İçindekiler

2. CCS PIC C Derleyici Programının Özellikleri

CCS PIC mikrodenetleyicileri için kullanılan bir C derleyicisidir. Program yazım editöründe C dilinde yazılan komutlar derlendiğinde HEX dosyanın yanında LST ve ERR dosyaları da üretilir. LST Dosya içerisinde C kodlarıyla birlikte ASM komutları da vardır. Bu dosya içerisine bakarak hangi komutla hangi REGISTER e müdahale edildiğini görebiliriz. ERR Dosyası içerisinde ise eğer C kodlarının yazımı esnasında bir hata oluşmuş ise bu dosya içerisinde hata görülebilir. C kodlarında hata düzeltilir ise bu dosyanın içeriği kendiliğinden değişir ve hata olmadığını yazar.

Derleyici ile 12 bit, 14 bit ve 18 serisi mikrodenetleyiciler için program yazılabilmektedir.

- **CCS PIC C Programının Kurulması**

Programın kurulması için EXE kurulum programı ve REG dosyaları (lisans dosyaları) olması gerekmektedir. Lisans dosyalarının program kurulum dosyası ile aynı klasörde olması gerekmektedir.

TERMINOLOJİ

Microcontroller: Yazılımsız hiç bir işe yaramayan plastik, metal ve saflaştırılmış kum yığını. Fakat içine yazılım girince hemen hemen sınırsız bir uygulama alanı var demektir.

I/O Giriş veya çıkış olarak yapılandırıla bilen ve dış dünyaya çıkış sağlayan bir bağlantı ucu. I/O microcontroller in bilgiyi okuması, kontrol etmesi ve iletişim kurabilmesi için gereklidir.

Software: Microcontroller in çalışa bilmesi için gereken bilgi dir.

Çıktı veya uygulamanın başarılı olabilmesi için Software (yazılım) hata ve kirlilikten uzak olmalıdır. Software C, Pascal veya Assembler gibi çeşitli dillerde yazılabilir.

Hardware Microcontroller, hafıza, arayüz, güç kaynağı, Sinyal koşullayıcı devreler ile Çalışması ve dış dünya ile alışveriş yapabilmesi için kendisine bağlanan diger bütün eleman parçalarıdır. Bir başka bakış açısıda çalışmadığı zaman Hardware in pencereden dışarı Şutlanmasıdır.

Simulator: MPLAB® gelişim çevresinin içine yerleştirilmiş. Kendi Simulatorü Microcontroller in iç çalışmasının bir kısmına giriş için izin verir. Olayların ne zaman olacağını biliyorsanız Simulator tasarımınızı test etmenin iyi bir yoludur.
Programın bir yerlerinde istenmeyen bir olay, tıkanma, takıntı meydana geliyorsa, Simulator ün uyardığını görürsünüz. Simulator ün full trace, step ve debug imkanları vardır.

SIM ICE (SIM In Circuit Emulator) ise 16C5x gelişimi için diğer bir üründür. ICE özelliklerinin bir kısmını veren bir hardware Simulator dür fakat bir başka özelliğide fiyatıdır.

In Circuit Emulator

(ICEPIC veya PICmicrocontroller®MCU MASTER) Microcontrollerin bulunduğu yerdeki soket ile PC niz arasına bağlanan çok faydalı bir ekipman parçasıdır. PC de yazılımın çalışmasını sağlar fakat devre kitinin ucunda bir Microcontroller gibi gözükür. ICE programın içine girmenize, microcontroller içinde hangi olayların meydana geldiğini ve dış dünya ile nasıl haberleştiğini izlemenize izin verir.

Programlayıcı

ICE nin yardımı dokunmadan microcontrollerin çalışmasını sağlayan hafıza bölümüne programı yükleyen birimdir. Farklı fiyat ve çeşitli şekil ve büyüklükde piyasaya çıkarlar.
Hem PICSTART PLUS hem de PROMATE II seri port a bağlanır.

Source File (Kaynak Program)

Assemblerin ve sizin anlayacağınız dilde yazılan program.
Microcontrollera yüklenmeden önce kaynak dosya Microcontrollerin anlayabileceği bir şekilde işlenmelidir.

Assembler / Compiler (Derleyici / Yorumlayıcı)

Kaynak dosyayı Object dosyaya çeviren bir yazılım paketidir. Derleme (Assembly) işlemi esnasında hatalar işaretlendikce programı hatalardan temizleyen (debugging) ve yoğun olarak kullanılan hata kontrolu Derleyicinin (Assemblerin) içinde yapılandırılmışdır. Microchip in çıkardığı MPASM bütün PIC lerin kullandığı en son derleyici (assembler) programıdır.

Object File (Hedef Dosya)

Bu dosya derleyici / yorumlayıcı tarafından üretilir ve görevini yerine getirmesini sağlamak üzere programlayıcı, simulator veya ICE nin algılayacağı bir biçimdedir. Dosyanın uzantısı assembler in (derleyicinin) verdiği direktife göre *.obj veya *.hex şeklindedir.

List Dosyası

Bu dosya Derleyici tarafından üretilir ve yantarafında yazdığınız komut ve heksodesimal değerleri ile birlikde Kaynak dosyadan gelen tüm talimatları içerir. Yazılım içinde ne olup bittiğini izleme şansınız kaynak dosya ya nazaran daha fazla olduğu için program hatalarınızı temizlemeye çalışırken en fazla yararlanacağınız dosyadır. Dosya uzantısı *.LST dir.

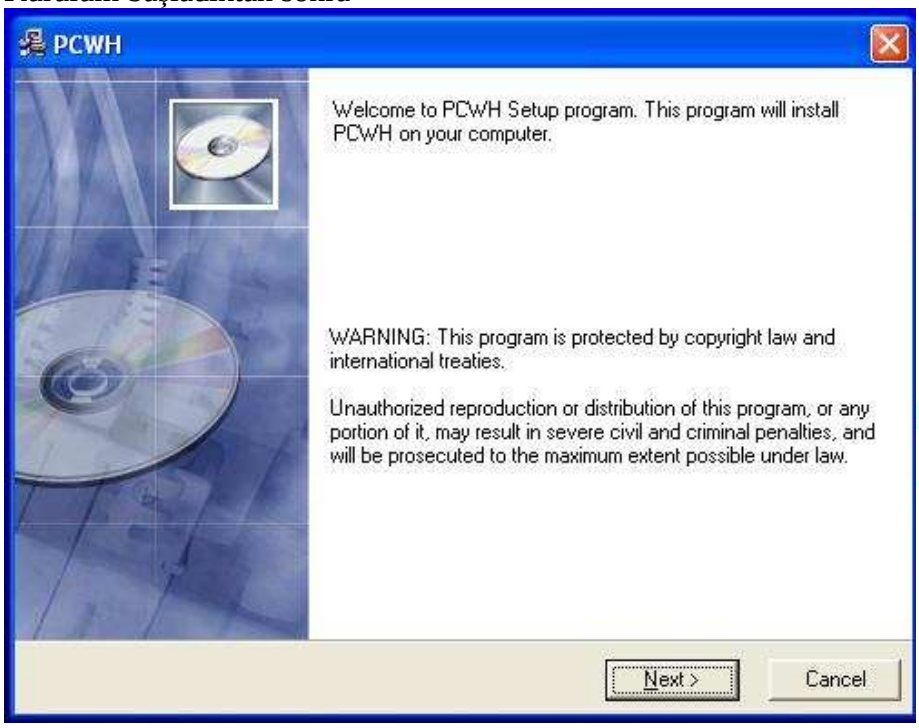
Diğer Dosyalar

Hata dosyası (*.ERR) hataları listeler fakat orjinlerini göstermezler. - .COD dosyası emulatör tarafından kullanılır.

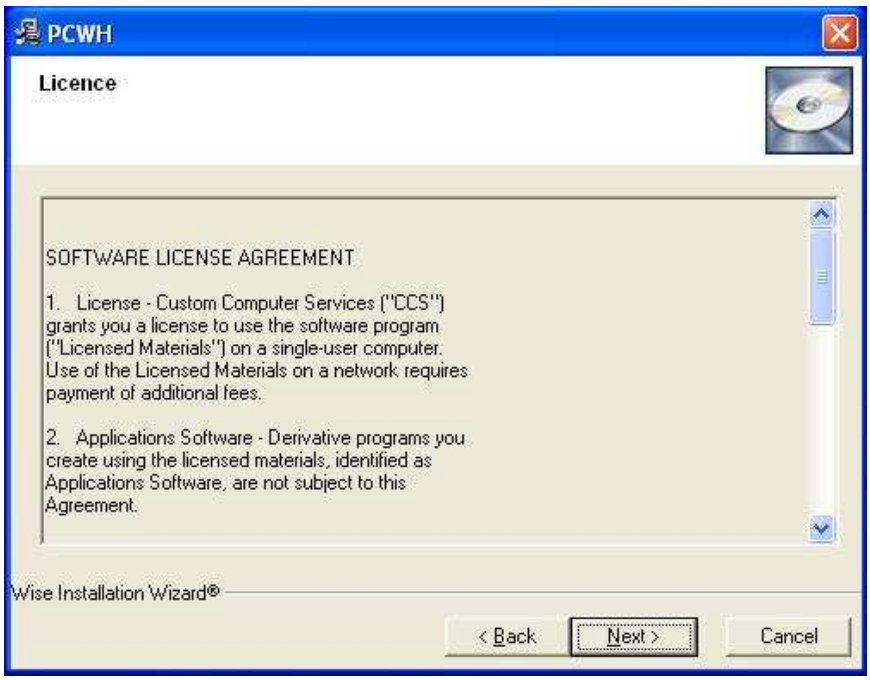
Bugs (Hatalar)

Programı yazanın yaptığı imla hataları veya daha önce tanıtımı yapılmayan, eksik verilen bilgilerdir. Bu hataların çoğu yorumlayıcı tarafından bulunur ve *.LST dosyasında gösterilir. Diğer hatalar dene yanıl metodu ile bulunup düzeltilmelidir.

