخل اشیاء عمومی مثل لباس، ساعت، دستبند و کار گذاشته میشود. اغلب ممکن است به شکل یک کلیه یا دسته کلید بنظر برسند.

در صورتیکه بخواهیم Tag ها را با در نظر گرفتن منبع انرژی تامین کننده شان دسته بندی کنیم به 4 دسته اصلی تقسیم بندی می شوند:

✓ Tag های غیر فعال Tags Passive که انرژی و برق مورد نیاز خود را از Reader ها بوسیله یکسری از روش های تراگیل بدست می آورند.

✓ Tag های فعال Tags Active که انرژی مورد نیازشان توسط یک باتری داخلی و جهت برقراری ارتباط دارای یک پردازنده، یک حافظه و حسگر می باشند.

✓ Tag. هایی نیمه غیر فعال Semi-Passive Tags که علاوه بر استفاده از باتری داخلی شان، میتوانند از انرژی منتقل

شده توسط Readerها نیز بهره مند شوند.

✓ Tag. های دو طرفه Tags way Two که علاوه بر استفاده از باتری داخلی شان میتوانند بدون کمک گرفتن از

Readerها دیگر اقسام هم شکل خود را نیز شناسایی کرده و با آنها به گفتگو پردازند.

▪ Reader چیست؟

قبلاً اشاره شد که Reader ها وسایل الکترونیکی هستند که حضور Tag ها را در محیط تشخیص داده و اطلاعات ذخیره شده در آنها را بازیابی میکنند.

سه دسته عمده Reader ها بصورت:

1. مدل ثابت Type Fixed

2. مدل دستی Hand held Type

3. مدل کارت PC Card Type

پیاده سازی سیستم RFID یا AVR :

هماون طور که در قسمت قبل اشاره شد سیستم RFID از سه بخش اصلی زیر تشکیل میشود:

❖ برچسب RFID یا Tag که بر روی کالای مورد نظر نصب میشود و حاوی یک شماره سریال است ، این شماره

سریال توسط کارخانه سازنده یا عوامل فروش یا... در حافظه ی Tag ریخته میشود (در ادامه راجع به Tag بیشتر

توضیح داده می شود).

❖ خواننده اطلاعات یا Hitag یا Reader که وظیفه ی تغذیه Tag و گرفتن شماره سریال و دادن آن به پردازنده را دارد.

❖ دستگاه های مدیریت اطلاعات یا پردازنده. این دستگاه ها اطلاعات گرفته شده از ریدر را پردازش می کنند و...

فرض کنید شما میخواهید یک سیستم rfid را راه اندازی کنید ، شما باید موارد بالا را تهیه کنید . فروشنده لوازم rfid ، تعدادی

tag ، یک دستگاه ریدر ، و یک نرم افزار در اختیار شما میگذارد ، همچنین گاتالوگی به شما میدهد که در آن خروجی

ریدر به ازای خواندن هر tag آورده شده است ، فرضا به ازای tag1 شماره سریال 01110111110110101 در اختیار شما

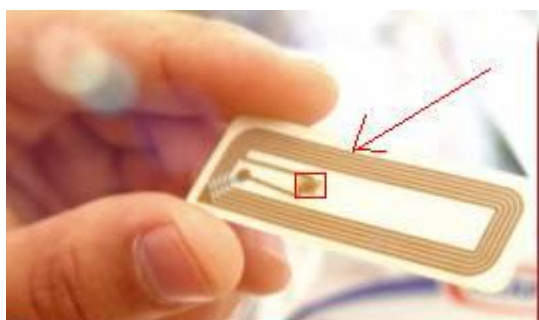
گذاشته میشود (هنگامی که tag شماره یک را نزدیک دستگاه ریدر میبرید ، دستگاه شماره سریال فوق را از آن میخواند) شما باید در نرم افزار ، مشخصات کالای را که میخواهید tag شماره یک را روی آن بچسبانید ،وارد کنید (برای کالا های از یک جنس شماره سریال ها یکی است).

■ ساخت Tag:

یکی از تراشه های پر کاربرد در صنعت RFID تراشه های سری em40xx میباشد ، این تراشه ها در اشکال یکسان با اندازه بسیار کوچک ساخته میشوند ، حافظه ی این ایسی ها قابل برنامه ریزی است (شما ایسی برنامه ریزی شده را خریداری میکنید) در زیر به بررسی em4102 که یک تراشه پر کاربرد از این خانواده است ، می پردازیم:

Em4102 یک ایسی با تکنولوژی cmos (بسیار کم مصرف) است که تنها برای خواندن دیتای rf مورد استفاده قرار میگیرد. این ایسی هنگامی که در یک میدان مغناطیسی قرار میگیرد ، بوسیله سیم پیچ خارجی که بین پایه های coil1 و coil2 متصل است ، تغذیه و روشن می شود و به وسیله ترمینال اول خود که به سر اول سیم پیچ متصل است ، از فرستنده ، کلاک (پالس هم زمانی ، این پالس از گیرنده به فرستنده ارسال میشود و به آن میفهماند که چه موقع اطلاعات را بفرستد) میگیرد و در صورت درست بودن آن ، 64 بیت اطلاعاتی که متصدی روی حافظه ی آرایه ای آن پروگرام کرده را به شکل مدولاسیون جریان برای ریدر میفرستد.

برنامه ریزی چیپ بوسیله ی لیزر و با ذوب کردن اتصالات پلی سلیکونی (polysilicon links) انجام میشود ، بنابراین هر چیپ دارای اطلاعات منحصر بفرد و مخصوص خود است.



این ایسی دارای چندین مد اختیاری برای مشخص کردن نوع کد و نرخ ارسال آن است . نرخ ارسال دیتا 64، 32، یا 16 بیتی است. به علت کم مصرف بودن هسته اصلی مدار ، نیازی به خازن برای جدا کردن تغذیه مدار نیست و تنها قطعه خارجی مورد نیاز سلف میباشد .

در زیر شکل پایه ها و کار هریک آورده شده است:

Pad	Name	Function	
1	COIL2	Coil terminal 2 / data output	
2	COIL1	Coil terminal 1 / clock input	
3	VDD	Positive internal supply voltage	
4	VSS	Negative internal supply voltage	

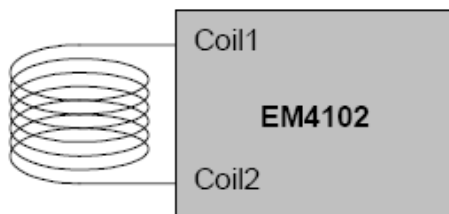
پایه 1: ترمینال متصل به سر دوم سیم پیچ (انتن) ، خروجی داده

پایه 2: ترمینال متصل به سر اول سیم پیچ (انتن) ، ورودی کلاک

پایه 3: پایه مثبت تغذیه داخلی (برای دستگاه های دیگر)

پایه 4: پایه منفی تغذیه داخلی (برای دستگاه های دیگر)

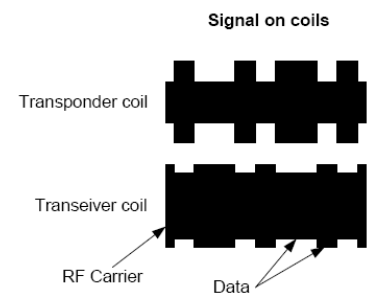
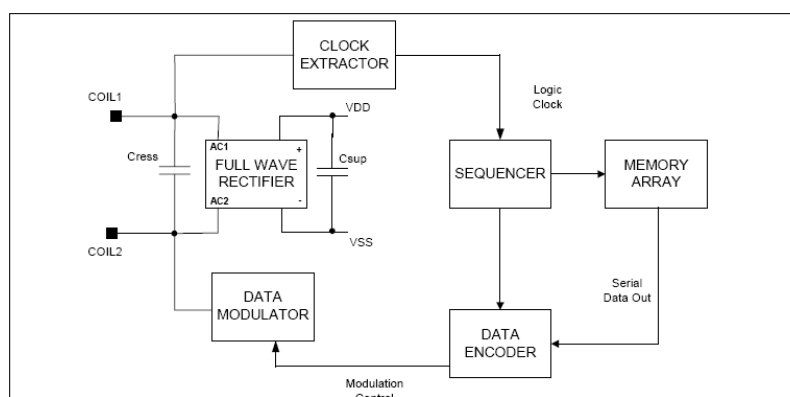
در زیر همچنین مشخصات قطعه و طریقه اتصال سیم پیچ به آن آورده شده است:



Absolute Maximum Ratings

Parameter	Symbol	Conditions
Maximum DC Current forced on COIL1 & COIL2	I_{COIL}	$\pm 30mA$
Power Supply	V_{DD}	-0.3 to 7.5V
Storage Temp. Die form	T_{store}	-55 to +200°C
Storage Temp. PCB form	T_{store}	-55 to +125°C
Electrostatic discharge maximum to MIL-STD-883C method 3015	V_{ESD}	2000V

بلوک داخلی قطعه و سیگنال موجود بر روی سیم پیچ در هنگام ارسال داده :



همانطور که قبلا نیز گفتیم ، شماره سریال مورد نیاز کاربر ، توسط کارخانه ی سازنده ی بر روی tag که احتمالا در داخل یک برچسب و در کنار آنتن مخصوص خود نصب شده ریخته میشود ، در این حالت نیازی به ساخت Tag و برنامه ریزی آن نمیشد ، در صورت نیاز شما میتوانید با مطالعه ی دیتا شیت این قطعه اطلاعات بیشتری را در مورد آن بدست آورید .

▪ ساخت خواننده اطلاعات یا hitag:

Hitag یا ریدر از سه قسمت زیر تشکیل شده است:

- ✓ آنتن ، این قسمت وظیفه انتشار امواج برای تغذیه tag و گرفتن اطلاعات از آن و تحویل به ریدر را به عهده ارد .
- ✓ یک ماژول الکترونیک RF که مسئول برقراری ارتباط بین آنتن و دستگاه پردازش است.
- ✓ یک ماژول کنترل کننده الکترونیکی که اطلاعات را از ماژول rf گرفته و آن را به داده قابل فهم برای کامپیوتر یا ... میکند .

ماژول های الکترونیک RF:

در این رابطه تراشه های زیادی از شرکت های مختلف ارائه شده است . که میکرو کنترلر AVR و نرم افزار بسکام از تراشه ی EM4095 ساخت شرکت SWATCH GROUP و تراشه HTRC110 ساخت شرکت Philips ، حمایت میکنند در زیر به بررسی این دو تراشه میپردازیم:

▪ EM4095:

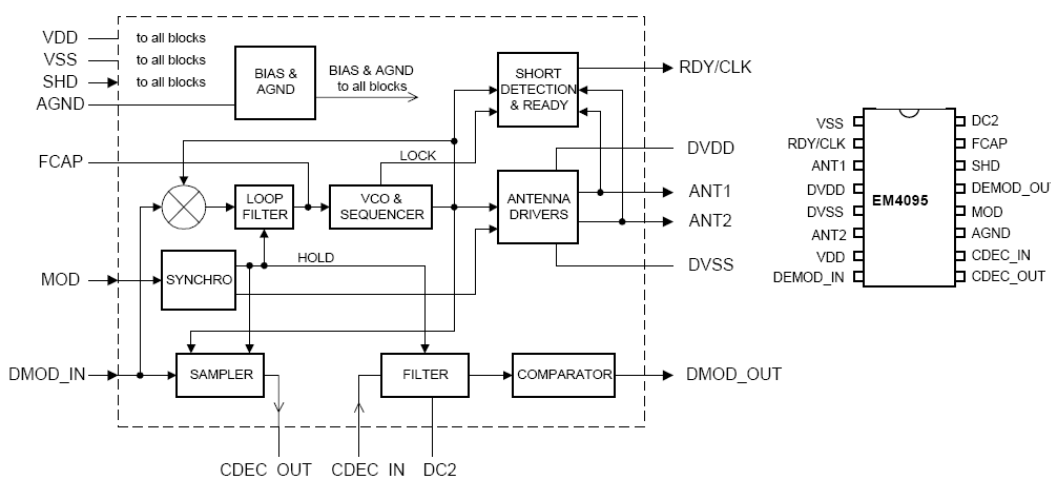
راه اندازی این تراشه بسیار آسان است ، چون کلیه امکانات داخل آن تعبیه شده اند . کافی است شما چند قطعه اولیه را به آن متصل کنید و سپس آن را به میکرو اتصال دهید . در زیر شکل و نام پایه ها و کار هر یک را مشاهده میکنید :

محل اتصال	شرح و توصیف	نام پایه	Pin
گراند	ولتاژ منفی تغذیه	VSS	1
ورودی کلاک دستگاه دیگر	خروجی کلاک برای همزمان سازی دیگر دستگاه های جانبی	RDY/CLK	2
پایه شماره 1 آنتن	پایه شماره 1 آنتن	ANT1	3
ولتاژ مثبت تغذیه	ولتاژ مثبت برای تغذیه آنتن	DVDD	4
گراند	ولتاژ منفی برای تغذیه آنتن	DVSS	5

6	ANT2	پایه 2 انتن	پایه شماره 2 انتن
7	VDD	ولتاژ مثبت تغذیه	ولتاژ مثبت تغذیه
8	DEM0D_IN	ورودی کنترل کننده ولتاژ انتن	پایه شماره 2 انتن
9	CDEC_OUT	بلوک اتصال خازن های خروجی	خروجی یک خازن
10	CDEC_IN	بلوک اتصال خازن های ورودی	ورودی یک خازن
11	AGND	گراند آنالوگ	با یک خازن به گراند
12	MOD	ولتاژ سطح بالا برای مدولاسیون انتن	گراند
13	DEM0D_OUT	سیگنال دیجیتال خروجی برای دستگاه پردازنده	ورودی داده ی پردازنده
14	SHD	ولتاژ بالا رونده برای بیدار کردن پردازنده از مدهای sleep	پیه وقفه پردازنده
15	FCAP	پایه اتصال به خازن برای راه اندازی pll داخلی	به خازن pll
16	DC2	پایه اتصال به خازن کوپلاژ خروجی	به خازن کوپلاژ

در زیر بلوک دیاگرام داخلی و شکل قطعه را مشاهده میکنید .

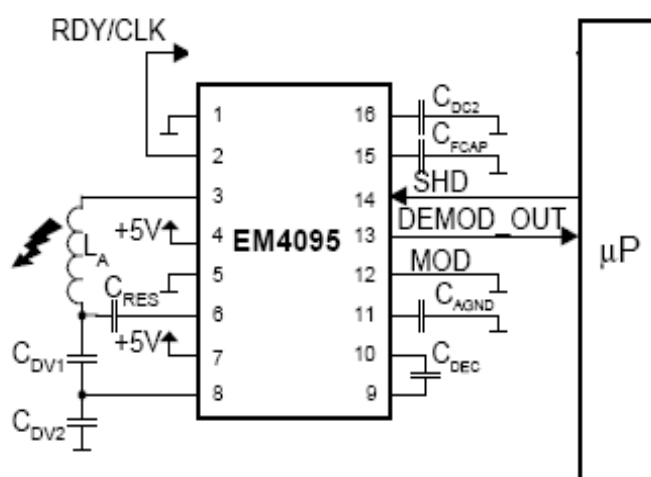
Block Diagram



مقادیر قطعات برای عمل کرد صحیح قطعه :

Parameter	Symb	Min	Typ	Max	Units
Operating junction temperature	T_J	-40		+110	$^{\circ}\text{C}$
Supply voltage	V_{DD}	4.1	5	5.5	V
Antenna circuit resonant frequency	F_{RES}	100	125	150	kHz
AC peak current on ANT1 & ANT2 pads	I_{ANT}			250	mA
C_{FCAP}		*	10	*	nF
C_{DEC}		*	100	*	nF
C_{DC2}		*	6.80	*	nF
C_{AGND}		100		220	nF
Package thermal resistor SO16	R_{thj-a}	69	70	71	$^{\circ}\text{C/W}$

با توجه به مطالب بالا مدارای که قطعه برای کار نیاز دارد به شکل زیر خواهد بود :



✓ مقادیر خازن های و C_{DV1} و C_{DV2} با توجه به نوع آنتن به کار رفته در مدار تعیین خواهد شد .

✓ L_a همان سیم پیچ آنتن است .

✓ این تراشه از طریق دو پایه SHD و DEMOD_OUT با میکرو ارتباط برقرار میکند .

✓ پایه DEMOD_OUT برای ارسال اطلاعات به میکرو و پایه SHD برای بیدار کردن میکرو میباشد .

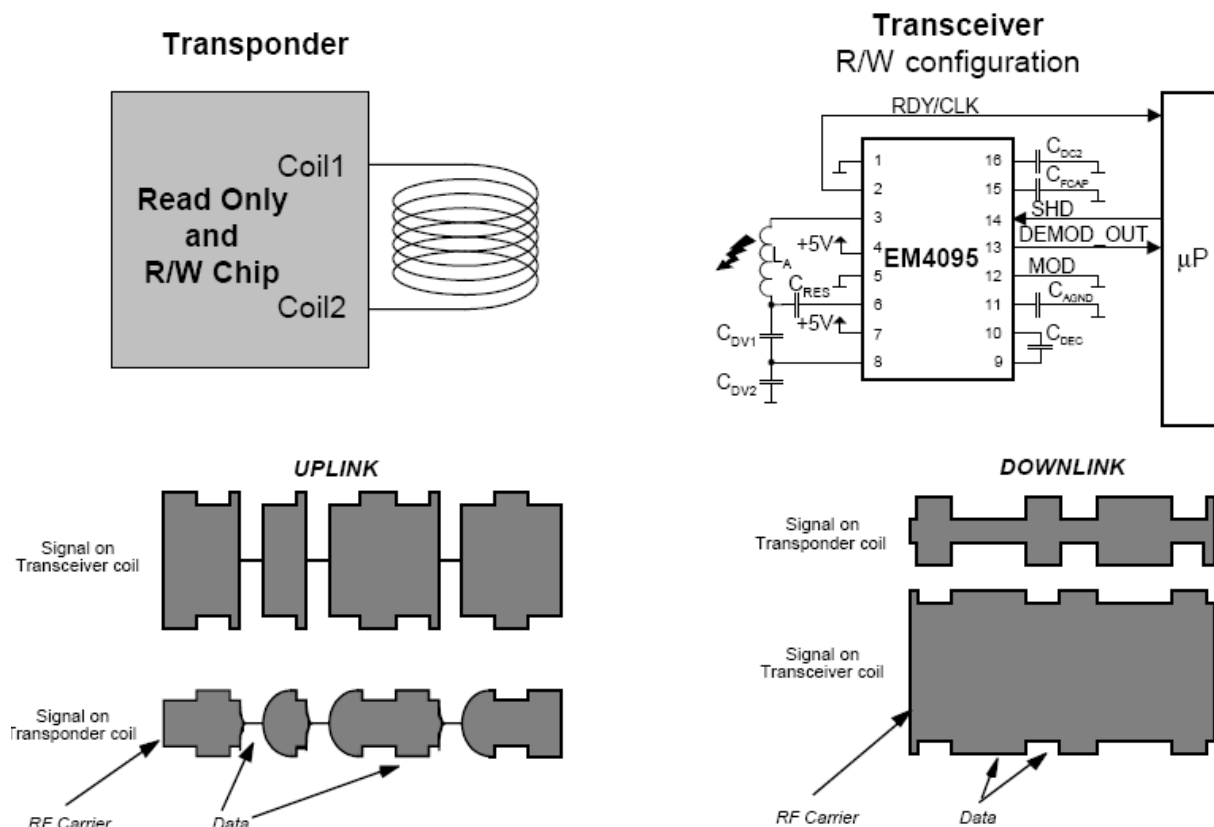
▪ نحوه عمل کرد مدار :

با اعمال تغذیه به مدار ، ولتاژ توسط سیم پیچ آنتن در فضای اطراف منتشر میشود . با قرار گرفتن اولین تگ در محدوده مدار

، ولتاژ در آن القا شده و تگ با کشیدن جریان اطلاعات خود (شماره سریال) را به مدار میفرستد ، در همین هنگام ، چیپ

پایه shd را یک میکند تا به میکرو میفهماند که ارسال اطلاعات تا چند سیکل دیگر آغاز خواهد شد و در ادامه میکرو شروع

به دریافت اطلاعات از طریق پایه DEMOD_OUT خواهد کرد .



■ راه اندازی چیپ EM4095 در محیط بسکام :

این چیپ با دستور زیر پیکربندی میشود :

CONFIG HITAG = prescale, TYPE=tp, DEMOD= demod, INT=@int

Prescale : مقدار دقت اندازه گیری تایمر 0 را مشخص میکند ، میکرو avr برای شمارش و ایجاد کلاک به تایمر 0 نیاز دارد .

مقدار prescale میتواند 0 تا 256 باشد .

TYPE=tp: مشخص کننده نوع چیپ میباشد که در این مورد چیپ مرد استفاده EM4095 است .

DEMOD= demod: به جای demod نام پایه ای که خروجی داده چیپ به آن متصل میشود ، نوشته میشود .

INT=@int : نام وقفه مورد استفاده است ، @int میتواند int1 یا int2 یا int0 باشد . پایه SHD چیپ به ورودی وقفه

مورد استفاده متصل می گردد. مثال :

Config Hitag = 64 , Type = Em4095 , Demod = Pind.3 , Int = @int1

در مثال بالا پایه DEMOD_OUT چیپ به پایه d.3 میکرو و پایه SHD به ورودی وقفه شماره 1 متصل میگردد .

دستور READHITAG :

این دستور به فرم زیر است و توسط آن میتوان شماره سریال تگ را خواند و از وجود یا عدم وجود آن آگاهی یافت :

READHITAG(var)

result = READHITAG(var)

result : یک متغیر عددی (از جنس بیت ، بایت یا ...) میباشد که در صورت عدم وجود تگ و شماره سریال مقدار آن صفر میباشد . در صورتی که شماره سریال وجود داشته باشد ، مقدار این متغیر 1 است و شماره سریال در متغیر var ذخیره میشود.

دستور _checkhitag :

با این دستور ، تگ چک میشود و در صورتی که اطلاعات و بیت های خطا و کلاک و ... صحیح باشند ، اجازه خروج داده

صادر میگردد ، این دستور به فرم زیر است :

Call _checkhitag

متغیر های رزرو شده برای EM4095 :

در بسکام تعدادی متغیر برای EM4095 تعیین شده است که با پیکربندی چیپ مقادیر داده به صورت خود کار در آنها قرار داده میشود و شما میتوانید از آنها استفاده نمایید :

_tagflag	هنگامی یک تگ از مقابل دستگاه عبور میکند ، این متغیر یک میشود .
_tag_insync	حالت بیت ها در این متغیر ذخیره میشود
_tag_bitcount	هنگامی که بین چیپ و میکرو همزمانی وجود ندارد ، کلیه بیت ها در این متغیر ذخیره میشوند
_tag_tbit	یک متغیر 8 بیتی برای ذخیره کردن بیت های دریافتی
_tag_par	بیت های توازن (تشخیص خطا) در این متغیر ذخیره میشود
_tag_timeout	مقدار زمانی که صرف شناسایی و خواندن تگ میشود ، در این متغیر ذخیره میگردد
_taglasttime	مقدار زمانی که از آخرین تحریک گذشته ، در این متغیر ذخیره میشود (تحریک توسط پایه SHD)
_tagid	آخرین داده ، که همان شماره سریال تگ میباشد در این متغیر ذخیره میشود

متغیر های که در بالا معرفی شدند ، صرفاً برای یافتن خطا ها و دادن گزارش استفاده میشوند ، مثلاً شما میتوانید با استفاده از متغیر `_tag_par` نوع بیت توازن را تعیین کنید (زوج یا فرد) ، یا با استفاده از بیت `_tag_timeout` سرعت دستگاه را اندازه گیری کنید .

مثال :

```
$regfile = "M88def.dat"
$baud = 19200
$crystal = 8000000
Dim Tags(5) As Byte 'make sure the array is at least 5 byte
Dim J As Byte
Config Hitag = 64 , Type = Em4095 , Demod = Pind.3 , Int = @int1
Print "Test EM4095"
'you could use the PCINT option too, but you must mask all pins out so it will only respond to our pin
' Pcmsk2 = &B0000_0100
' On Pinct2 Checkints
' Enable Pinct2
On Int1 Checkints Nosave 'we use the INT1 pin all regs are saved in the lib
Config Int1 = Change 'we have to config so that on each pin change the routine will be called
Enable Interrupts 'as last we have to enable all interrupts
Do
    Print "Check..."

    If Readhitag(tags(1)) = 1 Then 'this will enable INT1
        For J = 1 To 5
            Print Hex(tags(j)) ; " ";
        Next
        Print
    Else
        Print "Nothing"
    End If
    Waitms 500
Loop
Checkints:
Call _checkhitag 'in case you have used a PCINT, you could have other code here as well
Return
```

در مثال بالا ، چون هنوز تگی از مقابل سیستم عبور نکرده است مقدار (1)tags Readhitag برابر صفر میباشد ، پایه وقفه پیکربندی شده است و مدام چک می شود . با عبور اولین تگ پایه SHD تغییر وضعیت داده و میکرو به برجسب Checkints پرش میکند ، با دستور Call _checkhitag تگ چک میشود . در صورتی که اطلاعات آن درست باشد (اطلاعات ، کلاک و ...)cpu به حلقه اصلی برمیگردد ، اکنون مقدار (1)tags Readhitag برابر یک شده و شرط اجرا میشود . شماره سریال tag در متغیر tag0 قرار میگیرد و به پورت سریال ارسال میشود .در صورتی که تگ از محدوده سیستم خارج شود مقدار (1)tags Readhitag برابر صفر شده و سیستم تا آمدن تگ بعدی منتظر می ماند .

مثال :

```
$regfile = "M88def.dat"
$baud = 19200
$crystal = 8000000
$hwstack = 40
$swstack = 40
$framesize = 40
Declare Function Havetag(b As Byte ) As Byte      'Make SHD and MOD low
_md Alias Portd.4
Config _md = Output
_md = 0
_shd Alias Portd.5
Config _shd = Output
_shd = 0
Relay Alias Portd.2 : Config Relay = Output
S3 Alias Pinb.0 : S2 Alias Pinb.2 : S1 Alias Pinb.1
Portb = &B111                                ' these are all input pins and we activate the pull up resistor
Config Clock = Soft                            'we use a clock
Config Date = Dmy , Separator = -
Enable Interrupts                             ' the clock and RFID code need the int
Date$ = "15-12-07"                             ' just a special date to start with
Time$ = "00:00:00"
Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 , Db6 = Portc.4 , Db7 = Portc.5 , E = Portc.1 , Rs = Portc.0'Config Lcd Sets The
Portpins Of The Lcd
Config Lcd = 16 * 2                            '16*2 type LCD screen
Cls : Lcd " EM4095 sample" : Lowerline : Lcd "MCS Electronics"
Dim Tags(5) As Byte                            'make sure the array is at least 5 bytes
Dim J As Byte , Idx As Byte
Dim Eramdum As Eram Byte                       ' do not use first position
```

```

Dim Etagcount As Eram Byte          ' number of stored tags
Dim Etags(100) As Eram Byte         'room for 20 tags
Dim Stags(100) As Byte               'since we have enough SRAM store them in sram too
Dim Btags As Byte , Tmp1 As Byte , Tmp2 As Byte
Dim K As Byte , Tel As Byte , M As Byte
Config Hitag = 64 , Type = Em4095 , Demod = Pind.3 , Int = @int1
Print "EM4095 sample"
'you could use the PCINT option too, but you must mask all pins out so it will only respond to our pin
' Pcmsk2 = &B0000_0100 ' On Pcnt2 Checkints
On Int1 Checkints Nosave             'we use the INT1 pin all regs are saved in the lib
Config Int1 = Change                 'we have to config so that on each pin change the routine will be called
Enable Interrupts                   'as last we have to enable all interrupts
'read eeprom and store in sram
'when the program starts we read the EEPROM and store it in SRAM
For Idx = 1 To 100                  'for all stored tags
    Stags(idx) = Etags(idx)
    Print Hex(stags(idx)) ; ",";
Next
Btags = Etagcount                   ' get number of stored tags
If Btags = 255 Then                  ' an empty cell is FF (255)
    Print "No tags stored yet" : Btags = 0 : Etagcount = Btags          ' reset and write to eeprom
Else                                ' we have some tags
    For J = 1 To Btags
        Tmp2 = J * 5                'end
        Tmp1 = Tmp2 - 4              'start
        Print "RFID ; " ; J          ' just for debug
        For Idx = Tmp1 To Tmp2
            Print Hex(stags(idx)) ; ",";
        Next : Print : Next
    End If
Do
    Print "Check..."
    Upperline : Lcd Time$ ; " Detect"
    If Readhitag(tags(1)) = 1 Then    'this will enable INT1
        Lowerline
        For J = 1 To 5
            Print Hex(tags(j)) ; ","; Lcd Hex(tags(j)) ; ","
        Next
        M = Havetag(tags(1))          'check if we have this tag already
        If M > 0 Then
            Print "Valid TAG ; " ; M
            Relay = 1                'turn on relay
        End If
    End Do

```

```

    Waitms 2000 : Relay = 0 'wait 2 secs relay off
End If
Print
Else
    Print "Nothing"
End If
If S3 = 0 Then 'user pressed button 3
    Print "Button 3" : Cls : Lcd "Add RFID"
    Do
        If Readhitag(tags(1)) = 1 Then 'this will enable INT1
            If Havetag(tags(1)) = 0 Then 'we do not have it yet
                If Btags < 20 Then 'will it fit?
                    Incr Btags 'add one
                    Etagcount = Btags : Idx = Btags * 5 'offset
                    Idx = Idx - 4 : Lowerline
                    For J = 1 To 5
                        Lcd Hex(tags(j)) ; "," : Stags(idx) = Tags(j) : Etags(idx) = Tags(j) : Incr Idx
                    Next
                    Cls : Lcd "TAG stored" : Waitms 1000
                End If : End If
            Exit Do
        End If
    Loop
End If
If S2 = 0 Then : Print "Button 2" : End If
If S1 = 0 Then
    Print "Button 1"
End If
Waitms 500
Loop
'check to see if a tag is stored already 'return 0 if not stored 'return value 1-20 if stored
Function Havetag(b As Byte ) As Byte
    Print "Check if we have TAG : ";
    For K = 1 To 5
        Print Hex(b(k)) ; ","
    Next
    For K = 1 To 20
        Tmp2 = K * 5 'end address
        Tmp1 = Tmp2 - 4 'start
        Tel = 0
        For Idx = Tmp1 To Tmp2
            Incr Tel

```

```

If Stags(idx) <> B(tel) Then           'if they do not match
    Exit For                           'exit and try next
End If

Next

If Tel = 5 Then                         'if we did found 5 matching bytes we have a match
    Print "We have one" : Havetag = K   'set index
    Exit Function
End If

Next

Havetag = 0                            'assume we have nothing yet

End Function

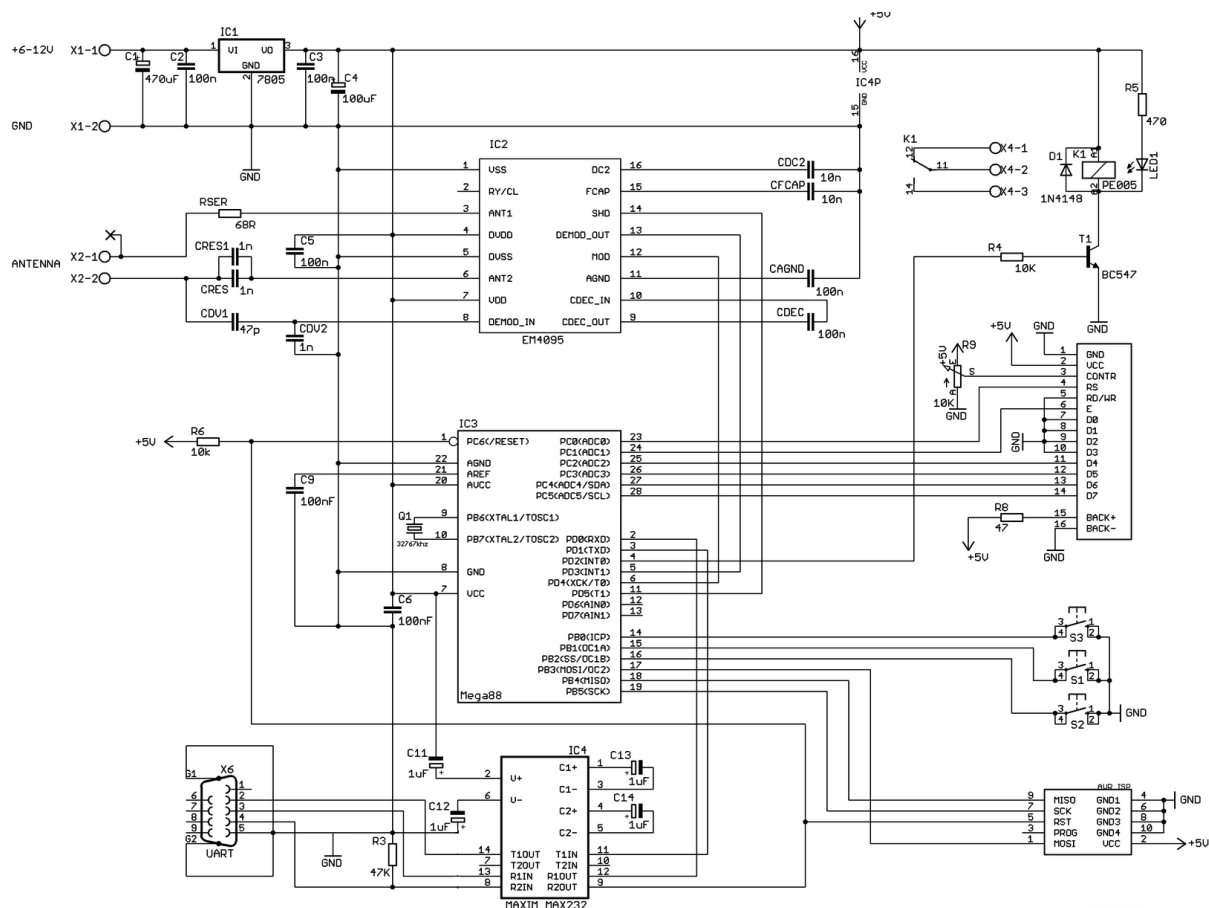
Checkints:
Call _checkhitag                       'in case you have used a PCINT, you could have other code here as well

Return

```

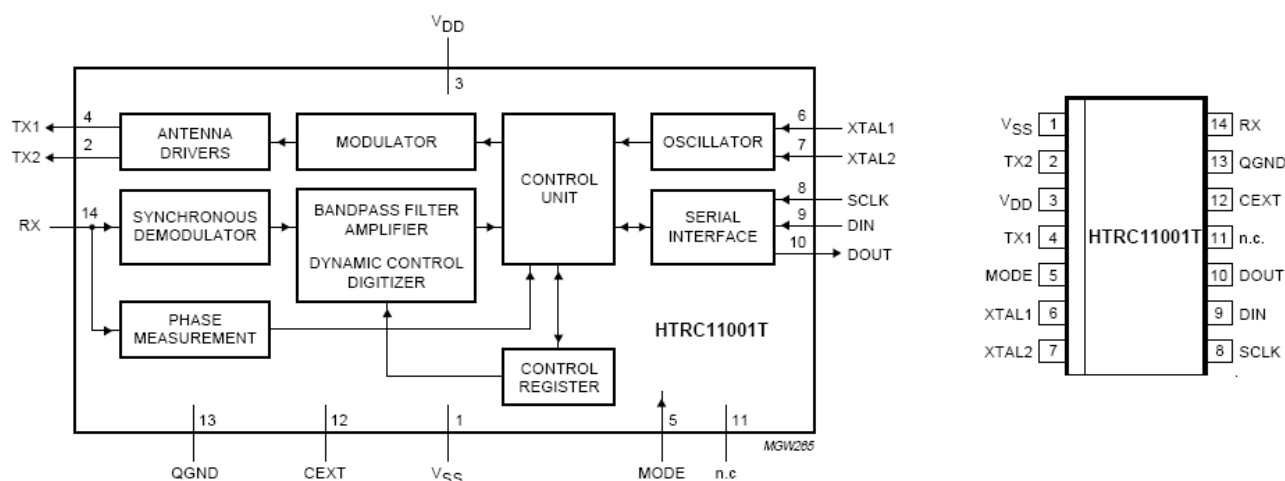
در مثال بالا که شما میتوانید با مراجعه به [این آدرس](#) اطلاعات بیشتری در مورد آن کسب کنید، یک سیستم کامل بر مبنای تکنولوژی RFID ایجاد شده است. در این سیستم میکروکنترلر ATMEGA88 اطلاعات موجود در خروجی چیپ EM4095 را دریافت کرده و آن را از طریق پورت سریال به کامپیوتر ارسال می کند.

نحوه ی کار این مدار به این صورت است که بعد از اعمال شدن ولتاژ تغذیه (شماتیک در ادامه آورده شده است) تمامی المان های مدار روشن شده و سیگنالی با فرکانس 125 کیلو هرتز در خروجی چیپ RFID ایجاد میشود ، با اتصال خروجی چیپ MAX232 به پورت سریال کامپیوتر خود میتوانید اطلاعات دریافتی و ارسالی مورد نیاز را در محیط هایپر ترمینال یا ترمینال ایمولاتور بسکام مشاهده نمایید . در زیر شماتیک مورد نیاز برای مثل بالا را مشاهده میکنید :



: HTRC110

راه اندازی این تراشه نیز مانند EM4095 بسیار ساده است ، این تراشه نیز مانند EM4095 به قطعات اندکی برای کار نیاز دارد .



در زیر شکل ظاهری ، نام پایه ها و بلوک دیاگرام قطعه را مشاهده میکنید :

نام پایه	شماره	محل اتصال
VSS	1	زمین مدار
		زمین چیپ

TX2	2	خروجی شماره 1 انتن	پایه شماره انتن
VDD	3	ولتاژ مثبت تغذیه	ولتاژ 5 ولت تغذیه
TX1	4	خروجی شماره 2 انتن	پایه شماره 2 انتن
MODE	5	فعال ساز چیپ (با پالس صفر)	زمین مدار
XTAL1	6	ورودی شماره 1 نوسان ساز	کریستال نوسان ساز
XTAL2	7	ورودی شماره 2 نوسان ساز	کریستال نوسان ساز
SCLK	8	ورودی کلاک از طرف میکرو کنترلر	به یکی از پایه های میکرو، که در برنامه مشخص میشود
DIN	9	ورودی داده از طرف میکرو کنترلر	به یکی از پایه های میکرو، که در برنامه مشخص میشود
DOUT	10	خروجی داده به میکرو کنترلر	به یکی از پایه های میکرو، که در برنامه مشخص میشود
n.c.	11	بدون اتصال	بدون اتصال
CEXT	12	محل اتصال خازن کوپلاژ	با یک خازن به زمین متصل میشود
QGND	13	محل اتصال خازن ایجاد کننده ی زمین انالوگ	با یک خازن به زمین متصل میشود
RX	14	ورودی دمولاتور	توسط یک خازن به انتن متصل میشود از آن نمونه برمیدارد

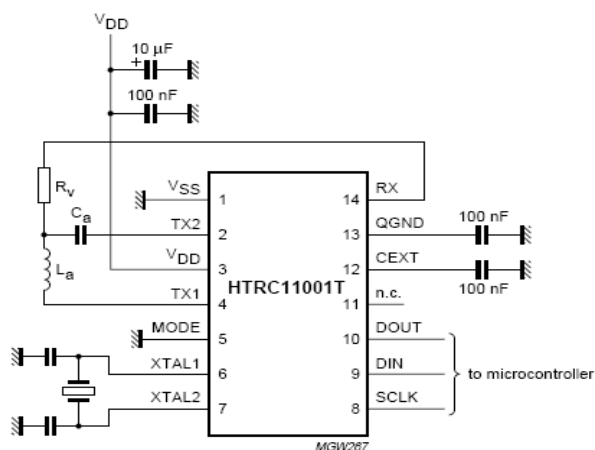
نکته :

✓ مقدار کریستال مورد استفاده 8 مگاهرتز میباشد، میکرو نیز با فرکانس 8 مگا هرتز کار میکند.

✓ در صورتی پایه شماره 5 به VCC متصل شود، چیپ از کار می افتد.

✓ با توجه به نمونه ای که توسط پایه 14 از سیگنال انتن گرفته میشود، داده خروجی ای چیپ از چیپ قبلی معتبر تر

است.



با توجه به موارد بالا شما برای راه اندازی یک سیستم rfid با

این چیپ به مدار مقابل نیاز دارید :

La : مقدار ضریب خود القایی سیم پیچ آنتن میباشد که

توسط سازنده آنتن مشخص خواهد شد

ca : مقدار این خازن از رابطه زیر بدست می آید . این خازن

و سلف باید یک تشکیل یک فیلتر بدهند :

$$125000 = \frac{1}{6.28\sqrt{LC}}$$

عدد 125 کیلو مقدار فرکانس ورودی یا همان فرکانس تشدید می باشد . مقدار L نیز ضریب خود القایی سیم پیچ اتن است .

Rv:مقاومت محدود کننده ولتاژ ورودی دمو دلاتور می باشد و مقدار آن برابر مقاومت پایه rx و زمین است (باید با اهم متر

مقاومت پایه rx و زمین اندازه گرفته شود)

مقدار کریستال نوسان ساز نیز معمولاً 8 مگا هرتز است (در هنگام استفاده از میکرو avr به عنوان پردازنده) و دو خازن متصل

به نیز وظیفه حذف نویز را برعهده دارند و معمولاً مقدار آنها 30 پیکو فاراد می باشد .

توضیحات بیشتر در دیتا شیت قطعه موجود است .

■ راه اندازی چیپ HTRC110 در محیط بسکام :

این چیپ با دستور زیر پیکربندی میشود :

CONFIG HITAG = prescale, TYPE=tp, DOUT = dout, DIN=din , CLOCK=clock, INT=int

Prescale : مقدار دقت اندازه گیری تایمر 0 را مشخص میکند ، میکرو avr برای شمارش و ایجاد کلاک به تایمر 0 نیاز دارد .

مقدار prescale میتواند 0 تا 256 باشد . (در فرکانس 8 مگا هرتز)

TYPE=tp: مشخص کننده نوع چیپ می باشد که در این مورد چیپ استفاده شده ، HTRC110 است.

DOUT = dout : نام پایه ای است که به پین dout چیپ HTRC110 متصل میشود . این پایه اطلاعات خروجی HTRC110 را

به میکرو وارد میکند .

DIN=din : نام پایه ای است که به پین din چیپ HTRC110 متصل میشود . این پایه اطلاعات خروجی میکرو را به HTRC110

منتقل میکند .

CLOCK=clock : نام پایه ای است که به پین clock چیپ HTRC110 متصل میشود . در صورتی که به ارتباط دقت کرده

باشید ، این چیپ از رابط spi نرم افزاری برای انتقال داده استفاده میکند . در این حالت میکرو به عنوان مستر و چیپ به عنوان

اسلیو عمل خواهد کرد .

INT=int : نام وقفه ای است که برای بیدار کردن میکرو استفاده میشود . نیازی به اتصال این پایه به میکرو نیست .

دستور READHITAG :

این دستور به فرم زیر است و توسط آن میتوان شماره سریال تگ را خواند و از وجود یا عدم وجود آن آگاهی یافت :

READHITAG(var)

result = READHITAG(var)

result : یک متغیر عددی (از جنس بیت ، بایت یا ...) میباشد که در صورت عدم وجود تگ و شماره سریال مقدار آن صفر میباشد . در صورتی که شماره سریال وجود داشته باشد ، مقدار این متغیر 1 است و شماره سریال در متغیر var ذخیره میشود.

دستور _checkhitag :

با این دستور ، تگ چک میشود و در صورتی که اطلاعات و بیت های خطا و کلاک و ... صحیح باشند ، اجازه خروج داده صادر میگردد ، این دستور به فرم زیر است :

Call _checkhitag

■ متغیر های رزرو شده برای HTRC110 :

در بسکام تعدادی متغیر برای HTRC110 تعیین شده است که با پیکربندی چیپ مقادیر داده به صورت خود کار در آنها قرار داده میشود و شما میتوانید از آنها استفاده نمایید :

نام متغیر	عمل کرد
_htr_statemachine	وضعیت دستگاه در این متغیر ذخیره می شود
_htcbit	این متغیر مانند یک حافظه موقت داده را در خود نگه میدارد .
_htcbitcount	تعداد شماره سریال دریافتی در این متغیر ذخیره میشود.
_htcmpulse	تعداد پالس های کلاک در این متغیر ذخیره میشود .
_htr_pulse_state	وضعیت پالس های دریافتی در این متغیر ذخیره میشود
_htc_retries	تعداد سریال های آسیب دیده در این متغیر ذخیره میشود (سریال های ناموفق)
_tagdelta	مقدار تاخیر زمانی که بین عبور دو عدد تگ وجود دارد در این متغیر ذخیره میشود
_tagtime	این متغیر مقدار شماره شده توسط تایمر یک ، در زمان ورود داده را در خود ذخیره میکند.
_taglasttime	مقدار زمانی که از آخرین داده دریافتی گذشته در این متغیر ذخیره می شود.
_tagparbit	بیت های توازن (تشخیص خطا) در این متغیر ذخیره میشود
_tagdata	آخرین داده ، که همان شماره سریال تگ میباشد در این متغیر ذخیره میشود
_tagid	آخرین داده ، که همان شماره سریال تگ میباشد در این متغیر ذخیره میشود

متغیر های که در بالا معرفی شدند ، صرفاً برای یافتن خطا ها و دادن گزارش استفاده میشوند ، مثلاً شما میتوانید با استفاده از متغیر `_tagdelta` ، محدود زمانی که هر وسیله از فروشگاه خارج میشود را مشخص کنید و

مثال :

```
$regfile = "m88def.dat"           ' specify chip
$crystal = 8000000                ' used speed
$baud = 19200                     'baud rate

'Notice that the CLOCK OUTPUT of the micro is connected to the clock input of the HTRC110
'PORTB.0 of the Mega88 can optional output the clock. You need to set the fusebit for this option
'This way all parts use the Mega88 internal oscillator
'The code is based on Philips(NXP) datasheets and code. We have signed an NDA to get the 8051 code
'You can find more info on Philips website if you want their code

Print "HTC110 demo"

Config Hitag = 64 , Type = Htrc110 , Dout = Pind.2 , Din = Pind.3 , Clock = Pind.4 , Int = @int0
'      ^ use timer0 and select prescale value 64
'      ^ we used htrc110 chip
'      ^-- dout of HTRC110 is connected to PIND.2 which will be set to input mode
'      ^ DIN of HTRC100 is connected to PIND.3 which will be set to output mode
'      ^clock of HTRC110 is connected to PIND.4 which is set to output mode
'      ^ interrupt

'the config statement will generate a number of constante and internal variables used by the code
'the htrc110.libx library is called

Dim Tags(5) As Byte              'each tag has 5 byte serial
Dim J As Byte                    ' a loop counter

'you need to use a pin that can detect a pin level change
'most INT pins have this option
'OR , you can use the PCINT interrupt that is available on some chips
'In case you want PCINT option
' Pcmsk2 = &B0000_0100          'set the mask to ONLY use the pin connected to DOUT
' On Pcnt2 Checkints            'label to be called
' Enable Pcnt2                  'enable this interrupt

'In case you want to use INT option
On Int0 Checkints                ' PIND.2 is INTO
Config Int0 = Change              'you must configure the pin to work in pin change intertupt mode
Enable Interrupts                ' enable global interrupts

Do
If Readhitag(tags(1)) = 1 Then     'check if there is a new tag ID
  For J = 1 To 5                 'print the 5 bytes
    Print Hex(tags(j)) ; ",";
  Next
```

Else 'there was nothing

Print "Nothing"

End If

Waitms 500 'some delay

Loop

'this routine is called by the interrupt routine

Checkints:

Call _checkhitag 'you must call this label

'you can do other things here but keep time to a minimum

Return

پروتکل TCP/IP

TCP/IP، (Internet Protocol / Transmission Control Protocol) یکی از مهمترین پروتکل های استفاده شده در شبکه های کامپیوتری است. اینترنت بعنوان بزرگترین شبکه موجود، از پروتکل فوق بمنظور ارتباط دستگاه های متفاوت استفاده می نماید. در ادامه به بررسی فرآیند انتقال اطلاعات، معرفی شبکه های موجود، نحوه استفاده از سوکت برای ایجاد تمایز در ارتباطات، چیپ W3100، ماژول IIM7000 و ماژول IIM7010 و نحوه ی اتصال آنها به AVR و شبکه خواهیم پرداخت.

شبکه چیست؟

شبکه مجموعه ای از سرویس دهنده ها و سرویس گیرنده های متعددی می باشد که به یکدیگر متصل هستند. در این بین سرویس دهنده ها (server) نقش سرویس دهنده و خدمات دهی و سرویس گیرنده ها (Client) نقش سرویس گیرنده یا همان مشتری را بازی می کنند.

انواع شبکه: شبکه ها را می توان به دو دسته ی «شبکه های محلی» LAN و شبکه های بزرگ تر از آن (WAN) تقسیم کرد.

شبکه های محلی: (Local Area Network)

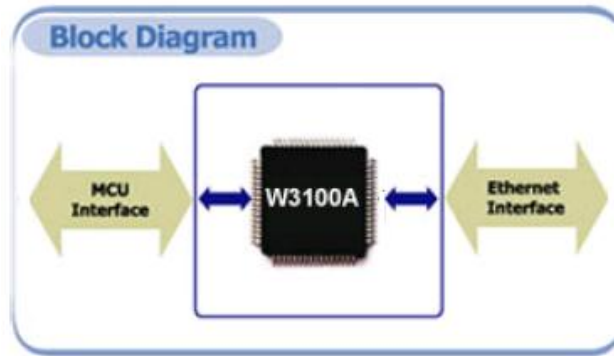
این نوع شبکه ها به شبکه های (LAN) معروف هستند. شبکه های محلی معمولاً میزبان 2 تا 20 کامپیوتر و در غالب Work Group میباشند. سرعت این نوع شبکه بسیار زیاد است (معمولاً 100 MB Per Sec) و می توان حجم داده های بالا را در مدت بسیار کم انتقال داد. نمونه ای از این شبکه ها در کافی نت ها، دانشگاه ها و... وجود دارد.

شبکه های گسترده: (Wide Area Network)

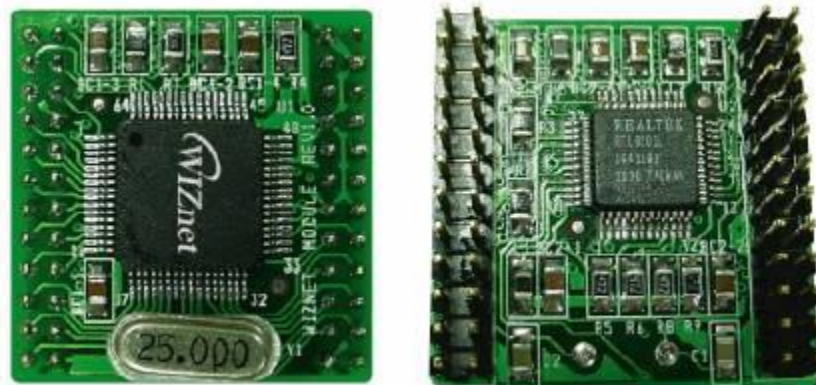
این نوع شبکه ها به شبکه های WAN معروف هستند. این شبکه ها بزرگتر از شبکه های LAN و اغلب برای امور عمومی از آن استفاده می شود. از جمله این شبکه ها میتوان شبکه های VAN و یا شبکه های بزرگتر مانند Internet و.. را نام برد. سرعت انتقال داده ها در این نوع شبکه ها نسبت به LAN کمتر میباشد.

