# به نام خدا

# آموزش برنامه نویسی میکروکنترلر های AVR به زبان بیسیک با استفاده از کامپایلر BASCOM

# سایت تخصصی مهندسی رباتیک

# WWW.ROBOTICS-ENGINEERING.IR

# محیط برنامه نویسی
# BASCOM
4

# معرفی منوهای محیط BASCOM

میکروکنترلرهای AVR

# منوی FILE

- **ایجاد فایل جدید (FILE NEW)**
  با انتخاب این گزینه یک پنجره جدید که شما قادر به نوشتن برنامه در آن هستید ایجاد می شود .
- **باز کردن فایل ( OPEN FILE )**
  با انتخاب این گزینه شما قادر به فراخوانی فایلی که در حافظه موجود است می باشید .
  BASCOM فایلها را بصورت استاندارد ASCII ذخیره می کند .بنابراین شمکا می توانید از ویرایشگری مثل NOTEPAD برای نوشتن برنامه استفاده کنید و سپس آنرا به محیط انتقال دهید.
- **بستن فایل ( CLOSE FILE )**
  این گزینه پنجره برنامه فعال را می بندد . اگر در فایل تغییری ایجاد کرده اید ابتدا باید قبل از بستن آن را ذخیره نمایید .
- **ذخیره فایل ( FILE SAVE)**
  با این گزینه شما قادر به ذخیره فایل بصورت ASCII در کامپیوتر خواهید بود .
- **ذخیره کردن بعنوان ( FILE SAVE AS)**
  با این گزینه قادر خواهید بود فایل موجود را با نام دیگر ذخیره کنید.

# ادامه منوی FILE...

- **نمایش پرینت فایل ( FILE PRINT PREVIEW)**

این گزینه نشان می دهد که فایل متنی موجود برنامه در هنگام پرینت به چه صورت خواهد بود .

- **پرینت فا یل ( FILE PRINT)**

با این گزینه شما می توانید فایل موجود در برنامه را پرینت نمایید .

- **بستن فایل ( CLOSE FILE )**

با این گزینه شما قادر خواهید بود از محیط BASCOM خارج شوید ولی در صورتی که شما در برنامه تان تییری داده اید و آن را ذخیره نکرده اید , پیش از خروج هشدار میدهد.

# منوی EDIT

- **EDIT UNDO**

با این گزینه شما می توانید دستکاری اخیرتان در برنامه را از بین ببرید .

- **EDIT REDO**

با این گزینه شما می توانید دستکاری اخیرتان را که از بین برده بودید دوباره برگردانید .

- **EDIT CUT**

با این گزینه شما می توانید متن انتخاب شده را بریده و به محل جدیدی انتقال دهید .

- **EDIT COPY**

با این گزینه شما می توانیدمتن انتخاب شده را کپی کرده و به محل جدیدی انتقال دهید .

- **EDIT PAST**

با این گزینه شما می توانید متنی را که قبلا COPY یا CUT کرده بودید در محل مورد نظر بچسبانید .

# منوی EDIT ...

- **EDIT FIND**

با این گزینه شما می توانید  متنی را در برنامه تان جستجو کنید .

- **EDIT FIND NEXT**

با این گزینه شما می توانید  متن مورد جستجو را دوباره جستجو نمایید .

- **EDIT REPLACE**

با این گزینه شما می توانید متنی را جایگزین متن موجود در برنامه نمایید یعنی در قسمت TEXT TO FIND متن مورد جستجو که باید توسط متن دیگری جایگزین شود را تایپ کنید و در قسمت  REPLACE WITH متنی را که باید جایگزین شود تایپ می کنیم .

- **EDIT  GOTO**

با این گزینه شما می توانیدمستقیما و به سرعت به خط دلخواهی بروید .

# منوی EDIT ...

- **EDIT TOGGLE BOOKMARK**

با این گزینه شما می توانید  شما می توانید در جاهای خاصی از برنامه که مورد نظر شماست نشانه گذاری کنید و به آنها توسط دستور دستور  EDIT GOTO BOOKMARKدسترسی پیدا کنید .

- **EDIT GOTO BOOKMARK**

با این گزینه شما می توانید  به نشانه هایی که قبلا گذاشته اید .

- **EDIT IDENT  BLOCK**

با این گزینه شما می توانید متن انتخاب شده را به اندازه یک TAB به سمت راست منتقل کنید .

- **EDIT  UNIDENT BLOCK**

با این گزینه شما می توانید متن انتخاب شده را به اندازه یک  TAB به سمت چپ منتقل کنید .

# منوی PROGRAM

- PROGRAM COMPILE

با این گزینه (یا کلید F7) شما قادر به ترجمه برنامه به زبان ماشین (COMPILE ) خواهید بود
.برنامه شما با انتخاب این گزینه پیش از COMPILE ذخیره خواهد شد و فایلهای زیر به انتخاب شما در OPTION COPILER SETTING ایجاد خواهند شد :

- XX.BIN فایل باینری که می تواند در میکروکنترلر PROGRAM شود .
- XX.DBG فایل DEBUG که برای نرم افزار شبیه ساز BASCOM مورد نیاز است .
- XX.OBJ فایل OBJECT که برای نرم افزار AVR STUDIO مورد نیاز است .
- XX.RPT فایل گزارشی
- XX.HEX فایل هگزادسیمال اینتل که برای بعضی از انواع PROGRAMMER ها مورد نیاز است .
- XX.ERR فایل خطا که فقط در هنگام بروز خطا ایجاد می شود.
- XX.EPP داده های که باید در EPROM برنامه ریزی شود در این فایل نگهداری میگردند .




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# منوی PROGRAM ...

اگر خطایی در برنامه موجود باشد شما پیغام خطا را در یک کادر محاوره ای دریافت خواهید کرد و COMPILE متوقف میشود. با کلیک بر روی هر کدام از انها به خطی که خطا در آن رخ داده پرش خواهید کرد .

- **PROGRAM SYNTAX CHECK**

بوسیله این گزینه برنامه شما برای نداشتن خطای املایی چک می شود .اگر خطایی وجود داشته باشد هیچ فایلی ایجاد نخواهد شد .

- **PROGRAM SHOW RESULT**

از این گزینه برای دیدن نتیجه COMPILE میتوان استفاده کرد .

گزینه OPTION COMPILE OUTPUT را برای تعیین اینکه کدام فایلها باید ایجاد شوند را ببینید . فایلهایی که محتوای آنها قابل مشاهده اند REPORT ERROR می باشند .

# منوی PROGRAM ...

- **PROGRAM SIMULATOR**

با فشردن کلید F2 یا این گزینه از منو PROGRAM شبیه ساز داخلی فعال خواهد شد .شما در برنامه با نوشتن کلمه کلیدی $SIM قادر به شبیه سازی سریعتر برنامه میباشید .در صورت تمایل شما می توانید از شبیه سازی های دیگر مانند AVR STUDIO نیز استفاده کنید . برای شبیه سازی فایلهای DBJ و OBJ باید ایجاد شده باشند . فایل OBJ در برنامه شبیه سازی AVR STUDIO و فایل DBJ برای شبیه ساز داخلی مورد استفاده قرار می گیرد .

- **SEND TO CHIP**

توسط این گزینه یا کلید F4 پنجره محیط برنامه ریزی ظاهر خواهد شد .شما می توانید توسط این گزینه میکرو مورد نظر خود را PROGRAM کنید .




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# منوی TOOLS

- **TERMINAL EMULATOR**

توسط این گزینه یا کلیدهای CTR + T با بالا آوردن TERMINAL EMULATOR می توانید از این محیط برای نمایش داده ارسالی و دریافتی در ارتباط سریال RS-232 بین میکرو و کامپیوتر استفاده نمایید .

- **LCD DESIGNER**

توسط این گزینه می توانید کاراکترهای دلخواه خود را طراحی نمایید و بر روی LCD نمایش دهید.

# منوی TOOLS ...

- GRAPHIC CONVERTOR

با کلیک بر روی این منو پنجره محیط GRAPHIC CONVERTOR برای تبدیل تصویر با پسوند *.BMP به تصویری با پسوند *.BGF که قابل نمایش بر روی GRAPHIC LCDاست ظاهر می شود .

فایل دلخواه خود را با پسوند *.BMP توسط دکمه LOAD وارد کرده و سپس با دکمه SAVE آنرا در کنار برنامه خود با پسوند *.BGF ( BASCOM GRAPHIC FILE) ذخیره کنید .فایل تبدیل شده بصورت سیاه و سفید دوباره نمایش داده می شود و با کلیک بر روی دکمه OK می توان از محیط خارج شد . فایل ذخیره شده با فراخوانی در برنامه قابل نمایش بر روی LCD گرافیکی است . انتخاب نوع LCD توسط قسمتLCD TYPE انجام می گیرد . فونت نوشتاری نیز می تواند 6*8 یا 8*8 پیکسل باشد .

# منوی OPTION

- **OPTION COMPILER**

با این منو شما می توانید گزینه های مختلف کامپایلر را طبق زیر اصلاح نمایید :

- **OPTION COMPILER CHIP**

انتخاب میکرو برای برنامه ریزی توسط این گزینه انجام می شود . در صورتی که از دستور $REGFILE در برنامه استفاده کرده اید به انتخاب میکرو توسط این گزینه نیازی نیست .

- **OPTION COMPILER OUTPUT**

با این گزینه می توان فایل هایی که مایل به ایجاد آنها پس از کامپایل هستیم را انتخاب کرد . با انتخاب گزینه SIZE WARNING زمانی که حجم CODE از مقدار حافظه FLASH ROM تجاوز کرد کامپایلر تولید WARNING می کند .

- **OPTION COMPILER 12C,SPI,1WIRE**

توسط این گزینه می توان پایه های مربوط به ارتباطات 12C SPI و 1WIRE را تعیین کرد .

