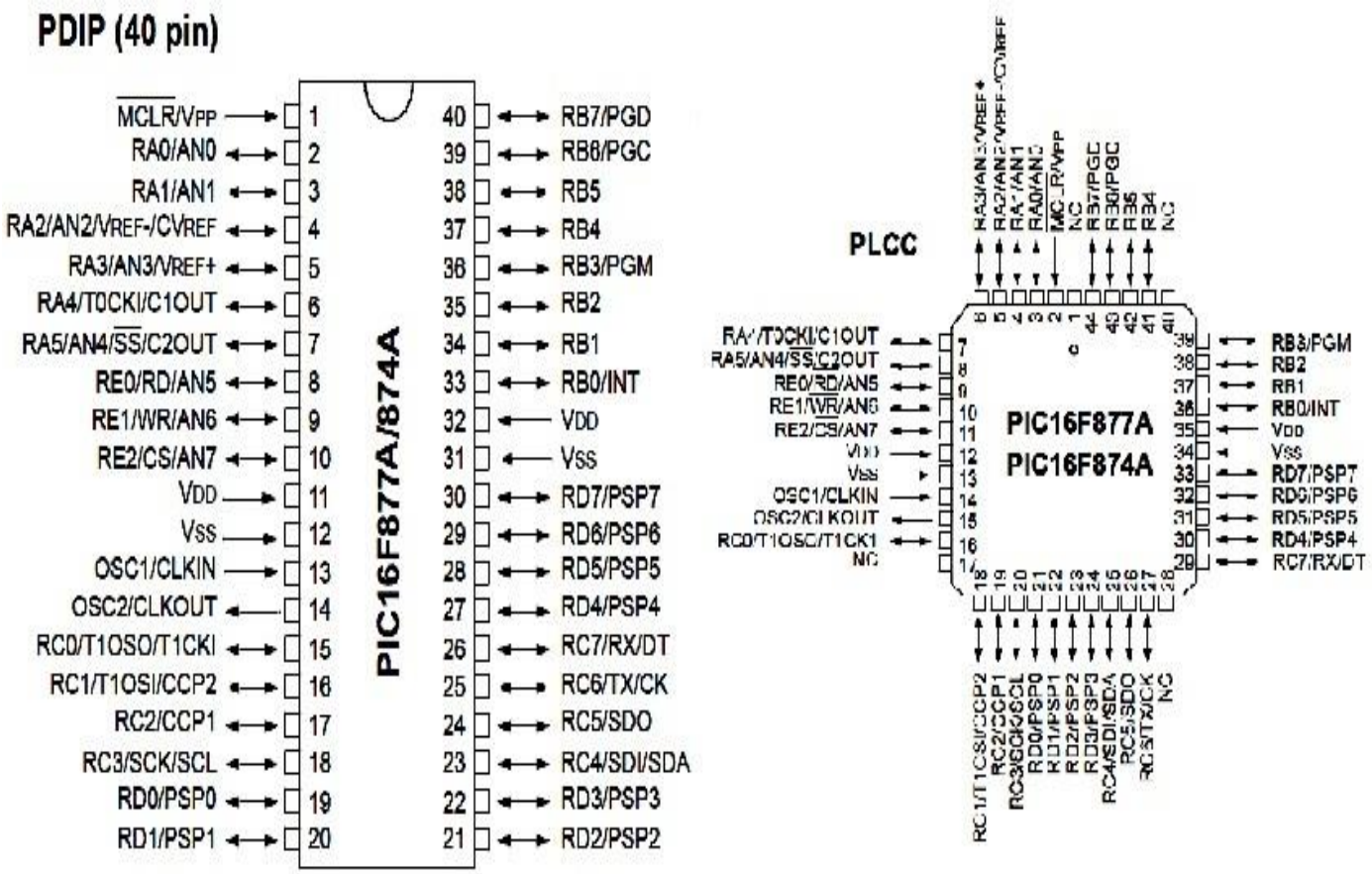
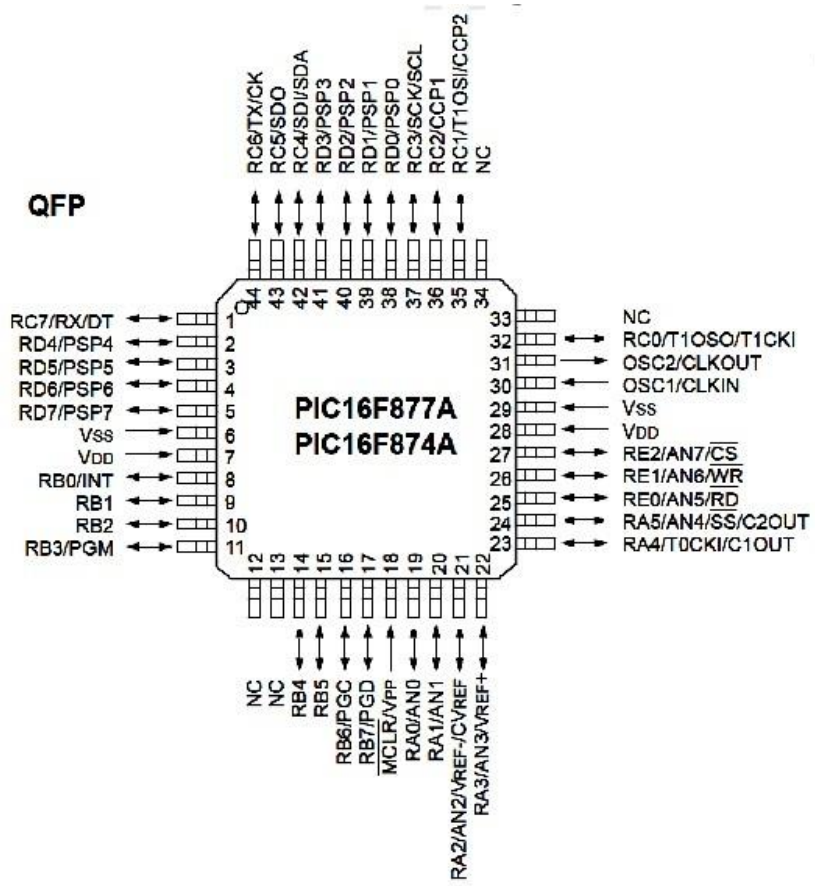
آموزش سریع و عملی میکروکنترلر pic