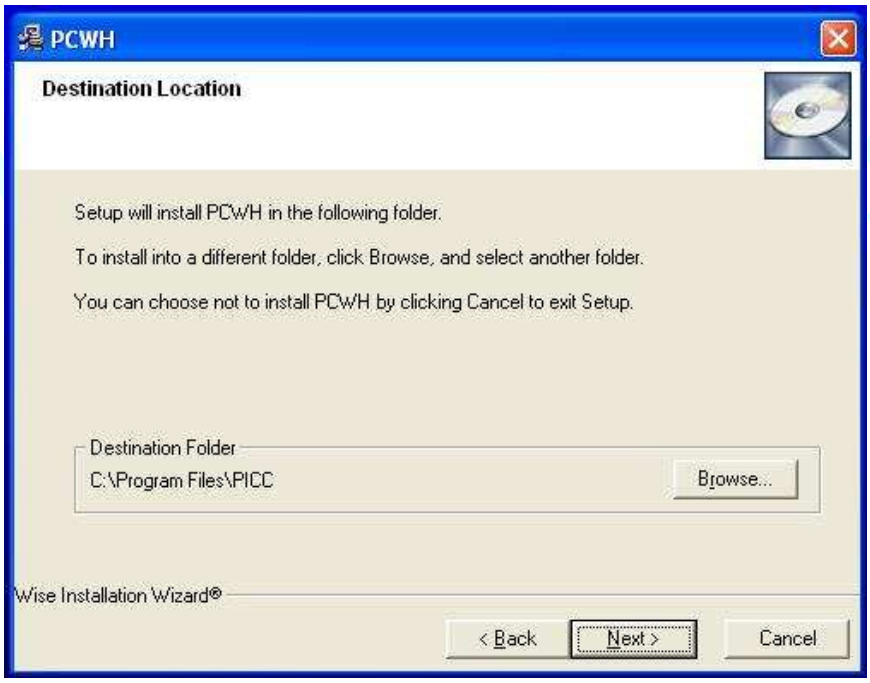
Kurulum başladıktan sonra



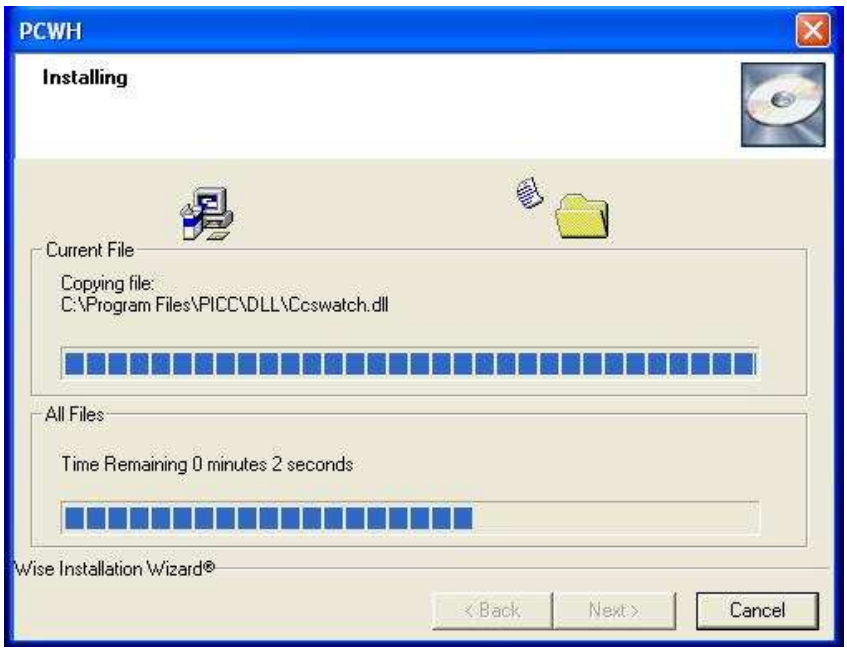
Next tıklanarak kuruluma devam edilmelidir.



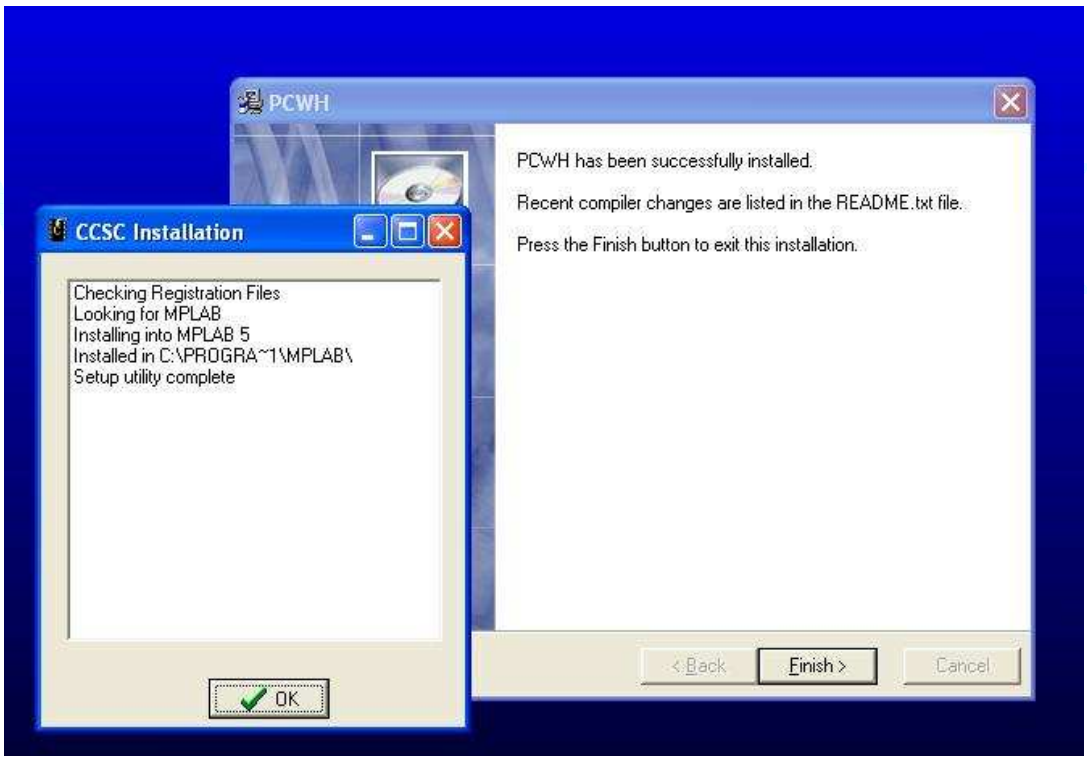
Burada programın lisanslı bir program olduğundan bahsedilmektedir.



Programın kurulacağı yer (hard disk alanı) seçilmelidir.



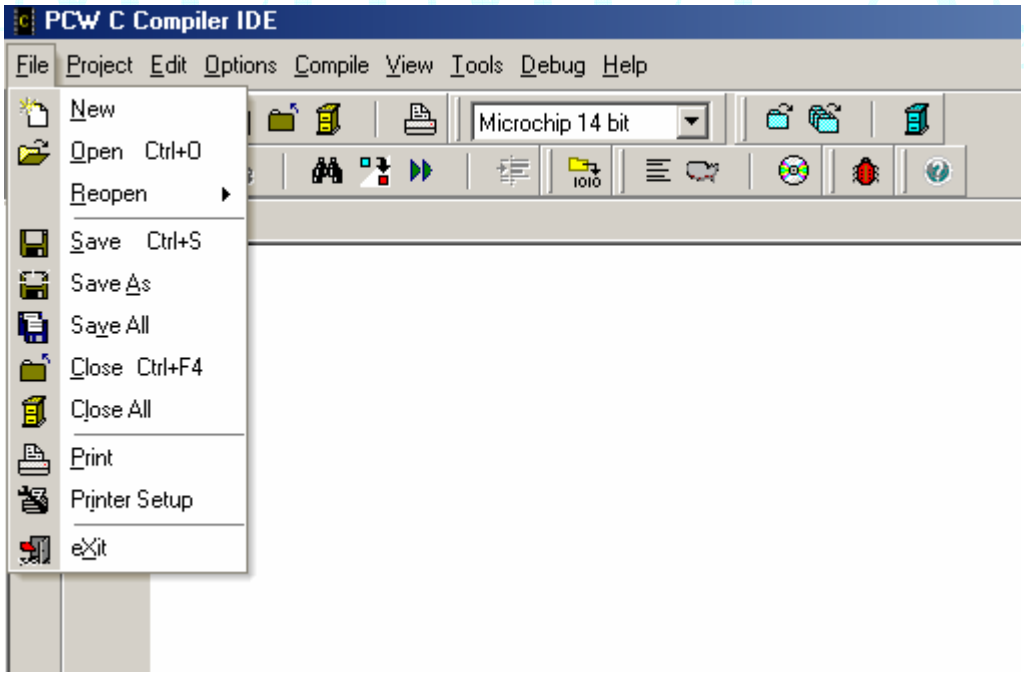
Dosyaların hard diske kopyalanmasına devam ediliyor.



Son olarak ta program bilgisayarda MPLAB programının kurulu olup olmadığını kontrol eder ve CCS PIC C programı kurulmuş olur.

PIC C PROGRAMI MENÜLERİ

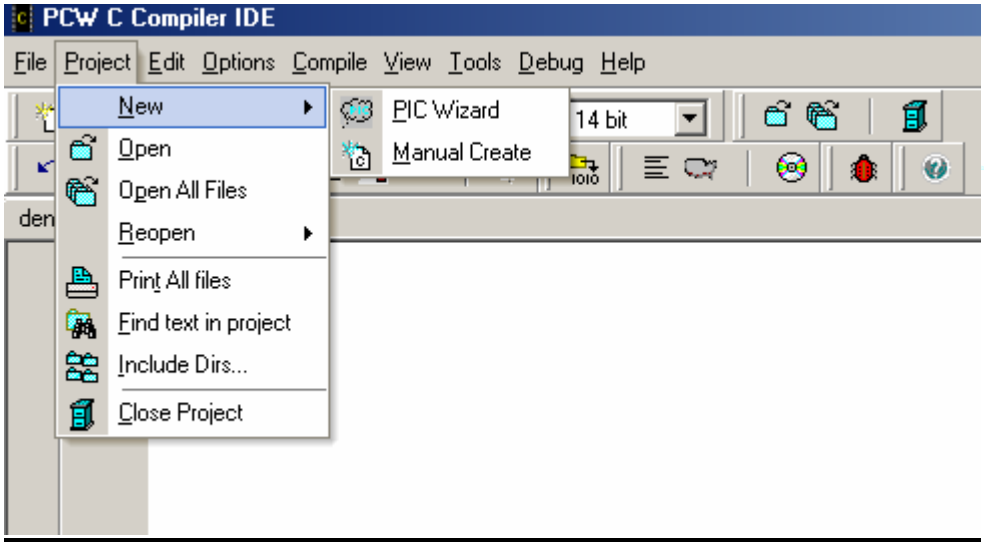
File Menu



New	Yeni dosya oluşturmak için kullanılır.
Open	Önceden oluşturulan dosyaları açmak için kullanılır.

Reopen	Daha önceden açılmış dosyalardan bir tanesini açmak için kullanılır.
Save	Yazılan programı kaydetmek için kullanılır.
Save As	Dosyayı farklı kaydetmek için kullanılır.
Save All	Açık olan dosyaların tamamını kaydetmek için kullanılır.
Close	Aktif olan pencereyi kapatmak için kullanılır.
Close All	Açık olan tüm pencereleri kapatır.
Print	Aktif sayfayı yazdırmak için kullanılır.
Printer Setup	Yazıcı ayarlarını yapmak için kullanılır.
Exit	Programdan çıkmak için kullanılır.