# منوی OPTION ...

- OPTION COMPILER COMMUNICATION

نرخ انتقال (BOUD RATE) ارتباط سزیال توسط این گزینه تعیین می شود که می توان یک نرخ جدید نیز تایپ کرد . گزینهFREQUENCY انتخاب فرکانس کریستال استفاده شده است که می تواند فرکانس اختیاری نیز باشد .

- OPTION COMPILER LCD

این گزینه دارای قابلیت های زیر می باشد :

در قسمت LCD TYPE نوع LCD را مشخص می کنیم .گزینه BUS MODE مشخص می کند LCD بصورت ۸ بیتی یا ۴ بیتی کار می کند .توسط گزینه DATA MODE تعیین می کنیم LCD بصورت PIN کار کند یا BUS و گزینه LCD ADDRESS مشخص کننده آدرس LCD در مد BUS است .

در صورت پیکره بندی هر یک از امکانات فوق در برنامه نیازی به تنظیم کردن آنها در این منو نیست .

- OPTION PROGRAMMER

•در این منو شما می توانید PROGRAMMER مورد نظر خود را انتخاب نمایید .

# معرفی محیط شبیه سازی
# (SIMULATOR)

میکروکنترلرهای AVR

# نوار ابزار در این محیط

- **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 دکمه 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 .

## نوار ابزار در این محیط ...

- **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 ) .

# نوار ابزار در این محیط ...

- **شبیه سازی سخت افزاری THE HARDWARE SIMULATOR**

با کلیک بر روی این گزینه ÷نجره ای ظاهر می شود . که قسمت بالایی یک LCD مجازی می باشد که برای نشان دادن داده های فرستاده شده به LCD استفاده می شود . نوار LED های قرمز رنگ پایین خروجی پورتها را نشان می دهد . با کلیک بر روی هر یک از LED های سبز رنگ که بعنوان ورودی هستند وضعیت آن معکوس می شود و روشن شدن LED بمنزله یک کردن پایه پورت است .یک صفحه کلید نیز تعبیه شده است که با دستور GETKBD( ) در برنامه قابل خواندن می باشد . در ضمن مقدار آنالوگ نیز هم برای مقایسه کننده آنالوگ و هم برای کانال های مختلف ADC قابل اعمال است.

- **REGISTERS**

این دکمه 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 .

- **I/O REGISTERS**
- برای نمایش ثباتهای I\O استفاده می شود . که مانند R قابل مقدار دهی است .

# نوار ابزار در این محیط ...

- **VARIABLES**

شما قادر به انتخاب متغیر با دو بار کلیک کردن در ستون VARIABLES میباشید . با فشار دکمه ENTER در هنگام اجرای برنامه قادر به مشاهده مقدار جدید متغیر در برنامه خواهید بود .
همچنین میتوانید مقدار هر متغیر را توسط VALUE تغییر دهید .
برای تماشای یک متغیر آرایه ای می توانید نام متغیر همراه با اندیس آنرا تایپ کنید و برای حذف هر سطر می توانید دکمه CTRL+DEL را فشار دهید .

- **WATCH**

این گزینه برای وارد کردن وضعیتی که قرار است در خلال شبیه سازی ارزیابی شود مورد استفاده قرار می گیرد و هنگامی که وضعیت مورد نظر صحیح شد شبیه سازی در حالت PAUSE قرار خواهد گرفت . حالت مورد نظر را در مکان مورد نظر تایپ نموده و دکمه ADD-BUTTON را فشار دهید . هنگامیکه دکمه MODIFY-BUTTON فشار داده شود , وضعیت مورد نظر را مورد بازنگری قرار میدهد و میتوان ارزش آنرا تغییر داد . برای حذف هر وضعیت شما باید آنرا انتخاب کرده و دکمه REMOVE را فشار دهید .

## نوار ابزار در این محیط ...

- **LOCAL**

  متغیرهای محلی موجود در SUB یا FUNCTION را نشان میدهد . البته نمیتوان متغیری را به آن اضافه نمود .

- **UP**

  وضعیت ثبات وضعیت ( STATUS REG) را نشان میدهد . FLAG ها را میتوان توسط کلیک بر روی CHECK BOX ها تغییر وضعیت داد .

- **INTERRUPTS**

  این گزینه منابع وقفه را نشان میدهد . هنگامیکه هیچ ISR برنامه نویسی نشده باشد , همه دکمه ها غیر فعال خواهند بود و اگر ISR نوشته شود , دکمه مربوط به آن فعال می شود و با کلیک بر روی هر کدام از دکمه ها , وقفه مربوطه اجرا می شود . در ضمن میتوان روی یک پایه خاص پالس نیز ایجاد نمود .

# معرفی محیط برنامه ریزی

میکروکنترلرهای AVR

# ISP STK PROGRAMMER

- پنجره ارسال برنامه به میکرو هنگامیکه RUN PROGRAMMER انتخاب می شود ظاهر میگردد .
- **منوی FILE**
  - **EXIT :** خروج از محیط برنامه ریزی .
  - **TEST :** یک کردن پایه های پورت . این گزینه تنها زمانی می تواند استفاده شود که از SAMPLR ELECTRONIC PROGRAMEEER استفاده شود .
- **منوی BUFFER**
  - **BUFFER CLEAR :** پاک کردن بافر.
  - **LOAD FROM FILE :** پر کردن بافر با فایل و برنامه ریزی آن در حافظه میکرو
  - **SAVE TO FILE :** ذخیره بافر در فایل دلخواه . بافر می تواند محتوای حافظه یک میکرو باشد .
- **منوی CHIP**
  - **CHIP IDENTIFY :** شناسایی میکرو متصل به PROGRAMMER .

# ISP STK PROGRAMMER...

- **WRITE BUFFER TO CHIP :** برنامه ریزی محتوای بافر در حافظه ROMیا EEPROM .
- **READ CLIPCODE INTO BUFFER :** خواندن داده حافظه کدی میکرو .
- **BLACK CHECK :** خالی بودن حافظه میکرو را مشخص می کند .
- **ERASE :** پاک کردن محتوای حافظه برنامه و داده EEPROM .
- **VERIFY :**این گزینه محتوای بافر و آنچه که در میکرو برنامه ریزی شده است را مقایسه می کند و در صورت تساوی پیغام VERIFY OK نمایش داده می شود .
- **AUTO PROGRAM :** حافظه میکرو را پاک کرده و برنامه مورد نظر را در حافظه FLASH برنامه ریزی می کند و سپس عمل VERIFY را به صورت خودکار انجام می دهد.
- **RESET :** میکرو متصل به PROGRAMMER را ریست می کند .

# معرفی محیط
# TERMINAL EMULATOR

میکروکنترلرهای AVR

# TERMINAL EMULATOR

- از این محیط می توان برای نمایش داده ارسالی و دریافتی در ارتباط سریال RS-232 بین میکرو و کامپیوتر استفاده نمود .

- اطلاعاتی که در این محیط تایپ می شود به میکرو ارسال و اطلاعاتی که از پورت کامپیوتر دریافت می شود دراین 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 به بیرون را دریافت کرده و نمایش داد و از صحت و سقم 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 .توجه داشته باشید که از BOUD RATE مشابه در میکرو و کامپیوتر استفاده نمایید .

## منوهای محیط

# TERMINAL EMULATOR

- **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 .
- **TERMINAL OPEN LOG :** فایل LOG 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 .

# دستورات و توابع محیط برنامه نویسی BASCOM

5

# بدنه یک برنامه درمحیط BASCOM

میکروکنترلرهای AVR

# بدنه یک برنامه درمحیط BASCOM ...
## معرفی میکرو

$REGFILE = VAR

برای شروع یک برنامه در محیط BASCOM ابتدا باید میکرو مورد نظر تعریف گردد . VAR نام چیپ مورد استفاده است که می تواند یکی از موارد زیر باشد .

$regfile = “ At12def.dat “ ‘ATtiny12 MCU
$regfile = “ At15def.dat “ ‘ATtiny15 MCU
$regfile = “ At22def.dat “ ‘ATtiny22 MCU
$regfile = “ At26def.dat “ ‘ATtiny26 MCU

$regfile = “ 2323def.dat “ ‘AT90s2323 MCU
$regfile = “ 2333def.dat “ ‘AT90s2333 MCU
$regfile = “ 2343def.dat “ ‘AT90s2343 MCU
$regfile = “ 4414def.dat “ ‘AT90s4414 MCU
$regfile = “ 4433def.dat “ ‘AT90s4433 MCU
$regfile = “ 4434def.dat “ ‘AT90s4434 MCU

# بدنه یک برنامه درمحیط BASCOM ...
# معرفی میکرو ...

```
$regfile = " 8515def.dat "          'AT90s8515 MCU
$regfile = " 8535def.dat "          'AT90s8535 MCU

$regfile = " M8535.dat "           'MEGA 8535 MCU
$regfile = " M8515.dat "           'MEGA 8515 MCU
$regfile = " M8def.dat "           'MEGA 8 MCU
$regfile = " M103def.dat "         'MEGA 103 MCU
$regfile = " M16def.dat "          'MEGA 16 MCU
$regfile = " M163def.dat "         'MEGA 163 MCU
$regfile = " M161def.dat "         'MEGA 161 MCU
$regfile = " M32def.dat "          'MEGA 32 MCU
$regfile = " M323def.dat "         'MEGA 323 MCU
$regfile = " M603def.dat "         'MEGA 603 MCU
$regfile = " M64def.dat "          'MEGA 64 MCU
$regfile = " M128def.dat "         'MEGA 128 MCU
$regfile = " M128103.dat "         'MEGA 128 IN MEGA 103 MODE MCU
```

# بدنه یک برنامه درمحیط BASCOM ...
# کریستال

برای مشخص کردن فرکانس کریستال استفاده شده برحسب هرتز از دستور زیر استفاده می نماییم .