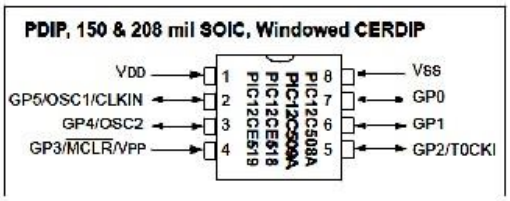
دانیال موسوی

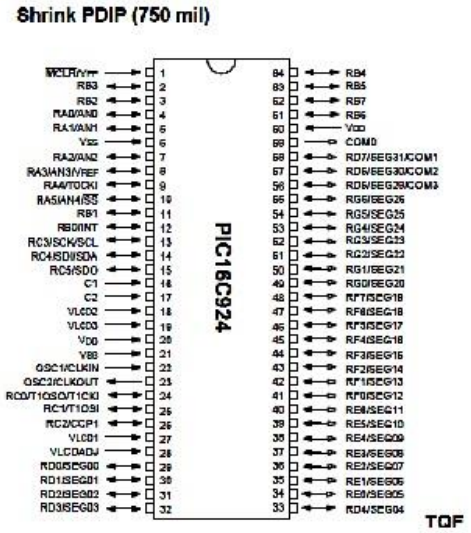
پروژه دات کام

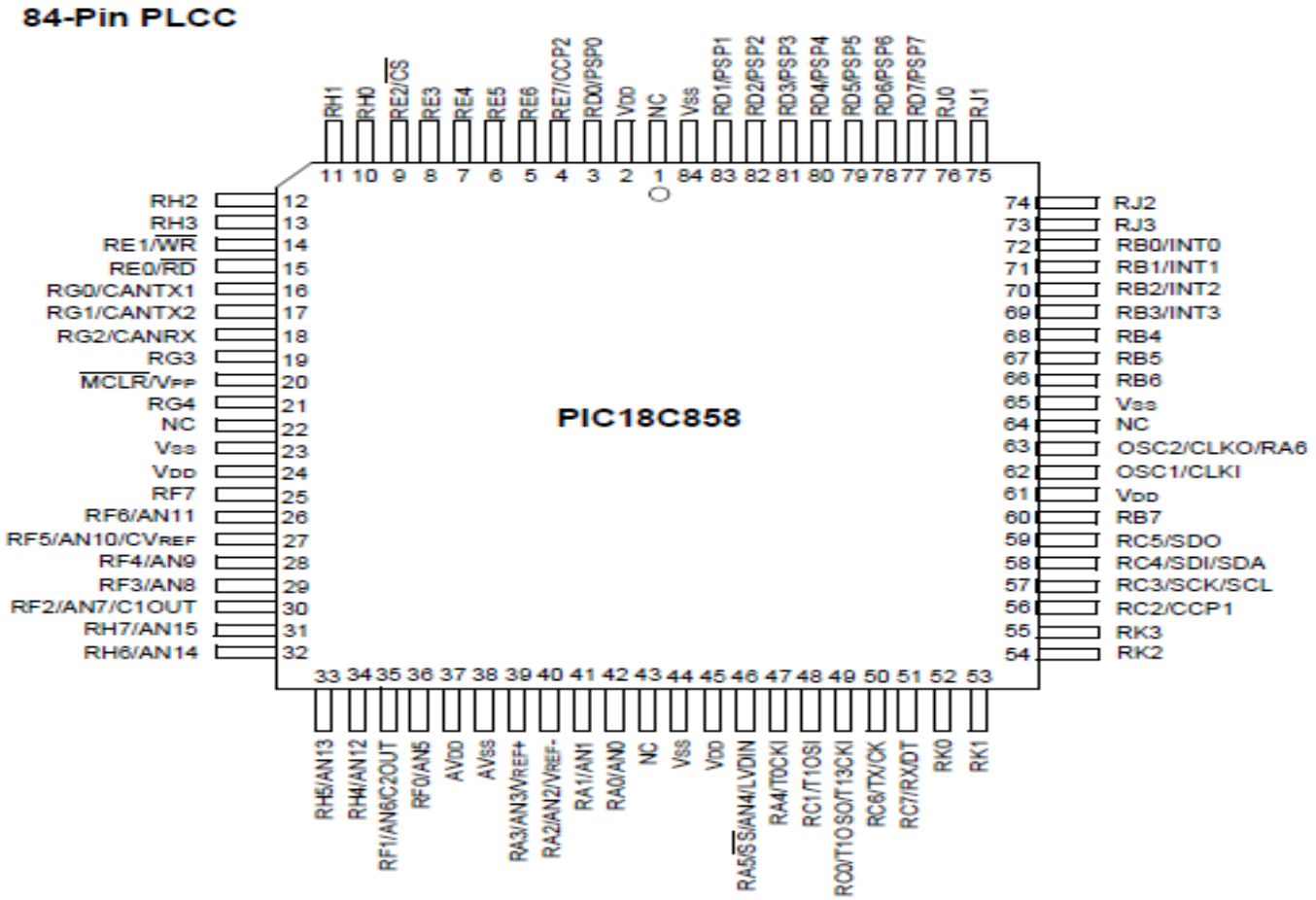
میتوان برای انتخاب میکرو مناسب به سایت microchip.com مراجعه کنید. نمایش شماتیک چند نمونه از این آی سی:











این آی سی میکرو کنترلر شبیه به AVR میباشد و قابل شبیه سازی با proteus میباشد. و در انواع 8 16 28 40 64 و84 پایه میباشد. میکروکنترلرهای PIC از نظر نوع حافظه به 4 دسته تقسیم می شوند:

- 1 دارای حافظه از نوع Flash هستند : آی سی هایی که دارای حرف F هستند مانند F8416
- 2 دارای حافظه از نوع Eprom هستند : آی سی هایی که دارای حرف C یا CE هستند مانند 16CE62516- C84
- 3 دارای حافظه از نوع Rom هستند:آی سی هایی که دارای حرف CR هستند مانند CR8416
- 4 دارای حافظه از نوع Eeprom هستند

جهت کار ابتدا باید یک زبان برنامه نویسی مانند c یا اسمبلی انتخاب نمود که در اینجا به زبان c با نرم افزار microc میپردازیم. کار با این نرم افزار بسیار ساده بوده ، به این صورت که با انتخاب New/Project و انتخاب نوع و فرکانس تراشه از آن استفاده میکنیم. (محدوده آنرا از کاتالوگ تراشه ببینید- در pic16f877 ماکزیمم فرکانس کاری 20Mhz میباشد-) سپس برنامه را نوشته ودر آخر روی Build در منوی بالا کلیک میکنیم تا خطایابی شود و فایل هگز خروجی برای پروگرامر یا پروتئوس فراهم شود. برای پروگرام کردن میتوان از پروگرمر های pikit موجود در بازار استفاده کرد.

قالب برنامه نویسی:1- تعریف کتابخانه 2- تعریف متغیر سراسری 3- تعریف تابع وقفه و توابع دیگر 4- تابع اصلی(شامل دستورات مورد نیاز و تعریف متغیرهای محلی)

مثال:

```
#include"name.h"

Char a,b=0;\\global variable's

Void interrupt(){statement's;}

Void main(){statement's;}
```

آخر دستورات از ; و برای توضیحات از // یا /*توضیحات*/ استفاده شود.

جهت برنامه نویسی ابتدا باید پورت های مورد نیاز را بصورت ورودی یا خروجی تعریف کرد توسط ثبات TRIS که برای حالت ورودی باید یک باشد و برای خروجی باید صفر شود.مثال:

```
TRISA=0; TRISB=0XFF; TRISC.F0=1;
```

توجه شود که برای نوشتن حروف هگزا دسیمال قبل عدد مورد نظر 0X و برای اعداد باینری 0B باید تایپ شود و ثبات ها و ریجیسترها با حروف بزرگ نوشته شوند. برای تعریف حالت(mod) پایه ای از پورت های میکرو باید آنرا بصورت بالا با Fx. تعریف کرد.

در صورت وصل کلید به ورودی باد آن پایه به یک مقاومت بالا کش(یک مقاومت بین پایه و منبع تغذیه- ترجیها 100k) یا پایین کش متصل شود.