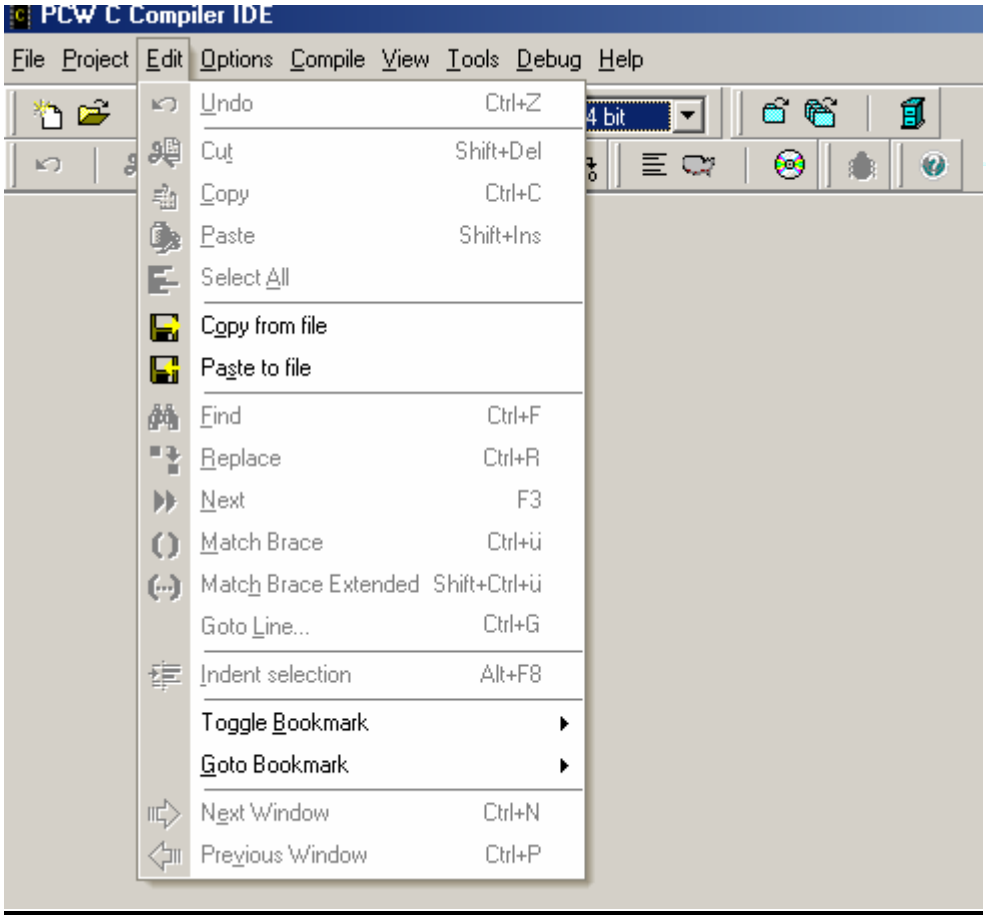
Project Menu



New	Yeni bir proje oluşturur.
New| PICWIZARD	Proje manuel (elle) yada proje sihirbazı (wizard) ile yapılır. Sihirbaz ile kullanıcı özel parametreler eklediğinde otomatik olarak header dosyaları (*. h) eklenir. Standart program kodları oluşturulur RS232, I/O 12C, timer seçenekleri, interrupt kurulumu A/D seçenekleri yapıldığında header dosyalar ve standart kodlar oluşturulur.
Open	A .PJT dosyası açılır. Derleme ve debug işlemi yapılabilir.

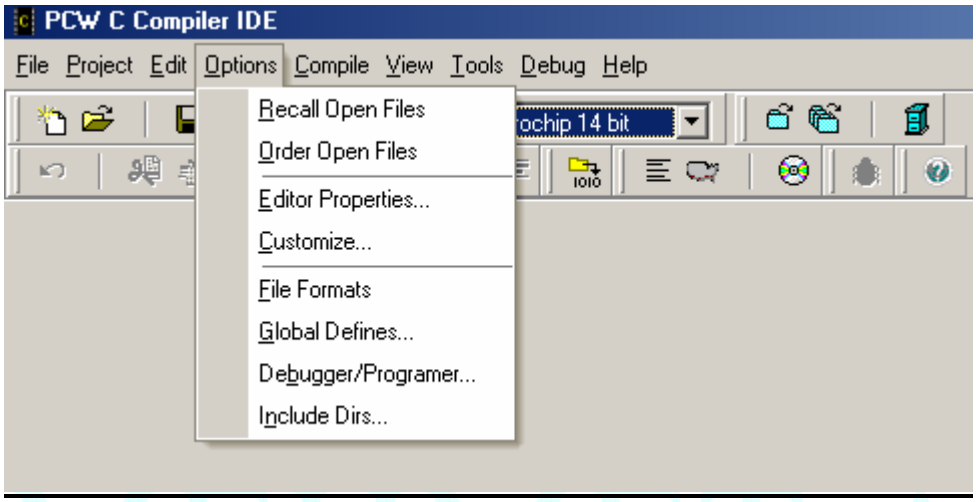
Open All Files	A.PJT dosyasına ait alt programların dosyalarını açar.
Reopen	Açılmış projelerden birini tekrar açmak için kullanılır.
Find Text In Project	Proje dosyası içerisinde bir kelime aramak için kullanılır.
Print All Files	Tüm sayfaları yazdırmak için kullanılır.
Include Dirs	Alt program klasörlerinin yarini belirtmek
Close Project	Tüm projeleri kapatır.

Edit Menu

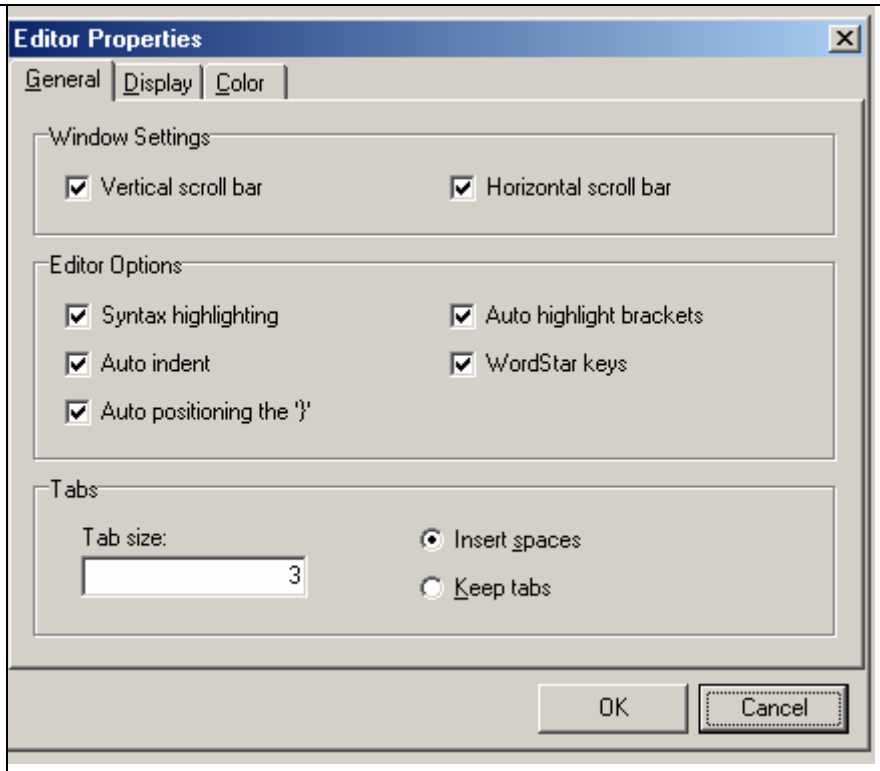


Undo	Geri al
Cut	Kes
Copy	Kopyala
Paste	Yapıştır.
Copy from File	Dosyadan Bulunulan dosyaya kopyalama
Paste to File	Seçilen kısım yeni bir dosyaya yapıştırır.
Find	Kelime bulmak için kullanılır.
Replace	Son kelime bulma işlemini tekrarlar.
Next	Sonraki kelimeleri bulur.
Find matching braces	Parantez başına ya da sonuna gider.
matching braces extended	{ veya). Parantezini işaretler.
Toggle Bookmark	Kursörün bulunduğu yere 0-9 arası işaret rakam verir.
Goto Bookmark	İşaret rakamı verilen yere gider.
Next Window	Birden fazla program penceresi açıksa sonraki pencereye geçmek için kullanılır.
Previous Window	Birden fazla program penceresi açıksa önceki pencereye geçmek için kullanılır
Indent Selection	Seçili alanın tanımlama ya da komut grubuna göre pragraf düzenlemesi yapar.

Options Menu



Recall Open Files	İşaretlendiği zaman program kapatılıp açıldığında en son çalışılan programlarla beraber açılır.
Editor Properties	Editör özellikleri penceresi.
General Tab:	**Window Settings:**



Pencere ayarları

Editor Options:

Komutların renkli görünmesini sağlar.

Auto Highlight braces

Kursörün bulunduğu parantezin renkli görünmesini sağlar.

Auto Indent

Otomatik paragraf düzenlemesi yapar.

WordStar keys

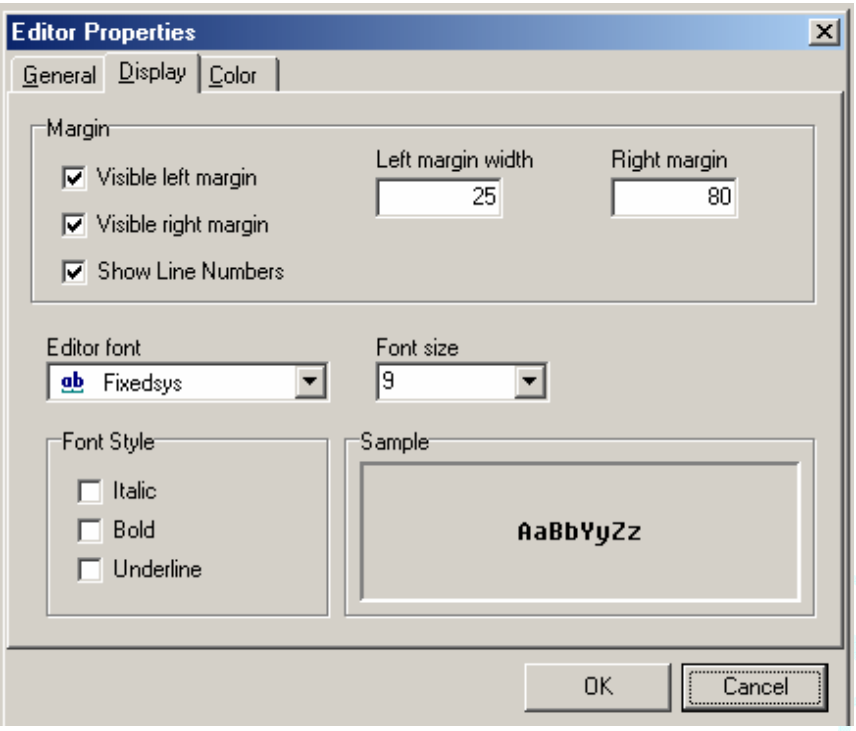
TABS:

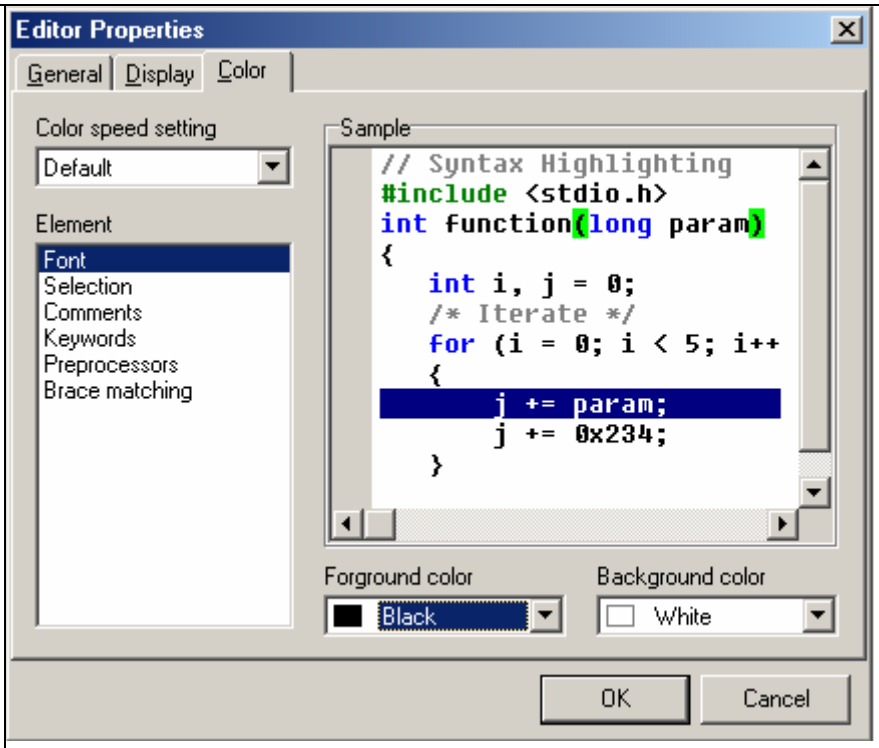
Tab size

Tab tuşuna basıldığında kaç karakter atlama yapılacağını gösterir.