$CRYSTAL = X

X فرکانس کریستال استفاده شده بر حسب هرتز است .


این دستور را حتی برای زمانی که با اسیلاتور داخلی میکرو کار میکنید بنویسید .

- **مثال**

```
$CRYSTAL = 14000000          '14MHZ external osc
$CRYSTAL = 8000000           '8MHZ external osc
$CRYSTAL = 1000000           '1MHZ internal osc
```

# بدنه یک برنامه درمحیط BASCOM ...
# اسمبلی و بیسیک ( اختیاری)

در صورت نیاز برای نوشتن برنامه اسمبلی در بین برنامه بیسیک از دستور زیر استفاده می نماییم

```
$ASM
        ASSEMBLY PROGRAMME
$ENDASM
```

با دستور $ASM می توان در برنامه شروع به نوشتن برنامه موردنظر اسمبلی کرده و پس از اتمام برنامه اسمبلی با دستور $ENDASM برنامه اسمبلی را به پایان رساند و به نوشتن ادامه برنامه پرداخت .

**⊡ مثال**

```
Dim c As Byte
Loadadr c,x              'load address of variable c into register x
$Asm                     'start assembly program
  Ldi r24,1               'load register R24 with the constant 1
  St x,R24               'store 1 into var c
$End Asm                 'end of assembly program
Print c                  'send c to serial port
End
```




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# بدنه یک برنامه درمحیط BASCOM ...
# یادداشت ( اختیاری)

گاهی نیاز است یادداشتهایی برای اطلاعات بیشتر در برنامه اضافه شود .

REM یا ‘

یاددلشتها و نوشته های بعد از این دستور غیر فعال بوده و در برنامه برای یادداشت به کار می رود و کامپایل نخواهد شد و همچنین به رنگ سبز در می آیند .

همچنین می توان از دو علامت برای شروع )‘ و از (‘ برای اتمام متن یادداشتی استفاده نمایید .

**⊡ مثال**

```
REM this sentence will not be compiled
Or
‘   this sentence will not be compiled
```

**⊡ مثال**

```
‘( start block comment
        This will not be compiled
‘) end block comment
```

# بدنه یک برنامه درمحیط BASCOM ...
## آدرس شروع برنامه ریزی حافظه FLASH ( اختیاری)

گاهی نیاز است که برنامه خود را از آدرسی دلخواه در حافظه FLASHROM قرار دهید .

$ROMSTART = ADDRESS

ADDRESS مکانی از حافظه است که برنامه HEX از این آدرس در حافظه میکرو کنترلر , شروع به نوشته شدن می شود . در صورتی که ازاین دستور استفاده نشود کامپایلر به طور خودکار آدرس &H0000 را در نظر می گیرد .

- **مثال**

$ROMSTART = &H4000

# بدنه یک برنامه درمحیط BASCOM ...
## تعیین کلاک ( اختیاری)

با این دستور در بعضی از میکروهای سری MEGA AVR از جمله MEGA103 یا MEGA603 به صورت نرم افزاری می توان کلاک سیستم را تغییر داد .تقسیم کلاک بطور مثال برای کاهش مصرف تغذیه استفاده می شود .

CLOCKDIVISION = var

Var مقادیر معتبر بین اعداد ۲ تا ۱۲۸ می تواند باشد .

اگر از این دستور استفاده نمایید , دستوراتی که مستقیما با کلاک سیستم کار می کنند ممکن است درست کار نکنند .

نکته

مثال

```
$boud = 2400
Clockdivision = 2
Print “Hello”
End
```

# بدنه یک برنامه درمحیط BASCOM ...
## پایان برنامه

END

این دستور در انتهای برنامه قرار می گیرد و اجرای برنامه را متوقف می کند . با این دستور تمام وقفه ها غیر فعال شده و یک حلقه بی نهایت تولید و برنامه خاتمه می یابد .

- **مثال**

```
PRINT  “ Hello”          ‘print this
END                      ‘ end program execution and disable all interrupt
```

# اعداد و متغیر ها و جداول

# LOOKUP

میکروکنترلرهای AVR

# اعداد و متغیر ها و جداولLOOKUP...

## دیمانسیون متغیرها

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

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

VAR نام متغیری  که در برنامه بکار برده میشود . در صورت استفاده از حافظه جانبی آنرا با XRAM مشخص کنید و  SRAM را زمانی اختیار کنید که می خواهید متغیرها را در حافظه  SRAM قرار دهید و  ERAM متغیر مورد نظر را در  EEROM داخلی جای میدهد . Data type نوع داده است که می تواند طبق جدول زیر BIT, BYTE ,INTEGER ,LONG ,WORD ,STRING یا SINGLE باشد .

در صورت استفاده از متغیر STRING , بیشترین طول آن نیز باید نوشته شود .گزینه اختیاری OVERLY متغیر تعریف شده را بصورت  POINTER  در نظر میگیرد و فضایی را برای متغیر در نظر نمی گیرد .

AT LOCATION به شما اجازه میدهد که متغیرتان را در آدرسی که میخواهید در حافظه ذخیره کنید زمانی که محل آدرسدهی اشغال باشد , اولین جای خالی در حافظه استفاده می شود .

# اعداد و متغیر ها و جداولLOOKUP...

## دیمانسیون متغیرها ...

| DATA TYPE | STORE AS | VALUE RANGE |
|---|---|---|
| BIT | A BIT | 0 OR 1 |
| BYTE | UNSIGNED 8 BITS | 0 TO 255 |
| INTEGER | SIGNED 16 BITS | -32767 TO 32767 |
| WORD | UNSIGNED 16 BITS | 0 TO 65535 |
| LONG | SIGNED 32 BITS | -214783648 TO 214783647 |
| SINGLE | SIGNED 32 BITS | 1.5 *10^-45 TO 3.4*10^38 |
| STRING | 0-245 BYTES | - |

# اعداد و متغیر ها و جداولLOOKUP...

## دیمانسیون متغیرها ...

- مثال

```
DIM B AS BIT        'BIT can be 0 or 1
DIM A AS BYTE    'BYTE range from  0  - 255
DIM K AS INTEGER AT 120      'you can specify the address of the       'variable .
                    The next dimensioned variable will be placed after A
```


عدد HEX را با علامت &H و عدد BINARY را با علامت &B نشان دهید

- مثال

```
A= &H01DE          'HEX NUM
B= &B01011011     'BIN NUM
```

- مثال

```
DIM B1 AS BYTE  AT $60 OVERLY
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور CONST

برای تعریف یک ثابت 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 ، STRINGCONST رشته انتساب یافته به SYMBOL و EXPRESSION میتواند عبارتی باشد که نتیجه آن به SYMBOL انتساب یابد .

**▣ مثال**

CONST S = “TEST”

CONST A = 5

CONST B1 =&B1001

CONST X = (B1 * 3 ) + 2

# اعداد و متغیر ها و جداولLOOKUP...

## دستور ALIAS

از این دستور برای تغیر نام متغیر استفاده می شود .

- **مثال**

DIRECTION ALIAS PORTB.1

حال شما می توانید بجای PORTB.1 از متغیر DIRECTION استفاده نمایید .

SET DIRECTION ‘is equal with SET PORTB.1

# اعداد و متغیر ها و جداولLOOKUP...

## دستور CHR

از این دستور برای تبدیل متغیر عددی یا یک ثابت به کاراکتر استفاده می شود . زمانی که قصد دارید یک کاراکتر بر روی LCD نمایش دهید از این دستور می توانید استفاده نماتید .

در صورتیکه از این دستور به این صورت استفاده نمایید PRINT CHR ( VAR ) کاراکتر اسکی VAR به پورت سریال فرستاده خواهد شد .

- **مثال**

```
DIM a AS Byte          ‘dim variable
A = 65                 ‘assign variable
Print a                ‘print value ( 65 )
Print HEX( a )         ‘print hex value (41)
Print Chr ( a )        ‘print ASKII character 65 (A )
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور INSTR

این دستورمحل 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 .

- **مثال**

```
DIM S AS String * 15, Z As String * 5
DIM Bp As Byte
S = “This is a test “
Z=  “is”
Bp = Instr (S , Z ) : Print Bp              ‘should print 3
Bp = Instr (4 , S ,Z ) : Print Bp           ‘should print 6
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور INCR

این دستور یک واحد به متغیر عددی VAR می افزاید .

INCR VAR

**مثال**

```
DO                  ‘ start loop
Incr A              ‘ increment  A by 1 A=A+1
Print A             ‘ print A
Loop Until A>10     ‘ repeat until A is greater than 10
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور DECR

این دستور متغیر VAR را یک واحد کم می کند .

DECR VAR

**مثال**

```
Dim A As Byte
A = 5                    ‘ assign value to a
Decr A                   ‘ decrement  by one A= A-1
Print A                  ‘ print A =4
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور CHECKSUM

این دستور مجموع کد دسیمال اسکی رشته VAR را برمی گرداند که البته اگر مجموع کد اسکی رشته از عدد ۲۵۵ بیشتر شود مقدار ۲۵۶ از مجموع کم می شود .

- **مثال**

```
Dim S As String*10          ‘ Dim Variable
S = “test”                  ‘ assign Variable
Print Checksum (S)          ‘ print value (192 )
S = ‘test next “            ‘ assign variable
Print Cecksum(S)            ‘ Print value 127 (127=383 – 256 )
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور LOW

این دستور LSB ( least significant byte) یک متغیر را برمی گرداند .

Var = LOW ( s)

LSB متغیر S در Var قرار می گیرد .

**⊡ مثال**

```
Dim I As Integer , Z As Byte
I = &h1001
Z = LOW (I )                    ‘ is 1
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور HIGH

این دستور MSB ( most significant byte) یک متغیر را برمی گرداند .

Var = HIGH ( s)

MSB متغیر S در Var قرار می گیرد .

**⊡ مثال**

```
Dim I As Integer , Z As Byte
I = &H1001
Z = HIGH (I )                ‘ Z is 16 z = &H10
I = &H1101
Z = HIGH (I )                ‘Z is 17  z = &H11
I = 1012
Z = HIGH (I )                ‘I = &H3F4 z is 3
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## 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 .