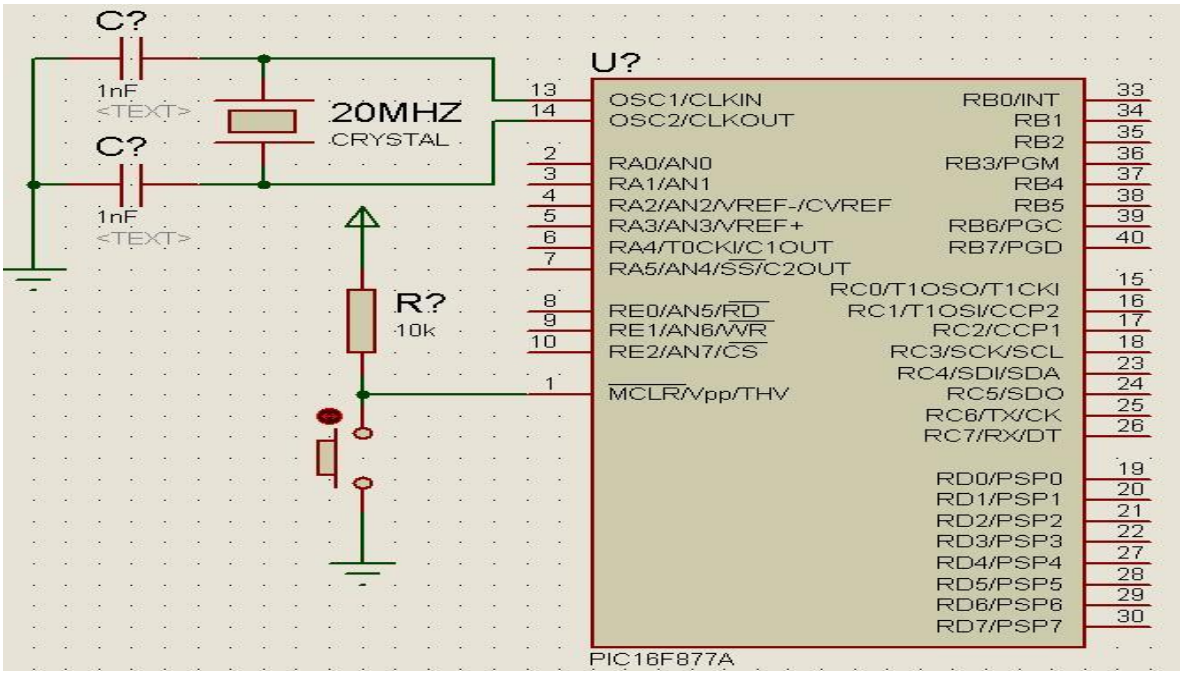
جهت قرار دادن اطلاعات در پورت خروجی باید به پورت آن مقدار دهیم: PORTA=55;

پورت GPIO:در میکرو های 8 پایه بجای پورت از GPIO استفاده میشود.6 بیت کم ارزش آن در دسترس است(بصورت i/o) پایه GP3 فقط بعنوان ورودی ست. GP0-3 دارای مقاومت بالا کش و درخواست وقفه است. با یک کردن INTCON.F3 میتوان این وقفه را فعال کرد. ثبات کنترلی حالت پورت های آن TRISGP میباشد.

پایه MCLR: پایه ریست میکرو است که باید به منبع وصل شود.

پایه osc1,2 :فرکانس کاری تراشه را بوسیله کریستال خارجی جهت فعالیت cpu فراهم میکنند.(خود میکرو اسیلاتور داخلی ندارد لذا باید از خارج بصورت زیر به تراشه متصل شود.)

فرکانس اسیلاتور(HZ)	مقدار خازن pf (C1,C2)
32k	33
200k	15-47-68pf
1M	15pf
4M	15pf
8M	15-33pf
20M	15-33pf



- بعنوان مثال برنامه زیر مقدار پورت A را در پورت B میریزد:

```
Void main(){TRICA=0XFF;TRICB=0;

        While(1){PORTB=PORTA;}}
```

پورت B : پورتی ست که دارای مقاومت بالاکش درونی میباشد که با بیت OPTION_REG.RBPU کنترل میشود که برای وصل مقاومت بالا کش مساوی 0 وبرای قطع آن مساوی 1 می شود.

پورت A: 6 بیتی میباشد.

وقفه INT:

این وقفه در PORTB.F0 وجود دارد ودر صورتی که ورودی این پایه یک شود بیت intcon.intf=1 می شود و زیر برنامه وقفه اجرا میشود.که باید ابتدا این پایه بصورت ورودی تعریف شود سپس INTCON.GIE=1; و INTCON.INTE=1; که حساس به لبه پایین رونده است چنانچه خواستیم حساس به لبه بالا رونده باشد OPTION_REG.INTEDG=1; . وصل یا قطع مقاومت بالا کش نیاز نیست.

تعریف تابع وقفه بصورت روبروست:

```
Void interrupt(){if(INTCON.INTF==1){ INTCON.INTF=0;
PORTA.F0=1;delay_ms(1000); PORTA.F0=0;}

Void main(){TRISA=0B000000;TRISB.F0=1;while(1);}
```

دستوراتی که میخواهید بعد آمدن وقفه اجرا شوند را درون تابع وقفه مینویسم.

اگر فقط از یک وقفه استفاده میکنید نیازی به گذاشتن شرط برای یک شدن پرچم وقفه ندارید:

```
Void interrupt(){INTCON.INTF=0; …..}

Void main(){……}
```

اگر از تابع وقفه نخواستید استفاده کنید و دستورات این وقفه را در دل تابع اصلی خاستید بنویسید وقفه INT را فعال نکنید و برنامه را بصورت زیر بنویسید:

```
Void main(){……while(1){ if(INTCON.INTF==1){ INTCON.INTF=0;
…//statements}

…..}}
```

وقفه تغیر در پورت B:

این وقفه مربوط به پایه های RB4-7 میباشد.میتوان در تابع وقفه برای اینکه کدامیک از این 4 پایه یک شده شرط گذاشت و تک تک آنها را چک کرد.برای این وقفه باید بیتهای وقفه سراسری (INTCON.GIE) و وقفه پورت B (INTCON.RBIE) را یک کرد و پایه های وقفه پورت B مورد نیاز را هم بعنوان ورودی تعریف کرد.

تایمر:

تایمر صفر:

تایمر 8 بیتی ست که میتوان از پالس ساعت داخلی یا از پالس خارجی استفاده کرد.درحالت استفاده از پالس خارجی محتوای ثبات TMR0 با هر پالس ورودی افزایش میابد و با سرریز این ثبات بیت وقفه INTCON.TMR0IF=1 میشود.فعال ساز این تایمر بیت intcon.tmr0ie است و OPTION_REG ثبات کنترلی این تایمر است.

OPTION_REG:

BRP U	INTED G	T0C S	T0S E	PS A	PS 2	PS 1	PS 0

T0CS :بایک شدن این بیت منبع پالس ساعت تایمر صفر داخلی در غیر اینصورت خارجی میشود.(بعدا توضیح داده خواهد شد)

T0SE :اگر یک شود محتوای تایمر صفر با لبه پایین رونده پالس ساعت خارجی روی پایه T0CLK افزایش میابد درغیر اینصورت با لبه بالا رونده.

PSA=0: تخصیص واحد پیش مقیا س دهنده به تایمر صفر. PSA=1: تخصیص واحد پیش مقیاس دهنده به تایمر نگهبان (whatchdog)

PS0-2 :نسبت واحد پیش مقیاس دهنده را تعین میکنند:

PS0-2	نسبت برای تایمر صفر	نسبت برای تایمر نگهبان
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

INTCON:

GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF

GIE: فعال ساز وقفه سراسری(در صورت یک شدن)

PEIE: فعال ساز وقفه جانبی

TMR0IE: فعال ساز وقفه سرریز

INTE:فعال ساز وقفه INT

RBIE: فعال ساز وقفه تغیر در پورت B

TMR0IF: پرچم وقفه سر ریز تایمر صفر

INTF: پرچم وقفه INT

RBIF: پرچم وقفه پورت B

مثال: یک شدن پورت B پس از سر ریز تایمر صفر:

```
Void interrupt(){PORTB=0XFF;INTCON.T0IF=0;}

Void
main(){TRISB=0;TMR0=0;OPTION_REG.RBPU=1;INTCON=0B10100000;

   While(1);}
```

تایمر یک :

16 بیتی ست و شامل دو ثبات TMR1H , TMR1L است که در صورت سر ریز PIR1.TMR1IF=1 میشود که اگر وقفه فعال باشد زیر برنامه آن اجرا میشود.برای فعال سازی وقفه: PIE1.TMR1IE=1; و INTCON.PEIE=1; میشود. ثبات کنترلی آن T1CON است.

T1CON:

-	-	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON

TMR1ON: فعال ساز تایمر یک

TMR1CS: فعال ساز نوسان ساز خارجی(یا کریستال) در صورت یک کردن این بیت نوسان ساز خارجی باید به پایه T1OSO/T1CLK اعمال شود.