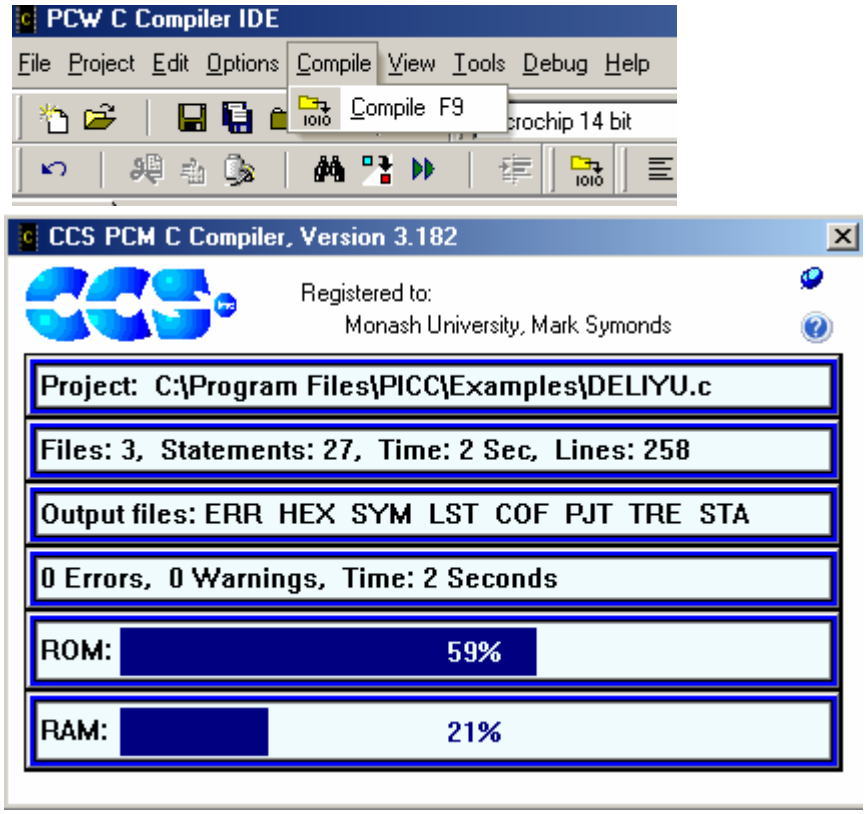
Keep Tabs

Tab konumunun korunması.

	Insert Spaces Sonraki Tab pozisyonuna geçer.
Display Tab:	**Magrin**  Visible left Margin Sol marjinin görülmesi. Visible Right Margin Sağ marjinin görülmesi. Editor Font Editör yazı fontu seçilir. Font Size: Yazı büyüklüğü seçilir. Font Style Yazı sitili seçilir. (Italic/Bold/Underline)

Color	 Editör renk ayarları yapılır.
File Formats	Compiler ve program dosya tipleri seçilir.
Include Dirs	INC dosyalarının klasörleri yeri belirlenir.
Debugger /Programmer	Debugger (hata ayıklama) ve programlama işlemi için firmanın programlayıcı devresi bilgisayara bağlı olmalıdır.
Global Definitions	Standart # tanımlamalarının ne anlama geldiği yazılır.

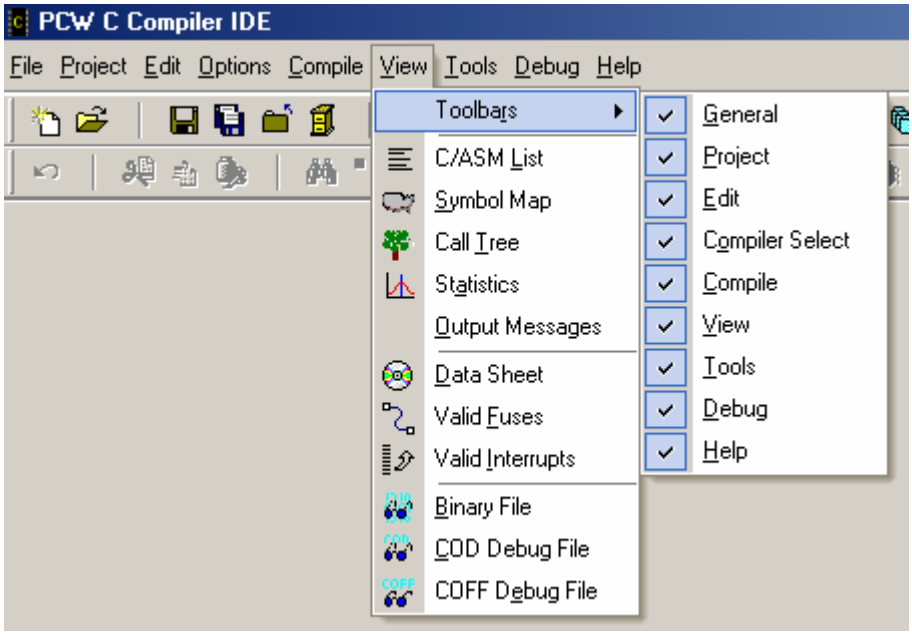
Compile



PCW Compile

Derleme işlemi yapılır. Derleme sonucunda ekranda programının bulunduğu klasör, oluşturulan kodlar, kullanılan dosya sayısı, hatalar, uyarılar, derleme süresi, ROM ve RAM belleklerde kapladığı yer gösteren bir uyarı penceresi açılır.

View Menu

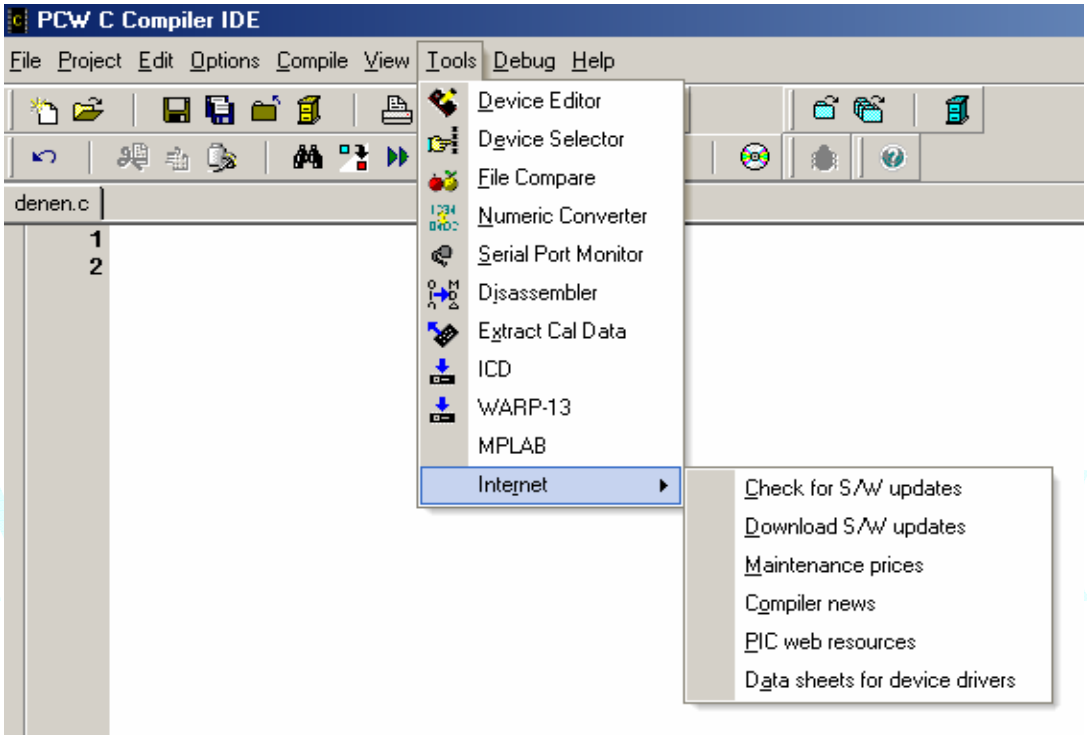


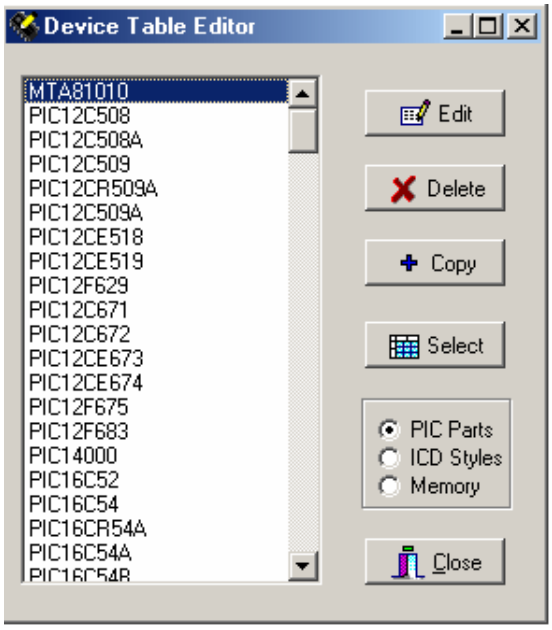
C/ASM	Programın ASM kodlarını gösterir. (C kodları ile birlikte) Örnek: ..... for(i=0;i<num_ms;i++) 0119: CLRF 31 011A: MOVF 2E,W 011B: SUBWF 31,W 011C: BTFSC 03.0 011D: GOTO 123 ..... delay_ms(250); 011E: MOVLW FA 011F: MOVWF 32 0120: CALL 0F6 0121: INCF 31,F 0122: GOTO 11A
Symbol Map	