**⊡ مثال**

```
Dim S As String * 12 , Z As String * 12
S = “Hello World “
Z = Lcase (S )                    ‘Z = hello world
Print Z
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## 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 .

**⊡ مثال**

```
Dim S As String * 12 , Z As String * 12
S = “Hello World “
Z = Ucase ( s )                    ‘Z = HELLO WORLD
Print Z
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور RIGHT

با این دستورقسمتی 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 .

**⊡ مثال**

```
Dim S As String * 15 , Z As String * 15
S = “ABCDEFG “
Z = Right( s , 2)                    ‘Z = FG
Print Z
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## 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**

```
Dim S As String * 15 , Z As String * 15
S = “abcdefg “
Z = Left( s , 5)                    ‘Z = abcde
Print Z
Z = Left( s , 1)                    ‘Z = a
Print Z
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور LEN

این دستور طول یا بعبارتی تعداد کاراکترهای یک رشته را برمیگرداند .

Var = Len(string )

طول رشته string در متغیر عددی VAR قرار می گیرد . رشته string نهایتا می تواند ۲۵۵ بایت طول داشته باشد . توجه داشته باشید که فضای خالی (SPACE BAR) خود یک کاراکتر به حساب می آید .

**⊡ مثال**

```
Dim S As String * 12
Dim A As Byte
S = “test “
A= Len(S )
Print A                    ‘Print 4
Print Len (S )             ‘Print 4
S=“test  “
A = Len ( A )
Print A                    ‘Print 5
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور LTRIM

این دستور فضای خالی یکرشته را حذف می کند .

Var = LTRIM( org )

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

**⊡ مثال**

```
Dim S As String * 10
S = “        AB “
Print LTRIM( s )              ‘AB
S = “        A  B “
Print LTRIM( s )              ‘A  B
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور SWAP

SWAP var1 , var2

با اجرای این دستور محتوای متغیر var1 در متغیر var2 و محتوای متغیر var2 در متغیر var1 قرار می گیرد .

نکته

دو متغیر var1وvar2 بایستی از یک نوع باشند .

**⊡ مثال**

```
Dim A As Integer , B1 As Integer
A = 1 :B1 = 2                    ‘assign two integer
SWAP A , B1                      ‘swap them
Print A                           ‘ prints 2
Print B1                          ‘ prints 1
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور 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 .

1- Var = Mid(var1,St[,L]

2- Mid(var , St[,L] = Var

۱- قسمتی از رشته var1 با شروع از کاراکتر stام و طول L برداشته شده و در متغیر var قرار می گیرد.

۲- 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 .

**مثال**

```
Dim A As XRAM String *15 , Z  As XRAM String *15
S = ‘ABCDEFG”
Z = Mid(S,2,3)
Print Z                          ‘BCD
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور ROTATE

دستور زیر تمام بیتها را به چپ یا راست منتقل می کند ولی تمام بیتها محفوظ هستند و هیچ بیتی بیرون فرستاده نمی شود .

ROTATE var ,LEFT/RIGHT [,shifts]

Var می تواند داده ای از نوع BYTE , INTEGER ,WORD ,LONG باشد . LEFT/RIGHT جهت چرخش بیتها و shift که اختیاری می باشد تعداد چرخش بیتها را مشخص می کند. در صورت قید نشدن مقدار یک در نظر گرفته می شود .

**⊡ مثال**

```
Dim A As Byte
A = 128
Rotate A, Left ,2
Print A                    'a=2
```

# اعداد و متغیر ها و جداولLOOKUP...

## دستور SPACE

برای ایجاد فضای خالی ازاین دستور استفاده می شود .

Var = SPACE (x )

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

**⊡ مثال**

```
Dim S As String *15
S = Space (5)
Print “{“ ;S ;”}”                    ‘{    } 5space
Print “{“ ; Space(6) ; “}”           ‘{     } 6 space
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## تابع FORMAT

این دستور یک رشته عددی را شکل دهی می کند .

target = Format (source , “mask”)

sourceرشته ای است که شکل دهی شود و نتایج در target قرار می گیرد .mask نوع شکل دهی است .

**⊡ مثال**

```
Dim S As String *10, I As Integer
S = “ 123 “
S= Format (s, “      “)          ‘5 space
Print S                          ‘s=“  123” two space first ,then 123
S =“12345”
S = Format(s , “000.000”)
Print S                           ‘s =“012.345
S = Format(s , “ + ”)
Print S                           ‘s =“+12345
End
```




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# اعداد و متغیر ها و جداولLOOKUP...

## تابع FUSING

از این دستور برای روند کردن رشته های عددی استفاده می شود .

target = Fusing (source , “mask”)

sourceرشته 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**

```
Dim S As Single,Z As  String*10
S = 123.45678
Print Fusing(S , .”#.# #”)          ‘Print 123.46
Print Fusing(S , .”#.& #”)          ‘Print 123.45
End
```

# اعداد و متغیر ها و جداولLOOKUP...

## جدول LOOKUP

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 برگشتنی از جدول در متغیر var قرار می گیرد .value =0 اولین داده در جدول را برمی گرداند . تعداد اندیس ها و مقدار داده برگشتی به ترتیب نهایتا می تواند ۲۵۵و ۶۵۵۳۵ باشد .

داده دو بایتی داخل جدول بایستی با علامت % پایان یابد .

نکته

مثال

```
Dim B1 As Byte, I As Integer
B1= lookup(2 , Dta)
Print B1                    'Print 2 (zero based )
I = lookupstr( 0, Dta2 )
Print I                     'Print 1000
Dta:
Data  1 , 2 , 3, 4 , 5
Dta2:
Data  1000% , 2000%
```

# اعداد و متغیر ها و جداولLOOKUP...

## جدول LOOKUPSTR

توسط این جدول می توان رشته دلخواهی را از جدولی برگرداند.

var = LOOKUPSTR(value , label )

Label برچسب جدول و value اندیس رشته دلخواه است . رشته برگشتنی از جدول در متغیر رشته ای var قرار می گیرد .value =0 اولین رشته در جدول را برمی گرداند . تعداد اندیس ها نهایتا می تواند ۲۵۵ باشد .

**▣ مثال**

```
Dim S As String*4 , Idx As Byte
Idx = 0 : S = lookupstr( idx , Sdata )
Print S                                ‘ This
End

Sdata:
Data  “This” , “is” , “a test”
```

# توابع ریاضی و محاسباتی

میکروکنترلرهای AVR

# توابع ریاضی و محاسباتی...
## عملگرهای ریاضی

از عملگرهای ریاضی روبرو می توانید در محیط BASCOM استفاده نماییدو عملیات ریاضی خود را انجام دهید .

| علامت | نماد |
|---|---|
| علامت ضرب | * |
| علامت جمع | + |
| علامت تفریق | - |
| علامت ممیز | . |
| علامت تقسیم | / |
| علامت کوچکتر از | < |
| علامت تساوی | = |
| علامت بزرگتر از | > |
| علامت بتوان | ^ |
| علامت کوچکتر یا مساوی | => |
| علامت بزرگتر یا مساوی | <= |
| علامت مخالف | <> |

# توابع ریاضی و محاسباتی...
# عملگرهای منطقی

عملگر های منطقی BASCOM به قرار زیر است :

| معرفی | نماد |
|---|---|
| CONJUNCTION | AND |
| DISJUNCTION | OR |
| EXCLUSIVE OR | XOR |
| COMPLIMENT | NOT |

- **مثال**

```
A = 63 and 19
Print A                    ''19 print
```

# توابع ریاضی و محاسباتی...
## تابع ABS

این دستور به معنای ریاضی  VAR = |VAR2|  (قدرمطلق) است .

- **مثال**

```
Dim A  As Integer , C As Integer
A = -1000
C = ABS (A)                    'c=|a|
Print C                        'C= 1000
End
```

# توابع ریاضی و محاسباتی...
## تابع EXP

Target = Exp (source)

Targetبرابر با e بتوان source است . Targetمتغیری از نوع داده SINGLE است .

▣ **مثال**

```
Dim X  As Single
X= Exp( 1.1)
Print X                    'Print 3.004166124
X = 1.1
X= Exp( X)
Print X                    'Print 3.004166124
End
```

# توابع ریاضی و محاسباتی...
## تابع LOG10

Target = Log10 (source)

لگاریتم پایه ۱۰ متغیر یا ثابت source در متغیر target قرار می گیرد . Target و source هر دو داده نوع single هستند .

**مثال**

```
Dim S1  As Single, S2 As Single
S1 = 0.01
S2 = Log10(S1)
Print S2
      For S1=1 to 100
            S2 = Log10(S1)
            Print S1 ;"          "   ;S2
      NEXT
End
```

# توابع ریاضی و محاسباتی...
## تابع 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 .ودر متغیر target قرار می گیرد .

- **مثال**

```
Dim X  As Single
X = Log(100)                    ‘4.605170
Print X
End
```

# توابع ریاضی و محاسباتی...

## تابع RND

این دستور یک عدد 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 .


limit باید یک عدد مثبت باشد .

**مثال**

```
Dim X  As Integer
Do
      I = Rnd (100)        'get random number
      Print I
      Wait 1
Loop
End
```

# توابع ریاضی و محاسباتی...
## تابع SIN

این دستور سینوس 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**

```
Dim X  As Single
Dim Vsin As Single
Const Pi= 3.14159265
X= Pi/2
Vsin = Sin (X)                    'Vsin = sin(p/2)
Print Vsin                        '0.9999332
End
```




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# توابع ریاضی و محاسباتی...
# تابع COS

این دستور کسینوس 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**