چیپ W3100 چیست ؟

در شبکه های کامپیوتری به دلیل نرخ بالای انتقال داده و زیاد بودن تعداد بیت های دریافتی / ارسالی، میکروکنترلر های 8 بیتی، به صورت مستقیم قادر به دریافت و پردازش اطلاعات نمیشوند، به همین دلیل نیاز است تا بین این میکروکنترلر ها و شبکه، ماژول های خاصی قرار داده شود.



W3100A یک چیپ پر قدرت ساخت کمپانی wiznet میباشد ، این چیپ میتواند داده سریال پورت شبکه را به داده موازی برای میکرو کنترلر تبدیل کند . شرکت wiznet برای راحتی کاربران این قطعه را در یک ماژول ارائه کرده است . برای کار با این ماژول فقط نیاز به اعمال تغذیه و متصل کردن آن به میکرو و پورت شبکه میباشد . کلیه توابع مربوط به کار با این چیپ و ماژول در برنامه بسکام وجود دارد . شما میتوانید با استفاده از این ماژول کار های زیادی ، از جمله اشکار کردن آدرس IP ، ارسال ایمیل ، ارسال داده ، کنترل کردن سرور ها ، کنترل کردن کامپیوتر های دیگر و.... را به سادگی انجام دهید .



شما میتوانید دیتاشیت این ماژول را از آدرس زیر دانلود نمایید :

http://www.wiznet.co.kr/Sub_Modules/en/product/Product_Detail.asp?cate1=&cate2=&cate3=&pid=1006&cType=2

<http://www.mcselec.com/index.php?option=content&task=view&id=18>

نام پایه ها :

شماره پایه	نام پایه	I/O	توصیف
1,48	3.3V	Power	ولتاژ تغذیه مثبت 3.3 ولت
2	INT	O	خروجی سیگنال وقفه برای فعال سازی میکرو

3	WR	I	سیگنال نوشتن
4	RD	I	سیگنال خواندن
5	CS	I	پالس انتخاب کننده ماژول
6	RESET	I	باز نشانی ماژول با پالس بالا رونده
7,10,9,12,11,14,15~23	A14~A9,A8~A0	I	خطوط آدرس
8,13,24,25,28,31,37,38,47	GND	Ground	گراند
26	RESET	I	بازنشانی ماژول با سیگنال پایین رونده
27	TPTX+	O	خروجی داده به شبکه
29	TPTX-	O	خروجی داده به شبکه
33	TPRX+	I	ورودی داده از شبکه
35	TPRX-	I	ورودی داده از شبکه
45,46,43,44,41,42,39,40	D7~D0	I/O	خطوط داده
30	COL_LED	O	نمایش گر وجود داده
32	100_LED	O	نمایش گر نرخ انتقال بالا (100M)
34	10_LED	O	نمایش گر نرخ انتقال پایین (10M)
36	LINK_LED	O	نمایشگر وجود خط

همانطور که قبلا نیز بیان شد ، این ماژول فقط داده سریال از پورت شبکه را به داده موازی برای میکرو تبدیل میکند ، همچنین داده موازی میکرو را به صورت سریال بر روی پورت میفرستد . پایه های WR و RD مشخص کننده دریافت یا ارسال داده خواهند بود . این ماژول فقط به میکرو کنترلر های که دارای پورت آدرس هستند (نظیر ATMEGA161L و ...) میتواند متصل شود .

توابع راه اندازی W3100/IIM7000/IIM7010 در بسکام :

دستورات کار با ماژول IIM7000 در نگاه اول ممکن است اندکی پیچیده به نظر برسد ، دلیل این پیچیدگی چیزی جز عدم آشنایی شما با شبکه های کامپیوتری نیست . در صورتی که احساس میکنید در درک عمل کرد دستورات مشکل دارید ابتدا مثال های آورده شده را بخوانید . (ارسال ایمیل ، خواندن ایمیل ، وب سرور) . برای کار با ماژول های ذکر شده از کتابخانه ی tcpip.lib استفاده میشود ، از آنجا که برای کار با شبکه پارمتر ها و متغیر های زیادی وجود دارد ، کامپایلر

بسکام برای پیکربندی این ماژول ها تنظیمات پیشفرضی را در نظر گرفته است که این تنظیمات باید قبل از پیکربندی ماژول در برنامه آورده شود، در زیر دستورات مربوطه آورده شده است :

```
$lib "tcpip.lib" \ specify the name of the tcp ip lib

`W3100A constants
Const Sock_stream = $01 \ Tcp
Const Sock_dgram = $02 \ Udp
Const Sock_ip1_raw = $03 \ Ip Layer Raw Sock
Const Sock_mac1_raw = $04 \ Mac Layer Raw Sock
Const Sel_control = 0 \ Confirm Socket Status
Const Sel_send = 1 \ Confirm Tx Free Buffer Size
Const Sel_recv = 2 \ Confirm Rx Data Size

`socket status
Const Sock_closed = $00 \ Status Of Connection Closed
Const Sock_arp = $01 \ Status Of Arp
Const Sock_listen = $02 \ Status Of Waiting For Tcp Connection Setup
Const Sock_synsent = $03 \ Status Of Setting Up Tcp Connection
Const Sock_synsent_ack = $04 \ Status Of Setting Up Tcp Connection
Const Sock_synrecv = $05 \ Status Of Setting Up Tcp Connection
Const Sock_established = $06 \ Status Of Tcp Connection Established
Const Sock_close_wait = $07 \ Status Of Closing Tcp Connection
Const Sock_last_ack = $08 \ Status Of Closing Tcp Connection
Const Sock_fin_wait1 = $09 \ Status Of Closing Tcp Connection
Const Sock_fin_wait2 = $0a \ Status Of Closing Tcp Connection
Const Sock_closing = $0b \ Status Of Closing Tcp Connection
Const Sock_time_wait = $0c \ Status Of Closing Tcp Connection
Const Sock_reset = $0d \ Status Of Closing Tcp Connection
Const Sock_init = $0e \ Status Of Socket Initialization
Const Sock_udp = $0f \ Status Of Udp
Const Sock_raw = $10 \ Status of IP RAW
```

برای مثال در کتابخانه tcpip.lib دستور Sock_closed به مفهوم بسته شدن کانکشن استفاده شده است، در پیکربندی سخت افزاری هنگامی که کانکشن بسته میشود، ماژول کد \$00 را به میکروکنترلر ارسال میکند.

با استفاده از این تنظیمات میتوانید ماژول W3100/IIM7000/IIM7010 را برای کار در پروتکل های TCP/IP و UDP (تحت TCP/IP) استفاده کنید.

این ماژول با دستور زیر راه اندازی میشود :

```
CONFIG TCP/IP = int , MAC = mac , IP = ip, SUBMASK = mask, GATEWAY = gateway, LOCALPORT= port, TX= tx, RX= rx
, NOINIT= 0|1 , TWI=address , Clock = speed [, baseaddress = address] [,TimeOut=tmOut]
```

Int: مشخص کننده پایه ی وقفه ای است که به پایه int ماژول متصل میشود. شما میتوانید از int0 یا int1 به جای این واژه استفاده کنید.

Mac: یک آدرس 6 بیتی میباشد که به ماژول اختصاص داده شده است. این آدرس شناسه ماژول در شبکه است، مثلاً آدرس 123.00.12.34.56.78 مربوط به یک نمونه ماژول w3100 است، در حالی دیگر ماژول w3100 دارای آدرس مستقل دیگری میباشد.

ip: برای کار با w3100 باید به آن یک آدرس ip اختصاص داد، آدرس ip میتواند یک آدرس 4 بیتی شبیه 192.168.0.10 باشد. با اتصال ماژول به شبکه، ماژول با این ip شناسی میشود و اطلاعات به این ip ارسال یا از آن دریافت میگردد.

SUBMASK: SUBMASK عددی 4 بیتی است که در شبکه های کامپیوتری برای مشخص کردن Net ID (شناسه یوزر های متصل به شبکه) و Host ID (شناسه ی هاست موجود در شبکه) به کار میرود. این عدد به صورت پیش فرض برای اتصال ماژول به یک کامپیوتر 255.255.255.0 است، در صورتی که قصد دارید این ماژول را به شبکه ای با چند کامپیوتر متصل کنید، با تایپ کردن دستور ipconfig در محیط cmd کامپیوتر یکی از کامپیوترها SUBMASK شبکه را بدست آورید.

GATEWAY: آدرس درگاه ورودی سیستم میباشد که میتوان با تایپ کردن دستور ipconfig در محیط cmd کامپیوتر مورد نظر آن را مشاهده کرد. برای مثال 192.168.0.1 درگاه یک سیستم کامپیوتری بر Ethernet adapter Local Area Connection 2: یک کامپیوتر در ویندوز 7 است، ما قصد داریم ماژول را به این کامپیوتر متصل کنیم.

LOCALPORT: عددی تا 5000 است که به پورت محلی ماژول اختصاص داده میشود، برای مثال در شبکه های کامپیوتری پورت 80 برای مشاهده ی صفحات وب استفاده میشود، در صورتی که کاربری قصد داشته باشد صفحه ی وبی را در کامپیوتر مذکور از طریق شبکه (توسط یک سیستم دیگر) مشاهده کند باید از پورت 80 استفاده کند.

TX: یک عدد یک بیتی است که اندازه ی بافر ارسال داده را تعیین میکند، این ماژول دارای 4 سوکت میباشد که عدد 00 مقدار 1024 بایت و عدد 01 مقدار 2048 بایت و 10 مقدار 4096 و عدد 11 مقدار 8192 بایت را به این بافر اختصاص میدهد.

برای مثال فرض کنید شما میخواهید از 4 سوکت در برنامه استفاده کنید، در این حالت باید چهار بافر داده با اندازه ی 2048 را برای کامپایلر معرفی کنید، در این حالت عدد 01010101 باید به جای TX در فرمت هگز قرار گیرد. چون این دستور برای پیکربندی IDE بسکام استفاده میشود، هنگام مقدار دهی باید در ابتدای رقم حرف \$ را قرار گیرد (این عدد به کتابخانه ارسال میشود) (Tx = \$55)

RX: یک عدد یک بیتی است که اندازه ی بافر دریافت داده را تعیین میکند، این ماژول دارای 4 سوکت میباشد که عدد 00 مقدار 1024 بایت و عدد 01 مقدار 2048 بایت و 10 مقدار 4096 و عدد 11 مقدار 8192 بایت را به این بافر اختصاص میدهد.

برای مثال فرض کنید شما میخواهید از 1 سوکت در برنامه استفاده کنید، در این حالت باید یک بافر داده با اندازه ی 8192 را برای کامپایلر معرفی کنید، در این حالت عدد 11 باید به جای TX در فرمت هگز قرار گیرد. چون این دستور برای پیکربندی IDE بسکام استفاده میشود، هنگام مقدار دهی باید در ابتدای رقم حرف \$ را قرار گیرد (این عدد به کتابخانه ارسال میشود) ($Rx = \$3$)

Noinit: در صورتی که قصد دارید تنظیمات مربوط به TCP و MAC و Subnetmask را در داخل برنامه انجام دهید، یا تنظیمات فعلی مربوط به این دستورات را تغییر دهید، به جای این دستور عدد 1 قرار دهید، این حالت برای مواردی مفید است که ماژول در شبکه های مختلفی استفاده شود، در این حالت کاربر میتواند در برنامه شرایط تغییر دستورات ذکر شده را ایجاد کند (مثلا یک کلید معرفی کند که با فشردن آن MAC مقدار دیگری را بگیرد).

TWI: آدرس TWI میکرو کنترلر اسلیو متصل شده به ماژول. (برای ماژول های که باس TWI دارند)

Clock: مقدار فرکانس کلاک باس TWI با این دستور مشخص میشود. (برای ماژول های که باس TWI دارند)

Baseaddress: آدرس انتخاب ماژول در باس TWI میباشد، این آدرس به صورت پیشفرض H8000 است.

TimeOut: رقمی اختیاری است که برای زمان TimeOut به کار میرود، با ارسال داده به ماژول برنامه متوقف شده و تا زمان TimeOut منتظر می ماند، در صورتی که بعد از سپری شدن این زمان داده ای منتقل نشود برنامه متوقف می گردد.

دستورات مربوط به کار با ماژول

Result = GETSOCKET(socket, mode, port, param)

دستور بالا عملیات اتصال ماژول به یک سوکت را انجام میدهد. با مقدار دهی mode با اعداد 1 تا 4 میتوان پروتکل ارتباطی ماژول را مشخص کرد، عدد یک برای پروتکل TCP/IP و عدد 2 برای پروتکل UDP استفاده میشود، شما همچنین میتوانید از دستورات sock_stream و sock_dgrm نیز به جای اعداد استفاده کنید.

در این دستور port، پورت محلی است که قصد اتصال به آن را دارید، در صورتی که قبلاً دستور LOCALPORT در پیکربندی TCP/IP را مقدار دهی کرده اید، به جای این دستور عدد صفر را قرار دهید.

همچنین Param برای پیکربندی پارامتر های داخلی ماژول استفاده میشود، برای استفاده از پارامتر های پیشفرض عدد 0 را به جای آن قرار دهید.

در شبکه های کامپیوتری داده های مختلفی در حال انتقال است، فایل های به اشتراک گذاشته شده، درخواست ها، خطا ها، ایمیل ها و ... نمونه های از این داده هستند. در این شبکه ها سوکت ارتباط بین درخواست کننده (کلاینت) و اجرا کننده (سرور) در لایه های پایین شبکه را برقرار کرده و پروتکل (UDP یا TCP/IP) نحوه ی انتقال داده را مشخص

میکند، در این دستور در صورتی که سوکت 0 تا 3 برای انتقال داده بر روی پورت Param با استفاده از پروتکل mode وجود داشته باشد، متغیر Result مقداری بین 0 تا 3 و در صورتی وجود هر گونه خطا (موجود نبودن سوکت، IP، آدرس MAC و...) مقدار 255 میگیرد.

Result = SOCKETCONNECT(socket, IP, port)

بعد از موفقیت آمیز بودن دریافت سوکت (255 نبودن Result) میتوان به سوکت مربوطه با دستور بالا متصل شد، در این دستور socket شماره ی سوکت و IP شماره ی IP سیستمی که قصد اتصال به آن را داریم و port شماره پورتی است که قصد داریم اطلاعات را از طریق آن ارسال کنیم می باشد.

در این دستور نیز در صورت موفقیت آمیز بودن اتصال متغیر Result دارای مقدار صفر و در صورت بروز هر مشکلی، دارای مقدار 1 خواهد شد.

Result = SOCKETSTAT(socket , mode)

بعد از اتصال به سوکت با دستور SOCKETSTAT میتوان اطلاعات مربوط به سوکت را دریافت کرد، در این دستور socket شماره ی سوکت و mode عددی بین 1 تا 3 است:

- SEL_CONTROL یا 0 : مقدار مربوط به رجیستر های وضعیت را بر میگرداند.
- SEL_SEND یا 1 : تعداد بایت های که در بافر ارسال داده قرار گرفته اند را بر میگرداند.
- SEL_RECV یا 2 : تعداد بایت های که در بافر دریافت داده ذخیره شده اند را بر میگرداند.

با انتخاب عدد صفر یا دستور SEL_CONTROL به جای mode یکی از مقادیر زیر در متغیر Result ذخیره میشود:

0	SOCK_CLOSED	Connection closed
1	SOCK_ARP	Standing by for reply after transmitting ARP request
2	SOCK_LISTEN	Standing by for connection setup to the client when acting in passive mode
3	SOCK_SYNSENT	Standing by for SYN,ACK after transmitting SYN for connecting setup when acting in active mode
4	SOCK_SYNSENT_ACK	Connection setup is complete after SYN,ACK is received and ACK is transmitted in active mode
5	SOCK_SYNRCV	SYN,ACK is being transmitted after receiving SYN from the client in listen state, passive mode
6	SOCK_ESTABLISHED	Connection setup is complete in active, passive mode
7	SOCK_CLOSE_WAIT	Connection being terminated
8	SOCK_LAST_ACK	Connection being terminated
9	SOCK_FIN_WAIT1	Connection being terminated

10	SOCK_FIN_WAIT2	Connection being terminated
11	SOCK_CLOSING	Connection being terminated
12	SOCK_TIME_WAIT	Connection being terminated
13	SOCK_RESET	Connection being terminated after receiving reset packet from peer.
14	SOCK_INIT	Socket initializing
15	SOCK_UDP	Applicable channel is initialized in UDP mode.
16	SOCK_RAW	Applicable channel is initialized in IP layer RAW mode
17	SOCK_UDP_ARP	Standing by for reply after transmitting ARP request packet to the destination for UDP transmission
18	SOCK_UDP_DATA	Data transmission in progress in UDP RAW mode
19	SOCK_RAW_INIT	W3100A initialized in MAC layer RAW mode

در صورتی که دستور Sock_established برگردانده شود، کاربر میتواند دستورات بعدی را به سرور ارسال کند.

Result = TCPREAD(socket , var , bytes)

در این دستور socket شماره ی سوکت و var متغیری از نوع String یا بایت برای ذخیره ی داده ی خواند شده و bytes تعداد بایت های خواند شده است (در صورتی که داده در فرمت String نباشد).

Result = TCPWRITE(socket , var , bytes)

Result = TCPWRITE(socket , EPROM, address , bytes)

در این دستور socket شماره ی سوکت، var متغیری از جنس string یا سایر متغیر های عددی و bytes مشخص کننده ی تعداد بایت های ارسالی میباشد. در شکل دیگر دستور EPROM مشخص کننده ی ارسال داده از EPROM میکرو کنترل و address آدر داده ی مورد نظر است.

در این دستور نیز در صورت موفقیت آمیز بودن اتصال متغیر Result دارای مقدار صفر و در صورت بروز هر مشکلی، دارای مقدار 1 خواهد شد.

Result = TCPWRITESTR(socket , var , param)

از این دستور برای ارسال داده های string (رشته) به سوکت استفاده میشود، در این دستور socket شماره ی سوکت و Var نام متغیری است که داده های string در آن ذخیره شده و param دستوری اختیاری است که میتواند مقدار 0 یا 255 داشته باشد، در این دستور عدد صفر تنها رشته ی string را ارسال میکند و عدد 255 باعث اضافه شدن دستور CR + LF به رشته و ارسال آن خواهد شد. در مثال ها توضیحات بیشتری در این مورد آورده شده است.

در این دستور Result یک متغیر از جنس word است که تعداد بایت های نوشته شده در سوکت را در خود ذخیره میکند، در صورتی که تعداد بایت ها از فضای موجود در متغیر بیشتر باشد، Result برابر با 0 خواهد شد.

SOCKETLISTEN socket

در صورتی که قصد داشته باشید از میکرو کنترلر و ماژول به عنوان سرور استفاده کنید، با استفاده از این دستور میتوانید یک سوکت ایجاد نمایید، در این دستور میتوانید به جای سوکت اعداد 0 تا 3 را قرار دهید. بعد معرفی شماره ی سوکت، سوکت مطابق تنظیمات دستور GetSocket آماده ی ارسال و دریافت داده خواهد شد. در مثال ها توضیحات بیشتری آورده شده است.

CloseSocket socket [, prm]

بعد از ارسال و دریافت اطلاعات، سوکت باید بسته شود (تا سایر دستگاه ها نیز بتوانند از آن استفاده کنند). در این دستور socket شماره ی سوکتی است که قبلاً باز شده است و prm یک متغیر اختیاری است که به جای آن میتوانید اعداد 1 و 2 و 3 را قرار دهید:

- 0- در صورتی که نیازی به دستورات اختیاری ندارید، prm را پاک کنید یا به جای آن عدد 0 قرار دهید
- 1- با قرار دادن عدد 2، بافر ارسال داده چک میشود و در صورت عدم وجود داده در آن سوکت بسته میشود
- 2- با قرار دادن عدد 2، دستور Close به سوکت ارسال میشود و بسته شدن سوکت چک می گردد، در صوتی که سوکت هنوز باز باشد، اجرای برنامه متوقف میگردد.

مثال:

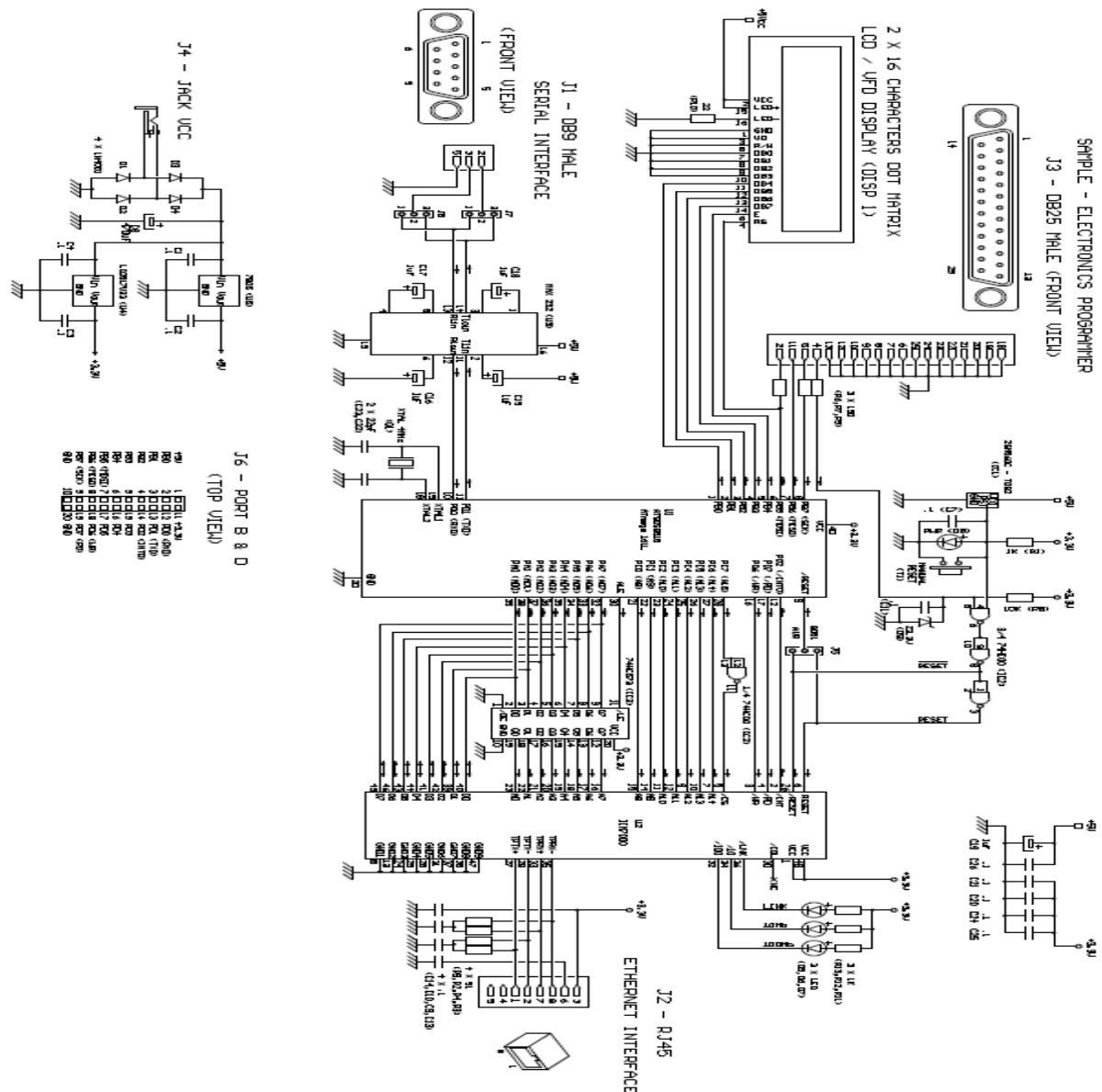
در این مثال میکرو کنترلر با استفاده از پروتکل TCP/IP به یک وب سرور تبدیل شده است که کاربر میتواند با مراجعه به آدرس های زیر صفحات index.htm و post.htm و notfound.htm را در مرور گر خود مشاهده کند:

<http://213.84.110.248:5000/index.htm>

<http://213.84.110.248:5000/post.htm>

<http://213.84.110.248:5000/notfound.htm>

شماتیک مورد نیاز برای این پروژه در صفحه ی بعد آورده شده است. (شماتیک در فرمت PDF با کیفیت بالا تر را از [این آدرس](#) دانلود کنید). این پروژه توسط شرکت MCS ELECTRONIC اجرا شده و همکنون سخت افزار آن به شبکه ی اینترنت متصل است.



```
$regfile = "m16l1def.dat"
$crystal = 3579545
$baud = 19200
$lib "tcpip.lib"
```

در بسکام کلیه پیکربندی های سخت افزاری مازول توسط کامپایلر (محیط IDE) انجام میشود ، برای دستیابی کاربر به برخی از تنظیمات ، دستورات زیر از کتابخانه به محیط برنامه آورده شده است . این دستورات مازول را به صورت پیشفرض برای پروتکل TCP/IP پیکربندی میکند ، در زیر مقادیر عددی به واژه های متناظر نسبت داده شده است :

```
Const Sock_stream = $01 ' Tcp
Const Sock_dgram = $02 ' Udp
Const Sock_ip1_raw = $03 ' Ip Layer Raw Sock
Const Sock_mac1_raw = $04 ' Mac Layer Raw Sock
Const Sel_control = 0 ' Confirm Socket Status
Const Sel_send = 1 ' Confirm Tx Free Buffer Size
Const Sel_recv = 2 ' Confirm Rx Data Size
```

بعد از برقراری ارتباط میکروکنترلر و مازول ، مازول برای برقراری ارتباط با شبکه آماده است ، در این حالت کاربر با

دستوراتی که در ادامه آورده شده است ماژول را به شبکه متصل میکند، در هنگام اتصال به شبکه وضعیت های مختلفی وجود دارد به عنوان مثال ارسال کد 00 از طرف ماژول به میکروکنترلر به معنای بسته شدن کانکشن است، در زیر این کدها به واژه های متناظر توسط دستور Const نسبت داده شده اند، از این به بعد Sock_closed معادل 00\$ است. اگر در شبکه ای، کد معادل بسته شدن کانکشن چیزی غیر از 00 باشد، باید آن را به جای این دستور درج کرد.

```
'socket status
Const Sock_closed = $00 ' Status Of Connection Closed
Const Sock_arp = $01 ' Status Of Arp
Const Sock_listen = $02 ' Status Of Waiting For Tcp Connection Setup
Const Sock_synsent = $03 ' Status Of Setting Up Tcp Connection
Const Sock_synsent_ack = $04 ' Status Of Setting Up Tcp Connection
Const Sock_synrecv = $05 ' Status Of Setting Up Tcp Connection
Const Sock_established = $06 ' Status Of Tcp Connection Established
Const Sock_close_wait = $07 ' Status Of Closing Tcp Connection
Const Sock_last_ack = $08 ' Status Of Closing Tcp Connection
Const Sock_fin_wait1 = $09 ' Status Of Closing Tcp Connection
Const Sock_fin_wait2 = $0a ' Status Of Closing Tcp Connection
Const Sock_closing = $0b ' Status Of Closing Tcp Connection
Const Sock_time_wait = $0c ' Status Of Closing Tcp Connection
Const Sock_reset = $0d ' Status Of Closing Tcp Connection
Const Sock_init = $0e ' Status Of Socket Initialization
Const Sock_udp = $0f ' Status Of Udp
Const Sock_raw = $10 ' Status of IP RAW
```

برای دیباگ کردن برنامه و مشاهده ی داده های ارسالی و دریافتی دستورات زیر به برنامه افزوده شده است، با اتصال پورت J1 به پورت Com کامپیوتر میتوانید رشته ها و متغیر های که با دستور print ارسال میشود را در محیط terminal Emulator مشاهده کنید، برای کسب اطلاعات بیشتر در این مورد به بخش پروتکل های ارتباطی و rs232 مراجعه کنید.

```
$eepleave ' do not delete the EEP file since we generated it with the converter tool
#if Debug
Print "init W3100A"
#endif
Enable Interrupts
Config TcpiP = Int0 , Mac = 00.00.12.34.56.78 , Ip = 192.168.0.10 , Submask = 255.255.255.0 ,
Gateway = 192.168.0.1 , Localport = 1000 , Tx = $55 , Rx = $55
```

با راه اندازی ماژول هنگامی که داده یا درخواستی به آن از سمت شبکه ارسال میشود، ماژول پایه ی int خود را فعال میکند تا میکروکنترلر از وجود داده با خبر شود، برای استفاده از وقفه فعال سازی وقفه ی سراسری الزامی است، در این پروژه داده های مربوط به صفحات در حافظه ی EEPROM میکروکنترلر ذخیره شده است. برای درک بهتر عمل کرد شبکه های کامپیوتری، آنها را مانند شبکه های مخابراتی تصور کنید، در شبکه های ارتباطی تلفن ویژگی های زیر وجود دارد:

- 1- هر فرد برای اتصال به شبکه ی تلفن به یک گوشی و یک خط تلفن نیاز دارد = برای ارتباط با شبکه نیاز به یک مودم (ماژول IIM7000) و یک پردازشگر برای پردازش داده (میکروکنترلر) است.
- 2- هر فرد در شبکه ی مخابرات دارای یک شماره ی منحصر بفرد است، فرد در شبکه با این شماره شناخته میشود = هر دستگاهی که به شبکه متصل میشود باید دارای یک Mac اختصاصی باشد تا در شبکه با آن شناخته شود.
- 3- با برقراری ارتباط در شبکه ی مخابرات (گرفتن یک شماره)، اپراتور مخابرات پلی را برای ارتباط ایجاد میکند، در یک سمت این پل گیرنده ی تماس و در سمت دیگر، پذیرنده ی تماس قرار میگیرد.. = در شبکه ها کامپیوتری برای ایجاد ارتباط، برای هر سیستم به یک ip نیاز داریم، برای مثال کامپیوتر شماره ی یک دارای IP به شماره ی 192.168.0.10 و کامپیوتر شماره ی 2 دارای IP به شماره ی 192.168.0.11 است، وقتی که بخواهیم از سیستم یک، فایلی را به سیستم 2 ارسال کنیم، باید آن را به IP شماره ی 192.168.0.11 بفرستیم.

در حالت بالا از روی آدرس Mac میتوان تشخیص داد که IP موجود مربوط به کدام کامپیوتر است .
در ارتباط تلفنی شماره ی تلفن مشابه ی آدرس های مک و درگاه های مسیر ایجاد مشابه آدرس های IP هستند . با برقراری ارتباط یک کانال صوتی با دو درگاه (دو آدرس IP) برای دو طرف ایجاد میگردد که کاربران میتوانند داده های صوتی خود را از طریق آن ارسال کنند .

4- در شبکه ی مخابراتی مشترکان در گروه های دسته بندی شده اند ، این دسته بندی به صورت محله ای یا خیابانی اعمال شده است ، برای مثال در شماره تلفن های خیابان X سه رقم اول مشترک است . در شبکه های کامپیوتری نیز اگر تعداد یوزر ها زیاد باشد ، تعداد IP به اندازه ی کافی وجود نخواهد داشت به همین دلیل از Submask برای دسته بندی کاربران استفاده میشود .

5- در شبکه های مخابراتی مختلف ، ممکن است پروتکل های ارتباطی ، کدهای شماره تلفن و ... متفاوت باشد ، به عنوان مثال فرض کنید قصد تماس با یکی از دوستان خارج از کشور را دارید در این حالت باید از طریق یک درگاه به شبکه ی مخابرات کشور مورد نظر متصل شوید = در صورتی که قصد داشته باشیم شبکه ی محلی خود را به شبکه ای بزرگتر متصل کنیم ، از Gateway استفاده میکنیم .

6- بر روی خطوط تلفن ، سرویس های دیگری نیز ارائه میشود ، سرویس مکالمه ی صوتی ، سرویس تلفن بانک ، سرویس دیال آپ ، سرویس فکس و ...) ، در تمامی این سرویس ها خطوط انتقال داده یکی بوده و پروتکل انتقال داده تفاوت دارد = در شبکه های کامپیوتری نیز کار های مختلفی میتوان انجام داد ، انتقال فایل ، مشاهده ی صفحات وب ، ارسال و دریافت ایمیل و ... برای انجام دادن هر یک از این موارد به یک پورت نیاز داریم ، برای مثال جهت خواند ایمیل از پورت 110 استفاده می شود

```
'dim used variables
Dim S As String * 140 At &H100 , Shtml As String * 15 , Sheader As String * 30
Dim Buf(120) As Byte At &H100 Overlay ' this is the same data as S but we can treat it as an array now
Dim Tempw As Word
Dim I As Byte , Bcmd As Byte , P1 As Byte , P2 As Byte , Size As Word
Dim Bcontent As Byte
```

متغیر های مورد نیاز در این پروژه در بالا معرفی شده اند .

```
I = Getsocket(0 , Sock_stream , 5000 , 0) 'get a socket
#If Debug
Print "socket : " ; I
#Endif
```

با دستورات بالا سوکتی برای ارتباط با کامپیوتر از طریق پروتکل TCP/IP ایجاد میشود ، برای ایجاد ارتباط در این پروتکل باید از پورت های استاندارد آن که شماره های 1024 تا 5000 و 6700 تا 6702 و 6880 هستند استفاده کرد که در اینجا از پورت 5000 استفاده شده است . در صورتی که ایجاد سوکت موفقیت آمیز باشد دارای مقداری بین 0 تا 4 خواهد بود ، وجود مقدار 255 در ا به معنای وجود خطا در ارتباط ، پروتکل و .. است . چیپ W3100A دارای 4 سوکت است ، این بدین معناست که این چیپ میتواند به صورت همزمان 4 پروسه را انجام دهند . با هر بار اجرای دستور فوق یکی از سوکت ها برای انجام عملیات های بعدی رزرو میشود و تا اجرای دستور CloseSocket نمیتوان از آن استفاده کرد .

```
Socketlisten I ' server mode

#If Debug
Print "listening"
```

```
#endif
```

اکنون میتوانیم با دستور Socketconnect به سوکت ایجاد شده متصل شویم. در دستور بالا از طریق پورت 5000 به کامپیوتری (یا مودم ADSL دارای پورت شبکه یا هر دستگاه استاندارد دیگر) با شماره ی Submask = 255.255.255.0 و Gateway = 192.168.0.0 متصل میشویم، این کامپیوتر به اینترنت متصل است و ما از طریق آن پورت 110 آدرس 64.77.0.0 را باز میکنیم، در صورتی که تمامی مراحل به درستی انجام شود متغیر ل برابر با 0 خواهد بود.

```
Do
Tempw = Socketstat(i , 0) ' get status
Select Case Tempw
Case Sock_established
```

دستور Socketstat وضعیت سوکت را بررسی میکند، دریافت پیغام Sock_established یا کد \$06 به معنای برقرار بودن اتصال است. هنگامی که شما آدرس www.google.com را در مرورگر خود وارد میکنید از طریق پورت 80 (برای خواند صفحات وب است) به سرور گوگل وارد می شوید. در این حالت گوگل صفحه ی index خود که همان صفحه ی جستجو است را برای شما ارسال میکند. در این پروژه نیز با باز کردن آدرس پورت 110 آدرس 64.77.0.0 سرور باید صفحه ی index خود را به ماژول ارسال کند، اطلاعات سرور به Submask = 255.255.255.0 و Gateway = 192.168.0.0 کامپیوتر با مودم adsl ارسال شده و سپس از طریق آن به آدرسی Ip = 192.168.0.10 سپس Mac = 00.00.12.34.56.78 ارسال میشود

```
#if Debug
Print "sock_est"
#endif
Tempw = Socketstat(i , Sel_recv) ' get received bytes
#if Debug
Print "receive buffer size : " ; Tempw
#endif
If Tempw > 0 Then ' if there is something received
Bcmd = 0
Do
Tempw = Tcpread(i , S) ' read a line
If Left(s , 3) = "GET" Then '
Bcmd = 1 ' GET /index.htm HTTP/1.1
Gosub Page
Elseif Left(s , 4) = "HEAD" Then
Bcmd = 2
Gosub Page
Elseif Left(s , 4) = "POST" Then
Bcmd = 3
Elseif Left(s , 15) = "Content-Length:" Then ' for post
S = Mid(s , 16) : Bcontent = Val(s)
Else
#if Debug
Print S
#endif
End If
Loop Until S = "" ' wait until we get an empty line
Tempw = Tcpwrite(i , "HTTP/1.0 200 OK{013}{010}")
If Bcmd = 3 Then
#if Debug
Print "Posted data"
#endif
Tempw = Tcpread(i , Buf(1) , Bcontent) ' read data
#if Debug
Bcontent = Bcontent + 1
Buf(bcontent) = 0 ' put string terminator at end of data so we can handle it as a string
Print S
#endif
Shtml = "/redirect.htm" ' redirect to www.mcselec.com
End If
Gosub Stuur ' GET or HEAD or POST feedback so send it
```

```

Print "closing socket"
Closesocket I ' close the connection
Print "done"
End If
Case Sock_close_wait
#if Debug
Print "CLOSE_WAIT"
#endif
Closesocket I ' we need to close
Case Sock_closed
#if Debug
Print "CLOSED"
#endif
I = Getsocket(0 , Sock_stream , 5000 , 0) ' get a new socket
Socketlisten I ' listen
#if Debug
Print "Listening on socket : " ; I
#endif
End Select
Loop
End

'get html page out of data
Page:
P1 = Instr(s , " ") ' find first space
P1 = P1 + 1 ' 4
P2 = Instr(p1 , s , " ") ' find second space
P2 = P2 - P1
Shtml = Mid(s , P1 , P2) ' dont use too long page names
Shtml = Lcase(shtml) ' make lower case
#if Debug
Print "HTML page:" ; Shtml
#endif
Return

'send data
Stuur:
Dim Woffset As Word , Bcontenttype As Byte , Wsize As Word , Bgenerate As Bit , Ihitcounter As Integer
Bgenerate = 0 ' by default
Select Case Shtml
Case "/index.htm"
Bcontenttype = 0 : Bgenerate = 1
Case "/redirect.htm"
Bcontenttype = 0 : Bgenerate = 1
Case "/post.htm" : Wsize = 277
bContentType = 0 : wOffset = 0
Case "/notfound.htm" : Wsize = 123
Bcontenttype = 0 : Woffset = 277
Case Else ' not found
Bcontenttype = 0 : Woffset = 277 : Wsize = 123
End Select

Select Case Bcontenttype
Case 0: ' text
Tempw = Tcpwrite(i , "Content-Type: text/html{013}{010}")
Case 1: ' gif
End Select
If Bgenerate = 0 Then ' data from eeprom
S = "Content-Length: " + Str(wsize) + "{013}{010}"
Tempw = Tcpwritestr(i , S , 255) ' add additional CR and LF
Tempw = Tcpwrite(i , Eeprom , Woffset , Wsize) ' write data
Else ' we generate the data
If Shtml = "/index.htm" Then
S = "<html><head><title>Easy TCP/IP</title></head><body><p><b>MCS webserver test<br></b>Hits : "
+ Str(ihitcounter) + "</p><p>&nbsp;</p><p>&nbsp;</p></body></html>"
Incr Ihitcounter 'increase hitcounter
Else
S = "<html><head><title></title></head><body onload='window.location.href=" + Chr(34) +
"http://www.mcselec.com" + Chr(34) + "></body></html>"
End If
Wsize = Len(s) ' size of body
Shheader = "Content-Length: " + Str(wsize) + "{013}{010}"
Tempw = Tcpwritestr(i , Shheader , 255) ' add additional CR and LF
Tempw = Tcpwrite(i , S , Wsize) ' send body
End If

```

Return

3- کار با شبکه را مانند کار با خط تلفن در نظر بگیرید، هنگامی که قصد دارید با یکی از دوستان ارتباط تلفنی داشته باشید، ابتدا باید پریز تلفن را پیدا کنید، در اطراف شما ممکن است پریز های مختلفی وجود داشته باشد (پریز برق، تلفن، آنتن، پریز های سالم، پریز های معیوب و..). هنگامی که پریز تلفن را پیدا کردید، (مقدار دهی socket با عدد صفر) شما میتوانید به شبکه ی مخابرات متصل شوید، اکنون باید تعیین کنید که چه قصد انجام چه کاری را دارید (مکالمه کنید، فکس ارسال کنید و..)

در این مثال میکرو کنترلر با استفاده از پروتکل POP3 به یک سرور ایمیل متصل شده و بعد از ورود به آن (لاگین شدن) ایمیل موجود در آن را بر روی LCD نمایش میدهد. شماتیک مورد نیاز برای این پروژه در صفحه ی بعد آورده شده است. (شماتیک در فرمت PDF با کیفیت بالا تر را از [این آدرس](#) دانلود کنید).

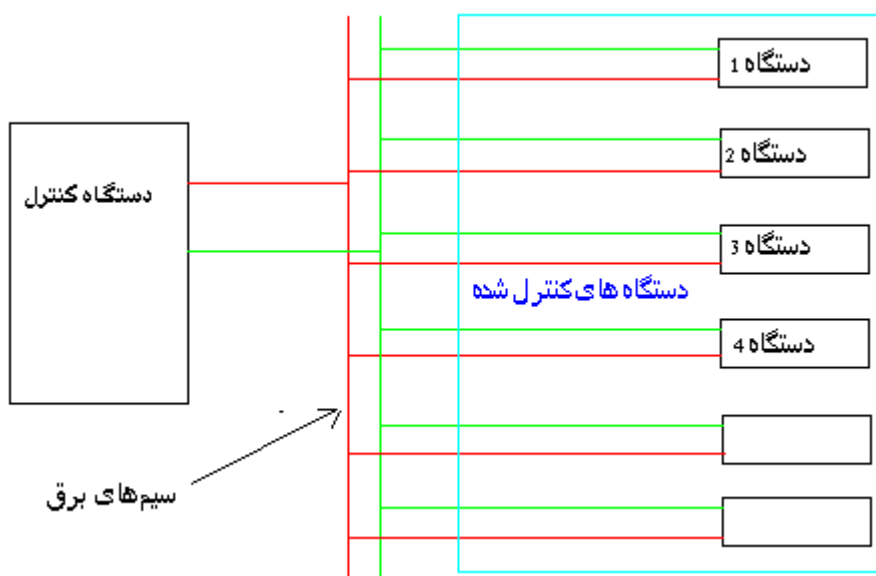
POP3 مخفف کلمه Post Office Protocol 3 یک موافقت نامه بین المللی برای دریافت و ارسال نامه های الکترونیکی می باشد. این توافق نامه توسط تمامی سرویس دهندگان اینترنت پشتیبانی و مورد استفاده قرار می گیرد. سیستم های پست الکترونیکی که بر مبنای این پروتکل طرح ریزی و طراحی می شوند بصورت مستقیم می توانند پذیرای درخواست های خارجی جهت خواندن و نوشتن ایمیل باشند، برای مثال نرم افزارهای Outlook میتواند از طریق پروتکل POP3 به سرویس ایمیل (مانند GMAIL) متصل شده و ایمیل های موجود در آن را بخواند (با توجه به نام کاربری و پسورد وارد شده در نرم افزار)

پروتکل x10 :

X10 چیست ؟

X10 یک استاندارد صنعتی برای کنترل دستگاه ها و لوازم برقی موجود در یک کارخانه یا خانه ، از یک یا چند محل میباشد . در این پروتکل سیم های انتقال برق به عنوان رابط ارتباطی انتخاب شده است . در این روش اطلاعات بصورت موجی با فرکانس بالا تر از فرکانس برق شهر روی آن اعمال میشوند .

X10 به عنوان اولین تکنولوژی demotic در سال 1975 توسط شرکت Pico Electronics اختراع شد که هنوز هم به عنوان پرکاربردترین تکنولوژی ها مطرح می باشد. امروزه از این تکنولوژی در سرتاسر دنیا به خصوص در مواقعی که امکان استفاده از شبکه های باسیم وجود ندارد استفاده می گردد. بلوک کلی این پروتکل را در زیر مشاهده میکنید:



Module Device ها :

بین دستگاه های کنترل شده و برق شهر ماژول هایی قرار میگیرند که بسته به باری که بایست کنترل شود نوع ماژول متفاوت است . این ماژول ها دیتا را از برق شهر میگیرد (از تغذیه ورودی) در صورتی که دیتای ورودی با دیتا موجود یکی بود فرمان مورد نظر انجام میشود و در غیر اینصورت ماژول عملی را انجام نمیدهد . ماژولهای گیرنده و فرستنده زیادی وجود دارد که هر کدام به یک روش دیتا را ارسال و دریافت میکند ، در زیر به بررسی یک از آنها میپردازیم:

در نرم افزار بسکام توابعی وجود دارد که شما با استفاده از آنها میتوانید میکرو avr را به عنوان فرستنده دیتا در این پروتکل به کار ببرید ، نام ماژولی که avr دیتا را به آن میفرستد TW-523 است . شما نمیتوانید میکرو avr را مستقیم به برق شهر متصل کنید ، برای اتصال avr به برق شهر (بطوری که دیتای خروجی از میکرو بدون آسیب به گیرنده TW-523 برسد) مدارات زیادی ارائه شده است که چند مورد در پوشه ضمیمه وجود دارد (شما میتوانید اطلاعات بیشتر و مدارات دیگر را از سایت www.x10.com دریافت کنید).مدار دارای پایه های خروجی زیر است ، نمونه های آماده این ماژول توسط شرکت x10 ارائه میشود که پایهای نمونه آماده نیز آورده شده است:

The TW-523 RJ-11 connector has the following pinout:

Pin	توصیف	Connect to micro	پایه (در نمونه آماده)
1	Zero Cross اشکار ساز عبور از صفر	Input pin. Add 5.1K pull up.	
2	GND	GND	گراند
3	RX	Not used.	گیرنده
4	TX	Output pin. Add 1K pull up.	فرستنده

پایه rx به دستگاه گیرنده اطلاعات متصل میشود.

پایه zero cross پایه اشکار ساز عبور از صفر میباشد ، (نشان میدهد که ولتاژ برق شهر در قله منفی است یه مثبت) شما باید



این پایه را با مقاومت 5.1 کیلو به vcc (5 ولت) متصل کنید.

پایه gnd زمین دستگاه است که باید به زمین میکرو متصل شود ،

نگران ولتاژ ac نباشید تمامی قسمتهای که به میکرو متصل میشود

توسط اپتو کوپلر ایزوله شده است.

پایه tx به فرستنده یا میکرو متصل میشود و اطلاعات را از میکرو به مدار و سپس به برق شهر منتقل میکند شما باید این پایه را

با مقاومت 1 کیلو به vcc (5 ولت) متصل کنید .

در نهایت با دستورات زیر پروتکل x10 راه اندازی میشود:

CONFIG X10 = pinZC , TX = portpin

PinZC : نشان دهنده پایه ای از میکرو است که پایه zero cross دستگاه به آن متصل میشود ، شما میتوانید پایه دلخواه از پایه

های میکرو را برای این کار انتخاب کنید.

Portpin : نشان دهنده پایه ای از میکرو است که پایه TX دستگاه به آن متصل میشود ، شما میتوانید پایه دلخواه از پایه های میکرو را برای این کار انتخاب کنید.

شما همچنین میتوانید با دستور زیر فرکانس شبکه ای که به آن مازول و میکرو را متصل کردید را پیدا کنید :

Result = X10DETECT()

Result : یک متغیر از جنس بایت میباشد که هنگام اتصال مازول به برق شهر یکی از اعداد صفر ، نشان دهنده عدم وجود ولتاژ در شبکه ، 1 ، نشان دهنده وجود ولتاژ در شبکه و فرکانس 50 هرتز ، 2 ، نشان دهنده وجود ولتاژ در شبکه و فرکانس 60 هرتز ، در آن ریخته میشود و در نهایت با دستور زیر دیتا از میکرو به مازول فرستنده و خطوط برق شهر ارسال میشود:

X10SEND house , code

House: کاراکتر اسکی میباشد که شناسنده دستگاه فرستنده به گیرنده است (ابتدا این کاراکتر ارسال میشود ، دستگاه گیرنده آن را میگیرد ، در صورتی که کد دریافتی با کد موجود در حافظه دستگاه یکی باشد ، دستگاه ادامه کدها که همان دستورات هستند را دریافت میکند و در غیر این صورت دستگاه در مقابل کدهای بعدی عکس العملی نشان نمیدهد)
CODE : در این نوع مازول گیرنده و فرستنده (TW-523) کدهای ارسالی و عملیاتی که گیرنده انجام میدهد مطابق جدول زیر است:

(کدها به فرم دسیمال نوشته شده اند ، این کدها به صورت پالس توسط میکرو به مازول فرستنده ارسال میشوند ، مازول آنها را روی برق شهر مدوله میکند ، این کدها توسط گیرنده از روی برق شهر جدا شد و دیکد میشود و عمل متناظر با آن در مازول گیرنده انجام میشود)

The following table lists all X10 codes.