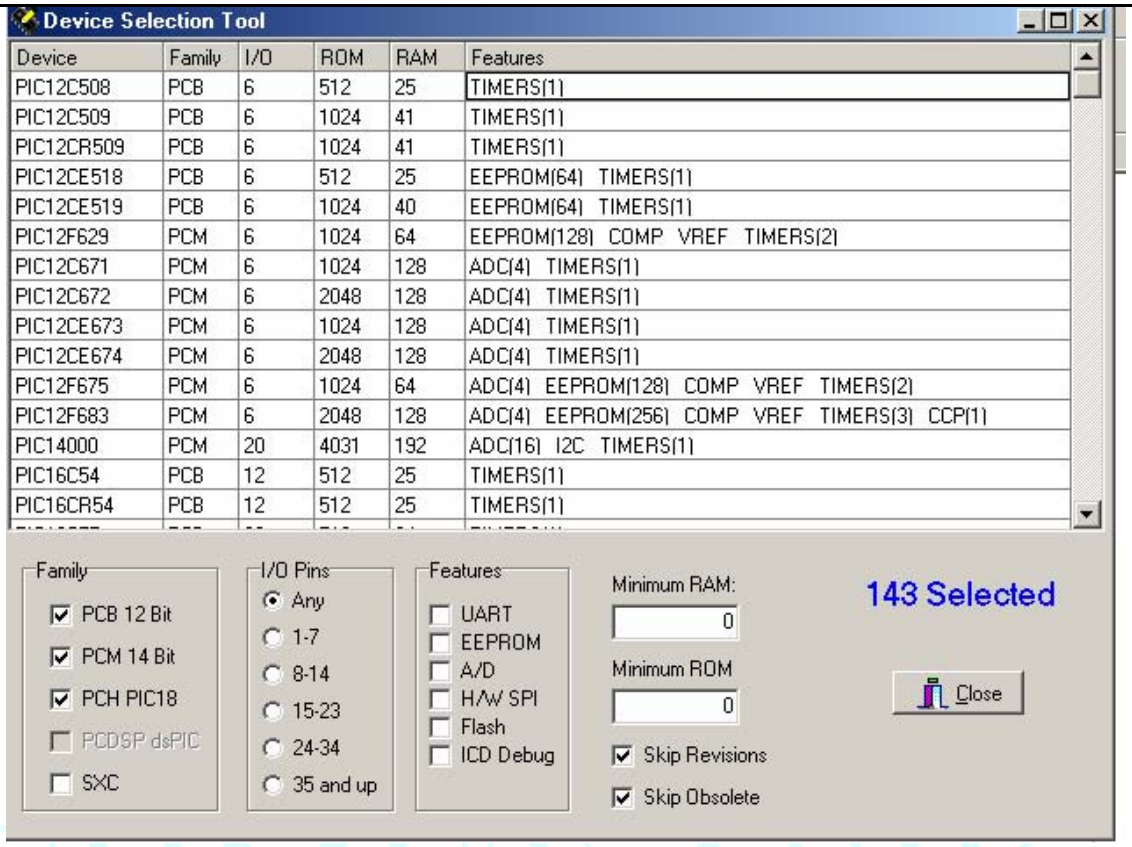
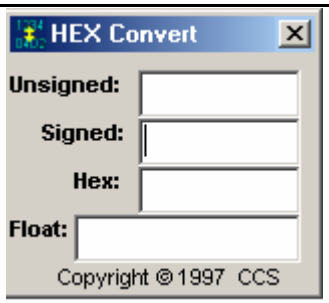
	Kullanılan yazmaçları ve kaydedilen adresleri gösterir. 00C @SCRATCH 00D @SCRATCH 00D _RETURN_ 00E @SCRATCH 00F @SCRATCH 010 @SCRATCH 027 generate_tone.num_us_delays 028 generate_tone.num_ms_delays 029 generate_tone.ms_delay_time
Call Tree	Program içerisindeki bağlantıları gösterir. Main 0/30 INIT 0/6 WAIT_FOR_HOST 0/23 (Inline) DELAY_US 0/12 SEND_DATA 0/65
Statistics	Program istatistiklerini gösterir. ROM used: 597 (58%) 427 (42%) including unused fragments RAM used: 14 (21%) at main() level 54 (79%) worst case Lines Stmts % Files 64 6 22 C:\Program Files\PICC\Examples\ARCEL.c 111 0 0 C:\PROGRAM FILES\PICC\devices\16F84.h 82 21 64 C:\PROGRAM FILES\PICC\drivers\tones.c
Data Sheet	Kullanılan mikrodenetleyici ile ilgili bilgi dosyası (data sheet) varsa onu

	açar.
Binary file	Dosyanın varsa binary kodlarını gösterir.
COD Debug file	Oluşturulan kodları gösterir.
Valid Fuses	#fuses direktifi ile kullanılabilecek sigortaları gösterir.
Valid Interrupts	#int_xxxx direktifi ile kesmeleri aktif/pasif yapar.

Tools Menu

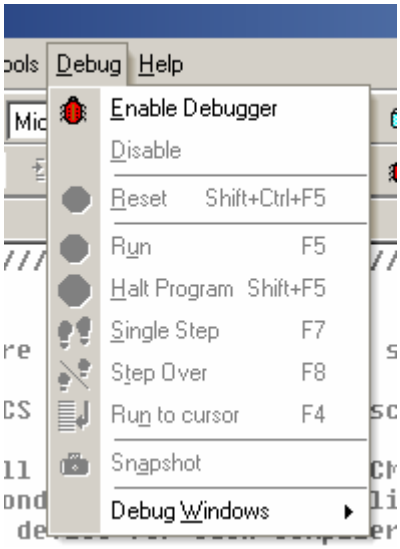


Device Editor	 Mikrodenetleyici seçim editörüdür.

Device Selector	 İstenilen özellikler belirlenir ve belirlenen ihtiyaçlar çerçevesinde kriterlere uyan mikrodenetleyiciyi gösterir.
File Compare	İstenilen iki dosyayı karşılaştırı.
Numeric Converter	 decimal, hex ve float ifadeleri birbirine dönüştürür.
Serial Port Monitor	Bilgisayarın seri portuna bilgi gönderme ya da almak için kullanılan pencere.
Disassembler	HEX kodunu ASM koduna dönüştürme işlemi yapar
Extract Cal Data	Harici veri girişi için ayarlama yapılır.
Program Chip	Yazılan ve derlenen program ccs firmasının programlayıcısı ile mikrodenetleyiciye yüklenir.

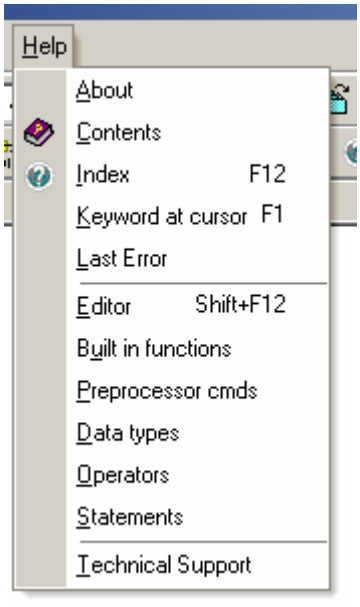
MPLAB	MPLAB programına geçiş yapar.
Internet	İnternete bağlanarak bilgi almak içindir.

Debug Menu



Hata ayıklama ve adım adım programın çalıştırılması için kullanılır. Bilgisayara programlayıcı takılı değilse çalışmaz.

Help Menu



About	Program hakkında bilgi verir.
Contents	Yardım dosyası içindekiler menüsü açılır.
Index	İndeks açılır.
Keyword at cursor	İstenilen yardım kelimesi yazılarak konu hakkında bilgi alınır.

Sabitler

123	Desimal
0123	Oktal
0x123	Heksadesimal
0b010010	Binary
'x'	karakter
'\010'	Oktal karakter
'\xA5	Hex karakter
'\c'	Özel karakter \n sonraki satıra geç \r satır başına git \t tab \b önceki karakteri siler \f başlangıç noktasına gider. \a uyarı sesi çıkarır \v tab ölçüsünde dikay boşluk brıtakır.

	\? Soru işareti yazdırır \' tek tırnak yazdırır \" çift tırnak yazdırır. \\ \ işareti yazdırır.
"abcdef"	String (karakter yazdırmak için kullanılır.

Operatörler

+	Toplama işlemi yapar
+=	Arttırma ve atama operatörü, x+=y >>>>> x=x+y
&=	Mantıksal işlem operatörü, x&=y, >>>>>>>> x=x&y
&	Adres operatörü
^=	Üs alma operatörü x^=y, >>>>>>>>>> x=x^y
?:	Koşul operatörü
- -	Azaltma operatörü
/=	Bölme işlemi, x\=y, x=x/y
/	Bölme operatörü
==	Eşitse operatörü
>	Büyüktür operatörü
>=	Büyük eşit operatörü
++	Arttırma operatörü
*	Çarpım operatörü
!=	Eşit değil operatörü
<<=	Sola kaydır ve eşitle , x<<=y x=x<<y
<	Küçüktür operatörü

<<	Sola kaydır
<=	Küçük eşit operatörü
&&	AND operatörü
!	NOT operatörü
ll	OR operatörü
%	Modül alma operatörü
>>=	Sağa kaydır eşitle , x>>=y x=x>>y
>>	Sağa kaydır
-=	Çıkar ve eşitle
-	Çıkarma işlemi

Veri Tipleri:

int1	1 Bitlik işaretsiz tam sayılar
int8	8 Bitlik işaretsiz tam sayılar
int16	16 Bitlik işaretsiz tam sayılar
int32	32 Bitlik işaretsiz tam sayılar
Char	8 Bit karakter tipi veriler.
Float	32 Bit ondalıklı sayılar
Short	1 Bitlik işaretsiz tam sayılar
Long	16 Bitlik işaretsiz tam sayılar
Unsigned	İşaretsiz pozitif sayıları tanımlamada kullanılır.
Signed	Pozitif veya negatif sayı tanımlamalarında kullanılır.

Tip	**Bit Genişliği**	**Alabileceği Değer Aralığı**
short	1	0 veya 1
short int	1	0 veya 1
int	8	0 - 255
char	8	0 - 255

unsigned	8	0 -255
unsigned int	8	0 -255
signed	8	-128 ile 127 arası
signed int	8	-128 ile 127 arası
long	16	0 - 65536
long int	16	0 - 65536
signed long	16	-32768 ile 32767 arası
float	32	3.4E-38 to 3.4E+38

C Programlama Dilindeki Bazı Özellikler :

1- C dilinde // işareti görüldüğü andan itibaren yazılanlar programa dahil edilmez yani açıklama satırı olarak algılanır.

/* açıklamalar */ /* */ İşaretleri arasında kalan yazılanlar açıklama olarak kabul edilir.

2- Komut satırları sonuna mutlaka ';' (noktalı virgül) konulur. Satır sonunda noktalı virgül kullanılmayan yerler etiket isimleri, for komutu sonu, while komutu sonu, if komutu sonudur.

3- Komutlar küçük harflerle yazılır. Büyük harflerle yazılmak istenirse Türkçe karakter kullanılmamalıdır.(Ör. i harfi I şeklinde yazılmalıdır.)

4- Etiket isimlerinde büyük küçük ayrımı yapılacağından kullanılan her yerde aynı olmalıdır. Büyük ise diğer satırlarda da büyük, küçük ise diğer satırlarda da küçük olmalıdır.

FONKSİYON YAZIMI:

Programların karmaşıklıktan kurtarılması ve aynı işlemlerin birden fazla yerde yapılması gerekiyorsa fonksiyonlar kullanılır. Fonksiyonlar ana program içerisinde olmazlar. Ayrıca fonksiyon içerisinde başka bir fonksiyon çalıştırılabilir fakat çalıştırılmak istenen fonksiyon daha önce yani programın üstünde yazılmalıdır.

```
void  zaman( )
{
delay_ms(200);
output_b(100);

buton( ); // kullanılamaz.
}

void buton( )
{

zaman( );

output_a(2);
```

}

Burada zaman ve buton isimli iki ayrı fonksiyon tanımlanmıştır. buton fonksiyonu içerisinde zaman fonksiyonu çalıştırılabilir fakat zaman fonksiyonu içerisinde buton fonksiyonu çalıştırılamaz. Bunun nedeni ise buton fonksiyonu daha sonra yazıldığından zaman fonksiyonu içerisinde tanımlanamamaktadır.

Dolayısıyla fonksiyonları ana programda kullanmak için void main( ) satırının daha üstüne yazmamız gerekir.

Kontrol ve Döngüler:

if Komutu : Kelime karşılığı "eğer"dir. C Dilinde if deyimini kullanmak için şu yol izlenir.

Önce if deyimine bir ifade verilir. Bu ifade değerlendirilir ve işlemin sonucu doğru ise if deyiminden sonra gelen satır çalıştırılır. Doğru değilse çalıştırılmaz.

Kullanımı:

if(şart)

Koşul doğru olduğunda çalıştırılacak satır;

Örnek:

```
if(a<2)

goto tek;
```

else eklenerek kullanmak:

if komutunda şart sağlanırsa bir ifade çalıştırılır. Şart sağlanmazsa çalıştırılmaz. Fakat else kullanarak şartın sağlanmadığı durumlarda yapılması gereken işlem belirtilir.