```
Dim X  As Single
Dim Vcos As Single
Const Pi= 3.14159265
X= Pi/2
Vcos = Cos (X)                  'Vcos = cos(p/2)
Print Vcos                      '0.0000066
End
```

# توابع ریاضی و محاسباتی...
## تابع TAN

Var = TAN (source )

این دستور 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**

```
Dim X  As Single
Dim Vtan As tangle
Const Pi= 3.14159265
X= Pi*2
Vtan = tan (X)                    'Vtan = tan(p*2)
Print Vtan                        ' -0.00000357
End
```

# توابع ریاضی و محاسباتی...
## تابع SINH

Var = SINH( source)

این دستور سینوس 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**

```
Dim X  As Single
Dim Y As Single
X=  0.512
Y  = Sinh (X)
Print  X ; “ ; “ ;Y
End
```

# توابع ریاضی و محاسباتی...
## تابع COSH

Var = COSH( source)

این دستور کسینوس 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**

```
Dim X  As Single
Dim Y As Single
X=  0.512
Y  = Cosh (X)
Print  X ; “ ; “ ;Y
End
```

# توابع ریاضی و محاسباتی...
## تابع TANH

Var = TANH( source)

این دستور 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**

```
Dim X  As Single
Dim Y As Single
X=  0.512
Y  = Tanh (X)
Print  X ; “ ; “ ;Y
End
```

# توابع ریاضی و محاسباتی...
## تابع ASIN

Var = ASIN( source)

این دستور آرکسینوس ثابت یا متغیر source را در متغیر var از نوع SINGLE قرار می دهد .
ورودی تابع عددی بین 1-و 1+ می باشد .

- **مثال**

```
Dim X  As Single
Dim Y As Single
X=  0.5
Y  = Asin (X)
Print  X ; “ ; “ ;Y
End
```

# توابع ریاضی و محاسباتی...
## تابع ACOS

Var = ACOS( source)

این دستور آرککسینوس 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**

```
Dim X  As Single
Dim Y As Single
X=  0.5
Y  = ACOS (X)
Print  X ; “ ; “ ;Y
End
```

# توابع ریاضی و محاسباتی...
## تابع ATN

Var = ATN( source)

این دستور آرک 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**

```
Dim X  As Single
Dim Y As Single
X=  1
Y  = atn (X) * 4
Print  X ; “ ; “ ;Y
End
```

# توابع ریاضی و محاسباتی...
## تابع DEG2RAD

Var =DEG2RAD( single)

برای تبدیل درجه به رادیان از این دستوراستفاده می شود .

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

- **مثال**

```
Dim X  As Single
Dim Y As Single
X=  180
Y  = Deg2rad (X)
Print  Y                          ‘3.141592
End
```

# توابع ریاضی و محاسباتی...
## تابع RAD2DEG

Var =RAD2DEG( single)

برای تبدیل رادیان به درجه از این دستوراستفاده می شود .

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

- **مثال**

```
Dim X  As Single
Dim Y As Single
X=  3.141592
Y  = Rad2Deg (X)
Print  Y                          ‘179.9999
End
```

# توابع ریاضی و محاسباتی...
## تابع 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**

Round(2.3) =2 ; Round(-2.3)= -2

Round(2.8)=3 ; Round(-2.8)= -3

# تبدیل کدها و متغیرها به یکدیگر

میکروکنترلرهای AVR

# تبدیل کدها و متغیرها به یکدیگر ...
# دستور ASC

Var = ASC (string)

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

- **مثال**

```
Dim A  As Byte , S As string
S= “ABC”
A = ASC(s)
Print A                    ‘will print 65
End
```

# تبدیل کدها و متغیرها 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 .

- **مثال**

```
Dim A  As Byte , S As string*10
A= 123
S= Hex(A)
Print S                    ‘7B will print
Print Hex(A)               ‘7B will print too
End
```

# تبدیل کدها و متغیرها به یکدیگر ...
# دستور HEXVAL

Var = HexVal (x)

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

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

**مثال**

```
Dim A  As Integer , S As string*15
S= “0A”
A = Hexval (S)
Print A                          ‘10 will  be print
End
```

# تبدیل کدها و متغیرها به یکدیگر ...
# دستور 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 .

**مثال**

```
Dim A  As Byte
A = 65
A = Makebcd (A)
Lcd A                          ‘101 will  show
End
```

# تبدیل کدها و متغیرها به یکدیگر ...
# دستور MAKEDEC

Var1 = MAKEDEC (Var2)

برای تبدیل یک داده BCD نوع INTEGER , WORD , BYTE به مقدار DECIMAL از این دستوراستفاده می شود .مقدار دسیمال متغیر یا ثابت var2 در متغیر var1 جای می گیرد .

**مثال**

```
Dim A  As Byte
A = 65
Lcd A
Lowerline
Lcd Bcd (A)
A = Makedec (A)                    ' A = 101
Lcd "       ";A
End
```

# تبدیل کدها و متغیرها به یکدیگر ...
# 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

**مثال**

```
Dim A  As Integer, I As Integer
A = 2
I = Makeint (A , 1)                ‘(1*256)+2 =258
End
```

# تبدیل کدها و متغیرها به یکدیگر ...
# دستور STR

Var = STR (X )

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

**مثال**

```
Dim A  As Byte , S As String*10
A = 123                          ‘ A is a num
S= Str (A )                      ‘now A is a string
Print S
End
```

# تبدیل کدها و متغیرها به یکدیگر ...
# دستور VAL

Var = VAL (S )

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

**مثال**

```
Dim A  As Byte , S As String*10
S= “123”                          ‘now S is a string
A = Val(S)                        ‘convert string to num
Print A
A = A*2                           ‘now you can use it as a num
Print A                           ‘ 246 Prints
End
```

# تبدیل کدها و متغیرها به یکدیگر ...
# 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 ۲۵۵ کاراکتر تولید می شود و.قرار دادن n = 0 قابل قبول نیست .

**مثال**

```
Dim S As String*15
S= String (5 , 65 )
Print S                    ‘AAAAA
End
```

# تبدیل کدها و متغیرها به یکدیگر ...
# 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 .

**مثال**

```
Dim B  As Byte
For B = 0 To 15
        Print Bin2grey (B )      '0  1  3  2   6  7  5  4  12  13  15  ...
Next
End
```

# تبدیل کدها و متغیرها به یکدیگر ...

## 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**

```
Dim B  As Byte
For B = 0 To 15
        Print Grey2bin (B )      '0  1  3  2   7  6   4   5  15  14   ...
Next
End
```

# رجیسترها و آدرس های حافظه

میکروکنترلرهای AVR

# رجیسترها و آدرس های حافظه...

تمام میکروهای AVR دارای ۳۲ رجیستر ۸ بیتی ( R0 –R31) همه منظوره در CPU خود هستند .

رجیسترهای R31(MSB) با R30(LSB) , R29(MSB) با R28(LSB و R27(MSB) با R26(LSB) تشکیل سه رجیستر 16 بیتی با ترتیب با نامهای X,Y,Z را می دهند .

# رجیسترها و آدرس های حافظه...
# دستور SET

Set Bit/Pin
Set Var.x

توسط این دستور می توان یک بیت را یک کرد .

Bit می تواند یک بیت و یا یک SFR مانند PORTB.1 باشدو Var متغیری از نوع داده , BYTE INTEGER , WORD , LONG باشد . X برای BYTE می تواند 0 تا 7 , 0 تا 15 برای WORD و برای LONG می تواند 0 تا 31 باشد .

**مثال**

```
Dim B1 As Bit , B2 As Byte , C As Word , L As Long
Set Portb.1                    ‘set bit 1 of port B
Set B1                         ‘bit variable
Set B2.1                       ‘set bit 1 of var b2
Set C.15                       ‘set highest bit of word
Set L.31                       ‘set MS bit of LONG
```




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# رجیسترها و آدرس های حافظه...

## دستور TOGGLE

این دستور مقدار منطقی یک پایه یا یک بیت را معکوس می کند .

TOGGLE pin/bit

PIN می تواند یک SFR مانند PORTB.1 و یا یک بیت باشد .

**مثال**

```
Dim VAR As Byte
Config Pinb.0 = output          ‘portb.0 is an output now
Toggle portb.0                  ‘toggle state
Waitms 1000                     ‘wait for 1 second
Toggle Portb.0                  ‘toggle state again
```

# رجیسترها و آدرس های حافظه...

## دستور RESET

توسط این دستور می توان یک بیت را صفر کرد .

RESET pin/bit

RESET Var.x

Bit می تواند یک بیت و یا یک SFR مانند PORTB.1 باشدو Var متغیری از نوع داده BYTE , INTEGER , WORD , LONG باشد . X برای BYTE می تواند 0 تا 7 , 0 تا 15 برای WORD و برای LONG می تواند 0 تا 31 باشد .

**مثال**

```
Dim B1 As Bit , B2 As Byte , I As Integer
reset Portb.3                    ‘reset bit 3 of port B
reset B1                         ‘bit variableres
reset B2.0                       ‘reset bit 0 of var b2
reset I.15                       ‘reset highest bit of I
```

# رجیسترها و آدرس های حافظه...

## دستور BITWAIT

BITWAIT X, SET/RESET

توسط این دستور اجرای برنامه تا زمانی که بیت X, SET( = 1) یا RESET (= 0 ) شود در خط جاری متوقف می ماند .در صورت TRUE شدن شرایط , اجرای برنامه از خط بعد ادامه می یابد . X می تواند یک بیت رجیستر داخلی مانند PORTB.Y باشد که Y می تواند بین اعداد صفر تا ۷ تغییر کند .

**مثال**