T1SYNC: هم زمان سازی پالس خارجی با داخلی.ابتدا با فعال سازیTMR1CSاین بیت فعال میشود.

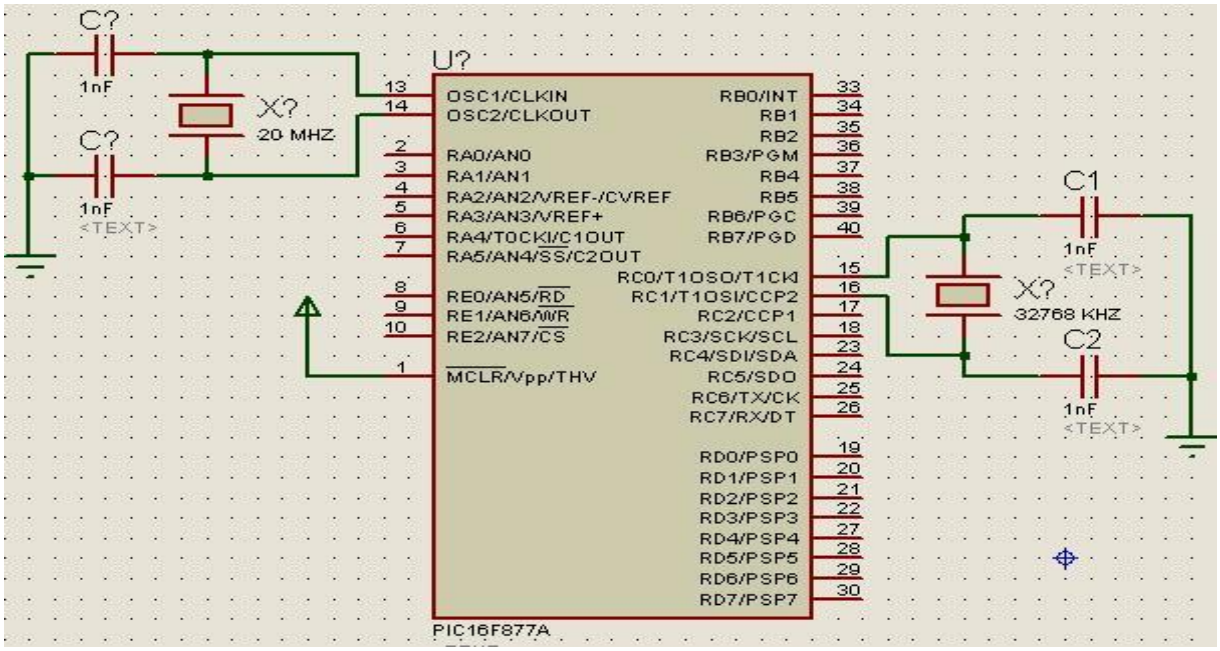
T1OSCEN: فعال ساز نوسان ساز خارجی تایمر یک که میتواند کریستال باشد (ماکزیمم 200MHZ)

T1CKPS0-1: نسبت تقسیم بسامد

نسبت تقسیم بسامد	T1CKPS0-1
1/1	00
1/2	01
1/4	10
1/8	11

بوسیله این نسبت تقسیم کلاک پالس تایمر یک تقسیم بر عدد موجود در جدول میشود. و سرعت شمارش میتواند تا 8 برابر کاهش یابد.

کریستال ساعت کریستالیست با فرکانس کاری 32768hz که برای ساخت ساعت از آن استفاده میشود یعنی موقعی که محتوای TMR1H=128 (32768/256=128) زمان 1 ثانیه سپری شده است. برای استفاده از کریستال باید پایه های آن بصورت خروجی و شمارنده هم زمان تعریف شوند.



فرکانس	c1,c2
23768 hz	33 pf
100,200 khz	15 pf

اگر T1OSCEN=1 , T1SYNC=1 باشد تایمر یک با هر لبه بالا رونده پالس روی پایه T1OSI افزایش می یابد و اگر T1OSCEN=0 باشد محتوای ثبات های تایمر یک با هر لبه بالارونده پایه T1OSO/T1CLK افزایش میابد.

نکته: فقط در دل تابع وقفه میتوان به TMR1H , TMR1L دسترسی داشت ودر ابتدای برنامه باید صفر شوند. در برنامه اگر T1CON.TMR1ON=0 شود شمارش PUSE شده وبا یک شدن آن شمارش ادامه میابد.پرچم سر ریز را باید پس از یک شدن صفر کرد.با هر تغیر در TMR1L برنامه وقفه اجرا میشود لذا باید دستوراتی که میخواهیم پس از سر ریز اجرا شوند را با شرط بیاوریم.

مثال: شمارنده غیر همزمانی که در صورت وقوع سر ریز LED را روشن کند و TMR1L را در پورت B نشان دهد:

```
#Define  LED  PORTC.F3

void main(){TRISB=0;TRISC.F3=0;TMR1L=0XF1;TMR1H=0XFF;
T1CON=0B00001111;

    while(1){ PORTB=TMR1L;   if (PIR1.TMR1IF){ LED=1; }}}
```

نکته: می توان در دل تابع اصلی نیز وقفه های سر ریز را چک کرد.

نکته: در نرم افزار proteus برای تراشه از کریستال نمیتوان استفاده کرد و باید یک سیگنال فرکانس پایین مثلا 256hz به ورودی RC0 اعمال کرد.

تایمر2:

تایمری 8 بیتیست که دو ثبات PR2 , TMR2 را مدام مقایسه کرده در صورت برابری وقفه آن فعال میشود. منبع پالس آن داخلی ست. دارای دو واحد: 1- پیش مقیاس: تقسیم بسامد نوسان ساز 2- پس مقیاس: تعداد دفعاتی که دو ثبات با هم برابر می شوند.در صورتی که واحد پس مقیاس سر ریز کند پرچم TMR2IF یک میشود و اگر وقفه آن فعال باشد (RIE1.TMR2IE=1) زیر برنامه اجرا میشود. ثبات کنترلی آن:

T2CON:

-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

TMR2ON: بیت فعال ساز تایمر 2

TOUTPS3:0	نسبت تقسیم بسامد واحد پس مقیاس دهنده		T2CKPS1:0	نسبت بسامد نوسان ساز در خروجی واحد پیش مقیاس
0000	1:1		00	1:1
0001	1:2		01	1:4
0010	1:3		10	1:16
0011	1:4		11	1:16
...	...			
1111	1:16			

:CCP

دارای 3 حالت مقایسه ، ثبت و PWM میباشد.هر واحد دارای ثبات 16 بیتی CCPRxL , CCPRxH است که x شماره واحد CCP میباشد. ثبات کنترلی آن CCPxCON :

-	-	CCP xX	CCP xY	CCPx M3	CCPx M2	CCPx M1	CCPx M0

CCPxX,Y : این بیتها همراه با ثبات CCPRxL زمان یک بودن سیگنال PWM را تعین میکنند.

حالت واحد CCP	عملکرد واحد CCP	CCPxM3:0
خاموش	خاموش	0000
ثبت	ثبت با هر لبه پایین رونده	0100
ثبت	ثبت هر لبه بالا رونده	0101
ثبت	ثبت با4 لبه بالا رونده	0110
ثبت	ثبت با 16 لبه بالارونده	0111
مقایسه	در صورت برابری مقایسه ، پایه CCPx و پرچم CCPIF یک میشوند	1000
مقایسه	در صورت برابری مقایسه ، پایه CCPx از یک به صفر تغیر میابد و پرچم CCPIF یک میشوند	1001
	در صورت برابری مقایسه ، پایه CCPx بدوم تغیر میماند و پرچم CCPIF یک میشوند و وقفه ای نرم افزاری ایجاد میشود.	1010
	واحد CCP در حالت تحریک رویداد خاص عمل کرده و بیت CCPIF یک میشود.	1011
PWM	حالت PWM	1100-1111

1- **حالت مقایسه:** در این حالت محتوای ثبات های CCP (CCPRxH , CCPRxL) با ثبات های تایمر 1 (TMR1H , TMR1L) مقایسه شده ، در صورت برابری نتیجه روی پایه CCPx طبق جدول نمایش میابد. در صورتی که بیت PIE1.CCPxIE=1 باشد برنامه وقفه اجرا شده و بیت

PIR1.CCPxIF یک میشود. پایه CCP باید بصورت خروجی و تایمر 1 در حالت همزمان تعریف شوند.

