

به نام خدا

آموزش AVR از صفر تا بینهایت !

www.forum.iranled.com

تهیه و جمع اوری : saeed450 – nex2004 از سایت ایران ویج

نویسنده : احمد فهیمی

قسمت اول آشنایی با میکروکنترلر AVR

سلام . برای اونایی که تا حالا با میکرو کار نکردن کمی مشکله که بعضی از مفاهیم رو یاد بگیرن برای همین من آشنایی مقدماتی با **avr** به زبان ساده در اوردم . راستش دیگه از این ساده تر به ذهنم نرسید . دوستانی که تمایل به یادگیری **avr** دارن باید بدونن که باید با میانی دیجیتال آشنایی داشته باشند حداقل اینکه با گیت های منطقی آشنایی داشته باشین برای آشنایی با دیجیتال کتاب میانی دیجیتال هنرستان یا کتاب طراحی دیجیتال نوشته موریس مانو خوبه. نوشته زیر مقدمه ای بر آشنایی با میکرو هست که گفتگوی بین دو دوست رو نشون میده که دارن درباره میکرو **avr** صحبت می کنن امیدوارم که مفید باشه.

1-سلام خوبی

2-سلام خوبم تو خوبی

1-اره خوبم . چکار می کنی کم پیدایی

2-اره یک چند وقتی دارم پروژه دانشگاه رو انجام می دم

1-چی هست این پروژت

2-هیچی ساخت یک دور سنج موتور با میکروکنترلر **AVR**

1-چی میکروکنترلر **avr**؟؟؟؟ میکرو کنترلر دیگه چیه

2-میکروکنترلر رو اگه بخام به زبون ساده بهت بگم یک کنترل کننده هستش که تقریبا هر چی ازش بخای برایت انجام میده

1-پس بگو یک رباته

2-نه منظورم هر کاری نبود ببین مثل یک کامپیوتر که بهش فرمان میدی اون هم انجام میده البته از نظر ظاهری که نگاهش کنی یک ایسی **40** پایه هستش

1-چی؟؟؟؟ مگه میشه یک کامپیوتر به اندازه یک ایسی چهل پایه باشه

2-نه خود کامپیوتر منظورم از کامپیوتر اینه که بهش فرمان می دی اونم انجام میده

1-بازم نفهمیدم بیشتر توضیح بده اصلا به چه دردی میخوره

2-بزار بیشتر برات توضیح بدم این یک ایسی هستش که ما به وسیله کامپیوتر برنامه نویسی می کنیم بعد برنامه رو می ریزیم توی میکرو . کاربردش هم خیلی زیاده مثلا باهاش می تونی ساعت . فرکانس متر. قفل رمز درست کنی

1-یکم بیشتر دربار خود میکرو توضیح بده

2-این میکرویی که بهت میگم یک خانواده هستش که انواع مختلفی داره که تشکیل شده از سه گروه **1-tiny**

2-90s 3-mega

1-خوب فرق این سه گروه باهم چیه مثلا **tiny** با **90s** یا با **mega** چه فرقی دارن

2-فرفشون توی امکاناتی که دارن هستش و همین فرق باعث شده که قیمت هاشونم باهم فرق کنه

1-منظورت از امکاناتشون چیه

2-ببین بزار بیشتر برات درباره خود میکرو توضیح بدم . از امکاناتی که این میکرو ها دارن میشه به تایمر -

کانتر - تولید موج **pwm** - حافظه ای که اطلاعات رو توی خودش حفظ کنه - سرعت بالای انجام دستورات -

I/O (یا همون ورودی خروجی خدمون) - مبدل آنالوگ به دیجیتال **adc** - مقایسه کننده آنالوگ و... . حالا که

با امکاناتش آشنا شدی همینو بهت بگم که بعضی از میکرو ها توی امکاناتشون و توی مقدار حافظه با هم فرق دارن برای همین قیمتهاشونم باهم فرق میکنه

1- خوب حالا کدوم از همه امکاناتش بیشتره

2- نوع **mega** از همه امکاناتش بیشتر و از نظر مقدار حافظه هم از همه حافظه اش بیشتره بعد نوع **s90** و در نهایت نوع **tiny**

1- این همه امکانات همه توی یک ایسی پس حتما باید یک صد هزار تومنی قیمتش باشه

2- نه اگه بهت بگم که بهترین نوع این میکرو که نوع **mega** هستش رو توی خود بازار ایران با سه هزار تومن می تونی بخری باورت نمی شه

1- چی فط سه هزار تومن اون هم بهترین نوعش اخه چطور ممکنه این همه امکانات فقط سه هزار تومن اون هم بهترین نوعش!!!!!!!!!!!!!!!!!!!!!!

2- خوب دیگه تکنولوژی دیگه

1- راستی گفتم یکی از امکاناتش حافظه هستش بگو ببینم اصلا این حافظه به چه دردی می خوره؟؟

2- یادت میاد که بهت گفته بودم که باید به این میکرو فرمان بدیم

1-اره

2- خوب این فرمان ها رو ما با یک نرم افزار مینویسیم

1- خوب چه ربطی داره به سوال من

2- یکم صبر داشته باش . این برنامه ای که مانوشتیم توی این حافظه قرار می گیره

1- بیشتر توضیح بده

2- حافظه میکرو **avr** دو تا هستش یکی حافظه بلند مدت که بهش می گن **eprom** و دومی هستش حافظه کوتاه مدت که بهش می گن **flash**

1- خوب فرقشون باهم چیه

2- فرقشون در اینه که در حافظه کوتاه مدت با قطع تغذیه (ولتاژ) میکرو اطلاعات از بین می ره ولی در حافظه بلند مدت با قطع شدن تغذیه میکرو اطلاعات از بین نمیره

1- اها فهمیدم . خوب یادم میاد بهم گفتمی که هر کاری ازش بخایم برامون انجام میده چطوری باید بهش بگیم که فلان کار رو انجام بده ؟

2- خوب بزار به صورت کلی بهت بگم که گیج نشی . ما برای کارهایی که از این میکرو می خایم باید بهش برنامه بدیم این برنامه توسط نرم افزار نوشته می شه

1- چه نرم افزاری؟

2- نرم افزارای مختلفی هستش که باهاش برای میکرو برنامه می نویسن برنامه هایی رو که من می شناسم

bascom –codvision و ... هستش که من با **bascom** کار می کنم. خوب حالا بزار ادامه مطلب رو بگم .

برنامه ای رو که ما می نویسیم زبان های مختلفی داره

1- منظورت انگلیسی یا فارسی هستش

2- نه بابا منظورم زبان های برنامه نویسی هستش که عبارتند از **basic-c** –اسمبلی

1- یعنی باید همه ای زبان های برنامه نویسی رو بلد باشی

2- نه بابا ناراحت نباش من خودم وقتی که می خاستم میکروکنترلر یاد بگیرم هیچ کدوم از این زبان ها رو یاد نداشتم . چقدر سوال می کنی منو از موضوع پرت کردی . هر کدوم از این زبان ها در یک نرم افزار خاص خودش

نوشته می شه مثلا اگه دوست داشتی برنامه رو با زبان بیسیک بنویسی باید با نرم افزار **basic** کار کنی اگه

خاستی با زبان **C** برنامه بنویسی باید با نرم افزار **codvision** برنامه بنویسی . خوب حالا وقتی که برنامه رو

نوشته باید برنامه رو بریزی توی میکرو . حتما می خای بررسی چطوری برنامه رو میریزن توی میکرو خوب بهت

می گم با یک پروگرامر ...

1-چی پروگرامر دیگه چیه؟

2-نترس پروگرامر وسیله ای هستش که میکرو رو روی اون قرار میدی ویک سر دیگش وصل می شه به پرینتر کامپیوتر بعد که به کامپیوتر نصب شد از طریق نرم افزار فرمان ارسال برنامه به داخل میکرو رو میدی مثلا در نرم افزار بیسکام که بعدا مفصلا بهت توضیح میدم با زدن کلید **F4** برنامه ریخته می شه داخل میکرو

1-ببین من نفهمیدم نقش این پروگرامر این وسط چیه

2-پروگرامر واسطه ای هست میان میکرو و کامپیوتر . خوب اخیه مرد حسابی برنامه ای که تو برای میکرو نوشتی باید بره توی میکرو یا نه . پس از روی هوا هم که نمی شه برنامه رو فرستاد پس میان یک پروگرامر وصل می کنند به کامپیوتر و میکرو رو روی این پروگرامر قرار میدن بعد از توی کامپیوتر فرمان ارسال برنامه رو می دن

1-ببین درست فهمیدم یا نه پروگرامر تنها نقشی که داره اینه که برنامه ای که ما توی کامپیوتر نوشیتیم رو بریزه توی میکروکنترلر

2-افرین

1-بازم سوال دارم

2-بگو

1-این امکاناتی رو که گفتمی میکرو داره چطوری می تونیم ازش استفاده کنیم

2-این کار رو موقع برنامه نویسی بهش می گیم . مثلا می خایم از کانتر میکرو (کانتر به معنی شمارنده) استفاده کنیم . باید در اول برنامه بنویسیم که از کانتر می خایم استفاده کنیم . به این کار در اصطلاح می گویند

config . البته بعدا درباره برنامه نویسیش - نحوه کار با نرم افزار - اتصال سخت افزاری میکرو برای کامل توضیح

می دم

1-یک مثال می تونی بزنی که کاربرد این میکرو و امکاناتش چیه؟؟؟

2-اره ولی قول بده که نترسی که برنامه شو بنویسم چون که بعدا درباره کل دستورات برنامه نویسی و نحوه کار با نرم افزار برات توضیح می دم

1-باشه

2-خوب گفتمی یک مثال بزنی منم یم مثال خیلی ساده میزنم . فرض کن که با دو تا میکروسویچ می خایم یک

led رو روشن خاموش کنیم . برای این کار اول باید برنامه توسط نرم افزار نوشته بشه که من در اینجا از زبان

بیسیک برای نوشتن برنامه استفاده میکنم :

کد:

```
"regfile = "8515DEF.DAT$
crystal = 8000000$
Config Porta = Output
Config Pinb.0 = Input
Config Pinb.1 = Input
Config Debounce = 25
Do
Debounce Pinb.0 , 1 , Sett , Sub
Debounce Pinb.0 , 1 , Reset , Sub
Loop
End
```

:Sett

Set Porta.0

Return

:Reset

Reset Porta.0

Return

- 1- خوب درباره این توضیح بده
- 2- صبر داشته باش آسیاب به نوبت گفتم که بعدا درباره تک تک دستورات و نحوه کار با نرم افزار برات توضیح می دم
- 1- قول می دی
- 2- قول می دم که تک تک دستورات رو با مثال زیاد برات توضیح بدم
- 1- خوب حالا من چکار کنم
- 2- فعلا برو نرم افزار بیسکام (bascom) رو نصب کن تا بعد
- 1- باشه پس فعلا تا بعد

پایان قسمت اول

قسمت دوم آموزشی

- 1- سلام چطوری
- 2- سلام ممنون تو چطوری
- 1- قربانت اقا این **avr** چی بوده ما نمی دونستیم
- 2- اره تازه کجاشودیدی بزار یکم راه بیفتی اونوقت می فهمی چی بوده
- 1- یادم می یاد بهم قول داده بودی که بهم میکرو یاد بدی
- 2- حتما به روی چشم . خوب حالا قرار بود از کجا شروع کنیم
- 1- یک لحظه وایستا اول به چندتا از سوالایی که برام پیش اومده جواب بده بعد شروع کن به یاد دادن
- 2- در خدمتم هر چی سوال داری بگو که منو یاد اون اولایی که می خاستم میکرو یاد بگیرم می ندازی
- 1- سوال اول اینکه اصلا ما چرا از میکروکنترلر استفاده می کنیم اصلا چه مزیتی داره و کجاها ازش استفاده میشه؟
- 2- خوب . همونطور که از اسمش پیداست میکرو کنترلر یعنی یک کنترل کننده . این کنترل کننده می تونه هر پدیده ای رو کنترل کنه فقط کافیه که بهش برنامه بدی (تکنولوژی دیگه) کنترل دقیق پدیده هایی همچون دما . فشار . نور . فاصله . و ... و کاربرد ان در بیشتر زمینه های رباتیک . ماشین های هوشمند و صنعت هستش
- 1- مثل **plc** ؟؟؟
- 2- بله با این تفاوت که میکرو کنترلر از نظر حجم و قیمت زمین تا آسمان با **plc** فرق می کند مثلا قیمت یک **plc** حدود هفتاد هزار تومنه اون هم با امکانات حداقل و ابعاد زیاد ولی میکروکنترلر با همه امکانات با قیمت **3000** تومان که ابعاد بسیار کمی هم داره در بازار ایران به وفور پیدا میشه . البته این تفاوت هیچ وقت از ارزش **plc** کم نخواهد کرد زیرا **plc** فقط در صنعت کاربرد دارد و یک المان صنعتی می باشد نسبت به میکروکنترلر بیشتر در صنعت استفاده می شه.
- 1- خوب یک سوال دیگه این میکروکنترلری که میگی سرعت اجرای فرمان هایی که باید اجرا کنه چقدر هستش
- 2- بستگی به فرکانس کاری میکرو داره . معمولا سرعت انجام دستورالعمل هایی که انجام می ده بین **8** تا **16** میلیون دستور العمل در ثانیه هستش

1- چی درست شنیدیم 8 تا 16 میلیون فرمان اونم فقط توی یک ثانیه ؟؟؟

2- اره درست شنیدی

1- خوب این 8 تا 16 میلیون فرمان یا به قول خودت دستورات عمل سرعتش به چی بستگی داره

2- مستقیما به فرکانش کاری میکرو بستگی داره

1- آخرین سوال بعد برو سراغ ادامه مطالب آموزشی

2- در خدمتم بگو

1- این امکاناتی که گفتی بعضی از میکرو ها دارن بعضی ها هم ندارن باید از کجا متوجه بشیم خودت حتما همه رو حفظ کردی

2- (با خنده) نه عزیز این میکرو ها هر کدومشون دارای یک دیتا شیت هستند که اطلاعاتی درباره میکرویی که می خای باهاش کار کنی بهت میده . اطلاعاتی از قبیل . ولتاژ کاری . فرکانس کاری . امکانات میکرو و...

1- ok حالا برو سر آموزش

2- خواهش می کنم به روی چشم . هزار یک سر فصل خیلی کلی برات بگم که چه چیزایی رو باید بهت بگم 1- اول باید با نرم افزار **bascom** آشنا بشی 2- و در اخر باید دستورات برنامه نویسی رو یاد بگیری

خوب اول می ریم سر نرم افزار **bascom** که خیلی ساده هستش . منم خیلی کلی برات نرم افزارو توضیح می دم بقیشو خودت یکم سیخ بزنی یاد می گیری

1- باشه هر طور که صلاح می دونی

2- اول هزار در باره این نرم افزار بیسکام (**bascom**) برات بگم . خوب من قبلا بهت گفتم کارهایی رو که ما از یک میکرو می خایم باید به صورت برنامه بهش بدیم . خوب حالا این برنامه رو باید با یک نرم افزاری نوشته بشه یا نه

1- بله

2- خوب این نرم افزار همین بود که بهت گفتم . ما به وسیله این نرم افزار خواسته هایی که از یک میکرو **AVR** داریم را به صورت برنامه نویسی اجرا می کنیم. خوب بهتره که زیاد طولش ندم و مستقیم برم سر نرم افزار. اول هزار یک مقدار از منو های کاری نرم **bascom1.11.7.4** برات توضیح بدم

File: اول روی گزینه **file** کلیک می کنی بعدش روی **NEW** کلیک می کنی . یک صفحه سفید باز میشه . این صفحه محیط برنامه نویسی هستش

منوی **EDIT** که چیز مهمی نداره

PROGRAM: با کلیک کردن روی این منو به اولین گزینه می رسیم که نوشته **COMPILE** . حتما با خودت می گی این **COMPILE** به چه دردی می خوره خوب الان من بهت می گم . گفتیم که ما در محیط برنامه نویسی برنامه مان را می نویسیم خوب ما بالخره باید بدانیم که این برنامه ای که نوشتیم دارای خطای نوشتاری هست یا نه خوب پس حالا فهمیدی به چه دردی می خوره .

1- نگفتی چطوری از گزینه **COMPILE** استفاده کنیم ؟؟

2- شما بعد ای که برنامه رو نوشتی باید از منوی **PROGRAM** گزینه **COMPILE** رو انتخاب کنی سپس به طور اتومات برنامه شما چک می شه اگه خطایی بود زیر نرم افزار می نویسه اگه نبود که هیچی در ضمن شما این کار رو می تونی با کلید میانبر **F7** هم انجام بدی

SIMULATE: این گزینه یک شبیه ساز هستش و برنامه ای رو که شما نوشتی رو برات شبیه سازی می کنه می کنه که من هیچ ازش خوشم نمی یاد چون دارای خطا هستش و بهتره کم ازش استفاده کنی . این کار رو می تونی

با کلید میانبر **F2** انجام بدی

SEND TO CHIP : شما بعد از این که برنامه رو توی محیط برنامه نویسی نوشتی و بعدش هم **COMPILE** کردی نیاز داری که برنامه رو بریزی داخلی میکرو این کار رو با کلیک روی گزینه **SEND TO CHIP** باید انجام بدی که خودش اتومات برنامه رو میریزه توی میکروکنترلر . این کار رو با زدن کلید **F4** هم می تونی انجام بدی

منو **TOOLS** : این منو باشه بعدها برات توضیح میدم چون فعلا باهاش سروکار نداری

منوی **OPTIONS**: با کلیک روی این منو اولین گزینه ای که به چشم می خوره **COMPILER** هستش که مهمه بقیه هم باهاش سروکار نداشتیم پس فکرتو مشغول بقیش نکن حرفه ای که شدی شاید بقیش برات کاربرد داشته باشه . خوب حالا این **COMPILER** رو که روش کلیک می کنی پنج گزینه دیگه ظاهر میشه . اولی نوشته **CHIP** : به معنی انتخاب میکرو هستش شما میکرویی رو که می خای ازش استفاده کنی رو توی ان قسمت انتخاب می کنی . گزینه بعدی **OUTPUT** هستش که باهاش کاری نداریم .گزینه بعدی **COMMUNICATION** هستش وقتی روی این گزینه کلیک می کنی یک پنجره باز می شه شما باید در قسمت **FERQUENCY** مقدار فرکانس کاری میکرو **AVR** رو بر حسب هر تزی اینجا وارد کنی

- 1- یک سوال از کجا بفهمم که فرکانس کاری میکرویی که ازش می خام استفاده کنم چنده؟؟
- 2- با خواندن دیتا شیت یا همان کاتالوگ خود میکرو کنترلر . گزینه بعدی **C12** هستش که با اونم کاری نداریم .

گزینه بعد **LCD** هستش

- 1- حتما باید این گزینه رو هم بی خیال شیم
- 2- برعکس گزینه **LCD** گزینه مهمی هستش . تو در این قسمت باید مشخص کنی که **LCD** باید به کدوم یکی از پایه های میکروکنترلر وصل بشه و اندازه **LCD** رو هم می تونی توی این قسمت مشخص کنی

- 1- خوب بیشتر توضیح بده در باره این گزینه
- 2- این گزینه باشه وقتی که **LCD** رو بهت توضیح دادم اونوقت این گزینه رو هم بهت یاد میدم

- 1- این **LCD** خیلی مشکله؟؟

- 2- نه اتفاقا بسیار آسون و کار کردن باهاش بسیار لذت بخشه. راستی یک گزینه دیگه توی منوی **TOOLS** هستش به نام **PROGRAMMER** . که مال انتخاب نوع پروگرامر هستش

این هم از قسمت های مهم نرم افزار **BSCOM**
اگه سوالی هست بگو

- 1- ما چرا باید از گزینه **COMPILE** استفاده کنیم
- 2- برای این که ببینیم برنامه ای رو که نوشتیم از لحاظ نوشتاری (لغت) مشکلی داره یا نه . اگه بعد از نوشتن برنامه **COMPILE** نکنی نرم افزار اجازه ریخت برنامه روی میکروکنترلر رو نمیده

- 1- اها گرفتم چی می گی
- 2- خوب حالا بریم سر اصل مطب که آشنایی با برنامه نویسی هستش که مهمترین بخشه پس خوب گوش کن

- 1- ای به چشم

- 2- پس بزار اول یک مقدمه ای بگم بعد . ببین برنامه ای رو که شما می خای برای میکروکنترلر توی محیط برنامه نویسی نرم افزار بنویسی به زبان های مختلف نوشته می شه و هر زبان برنامه نویسی نرم افزار مخصوص خودشو داره مثلا برای نوشتن برنامه به زبان بیسیک از نرم افزار **BASCOM** . برای نوشتن با زبان **C** از نرم افزار **CODVISION** استفاده می کنیم . که من بیسیک رو دوست دارم و بهت یاد می دم . چون زبان برنامه نویسی بسیار ساده ای هستش . خوب دیگه وقتشه بریم سر دستورات برنامه نویسی.

دستور **REGFILES**: اولین حرکتی که باید در محیط برنامه نویسی انجام بدی اینه که میکروکنترلری رو که می خای ازش استفاده کنی رو باید در محیط برنامه نویسی ذکر کنی مثلا تو یک میکرو کنترلر **ATMEGA16**

خریدی پس باید در محیط برنامه نویسی اینطور میکرو رو معرفی کنی

ATMEGA16 برای "REGFILE = "M16DEF.DAT\$

AT90S8535 برای "REGFILE = "8535DEF.DAT\$

ATTINY12 برای "REGFILE="AT12DEF.DAT\$

یک راه دیگه هم هست که میکرو رو از داخل تنظیمات خود نرم افزار معرفی می کنی

1- چطوری . یعنی دیگه نمی خاد دستور **REGFILES** رو بنویسیم

2- نه لازم نیست . برای این کار به منوی **OPTIONS** نرم افزار مراجعه می کنی سپس گزینه **COMPILR** و

بعد گزینه **CHIP** رو کلیک می کنی یک پنجره باز می شه که اول صفحه نوشته **CHIP** . که جلوش یک کادری

هستش که میکرو رو اونجا انتخاب می کنی سپس **OK** می کنی

1- پس همیشه در برنامه اولین کاری که باید بکنم اینه که میکروکنترلر رو معرفی کنم طبق روش بالا که گفتی نه ؟

2-اره کاملا درسته

دستور **CRYSTALS**: دومین گامی که بعد از معرفی میکرو باید انجام بدی اینه که فرکانس کاری میکرو رو برای

نرم افزار مشخص کنی که برای این کار از دستور **CRYSTAL** استفاده میکنی مثلا فرض کن فرکانس میکرو

MEGA16 تو **8** مگاهرتز هستش یعنی **8000000** هر تزی که با این دستور فرکانس کاری رو معرفی می کنی

البته به هر تزی :

CRYSTAL = 8000000\$

فرکانس کاری میکرو رو هم می تونی از داخل نرم افزار انتخاب کنی به این صورت که می روی داخل منوی

OPTIONS سپس گزینه **COMPILER** سپس گزینه **COMMUNICATION** رو کلیک می کنی یک

پنجره باز می شه در داخل این پنجره یک جایی نوشته **FERQUENCY** که جلوش شما باید فرکانس رو انتخاب

کنی و بعد **OK** کنی

END : در گام سوم پایان هر برنامه ای باید از دستور **END** استفاده کنی یعنی اتمام برنامه

1- خوب بگو ببینم که گام چهارم چی هستش

2- گام چهارمی وجود نداره خیط شدی . در واقع برنامه نویسی یعنی قسمت شیرین کار از اینجا شروع میشه.

خوب حالا بهت توضیح میدم که برای نوشتن یک برنامه باید چکار کرد اولین چیزی رو که باید بهت یاد بدم

متغیرها هستند . همونطور که از اسمش پیداست یعنی قابل تغییر هستش . انواع متغیرها عبارتند از: **1- BIT**

2- BYTE 3- WORD 4- INTEGER 5- LONG 6- SINGLE 7- STRING

BIT که یعنی **0** و **1**

BYTE تشکیل شده از هشت بیت و هر **0** **BYTE** تا **255** مقدار دارد

WORD تشکیل شده از دو **BYTE** و هر **0** **WORD** تا **65535** مقدار دارد

با بقیه متغیرها هم فعلا کاری نداریم

1- خوب درست کارش چی هست و کجا کاربرد داره

2- بزار وقتی که یکم رفتیم جلوتر کم کم می فهمی خوب ادامه مطلب رو گوش کن

دستور **DIM : DIM** به معنی معرفی اسم متغیر است که می تونه هر اسمی باشه مثلا : **A** یا **B** یا **ALI** یا

TEMO حتی اسم خودت فقط برای اینه که متغیر رو یک نامی بهش بدی که با بقیه متغیرها قاطی نشه

دستور **AS : AS** به معنی نوع متغیر هستش که باید یکی از انواع متغیرهای **BIT** یا **BYTE** یا **INTEGER**

باشه

مثال: **DIM A AS BYTE**

DIM B AS BIT
DIM ALI AS BYTE
DIM N AS SINGLE

1- یک مثال مفهومی تر بزن

2- باشه مثلا در **DIM A AS BYTE** . به این معنی که متغییری که نامش هست **A** از نوع بایت (**BYTE**) می باشد

DIM ALI AS BIT . یعنی متغییری که نامش هست **ALI** از نوع بیت **BIT** می باشد .

1- حالا فهمیدم

2- بزار بریم جلوتر مثال روز برات روشن میشه که این دستورات کجا کاربرد داره
در ضمن شما گاهی لازم است که به متغییر مقدار هم بدی که به ترتیب زیر عمل می کنی
مثلا شما یک متغییر داری از نوع بایت

DIM S AS BYTE

که می تونی از **0** تا **255** بهش مقدار بدی به ترتیب زیر

S = 12 یا **S = 0** یا **S = 255** یا هر رقمی که از **0** تا **255** دوست داشتی البته اگه لازم بود
دستور **INCR** : با نوشتن این دستو شما می توانید یک متغییر را افزایش بدی به مثال زیر توجه کن بهتر می

فهمی

DIM B AS BYTE

B=0

INCR B

بزار از خط اول برات توضیح بدم برنامه ای که ما نوشتیم رو میکرو میاد از خط اول شروع می کنه به خوندن خط

اول یعنی ما یک متغییری داریم به نام **B** و از نوع **BYTE**

خط دوم به متغییر **B** مقدار دادیم

خط سوم دستور دادیم که متغیر **B** رو یک واحد افزایش بده وقتی که میکرو به خط سوم رسید و اونو خوند اگه

متغییر ما بوده **0** الان میشه **1**

1- کاربردش چیه این دستور **INCR**

2- کاربردش توی شمارنده هستش

دستور **DECR** : این دستور برعکس دستور قبل هستش و یک واحد از متغییر ما کم می کنه

مثال :

DIM U AS BYTE

U = 20

DECR U

خط اول یعنی ما یک متغییری داریم که اسمش هست **U** و از نوع بایت

خط دوم به **U** مقدار دادیم

خط سوم یک واحد از مقدار **U** کم کردیم

بزار یک مثال کاربردی تر بزنم :

"REGFILE = "M16DEF.DAT\$

CRYSTAL = 8000000\$

DIM K AS BYTE

K = 100

INCR K

DECR K

END

خوب حال خوب گوش کن

خط اول ما اومدیم نوع میکرو مونو مشخص کردیم که هست مگا **16**

خط دوم فرکانس کاری میکرو رو مشخص کردم
خط سوم گفتیم که ما یک معییری داریم به نام **K** و از نوع بایت
خط چهارم برای متغییر بایت یک مقدار دادیم
خط پنجم دستور دادیم که یک واحد به متغییر **K** اضافه کن یعنی اگه **100** بوده الان می شه **101**
خط ششم دستور دادیم که یک واحد از متغییر **K** کم کن یعنی الان که هست **101** یک واحد که کم بشه میشه
100
خط هفتم دستور پایان برنامه رو دادیم
1- سوال دارم اگه مثلا در مثال بالا دستور INCR و یا DECR رو دو بار زیر هم بنویسیم چی می شه
2- برای پاسخ به سوال به یک مثال دیگه توجه کن

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 800000$
DIM K AS BYTE
K = 100
INCR K
DECR K
DECR K
DECR K
INCR K
END
```

خوب حال خوب گوش کن
خط اول ما اومدیم نوع میکرو مونو مشخص کردیم که هست مگا**16**
خط دوم فرکانس کاری میکرو رو مشخص کردم
خط سوم گفتیم که ما یک معییری داریم به نام **K** و از نوع بایت
خط چهارم برای متغییر بایت یک مقدار دادیم
خط پنجم دستور دادیم که یک واحد به متغییر **K** اضافه کن یعنی اگه **100** بوده الان می شه **101**
خط ششم دستور دادیم که یک واحد از متغییر **K** کم کن یعنی الان که هست **101** یک واحد که کم بشه میشه
100
خط هفتم دوباره دستور دادیم که یک واحد از متغییر **K** کم کن خوب متغییر **K** که الان هستش **100** میشه **99**
خط هشتم دوباره دستور دادیم که یک واحد از متغییر **K** کم کن خوب متغییر **K** که **99** بوده میشه الان **98**
1- فهمیدم بزار خط نهم و دهم رو خودم بگم ببینم یاد گرفتم یا نه
خط نهم یعنی خط نهم دستور دادیم یک واحد به متغییر **K** اضافه بشه یعنی الان که **K** هستش **98** بعد از خونده
شدن دستور خط نهم میشه **99** و خط دهم یعنی اتمام برنامه
2- احسنست افرین داری راه میافتی ها
1- دستور بعدی چیه
2- دستور بعدی در مورد LCD هستش . بزار اول در مورد خود LCD برات بگم بعد دستوراتش رو با هم مرور می کنیم
LCD ها انواع مختلفی دارند اولی گرافیکی - دومی کارکتری که من با کارکتری کار می کنم شبیه **lcd** ماشین
حساب هستش که از نظر ابعاد هم باهم فرق دارن مثلا **16** در **1 - 16** در **2 - 16** در **4 - 40** در **4** و...
1- خوب مثلا این شماره ها برای چیه مثلا همین 16 در 2
2- عدد اول یعنی 16 به ما میگه که این lcd ما 16 ردیف (افقی) داره عدد دوم یعنی 2 به ما می گه این lcd دو ستون (عمودی) داره

1- حالا گرفتم

2- این lcd ها 14 پایه دارند که ما فقط از 9 پایه ان استفاده می کنیم که برای استفاده از ان و وصل ان به

میکروکنترلر به ترتیب زیر عمل می کنیم

پایه اول **vss** نام دارد که به زمین وصل میشه

پایه دوم **vdd** هستش که به ولتاژ 5 ولت وصل میشه

پایه سوم **vee** هستش که به زمین وصل میشه

پایه چهارم **rs** هستش که به میکرو وصل می شه . بعدا بهت می گم به کجای میکرو وصل میشه

پایه پنجم **rw** هستش که اونم به زمین وصل میشه

پایه ششم **E** هستش که اونم به میکرو وصل می شه

پایه های 10 9 8 7 که هیچی به جایی وصل نمیشن

پایه 11 **DB4** اسمش هست اونم به میکرو وصل میشه

پایه 12 **DB5** اسمش هست اونم به میکرو وصل میشه

پایه 13 **DB6** اسمش هست اونم به میکرو وصل میشه

پایه 14 **DB7** اسمش هست اونم به میکرو وصل میشه

1- یک سوال بعضی از پایه های بالا رو که معرفی کردی گفتمی وصل میشه به **LCD** ولی نگفتمی به کدوم پایه

LCD باید وصل بشه

2- بله نگفتم ولی اگر یکم صبر کنی بهت میگم . اگه یادت باشه وقتی که داشتیم نرم افزار **BASCOM** رو برات

توضیح می دادم گفتم که توی منوی **OPTINS** بعد **COMPILER** یک گزینه هست به نام **LCD** شما اونجا

مشخص می کنی که پایه های میکروکنترلر به کدوم پایه **LCD** وصل بشه که باید به ترتیب زیر عمل کنی بعد از

این که وارد گزینه **LCD** توی نرم افزار شدی یک پنجره باز میشه که سمت راست اون شش کادر هستش که شما

باید اونجا مشخص کنی که **LCD** به کدوم پایه های میکرو وصل بشه . می دانیم که اکثر میکرو ها دارای چهار

PORT هستند البته به غیر از خانواده **TINY**

1- پورت چیه واز؟؟

2- به هر هشت پایه میکرو یک پورت میگن که هر پورت باز خودش یک اسم داره مثلا **ATMEGA32** دارای

چهار پورت هستش به نام های **D -C -B -A** هر کدو از این پورت ها هشت پایه هستند

1- از کجا بفهمیم که مثلا پورت **A** یا پورت **C** کجاست

2- باز هم از روی دیتاشیت خود ایسی

1- خوب ادامه بده

2- هنگامی که شما می خای **LCD** رو به میکرو وصل کنی باید در گزینه **LCD** یک پورت رو به دلخواه انتخاب کنی

1- فرقی نمیکنه که کدوم پورت باشه

2- نه هر کدو از پورت ها که دوست داشته مثلا پورت **A** یا پورت **D**

1- اها خوب ادامه بده

2- مثلا حال کردی که **LCD** رو به پورت **C** وصل کنی برای این کار در نرم افزار وقتی که به قسمت **LCD** رفتی و

یک پنجره باز شد در قسمت راست پنجره شش گزینه برای انتخاب وجود داره که باید به ترتیب زیر عمل کنی :

اولین گزینه از بالا نوشته **ENABLE** که باید **PORTC.5** رو انتخاب کنی

دومین گزینه نوشته **RS** که باید **PORTC.4** رو انتخاب کنی

سومین گزینه نوشته **DB7** که باید **PORTC.3** رو انتخاب کنی

چهارمین گزینه نوشته **DB6** که باید **PORTC.2** رو انتخاب کنی

پنجمی گزینه نوشته **DB5** که باید **PORTC.1** رو انتخاب کنی

ششمین گزینه نوشته **DB4** که باید **PORTC.0** رو انتخاب کنی

حال فهمیدی کدوم پایه های میکرو به **LCD** وصل میشه

- 1- اها پس سخت افزاری هم باید به ترتیبی که مثل بالا مشخص کردیم **LCD** رو به میکروکنترلر وصل کنیم
- 2- کاملا درسته. شما مختاری که هر **PORT** که دوست داشتی رو به میکرو وصل کنی ولی باید هر پورته که توی ذهننت در نظر داری رو توی نرم افزار مثل شکل بالا انتخاب کنی

یک نکته دیگه اینکه در قسمت سمت چپ همین پنجره **LCD** یک گزینه هست به نام **LCD TYPE** که شما اونجا اندازه **LCD** تو مشخص می کنی مثلا یک **LCD** رفتی خریدی که اندازه هست **16** در **2** توی این گزینه باید **16*2** رو انتخاب کنی بعدی **OK** رو بزنی . خوب حال بریم سر دستورات **LCD** :

دستور **CLS** : این دستور کل **LCD** رو پاک و آماده نوشتن می کنه که همیشه باید قبل از دستور نوشت روی **LCD** این دستور نوشته بشه

دستور **LCD** : خوب با این دستور شما می تونی یک عبارت رو روی **LCD** نمایش بدی مثلا یک عدد یا یک اسم یا یک کلمه یا یک متغییر . البته قبل این دستور باید حتما **CLS** نوشته بشه این مثال توجه کن

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 8000000$
CLS
" LCD " ALI
END
```