```
Dim A As Bit
Bitwait A , .Set                ‘ wait until Bit A is Set
Bitwait PortB.7 , reset         ‘ wait until Bit 7 of Port B is 0
```

# رجیسترها و آدرس های حافظه...
# دستور CPEEK

Var = CPEEK(address)

از این دستور برای برگرداندن بایتی که در ادرسی از حافظه کدی ذخیره شده است استفاده می کنیم .با این دستور می توانید به رجیسترهای داخلی نیز دسترسی پیدا کنید . البته با این دستور نمی توان در حافظه داخلی چیزی نوشت .

- **مثال**

```
Dim I As Integer , B1 As Byte
For I = 0 To 31
      B1 = Peek (I )          ' only 32 registers in AVR
      Print Hex (b1)          ' get byte from internal memory (r0-r31)
Next
```

# رجیسترها و آدرس های حافظه...
# دستور CPEEKH

Var = CPEEKH(address)

با این دستور می توان بایت ذخیره شده در صفحه بالای حافظه کدی ( FLASH MEM) میکرو MEGA103 یا دیگر میکروها که دارای 128 K حافظه است را خواند .

ADDRESS آدرس حافظه و محتوای آدرس در متغیر یک بایت VAR قرار می گیرد .

Cpeek( 0) محتوای اولین بایت حافظه بالای 64 K را برمی گرداند.

# رجیسترها و آدرس های حافظه...
# دستور LOADADR

LOADADR var ,reg

با این دستور می توانید آدرس یک متغیر را در یک جفت رجیستر ذخیره کنید . Var متغیری است که آدرس آن در متغیرهای دوبایتی X,Y,Z ذخیره می شود و REG رجیسترهای X,Y, Z هستند .

این دستور جز دستورات اسمبلی است و برای کمک به برنامه نویسان اضافه شده است .

- **مثال**

```
Dim S As String ,A  As Byte
$asm
        Loadadr S , X          'load address into R26 and R27
        ld _temp1 , X          'load value of location R26/R27 into
                               'R24 ( _temp1)
$end asm
End
```

# رجیسترها و آدرس های حافظه...
# دستور OUT

OUT address , value

توسط این دستور می توان یک بایت به یک پورت سخت افزاری یا آدرس حافظه داخلی /خارجی ارسال کرد .

Value به آدرس address که می تواند بین 0H – FFFF H باشد فرستاده می شود . دستور OUT می تواند در تمام مکانهای حافظه AVR بنویسد . توجه کنید که برای address یک WORD تعریف می شود .

- **مثال**

```
Dim A As Byte
Out &H8000 , 1      ‘send 1 to the databus (d0 – d7) at address 8000
End
```

# رجیسترها و آدرس های حافظه...
# دستور INP

Var = INP (address)

توسط این دستور می توان یک بایت از پورت سخت افزاری یا آدرس حافظه داخلی /خارجی خواند .

محتوای آدرس addressکه می تواند بین 0H – FFFF H باشد خوانده شده و در متغیر var قرار می گیرد .دستور INPمی تواند از تمام مکانهای حافظه AVR بخواند .

- **مثال**

```
Dim A As Byte
A = INP (&H8000 )      ‘read value is placed on databus(d0 – d7) at
                       ‘address 8000
Print A
End
```




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# رجیسترها و آدرس های حافظه...
# دستور PEEK

Var = PEEK (address)

این دستور محتوای یک رجیستر را برمی گرداند .

Address آدرس رجیسترهای R0 – R7 است که بین 0 - 31 می باشد .محتوای رجیستر در متغیر var جای می گیرد . دستور ( ) PEEK فقط می تواند محتوای رجیسترها را بخواند ولی INP( ) می تواند از تمام مکانهای حافظه بخواند .

- **مثال**

```
Dim A As Byte
A = PEEK (0 )      ‘return the first byte of the internal memory (R0) End
```

# رجیسترها و آدرس های حافظه...
# دستور POKE

POKE address , value

با این دستور می توانیم یک بایت داده را در یکی از رجیسترها بنویسیم .

مقدار متغیر یا ثابت یک بایتی معتبر در آدرس address که بین 0 - 31 برای رجیسترهای R0 – R7 است نوشته می شود .

- **مثال**

```
Poke 1 , 5              'write 5 to R1
End
```

# رجیسترها و آدرس های حافظه...
# دستور VARPTR

Var = VARPTR (var2 )

این دستور آدرس یک متغیر را در مکان حافظه بر می گرداند .
آدرس متغیر var2 در مکان حافظه بدست آمده و در متغیر var قرار می گیرد .

⊡ **مثال**

```
Dim B As Xram Byte At &H300 , I As Integer , W As Word
W = Varptr (b)
Print Hex(W)                    'Print &H0300
End
```

# دستورالعملهای حلقه و پرش

میکروکنترلرهای AVR

# دستورالعملهای حلقه و پرش ...
# دستورالعمل JMP و GOTO

GOTO label
JMP label

با این دستورات می توان به برچسب label پرش کرد .برچسب label باید با علامت : (colon) پایان یابد و می تواند تا ۳۲ کارکتر طول داشته باشد . به خاطر داشته باشید زمانیکه از دو label هم نام استفاده شود کامپایلر به شما warning می دهد . دستور return برای برگشت از برچسب وجود ندارد .

- **مثال**

```
Start :                 ‘A label must  end with a colon
A = A +1                ‘Increment A
If A <10
Goto start               ‘Or Jmp start
End If                   ‘Close If
End
```

# دستورالعملهای حلقه و پرش ...
# دستورالعمل DO-LOOP

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

```
DO
  statements
LOOP [ UNTIL expression ]
```

دستورالعمل statement تا زمانی که expression دارای ارزش TRUE یا غیر صفر باشد تکرار خواهد شد .بنابراین این نوع حلقه حداقل یکبار تکرار می شود. DO –LOOP بتنهایی یک حلقه بینهایت است که با EXIT DO می توان از درون حلقه خارج شد و اجرای برنامه در خط بعد از حلقه ادامه یابد .

⊡ **مثال**

```
Dim A As Byte
Do                          ‘start the loop
A = A + 1                   ‘Increment A
Print A                     ‘Print It
Loop Until A = 10            ‘repeat until A = 10
Print A
```




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# دستورالعملهای حلقه و پرش ...
# دستورالعمل FOR-NEXT

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

```
FOR var = start TO end [STEP VALUE ]
  statements
NEXT var
```

Var بعنوان یک کانتر عمل می کند که start مقداراولیه 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**

```
Dim A As Byte , B1 As Byte , C As Integer
For A = 1 To 10 Step 2
      Print “this is a A “ ; A
Next A
For C = 10 To -5 Step -1
      Print “this is a C “ ; C
Next
For B1 = 1 To 10
      Print “this is a B1 “ ; B1
Next
```

# دستورالعملهای حلقه و پرش ...
# دستورالعمل WHILE-WEND

```
WHILE condition
  statements
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 حالتهایی اصلا اجرا نشود . بخش statement تا وقتی که حاصل condition صفر یا FALSE نشده است تکرار خواهد شد .

**مثال**

```
Dim A As Byte
A = 1
While A <10
        Print A
        Incr A
Wend
```

# دستورالعملهای حلقه و پرش ...
# دستورالعمل IF

در کلیه حالتهای زیر عبارت statement می تواند یک دستورالعمل ساده یا چند دستورالعمل مرکب باشد .

حالت 0:

```
If Expression THEN statement
```

دستورالعمل statement زمانی اجرا می شود که عبارت expression دارای ارزش TRUE باشد .

حالت 1:

```
If Expression Then
        statement1
Else
        statement2
End If
```

درصورتی که عبارت expression دارای ارزشTRUE باشد دستورالعمل statement1 اجرا خواهد شد , در غیر این صورت دستورالعمل statement2 اجرا می شود .

# دستورالعملهای حلقه و پرش ...
# دستورالعمل IF ...

حالت 2:

```
If Expression1 Then
        statement1
Elseif [Expression2 Then]
        statement2
Else
        statement3
End If
```

درصورتی که عبارت expression1 دارای ارزشTRUE باشد دستورالعمل statement1 اجرا خواهد شد ,در صورتی که عبارت expression1 دارای ارزش FALSE ولی عبارت اختیاری expression دارای ارزش TRUE باشد دستورالعمل statement2 اجرا می شود و در غیر این صورت دستورالعمل statement3 اجرا خواهد شد .

همچنین با دستور IF می توان صفر یا یک بودن یک بیت از یک متغیر را امتحان کرد .

```
IF bit =1 THEN     OR IF bit =0 THEN
```

# دستورالعملهای حلقه و پرش ...
# دستورالعمل IF ...

- مثال

```
Dim Var As Byte , Idx As Byte
Idx = 1
If Var.Idx = 1 then
Set portb.0
Else ....

Dim A As Integer
A = 10
If A = 10 then
        Print “this part is executed “
Else
        Print “ this will never be executed”
End if
```

# دستورالعملهای حلقه و پرش ...
# دستورالعمل CASE

اگر متغیر VAR با مقدار test1 برابر باشد statement1 اجرا می شود و سپس اجرای برنامه بعد از end selectادامه می یابد .
در غیر اینصورت اگر متغیر var با مقدار test1 برابر نباشد ولی با مقدار test2 برابر باشد statement2 اجرا می شود و سپس اجرای برنامه بعد از end select ادامه می یابد.
و نهایتا اگر متغیر var با هیچکدام از مقادیر test1 و test2 برابر نباشد ,statement3 اجرا می شود و سپس اجرای برنامه بعد از end select ادامه می یابد .

شما می توانید به صورتهای زیر نیز متغیر را امتحان کنید :
اگر متغیر موردنظر بزرگتر از ۲ باشد . Case is >2
و یا می توان محدوده ای را برای امتحان کردن در نظر گرفت :
اگر متغیر موردنظر بین ۲ تا ۵ باشد . Case 2 to 5

# دستورالعملهای حلقه و پرش ...
# دستورالعمل CASE....

- مثال

```
Dim X As Byte
Do
    Input " X?" , X
    Select Case X                                   'test X
        Case 1 To 3     :Print " 1 or  2   or  3"
        Case 4          :Print "4"
        Case Is >10     :Print "> 10"
        Case Else       :Print "no "
    End Select
Loop
End
```

# دستورالعملهای حلقه و پرش ...
# دستور EXIT

با این دستور می توانید فقط از یک ساختار یا حلقه خارج شوید و ادامه برنامه را بعد از ساختار یا حلقه ادامه دهید .