Kullanımı:

```
if(şart)
Koşul1;
else
Koşul2;
```

Örnek:

```
if (x==25)
   x=1;
else
   x=x+1;
```


örnek program:

```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
void main()
{
tek:
if(input(PIN_A0))
output_b(255);
else
output_b(1);
goto tek;
```

}

Bu örnekte A portunun 0. bitine bağlı olan buton 1 olmuşsa B portunun tüm uçları 1 yapılır. Buton 1 olmamışsa B portunun 0. biti 1 olmaktadır.

While Komutu : Belirtilen şart geçerli olduğu müddetçe işlem tekrarlanır.

Kullanımı:

While(şart)

{

işlemler;

}

Örnek:

while(x<200)

{

output_a (255);

}

x değişkeni 200 den küçük olduğu sürece parantez içerisindeki komut satırları tekrarlanır.

a=0;

while(a<4)

{

a++;

output_b (1);

}

yukarıdaki komut satırları B portuna 1 bilgisini 4 defa gönderir.

Do while komutu: while döngüsüne çok benzer ve aynı mantıkla çalışır. While komutunda koşul döngüye girmeden önce yapılıyordu. Yani önce koşul kontrol ediliyor, eğer doğru olarak değerlendirilir ise döngüye giriliyordu. do while döngüsünde ise önce döngüye girilir, döngü en az bir kere çalıştırılır, koşul kontrolü ise döngü sonunda yapılır.

Kullanımı:

Do

```
{

çalıştırılacak satırlar;

} while (şart);
```

Örnek:

```
do

{

a++;

output_b(255);

} while(a!=10);
```

Bu örnekte a sayısı bir arttırılarak b portuna gönderilir. Eğer a 10 olursa işleme alt satırlardan devam edilir. 10 değilse tekrar arttırılır.

Örnek program:

```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)

void main()

{

char a=0;

do

{

a=a+1;

delay_ms(250);

output_b(a);

} while(a!=3);
```

```
delay_ms(500);

output_b(255);

tek:

goto tek;

}
```

For Komutu : Belirlenen sınırlar içerisinde istenilen işlemleri yapmak için kullanılır.

```
x=0;

for(a=0; a<10;a=a+2)

{

x=x+1;

output_a (x);

}
```

yukarıdaki örnekte alt sınır 0 (a=0), üst sınır 10 (a<10) ve a nın her döngüde artış miktarıdır.(a=a+2)

0 dan 10 ye kadar her döngüde 2 arttırarak 5 defa B portuna her defasında x e bağlı olarak farklı bilgiler gönderir. 0, 1,2,3,4 bilgileri B portuna sırasıyla gönderilir.

Örnek program

```
#include <16F84.h>

#fuses XT,NOWDT,NOPROTECT

#use delay(clock=4000000)


void main( )

{

char a,x=0;

for(a=0; a<10;a++)

{

x=x+1;

output_b (x);
```

```
delay_ms(250);

}

}
```

switch deyimi:

if deyiminin kullanımına çok benzer. iç içe if deyimlerinin kullanılması gereken yerlerde switch deyimi kullanmak daha doğru olur. Fakat bu olay switch deyiminin if deyiminden daha üstün olduğunu göstermez. Her ikisinin de üstün olduğu kullanım alanları vardır.

Kullanımı:

```
Switch( koşul) {

case  sabit 1: çalıştırılacak satır;

        break;

case sabit 2:    çalıştırılacak satır;

        break;

}
```


Örnek: a değişkenine bağlı olarak a nın 0.1.2 durumlarda b portuna 0.1.2 bilgisini gönderen program.

```
#include <16F84.h>

#fuses XT,NOWDT,NOPROTECT

#use delay(clock=4000000)

void main() {

char a=0;

tek:

      switch (a)

     {

      case 0   : output_b(0);

                 delay_ms(250);

                 break;
```

```
        case 1   : output_b(1);
                   delay_ms(250);
                   break;
        case 2   : output_b(2);
                   delay_ms(250);
                   break;
      }
a=a+1;
  goto tek;
}
```

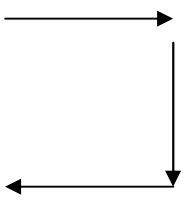

break ve continue deyimleri:

Tüm döngüler içerisinde kullanılan iki deyimdir.

Break deyimi; içinde kullanıldığı döngüyü sonlandırmak için kullanılır. Program içerisinde break komutu görüldüğüne döngü içerisindeki işlemler tamamlanmadan döngü sonuna gidilir.

Örnek:

```
Char a=0;
while (a<5)  {
output_b(255);
delay_ms(250);
break;
a=a+1;
}
```

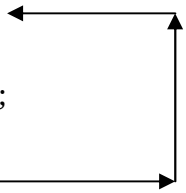


Örnekte ilk olarak a değişkeni kontrol edilir. a değeri 5 ten küçük olduğu için b portuna 255 gönderilir daha sonra 250ms beklenir ve işlem bitirilir. Break komutu olmasaydı bu işlem a nın 5 değerine ulaşmasına kadar devam edecekti.

Continue deyimi de benzer özellikler gösterir. Program içerisinde program akışının döngü başına dönmesini sağlar.

Örnek:

```
char  a=0;

while(a<5) {

output_b(255);

continue;

a++;

}
```



Örnekte a nın 5 ten küçük olduğu kontrol edilir. b portuna 255 gönderilir. Continue komutu görülünce döngü başına dönülür dolayısıyla b portuna sürekli 255 gönderilmiş olunur.

Sonsuz döngüler:

```
tek :

goto tek;
```

ya da

```
char a=2;

while(a>1) {

……

……

}
```

burada a nın yani 2 nin 1 den büyük olduğu durumlarda parantez içerisindeki işlemler gerçekleştirilecek yani sürekli devam edecek.

Diziler:

Tek Boyutlu Diziler:

DiziTipi DiziAdı[n]={dizi elemanları };

n değeri dizideki eleman sayısını gösterir. n yazılmadan da dizi oluşturulabilir.

char dizi[3]={63,6,91};

char dizi[ ]={63,6,91}; // Şeklinde de yazılabilir.

output_b(dizi[0]);// Yazılır ise B portuna dizinin 0. elemanı yani 63 bilgisi gönderilir.

char memo[]="Mehmet Aşık";// memo isimli yazı türünde bir dizi oluşturulmuştur. Dizi yazı türünde dizi olursa "Mehmet Aşık" tırnak içerisinde yazılmalıdır.

Const char dizi [5] = {2,332,12,43,55}; // dizi const şeklinde tanımlanırsa dizi RAM bellekte değil de program belleğinde saklanır.

Çok Boyutlu Diziler:

Char dizi[ ] [ ]=

```
char const key[4][3] = {{'1','2','3'},
                        {'4','5','6'},
                        {'7','8','9'},
                        {'*','0','#'}};
```

Örnekte 3 elemanlı 4 ayrı dizi olduğunu göstermektedir.
Kullanırken
output_b(key[2][2]); şeklinde kullanılabilir. Böylelikle b portuna 9 değeri gönderilmiş olunur.
output_b(key[0][0]);   ile B portuna 1 değeri gönderilir.

İŞARETÇİLER(POINTER):

Pointer tipinde değişkenin tanımlanmasında önce değişkenin ne tip bir değeri gösterdiğini tanımlamak gerekir. Bunun yanında pointer tipi değişkenler her zaman * işareti ile kullanılır:

veri tipi *d_adi;

Pointer işlemlerinde 2 ayrı operatör kullanılır. Bunlardan biri değişkenin bir adres bilgisi tuttuğunu gösteren ve o adresteki değeri veren *, diğeri de değişkenin hafızadaki adresini gösteren & operatörüdür. Örneğin;

m = &count;

kodu m değişkenine count'un adresini yazar. Bu işlemin count'un değeri ile hiç bir işi yoktur. Yaptığı tek şey count değişkeninin tutulduğu adresi alıp m değişkenine atamaktır. Bunun yanında;

n = *m;

kodu da m adresinde tutulan değişkenin değerini alır ve n'e atar. Bir önceki kod ile bu kod beraber düşünülürse birincisi count'un adresini m'e atıyor, ikincisi de m adresindeki değeri, yani count'un tutulduğu adresteki değeri, yani count'un değerini n'e atıyor.

Pointer türleri aynı ise matematiksel işlemler yapılabilir.

Örnek:

```
int x,y;
int p1=20;
x = &p1;  // p1 değişkeninin adres bilgisi x değişkenine yüklenir. ( adres 35 varsayalım)
y =*x;     // x değişkeninde tutulan adresin yani 35 nolu adresin içerisindeki bilgiyi y ye yükler.
```

Bu durumda x = 35 ve y = 20 (p1 değeri) olur.

Örnek:

```
int p1=10;
int p2= 20;

x = &p1+p1;
```

x değişkenine p1 ve p1 in tutulduğu adres değeri toplanır.

STRUCTURE:

Structure yapısı, birbiri ile ilişkili farklı veri tipindeki değişkenlerin tek bir yapı altında tutulması için kullanılır. Structure yapısı programcı tarafından tanımlanan bir api olduğundan, kullanılmadan önce içeriğinin tanımlanması gerekir. Structure'i oluşturan değişkenlere eleman adı verilir.

Bir structure'i oluşturan değerler genelde birbiriyle ilişkilidir. Structure tanımının genel yapısı aşağıdaki gibidir;

```
struct structure_adi

 {

 vtipi1 deg_ad1;

 vtipi2 deg_ad2;

 …

 vtipi3 deg_ad3;

 };
```

Structure tanımı mutlaka **;** ile bitmelidir. Bu tanım esnasında hiçbir değişken tanımlaması yapılmamıştır ama bu tanım yapıldıktan sonra structure_adı adı ile bir veri tipi oluşturulmuş olur ve bu structure tipi değişken tanımlamalarında kullanılabilir. Bu veri tipine sahip bir değişken yaratmak için ise;

struct structure_adi degisken_adi;

satırı gereklidir. Bu satir önceden tanımlanmış yapıda bir değişken tanımlar ve bu değişken tanımlı olan structure'in bütün elemanlarına sahiptir. Derleyici tarafından değişken tiplerinde tanımlama yapıldığında gerçekleşen işlemler burada da gerçekleşir. Derleyici tanımlanan değişkenin structure tipinin boyutu kadar hafıza bloğunu değişkene ayırır. Örneğin;

```
struct addr

  {

  char name[30];

  char street[40];

  char city[20];

  char state[3];

  unsigned long int zip;

  };
```

Yapısında bir structure tanımladıktan sonra

struct addr addr_var;

Bir structure tipinde değişken tanımlamak için, tanımlanacak değişkenlerin isimleri, struct tanımından hemen sonra da verilebilir. Örneğin;

```
 struct addr {

  char name[30];

  char street[40];

  char city[20];

  char state[3];

  unsigned long int zip;

  } addr_info, binfo, cinfo;
```

Satırları hem addr tipinde bir structure tanımlar hem de bu veri yapısına sahip addr_info, binfo, cinfo değişkenlerini tanımlar. Ayni şekilde istenilen yapıya sahip tek bir değişken tanımlanacaksa, struct'tan sonra kullanılan veri tipi ismine ihtiyaç yoktur - sadece istenilen değişkenin adi verilebilir;

```
struct {

  char name[30];

  char street[40];

  char city[20];
```

```
  char state[3];

  unsigned long int zip;

  } addr_info;
```

Structure elemanlarına erişim

Bir structure'da tanımlı olan her elemana . operatörü ile ulaşılır. Örneğin bir önceki örnekte tanımlanan addr_var değişkenindeki zip alanını değiştirmek istersek;

addr_var.zip = 12345;

İfadesini kullanırız. Elemanlar üzerindeki işlemler ise tamamen elemanın veri tipine bağlıdır. Normal bir tanımlama ile kullanılan değişkenler üzerinde yapılabilen bütün işlemler elemanlar üzerinde de yapılabilir. Örneğin yazdırma işlemi;

printf("%d", addr_info.zip);

Şeklinde yapılırken, name stringinin bütün elemanları da;

```
 for(t=0; addr_info.name[t]; ++t)

 putchar(addr_info.name[t]);
```

ifadesi ile yazdırılabilir.