دو خط اول رو که می دونی چی هستش. در خط سوم دستور پاک کردن **LCD** رو با دستور **CLS** دادیم در خط چهارم یعنی اینکه روی صفحه **LCD** بنویس **ALI** . البته به جای **ALI** هر چیز دیگه ای هم می شه نوشت از قبیل عدد . کلمه جمله

مثل مثال زیر

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 8000000$
CLS
" LCD " 999
END
```

مثل مثال قبل

1- اها فهمیدم

2- یادت میاد که بهت گفتم نرم افزار بیسکام یک جایی داره به نام شبیه ساز

1-اره چطور مگه

2- خوب تو می تونی اونجا این دستوراتی رو که امروز یادت دادم رو اونجا شبیه سازی کنی

1- تو که گفتی از این شبیه سازش خوشت نمی یاد

2- اره منظورم این بود که توی کار های بزرگ همیشه ازش استفاده کرد ولی توی کارهای کوچیک همیشه ازش استفاده کرد

1- حالا چی کار می کنه این شبیه ساز نرم افزار

2- برنامه ای رو که تو نوشتی برات شبیه سازی می کنه داخلش یک **LCD** هم داره برو حال کن

1- چطوری ازش استفاده کنم

2- بعد از این که تو برنامه رو نوشتی باید برنامتو از نظر نداشتن خطا چک کنی

1- چطوری

2- با زدن کلید **F7** . بعد اگه برنامت اشکالی نداشت کلید **F2** رو می زنی و وارد محیط شبیه ساز یا همون

SIMULATOR میشی

1- اینطوری گیج میشم یک مثال هم بزن

2- باشه به روی چشم فرض کن که ما یکی از برنامه های بالا رو نوشتیم مثل برنامه زیر:

```
"REGFILE = "M16DEF.DAT$  
CRYSTAL = 8000000$  
CLS  
" LCD " ALI  
END
```

درضمن تنظیمات روهم مثل بالا که گفتم پایه های LCD رو توی نرم افزار انتخاب کن رو مثل بالا انتخاب می کنی بعد دکمه F7 رو بزن تا برنامه چک بشه بعد دکمه F2 رو بزن تا وارد محیط شبیه ساز بشی . توی محیط شبیه ساز خیلی دکمه هستش که من فقط چند تارو که کاربرد داره برات میگم دکمه اوایل از بالا سمت چپ شبیه دکمه PLAY ویدو هست که اسمش هست RUN PROGRAM که برای اجرای برنامه شبیه ساز هستش دکمه بعدی که هچی دکمه بقلیش شبیه دکمه استپ ویدو هست که اسمش هست STOP PROGRAM که برای توقف شبیه سازی هستش چند تا دکمه اون طرفتر یک دکمه که داخلش یک مستطیل ابی رنگی هستش و روش نوشته LCD . این دکمه رو وقتی که میزنی یک LCD توشه خوب ادامه حالا که وارد محیط شبیه ساز شدی اول برای شبیه سازی مثال بالا دکمه LCD رو انتخاب می کنی بعد دکمه RUN PROGRAM رو میزنی باید یک ده پانزده ثانیه صبر می کنی تا روی LCD نوشته بشه ALI برای امروزدیگه بسه باشه بقیش برای یک فرصت دیگه فعلا بای

قسمت سوم آموزشی

1-سلام

2-سلام خوبی

1-قربانت

2-چه خبر اون چیزایی رو که بهت گفته بودم رو کار کردی مثال حل کردی ازش

1-اره . کار کردم اونم از همش . اقا یک سوال اگه بخایم مقدار یک متغییر رو روی LCD نمایش بدیم باید چطوری

عمل کنیم

2-این که خیلی اسونه با چند تا مثال جواب سواتو میدم

مثال 1:

```
"REGFILE = "M16DEF.DAT$  
CRYSTAL = 8000000$  
DIM A AS BYTE  
A = 3  
CLS  
LCD A  
END
```

1-خوب تحلیلش کن

2-باشه

خط اول که معرفی میکرو هستش

خط دوم فرکانس کاری میکرو هستش

خط سوم ما یک متغییر را که نامش هست **A** و از نوع بایت می باشد را تعریف کردیم

خط چهارم به متغییر مقدار دادیم

خط پنجم با دستور **CLS** ال سی دی را برای نوشتن جدید پاک کردیم

خط ششم دستور داریم که مقداری را که به متغییر **A** داده ایم روی **LCD** نمایش داده شود

خط هفتم با دستور **END** برنامه را به پایان رساندیم.

1- اها یعنی به متغییر **A** هر مقداری که داده باشیم مثلا **1** یا **0** یا **1001** روی **LCD** نمایش داده می شود

2- کاملا درسته واما مثال دوم: این دفعه می خایم دو تا متغییر رو یکی پس از دیگری روی **LCD** نمایش بدیم

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 8000000$
DIM ALI AS BIT
DIM IRAN AS BYTE
ALI = 0
IRAN = 224
CLS
LCD ALI
CLS
LCD IRAN
END
```

خط اول که معرفی خود میکرو

خط دوم معرفی فرکانس کاری میکرو

خط سوم اومدیم یک متغییر از نوع **BIT** که نامش است **ALI** رو معرفی کردیم

خط چهارم هم اومدیم یک متغییر از نوع بایت که نامش هست **IRAN** رو معرفی کردیم

خط پنجم به متغییر **BIT** که نامش بود **ALI** یک مقدار دادیم

خط ششم هم مثل خط پنجم

خط هفتم با دستور **CLS** ال سی دی را برای نوشتن آماده کردیم

خط هشتم به میکرو دستور دادیم که مقدار متغییر **ALI** رو روی **LCD** نمایش بدهد

خط نهم دوباره با دستور **CLS** ال سی دی رو برای نوشتن دوباره پاک کردیم

خط دهم هم به میکرو دستور دادیم که مقدار متغییر **IRAN** رو روی ال سی دی نمایش بده

و در خط آخر نیز با دستور **END** برنامه رو به اتمام رساندیم

2- مثال اخر :

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 8000000$
DIM G AS WORD
G = 12
CLS
LCD G
INCR G
CLS
LCD G
END
```

خط اول و دوم رو که دیگه تکراری شده که بخام توضیح بدم

خط سوم هم تقریبا مثل قبل اومدیم یک متغییر که نامش هست **G** و از نوع **WORD** می باشد را برای میکرو

تعریف کردیم

خط چهارم به متغییر **G** یک مقدار دلخواه دادیم
خط پنجم ال سی دی را با دستور **CLS** برای آماده شدن پاک کردیم
خط ششم با دستور **LCD G** مقدار فعلی متغییر **G** را که هست **12** را روی **LCD** نمایش دادیم
خط هفتم به متغییر **G** یک واحد اضافه کردیم یعنی شد **13**
خط هشتم با دستور **CLS** ال سی دی را برای نوشتن دوباره پاک کردیم
خط نهم به میکرو دستور دادیم مقدار متغییر **G** را که الان به آن یک واحد اضافه شده است یعنی شده **13** رو
روی ال سی دی نمایش بده

خط دهم هم یعنی پایان برنامه

1- همه این مثالایی رو که گفتی توی محیط شبیه سازی قابل اجراست

2-اره قابل اجراست و میتونی اونجا همه این مثال های رو مشاهده کنی.

خوب حال بریم سر مبحث بعدی

1- دستورات **LCD** همینا بود

2- نه هنوز چند تا دیگه هست که به موقش میگم

1- خوب برو سر مبحث بعدی

2- باشه . ببین گاهی وقتا لازم هستش که در بین برنامه هایی که داریم می نویسیم یک تاخیر ایجاد کنیم که این تاخیر می تونه یک ثانیه . ده ثانیه . یک میلی ثانیه . یک میکرو ثانیه یا هر تایمی که عشقت بود رو تاخیر ایجاد کنی

1- همین جا وایستا که یک سوال دارم . این تاخیر به چه درد می خوره لطفا فقط یک مثال کاربردی هم بزن

2- ببین مثلا شما می خای یک **LED** رو روشن کنی و می خای این **LED** بعد از مثلا ده ثانیه خاموش بشه خوب

حالا ارزش این دستور باید حس کنی

1- اها فهمیدم حالا ادامه بده

2- تاخیراتی را که ما میتوانیم ایجاد کنیم بر حسب ثانیه - میلی ثانیه - میکرو ثانیه می باشد

1 WAIT تاخیر برای مدت زمان یک ثانیه

1 WAITMS تاخیر برای مدت زمان یک میلی ثانیه

1 WAITUS تاخیر برای مدت زمان یک میکرو ثانیه

متاسفانه این مدت زمان های تاخیر زیاد هم دقیق نیستند مثلا شما دستور میدی که یک ثانیه تاخیر داشته باش

میکرو میاد دوازده ثانیه تاخیر ایجاد میکنه

1- خوب پس چاره چیه؟؟

2- برای حل این مشکل شما باید از میلی ثانیه یا میکرو ثانیه بیشتر استفاده کنی مثال اگه می خای یک تاخیر

یک ثایه ایجاد کنی باید بنویسی **WAITMS 30** البته این یک مثال بود که اگه یکم با این دستورات ور بری

همش دستت میاد

خوب حالا بریم سر مثال ها

مثال اول:

```
"REGFILE = "M16DEF.DAT$  
CRYSTAL = 8000000$  
DIM A AS BYTE  
A = 112  
CLS  
LCD A  
INCR A  
WAITMS 500  
CLS
```

LCD A END

حالا تحلیل برنامه

خط اول و دوم که خودت میدونی

خط سوم هم که باید بدونی

خط چهارم هم باز باید بدونی ولی می گم که دلت نشکنه در خط چهارم ما به متغیر **A** یک مقدار دادیم

خط پنجم با دستور **CLS** ال سی دی را برای نوشتن و آماده شدن پاک کردیم

خط ششم با دستور **LCD A** مقدار متغییر **A** را که هست **112** روی ال سی دی نمایش دادیم

در خط هفتم با دستور **INCR A** یک واحد به متغییر **A** اضافه کردیم یعنی بود **112** حالا شد **113**

در خط هشتم با دستور **WAITMS 500** یک تاخیر پانصد میلی ثانیه ای ایجاد کردیم یعنی اینکه برنامه فعلا

هیمنجا متوقف است و بعد از **500** میلی ثانیه برنامه از خط بعدی خوانده می شود

در خط نهم با دستور **CLS** ال سی دی را پاک کردیم

در خط دهم مقدار جدید متغییر **A** را که الان هست **113** روی ال سی دی نمایش دادیم

در خط آخر هم با دستور **END** برنامه رو پایان دادیم

```
"REGFILE = "M16DEF.DAT$  
CRYSTAL = 800000$  
DIM S AS BYTE  
S = 112  
CLS  
LCD S  
INCR S  
WAITMS 500  
CLS  
WAITMS 100  
LCD S  
END
```

از خط اول تا خط هشتم

خط نهم ال سی دی رو پاک کردیم

خط دهم یک تاخیر در برنامه با مدت زمان **100** میلی ثانیه ایجاد کردیم و بعد از اینکه صد میلی ثانیه گذشت

برنامه از خط بعدی ادامه پیدا می کنه

خط یازدهم دستور دادیم که مقدار جدید **S** را که **113** هستش رو روی ال سی دی نمایش بده

خط آخر هم پایان برنامه

1 - اینم توی محیط شبیه ساز قابل اجراست

2 - بله حتما. موافقی که بریم سر مبحث بعدی

1 - اره بریم

2 - خوب حالا رسیدیم به قسمت های جون دارتر میکرو. اگه یادت باشه اون اوایل بهت گفته بودم که

میکروکنترلر دارای یک سری امکانات هستش مثل تایمر - کانتر - مبدل انالوگ به دیجیتال همچنین گفتم که

یکی از امکاناتش ورودی خروجی یا همون **I/O** هستش که کاربردی فراوان داره. مثلا در ساده ترین حالت شما با

استفاده از این امکان می تونی یک **LED** رو روشن یا خاموش یا هر دو حالت رو انجام بدی

1 - **I/O** چی هست

2 - **I** یعنی ورودی **O** هم یعنی خروجی البت هر دو از نوع دیجیتال هستند

- 1- ها فهمیدم . همیشه یک مثال دیگه از کاربردش بزنی
- 2- چراکه نه حتما. فرض کن می خای با یک میکرو سویچ (یک نوع کلید) یک موتور رو روشن خاموش کنی و حتما باید با یک عدد میکروسویچ یک بار که میزنی روشن و دفعه بعد که میزنی خاموش شود

1- میکرو سویچ همون شسی هستش ؟

2- اره میکروسویچ همون کلید زنگ در خونتون هستش

2- در کل برای استفاده از امکانات میکروکنترلر باید از روش پیکره بندی استفاده شود که بهش می گن **CONFIG** یعنی شما با این کار به میکروکنترلر اعلام می کنی که می خاهم از فلان امکانات استفاده کنم . بزار

اول یک سری دستورات رو لیست وار معرفی کنم بعد یکی یکی اونها رو توضیح بدم

PORT

PIN

SET

RESET

TOGGLE

خوب حالا درباره هر کدوم توضیح میدم

PORT به معنی خروجی هستش هر وقت که خاستی از میکرو یک خروجی بگیري از این دستور باید استفاده کنی . مثلا یک **LED** رو می خای روشن کنی یا هر چیز دیگه

PIN به معنی ورودی هستش هر وقت که خاستی یه میکروکنترلر یک ورودی بدی باید از این دستور استفاده کنی مثلا می خای یک کلید به ورودی وصل کنی البته این دستور **PIN** و **PORT** به تنهایی کاربرد ندارن که در ادامه میگویم که باید چیکار کنی

SET که همون یک دیجیتال خودمونه

RESET که همون صفر دیجیتال خودمونه

TOGGLE یعنی عکس حالت فعلی یعنی اگر قبلا یک بوده حالا صفر میشه اگه قبلا صفر بوده حالا یک میشه

بزار قبل از اینکه توضیحاتمو کامل کنم یک سری اطلاعات درباره پایه های میکرو بدم . ببین هر میکرو بسته به نوعش یک تعداد پایه داره که از **8** پایش هست تا **40** پایه که هر هشت پایه رو یک پورت می نامند البته ربطی به اون پورتی که توی دستورات بهت گفتم نداره هر پورت رو که هشت پایه هستش رو با یک اسم نام گذاری می کنن که استاندارد هستش به نام های پورت **A** پورت **B** پورت **C** پورت **D** میکرو ها حداکثر چهار پورت ورودی خروجی دارن. در ضمن هر کدام از پورت ها می توانند ورودی یا خروجی باشند که بستگی به انتخاب برنامه نویس داره علامت پورت ها به ترتیب زیر هستش:

A,0

A.1

A.2

A.3

A.4

A.5

A.6

A.7

برای پورت های **C B** و **D** هم مثل بالا هستش

1- یک سوال . اگه بخاد حداکثر چهار پورت داشته باشه و هر پورت هم که هشتا پایه هستش پس در کل همیشه

32 پایه پس چرا بعضی از میکرو ها **40** پایه هستند

2- خوب **32** پایه هستش با به عبارتی چهار پورت بقیش مال **VCC GND** و... هستش

1- اها فهمیدم ادامه بده

2- خوب گفتیم که برای استفاده از امکانات میکرو باید آنها را پیکره بندی کنیم یا به عبارتی **CONFIG** کنیم. حالا ما برای استفاده از ورودی خروجی میکروکنترلر **I/O** هم باید اونها پیکره بندی کنیم یعنی به میکرو اعلام کنیم که اقا ما میخاهیم از فلان پایه تو به عنوان ورودی یا مثلا خروجی استفاده کنیم. اگه خاستیم از یک پورت به عنوان خروجی استفاده کنیم باید به ترتیب مثال زیر عمل کنیم:

مثال: مثلا می خاهیم از پورت **A** به عنوان خروجی استفاده کنیم:

CONFIG PORTA = OUTPUT

حالا تحلیل این دستور. **CONFIG** که بهت گفتم یعنی چی **PORTA** یعنی می خاهیم از پورت **A** به عنوان خروجی استفاده کنیم به جای **A** از هر پورت دلخواه دیگه هم می تونیم استفاده کنیم. علامت مساوی = رو که باید همیشه بزاری. عبارت **OUTPUT** هم یعنی خروجی

اگه خاستیم از یک پورت به عنوان ورودی استفاده کنیم باید به ترتیب مثال زیر عمل کنیم:

مثال: مثلا می خاهیم از پورت **B** به عنوان ورودی استفاده کنیم

CONFIG PINB.0 = INPUT

حالا گوش فرا بده به تحلیل این دستور. **CONFIG** که گفتم قبلا. خوب **PINB.0** یعنی ما می خاهیم از پایه **B.0** به عنوان ورودی استفاده کنیم. = هم که هیچی همیشه باید بزاری. کلمه **INPUT** را وقتی میزاری که خاسته باشی از یک پایه به عنوان ورودی استفاده کنی

3- خوب بزار یک مرور بکنیم. اگر خاستیم که از یک پایه به عنوان ورودی استفاده کنیم از دستور

CONFIG PINC.0 = INPUT استفاده کنیم که ما در این مثال از پورت **C.0** به عنوان ورودی استفاده

کردیم نکته اینجاست که هر وقت که خاستی از یک پایه به عنوان ورودی استفاده کنی باید شماره پایه رو هم مثل همین مثال بالا که برای زد استفاده کنی به جای **PINB.0** هم می تونی مثال بزاری **PINB.1** یا **PINB.3** یا بزاری **PINB.7** که بستگی داره به انتخاب تو

اگر خاستیم که از یک پورت به عنوان خروجی استفاده کنیم از دستور زیر استفاده می کنیم باید از دستور زیر استفاده کنی

CONFIG PORT C = INPUT ما در این مثال به میکرو فهمانیدیم می خاهیم از پورت **C** تو به عنوان

خروجی استفاده کنیم

1- شماره پایه رو نباید بنویسیم

2- نه نباید بنویسیم بعدا ما بین برنامه مشخص می کنیم که کدام پایه باید خروجی بدهد

بزار ادمه درس رو با یک مثال برات بگم فرض کن ما میخایم از پایه **D.4** میکرو کنترلر یک خروجی بگیرم و یک **LED** رو روشن کنیم برای این کار از برنامه زیر استفاده می کنیم:

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 8000000$
CONFIG PORTD = OUTPUT
SET PORTD.4
END
```

خط اول و دوم که هیچ

خط سوم ما اومدیم به میکرو اعلام کردیم که می خاهیم از پورت **C** تو به عنوان یک خروجی استفاده کنیم

خط چهارم نوشتیم **SET PORTD.4** ما بانوشتن کلمه **SET** این منظور را رساندیم که می خاهیم پورت **D.4** یک شود. و همچنین جلوی **SET** نام پایه ای رو که باید یک شود یا به عبارتی دیگر **SET** شود رو هم اعلام کردیم

در خط آخر هم با دستور **END** برنامه را به پایان رساندیم

1- یعنی اگه یک **LED** رو به پایه **D.4** میکروکنترلر وصل می کردیم **LED** روشن می شد

2-اره روشن می شد

مثال دوم: می خواهیم همان پورت **D.4** را یک بار که روشن شد بعد از دو ثانیه خاموش شود

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 8000000$
CONFIG PORTD = OUTPUT
SET PORTD.4
WAITMS 2000
RESET PORTD.4
END
```

خط اول دوم که هیچ

خط سوم هم اومدیم تعرف کردیم که از پایه **D.4** می خواهیم به عنوان خروجی استفاده کنیم

خط چهارم با دستور **SET PORTD.4** پایه **D.4** رو یک کردیم

خط پنجم با دستور **WAITMS 2000** این را رساندیم که می خواهیم **2000** میلی ثانیه به عبارتی دو ثانیه در

همین جا تاخیر ایجاد شود در طول این دو ثانیه **PORTD.4** در همان حالت یک باقی می ماند. بعد از دو ثانیه

برنامه از خط بعد ادامه پیدا می کند

خط ششم با دستور **RESET PORTD.4** به میکرو دستور دادیم که همان پایه **D.4** را ریست کن یعنی صفر

کن

خط آخر هم که یعنی پایان برنامه. در این مثال اگر که ما یک **LED** به پایه **D.4** وصل می کردیم **LED** برای بار

اول با دستور **SET** روشن می شد و این روشن بودن با دستور **WAITMS 2000** دو ثانیه به طول می انجامید

و بعد از دو ثانیه با دستور **RESET PORTD.4 LED** خاموش می شد چون پایه را صفر کردیم.

مثال سوم: فرض کن می خواهیم دو تا پایه رو هم زمان یک کنیم و بعد از دو ثانیه فقط یکی از آنها را صفر کنیم.

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 8000000$
CONFIG PORTC = OUTPUT
SET PORTC.2
SET PORTC.7
WAITMS 2000
RESET PORTC.7
END
```

خط اول و دوم که هیچ

خط سوم هم که باید حتما یاد داشته باشی

خط چهارم اومدیم فرمان دادیم که پایه **C.2** را یک کن

خط پنجم هم اومدیم فرمان دادیم که پایه **C.7** رو یک کن

خط ششم دو ثانیه تاخیر در برنامه ایجاد کردیم که در طول این مدت پایه های **C.2** و **C.7** در حالت یک است

خط هفتم با دستور **RESET PORTC.7** فقط پایه **C.7** رو صفر کردیم. ولی پایه **C.2** در همان حالت یک باقی

مانده است

خط آخر هم که هیچ

مثال سوم: این مثال رو ببینم که می تونی تحلیل کنی

```
"REGFILE = "M16DEF.DAT$
CRYSTAL = 8000000$
CONFIG PORTA = OUTPUT
CONFIG PORTC = OUTPUT
SET PORTA.1
SET PORT C.0
```

حالا خودت تحلیلش کن

1-خط اول و دوم رو که بلدم

خط سوم اومدیم به میکرو اعلام کردیم که می خواهیم از پوت **A** به عنوان خروجی استفاده کنیم

خط چهارم هم مثل خط سوم

خط ششم دستور یک شدن پایه **A.1** رو صادر کردیم

خط هفتم هم دستور یک شدن پایه **C.0** رو صادر کردیم

خوب بود

2- عالی بود

1- حال ورودی رو برام توضیح بده

2- ورودی باشه برای وقتی که چندتا از دستورات شرط و چندتا دستور دیگه رو گفتیم برات میگم

1- خوب حالا این صفر و یک شدن پایه ها رو همیشه تو شبیه ساز نرم افزار تماشا کرد

2- متاسفانه نه همیشه

1- پس باید چکار کنم

2- دو راه داری اول اینکه بری یک میکرو بخری و خودت امتحان کنی دوم اینکه با استفاده از نرم افزار پروتوس

اونو شبیه سازی کنی .

1- خوب این پروتوس رو برام توضیح بده

2- اینو دیگه خودت برو یاد بگیر توی انجمن های برق و الکترونیک ایرانی که تعدادشون هم زیاده به خوبی

توضیح دادن

1- باشه حالا چی رو می خای برام توضیح بدی

2- فعلا تا همین جاشو داشته باش تا بعدا ادامشو برات توضیح بدم

پایان قسمت سوم

تمرینات قسمت سوم:

بنا به درخواست یکی از دوستان از این به بعد در پایان هر قسمت تمرینات مربوط به همان قسمت را می زارم و در

قسمت بعدی جواب آنها را می نویسم . دوستان سعی کنند خودشان تمرینات را جواب دهد و در قسمت بعدی

پاسخ ان را ببینند تا اشکالات بر طرف شود.

1- برنامه ای بنویسید که روی LCD نوشته شود MICRO

2- برنامه ای بنویسید که ابتدا روی LCD نوشته شود AVR . و بعد از 30 میلی ثانیه تاخیر LCD پاک شود و

دوباره روی ان نوشته شود **ALI**

3- برنامه ای بنویسید که در ان پایه A.0 میکرو SET شود (منظور PORTA.0 میباشد)

4- برنامه ای بنویسید که در ان پایه A.1 یک بار SET شود و بعد از 50 میلی ثانیه تاخیر RESET شود

5- برنامه ای بنویسید که در ان پایه های B.5 و D.7 میکرو یک بار هم زمان باهم SET شوند و بعد از 60 میلی

ثانیه تاخیر RESET شوند.

درضمن دوستان توجه داشته باشند اگر در جایی به اشکال برخوردید حتما سوال کنید . یک نکته اینکه **SET** و

RESET ورودی خروجی های میکرو رو همیشه تویه محیط شبیه ساز تماشا کرد و برای شبیه سازی بهتره از نرم

افزار **PROTEUS** استفاده کنید خوشبختانه یک فایل آموزشی این نرم افزار رو با فرمت **PDF** و به زبان فارسی

دارم که اگر تونستم اپلودش کنم براتون میزارم یا میتونید به لینک زیر مراجعه کنید .