2- حالت ثبت: اگر روی پایه CCPx رخدادی اتفاق بیفتد محتوای ثبات تایمر1 در ثبات CCP ثبت شده (طبق جدول) و پرچم PIR1.CCPxIF=1 میشود و اگر وقفه CCP فعال باشد (PIE1.CCPxIE=1) زیر برنامه آن اجرا میشود. پایه CCP در حالت ورودی و تایمر1 در حالت همزمان تعریف شود. ریجیستر INTCON مقدار دهی نشود.

برای ساخت فرکانس متر حالت ثبت با 16 لبه بالا رونده برنامه ریزی شود.

3- حالت PWM:در این حالت یک سیگنال PWM روی پایه CCP قرار میگیرد.(پایه CCP بصورت خروجی تعریف شود) در این حالت از تایمر2 استفاده میشود. تایمر دو دارای دو ثبات کنترلی , TMR2 PR2 میباشد. PR2 فرکانس سیگنال PWM را تعین میکند طبق فرمول زیر:

$$F_{PWM} = \frac{1}{(PR2+1) \times 4 \times T_{OSC} \times P_{r\,SCALER_{TIMER2}}}$$

دیوتی سایکل(زمان یک بودن) سیگنال pwm توسط ثبات CCPRxL و بیتهای CCPxX , CCPxY طبق فرمول زیر تعین میشود.

$$= (\text{CCPRxL \& CCPxX:CCPxY}) \times T_{OSC} \times P_{r\,SCALER_{TIMER2}} T_{ON}$$

مثال: سیگنالی PWMو 50KHZ با دیوتی سایکل 40درصد تولید کنید.(فرکانس اسیلاتور 8MHZ)

$$= 1/50 = 20\mu S \; T_{PWM}$$

$$T_{OSC} = \frac{1}{8M} = .125\mu S$$

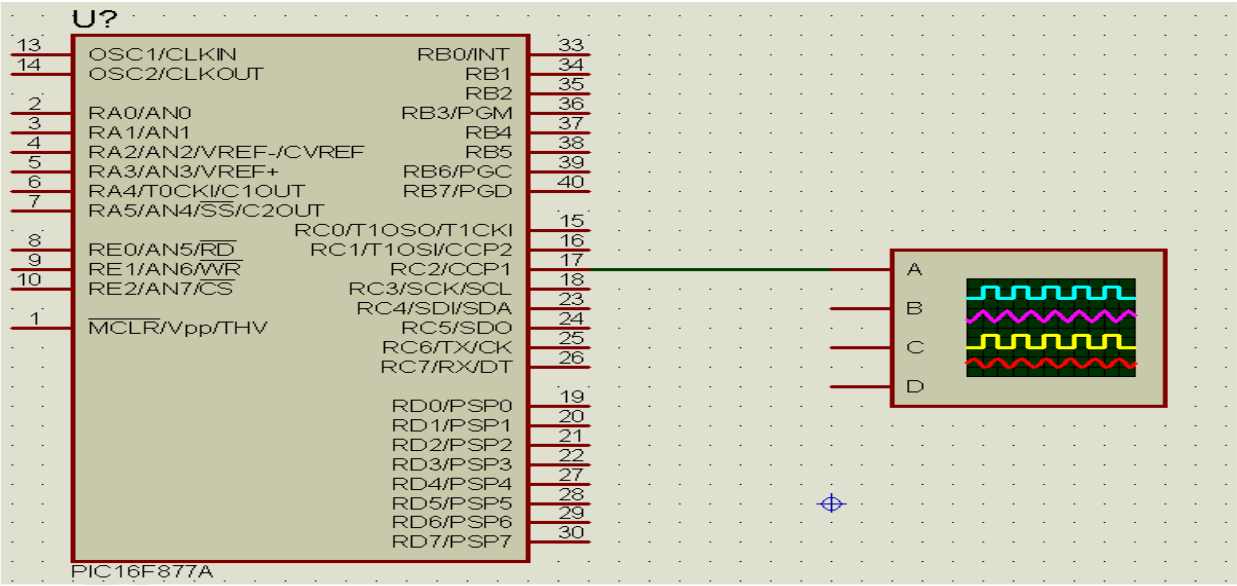
$$20 = (PR2+1) \times 4 \times .125 \times 1 \Rightarrow PR2 = \frac{20}{4 \times .125} - 1 = 39$$

$$\text{D.C}=.4 \Rightarrow T_{ON} = T_{PWM} \times .4 = 8\mu S$$

$$8 = (\text{CCPRxL \& CCPxX:CCPxY}) \times .125 \times 1 \Rightarrow \text{CCPRxL \& CCPxX: CCPxY} = 64$$

$$= 0001001100$$

یعنی بیتهای CCPxX,Y صفر و CCPRxL=0B00010011



Void main( ){ TRISC.F2=0; CCPR1L=0B00010011; PR2=39;

CCP1CON=0B00001100;

T2CON.TMR2ON=1; while(1); }

واحد مرجع ولتاژ:

وظیفه این واحد تولید ولتاژ مرجع (خروجی آنالوگ) است. دارای ثبات کنترلی زیر است:

CVRCON:

CVR EN	CVR OE	CVR R	-	CV R3	CV R2	CV R1	CV R0

CVREN: فعال ساز ولت مرجع

CVROE: اگر یک شود ولت مرجع بصورت داخلی به ورودی مقایسه کننده و پایه - VREF اعمال میشود.

CVRR: اگر یک باشد ولت مرجع بین 0 تا $0.75\ V_{DD}$ به فاصله $\frac{V_{DD}}{24}$ تولید میشود و اگر صفر باشد ولت مرجع بین $.25V_{DD} - .75V_{DD}$ به فاصله $\frac{V_{DD}}{32}$ تولید میشود.

CVR3:0: تعین ولت مرجع طبق فرمول زیر:

CVRR=0 => Vref=$.25V_{DD} + \frac{CVR3:0}{32} V_{DD}$ CVRR=1 => Vref= $\frac{CVR3:0}{24} V_{DD}$

واحد مقایسه کننده:

دو واحد مقایسه با هم مقایسه شده و نتیجه در خروجی نمایش میابد. در PIC16F877 ورودی های مقایسه کننده1 AN3 , AN0 و مقایسه کننده2 AN1 , AN2 میباشند.

CMCON:

C2O UT	C1O UT	C2I NV	C1I NV	CI S	CM 2	CM 1	CM 0

C2OUT: اگر AN1<AN2 باشد C2OUTیک میشود. AN1 پایه +Vin و AN2 پایه -Vin میباشد.

C1OUT: اگر AN0 (پایه -Vin) کوچکتر از AN3 (پایه +Vin) شود این بیت یک میشود.