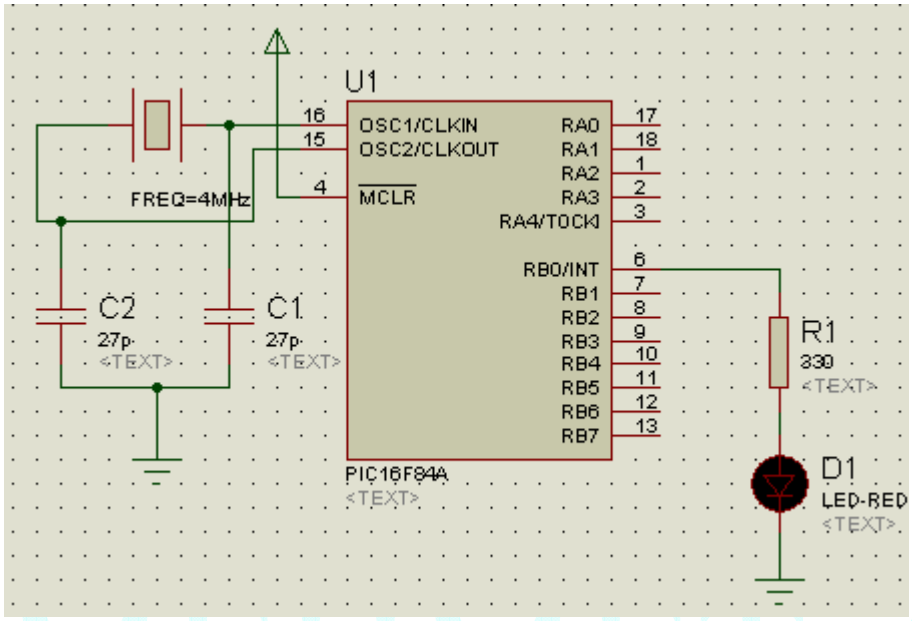
Örnek:

```
void main( ) {

   struct {
     long kimlik;
     int yas;

   }bdate;

bdate.kimlik=12;
bdate.yas=30;

   printf("\r\nKimlik No  : %lu",bdate.kimlik);
   printf("\r\nYas : %d",bdate.yas);

   while(TRUE);

}
```

4. UYGULAMALAR

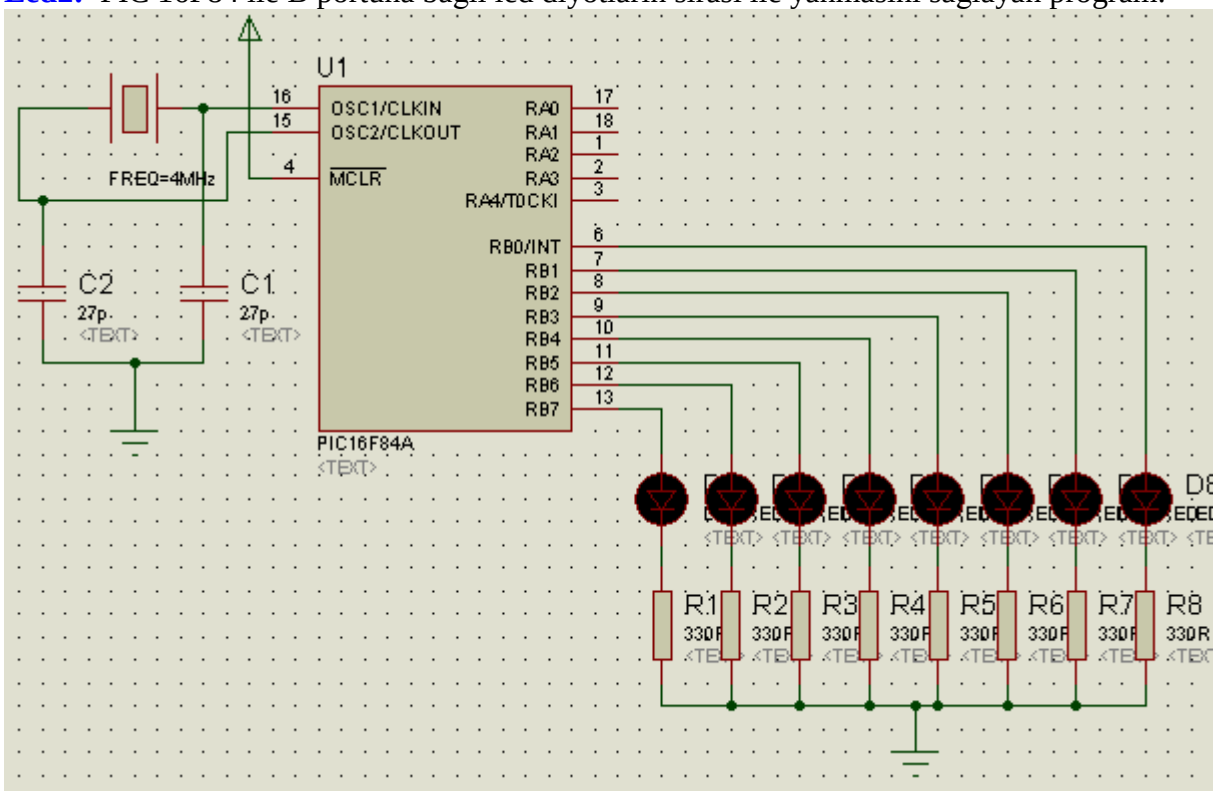
Led Diyod-Buton Uygulamaları

Led1: PIC 16F84 ile B portunun 0. bitine bağlı led diyodu 500 mili Saniye yakıp söndüren program.




```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)


void main()
{

while(1)
{
output_b(1);
delay_ms(500);
output_b(0);
delay_ms(500);
}


}
```

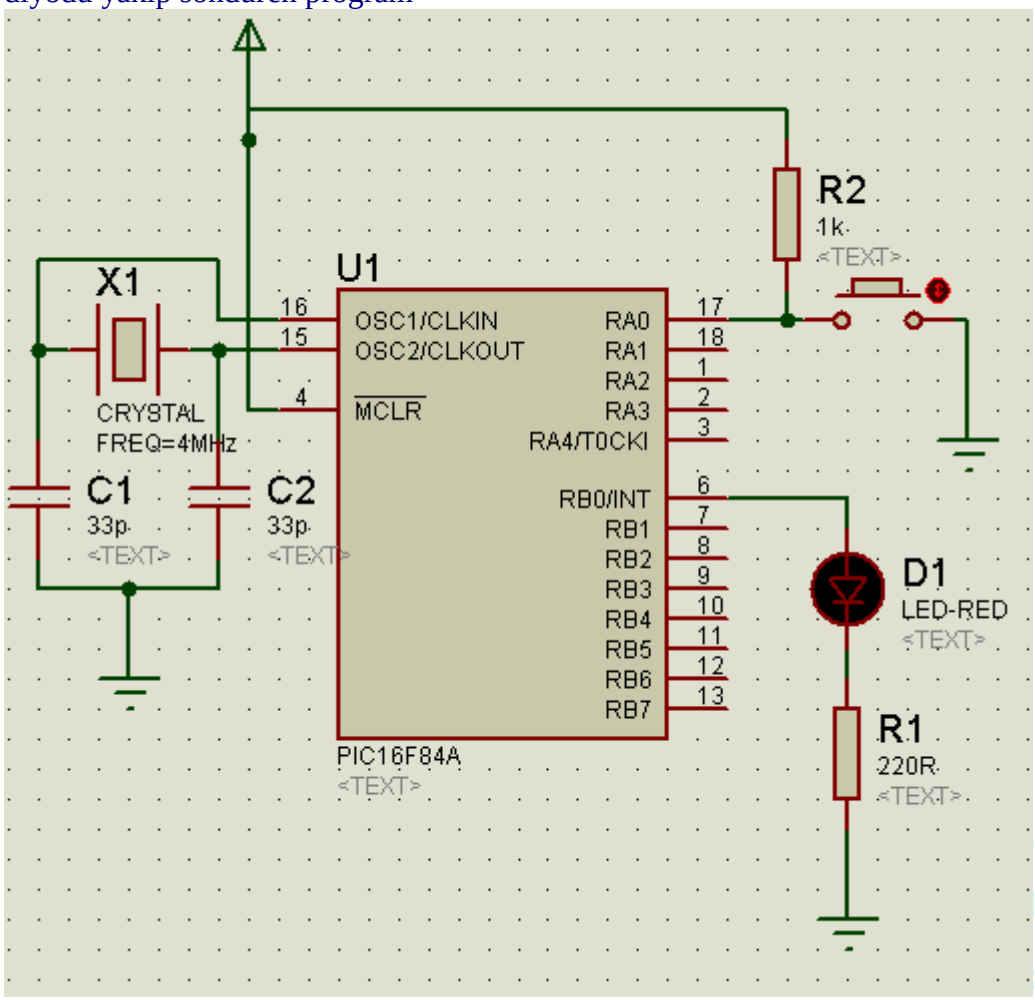
Led2: PIC 16F84 ile B portuna bağlı led diyotların sırası ile yanmasını sağlayan program.




```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)

char a;
void main()
{
a=1;
while(1)
{

output_b(a);
a=a<<1;
delay_ms(250);
if(a==256) a=1;
}

}
```