<http://www.eca.ir/forum2/index.php?board=135.0>

آغاز قسمت چهارم :

1-سلام خوبی

2-سلام قربانت تو چطوری چکار می کنی با این میکروکنترلر

1-منم خوبم . با این میکروکنترلر داریم حال می کنیم والا قبلا برای طراحی یک مدار ساده دیجیتال مجبور بودم چند روز روش کار کنم ولی حالا دیگه نه . بعدشم یک چیزو یادت رفته بود بگی

2-چی؟؟

1-از **TOGGLE** مثال نزدی

2-باشه توی این درسی رو که امروز بهت میدم از این دستور هم استفاده می کنم که یاد بگیری . خوب شروع کنیم

1-اره

2-خوب قرار بود از کجا شروع کنیم

1-قرار بود ورودی رو بهم بگی . گفته بود قبل از اینکه ورودی رو درس بدی باید چند تا دستور شرط و ... رو برام بگی

2-اره این جلسه قبل از اینکه ورودی یا همون **PIN** رو بهت بگم باید . دستور حلقه - دستور پرش رو اول بگم بعد بریم سر ورودی . چون که این طوری بهتره

1-حرفی نیست بریم

2-خوب اول بریم سراغ حلقه . ببین به یک چیز توی برنامه های قبلی توجه کرده بودی

1-نه چی؟؟

2-ما میومدیم یک برنامه رو می نوشتیم این برنامه هم یک کاری رو برامون انجام میداد و در نهایت برناه به **END** می رسید و برنامه کلا تمام می شد . و برای شروع مجدد مجبور بودیم برنامه رو دوباره اجرا کنیم . خوب حالا فرض کن ما یک برنامه ای رو میخاییم که مثلا **PORTB.0** رو هر **40** میلی ثانیه یک بار روشن و خاموش کنه و این کار ادامه داشته باشه یعنی اینکه **40** میلی ثانیه روشن باشه و **40** میلی ثانیه خاموش باشه . برای این کار باید از دستور استفاده کنیم که برنامه هیچ وقت به پایان نرسه که این دستور اسمش هست حلقه

1-خوب حالا این حلقه چکار میکنه

2-این حلقه باعث میشه برنامه توی این قسمت دور بزنه و هیچ وقت به **END** نرسه یعنی برنامه تمام نشه مگر اینکه ما یک شرط بزاریم که از حلقه بیرون بزاریم

1-چه شرطی

2-اونو تو قسمت دستور شرط برات می گم

2- حلقه **DO - LOOP**

برای گذاشتن حلقه **DO** رو اول می نویسیم و در خط های بعدی برنامه ای که باید در حلقه باشد رو می نویسیم و در نهایت **LOOP** رو می نویسیم

1-من که چیزی نفهمیدم . یک مثال بزن

همون مثالی رو که **PORTB.0** باید **SET** و **RESET** بشه رو برات می نویسم . از این به بعد هم دیگه

```
"REGFILE = "M16DEF.DAT$
```

```
CRYSTAL = 8000000$
```

خوب حالا برنامه ای که گفتیم:

```
CONFIG PORTA = OUTPUT
```

```
DO
TOGGLE PORTA.0
WAITMS 40
LOOP
END
```

خوب حالا تحلیل برنامه :

این برنامه هیچ گاه به **END** نمی رسد تا برنامه تمام شود
خط اول رو که **PORTA** رو به عنوان خروجی پیکره بندی کردیم
خط دوم **DO** را گذاشتیم

خط سوم با نوشتن **TOGGLE PORTA.0** این را رساندیم که **PORTA.0** در هر حالتی از حالت ها صفر یا یک که بوده باید وضعیتش معکوس شود یعنی اگه صفر بوده حالا باید یک شود . اگر یک بوده حالا باید صفر شود
خط چهارم هم اخر حلقه را گذاشتیم

خط پنجم هم نوشتیم **END** البته برنامه هیچ گاه به خاطر این که ما یک حلقه گذاشتیم به **END** نمی رسد
توجه کن که ما برنامه ای رو که می خاستیم رو داخل این حلقه گذاشتیم و برنامه داخل این حلقه دور میزند . بار اول برنامه **TOGGLE PORTA.0** رو می خونه و پایه **A.0** رو **SET** یا همون یک می کنه بعد **40** میلی ثانیه صبر می کنه و برنامه از خط بعد شروع می شه خط بعد **LOOP** هستش که میکرو با خوندن **LOOP** دوباره بر میگردد برنامه رو از خط پایینی **DO** شروع به خوندن می کنه یعنی دوباره همون **TOGGLE PORTA.0** رو می خونه این بار چون پایه **A.0** در حالت **SET** یا همون یک بوده این بار یاد بشه عکس همین حالت بشه یعنی **RESET** یا صفر میشه بعد **40** میلی ثانیه صبر میکنه و میکرو برنامه رو از خط بعدی میخونه یعنی دستور **LOOP** رو می خونه . با خوانده شدن دستور **LOOP** میکرو برنامه رو از خط زیری **DO** شروع به خوندن می کنه یعنی دوباره **TOGGLE PORTA.0** رو می خونه این بار چون در حالت قبلی پایه **A.0** در حالت صفر بوده این بار یک می شه برنامه به همین منوال ادامه داره و ادامه پیدا می کنه و هیچ وقت به پایان نمی رسه

1-گرفتم چی می گی ولی اگه ممکنه یک مثال دیگه هم بزن
2-باشه

مثال دوم:

```
CONFIG PORTB = OUTPUT
DO
SET PORTB.5
WAITMS 60
RESET PORTB.5
WAITMS 60
LOOP
END
```

خوب حالا بریم سر تحلیل برنامه:

برنامه از خط اول شروع میشه و میکرو خط اول رو می خونه

خط دوم که **DO** هستش که ابتدای حلقه ما هستش

خط سوم **SET PORTB.5** یعنی پایه **B.5** یک بشه

خط چهارم یک تاخیر در برنامه با مدت زمان **60** میلی ثانیه گذاشتیم

خط پنجم **RESET PORTB.5** یعنی پایه **B.5** صفر بشه

خط ششم با گذاشتن دستور **LOOP** انتهای حلقه رو مشخص کردیم

خط هفتم هم نوشتیم **END** که برنامه به خاطر داشتن حلقه هیچ گاه به **END** نمی رسد

برنامه به این صورت است که از خط اول شروع به خونده شدن توسط میکرو میشه و پیکره بندی رو می خونه . بعد میوفته توی حلقه و اولین عبارتی رو که می خونه **SET PORTB.5** هستش که در این زمان میکرو پایه **B.5** رو

یک می کنه بعد توسط دستور **WAITMS 60** حدود **60** میلی ثانیه تاخیر ایجاد می کنه بعد از سپری شدن این مدت زمان میکرو برنامه رو از خط بعدی شروع به خواندن می کنه یعنی **RESET PORTB.5** که میکرو توسط این دستور پایه **B.5** رو صفر می کنه بعد توسط دستور **WAITMS 60** میکرو **60** میلی ثانیه در اجرای برنامه دوباره تاخیر ایجاد می کنه بعد از سپری شدن این مدت زمان میکرو برنامه رو از خط بعدی یعنی **LOOP** شروع به خواندن می کنه که با خوانده شدن **LOOP** برنامه می پره از خط بعدی یعنی **DO SET PORTA.0** شروع میشه و دوباره پایه **B.5** یک میشه و دوباره **60** میلی ثانیه تاخیر ایجاد میشه دو باره پایه **B.5** صفر میشه و **60** میلی ثانیه تاخیر ایجاد میشه بعد دوباره به **LOOP** می رسه و دوباره میکرو میپره و برنامه رو از خط بعدی **DO** شروع به خواندن میکنه و الی بی نهایت این برنامه تکرار می شه

1- عالی بود کامل گرفتم

2- خدارو شکر. جهنم ضرر یک مثال دیگه هم میزنم که **LCD** هم توش باشه

```
DO
CLS
"LCD "AHMAD
WAITMS 50
CLS
"LCD "MICRO
WAITMS 60
LOOP
END
```

خوب حالا نوبت می رسه به تحلیل برنامه :

برنامه از **DO** شروع میشه **DO** یعنی ابتدای حلقه . در خط بعد با دستور **CLS** ال سی دی پاک می شه و آماده نوشتن . در خط بعدی به میکرو دستور می دیم که عبارت **AHMAD** رو روی **LCD** نمایش بده . در خط بعد فرمان ایجاد تاخیر به مدت زمان **50** میلی ثانیه را می دهیم پس از سپری شدن این مدت زمان میکرو برنامه را از خط بعد می خواند . خط بعدی با دستور **CLS** صفحه **LCD** را پاک کردیم تا برای نوشتن دوباره آماده شود. در خط بعد فرمان دادیم که عبارت **MICRO** روی صفحه **LCD** نمایش داده شود. در خط بعد دستور دادیم که **60** میلی ثانیه تاخیر ایجاد شود که پس از سپری شدن این تاخیر برنامه از خط بعد اجرا می شود. به خط بعدی یعنی **LOOP** که می رسیم میکرو تا اینکه این دستور **LOOP** رو می بیند بی درنگ می پره میره برنامه رو از خط بعدی **DO** شروع می کنه به اجرا کردن . یعنی از خط **CLS** و همینطور ادامه می ده

3- رو حساب رفاقت می خام یک مثال دیگه هم بزنم که حالشو ببی

```
DIM G AS BYTE
G = 0
DO
INCR G
CLS
LCD G
WAITMS 50
LOOP
END
```

خط اول رو که خودت می دونی یعنی اینکه ما یک متغییر داریم که اسمش هست **G** و از نوع بایت

خط دوم به متغییر **G** مقدار دادیم

خط سوم ابتدای حلقه را مشخص کردیم

خط چهارم یک واحد به متغییر **G** اضافه کردیم

خط پنجم با دستور **CLS** ال سی دی را برای نوشتن پاک کردیم

خط ششم دستور دادیم که مقدار متغییر **G** روی **LCD** نمایش داده شود

خط هفتم یک تاخیر 50 میلی ثانیه ای در برنامه ایجاد کردیم
خط هشتم با دستور LOOP انتهای حلقه را مشخص کردیم
خوب حالا تحلیل اصلی رو گوش کن :

میکرو میاد برنامه رو از خط اول شروع می کنه به خوندن و می فهمه که یک متغیر از نوع بایت ایجاد شده . بعد خط دوم رو می خونه که نوشته مقدار این متغیر هست صفر . خط سوم ابتدای حلقه مشخص شده است . خط چهارم با دستور INCR G یک واحد به متغیر G اضافه کردیم یعنی اگه بوده صفر حالا که یک واحد بهش اضافه شده میشه 1 . خط بعدی با دستور CLS ال سی دی را پاک کردیم تا برای نوشتن حاضر شود . خط بعدی دستور دادیم که مقدار متغیر G که الان هست 1 روی LCD نمایش داده شود . خط بعدی با دستور WAITMS 50 دستور تاخیر به مدت 50 میلی ثانیه را صادر کردیم . خط بعدی هست LOOP که میکرو وقتی که به این دستور می رسه می پره میره و برنامه رو از خط زیری DO شروع می کنه به خوندن توجیه کن که از خط بعدی DO شروع می کنه به خوندن نه اینکه از بالا . داریم به جای جالبش می رسیم . خوب حالا باید برنامه از خط بعدی DO خونده بشه خط بعد از DO نوشته چی نوشته INCR G خوب یعنی اینکه یک واحد به متغیر G اضافه بشه خوب حالا آخرین مقدار متغیر G بوده 1 (به دلیل اینکه دفعه قبل با دستور INCR G یک واحد به ان اضافه کرده بودیم) و حالا که دوباره دستور دادیم که یک واحد دیگر به متغیر G اضافه بشه اینبار میشه 2 . خط بعدی با دستور CLS ال سی دی را برای نوشتن دوباره پاک می کنیم . در خط بعدی با دستور LCD G مقدار متغیر G را روی ال سی دی نمایش می دهیم یعنی مقدار جدید که همان 2 است . خط بعدی دوباره یک تاخیر ایجاد می شود و پس از سپری شدن این مدت زمان برنامه از خط بعدی اجرا می شود . خط بعدی هست LOOP باز هم مثل دفعه قبل برنامه از خط زیری DO شروع می شه و روز از نو روزی از نوع همینطور به متغیر G یکسره اضافه می شه تا جایی که متغیر G برسه به 255 وقتی که به این مقدار رسید متغیر G پر می شود (چون از نوع بایت است و با 255 پر می شود و با 256 سرریز می شود) . وقتی که این 255 برسه به 256 متغیر سر ریز میشه و دوباره از صفر شروع به شمردن می کنه . دیگه از این ساده تر نمی شد توضیح داد.

1- اقا یک سوال

2- بپرس

1- توی مثال به یک نکته توجه کردی

2- نه چی

1- مثال توی قسمتی که می خاستی توی یک حلقه مثال LCD رو بزنی نوشتی "ALI LCD" ولی وقتی که

خاستی مقدار یک متغیر رو نشون بدی این نوشتی LCD G که مقدار متغیر G رو نشون بده حالا سوال

اینجاست چرا ALI رو داخل " " گذاشتی ولی متغیر G رو داخل " " نذاشتی

2- سوال خوبی بود . ببین هر وقت که خاستی یک متغیر رو نشون بدی لازم نیست که اونو داخل " " قرار بدی

یعنی نباید داخل " " قرار بدی ولی هر وقت که خاستی یک عبارت رو روی ال سی دی نمایش بدی باید اونو داخل

" " قرار بدی

1- فهمیدم . این حلقه تموم شد

2- اره

1- پس اگه خاستیم از داخل این حلقه بیرون بیایم باید چکار کنیم

2- اینو برات می گم وقتی که دستورات شرط رو گفتم

خوب حالا رسیدیم به دستورات شرط

1- چی هست این دستور شرط

2- گاهی لازم است که اگر یک حالتی که برای ما مد نظر است اتفاق بیفته دستور شرط یک کاری رو مثلا یک پایه

رو برامون یک می کنه

1- ببین نفهمیدم دوباره بگو

2- باشه برمیگردیم سر دو تا مثال قبل که یک متغییر داخل یک حلقه بود و هر دفعه که دور می زد یک واحد به متغییر اضافه می شد . خوب اگه این کار ادامه پیدا کنه این متغییر هی پرو میشه با 255 و دوباره از صفر شروع می کنه دوباره پرو می شه با 255 و دوباره از صفر شروع می کنه به اضافه شدن حالا ما حال کردیم که اگر این متغییر **G** به مثلا 22 رسید یک کاری رو برامون انجام بده حالا هر کاری رو که حال کردیم رو بره برامون انجام بده مثلا یک پورت رو بیاد برامون **SET** کنه یا اینکه بره رویه **LCD** یک چیزی بنویسه . که ما باید از دستور شرط استفاده کنیم . یک مثال ساده دیگه که می تونم برات بزنم اینه که مثلا بابای تو بهت می گه که اگر آخر سال معدل تو 20 شد برات دو چرخه می خرم . خوب بابات برای تو یک شرط گذاشته که اگر این شرط که 20 شدن معدل تو است اجرا بشه . دستور شرط که خریدن دو چرخه توسط بابات هست اجرا میشه . حالا فهمیدی

1- بابا ما از کوچیکی داریم 20 می گیرم بابامون برامون هیچی نگرفته . ولی مطلبو خوب گرفتم

2- عالیه . خوب بریم سر دستور شرط

IF همان دستور شرط است

THEN یعنی اگر شرط اجرا شود این **THEN** میاد برای ما یک کاری رو انجام می ده

1- مثال برن

2- همون مثال دو چرخه رو می زنم . خوب بابای فلانی بهش گفته که اگر معدلش 20 بشه یعنی شرط **IF** ... براش یک دچرخه می خره یعنی یک کاری رو در قبال شرطی رو که گذاشته برای پسرش انجام می ده یعنی **THEN**

1- اها گرفتم. از مثال های برنامه هم برام بگو

2- باشه . بریم سر اون مثالی که متغییر **G** رو توی یک حلقه قرار داده بودیم حالا ما می خایم که اگر این متغییر **G** به 6 برسه بره و **PORTA.0** رو **SET** کنه :

```
CONFIG PORTA = OUTPUT
DIM G AS BYTE
G = 0
DO
  INCR G
  CLS
  LCD G
IF G = 6 THEN SET PORTA.0
  WAITMS 50
LOOP
END
```

برنامه بالا رو که تا حدودی می دونی چیه چون قبلا توضیحشو دادم فقط می مونه خط هفتم که نوشته **IF G = 6 THEN SET PORTA.0** ما در اینجا با گذاشتن شرط یعنی **IF G = 6** شرط گذاشتیم که اگر متغییر **G** به عدد 6 رسید . ادامه نوشتیم **THEN SET PORTA.0** بروی **SET** کن **PORTA.0** . فهمیدی

1-اره ولی مگه شرط بندی حروم نیست

2-چرا حرومه . ولی باید مواظب باشی که شرط رو نبازی

خوب رسیدیم به دستورات زیر برنامه (**GOTO - GOSUB**)

1-چی هستن این زیر برنامه ها

2-با یک مثال برات توضیح می دم توی مثال قبل اگر دقت کرده باشی متغییر **G** وقتی که به عدد 6 می رسید می رفت و **PORTA.0** رو **SET** می کرد و همینطوری به متغییر **G** اضافه می شد . حالا اگر که ما حال کردیم بعد از این که متغییر **G** به عدد 6 رسید بره همون کاره خودشو یعنی **SET** کردن **PORTA.0** رو انجام بده ولی برنامه دیگه تموم بشه و دیگه شمارش نکنه یا به عبارتی به **END** برسه . خوب برای این کار باید از زیر برنامه استفاده کنیم به مثال زیر توجه کن :

```
CONFIG PORTA = OUTPUT
DIM G AS BYTE
```

```

G = 0
DO
INCR G
CLS
LCD G
IF G = 6 THEN GOTO AHMAD
WAITMS 50
LOOP
:AHMAD
SET PORTA.0
END

```

خوب حالا بریم تحلیل برنامه . برنامه رو که می دونی چجوریه منظورم معنی خط ها هستش البته به غیر از خط هشتم که نوشته

```
IF G = 6 THEN GOTO AHMAD
```

خوب این خط یعنی اینکه اگر متغییر **G** رسید به عدد **6** دستور شرط را اجرا کن که دستور شرط این است **GOTO AHAD** یعنی بپر برو به زیربرنامه ای که استش هست احمد . توجه کن که به جای اسم **AHMAD** هر اسم دیگه ای می تونی بزاری هر اسمی که حال کردی . بعد از اینکه متغییر **G** به عدد **6** رسید دستور شرط اجرا می شود و میکروکنترلر برنامه را از خط بعدی زیربرنامه **AHMAD** می خواند یعنی خط **SET PORTA.0** را می خونه و پورت **A.0** رو ست می کنه بعد از این که ست کرد خط بعدی رو می خونه که نوشته **END** به محض اینکه این دستور **END** رو خوند برنامه متوقف می شه و به پایان می رسه .

پس فهمیدی که زیر برنامه برای وقتی هستش که ما می خوایم یک دستور شرط رو اجرا کنیم به یک مثال دیگه توجه کن :

```

CONFIG PORTA = OUTPUT
DIM G AS BYTE
:METAL
G = 0
RESET PORTA.0
DO
INCR G
CLS
LCD G
IF G = 6 THEN GOTO AHMAD
WAITMS 50
LOOP
:AHMAD
SET PORTA.0
WAITMS 50
GOTO METAL
END

```

خوب حالا تحلیل برنامه . میکرو میاد برنامه رو از خط اول شروع میکنه به خوندن که خط اول مربوط به پیکره بندی **PORTA** به عنوان خروجی هستش . بعد میاد خط دوم رو می خونه که در این خط ما یک متغییر که اسمش هست **G** و از نوع بایت می باشد رو تعریف کرده ایم. خط سوم یک زیر برنامه هستش به نام **METAL** که فعلا بهش کار ندارم تا بعد در ضمن این نکته رو باید یادت باشه که هر وقت یک زیر برنامه رو می نویسی باید جلوش دو نقطه رو بزاری اینطوری **METAL** . خط بعدی به متغییر یک مقدار دادیم که صفر است . خط بعدی نوشتیم **RESET PORTA.0** که به این هم فعلا کار نداریم . خط بعدی **DO** هستش که ابتدای حلقه را مشخص می کنه . خط بعدی نوشته **INCR G** یعنی به مقدار متغییر **G** یک واحد اضافه کن یعنی صفر که بوده حالا که یک

واحد اضافه بشه می شه یک . خط بعدی با دستور **CLS** ال سی دی را پاک کردیم تا برای نوشتن دوباره آماده شود . خط بعدی با فرمان **LCD G** این را رساندیم که مقدار متغییر **G** رو روی ال سی دی نمایش بده که الان باید عدد یک رو روی ال سی دی نمایش بده . در خط بعدی نوشتیم **IF G = 6 THEN GOTO AHMAD** یعنی اگر متغییر **G** ما به عدد **6** رسید برو دستور شرط رو که هست **THEN GOTO AHMAD** اجرا کن یعنی ببر برو توی زیر برنامه **AHMAD** البته وقتی دستور شرط اجرا میشه که متغییر **G** به عدد **6** برسه . خط بعدی یک تاخیر در برنامه با مدت زمان **50** میلی ثانیه را گذاشتیم که بعد از سپری شدن این مدت زمان برنامه از خط بعدی شروع می شه خوب تا این جا که متغییر ما عدد **1** هستش . بعد که میکرو میرسه به دستور **LOOP** دوباره دور می زنه و خط زیری **DO** رو می خونه که نوشته **INCR G** یعنی یک واحد به متغییر **G** اضافه کن خوب متغییر **G** الان هستش **1** بعد که به این دستور رسید این دفعه یک واحد بهش اضافه می شه و می شه دو . این عمل انقدر ادامه می یابد تا متغییر **G** برسه به عدد **6** . همین که به عدد **6** رسید از اونجایی که ما یک شرط با دستور **IF G = 6 THEN GOTO AHMAD** گذاشتیم یعنی اینکه وقتی متغییر **G** به عدد شش رسید یعنی **IF G = 6 THEN GOTO AHMAD** اجرا کن یعنی اینکه ببر برو توی زیر برنامه **AHMAD** خوب حالا میکرو میبره میره برنامه رو از خط زیری زیر برنامه **AHMAD** می خونه که نوشته **SET PORTA.0** یعنی پایه **A.0** رو یک کن . بعدش نوشته **WAITMS 50** یعنی اینکه به مدت زمان **50** میلی ثانیه در برنامه تاخیر ایجاد کن که در طول این مدت **PORTA.0** در وضعیت **SET** قرار دارد بعد از این که این مدت زمان سپری شد برنامه از خط بعدی اجرا می شه خط بعدی یعنی **GOTO METAL** یعنی اینکه ببر برو توی زیر برنامه **METAL** و میکرو هم همین کار رو می کنه و می پره می ره توی زیر برنامه **METAL** و برنامه از خط بعدی این زیر برنامه که هستش **G = 0** ادامه پیدا می کنه این خط به این معنی هستش که متغییر **G** رو که بوده **6** حالا بیا صفرش کن . در خط بعدی یعنی **RESET PORTA.0** یعنی اینکه پایه **A.0** رو که ست کرده بودی حالا بیا ریستش کن یعنی صفر کن . خوب حالا هم متغییر **G** ما صفر هستش هم اینکه **PORTA.0** ما صفر یا همون ریست هستش . بعد میکرو می یاد برنامه رو از خط بعدی شروع میکنه به خوندن که نوشته **DO** یعنی اول زیر برنامه . خط بعدی نوشته **INCR G** یعنی یک واحد به متغییر **G** ما که الان دوباره صفرش کردیم اضافه کن که بشه یک . دوباره روز از نو روزی از نو یعنی اینکه دوباره شروع می کنه به اضافه کردن متغییر تا دوباره برسه به **6** بعد که رسید به **6** دوباره بیره بره به زیر برنامه **AHMAD** اونجا باز **PORTA.0** رو یک می کنه بعدش تاخیر ایجاد می کنه بعد با دستور **GOTO METAL** می پره میره به زیر برنامه **METAL** که در بالای برنامه هستش بعد دوباره متغییر رو که **6** هستش رو صفر می کنه به دنبالش **PORTA.0** رو هم که یک بوده صفر می کنه باز دوباره.....

حالا رسیدیم به دستور **GOSUB**

این دستور همان دستور **GOTO** هستش با این تفاوت که یک دستور دیگه همراهش هست به نام اقای

RETURN که این **RETURN** رو هر جای برنامه که بنویسیم میکرو میاد برنامه رو از خط بعدی **GOSUB**

شروع می کنه به خوندن . که دربارش وقتی که خواستم ورودی رو بهت درس بدم می گم .

ورودی **PIN** :

ورودی در میکرو کار بردای فراوونی داره مثلا در ساده ترین حالت می خای یک میکروسویچ به میکروکنترلر وصل کنی که با هر بار زدن این میکروسویچ یک **LED** روشن و خاموش بشه در مراحل پیشرفته هم مثلا در ساخت یک ربات کاربرد داره به این صورت که یک سنسور مادون قرمز بهش وصل بشه و مثلا به یک خط سیاه حساس باشه و از این طور کارا

خوب بریم سر دستور ورودی . باز هم می خوام با یک مثال برات توضیح بدم

فرض کن که می خایم با یک میکرو سوییچ که به میکرو وصل کردیم یک **LED** رو یک بار روشن و یک بار

خاموش کنیم که برنامه به صورت زیر است و خیلی خیلی ساده :

CONFIG PINB.0 = INPUT

DIM A AS BYTE
.Ali
DO
DEBOUNCE PINB.0 , 1 , SONY
LOOP
END
:SONY
INCR A
CLS
LCD A
GOTO ALI

خوب حالا وقت تحلیل برنامه هستش: در خط اول ما اومدیم با نوشتن دستور **CONFIG PINB.0 = INPUT** پایه **B.0** رو به عنوان ورودی تعریف کردیم . در خط بعد یک متغییر از نوع بایت که اسمش رو گذاشتیم **A** تعریف کردیم . در خط بعدی یک زیر برنامه به نام **ALI** گذاشتیم که فعلا بهش کار نداریم و در ادامه توضیحش می دم. در خط بعدی با نوشتن **DO** ابتدای حلقه را مشخص کردیم . در خط بعدی نوشتیم که **DEBOUNCE PINB.0 , 1 , SONY** که هر وقت خاستی که یک ورودی رو تعریف کنی بهتره از این دستور استفاده کنی یا به عبارتی حتما استفاده کنی خوب حالا لابد از خودت می پرسی که این دستورات چی هست این دستور که همش باید توی یک خط نوشته بشن رو یکی یکی برات بازش می کنم **DRBOUNCE** رو همیشه وقتی که می خای یک ورودی رو تعریف کنی اولش باید بزاری و کاربردشم اینه که یک تاخیر خیلی خیلی کوچیک که به چشم هم نمیاد ایجاد می کنه و باعث می شه که پایه ورودی نویز نگیره یک بار دیگه دوباره می گم ما نوشتیم که **DEBOUNCE PINB.0** یعنی پایه **B.0** رو از نظر نویز محافظت کن خوب این تا بعدش علامت , رو گذاشتیم تا جملات از هم تفکیک شوند که باید حتما علامت , رو بزاری در ادامه ما نوشتیم که **1** خوب حالا این **1** یعنی چی . این **1** یعنی اگر پایه **B.0** یا یک **1** شد یا به عبارتی **SET** شد با میتونیم بگیم که یک نوع شرط است که در مقابل یک شدن پایه **B.0** گذاشته ایم و جلوی این **1** دوباره علامت , گذاشتیم تا از جمله بعدیش تفکیک بشه . در جمله بعدیش نوشتیم **SONY** که این یعنی اینکه اقا اگر این پایه **B.0** ما **1** شد پیر برو توی زیربرنامه **SONY** که به جای این **SONY** هر اسم دخواه دیگه ای هم می تونی بزاری . یک بار دیگه برات این جمله رو به صورت کلی توضیح می دم معنی این جمله دستوری این هستش که اگر **PINB.0** ما **1** شد پیر برو توی زیربرنامه **SONY** و اگر صفر بود که هیچی . در خط بعدی نوشتیم **LOOP** که انتهای حلقه را با این کلمه مشخص کردیم . در خط بعدی نوشتیم **END** که برنامه هیچگاه به **END** نمی رسد . در خط بعدی نوشتیم **SONY** که **SONY** یک زیربرنامه است که می تونست هر اسم دیگه ای باشه . در خط بعدی نوشتیم **INCR A** که به معنی اضافه کردن یک واحد به متغییر **A** می باشد . در خط بعدی نوشتیم که **CLS** یعنی پاک کن کل صفحه ال سی دی رو تا برای نوشتن آماده بشه. در خط بعدی نوشتیم **GOTO ALI** یعنی پیر به زیربرنامه **ALI** . خوب حالا بریم سر طرز کار مدار : این مدار در واقع یک شمارنده هستش . طرز کارش هم اینطوریه که میکرو میاد سه خط اول رو می خونه بعد میره می یوفته توی حلقه **DO - LOOP** و اونجا داخل این حلقه هی دور میزنه تا کی تا وقتی که **PINB.0** ما یک بشه یعنی شرطی که گذاشته ایم اجرا بشه که شرط ما این هستش که پایه **B.0** یک بشه بعد از اینکه این شرط اجرا شد میکرو میاد دستور شرط رو اجرا می کنه دستور شرط کدومه همونی که بهت گفتم یعنی پیرش به زیر برنامه **SONY** . و میکرو که گوش به فرمان برنامه هست می پره می ره داخل زیر برنامه **SONY** و از خط بعدی زیر برنامه **SONY** که نوشته **INCR A** رو می خونه و طبق این دستور یک واحد به این متغییر **A** اضافه می کنه یعنی میکشش **1** . بعدش می یاد خط بعدی رو می خونه که نوشته **CLS** یعنی اینکه کل ال سی دی پاک بشه تا برای نوشتن حاضر بشه . در خط بعدی نوشتیم که

LCD A به این معنی که مقدار متغییر **A** رو که هستش یک روی **LCD** نمایش بده . در خط بعدی نوشته **GOTO ALI** یعنی اینکه پیر برو توی زیر برنامه **ALI** و میکرو هم همین کار رو می کنه و می پره می ره اولین خط بعد از زیر برنامه **ALI** رو که نوشته **DO** رو می خونه و دوباره میوفته توی این حلقه و باز هم دوباره انقدر توی این حلقه هستش و توی این حلقه دور می زنه تا وقتی که دوباره **PINB.0** یک بشه یا به عبارتی **SET** بشه اونوقت دوباره روز از نو روزی از نو و میپره دوباره توی زیر برنامه **SONY** و دستورات داخل این زیر برنامه رو دباره اجرا می کنه که در خط اول این زیر برنامه نوشته

INCR A یعنی دوباره یک واحد به متغییر **A** اضافه کن و میکرو هم این کار رو میکنه و متغییر **A** که یک بوده الان میشه **2** بعد میاد دستور **CLS** رو می خونه که لازم به توضیح نیست . بعد می با دستور **LCD A** رو می خونه که مقدار جدید این متغییر **A** رو که الان هستش **2** رو روی **LCD** نمایش می ده بعد میاد دستور **GOTO ALI** رو می خونه و می پره می ره توی زیر برنامه **ALI** دوباره میوفته توی حلقه و الی اخر . با استفاده از این برنامه می تونی یک شما رنده درست کنی که می تونه از عدد **0** تا **255** رو بشماره و اگر خاصی رنج شمارشش از **255** بیشتر باشه می تونی به جای **BYTE** توی برنامه بنویسی **WORD** . این برنامه در کارخونه هم کاربرد داره البته با کمی تغییرات کوچیک در برنامه و مثلا بایک سنسور می شه قطعات عبوری از روی خط تولید رو شمارش کرد.

1- از نظر اتصالات سخت افزاریش هم توضیح بده برام

2- این که خیلی ساده هستش کافیه تغذیه میکرو رو وصل کنی و با یک میکروسویچ که یک سرش رفته باشه به **VCC 5** ولت و سر دیگه میکروسویچ هم رفته باشه به پایه **B.0** و اتصالات ال سی دی این مدار رو کاملش کنی.

1- جالب بود

2- تو به این می گی جالب بزار یکم راه بیوفتی اونوقت این طور مدارا برات پیش پا افتاده هستش . وقتی که میتونی میکروکنترلر رو از طریق تلفن **SMS** و اینترنت کنترلر کنی دیگه این مدارات در نظرت بچه بازی هستش

1- چطوری از طریق اینایی که گفتی کنترل می کنن

2- دیگه باید یکم اطلاعاتت در مورد پورتهای کامپیوتر و **PC INTERFACES** کامل باشه که بتونی از طریق

مثلا اینترنت یک موتور رو کنترل کنی وگرنه برنامه نویسی میکرو که چیز نیست

1- با این **PC INTERFACES** چه کار میتونی بکنی و اصلا چی هستش

2- اندازه گیری و کنترل با استفاده از پورتهای استاندارد تحت ویندوز رو می گن **PC INTERFACES** مثلا شما

توی محیط نرم افزار ویرژال بیسیک یک برنامه ای می نویسی که اگر مثلا فلان دکمه کیبرد فشرده شد خروجی

فلان پورت کامپیوتر رو یک کن که این خروجی ها از نظر استاندارد با استاندارد میکروکنترلر فرق دارنند و باید از

ای سی **MAX232** استفاده کنی که از نظر استاندارد با هم مطابق شوند .

1- علم چقدر پیشرفت کرده ها

2- بله . بزار یک مثال دیگه هم بزنم ولی توضیحش رو خیلی خلاصه برات می گم

CONFIG PORTA = OUTPUT

CONFIG PINB.0 = INPUT

DIM A AS BYTE

:Ali

DO

DEBOUNCE PINB.0 , 1 , SONY

LOOP

END

:SONY

INCR A

CLS

LCD A

TOGGLE PORTA

GOTO ALI

در برنامه بالا فقط یک چیز اضافه شده اونم دستور **TOGGLE PORTA** که با وجود این دستور هر وقت که میکرو به زیر برنامه **SONY** پرش می کنه علاوه براین که یک واحد به متغییر **A** اضافه می کنه و روی ال سی دی هم نمایش می ده موقعیت **PORTA** رو هم عوض می کنه یعنی اگر صفر باشه یک می شه و اگه یک باشه صفر می کنه و دفعه بعد که به این زیر برنامه رفت دوباره اگر صفر باشه یکش می کنه و اگه یک باشه صفرش می کنه بزار منظورمو ساده بگم اگر شما یک **LED** به پایه های **A.0** تا **A.7** وصل کنی به ازای هر بار زدن میکروسویچ این **LED** ها یک بار روشن و باردیگه خاموش می شن.