Code value	Description
1-16	Used to address a unit. X10 can use a maximum of 16 units per house code.
17	All units off
18	All lights on
19	ON
20	OFF
21	DIM
22	BRIGHT
23	All lights off
24	Extended ode
25	Hail request
26	Hail acknowledge
27	Preset dim
28	Preset dim
29	Extended data analog
30	Status on
31	Status off
32	Status request

مثال:

```

$regfile = "m16def.dat"
$crystal = 8000000
Config Kbd = Porta
Config X10 = Pinc.7 , Tx = Portc.6
Dim A As Byte , X As Byte
Const House = "M"
X = X10detect()
Print X
Q:
A = Getkbd()
If A > 15 Then : Goto Q : End If
X10send House , A
Goto Q
End

```


کار با magnetic card (کارت های مغناطیسی):

کارت های مغناطیسی برخلاف اسمارت کارت ها و تگ های RFID از خاصیت نوار مغناطیسی خود برای انتقال اطلاعات به کارخوان استفاده میکنند، در واقع عمل کرد آنها تقریباً شبیه به نوار های کاست میباشد که اطلاعات (شماره سریال یا ...) بر روی آن توسط یون های مغناطیسی ایجاد شده و در صورتی که در مقابل یک میدان مغناطیسی کنترل شده قرار گیرد باعث تغییر در شدت میدان خواهد شد. ویژگی اصلی این کارت ها دارا بودن یک نوار سیاه رنگ است. کارت خوان این دستگاه که با نام ریدر کارت مغناطیسی (magnetic card reader) در بازار موجود است. دارای 5 سیم به شرح زیر است:

1- قرمز = تغذیه 5 ولت کارت ریدر

2- مشکی = تغذیه گراند کارت

3- زرد = انتخاب تراشه

4- سبز = کلاک کارت ریدر

5- ابی = خروجی دیتا برای میکرو

با دستورات زیر میتوان کارت ریدر را برای میکرو معرفی کرد:

```
_mport Alias Piny
```

```
_mdata Alias X
```

```
_mcs Alias X
```

```
_mclock Alias X
```

```
Config Porty = Input
```

```
Porty = 255
```

```
READMAGCARD var , count , coding
```

_mport Alias Piny: با این دستور پورتی که کارت ریدر به آن متصل است مشخص میشود:

```
mport Alias PinB_
```

_mdata Alias X: نشان میدهد که پایه داده به کدام یک از پایه های میکرو متصل شده است، X شماره پایه است که میتواند

بین 0 تا 7 باشد (سیم ابی)

_mcs Alias X : نشان میدهد که پایه انتخاب به کدام یک از پایه های میکرو متصل شده است ، X شماره پایه است که میتواند

بین 0 تا 7 باشد (سیم زرد)

_mclock Alias X : نشان میدهد که پایه کلاک به کدام یک از پایه های میکرو متصل شده است ، X شماره پایه است که

میتواند بین 0 تا 7 باشد (سیم سبز)

Config Porty = Input : پورتهی که کارت ریدر به آن متصل است حتما باید به عنوان ورودی تعریف شود

Porty = 255 : تمام ورودی های پورت را یک میکند (تا فقط به صفر شدن واکنش نشان دهند)

با دستور زیر میتوان خروجی کارت ریدر را خواند:

READMAGCARD var , count , coding

Var:در این متغیر بایت خوانده شده قرار میگیرد

Returned number	ISO characterT
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	hardware control
11	start byte
12	hardware control
13	separator
14	hardware control
15	stop byte

Count:در این متغیر شماره بیت خوانده شده قرار میگیرد ، شماره بیت طبق جدول زیر است :

داده های خوانده شده در کنار هم قرار میگیرند و یک رقم را به وجود میاورد ، این رقم ها میتواند رمز عبور، کد شناسایی یا

سایر موارد مورد نیاز باشد.

Coding: بر روی کارت های مغناطیسی سه خط وجود دارد ، در صورتی که شما از خط اول استفاده کنید باید به جای

Coding رقم 7 را قرار دهید و در صورتی که از خط دوم یا سوم استفاده کنید باید به جای Coding رقم 5 را قرار دهید .

مثال:

```

$regfile = "m16def.dat" : $crystal = 1000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Pina.0 , Db5 = Pina.1 , Db6 = Pina.2 , Db7 = Pina.3 , Rs = Pina.4 , E = Pina.5

Dim A As Byte , B As Byte

_mport Alias Pinb

_mdata Alias 0

_mcs Alias 1

_mclock Alias 2

Config Portb = Input

Portb = 255 : Do

Reset Porta.0 : Locate 1 , 1 : Lcd "Insert your card"

Readmagcard A , B , 5

Select Case A

Case 12

Locate 1 , 1 : Lcd "welcome mr a" : Wait 2 : Cls

Case 13:

Locate 1 , 1 : Lcd "welcome mr b" : Set Porta.0 : Wait 5 : Cls

Case 14 :

Locate 1 , 1 : Lcd "welcome mr c" : Set Porta.0 : Wait 5 : Cls

Case 15 :

Locate 1 , 1 : Lcd "welcome mr d" : Set Porta.0 : Wait 5 : Cls

Case Else :

Locate 1 , 1 : Lcd "card not right" : Reset Porta.0 : Wait 5 : Cls

End Select : Loop

End

```

برنامه بالا مربوط به یک سیستم نگهبانی است ، فقط افرادی که دارای کارت میباشند و مقداریر حافظه کارت آنها در دستگاه ثبت شده میتوانند وارد شوند .

اتصال avr به عنوان کیبرد به کامپیوتر:

کیبرد و موس یکی از قطعات جانبی کامپیوتر است که توسط آنها میتوان داده های مختلفی را به کامپیوتر وارد نمود ، این قطعات معمولاً از دو خط برای انتقال پالس کلاک و داده و دو خط ولتاژ تغذیه برای ارتباط به کامپیوتر استفاده میکنند ، در این بخش سعی داریم شما را با نحوه ی ساخت این قطعات توسط میکرو کنترلر های avr و کامپایلر بسکام آشنا کنیم .

در کامپایلر بسکام میتوانیم با استفاده از دستور زیر دو پایه ی دلخواه از میکرو کنترلر را به عنوان پایه ی دریافت پالس همزمانی و ارسال داده پیکربندی کنیم :

CONFIG Atemu = int , DATA = data, CLOCK=clock

Int : شما میتوانید از وقفه صفر یا یک برای این مورد استفاده کنید (int0 یا int1) ، در این حالت هنگامی که دستوری از سمت کامپیوتر ارسال میشود ، cpu با تحریک شدن منبع وقفه به زیر برنامه ی پاسخ گویی به کامپیوتر میرود .

Data : نام پایه ای از میکرو است که سیم دیتای پورت کیبرد کامپیوتر به آن متصل میشود (سیم دیتای سوکت کیبرد باید به پایه ورودی وقفه متصل شود)(شما مجازید از دوپایه int0 و int1 (پایه 16 و 17 میکرو مگا 16) استفاده کنید))

Clock : نام پایه ای از میکرو است که سیم کلاک پورت کیبرد کامپیوتر به آن متصل میشود. با دستور زیر میتوانید کد های دلخواه را به کیبرد را به کامپیوتر ارسال کنید

SENDSCANKB label | var

Label نام برجسبی است که کد در آن قرار دارد (شما میتوانید چند کد را مانند مثال پشت سر هم نوشته و آنها را به کامپیوتر بفرستید)

Var : شما همچنین میتوانید یک متغیر را به کامپیوتر بفرستید. در زیر نام کلید های صفحه کلید و کدی که هر یک میسازند را مشاهده میفرمایید:

KEY	MAKE	BREAK	KEY	MAKE	BREAK	KEY	MAKE	BREAK
A	1C	F0,1C	9	46	F0,46	[54	F0,54
B	32	F0,32	`	0E	F0,0E	INSERT	E0,70	E0,F0,70
C	21	F0,21	-	4E	F0,4E	HOME	E0,6C	E0,F0,6C
D	23	F0,23	=	55	F0,55	PG UP	E0,7D	E0,F0,7D
E	24	F0,24	\	5D	F0,5D	DELETE	E0,71	E0,F0,71
F	2B	F0,2B	BKSP	66	F0,66	END	E0,69	E0,F0,69
G	34	F0,34	SPACE	29	F0,29	PG DN	E0,7A	E0,F0,7A
H	33	F0,33	TAB	0D	F0,0D	U ARROW	E0,75	E0,F0,75
I	43	F0,43	CAPS	58	F0,58	L ARROW	E0,6B	E0,F0,6B
J	3B	F0,3B	L SHFT	12	F0,12	D ARROW	E0,72	E0,F0,72
K	42	F0,42	L CTRL	14	F0,14	R ARROW	E0,74	E0,F0,74
L	4B	F0,4B	L GUI	E0,1F	E0,F0,1F	NUM	77	F0,77
M	3A	F0,3A	L ALT	11	F0,11	KP /	E0,4A	E0,F0,4A
N	31	F0,31	R SHFT	59	F0,59	KP *	7C	F0,7C
O	44	F0,44	R CTRL	E0,14	E0,F0,14	KP -	7B	F0,7B
P	4D	F0,4D	R GUI	E0,27	E0,F0,27	KP +	79	F0,79
Q	15	F0,15	R ALT	E0,11	E0,F0,11	KP EN	E0,5A	E0,F0,5A
R	2D	F0,2D	APPS	E0,2F	E0,F0,2F	KP .	71	F0,71
S	1B	F0,1B	ENTER	5A	F0,5A	KP 0	70	F0,70
T	2C	F0,2C	ESC	76	F0,76	KP 1	69	F0,69
U	3C	F0,3C	F1	05	F0,05	KP 2	72	F0,72
V	2A	F0,2A	F2	06	F0,06	KP 3	7A	F0,7A
W	1D	F0,1D	F3	04	F0,04	KP 4	6B	F0,6B
X	22	F0,22	F4	0C	F0,0C	KP 5	73	F0,73
Y	35	F0,35	F5	03	F0,03	KP 6	74	F0,74
Z	1A	F0,1A	F6	0B	F0,0B	KP 7	6C	F0,6C
0	45	F0,45	F7	83	F0,83	KP 8	75	F0,75
1	16	F0,16	F8	0A	F0,0A	KP 9	7D	F0,7D
2	1E	F0,1E	F9	01	F0,01]	5B	F0,5B
3	26	F0,26	F10	09	F0,09	;	4C	F0,4C
4	25	F0,25	F11	78	F0,78	'	52	F0,52
5	2E	F0,2E	F12	07	F0,07	,	41	F0,41
6	36	F0,36	PRNT	E0,12,	E0,F0,	.	49	F0,49
			SCRN	E0,7C	7C,E0,			
					F0,12			
7	3D	F0,3D	SCROLL	7E	F0,7E	/	4A	F0,4A
8	3E	F0,3E	PAUSE	E1,14,77,	-NONE-			
				E1,F0,14,				
				F0,77				

در مثال زیر میخواهیم عبارت 1nafar را به کامپیوتر بفرستیم ، بعد از ساخت سخت افزار و اتصال آن به کامپیوتر در صورتی

که برنامه Notepad یا word را باز کنید ، عبارت مذکور در آن نوشته میشود.

مانند تمامی برنامه ها ابتدا میکرو و کریستال را معرفی میکنیم:

```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

قدمی بعدی فعال سازی وقفه سراسری (هنگامی که میخواهیم از وقفه استفاده کنیم باید آن را فعال کنیم ، با دستور که نام

برده میشود تمامی وقفه ها فعال میشوند) است:

```
Enable Interrupts
```

مرحله بعدی معرفی پایه های میکرو است که باید به پورت کیبرد موجود در پشت کامپیوتر متصل شود:

Config Atemu = Int1 , Data = Pind.3 , Clock = Pinb.0

در حالت بالا پایه کلاک پورت کیبرد به پورت b.0 و پایه دیتا آن به ورودی وقفه 1 (پورت d.3) متصل است، (در صوتی

که از وقفه 0 استفاده شود، پایه دیتا باید به پورت d.2 متصل شود)

در مرحله بعد یک حلقه ایجاد میکنیم تا میکرو مدام عبارت را به کامپیوتر ارسال کند:

Do

و در نهایت با دستور زیر برچسبی که در آن عبارت "1nafar" وجود دارد به کامپیوتر ارسال میشود:

Sendscankbd data1

یک تاخیر زمانی، برای اینکه صفحه مدام پر نشود:

Wait 2

پایان حلقه و پایان برنامه:

Loop

End

و در نهایت ایجاد برچسب:

Data1:

Data 18 , &H16 , &HF0 , &H16 , &H31 , &HF0 , &H31 , &H1C , &HF0 , &H1C , &H2B , &HF0 , &H2B , &H1C , &HF0 , &H1C , &H2D , &HF0 , &H2D

'data tedad ersal ,1 ,n , a , f , a , r

اولین داده جدول تعداد بایت ارسالی است، در اینجا تمامی بایت ها (که تعداد آنها 18 تاست) با هم ارسال شده اند (منظور

پشت سرهم است)، شما میتوانید مانند جدول lookup، جدول را آدرس دهی کنید و داده های مورد نظر را بفرستید، توجه

داشته باشید که تعداد بایت نباید از 3 کمتر باشد، در این صورت داده کامل منتقل نمیشود. داده های بعدی، اعداد و حروف

میباشند، مثلا برای عدد 1 باید، مطابق جدول بالا (نام کلید های صفحه کلید و کدی که هر یک میسازند) کد &H16 , &HF0

&H16 , ارسال شود و برای کلید DELETE باید کد &H71 , &HE0 , &HF0 ارسال شود

ودر زیر کد مربوط به کلید های مدیاپلیر و دیگر کلید های موجود بر روی صفحه کلید را مشاهده می فرمایید:

Key	Make Code	Break Code
Next Track	E0, 4D	E0, F0, 4D
Previous Track	E0, 15	E0, F0, 15
Stop	E0, 3B	E0, F0, 3B
Play/Pause	E0, 34	E0, F0, 34
Mute	E0, 23	E0, F0, 23
Volume Up	E0, 32	E0, F0, 32
Volume Down	E0, 21	E0, F0, 21
Media Select	E0, 50	E0, F0, 50
E-Mail	E0, 48	E0, F0, 48
Calculator	E0, 2B	E0, F0, 2B
My Computer	E0, 40	E0, F0, 40
WWW Search	E0, 10	E0, F0, 10
WWW Home	E0, 3A	E0, F0, 3A
WWW Back	E0, 38	E0, F0, 38
WWW Forward	E0, 30	E0, F0, 30
WWW Stop	E0, 28	E0, F0, 28
WWW Refresh	E0, 20	E0, F0, 20
WWW Favorites	E0, 18	E0, F0, 18

Key	Make Code	Break Code
Power	E0, 37	E0, F0, 37
Sleep	E0, 3F	E0, F0, 3F
Wake	E0, 5E	E0, F0, 5E

میخواهیم با یک کلید صدا را کم و با کلید دیگر صدا را زیاد کنیم برای این کار چند روش وجود دارد که یکی از آنها در زیر آمده است:

```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

```
Enable Interrupts
```

```
Config Atemu = Int1 , Data = Pind.3 , Clock = Pinb.0
```

```
Waitms 500
```

```
Config Porta = Input
```

```
Q:
```

```
Debounce Pina.0 , 1 , Vup
```

```
Debounce Pina.0 , 1 , Vdown
```

```
Goto Q
```

```
Vup:
```

```
Sendscankbd Data1
```

```
Waitms 500
```

```
Goto Q
```

```
Vdown:
```

```
Sendscankbd Data2
```

```
Waitms 500
```

```
Goto Q
```

```
End
```

```
Data 5 , &HE0 , &H32 , &HE0 , &HF0 , &H32
```

```
Data2:
```

```
Data 5 , &HE0 , &H21 , &HE0 , &HF0 , &H21
```

اتصال avr به عنوان موس به کامپیوتر:

با مجموعه دستورات زیر میتوانید میکرو کنترلر های avr را طوری در محیط بسکام برنامه ریزی کنید که بتوان از آنها به عنوان موس استفاده کرد ، در این حالت توسط توابع میتوان مختصات لازم و وضعیت کلید های موجود بر روی موس های واقعی را به کامپیوتر ارسال نمود .

راه اندازی این مورد با دستور زیر انجام میشود:

```
CONFIG PS2EMU= int , DATA = data, CLOCK=clock
```

Int : شما میتوانید از وقفه صفر یا یک برای این مورد استفاده کنید (int0 یا int1)

Data : نام پایه ای از میکرو است که سیم دیتای پورت موس کامپیوتر (که شکل آن را در بالا مشاهده فرمودید) به آن متصل

میشود. (سیم دیتای سوکت موس باید به پایه ورودی وقفه متصل شود) (شما مجازید از دو پایه int0 و int1 (پایه 16 و 17 میکرو

مگا 16 استفاده کنید))

Clock : نام پایه ای از میکرو است که سیم کلاک پورت موس کامپیوتر به آن متصل میشود.

با دستور زیر میتوانید کدهای دلخواه را به کامپیوتر ارسال کنید

```
PS2MOUSEXY X , Y, button
```

X و Y مختصات مکان توقف اشاره گر موس است که از -255 تا 255 میباشد.

Button : عدد معادل کلید فشرده شده میباشد (در روی موس 3 کلید اصلی وجود دارد : کلید راست ، کلید چپ، کلید وسط

(که رقم

معادل هر کلید در زیر آمده است

no buttons pressed - 0 صفر :هیچ کلیدی فشرده نشده است

left button pressed - 1 یک : کلید سمت چپ فشرده شده است

right button pressed - 2 دو : کلید سمت راست فشرده شده است

middle button pressed - 4 چهار : کلید وسط فشرده شده است

با دستور زیر نیز میتوانید کدهای مربوط به موس را از جدول بخوانید و به کامپیوتر ارسال کنید:

```
Sendscan lable
```

Lable : نام برجسیبی است که کدها در آن وجود دارد . مثال :

```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

```
Enable Interrupts
```

```
Config Ps2emu = Int0 , Data = Pind.2 , Clock = Pind.1
```

```
Config Porta = Input
```

```
Dim X As Integer , Y As Integer , Button As Byte
```

```
Q:
```

```
Debounce Pina.0 , 1 , Mright
```

```
Debounce Pina.1 , 1 , Mleft
```

```
Debounce Pina.2 , 1 , Mup
```

```
Debounce Pina.3 , 1 , Mdown
```

```
Debounce Pina.4 , 1 , Rpressed
```

```
Debounce Pina.5 , 1 , Lpressed
```

```
Debounce Pina.6 , 1 , Mpressed
```

```
Ps2mousexy X , Y , Button
```

```
Goto Q
```

```
Mright:
```

```
Incr X : Waitms 500 : Goto Q
```

```
Mleft:
```

```
Decr X : Waitms 500 : Goto Q
```

```
Mup:
```

```
Incr Y : Waitms 500 : Goto Q
```

```
Mdown:
```

```
Decr Y : Waitms 500 : Goto Q
```

```
Rpressed:
```

```
If Button = 2 Then : Button = 0 : Else : Button = 2 : End If
```

```
Waitms 500 : Goto Q
```

```
Lpressed:
```

```
If Button = 1 Then : Button = 0 : Else : Button = 1 : End If
```

```
Waitms 500 : Goto Q
```

```
Mpressed:
```

```
If Button = 4 Then : Button = 0 : Else : Button = 4 : End If
```

```
Waitms 500 : Goto Q
```

```
End
```

مثال:

```

$regfile = "m16def.dat"

$crystal = 8000000

Enable Interrupts

Config Ps2emu = Int0 , Data = Pind.2 , Clock = Pind.1

Dim A As Byte

Q:

Incr A

Wait 1

Select Case A

Case 1 : Ps2mousexy 0 , 10 , 0      ' up
Case 2 : Ps2mousexy 0 , -10 , 0    ' down
Case 3 : Ps2mousexy -10 , 0 , 0    ' left
Case 4 : Ps2mousexy 10 , 0 , 0     ' right
Case 5 : Ps2mousexy 0 , 0 , 1      ' left button pressed
Ps2mousexy 0 , 0 , 0              ' left button released

Case 6 : Sendscan W

End Select

Goto Q

W:

Data 3 , &H08 , &H00 , &H01

```

نکته: شما مجازید فقط یک موس یا کیبرد به کامپیوتر خود متصل کنید ، اتصال سخت افزار بیشتر ممکن است به کامپیوتر

شما آسیب بزنند.

کتابخانه های مورد نیاز برای دو مبحث بالا را در پوشه ی پیوست مشاهده نمایید .

سروو موتور:

سروو ها نوعی موتور بسیار پر قدرت است که میتواند حول یک زاویه خاص با دقت بالا بچرخند ، از سروو ها برای بازوی ربات و باز و بسته کردن درب و دریچه و ... استفاده میشود ، این موتور ها از یک موتور DC ، یک مدار کنترل کننده و یک مدار فیدبک ساخته میشود ، همچنین برای افزایش قدرت گشتاور خروجی یک گیربکس در خروجی موتور DC قرار میگیرد .

سروو ها دارای سه سیم میباشند که دوتای آنها تغذیه و دیگری برای کنترل درجه چرخش مورد استفاده قرار میگیرد (در صورتی که سروو شما دارای 5 سیم است ، دو تا از آنها برای تغذیه مدار داخلی سروو و دوتای دیگر برای تغذیه خود سروو و سیم آخر برای کنترل میباشد ، بهتر است دیتا شیت سروو را از فروشنده دریافت کنید) .

راه اندازی سروو در بسکام با دستور زیر انجام میشود:

Config Servos = X , Servo1 = Portx.y , Servo2 = Portx.y , Servon = Portx.y , Reload = RI

Config Servos = X : نشان دهنده تعداد سروو های استفاده شده میباشد که بیشترین تعداد میتواند 14 باشد.(به جای x تعداد

گذاشته میشود مانند : Config Servos = 3)

Servo1 = Portx.y : پایه کنترل سروو به یکی از پین های میکرو که با Portx.y مشخص شده متصل میشود.

Reload: نشان دهنده زمانی است که میکرو دوباره اطلاعات مربوط به سرو ها را روی پین مورد نظر میفرستد(این زمان بر حسب میکروثانیه است).

نکته: این دستور از تایمر صفر برای راه اندازی (تولید زمان روشن بودن سروو) استفاده میکند و هنگامی که سروو را پیکربندی کردید دیگر نمیتوانید از تایمر صفر استفاده کنید.

بعد از پیکربندی سروو نوبت به راه اندازی آن است برای این کار از دستور زیر استفاده میشود:

Servo(x) = y

X شماره سروو است که میتواند از 1 تا 14 باشد و y ضرب در 10 زمان روشن بودن سروو را نشان میدهد، که میتواند یک

متغیر یا عدد صحیح باشد.

مثال:

```
$regfile = "m16def.dat"
```

```
$crystal = 12000000
```

```
Config Servos = 14 , Servo1 = Portd.0 , Servo2 = Portd.1 , Servo3 = Portd.2_
```

```
, Servo4 = Portd.3 , Servo5 = Portd.4 , Servo6 = Portd.5 , Servo7 = Portd.6_
```

```
, Servo8 = Portd.7 , Servo9 = Portc.7 , Servo10 = Portc.6 , Servo11 = Portc.5_
```

```
, Servo12 = Portc.4 , Servo13 = Portc.3 , Servo14 = Portc.2 , Reload = 100
```

```
Config Portd = Output , Portc = Output
```

```
Enable Interrupts
```

```
Dim A As Word
```

```
A=10
```

```
Do
```

```
Servo(1) = 1 : Wait 1
```

```
Servo(2) = 5 : Wait 1
```

```
Servo(3) = a : Wait 1
```

```
Servo(4) = 15 : Wait 1
```

```
Servo(5) = 20 : Wait 1
```

```
Servo(6) = 25 : Wait 1
```

```
Servo(7) = 30 : Wait 1
```

```
Servo(8) = 35 : Wait 1
```

```
Servo(9) = 40 : Wait 1
```

```
Servo(10) = 45 : Wait 1
```

```
Servo(11) = 50 : Wait 1
```

```
Servo(12) = 55 : Wait 1
```

```
Servo(13) = 60 : Wait 1
```

```
Servo(14) = 65 : Wait 1
```

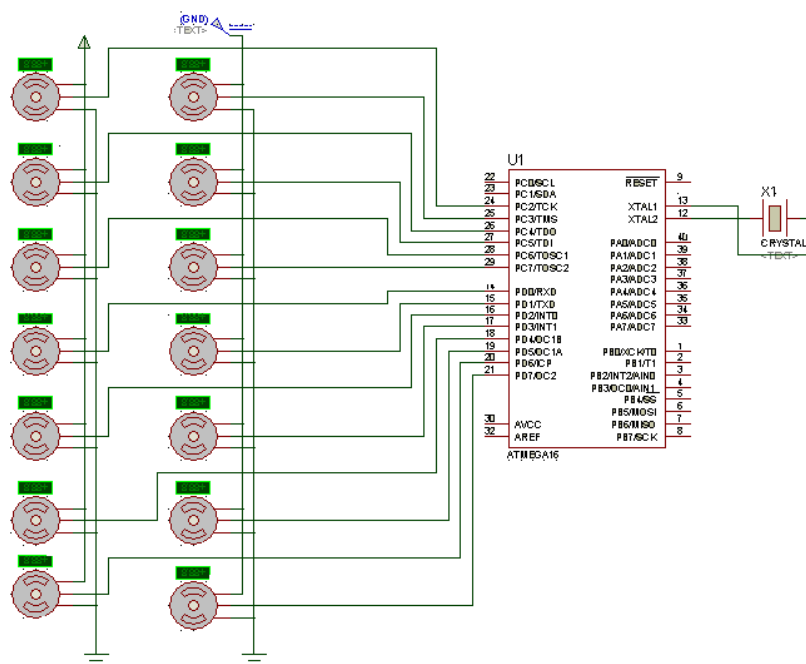
```
Loop
```

```
End
```

در مثال بالا تعداد 14 عدد سروو به میکرو مگا 16 متصل شده است ، مدت زمان روشن بودن سروو ها به ترتیب از سروو 1 به

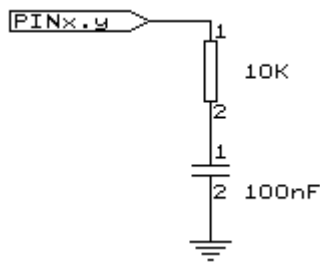
این شرح است: 10 و 50 و 100 و 150 و 200 و 250 و 300 و 350 و 400 و 450 و 500 و 550 و 600 و 650 میکرو ثانیه است. (یک

سرو فقط در یک زاویه خاص میچرخد ، مثلا از 0 تا 270 درجه یا از 0 تا 360 درجه، زمان که در بالا گفته شد ، بیان میکند که چقدر طول میکشد تا سرو به یک زاویه خاص برسد ، برای فهمیدن این زمان باید به دیتاشیت سرو مراجعه کنید ، مثلا نوعی سرو در هر 10 میکرو ثانیه 1 درجه حرکت میکند و زاویه چرخش آن از 0 تا 10 است، بنابراین برای رسیدن به زاویه 120 درجه باید به جای y عدد 120 را قرار دهیم، در صورتی که به جای y عدد بیشتر از 180 قرار دهیم سرو روی 180 قفل میشود. بیشتر مقداری که به جای y میتوانید قرار دهید 255 است. مدار مثال بالا:



اندازه گیری یک خازن یا مقاومت:

در صورتی که یک مقاومت و یک خازن را مطابق تصویر زیر به صورت موازی به هم متصل کنید و سپس آنها را به یک منبع ولتاژ وصل کنید ، خازن در مدت زمانی کوتاه شارژ شده و جریان عبوری از مقاومت برابر با صفر میشود . اکنون اگر خازن را دشارژ نمایید و مدت زمانی که مقدار ولتاژ خازن برابر صفر میشود را اندازه گیری کنید میتوانید با استفاده از فرمول $t=R.C$ مقدار مقاومت یا خازن مجهول را محاسبه کنید .



در کامپایلر بسکام شما با استفاده از دستور زیر میتوانید مقدار ثابت زمانی مقاومت و خازنی که به پایه دلخواه میکرو کنترلر متصل شده است را بدست آورید (مقدار t در فرمول بالا) :

`var = GETRC(pin , number)`

var: یک متغیر از جنس word میباشد که مقدار ثابت زمانی در آن ریخته میشود.

Pin: نام پورتهی است که خازن و مقاومت به آن متصل است (مانند porta یا portd).

Number: شماره پایه ای است که مقاومت و خازن به آن متصل شده است (مانند 1 یا 2) (این مقدار نمیتواند از 7 بیشتر شود

((در مدارات مقاومت یا خازن که به اختصار به آن rc میگویند ، خازن بعد از 5 ثابت زمانی شارژ میشود (بعد از $5t$) مقدار

دقیق این ثابت زمانی به مقدار خازن و مقدار مقاومت بستگی دارد و فرمول آن به شکل $t=rc$ است ، میکرو مقدار ثابت

زمانی را اندازه میگیرد ، شما با داشتن مقدار یکی از المانها میتوانید مقدار دیگر را بدست آورید ، مثال:

```
$regfile = "M16DEF.DAT"
```

```
$crystal = 8000000
```

```
Config Lcdpin = Pin , Db4 = Pinc.1 , Db5 = Pinc.2 , Db6 = Pinc.3 , Db7 = Pinc.4 , E = Pind.2 , Rs = Pind.3
```

```
Config Lcd = 16 * 2
```

```
Config Porta = Output
```

```
Dim W As Word
```

```
Do
```

```
W = Getrc(pina , 7) : W = W / 1000 : Locate 1 , 1 : Lcd W : Wait 2
```

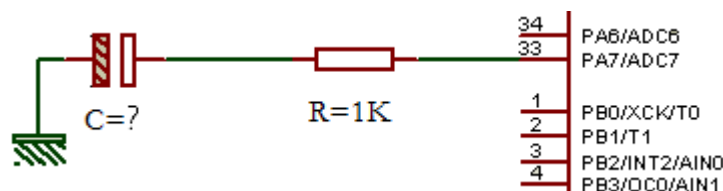
```
Loop
```

```
end
```

در مثال بالا مقدار یک خازن مجهول اندازه گرفته شده است ، در این مثال خازن مجهول با یک مقاومت 1 کیلو اهم سری

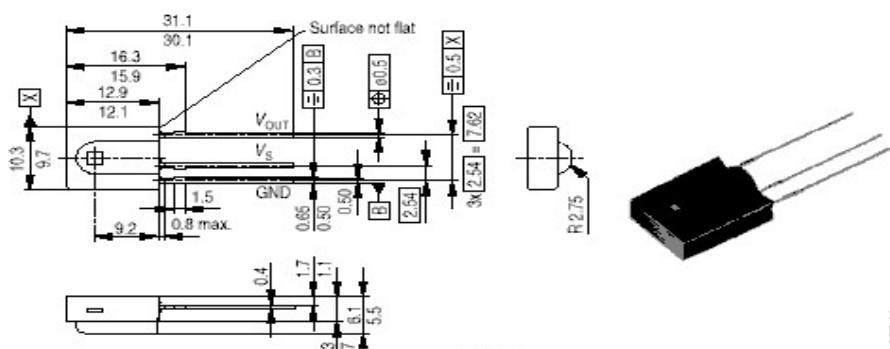
شده و مانند تصویر زیر به porta.7 متصل شده است ، (شما همچنین میتوانید مقدار مقاومت را نیز اندازه گیری کنید ، به

شرطی که مقدار خازن را بدانید):

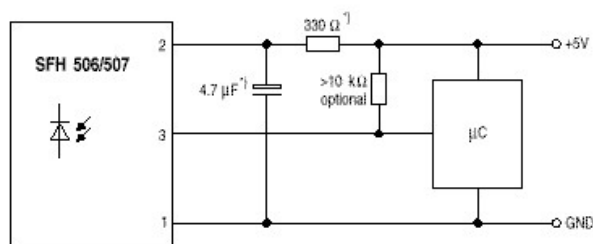


راه اندازی فرستنده / گیرنده RC5 :

امروزه گیرنده و فرستنده های مادون قرمز که آنها را با نام تجاری گیرنده و فرستنده های RC5 میشناسند رواج فوق العاده ای پیدا کرده اند ، از آنها در کنترل از راه دور ، سنسور های شمارنده ، ربات ها و استفاده فروانی میشود . در زیر شکل گیرنده (یا فتوترانزیستور) آن را ببینید:

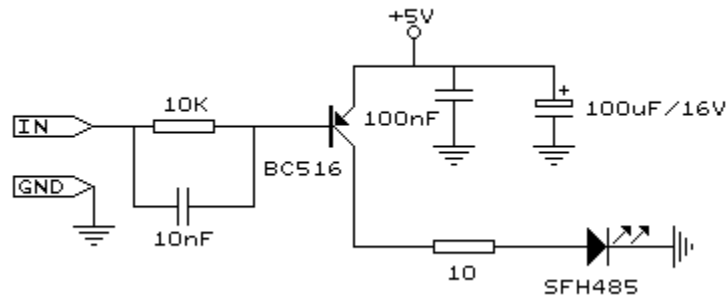


فرستنده (یا ir-led) دقیقا مانند یک led میباشد که رنگ آن سفید است ، اتصال گیرنده به میکرو مطابق شکل زیر است:



(نوع 2 پایه این سنسور نیز موجود میباشد که به نام گیرنده مادون قرمز معروف است ، اتصال نوع دوپایه بدون V_{CC} صورت میگیرد ، نوع دوپایه باعث ایجاد خطای غیر قابل چشم پوشی میشود ، به طوری که از آن فقط در موارد تشخیص مانع استفاده میگردد)

اتصال فرستنده به میکرو مانند شکل زیر است :



ترانزیستور نقش تقویت پالسها و دیگر قطعات وظیفه حذف نویز را به عهده دارند .

در صورت نیاز شما میتوانید داده ی خارج شده از میکرو را با یک پالس 40 کیلو هرتز مدوله کنید ، تا داده ی ارسالی مصونیت بیشتر در برابر نویز داشته باشد ، در ادامه بیشتر در این باره توضیح داده ایم .

در کامپایلر بسکام ، فرستنده مادون قرمز با دستور زیر راه اندازی می شود:

```
RC5SEND togglebit, address, command
```

Togglebit: به جای این واژه شما باید 0 یا 32 قرار دهید ، این اعداد نشان دهنده سطح شروع ارسال پالس می باشند.

Address: نشان دهنده آدرس دستور است که میتواند به فرم باینری هگز یا دسیمال باشد .(در گیرنده و فرستنده باید آدرس

دستور فرستاده شده و دستور گرفته شده یکی باشد ، تا دستور اجرا شود)

Command: نشان دهنده فرمان است که میتواند به فرم باینری ، هگز یا دسیمال باشد. (مثلا شما در گیرنده دستور زیر را

ارسال می کنید :

```
Rc5send 0 , 20 , 12
```

در گیرنده با دستوراتی که بعدا گفته می شود ، این کد را دریافت می کنید ، سپس با یک دستور if می توانید هر کاری که خواستید انجام دهید ،(مثلا اگر دستور 12 و آدرس 20 بود پین b.0 را یک کن) .

توجه داشته باشید که وقتی 5 rc پیکر بندی میشود ، پایه OC1(A) به عنوان خروجی داده قرار میگیرد و فرستنده باید به این

پایه متصل شود ، در این حالت دیگر نمیتوان از وقفه های تایمر 1 استفاده کرد. در مثال زیر با نحوه کار این فرستنده بیشتر

آشنا می شوید:

```
$regfile = "2313def.dat"
```

```
$crystal = 4000000
```

```
Config Portd = Input
```

```

Do
Debounce Pind.0 , 0 , Q
Debounce Pind.1 , 0 , W
Loop
Q:
Rc5send 32 , 0 , 12
Wait 1
Return
W:
Rc5send 32 , 0 , 13
Wait 1
Return
End

```

در حالت قبل کد ها نوشته شده بدون هیچ تغییری به خروجی ارسال میشد ، حالتی وجود دارد که شما میتوانید کد ارسالی را با یک کد باینری ترکیب کنید (کد را به صورت رمز در آورید) برای این کار از دستور زیر استفاده میشود:

RC5SENDEXT togglebit, address, command

همه چیز مانند حالت قبل است فقط به جای togglebit میتوانید هر عددی که دلتان میخواهد قرار دهید تا با دستور ترکیب شده و ارسال شود (در گیرنده باید عدد گذاشته شده را بردارید که در ادامه توضیح داده می شود .

مثال :

```

$regfile = "2313def.dat"
$crystal = 4000000
Config Portd = Input
Do
Debounce Pind.0 , 0 , Q
Debounce Pind.1 , 0 , W
Loop
Q:
Rc5sendext &B11000000 , 0 , 26
Wait 1
Return
W:
Rc5sendext 9 , 0 , 30
Wait 1
Return
End

```

راه اندازی گیرنده rc5

گیرنده rc5 که شکل و طریقه اتصال آن به میکرو را در بالا مشاهده کردید با دستور زیر راه اندازی می شود:

CONFIG RC5 = pin

که oin نام پایه دلخواه میکرو میباشد که پایه خروجی گیرنده سه پایه rc5 به آن متصل میشود. با دستور زیر میتوان اطلاعات دریافتی توسط گیرنده rc5 را آشکار کرد:

GETRC5(address, command)

Address و command اطلاعات مربوط به دستور و آدرسی میباشد که توسط فرستنده ارسال شده و توسط گیرنده دریافت میشود، این اطلاعات باید در متغیرهای مناسب که میتواند از جنس bayt یا word باشد ریخته شوند و مورد استفاده قرار گیرند .

مثال : (این برنامه برای گیرنده مثال قبلی میباشد)

```
$regfile = "2313def.dat" : $crystal = 4000000
Config Portd = Output
Config Rc5 = Pind.7
Enable Interrupts
Dim Address As Byte , Command As Byte
Do
Getrc5(address , Command)
If Command = 12 Then
Set Portd.0 : Reset Portd.1
End If
If Command = 13 Then
Set Portd.1 : Reset Portd.0
End If
Loop
End
```

ساخت کنترلر تلویزیون و سیدی SONY:

توسط دستور زیر میتوان دستورات مخصوص کنترلر تلویزیون و cd سونی را به این دستگاه ارسال کرد.

SONYSEND address

address : آدرس نام دستور است که با ارسال آن به تلویزیون کار مخصوص آن انجام میشود ، در زیر جدول کدهای

مخصوص تلویزیون های sony آمده است:

---[SONY TV Infrared Remote Control codes (RM-694)]-----		
نام کلید	کد هگز	کد باینری
program +	&H090	0000 1001 0000
program -	&H890	1000 1001 0000
volume +	&H490	0100 1001 0000
volume -	&HC90	1100 1001 0000
Power	&HA90	1010 1001 0000
sound on/off	&H290	0010 1001 0000
1	&H010	0000 0001 0000
2	&H810	1000 0001 0000
3	&H410	0100 0001 0000
4	&HC10	1100 0001 0000
5	&H210	0010 0001 0000
6	&HA10	1010 0001 0000
7	&H610	0110 0001 0000
8	&HE10	1110 0001 0000
9	&H110	0001 0001 0000
0	&H910	1001 0001 0000
-/--	&HB90	1011 1001 0000

SONY CD Infrared Remote Control codes (RM-DX55)

Function	Hex	Bin
Power	A91	1010 1001 0001
Play	4D1	0100 1101 0001
Stop	1D1	0001 1101 0001
Pause	9D1	1001 1101 0001
Continue	B91	1011 1001 0001
Shuffle	AD1	1010 1101 0001
Program	F91	1111 1001 0001
Disc	531	0101 0011 0001
1	011	0000 0001 0001
2	811	1000 0001 0001
3	411	0100 0001 0001
4	C11	1100 0001 0001
5	211	0010 0001 0001
6	A11	1010 0001 0001
7	611	0110 0001 0001
8	E11	1110 0001 0001
9	111	0001 0001 0001
0	051	0000 0101 0001
>10	E51	1110 0101 0001
enter	D11	1101 0001 0001
clear	F11	1111 0001 0001
repeat	351	0011 0101 0001
disc -	BD1	1011 1101 0001
disc +	H7D1	0111 1101 0001
<<	0D1	0000 1101 0001
>>	8D1	1000 1101 0001
<<	CD1	1100 1101 0001
>>	2D1	0010 1101 0001
SONY Cassette	RM-J901)	
Deck A		
stop	1C1	0001 1100 0001
play >	4C1	0100 1100 0001
play <	EC1	1110 1100 0001
>>	2C1	0010 1100 0001
<<	CC1	1100 1100 0001
record	6C1	0110 1100 0001
pause	9C1	1001 1100 0001
Dec B		
stop	18E	0001 1000 1110
play >	58E	0101 1000 1110
play <	04E	0000 0100 1110
>>	38E	0011 1000 1110
<<	D8E	1101 1000 1110
record	78E	0111 1000 1110
pause	98E	1001 1000 1110

مثال

\$regfile = "m8def.dat"

\$crystal = 4000000

Config Kbd = Portd

Dim A As Byte

```
Dim B As Byte
```

```
Q:
```

```
A = Getkbd()
```

```
If A > 15 Then
```

```
Goto Q
```

```
End If
```

```
B = Lookup(a , Dat)
```

```
Sonysend B
```

```
Goto Q
```

```
End
```

```
Dat:
```

```
Data &H090 , &H890 , &H490 , &HC90 , &HA90 , &H290 , &H010 , &H810 , &H410 , &HC10 , &H210 , &HA10 , &H610 , &HE10 ,  
&H110 , &H910
```

```
'program +,program -,volume +,volume -,power,sound on/off,1,2,3,4,5,6,7,8,9,0
```

در برنامه بالا با استفاده از میکرو مگا 8 و فرستنده rc5 یک کنترل مخصوص تلوزیون سونی ساخته شده است ، فرستنده rc5 مطابق مداری که در بالا معرفی شد به پایه oc1a میکرو (پایه 15) متصل میشود شما میتوانید از میکرو ATMEGA8L استفاده کرده و این مدار را با 3 ولت راه اندازی کنید ، کد مربوط به دیگر دستگاه های شرکت SONY را میتوانید از آدرس زیر بدست آورید:

<http://www.fet.uni-hannover.de/purnhage/>

فرستندهای RC6 :

این پروتکل ، برخلاف RC5 در اکثر دستگاه های صوتی تصویری جدید(تمامی دستگاه های CD چینی را پشتیبانی میکند) استفاده می شود . در این پروتکل تقریباً تمامی جزییات مانند RC5 است و فقط نحوه ارسال داده فرق دارد .
با دستور زیر میتوان دیتا را توسط این رابط به گیرنده ارسال کرد:

```
RC6SEND togglebit, address, command
```

Togglebit : نشان دهنده وضعیت پایه بعد از ارسال دستور است که میتواند صفر یا یک باشد .

Address : این مورد میتواند یکی از موارد جدول زیر باشد:

Device	Address
TV	0
VCR	5
SAT	8
DVD	4

Command : این گزینه مشخص کننده دستور ارسالی است (دستور مورد نظر به جای این کلمه نوشته میشود) دستورات

مخصوص هر کنترل را می توانید از سایت سازنده دریافت کنید در زیر کدهای مخصوص نوعی VCR که در HELP بسکام

موجود بود را مشاهده می فرمایید:

Command	Value	Command	Value
Key 0	0	Balance right	26
Key 1	1	Balance left	27
Key 2-9	2-9	Channel search+	30
Previous program	10	Channel search -	31
Standby	12	Next	32
Mute/un-mute	13	Previous	33
Personal preference	14	External 1	56
Display	15	External 2	57
Volume up	16	TXT submode	60
Volume down	17	Standby	61
Brightness up	18	Menu on	84
Brightness down	19	Menu off	85
Saturation up	20	Help	129
Saturation down	21	Zoom -	246
Bass up	22	Zoom +	247
Bass down	23		
Treble up	24		
Treble down	25		

مثال

```
$regfile = "m8def.dat"
```

```
$crystal = 4000000
```

```
Config Kbd = Portd
```

```
Dim A As Byte
```

```
Dim B As Byte
```

```
Q:
```

```
A = Getkbd()
```

```
If A > 15 Then
```

```
Goto Q
```

```
End If
```

```
B = Lookup(a , Dat)
```

```
Rc6send 0 ,5 , B
```

```
Goto Q
```

```
End
```

```
Dat:
```

```
Data 30 , 31 , 16 , 17 , 61 , 12 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 0
```

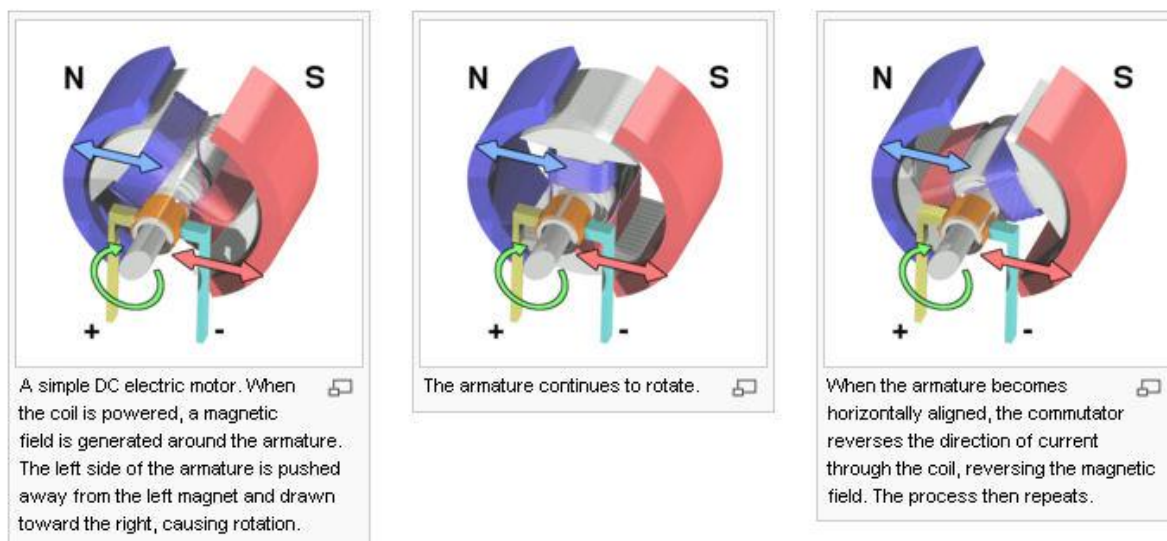
```
'program +,program -,volume +,volume -,power,sound on/off,1,2,3,4,5,6,7,8,9,0
```

راه اندازی انواع موتور های dc و پله ای

یکی از قطعات پرکاربرد در صنعت ، رباتیک و کنترل موتور های الکتریکی میباشند ، این موتور ها که خود به دو دسته ی عمده ی موتور های dc و ac (که خود اینها نیز به دسته های بزرگی تقسیم میشوند) تقسیم میشوند میتوانند با استفاده از کنترل کننده ی خود در یک زاویه ی خاص و به صورت چپ گرد یا راستگرد بچرخند . در این بخش به بررسی نحوه ی کنترل برخی از موتور های پرکاربرد پرداخته ایم .

موتورهای DC :

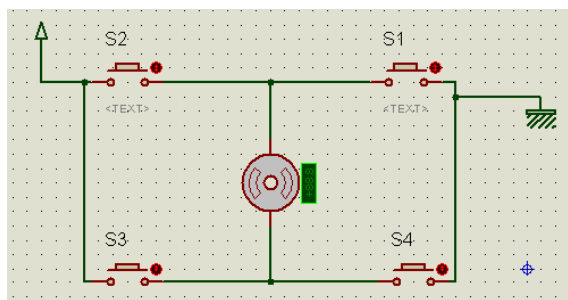
این موتور از یک آهنربای دائم و چند آهنربای الکتریکی تشکیل شده است ، در ساختمان این موتور آهن ربا های دائم از جنس آهن مغناطیسی و آهن ربا های مغناطیسی از سیم پیچ های که بر روی یک هسته پیچیده شده اند بوجود می آید . با اتصال پایه های موتور به ولتاژ DC ، جریان الکتریکی از طریق دو عدد جاروبک به سیم پیچ ها اعمال شده و در آنها یک میدان بوجود میاورد که این میدان باعث گردش موتور میشود.



در این نوع موتور ها ، سرعت وابسته به ولتاژ و گشتاور وابسته به جریان است و شما میتوانید جهت گردش ، و سرعت چرخش و مقدار گشتاور موتور را کنترل کنید .

کنترل جهت گردش موتور DC :

شما میتوانید با تغییر جهت جریان عبوری از موتور (برعکس کردن قطب ها ، (جای دوسیم موتور را عوض کنید)) جهت گردش آن را تغییر دهید :



در مدار بالا اگر کلید های S1 و S3 فشرده شوند موتور به سمت راست و اگر کلید های S2 و S4 فشرده شوند موتور به سمت چپ می چرخد. عملیات بالا را توسط میکرو ، و دو کلید انجام میدهیم، برنامه میکرو و مدار را در زیر مشاهده میفرمایید:

```
$regfile = "M8DEF.DAT" : $crystal = 8000000
```

```
Config Portc = Output : Config Portd = Input
```

```
Do
```

```
Debounce Pind.0 , 1 , Q :
```

```
Debounce Pind.1 , 1 , W
```

```
Loop
```

```
End
```

```
Q:
```

```
Reset Portc.1
```

```
Waitms 100 : Set Portc.0
```

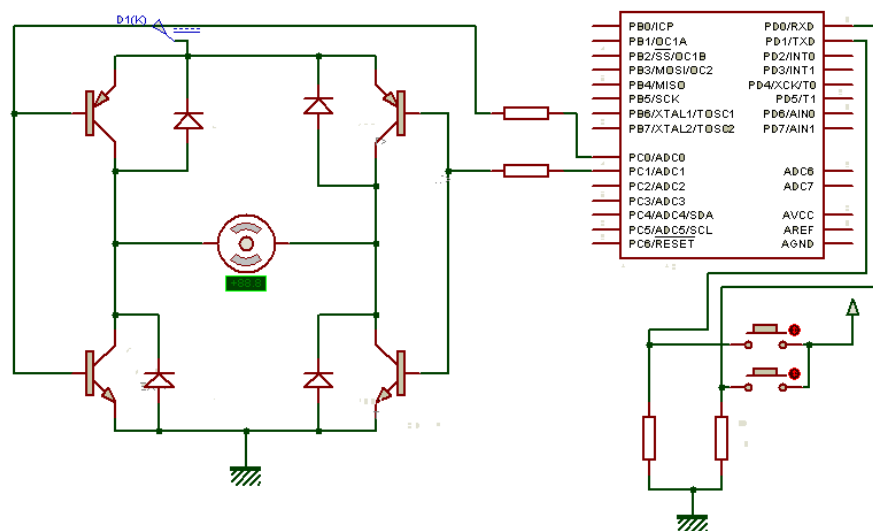
```
Return
```