EXIT FOR
EXIT DO
EXIT WHILE
EXIT SUB
EXIT FUNCTION

- **مثال**

```
Do
        A = A +1
       If A = 100 Then
            Exit Do
      End If
Loop
End
```

# دستورالعملهای حلقه و پرش ...
# دستورالعمل ON VALUE

با این دستور با توجه به مقدار متغیر می توان به توابع یا برچسب های مختلفی پرش کرد .

ON var [GOTO] [GOSUB] label1 [,label2]

Var متغیر مورد نظر برای امتحان شدن که می تواند یک SFR مانند PORTD باشد و LABEL1 و LABEL2 و .. برچسب هایی هستند که با توجه به مقدار VAR به آنها پرش می شود .

- **مثال**

```
Dim X As Byte
X = 1
ON X Gosub Lbl2,Lbl3              'jump to sub lbl3
X=0
ON X Goto  Lbl1, Lbl4             'jump to label lbl1
Lbl1:
Incr X
Print X
Lbl2:
End
Print X
return
```




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# ایجاد تاخیر در برنامه

میکروکنترلرهای AVR

# ایجاد تاخیر در برنامه ...
# دستور DELAY

این دستور برای مدت کوتاهی به مقدار ۱۰۰۰ میکرو ثانیه در اجرای برنامه تاخیر ایجاد می کند .

- **مثال**

```
DELAY                    'Wait for hardware to be ready
```

# ایجاد تاخیر در برنامه ...
# دستور WAITus

برای ایجاد تاخیر در برنامه از این دستور می شود .

WAITus microsecond

اجرای برنامه به مدت microsecond میکرو ثانیه متوقف می شود .پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه می یابد. Microsecond می تواند عددی بین 1 – ( 255 باشد .

نکته

دستورات تاخیری زمان دقیق را به شما نمی دهد . برای بدست آوردن زمان دقیق از تایمرها استفاده کنید .

**مثال**

```
Waitus 10
Print “BASCOM”
End
```




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# ایجاد تاخیر در برنامه ...
# دستور WAITms

برای ایجاد تاخیر در برنامه از این دستور می شود .

WAITms milisecond

اجرای برنامه به مدت milisecond میلی ثانیه متوقف می شود .پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه می یابد. Milisecond می تواند عددی بین ( 65535- 1) باشد .

**مثال**

```
Waitms 10
Print “BASCOM”
End
```

# ایجاد تاخیر در برنامه ...
# دستور WAITus

برای ایجاد تاخیر در برنامه از این دستور می شود .

WAIT second

اجرای برنامه به مدت second ثانیه متوقف می شود .پس از سپری شدن زمان مشخص شده اجرای برنامه از خط بعد ادامه می یابد.

- **مثال**

```
Wait 3
Print “BASCOM”
End
```

# زیر برنامه و تابع

میکروکنترلرهای AVR

# زیر برنامه و تابع ...

## معرفی تابع 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 تغیری ایجاد نشود .ولی در حالت BYREF آدرس متغیر ارسال و تغیرات 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 وtype2 نوع داده برگشتی است . که هر دو می توانند داده نوع , BYTE , INTEGER, WORD LONG ,STRING باشند .

# زیر برنامه و تابع ...
# معرفی تابع DECLARE FUNCTION...

- **مثال**

در مثال زیر I بصورت BYVAL فرستاده شده است بنابراین یک کپی از مقدار I به زیر تابع فرستاده می شود و هیچ تغیری در محتوای آن ایجاد نمی شود. S بصورت BYREF فرستاده می شود و تغیر در آن صورت می گیرد .فراخوانی تابع MYFUNCTION با K وZ 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 .

# زیر برنامه و تابع ...
# معرفی تابع DECLARE FUNCTION...

- مثال

```
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 )
Lcd T                    ‘25
Lcd Z                    ‘Bascom
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
```

# زیر برنامه و تابع ...
# 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 برنمی گرداند .در زمان ارسال داده بصورت BYREF آدرس داده به زیر برنامه فرستاده می شود و در محتوای آن تغیر ایجاد می شود .ولی در حالت BYVAL یک کپی از داده فرستاده می شود و به هیچ وجه در محتوای آن تغیری ایجاد نمی شود . TEST نام زیربرنامه و VARنام 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 Subپایان می یابد .




www.IrPDF.com
www.IrPDF.com
www.IrPDF.com

# زیر برنامه و تابع ...
# معرفی زیربرنامه DECLARE SUB...

- مثال

```
Dim A As Byte , B1 As Byte , C As Byte
Declare Sub Test ( A As Byte )
A =1 : B1 = 2 : C = 3
Print  A ; B1;C                    ,123 will print
Call Test (B1)
Print  A ; B1;C                    ‘ 223 will print
End

Sub test (A As Byte )
        Print A ; B1 ; C          ‘123 will print
End Sub
```

# زیر برنامه و 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 .

# زیر برنامه و تابع ...
# فراخوانی CALL ...

- مثال

```
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
Print  A                    ‘ will print A = 10
End

Sub Test ( B1 As Byte , Byval B2 As Byte )
Lcd B1
LowerLine
Lcd BCD(B2)
B1 = 10
B2 = 15
End Sub
```

# زیر برنامه و تابع ...
# بکارگیری متغیر محلی یا LOCAL

از این دستور برای تعریف متغیر محلی در زیربرنامه 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 .


متغیرهای بیتی نمی توانند بصورت محلی تعریف شوند .

# زیر برنامه و تابع ...
# بکارگیری متغیر محلی یا LOCAL...

- مثال

```
Declare Sub Test2
Do
Call test2
Loop
End

Sub Test2
    Local A As Byte
    Incr A
    Lcd A
End Sub
```

# زیر برنامه و تابع ...
# بکارگیری متغیر محلی یا LOCAL

این دستور به زیربرنامه پرش می کند و اجرای برنامه را از آدرس برچسب ادامه می دهد .

GOSUB label

LABEL نام برچسبی زیر برنامه است که به آن پرش می شود .توسط دستور RETURN می توان از SUB برگشت کرد و اجرای برنامه بعد از دستور GO SUB ادامه می یابد .

- **مثال**

```
Dim X As Byte
Gosub Routine          'Jump to routine
Print " Hello"         'After come back from routine Print "Hello"
End

Routine
X = X + 2
Print X
Return
```

# پیکره بندی و کار با امکانات AVR در BASCOM

6

# پیکره بندی پورت ها

میکروکنترلرهای AVR

# پیکره بندی پورت ها ...

برای تعیین جهت پایه پورتها از این پیکره بندی استفاده می نماییم .جهت یک پایه می تواند ورودی یا خروجی باشد .

Config Portx = State

Config Pinx.y = State

X , Y بسته به میکرو می توانند به ترتیب پایه های 7- 0 پورت های F, E, D, C, B, A باشند.

State می تواند یکی از گزینه های زیر باشد :

INPUT یا 0 : در این حالت رجیستر جهت داده پایه یا پورت انتخاب شده صفر می شود و پایه یا پورت بعنوان ورودی استفاده می شود .

OUTPUT یا 1: در این حالت رجیستر جهت داده پایه یا پورت انتخاب شده یک مشود و پایه یا پورت بعنوان خروجی استفاده می شود .

زمانیکه بخواهید از پورتی بخوانید بایستی از رجیستر PIN پورت مربوطه استفاده کنید و در هنگام نوشتن در پورت بایستی در رجیستر PORT بنویسید .

# پیکره بندی پورت ها ...

- مثال

```
Dim A As Byte , Count  As Byte
Config Portd = input                    'configure PORT D for input mode
A = Pind                                'Read data on Portd
A = A And Portd                         'A = A & PORTD
Print A
Bitwait Pind.7 , reset                  'wait until bit is low
Config portb = output
Portb = 10                              'set portb to 10
Portb = Portb And 2
Set Portb.0                             'set bit 0 of portb to 1
Incr Portb
```

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

- **پورت A**

پورت A یک I/O دو طرفه ۸ بیتی است . سه آدرس از مکان حافظه I/O اختصاص به پورت A دارد .یک آدرس برای رجیستر داده PORTA , دومی رجیستر جهت داده DDRA و سومی پایه ورودی پورت A , PINA است .آدرس پایه های ورودی پورت A فقط قابل خواندن است در صورتی که رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . تمام پایه های پورت دارای مقاومت Pull up مجزا هستند .

PINA یک رجیستر نیست . این آدرس دسترسی به مقدار فیزیکی بر روی هر یک از پایه های پورت A را ممکن می سازد .زمانیکه PORTA خوانده می شود , داده لچ پورت A خوانده می شود و زمانیکه از PINA خوانده می شود مقدار منطقی که بر روی پایه ها موجود است خوانده می شود .