برای امروز دیگه کافیه بقیش باشه برای جلسه بعدی

پایان قسمت چهارم

قسمت پنجم آموزشی

1-سلام خوبی

2-سلام من خوبم تو چطوری چکار می کنی

1-خوبم ولی یکم ناراحتم

2-چرا

1-بابا تمام این مثال هایی رو که گفته بودی توی نرم افزار **BASCOS** تست کردم همشون هم درست بود ولی

چه فایده وقتی که نمی تونم توی محیط **BASCOS** شبیه سازیشون کنم

2-خوب من که بهت گفته بودم که برو توی نرم افزار پروتوس تستش کن

1-اره گفته بودی ولی من که یاد ندارم باهاش کار کنم

2-خوب بهت یک لینک داده بودم که بری اونجا دربارش مطلب زیاده

1-خوب درست ولی اگه توضیح بدی اونایی رو هم که از توی لینک گرفتم باهاش اطلاعاتم کامل می شه

2-خیلی خوب چاره چیه ما که این همه گفتیم آموزش پروتوس هم می گیم دیگه چی؟؟

1-دمت گرم بابا خیلی حال دادی

2-فقط یک چیزی رو از همین الان بگم من این نرم افزار پروتوس رو بهت می گم و از اونجایی که این نرم افزار

بعضی وقتا **EROR** های بدی میباید مشکلتو خودت حل کنی یا توی انجمن مشکلت رو بگی چون من

فقط در حد کارکردن با این نرم افزار سرم میشه .

1-باشه هر چی که تو بگی

2-خوب پس این جلسه نحوه نصب و کارکردن با نرم افزار پروتوس ورژن **6** رو برات توضیح می دم

قبل از اینکه نحوه نصب رو برات توضیح بدم بزار یک مقدار درباره کاربرد نرم افزار برات بگم . این نرم افزار

پروتوس یک نوع شبیه ساز هستش که علاوه بر شبیه سازی قاده یک مدار رو انالیز کنه و همچنین امکان

کشیدن پشت فیبر مدار چاپی رو هم میشه از امکاناتش شمرد . ما در اینجا فقط می خوایم از امکان شبیه سازیش

استفاده کنیم پروتوس یک شبیه ساز بسیار قدرت مند هستش که شما می تونی هر مداری رو که می خای توش

شبیه سازی کنی و نتیجه کارت رو ببینی . خود من هر وقت که می خام یک مداری رو درست کنم اول با پروتوس

شبیه سازیش می کنم بعد از اینکه نتیجه گرفتم می روم و اونو روی برد پیاده می کنم . یکی از امکاناتی که نرم

افزار پروتوس در اختیار ما قرار داده شبیه سازی مدارات میکروکنترل هستش و شما قبل از این که خاسته باشی مدار تو روی برد سوار کنی می تونی توی پروتوس شبیه سازی کنی و بعد از اینکه نتیجه دلخواهت رو گرفتی اونو روی برد پیاده کنی . شما داخل این پروتوس هر قطعه ای رو که حتی تا حالا ندیدیش پیدا می کنی و باهاش کار کنی .

خوب حالا بریم سر نصب نرم افزار پروتوس ورژن 6 :

برای نصب نرم افزار **PROTEUS** ابتدا با اجرا کردن **INSTALL** یا **SETUP** انرا نصب می کنیم بعد از اینکه نصب شد می ریم داخل پوشه **CRACK** که در داخل **CD** نرم افزارش هست . سه تا **CRACK** داخلش هست که هر سه تاشون شبیه ادمک هستند و هر سه تا رو باید کپی کنی و در داخل پوشه **BIN** بریزی یعنی **PASTE** کنی حالا این فایل **BIN** رو حتما از خود می پرسی که کجاست پوشه **BIN** همون جایی هستش که نرم افزار رو ریختی بعد از اینکه این کارا رو کردی نوبت می رسه به اجرا کردن خود **CRACK** هایی که ریختی که باید یکی یکی اونا رو اجرا کنی بعد از اینکه هر کدوم رو اجرا کردی یک کادر کوچیک باز می شه که سه گزینه داره اولی **NEXT** دومی **RESTORE** و سومی **PATCH** می باشد که بعد از باز شدن پنجره باید روی گزینه **RESTORE** یا **PATCH** که دقیقا یادم نیست کدومشون هست کلیک کنی که یک پنجره باز می شه داخل این پنجره جدید یک فایل هستش که باید اونو انتخاب کنی و هی **NEXT** رو بزنی تا وقتی که پیغام بده **CRACK SUCCESSFUL** بعد از این که هر سه تا رو اجرا کردی نرم افزار آماده اجرا هستش .

1- خوب تا اینجا که راحت بود حالا برو سر توضیح خود نرم افزار

2- باشه . خوب قصد من از آموزش این نرم افزار **PROTEUS** فقط شبیه سازی خود میکروکنترلر **AVR** هستش . برای این که بتونی این کار رو بکنی پس از اینکه پروتوس رو باز کردی باید روی یک شکل که در بالای نرم افزار است کلیک کنی که این شکل شبیه یک **OPAMP** است و از سمت چپ اولین شکل هستش که اگر روی این شکل موس رو نگه داری نوشته **COMPONENT** . پس از این که روی این شکل کلیک کردی یکم پایین تر از این شکل دو تا گزینه پیدا میشه که اولیش هست **P** و دومیش هست **L** که تو باید روی **P** کلیک کنی تا کتابخانه قطعت ضاهر بشه که بالای پنجره این کتابخانه نوشته **PICK DEVICES** . خوب تو باید از داخل این پنجره قطعاتی رو که می خای انتخاب کنی مثلا اگر یک میکرو کنترلر **AT90S8535** می خای باید ابتدا روی گزینه **MICRO** کلیک کنی تا لیست کل میکروکنترلر ها اعم از **AVR-PIC-8051** رو برات نمایش بده که ما برای انتخاب میکروکنترلر **AT90S8535** باید روی ان دوبار کلیک کنیم تا در کادر کناری سمت چپ نرم افزار ذخیره شود . اگر میکروسویچ می خاستی در کادر بالایی همین پنجره گزینه **ACTIVE** رو انتخاب کنی که توش قصعاتی هستش که کاربرد زیادی دارند مثل باطری - موتور **DC-LED** - و همچنین میکروسویچ . خوب برای این که میکروسویچ خاستی بیاری باید روی **BUTTON** دو بار کلیک کنی تا اون هم در کادر سمت چپ نرم افزار ذخیره بشه منظورم از کادر سمت چپ نرم افزار کادی هستش که بالاش نوشته **DEVICES** و دقیقا زیر همون **L** که بهت گفتم قرار داره . در ضمن برای بزرگ نمایی صفحه نرم افزار باید از **F6** و **F7** استفاده کنی . یک نکته دیگه اون هم اینکه برای این که از **VCC** و **GND** نرم افزار استفاده کنی باید بری روی یک شکل دیگه که علامت دوتا پیکان هستش که مخالف جهت یکدیگردن و این شکل **7** تا شکل بعد از اون شکلی رو که گفتم شکل **OPAMP** هستش به سمت راست قرار داره که اگه روی این شکل با موس وایستی نوشته **INTER-SHEET TERMINAL** که بعد از کلیک روی این گزینه یک کادر در سمت چپ نرم افزار باز میشه که هم **GND** داره و هم **VCC** برای اینکه بتونی این دوتارو روی صفحه بیاری کافیه روش یک بار کلیک چپ کنی بعد بیای روی صفحه نرم افزار دو باره کلیک چپ کنی تا روی صفحه نرم افزار ضاهر بشه . برای دیگر قطعات هم باید همین کار رو بکنی یعنی قطعاتی رو که ذخیره کردی رو یک بار روش کلیک چپ کنی تا انتخاب بشه بعد بیای روی صفحه نرم افزار دوباره کلیک چپ کنی تا ضاهر بشه . برای پاک کردن قطعه از روی صفحه برنامه باید روی اون دوبار کلیک راست کنی . برای تنظیمات یک قطعه مثلا عوض کردن مقدار یک مقاومت یا یک باطری باید ابتدا یک بار روی اون کلیک راست کنی تا قطعه قرمز بشه بعد دوباره روش کلیک چپ کنی تا یک پنجره باز بشه و بتونی مقدار یا هر

چیز دیگه شو عوض کنی ولی مواظب باش که یک وقت دوبار روی اون کلیک راست نکنی چون قطعه پاک بشه و اگر یک مومقع این اشتباه رو کردی می تونی **ctrl + z** رو از روی صفحه کلید بزنی تا قطعه برگرده . برای این که جای یک قطعه رو روی صفحه عوض کنی باید ابتدا روی قطعه یک بار کلیک راست کنی تا انتخاب بشه بعد با کلیک چپ هر طرف که خاستی ببریش .

بزار یک کار عملی انجام بدیم تا طرز راه اندازی یا شبیه سازی میکروکنترلر رو توی محیط برنامه یاد بگیریم فرض کن که می خایم یک مدار ساده مثلا یک چشمک زن رو با **AVR** درست کنی که از این ساده تر دیگه وجود نداره این چشمک زن رو می خایم با هشت **LED** درست کنی که همه هم زمان با هم روشن خاموش می شن. برای این کار اول می یایم توی محیط پروتوس تا قطعات رو به هم وصل کنیم . خوب طبق همون روشی که گفته بودم قطعات رو می یاری میکروکنترلر رو از نوع **AT90S8535** انتخاب می کنیم و همچنین اول میکرو کنترلر رو میاریم در ضمن اند این هشتا **LED** رو به ترتیب به پایه های **A.0** تا **A.7** وصل می کنیم و سر دیگر این **LED** ها رو به **GND** وصل کن **GND** رو هم همون طور که بهت یاد دادم بیار بعد برو روی توی منوی **FILE** و روی گزینه **SAVE DESING AS** کلیک کن و مسیری رو که برای **SAVE** کردن مدار ازت می خاد رو بده همون جایی که نرم افزار رو نصب کردی یعنی توی پوشه **SAMPLE** البته بهتره توی این پوشه قبلش یک پوشه جدید ایجاد کنی و مدار تو توی این پوشه جید **SAVE** کنی .

1- من یک چیز رو نفهمیدم برنامه ای نمی خای براش بنویسی و برنامه رو چطوری می خای توش بریزی
2- اگه یکم صبر کنی به اونجاشم می رسیم . خوب تا اینجا که قسمت سخت افزاری مدار رو انجام دادیم حالا باید برای این مدار یک برنامه هم بنویسیم که اونم خیلی راحتیه خوب **BASCOM** رو باز کن و یک صفحه جدید ایجاد کن و برنامه زیر رو بنویس :

```
CONFIG PORTA = OUTPUT
DO
TOGGLE PORTA
WAITMS 100
LOOP
```

برای انتخاب نوع میکروکنترلر همونطور که بهت یاد دادم برو از داخل خود **BASCOM** انتخابش کن و حتما **AT90S8535** رو انتخاب کن و برای انتخاب فرکانس کاری هم بازم همونطوری که بهت یاد دادم از داخل خود نرم افزار انتخابش کن و بزارش روی **8000000** هر تری بعد که این تنظیمات رو انجام دادی و **OK** کردی بیا اول دکمه **F7** رو بزنی تا برنامه چک بشه تا خطا نداشته باشه وقتی که این دکمه **F7** رو میزنی قبل از اینکه برنامه رو چک کنه ازت یک مسیر میخاد که برنامه رو کجا ذخیره کنه که شما باید دقیقا ادرس رو همون جایی بدی که مدار تو اونجا **SAVE** کردی خوب کار ما با **BASCOM** تمام شد و اگه حال کردی می تونی ببندیش و بری سر نرم افزار پروتوس . اول پروتوس رو باز کن بعد برو توی منوی **FILE** و گزینه

LOAD DESING رو انتخاب کن تا یک پنجره باز بشه بعد از طریق این پنجره برو اون جایی که مدار تو **SAVE** کردی و مدار تو انتخاب کن تا روی صفحه بیاد . خوب گفتی که چطوری برنامه ای رو که نوشتیم می ریزیم توی میکروکنترلر خوب الان بهت می گم برای این کار باید روی میکروکنترلر یک بار کلیک راست کنی تا انتخاب بشه و به رنگ قرمز در بیاد بعد یک بار کلیک چپ می کنی تا یک پنجره باز بشه داخل این پنجره یک جایی نوشته **PROGRAM FILE** که جلوش یک کادر هستش و جلوی این کادر یک شکل زرد رنگ هستش که باید روی این شکل کلیک کنی تا دوباره یک کادر دیگه باز بشه . به وسیله این کادری که الان باز شد باید بری برنامه ای که نوشتی رو از اونجایی که **SAVE** قبلا کردی انتخاب کنی منظور فایلی هستش که با پسوند **HEX** هستش بعد که انتخاب کردی و **OK** رو زدی برنامه خودکار توی میکرو قرار می گیره بعد باید برای انتخاب فرکانس کاری میکرو باید در کادری که نوشته **CLOCK FREQUENCY** مقدار فرکانس میکرو رو بنویسی البته به مگا هر تری **MHZ8** . خوب کار دیگه تموم شد **OK** رو می زنی تا پنجره بسته بشه بعد برای اجرای این مدار باید بری پایین صفحه و روی علامتی که شبیه **PLAY** ضبط هستش کلیک کنی که شبیه مثلث هستش و می بینی که هر هشتا

LED دارن باهم ديگه هم زمان چشمک می زنن. در ضمن اگر برنامه **EROR** داد این ديگه تقصير من نيست چون من به قلق نرم افزاره مسلط نيستم و بايد بری توی این تاپيکی که بهت الان لينکشو می دم و مشکلتو اونجا مطرح کنی <http://www.eca.ir/forum2/index.php/board,181.0.html> موفق باشی.

پایان قسمت پنجم

قسمت ششم:

1-سلام خوبی

2-سلام قربانت تو چطوری

1-خوبم . امروز چی می خای یادم بدی

2-اول بگو که با این پروتوس مشکلی نداشتی

1-این پروتوس اوایل **eror** میداد ولی رفتم مشکلم رو حل کردم

2-خوبه . امروز می خام طرز اتصال صفحه کلید ماتریسی به میکروکنترلر همراه با برنامه اون بهت بگم

1-صفحه کلید ماتریسی ديگه چیه

2-همون صفحه کلیدی که ماشین حسابا و تلفن ها دارن

1-اها ادامه بده

2-این صفحه کلید ها از نظر سخت افزاری **8** تا پایه دارن که مستقیم وصل میشن به میکروکنترلر و یک پوت کامل رو اشغال می کنن

1-منظورت از اینکه یک پورت کامل رو اشغال می کنن چیه ؟

2-منظورم اینه که باید وصل بشن به هشتا از پایه های میکرو مثلا وصل بشن به پورت **A** یا پورت **B** یا هر پورتهی فرقی نداره کدوم پورت ولی باید اگه می خای به پورت **A** وصل کنی باید فقط به پایه های پورت **A** وصل بشن نباید مثلا چندتا رو به پایه های **A** و چندتای ديگشو وصل کنی مثلا به پایه های پورت **B** .

خوب بزار بهتر بهت بگم که باید چطوری وصلش کنی

1-یک لحظه وایستا اصلا چرا ما باید کیبرد رو وصل کنیم به میکرو خوب کاربردشم بگو ديگه مارو گیج کردی
2-باشه معذرت می خام . فرض کن که می خای یک ماشین حساب با میکروکنترلر درست کنی یا اینکه می خای اعداد **0** تا **15** رو روی **LCD** نمایش بدی . یا یکم پیشرفته تر می تونی یک قفل رمز الکترونیکی درست کنی
1-حالا فهمیدم ادامه بده

2-خوب می خاستم بهت بگم که چطوری کیبرد رو به میکرو وصل کنی . خوب این خیلی ساده است ابتدا یک پورتهی که حال کردی کیبرد رو بهش وصل کنی رو در نظر می گیری خوب گفتم که کیبرد **8** تا پایه داره خوب میکرو هم هر پورتهش **8** تا پایه داره مثلا می خای به پورت **A** وصل کنی . خوب برای این کار پایه اول کیبرد رو وصل می کنی به پایه **A.0** پایه دوم رو وصل می کنی به پایه **A.1** و الی آخر که باید پایه آخر یعنی پایه هشتم کیبرد رو وصل کنی به پایه **A.7**

1-خوب این طرز اتصال شو فهمیدم برنامه چطوری به میکرو بدیم

2-عجله می کنی یا . خوب حالا می ریم سر برنامه . قبلش بگم که وصل کردن صفحه کلید به میکرو رو در اصطلاح بهش

میگن اسکن صفحه کلید . اینو گفتم که یک موقع جایی شنیدی نگی این ديگه چه اصطلاحیه . خوب برنامه اتصال صفحه کلید ماتریسی به میکرو ساده هستش. طرز کار به این صورت هستش که ما اول یک متغیر رو از نوع بایت تعریف می کنیم بعد مقدار خونده شده از کیبرد رو می ریزیم توی این متغیر بعد با فرمان **CLS** و **LCD** اونو روی

LCD نمایش می‌دهیم به همین سادگی. خوب اول باید ما برای میکرو تعریف کنیم که اقا ما می‌خواهیم یک کیبرد ماتریسی به جنابعالی وصل کنیم تا طفلی بفهمه که این جسمی که بهش وصل شده چیه. یادت می‌یاد گفتیم که برای استفاده از امکانات میکرو باید اونها رو پیکره بندی کنی. یا به اصطلاح **CONFIG** کنی

1-اره

2-یادت میاد که برای استفاده از امکان ورودی خروجی میکرو از دستور **CONFIG PORTA = OUTPUT** برای خروجی و **CONFIG PINA.0 = INPUT** برای ورودی استفاده می‌کردیم

1-بازم اره

2-خوب یکی از امکانات میکروکنترلر **AVR** اینکه که می‌تونه کلید هایی رو که روی کیبرد فشرده میشه رو بخونه که چه عددی بوده و برای استفاده از این امکان هم مثل سایر امکانات دیگه باید از دستور **CONFIG** استفاده کنیم

CONFIG KBD = PORTB

با دستور بالا ما به میکرو فهماندیم که اقا ما می‌خایم یک کیبرد به شما متصل کنیم اون هم به پورت **B**. توجه کن که پورتی رو که جلوی علامتی مساوی می‌نویسی می‌تونه یکی از چهار پورت **A** یا **B** یا **C** یا **D** باشه ولی هر پورتی رو که می‌نویسی کیبرد رو هم باید به همون وصل کنی.

DIM A AS BYTE

خوب ما یک متغیر تعریف کردیم که مقدار خونده شده از کیبرد رو بریزیم توی این متغیر

1-منظورت از این که مقدار خونده شده از کیبرد رو می‌ریزیم توی متغیر چیه

2-خوب ببین ما وقتی که کار برنامه نویسیمون تموم شد و مدار رو راه اندازی کردیم مثلا یک کلید رو از روی صفحه کلید فشار دادیم میکرو قادره مقدار عددی این کلید فشرده شده رو بخونه مثلا کلید شما ره یک رو فشار دادیم میکرو می‌فهمه که کلید شماره یک فشرده شده خوب این مقدار عددی باید در یک جا قرار بگیره و ما جایی بهتر از اینکه این مقدار عددی رو انتقال بدیم به یک متغیر گیر نیاوردیم. البته وقتی که مطلب کیبرد رو تموم کردم کامل می‌فهمی چی به چیه. بزار یک مثال بزنم. فرض کن که ما یک کیبرد داریم بایک میکروکنترلر و یک **LCD** می‌خایم این سه تا رو به هم وصل کنیم وهنگامی که یک کلید از روی صفحه کلید فشرده شد میکرو اونو بخونه و مقدار اون عدد رو روی **LCD** نمایش بده که برنامه به صورت زیر هستش:

CONFIG KBD = PORTA

DIM A AS BYTE

:MAIN

DO

()A = GETKBD

IF A < 16 THEN GOTO MASHHAD

LOOP

END

: MASHHAD

CLS

LCD A

DO

()A = GETKBD

LOOP UNTIL A = 16

GOTO MAIN

کل برنامه همین هستش البته به غیر از معرفی میکرو و انتخاب فرکانس کاری که باید در اول برنامه نوشته بشن. خوب حالا بریم سر تحلیل برنامه.

خط اول که ما پیکره بندی کیبرد رو مشخص کردیم که می‌خاهد به پورت **A** متصل بشه

خط دوم ما یک متغییر تعریف کردیم که مقدار خوانده شده از کیبرد ریخته بشه یا به عبارتی انتقال داده بشه به داخل این متغییر

خط سوم که یک زیر برنامه تعریف کردیم

خط بعدی نوشتیم **DO** یعنی ابتدای حلقه

خط بعد نوشتیم **A = GETKBD** () با این دستور مقداری که میکروکنترلر از کیبرد می خونه میریزه توی

متغییر **A**

خط بعد نوشتیم **IF A < 16 THEN GOSUB MASHHAD** خوب با این دستور این شرط را گذاشتیم که اگر اعدادی کوچکتر از **16** رو خوندی بروی دستور شرط رو که پرش به زیر برنامه **MASHHAD** هستش رو اجرا کن خوب حالا ممکنه که از خودت بررسی که چرا کوچکتر از **16** خوب الان بهت می گم چرا . ببین ما گفتیم که کیبرد ما **15** دکمه داره که اعدادی که داره یعنی **0** تا **15** که کلا می شه **16** کلید پس ما می تونیم اعداد **0** تا **15** رو روی باهاش نشون بدیم پس باید یک دستوری رو بنویسیم که اگر عددی کوچکتر از **16** رو خونده بره دستور شرط رو اجرا کنه.

خط بعدی نوشتیم **LOOP** که پایان حلقه هستش

خط بعدی نوشتیم **END** که برنامه هیچ وقت به این **END** نمی رسه

خط بعدی نوشتیم **MASSHAD**: که همان زیر برنامه **MASSHAD** هستش که هر اسم دیگه ای می تونیم

براش بزاریم

خط بعدی نوشتیم **CLS** که باید بدونی کارش چیه

خط بعدی نوشتیم **LCD A** که با این دستور مقدار عددی که میکرو از کیبرد خونده و ریخته توی متغییر **A** رو می تونیم بخونیم امیدوارم که فهمیده باشه که چرا مقداری رو که میکرو می خونه باید بریزیم توی یک متغییر. فرض کن که ما کلیدی از کیبرد رو که روش نوشته **4** رو فشار می دیم میکرو میاد این مقدار رو میخونه و انتقالش می ده توی متغییر **A** بعد ما برای اینکه ببینیم کلیدی رو که فشردیم چه عددی بوده باید با دستور **LCD A** مقدار

متغییر **A** رو روی **LCD** نمایش بدیم

در خط های بعدی نوشتیم

DO

() A = GETKBD

LOOP UNTIL A = 16

GOTO MAIN

که این دستورات هیچ توضیحی نداره و همیشه باید بنویسیم . این دستورات برای اینه که وقتی شما یک دکمه رو فشار می دی میکرو نیاد هی اون عدد رو روی **LCD** تند تند نمایش بده . این سه خط دستور طور کلی میاد یک حالتی رو در برنامه ایجاد می کنه که تا وقتی که ما دستمون رو از روی دکمه فشرده شده برداشتیم عدد فیکس بمونه بعد وقتی که دستمون رو از روی کلید برداشتیم میاد خط بعدی رو می خونه خط اخر هم نوشتیم **GOTO MAIN** که با این دستور دوباره برمی گردیم توی حلقه اول و انقدر توی این حلقه می مومنه تا یک دکمه دیگه فشرده بشه و بره اونو بخونه به عبارتی روز از نو روزی از نو.

1- کل برنامه همین بود

2- اگه شک داری می تونی بری توی پروتوس امتحانش کنی

1- این کیبرد کجای پروتوس هستش

2- وقتی که کتابخانه نرم افزار رو باز کردی اول می ری تو قسمت **ACTIVE بعد از اونجا میتونی یک کیبرد**

انتخاب کنی

1- ممنون

2-قربانت . بزار یک مثال کاربدی تر بزنم همین برنامه رو یکم پیشرفته ترش می کنیم مثلا می خایم هر وقت که دکمه شماره 3 رو فشار دادیم یک LED روشن بشه و هر وقت که دکمه شماره 12 رو فشار دادیم LED خاموش بشه . پس ما به طور دلخواه باید یک پورت رو مثلا پورت A رو برای اتصال کیبرد به میکرو در نظر بگیریم و یک پورت دیگه هم مثلا پورت B رو برای LED در نظر بگیریم البته ما از پورت B فقط یک پایه لازم داریم که مثلا می تونه B.0 باشه . LCD هم که حال می کنیم وصل بشه به پورت C . خوب برنامه ای که ما باید بنویسیم به شرح زیر است :

```

CONFIG PORTB = OUTPUT
CONFIG PORTA = KBD
DIM A AS BYTE
:MAIN
DO
( ) A = GETKBD
IF A < 16 THEN GOTO SHOW1
IF A = 3 THEN GOTO SHOW2
IF A = 12 THEN GOTO SHOW3
LOOP
END
:SHOW1
CLS
LCD A
DO
( ) A = GETKBD
LOOP UNTIL A = 16
GOTO MAIN
:SHOW2
SET PORTB.0
CLS
LCD A
DO
( ) A = GETKBD
LOOP UNTIL A = 16
GOTO MAIN
:SHOW3
RESET PORTB.0
CLS
LCD A
DO
( ) A = GETKBD
LOOP UNTIL A = 16
GOTO MAIN

```

خوب این هم یک برنامه کامل برای این مثالی که زدیم فقط باید نوع میکرو و فرکانس کاری میکرو رو خودت بنویسی یا اینکه از داخل نرم افزار تنظیمشون کنی . خوب حالا بریم سر تحلیل برنامه:

میکرو میاد از خط اول برنامه شروع می کنه به خوندن یعنی عبارت **CONFIG PORTB = OUTPUT** و می فهمه که ما یک خروجی رو پیکره بندی کردیم . بعد میاد خط دوم رو می خونه که نوشته **CONFIG PORTA = KBD** و می فهمه که قراره یک کیبرد بهش وصل بشه. بعد خط بعدی رو می خونه **DIM A AS BYTE** این

یعنی اینکه ما یک متغییر تعریف کردیم برای همون که قبلا بهت گفتم که می خایم عددی رو که میکرو از کیبرد می خونه بریزه توی یک متغییر برای همون یک متغییر تعریف کردیم . خط بعدی یک زیر برنامه به اسم **MAIN** تعریف کردیم که می تونست هر اسم دخواه دیگه ای با شه کاربردش هم در ادامه بهت می گم که چرا گذاشتیمش. خط بعدی نوشتیم **DO** که با این کار ابتدای یک حلقه رو تعریف کردیم. خط بعد نوشتیم که **A = GETKBD** () که این جمله کوچیک رو می تونم اینطوری ترجمه کنم که آقای میکرو هر دکمه ای رو که فشرده شد عدد او دکمه فشرده شده رو بریز توی این متغییر **A** . در خط های بعدی سه شرط گذاشتیم که شرط های ما همیشه باید در داخل حلقه **DO-LOOP** قرار بگیره

1-چرا باید داخل حلقه قرار بگیره

2-چون که یکسره میکرو بیاد این شرط ها رو چک کنه و اگر که شرطی اجرا شد بره دستور شرط رو اجرا کنه
1-اها ادامه بده

خوب گفتیم که ما داخل این حلقه سه تا شرط گذاشتیم . اولی شرط نوشتیم که **IF A < 16 THEN GOTO SHOW1**

به این معنی که اگر **A** کوچکتر از **16** بود برو دستور شرط رو که بعد از **THEN** نوشتیم رو برو اجرا کن خوب حالا دستور شرط چیه اینه : **GOTO SHOW1** یعنی اگر شرط مورد نظر اجرا شد پیر برو توی زیر برنامه **SHOW1** حال توی زیر برنامه **SHOW1** چیکار باید انجام بده رو بهش می رسیم که از این شرط برای خواندن عددی که دکمه متناظر با اون فشرده شده استفاده میشه.

شرط دوم نوشتیم که **IF A = 3 THEN GOTO SHOW2** که با این جمله این مطلب رو به میکرو فهموندیم که اگر خدای ناکرده دکمه شماره **3** فشرده شد برو دستور شرط رو که پرش به زیر برنامه **SHOW2** هستش رو اجرا کن.

شرط سومی که گذاشتیم اینه **IF A = 12 THEN GOTO SHOW3** یعنی اگر دکمه شماره **12** فشرده شد برو دستور شرط رو که پرش به زیر برنامه **SHOW3** هستش رو اجرا کن. بعد از این که شروط رو نوشتیم **LOOP** رو که پایان حلقه هستش رو گذاشتیم . در خط بعد نوشتیم **END** که برنامه ما هیچ وقت به این **END** نمیرسه و نباید هم برسه . در خط های بعدی سه تا زیر برنامه به نام های **SHOW1** و **SHOW2** و **SHOW3** تعریف کردیم البته که در داخل این زیر برنامه ها چند تا دستور نوشتیم که توضیحات این زیر برنامه ها رو حین تحلیل دوم برنامه بهت می گم .