```
W:
```

```
Reset Portc.0
```

```
Waitms 100 : Set Portc.1
```

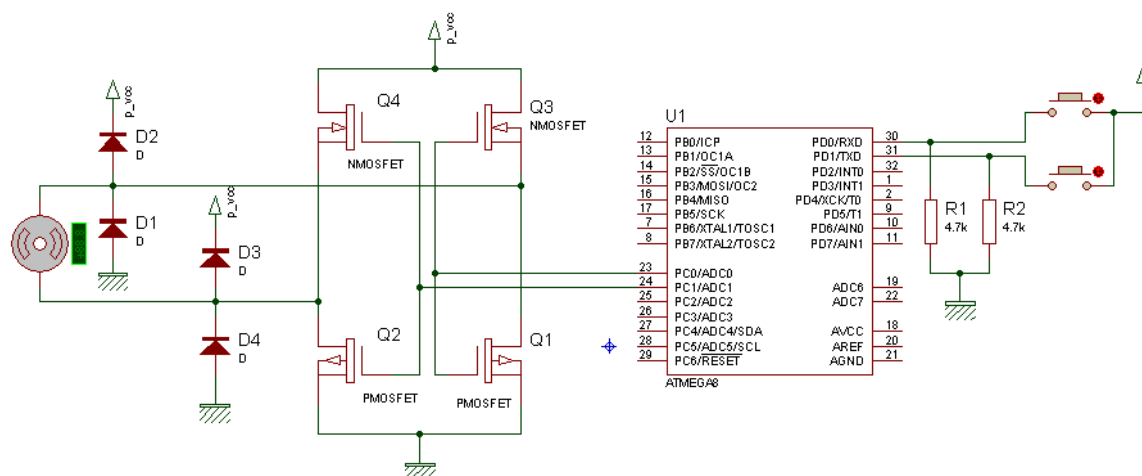
```
Return
```



در این مدار نقش دیود ها حفاظت از ترانزیستور ها در برابر ولتاژ القای موتور میباشد . از این مدار میتوانیم برای راه

اندازی موتور های کم قدرت که به جریان کمی نیاز دارند استفاده کنیم ، برای راه اندازی موتور های قوی تر شما میتوانید

از زوج دارلینگتون مانند 2N3055 یا از MOSFET مانند مدار زیر استفاده کنید:



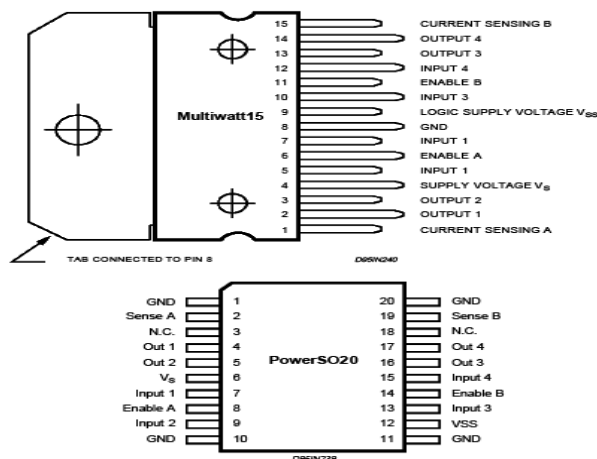
MOSFET ها و ترانزیستورها علاوه بر قیمت کم در توان های گوناگون وجود دارند که شما میتوانید مطابق نیاز خود آن را تهیه کنید .

روش دیگر برای راه اندازی موتور های DC استفاده از درایو موتور میباشد . در دنیای الکترونیک درایور های مختلفی برای راه اندازی موتور های مختلف ساخته شده است که از این دسته میتوانیم به درایو های مانند L298 و L293 و L293 و uln2003 و... که معمولا در اکثر قطعه فروشی ها وجود دارد اشاره کنیم در زیر توضیحات مربوط به ای سی L298 آورده شده است :

ای سی L298 :

ای ایسی در دونوع بسته بندی PowerSo20 و Multiwatt15 تولید و عرضه میشود ، ولتاژ تغذیه ی این ایسی 7 ولت بوده و میتواند 2 موتور را بصورت هم زمان راه اندازی کند . بیشترین ولتاژ و جریانی که برای هر موتور فراهم میشود ، به ترتیب 46 و 2 آمپر میباشد و محدوده دمای کار آن بین -40 تا 150 درجه سانتی گراد است .

در زیر شماره پایه ها و شکل قطعه را مشاهده میکنید:



Multiwatt15	PowerSo20	حس کننده a و b	وظیفه پایه
1;15	2;19	از این دو پایه به عنوان سنسور جریان استفاده میشود ، بین این دو پایه و زمین مقاومت متغیری قرار میگیرد ، بوسیله ی این مقاومت میتوان جریان بار را کنترل کرد (مقدار مقاومت بستگی به دقت کنترل دارد ، در صورتی که کنترل جریان	

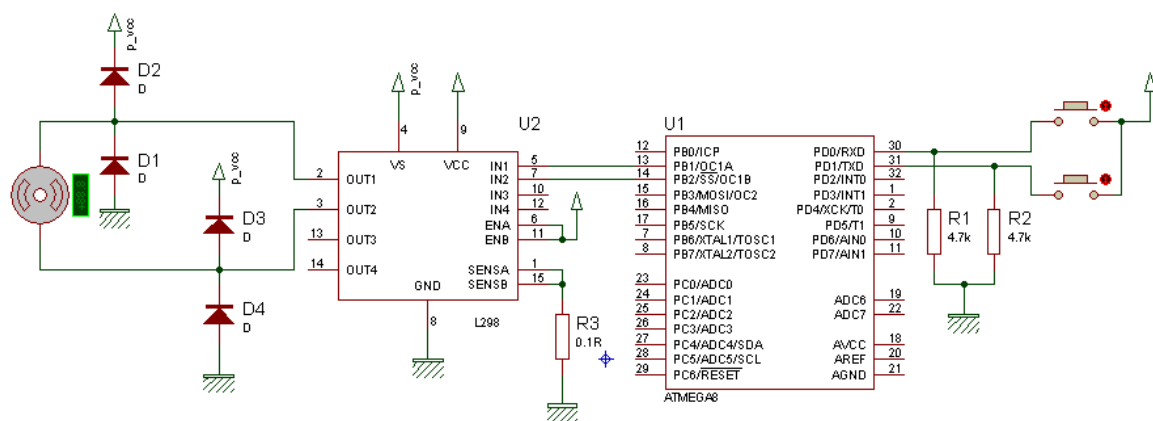
			مهم نیست این دو پایه را مطابق شکل با یک مقاومت به زمین متصل کنید.
2;3	4;5	خروجی 1 و 2	این دو پایه خروجی موتور a میباشند ، باید این دو پایه را مستقیما به موتور a متصل کنید.
4	6	تغذیه موتور	پایه 4 پایه تغذیه موتور ها می باشد ، شما میتوانید بسته به نوع موتور ولتاژ بین 1.5 تا 50 ولت را به این پایه اعمال کنید ، برای کاهش نویز بین این پایه و زمین حتما یک خازن 100nf قرار دهید.(این ولتاژ 1 ولت از تغذیه موتور بیشتر باشد)
5;7	7;9	ورودی 1 و 2	این دو پایه ورودی برای کنترلر موتور a می باشد ، این دو پایه از میکرو یا ... گرفته میشود.
6;11	8;14	فعال ساز خروجی a و فعال ساز خروجی b	این دو پایه فعال ساز خروجی های a و b میباشند ، با دادن یک ولتاژ بین 2.3 تا 7 ولت به پایه 6 (8) موتور a فعال میشود ، با صفر کردن پایه موتور غیر فعال میگردد و با دادن یک ولتاژ بین 2.3 تا 7 ولت به پایه 11 (14) موتور b فعال میشود ، با صفر کردن پایه موتور غیر فعال میگردد.
8	1,10,11,20	گراند	پایه های گراند مدار هستند ، گراند تغذیه موتور ها با گراند تغذیه ایسی ، باید به هم متصل شوند.
9	12		این پایه، پایه تغذیه ی ای سی است که 7 ولت می باشد، برای کاهش نویز بین این پایه و زمین حتما یک خازن 100nf قرار دهید.
10 ; 12	13 ; 15	وردی 3 و 4	این دو پایه ورودی برای کنترلر موتور b می باشد ، این دو پایه از میکرو یا ... گرفته میشود.
13;14	16;17	خروجی 3 و 4	این دو پایه خروجی موتور b میباشند ، باید این دو پایه را مستقیما به موتور b متصل کنید.
-	3;18	بدون اتصال	چیزی متصل نمیشود

همانطور که در جدول بالا مشاهده میکنید ، این قطعه دارای 4 پایه ی ورودی و چهار پایه ی خروجی و دو پایه ی توانا ساز میباشد ، هنگامی که شما یکی از پایه های ورودی را یک میکنید پایه ی خروجی متناظر آن درایو شده و میتواند ولتاژی تا منبع تغذیه (1.5 تا 50 ولت) با جریان حداکثر 2 آمپر را در خروجی ایجاد کند . در زیر وضعیت پایه های ورودی/خروجی و کنترل را مشاهده میکنید :

وضعیت پایه ها برای کنترل موتور :

موتور B	وضعیت پایه 2	وضعیت پایه 1	...	موتور A	وضعیت پایه 3	وضعیت پایه 2
متوقف	0	0	...	متوقف	0	0
راستگرد	1	0	...	راستگرد	1	0
چپگرد	0	1	...	چپگرد	0	1
متوقف	1	1	...	متوقف	1	1

منظور از صفر این است که پایه به گرانند متصل شده و منظور از یک وصل شدن پایه به ولتاژی بین 2.3 تا 7 ولت است . در میکرو کنترلر سطح صفر و یک توسط دستورات برنامه نویسی تعیین میشود ، با مشاهده ی مثال زیر میتوانید مطالب موجود را بهتر درک کنید :



```
$regfile = "M8DEF.DAT"
$crystal = 8000000
Config Portb = Output
Config Portd = Input
Do
If Pind.0 = 1 Then
Set Portb.1
```

```

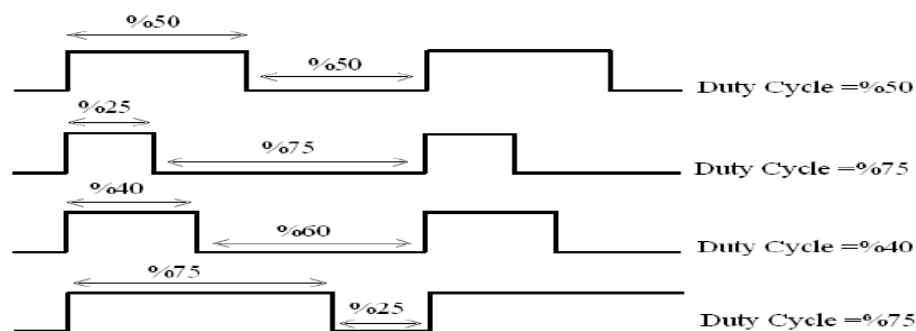
Reset Portb.2
End If
If Pind.1 = 1 Then
Reset Portb.1
Set Portb.2
End If
Loop
End

```

توجه داشته باشید که این درایور فاقد دیود های هرزگرد داخلی است و برای خنثی کردن ولتاژ القایی موتور و حفاظت از درایو شما باید از دیود های خارجی مانند مدار بالا استفاده کنید .

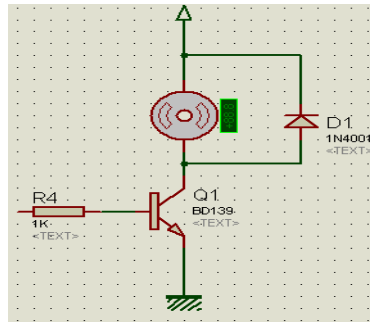
کنترل سرعت موتور :

قبلا گفتیم که سرعت موتور تابعی از ولتاژ دو سر آن است ، یعنی هرچه ولتاژ دو سر موتور بیشتر باشد سرعت آن بیشتر است و هر چقدر این ولتاژ کمتر باشد سرعت کم میشود ، در عمل برای کنترل دور موتور از مدولاسیون عرض پالس (PWM) که در بخش تایمر و کانتر گفته شد استفاده میشود. PWM پالسی است با فرکانس ثابت و پهنای متغیر که در زیر نمونه های از پالس PWM را مشاهده میکنید :



در صورتی که انتگرال میزان ولتاژ زیر نمودار در چهار حالت گفته شده را محاسبه کنید ، خواهید دید که مقدار ولتاژ متوسط در حالتی که عرض پالس بیشتر است (Duty cycle = %75) حداکثر مقدار خود را دارد .

در زیر یک مدار ساده برای کنترل سرعت یک موتور DC آورده شده است، نقش ترانزیستور در این مدار تقویت جریان است :



خروجی پالس PWM از میکرو به مقاومت R4 اعمال میشود ، در این حالت با زیاد شدن ولتاژ بیس (هنگامی که پالس pwm در سطح یک قرار دارد) ترانزیستور به اشباع رفته و ولتاژ تغذیه به دو سر موتور اعمال میشود ، هنگامی که ولتاژ ورودی از 0.6 ولت کمتر میشود (pwm به سمت صفر میل میکند) ترانزیستور وارد ناحیه ی قطع شده و پایه ی گراند موتور قطع میشود . شما همچنین میتوانید با استفاده از دو پالس PWM و پل H یا درایو علاوه بر کنترل جهت گردش دور موتور را نیز کنترل کنید.

در زیر مدار و برنامه کنترل سرعت و جهت چرخش یک موتور DC را مشاهده میکنید :

با استفاده از کلید تکی جهت گردش موتور معین میشود ، و با استفاده از دو کلید دیگر ، سرعت آن تغییر میکند.

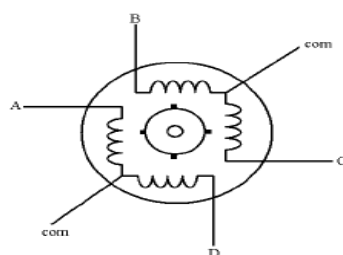
عملکرد	In2	In1
ترمز	0	0
راستگرد	پالس PWM	0
چپگرد	0	پالس PWM
ترمز	1	1

برنامه:

```
$regfile = "M8DEF.DAT" : $crystal = 8000000
Config Portb = Output : Config Portd = Input
Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Up , Compare B Pwm = Clear Down
, Prescale = 8
Dim A As Word , B As Word
A = 0 : B = 0
Do
If Pind.0 = 1 Then : A=A+10 : B=B+10 : Waitms 100 : End If
If Pind.1 = 1 Then : A = A -10 : B = B -10 : Waitms 100 : End If
If Pind.2 = 1 Then : B = 0 : A = 100 : Else : A = 0 : B = 100 : End If
Pwm1a = A : Pwm1b = B
Loop
End
```

موتورهای پله ای:

نوع دیگری از موتورهای الکتریکی موتور پله ای است، که در آن یک روتور درونی، شامل آهنرباهای دائمی توسط یک دسته از آهنرباهای خارجی که به صورت الکترونیکی روشن و خاموش می شوند، کنترل می شود. یک موتور پله ای ترکیبی از یک موتور الکتریکی DC و یک سلونوید است. موتورهای پله ای ساده توسط بخشی از یک سیستم دنده ای در حالت های موقعیتی معینی قرار می گیرند، اما موتورهای پله ای نسبتاً کنترل شده، می توانند بسیار آرام بچرخند. در زیر ساختمان یک موتور پله ای ساده را مشاهده میکنید:



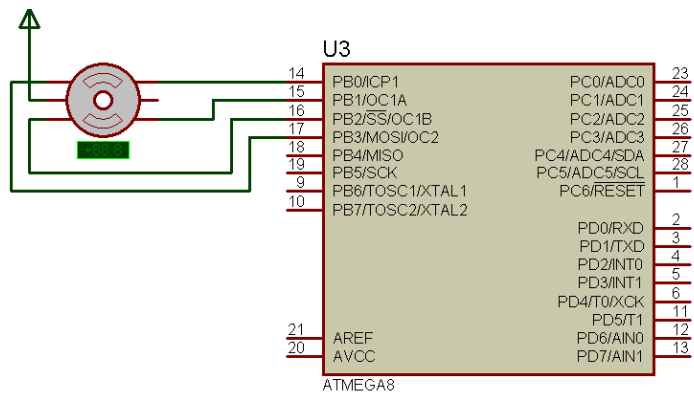
ساختمان یک موتور پله ای ساده

نحوه عمل کرد یک موتور پله ای با موتور DC تفاوت چندانی ندارد، برای راه اندازی این نوع موتور کافی است به ترتیب به سیم پیچ ها ولتاژ دهید، در اکثر موتور ها سیم های گراند از داخل به هم متصل میباشند، جدول زیر شما را در مورد طریقه دادن پالس راه نمایی میکند:

A	B	C	D	جهت موتور	A	B	C	D	جهت موتور
1	0	0	0	در جهت	0	0	0	1	خلاف جهت
0	1	0	0	عقربه های	0	0	1	0	عقربه های
0	0	1	0	ساعت	0	1	0	0	ساعت
0	0	0	1		1	0	0	0	

در زیر برنامه ای برای راه اندازی یک موتور پله ای 5 سیمه آورده شده است:

```
$regfile = "M8DEF.DAT" : $crystal = 8000000
Config Portb = Output
Do
Portb = &B00000001 : WAITMS 900
Portb = &B00000010 : WAITMS 900
Portb = &B00000100 : WAITMS 900
Portb = &B00001000 : WAITMS 900
```

Loop

End

برای کنترل دقیق تر زاویه حرکت موتور ،

آن را به صورت نیم پله راه اندازی میکنند ،

در جدول زیر طریقه پالس دهی را مشاهده میکنید:

A	B	C	D	جهت موتور	A	B	C	D	جهت موتور
0	0	0	1	در خلاف جهت عقربه های ساعت	1	0	0	1	موافق جهت عقربه های ساعت
0	0	1	1		1	0	0	0	
0	0	1	0		1	1	0	0	
0	1	1	0		0	1	0	0	
0	1	0	0		0	1	1	0	
1	1	0	0		0	0	1	0	
1	0	0	0		0	0	1	1	
1	0	0	1		0	0	0	1	

در زیر برنامه برای راه اندازی موتور به صورت نیم پله را مشاهده میکنید:

```
$regfile = "M8DEF.DAT" : $crystal = 8000000
```

```
Config Portb = Output
```

```
Do
```

```
Portb = &B00000001 : Waitms 50
```

```
Portb = &B00000011 : Waitms 50
```

```
Portb = &B00000010 : Waitms 50
```

```
Portb = &B00000110 : Waitms 50
```

```
Portb = &B00000100 : Waitms 50
```

```
Portb = &B00001100 : Waitms 50
```

```
Portb = &B00001000 : Waitms 50
```

```
Portb = &B00001001 : Waitms 50
```

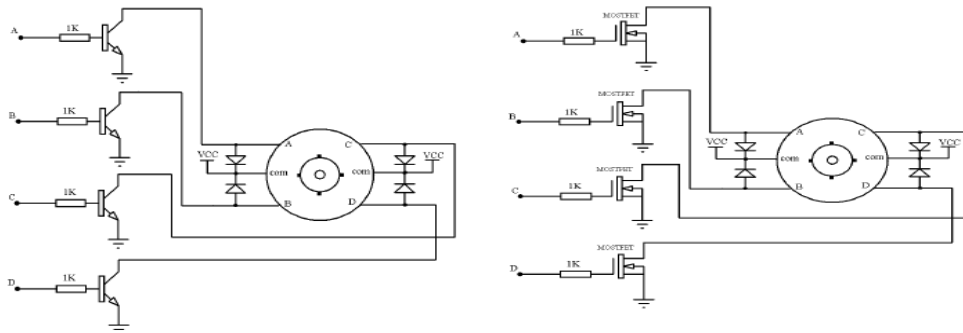
```
Loop
```

```
End
```

تاخیر زمانی سرعت موتور را معین میکند (چقدر طول میکشد تا موتور یک پله حرکت کند) .

برای راه اندازی موتور های قوی به یک راه انداز نیاز است (حداکثر جریان دهی میکرو 20 میلی آمپر است) شما میتوانید از

درایو های ULNXXXX یا ترانزیستورهای bjt یا fet استفاده کنید:

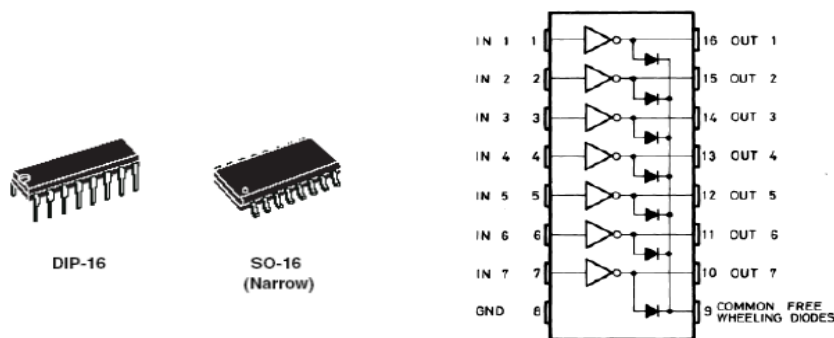


موتور های پله ای تک قطب :

درايوهاى ulnxxxx:

اين درايوها در ولتاژ ها و جريان هاى مختلف توليد ميشوند يکى از معروف ترين نوع آنها uln2003 است که در زير به

بررسى آن ميپردازيم:



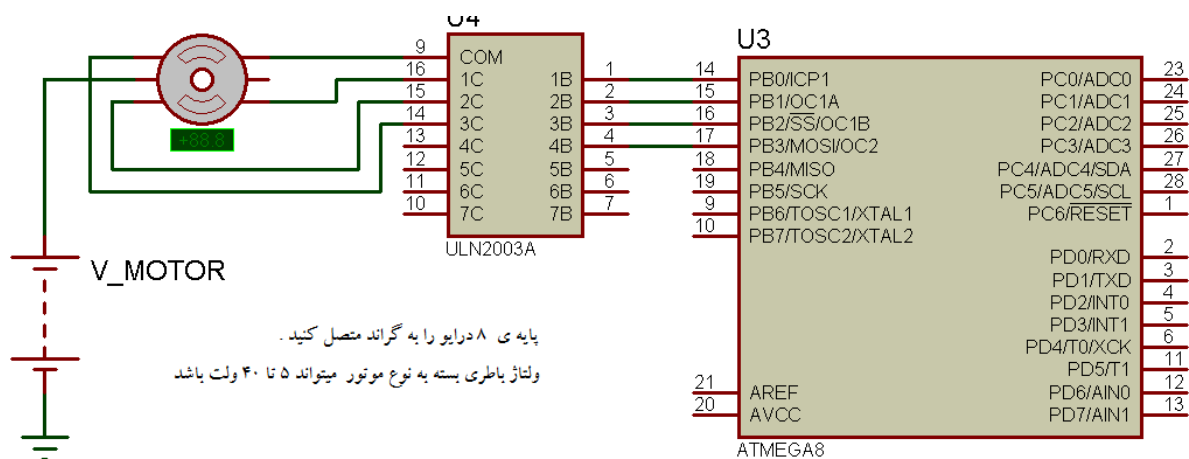
مشخصات:

Symbol	Parameter	Value	Unit
V_O	Output voltage	50	V
V_I	Input voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_C	Continuous collector current	500	mA
I_B	Continuous base current	25	mA

حداکثر جريان خروجى اين دراىو 500 ميلى آمپر ميباشد و ولتاژ خروجى آن به ميزان ولتاژ تغذيه موجود بين پايه گراند ايسى

و سيم مشترک موتور است و ميتواند حداکثر 50 ولت باشد ، ولتاژ ورودى آن نيز حداکثر 30 ولت است و ايسى از ورودى

جريان 25 ميلى آمپر را به ازاي بيشترين ولتاژ میکشد.در زير نحوه اتصال میکرو به ايسى وموتور را مشاهده ميکنيد:



برنامه نیز مانند برنامه قبلی است. (دیگر اطلاعات در مورد درایو ها و موتور را میتوانید در دیتاشیت آنها ببینید).

موتور های پله ای دو قطب :

کار با حافظه داخلی میکرو (eeprom):

تعداد زیادی از میکرو های avr دارای حافظه داخلی میباشند ، این حافظه دارای ویژگی های زیر است :

- ✓ اطلاعات این حافظه بر اثر قطع برق پاک نمیشود و میتواند تا سالهای زیادی محفوظ بماند .
- ✓ برای نگهداری این اطلاعات نیازی به باتری خارجی و ... نمیشود .
- ✓ دست یابی به حافظه مستقیما و توسط CPU انجام میشود .
- ✓ در بانک های 16 تا 256 بیتی در دسترس است و حجم آن بسته به میکرو کنترلر های مختلف متغیر است .
- ✓ این حافظه در تمامی اعضای خانواده های Atmega و atxmega و برخی از اعضای At91s وجود دارد .

Eeprom داخلی میکرو همیشه آماده به کار است و کافی است شما در آن بنویسید یا از آن بخوانید ، نوشتن در eeprom با دستور زیر انجام می شود :

```
WRITEEEPROM var , address
```

Var: متغیر یا عدد ثابتی است که قصد ذخیره آن را دارید.

Address: مکانی از حافظه میباشد که متغیر در آن ذخیره می شود .(در صورتی که آدرسی درج نشود متغیر در اولین مکان خالی ذخیره می شود و پیدا کردن آن با خداست).

در سری AVR عملیات نوشتن در حافظه از آدرس 00 شروع شده و تا پر شدن حافظه ادامه پیدا می کند ، در این حالت اگر داده ای در یک آدرس خاص نوشته شود ، جایگزین داده قبلی خواهد شد .

شما همچنین با دستور زیر میتوانید اطلاعات داخل حافظه را بخوانید:

```
READEEPROM var , address
```

Var: یک متغیر متناسب با مقدار اطلاعات میباشد ، که اطلاعات خوانده شده از آدرس درج شده ، در آن ریخته میشود .

Address: آدرسی است که باید اطلاعات از آن خوانده شود.

مثال:

```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
```

```
Dim A As Byte , C As Byte
```

```
C = 12
```

```
Writeeprom C , 1
```

```
Wait 1
```

```
Readeeprom A , 1
```

```
Locate 1 , 1 : Lcd A
```

```
End
```

در این مثال مقدار 12 در متغیر c ریخته شده است ، من این متغیر را در آدرس 1 حافظه داخلی ذخیره کردم ، و بعد از گذشت 1 ثانیه خانه 1 حافظه را خواندم و حاصل را در متغیر a ریختم و آن را روی lcd نمایش دادم . همانگونه که مشاهده خواهید کرد عدد خوانده شده برابر 12 میشود . مثالی دیگر:

```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

```
Config Lcd = 16 * 2
```

```
Config Lcdpin = Pin , Db4 = Portd.0 , Db5 = Portd.1 , Db6 = Portd.2 , Db7 = Portd.3 , E = Portd.4 , Rs = Portd.5
```

```
Config Adc = Single , Prescaler = Auto
```

```
Dim A As Word , B As Byte , C As Byte
```

```
Start Adc
```

```
Do
```

```
A = Getadc(0) : A = A / 2
```

```
Locate 1 , 1 : Lcd A
```

```
Wait 1 : Incr B
```

```
If B > 180 Then
```

```
Writeeprom A , C
```

```
C = C + 1 : B = 0
```

```
End If
```

```
If C > 3 Then
```

```
Readeeprom A , 0 : Locate 1 , 1 : Lcd A
```

```
Readeeprom A , 1 : Locate 1 , 9 : Lcd A
```

```
Readeeprom A , 2 : Locate 2 , 1 : Lcd A
```

```
Readeeprom A , 3 : Locate 2 , 9 : Lcd A
```

```
C = 0 : Else : Loop
```

```
End If
```

```
End
```

در برنامه بالا هر سه ساعت یک بار دمای محیط اندازه گرفته میشود و در حافظه داخلی میکرو ذخیره میشود ، بعد از گذشت 12 ساعت دما های ذخیره شده بر روی lcd به نمایش در میاید .

در کامپایلر بسکام امکان معرفی متغیر از نوع ERAM نیز وجود دارد ، این متغیر آخرین مقدار خود در اولین فضای خالی حافظه ی فلش ذخیره میکند ، مثال :

```
Dim b as byte, bx as ERAM byte
```

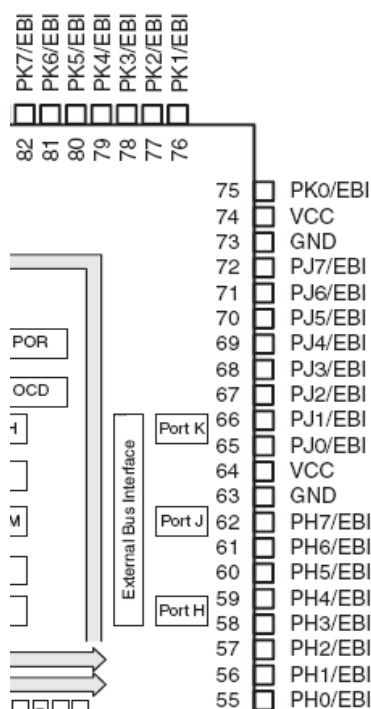
```
B= 1
```

```
Bx=b ' write to EEPROM
```

```
B=bx ' read from EEPROM
```

در سری Atxmega حافظه ی eeprom میتواند در حالت mapped پیکربندی شود ، ما نحوه ی راه اندازی این حافظه در سری Atxmega را بعد از تشریح شبکه های میکرو کنترلری بیان خواهیم نمود .

اتصال حافظه ی خارجی به میکروکنترلر :



برخی از میکروکنترلر های AVR دارای باس EBI (External Bus Interface) هستند . کاربرد با استفاده از باس های EBI که معمولاً به صورت 8 یا 16 یا 32 بیتی ارائه میشوند ، میتوانند انواع مختلفی از حافظه های خارجی همچون SRAM و SDRAM و تجهیزات الکترونیکی همچون LCD را با میکروکنترلر کنترل نمایند .

برای مثال میکروکنترلر ATXMEGA128A1 مطابق تصویر روبرو دارای سه باس EBI هشت بیتی است و توسط آن میتواند حافظه های 256 بایت تا 16 مگابایت را به سادگی بخواند و بنویسد . در این میکروکنترلر تمامی پایه های باس EBI دو جهته بود و میتوانند برای خواندن و نوشتن استفاده یا آدرس دهی استفاده شوند . برای کسب اطلاعات بیشتر در مورد نحوه ی راه اندازی حافظه های خارجی عبارت CONFIG XRAM را در راهنمای نرم افزار بسکام جستجو کنید .

راه اندازی WATCHDOG:

یکی از موارد مهم در هنگام انتخاب یک میکرو کنترلر قابلیت ثبات پردازنده (cpu) یا اصطلاحاً عدم هنگ کردن آن در شرایط مختلف (مانند نویز یا وجود ورودی های متناقض یا ...) میباشد. در تمامی منطق های دیجیتالی امکان بروز خطا، یکسان نبودن سرعت انتقال داده در لوازم جانبی و... وجود دارد، در این شرایط باید هنگ کردن cpu به عنوان هسته ی اصلی یک میکرو کنترلر را پذیرفت و برای بازنشانی آن به دنبال چاره بود

برای حل کردن مشکل بالا تایمری به نام WATCHDOG در میکرو کنترلر های مختلف تعبیه شده است، WATCHDOG یکی از تایمر های میکرو است که میتواند تا یک زمان خاص بشمارد و میکرو را ریست کند، این تایمر میتواند تا 8 زمان 16، 32، 64، 128، 256، 512، 1024 و 2048 و در بعضی از میکروها 4096، 8000 میلی ثانیه بشمارد، بعد از سپری شدن زمان میکرو ریست میشود و برنامه دوباره از ابتدا اجرا میشود، عمل این تایمر کاملاً مجزا از CPU و سایر واحد های جانبی بوده و کلاک آن نیز به صورت مستقیم از منبع کلاک تعیین شده برای پردازنده تامین میشود.

راه اندازی WATCHDOG به فرم زیر است :

CONFIG WATCHDOG = time

Time : یکی از زمان های گفته شده در بالا میباشد (16، 32، 64، 128، 256، 512، 1024 و 2048 و در بعضی از میکروها 4096، 8192 میلی ثانیه)

با دستور Start Watchdog تایمر شروع به شمارش میکند و پس از سپری شدن زمان time میکرو ریست میشود.

مثال:

```
$regfile = "m16def.dat"
$crystal = 8000000
Config Porta.0 = Input
Config Porta.1 = Output
Config Watchdog = 1024
Do
If Pina.0 = 0 Then
Set Porta.1
Else
```


Start Watchdog

End If

Loop

End

در مثال بالا پایه a.0 (که در حالت عادی 1 است) مدام چک میشود و در صورتی که پایه 0 شود میکرو بعد از 1024 میلی ثانیه ریست میشود.

بعد از رخ دادن ریست ، کاربر باید تایمر watchdog را با استفاده از دستور Reset watchdog ریست نماید ، کاربر همچنین می تواند با استفاده از دستور STOP WATCHDOG ، تایمر watchdog را متوقف کند .

در برخی از میکرو کنترلر های avr پرچم های برای این تایمر در نظر گرفته شده است ، کاربر میتواند با خواندن و مقدار دهی این پرچم ها کنترل بیشتری بر روی این تایمر داشته باشد . مثال :

```
$regfile = "xm128a1def.dat"

$crystal = 32000000

$hwstack = 64

$swstack = 64

$framesize = 64

'include the following lib and code, the routines will be replaced since they are a workaround

$lib "xmega.lib"

$external _xmegafix_clear

$external _xmegafix_rol_r1014 'First Enable The Osc Of Your Choice

Config Osc = Enabled , 32mhzosc = Enabled 'configure the systemclock

Config Sysclock = 32mhz , Prescalea = 1 , Prescalebc = 1_1

Config Com1 = 19200 , Mode = Asynchronous , Parity = None , Stopbits = 1 , Databits = 8

Config Input1 = Cr , Echo = CrLf ' CR is used for input, we echo back CR and LF

Open "COM1:" For Binary As #1

' ^^^^ change from COM1-COM8

Print #1 , "Xmega revision:" ; Mcu_revid ' make sure it is 7 or higher !!! lower revs have many flaws

Config Watchdog = 4000 'after 4 seconds a reset will occur if the watchdog is enabled

'possible value : 8 , 16,32,64,125,250,500,1000,2000,4000,8000

'these values are clock cycles, based on a 1 KHz clock !!!

Dim W As Word , B As Byte
```

Do

W = W + 1

Print W

Waitms 500

B = Inkey()

If B = "a" Then

Start Watchdog

Print "start"

Elseif B = "b" Then

Stop Watchdog

Print "stop"

Elseif B = "c" Then

Config Watchdog = 8000

Print "8 sec"

Elseif B = "d" Then

Reset Watchdog

Print "reset"

End If

Loop

End

بهینه سازی مصرف توان :

در مدارات الکترونیکی ممکن است در برخی از دوره های زمانی میکروکنترلر فعالیتی را انجام نداده و اصطلاحاً در حالت آماده به کار باشد ، این حالت ممکن است برای برخی از بخش های داخلی میکروکنترلر نیز وجود داشته باشد . این حالت در مداراتی که تغذیه ی خود را از باتری تامین میکنند باعث افزایش جریان مصرفی از باتری و کاهش زمان عمل کرد مدار خواهد شد .

در میکروکنترلر های خانواده ی AVR میتوان با استفاده از دو روش زیر جریان مصرفی میکروکنترلر تا حد میکروآمپر کاهش داد .

مد های کم مصرفی :

در بسکام با دستور زیر میتوان مد کاری میکروکنترلر را تعیین کرد :

CONFIG POWERMODE mode

در این دستور MODE یکی از دستورات زیر است :

IDLE : در این مد CPU میکروکنترلر متوقف میشود اما واحد های SPI و USART و Analog Comparator و ADC و 2-wire Interface Serial و Timer/Counters و Watchdog و سیستم وقفه در صورتی که قبلاً فعال شده باشند همچنان به کار خود ادامه میدهند. در این مد کلاک حافظه ی فلش و کلاک CPU متوقف میشود . رخ دادن هر یک از وقفه ها ، سر ریز شدن تایمر / کانتر ، ورود داده ی جدید به بافر دریافت داده و ... باعث استارت CPU و ادامه ی روند عادی کار خواهد شد .

ADCNOISE : در این مد نیز CPU متوقف میشود اما واحد های مبدل آنالوگ به دیجیتال ، ورودی های وقفه ی خارجی ، تایمر کانتر شماره ی 2 و تایمر واچ داگ و کنترل کننده ی واسط سریال دو سیمه در صورتی که قبلاً فعال شده باشند به کار خود ادامه میدهند ، در این مد کلاک پورت های ورودی / خروجی ، کلاک حافظه ی فلش و کلاک CPU متوقف میشود . این مد در سری atxmega وجود ندارد .

در این مد ADC با دقت بالا شروع به اندازه گیری نویز محیط می کند و با اتمام شدن تبدیل و رخ دادن وقفه ی پایان تبدیل CPU مجدداً شروع به کار میکند ، علاوه بر وقفه ی ADC ، رخ دادن ریست خارجی ، بازنشانی تایمر واچ داگ ، وقفه ی تایمر واچ داگ ، فیوز بیت Brown-out ، دریافت داده توسط واسط سریال 2 سیمه ، رخ دادن وقفه ی تایمر / کانتر 2 ، وقفه ی EEPROM و رخ دادن وقفه بر روی پایه های وقفه ی خارجی نیز میتواند CPU را راه اندازی کند .

POWERDOWN: در این مد نوسان ساز خارجی متوقف میشود، اما ورودی های وقفه ی خارجی، تایمر واچ داگ و واسط سریال دو سمیه در صورتی که قبلا فعال شده باشند به کار خود ادامه میدهد. رخ دادن ریست خارجی، بازنشانی تایمر واچ داگ، وقفه ی تایمر واچ داگ، فیوز بیت Brown-out، دریافت داده توسط واسط سریال 2 سیمه و رخ دادن وقفه بر روی پایه های وقفه ی خارجی میتواند CPU را راه اندازی کند.

در صورتی که برای بیدار کردن CPU از ورودی های وقفه استفاده شود، زمان اعمال پالس باید از زمان بیدار باش تعیین شده در فیوز بیت CKSEL بیشتر باشد. برای کسب اطلاعات بیشتر در این رابطه به دیتاشیت فارسی میکروکنترلر های Avr که در بخش ضمایم آورده شده است مراجعه نمایید.

POWERSAVE: این مد تقریبا مشابه با مد قبلی است، در این مد تایمر / کانتر 2 نیز به امکاناتی که در مد قبلی فعال بودند اضافه میشود. در صورتی که تایمر / کانتر 2 در مد synchronous پیکربندی شود، کلاک خود را از منبع مجزا دریافت خواهد کرد، در این حالت سر ریز شدن تایمر یا رخ دادن هر یک از وقفه های آن میتواند به عوامل فعال ساز cpu در مد قبلی افزوده شود.

STANDBY: این مد مشابه مد POWERDOWN است، ولی در آن نوسان ساز اصلی سیستم متوقف نمیشود. در این مد cpu بعد از 6 پالس کلاک از حالت توقف خارج میشود.

ExtendedStandby: این مد مشابه با مد Power-save است، ولی در آن نوسان ساز اصلی سیستم متوقف نمیشود. در این مد cpu بعد از 6 پالس کلاک از حالت توقف خارج میشود.

در میکرو کنترلر های سری atxmega علاوه بر مد های کاری بالا، با استفاده از دستور زیر میتوان بخش های مختلف میکروکنترلر را خاموش یا روشن نمود:

CONFIG POWER_REDUCTION= dummy, device=ON|OFF

خاموش یا روشن کردن هر بخش با نوشتن نام آن به جای device و قرار دادن دستور on یا off در مقابل آن انجام میشود. با خاموش شدن هر بخش کلاک مربوط به آن قطع شده و امکان دریافت یا ارسال داد به آن وجود نخواهد داشت.

AES: واحد زمر گذاری پیشرفته

EBI: پورت های موازی جهت ادرس دهی حافظه های خارجی

RTC: واحد زمان شمار

EVSYS: سیستم ثبت رخ داد

DMA: گذرگاه dma (دستیابی مستقیم به حافظه)

DACA, DACB: مبدل های دیجیتال به آنالوگ

ACA, ACB: مقایسه کننده های آنالوگ

ADCA, ADCB: مبدل های آنالوگ به دیجیتال

TWIC, TWID, TWIE, TWIF: رابط های سریال دو سمیه

USARTC0, USARTC1, USARTD0, USARTD1, USARTE0, USARTE1, USARTF0, USARTF1: پورت های سریال

SPIC, SPID, SPIE, SPIF: رابط های سریال سنکرون

TCC0, TCC1, TCD0, TDC1, TCE0, TCE1, TCF0, TCF1: تایمر ها / کانتر ها

High Resolution Extension on port : HIRESC, HIRESD, HIRESE, HIRESF

از جمله بخش های هستند که میتوانند توسط دستور بالا روشن و خاموش شوند. مثال :

```
$regfile = "xm128a1def.dat"
$crystal = 2000000 '2MHz
$hwstack = 64
$swstack = 40
$framesize = 40
$lib "xmega.lib" : $external _xmegafix_clear : $external _xmegafix_rol_r1014
Config Osc = Enabled
Config Sysclock = 2mhz '2MHz
' YOU CAN MINIMIZE POWER CONSUMPTION FOR EXAMPLE WITH :
' 1. Use Low supply voltage
' 2. Use Sleep Modes
' 3. Keep Clock Frequencys low (also with Precsalers)
' 4. Use Powe Reduction Registers to shut down unused peripherals
'With Power_reduction you can shut down specific peripherals that are not used in your application
'Paramters: aes,dma,ebi,rtc,evsys,daca,dacb,adca,adcb,aca,acb,twic,usartc0,usartc1,spic,hiresc,tcc0,tcc1
Config Power_reduction = Dummy , Aes = Off , Twic = Off , Twid = Off , Twie = Off , Aca = Off , Adcb = Off , Tcc0 = Off , Tcc1 = Off , Dma = Off
'For the following we need the EVENT System therefore we do not shut down EVENT SYSTEM
Config Com1 = 9600 , Mode = Asynchronous , Parity = None , Stopbits = 1 , Databits = 8
Open "COM1:" For Binary As #1
Waitms 2
Print #1 , " aes,dma,ebi,rtc,evsys,daca,dacb,adca,adcb,aca,acb,twic,usartc0,usartc1,spic,hiresc,tcc0,tcc1 shut down "
End
```

ضمائم:

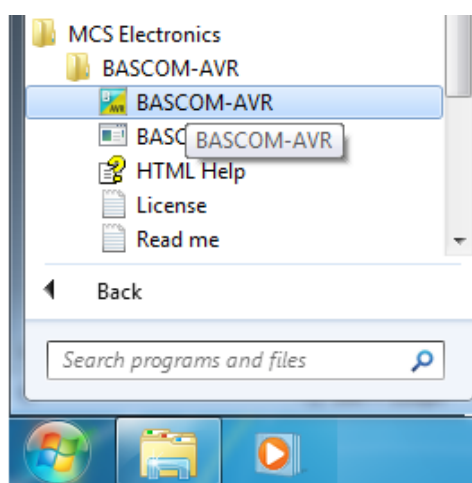
- ✓ در این فصل سایر مطالب و نکاتی که شما برای کار با میکرو کنترلر های AVR نیاز دارید آورده شده است .
- ✓ برخی از مطالب به صورت خلاصه از دیگر کتاب های آموزشی برداشته شده اند ، شما میتوانید با مراجعه به مرجع ، آموزش کامل را مشاهده کنید .
- ✓ قبل از خواندن مطالب این بخش باید فصل های قبلی را به صورت کامل خوانده باشید ، وگرنه در درک مطالب دچار مشکل میشوید .

ضمیمه 1: طریقه ی نصب بسکام :

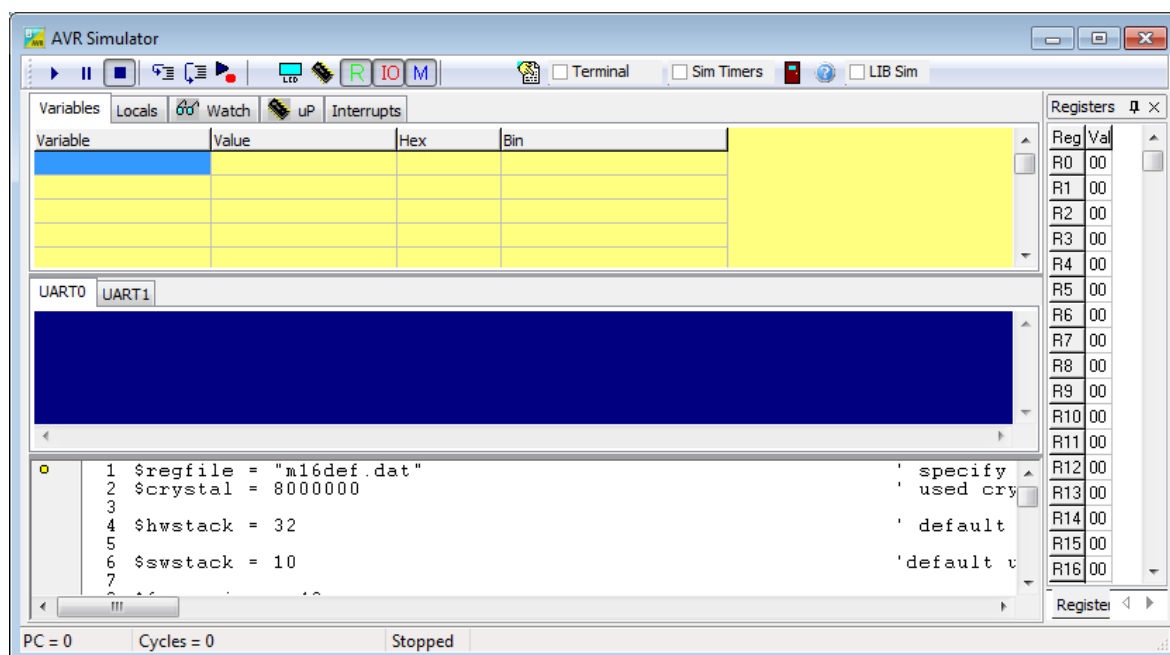
با مراجعه به سایت www.mcselec.com نسخه ی دموی نرم افزار را دانلود کنید یا نسخه ی کامل آن را خریداری نمایید .

در سایت mcselec مراحل نصب برای آخرین ورژن آورده شده است .

بعد از انجام مراحل بالا نرم افزار بسکام از منوی استارت و مسیر all programs در دسترس شماست :

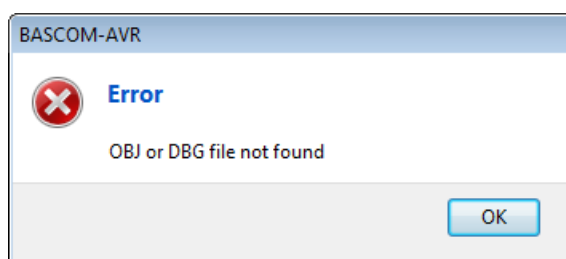


ضمیمه 2: آشنایی با محیط شبیه سازی بسکام (simulate):



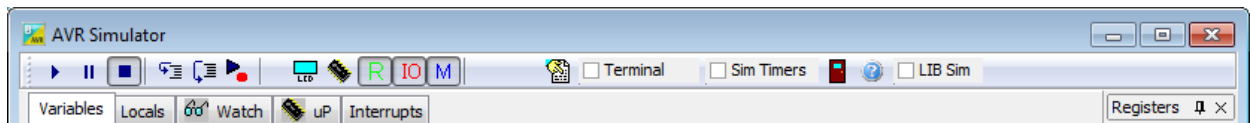
از این محیط برای شبیه سازی برنامه های نوشته شده در نرم افزار بسکام استفاده میشود. هر چند این محیط در مقایسه با سایر نرم افزار های شبیه ساز (نظیر پروتوس) دارای امکانات کمتری میباشد، اما یادگیری نحوه ی کار با آن شما را تست آنی برنامه و رفع اشکالات احتمالی آن یاری میدهد.

شما میتوانید از منوی `Program > simulate` یا با زدن کلید `f2` به این بخش وارد شوید، البته قبل از انتخاب این بخش باید برنامه ی خود را کامپایل نمایید. محیط شبیه ساز برای اجرای برنامه نیاز به دو فایل `name.OBJ` و `name.DBG` دارد، در صورتی که هنگام شبیه سازی با پیغام زیر شدید آن را تایید کرده و مراحل موجود در ادامه را دنبال کنید:



ابتدا برنامه را کامپایل کنید و دوباره محیط شبیه سازی را فعال کنید ، در صورتی که مشکل حل نشد به مسیر Options>compiler>output رفته و گزینه های Debug file و avr Studio Object file را تیک بزنید ، در صورتی که مشکل حل نشد محل ذخیره برنامه را تغییر بدهید و مجدداً آن را کامپایل کنید .

اولین بخشی که بعد از ورود به محیط simulator مورد توجه قرار میگیرد تولبار کنترل میباشد :



RUN : با فشردن این دکمه شبیه سازی آغاز می شود .

PAUSE : باعث توقف موقت شبیه سازی می شود و با فشردن دکمه RUN شبیه سازی ادامه پیدا می کند .

STOP : باعث توقف کامل شبیه سازی برنامه جاری می شود .

STEP INTO CODE : با استفاده از این کلید می توان برنامه را خط به خط اجرا نمود و هنگام فراخوانی توابع به داخل

آنها رفته و مراحل اجرای آنها را بررسی کرد . این کار را با فشردن کلید F8 نیز می توانید انجام دهید . بعد از هر بار اجرای این دستور شبیه سازی به حالت PAUSE می رود .

STEP OVER : این دکمه شبیه دکمه قبلی است با این تفاوت که در هنگام فراخوانی توابع به داخل SUB ROUTINE

نخواهید رفت . این کار را می توانید با فشردن کلید SHIFT+ F8 نیز انجام دهید .

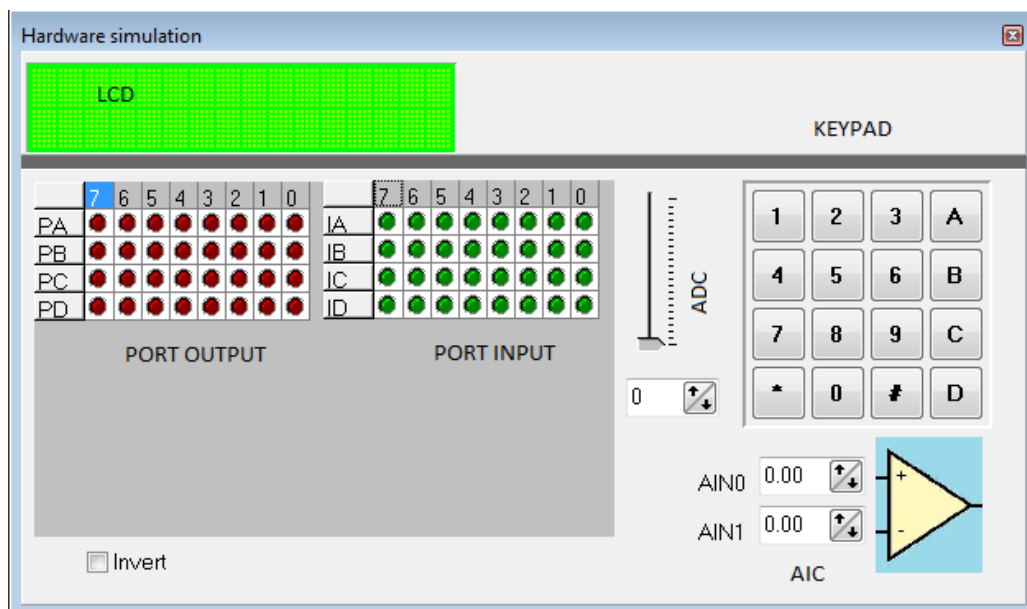
RUN TO : دکمه RUN TO شبیه سازی را تا خط انتخاب شده انجام میدهد و سپس به حالت PAUSE میرود (خط جاری

باید شامل کدهای قابل اجرا باشد) .

Show hardware emulation : با کلیک بر روی این گزینه پنجره زیر باز می شود . در این پنجره امکانات سخت افزاری

نظیر lcd ، مبدل آنالوگ به دیجیتال ، کیپد 4*4 ، پورت ها و همچنین ورودی مقایسه کننده ی آنالوگ موجود میباشد .

در این بخش LED های قرمز رنگ نماد پورت های خروجی و LED های سبز رنگ نماد پین های ورودی میباشند ، در صورتی که روی آنها کلیک کنید آنها تغییر رنگ میدهند (مانند این است که پایه را یک (led روشن) یا صفر کرده اید).