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

- **رجیسترهای پورت A**
  - رجیستر داده پورت A - PORTA [PORT A DATA REGISTER ]

| PORTA7 | PORTA6 | PORTA5 | PORTA4 | PORTA3 | PORTA2 | PORTA1 | PORTA0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

  - رجیستر جهت داده پورت A - DDRA [PORT A DATA DIRECTION REGISTER ]

| PORTA7 | PORTA6 | PORTA5 | PORTA4 | PORTA3 | PORTA2 | PORTA1 | PORTA0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

  - بایت آدرس پایه های ورودی پورت PINA- A [PORT A INPUT PINS ADDRESS ]

| PORTA7 | PORTA6 | PORTA5 | PORTA4 | PORTA3 | PORTA2 | PORTA1 | PORTA0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

- **استفاده از پورت A به عنوان یک I/O عمومی دیجیتال**

تمام ۸ پایه موجود زمانیکه بعنوان پایه های I/O دیجیتال استفاده می شوند دارای عملکرد مساوی هستند . Pan , پایه I/O عمومی : بیت DDAn در رجیستر DDRA مشخص کننده جهت پایه است . اگر DDAnیک باشد , PAn بعنوان یک پایه خروجی مورد استفاده قرار می گیرد و اگر DDAnصفر باشد , PAn بعنوان یک پایه ورودی در نظر گرفته می شود .اگر PORTAn یک باشد هنگامیکه پایه بعنوان ورودی تعریف می شود ,مقاوت Pull-up فعال می شود . برای خاموش کردن مقاومت باید PORTAn صفر شود یا اینکه پایه بعنوان خروجی تعریف شود .پایه های پورت زمانیکه ریست اتفاق می افتد به حالت Tri-state می روند .

| DDAn | PORTAn | I/O | Pull up | Comment |
|---|---|---|---|---|
| 0 | 0 | Input | No | Tri-state |
| 0 | 1 | Input | Yes | PAn will source current if ext. pull up low |
| 1 | 0 | Output | No | Push-Pull Zero output |
| 1 | 1 | Output | No | Push-Pull one output |

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

- **پورت B**

پورت B یک I/O دو طرفه ۸ بیتی است . سه آدرس از مکان حافظه I/O اختصاص به پورت B دارد .یک آدرس برای رجیستر داده PORTB , دومی رجیستر جهت داده DDRB و سومی پایه ورودی پورت B , PINB است .آدرس پایه های ورودی پورت B فقط قابل خواندن است در صورتی که رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . تمام پایه های پورت دارای مقاومت Pull up مجزا هستند .

PINB یک رجیستر نیست . این آدرس دسترسی به مقدار فیزیکی بر روی هر یک از پایه های پورت B را ممکن می سازد .زمانیکه PORTB خوانده می شود , داده لچ پورت B خوانده می شود و زمانیکه از PINB خوانده می شود مقدار منطقی که بر روی پایه ها موجود است خوانده می شود .

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

- **رجیسترهای پورت B**
  - رجیستر داده پورت B - PORTB [PORT B DATA REGISTER ]

| PORTB7 | PORTB6 | PORTB5 | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

  - رجیستر جهت داده پورت B - DDRB [PORT B DATA DIRECTION REGISTER ]

| PORTB7 | PORTB6 | PORTB5 | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

  - بایت آدرس پایه های ورودی پورت PINB- B [PORT B INPUT PINS ADDRESS ]

| PORTB7 | PORTB6 | PORTB5 | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

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

تمام ۸ پایه موجود زمانیکه بعنوان پایه های I/O دیجیتال استفاده می شوند دارای عملکرد مساوی هستند . Pbn , پایه I/O عمومی : بیت DDBn در رجیستر DDRB مشخص کننده جهت پایه است . اگر DDBnیک باشد , PBn بعنوان یک پایه خروجی مورد استفاده قرار می گیرد و اگر DDBnصفر باشد , PBn بعنوان یک پایه ورودی در نظر گرفته می شود .اگر PORTBn یک باشد هنگامیکه پایه بعنوان ورودی تعریف می شود ,مقاوت Pull-up فعال می شود . برای خاموش کردن مقاومت باید PORTBn صفر شود یا اینکه پایه بعنوان خروجی تعریف شود .پایه های پورت زمانیکه ریست اتفاق می افتد به حالت Tri-state می روند .

| DDBn | PORTBn | I/O | Pull up | Comment |
|---|---|---|---|---|
| 0 | 0 | Input | No | Tri-state |
| 0 | 1 | Input | Yes | PBn will source current if ext. pull up low |
| 1 | 0 | Output | No | Push-Pull Zero output |
| 1 | 1 | Output | No | Push-Pull one output |

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

- **پورت C**

پورت C یک I/O دو طرفه ۸ بیتی است . سه آدرس از مکان حافظه I/O اختصاص به پورت C دارد .یک آدرس برای رجیستر داده PORTC , دومی رجیستر جهت داده DDRB و سومی پایه ورودی پورت C , PINC است .آدرس پایه های ورودی پورت C فقط قابل خواندن است در صورتی که رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . تمام پایه های پورت دارای مقاومت Pull up مجزا هستند .

PINC یک رجیستر نیست . این آدرس دسترسی به مقدار فیزیکی بر روی هر یک از پایه های پورت Cرا ممکن می سازد .زمانیکه PORTC خوانده می شود , داده لچ پورت C خوانده می شود و زمانیکه از PINC خوانده می شود مقدار منطقی که بر روی پایه ها موجود است خوانده می شود .

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

⊡ **رجیسترهای پورت C**

❖ رجیستر داده پورت C - PORTC [PORT C DATA REGISTER ]

| PORTC7 | PORTC6 | PORTC5 | PORTC4 | PORTC3 | PORTC2 | PORTC1 | PORTC0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

❖ رجیستر جهت داده پورت C - DDRC [PORT C DATA DIRECTION REGISTER ]

| PORTC7 | PORTC6 | PORTC5 | PORTC4 | PORTC3 | PORTC2 | PORTC1 | PORTC0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

❖ بایت آدرس پایه های ورودی پورت [PORT C INPUT PINS ADDRESS ] PINC- C

| PORTC7 | PORTC6 | PORTC5 | PORTC4 | PORTC3 | PORTC2 | PORTC1 | PORTC0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

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

تمام ۸ پایه موجود زمانیکه بعنوان پایه های I/O دیجیتال استفاده می شوند دارای عملکرد مساوی هستند . PCn , پایه I/O عمومی : بیت DDCn در رجیستر DDRC مشخص کننده جهت پایه است . اگر DDCnیک باشد , PCn بعنوان یک پایه خروجی مورد استفاده قرار می گیرد و اگر DDCnصفر باشد , PCn بعنوان یک پایه ورودی در نظر گرفته می شود .اگر PORTCn یک باشد هنگامیکه پایه بعنوان ورودی تعریف می شود ,مقاوت Pull-up فعال می شود . برای خاموش کردن مقاومت باید PORTCn صفر شود یا اینکه پایه بعنوان خروجی تعریف شود .پایه های پورت زمانیکه ریست اتفاق می افتد به حالت Tri-state می روند .

| DDCn | PORTCn | I/O | Pull up | Comment |
|---|---|---|---|---|
| 0 | 0 | Input | No | Tri-state |
| 0 | 1 | Input | Yes | PCn will source current if ext. pull up low |
| 1 | 0 | Output | No | Push-Pull Zero output |
| 1 | 1 | Output | No | Push-Pull one output |

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

- **پورت D**

پورت D یک I/O دو طرفه ۸ بیتی است . سه آدرس از مکان حافظه I/O اختصاص به پورت D دارد .یک آدرس برای رجیستر داده PORTD , دومی رجیستر جهت داده DDRB و سومی پایه ورودی پورت D , PIND است .آدرس پایه های ورودی پورت D فقط قابل خواندن است در صورتی که رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . تمام پایه های پورت دارای مقاومت Pull up مجزا هستند .

PIND یک رجیستر نیست . این آدرس دسترسی به مقدار فیزیکی بر روی هر یک از پایه های پورت Dرا ممکن می سازد .زمانیکه PORTD خوانده می شود , داده لچ پورت D خوانده می شود و زمانیکه از PIND خوانده می شود مقدار منطقی که بر روی پایه ها موجود است خوانده می شود .

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

- **رجیسترهای پورت D**
  - رجیستر داده پورت D - PORTD [PORT D DATA REGISTER ]

| PORTD7 | PORTD6 | PORTD5 | PORTD4 | PORTD3 | PORTD2 | PORTD1 | PORTD0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

  - رجیستر جهت داده پورت D - DDRD [PORT D DATA DIREDTION REGISTER ]

| PORTD7 | PORTD6 | PORTD5 | PORTD4 | PORTD3 | PORTD2 | PORTD1 | PORTD0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

  - بایت آدرس پایه های ورودی پورت D- PIND [PORT D INPUT PINS ADDRESS ]

| PORTD7 | PORTD6 | PORTD5 | PORTD4 | PORTD3 | PORTD2 | PORTD1 | PORTD0 |
|---|---|---|---|---|---|---|---|
| R/W | R/W | R/W | R/W | R/W | R/ W | R/W | R/W |

# پیکره بندی پورت ها ...
# بررسی پورتهای میکرو ATMEGA32

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

تمام ۸ پایه موجود زمانیکه بعنوان پایه های I/O دیجیتال استفاده می شوند دارای عملکرد مساوی هستند . PDn , پایه I/O عمومی : بیت DDDn در رجیستر DDRD مشخص کننده جهت پایه است . اگر DDDnیک باشد , PDn بعنوان یک پایه خروجی مورد استفاده قرار می گیرد و اگر DDDnصفر باشد , PDn بعنوان یک پایه ورودی در نظر گرفته می شود .اگر PORTDn یک باشد هنگامیکه پایه بعنوان ورودی تعریف می شود ,مقاوت Pull-up فعال می شود . برای خاموش کردن مقاومت باید PORTDn صفر شود یا اینکه پایه بعنوان خروجی تعریف شود .پایه های پورت زمانیکه ریست اتفاق می افتد به حالت Tri-state می روند .

| DDDn | PORTDn | I/O | Pull up | Comment |
|---|---|---|---|---|
| 0 | 0 | Input | No | Tri-state |
| 0 | 1 | Input | Yes | PDn will source current if ext. pull up low |
| 1 | 0 | Output | No | Push-Pull Zero output |
| 1 | 1 | Output | No | Push-Pull one output |

# سایت تخصصی مهندسی رباتیک

# WWW.ROBOTICS-ENGINEERING.IR

پایان