تحلیل دوم برنامه : خوب فرض کن که برنامه رو نوشتیم و ریختیم توی میکروکنترلر حالا بهت می گم که می کرو با خواندن این دستورات چیکار می کنه یا به عبارتی دیگه چکار باید بکنه . در دو خط اول که ما یک پورت برای خروجی تعریف کردیم و یک پورت برای اتصال به کیبرد که میکرو این دو خط رو می خونه و می ره سر خط سوم در این خط یک متغییر تعریف کردیم که این خط رو هم می خونه و میره سر خط بعدی. خط بعدی زیر برنامه **MAIN** رو می خونه و می ره سر خط بعدی . خط بعدی که ما یک **DO** گذاشتیم و میکرو این رو هم می خونه و هم زمان که اینو خوند می فهمه که ابتدای یک حلقه رسیده و همونجا

میوفته توی این حلقه و میکرو میاد کل چهار خط این حلقه رو می خونه

1-منظورت از چهار خط داخل این حلقه همون خط اول + سه تا شرط بعدیشه

2-اره عزیز . بعد که این چهار خط رو خوند میرسه به LOOP می فهمه که باید بره دوباره از خط زیر **DO** شروع کنه به خواندن بعد دوباره میاد این کار رو می کنه و باز دوباره اون چهار خط رو میخونه و بعدش دوباره می رسه به **LOOP** و دوباره می فهمه که باید بره برنامه رو از خط زیری **DO** شروع کنه به خواندن بعد دوباره این کار رو می کنه . به عبارت دیگه می تونم بهت بگم که داخل این حلقه یکسره دور می زنه

1-خوب پس کی بیرون میاد

2-اها این همون سوالی بود که الان می خواستم جوابشو بگم . وقتی بیرون میادش که یکی از دستورات شرط اجرا بشه فرقی هم نمی کنه که کدوم یکی از سه تا شرط باشه

1-حالا شما فرض کن که شرط اولی اجرا شده خوب چه اتفاقی میوفته

2- شرط اولی هستش **IF A < 16 THEN GOTO SHOW1** که مربوط به خوندن عدد دکمه ای است که روی کیبرد فشرده شده. اگر این شرط اجرا بشه دستور شرط هم اجرا می شه دستور شرط همون جمله بعد از **THEN** هستش که گفته پیر برو توی زیر برنامه **SHOW1** و میکرو هم همین کار رو میکنه و مییره میره توی زیر برنامه **SHOW1** و دستوراتی رو که داخل این زیر برنامه نوشتیم رو اجرا میکنه حالا توی زیر برنامه **SHOW1** نوشتیم که:

```
CLS
LCD A
DO
  ( ) A = GETKBD
LOOP UNTIL A = 16
GOTO MAIN
```

خط اول داخل زیر برنامه **SHOW1** هستش **CLS** که میکرو با این دستور می ره و صفحه **LCD** رو پاک می کنه خط دوم نوشتیم **LCD A** یعنی این که مقدار متغییر **A** رو که عدد دکمه ای هست که فشرده اونها ریخته شده رو با این دستور میره روی **LCD** نمایش می ده حالا هر عددی که باشه فرقی نداره می ره و نشون می ده سه خط بعدی هم که قبلا برات توضیح دادم که برای اینه که تا وقتی که دستمون رو از روی دکمه برنداشتیم همون عدد رو روی **LCD** نمایش بده

خط بعدی نوشتیم **GOTO MAIN** که در اخر این زیر برنامه قرار داره به این معنی که پیر برو توی زیر برنامه **MAIN** و میکرو هم میره همین کار رو میکنه . بعد از پریدن توی زیر برنامه **MAIN** میکرو میاد خط بعدیشو می خونه نوشته **DO** و باز دوباره میوفته توی حلقه و دوباره توی این حلقه دور می زنه تا یکی از شروط اجرا بشه . فرض کن که این دفعه شرط دوم اجرا بشه البته فرقی نمیکنه که کدوم شرط اجرا بشه خوب گفتیم که فرض رو بر این میزاریم که شرط دوم اجرا شده یعنی

IF A = 3 THEN GOTO SHOW2 که ما با این دستور این شرط را گذاشته بودیم که اگر دکمه شماره **3** فشرده شد برو دسترو شرط رو اجرا کن که اگر این شرط اجرا بشه دستور شرط هم که پرش به زیر برنامه **SHOW2** هستش هم اجرا میشه و میکرو هم همین کار رو می کنه و میاد مییره توی زیر برنامه **SHOW2** و دستورات داخل این زیر برنامه رو اجرا می کنه که دستورات داخل این زیر برنامه به قرار زیر است :

```
:SHOW2
SET PORTB.0
CLS
LCD A
DO
  ( ) A = GETKBD
LOOP UNTIL A = 16
GOTO MAIN
```

خط اول نوشتیم که **SET PORTB.0** به این معنی که پایه **B.0** رو یک کن

خط بعدی نوشته شده **CLS** یعنی اینکه صفحه **LCD** رو پاک کن

خط سوم نوشتیم که **LCD A** یعنی نشون بده مقدار متغییر **A** رو که باید عدد **3** رو نشون بده

1- چرا **3**

2- چون که ما داخل اون حلقه شرط گذاشته بودیم که اگه دکمه شماره **3** فشرده شده بود پیر برو توی زیر برنامه **SHOW2** خوب پس دکمه ای که فشرده شده شماره سه هستش دیگه پس باید حتما عدد **3** رو نشون بده

1- اخر ما نفهمیدیم که این شرط بندی حلاله یا حرام

2- تو این جور موارد حلاله . در سه خط بعدی نوشتیم :

DO

() A = GETKBD

LOOP UNTIL A = 16

که اینو بهت گفتم برای چی . برای فیکس شدن عدد روی **LCD** تا موقعی که دستمون رو از روی دکمه برداشتیم در خط بعدی هم نوشتیم **GOTO MAIN** که میکرو با خوندن این دستور می پره میره اولین خط بعد از این زیربرنامه که نوشته **DO** رو می خونه و می فهمه که ابتدای دردرس یعنی دور زدن توی این حلقه شروع شده

1-خوب این LED کی خاموش می شه

2-وقتی که شرط سوم یعنی فشرده شدن دکمه شماره 12 اجرا بشه که شرطی که براش گذاشتیم از این قراره

IF A = 12 THEN GOTO SHOW3 این شرط رو همیشه اینطوری ترجمه کرد که اگه دکمه شماره **12**

ازروی کیبرد فشرده شد پپر برو توی زیربرنامه **SHOW3** . حالا توی این زیربرنامه **SHOW3** این دستورات

هستش :

:SHOW3

RESET PORTB.0

CLS

LCD A

DO

() A = GETKBD

LOOP UNTIL A = 16

GOTO MAIN

خط اول بعد از زیر برنامه نوشته شده **RESET PORTB.0** به این معنی که پایه **B.0** رو **0** کن و در اینجا اون **LED** که گفتیم خاموش میشه .

خط بعدی نوشتیم **CLS** یعنی پاک کن صفحه ال سی دی رو

خط بعدی **LCD A** هستش یعنی مقدار متغییر **A** رو نشون بده که باید مقدار **12** رو روی **LCD** نمایش بده

سه خط بعدی هم از قرار زیر هستش :

DO

() A = GETKBD

LOOP UNTIL A = 16

که باید دیگه بدونی مال چی هستش

خط اخر هم **GOTO MAIN** هستش که بعد از این که دستورات داخل زیربرنامه اجرا شد و میکرو به این خط

رسید با خوندن این خط میره میره توی زیربرنامه **MAIN** و میکرو هم

1-بزار من بقشو بگم ببین یاد گرفتم یا نه

2-بفرمایید

1-بعد از این که پرید توی زیربرنامه MAIN اولین خط نوشته DO که دوباره میوفته توی این حلقه . بقیشو

خودت بگو

2-خوب بود معلومه که یاد گرفتی . بعد توی این حلقه اونقدر دور میزنه تا دوباره یک شرط دیگه از سه شرط اجرا بشه .

این هم از تفسیر برنامه گمون نکنم که از این ساده تر بشه تفسیرش کرد.

یک نکته ای که من باید بگم اینه که تو هر دستور شرطی رو می تونی توی این زیر برنامه ها قرار بدی مثلا بعد از

این که **PORTB.0** رو **SET** کرد یک پشندش یعنی خط پایینیش بنویسیم که **CLS**

"LCD "AHMAD که با اینکار میداد هم پایه **B.0** رو **SET** میکنه و هم روی **LCD** یک چیزی می نویسه و

خیلی کارای دیگه ای هم می تونی بکنی که بستگی به خودت داره

یک نکته دیگه اینکه گاهی وقتا لازم هستش که دکمه های صفحه کلید رو مثل ماشین حساب پیکره بندی کنی که باید از جدول **LOOKUP** استفاده کنی که من الان حضور ذهن ندارم که چطوری بودش یادم رفته باید مراجعه کنم استادم.

درس بعدی استفاده از توابع ریاضی هستش که من همه اونا رو بهت نمی گم چون که کاربرد نداره علامت جمع + : به مثال زیر توجه کن

```
DIM A AS BYTE
A = 5 + 1
CLS
LCD A
END
```

این برنامه هیچ نیازی به توضیح ندارد چون همه چیزش شفاف هستش . فقط همینو بگم که در مثال بالا ما یک متغییر تعریف کردیم مقداری رو که به متغییر دادیم **5+1** هستش میکرو میاد با امکاناتی که داخل خودش داره این دو عدد رو باهم جمع می کنه و حاصل که عدد **6** هست رو هنگامی که ما دستور **LCD A** رو می خونه یعنی عدد **6** روی **LCD** نمایش میده . بزار یکم پیشرفته ترش کنیم

```
CONFIG PORTB = OUTPUT
DIM A AS BYTE
A = 5+1
IF A = 5 THEN SET PORTB.0
IF A = 6 THEN SET PORTB.1
IF A = 7 THEN SET PORTB
END
```

در برنامه بالا ما یک خروجی تعریف کردیم . بعد یک متغییر از نوع بایت تعریف به نام **A** تعریف کردیم بعد اومدیم برای مقدار دهی به متغییر **A** از جمع دوعدد استفاده کردیم یعنی مقدار متغییر **A** هست **5+1** . بعد اومدیم سه تا شرط و سه تا دستور شرط گذاشتیم اولین شرطی که گذاشتیم این بود که اگر مقدار متغییر **A** شد **5** برو و دستور شرط رو که هست **SET PORTB.0** رو اجرا کن . شرط دوم اومدی گفتیم که اگه مقدار متغییر **A** شد **6** برو دستور شرط رو که هست **SET PORTB.1** رو اجرا کن . شرط سوم گفتیم که اگه مقدار متغییر **A** شد **7** برو و دستور شرط رو که هست **SET PORTB** رو اجرا کن که یعنی یک کردن کل هشتا پایه پورت **B** . برای امروز دیگه کافیه بقیش باشه برای جلسه بعدی پایان قسمت ششم

قسمت هفتم :

1-سلام چطوری

2-سلام ممنون چی کار می کنی

1-قربانت هیچکار . امروز اومدم که بقیه میکرو رو بهم یاد بدی

2-چشم . حالا کجا بودیم

1-توی قسمت جمع و تفریق و ...

2-اها یادم اومد جمع رو گفتیم بقیش موند . باشه . عملیات بعدی تفریق هستش که بایک مثال بهت می گم که

چطوری باید عمل کنی

```
DIM G AS BYTE
```


G = 6 - 2
CLS
LCD G
END

مثال بالا یک مثال خیلی ساده هستش . ما در خط اول یک متغییر تعریف کردیم به نام **G** در خط دوم بهش مقدار **2-6** رو دادیم که یعنی **4** . در خط بعدی با فرمان **CLS** ال سی دی را پاک کردیم تا آماده نوشتن بشه . در خط بعدی با فرمان **LCD G** از میکرو خواستیم تا مقدار متغییر **G** را روی **LCD** نمایش دهد که میکرو برای ما عدد **4** را نمایش می دهد یعنی حاصل عبارت **2-6** را نشان می دهد . در خط آخر هم با دستور **END** برنامه رو به پایان رساندیم . این برنامه بالا رو اگه دوست داشتی می تونی با شبیه ساز خود **BASCOM** نگاه کنی .

یک مثال دیگه:

DIM L AS BYTE
DIM N AS BYTE
L = 12 - 3
N = 2 - 2
CLS
LCD L
WAITMS 100
CLS
LCD N
END

خوب توی برنامه بالا ما اومدیم دو تا متغییر به نام های **N** و **L** از نوع بایت تعریف کردیم . در خط سوم به متغییر **L** مقدار **12 - 3** دادیم . در خط بعدی به متغییر **N** مقدار **2-2** دادیم . در خط بعدی با فرمان **CLS** ال سی دی را پاک کردیم و در خط بعد با فرمان **LCD L** به میکرو دستور دادیم که مقدار متغییر **L** را که هست **12 - 3** یعنی **9** رو نمایش بده و میکرو مقدار **9** را برای متغییر **L** نمایش میدهد . در خط بعد ما یک تاخیر در برنامه گذاشتیم . در خط بعدی با فرمان **CLS** ال سی دی را مجددا پاک کردیم و در خط بعدی با فرمان **LCD N** به میکرو دستور دادیم که مقدار متغییر **N** رو که هست **2-2** یعنی **0** رو به ما نشون بده که میکرو مقدار **0** رو روی **LCD** نمایش میدهد و در خط آخر هم برنامه به پایان می رسه .

یک مثال دیگه هم رو حساب معرفت میزنم :

DIM L AS BYTE
DIM N AS BYTE
DIM AHMAD AS BYTE
L = 12 - 3
N = 10 - 2
AHMAD = L + N
CLS
LCD L
WAITMS 100
CLS
LCD N
WAITMS 100
CLS
LCD AHMAD
END

این برنامه رو خودت می دونی که چی به چیه ولی بازم خودم می گم . ببین ما سه تا متغییر تعریف کردیم هر سه تا هم از نوع بایت هستن به نام های **L** و **N** و **AHMAD** . به متغییر های **L** و **N** مقدار دادیم و به متغییر **AHMAD** مقداری که دادیم این بود که آقای میکروکنترلر حاصل متغییر های **L** و **N** رو با هم جمع کن و حاصل را مقدار

متغییر **AHMAD** قرار بده خوب مقدار متغییر **L** هست **12 - 3** یعنی **9** و مقدار متغییر **N** هست **10 - 2** یعنی **8**. ما با دستور **AHMAD = N + L** این فرمان را دادیم که مقدار متغییر **AHMAD** هست **9 + 8** یعنی **17**.

خوب بقیه دستورات هم به این شکل هستند فقط به جای علامت جمع و تفریق کافیه علامت های ضرب و تقسی رو بزاری که علامت تقسیم هستند /

و علامت ضرب هستند *

همینطور اینها رو با هم جمع و تفرق و یا ضرب و تقسیم کن تا کامل دستت بیاد یا یک شرط بزار که مثال اگه مقدار عبارت

4 - 1 شد عدد **3** برو فلان کار رو انجام بده یا خیلی کارای دیگه میتونی انجام بدی.

1 - خوب درس بعدی چیه ؟

2 - راستش بخای من قبل از این که بخوام تایمر و کانتر رو بهت بگم باید یک سری از چیزهایی رو که قبلا نگفتم بهت بگم مثلا بعضی از دستورات مربوط به **LCD** رو نگفتم بعضی از دستورات پرش رو نگفتم که باید همون اول می گفتم

1 - خوب چرا همون اول نگفتمی

2 - خوب ترسیدم که باهم قاطی کنی بلایی که سرخودم تو کلاس اومده . استاد میومد چند تا دستور شبیه به هم رو که همش یک کار رو انجام می داد باهم می گفت بقیه هم قاطی می کردن

1 - مثلا چی

2 - مثلا همین دستور **GOTO** که پرش به یک زیر برنامه هستند رو با دستور **GOSUB** هم می شه انجام داد که هر دو تاش یک کار رو انجام میدن . یا بعضی از دستورات مربوط به **LCD** رو مثلا خاموش روشن کردن **LCD** - خاموش روشن شدن کرسر

و چند تا دستور دیگه .

1 - اینا خیلی سخته

2 - نه بابا از دستورات جمع و تفرق اسون تره . بعد که اینا رو گفتم می ریم سر تایمر و کانتر.

اول بریم سر بقیه دستورات **LCD** : خوب ما از ال سی دی فقط دو دستور **CLS** و **LCD** رو با هم مرور کردیم دستورات بعدی به این شرح هستند :

دستور **CONFIG LCD** : توسط این دستور می شه اندازه **LCD** رو مشخص کرد . مثال :

DIM A AS BYTE

A = 10

CONFIG LCD = 16 * 2

CLS

LCD A

END

خوب برنامه بالا خیلی راحت ما یک متغییر تعریف کردیم بعد بهش مقدار دادیم . بعد اندازه **LCD** رو مشخص کردیم . وبعد مقدار متغییر رو روی **LCD** نمایش دادیم. در نهایت برنامه رو به پایان رسوندیم .

1 - هر اندازه ای که خاستیم می تونیم بدیم ؟

2 - نه اندازه ها باید استاندارد باشه و باید یکی از انواع اندازه های زیر باشه :

1 * 16 - 2 * 16 - 4 * 20 - 2 * 20 - 4 * 40 - 2 * 40 * 4

درضمن این اندازه ها رو می تونی از داخل نرم افزار که قبلا توضیح دادم هم تنظیم کنی

دستور بعدی **DISPLAY** هستند که جلوی این دستور باید یکی از دو گزینه **ON** یا **OFF** قرار بگیره . توسط

این دستور میشه ال سی دی رو روشن یا خاموش کرد . مثال :

DIM AHMAD AS BYTE

AHMAD = 3
CLS
LCD AHMAD
WAITMS 100
DISPLAY OFF
END

خوب در این برنامه ما یک متغییر به نام **AHMAD** از نوع بایت تعریف کردیم . بعد در خط دوم به متغییر **AHMAD** مقدار **3** رو دادیم . در خط بعدی با دستور **CLS** ال سی دی رو پاک کردیم . در خط بعدی توسط دستور **LCD AHMAD** دستور دادیم که مقدار متغییر **AHMAD** رو روی ال سی دی نمایش بده . در خط بعدی یک تاخیر **100** میلی ثانیه ای در برنامه ایجاد کردیم بعد از گذشت این مدت زمان میکرو خط بعدی رو می خونه که نوشته **DISPLAY OFF** و توسط این دستور **LCD** خاموش میشه. و در خط آخر هم برنامه به پایان می رسه. در ضمن هر وقت که خاصی **LCD** رو که با دستور **DISPLAY OFF** خاموش کرده بودی دوباره روشنش کنی باید با دستور **DISPLAY ON** این کار رو بکنی.

دستور بعدی **CURSOR** هستش که جلوی این دستور یکی از چهار گزینه زیر باید قرار بگیره :

ON : که اگه **ON** جلوی **CURSOR** قرار بگیره **CURSOR** فعال میشه

OFF : که اگه **OFF** جلوی **CURSOR** قرار بگیره **CURSOR** خاموش میشه

BLINK : اگر **BLINK** جلوی **CURSOR** قرار بگیره **CURSOR** چشمک میزنه

NOBLINK : که اگه **NOBLINK** جلوی **CURSOR** قرار بگیره **CURSOR** چشمک نمی زنه

1- خوب حالا این CURSOR چی هستش

2-CURSOR یا به فارسی کرسر همون خطی هستش که وقتی داری مثلا با برنامه **WORDPAD** تایپ می کنی هی چشمک می زنه توی **LCD** هم این کرسر قرار داره که می تونه خاموش یا روشن یا چشمک زن باشه که دست خودمونه که چه بلایی سرش بیاریم .

مثال :

DIM A AS BYTE
A = 100
CURSOR BLINK
CLS
LCD A
END

خوب ما یک متغییر تعریف کردیم به نام **A** و از نوع بایت . در خط دوم مقدار دادیم . در خط سوم توسط دستور **CURSOR BLINK** کرسر را از نوع چشمک زن انتخاب کردیم . در خط چهارم با دستور **CLS** ال سی دی رو پاک کردیم . در خط بعد با دستور **LCD A** مقدار متغییر **A** رو روی ال سی دی نمایش دادیم. در خط آخر هم توسط دستور **END** برنامه رو به پایان رساندیم.

دستور بعدی **SHIFLCD** هست که بعد از این دستور باید از یکی از دو فرمان **LEFT** یا **RIGHT** استفاده کرد . توسط این دستور می توان کل صفحه نمایش رو به اندازه یک واحد به چپ یا راست منتقل کرد . مثال :

CLS
"LCD "AHMAD
SHIFTLCD RIGHT
END

این یک برنامه ساده هستش که ما اول اومدیم توسط دستور **CLS** ال سی دی رو پاک کردیم بعد توسط دستور **"LCD "AHMAD** اومدیم ثابت **AHMAD** رو روی ال سی دی نمایش دادیم در خط بعد توسط دستور **SHIFTLCD RIGHT** اومدیم کل صفحه ال سی دی رو یک واحد به سمت راست انتقال دادیم .

1-این ثابت AHMAD رو که گفتی یعنی چی ؟

2- هر چیز که در داخل " " قرار بگیره همیشه یک ثابت

1- یک سوال دیگه اگه ما خواستیم که دو یا چند واحد کل صفحه ال سی دی رو به سمت راست یا چپ هدایت کنیم باید چکار کنیم؟

2- به ازای هر چند واحد که خواستیم کل صفحه ال سی دی رو به سمت راست هدایت کنیم باید دستور **SHIFTLCD** رو زیر هم بنویسیم. مثلا می خایم **4** واحد به سمت راست هدایت بشه از روشی که گفتیم باید به صورت زیر عمل کنیم:

CLS
"LCD "AHMAD
SHIFTLCD RIGHT
SHIFTLCD RIGHT
SHIFTLCD RIGHT
SHIFTLCD RIGHT
END

که میکرو به ازای هر دفعه که دستور **SHIFTLCD** رو می بینه یک واحد کل صفحه ال سی دی رو به سمت راست هدایت می کنه

1- اگه این دستور **SIFTLCD** رو هزار بار زیر هم بنویسم چی میشه

2- مگه مرض داری ولی خوب در نهایت بعد چند بار اون نوشته روی ال سی دی از **LCD** خارج میشه.

دستور بعدی **LOCATE** هستش که جلوی این دستور باید مقدار بدی که در ادامه می گم چه مقداری باید بدی. توسط این دستور می تونی مکان نوشته ای که می خای روی ال سی دی نوشته بشه رو انتخاب کنی یا تغییر بدی مثلا شاید حال کردی که یک کلمه رو وسط ال سی دی نمایش بدی یا پایین سمت چپ یا هر جای ال سی دی که حال کردی نمایش بدی اونجاست که این دستور به کارت میاد. بین هر ال سی دی بسته به این که اندازه ان چند در چند باشه تعدادی ستون افقی و تعدادی ستون عمودی داره مثلا در ال سی دی **16 * 2** تعداد ستون افقی **16** و تعداد ستون عمودی **2** هستش یا در ال سی دی **40 * 4** تعداد ستون افقی **40** و تعداد ستون عمودی **4** هست. خوب حالا که فهمیدی اندازه هایی که برای ال سی دی مشخص می کنن چه مفهومی داره می تونی اون مقداری رو که گفتیم باید جلوی دستور **LOCATE** بنویسی رو به دست بیاری که باید به ترتیب زیر عمل کنی :

LOCATE 1 , 1

در بالا من یک نمونه مکان نما رو نوشتم البته هر مقداری که خواستی می تونی بدی به شرط این که مقدار رو خیلی زیاد ندی چون ممکنه اون کلمه ای که قرار روی ال سی دی نوشته بشه از ال سی دی بره بیرون. عدد اولی رو که نوشتم به این منظور بوده که مقدار مکان عمودی رو مشخص کردم. عدد دومی که جلوش نوشتم مقدار پیشروی کلمه به جلو در سمت ستون های افقی هستش. این اعداد برای هر نوع ال سی دی ممکنه فرق کنه مثلا در ال سی دی نوع **16 * 2** به علت این که تعداد ستون های عمودی ما **2** تا بیشتر نیست برای مقدار دهی به دستور **LOCATE** ما مجاز نیستیم که برای ستون عمودی بیشتر از **2** و برای ستون افقی بیشتر از **16** بدهیم. ویا در ال سی دی **16 * 1** فرق میکنه و به علت اینکه ما یک عدد ستون عمودی داریم نمی توانیم برای مقدار دهی به دستور **LOCATE** برای ستون عمودی بیشتر از **1** بدهیم و برای مقدار دهی برای ستون افقی بیشتر از **16** بدهیم.

1- خوب حالا فکر کن که من از سر کنجکاوای بیشتر از این عدد های مجاز دادم چه اتفاقی میوفته؟

2- اتفاق خاصی نمیوفته فقط شما چیز روی ال سی دی نمی بینی

مثال :

CONFIG LCD = 16 * 2
DO
CLS
LOCATE 2 , 1
"LCD "AHMAD

WAITMS 50
CLS
LOCATE 1 , 1
"LCD "AHMAD
WAITMS 50
LOOP
END

برنامه بالا میاد یک بار کلمه ثابت **AHMAD** رو یک بار در قسمت پایین ال سی دی نمایش می ده و یک بار بالای ال سی دی .

خوب حال تحلیل برنامه :

در خط اول ما اومدیم به میکرو فهموندیم که آقای میکروکنترلر ال سی دی که ما داریم استفاده می کنیم از نوع **16 * 2** هستش

در خط دوم ما با دستور **DO** ابتدای حلقه رو مشخص کردیم

در خط سوم با دستور **CLS** ال سی دی رو پاک کردیم

در خط چهارم با دستور **LOCATE 2 , 1** مکان یا نقطه ای از ال سی دی رو که می خای چیز رو روش نمایش بدیم رو مشخص کردیم که با نوشتن عدد **2** این رو به میکرو فهموندیم که باید در خط دوم عمودی یعنی پایین ال سی دی بنویسی . و با نوشتن عدد **1** بعد از عدد دو این رو به میکرو فهموندیم که باید در اول ستون افقی ال سی دی چیزی بنویسی .

در خط پنجم با دستور **"LCD "AHMAD** کلمه **AHMAD** رو در خط پایین اول ال سی دی نمایش دادیم

در خط ششم با دستور **WAITMS 50** یک تاخیر **50** میلی ثانیه ای در اجرای برنامه ایجاد کردیم .

در خط هفتم با دستور **CLS** دوباره ال سی دی رو پاک کردیم .

در خط هشتم با دستور **LOCATE 1 , 1** دوباره یک ادرس مکان برای جایی که میکرو باید روی ال سی دی نمایش بدهد دادیم که هر دو رو عدد **1** دادیم یعنی اول ال سی دی

در خط بعدی با دستور **"LCD "AHMAD** دوباره کلمه **AHMAD** رو روی ال سی دی نمایش دادیم با این تفاوت که مکانی که این دفعه ال سی دی کلمه **AHMAD** رو نمایش میده فرق کرده و اول ال سی دی داره نمایش می ده نه این که در خط پایین

در خط بعدی با دستور **WAITMS 50** یک تاخیر **50** میلی ثانیه ای در اجرای برنامه ایجاد کردیم

در خط بعدی هم با دستور **LOOP** انتهای حلقه رو مشخص کردیم که برنامه ما در این حلقه دور می زنه و کارش رو از اول شروع می کنه و به **END** هیچ وقت نمی رسه و برنامه به پایان نمی رسه .

یک نکته ای رو که باید بگم و تو هم سوال نکردی این هستش که همیشه باید دستور **LOCATE** بین **CLS** و

LCD نوشته بشه مثل زیر

CLS
LOCATE 2 , 1
"LCD "AHMAD

خوب اینم بقیه دستورات ال سی دی که تموم شد .

ادامه دستورات که مربوط به دستور پرش و چندتا چیز دیگه هستش باشه برای بعد

پایان قسمت هفتم

قسمت هشتم

1-سلام خوبی

2-سلام خوبم تو چطوری

1-عالی راستی امروز چی می خای بگی

2-امروز باید دستور حلقه و پرش رو که قبلا بهت نگفتم بهت بگم که خیلی راحت هست

1-اماده ام بگو

2-اولین دستور **GOSUB** هستش توسط این دستورشما می تونی بپری توی یک زیر برنامه و دستورات داخل اون زیر برنامه رو انجام بدی دقیقا همان کاری که دستور **GOTO** برای ما انجام می داد این دستور **GOSUB** هم برای ما انجام می ده با یک تفاوت و اون هم اینکه در دستور **GOSUB** ما با دستور **RETURN** می توانیم دوباره برگردیم **GOSUB** رو بخونیم کاری که با دستور **GOTO** نمی تونستیم انجام بدیم

1-یک مثال بزن که نفهمیدم

2-باشه اینم مثال :

```
CONFIG PINB.0 = INPUT
CONFIG PORTA = OUTPUT
DO
IF PINB.0 = 1 THEN GOSUB BENZ
LOOP
END
:BENZ

TOGGLE PORTA
WAITMS 50
RETURN
```

توسط برنامه بالا می تونیم با زدن میکرو سویچ پورت **A** رو **TOGGLE** کنیم. یعنی برای بار اول که میکروسویچ رو میزنیم پورت **A** یک می شه برای بار دوم که میزنی صفر میشه برای بار سوم که می زنی دوباره یک می شه برای بار چهارم که میزنی صفر میشه و الی اخر. در خط اول ما پایه **B.0** رو به عنوان ورودی پیکره بندی کردیم. در خط دوم پورت **A** رو برای خروجی پیکره بندی کردیم. در خط سوم ما اول حلقه رو با نوشتن دستور **DO** عنوان کردیم. در خط چهارم ما نوشتیم

IF PINB.0 = 1 THEN GOSUB BENZ به معنی که اگر پایه **B.0** میکروکنترلر یک شد پیر برو توی زیر برنامه **BENZ**. در خط پنجم با دستور **LOOP** انتهای حلقه رو مشخص کردیم. در خط ششم نوشتیم **END** که برنامه ما هیچ وقت به اینجا نمی رسه. در خط بعدی ما یک زیر برنامه به اسم **BENZ** مشخص کردیم. خط هفتم نوشتیم **TOGGLE PORTA** یعنی وضعیت پورت **A** رو هر چی که هست معکوسش کن یعنی اگه یک هستش بکنش صفر و اگه صفر هستش بکنش یک. در خط هشتم ما یک تاخر **50** میل ثانیه ای گذاشتیم. در خط نهم نوشتیم **RETURN** که در ادامه بهت که توی این برنامه ضیفش چیه.

خوب میکروکنترلر میاد از خط اول برنامه شروع میکنه به خوندن خط اول خط دوم رو می خونه. به خط سوم که می رسه یعنی **DO** می فهمه که یک حلقه سر راهشه و توی این حلقه یکسره دور میزنه و خارج نمی شه تا وقتی که اون شرطی که گذاشتیم اجرا بشه یعنی **IF PINB.0 = 1 THEN GOSUB BENZ**. حالا فرض می کنیم که ما میکروسویچ رو فشار دادیم و با این کارمون **PINB.0** رو یک کردیم در اینجا میکرو میاد دستور شرط رو یعنی **THEN GOSUB BENZ** رو اجرا میکنه یعنی میبره می ره توی زیر برنامه **BENZ** و دستورات داخل این زیر برنامه رو اجرا میکنه. خوب حالا داخل این زیر برنامه ما این دستورات رو نوشتیم **TOGGLE PORTA** یعنی وضعیت پورت **A** رو هر چی که هستش معکوسش کن و در اینجا میکرو میاد پورت **A** رو یک

میکنه و اگه شما به پورت **A LED** وصل کرده باشی می بینی که روشن می شه دستور بعدی که داخل این زیربرنامه هست **WAITMS 50** که میکرو با خوندن این خط به مدت **50** میلی ثانیه در اجرای برنامه تاخیر ایجاد میکنه . دستور بعدی که ما داخل این زیربرنامه نوشتیم **RETURN** هستش که میکرو بعد از گذشت **50** میلی ثانیه که این دستور رو می خونه برمی گرده توی حلقه . و دوباره توی اون حلقه دور می زنه تا وقتی که دوباره شرط اجرا بشه و دوباره بپره بره توی زیربرنامه.

برنامه بالا رو اینطوری هم میشه نوشت که این روش منطقی تره :

-3

```

CONFIG PINB.0 = INPUT
CONFIG PORTA = OUTPUT
DO
DEBOUNCE PINB.0 , 1 , BENZ , SUB
LOOP
END
:BENZ

TOGGLE PORTA
RETURN

```

برنامه بالا رو فقط تاخیر **50** میلی ثانیه ای شو حذف کردم . برنامه بالا رو تنها تغییری که توش انجام دادم خط چهارمش هستش یعنی **DEBOUNCE PINB.0 , 1 , BENZ , SUB** که ما با نوشتن این خط این رو به میکرو فهمونیدم که اگر یک موقع **PINB.0** ما یک شد بپر برو توی زیربرنامه **BENZ** . اون **SUB** هم نوشتیم که به میکرو بفهمونیم که ما دستور **GOSUB** می خایم که این پرش صورت بگیره و اگه ننویسیم میکرو خودش پیش فرض با دستور **GOTO** میپره توی زیربرنامه . خوب حالا فرض کن که این شرط یعنی پایه **B.0** یک شده در اینجا میکرو میپره میره توی زیربرنامه **BENZ** و دستورات داخل این زیربرنامه رو اجرا می کنه و وقتی که به دستور **RETURN** میرسه مفهمه که باید بپره بره توی حلقه و خط

```

DEBOUNCE PINB.0 , 1 , BENZ , SUB

```

رو دوباره بخونه و داخل این حلقه دوباره گیر میکنه تا وقتی که پایه **B.0** دوباره یک بشه و دوباره بپره بره توی زیربرنامه و روز از نو روزی از نو.