PORT OUTPUT : خروجی پورت های میکرو میباشد و تعداد آن بستگی به تعداد پایه های میکرو دارد (مثلا برای ATMEGA8

به 3 ردیف کاهش می یابد) هنگامی که شما در برنامه از دستورات set یا ... استفاده میکنید وضعیت این led ها تغییر میکند)

PORT INPUT : این قسمت به عنوان ورودی پورت ها میباشد و تعداد آن مانند حالت قبل به تعداد پایه های میکرو بستگی

دارد ، هنگامی شما در برنامه از دستور debounce یا ... (برای اتصال کلید به میکرو یا ...) استفاده میکنید ، میتوانید با کلیک

کردن روی led مورد نظر وضعیت آن را تغییر دهید

ADC : این قسمت به عنوان ورودی مبدل آنالوگ به دیجیتال استفاده میشود . با تغییر دادن میتوانید به کانال

دلخواه بروید و با تغییر دادن میتوانید مقدار ورودی را تغییر دهید.

KEYPAD : این قسمت مانند یک کیبورد 4*4 است که به یکی از پایه های میکرو متصل شده ، هنگامی که در برنامه دستورات


پیکربندی KBD را وارد میکنید این وسیله به پورت ذکر شده در برنامه متصل میشود .

AIC : این بخش ورودی مقایسه کننده ی آنالوگ به کار میباشد .


LCD : این بخش یک lcd است که متغیر یا اشکالی که در برنامه نوشته اید ("lcd "fhdhgd") را نمایش میدهد ، سطر

و ستون آن توسط دستور $\text{Config Lcd} = x * y$ در برنامه تعیین میشود .


INVERT : این قسمت برای معکوس کردن وضعیت پایه های ورودی و خروجی (led های سبز و قرمز) استفاده میشود.

Enable/disable real hardware emulation  : با انتخاب این گزینه شما میتوانید میکرو را به پورت سریال کامپیوتر خود


متصل کرده و برنامه داخل آن را اجرا کنید (این امکان فقط در بعضی از میکرو ها نظیر 2313 و... وجود دارد)


display REGISTERS windows  : این گزینه پنجره ثباتها را با مقادیر قبلی نمایش می دهد . مقدارهای نشان داده شده در

این پنجره هگزادسیمال می باشد که برای تغییر هر کدام از آنها روی خانه مربوطه کلیک کرده و مقدار جدید را وارد کنید


Display I/O REGISTERS windows  : برای نمایش ثباتهای ورودی ها و خروجی ها استفاده می شود . که مانند مورد

قبلی قابل مقدار دهی است.

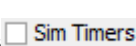
Display memory windows  : برای نمایش مکان های مختلف حافظه استفاده میشود .


Refresh variables  : این گزینه برای تازه سازی متغیر ها استفاده میشود (هنگامی که متغیری تغییر کرده باشد ، بازدن

این گزینه تغییرات اعمال میشوند .

Terminal ☐  : این گزینه برای استفاده از محیط ترمینال ایمولاتور میباشد (شما میتوانید با فعال کردن این گزینه از

محیط ترمینال برای شبیه سازی واقعی استفاده کنید).

Sim Timers ☐  : هنگامی قصد شبیه سازی تایمر ها را دارید باید این گزینه را تیک بزنید.

Exet  : بازدن این گزینه محیط شبیه سازی بسته میشود.

قسمت دوم بخش کنترل داده ها میباشد که تصویر آن را در زیر مشاهده میکنید :

Variables			
Locals	Watch	uP	Interrupts
Variable	Value	Hex	Bin

Variables: شما میتوانید با نوشتن نام متغیر های موجود در برنامه و زدن کلید enter میتوانید مقادیر دسیمال و مقدار hex و مقدار باینری آن را در بخش های مربوطه مشاهده کنید . شما همچنین میتوانید مقداری را به آن نسبت دهید ، برای این کار مقدار را در قسمت های گفته شده به فرم مورد نیاز بنویسید .

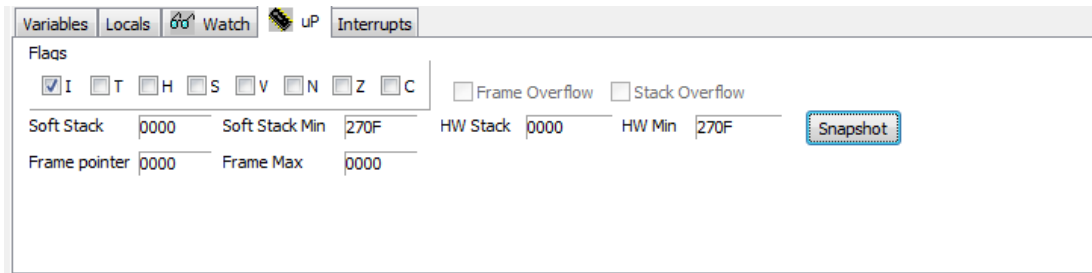
Locals: این گزینه مانند حالت قبل است ، تنها تفاوت در این است که مورد قبل متغیر های موجود در حلقه اصلی و این مورد متغیر های موجود در زیر برنامه ها را نشان میدهد:

Variable	Value	Hex	Bin

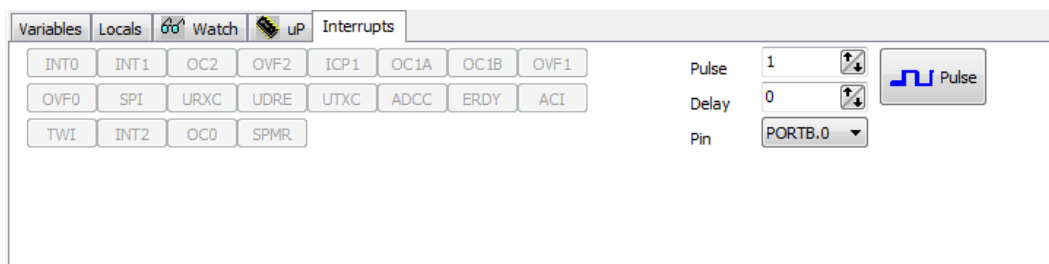
WATCH: این گزینه برای وارد کردن وضعیتی که باید ارزیابی شود مورد استفاده قرار می گیرد . برای استفاده از این بخش حالت مورد نظر را در مکان مورد نظر تایپ نموده و دکمه add را فشار دهید تا وضعیت در کادر نمایش داده شود . هنگامیکه دکمه MODIFY فشار داده شود ، وضعیت مورد نظر مورد بازنگری میشود و عمل متناظر با آن انجام میشود. برای حذف هر وضعیت آنرا در کادر انتخاب کرده انتخاب کرده و دکمه REMOVE را فشار دهید . هنگامی که وضعیت مورد نظر صحیح شد شبیه سازی در حالت PAUSE قرار خواهد گرفت.

Variables	Locals	Watch	uP	Interrupts
<div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border: 1px solid black; width: 400px; height: 100px;"></div> <div> <div>Add</div> <div>Modify</div> <div>Remove</div> <div style="border: 1px solid black; width: 150px; height: 20px; margin-top: 5px;"></div> </div> </div>				

up: در این قسمت میتوان وضعیت رجیستر های داخلی میکرو را مشاهده کرد (رجیستر تایمر ها ، adc و ...). ابتدا بر روی Snapshot کلیک کنید ، (به طور پیش فرض انتخاب شده است) و در زمان دلخواه دوباره روی آن کلیک کنید ، کادری باز میشود و در آن وضعیت رجیستر های انتخاب شده نمایش داده میشود ، برای اجرای دوباره روی کلید stop کلیک کنید



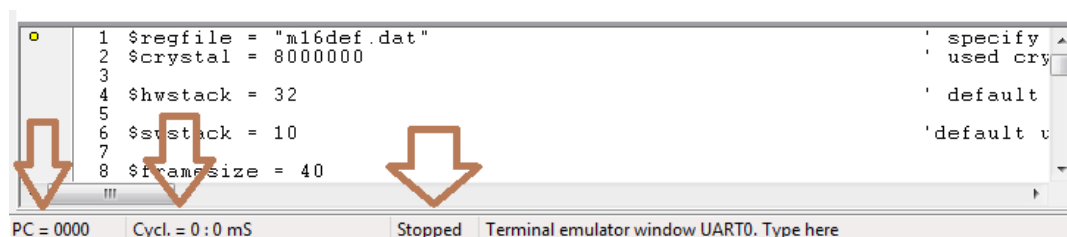
Interrupts: در این قسمت منابع وقفه خارجی میکرو مورد استفاده نمایش داده میشود (تنها در صورتی که در برنامه از آنها استفاده کنید فعال میشوند) همچنین شما میتوانید به پایه های ورودی کانتر های میکرو، یک پالس مریعی با فرکانس دلخواه اعمال نمایید.



قسمت سوم uart emulator میباشد که از آن برای نمایش داده دریافتی از پورت سریال میکرو کنترلر استفاده میشود. توسط این بخش میتوانید برنامه های را که برای پورت سریال نوشته اید را شبیه سازی کنید.



آخرین قسمت در این بخش محل نمایش برنامه میباشد، در زیر این قسمت وضعیت cpu میکرو و زمان که از آغاز شبیه سازی گذشته نمایش داده میشود:



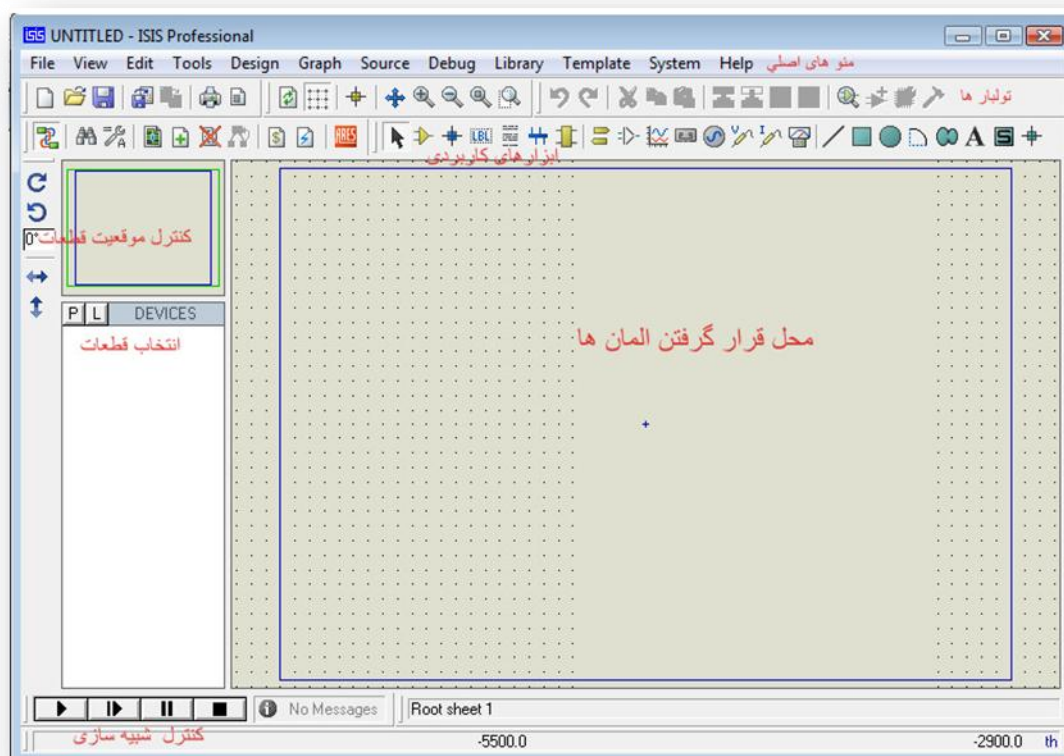
شبیه سازی یک برنامه:

ضمیمه 3: شبیه سازی میکرو کنترلر AVR با نرم افزار پروتوس (آشنایی مقدماتی)

بعد از نصب و فعال کردن نرم افزار پروتوس روی ایکون آن در منوی استارت کلیک کنید تا نرم افزار باز شود:



محیط نرم افزار به شکل زیر است :



✓ در قسمت منابع اندازه گیری ، مجموعه ای از منابع ولتاژ، جریان، اسیلوسکوپ، ولت متر ، سیگنال ژنراتور ، منابع پالس (ac، مربعی ،...) وجود دارد که شما میتوانید آنها را انتخاب کنید و در صفحه بگذارید (ابتدا روی آنها کلیک کنید ، سپس به صفحه آمده و در نقطه مورد نظر دوباره کلیک کنید تا قطعه در آنجا گذاشته شود).

✓ در قسمت ابزار ویرایش ، ابزاری برای ذخیره و تهیه پرینت و ... و بزرگ و کوچک کردن صفحه و .. وجود دارد که در صورت نیاز میتوانید آنها را به کار ببرید.

✓ در قسمت کنترل شبیه سازی 4 کلیک برای شروع و مکث و توقف و دیباگ کردن برنامه وجود دارد .

✓ در قسمت انتخاب قطعه ، شما میتوانید قطعات بکار رفته در مدار مورد نظرتان را انتخاب کنید و سپس آنها را طبق

نقشه به هم متصل کرده و شبیه سازی کنید ، در زیر با یک مثال این عمل توضیح داده میشود:

مدار مربوط به یک شمارنده می باشد ، که با استفاده از تایمر/کانتر 1 پالس های اعمالی به پایه T1 میکرو مگا 16 را می شمارد و

روی یک 16*2 lcd نمایش میدهد (در صورتی که مفهوم دستورات زیر را نمیدانید ، نگران نباشید ؛ در آینده ای نزدیک

تمامی آنها را خواهید آموخت) :

```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

```
Config Lcdpin = Pin , Db4 = Portd.2 , Db5 = Portd.3 , Db6 = Portd.4 , Db7 = Portd.5 , E = Portd.1 , Rs = Portd.0
```

```
Config Lcd = 16 * 2
```

```
Config Timer1 = Counter , Edge = Rising
```

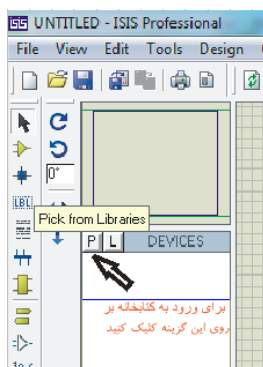
```
Do
```

```
Locate 1 , 1 : Lcd Counter1
```

```
Loop
```

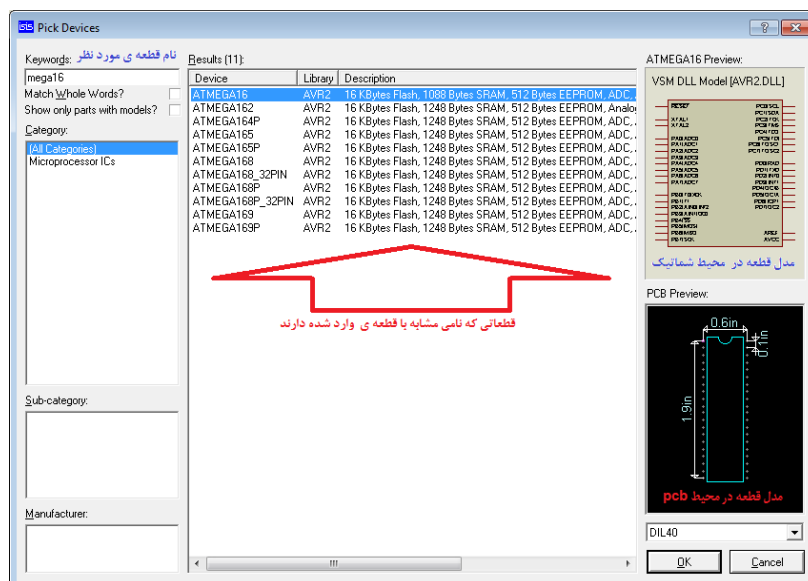
```
End
```

اولین قدم آوردن قطعات به صفحه ی شماتیک میباشد ، برای این کار روی گزینه p کلیک کنید .



توجه داشته باشید که برای کلیک روی ایکون pick from li... (p) باید ابتدا روی ایکون Selection mode (موس مشکی) کلیک

کنید. بعد از کلیک روی ایکون p پنجره لایبری باز میشود :



در قسمت "محل نوشتن نام قطعه" اسم قطعات مورد نیاز را تک تک بنویسید و سپس بر روی آنها دو بار کلیک کنید ، قطعات مورد نیاز:

1- میکرو کنترلر mega16 (atmega16)

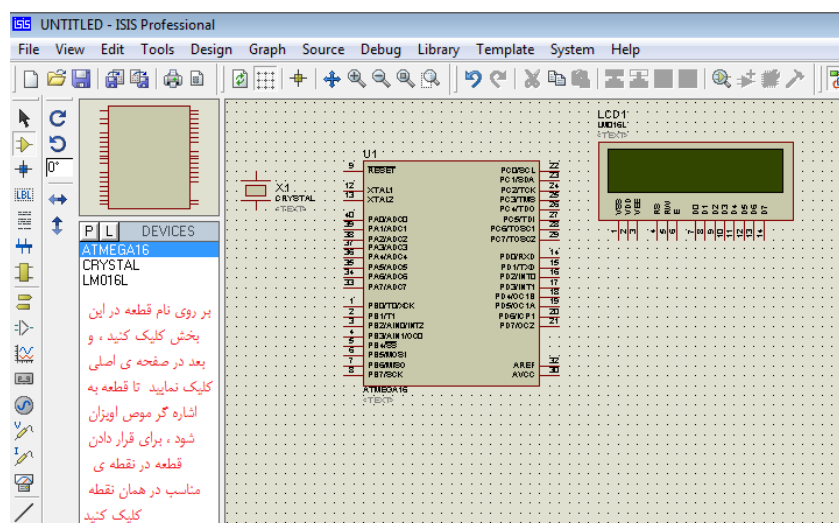
2- Lcd 2*16 (LM016L)

3- کریستال 8 مگاهرتز (CRYSTAL)

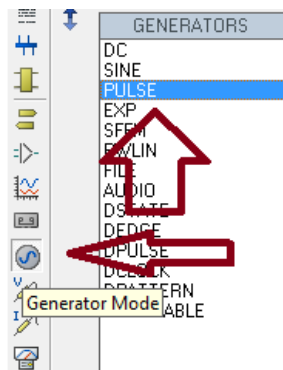
4- منبع پالس

برای نمونه نام میکرو را در قسمت گفته شده تایپ کنید ، نرم افزار قطعه را به شما معرفی میکند (مانند شکل بالا) روی نام آن دوبار کلیک کنید تا نام آن به قطعات انتخاب شده افزوده شود . این کار را برای دیگر قطعات نیز انجام دهید و در آخر روی OK کلیک کنید تا کتابخانه بسته شود.

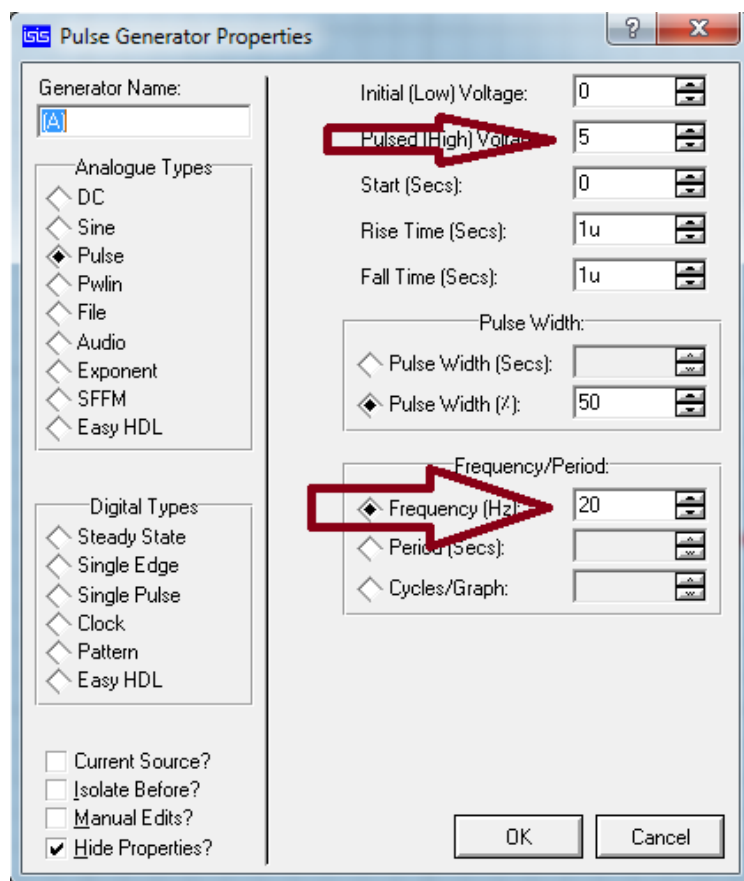
برای انتقال قطعات روی صفحه اصلی اول روی نام آنها در پالت قطعات کلیک کنید و سپس در نقطه دلخواه از صفحه دوباره کلیک کنید تا قطعه در آنجا گذاشته شود ، همه قطعات را به صفحه انتقال دهید :



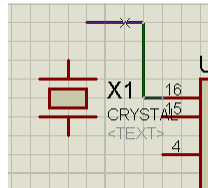
برای آوردن منبع پالس در بخش ابزار های کاربردی بر روی آیکن generator mode کلیک نموده و از پالت باز شده گزینه ی pulse را انتخاب نمایید و آن را در صفحه ی اصلی قرار دهید :



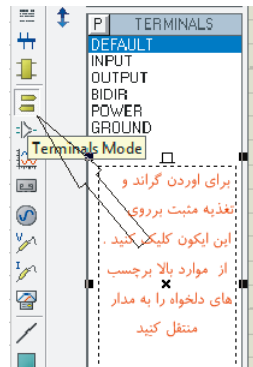
سپس بر روی آن دوبار کلیک کنید و داده ی موجود در تصویر زیر را در پنجره ی باز شده وارد نمایید :



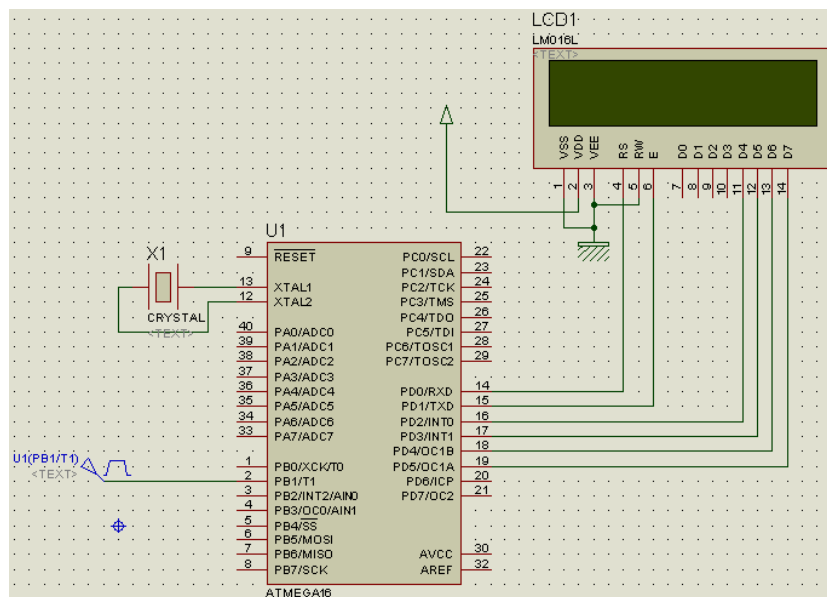
برای کشیدن سیم های بین پایه ها ابتدا روی آیکن Selection mode (موس مشکی) کلیک کنید سپس موس را بر روی پایه ای که می خواهید آن را به پایه دیگر متصل کنید ببرید می بیند که آیکن موس شبیه به یک مداد میشود ، روی پایه کلیک کنید و آن را به پایه دیگر متصل کنید ، خط را کشیده و بر روی پایه دیگر دوباره کلیک کنید ، برای حذف کردن یک قطعه دوبار روی آن کلیک چپ نمایید.



برای آوردن VCC و گراند، روی ایکون terminals mode کلیک کنید و در اینجا گزینه ای POWER و GROUND را انتخاب کنید.

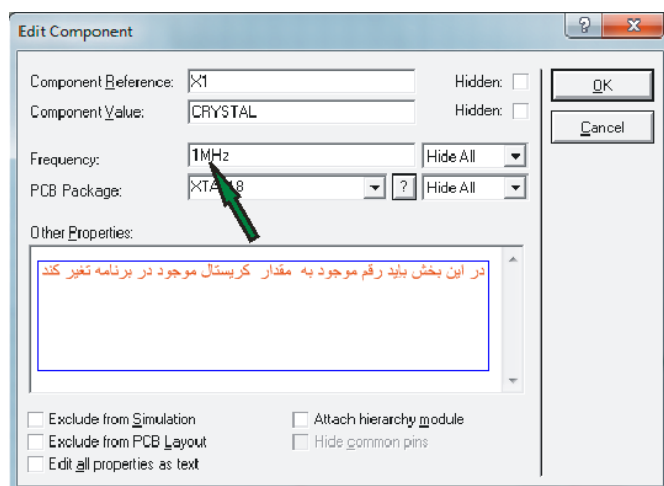


مدار شما باید طبق برنامه به شکل زیر باشد :



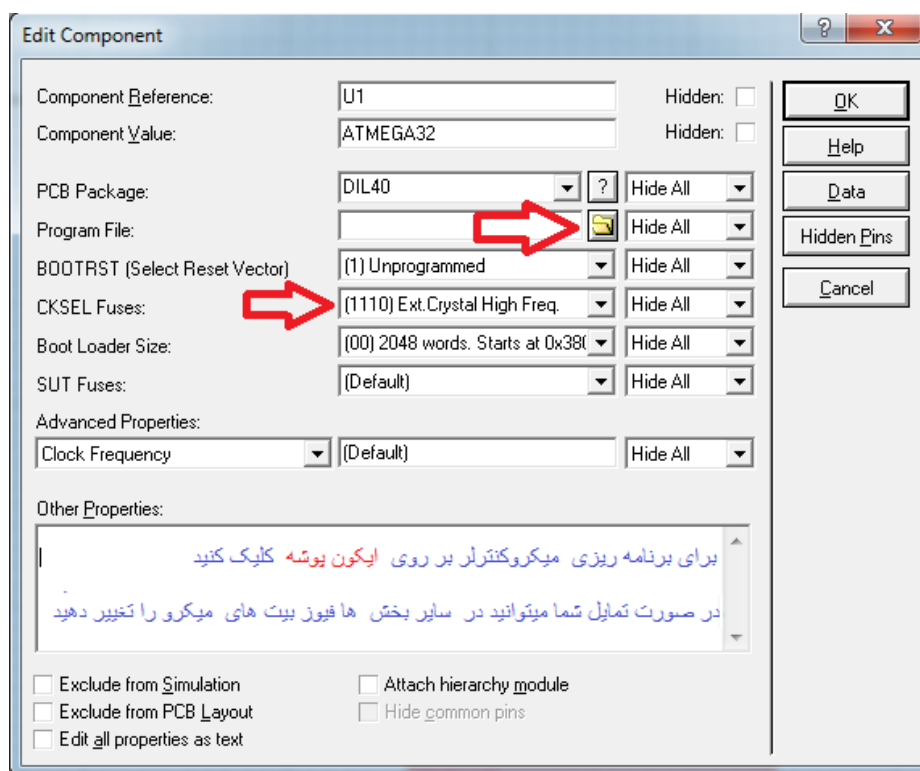
بعد از طراحی مدار، مشاهده میکنید که کلیه قطعات دارای مقادیر پیش فرض میباشند، مثلاً مقدار کریستال 1 مگا هرتز است و.....، ما باید همه آنها را به مقادیر دلخواه خودمان تغییر دهیم، برای این کار روی قطعه (در صفحه اصلی) دو بار کلیک کنید، پنجره ای باز میشود که در آن مشخصات قطعه موجود میباشد.

من بر روی کریستال دو بار کلیک کردم و پنجره زیر باز شد، در این پنجره باید فرکانس کریستال را به مقدار نوشته شده در برنامه تغییر داد، برای این کار فرکانس جدید را در قسمت نمایش داده شده تایپ کنید:



بعد از رسم کامل مدار و تغییر دادن مقدار قطعات ، نوبت به برنامه ریزی میکرو میرسد .

برای ریختن کد هگز روی میکرو ، روی آن دوبار کلیک چپ کنید ، در پنجره باز شده بر روی آیکن مشخص شده کلیک نمایید و در پنجره ی جدید به محل ذخیره ی کد هگز رفته و آن را باز کنید ، همچنین فرکانس کریستال را به 8 مگا هرتز تغییر دهید:



کد هگز با پسوند hex. بعد از کامپایل کردن برنامه ی فوق ایجاد میشود ، شما باید برنامه ای که در بالا موجود است را در محیط بسکام کپی کرده و آن را کامپایل کنید ، با کلیک کردن بر روی کلید شروع شبیه سازی ، برنامه اجرا شده و.....

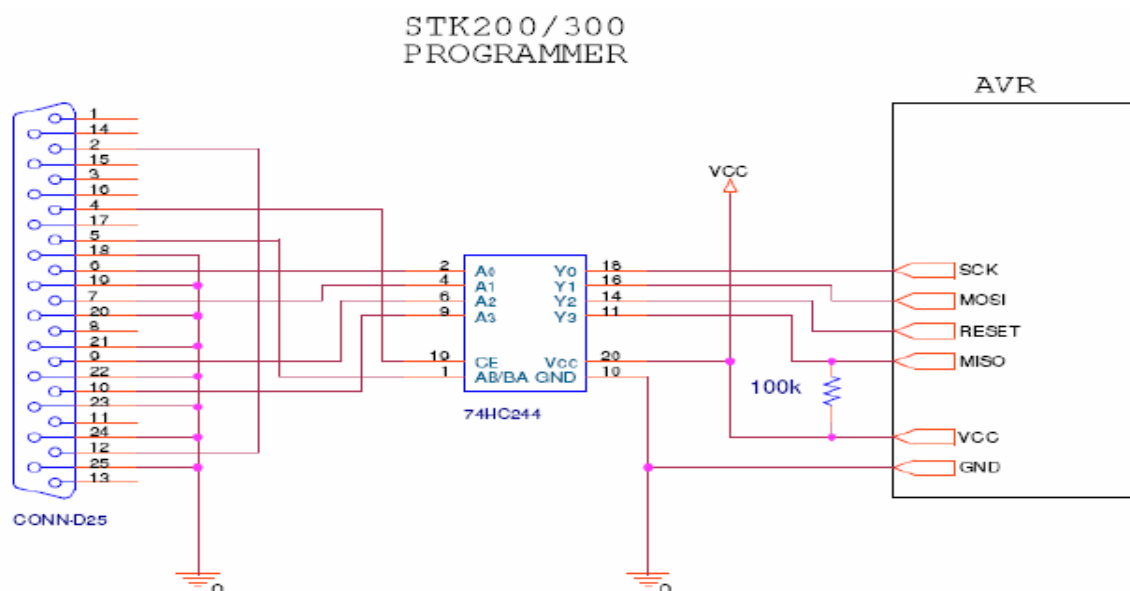
ضمیمه 4 برنامه ریزی میکرو کنترلر و معرفی پروگرامر ها

بعد از نوشتن برنامه و تست آن نوبت به برنامه ریزی میکرو کنترلر میرسد ، برای این کار به دستگاهی به نام پروگرامر نیاز دارید تا بین کامپیوتر و میکرو کنترلر قرار گرفته و کد هگز موجود را از کامپیوتر به میکرو انتقال دهد. پروگرامر ها در نمونه ها و قیمت های متنوعی ساخته و عرضه شده اند ، شما میتوانید نمونه مناسب با پورت کامپیوتر (پروگرامرها به یکی از پورت های کامپیوتر متصل میشوند ، این پورت ممکن است پورت موازی (lpt یا چاپگر) ، پورت سریال (com) یا پورت usb باشد) خود را انتخاب کرده و از آن استفاده کنید . در ادامه به بررسی دو پروگرامر معروف stk200/300 و mk II پرداخته ایم . کلیه پروگرامر های مذکور دارای مدار و سورس رایگان بوده و امکان ساخت آن برای شما وجود دارد .

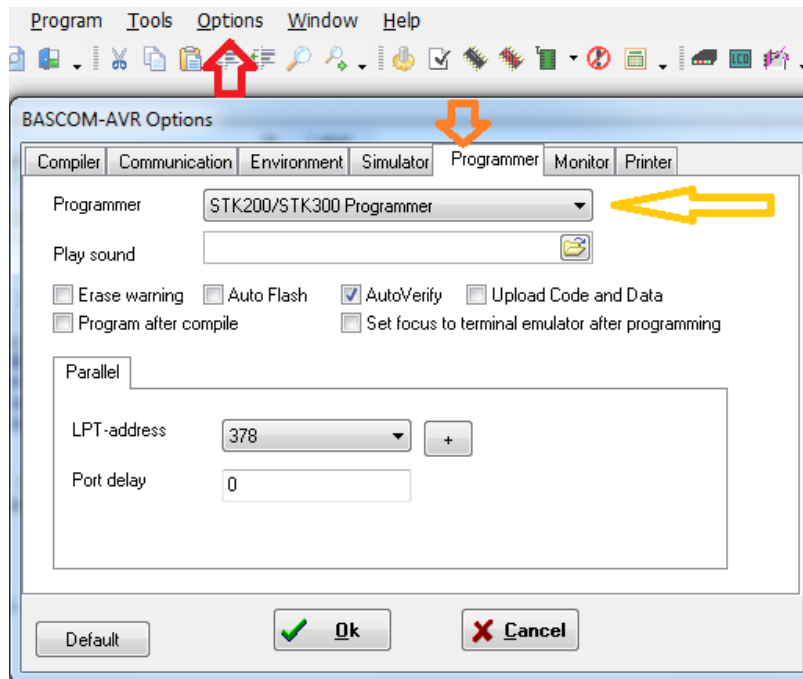
قبل از خواندن ادامه مطلب ضمیمه ی طراحی مدار با میکرو کنترلر های AVR را بخوانید .

پروگرامر STK200/300 :

یکی از پروگرامر های بسیار ساده برای میکرو کنترلر های AVR پروگرامر stk200/300 میباشد ، این پروگرامر دارای سخت افزاری به شکل زیر است و از طریق پورت lpt با میکرو کنترلر تبادل داده میکند :



با استفاده از این پروگرامر می‌توانید برنامه‌ی خود را از داخل کامپایلر بسکام به میکرو کنترلر منتقل نمایید، برای راه اندازی این پروگرامر به مسیر Options>Programmer و تنظیمات پنجره‌ی موجود را مانند عکس زیر تغییر دهید:



همچنین شما باید به کنترل پانل ویندوز بروید و در آنجا یک پرینتر نصب کنید، برای این کار به مسیر زیر بروید:

Start menu>control panel>Printers and Faxes(or Control Panel\All Control Panel Items\Devices and Printers)

در پنجره باز شده بر روی گزینه add a printer کلیک کنید:

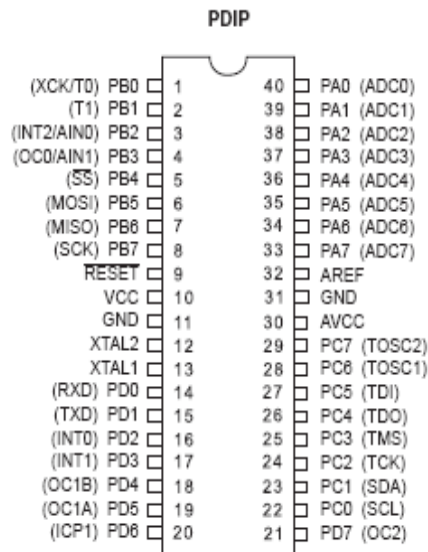


پنجره‌ی باز شده، در همه پنجره‌ها فقط گزینه next را بزنید، و در نهایت روی finish کلیک کنید.

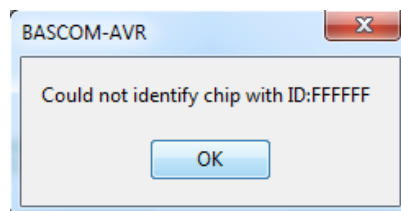
تنظیمات نرم افزاری پروگرامر انجام شد، اکنون شما باید سخت افزار بالا را ساخته و آن را به پورت پرینتر سیستم خود متصل نمایید، فایل شماتیک و pcb کامل این مدار را از انجمن سایت IRANMICRO.IR دانلود کنید.

اکنون شما باید پایه‌های sck و mosi و miso و reset و vcc و gnd پروگرامر را به پایه‌های متناظر میکرو کنترلر خود متصل

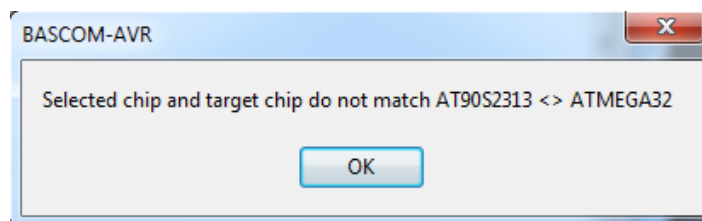
نمایید، در زیر تصویر میکرو کنترلر ATMEGA16 را مشاهده می‌کنید:



برای دستیابی به نرم افزار پروگرامر، در محیط بسکام به آدرس `program>send to chip` بروید یا کلید `f4` را فشار دهید، در صورتی که تنظیمات سخت افزاری و نرم افزاری به درستی انجام شده باشد، پنجره ی `AVR ISP STK programmer` باز میشود. در صورت وجود هر گونه اشکال در تنظیمات یا سخت افزار با یکی از پیغام های زیر روبرو خواهید شد:



این پیغام، هنگامی که سخت افزار پروگرامر به درستی به کامپیوتر متصل نشده باشد نمایش داده میشود (ممکن است میکرو به پروگرامر، یا پروگرامر به کامپیوتر متصل نشده باشد).



این پیغام هنگامی که میکرو نوشته شده در برنامه با میکرو ی متصل شده به پروگرامر یکی نباشد نمایش داده میشود. برای رفع کردن خطاهای فوق شما باید موارد زیر را چک کنید:

✓ حداکثر طول کابل پروگرامر (از pc به برد پروگرامر و از پروگرامر به میکرو) از 50 سانتی متر بیشتر نباشد.

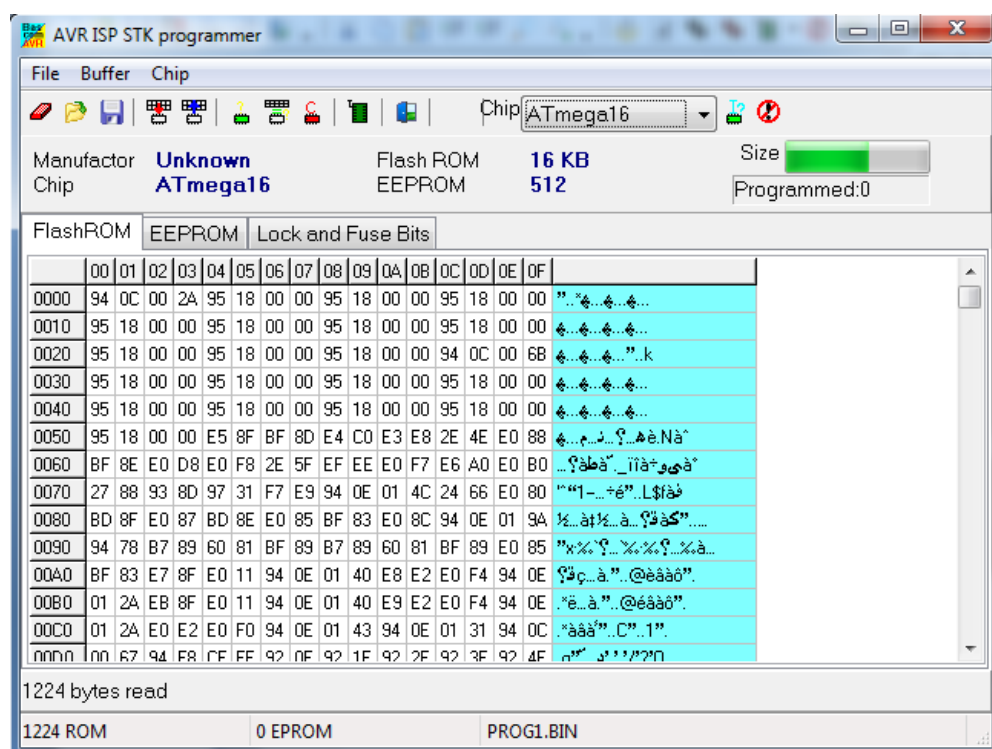
✓ برای راه اندازی مدار پروگرامر و میکرو به تغذیه ی 5 ولت نیاز دارید که میتوانید آن را از پورت usb کامپیوتر تامین کنید .

✓ در صورتی که کانکتور مادگی پورت lpt را مقابل خود بگیرید شماره ی پایه ها در کنار آنها نوشته شده است .

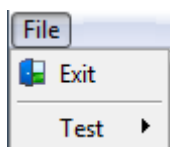
✓ این پروگرامر فقط از میکروکنترلر های که پورت spi دارند پشتیبانی میکند .

در زیر تصویر پنجره ی AVR ISP STK programmer را مشاهده میکنید ، در ادامه به بررسی گزینه های موجود در این

پنجره پرداخته ایم :

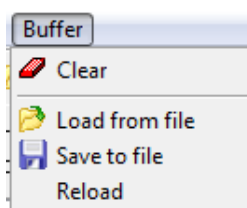


منوی FILE



EXIT: با زدن این گزینه پنجره پروگرامر بسته میشود .

TEST: با زدن این گزینه پایه های پورت lpt 1 میشود و میتوان پورت را امتحان کرد.



منوی BUFFER

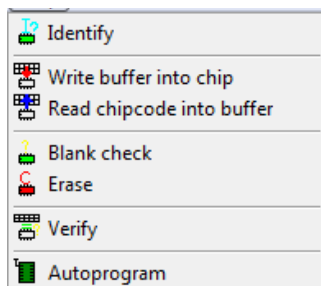
BUFFER CLEAR: با زدن این گزینه اطلاعات موجود بر روی حافظه نرم افزار پروگرامر پاک

میشود.

LOAD FROM FILE: با زدن این گزینه شما میتوانید یک فایل هگز یا باینری را باز کنید و آن را بر روی میکرو پروگرام کنید.

SAVE TO FILE: شما میتوانید محتوای حافظه نرم افزار را در مکان دلخواه با زدن این گزینه ذخیره کنید.

Reload: با انتخاب این گزینه کد هگز برنامه ی موجود در بسکام برای انتقال به میکرو باز میشود.



منوی CHIP

CHIP IDENTIFY: با زدن این گزینه میکرو متصل به پروگرامر شناسایی میشود.

WRITE BUFFER TO CHIP: با زدن این گزینه محتوای حافظه نرم افزار (کد هگز) که باز

کردید یا بوجود آمده بود، در حافظه میکرو ریخته میشود.

READ CLIPCODE INTO BUFFER: با زدن این گزینه کد هگز موجود در حافظه میکرو خوانده میشود و در حافظه نرم افزار

قرار میگیرد، شما میتوانید با زدن گزینه SAVE TO FILE آن را در مکان دلخواه ذخیره کرده و بعداً آن را روی میکرو دیگر

بریزید.

BLACK CHECK: با زدن این گزینه پر یا خالی بودن حافظه میکرو مشخص میشود.

ERASE: با زدن این گزینه حافظه میکرو پاک میشود.

VERIFY: این گزینه کد هگز موجود را با کد ریخته شده در حافظه میکرو مقایسه میکند، در صورتی که کدها با هم تفاوت

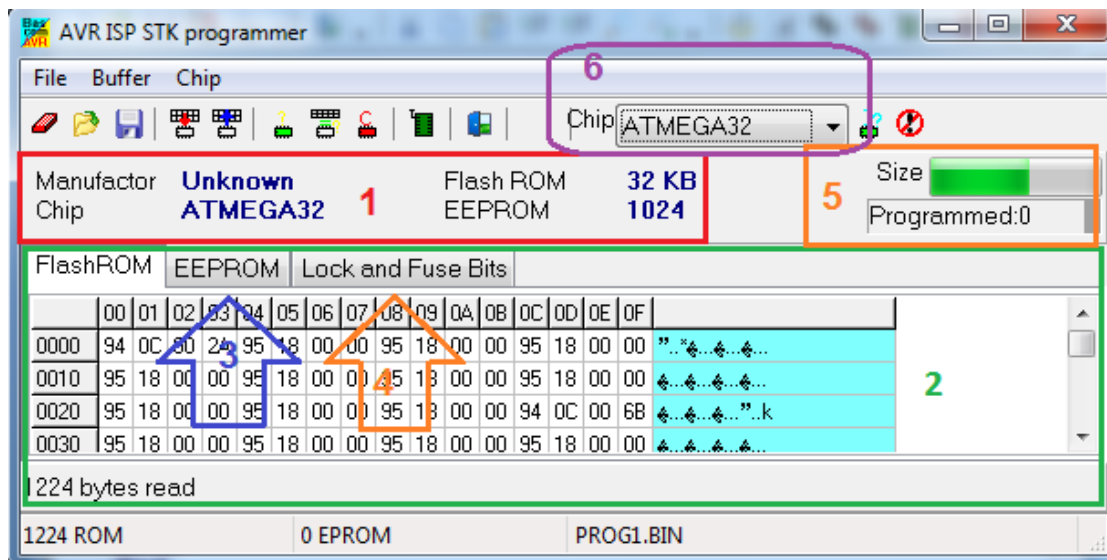
داشته باشند پیغامی ظاهر میشود.


AUTO PROGRAM: با زدن این گزینه حافظه میکرو پاک شده و کد هگز در آن ریخته میشود و بعد عمل VERIFY را به

صورت خود کار انجام می دهد.

RESET: میکرو متصل به PROGRAMMER را ریست می کند

دیگر منوها:



- 1- اطلاعات مربوط به میکرو، شامل نام و مقادیر حافظه ی فلش و eeprom را نشان میدهد.
 - 2- کد های موجود در خانه های حافظه ی فلش / ایپرام / فیوز بیت ها در این بخش نمایش داده میشود.
 - 3- با زدن این گزینه محتویات حافظه eeprom نمایش داده میشود.
 - 4- با زدن این گزینه وارد محیط پروگرام کردن فیوز بیت ها میشوید.
 - 5- تعداد دفعاتی که با این پروگرامر میکرو را پروگرام کرده اید و سائز واقعی کد هگز موجود در این بخش نمایش داده می شود. توجه داشته باشید که سائزی که ویندوز با انتخاب گزینه ی Properties به شما نشان میدهد حجم واقعی نیست.
 - 6- نام میکرو را نشان میدهد، هنگامی که این پنجره را باز میکنید، در صورت درست بودن سخت افزار، نام میکرو به صورت خودکار نمایش داده میشود.
- سایر گزینه های موجود در این بخش، در منوها مورد بررسی قرار گرفت، برای برنامه ریزی سریع میکرو، کافی است به این محیط وارد شده و گزینه ی AUTO PROGRAM را از منوی CHIP انتخاب نمایید (یا بر روی آیکون  کلیک کنید) تا برنامه به حافظه ی فلش میکرو منتقل شده و بعد از مقایسه و تایید صحت، پیغام تایید برنامه ریزی به شما نمایش داده شود.

در این پنجره شما میتوانید فیوز بیت های میکرو کنترلر را نیز برنامه ریزی نمایید، برای اینکار کافی است به دیتاشیت میکرو مراجعه کرده و فیوز بیت دلخواه خود را انتخاب نمایید. در ادامه این مورد را بیشتر توضیح داده ایم ...

آموزش کار با پروگرامر MK II :

این راهنما به شما می آموزد که چگونه پروگرامر MK II را راه اندازی کرده و با آن سری XMEGA را برنامه ریزی کنید. برای دریافت راهنمایی کامل این پروگرامر به آدرس زیر مراجعه کنید :

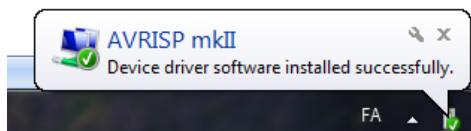
<http://www.kavirelectronic.ir/eshop/?>

MK II (بخوانید MK TWO) یکی از پروگرامر های تجاری تولید شده توسط شرکت اتمل است که میتواند از طریق واسط های PDI و ISP میکرو کنترلر های 8 و 16 بیتی تولید شده توسط این شرکت را برنامه ریزی کند. قیمت نسخه ی اصلی این پروگرامر در حدود 34 دلار میباشد که در ایران با قیمت 60000 (با دلار 1400 تومانی) در دسترس شما است.

در اولین ماه های سال 2010 تیم لیفا، نسخه ی کلون این پروگرامر را در دسترس همگان قرار داد، با ارائه ی این نسخه، کلیه مدارات و سورس های موجود توسط گروه کویر الکترونیک تهیه و تست گردید و از طریق مجلات PMM در دسترس کاربران قرار گرفت. شما میتوانید کلیه مستندات فارسی این پروگرامر + شماتیک 100٪ عملی آن را در مجله ی PMM شماره ی 14 مشاهده کنید و مدار پروگرامر را خودتان بسازید.

همچنین شما میتوانید این پروگرامر را از طریق فروشگاه تخصصی گروه کویر الکترونیک با بسیار مناسب تهیه نمایید. برای کار با این پروگرامر به نرم افزار AVR Studio 4.18 SP2 و درایو پروگرامر نیاز دارید، که میتوانید آنها را از انجمن سایت ایران میکرو دانلود کنید.

بعد از نصب نرم افزار (نصب نرم افزار مانند سایر نرم افزار های ویندوز بوده و نکته ی خاصی ندارد) پروگرامر را به پورت USB کامپیوتر خود متصل نمایید و جامپر ON/OFF موجود در کنار پورت USB دستگاه را متصل نمایید. با اتصال پروگرامر ویندوز به صورت خودکار پروگرامر را شناخته و درایو آن را نصب میکند :



آشنایی با بخش های مختلف پروگرامر:



در صورتی که مطالب موجود در مجله ی PMM14 را مطالعه کنید ، متوجه خواهید شد که شماتیک ارائه شده دقیقا با نمونه ی ارائه شده در فروشگاه یکسان میباشد ، البته ما یک کانکتور IDC برای سازگاری پروگرامر با برد های آموزشی EKE2-XMEGA و EKE2-AVR/8051 اضافه کرده ایم که شما میتوانید در صورت نیاز آن را حذف کنید .

✓ در این مدار از یک میکرو کنترلر AT91USB162 و یک بافر به شماره ی GTL2003 استفاده شده است . میکرو کنترلر وظیفه ی دریافت داده از پورت USB و انتقال آن به پورت PDI یا ISP را به عهده دارد ، قطعه ی GTL2003 یک بافر دوطرف میباشد که میتواند از دو طرف مدار در برابر اتصال کوتاه یا اضافه ولتاژ محافظت نماید .

✓ قطعه ی ZIF40PIN1 یک زیف سوکت 40 پایه میباشد که میکرو کنترلر های سری MEGA ، AT90S و TINY روی آن قرار میگیرند ، همچنین شما میتوانید با استفاده از کانکتور ISP و یک کابل IDC میکرو کنترلر را در داخل برد برنامه ریزی نمایید . این کانکتور با برد های EKE2-XMEGA و EKE2-AVR/8051 گروه کویرالکترونیک سازگاری دارد و کافی است آن را با کابل IDC به برد متصل نمایید .

✓ برای برنامه ریزی سری XMEGA از سوکت PDI استفاده میشود .

✓ LED های موجود وضعیت برنامه ریزی و انتقال داده را مشخص میکنند .

✓ نقش جامپر ON/OFF خاموش/روشن کردن پروگرامر میباشد، همچنین با اتصال جامپر POWER MODE میتوانيد تغذيه ی میکرو کنترلر را از پروگرامر تامین نمایید.

✓ به دلیل وجود تداخل در قرار گرفتن برخی از میکرو کنترلر بر روی یک زيف سوکت، استفاده از جامپر SCK الزامی است، این جامپر باید در هنگام برنامه ریزی میکرو کنترلر های 28 پایه قطع باشد.

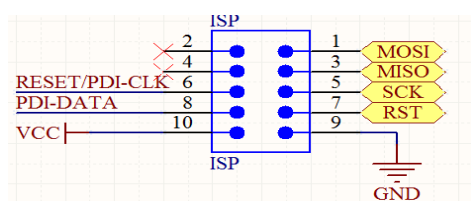
✓ کریستال Y2 برای تامین کردن کلاک میکرو کنترلر های که منبع کلاک آنها بر روی کریستال خارجی تنظیم شده است به کار میرود.

✓ کلید های RESET و HWB برای برنامه ریزی میکرو کنترلر At91usb162 به کار میرود، همچنین با استفاده از جامپر SELF PROGRAM میتوانيد برنامه ی دلخواه خود را بر روی میکرو کنترلر برنامه ریزی نمایید، شما فقط در هنگام آپدیت کردن برنامه ی میکرو کنترلر به این موارد نیاز خواهید داشت.

✓ پین CLK برای تامین کلاک سایر اداوات جانبی تعبیه شده است.

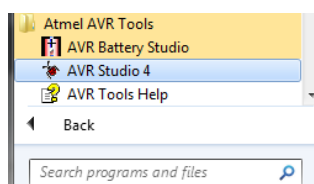
✓ نحوه ی قرار دادن میکرو بر روی سوکت زيف در راهنمای کامل پروگرامر آورده شده است. میکرو کنترلر Atmega16 را به گونه ای در داخل پروگرامر قرار دهید که پایه ی یک آن به سمت پایه ی سوکت zif (سوکت سبز رنگ) باشد. پایه ی یک میکرو کنترلر با یک فرو رفتگی مشخص شده است.

در زیر وضعیت پایه های سوکت ISP را مشاهده میکنید:

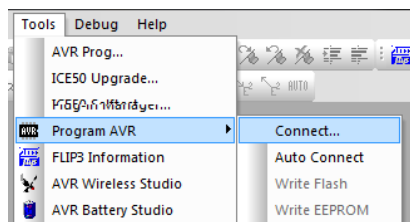


موقعیت پایه ها را بر روی تصویر صفحه ی قبل مشاهده کنید، پایه ی شماره ی 1 و 10 با فلش مشخص شده است.

بعد از راه اندازی نرم افزار و سخت افزار و اتصال میکرو کنترلر به پروگرامر، نرم افزار AVR Studio را باز کنید:

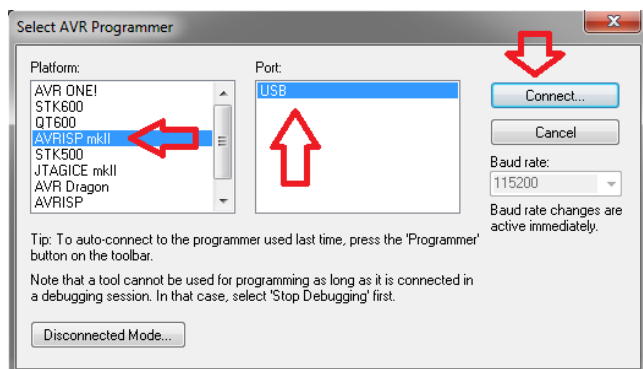


در محیط نرم افزار از منوی TOOLS و زیر منوی AVR program گزینه ی Connect را انتخاب کنید:

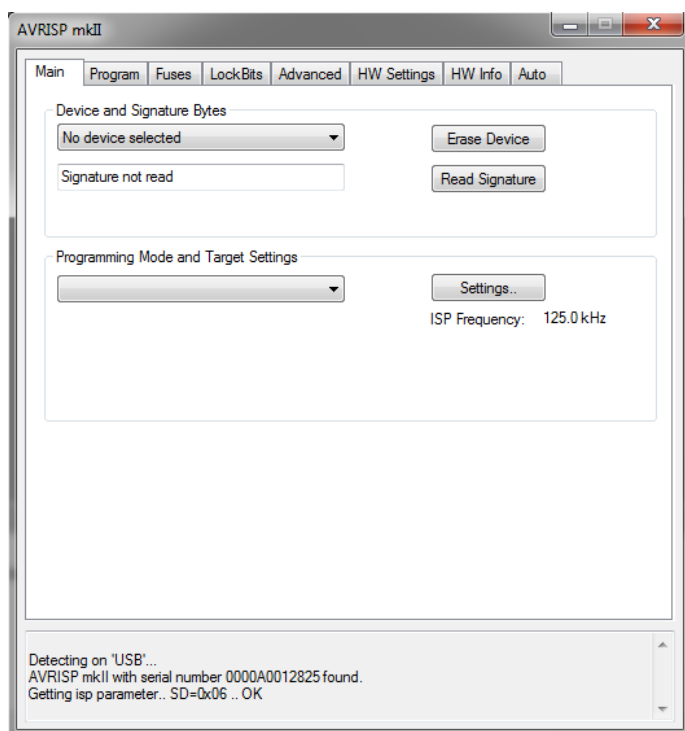


در پنجره ی باز شده از بخش Platform گزینه ی AVRISP mkII و از بخش پورت گزینه ی USB را انتخاب نمایید و

سپس بر روی Connect کلیک کنید :



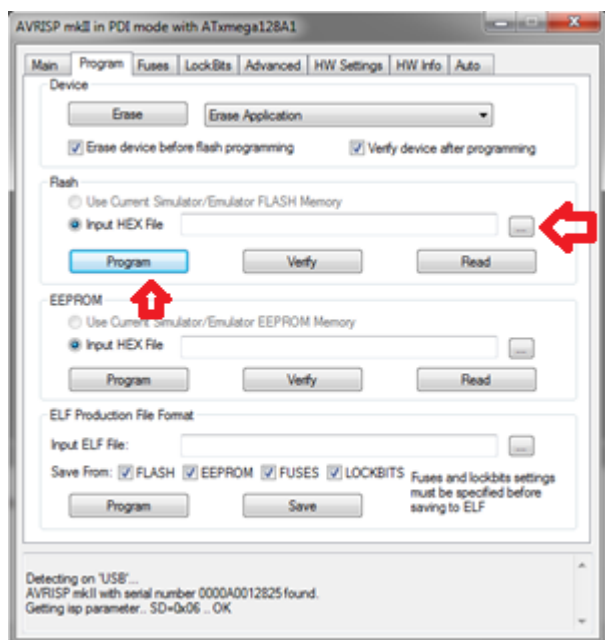
باز شدن پنجره ی AVRISP mkII به مفهوم درست بودن تنظیمات و شناخته شدن پروگرامر توسط نرم افزار است :



در پنجره ی بالا از بخش device and signature bytes میکرو کنترلر نصب شده بر روی برد (مثلا میکرو کنترلر

Atxmega128a1 یا atmega16 یا هر میکرو کنترلر دیگری که قصد برنامه ریزی آن را دارید) را انتخاب نمایید.

به پالت program بروید، در این بخش میتوانید توسط گزینه های موجود کد هگز ساخته شده توسط کامپایلر های دیگر را باز کرده و به میکرو منتقل کنید.



در بخش path بر روی گزینه ی browse کلیک نمایید و فایل hex یا bin ایجاد شده توسط کامپایلر بسکام که در محل ذخیره ی برنامه ایجاد شده است را باز کنید (این کد در کنار برنامه ی اصلی و بعد از کامپایل کرن برنامه ایجاد میشود) و سپس بر روی کلید program کلیک کنید تا برنامه به میکرو منتقل شود.

با انتخاب تب های fuse و lock bit میتوانید فیوز بیت های و قفل های حافظه ی میکرو کنترلر را برنامه ریزی نمایید.

فیوز بیت های اصلی میکرو کنترلر

فیوز بیت ها فقط در سری های atmega و at9s و tiny وجود دارند و تعداد و نحوه ی عملکرد آنها برای میکرو کنترلر ها خانواده های مذکور متغیر است.

در اغلب میکرو کنترلر های AVR دو فیوز بیت jtag و فیوز بیت کلاک وجود دارد که کاربر باید با توجه به برنامه ی خود باید آنها را تغییر دهد، سایر فیوز بیت ها باید بعد از مطالعه ی دیتاشیت تنظیم می شوند (معمولاً در اکثر مدارات این فیوز بیت ها در حالت پیش فرض استفاده میشود)، در ضمیمه "دیتاشیت فارسی میکرو کنترلر های AVR" به بررسی فیوز بیت های برخی از میکرو کنترلر های پر استفاده پرداخته شده است.

در میکروکنترلر های که دارای پورت jtag هستند ، به صورت پیش فرض ورودی / خروجی های پورت C به عنوان پورت های داده این پروتکل پیکربندی شده اند که کاربر برای استفاده از این پورت به عنوان ورودی خروجی باید فیوز بیت JTAG را غیر فعال کند .

JTAG یک پروتکل استاندارد برای برنامه ریزی و دیباگ کردن برنامه ی میکروکنترلر است .

میکروکنترلر های AVR به صورت پیش فرض با فرکانس یک مگا هرتز کار میکنند . به عنوان مثال اگر میکروکنترلی با برنامه زیر پروگرام شود :