C2INV: معکوس کننده خروجی مقایسه کننده2

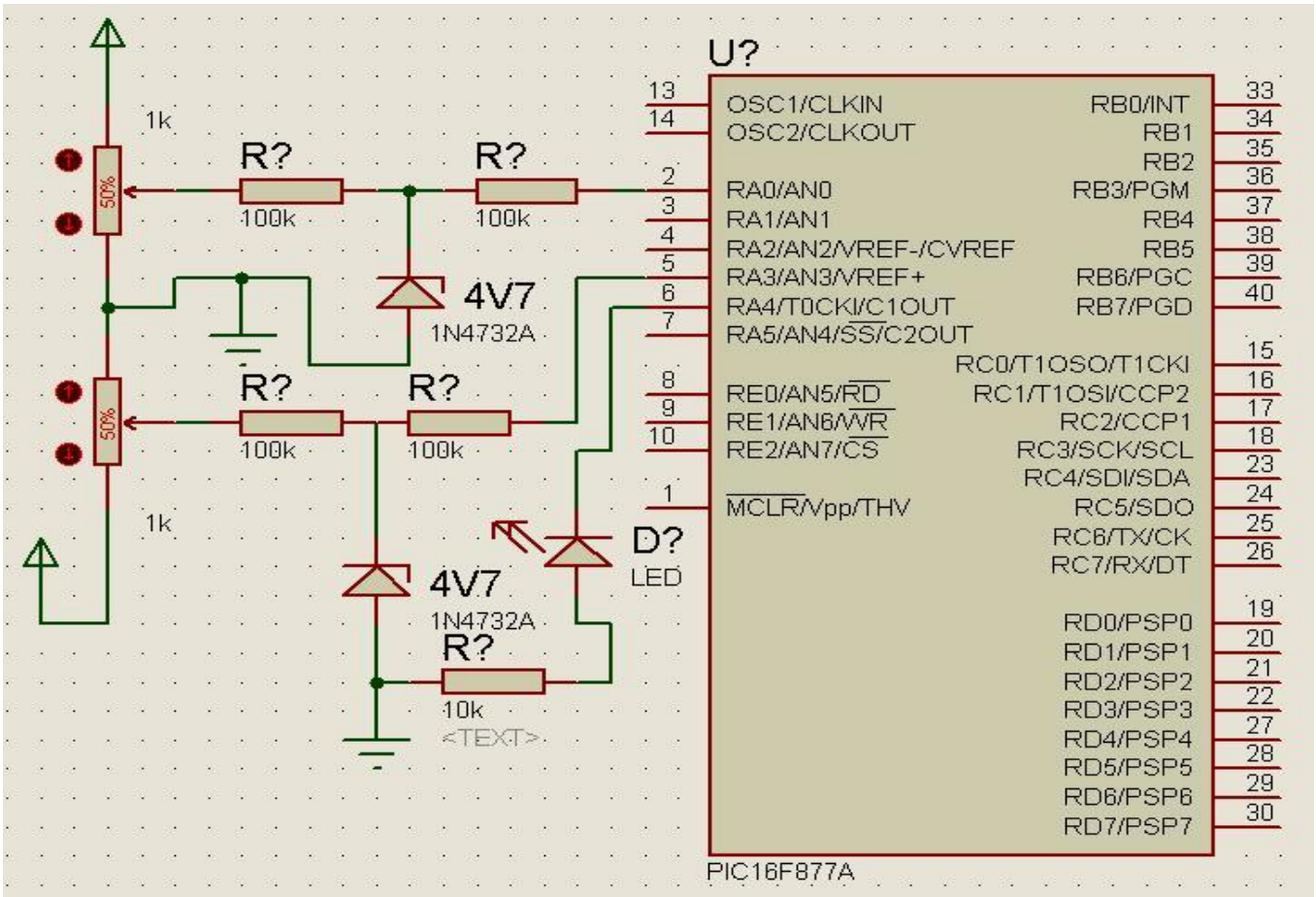
C1INV: معکوس کننده خروجی مقایسه کننده1

CIS: انتخاب کننده ورودی مقایسه کننده. اگر CM2:0=110 باشد و CIS=0: ورودی منفی مقایسه کننده1(-Vin) به پایه AN0/RA0 ورودی منفی مقایسه کننده2(-Vin) به پایه AN1/RA1وصل میشود. اگر CM2:0=110 باشد و CIS=1: : ورودی منفی مقایسه کننده1(-Vin) به پایه AN3/RA3 ورودی منفی مقایسه کننده2(-Vin) به پایه AN2/RA2وصل میشود.

CM2:0	حالت مقایسه کننده
0000	مقایسه کننده ها خاموش
0001	مقایسه کننده1 روشن و 2 خاموش نتیجه روی پایه RA4/C1OUT
0010	دو مقایسه کننده بصورت مستقل فعال
0011	دو مقایسه کننده بصورت مستقل فعال اند و نتیجه روی پایه های خروجی نمایش میابد

0100	دو مقایسه کننده فعال و مرجع ولتاژ یکسانی دارند
0101	دو مقایسه کننده فعال و مرجع ولتاژ یکسانی دارند و نتیجه روی خروجی نمایش میابد
110	دو مقایسه کننده فعال همراه با 4 ورودی که توسط cis کنترل میشوند
111	مقایسه کننده ها خاموش

فعال سازی وقفه مقایسه کننده توسط بیت های PIE2.CMIE , GIE , PEIE , INTCON است(باید یک شوند). هر تغیری در پایه خروجی باعث فعال شدن پرچم وقفه مقایسه کننده میشود(CMIF.PIR2). محدوده ولتاژ ورودی باید بین Vss تا VDD باشد.برای حفاظت از میکرو مدار مدار ورودی را بصورت زیر ببندید.



مثال: طبق مدار بالا برنامه ای بنویسید که ورودی های مقایسه کننده یک را مقایسه و نتیجه را در خروجی نشان دهد:

```
void main(){ TRISA.F4=0; CMCON=0B00000001; }
```

واحد مبدل آنالوگ به دیجیتال :

وظیفه این واحد گرفتن سیگنال آنالوگ از یکی از پایه (کانال)های ورودی آنالوگ (ANx) و تبدیل آن به عددی بین 0 – 1111 1111 (در نوع 8 بیتی) میباشد. مثلا اگر ورودی ماکزیمم 5 ولت باشد و 5 ولت به ورودی اعمال شود همه بیتهای ریجیستر مربوطه یک میشود معادل 255:

$\frac{255}{5}$ =51 => $\frac{255}{51} = 5$ V OR $\frac{100}{51} = 1.961\,V$

همانگونه که نشان میدهد عدد تبدیل شده توسط این واحد که در ریجیستر مربوطه ذخیره شده را تقسیم بر 51 میکنیم و مقدار ولتاژ ورودی مشخص میشود.

انواع مبدل های آنالوگ به دیجیتال:

1- ده بیتی با 8 کانال ورودی
2- 8 بیتی با 4 کانال ورودی
3- 8 بیتی با 8 کانال ورودی

PIC16F877 دارای نوع 10 بیتی با 8 کانال ورودیست. طرز کار این واحد به این صورت است که از یکی از کانال های ورودی سیگنال آنالوگ دریافت و کد دیجیتالی متناسب با آن در , ADRESL ADRESH ذخیره میشود. ثبات های کنترلی آن:

ADCON:

ADCS 1	ADCS 0	CHS2	CHS1	CHS0	GO/ *DONE*	-	ADON

ADON: فعال ساز مبدل آنالوگ به دیجیتال

GO/DONE: با یک شدن این بیت عمل تبدیل A/D شروع میشود.

CHS2:0	تعین کانال ورودی
000	کانال 0 (AN0)
001	کانال 1 (AN1)
010	کانال 2 (AN2)
011	کانال 3 (AN3)

100	کانال 4 (AN4)
101	کانال 5 (AN5)
110	کانال 6 (AN6)
111	کانال 7 (AN7)

ADCS2:0	تعین کلاک پالس مبدل A/D
000	FOSC/2
001	FOSC/8
010	FOSC/32
011	FRC
100	FOSC/4
101	FOSC/16
110	FOSC/64
111	FRC

ADCS2 در ثبات ADCON1 است.