دستور بعدی حلقه **FOR –NEXT** هستش که با این دستور می تونی یک حلقه شمارشی ایجاد کنی اگه توجه کنی تا حالا من یک حلقه **DO-LOOP** بهت گفتم که یک حلقه نامحدود بود ولی این حلقه **FOR-NEXT** یک حلقه محدود هستش یعنی میاد توی یک حلقه ایجاد میکنه البته تا یک زمان مشخصی این کار رو میکنه که این زمان دست خودمون هستش که چقدر باشه به مثال زیر توجه کن :

```

DIM A AS BYTE
FOR A = 1 TO 10 STEP 1
CLS
LCD A
WAITMS 30
NEXT A
END

```

خوب بریم سر تحلیل برنامه . خط اول که ما یک متغیر به اسم **A** و از نوع بایت تعریف کردیم . در خط دوم ما نوشتیم که

FOR A = 1 TO 10 STEP 2 به این معنی که متغیر **A** رو افزایش بده از عدد **1** تا عدد **10** . و دستور

STEP 1 که نوشتیم به این معنی هستش که یکی یکی بشمار یا به عبارتی یک پله یک پله بشمار

1-ببین نفهمیدم این STEP 1 رو دوباره توضیح بده

2- یعنی این که یکی یکی بشمار 1 و 2 و 3 و 4 و 5 و 6 تا عدد 10. آگه ما به جای STEP 1 می دادیم STEP 2 دو تا دو تا می‌شمرد اینطوری 1 و 3 و 5 و 7 و 9

خوب بریم سر ادامه وقتی که میکرو خط FOR A = 1 TO 10 STEP 2 رو می خونه شروع می کنه به افزایش متغیر A. بعد میاد خط بعدی که نوشته CLS رو می خونه و در این جا LCD رو پاک میکنه. بعد میاد خط LCD A رو میخونه و می فهمه که باید مقداری که داره افزایش پیدا میکنه رو نشون بده. خط بعدی نوشتیم WAITMS 30 که یک تاخیر 30 میلی ثانیه ای در اجرای برنامه انجام میده و کارش این هستش که بیاد مدت زمان افزایش رو تند یا کند کنه یا به عبارتی اهسته یا سریع کنه. خط بعدی نوشتیم NEXT A که میکرو با خوندن این خط دوباره برمیگرده و خط دوم رو یعنی خط

FOR A = 1 TO 10 STEP 2 رو می خونه. خط بعدی نوشتیم END که برنامه ما فعلا به این قسمت نمی

رسه تا وقتی که شمارش تموم بشه

1- من که چیزی نفهمیدم

2- بزار تحلیلش کنم می فهمی

خوب حالا تحلیل برنامه: میکرو میاد از خط اول شروع میکنه به خوندن و می فهمه که ما یک متغیر به اسم a و از نوع بایت تعریف کردیم. در خط دوم که ما نوشتیم FOR A = 1 TO 10 STEP 1 میکرو میاد شروع میکنه به شمارش یا همون افزایش متغیر یک نکته رو بزار همیجا بگم که فرقی نداره که عددها چقدر باشه حتی می تونیم بگیم که از 10 شروع به شمارش کن تا مثال 110 هیچی محدودیتی وجود نداره البته حداکثر تا 255 میتونی بدی. خوب بری سر دستوری که نوشتیم. خوب بعد میکرو میاد دفعه اول خود مقدار 1 رو میزازه توی متغیر A بعد میاد با فرمان CLS ال سی دی رو پاک میکنه و با دستور LCD A میاد مقدار متغیر A رو که الان هست 1 رو روی ال سی دی نمایش می ده بعد با دستور WAITMS 30 میاد یک تاخیر 30 میلی ثانیه ای در اجرای برنامه ایجاد می کنه و در خط بعد تا که دستور NEXT A رو می خونه برمی گرده توی اون خط FOR A = 1 TO 10 STEP 1 و دوباره که این دستور رو خوند مقدار متغیر A رو یک واحد بیشتر می کنه یعنی 2 و دوباره با دستور CLS ال سی دی رو پاک می کنه و با دستور LCD A میاد مقدار متغیر A رو که الان هست 2 رو روی ال سی دی نمایش می ده. این روال همینطور ادامه داره تا این که مقدار متغیر A بشه 10 یا به عبارتی مساوی بشه با مقداری که در پایان شمارش بهش دادیم (که در اینجا عدد 10 رو دادیم) و وقتی که این اتفاق افتاد میکرو میاد خط زیری NEXT A رو می خونه که ما در اینجا دستور END رو گذاشتیم که برنامه به پایان برسه ولی شما آگه حال کردی که شمارش دوباره آغاز بشه و دوباره بشماره می تونی قبل از دستور FOR یک زیربرنامه تعریف کنی و در بعد دستور NEXT A هم یک دستور پرش به این زیربرنامه رو بزاری که هر وقت که شمارش به پایان رسید میکرو پیره بره توی این زیربرنامه و شمارش رو از دوباره آغاز کنه مثل مثال زیر

DIM A AS BYTE

:MAIN

FOR A = 1 TO 10 STEP 1

CLS

LCD A

WAITMS 30

NEXT A

GOTO MAIN

END

دستور بعدی WHILE – WEND هستش این دستور همون کاری رو می کنه که دستور DO – LOOP انجام

می داد ولی با یک تفاوت اون هم اینکه برای داخل شدن به این حلقه بی نهایت باید شرط مورد نظری که جلوی

WHILE نوشته میشه اول اجرا بشه تا اجازه داخل شدن به این حلقه داده بشه. یک چیز دیگه هم که هست

اینه که **WHILE** ابتدای حلقه هستش مثل همان **DO** که ابتدای حلقه بود و **WEND** هم انتهای حلقه مثل **LOOP** که انتهای حلقه بود به مثال زیر توجه کن که خیلی هم ساده هست.

1- یک سوال فرق بین دستور **WHILE – WEND** با دستور **DO – LOOP** چی هستش

2- در دستور **DO – LOOP** شما برای ورود به حلقه نیاز به اجازه نداشتی ولی توی دستور **WHILE –**

WEND نیاز به کسب اجازه داری حالا این اجازه کی صادر میشه زمانی که شرطی رو که جلوی **WHILE**

گذاشتیم اجرا بشه . اگه بخای به یک زبون دیگه حالت کنم باید بگم که فرق این دوتا دستور مثل ابتدایی و دانشگاه هستش شما توی ابتدایی برای بیرون رفتن از کلاس باید از معلمتون اجازه می گرفتی ولی توی دانشگاه اجازه نمی گیری و سرتو میندازی پایین و می ری بیرون و اگه یک زمانی توی دانشگاه از استاد اجازه بگیری بقیه بهت می خندن توی میکرو هم همینطور ممکنه که متغییرا بهت بخندن.

خوب بری یک مثال ساده هم برات بزنیم که حال کنی :

DIM K AS BYTE

K = 3

WHILE K = 3

CLS

"LCD "OK

WEND

END

خط اول ما یک متغییر به اسم **K** و از نوع بایت تعریف کردیم .

خط دوم به متغییر **K** مقدار دادیم . مقدار **3** دادیم

خط سوم با دستور **WHILE** ابتدای حلقه رو مشخض کردیم البته برای ورود به حلقه این حلقه یک شرط

گذاشتیم که این شرط هر چیزی میتونه باشه که این شرط رو ما باید جلوی دستور **WHILE** با یکم فاصله

بنویسیم . شرطی که نوشتیم **K = 3** هستش یعنی اگر که مقدار **K** عدد **3** بود اجازه ورود به حلقه داده میشه .

خط چهارم که داخل حلقه قرار داره **CLS** هستش که می دونی چی کار می کنه

خط پنجم هم **"LCD "OK** هستش که این دستور هم داخل حلقه قرار داره

خط ششم **WEND** هستش یعنی پایان حلقه

خط هفتم هم **END** هستش که تا زمانی که شرط برقرار باشه برنامه به این دستور نمی رسه

تحلیل این برنامه هم خیل ساده هستش میکرو میاد از خط اول شروع می کنه به خوندن خط اول خط دوم رو می

خونه می رسه به خط سوم این جا رو گوش کن که خیلی مهمه اگه شرط برقرار بود یعنی مقدار **K = 3** بود که

الان هست اجازه ورود به این حلقه داده می شه ولی اگه شرط برقرار نبود میکرو میره و خط بعدی **WEND** رو

می خونه که ما اینجا حال کردیم که اگه شرط برقرار نبود برنامه به **END** برسه و تموم بشه .

دستور بعدی **DELAY** هستش که این دستور یک دستور تاخیر در اجرای برنامه هستش همانند دستور های

WAITMS – WAIT –WAITUS که دستور **DELAY** به مدت **1000** میکروثانیه در اجرای برنامه تاخیر

ایجاد میکنه که نیاز هم به مثال نیست.

فعلا همینا رو داشته باش تا بعد !!!

1- سلام خوبی

2- خوبم تو چطوری

1- ای بد نیستم خوب قراره امروز چی یادم بدی

2- امروز می خوام کار با حافظه **EEPROM** رو بهت بگم

1- چی هستش

2- حافظه بلند مدت یا حافظه پایدار میکروکنترلر . اگه یادت باشه برنامه ای رو که ما می نوشتیم و بعد پروگرام می کردیم و با این کار برنامه رو می ریختیم توی قسمت حافظه . **FLASH** ما در **AVR** می توانیم برنامه ای رو که نوشتیم یا توی حافظه **FLASH** بریزم و یا توی حافظه **EEPROM** بریزیم بستگی داره که دوست داشته باشیم اطلاعات ذخیره بشه یا نه

1- من که نفهمیدم چی شد بالاخره توی کدوم حافظه بریزیم بهتره

2- ببین این دوتا حافظه نسبت به هم یک سری خوبی ها بی دارن و یک سری بدی ها مثلا حافظه **FLASH** نسبت به حافظه **EEPROM** از سرعت بالاتری برخورداره و ما برنامه ای رو که قبلا توی این حافظه ریخته بودیم می تونیم از طریق نرم افزار پاکش کنیم بدی حافظه **FLASH** هم توی همین هستش که اطلاعاتی رو که توی اون ریختیم بعد از این که تغذیه میکروکنترلر قطع بشه از بین می ره

1- منظورت از اطلاعات همون برنامه ای هستش که توش ریختیم

2- نه اشتباه نشه منظورم اینه که مثلا شما داری یک متغییر رو افزایش می دی و روی ال سی دی هم نمایش می دی بعد تغذیه میکروکنترلر رو بر می داری بعد دوباره تغذیه رو وصل می کنی ایا مقدار متغییری که قبلا افزایش داده بودی از صفر می یاد یا آخرین عددی که افزایش پیدا کرده بوده خوب مسلما از صفر چون اطلاعاتش پاک شده. خوب بریم سر بحث خودمون حافظه **EEPROM** هم نسبت به حافظه **FLASH** یک سری مزیت داره اون هم اینه که می تونیه اطلاعات رو در خودش نگه داره یک بدی هم که داره اینه که سرعتش نسبت به حافظه **FLASH** پایین هستش و برنامه ای رو که شما توی این حافظه می ریزی نمی تونی با نرم افزار پاکش کنی و باید با اشعه پاک بشه . بزار یک مثال بزنم . فرض کن که داری توی برنامه **WORD** یک متن رو تایپ می کنی و کلی هم تایپ کردی و ناگهان برق

می ره و بعد از دو ساعت برق میاد می ری کامپیوتر رو روشن می کنی ایا اون متنی رو که تایپ کرده بودی هنوز هستش یا اینکه از بین رفته

1- خوب مسلما چون که ذخیره نکرده بودم از بین رفته

2- آفرین . حالا بیا فرض کنیم که شما بعد از این که این متن رو تایپ کردی رفتی توی یک درایوی ذخیرهش کردی و بعد از این که ذخیرهش کردی یک دفعه برق می ره بعد از دو ساعت برق می ره می ری کامپیوترت رو روشن می کنی بعد می ری سراغ اون متنی که قبلا ذخیره کرده بودی ایا پاک شده

1- نه چون که ذخیرهش کرده بودم

2- احسن حافظه **EEPROM** هم همینظوره و می تونه آخرین اطلاعاتی رو که بهش دادی تا مدت های نا

محدود در خودش نگه داره

1- بابا یک مثال هم از نحوه برنامه نویسیش بزن

2- چشم این هم یک مثال

Dim A As Byte
Readeeprom A , 0
Do
Debounce Pinb.0 , 1 , Show , Sub

Debounce Pinb.1 , 1 , Show2 , Sub

Loop
End

```
show:  
Incr A  
Writeeprom A , 0  
Waitms 5  
Readeeprom A , 0  
Cls  
Lcd A  
Return  
show2:  
A = 0  
Writeeprom A , 0  
WAITMS 5  
Cls  
Lcd A  
Return
```

در برنامه بالا ما به وسیله دو میکروسویچ که به پایه های **b.0** و **b.1** وصل هستند یک متغیر رو زیاد می کنیم و هر وقت که دوست داشتیم صفرش می کنیم میکروسویچی که به پایه **B.0** وصل هستش برای زیاد کردن متغیر هستش و به کمک این میکروسویچ می تونیم از صفر تا **225** رو به کمک میکروسویچ زیاد کنیم میکروسویچی که به پایه **B.1** وصل می شه برای این هستش که ما در هر جایی که از برنامه باشیم بتوانیم متغیر رو صفر کنیم

جالبی این برنامه این هستش که ما به دلیل این که برنامه رو توی حافظه **EEPROM** میریزیم اگه یک وقتی تغذیه میکروکنترلر رو قطع کنیم و مجددا وصل کنیم و دوباره میکروسویچی که برای زیاد کردن متغیر هستش رو تحریک کنیم متغیر از آخرین عددی که قبلا زیاد شده بوده شروع می کنه به زیاد شدن

1- همین تیکه رو یک بار دیگه بگو

2- فرض کن که با میکرو سوییچ متغیر رو زیاد کردی و رسوندی به عدد مثلا **10** و روی ال سی دی نمایش دادی بعد تغذیه میکروکنترلر رو قطع کردی و بعد از چند ساعت میای دوباره تغذیه میکروکنترلر رو وصل می کنی و دوباره میکروسویچ رو می زنی حتما فکر میکنی که از صفر دوباره شروع میکنه به زیاد کردن . باید بگم اگه اینطوری فکر می کنی اشتباه کردی چون که بازدن میکروسویچ به دلیل اینکه این مقدار متغیر در حافظه **EEPROM** ذخیره شده متغیر شروع میکنه به زیاد شدن البته از ادامه کار یعنی ادامه **10** . خوب حالا شاید خواستیم که از صفر شروع کنیم برای این کار میکروسویچی رو که به پایه **B.1** وصل هستش رو فشار می دیم و دوباره مقدار متغیر می شه **0** خوب حالا بریم سر تحلیل برنامه:

میکرو میاد برنامه رو از خط اول شروع می کنه به خواندن خط اول ما یک متغیر از نوع بایت و به اسم **A** تعریف کردیم خط دوم نوشتیم **Readeeprom A , 0** یه این معنی که بخوان مقدار متغیر **A** رو از مکان حافظه **0** **EEPROM** که به جای **0** هر عدد دیگری میتونی بدی . میکرو که این دستور رو میخونه می فهمه که باید مقدار متغیر **A** رو که قبلا ذخیره کرده بخونه خوب برای بار اول که ما چیزی رو ذخیره نکردیم . پس مقدرا متغیر **A** رو در حله اول صفر در نظر می گیره

بعد میاد میوفته توی حلقه **DO - LOOP** و اون تو دور می زنه تا وقتی که یکی از پایه های **B.0** یا **B.1** ما

یک بشه که این پایه ها مستقیما وصل شده به میکروسویچ ها

فرض کن که میکروسویچ اول که وصل شده به پایه **B.0** تحریک بشه در این جا میکروکنترلر میفهمه که پایه **B.0** یک شده و می ره دستور شرط رو که پرش به زیر برنامه **SHOW** هشتش رو اجرا می کنه و میپره میره داخل این زیر برنامه خوب حالا باید دستورات داخل این زیر برنامه رو اجرا کنه

اولین دستور داخل این زیر برنامه هشتش **INCR A** و میکرو هم این کار رو میکنه و یک واحد به مقدار متغییر **A** اضافه می کنه یعنی می شه عدد **1**

دستور دوم هشتش **Writeeprom A , 0** این دستور به این معنی هشتش که مقدار متغییر **A** رو ذخیره کن داخل مکان حافظه **0** و میکرو هم این کار رو میکنه و مقدار عدد **1** رو توی این مکان از حافظه **EEPROM** ذخیره میکنه. دستور بعدی داخل این زیر برنامه نوشته **Waitms 5** یعنی این که یک تاخیر **5** میلی ثانیه ای ایجاد کن این تاخیر رو همیشه بعد از دستور **Writeeprom** بزار چون که یک تاخیر ایجاد بشه و عملیات ذخیره سازی به درستی انجام بشه. دستور بعدی داخل این زیر برنامه نوشته **Readeeprom A , 0** این دستور به این معنی هشتش که بخوان مکان حافظه **0** از حافظه **EEPROM** دقیقه همان مکانی که مقدار متغییر **A** رو اونجا ذخیره کرده بودیم و میکرو هم این مکان رو می خونه و می فهمه که مقدرا متغییر **A** هشتش **1**. دستور بعدی داخل این زیر برنامه هشتش **CLS** که باید بدونی برای چی هشتش. خط بعدی داخل این زیر برنامه نوشته **LCD A**

به این معنی که مقداری رو که توسط دستور **Readeeprom A , 0** خواندی رو روی ال سی دی نمایش بده و میکرو هم این کار رو می کنه توجه داشته باش که هر وقت خواستی مقدار متغییری رو که قبلا ذخیره کردی رو بخونی و روی ال سی دی نمایش بدی اول باید با دستور **Readeeprom**

مقدار ذخیره شده رو بخونی بعد با دستور **LCD** مقدرا رو روی ال سی دی نمایش بدی. دستور بعدی داخل این زیر برنامه هشتش **RETURN** به این معنی که پیر برو سر همون خطی که قبلا ازش پرش کرده بودی یعنی خط **Debounce Pinb.0 , 1 , Show , Sub** دوباره توی این حلقه دور می زنه تا این که دوباره یکی از دو میکروسویچ ها تحریک بشه و بپره بره توی زیر برنامه ای که براشون تعریف شده. فرض کن که همیجا یک دفعه دستت میخوره به سیم منبع تغذیه ای که به میکروکنترلر وصل هشتش و تغذیه میکروکنترلر قطع میشه و تو دوباره تغذیه رو وصل می کنی و میکروکنترلر هم از خط اول برنامه شروع میکنه به خواندن برنامه و دستور **READEEPROM A , 0** رو می خونه و میکروکنترلر با خواندن این دستور می فهمه که باید بره مقدار متغییر **A** رو از مکان **0** حافظه **EEPROM** بخونه یعنی دقیقا همانجایی که مقدار متغییر **A** رو در اونجا ذخیره کرده بود. خوب ما آخرین مقداری رو که با تحریک کردن میکروسویچ به متغییر **A** دادیم عدد **1** بود خوب اینجا میکرو می فهمه که ما آخرین مقداری رو که به متغییر **A** داده بودیم عدد **1** بوده از این به دیگه از صفر شروع نمی کنه به افزایش متغییر بلکه از ادامه عدد **1** شروع میکنه به شمردن. خوب ما یک میکروسویچ دیگه هم داشتیم که وصل بود به پایه **B.1** و این میکرو سویچ برای این بود که مقدار متغییر ما در هر وضعیتی که بود بیاد صفرش کنه به این ترتیب که با تحریک میکروسویچ میکروکنترلر می فهمه که پایه **B.1** یک شده و باید از داخل حلقه **DO - LOOP** بپره بره توی زیر برنامه **SHOW2** و دستورات داخل این زیر برنامه رو اجرا کنه خوب حالا داخل این زیر برنامه چی هشتش الان می گم اولین دستوری که داخل زیر برنامه **SHOW2** هشتش **A = 0** هشتش که میکروکنترلر با خواندن این دستور میاد مقدار متغییر **A** رو صفر می کنه حالا مقدار متغییر **A** هر چقدر که بوده فرقی نداره و صفرش می کنه. دستور بعدی داخل این زیر برنامه هشتش **WAITMS 5** که میکروکنترلر با خواندن این دستور یک تاخیر **5** میلی ثانیه ای در اجرای برنامه ایجاد می کنه که این تاخیر رو همونطور که گفتم باید بعد از دستور **Writeeprom** همیشه بزاری تا اطلاعاتی که می خواد ذخیره بشه به درستی ذخیره بشه. دستور بعدی داخل این زیر برنامه هست **CLS** که می دونی برای چی هشتش. دستور بعدی هشتش **LCD A** و میکرو میاد مقدار متغییر **A** رو که ما الان صفرش کردیم رو روی ال سی دی نمایش می ده یعنی عدد صفر رو نمایش می ده. دستور بعدی داخل این زیر برنامه هشتش **RETURN** که میکرو با خواندن این

دستور میاد میبیره کجا همونجایی که قبلا ازش پریده داخل این زیربرنامه یعنی خط
Debounce Pinb.1 , 1 , Show2 , Sub و دوباره می یوفته داخل این حلقه **DO -LOOP** و روز از نو

روزی از نو

1- یک سوال اگه ما مقدار متغییر رو با زدن میکروسویچ رسوندیم به مثلا عدد **20** بعد توسط میکروسویچ دوم اومدیم مقدار میکروسویچ رو صفر کردیم بعد تغذیه میکروکنترلر رو قطع کردیم و دوباره وصل کردیم حالا سوال اینجاست که آخرین عددی که در حافظه **EEPROM** ذخیره می شه و بعد از برقراری تغذیه مجددا توسط دستور **READEEPROM** خوانده می شه چه عددی هستش ؟

2- خوب این که خیلی راحتته شما بعد از این که مقدار متغییر رو رسوندی به عدد **20** اومدی بازدن میکروسویچ دوم مقدار متغییر رو صفر کردی پس آخرین مقداری که در این متغییر بوده و ذخیره هم شده عدد صفر هستش و با برقراری تغذیه مجدد و خوانده شدن دستور **READEEPROM** توسط میکروکنترلر میکروکنترلر میفهمه که مقدار متغییر هم اکنون هستش صفر . اینو یادت باشه که آخرین مقداری که به متغییر می دی توی حافظه **EEPROM** ذخیره میشه و فرقی نداره که چه عددی باشه .

خوب تا همین جا داشته باش تا جلسه بعدی

قسمت دهم

1-سلام خوبی

2-سلام ای بد نیستیم تو خوبی

1-قربانت . امروز چه مبحثی رو می خای بگی

2-دستور ROTATE رو اول می گم بعد می ریم سر بحث شیرین تایمر کانتر

1-برو که رفتیم

2-دستور ROTATE : توسط این دستور می شه بیت ها رو به سمت چپ یا راست ببری که کاربرد زیادی هم

داره مثلا در کنترلر استپ موتور . بزار یک مثال بهت بگم تا به اهمیت این دستور پی ببری

فرض کن که هشتا **LED** داری و می خای اولی روشن بشه بعد از چند ثانیه خاموش بشه و همزمان با خاموش شدنش دومین **LED** روشن بشه بعد از چند ثانیه خاموش بشه همزمان با خاموش شدنش **LED** سوم روشن بشه و الی اخر تا این که **LED** هشتم روشن بشه و چند ثانیه روشن بمونه و خاموش بشه و از دوباره **LED** اول روشن بشه و این ریتم از دوباره تکرار بشه . خوب اگه ما بخایم این کار رو انجام بدیم باید از برنامه زیر استفاده کنیم در برنامه زیر **LED** ها به پورت **A** متصل شده اند

Config Porta = Output

Do

Set Porta.0

Waitms 25

Reset Porta.0

Set Porta.1

Waitms 25

Reset Porta.1

Set Porta.2

Waitms 25

Reset Porta.2

Set Porta.3

```

Waitms 25
Reset Porta.3
Set Porta.4
Waitms 25
Reset Porta.4
Set Porta.5
Waitms 25
Reset Porta.5
Set Porta.6
Waitms 25
Reset Porta.6
Set Porta.7
Waitms 25
Reset Porta.7
Loop
End

```

برنامه بالا کار می کند ولی از نظر من غلط هستش چون که ما با دستور **ROTATE** می تونیم این کار رو خیلی راحت انجام بدیم و نیاز به نوشتن این همه دستور نیست به مثال زیر توجه کن

```

Config Porta = Output
Dim B As Byte
B = &B1000000
Do
Rotate B , Right
Porta = B
Waitms 100
Loop
End

```

برنامه بالا همون کار برنامه قبلی رو انجام می ده می بینی که چقدر دستورات کم شد البته من فقط یک مثال زدم که باید در مورد ستور هم بهت بگم که چی به چیه. گفتیم که یک بایت تشکیل شده است از هشت بیت. ما در این برنامه یک متغیر به اسم **B** و از نوع بایت تعریف کردیم. این بیت ها رو می تونیم به یک پورت هم ارتباط بدیم.

حالا بزار بریم سر برنامه بهت می گم که چه اتفاقی افتاده .

خط اول ما اومدیم پورت **A** رو به عنوان خروجی تعریف کردیم

خط دوم اومدیم یک متغیر از نوع بایت و به اسم **B** تعریف کردیم

خط سوم با دستور **B = &B1000000** کد باینری برای متغیر **B** انتخاب کردیم اگه توجه کنی متوجه

میشی که این عدد یک عدد هشت رقمی است یعنی هشت بیتی هستش اون یکی که اول نوشته شده همون

منطق یک دیجیتال و اون صفرهایی که نوشته شده همون منطق صفر دیجیتال هستش. این دستور یک نوع

مقدار دهی به متغیر هستش. توضیحات این دستور هنوز ادامه داره که در ادامه توضیح میدم.

خط بعدی با دستور **DO** اومدی و ابتدای حلقه رو مشخص کردیم

خط بعدی با دستور **ROTATE B , RIGHT** به میکرو دستور دادیم که بیت های متغیر **B** رو که در خط قبل

توضیح دادیم یک گام به سمت راست جلوتر ببر یعنی میشه **01000000** اگه توجه کنی متوجه می شی که

عدد یک **1** یک گام جلو تر رفته البته صفرها هم جلوتر می رن ولی هیچ بیتی خارج نمی شه و اگه به اخر برسه

دوباره دور می زنه و میاد اول قرار می گیره.

خط بعد با دستور **PORTA = B** به میکرو فرمان دادیم که مقدار بیت های متغیر **B** رو روی پایه های پورت **A**

بار گذاری کن. حتما از خود می پرسی که یعنی چی. خوب الان من بهت می گم یعنی چی. همونطور که می

دونی هر پورت هشت پایه داره و هر متغیر از نوع بایت هم هشت بیت داره ما اومدیم با این دستوری که نوشتیم

به ترتیب بیت ها رو روی پایه های **A.0** تا **A.7** بار گذاری کردیم در اینجا عدد باینری ما هستش **01000000** پس پایه **A.0** میشه **0** پایه **A.1** میشه **1** پایه **A.2** میشه **0** پایه **A.3** میشه **0** و الی آخر که بقیه پایه ها هم صفر هستش چون ادامه عدد باینری صفر هستش صفر در اینجا به معنی خاموش بودن پایه مورد نظر و یک در اینجا به معنی روشن شدن پایه مورد نظر است . پس تا اینجا پایه **A.1** ما **1** هستش یعنی روشن هستش پس اون **LED**ی که به پایه **A.1** وصل هستش باید روشن بشه و تمام **LED** های دیگه خاموش باشه .

خط بعدی با دستور **WAITMS 100** یک تاخیر صد میلی ثانیه ای در اجرای برنامه ایجاد کردیم که با تغییر این عدد می تونی سرعت روشن روشن و خاموش بودن **LED** ها رو تغییر بدی .

خط بعدی با دستور **LOOP** انتهای حلقه رو مشخص کردیم که میکروکنترلر وقتی به این دستور می رسه بر می گرده و از خط بعدی **DO** شروع می کنه به خوندن مجدد دستورات .

خط بعدی نوشتیم **END** که برنامه هیچ زمان به **END** نمی رسه .

گفتیم که برنامه وقتی به **LOOP** رسید بر می گرده می ره خط بعدی **DO** رو می خونه خط بعدی **DO** هستش **Rotate B , Right** به محض این که این دستور توسط میکرو خونده شد بیت ها دوباره یک واحد به سمت راست هدایت می شوند حالا قبلا بیتها بودن **01000000** که بعد از خونده شدن دستور **Rotate B , Right** یک واحد به سمت راست هدایت می شن یعنی **00100000** اگه توجه کنی می بینی که این دفعه بیت سوم از سمت چپ یعنی **1** جلوتر رفت اونم به سمت راست

خط بعدی **PORTA = B** هستش که قبلا توضیح شو دادم به این معنی هستش که مقدار فعلی بیت ها رو روی پورت **A** بارگذاری کن به محض خونده شدن این دستور میکرو میاد این بیت ها رو به ترتیب از سمت چپ روی پورت **A** بار گذاری می کنه این دفعه بیت سوم از سمت چپ شده **1** و بیت دوم از سمت چپ که در دفعه قبل **1** بوده شده صفر پس پایه **A.1** که قبلا یک بوده یعنی روشن بوده میشه صفر و پایه **A.2** که قبلا صفر بوده حالا میشه یک و **LED** ی که بهش وصل هستش روشن میشه

1- همین جا وایستا که یک سوال دارم چه اتفاقی افتاد که بیتی که قبلا یک بوده شد صفر و بیتی که قبلا صفر بود شده یک

2- به دلیل همون دستور **Rotate B , Right** که نوشتیم بوده . ببین ما با این فرمان به میکرو دستور دادیم که بیت ها رو یک واحد به سمت راست هدایت کن و از اونجایی هم که گفتیم بیت ها خارج نمیشن و دور می زنن هر بیتی که اخر از همه بود و یک واحد به سمت راست هدایت شد بر میگردد میره اول .

خوب بزار ادامه تحلیل رو بگم:

خط بعدی هستش **WAITMS 100** که قبلا دربارش توضیح کافی داده شد

خط بعدی که توسط میکروکنترلر خونده میشه **LOOP** هستش و با خونده شدن این دستور میکرو میبره دوباره می ره از خط زیری **DO** شروع می کنه به خوندن یعنی خط **Rotate B , Right** و با خونده شدن این دستور دوباره تمام بیت ها یک واحد به سمت راست هدایت میشن یعنی **00010000** و الی آخر و هی این برنامه دور مزنه و **LED** ها یکی پس از دیگری روشن و خاموش می شه

1- سوال دارم بعد از این که این **1** به اخر رسید یعنی شد **00000001** و دوباره میکرو به دستور **Rotate B , Right** رسید تکلیف این **1** چی می شه چون که باید یک واحد به سمت راست هدایت بشه

2- این که خیلی راحت خوب **1** دوباره بر می گرده سر خط یعنی **10000000**

یکم این مثال رو دستکاری کن تا همه چیز دستت بیاد مثلا در اونجایی که ما نوشتیم **B = &B10000000** می تونی دستکاری کنی و یک مقدار دیگه بدی مثلا **B = &B 10101010** یا هر چیز دیگه .

خوب رسیدیم به مبحث تایمر کانتر در میکرو کنترلر : **AVR**

در میکرو کنترلر **AVR** نهایتا دو سه تا تایمر و کانتر بیشتر نداریم البته در میکروکنترلر نوع **ATMEGA8** چهار تایمر کانتر داریم که در این نوع استسنا هستش

1- تایمر رو فهمیدم ولی کانتر چی هست

2-کانتر (COUNTER) به معنی شمارنده هستش که در میکروکنترلر هم وظیفه این کانتر شمردن پالس هایی هستش که به ورودیش اعمال میشه که سر فرصت بهم می گم . من اینجا فقط تایمر صفر **0** و تایمر یک **1** رو بهت می گم تایمر **3** هم به کار نیومده که برم دنبالش به کار توهم نیما. این تایمر کانتر صفر و یک دو تا فرق باهم دارن و اون اینه که تایمر کانتر صفر **8** بیتی هسش و تایمر کانتر یک **16** بیتی .

1-موارد کاربردش کجا هستش

2-موارد کاربردش ساخت ساعت . شمارنده . فرکانس متر و...

اول میرم سر تایمر کانتر صفر .بین یزار یک چیز رو اول بهت بگم که بعضی از دستورات رو نمی شه موشکافی کنیم پس زیاد گیر نده .

1-باشه

2-اول تایمر صفر : برای این که شما بتونی از تایمر کانتر برای زمان های دقیق استفاده کنی مجبوری بری و

یک **RTC** از بازار بخری و وصل کنی به دو تا از پایه های مخصوص میکروکنترلر که بهش بعدا اشاره می کنم . برای ایجاد یک تایم مثلا یک ثانیه ای شما باید ابتدا به میکروکنترلر با دستور پیکره بندی **CONFIG** حالی کنی که اقای میکروکنترلر من می خام از تایمر صفر شما استفاده کنم . برای ایجای زمان یک ثانیه توسط تایمر صفر به ترتیب زیر باید عمل کنی

اول باید ببینی فرکانس کاری میکروکنترلرت چند هستش که اکثر میکروکنترلرها **8** مگا هستن .