```
$regfile = "m16def.dat"
```

```
$crystal = 8000000
```

```
Config PORTA = Output
```

```
A:
```

```
Set PORTA.1
```

```
Wait 1
```

```
RESet PORTA.1
```

```
Wait 1
```

```
Goto A
```

```
End
```

و فیوز بیت های آن برنامه ریزی نشوند ، علی رقم استفاده از تاخیر 1 ثانیه ، پایه ی 1 از پورت A هر 8 ثانیه تغییر وضعیت میدهد .

با تغییر دادن فیوز بیت CKSEL0 تا 3 به مقدار مناسب میتوان فرکانس کاری سخت افزار را با فرکانس تعیین شده در نرم افزار برابر کرد .

در دیتاشیت فارسی میکروکنترلر های AVR اطلاعات بیشتری در رابطه با فیوز بیت ها وجود دارد .

ضمیمه 5: طراحی مدار با میکروکنترلر های AVR :

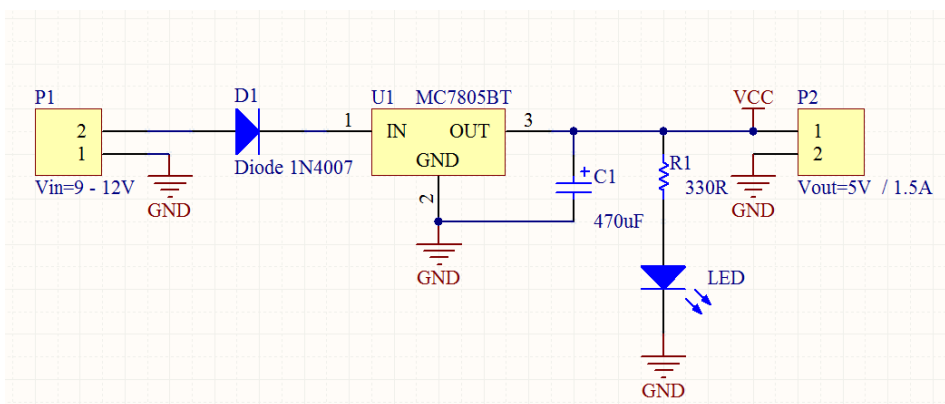
حداقل سخت افزار مورد نیاز برای راه اندازی میکروکنترلر های سری tiny و at90s و atmega :

در ضmann بعدی دیتاشیت فارسی برخی از میکروکنترلر های AVR آورده شده است با مرور کلی یکی از آنها مشاهده خواهید کرد که در تمامی آنها پایه های به نام RESET و VCC و GND و XTAL و AVCC و AGND و AREF وجود دارد .

ولتاژ تغذیه :

در میکروکنترلر های AVR پایه ی VCC و GND باید به ولتاژ 5 ولت متصل شوند . بسته به نوع میکروکنترلر ولتاژ موجود میان این دو پایه میتواند بین 1.8 – 5.5V و 2.7 – 5.5V و 4.0 – 5.5V باشد .

- ✓ در صورتی که ولتاژ تغذیه از 5.5 ولت بیشتر شود ، میکروکنترلر آسیب خواهد دید .
 - ✓ معکوس شدن پلاریته ی ولتاژ اعمالی به پایه های VCC و گراند باعث میشود میکروکنترلر آسیب ببیند .
 - ✓ کاهش ولتاژ از کمترین حد تعیین شده منجر به خاموشی میکروکنترلر میشود .
 - ✓ کاهش و افزایش ناگهانی ولتاژ تغذیه منجر به ایجاد وقفه در عمل کرد میکروکنترلر میشود (هنگ کردن) (مثلاً اگر ولتاژ تغذیه به صورت ناگهانی از 5 ولت به 3.5 ولت کاهش یابد) .
- یکی از روش مرسوم برای اعمال کردن تغذیه به میکروکنترلر های AVR استفاده از مدار زیر است :



در این مدار از یک رگولاتور ولتاژ مثبت به شماره ی 7805 استفاده شده است ، رگولاتور به ازای ولتاژ اعمال شده به ورودی که میتواند بین 9 تا 12 ولت DC باشد ، ولتاژ ثابت 5 ولت را به خروجی تحویل میدهد . در این مدار برای نمایش وجود ولتاژ در خروجی رگولاتور از یک نمایشگر LED و یک مقاومت کاهنده استفاده شده است ، که شما میتوانید در صورت عدم نیاز به این قابلیت ، آنها را حذف کنید (R1 و LED) .

در صورتی که به تازگی کار خود را با میکروکنترلر های AVR شروع نموده اید ، میتوانید ولتاژ 5 ولت مورد نیاز را از خروجی ولتاژ پروگرامر تامین کنید ، در ادامه توضیحات بیشتری در این رابطه آورده شده است .

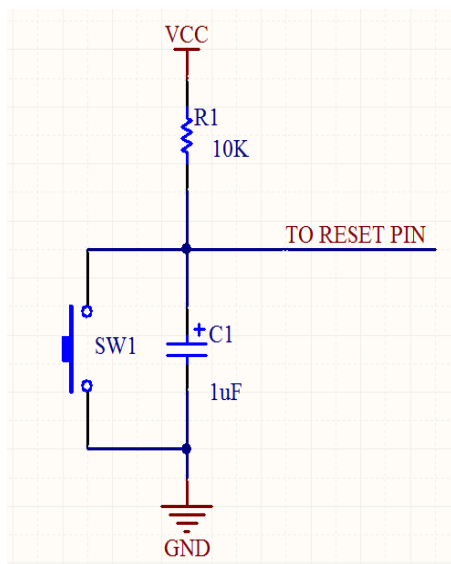
پایه های کریستال :

میکرو کنترلر های AVR دارای یک نوسان داخلی یک تا 8 مگاهرتز هستند ، این میکرو کنترلر ها همچنین می توانند کلاک خود را از پایه های XTAL1 و XTAL2 تامین کنند . در صورتی که به تازگی کار خود را با میکرو کنترلر های AVR شروع نموده اید ، نیازی به استفاده از کریستال خارجی نخواهید داشت و میتوانید میکرو کنترلر را با کریستال داخلی که نحوه ی فعال سازی آن در ضمیمه ی شماره ی 4 شرح داده شد ، فعال کنید . برای استفاده از کریستال خارجی باید منبع کلاک مورد نظر خود که یکی از گزینه های کریستال خارجی نوسان ساز سرامیکی یا نوسان ساز کریستالی فرکانس پایین یا نوسان ساز RC خارجی یا کلاک خارجی است را مطابق اطلاعات درج شده در دیتاشیت میکرو کنترلر به پایه های مربوطه متصل کنید ، با مراجعه به دیتاشیت فارسی میکرو کنترلر های AVR و بخش طریقه اتصال نوسان ساز به میکرو میتوانید اطلاعات بیشتری را در این زمینه بدست آورید .

توجه داشته باشید که برای تغییر دادن منبع کلاک میکرو کنترلر نیاز به ایجاد تغییر در فیوز بیت ها دارید ، تنظیم نادرست فیوز بیت ها ممکن است میکرو کنترلر را برای همیشه از کار بیندازد .

پایه RESET :

از این پایه برای بازنشانی میکرو کنترلر استفاده میشود . با صفر شدن این پایه محتوای تمامی متغیر ها و رجیستر ها صفر



شده و برنامه مجدداً از اولین خط اجرا میشود .

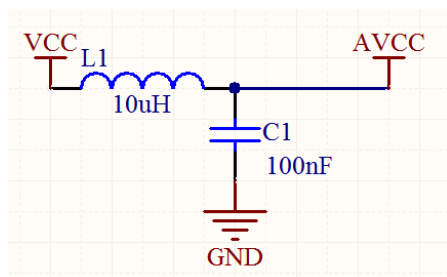
برای بازنشانی میکرو کنترلر به مدار روبرو نیاز خواهید داشت . با فشردن کلید پایه ی reset صفر میشود . در صورتی که به این قابلیت نیازی ندارید ، پایه ی RESET را توسط یک مقاومت 10 کیلو به پایه ی VCC (ولتاژ 5 ولت) متصل کنید .

قطع و وصل تغذیه ی میکرو یا برنامه ریزی مجدد آن نیز میتواند باعث RESET شدن میکرو کنترلر شود .

پایه های AVCC و AGND و AREF :

در صورتی که قصد داشته باشید از مبدل آنالوگ به دیجیتال داخلی

میکرو کنترلر استفاده کنید ، باید پایه ی AVCC را ولتاژ 5 ولت و پایه ی AGND



را به صفر ولت و پایه ی AREF را ولتاژ مرجع مشخص شده در برنامه متصل کنید. مدار مقابل یک فیلتر پایین گذر را نمایش می دهد که با استفاده از آن میتوان از ورود نویز های احتمالی ولتاژ تغذیه به ADC جلوگیری کرد. در صورتی که ولتاژ تغذیه بدون نویز باشد میتوانید پایه های AVCC و AREF را مستقیماً به VCC و پایه ی AGND (در صورت وجود) را مستقیماً به GND میکرو کنترلر متصل کنید.

پایه های مورد نیاز برای برنامه ریزی میکرو کنترلر :

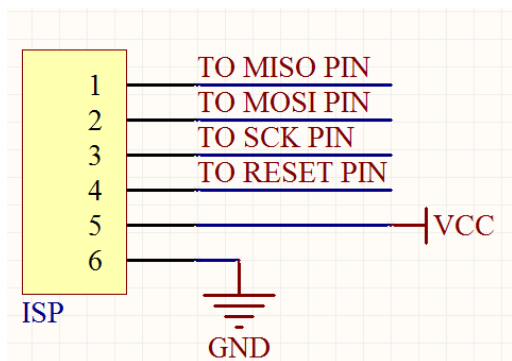
میکرو کنترلر های AVR معمولاً با دوروش JTAG یا ISP برنامه ریزی میشوند، که کاربر میتواند با استفاده از دستگاهی به نام پروگرامر برنامه ی خود، که توسط کامپایلر به کد هگز تبدیل شده است را به میکرو کنترلر منتقل کند. پروگرامر از یک سمت به یکی از پورت های کامپیوتر و از سمت دیگر به پورت JTAG یا ISP میکرو کنترلر متصل میشود.

برای JTAG از پایه های TCK و TMS و TDO و TDI و RESET + پایه های VCC و Gnd و برای ISP از پایه های MISO و MOSI و SCK و RESET + پایه های VCC و Gnd استفاده میشود.

واسط JTAG بر روی پورت C قرار دارد و در صورت فعال بودن آن دیگر نمیتوان از این پورت به عنوان ورودی و خروجی استفاده نمود، همچنین این واسط بر روی تعداد کمی از میکرو کنترلر وجود دارد (میکرو کنترلر های که بیشتر از 4 پورت دارند). این درگاه امکانات مختلفی همچون قابلیت اجرای خط به خط برنامه در داخل میکرو کنترلر، قابلیت اشکال یابی برنامه و... را در اختیار کاربران حرفه ای میدهد، در برگه ی اطلاعاتی موجود در فایل پیوست

به بررسی قابلیت های این پروگرامر و نحوه ی کار با آن پرداخته

شده است.



درگاه ISP روش ساده و ارزان قیمت برای برنامه ریزی میکرو کنترلر

های AVR است که برای استفاده از آن به یک کانکتور 6 پایه نیاز

خواهید داشت. درگاه ISP تقریباً در تمامی میکرو کنترلر های AVR

(به جز سری ATXMEGA) وجود دارد، به عنوان مثال در میکرو کنترلر ATMEGA16 یا ATMEGA32 پایه های 1 تا 6

کانکتور بالا به پایه های 6 تا 11 میکرو کنترلر متصل میشوند.

پایه های VCC و GND ولتاژ تغذیه ی میکرو کنترلر هستند که از طرف پروگرامر تامین خواهند شد. شماتیک یکی از پروگرامر های معروف AVR در ضمیمه ی شماره 4 آورده شده است.

اتصال پایه های ورودی / خروجی :

میکرو کنترلر های AVR معمولاً دارای چندین پورت 8 بیتی هستند که کاربر میتواند هر پین را در نقش اولیه و به عنوان ورودی یا خروجی یا در نقش ثانویه (مثلاً در یکی از پروتکل های ارتباطی) در برنامه ی نوشته شده تعیین کند. هر پین بعد از پیکربندی باید به یک سخت افزار جانبی متصل شود تا داده ای را از بیرون به میکرو کنترلر یا از میکرو کنترلر به دستگاه جانبی منتقل کند. در اتصال پایه ها به لوازم جانبی مختلف کاربر باید موارد زیر را در نظر بگیرد:

✓ حداکثر جریان مجاز خروجی پایه ها ورودی / خروجی برابر با 18 میلی آمپر است و کنترل کننده ی پورت میتواند این جریان را در سطح صفر یا یک تامین کند.

✓ حداکثر ولتاژ اعمالی به پایه ها برابر با 5.5 ولت و حداقل آن برابر با 0 ولت است.

✓ مقدار ولتاژ هر پایه در سطح یک برابر با VCC و در سطح صفر برابر با GND است.

✓ مقاومت ورودی پایه ها را میتوان بینهایت در نظر گرفت.

با توجه به موارد بالا بهتر است:

- برای راه اندازی امکانات جانبی همچون LED یا 7SEGMENT یا رله یا انواع موتور و... از درایو استفاده شود..
- کلیه پایه های که به عنوان ورودی معرفی میشوند با یک مقاومت 10 کیلو اهم به پایه های VCC یا GND متصل شوند.

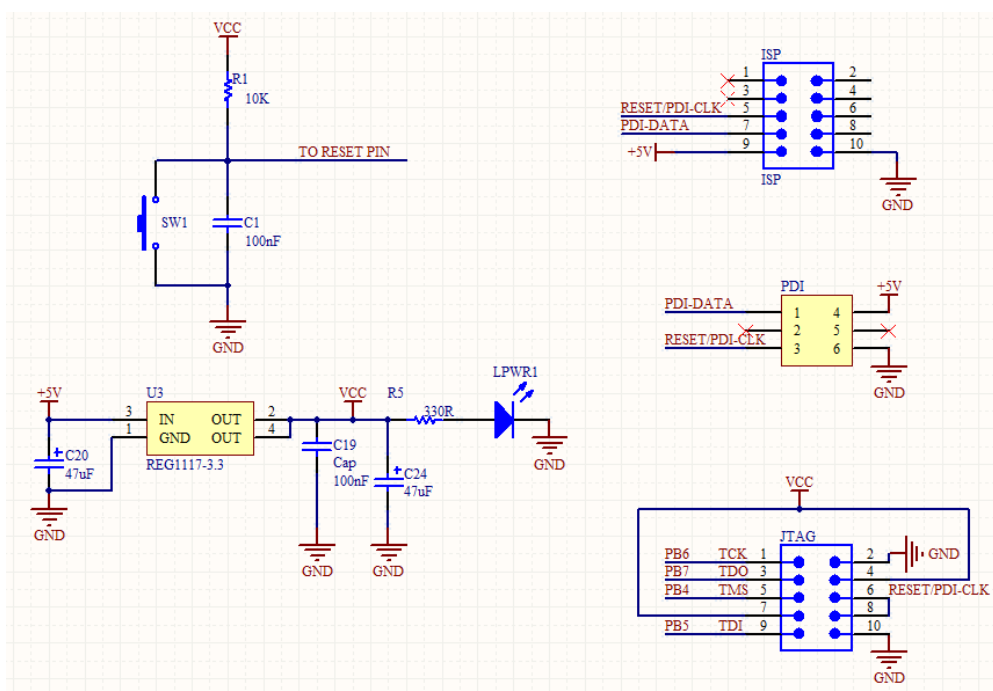
حداقل سخت افزار مورد نیاز برای راه اندازی میکرو کنترلر های سری ATxmega

➤ برای راه اندازی این میکرو کنترلر کافی است، مطابق نقشه ی صفحه ی بعد کلیه ی پایه های VCC را به ولتاژ 3.3 ولت و کلیه پایه های GND را به گرانند تغذیه متصل نمایید. شما میتوانید ولتاژ 3.3 ولت را توسط یک رگولاتور LF33 یا M1117.3.3 از ولتاژ 5 ولت USB یا ... تامین نمایید.

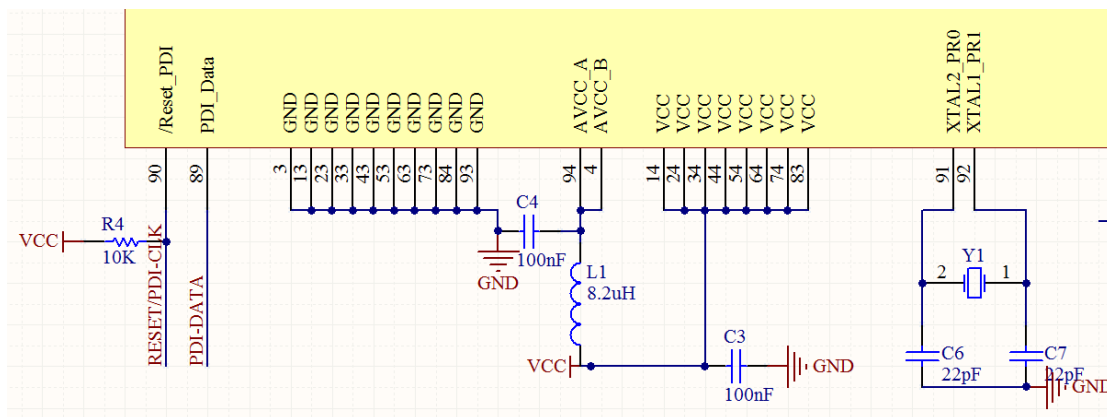
➤ پایه های AVCC جهت تامین کردن تغذیه ی آنالوگ مربوط به واحد ADC تعبیه شده اند، قطعا هر چقدر این ولتاژ پایدار تر باشد دقت تبدیل ADC بیشتر خواهد بود، شما میتوانید این دو پایه را با یک فیلتر میان گذر به پایه های VCC متصل نمایید

➤ سری XMEGA دارای یک اسیلاتور داخلی با فرکانس پیش فرض 2 مگاهرتز میباشد. شما می توانید در صورت نیاز به تامین کلاک میکرو از منبع خارجی، کریستالی را به پایه های PRO و PR1 (پایه های 91 و 92) یا به پایه های PQ0 و PQ1 (پایه های 85 و 86) متصل نمایید تا کلاک میکرو به ترتیب از کریستال خارجی با فرکانس 2 تا 16 مگاهرتز یا کریستال 32.687 کیلو هرتز (در مد های کم مصرفی) تامین شود.

➤ برای برنامه ریزی این دو خانواده دو روش JTAG و PDI در نظر گرفته شده است، در روش اول از خطوط TCK (JTAG Test Clock) و TDI (JTAG Test Data In) و TDO (JTAG Test Data Out) و TMS (JTAG Test Mode Select) به همراه RESET استفاده میشود. در این روش امکان دستیابی به مکان های مختلف حافظه و دیباگ و اشکال زدایی برنامه توسط پروگرامر و دیباگر JTAG وجود دارد. روش دوم از پین های PDI_CLK / RESET (Program and Debug Interface Clock pin) و PDI_DATA (Program and Debug Interface Data pin) برای برنامه ریزی میکرو توسط پروگرامر MK II استفاده میشود.



شماتیک زیر مطابق پایه های میکرو کنترلر ATXMEGA128A1 طراحی شده است و شما میتوانید پایه های متناظر برای میکرو کنترلر مورد نظر خود را در دیتاشیت آن مشاهده کنید.



در طراحی مدارات الکترونیک سه عامل مهم وجود دارد که تعیین کننده ی کیفیت کار و پایداری مدار میباشد:

1- قیمت نهایی پروژه

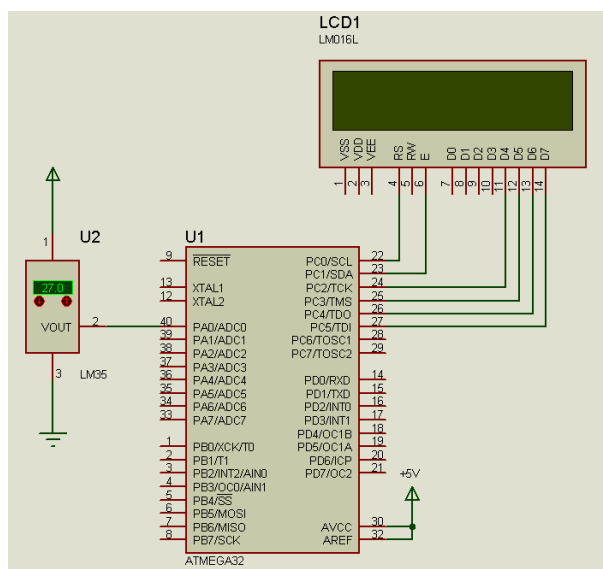
2- رعایت اصول طراحی الکترونیک

3- هدف طراحی مدار

رنج وسیع قطعات الکترونیک و تنوع قیمت آنها از جمله عواملی است که همواره طراحان را در انتخاب بهترین قطعه دچار گمراهی میکند ، به عنوان مثال ممکن است در یک پروژه اندازه گیری دما ، راهکار های زیر پیش روی طراح باشد :

✓ استفاده از یک سنسور دمای معمولی (مانند LM35 یا سنسور های PTC و NTC) و اتصال مستقیم آن به ADC

میکروکنترلر



قطعا در این روش در صورتی که سنسور در یک مکان نويز دار (مثلا در نزديك يك موتور القايي) نصب شود ، داده خروجی آن به هيچ وجه قابل اعتماد نخواهد بود ، همچنين نويز های الکترومغناطیسی ایجاد شده میتواند از طریق سیم های تغذیه ی سنسور به میکروکنترلر وارد شده و عملکرد صحیح آن را به خطر بیاندازند .

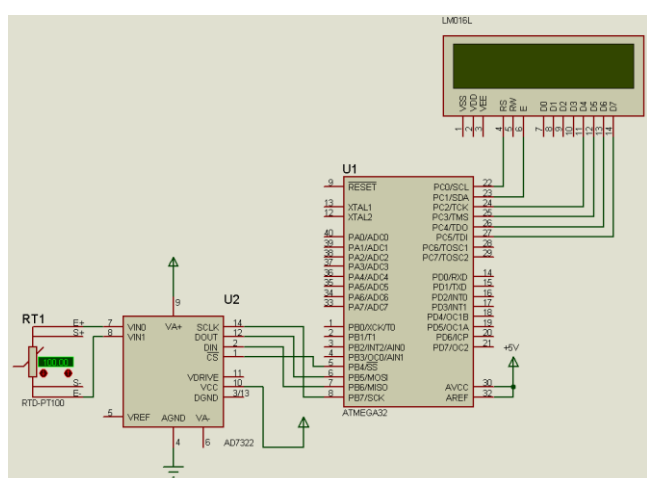
استفاده از کریستال داخلی برای میکروکنترلر ، آزاد بودن

پایه ی ریست میکرو کنترلر ، استفاده از منبع تغذیه ی نامناسب و... از دیگر عواملی هستند که عمل کرد صحیح مدار را تهدید میکنند .

این روش بیشتر برای پروژه های دانشجویی و مقاصد آموزشی مناسب است و به هیچ عنوان نمیتوان از آن برای مقاصد صنعتی استفاده نمود .

✓ استفاده از یک سنسور صنعتی (مانند PT100 یا مازول های اندازه گیری دما) و اتصال آن به یک ADC خارجی

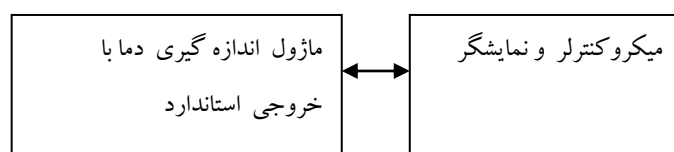
و اتصال خروجی ADC خارجی به میکرو کنترلر



در این روش هزینه طراحی 4 تا 6 برابر روش اول خواهد بود . در این روش به دلیل طراحی غیر اصولی و عدم آگاهی طراح ، عملا کلیه هزینه های پروژه به هدر رفته است و این مدار نیز دارای معایبی مانند مدار قبلی است .

✓ طراحی مازولی مجزا برای اندازه گیری دما و یک برد مجزا برای کنترل دما و اتصال مازول و برد مطابق

استاندارد های صنعتی (پروتکل RS485 یا ...)



در این روش هزینه ی طراحی 10 تا 15 برابر روش اول است و در آن به دلیل طراحی اصولی مدارات ، طراح میتواند عمل کرد پایدار آن را به صورت 100 درصد در کلیه شرایط تضمین کند ، در این روش ملاحظات زیر در طراحی لحاظ شده است :

ماژول اندازه گیری دما دارای یک سنسور صنعتی و یک مبدل آنالوگ به دیجیتال میباشد و قرار است در محل اندازه گیری دما نصب شود . در این مازول سنسور دما را اندازه گیری کرده و داده ی آنالوگ مربوط به آن را به مبدل

دیجیتال به آنالوگ میدهد. مبدل آنالوگ به دیجیتال داده آنالوگ را به دیجیتال تبدیل کرده و در نهایت این داده به صورت دیجیتال از طریق پروتکل RS485 یا به صورت آنالوگ (جریان 4 تا 20 میلی آمپر) به برد میکروکنترلر که در اتاق فرمان (یا محلی که دارای کمترین نویز است) ارسال می شود. در برد میکروکنترلر با استفاده از اپتوکوپلر داده ی دیجیتال به میکروکنترلر وارد میشود، در صورتی که داده ی دریافتی از سنسور آنالوگ باشد، جریان دریافتی ابتدا توسط یک ADC دیگر به دیجیتال تبدیل شده و بعد با استفاده از اپتوکوپلر به میکروکنترلر تحویل داده میشود. در این مدار میکروکنترلر به صورت مداوم خروجی ماژول را کنترل میکند و در صورت وجود خطا در داده ی خروجی آن میتواند آن را ریست کرده یا اپراتور را از وجود خطا آگاه کند.

در این مدار تغذیه یخس های آنالوگ و دیجیتال از یکدیگر مجزا هستند و تبادل داده میان آنها توسط ایزوله کننده ی نوری (اپتوکوپلر) انجام می شود. وجود الگوریتم های مختلف خطایابی برای داده ی دریافتی از ماژول در برنامه ی میکروکنترلر، استفاده از تایمر و اچداگ و فعال سازی فیوز بیت های مربوط به تغذیه ی میکروکنترلر، استفاده از قطعات uP Supervisory Circuits مانند MAX707 و طراحی بهینه و اصولی فیبر مدار چاپی و... از جمله عواملی هستند که پایداری مدار را افزایش میدهند.

در تمامی پروژه قیمت و کیفیت در اولویت قرار دارند و با توجه به رابطه ی مستقیم این دو کمیت، طراح با توجه به نوع پروژه (دانشجویی یا صنعتی یا آموزشی)، قیمت تمام شده طرح و سطح دانش خود میتواند:

1- پروژه را به صورت اصولی و حرفه ای انجام دهد.

2- پروژه را قبول نکرده و آبروی طراحان داخلی را حفظ کند.

ضمیمه ی شماره 6: دیتاشیت فارسی میکروکنترلر های AVR

در این ضمیمه ترجمه ی فارسی دیتاشیت تعدادی از میکروکنترلر های معروف AVR آورده شده است . در این دیتاشیت به معرفی امکانات جانبی میکروکنترلر و نقش پایه های آن و معرفی فیوز بیت ها پرداخته شده است . برای کسب اطلاعات بیشتر در مورد میکروکنترلر میتوانید به دیتاشیت اصلی آن که از سایت شرکت اتمل به نشانی WWW.ATMEL.COM قابل دریافت است مراجعه نمایید .

میکروکنترلر ATMEGA8 :

--ویژگی :

- 1- کارای بالا و توان مصرفی کم
 - 2- دارای 130 دستور که اکثر آنها در یک سیکل اجرا میشوند
 - 3- 32*8 رجیستر کاربردی
 - 4- حداکثر کریستال مورد استفاده 16مگاهرتز atmega8 و 8 مگاهرتز برای atmega8l
 - 5- سرعتی تا 16mips در فرکانس 16مگاهرتز
- حافظه ، برنامه و داده غیر فرار:

- 1- 8k بایت حافظه فلش داخلی قابل برنامه ریزی
- این حافظه میتواند تا 10000 بار نوشته و پاک شود (قابلیت پروگرم کردن تا 10000 بار)
- 2- 1024 بایت حافظه sram داخلی
- 3- 512 بایت حافظه eeprom داخلی برای ذخیره اطلاعات
- این حافظه میتواند تا 1000000 بار نوشته و پاک شود
- 4- قفل برنامه داخل حافظه flash و eeprom برای جلوگیری از خواندن آن

--خصوصیات جانبی:

- 1- دو تایمر/ کانتر 8 بیتی با prescaler مجزا و دارای مد compare (تایمر / کانتر 0 و 2)
- 2- یک تایمر/ کانتر 16 بیتی با prescaler مجزا و دارای مد compare , capture (تایمر / کانتر 1)
- 3- سه کانال pwm
- 4- 8 کانال مبدل آنالوگ به دیجیتال در بسته بندی های mlf و tqfp (دو نوع بسته بندی نصب سطحی میباشند)

- 6- کانال با دقت 10 بیت
- 2- کانال با دقت 8 بیت
- 5- 6 کانال مبدل آنالوگ به دیجیتال در بسته بندی های pdip (نوع بسته پایه دار)
- 4- کانال با دقت 10 بیت
- 2- کانال با دقت 8 بیت
- 6- دارای rtc (نوعی ساعت است که زمان و تاریخ را مستقل از عملکرد میکرو محاسبه میکند) با اسیلاتور مجزا
- 7- یک مقایسه کننده آنالوگ داخلی
- 8- Usart قابل برنامه ریزی
- 9- Watchdog قابل برنامه ریزی با اسیلاتور داخلی
- 10- ارتباط سریال isp برای برنامه ریزی (پروگرام کردن) داخل مدار (هنگامی که میکرو داخل مدار است با پروگرامر isp میتوانید میکرو را برنامه ریزی کنید، برای برنامه ریزی از چهار خط miso و mosi و sck و reset استفاده میشود)
- 11- قابلیت ارتباط سریال isp به صورت master یا slave
- خصوصیات ویژه میکرو:
- 1- Reset شدن میکرو بعد از روشن شدن
- 2- دارای 5 مد در حالت بیکاری برای مصرف کمتر انرژی و راندمان بیشتر
- 3- منبع وقفه داخلی و خارجی
- 4- دارای نوسان ساز داخلی کالیبره شده (حداکثر فرکانس این نوسان ساز 8 مگا هرتز است)
- انواع بسته بندی و تعداد پایه ها:
- 1- 23 خط ورودی و خروجی (23 پین و سه پورت b (8 پایه) و c (7 پایه) و d (8 پایه))
- 2- 5 پایه مربوط به تغذیه ها در بسته بندی pdip
- 3- 7 پایه مربوط به تغذیه ها در بسته بندی mlf و tqfp
- 4- دوپایه مربوط به adc در بسته بندی mlf و tqfp
- 5- جمع پایه 32 پایه در بسته بندی mlf و tqfp و 28 پایه در بسته بندی pdip
- حداکثر کریستال مورد استفاده
- 16-مگاهرتز atmega8

8- مگا هرتز برای atmega8l

--ولتاژ کاری

- 2.7 تا 5.5 ولت برای atmega1

- 4.5 تا 5.5 ولت برای atmega8

شکل و شرح میکرو بسته بندی نوع pdip:

پایه شماره 1 - reset / portc.6 این پایه علاوه بر نقش پین ورودی و خروجی c.6 ، (portc.6) نقش پایه reset (باز نشانی) رانیز به عهده دارد اگر به این پایه یک پالس یک به صفر داده شود برنامه از اول اجرا میشود

پایه شماره 2 - portd.0/rxd این پایه علاوه بر نقش پین ورودی و خروجی d.0 ، (portd.0) نقش پایه rxd (گیرنده اطلاعات) در ارتباط سریال رانیز به عهده دارد (این پایه و پایه txd در ارتباط سریال با هم استفاده میشوند) و هنگامی که از ارتباط سریال استفاده می شود از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد

PDIP

(RESET) PC6	1	28	PC5 (ADC5/SCL)
(RXD) PD0	2	27	PC4 (ADC4/SDA)
(TXD) PD1	3	26	PC3 (ADC3)
(INT0) PD2	4	25	PC2 (ADC2)
(INT1) PD3	5	24	PC1 (ADC1)
(XCK/T0) PD4	6	23	PC0 (ADC0)
VCC	7	22	GND
GND	8	21	AREF
(XTAL1/TOSC1) PB6	9	20	AVCC
(XTAL2/TOSC2) PB7	10	19	PB5 (SCK)
(T1) PD5	11	18	PB4 (MISO)
(AIN0) PD6	12	17	PB3 (MOSI/OC2)
(AIN1) PD7	13	16	PB2 (SS/OC1B)
(ICP1) PB0	14	15	PB1 (OC1A)

پایه شماره 3 - portd.1/txd این پایه علاوه بر نقش پین ورودی و خروجی

d.1 ، (portd.1) نقش پایه txd (فرستنده اطلاعات) در ارتباط سریال رانیز به عهده دارد (این پایه و پایه rxd در ارتباط سریال با هم استفاده میشوند) و هنگامی که از ارتباط سریال استفاده می شود از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 4 - portd.2/int0 این پایه علاوه بر نقش پین ورودی و خروجی d.2 ، (portd.2) نقش منبع وقفه خارجی 0 را نیز به عهده دارد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که وقفه خارجی راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 5 - portd.3/int1 این پایه علاوه بر نقش پین ورودی و خروجی d.3 ، (portd.3) نقش منبع وقفه خارجی 1 را نیز به عهده دارد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که وقفه خارجی راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره portd.4/xck/t0-6 این پایه علاوه بر نقش پین ورودی و خروجی d.4، (portd.4) دونقش دیگر نیز دارد t0-1: ورودی کلاک برای تایمر/ کانتر 0 است.

2- xck: به عنوان کلاک خارجی USART استفاده میشود. این پایه فقط زمانی که USART در مد اسنکرون کار میکند فعال میشود (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که یکی از امکانات گفته شده راه اندازی شود این پایه فقط کار مربوط به آن پایه را انجام میدهد و دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد).

پایه شماره VCC-7 این پایه، یکی از پایه های تغذیه میکرو می باشد که باید به VCC (5 ولت) مدار متصل شود (هر دو VCC میکرو از داخل به هم متصل میباشد)

پایه شماره GND-8 این پایه یکی از پایه های تغذیه میکرو می باشد که باید به GND (صفر ولت) مدار متصل شود (هر سه GND میکرو از داخل به هم متصل میباشد)

پایه شماره 9-10 portb.6/xtal1/tosc1 در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که از تایمر/ کانتر 2 در مد اسنکرون (میکرو به مد sleep میرود اما تایمر/ کانتر 2 به شمارش ادامه می دهد) استفاده میشود به این پایه و پایه 10 کریستال ساعت متصل میشود و همچنین در حالتی که از کریستال خارجی استفاده میشود به این پایه و پایه 10 کریستال کوارتز متصل میشود و دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد. (توجه داشته باشید که در آن واحد نمی توان از چندین نقش پورت استفاده کرد)

پایه شماره 10-11 portb.7/xtal2/tosc2 در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که از تایمر/ کانتر 2 در مد اسنکرون (میکرو به مد sleep میرود اما تایمر/ کانتر 2 به شمارش ادامه می دهد) استفاده میشود به این پایه و پایه 9 کریستال ساعت متصل میشود و همچنین در حالتی که از کریستال خارجی استفاده میشود به این پایه و پایه 9 کریستال کوارتز متصل میشود و دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد. (توجه داشته باشید که در آن واحد نمی توان از چندین نقش پورت استفاده کرد)

پایه شماره 11-12 portd.5/t1-5، d.5 (portd.5) نقش ورودی و خروجی کلاک برای تایمر/ کانتر 1 را هم عهده دار است (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که تایمر/ کانتر 1 با کلاک خارجی راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد).

پایه شماره portd.6/ain0-12 این پایه علاوه بر نقش پین ورودی و خروجی d.6 ، (portd.6) نقش ورودی مثبت مقایسه کننده آنالوگ را نیز به عهده دارد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که مقایسه کننده آنالوگ راه اندازی میشود دیگر از این پایه نمی توان به عنوان وردی یا خروجی استفاده کرد)

پایه شماره portd.7/ain1-13 این پایه علاوه بر نقش پین ورودی و خروجی d.7 ، (portd.7) نقش ورودی منفی مقایسه کننده آنالوگ را نیز به عهده دارد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که مقایسه کننده آنالوگ راه اندازی میشود دیگر از این پایه نمی توان به عنوان وردی یا خروجی استفاده کرد)

پایه شماره portb.0/icp-14 این پایه علاوه بر نقش پین ورودی و خروجی b.0 ، (portb.0) نقش ورودی capture تایمر/کانتر 1 را نیز به عهده دارد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که capture تایمر/کانتر 1 راه اندازی میشود دیگر از این پایه نمی توان به عنوان وردی یا خروجی استفاده کرد)

پایه شماره portb.1/oc1a-15 این پایه علاوه بر نقش پین ورودی و خروجی b.1 ، (portb.1) نقش خروجی مد مقایسه ای تایمر/کانتر 1 را نیز به عهده دارد همچنین این پایه به عنوان خروجی مد pwm تایمر/کانتر 1 مورد استفاده قرار میگیرد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که پایه pb0 با یک شدن ddd1 برای خروجی مد مقایسه ای تایمر/کانتر 1 ، راه اندازی میشود، یا به عنوان خروجی pwm تعریف میشود دیگر از این پایه نمی توان به عنوان وردی یا خروجی استفاده کرد)

پایه شماره portb.2/ss/oc1b-16 این پایه علاوه بر نقش پین ورودی و خروجی b.2 ، (portb.2) نقش خروجی دیگر مد مقایسه ای تایمر/کانتر 1 را نیز به عهده دارد همچنین این پایه به عنوان خروجی مد pwm تایمر/کانتر 1 مورد استفاده قرار میگیرد هم چنین زمانی که ارتباط spi راه اندازی میشود این پایه در میکرو slave ورودی تعریف میشود و با صفر شدن این پایه spi فعال میگردد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که پایه pb2 با یک شدن ddd2 برای خروجی مد مقایسه ای تایمر/کانتر 1 ، راه اندازی میشود، یا به عنوان خروجی pwm تعریف میشود، دیگر از این پایه نمی توان به عنوان وردی یا خروجی استفاده کرد اما در ارتباط spi میتوان در میکرو master ان را به عنوان ورودی یا خروجی تعریف کرد)

پایه شماره portb.3/mosi/oc2-17 این پایه علاوه بر نقش پین ورودی و خروجی b.3 ، (portb.3) نقش خروجی مد مقایسه ای تایمر/کانتر 2 را نیز به عهده دارد همچنین این پایه به عنوان خروجی مد pwm تایمر/کانتر 2 مورد استفاده قرار میگیرد و در ارتباط spi این پایه برای ورودی داده slave و خروجی داده master استفاده میشود (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که پایه pb3 با یک شدن ddd3 برای خروجی مد مقایسه ای تایمر/کانتر 2 ، راه اندازی میشود، یا به عنوان خروجی pwm تعریف میشود دیگر از این پایه نمی توان به عنوان وردی یا خروجی

استفاده کرد زمانی که میکرو در ارتباط spi به صورت master شکل دهی میشود خروجی است و زمانی که به صورت slave شکل دهی میشود این پایه ورودی است.)

پایه شماره portb.4/miso-18 این پایه علاوه بر نقش پین ورودی و خروجی b.4، (portb.4) در ارتباط spi این پایه برای ورودی داده master و خروجی داده slave استفاده میشود (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما زمانی که میکرو در ارتباط spi به صورت master شکل دهی میشود ورودی است و زمانی که به صورت slave شکل دهی میشود این پایه خروجی است.)

پایه شماره portb.5/sck-19 این پایه علاوه بر نقش پین ورودی و خروجی b.5، (portb.5) در ارتباط spi این پایه برای کلاک خروجی میکرو master و کلاک ورودی میکرو slave استفاده میشود (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما زمانی که میکرو در ارتباط spi به صورت master شکل دهی میشود خروجی است و زمانی که به صورت slave شکل دهی میشود این پایه ورودی است.)

پایه شماره avcc-20 این پایه برای تغذیه مبدل آنالوگ به دیجیتال استفاده میشود، زمانی که از مبدل آنالوگ به دیجیتال استفاده نمی شود این پایه آزاد است، و زمانی که از مبدل آنالوگ به دیجیتال (adc) استفاده میشود این پایه باید به vcc متصل گردد (اختلاف ولتاژ این پایه با vcc نباید بیشتر از 3. ولت باشد (در غیر این صورت محاسبات دقیق نخواهد بود))

پایه شماره aref-21 این پایه برای ولتاژ مبدل آنالوگ به دیجیتال استفاده میشود، زمانی که از مبدل آنالوگ به دیجیتال استفاده نمی شود این پایه آزاد است، و زمانی که از مبدل آنالوگ به دیجیتال (adc) استفاده میشود این پایه باید طبق برنامه نوشته شده به vcc یا ولتاژ مرجع متصل گردد (در صورتی که از ولتاژ مرجع استفاده نشود (ولتاژ مرجع زمین گرفته شود) این پایه آزاد خواهد بود)

پایه شماره gnd - 22 دیگر پایه gnd میکرو می باشد که باید به gnd (صفر ولت) مدار متصل شود (هر دو gnd میکرو از داخل به هم متصل میباشد)

پایه شماره portc.0/adc(0)-23 این پایه علاوه بر نقش پین ورودی و خروجی c.0، (portc.0) به عنوان ورودی مبدل آنالوگ به دیجیتال صفر (adc(0)) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره portc.1/adc(1)-24 این پایه علاوه بر نقش پین ورودی و خروجی c.1، (portc.1) به عنوان ورودی مبدل آنالوگ به دیجیتال صفر (adc(1)) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

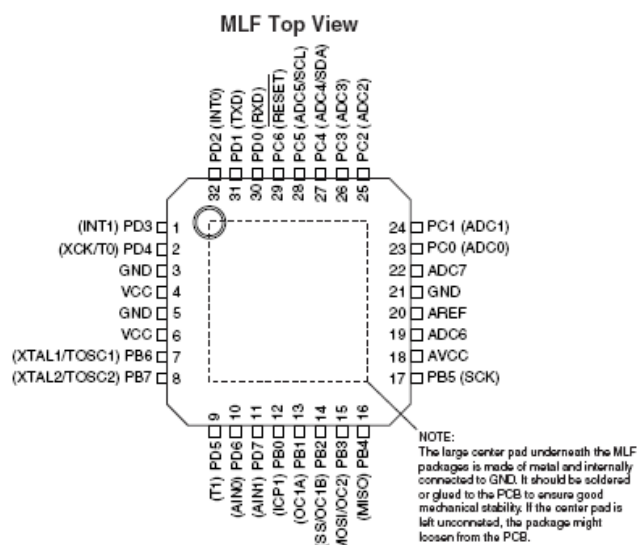
پایه شماره 25 - portc.2/adc(2) این پایه علاوه بر نقش پین ورودی و خروجی c.2، (portc.2) به عنوان ورودی مبدل آنالوگ به دیجیتال صفر (adc(2)) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 26 - portc.3/adc3 این پایه علاوه بر نقش پین ورودی و خروجی c.3، (portc.3) به عنوان ورودی مبدل آنالوگ به دیجیتال صفر (adc(3)) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

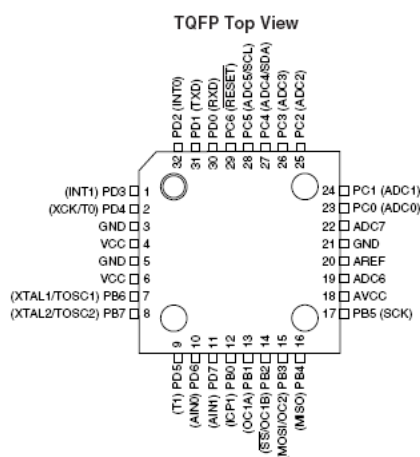
پایه شماره 27 - portc.4/adc(4)/sda این پایه علاوه بر نقش پین ورودی و خروجی c.4، (portc.4) به عنوان ورودی مبدل آنالوگ به دیجیتال صفر (adc(4)) استفاده میشود همچنین در زمان ارتباط 2-wire به عنوان خط داده استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 28 - portc.5/adc(5)/scl این پایه علاوه بر نقش پین ورودی و خروجی c.5، (portc.5) به عنوان ورودی مبدل آنالوگ

به دیجیتال صفر (adc(5)) استفاده میشود همچنین در زمان ارتباط 2-wire به عنوان خط کلاک استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)



در بالا دو نوع بسته بندی دیگر را مشاهده می کنید کار پایه ها مشابه بسته بندی pdip می باشد و فقط جای آنها عوض شده است .

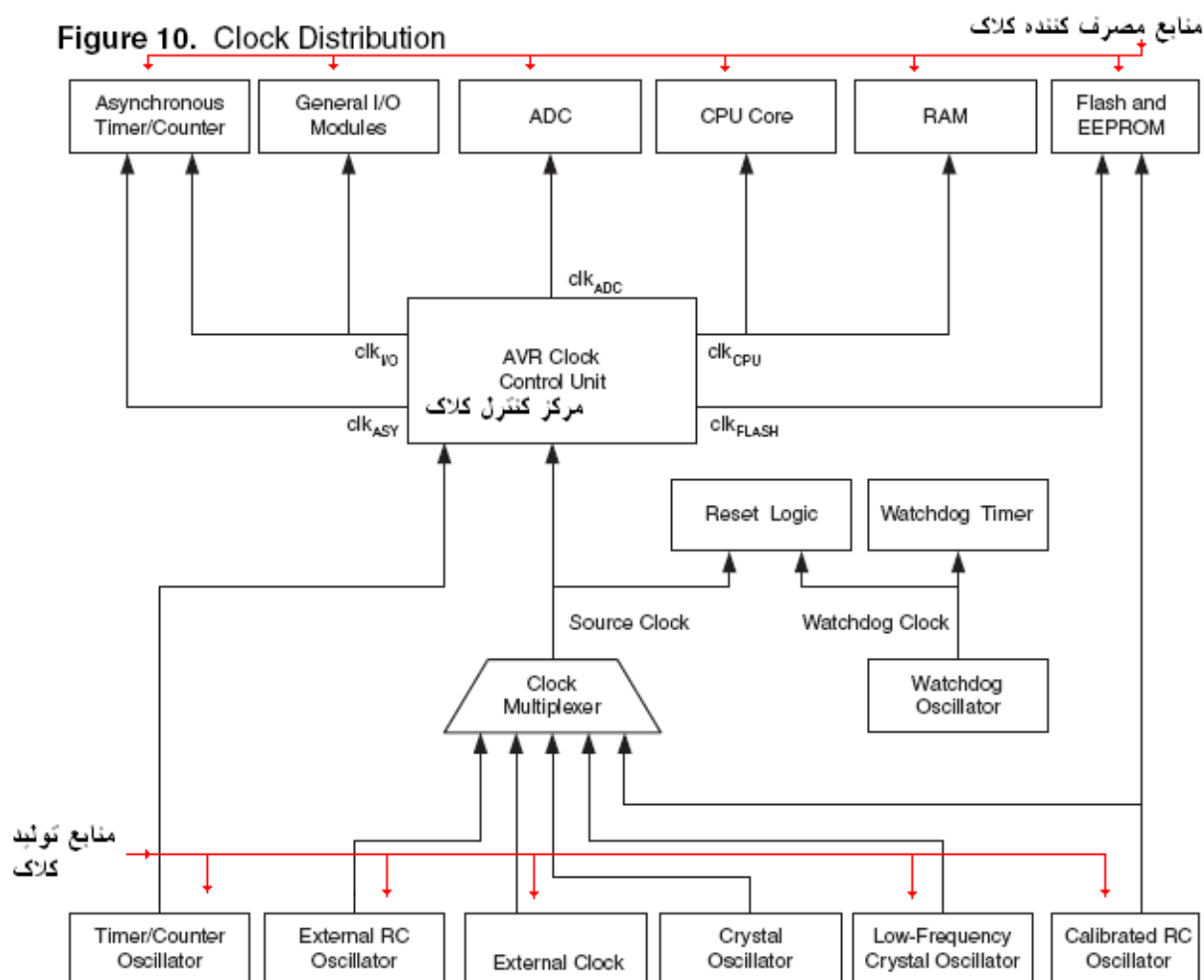


پایه های 19 و 22 در این بسته بندی ورودی مبدل آنالوگ به دیجیتال کانال 6 و 7 می باشند و فقط موقعی که مبدل آنالوگ به دیجیتال راه اندازی میشود می توان از آنها استفاده کرد .

در این بسته بندی هر سه پایه گراند از داخل به هم متصل میباشد ،این موضوع در مورد VCC نیز صدق میکند.

منابع تولید کلاک سیستم:

کلاک سیستم در این میکرو مطابق شکل زیر توضیح شده است:



انواع منابع تولید کلاک در avr:

Device Clocking Option	نام سخت افزار نوسان ساز	CKSEL3..0
External Crystal/Ceramic Resonator	کریستال خارجی نوسان ساز سرامیکی /	1111 - 1010
External Low-frequency Crystal	نوسان ساز کریستالی فرکانس پایین	1001
External RC Oscillator	نوسان ساز rc خارجی	1000 - 0101
Calibrated Internal RC Oscillato	نوسان ساز rc داخلی میکرو	0100 - 0001
External Clock	کلاک خارجی	0000

میکرو های avr دارای چندین منبع برای تولید پالس کلاک میباشند که کاربر میتواند با برنامه ریزی فیوز بیت ها ، یکی از آنها را به عنوان منبع تامین کلاک انتخاب کند . ، در تمام جداول فیوز بیت ، صفر به معنای برنامه ریزی شده و 1 به معنای عدم برنامه ریزی است

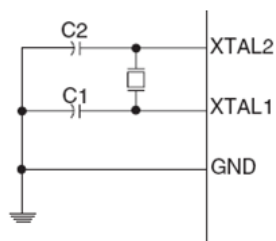
طریقه اتصال نوسان ساز به میکرو:

1- کریستال خارجی / نوسان ساز سرامیکی:

در این حالت کریستال کوارتز یا نوسان ساز سرامیکی به دو پایه xtal 1 و xtal 2 متصل میشود

خازن های c1 و c2 برای جلوگیری از تاثیر نویز محیط بر روی نوسان ساز می باشد که مقدار آنها بستگی به مقدار نویز محیط دارد مقدار پیشنهادی این خازن ها برای فرکانس های مختلف در جدول زیر آورده شده است:

Crystal Oscillator Connections



Frequency Range(MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
0.4 - 0.9	—
0.9 - 3.0	12 - 22
3.0 - 8.0	12 - 22
$1.0 \leq$	12 - 22

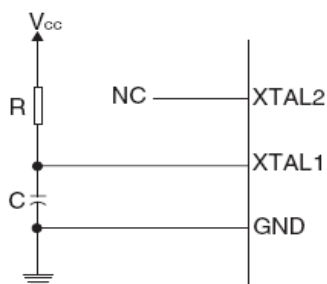
2- نوسان ساز کریستالی فرکانس پایین:

این نوع نوسان ساز که به کریستال ساعت نیز معروف است (32.768khz) مطابق شکل بالا به دو پایه xtal 1 و xtal 2 متصل میشود ، برای این کریستال مقدار خازن ها 36 pf است.

3- نوسان ساز rc خارجی:

فرکانس این نوسان ساز از معادله $f=1/(3rc)$ بدست میاید نحوه اتصال این نوسان ساز به میکرو در شکل زیر آورده شده است .

External RC Configuration



کمترین مقدار خازن باید 22pf باشد تا نوسانات پایدار بماند.
این نوع نوسان ساز می تواند در 4 مد فرکانسی کار کند که این فرکانسها با تنظیم فیوز بیت های 0...3 cksels قابل انتخاب است در جدول زیر مد های عملیاتی نوسان ساز rc خارجی آورده شده است.

External RC Oscillator Operating Modes

CKSEL3..0	Frequency Range (MHz)
0101	0.1 - 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0

4-نوسان ساز rc داخلی میکرو:

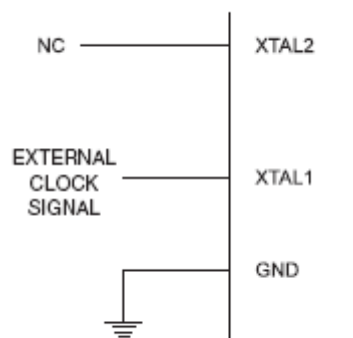
این نوسان ساز کلاک های نامی 1 و 2 و 4 و 8 مگاهرتز را در ولتاژ 5 ولت و دمای 25 درجه سانتی گراد تولید میکند در حالت عادی فیوز بیت مربوط به این نوع نوسان ساز برنامه ریزی شده است و میکرو با این نوسان ساز کار میکند (با فرکانس 1 مگا هرتز)، شما میتوانید با برنامه ریزی فیوز بیت های ckse3...0 طبق جدول زیر مقدار فرکانس را در رنج مربوطه قرار دهید