ADCON1:

ADFM	ADCS2	-	-	PCFG3	PCFG2	PCFG1	PCFG0

ADRESH								ADRESL								AD FM
-	-	-	-	-	-	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	0
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	-	-	-	-	-	-	1

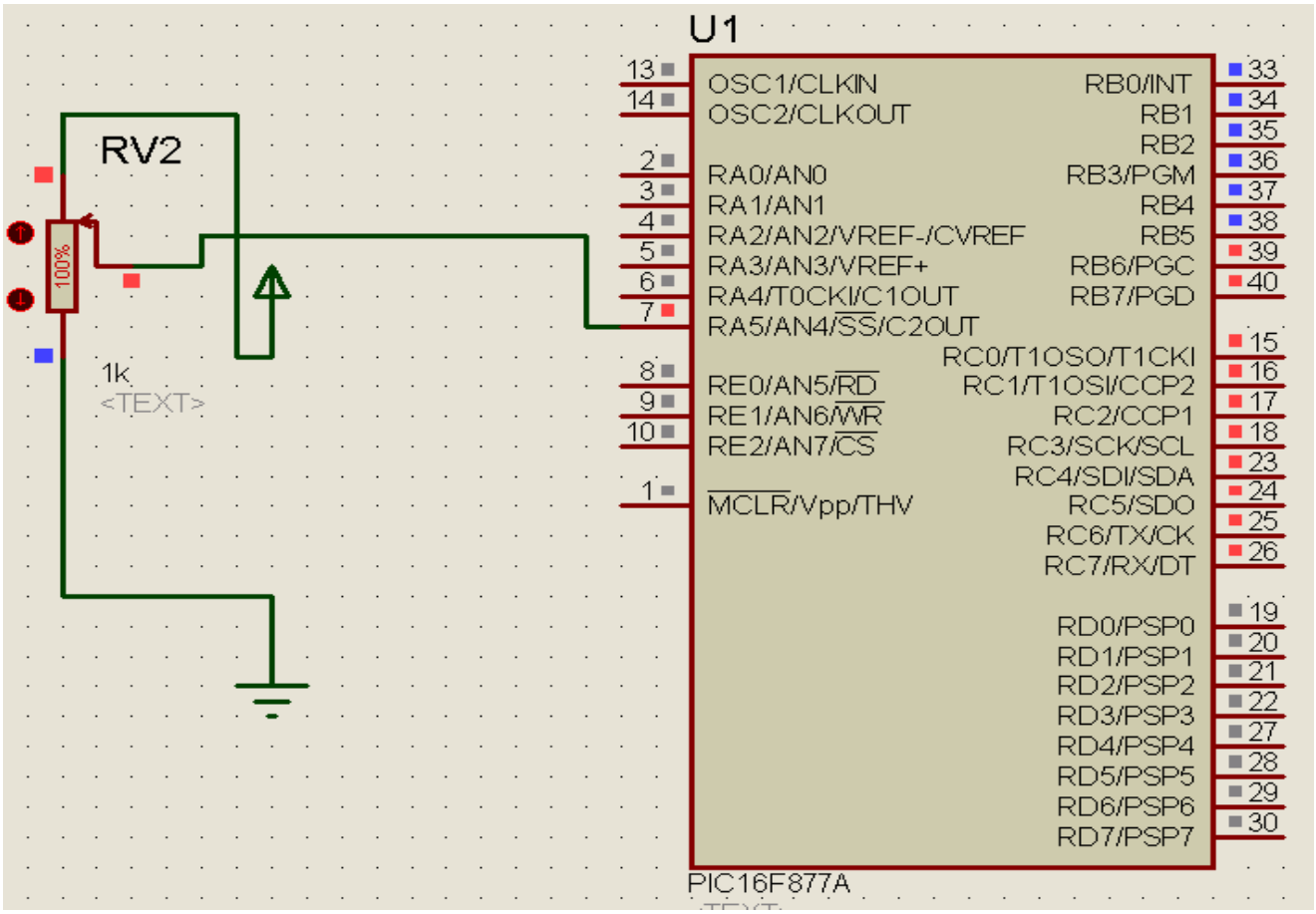
پیکره بندی پایه های مبدل A/D:

PCFG3:0	AN	AN	AN	AN	AN	AN	AN	AN	VREF+	VREF-	C/R

	7	6	5	4	3	2	1	0			
0000	A	A	A	A	A	A	A	A	**V**DD	**V**SS	8/0
0001	A	A	A	A	**V**REF+	A	A	A	AN3	**V**SS	7/1
0010	D	D	D	A	A	A	A	A	**V**DD	**V**SS	5/0
0011	D	D	D	A	**V**REF+	A	A	A	AN3	**V**SS	4/1
0100	D	D	D	D	A	D	A	A	**V**DD	**V**SS	3/0
0101	D	D	D	D	D	**V**REF-	A	A	AN3	**V**SS	2/1
0110_0111	D	D	D	D	D	D	D	D	-	-	0/0
1000	A	A	A	A	**V**REF+	**V**REF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	**V**DD	**V**SS	6/0
1010	D	D	A	A	**V**REF+	A	A	A	AN3	**V**SS	5/1
1011	D	D	A	A	**V**REF+	**V**REF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	**V**REF+	**V**REF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	**V**REF+	**V**REF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	**V**DD	**V**SS	1/0
1111	D	D	D	D	**V**REF+	**V**REF-	D	A	AN3	AN2	1/2

پس از انجام عمل تبدیل A/D و قرار گرفتن نتیجه در ADRESL , ADRESH پرچم PIR1.ADIF یک میشود و زیر برنامه وقفه اجرا میشود که باید آنرا صفر کرد. فعال سازی وقفه واحد A/D توسط PIE.ADIE پایه کانال ورودی باید بصورت ورودی تعریف شود. تابع Y=Adc_Read(X) یک تابع 10 بیتی ست که حاصل تبدیل A/D را در متغیر مثلا Y (باید حداقل از نوع INT باشد) قرار میدهد ، X نیز مشخص کننده کانال ورودیست که میتواند عدد باشد. این تابع میکرو های از نوع 18F با شماره های 2331 و 2431 و 4331 و 4431 را پشتیبانی نمیکند.

مثال:




```
void main( ) {TRISB=0;TRISC=0;TRISA.F5=1;
ADCON0=0B00100001;ADCON1=0B100000;

while( 1){ADCON0.F2=1;while( PIR1.ADIF==0);PIR1.ADIF=0;PORTB=ADRESL;
        PORTC=ADRESH;}}
```

حافظه EEPROM:

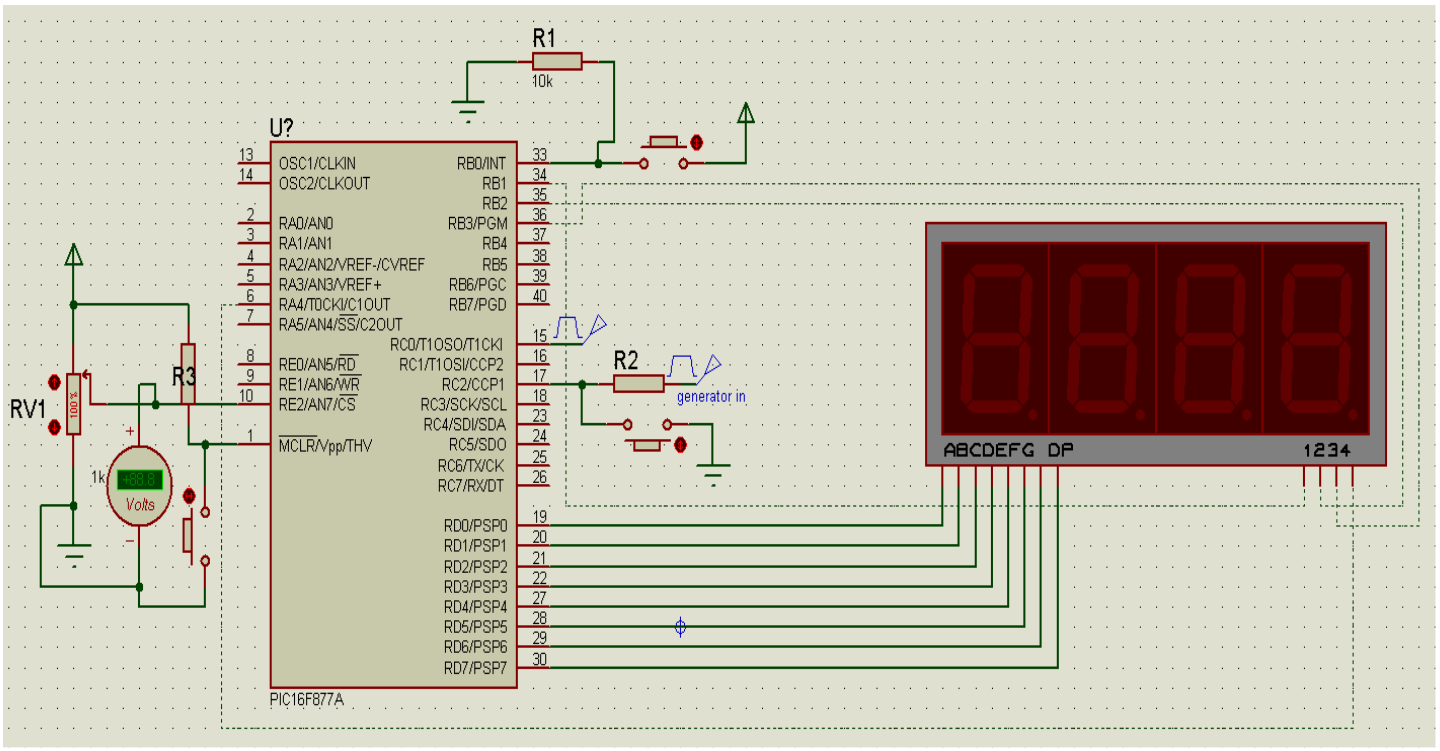
این حافظه ، بخاطر فرار بودن حافظه داخلی میکرو کارایی دارد و در صورت قطع برق یا ریست شدن تراشه مقدار خود را حفظ میکند که جهت کار با آن در زیر دو تابع معرفی میشود. PIC16F877 دارای 256بایت حافظه EEPROM است.

Eeprom_Write(address , data) در دو متغیر این تابع آدرس و اطلاعات را میریزیم که میتوانند بجای متغیر عدد باشند.

Eeprom_Read(address) که فقط آدرس متغیر مورد نظر را میدهیم و میتوان آنرا در متغیری دیگر ذخیره کرد.

نکته: هنگامی که میخواهیم از این دو تابع بصورت پی در پی استفاده کنیم باید از تاخیر حداقل 20ms ای بین این توابع استفاده کنیم.

مثال: برنامه ای بنویسید که با یک شستی و یک سون سگمنت ولتاژ ، فرکانس و ساعت کاری پالس ورودی را اندازه بگیرد و جدا جدا بوسیله شستی کنترلی آنها را نشان دهد(فرکانس روی , RC0=500 RC2=50 هرتز میباشند) :




```
#include "seg.h"
float a=0;int fr=0,cc=5;
```

```
unsigned char
bb=3,l=2,nn,mm=0,min_l=0,min_h=0,hr_l=0,hr_h=0,hr=0,aa=0,ab=0,ac=0,i=0,fr_l=0,fr_h=0,m=0,n=0;

void
main(){TRISC.F0=1;TRISC.F1=1;TRISC.F2=1;TRISB.F0=1;TRISB.F1=0;TRISB.F2=0;TRISB.F3=0;
TRISA.F4=0;TRISD=0;TRISE.F2=1;INTCON=0B10000000;PORTD=0XFF;PORTA.F4=1;PORTB.f1=1;
PORTB.F2=1;PORTB.F3=1;TMR1L=0;TMR1H=0;T1CON=0B00001111;CCP1CON=0B00000101;
PIE1.CCP1IE=1;CCPR1L=0;CCPR1H=0;ADCON0=0B11111001;ADCON1=0B10000000;ADRESH=0;ADRESL=0;

  while(1){a=Adc_Read(7)/(204.6);aa=a;ab=(a*10-aa*10);ac=(a*100-ab*100);

if(INTCON.INTF==1){INTCON.INTF=0;m=m++;delay_ms(300);if(m==7){m=0;}}
if(PIR1.CCP1IF==1){PIR1.CCP1IF=0;TMR1L=0;TMR1H=0;fr=500/(((CCPR1H&0X00FF)<<8)|(CCPR1L&0X00FF));fr_l=fr%10;fr_h=fr/10;
hr++;if(hr==5){nn=Eeprom_Read(0);delay_ms(20);if(nn=0){Eeprom_Write(0,1);delay_ms(20);Eeprom_Write(1,0);delay_ms(20);Eeprom_Write(2,0);delay_ms(20);
Eeprom_Write(3,0);delay_ms(20);Eeprom_Write(4,0);delay_ms(20);}if(mm==0){mm=1;min_l=Eeprom_Read(1);delay_ms(20);min_h=Eeprom_Read(2);delay_ms(20);

hr_l=Eeprom_Read(3);delay_ms(20);hr_h=Eeprom_Read(4);delay_ms(20);}
hr=0;min_l++;Eeprom_Write(1,min_l);delay_ms(20);if(min_l==10){min_l=0;min_h++;
```

```
Eeprom_Write(2,min_h);delay_ms(20);if(min_h==6){min_h=0;hr_l++;Eepro
m_Write(3,hr_l);delay_ms(20);if(hr_l==10){hr_l=0;hr_h++;Eeprom_Write(4,
hr_h);

delay_ms(20);if(hr_h==10){hr_h=0;Eeprom_Write(4,hr_h);delay_ms(20);}}}
}}}

if(m==3)/*Vdc*/{seg(aa);PORTB.F2=0;delay_us(400);PORTD=0B10000000;delay_us(400);PORT
B.F2=1;PORTD.F7=0;PORTB.F3=0;seg(ab);delay_us(400);PORTB.F3=1;seg(ac);PORTA.F4=0;dela
y_us(400);PORTA.F4=1;}
if(m==1)/*frequensy*/{seg(fr_l);PORTA.F4=0;delay_us(400);PORTA.F4=1;se
g(fr_h);PORTB.F3=0;delay_us(400);PORTB.F3=1;}
if(m==0)/*hour's*/{PORTA.F4=0;seg(min_l);delay_us(400);PORTA.F4=1;PORTB.F3=0;seg
(min_h);delay_us(400);PORTB.F3=1;PORTB.f2=0;seg(hr_l);delay_us(400);PORTD=0B10
000000;delay_us(400);
PORTB.F2=1;PORTB.f1=0;seg(hr_h);delay_us(400);PORTB.F1=1;}}}
```

فایل Seg.hقبلا در یک تکست نوشته شده و با این نام در فایلی که برنامه در آن است ذخیره شده.

کتابخانه ای

داخل آن برنامه زیر نوشته شده:

```
void seg(unsigned char a){
  switch(a){case 0:PORTD=0B00111111;break;case
1:PORTD=0B00000110;break;
  case 2:PORTD=0B01011011;break;case 3:PORTD=0B01001111;break;
  case 4:PORTD=0B01100110;break;case 5:PORTD=0B01101101;break;
  case 6:PORTD=0B01111101;break;case 7:PORTD=0B00000111;break;
  case 8:PORTD=0B01111111;break;case 9:PORTD=0B01101111;break;}}
```

خطاهای برنامه نویسی:

Demo limit: خطاییست که در صورت کرک نشدن نرم افزار میکروسی پس از زیاد شدن دستورات از حد معینی داده میشود.

Routin too large: در صورتی نرم افزار میکروسی این خطا را میدهد که تعداد دستورات از حد معینی گذشته و لایسنز آن مشکل داشته باشد.

Stack over flue: سر ریز پشته