Config Timer0 = Timer , Prescale = 256

Dim A As Byte

Dim B As Byte

Enable Interrupts

Enable Timer0

On Timer0 Show

Do

Loop

End

Show:

Timer0 = 6

Incr A

If A = 125 Then

Incr B

Cls

Lcd B

Timer0 = 0

A = 0

End If

Return

این برنامه باید برای ما یک تایم یک ثانیه ایجاد می کنه ولی در عمل این طور نیست و زمان ها بیشتر از یک ثانیه طول می کشه که این مشکل بایک **RTC** قابل حل هستش .

در خط اول برنامه ما با دستور **Config Timer0 = Timer , Prescale = 256** اومدیم و تایمر صفر رو برای مد تایمر پیکره بندی کردیم البته با این دستور **CONFIG TIMER0 = TIMER** در ادامه همین

خط بعد از گذاشتن علامت . با دستور **Prescale = 256** اومدیم و مقدار فرکانس کاری میکرو رو بر **256**

تقسیم کردیم توجه کن که به جای این عدد از اعداد **1** و **8** و **64** و **1024** هم می تونی استفاده کنی که

گذاشتن یکی از این اعداد بعد **PRESCALE** اجباری هست با این کار ما فرکانس کاری میکروکنترلر رو بر

256 تقسیم کردیم یعنی $8000000 / 256 = 31250$ خوب حتما از خودت می پرسی که ما چرا این

تقسیم رو انجام دادیم .باید بهت بگم که برای این که فرکانس یکم کوچیکتر بشه و بتونیم ازش برای ایجاد یک

ثانیه استفاده کنیم.

خط دوم اومدیم یک متغییر از نوع بایت به نام **A** تعریف کردیم

خط سوم هم اومدیم یک متغییر از نوع بایت به نام **B** تعریف کردیم که در ادامه می گم که این متغییر ها رو برای چه کاری هستن.

خط بعدی با دستور **ENABLE INTERRUPTS** اومدیم وقفه میکرو کنترلر رو فعال کردیم توجه کن که ما در میکرو کنترلر یک وقفه داریم که انواع مختلف هم داره که بعدا در بارش توضیح می دم ولی همین رو بدون که هر وقت از تایمر و کانتر استفاده می کنی باید این دستور رو در اوایل برنامه بزاری تا هر وقت که وقفه تایمر خورد به میکرو اعلام بشه که زمان یک ثانیه شد. در ضمن با دستور

DISABLE INTERRUPTS هم می تونیم این فرمان رو غیر فعال کنیم

خط بعد با دستور **ENABLE TIMERO** تایمر **0** رو فعال کردیم. با دستور **DISABLE TIMERO** هم می تونیم این تایمر **0** رو غیر فعال کنیم.

خط بعدی نوشتیم **ON TIMERO SHOW** این دستور به این معنی هستش که اگر تایمر صفر سرریز شد پیر برو توی زیر برنامه **SHOW** توجه داشته باش که این سرریز شدن تایمر به منزله زمان یک ثانیه نیست که در ادامه می گم زمان یک ثانیه کی به وجود میاد.

خط بعدی نوشتیم **DO** و پایین **DO** هم نوشتیم **LOOP**. این حلقه **DO-LOOP** رو که تو حلقه هم خالی گذاشتیم برای این هستش که اگه برنامه دیگه ای هم داشتیم بزاریم داخل این حلقه مثلا شاید ما یک برنامه ای نوشته باشیم که علاوه بر تایمر یک کیبرد هم در برنامه استفاده کرده باشیم که برنامه مثلا کیبرد هم داخل این حلقه باید نوشته بشه.

در خط بعدی نوشتیم **END** که برنامه ما هیچ وقت به این **END** نمی رسه

خط بعدی نوشتیم **SHOW:** که زیر برنامه ما رو مشخص کرده که داخل این زیر برنامه عبارات زیر نوشته شده:

اولی دستور داخل این زیر برنامه هستش **TIMERO = 6** بین ما گفتیم که تایمر صفر یک تایمر **8** بیتی هستش یعنی یک بایتی خوب یک بایت هم با **255** پر میشه و با **256** تا هم سرریز میشه و دوباره صفر میشه ما اومدیم برای این که این **256** سرراست بشه تایمر صفر رو با **6** تا پر کردیم که تقسیمات هم اعشاری نشه. قبل از این که بخام خط بعدی یعنی **INCR A** رو توضیح بدم بزار یک چیزی رو اول بگم بعد برم سراغش چون که به هم ربط دارن. ما یک تقسیم دیگه هم باید انجام بدیم یعنی $31250 / 250 = 125$ که حاصل شد **125** این همون عدد تایمر هستش که در دستور قبل با فرمان **TIMERO = 6** شش واحد بهش اضافه کرده بودیم که تقسیم ما سرراست بشه نه این که اعشاری بشه. حالا ما این تقسیم رو برای این انجام دادیم که بفهمیم یک متغییر رو چند بار باید افزایش بده تا تایم یک ثانیه ایجاد بشه که در اینجا عدد **125** به دست اومد یعنی برای این که زمان یک ثانیه ایجاد بشه متغییر (در این جا متغییر **A**) باید **125** تا پر بشه. حالا بریم سر خط دستور بعدی.

خط بعدی نوشتیم **INCR A** به این معنی که هر وقت تایمر ما وقفه خورد بیاد و یک واحد به این متغییر **A** اضافه کنه

خط بعدی نوشتیم **IF A = 125 THEN** ما گفتیم که برای به دست اومدن زمان یک ثانیه طبق تقسیماتی که انجام دادیم متغییر **A** باید **125** تا پر بشه برای همین دستور شرط رو نوشتیم که اگر متغییر **A** مساوی **125** شد برو دستور شرط رو اجرا کن این دستور شرط هر چیزی می تونه باشه مثال برو یک پورت رو یک کن یا این که یک واحد به یک متغییر دیگه اضافه کن. یا یک پایه رو صفر کن یا هر چی که دوست داشتی فقط ما این جا اومدیم و دستورات شرط رو در خط های پایین نوشتیم که دستورات شرطی که نوشتیم به این ها هستن:

اولین دستور شرط این هستش **INCR B** یعنی یک واحد به متغییر **B** اضافه کن که هر دستور دیگه ای هم می تونستیم بزاریم مثلا این که یک پایه رو **TOGGLE** کن که ما حال کردیم بعد از یک ثانیه یک متغییر دیگه رو

بره اضافه کنه.

دستور شرط بعدی **CLS** هستش یعنی ال سی دی رو پاک کن
دستور شرط بعدی **LCD B** هستش که ما به میکرو دستور دادیم که مقدار متغییر **B** رو روی ال سی دی
نمایش بده.

دستور شرط بعدی **TIMERO = 0** قرار دادیم تا برای شروع مجدد صفر بشه
دستور شرط بعدی **A = 0** هستش که با این کار مقدار متغییر **A** رو صفر کردیم تا برای بار بعدی که می خاد
متغییر رو اضافه کنه دوباره از صفر شروع به اضافه کردن کنه نه این که از ادامه **125**
خط بعدی **END IF** هستش که ما هر وقت دستورات شرط تمام میشه باید از این دستور استفاده کنیم تا میکرو
بفهمه که دستورات شرط تموم شده . پس از این که میکرو به این خط رسید و این دستور رو خونده هیچ کاری
انجام نمی ده و فقط از خط پایین این دستور شروع به خوندن ادامه برنامه می کنه
خط اخر نوشتیم **RETURN** ببین ما برای این که دوباره یک تایم یک ثانیه دیگه به وجود بیاد مجبوریم که
دوباره برگردیم به اول برنامه در تایمر ها با دستور **RETURN** این کار رو انجام می دیم . و میکرو میره به اول
برنامه از دوباره روز از نو روزی از نو و این ایجاد زمان یک ثانیه ادامه داره .
1- یک سوال اگه ما خاستیم که یک زمان بیشتر از یک ثانیه ایجاد کنیم مثلا دو ثانیه یا بیشتر باید چکار کنیم

2- سوال خوبی کردیم این کار خیلی راحتیه در خطی که نوشتیم **IF A = 125** و این **125** برای یک ثانیه
بود رو دو برابر کنیم تا زمان دو ثانیه حاصل شه یعنی بزاریم **250** . یک چیز دیگه هم باید بهت بگم که اگه
زمان بیشتری مثلا از دو ثانیه بیشتر خاستی باید **125** رو ضرت در اون تایمی که می خای بکنی
مثلا زمان سه ثانیه می خای **125 * 3 = 375** که حاصل شد **375** که این جا چون متغییر **A** فقط تا **255** تا
می تونه پر بشه و با **256** تا سرریز میشه ما نمی تونیم از این نوع متغییر استفاده کنیم و مجبوریم از یک متغییر
دیگه استفاده کنیم که از نوع **WORD** هستش که ظرفیت خیلی بالا داره . حالا کافیه در اوایل برنامه به
جای دستور **DIM A AS BYTE** بنویسیم **DIM A AS WORD** و به جای دستور
IF A = 125 بنویسیم **IF A = 375** البته برای تایم سه ثانیه)
خوب برای امروز دیگه کافیه بقیه باشه برای جلسه بعدی



سلام دوستان اینم قسمت یازدهم

- 1- سلام خوبی
- 2- سلام ممنون
- 1- خوب امروز می خای چه مبحثی رو شروع کنی
- 2- امروز می خام در مورد کانتر یا همون شمارنده بگم
- 1- حالا این کانتر چکار می کنه؟
- 2- کانتر پالس ها رو میشماره مثلا شما می خای یک پاسی رو که از یک جا تولید شده رو بشماری از کانتر
استفاده می کنی کانتر یعنی شمارنده .
- 1- کاربردش چیه؟
- 2- اولین کاربردی که به ذهن من میرسه فرکانس متر هستش . به این صورت که ما مقدار پالس هایی که در مدت

زمان یک ثانیه تولید میشه رو اندازه گیری می کنیم بعد روی **LCD** نمایش می دیم
-1 پس می تونیم مثلا از این کانتر در خط تولید یک کارخانه هم استفاده کنیم که تولیدات یک خط تولید رو بشماره

-2اره میشه ولی وقتی که با دستور **INCR** می تونی این کار رو بکنی چرا با کانتر این کار رو انجام بدی
-1راست میگی یا . گفتی که یکی از کاربرداش فرکانس متر هستش میشه یکم بیشتر توضیح بدی
-2تعریف فرکانس تعداد پالس هایی که در مدت زمان یک ثانیه تولید میشه هستش . یعنی اگه ما بتونیم تعداد پالس هایی رو که در مدت زمان یک ثانیه تولید میشه رو اندازه گیری کنیم میشه گفت که فرکانس متر درست کردیم . به طور کلی برای این کار میایم کانتر و تایمر رو همزمان با هم راه اندازی می کنیم تایمر رو روی یک ثانیه می زاریم که هر یک ثانیه بره و مقدار کانتر رو بخونه و روی **LCD** نمایش بده به این صورت که ما پالس ها رو به پایه کانتر می دیم این پالس ها در مدت زمان یک ثانیه توسط کانتر شمرده می شه . بعد از یک ثانیه تایمر و کانتر **STOP** میشن و مقدار کانتر در مدت زمان یک ثانیه روی ال سی دی نمایش می دیم .
من اینجا نحوه کار با کانتر یک رو برات می گم :

پالس هایی رو که ما می خاهیم توسط کانتر یک بشماریم رو باید به پایه **T1** بدیم .
یک نکته مهم این که کانتر یک با **65535** تا پر میشه و با **65536** تا سرریز میشه یا به عبارتی صفر میشه . به زبان ساده تر اگه ما به پایه **T1** که مربوط به کانتر یک هستش **65535** تا پالس بدیم کانتر پر میشه و اگه **65536** تا پالس بدیم کانتر سرریز میشه و دوباره از صفر شروع میشه .
-2خوب حالا بریم سر دستورات و پیکره بندی کانتر : (**counter**) ما در اینجا از تایمر یک به عنوان کانتر استفاده کرده ایم

Config Timer1 = Counter , Edge = Rising

Config Timer1 = Counter : ما در این جا به میکرو دستور دادیم که می خواهیم از تایمر یک به

عنوان کانتر استفاده کنیم

Edge = Rising : ما در این جا به میکرو دستور دادیم که می خواهیم لبه بالا رونده رو بشماریم . اگه

خواسته باشیم که لبه پایین رونده رو بشماریم به جای **RISING** می نویسم **FALLING**

ENABLE INTERRUPTS : این دستور رو هر بار که از تایمر کانتر میکرو استفاده می کنی باید بنویسی با

این کار وقفه ها رو فعال می کنیم

ENABLE COUNTER1 : با این دستور کانتر یک رو فعال کردیم که همیشه باید بنویسیم .

ON COUNTER1 SHOW : با این دستور به میکرو دستور می دیم که هر وقت که وقفه کانتر **1** خورد برو توی زیر برنامه **SHOW** یا به عبارت دیگه با این دستور به میکرو دستور می دیم که هر وقت کانتر سرریز شد (با همان مقدار **65536** تا) برو توی زیر برنامه **SHOW**

COUNTER1 = X : با این دستور می تونیم به کانتر یک مقدار بدیم و یعنی به جای **X** عدد بزاریم مثلا

COUNTER = 65530 با این کار ما کانتر یک رو با مقدار **65530** تا پر کردیم و کانتر از این مقدار شروع

به شمردن می کند یعنی اگه یک پالس به پایه **T1** بدم مقدار کانتر میشه **65531** اگه **6** تا پالس به پایه **T1** بدیم کانتر سرریز می کنه و صفر میشه .

STOP TIMER1 : با این دستور می تونیم عملیات شمردن کانتر رو متوقف کنیم . توجه کن که اگه با این

دستور عملیات شمردن کانتر رو متوقف کنیم مقدار کانتر صفر نمی شه بلکه روی همون مقداری که تا الان

شمرده استپ می شه .

START TIMER1 : این دستور عکس **STOP TIMER1** هستش یعنی هر جا که ما عملیات شمارش رو

توسط دستور **STOP TIMER1** متوقف کرده باشیم با دستور **START TIMER1** می تونیم شمردن رو

آغاز کنیم .

بزار یک مثال بزنم که بهتر بفهمی:
برنامه زیر به ازای هر پالسی که به پایه **T1** میکروکنترلر می خوره یک متغییر رو یک واحد اضافه می کنه

```
Config Timer1 = Counter , Edge = Rising
Enable Interrupts
Enable Counter1
On Counter1 Show
Counter1 = 65535
Dim A As WORD
Do
Loop
End
SHOW:
Counter1 = 65535
Incr A
Cls : Lcd A
Return
```

حالا تحلیل برنامه :

در خط اول ما تایمر یک رو در مد کانتر پیکره بندی کردیم یعنی به میکرو گفتیم که از تایمر یک می خواهیم به عنوان کانتر (شمارنده) استفاده کنیم و در ادامه همین خط نوشتیم **Edge = Rising** که با این دستور به میکرو فهموندیم که لبه بالا رونده رو می خایم بشماریم اگه ما به جای **RISING** می نوشتیم **FALLING** میکرو فقط لبه پایین رونده پالس رو می شمرد.

خط دوم : **Enable Interrupts** که با این دستور وقفه سراسری رو فعال می کنیم این دستور رو هر وقت که از کانتر یا تایمر استفاده می کنی باید بزاری.

خط سوم : **Enable Counter1** با این دستور کانتریک رو فعال می کنیم این دستور رو هم هر وقت که از کانتر استفاده می کنی باید بنویسی.

خط چهارم : **On Counter1 Show2** با این دستور به میکرو دستور می دیم که هر وقت کانتر سرریز شد (با همان مقدار **65536** تا) بپر برو توی زیر برنامه **SHOW (SHOW)** اسم زیر برنامه است که هر اسم دلخواه دیگر می تواند باشد مثلاً. (**AHMAD**)

خط پنجم : **Counter1 = 65535** به کانتریک مقدار **65535** دادیم یعنی کانتر از ادامه این عدد شروع به شمردن می کند به جای این عدد هر عدد دخواه دیگر هم می توانیم بدهیم.

خط ششم : **Dim A As WORD** ما با این دستور یک متغییر به اسم **A** و از نوع **WORD** تعریف کردیم که در ادامه می گم که چه کاری رو انجام میدهد.

خط هفتم : **DO** با نوشتن این دستور ابتدای حلقه رو مشخص کردیم.

خط هشتم : **LOOP** با این دستور انتهای حلقه رو مشخص کردیم.

خط نهم : **END** این دستور یعنی انتهای برنامه که برنامه هیچ وقت به این دستور نمی رسه.

خط دهم : **SHOW** در اینجا یک زیر برنامه به اسم **SHOW** تعریف کردیم که داخل این زیر برنامه دستورات زیر نوشته شده:

خط یازدهم : **Counter1 = 65535** اولین دستور داخل زیر برنامه **SHOW** است که در این جا به کانتریک مقدار **65535** دادیم یعنی کانتر از ادامه این عدد شروع به شمردن می کند به جای این عدد هر عدد دخواه دیگر هم می توانیم بدهیم.

خط دوازدهم : **INCR A** دومین دستور داخل زیر برنامه **SHOW** است که وقتی میکرو این خط رو می خونه

مقدار متغییر **A** رو یک واحد اضافه میکنه.
خط سیزدهم **Cls : Lcd A** سومین دستور داخل زیربرنامه **SHOW** است که با این دستور میکرو مقدار متغییر **A** رو روی **LCD** نمایش می ده.
خط چهاردهم **RETURN** چهارمین دستور داخل زیربرنامه **SHOW** است که با این دستور میکرو بر می گرده اول برنامه.

تحلیل دوم:

برنامه از ابتدا توسط میکرو خوانده میشه:
خط اول ما اومدیم تایمر یک رو در مد کانتر استفاده کردیم
خط دوم **Enable Interrupts** ما در این جا وقفه سراسری رو فعال کردیم توجه کن که هر وقت از تایمر یا کانتر استفاده می کنی این دستور رو حتما بنویسی تا وقفه سراسری فعال بشه.
خط سوم **Enable Counter1** با این دستور میکرو کانتر یک رو فعال می کنه و میره سر خط بعدی
خط چهارم **On Counter1 Show** میکرو این دستور رو می خونه ولی فعلا تا زمانی این دستور رو اجرا نمی کنه یعنی به زیر برنامه **SHOW** پرش نمی کنه تا زمانی که وقتش برسه
-1 وقتش چه وقت می رسه
-2 هر وقت که کانتر یک با مقدار **65536** تا سرریز کنه بدون درنگ این دستور توسط میکرو خوانده و میپره توی زیر برنامه **SHOW**
خط پنجم **Counter1 = 65535** با این دستور میکرو میکرو کانتر یک رو تا مقدار **65535** تا پر می کنه. این دستور مثل این هستش که ما **65535** تا پالس به پایه **T1** میکرو داده باشیم و کانتر هم به اندازه همین مقدار پر شده باشه.

خط ششم **DIM A AS WORD** میکرو وقتی که به این خط از برنامه می رسه می فهمه که ما یک متغییر از نوع **WORD** به اسم **A** انتخاب کرده ایم و میره خط بعدی رو می خونه.
توی خط هفتم و هشتم ما یک حلقه **DO - LOOP** گذاشتیم میکرو به محض این که این دستور رو میخونه میوفته توی این حلقه و از این حلقه بیرون نمیاره. خوب حالا رسیدیم به قسمت جذاب ماجرا. توی این گیرورداری که میکروی بیچاره افتاده توی حلقه ما تنها کمکی که به این میکروی بیچاره میتونیم بکنیم اینه که یک پالس ناقابل بدیم به پایه **T1** میکرو.

-1 خوب چه اتفاقی میوفته
-2 اها یادت میاد که ما مقدار کانتر رو تا **65535** تا پر کردیم و بهتم گفتیم که مقدار کانتر اگه به **65536** برسه کانتر سرریز میشه و از اونجایی هم که ما توسط دستور **On Counter1 Show** به میکرو دستور دادیم که اگه سرریز کردی ببر برو و توی زیر برنامه **SHOW**

-1 خوب اره چه ربطی داره ؟
-2 ربطش به اینه که ما با دادن یک پالس به پایه **T1** یک واحد به مقدار کانتر یک اضافه کردیم یعنی **65535**
1 + 65536 فهمیدی حالا
-1 اها حالا فهمیدم

-2 پس بزار بقیه ماجرا رو بگم. بعد از این که ما با دادن یک پالس به پایه **T1** مقدار کانتر رو به **65536** رساندیم طبیعتا کانتر سرریز میشه و چون قبلا بهش دستور داده بودیم که اگه کانتر سرریز شد ببر توی زیر برنامه **SHOW** میکرو هم بدون استخاره میپره میره توی زیر برنامه **SHOW** و دستوراتی رو که ما داخل زیربرنامه **SHOW** نوشتیم رو اجرا میکنه. حالا فرض رو بر این میزاریم که ما یک پالس به پایه **T1** دادیم و میکرو سرریز شده و پریده توی زیربرنام **SHOW** و بحث رو از اونجا دوباره پی میگیریم.
بعد از این که میکرو پرید توی زیربرنامه **SHOW** میاد دستورات داخل این زیربرنامه رو اجرا میکنه به ترتیب

زیر:

اولین دستور داخل زیربرنامه **SHOW** هست : **Counter1 = 65535** چون که کانتر یک قبلا سرریز شده و مقدارش صفر شده ما باید دوباره بیایم مقدار کانتر یک رو برگردونیم به حالت قبل یعنی **65535**.
دومین دستور داخل زیربرنامه **SHOW** هست **INCR A** یعنی یک واحد به مقدار متغیر **A** اضافه کن.
سومین دستور داخل زیر برنامه **SHOW** هست **CLS:LCD A** به این معنی که مقدار متغیر **A** رو روی **LCD** نمایش بده

چهارمین یا به عبارت دیگر آخرین دستور داخل زیربرنامه **SHOW** هست **REUTURN** که با این دستور میکرو دوباره میره توی زیر برنامه **SHOW** و دوباره منتظر یک پالس دیگه میمونه تا دوباره سرریز بشه و دوباره بپره بره توی زیربرنامه **SHOW** و دستورات زیربرنامه **SHOW** رو دوباره اجرا کنه.

1- یک سوال اگه ما خواستیم که کانتر به جای این که به ازای هر یک پالس بپره توی زیربرنامه **SHOW** بیاد و به ازای هر دو تا پالس این عمل رو انجام بده باید چه کار کنیم ؟

2- برای این کار ما باید یک کاری انجام بدیم که مقدار کانتر به ازای هر دو تا پالس سرریز بشه یعنی بعد از این که دو تا پالس به پایه **T1** خورد سرریز بشه . پس باید به جای **Counter1 = 65535** بنویسیم **Counter1** **65534 =** یعنی ما مقدار کانتر رو با **65534** تا پر کردیم و برای سرریز شدن کانتر تا **65536** به دو تا پالس نیاز داریم . اینطوری پالس اولی که به پایه **T1** بخوره مقدار کانتر یک واحد افزایش پیدا می کنه و از **65534** میرسه به **65535** و هیچ اتفاق خاصی نمیوفته ولی با دادن پالس دومی مقدار کانتر یک واحد دیگه افزایش پیدا می کنه یعنی از **65535** میرسه به **65536** و اینجاست که کانتر سر ریز میکنه و میپره توی زیربرنامه

SHOW

1- یک سوال دیگه از کجا بفهمیم که پایه **T1** میکرو کجاست ؟

2- از روی دیتاشیت یا برگه مشخصات میکرو یک کار دیگه هم می تونی انجام بدی و اون این که بری توی سایت **www.atmel.com** و اونجا می تونی برگه مشخصات اون نوع میکروبی که باهش کار می کنی رو دانلود کنی.

1- بازم سوال چرا توی از دستور **Counter1 = 65535** دوبار توی برنامه استفاده کردی

2- ببین وقتی که ما تغذیه میکرو رو وصل می کنیم میکروکنترلر یک بار برنامه رو از اول شروع میکنه به خوندن و برای اولین بار دستور **Counter1 = 65535** رو که ما در خط پنجم نوشتیم رو می خونه و با این دستور کانتر رو با مقدار **65535** تا پر می کنه و بعدش می یوفته توی حلقه **DO-LOOP** بعدش اونقدر توی این حلقه می مونه تا ما یک پالس به پایه **T1** بدیم تا این که سرریز بشه و بره توی زیر برنامه **SHOW** حالا بیا فرض رو بر این بزاریم که ما دستور **Counter1 = 65535** رو در خط یازدهم نوشته باشیم . خوب اگه ما این دستور رو در خط یازدهم ننویسیم میکرو میاد و تمام دستورات داخل زیربرنامه رو انجام می ده و با دستور **RETURN** که ما در اخر برنامه نوشته بودیم برمی گرده میره تو حلقه **DO-LOOP** و از اونجایی که کانتر یک بار سرریز شده یعنی مقدارش صفر شده دیگه با پالس بعدی نمیره توی زیر برنامه چون الان مقدار کانتر صفر هستش و ما باید **65536** تا پالس به پایه **T1** بدیم تا بره توی زیر برنامه . حالا ما برای جلوگیری از این کار اومدیم دوبار دستور **Counter1 = 65535** رو نوشتیم تا هر بار که دستورات زیر برنامه رو انجام داد مقدار کانتر رو برگردونه به **65535** و دوباره آماده یک پالس بعدی باشه.

پایان 

1- سلام خوبی

2- سلام ممنون تو چطوری

1- این بد نیستیم . خوب برو سر اصل مطلب امروز می خای چیرو بهم باد بدی

2- امروز می خام برات از INTO یا وقفه خارجی میکرو برات بگم

1- حالا چی هست کارش چیه ؟

2- کارش فقط و فقط اشکار کردن هستش . یعنی شما در هر جای برنامه که باشی با تحریک کردن این پایه

INTO می تونی به یک زیر برنامه بپری و ...

کاربردش کجا هستش کاربردش توی جا که بخای . به عنوان مثال شما یک برنامه برای یک دستگاه پرس پیشرفته نوشتی که قرار میکروکنترلر روی این دستگاه نصب بشه که توی این برنامه هم کیبرد هستش هم اندازه گیری دما هم تایمر و چندتا کار دیگه . یکی از کارهایی که این دستگاه پرس باید انجام بده اینه که وقتی که پرس داره به سمت پایین میاد یک سنسور فعال میشه و با فعال شدن یا سنسور یک تایمر شروع به شمارش کنه و بعد از مثلا پنج ثانیه پرس برمیگرده به حالت اول . با این اوصاف ما باید برنامه رو طوری طراحی کنیم که برنامه ما در هر وضعیتی که بود بره و تایمر رو فعال کنه . حالا تو اگه باشی برای این قسمت از برنامه چه کار میکنی

1- میام و سنسور رو وصل می کنم به یکی از پایه های میکرو بعد برنامه رو به از همون دستوری بود که ورودی

برای میکرو تعریف میکردیم یعنی این دسترو:

DEBOUNCE PINB.0 , 1 , STAR

بعد میکرو میبره میره توی زیر برنامه STAR و کاری رو که گفتم یعنی تایمر رو فعال میکنه

2- بخش سخت افزاری رو خوب اومدی ولی این دستور DEBOUNCE که گفتم کاملا توی این قسمت

اشتباه بود می دونی چرا

1- نه چرا؟؟؟؟

2- مگه ما نگفتیم که این میکروکنترلر قراره بهش کیبرد ماتریسی - سنسور دما و چنتا چیز دیگه وصل بشه .

خوب با این اوصاف میکرو یکسره در حال خوندن این کیبرد و اندازه گیری دما هستش کی وقت میکنه بیاد

دستور DEBOUNCE رو بخونه البته نه اینکه دستور DEBOUNCE رو نخونه منظورم اینه که ممکنه که

موقعی که سنسور فعال شد میکرو در حال خوندن یک قسمت دیگه از برنامه باشه . پس ما باید یک ترتیبی بدیم

که برنامه میکرو در هر قسمتی از برنامه که باشه بره اون کاری رو که ما ازش خاستیم رو انجام بده و این کار رو با

استفاده از INTO انجام میده . حالا بزار چند تا مثال میزنم می فهمی که چی به چی

CONFIG INTO = X

گفتیم که برای استفاده از بعض امکانات میکرو نیاز به پیکره بندی ان داریم که ما با نوشتن دستور بالا این کار رو

انجام دادیم . و به میکرو فهماندیم که می خاهیم از امکان INTO ان استفاده کنیم بعد از علامت مساوی نوشتیم

X که به جای X باید یکی از گزینه های زیر رو بنویسی:

RISING : برای اشکار کردن لبه بالا رونده از RISING استفاده می کنیم .

FALLING : برای اشکار کردن لبه پایین رونده از FALLING استفاده میکنیم .

LOW LEVEL : برای اشکار کردن سطح صفر از LOW LEVEL استفاده می کنیم .

1- والا ما که نمیدونیم این لبه بالا رونده - لبه پایین رونده و سطح صفر چی هستش؟؟

2- الان میگم برات میگم . به این شکل موج توجه کن (برای دیدن شکل روی لینک زیر کلیک کنید)

<http://www.4shared.com/file/21102873/ca60357/RISING.html>

این یک پالس می تونه توسط یک شسی یا یک سنسور یا هر چیز دیگه ایجاد شده باشه . حالا اون قسمتی از شکل موج رو که قرمز هستش رو میگن سطح صفر یا LOW LEVEL اون قسمت ابی رنگ رو بهش میگن لبه

بالا رونده یا **RISING** اون قسمت سیاه رنگ رو بهش میگویند سطح یک یا **HIGH LEVEL** و اون قسمتی که سبز رنگ هستش رو هم میگویند پایین رونده یا **FALLING**

1- فهمیدم ادامه بده . راستی کاربردهای دیگه ای هم داره

2- اره یکی از این کاربرد ها در دیمر دیجیتال هستش که برای اینکه ما بفهمیم شکل موج **AC** از صفر عبور کرده یا نه از **INT** استفاده می کنیم .

1- این پایه **int0** کدوم یکی از پایه های میکرو هستش ؟

2- در هر میکرو فرق میکنه کافیه به دیتاشیت یا همون برگه مشخصاتش مراجعه کنی یک مثال ساده برات میزنم بعد میریم سر مثال های دیگه .

در برنامه زیر به ازای هر بار که شسی رو فشار بدی خروجی **TOGGLE** میشه :

```

Config int0 = rising
Config portb = output
Enable Interrupts
Enable Int0
On Int0 eca
Reset portb
Do
Loop
End
eca:
Toggle Portb
Return

```

خوب حالا بریم سر تحلیل برنامه :

در خط اول ما با دستور **Config int0 = rising** پیکره بندی **int0** رو انجام دادیم یعنی به میکرو گفتیم که ما می خواهیم از **int0** استفاده کنیم . راستی ما بعد از علامت مساوی نوشتیم **rising** و به معنی این هستش که ما می خواهیم پایه **int0** نسبت به لبه بالا رونده حساس باشه

در خط دوم با دستور **config portb = output** ما پورت **b** رو به عنوان خروجی تعریف کردیم .

در خط سوم با دستور **enable interrupts** وقفه سراسری رو فعال کردیم . راستی اینو یادم رفت که بگم هر وقت که خاستی از **int0** استفاده کنی باید این وقفه سراسری رو فعال کنی .