Internal Calibrated RC Oscillator Operating Modes

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

5- کلاک خارجی :

برای راه اندازی میکرو توسط کلاک خارجی پایه 1 xtal باید مطابق شکل زیر وصل شود در این مد، کلاک خارجی باید دارای ثبات بالا باشد، در صورتی که فرکانس تغییر کند میکرو رفتار غیر قابل انتظاری از خود نشان میدهد



حال که با طریقه برنامه ریزی فیوز بیت های مربوط به کریستال آشنا شدید ، نکات زیر را مد نظر داشته باشید:

- 1- در صورتی که فیوز بیت مربوط به یکی از نوسان ساز ها برنامه ریزی شود میکرو فقط با آن نوسان ساز راه اندازی میشود. مثلا اگر شما فیوز بیت cksel را روی 0000 برنامه ریزی کنید ، میکرو فقط با کلاک خارجی راه اندازی

میشود، حتی اگر موقع کار کلاک خارجی قطع شود، میکرو خاموش میگردد، این حالت برای پروگرام کردن میکرو نیز صادق است (بدون کلاک خارجی میکرو پروگرام نمیشود).

2- برای اطمینان از پروگرام کردن فیوز بیت ها میتوانید کلاک را قطع کنید (نوسان ساز را از میکرو جدا کنید)، اگر میکرو به کار خود ادامه داد فیوز بیت مربوطه درست برنامه ریزی نشده است و اگر میکرو خاموش شد، فیوز بیت مربوطه درست برنامه ریزی شده است.

3- نوسان ساز های سرامیکی در انواع مختلف ساخته میشود و نمی توان از هر خازنی به عنوان خازن نویز گیر استفاده کرد، شما فقط میتوانید از خازن های پیشنهادی کارخانه تولید کننده استفاده کنید.

4- در این میکرو هنگامی که از کریستال خارجی استفاده میشود، نمیتوان از پایه های portb.6 و portb.7 به عنوان ورودی یا خروجی استفاده کرد.

فیوز بیت های دیگر این میکرو :

Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
RSTDISBL	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)

1-Rstdisbl: در حالت پیش فرض portc.6 پایه reset میکرو میباشد، با برنامه ریزی این فیوز بیت این پایه به عنوان

ورودی و خروجی دیجیتال استفاده میشود، اگر این فیوز بیت برنامه ریزی شود دیگر نمی توان میکرو را با پروگرامر های معمولی برنامه ریزی کرد.

2-Wdton: اگر این فیوز بیت برنامه ریزی شود تایمر watchdog را برای همیشه روشن میکند (این تایمر را به صورت

نرم افزاری میتوان روشن یا خاموش کرد) (این تایمر میکرو را پس از یک زمان مشخص ریست میکند)

3-Spien: این فیوز بیت در حالت پیش فرض برنامه ریزی شده و می توان میکرو از طریق ارتباط isp برنامه ریز کرد

در صورتی که این فیوز بیت پاک شود، دیگر نمیتوان میکرو از طریق ارتباط isp برنامه ریزی کرد (این فیوز بیت

با پروگرامر های خاص برنامه ریزی میشود.

Ckopt-4: اگر این فیوز بیت برنامه ریزی شود خازن های داخلی میکرو بر روی دو پایه xtal1 و xtal2 راه اندازی میشوند و دیگر نیازی به استفاده از خازن بروری دو پایه xtal1 و xtal2 نیست. (فعال کردن این فیوز بیت باعث افزایش مصرف انرژی میشود) (این بیت در حالت پیش فرض برنامه ریزی نشده است).

Eesave-5: اگر این فیوز بیت برنامه ریزی شود در زمان پاک کردن حافظه فلش میکرو (erase کردن فلش در هنگام پروگرام کردن) محتویات eeprom محفوظ می ماند (این بیت در حالت پیش فرض برنامه ریزی نشده است).

bootsz1-6: این فیوز بیت و فیوز بیت بعدی برای انتخاب میزان حافظه boot طبق جدول زیر برنامه ریزی میشوند:

Boot Size Configuration							
BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	128 words	4	0x000 - 0xF7F	0xF80 - 0xFFF	0xF7F	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xEFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xDFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xBFF	0xC00

(حافظه فلش میکرو های avr که دارای Bootloader هستند از دو بخش اصلی Application و Bootloader تشکیل شده است که برنامه های کاربردی در بخش Application ذخیره میشود و برنامه پشتیبان در حافظه Boot).

Bootsz0-7: این فیوز بیت و فیوز بیت قبلی برای انتخاب میزان حافظه boot طبق جدول بالا برنامه ریزی میشوند:

Boostrst-8: این فیوز بیت ادرس بردار ریست را تغییر میدهد. (در حالت عادی (هنگامی که این فیوز بیت برنامه ریزی نشده باشد) بعد از ریست شدن میکرو برنامه از خانه 0000 حافظه شروع به اجرا میکند، اما اگر این فیوز بیت برنامه ریزی شود بعد از ریست شدن میکرو برنامه از ادرسی که بوسیله دو فیوز بیت bootsz1 و bootsz0 تعیین شده شروع میشود

Bodlevel -9: در حالت عادی (هنگامی که این فیوز بیت برنامه ریزی نشده باشد) اگر ولتاژ تغذیه میکرو از 2.7 ولت پایین تر بیاید میکرو ریست میشود، اما اگر این فیوز بیت برنامه ریزی شود، هنگامی که ولتاژ تغذیه میکرو از 4 ولت کمتر شود میکرو ریست میشود (این فیوز بیت مخصوص atmega8l میباشد)

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⁾

10- Boden: این فیوز بیت در حالت پیش فرض برنامه ریزی نشده است اما اگر برنامه ریزی شود سیستم brown-out

راه اندازی میشود (این سیستم یک اشکار ساز است که در طول عملکرد میکرو سطح ولتاژ منع تغذیه را با یک

ولتاژ مرجع داخلی مقایسه میکند و در صورتیکه V_{CC} از ولتاژ مرجع بیشتر شود میکرو ریست میشود اگر این فیوز

بیت به صورت 01 برنامه ریزی شود ولتاژ مرجع 2.7 ولت است و اگر به صورت 00 برنامه ریزی شود ولتاژ مرجع 4 ولت

است و اگر به صورت 11 یا 10 برنامه ریزی شود غیر فعال میگردد

11- Sut1-12 و sut0: این دو فیوز بیت برای انتخاب زمان start-up استفاده میشود. (طبق جدول زیر)

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

زمان start-up با توجه نوع کریستال تعیین میشود (هنگامی که میکرو ریست میشود (میکرو در هنگام روشن شدن نیز ریست

میشود) چند میلی ثانیه طول میکشد تا نوسانات کریستال پایدار شود بعد از اینکه نوسانات نوسان ساز پایدار شد cpu شروع به

اجرای برنامه از اولین خانه حافظه میکند به مدت زمانی که طول میکشد تا نوسانات پایدار شود زمان start-up میگویند و این

زمان برای انواع کریستال متفاوت میباشد)

13-14-15-16-cksel0 وcksel1 وcksel2 وcksel3: این چهار فیوز بیت مربوط به انتخاب نوسان ساز میباشد که در بالا گفته شد

دیگر خصوصیات:

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to $V_{CC}+0.5V$
Voltage on $\overline{\text{RESET}}$ with respect to Ground.....	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins.....	200.0 mA

دیتاشیت فارسی atmega32

--ویژگی:

- 6- کارای بالا و توان مصرفی کم
- 7- دارای 131 دستور که اکثر آنها در یک سیکل اجرا میشوند
- 8- 32*8 رجیستر کاربردی
- 9- حداکثر کریستال مورد استفاده 16 مگاهرتز atmega32 و 8 مگاهرتز برای atmega32l
- 10- سرعتی تا 16mips در فرکانس 16 مگاهرتز

--حافظه، برنامه و داده غیر فرار:

- 5- 32 k بایت حافظه فلش داخلی قابل برنامه ریزی
- این حافظه میتواند تا 10000 بار نوشته و پاک شود (قابلیت پروگرام کردن تا 10000 بار)
- 6- 2k بایت حافظه sram داخلی
- 7- 1024 بایت حافظه eeprom داخلی برای ذخیره اطلاعات
- این حافظه میتواند تا 1000000 بار نوشته و پاک شود
- 8- قفل برنامه داخل حافظه flash و eeprom برای جلوگیری از خواندن آن

--خصوصیات جانبی:

- 12- دو تایمر / کانتر 8 بیتی با prescaler مجزا و دارای مد compare (تایمر / کانتر 0 و 2)
- 13- یک تایمر / کانتر 16 بیتی با prescaler مجزا و دارای مد compare , capture (تایمر / کانتر 1)
- 14- چهار کانال pwm
- 15- 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
- 8- کانال single-ended
- دارای 7 کانال تفاضلی در بسته بندی tqfp (این نوع adc اختلاف بین دو ولتاژ را اندازه میگیرد در حالی که adc های --معمولی ولتاژ ورودی را نسبت به زمین اندازه میگیرد)
- دارای دو کانال تفاضلی با گین 1x و 10x یا 200x
- 16- دارای rtc (نوعی ساعت است که زمان و تاریخ را مستقل از عملکرد میکرو محاسبه میکند) با اسلاتور مجزا
- 17- یک مقایسه کننده آنالوگ داخلی

18- Usart قابل برنامه ریزی

19- Watchdog قابل برنامه ریزی با اسیلاتور داخلی

20- ارتباط سریال isp برای برنامه ریزی (پروگرام کردن) داخل مدار (هنگامی که میکرو داخل مدار است با پروگرامر

isp میتوانید میکرو را برنامه ریزی کنید، برای برنامه ریزی از چهار خط miso و mosi و sck و reset استفاده

میشود)

21- قابلیت ارتباط سریال isp به صورت master یا slave

22- قابلیت ارتباط jtag (یک نوع ارتباط است که از طریق آن می توان کلیه حافظه های قابل برنامه ریزی میکرو را

خواند یا نوشت)

خصوصیات ویژه میکرو:

5- Reset شدن میکرو بعد از روشن شدن

6- دارای 5 مد در حالت بیکاری برای مصرف کمتر انرژی و راندمان بیشتر

7- منبع وقفه داخلی و خارجی

8- دارای نوسان ساز داخلی کالیبره شده (حداکثر فرکانس این نوسان ساز 8 مگا هرتز است)

--انواع بسته بندی و تعداد پایه ها:

6- 32 خط ورودی و خروجی (32 پین و چهار پورت a (8 پایه) و b (8 پایه) و c (8 پایه) و d (8 پایه))

7- 3 پایه مربوط به تغذیه ها در بسته بندی pdip

8- 7 پایه مربوط به تغذیه ها در بسته بندی mlf و tqfp

9- دو پایه مربوط به کریستال در بسته بندی mlf و tqfp و pdip

10- یک پایه مربوط به میکرو reset در بسته بندی mlf و tqfp و pdip

11- دو پایه مربوط به تغذیه adc و ولتاژ مرجع آن در بسته بندی mlf و tqfp و pdip

12- جمع پایه 44 پایه در بسته بندی mlf و tqfp و 40 پایه در بسته بندی pdip

--حداکثر کریستال مورد استفاده

16-مگا هرتز atmega32

8- مگا هرتز برای atmega32l

--ولتاژ کاری

– 2.7 تا 5.5 ولت برای atmega32l

– 4.5 تا 5.5 ولت برای atmega32

شکل و شرح میکرو بسته بندی نوع pdip:

پایه شماره 1 – portb.0/xck/t0 این پایه علاوه بر نقش پین ورودی و خروجی b.0، portb.0)دو نقش دیگر نیز دارد 1 – t0: ورودی کلاک برای تایمر / کانتر 0 است.

2 – xck: به عنوان کلاک خارجی usart استفاده میشود. این پایه فقط زمانی که Usart در مد اسنکرون کار میکند فعال میشود (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که یکی از امکانات گفته شده راه اندازی شود این پایه فقط کار مربوط به آن پایه را انجام میدهد و دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد).

پایه شماره 2 – portb.1/t1 این پایه علاوه بر نقش پین ورودی و خروجی b.1، portb.1) نقش ورودی کلاک برای تایمر / کانتر 1 را هم عهده دار است (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که تایمر / کانتر 1 با کلاک خارجی راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد).

پایه شماره 3 – portb.2/int2/ain0 این پایه علاوه بر نقش پین ورودی و خروجی b.2، portb.2) نقش ورودی مثبت مقایسه

کننده آنالوگ را نیز به عهده دارد، نقش دیگر این پایه به عنوان منبع وقفه

خارجی دو است. (در حالت عادی این پایه به عنوان ورودی و خروجی

دیجیتال (i/o) استفاده می شود اما وقتی که وقفه خارجی یا مقایسه کننده

آنالوگ راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا

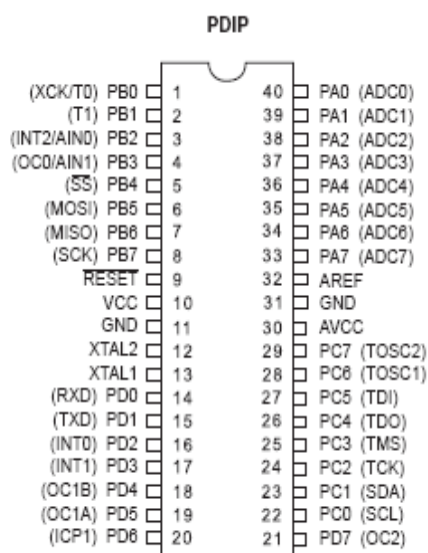
خروجی استفاده کرد) پایه شماره 4 – portb.3/oc0/ain1 این پایه علاوه

بر نقش پین ورودی و خروجی b.3، portb.3) نقش ورودی منفی مقایسه

کننده آنالوگ را نیز به عهده دارد، این پایه خروجی مد مقایسه ای

تایمر / کانتر 2 را نیز به عهده دارد همچنین این پایه به عنوان خروجی مد

pwm تایمر / کانتر 2 مورد



استفاده قرار میگیرد. (در حالت عادی این پایه به عنوان ورودی و خروجی

دیجیتال (i/o) استفاده می شود اما وقتی که وقفه خارجی یا مقایسه کننده آنالوگ یا تایمر دو در مد مقایسه ای یا pwm راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد (پایه شماره 5- portb.4/ss این پایه علاوه بر نقش پین ورودی و خروجی 4.b، (portb.4) زمانی که ارتباط spi راه اندازی میشود این پایه در میکرو slave ورودی تعریف میشود و با صفر شدن این پایه spi فعال میگردد (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما، در ارتباط spi نمیتوان در میکرو slave از این پایه به عنوان ورودی یا خروجی استفاده کرد اما در میکرو master میتوان آن را به عنوان ورودی یا خروجی تعریف کرد)

پایه شماره 6- portb.5/mosi این پایه علاوه بر نقش پین ورودی و خروجی 5.b، (portb.5) در ارتباط spi این پایه برای ورودی داده slave و خروجی داده master استفاده میشود (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود زمانی که میکرو در ارتباط spi به صورت master شکل دهی میشود خروجی است و زمانی که به صورت slave شکل دهی میشود این پایه ورودی است).

پایه شماره 7- portb.6/miso این پایه علاوه بر نقش پین ورودی و خروجی 6.b، (portb.6) در ارتباط spi این پایه برای ورودی داده master و خروجی داده slave استفاده میشود (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود زمانی که میکرو در ارتباط spi به صورت master شکل دهی میشود ورودی است و زمانی که به صورت slave شکل دهی میشود این پایه خروجی است).

پایه شماره 8- portb.7/sck این پایه علاوه بر نقش پین ورودی و خروجی 7.b، (portb.7) در ارتباط spi این پایه برای کلاک خروجی میکرو master و کلاک ورودی میکرو slave استفاده میشود (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما زمانی که میکرو در ارتباط spi به صورت master شکل دهی میشود خروجی است و زمانی که به صورت slave شکل دهی میشود این پایه ورودی است).

پایه شماره 9- reset: نقش پایه reset (باز نشانی میکرو) رانیز به عهده دارد (اگر به این پایه یک پالس یک به صفر داده شود برنامه از اول اجرا میشود)

پایه شماره 10- vcc این پایه، یکی از پایه های تغذیه میکرو می باشد که باید به vcc (5 ولت) مدار متصل شود (هر دو vcc میکرو از داخل به هم متصل میباشد)

پایه شماره 11- gnd این پایه یکی از پایه های تغذیه میکرو می باشد که باید به gnd (صفر ولت) مدار متصل شود (هر سه gnd میکرو از داخل به هم متصل میباشد)

پایه شماره 12- xtla2 کریستال خارجی بین این پایه و پایه xtla1 قرار میگیرد

پایه شماره 13- xtal1 کریستال خارجی بین این پایه و پایه xtal2 قرار میگیرد

پایه شماره 14- portd.0/txd این پایه علاوه بر نقش پین ورودی و خروجی d.0، (portd.0) نقش پایه rxd (گیرنده اطلاعات) در ارتباط سریال رانیز به عهده دارد (این پایه و پایه txd در ارتباط سریال با هم استفاده میشوند) و هنگامی که از ارتباط سریال استفاده می شود از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 15- portd.1/txd این پایه علاوه بر نقش پین ورودی و خروجی d.1، (portd.1) نقش پایه txd (فرستنده اطلاعات) در ارتباط سریال رانیز به عهده دارد (این پایه و پایه rxd در ارتباط سریال با هم استفاده میشوند) و هنگامی که از ارتباط سریال استفاده می شود از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 16- portd.2/int0 این پایه علاوه بر نقش پین ورودی و خروجی d.2، (portd.2) نقش منبع وقفه خارجی 0 را نیز به عهده دارد (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که وقفه خارجی راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 17- portd.3/int1 این پایه علاوه بر نقش پین ورودی و خروجی d.3، (portd.3) نقش منبع وقفه خارجی 1 را نیز به عهده دارد (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که وقفه خارجی راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 18- portd.4/oc1b این پایه علاوه بر نقش پین ورودی و خروجی d.4، (portd.4) نقش خروجی دیگر مد مقایسه ای تایمر/کانتر 1 را نیز به عهده دارد همچنین این پایه به عنوان خروجی مد pwm تایمر/کانتر 1 مورد استفاده قرار میگیرد (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود، اما وقتی که پایه به عنوان خروجی pwm تعریف میشود، دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد اما در ارتباط spi میتوان در میکرو master آن را به عنوان ورودی یا خروجی تعریف کرد)

. پایه شماره 19- portd.5/oc1a این پایه علاوه بر نقش پین ورودی و خروجی d.5، (portd.5) نقش خروجی مد مقایسه ای تایمر/کانتر 1 را نیز به عهده دارد همچنین این پایه به عنوان خروجی مد pwm تایمر/کانتر 1 مورد استفاده قرار میگیرد (در حالت عادی این پایه به عنوان ورودی و خروجی دیجیتال (i/o) استفاده می شود اما وقتی که پایه pd5 با یک شدن ddd5 برای خروجی مد مقایسه ای تایمر/کانتر 1، راه اندازی میشود، یا به عنوان خروجی pwm تعریف میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 20-1 portd.6/icp این پایه علاوه بر نقش پین ورودی و خروجی d.6، (portd.6) نقش ورودی capture تایمر/کانتر 1 را نیز به عهده دارد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که capture تایمر/کانتر 1 راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 21-2 portd.7/oc این پایه علاوه بر نقش پین ورودی و خروجی d.7، (portd.7) نقش خروجی مد مقایسه ای تایمر/کانتر 2 را نیز به عهده دارد همچنین این پایه به عنوان خروجی مد pwm تایمر/کانتر 2 مورد استفاده قرار میگیرد (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که پایه pb3 با یک شدن ddd3 برای خروجی مد مقایسه ای تایمر/کانتر 2، راه اندازی میشود، یا به عنوان خروجی pwm تعریف میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد).

پایه شماره 22 -2 portc.0/scl این پایه علاوه بر نقش پین ورودی و خروجی c.0، (portc.0) در زمان ارتباط 2-wire به عنوان خط کلاک استفاده میشود (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که پایه ارتباط 2-wire راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد).

پایه شماره 23 -1 portc.1/sda این پایه علاوه بر نقش پین ورودی و خروجی c.1، (portc.1) در زمان ارتباط 2-wire به عنوان خط داده استفاده میشود (در حالت عادی این پایه به عنوان ورودی خروجی دیجیتال (i/o) استفاده می شود اما وقتی که پایه ارتباط 2-wire راه اندازی میشود دیگر از این پایه نمی توان به عنوان ورودی یا خروجی استفاده کرد).

پایه شماره 24 -2 portc.2/tck این پایه علاوه بر نقش پین ورودی و خروجی c.2، (portc.2) هنگامی که ارتباط jtag استفاده میشود این پایه به عنوان خط کلاک استفاده میشود و دیگر نمی توان از آن به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 25 -3 portc.3/tms این پایه علاوه بر نقش پین ورودی و خروجی c.3، (portc.3) هنگامی که ارتباط jtag استفاده میشود این پایه به عنوان خط فرمان استفاده میشود و دیگر نمی توان از آن به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 26 -4 portc.4/tdo این پایه علاوه بر نقش پین ورودی و خروجی c.4، (portc.4) هنگامی که ارتباط jtag استفاده میشود این پایه به عنوان خط خروجی داده سریال استفاده میشود و دیگر نمی توان از آن به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 27 -5 portc.5/tdi این پایه علاوه بر نقش پین ورودی و خروجی c.5، (portc.5) هنگامی که ارتباط jtag استفاده میشود این پایه به عنوان خط ورودی داده سریال استفاده میشود و دیگر نمی توان از آن به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 28 — portc.6/tosc1 این پایه علاوه بر نقش پین ورودی و خروجی c.6، (portc.6) وقتی که از تایمر / کانتر 2 در مد اسنکرون (میکرو به مد sleep میرود اما تایمر / کانتر 2 به شمارش ادامه می دهد) استفاده میشود به این پایه و پایه 29 کریستال ساعت متصل میشود و دیگر نمی توان از آن به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 29 — portc.7/tosc2 این پایه علاوه بر نقش پین ورودی و خروجی c.7، (portc.7) وقتی که از تایمر / کانتر 2 در مد اسنکرون (میکرو به مد sleep میرود اما تایمر / کانتر 2 به شمارش ادامه می دهد) استفاده میشود به این پایه و پایه 28 کریستال ساعت متصل میشود و دیگر نمی توان از آن به عنوان ورودی یا خروجی استفاده کرد

پایه شماره 30 — avcc این پایه برای تغذیه مبدل آنالوگ به دیجیتال استفاده میشود، زمانی که از مبدل آنالوگ به دیجیتال استفاده نمی شود این پایه آزاد است، و زمانی که از مبدل آنالوگ به دیجیتال (adc) استفاده میشود این پایه باید به vcc متصل گردد (اختلاف ولتاژ این پایه با vcc نباید بیشتر از 3. ولت باشد (در غیر این صورت محاسبات دقیق نخواهد بود))

پایه شماره 31 — gnd این پایه یکی از پایه های تغذیه میکرو می باشد که باید به gnd (صفر ولت) مدار متصل شود (هر سه gnd میکرو از داخل به هم متصل میباشد)

پایه شماره 32 — aref این پایه برای ولتاژ مبدل آنالوگ به دیجیتال استفاده میشود، زمانی که از مبدل آنالوگ به دیجیتال استفاده نمی شود این پایه آزاد است، و زمانی که از مبدل آنالوگ به دیجیتال (adc) استفاده میشود این پایه باید طبق برنامه نوشته شده به vcc یا ولتاژ مرجع متصل گردد (در صورتی که از ولتاژ مرجع استفاده نشود (ولتاژ مرجع زمین گرفته شود) این پایه آزاد خواهد بود)

پایه شماره 33 — porta.7/adc(7) این پایه علاوه بر نقش پین ورودی و خروجی a.7، (porta.7) به عنوان ورودی مبدل آنالوگ به دیجیتال صفر (adc(7)) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 34 — porta.6/adc(6) این پایه علاوه بر نقش پین ورودی و خروجی a.6، (porta.6) به عنوان ورودی مبدل آنالوگ به دیجیتال صفر (adc(6)) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

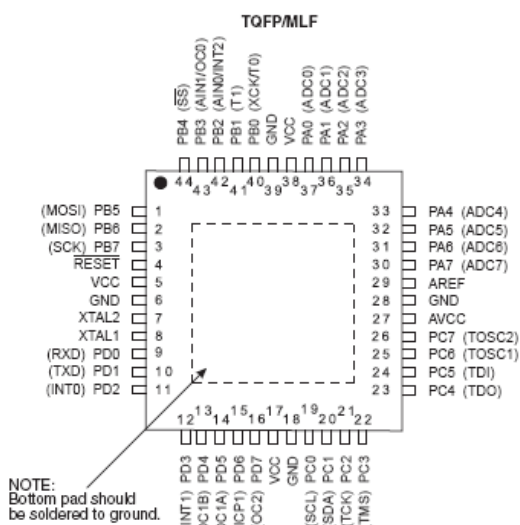
پایه شماره 35 — porta.5/adc(5) این پایه علاوه بر نقش پین ورودی و خروجی a.5، (porta.5) به عنوان ورودی مبدل آنالوگ به دیجیتال صفر (adc(5)) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 36 — $\text{porta.4}/\text{adc}(4)$ این پایه علاوه بر نقش پین ورودی و خروجی a.4 ، porta.4 به عنوان ورودی مبدل آنالوگ به دیجیتال صفر ($\text{adc}(4)$) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 37 — $\text{porta.3}/\text{adc}(3)$ این پایه علاوه بر نقش پین ورودی و خروجی a.3 ، porta.3 به عنوان ورودی مبدل آنالوگ به دیجیتال صفر ($\text{adc}(3)$) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

پایه شماره 38 — $\text{porta.2}/\text{adc}(2)$ این پایه علاوه بر نقش پین ورودی و خروجی a.2 ، porta.2 به عنوان ورودی مبدل آنالوگ به دیجیتال صفر ($\text{adc}(2)$) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)

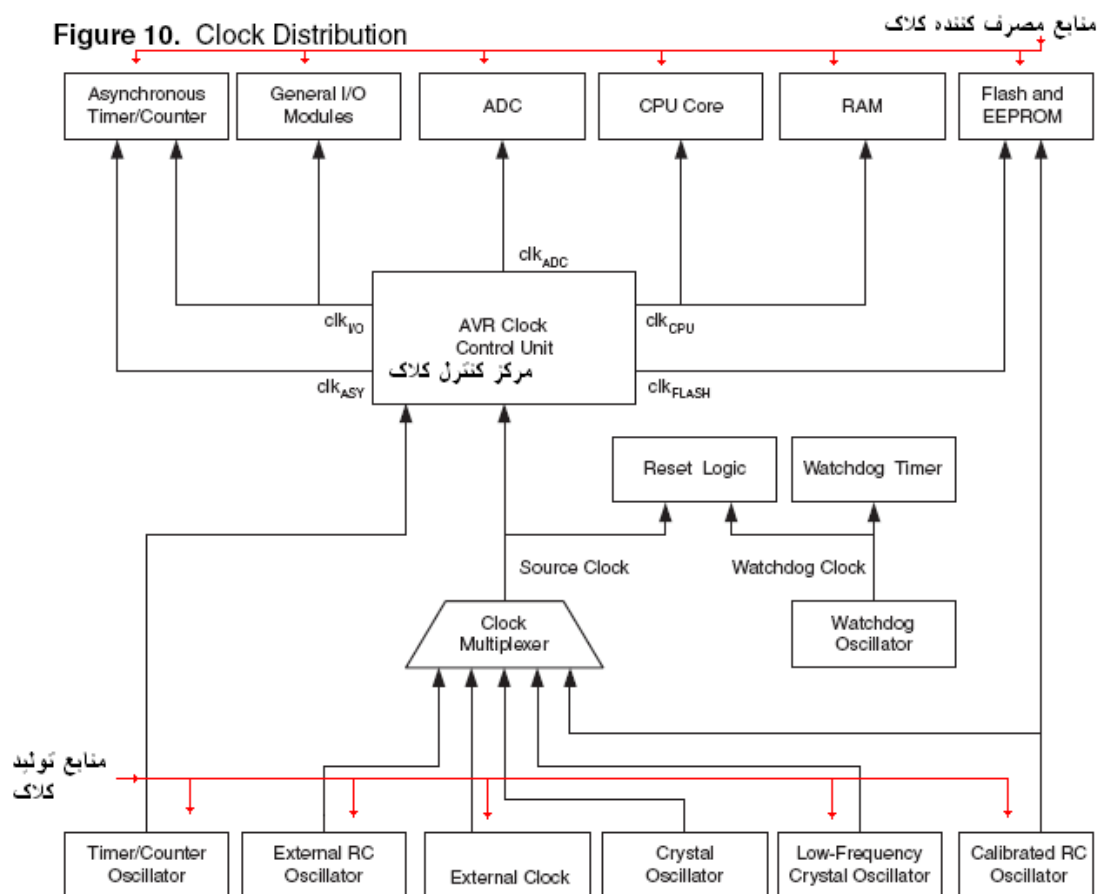
پایه شماره 39 — $\text{porta.1}/\text{adc}(1)$ این پایه علاوه بر نقش پین ورودی و خروجی a.1 ، porta.1 به عنوان ورودی مبدل آنالوگ به دیجیتال صفر ($\text{adc}(1)$) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد)



پایه شماره 40 — $\text{porta.0}/\text{adc}(0)$ این پایه علاوه بر نقش پین ورودی و خروجی a.0 ، porta.0 به عنوان ورودی مبدل آنالوگ به دیجیتال صفر ($\text{adc}(0)$) استفاده میشود (زمانی که مبدل آنالوگ به دیجیتال راه اندازی میشود از این پایه و سایر پایه های ورودی مبدل آنالوگ به دیجیتال نمی توان به عنوان ورودی یا خروجی استفاده کرد) در شکل زیر نوع بسته بندی دیگر را مشاهده می کنید کار پایه ها مشابه بسته بندی pdip می باشد و فقط جای آنها عوض شده است. پایه های هر چهار گراند از داخل به هم متصل میباشد، این موضوع در مورد vcc نیز صدق میکند

منابع تولید کلاک سیستم:

کلاک سیستم در این میکرو مطابق شکل زیر توضیح شده است:



انواع منابع تولید کلاک در avr:

Device Clocking Option	نام سخت افزار نوسان ساز	CKSEL3..0
External Crystal/Ceramic Resonator	کریستال خارجی نوسان ساز سرامیکی	1111 - 1010
External Low-frequency Crystal	نوسان ساز کریستالی فرکانس پایین	1001
External RC Oscillator	نوسان ساز rc خارجی	1000 - 0101
Calibrated Internal RC Oscillator	نوسان ساز rc داخلی میکرو	0100 - 0001
External Clock	کلاک خارجی	0000

میکرو های avr دارای چندین منبع برای تولید پالس می باشد که میتوان از هر کدام استفاده کرد.

برای استفاده از هر یک باید فیوز بیت مربوط به آن را پروگرام کرد، در تمام جداول فیوز بیت، صفر به معنای برنامه ریزی

شده و 1 به معنای عدم برنامه ریزی است

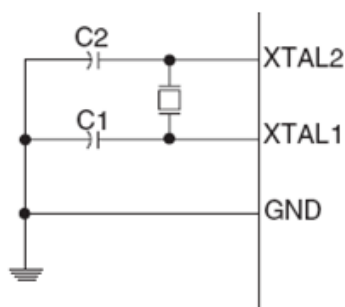
طریقه اتصال نوسان ساز به میکرو:

کریستال خارجی / نوسان ساز سرامیکی:

در این حالت کریستال کوآرتز یا نوسان ساز سرامیکی به دو پایه XTAL 1 و XTAL 2 متصل میشود

خازن های C1 و C2 برای جلوگیری از تاثیر نویز محیط بر روی نوسان ساز می باشد که مقدار آنها بستگی به مقدار نویز محیط دارد مقدار پیشنهادی این خازن ها برای فرکانس های مختلف در جدول زیر آورده شده است

Crystal Oscillator Connections



Frequency Range(MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
0.4 - 0.9	—
0.9 - 3.0	12 - 22
3.0 - 8.0	12 - 22
$1.0 \leq$	12 - 22

نوسان ساز کریستالی فرکانس پایین:

این نوع نوسان ساز که به کریستال ساعت نیز معروف است (32.768kHz) مطابق شکل بالا به دو پایه XTAL 1 و XTAL 2 متصل میشود ، برای این کریستال مقدار خازن ها 36 pf است.

نوسان ساز RC خارجی:

فرکانس این نوسان ساز از معادله $f=1/(3RC)$ بدست میاید نحوه

اتصال این نوسان ساز به میکرو در شکل روبرو آورده شده است .

کمترین مقدار خازن باید 22pf باشد تا نوسانات پایدار بماند.

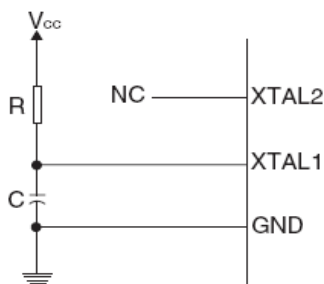
این نوع نوسان ساز می تواند در 4 مد فرکانسی کار کند که

این فرکانسها با تنظیم فیوز بیت های 0...3cksel قابل انتخاب

است در جدول زیر مد های عملیاتی نوسان ساز RC خارجی

آورده شده است.

External RC Configuration



External RC Oscillator Operating Modes

CKSEL3..0	Frequency Range (MHz)
0101	0.1 - 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0

نوسان ساز rc داخلی میکرو:

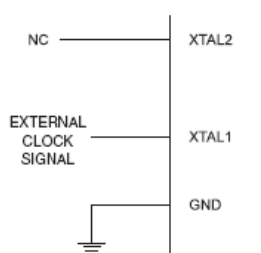
این نوسان ساز کلاک های نامی 1 و 2 و 4 و 8 مگاهرتز را در ولتاژ 5 ولت و دمای 25 درجه سانتی گراد تولید میکند در حالت عادی فیوز بیت مربوط به این نوع نوسان ساز برنامه ریزی شده است و میکرو با این نوسان ساز کار میکند (با فرکانس 1 مگاهرتز)، شما میتوانید با برنامه ریزی فیوز بیت های ckse3...0 طبق جدول زیر مقدار فرکانس را در رنج مربوطه قرار دهید

Internal Calibrated RC Oscillator Operating Modes

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

کلاک خارجی :

برای راه اندازی میکرو توسط کلاک خارجی پایه xtal 1 باید مطابق شکل زیر وصل شود در این مد ، کلاک خارجی باید دارای ثبات بالا باشد، در صورتی که فرکانس تغییر کند میکرو رفتار غیر قابل انتظاری از خود نشان میدهد



حال که با طریقه برنامه ریزی فیوز بیت های مربوط به کریستال آشنا شدید ، نکات زیر را مد نظر داشته باشید:

✓ در صورتی که فیوز بیت مربوط به یکی از نوسان ساز ها برنامه ریزی شود میکرو فقط با آن نوسان ساز ره اندازی میشود. مثلاً اگر شما فیوز بیت ckse1 را روی 0000 برنامه ریزی کنید ، میکرو فقط با کلاک خارجی راه اندازی میشود ، حتی اگر موقع کار کلاک خارجی قطع شود ، میکرو خاموش میگردد، این حالت برای پروگرام کردن میکرو نیز صادق است (بدون کلاک خارجی میکرو پروگرام نمیشود).

✓ برای اطمینان از پروگرام کردن فیوز بیت ها میتوانید کلاک را قطع کنید (نوسان ساز را از میکرو جدا کنید)، اگر میکرو به کار خود ادامه داد فیوز بیت مربوطه درست برنامه ریزی نشده است و اگر میکرو خاموش شد، فیوز بیت مربوطه درست برنامه ریزی شده است.

✓ نوسان ساز های سرامیکی در انواع مختلف ساخته میشود و نمی توان از هر خازنی به عنوان خازن نویز گیر استفاده کرد ، شما فقط میتوانید از خازن های پیشنهادی کارخانه تولید کننده استفاده کنید.

✓ در این میکرو هنگامی که از کریستال خارجی استفاده میشود ، نمیتوان از پایه های portb.6 و portb.7 به عنوان ورودی یا خروجی استفاده کرد.

فیوز بیت های دیگر این میکرو :

Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
OCDEN	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (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)

ocden : برنامه ریزی این فیوز بیت به همراه فیوز بیت jtagen به قسمت های مختلف میکرو این امکان را می دهد

تا در مد sleep کار کنند که کار کردن در این مد باعث کاهش مصرف انرژی میگردد.

jtagen: این فیوز بیت در حالت پیش فرض فعال است و به میکرو امکان برنامه ریزی از طریق واسط jtag را میدهد

توجه داشته باشید که در حالت پیش فرض پورت C برای ارتباط با واسط JTAG پیکربندی شده و با فعال بودن این فیوز امکان استفاده از این پورت به عنوان ورودی / خروجی وجود نخواهد داشت .

Spie : این فیوز بیت در حالت پیش فرض برنامه ریزی شده و می توان میکرو را از طریق ارتباط برنامه ریز کرد در

صورتی که این فیوز بیت پاک شود ، دیگر نمیتوان میکرو از طریق ارتباط isp برنامه ریزی کرد (این فیوز بیت با پروگرامر های خاص برنامه ریزی میشود).

Ckopt : اگر این فیوز بیت برنامه ریزی شود خازن های داخلی میکرو بر روی دو پایه xtal1 و xtal2 راه اندازی میشوند و دیگر نیازی به استفاده از خازن بروری دو پایه xtal1 و xtal2 نیست. (فعال کردن این فیوز بیت باعث افزایش مصرف انرژی میشود) (این بیت در حالت پیش فرض برنامه ریزی نشده است).

Eesave : اگر این فیوز بیت برنامه ریزی شود در زمان پاک کردن حافظه فلش میکرو (erase کردن فلش در هنگام پروگرام کردن) محتویات eeprom محفوظ می ماند (این بیت در حالت پیش فرض برنامه ریزی نشده است).

bootsz1: این فیوز بیت و فیوز بیت بعدی برای انتخاب میزان حافظه boot طبق جدول زیر برنامه ریزی میشوند:

Boot Size Configuration							
BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	128 words	4	0x000 - 0xF7F	0xF80 - 0xFFF	0xF7F	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xEFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xDFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xBFF	0xC00

(حافظه فلش میکرو های avr که دارای Bootloader هستند از دو بخش اصلی Application و Bootloader تشکیل شده است که برنامه های کاربردی در بخش Application ذخیره میشود و برنامه پشتیبان در حافظه Boot).

Bootsz0: این فیوز بیت و فیوز بیت قبلی برای انتخاب میزان حافظه boot طبق جدول بالا برنامه ریزی میشوند:

Bootrst: این فیوز بیت ادرس بردار ریست را تغییر میدهد. (در حالت عادی (هنگامی که این فیوز بیت برنامه ریزی نشده باشد) بعد از ریست شدن میکرو برنامه از خانه 0000 حافظه شروع به اجرا میکند، اما اگر این فیوز بیت برنامه ریزی شود بعد از ریست شدن میکرو برنامه از ادرسی که بوسیله دو فیوز بیت bootsz1 و bootsz0 تعیین شده شروع میشود

Bodlevel: در حالت عادی (هنگامی که این فیوز بیت برنامه ریزی نشده باشد) اگر ولتاژ تغذیه میکرو از 2.7 ولت پایین تر بیاید میکرو ریست میشود، اما اگر این فیوز بیت برنامه ریزی شود، هنگامی که ولتاژ تغذیه میکرو از 4 ولت کمتر شود میکرو ریست میشود (این فیوز بیت مخصوص atmega32l میباشد)

Boden: این فیوز بیت در حالت پیش فرض برنامه ریزی نشده است اما اگر برنامه ریزی شود سیستم brown-out راه اندازی میشود (این سیستم یک اشکار ساز است که در طول عملکرد میکرو سطح ولتاژ منبع تغذیه را با یک ولتاژ مرجع داخلی مقایسه میکند و در صورتیکه V_{CC} از ولتاژ مرجع بیشتر شود میکرو ریست میشود اگر این فیوز بیت به صورت 01 برنامه ریزی شود ولتاژ مرجع 2.7 ولت است و اگر به صورت 00 برنامه ریزی شود ولتاژ مرجع 4 ولت است و اگر به صورت 11 یا 10 برنامه ریزی شود غیر فعال میگردد.

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⁾

Sut1- و sut0: این دو فیوز بیت برای انتخاب زمان start-up استفاده میشود. (طبق جدول زیر)

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

زمان start-up با توجه نوع کریستال تعیین میشود (هنگامی که میکرو ریست میشود (میکرو در هنگام روشن شدن نیز ریست میشود) چند میلی ثانیه طول میکشد تا نوسانات کریستال پایدار شود بعد از اینکه نوسانات نوسان ساز پایدار شد cpu شروع به اجرای برنامه از اولین خانه حافظه میکند به مدت زمانی که طول میکشد تا نوسانات پایدار شود زمان start-up میگویند و این زمان برای انواع کریستال متفاوت میباشد)

cksel0 و cksel1 و cksel2 و cksel3: این چهار فیوز بیت مربوط به انتخاب نوسان ساز میباشد که در بالا گفته شد

دیگر خصوصیات:

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to $V_{CC}+0.5V$
Voltage on $\overline{\text{RESET}}$ with respect to Ground.....	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins.....	200.0 mA

--ویژگی :

کارای بالا و توان مصرفی کم
دارای 90 دستور که اکثر آنها در یک سیکل اجرا میشوند
32*8 رجیستر کاربردی
سرعتی تا 8mips در فرکانس 8 مگاهرتز

--حافظه ، برنامه و داده غیر فرار:

1k بایت حافظه فلش داخلی قابل برنامه ریزی
این حافظه میتواند تا 1000 بار نوشته و پاک شود (قابلیت پروگرام کردن تا 1000 بار)
64 بایت حافظه eeprom داخلی برای ذخیره اطلاعات
این حافظه میتواند تا 1000000 بار نوشته و پاک شود
قفل برنامه داخل حافظه flash و eeprom برای جلوگیری از خواندن آن

--خصوصیات جانبی:

تایمر/ کانتر 8 بیتی با prescaler مجزا و (تایمر / کانتر 0)
یک مقایسه کننده آنالوگ داخلی
Usart قابل برنامه ریزی
Watchdog قابل برنامه ریزی با اسلاتور داخلی
ارتباط سریال isp برای برنامه ریزی (پروگرام کردن) داخل مدار (هنگامی که میکرو داخل مدار است با پروگرامر isp میتوانید میکرو را برنامه ریزی کنید، برای برنامه ریزی از چهار خط miso و mosi و sck و reset استفاده میشود)
قابلیت ارتباط سریال isp به صورت master یا slave
خصوصیات ویژه میکرو:
Reset شدن میکرو بعد از روشن شدن
دارای 5 مد در حالت بیکاری برای مصرف کمتر انرژی و راندمان بیشتر
منبع وقفه داخلی و خارجی
دارای نوسان ساز داخلی کالیبره شده (حداکثر فرکانس این نوسان ساز 8 مگاهرتز است)

--انواع بسته بندی و تعداد پایه ها:

6 خط ورودی و خروجی (6 پایه در پورت b)

2 پایه مربوط به تغذیه

--حداکثر کریستال مورد استفاده

0 - 1.2 MHz برای (ATtiny12V-1)

0 - 2 MHz برای (ATtiny11L-2)

0 - 4 MHz برای (ATtiny12L-4)

0 - 6 MHz برای (ATtiny11-6)

--ولتاژ کاری

0 - 8 MHz (ATtiny12-8)

1.8 - 5.5V برای ATtiny12V-1

2.7 - 5.5V برای ATtiny11L-2 و ATtiny12L-4

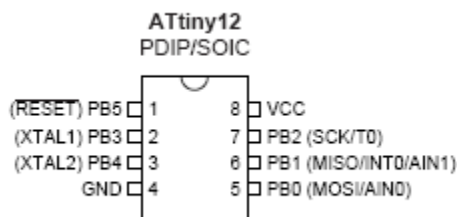
4.0 - 5.5V برای ATtiny11-6 و ATtiny12-8

شکل و شرح میکرو بسته بندی نوع pdip:

پایه شماره 1 — portb.5/reset : این پایه علاوه بر نقش پین ورودی و خروجی b.5، (portb.5) نقش پایه باز نشانی میکرو نیز دارد ، بآدادن یک پالس صفر به یک به این پایه میکرو ریست شده و برنامه از ابتدای حافظه فلش دوباره اجرا میشود.

پایه شماره 2 — portb.3/xtal1 : این پایه علاوه بر نقش پین ورودی و خروجی b.3، (portb.3)

نقش پایه ورودی کلاک کریستال را نیز به عهده دارد (منابع کریستال در ادامه آمده است).

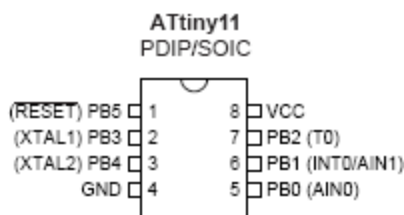


پایه شماره 3 — portb.4/xtal2 : این پایه علاوه بر نقش پین ورودی و خروجی b.4، (portb.4)

نقش پایه ورودی کلاک کریستال را نیز به عهده دارد (منابع کریستال در ادامه آمده است).

پایه شماره 4 — gnd این پایه یکی از پایه های تغذیه میکرو می باشد که باید به gnd (صفر ولت

(مدار متصل شود



پایه شماره 5 — portb.0/mosi/ain0 : این پایه علاوه بر نقش پین ورودی و خروجی b.0 ، (portb.0)

نقش ورودی مثبت مقایسه کننده آنالوگ را نیز به عهده دارد، نقش دیگر این پایه به عنوان خروج داده

مستر و ورودی داده اسلیو در پروتکل isp است.

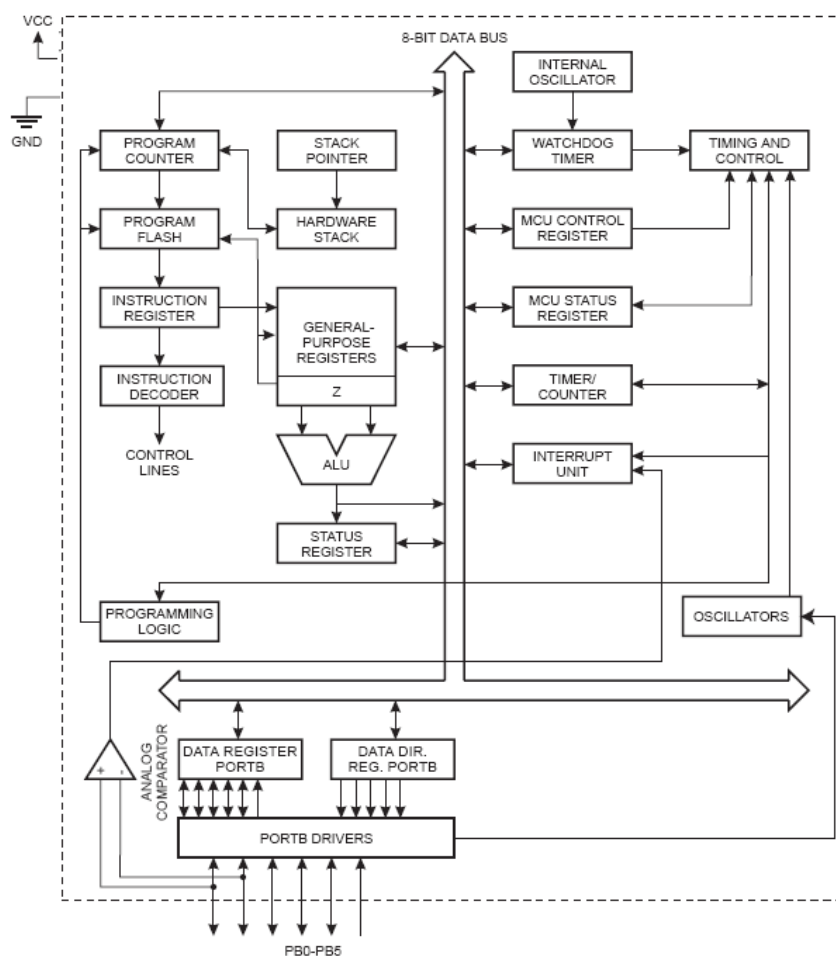
پایه شماره 6 — portb.1/MISO/INT0/ain1 این پایه علاوه بر نقش پین ورودی و خروجی b.1، (portb.1) نقش ورودی منفی مقایسه کننده آنالوگ را نیز به عهده دارد، این پایه همچنین خروجی داده اسلیو و ورودی داده مستر در پروتکل isp است. دیگر نقش این پایه ورودی منبع وقفه صفر است.

پایه شماره 7 — portb.2/sck/t0 این پایه علاوه بر نقش پین ورودی و خروجی b.2، (portb.2) زمانی که ارتباط spi راه اندازی میشود نقش پایه کلاک را به عهده دارد (بر روی پایه کلاک پالس تولید میشود که باعث هم زمانی در دستگاه های که باهم شبکه شده اند میگردد) این پایه نقش ورودی کانتر صفر را نیز به عهده دارد.

پایه شماره 8 — VCC این پایه، یکی از پایه های تغذیه میکرو می باشد که باید به VCC (5 ولت) مدار متصل شود (هر دو VCC میکرو از داخل به هم متصل میباشد)

منابع تولید کلاک سیستم

کلاک سیستم در این میکرو مطابق شکل زیر توضیح شده است:



انواع منابع تولید کلاک در avr:

Device Clocking Option	نام سخت افزار نوسان ساز	CKSEL3..0
External Crystal/Ceramic Resonator	کریستال خارجی نوسان ساز سرامیکی	1111 - 1010
External Low-frequency Crystal	نوسان ساز کریستالی فرکانس پایین	1001
External RC Oscillator	نوسان ساز rc خارجی	1000 - 0101
Calibrated Internal RC Oscillator	نوسان ساز rc داخلی میکرو	0100 - 0001
External Clock	کلاک خارجی	0000

میکرو های avr دارای چندین منبع برای تولید پالس می باشد که میتوان از هر کدام استفاده کرد.

برای استفاده از هر یک باید فیوز بیت مربوط به آن را پروگرام کرد ، در تمام جداول فیوز بیت ، صفر به معنای برنامه ریزی شده و 1 به معنای عدم برنامه ریزی است.

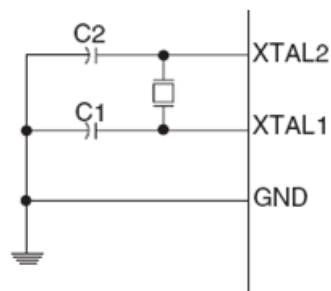
طریقه اتصال نوسان ساز به میکرو:

کریستال خارجی / نوسان ساز سرامیکی:

در این حالت کریستال کوآرتز یا نوسان ساز سرامیکی به دو پایه xtal 1 و xtal 2 متصل میشود

خازن های C_1 و C_2 برای جلوگیری از تاثیر نویز محیط بر روی نوسان ساز می باشد که مقدار آنها بستگی به مقدار نویز محیط دارد مقدار پیشنهادی این خازن ها برای فرکانس های مختلف در جدول زیر آورده شده است:

Crystal Oscillator Connections

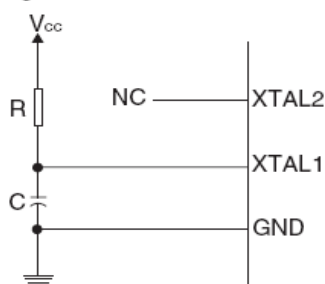


Frequency Range(MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
0.4 - 0.9	—
0.9 - 3.0	12 - 22
3.0 - 8.0	12 - 22
$1.0 \leq$	12 - 22

نوسان ساز کریستالی فرکانس پایین:

این نوع نوسان ساز که به کریستال ساعت نیز معروف است (32.768khz) مطابق شکل بالا به دو پایه xtal 1 و xtal 2 متصل میشود ، برای این کریستال مقدار خازن ها 36 pf است.

External RC Configuration



نوسان ساز rc خارجی:

فرکانس این نوسان ساز از معادله $f=1/(3RC)$ بدست میاید نحوه اتصال این نوسان ساز به میکرو در شکل روبرو آورده شده است . کمترین مقدار خازن باید 22pf باشد تا نوسانات پایدار بماند.

این نوع نوسان ساز می تواند در 4 مد فرکانسی کار کند که این فرکانسها با تنظیم فیوز بیت های CKSEL3...0 قابل انتخاب است در جدول زیر مد های عملیاتی نوسان ساز RC خارجی آورده شده است.

External RC Oscillator Operating Modes

CKSEL3..0	Frequency Range (MHz)
0101	0.1 - 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0

نوسان ساز RC داخلی میکرو:

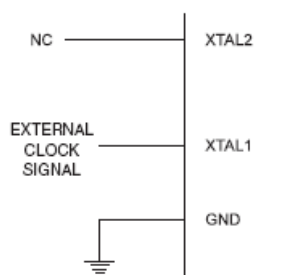
این نوسان ساز کلاک های نامی 1 و 2 و 4 و 8 مگاهرتز را در ولتاژ 5 ولت و دمای 25 درجه سانتی گراد تولید میکند در حالت عادی فیوز بیت مربوط به این نوع نوسان ساز برنامه ریزی شده است و میکرو با این نوسان ساز کار میکند (با فرکانس 1 مگا هرتز)، شما میتوانید با برنامه ریزی فیوز بیت های CKSEL3...0 طبق جدول زیر مقدار فرکانس را در رنج مربوطه قرار دهید

Internal Calibrated RC Oscillator Operating Modes

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

کلاک خارجی:

برای راه اندازی میکرو توسط کلاک خارجی پایه XTAL1 باید مطابق شکل زیر وصل شود در این مد، کلاک خارجی باید دارای ثبات بالا باشد، در صورتی که فرکانس تغییر کند میکرو رفتار غیر قابل انتظاری از خود نشان میدهد



➤ فیوز بیت های دیگر این میکرو:

ATTiny12 Clock Options and Start-up Times

CKSEL3..0	Clock Source	Start-up Time, $V_{CC} = 1.8V$, BODLEVEL Unprogrammed	Start-up Time, $V_{CC} = 2.7V$, BODLEVEL Programmed
1111	Ext. Crystal/Ceramic Resonator ⁽¹⁾	1K CK	1K CK
1110	Ext. Crystal/Ceramic Resonator ⁽¹⁾	3.6 ms + 1K CK	4.2 ms + 1K CK
1101	Ext. Crystal/Ceramic Resonator ⁽¹⁾	57 ms 1K CK	67 ms + 1K CK
1100	Ext. Crystal/Ceramic Resonator	16K CK	16K CK
1011	Ext. Crystal/Ceramic Resonator	3.6 ms + 16K CK	4.2 ms + 16K CK
1010	Ext. Crystal/Ceramic Resonator	57 ms + 16K CK	67 ms + 16K CK
1001	Ext. Low-frequency Crystal	57 ms + 1K CK	67 ms + 1K CK
1000	Ext. Low-frequency Crystal	57 ms + 32K CK	67 ms + 32K CK
0111	Ext. RC Oscillator	6 CK	6 CK
0110	Ext. RC Oscillator	3.6 ms + 6 CK	4.2 ms + 6 CK
0101	Ext. RC Oscillator	57 ms + 6 CK	67 ms + 6 CK
0100	Int. RC Oscillator	6 CK	6 CK
0011	Int. RC Oscillator	3.6 ms + 6 CK	4.2 ms + 6 CK
0010	Int. RC Oscillator	57 ms + 6 CK	67 ms + 6 CK
0001	Ext. Clock	6 CK	6 CK
0000	Ext. Clock	3.6 ms + 6 CK	4.2 ms + 6 CK

FSTRT: این فیوز بیت زمان START – UP را طبق جدول زیر مشخص میکند (هنگامی میکرو شروع به کار میکند مدت زمان کوتاهی طول میکشد تا نوسانات کریستال پایدار شود ، شما باید با توجه به نوع کریستال و زمان پایداری آن ، طبق جدول این فیوز بیت را برنامه ریزی کنید)

Start-up Times for the ATTiny11 ($V_{CC} = 2.7V$)

Selected Clock Option	Start-up Time t_{OUT}	
	FSTRT Unprogrammed	FSTRT Programmed
External Crystal	67 ms	4.2 ms
External Ceramic Resonator	67 ms	4.2 ms
External Low-frequency Crystal	4.2 s	4.2 s
External RC Oscillator	4.2 ms	67 μ s
Internal RC Oscillator	4.2 ms	67 μ s
External Clock	4.2 ms	5 clocks from reset, 2 clocks from power-down

12 – RSTDISBL: با برنامه ریزی این فیوز بیت میتوان از پایه RESET (پورت B.5) به عنوان ورودی و خروجی استفاده کرد (

با این کار میکرو را نمیتوان از طریق واسطه ISP پروگرام کرد)

13 – Spien: (مخصوص ATTINY12) این فیوز بیت در حالت پیش فرض برنامه ریزی شده و می توان میکرو را از طریق

ارتباط isp برنامه ریز کرد در صورتی که این فیوز بیت پاک شود ،دیگر نمیتوان میکرو از طریق ارتباط isp برنامه

ریزی کرد (این فیوز بیت با پروگرامر های خاص برنامه ریزی میشود.)

14- Bodlevel : (مخصوص ATTINY12) در حالت عادی (هنگامی که این فیوز بیت برنامه ریزی نشده باشد) اگر ولتاژ

تغذیه میکرو از 2.7 ولت پایین تر بیاید میکرو ریست میشود، اما اگر این فیوز بیت برنامه ریزی شود، هنگامی که ولتاژ

تغذیه میکرو از 4 ولت کمتر شود میکرو ریست میشود (این فیوز بیت مخصوص نوع L و V میباشد)

15- Boden : (مخصوص ATTINY12) این فیوز بیت در حالت پیش فرض برنامه ریزی نشده است اما اگر برنامه ریزی شود

سیستم brown-out راه اندازی میشود (این سیستم یک اشکار ساز است که در طول عملکرد میکرو سطح ولتاژ منبع

تغذیه را با یک ولتاژ مرجع داخلی مقایسه میکند و در صورتیکه V_{CC} از ولتاژ مرجع بیشتر شود میکرو ریست میشود اگر

این فیوز بیت به صورت 01 برنامه ریزی شود ولتاژ مرجع 2.7 ولت است و اگر به صورت 00 برنامه ریزی شود ولتاژ

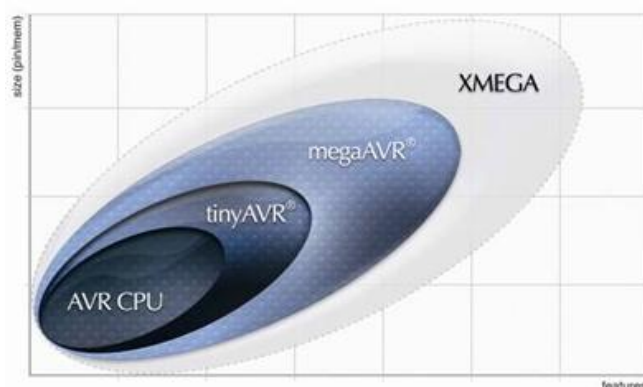
مرجع 4 ولت است و اگر به صورت 11 یا 10 برنامه ریزی شود غیر فعال میگردد

4-5-6-7-cksel0 وcksel1 وcksel2 وcksel3 : این چهار فیوز بیت مربوط به انتخاب نوسان ساز میباشد (انتخاب نوع

نوسان ساز در بالا گفته شد)

دیتاشیت فارسی میکرو کنترلر ATXMEGA128A1 :

این میکرو کنترلر ها که دارای قدرت پردازش 16/8 بیتی میباشند و تعداد i/o آنها گاهها" تا 100 عدد میرسد ، قرار است در پروسه های اندازه گیری ، کنترل و انتقال داده به کار روند .



این میکرو کنترلر ها تقریباً به روش میکرو کنترلر های قبلی (سری های mega و at91s و...) ساخته میشوند و شرکت اتمل برای بهبود کارایی موارد زیر را به آنها افزوده است :

- ✓ کاهش توان مصرفی .
- ✓ بالا بردن کلاک و سرعت cpu
- ✓ افزودن 4 کانال DMA
- ✓ تنظیمات 100٪ قابل پیش بینی (عمل کرد ثابت)
- ✓ دارای ADC و DAC های 12 بیتی
- ✓ دارای امکانات مختلف برای پنهان کردن داده ها (DES و AES Data Encryption Standard و Address Encryption Standard)
- ✓ حذف فیوز بیت های مربوط به منبع کلاک و افزودن واحد اسیلاتور و PLL

خانواده ی XMEGA دارای ویژگی های بیشتری نسبت به سایر میکرو کنترلر های avr است ، این دسته برای فعالیت در فرکانس 32 مگاهرتز فقط به ولتاژ تغذیه ی 1.6 ولت نیاز دارند ، اندازه ی حافظه ی فلش در آنها بین 16 تا 384 کیلو بایت است و در بسته بندی های 44 تا 100 پایه تولید میشوند . خانواده ی xmega عملکرد کاملاً ثابتی دارند و از آنها میتوان در کاربرد های همچون سیستم های صوتی ، دستگاه های پزشکی ، منابع تغذیه ، برد های کنترل کننده ، شبکه های انتقال داده ، کنترل دور موتور ، فرستنده و گیرنده های نوری و هر چیزی که به یک کنترل کننده ی پایدار و کم مصرف نیاز داشته باشد استفاده کرد .

برای آشنایی بیشتر شما با این خانواده ، در ادامه دیتا شیت فارسی ATxmega192A1 ، ATxmega256A1 ، ATxmega384A1 ، ATxmega64A1 ، ATxmega128A1 آورده شده است ، در این دیتا شیت ویژگی ها ، نوع بسته بندی و چیدمان و وظیفه ی پایه های میکرو کنترلر موجود میباشد .

ویژگی :

کارای بالا و توان مصرفی کم

64 تا 384 کیلو بایت بایت حافظه فلش داخلی قابل برنامه ریزی

این حافظه میتواند تا 10000 بار نوشته و پاک شود (قابلیت پروگرام کردن تا 10000 بار)

4 تا 8 کیلو بایت حافظه ی boot با قابلیت قفل شدن (Lock Bits)

2 تا 4 کیلو بایت حافظه eeprom داخلی برای ذخیره اطلاعات

4 تا 32 کیلو بایت حافظه sram داخلی

دارای باس مجزا برای اتصال sram خارجی تا بالای 16 مگا بایت

دارای باس مجزای برای اتصال sdram خارجی تا بالای 128 مگا بیت

قفل برنامه داخل حافظه flash و eeprom برای جلوگیری از خواندن آن

خصوصیات جانبی:

چهار کانال DMA با قابلیت پشتیبانی از درخواست خارجی

8 کانال Event System

8 تایمر / کانتر 16 بیتی

چهار کانال تایمر/کانتر با 4 عدد خروجی مقایسه ای یا ورودی تحریک پذیر

چهار کانال تایمر/کانتر با 2 عدد خروجی مقایسه ای یا ورودی تحریک پذیر

دقت بسیار بالا در تمامی تایمر/کانترها

تولید موج پیشرفته در دو تایمر/کانتر

چهار کانال pwm

2*8 کانال مبدل آنالوگ به دیجیتال 12 بیتی

2*2 کانال مبدل دیجیتال به آنالوگ 12 بیتی

دارای rtc شانزده بیتی با اسیلاتور مجزا

4 مقایسه کننده آنالوگ داخلی

8 کانال Usart قابل برنامه ریزی

پشتیبانی از مدولاسیون/دمدولاسیون IrDA در یکی از کانال های usart

Watchdog قابل برنامه ریزی با اسیلاتور داخلی

5 کانال پورت i2c با ادرس دابل (I2C and SMBus compatible)

5 کانال رابط سریال SPI به صورت master یا slave

قابلیت ارتباط jtag (یک نوع ارتباط است که از طریق آن می توان کلیه حافظه های قابل برنامه ریزی میکرو را خواند یا نوشت)

منبع وقفه داخلی و خارجی ، وجود وقفه ی خارجی بر روی تمامی پایه ها
خصوصیات ویژه ی میکرو کنترلر

Reset شدن میکرو بعد از روشن شدن

دارای منابع کلاک داخلی و خارجی به همراه pll

دارای 5 مد در حالت بیکاری برای مصرف کمتر انرژی و راندمان بیشتر

برنامه ریزی و دیباگ پیشرفته :

JTAG (IEEE 1149.1 Compliant) برای برنامه ریزی و دیباگ کردن

PDI (Program and Debug Interface) برای برنامه ریزی و دیباگ کردن

تعداد ورودی/خروجی در هر بسته بندی :

➤ 78 Programmable I/O Lines

➤ 100 - lead TQFP

➤ 100 - ball CBGA

➤ 100 - ball VFBGA

ولتاژ کاری :

➤ 1.6 تا 3.6 ولت

فرکانس کاری :

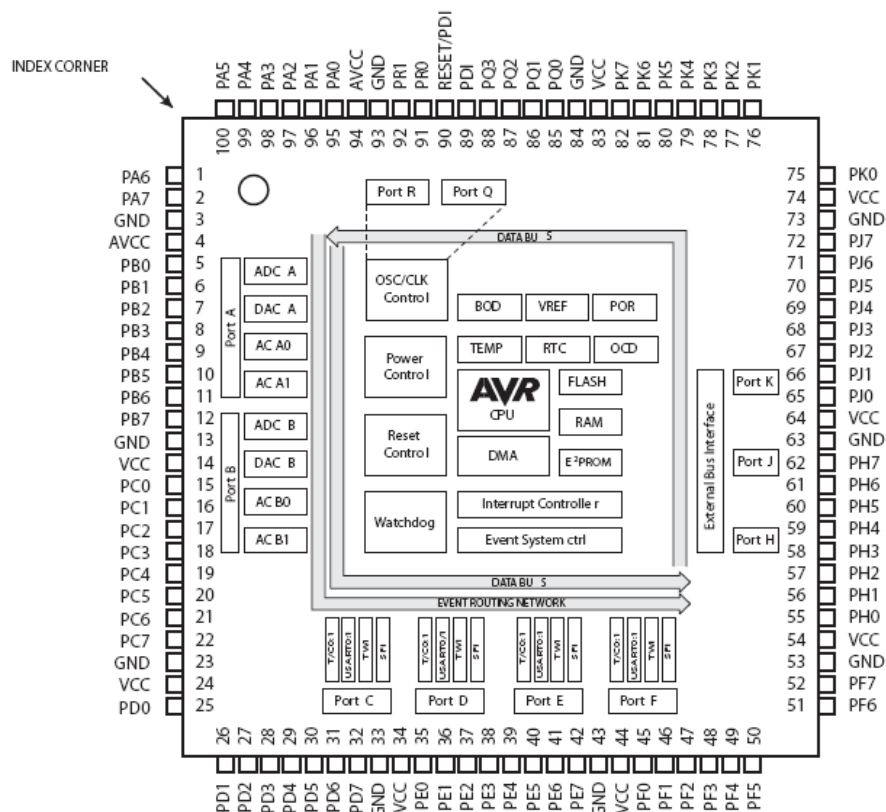
➤ 0 – 12 MHz @ 1.6 – 3.6V

➤ 0 – 32 MHz @ 2.7 – 3.6V

سایر اطلاعات :

بسته بندی :

Ordering Code	Flash (B)	E ²	SRAM	Speed (MHz)	Power Supply	Package ⁽¹⁾⁽²⁾⁽³⁾	Temp
ATxmega384A1-AU	384K + 8K	4 KB	32 KB	32	1.6 - 3.6V	100A	-40°C - 85°C
ATxmega256A1-AU	256K + 8K	4 KB	16 KB	32	1.6 - 3.6V		
ATxmega192A1-AU	192K + 8K	2 KB	16 KB	32	1.6 - 3.6V		
ATxmega128A1-AU	128K + 8K	2 KB	8 KB	32	1.6 - 3.6V		
ATxmega64A1-AU	64K + 4K	2 KB	4 KB	32	1.6 - 3.6V		
ATxmega384A1-CU	384K + 8K	4 KB	32 KB	32	1.6 - 3.6V	100C1	
ATxmega256A1-CU	256K + 8K	4 KB	16 KB	32	1.6 - 3.6V		
ATxmega192A1-CU	192K + 8K	2 KB	16 KB	32	1.6 - 3.6V		
ATxmega128A1-CU	128K + 8K	2 KB	8 KB	32	1.6 - 3.6V		
ATxmega64A1-CU	64K + 4K	2 KB	4 KB	32	1.6 - 3.6V		
ATxmega128A1-C7U	128K + 8K	2 KB	8 KB	32	1.6 - 3.6V	100C2	
ATxmega64A1-C7U	64K + 4K	2 KB	4 KB	32	1.6 - 3.6V		



نقش پایه ها :

میکرو کنترلر های معرفی شده ، دارای بسته بندی TQFP یا BGA و دارای 100 پایه میباشند ، در جدول زیر کاربرد کلیه پایه های میکرو کنترلر آورده شده است ، شما میتوانید موقعیت هر پایه را در تصویر بالا پیدا کنید :

توضیح	کاربرد	نام پایه	شماره ی پین
گراند میکرو کنترلر	Ground	GND	93
ورودی ولتاژ تغذیه ی adc و dac	Analog supply voltage	AVCC	94
ورودی - خروجی همزمان پورت A.0 / ورودی مبدل آنالوگ به دیجیتال شماره ی 0 / ورودی مقایسه کننده ی شماره ی 0 / ورودی ولتاژ مرجع .	SYNC/ ADC0 /AC0 /AREF	PA0	95
ورودی - خروجی همزمان پورت A.1 / ورودی مبدل آنالوگ به دیجیتال شماره ی 1 / ورودی مقایسه کننده ی شماره ی 1	SYNC/ ADC1/AC1	PA1	96
ورودی همزمان - غیر همزمان پورت A.2 / ورودی مبدل آنالوگ به دیجیتال شماره ی 2 / ورودی مقایسه کننده ی شماره ی 2 / خروجی مبدل دیجیتال به آنالوگ شماره ی 0	SYNC/ASYN/ ADC2/AC2/ DAC0	PA2	97
ورودی - خروجی همزمان پورت A.3 / ورودی مبدل آنالوگ به دیجیتال شماره ی 3 / ورودی مقایسه کننده ی شماره ی 3 / خروجی مبدل دیجیتال به آنالوگ شماره ی 1	SYNC/ ADC3/AC3 /DAC1	PA3	98
ورودی - خروجی همزمان پورت A.4 / ورودی مبدل آنالوگ به دیجیتال شماره ی 4 / ورودی مقایسه کننده ی شماره ی 4	SYNC /ADC4/AC4	PA4	99
ورودی - خروجی همزمان پورت A.5 / ورودی مبدل آنالوگ به دیجیتال شماره ی 5 / ورودی مقایسه کننده ی شماره ی 5	SYNC /ADC5/AC5	PA5	100
ورودی - خروجی همزمان پورت A.6 / ورودی مبدل آنالوگ به دیجیتال شماره ی 6 / ورودی مقایسه کننده ی شماره ی 6	SYNC /ADC6/AC6	PA6	1
ورودی - خروجی همزمان پورت A.7 / ورودی مبدل آنالوگ به دیجیتال شماره ی 7 / ورودی مقایسه کننده ی شماره ی 7 / خروجی مقایسه کننده ی آنالوگ شماره ی 0	SYNC/ ADC7/AC7 /AC0OUT	PA7	2
گراند	Ground	GND	3
ورودی - خروجی ولتاژ تغذیه ی adc و dac	Analog supply voltage	AVCC	4
ورودی - خروجی همزمان پورت B.0 / ورودی مبدل آنالوگ به دیجیتال شماره ی 0 / ورودی مقایسه کننده ی شماره ی 0 / ورودی ولتاژ مرجع .	SYNC/ ADC0/ AC0/ AREF	PB0	5
ورودی - خروجی همزمان پورت B.1 / ورودی مبدل آنالوگ به دیجیتال شماره ی 1 / ورودی مقایسه کننده ی شماره ی 1	SYNC / ADC1/ AC1	PB1	6
ورودی - خروجی همزمان - غیر همزمان پورت B.2 / ورودی مبدل آنالوگ به دیجیتال شماره ی 2 / ورودی مقایسه کننده ی شماره ی 2 / خروجی مبدل دیجیتال به آنالوگ شماره ی 0	SYNC-ASYN/ ADC2/ AC2/ DAC0	PB2	7
ورودی - خروجی همزمان پورت B.3 / ورودی مبدل آنالوگ به دیجیتال شماره ی 3 / ورودی مقایسه کننده ی شماره ی 3 / خروجی مبدل دیجیتال به آنالوگ شماره ی 1	SYNC/ ADC3 /AC3/DAC1	PB3	8
ورودی همزمان پورت B.4 / ورودی مبدل آنالوگ به دیجیتال شماره ی 4 / ورودی مقایسه کننده ی شماره ی 4 / پایه Test Mode Select در مد JTAG	SYNC/ ADC4/ AC4/ TMS	PB4	9
ورودی همزمان پورت B.5 / ورودی مبدل آنالوگ به دیجیتال شماره ی 5 / ورودی مقایسه کننده ی شماره ی 5 / پایه Test Data In در مد JTAG	SYNC / ADC5/ AC5/ TDI	PB5	10

11	PB6	SYNC /ADC6 / AC6/ TCK	ورودی - خروجی همزمان پورت B.6 / ورودی مبدل آنالوگ به دیجیتال شماره ی 6 / ورودی مقایسه کننده ی شماره ی 6 / پایه Test Clock در مد JTAG
12	PB7	SYNC / ADC7/ AC7/ AC0OUT /TDO	ورودی - خروجی همزمان پورت B.7 / ورودی مبدل آنالوگ به دیجیتال شماره ی 7 / ورودی مقایسه کننده ی شماره ی 7 / خروجی مقایسه کننده ی آنالوگ شماره ی 0 / پایه Test Data Out در مد JTAG
13	GND	Ground	گراند
14	VCC	Digital supply voltage	ورودی ولتاژ تغذیه ی میکرو کنترلر
15	PC0	SYNC /OC0A /OC0ALS/ SDA	ورودی - خروجی همزمان پورت C.0 / خروجی مقایسه ای شماره ی 1 تایمر-کانتر 0 / خروجی مقایسه ای حالت Low شماره ی 1 تایمر-کانتر 0 / پایه ی Serial Data در پروتکل I2C
16	PC1	SYNC/ OC0B/ OC0AHS/ XCK0/ SCL	ورودی - خروجی همزمان پورت C.1 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 0 / خروجی مقایسه ای حالت High شماره ی 1 تایمر-کانتر 0 / انتقال کلاک واحد USART شماره ی 0 / پایه ی Serial Clock در پروتکل I2C
17	PC2	SYNC-ASYNC /OC0C /OC0BLS/ RXD0	ورودی - خروجی همزمان - غیر همزمان پورت PC.2 / خروجی مقایسه ای شماره ی 3 تایمر-کانتر 0 / خروجی مقایسه ای حالت Low شماره ی 2 تایمر-کانتر 0 / پایه ی Receiver Data واحد USART0
18	PC3	SYNC/ OC0D/ OC0BHS/ TXD0	ورودی - خروجی همزمان پورت PC.3 / خروجی مقایسه ای حالت Low شماره ی 3 تایمر-کانتر 0 / خروجی مقایسه ای شماره ی 4 تایمر-کانتر 0 / پایه ی Transmitter Data واحد USART0
19	PC4	SYNC/OC0CLS /OC1A /SS	ورودی - خروجی همزمان پورت C.4 / خروجی مقایسه ای شماره ی 1 تایمر-کانتر 1 / پایه ی انتخاب اسلیو (CHIP SELECT) واحد SPI
20	PC5	SYNC/ OC0CHS /OC1B/ XCK1 /MOSI	ورودی - خروجی همزمان پورت C.5 / خروجی مقایسه ای حالت High شماره ی 3 تایمر-کانتر 0 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 1 / انتقال کلاک واحد USART شماره ی 1 / پایه ی Master In Slave Out واحد SPI
21	PC6	SYNC /OC0DLS /RXD1/ MISO	ورودی - خروجی همزمان پورت C.6 / خروجی مقایسه ای حالت Low شماره ی 4 تایمر-کانتر 0 / پایه ی Receiver Data واحد USART1 / پایه ی Master In Slave Out واحد SPI
22	PC7	SYNC/ OC0DHS /TXD1/ SCK /CLKOUT/ EVOUT	ورودی - خروجی همزمان پورت C.7 / خروجی مقایسه ای حالت High شماره ی 4 تایمر-کانتر 0 / پایه ی Transmitter Data واحد USART1 / پایه ی Serial Clock واحد SPI / خروجی پالس کلاک برای وسیله ی جانبی / خروجی رویداد شماره ی 0
23	GND	Ground	گراند
24	VCC	Digital supply voltage	ورودی ولتاژ تغذیه ی میکرو کنترلر
25	PD0	SYNC /OC0A/ SDA	ورودی - خروجی همزمان پورت D.0 / خروجی مقایسه ای شماره ی 1 تایمر-کانتر 0 / پایه ی Serial Data در پروتکل I2C
26	PD1	SYNC/ OC0B /XCK0/ SCL	ورودی - خروجی همزمان پورت D.1 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 0 / انتقال کلاک واحد USART شماره ی 0 / پایه ی Serial Clock در پروتکل I2C
27	PD2	SYNC-ASYNC/ OC0C/ RXD0	ورودی - خروجی همزمان-غیر همزمان پورت D.2 / خروجی مقایسه ای شماره ی 3 تایمر-کانتر 0 / پایه ی Receiver Data واحد USART0
28	PD3	SYNC /OC0D /TXD0	ورودی - خروجی همزمان پورت D.3 / خروجی مقایسه ای شماره ی 4 تایمر-کانتر 0 / پایه ی Transmitter Data واحد USART0
29	PD4	SYNC/ OC1A /SS	ورودی - خروجی همزمان پورت D.4 / خروجی مقایسه ای شماره ی 1 تایمر-کانتر 1 / پایه ی انتخاب اسلیو (CHIP SELECT) واحد SPI
30	PD5	SYNC/ OC1B/ XCK1/ MOSI	ورودی - خروجی همزمان پورت D.5 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 1 / انتقال کلاک واحد USART شماره ی 1 / پایه ی Master In Slave Out واحد SPI
31	PD6	SYNC/ RXD1/ MISO	ورودی - خروجی همزمان پورت D.6 / پایه ی Receiver Data واحد USART1 / پایه ی Master In Slave Out واحد SPI

32	PD7	SYNC/ TXD1/ SCK/ CLKOUT/ EVOUT	ورودی - خروجی همزمان پورت D.7 / پایه ی Transmitter Data واحد USART1 / پایه ی Serial Clock واحد SPI / خروجی پالس کلاک برای وسیله ی جانبی / خروجی رویداد شماره ی 0
33	GND	Ground	گراوند
34	VCC	Digital supply voltage	ورودی ولتاژ تغذیه ی میکرو کنترلر
35	PE0	SYNC/ OC0A/ OC0ALS/ SDA	ورودی - خروجی همزمان پورت E.0 / خروجی مقایسه ای شماره ی 1 تایمر-کانتر 0 / خروجی مقایسه ای حالت Low شماره ی 1 تایمر-کانتر 0 / پایه ی Serial Data در پروتکل I2C
36	PE1	SYNC/ OC0B/ OC0AHS/ XCK0/ SCL	ورودی - خروجی همزمان پورت E.1 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 0 / خروجی مقایسه ای حالت High شماره ی 1 تایمر-کانتر 0 / انتقال کلاک واحد USART / پایه ی Serial Clock در پروتکل I2C
37	PE2	SYNC-ASYNC/ OC0C/ OC0BLS/ RXD0	ورودی - خروجی همزمان پورت E.2 / خروجی مقایسه ای شماره ی 3 تایمر-کانتر 0 / خروجی مقایسه ای حالت Low شماره ی 2 تایمر-کانتر 0 / پایه ی Receiver Data واحد USART0
38	PE3	SYNC/ OC0D/ OC0BHS/ TXD0	ورودی - خروجی همزمان پورت E.3 / خروجی مقایسه ای شماره ی 4 تایمر-کانتر 0 / خروجی مقایسه ای حالت High شماره ی 2 تایمر-کانتر 0 / پایه ی Transmitter Data واحد USART0
39	PE4	SYNC/ OC0CLS/ OC1A/ SS	ورودی - خروجی همزمان پورت E.4 / خروجی مقایسه ای حالت Low شماره ی 3 تایمر-کانتر 0 / خروجی مقایسه ای شماره ی 1 تایمر-کانتر 1 / پایه ی انتخاب اسلیو (CHIP SELECT) واحد SPI
40	PE5	SYNC/ OC0CHS/ OC1B/ XCK1/ MOSI	ورودی - خروجی همزمان پورت E.5 / خروجی مقایسه ای حالت High شماره ی 3 تایمر-کانتر 0 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 1 / انتقال کلاک واحد USART شماره ی 1 / پایه ی Master Out Slave In واحد SPI
41	PE6	SYNC/ OC0DLS/ RXD1/ MISO	ورودی - خروجی همزمان پورت E.6 / خروجی مقایسه ای حالت Low شماره ی 4 تایمر-کانتر 0 / پایه ی Receiver Data واحد USART1 / پایه ی Master In Slave Out واحد SPI
42	PE7	SYNC/ OC0DHS/ TXD1 /SCK /CLKOUT /EVOUT	ورودی - خروجی همزمان پورت E.7 / خروجی مقایسه ای حالت High شماره ی 4 تایمر-کانتر 0 / پایه ی Transmitter Data واحد USART1 / پایه ی Serial Clock واحد SPI / خروجی پالس کلاک برای وسیله ی جانبی / خروجی رویداد شماره ی 0
43	GND	Ground	گراوند
44	VCC	Digital supply voltage	ورودی ولتاژ تغذیه ی میکرو کنترلر
45	PF0	SYNC/ OC0B /XCK0/ SCL	ورودی - خروجی همزمان پورت F.0 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 0 / انتقال کلاک واحد USART0 / پایه ی Serial Clock در پروتکل I2C
46	PF1	SYNC/ OC0B /XCK0	ورودی - خروجی همزمان پورت F.1 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 0 / انتقال کلاک واحد USART شماره ی 0 /
47	PF2	SYNC-ASYNC/ OC0C /RXD0	ورودی - خروجی همزمان غیر همزمان پورت F.2 / خروجی مقایسه ای شماره ی 3 تایمر-کانتر 0 / پایه ی Receiver Data واحد USART0
48	PF3	SYNC/ OC0D /TXD0	ورودی - خروجی همزمان پورت F.3 / خروجی مقایسه ای شماره ی 4 تایمر-کانتر 0 / پایه ی Transmitter Data واحد USART0
49	PF4	SYNC/ OC1A /SS	ورودی - خروجی همزمان پورت F.4 / خروجی مقایسه ای شماره ی 1 تایمر-کانتر 1 / پایه ی انتخاب اسلیو (CHIP SELECT) واحد SPI
50	PF5	SYNC/ OC1B /XCK1 /MOSI	ورودی - خروجی همزمان پورت F.5 / خروجی مقایسه ای شماره ی 2 تایمر-کانتر 1 / انتقال کلاک واحد USART شماره ی 1 / پایه ی Master Out Slave In واحد SPI
51	PF6	SYNC/ RXD1/ MISO	ورودی - خروجی همزمان پورت F.6 / پایه ی Receiver Data واحد USART1 / پایه ی Master In Slave Out واحد SPI
52	PF7	SYNC/ TXD1 /SCK	ورودی - خروجی همزمان پورت F.7 / پایه ی Transmitter Data واحد USART1 / پایه ی Serial Clock واحد SPI
53	GND	Ground	گراوند

54	VCC	Digital supply voltage	ورودی ولتاژ تغذیه ی میکرو کنترلر
55	PH0	SYNC / WE	ورودی - خروجی همزمان پورت H.0 / External Data Memory Write
56	PH1	SYNC/ CAS/ RE	ورودی - خروجی همزمان پورت H.1 / Read Enable / Column Access Strobe
57	PH2	SYNC-ASYNC/ RAS/ ALE1	ورودی - خروجی همزمان- غیر همزمان پورت H.2 / Address Latch Enable 1 / Row Access Strobe
58	PH3	SYNC /DQM/ ALE2	ورودی - خروجی همزمان پورت H.3 / Address Latch Enable 2/ Data Mask Signal/Output Enable
59	PH4	SYNC /BA0 / CS0/ CS0-A16	ورودی - خروجی همزمان پورت H.4 / Address line 16 / Chip Select 0 / Bank Address 0
60	PH5	SYNC /BA1/ CS1-A17 /CS1	ورودی - خروجی همزمان پورت H.5 / Address line 16 / Chip Select 1- Address line 17 / Bank Address 1
61	PH6	SYNC/ CKE/ CS2-A18/ CS2	ورودی - خروجی همزمان پورت H.6 / Chip / Chip Select 2- Address line 18 / SDRAM Clock Enable / Select 2
62	PH7	SYNC/ CLK /CS3-A19 /CS3	ورودی - خروجی همزمان پورت H.7 / Chip Select 3 / Chip Select 3- Address line 19 // SDRAM Clock
63	GND	Ground	گراند
64	VCC	Digital supply voltage	ورودی ولتاژ تغذیه ی میکرو کنترلر
65	PJ0	SYNC /D0 /D0-A0/ D0-A0-A8	ورودی - خروجی همزمان پورت J.0 / خط داده 0 / خط داده و آدرس 0 - آدرس 8
66	PJ1	SYNC /D1 /D1-A1 /D1-A1-A9	ورودی - خروجی همزمان پورت J.1 / خط داده 1 / خط داده و آدرس 1 - آدرس 9
67	PJ2	SYNC-ASYNC /D2/ D2-A2 /D2-A2-A10	ورودی - خروجی همزمان- غیر همزمان پورت J.2 / خط داده 2 / خط داده و آدرس 2 - آدرس 10
68	PJ3	SYNC /D3 /D3-A3 / D3-A3-A11	ورودی - خروجی همزمان پورت J.3 / خط داده 3 / خط داده و آدرس 3 - آدرس 11
69	PJ4	SYNC /A8/ D4/D4-A4 /D4-A4-A12	ورودی - خروجی همزمان پورت J.4 / خط داده 4 / خط داده و آدرس 4 - آدرس 12
70	PJ5	SYNC/ A9/D5/ D5-A5/ D5-A5-A13	ورودی - خروجی همزمان پورت J.5 / خط آدرس 9 / خط داده 5 / خط داده و آدرس 5 - آدرس 13
71	PJ6	SYNC/ A10/D6 /D6-A6/D6-A6-A14	ورودی - خروجی همزمان پورت J.6 / خط آدرس 10 / خط داده 6 / خط داده و آدرس 6 - آدرس 14
72	PJ7	SYNC/ A11/ D7 /D7-A7 /D7-A7-A15	ورودی - خروجی همزمان پورت J.7 // خط آدرس 11 / خط داده 7 / خط داده و آدرس 7 - آدرس 15
73	GND	Ground	گراند
74	VCC	Digital supply voltage	ورودی ولتاژ تغذیه ی میکرو کنترلر
75	PK0	SYNC/ A0/A0-A8/A0-A8-A16/A8	ورودی - خروجی همزمان پورت K.0 / خط آدرس 0 / خط آدرس 0 - آدرس 8 / خط آدرس 8
76	PK1	SYNC/ A1/A1-A9/A1-A9-A17/A9	ورودی - خروجی همزمان پورت K.1 / خط آدرس 1 / خط آدرس 1 - آدرس 9 / خط آدرس 9
77	PK2	SYNC-ASYNC/ A2/A2-A10/A2-A10-A18/ A10	ورودی - خروجی همزمان- غیر همزمان پورت K.2 / خط آدرس 2 / خط آدرس 2 - آدرس 10 / خط آدرس 10
78	PK3	SYNC/ A3/A3-A11/A3-A11-A19/A11	ورودی - خروجی همزمان پورت K.3 / خط آدرس 3 / خط آدرس 3 - آدرس 11 / خط آدرس 11
79	PK4	SYNC /A4/A4-A12/A4-A12-A20/A12	ورودی - خروجی همزمان پورت K.4 / خط آدرس 4 / خط آدرس 4 - آدرس 12 / خط آدرس 12
80	PK5	SYNC/ A5/A5-A13/A5-A13-A21/A13	ورودی - خروجی همزمان پورت K.5 / خط آدرس 5 / خط آدرس 5 - آدرس 13 / خط آدرس 13

81	PK6	SYNC/ A6/A6-A14/A6-A14-A22/A14	ورودی - خروجی همزمان پورت K.6 / خط آدرس A6 / خط آدرس A6-A14 / خط آدرس A6-A14-A22 / خط آدرس A14
82	PK7	SYNC/ A7/A7-A15/A7-A15-A23/A15	ورودی - خروجی همزمان پورت K.7 / خط آدرس A7 / خط آدرس A7-A15 / خط آدرس A7-A15-A23 / خط آدرس A15
83	GND	Ground	گراند
84	VCC	Digital supply voltage	ورودی ولتاژ تغذیه ی میکرو کنترلر
85	PQ0	SYNC / TOSC1	ورودی - خروجی همزمان پورت Q.0 / اتصال به کریستال ساعت خارجی
86	PQ1	SYNC / TOSC2	ورودی - خروجی همزمان پورت Q.1 / اتصال به کریستال ساعت خارجی
87	PQ2	SYNC-ASYNC	ورودی - خروجی همزمان-غیرهمزمان پورت Q.2
88	PQ3	SYNC	ورودی - خروجی همزمان پورت Q.3
89	PDI	PDI_DATA	خط انتقال داده پرورگرامر PDI
90	RESET	PDI_CLOCK	ورودی ریست / اتصال به خط کلاک پرورگرامر PDI
91	PRO	SYNC / XTAL2	ورودی - خروجی همزمان پورت R.0 / اتصال به کریستال خارجی
92	PR1	SYNC / XTAL1	ورودی - خروجی همزمان پورت R.1 / اتصال به کریستال خارجی

منابع و مآخذ:

کتاب میکرو کنترلر های AVR نوشته مهندس علی کاهه

راه اندازی موتور های DC و پله ای توسط AVR به همراه درایو هر یک از آنها نوشته آقای حمید بادامی نژاد

آموزش سریع میکرو کنترلر های avr نوشته ی رضا سپاس یار

مطالب موجود در سایت های :

www.eca.ir

www.iranmicro.ir

www.rfid.persianblog.ir

....

HELP نرم افزار بسکام