در خط چهارم با دستور **enable int0** ما **int0** رو فعال کردیم . در ضمن اگر خاستی که در هر جای برنامه **int0** رو غیر فعال کنی می تونی به جای **enable** از **disble** استفاده کنی

در خط پنجم نوشتیم **On Int0 eca** این دستور این مطلب رو می رسونه که اگر پایه **int0** با لبه بالا رونده تحریک شد میکرو بپره بره توی زیر برنامه **eca**

خط ششم نوشتیم **reset portb** به این معنی که پورت **b** رو صفر کن

خط هفتم و هشتم هم به ترتیب **do-loop** هستش که یک حلقه بینهایت هستش ما این حلقه رو برای این گذاشتیم که میکرو اول برنامه که همه دستورات پیکره بندی رو خونند و رسید به این حلقه بیوفته توی این حلقه و تا زمانی که وقفه **int0** نخورد توی این حلقه گرفتار باشه حالا وقفه کی می خوره الان بهت می گم وقفه زمانی می خوره که پایه **int0** توسط یک لبه بالا رونده تحریک بشه .

خط نهم نوشتیم **end** که خودت می دونی چی هستش در ضمن برنامه ما هیچ وقت به **end** نمی رسه .

خط دهم نوشتیم **ahmad:** که یک زیر برنامه هستش که توی تحلیل دوم می گم که چرا نوشتمش .

خط یازدهم نوشتیم **toggle portb** به این معنی که حالت پورت **b** رو در هر وضعیتی که هست (منظورم صفر یا یک) معکوسش کن یعنی اگر یک بوده حالا صفرش کن و اگر صفر بوده حالا یکش کن .

خط دوازدهم نوشتیم **return** به این معنی که برگرد برو سر جای اولت یعنی دوباره برو توی حلقه .

خوب حالا بریم سر تحلیل دوم: میکروکنترلر میاد برنامه رو از خط اول شروع میکنه به خوندن خط اول نوشتیم **config int0 = rising** در اینجا میکرو میفهمه که ما **int0** رو پیکره بندی کردیم. در ادامه میاد خط دوم رو میخونه یعنی **config portb = output** در اینجا ما پورت **b** رو به عنوان خروجی تعریف کردیم که به این پورت **b** میتونی هر چیزی وصل کنی مثلا **LED** در خط سوم نوشتیم **enable interrupts** و با این کار به میکرو فرمان فعال کردن وقفه سراسری رو دادیم. راستی اینو بگم که هر وقت از **int0** خاستی استفاده کنی این وقفه سراسری رو هم با همین دستوری که نوشتیم یعنی **enable interrupts** فعال کن.

خط چهارم نوشتیم **enable int0** میکرو با خوندن این دستور **int0** رو برای ما فعال میکنه بعد میره به خط پنجم.

خط پنجم ما نوشتیم **on int0 eca** میکرو با خوندن این دستور می فهمه که هر وقت که به پایه **int0** یک پالس با لبه بالا رونده خورد باید بیره بره توی زیر برنامه **eca** و دستورات داخل این زیر برنامه رو اجرا کنه.

1- یک سوال میکرو از کجا میفهمه که وقتی که پالس با لبه بالا رونده به پایه **int0** خورد باید بره توی زیر برنامه چرا بالبه پایین رونده نره توی زیر برنامه

2- از اونجایی که ما در اول برنامه براش با دستور **config int0 = rising** مشخص کردیم که باید به لبه بالا رونده حساس باشه و میکرو هم تابع دستوری هستش که ما بهش میدیم در مورد این که چرا با لبه پایین رونده این کار رو نکنه باید بگم که کافیه به جای **rising** بدیم **falling** اون وقته که به لبه پایین رونده حساس میشه. خوب حال بریم سر ادامه تحلیل برنامه

خط ششم ما نوشتیم **reset portb** به این معنی که کل پورت **b** رو صفر کن. میکرو این خط رو میخونه و میره سراغ خط بعدی

خط هفتم و هشتم ما یک حلقه بینهایت یعنی **do-loop** نوشتیم که بدبختی میکرو اینجا شروع میشه میکرو با رسیدن به این دستورات میوفته توی یک حلقه بینهایت و اگه قسمت باشه تا ابد اونجا گرفتار میشه. مگر اینکه یک پالس فداکار بیاد و نجاتش بده.

1- منظورت رو نفهمیدم تا کی توی این حلقه **do-loop** هستش

2- تا زمانی که ما به پایه **int0** یک پالس با لبه بالا رونده بدیم. به محض اینکه ما این عمل رو انجام بدیم میکرو میفهمه که وقفه خورد و باید بره توی اون زیر برنامه **eca** که ما از قبل تعیین کرده بودیم. حالا بیا فرض رو بر این بگیریم که ما یک پالس با لبه بالا رونده به پایه **int0** دادیم و میکرو هم رفت توی زیر برنامه **eca** پس تحلیل برنامه رو از زیر برنامه **eca** که در خط دهم هستش پی می گیریم.

در خط دهم ما زیر برنامه **eca** رو داریم که میکرو الان پریده توی این زیر برنامه و می خاد دستورات داخل این زیر برنامه رو انجام بده:

اولین دستور داخل این زیر برنامه هستش **toggle portb** میکرو با خوندن این دستور پورت **b** رو **toggle** میکنه. چون در اول برنامه ما با دستور **reset portb** پورت **b** رو صفر کرده بودیم حالا که میکرو دستور

toggle portb رو انجام میده پورت **b** میشه یک یا به عبارتی عکس حالت قبل. در اینجا اگه که یک **led** رو به یک از پایه های پورت **b** وصل کرده باشی می بینی که **led** روشن میشه.

دومین یا به عبارتی آخرین دستور داخل این زیر برنامه هستش **return** میکرو با خوندن این دستور بر می گرده توی حلقه **do-loop** و مثل دفعه قبل توی این حلقه اونقدر میمونه تا دوباره یک پالس با لبه بالا رونده به پایه

int0 بخوره تا دوباره بره توی زیر برنامه **eca** و دوباره دستورات داخل این زیر برنامه رو اجرا کنه اگه این میره دوباره **portb** رو **toggle** کنه پورت **b** این دفعه صفر میشه به عبارتی عکس حالت قبل. یعنی اگه شما یک

led یکی از پایه های پورت **b** وصل کرده باشی **led** خاموش میشه.

اینم عکس سخت افزار مداری

<http://www.4shared.com/file/21102971/e36a084c/no1.html>

خوب حضری بریم سر مثال بعدی

1-اره ولی بزار این دفعه من یک مدار بگم تو برنامه شو بنویس

2-فکر خوبیه بگو

1-من یک مداری میخام به این صورت کار کنه . وقتی که تغذیه میکروکنترلر رو وصل میکنیم یک **led** روشن خاموش بشه به عبارت دیگر چشمک بزنه و یک **led** دیگه هم توی مدار باشه که در حالت اول خاموش باشه . یک میکروسویچ هم توی مدارمون باشه که باید وصل بشه به پایه **int0** این میکروسویچ رو هر بار که میزنیم **led** دوم **toggle** بشه یعنی برای بار اول که میکروسویچ رو زدیم **led** روشن بشه و برای بار بعدی که زدیم **led** خاموش بشه و این سیکل ادامه داشته باشه .

2-خوب من توی این مداری که گفتم **led** اولی که قرار یکسره چشمک بزنه رو وصل میکنم به پایه **b.0** و

led دومی که قراره به ازای هر بار که ما میکروسویچ رو میزنیم **toggle** بشه رو وصل کردم به پایه **b.1**

میکروسویچ رو هم که قراره به پایه **int0** وصل بشه رو هم وصل کردم به پایه **int.0**

حالا بریم سر برنامه :

```
Config Int0 = Rising
Config Portb = Output
Enable Interrupts
Enable Int0
On Int0 led
Do
Toggle Portb.0
Waitms 100
Loop
End
Led:
Toggle Portb.1
Return
```

تحلیل اول برنامه:

خط اول **config int0 = rising** با این دستور پایه **int0** رو برای پالس با لبه بالارونده پیکره بندی کردیم

خط دوم **config portb = output** در این دستور ما پورت **b** رو به عنوان خروجی پیکره بندی کردیم خط سوم **enable interrupts** در این دستور ما وقفه سراسری رو فعال کردیم که در هر جا که خاسته باشیم از **int0** استفاده کنیم باید این وقفه سراسری رو هم فعال کنیم

خط چهارم **enable int0** در این دستور ما **int0** رو فعال کردیم

خط پنجم **on int0 led** این دستور به این معنی هستش که هر وقت وقفه **int0** خورد بپر برو توی زیر برنامه . **led** در ضمن وقفه **int0** موقعی می خوره که ما یک پالس با لبه بالارونده به پایه **int0** بدیم .

خط ششم **do** این دستور اول یک حلقه بی نهایت هستش

خط هفتم **Toggle Portb.0** این دستور پایه **b.0** رو **toggle** میکنه

خط هشتم **Waitms 100** این دستور یک تاخیر **100** میلی ثانیه ای در اجرای برنامه ایجاد میکنه

خط نهم **loop** انتهای حلقه بینهایت رو مشخص میکنه

خط دهم **end** به معنی انتهای برنامه هستش که میکرو هیچ وقت این دستور رو نمی تونه بخونه

خط یازدهم **led:** ما در اینجا یک زیر برنامه با اسم **led** مشخص کردیم که درون این زیر برنامه دو دستور وجود دارد که در خط دوازدهم و سیزدهم هستند
خط دوازدهم **toggle portb.1** اولین دستور داخل زیر برنامه هستند و همونطور که از اسمش پیداست پایه **b.1** و **toggle** میکنه
خط سیزدهم **return** دومین و آخرین دستور داخل زیر برنامه هستند میکرو با خواندن این دستور برمیگرده توی حلقه بینهایت.

تحلیل دوم:

بین نمی خوام زیاد کشش بدم برای همین خیلی خلاصه می گم.
میکرو میاد از اول برنامه شروع میکنه به خواندن خط اول تا پنجم رو می خونه و به این نتیجه می رسه که ما یک از **int0** می خواهیم استفاده کنیم و در ضمن می خواهیم از پورت **b** به عنوان خروجی استفاده کنیم.
در خط ششم یعنی **do** که اول حلقه هستند میوفته توی حلقه و اولین دستور داخل این حلقه رو که هست **toggle portb.0** رو اجرا میکنه یعنی وضعیت پایه **b.0** رو عکس حالت قبلش میکنه از اونجایی که حالت قبل یا حالت فعلی هستند صفر این دفعه میاد پایه **b.0** رو یک میکنه و شما محلاضه میکنی که اون **led** ی که به این پایه وصل هستند روشن میشه .
بعد میره خط هفتم رو می خونه **waitms 100** این دستور **100** میلی ثانیه توی اجرای برنامه تاخیر ایجاد میکنه توی این **100** میلی ثانیه تاخیر پایه **b.0** در همون حالت یک بودن هستند . البته اینو بگم که در عمل از **100** میلی ثانیه بیشتر طول میکشه . بعد میره خط هشتم رو می خونه
خط هشتم هستند **loop** این دستور آخر حلقه رو مشخص کرده میکرو به محض اینکه این دستور رو می خونه می فهمه که باید بره از خط ششم دوباره شروع به خواندن کنه یعنی از دستور **do** این دستور رو می خونه و دوباره میوفته توی این حلقه و دستورات داخل این حلقه رو انجام میده و دوباره **portb.0** و **toggle** می کنه و سپس **100** میلی ثانیه تاخیر ایجاد می کنه و می رسه به **loop** و با خواندن این دستور **loop** همانند قبل دوباره می ره به خط ششم یعنی **do** و این سیکل همینطور ادامه داره . این قسمت از برنامه که داخل حلقه هستند مربوط به قسمت چشمک زن هست اگه توجه کنی **portb.0** یکسره داره **toggle** میشه البته با تاخیر **100** میلی ثانیه ای و با این کارش اون **led** که به پایه **b.0** وصل هستند رو روشن خاموش می کنه.
1- سوال دارم . این تاخیر **100** میلی ثانیه ای رو برای چی گذاشتی ؟
2- برای اینکه سرعت **toggle** شدن رو کم کنه یا به عبارت دیگه سرعت چشمک زدن **led** رو کم کنه
تا ما بتونیم اونو ببینیم .

1- یعنی ما میتونیم با کم و زیاد کردنش سرعت چشمک زدن **led** رو کنترل کنیم ؟
2- اره مثلا میتونی به جای **waitms 100** بنویسی **waitms 50** اینطوری سرعت چشمک زدن **led** زیاد میشه یا اگر عددش رو زیادتر کنی مثلا **200** یا **300** میتونی سرعت چشمک زدن **led** رو کم کنی . حالا بریم سر ادامه تحلیل برنامه

خب اونجایی بودیم که برنامه ما توی حلقه بی نهایت **do-loop** گیر کرده بود و با این کارش یک **led** رو برای ما روشن خاموش میکرد . حالا با فرض کنیم ما میکروسویچ رو فشار دادیم و با این کارمون یک پالس با لبه بالا رونده ایجاد کرده ایم که در این حالت وقفه **into** میخوره و میکرو میپره میره توی اون زیر برنامه ای که ما براش مشخص کردیم یعنی زیر برنامه **led** و دستورات داخل این زیر برنامه رو اجرا میکنه . حالا بیا فرض کنیم که این اتفاق هم افتاده و میکرو میخاد بره توی زیر برنامه **led**

اولین دستور داخل زیر برنامه **led** هستند **toggle portb.1** میکرو با خواندن این دستور پایه **b.1** رو **toggle** میکنه یعنی عکس وضعیت فعلی و از اونجایی که در وضعیت فعلی پایه **b.1** صفر بوده این دفعه توسط دستور **toggle portb.1** یک میشه و میبینی که اون **led** که به این پایه وصل شده روشن میشه . سپس

میکرو میره خط بعدی رو اجرا میکنه یعنی دستور **return** و با خوندن این دستور اونو اجرا میکنه و برمیگرده داخل همون حلقه **do-loop** و دوباره شروع میکنه به انجام دادن دستورات داخل همون حلقه . **do-loop** اگه دوباره میکروسویچ رو فشار بدی میکرو باز وقفش میخوره و میاد توی زیر برنامه **led** و دوباره پایه **b.1** رو **toggle** میکنه و چون در وضعیت فعلی پایه **b.1** یک هستش این دفعه با دستور **toggle portb.1** این پایه صفر میشه و میبینی که اون **led** که به این پایه وصل هستش خاموش می شه .

اینم شکل سخت افزاری مدار:

<http://www.4shared.com/file/21103003/f548a34d/no2.html>

-2- حالا اگه سوالی هستش بپرس

- 1- سوالم اینه که ایا ما توی این مدار با فشار دادن میکروسویچ یک لبه بالا رونده ایجاد میکردیم یا نه؟
 - 2- ره ببین ما تا وقتی که کاری به میکروسویچ نداشته باشیم وضعیت در حالت سطح صفر هستش . به محض اینکه میکرو سویچ رو فشار دادیم یک لبه بالا رونده اینجاد میشه . و اگر همچنان دستمون رو روی میکرو سویچ نگه داریم وضعیت در حالت سطح یک قرار داره . به محض اینکه ما دستمون رو از روی میکرو سویچ برداریم یک لبه پایین رونده ایجاد میشه که و دوباره به سطح منطقی صفر بر میگردد.
 - 2- خوب حالا که سوالات تموم شد بزار یک چیز رو دوباره یادآوری کنم و اون اینکه از **int0** در مواقعی استفاده میشه که ما خاسته باشیم هر وقت که پایه **int0** تحریک شد میکرو در هر وضعیتی که باشه بره و اون کاری رو که ما ازش خاستیم رو انجام بده که توی این دوتا مثالی که برات زدم کاملا مشخص هستش. یعنی میکرو اب دستشه بزاره و بره اون کاری رو که ما میگییم انجام بده.
- فعلا خدانگهدار

قسمت سیزدهم



سلام

2- سلام امروز می خام در مورد تایمر **2** برای ایجاد زمان خیلی دقیق برات بگم که اگه یک موقع خاستی یک زمان دقیق رو ایجاد کنی بتونی با تایمر کانتر دو این زمان رو ایجاد کنی از تایمر دقیق می تونی مثلا توی یک ساعت دقیق استفاده کنی.

1- بگو سرو پا گوشم

2- برای اینکه بتونی از تایمر کانتر **2** برای زمان دقیق استفاده کنی نیاز به یک **RTC** داری که مقدارش هم باید **32768** هر تز

باشه . این **RTC** یک قطعه دو پایه هستش که باید به پایه های **TOC1** و **TOC2** وصل بشه . شکل زیر رو نگاه کن:

<http://www.4shared.com/file/22309579/4f8a458d/RTC-AVR.html?dirPwdVerified=c95a46aa>

1- مدار خاص دیگه نیاز نداره ؟

2- نه هیچی نیاز نداره فقط کافیه این **RTC** رو به اون پایه هایی که گفتم وصل کنی

1- این **RTC** پایه مثبت یا منفی یا پایه تغذیه ای نداره ؟

2- نه فقط دوتا پایه داره که باید وصل کنی به پایه های **TOC1** و **TOC2** و هیچ چیز خاص دیگه ای نداره

1- این پایه های **TOC1** و **TOC2** کدوم یکی از پایه های میکرو هستن ؟

2- اینو دیگه باید از توی برگه مشخصاتش در بیاری ببینی که کجا هستش مثلا توی **MEGA8** این دوتا پایه

پایه های **9** و **10** هستن یا مثلا توی **MEGA32** پایه های **28** و **29** هستن

2- خوب این شد از سخت افزار مدار حالا بریم سر برنامه ای که باید نوشت:

برنامه رو با یک مثال ساده برات شروع می کنم برنامه به این صورت هستش که **PORTD** رو توی زمان دقیق یک ثانیه **TOGGLE** کنیم

مثال اول:

```
CONFOG TIMER2 = TIMER , ASYNC = ON , PRESCALE = 128
CONFIG PORTD = OUTPUT
ENABLE INTERRUPTS
ENABLE TIMER2
ON TIMER2 NEX
DO
LOOP
END
NEX:
TOGGLE PORTD
RETURN
```

برنامه همینیه هستش که می بینی دیدی که چقدر راحت هستش . خوب حالا بریم سر تحلیل برنامه:

توی خط اول که ما نوشتیم **CONFOG TIMER2 = TIMER , ASYNC = ON , PRESCALE =**

128 با این کار ما تایمر دو رو توی مد تایمر انتخاب کردیم **ASYNC = ON** این دستور که توی همین خط اول نوشتیم رو حتما باید بنویسی تا تایمر به صورت اسنکرون بیکره بندی بشه و کلاک رو از **RTC** دریافت کنه .

PRESCALE = 128 این دستور رو که قبلا توی بحث تایمر کانتر یک بررسی کردیم این **128** رو هم که

نوشتیم برای این هستش که بتونیم زمان یک ثانیه رو بدست بیاریم وگرنه میتونی یکی از اعداد **1 / 8 / 64 /**

128 / 256 / 1024 رو هم بنویسی تا زمان های کمتر یا بیشتری به دست بیاری

خط دوم **CONFIG PORTD = OUTPUT** در اینجا ما **PORTD** رو به عنوان خروجی تعریف کردیم

خط سوم **ENABLE INTERRUPTS** در این جا ما وقفه سراسری رو فعال کردیم که توی همه تایمرهایی که ما استفاده می کنیم باید این وقفه رو فعال کنیم.

خط چهارم **ENABLE TIMER2** چون که ما از تایمر شماره **2** میکرو میخایم استفاده کنیم باید خود

TIMER2 رو توسط این دستور فعال کنیم

خط پنجم **ON TIMER2 NEX** این دستور رو هم به این خاطر نوشتیم که زمانی که وقفه تایمر خورد پیره بره

توی زیر برنامه **NEX**

1- حالا این وقفه کی می خوره

2- صبر داشته باش بهت می گم. وقفه زمانی میخوره که زمان یک ثانیه رسیده باشه

2- خط ششم و هفتم ما یک حلقه **DO -LOOP** گذاشتیم که در ادامه می گم برای چه کاری هستش

خط هشتم **END** که میکرو هیچوقت این دستور رو نمی خونه یعنی نباید هم بخونه

خط نهم: **NEX:** این یک زیر برنامه هستش که اینم در ادامه توضیح میدم

خط دهم **TOGGLE PORTD** این دستور به این معنی هستش که پورت **D** رو **TOGGLE** کن یا به عبارتی

وضعیت پورت **D** رو از نظر صفر یا یک بودن معکوس کن یعنی اگه پورت صفر هستش حالا یکش کن یا اگه یک

هستش حالا صفرش کن

خط یازدهم **RETURN** با این دستور بر می گردیم تو حلقه **DO-LOOP**

حالا بریم سر تحلیل دوم راستی قبل از این که تحلیل دوم رو شروع کنم اینو بهت بگم که این برنامه رو نمی تونی

با شبیه ساز **BASCOM** و نه با پروتوس به طور دقیق تست کنی و باید حتما روی یک بردبرد ببندیش اینو گفتم

که یک موقع نری با این شبیه ساز تستش کنی بعد که زمان های غیر عادی به دست آوردی تو دلت به ما فحش

بدی.

تحلیل دوم برنامه:

میکرو میاد برنامه رو از خط اول شروع می کنه به خوندن خط اول رو می خونه و می فهمه که ما تایمر 2 را اون هم در حالت اسنکرون پیکره بندی کردیم بعد می ره خط دوم.

توی خط دوم میکرو می فهمه که ما PORTD رو به عنوان خروجی تعریف کردیم بعد می ره خط سوم توی خط سوم که ENABLE INTERRUPTS هستش میکرو وقفه سراسری رو فعال میکنه بعد میره خط چهارم

توی خط چهارم که ENABLE TIMER2 هستش میکرو تایمر 2 رو هم فعال میکنه بعد میره خط پنجم توی خط پنجم که ON TIMER2 NEX هستش میکرو می فهمه که هر وقت وقفه تایمر 2 خورد باید بپره بره توی زیر برنامه NEX که به جای NEX هر اسم دلخواه دیگه ای هم می تونی بزاری. در ضمن وقفه تایمر دو توی این برنامه زمانی می خوره که به زمان یک ثانیه رسیده باشیم. بعد میکرو میره به خط ششم. توی خط ششم و هفتم ما یک حلقه بی نهایت DO-LOOP گذاشتیم این حلقه رو به این به دو دلیل گذاشتیم اول به خاطر این که ما شاید برنامه های دیگه این هم داشته باشیم مثال خوندن دما یا خوندن کیبرد ماتریسی یا ... دلیل دوم اینکه میکرو اونقدر توی این حلقه بمونه تا وقفه تایمر 2 بخوره. خوب حالا بیا فرض رو بر این بگیریم که زمان یک ثانیه ایجاد شده و وقفه تایمر 2 خورده که در این جا میکرو می فهمه که وقفه خورده و زمان یک ثانیه سررسیده و باید بپره بره توی زیر برنامه NEX و دستورات داخل این زیر برنامه رو انجام بده. پس ادامه برنامه رو از خط زیر برنامه NEX یا همون خط نهم ادامه میدم:

اولین دستور داخل زیر برنامه NEX هستش TOGGLE PORTD در این جا میکرو میاد وضعیت پورت D رو معکوس میکنه چون وضعیت قبلی صفر بوده اینجا میاد پورت D رو یکش می کنه. اگه یک LED به یکی از پایه های PORTD وصل کنی می بینی که LED روشن میشه.

دومین دستور داخل زیر برنامه NEX هستش RETURN که میکرو وقتی به این دستور رسید میره توی اون حلقه ی DO-LOOP

اینم از برنامه دیدی که چقدر راحت بود

1-اره

2-خوب حالا که راحت بود یک دستور هستش که دیگه از این هم راحت تره حجم برنامه رو از اینی هم که

هستش راحت تر می کنه تازه یک سری امکانات هم داره

خوب بزار همین مثالی رو که بالا برات زدم رو توسط این دستور برات بنویسیم در ضمن توی سخت افزار هیچ

تغییری انجام نمیشه

```
CONFIG CLOCK = SOFT , GOSUB = SECTIC
CONFIG PORTD = OUTPUT
ENABLE INTERRUPTS
DO
LOOP
END
SECTIC:
TOGGLE PORTD
RETURN
```

همین . دیدی چقدر برنامه راحتی هستش

برنامه ساده هستش و به جز خط اول بقیه نیاز به توضیح نیست

توی خط اول ما نوشتیم CONFIG CLOCK = SOFT , GOSUB = SECTIC این دستور میاد تایمر

رو در مد سنکرون پیکره بندی میکنه اونجایی هم که نوشتیم GOSUB = SECTIC این معنی رو میده که

وقتی که به یک ثانیه رسیدیم بپر برو توی زیر برنامه SECTIC و دستورات داخل این زیر برنامه رو انجام بده

یک نکته مهم رو هم بگم که اون زیر برنامه که ما اسمش **SECTIC** هستش رو یک موقع عوض نکنی اسمشو ها که برنامه **EROR** میده . راستی این دستوراتی که نوشتیم فقط مخصوص یک ثانیه هستش و این این دستور بیشتر برای ساخت ساعت استفاده می شه .

یکی از امکاناتی که این دستور **CONFIG CLOCK = SOFT** در اختیار ما قرار می ده ساخت ساعت همراه با تاریخ هستش که با نوشتن دو دستور زیر می تونیم این کار رو انجام بدیم

```
DATE$ = "21/05/86"
```

```
TIME$ = "10:35:58"
```

توسط دو دستور بالا می توان توسط یک برنامه ای که الان می نویسم یک ساعت همراه با تاریخ درست کرد:

```
CONFIG CLOCK = SOFT , GOSUB = SECTIC
```

```
ENABLE INTERRUPTS
```

```
DATE$ = "21/05/86"
```

```
TIME$ = "10:35:58"
```

```
DO
```

```
LOOP
```

```
END
```

```
SECTIC:
```

```
LCD TIME$
```

```
LOCATE 2,1
```

```
LCD DATE$
```

```
RETURN
```

برنامه بالا زیاد نیاز به توضیح نداره

خط اول که ما تایمر رو در مد سنکرون پیکره بندی کردیم و به میکرو فهموندیم که فرکانس مورد نیاز تایمر رو باید از **RTC** دریافت کنه

خط دوم که وقفه سراسری رو فعال کردیم

خط سوم هم مقدار اولیه تاریخ رو دادیم که هر مقدار دیگه ای هم به غیر از اون هم می تونی بدی

خط چهارم هم مقدار اولیه ساعت رو دادیم که هر مقدار دیگه ای هم می تونی بدی

خط پنجم و ششم هم یک حلقه **DO-LOOP** گذاشتیم که میکرو اونتو بمونه تا وقتی که وقفه تایمر بخوره در

ضمن وقفه تایمر راس یک ثانیه می خوره. حالا فرض کن که وقفه تایمر خورده و پریده توی زیربرنامه

SECTIC . پس بریم که از میکرو عقب نمونیم بریم ببینیم که میکرو توی این زیربرنامه چکار می خاد انجام

بده .

اولین دستور داخل زیر برنامه **SECTIC** هستش **LCD TIME\$** میکرو باخوندن این دستور یک ثانیه به مقدار پیش فرضی که ما در خط های سه و چهار گذاشتیم اضافه می کنه وساعت رو به اصطلاح به روز می کنه مثلا ما در پیش فرض ساعت **10 : 35 : 58** رو نوشتیم ولی بعد از این که وقتی که وقفه تایمر خورد و اومد توی

زیر برنامه ساعت میشه **10 : 35 : 59**

دومین دستور داخل زیر برنامه **SECTIC** هستش **LOCATE 2,1** این دستور مکان نمای **LCD** رو به خط

دوم یعنی خط پایینی میاره و ادامه نوشتن رو از این خط ادامه میده در مورد این دستور در جلسات قبلی

توضیحات کافی دادم

سومین دستور داخل زیربرنامه **SECTIC** هستش **LCD DATE\$** این دستور هم تاریخ رو البته به صورت اتومات به روز می کنه البته از ادامه پیش فرضی که بهش دادیم عملکرد این دستور هم مثل دستور

LCD TIME\$ هستش .

آخرین دستور داخل زیربرنامه **SECTIC** هستش **RETURN** میکرو به محض خوندن این دستور بر میگره به

داخل حلقه **DO-LOOP**

1- همه اینایی که گفتمی درست ولی من یک چیز مهم رو نفهمیدم و اون اینکه توی مثال اولی که زدی توی برنامه

رو نوشتی بعد گفتی که این برنامه هر یک ثانیه می ره تو ی زیر برنامه و دستورات داخل این زیر برنامه رو انجام می ده حالا سوالی که برای من پیش اومده اینه که تو چطوری یک کاری کردی که هر یک ثانیه میکرو بره تو ی زیر برنامه ایا فرمول خاصی داره زود بگو که همین مشکل داره گیجم میکنه؟؟؟؟؟؟؟؟

2-اره فرمول داره فرمولش هم اینه $128 * 256 / 32768 = 1$ فرمولش همینیه که می بینی چیز خاص دیگه ای نداره حالا بریم سر اینکه اصلا این عدد ها چی هستش تا گیج نشی:

من گفتم که یک RTC به مقدار 32768 هر تری وصل کن به پایه های TOC1 و TOC2 بعد یک دستوری داشتیم تو ی خط اول **ASYNC=ON**

که با نوشتن این دستور تایمر 2 کلاکش رو دیگه از کریستال داخلی میکرو نمی گیره بلکه RTC که مقدارش هست 32768 هر تری دریافت می کنه . حالا فکر میکنی که تایمر 2 با این فرکانسی که از RTC دریافت می کنه چکار می کنه الان بهت می گم . ما تو ی خط اول یک دستوری داشتیم به نام

PRESCALE و مقداری که بهش دادیم این بود **PRESCALE = 128** ما با نوشتن این دستور به میکرو دستور دادیم که آقای AVR عزیز این فرکانسی رو که داری از RTC دریافت میکنی که مقدارش هم هست 32768 تقسیم بر عدد 128 کن . و میکرو هم این کار رو می کنه طبق این محاسبه : $32768 / 128 = 256$ یعنی در هر ثانیه دقیقا 256 هر تری به تایمر میرسه یا به عبارت دیگه در هر ثانیه 256 تا کلاک به تایمر 2 می خوره و تایمر این مقدار رو تو ی خودش می ریزه حالا از اونجایی که تایمر 2 با 255 تا کلاک پر و با 256 تا کلاک سرریز میشه وقتی که 256 تا کلاک بهش خورد سرریز میشه و وقفه می خوره و می ره تو ی اون زیر برنامه ای که براش تعریف کرده بودیم. در ضمن این 256 تا کلاک تو ی یک ثانیه ایجاد میشه یا به عبارت دیگه در هر ثانیه 256 تا کلاک به تایمر 2 می خوره طبق همون رابطه ای که نوشتیم منظورم رابطه ی $32768 / 128 = 256$ هستش.

1- تو نوشتی که **PRESCALE = 128** به جای عدد 128 هر عدد دیگه ای هم می تونیم بزاریم
2- هر عددی نه فقط اعداد 1 یا 8 یا 64 یا 128 یا 256 یا 1024 رو می تونیم بزاریم مثلا اگه به جای 128 عدد 64 رو بزاری تایمر 2 هر

0.5 ثانیه سرریز میشه طبق این رابطه $64 * 256 / 32768 = 0.5$ که در این رابطه 64 عدد **PRESCALE** هستش 256 مقدار گنجایش تایمر هستش که به ازای این مقدار تایمر سرریز میشه عدد 32768 هم که مقدار فرکانس یا کلاک RTC هستش. یا مثلا اگه عدد 256 رو بزاری تایمر 2 هر 2 ثانیه سرریز میشه

طبق این رابطه $256 * 256 / 32768 = 2$

1- پس مثال اگه عدد **PRESCALE = 1024** باشه طبق این رابطه ای که گفتی $1024 / 256 * 256 = 8$ یعنی تایمر هر هشت ثانیه سرریز میشه درست گفتم یا نه ؟؟

2-اره کاملا درسته درسته . دوباره تاکید می کنم که این برنامه رو تو ی پروتوس یک موقع تست نکنی که تاخیرش زیاد هستش و حتما روی برد برد مدارشو ببند.

فلا 

تهیه و جمع اوری : saeed450 – nex2004 از سایت ایران ویج

نویسنده : احمد فهیمی