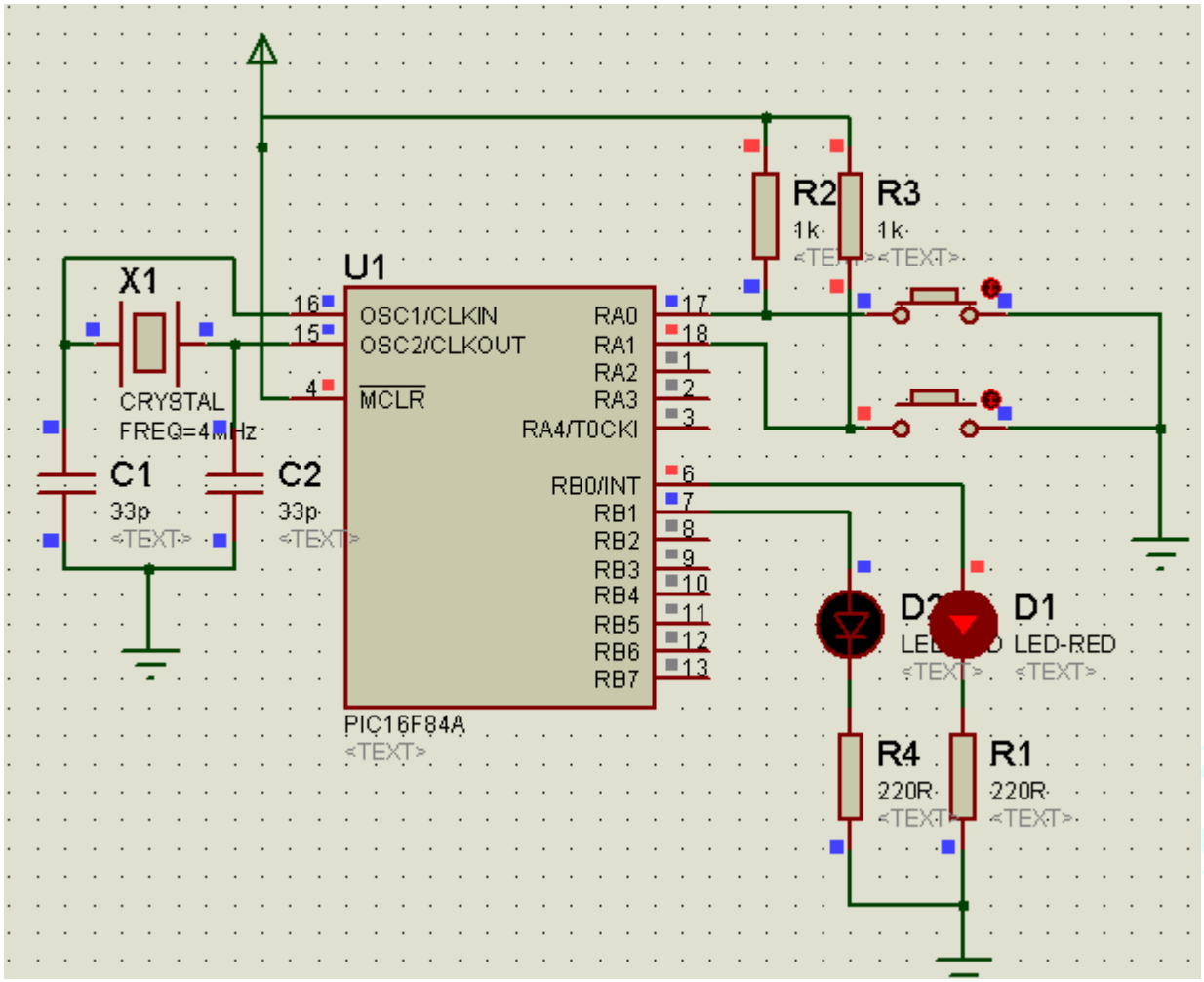
Programda ledlerin sağdan sola değil de soldan sağa ilerleyerek yanması istenseydi a=128 değeri atanır ve a=a>>1; şeklinde değiştirilirdi. Ayrıca if kontrolü de if(a= =0) a=128; yapılırdı.

Buton 1 : A portunun 0. bitine bağlı olan butona basıldığında B portunun 0.bitine bağlı olan led diyodu yakıp söndüren program




```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)


void main()
{

while(1)
{
if(!input(PIN_A0))
{
output_b(1);
delay_ms(250);
output_b(0);
delay_ms(250);
}

}

}
```

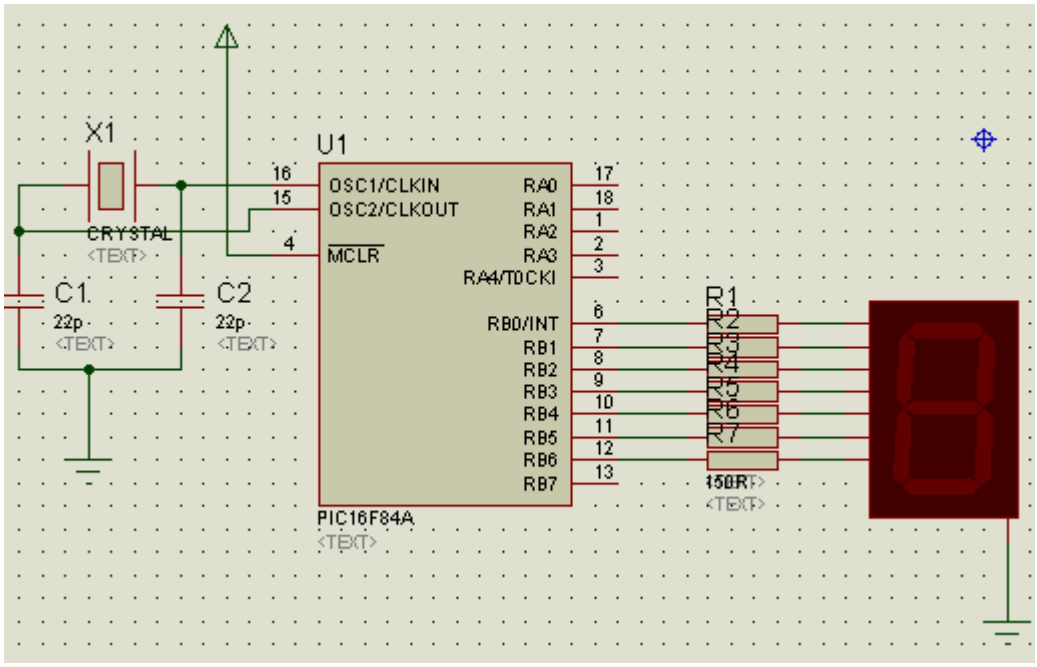
Buton 2 : A portunun 0. bitine bağlı olan butona basıldığında B portunun 0. bitine bağlı olan led diyodu yakan, A portunun 1. bitine bağlı olan butona basıldığında B portunun 1. bitine bağlı olan led diyodu yakan program.




```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)


void main()
{

while(1)
{
if(!input(PIN_A0)) output_high(PIN_B0);
else output_low(PIN_B0);
if(!input(PIN_A1)) output_high(PIN_B1);
else output_low(PIN_B1);

}

}
```

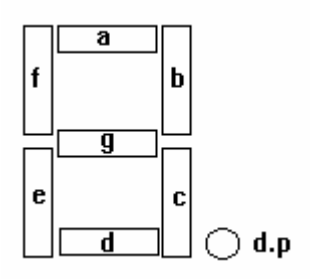
Display ugulamaları

7 Segment 1: 7 Segment displayde sabit bir sayının gösterilmesi (örnek 5 rakamı)



Ortak katotlu display kullanılmıştır.


```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)

void main()
{

while(1)
{
output_b(109);
}


}
```



5 Rakamı göstermesi için a,c,d,f,g segmentlerinin lojik1, diğer segmentlerin lojik 0 olması gerekmektedir. PIC 16F84 uçlarına bağlantı şekline göre (Örnekte RB0 dan başlanarak bağlantı yapılmıştır.) porta gönderilecek değer hesaplanır.

Şekildeki tabloya bakılarak hangi rakam için porta hangi bilginin gönderilmesi gerektiği bulunur.

Rakam	*H*	*G*	*F*	*E*	*D*	*C*	*B*	*A*	*Binary*	*Desimal*
0	0	0	1	1	1	1	1	1	00111111	63
1	0	0	0	0	0	1	1	0	00000110	6
2	0	1	0	1	1	0	1	1	01011011	91
3	0	1	0	0	1	1	1	1	01001111	79
4	0	1	1	0	0	1	1	0	01100110	102
5	0	1	1	0	1	1	0	1	01101101	109
6	0	1	1	1	1	1	0	1	01111101	125
7	0	0	0	0	0	1	1	1	00000111	7
8	0	1	1	1	1	1	1	1	01111111	127
9	0	1	1	0	1	1	1	1	01101111	111

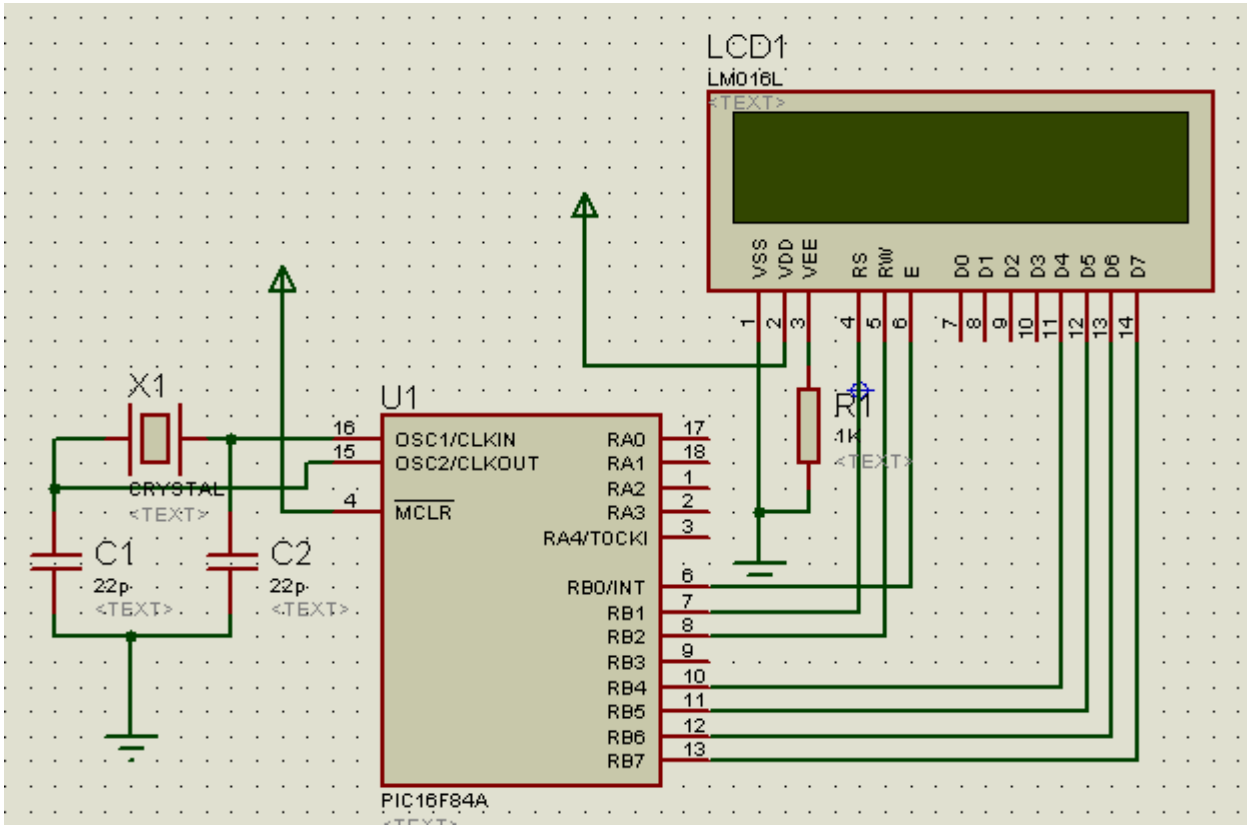
7 Segment 2: 7 Segment displayde 0-9 arası sayma işleminin yapılması

Uygulama devresi uygulama 3 teki ile aynı olacak.

```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)

char dizi[]={63,6,91,79,102,109,125,7,127,111};
char x=0;

void main()
{

while(1)
{
output_b(dizi[x]);
delay_ms(500);
x++;
if(x>9) x=0;
}
}
```

Bu uygulamada 0-9 arası sayma yapmak için 7 segmente gönderilecek değerler bir dizide gösterilmiştir. Dizi elemanları sırasıyla porta gönderilerek ve arada gecikme süresi bırakılarak sayma işlemi yaptırılmıştır.

LCD 1: 2x16 lık bir LCD displayde sabit bir yazının yazdırılması.




```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
#include<lcd.c>

void main()
{
lcd_init();

while(1)
{
lcd_gotoxy(1,2);   // LCD nin 1. sütun, 2 satırına yazdırmak içindir.
lcd_putc("\ mehmet\ ");
}
}
```

LCD uygulamalarında LCD.C include dosyası kullanılmalıdır. Bu dosya genel bir dosyadır. Hem B portu için hemde D portu için kullanılabilmektedir. Eğer B portu kullanılacak ise driver klasörü içerisinden LCD.C dosyası açılmalı ve B portu kullanılacak şekilde ayarlanmalıdır. Ayarlama aşağıdaki satırın önündeki // işaretlerinin kaldırılması ile olacaktır.

```
//#define use_portb_lcd TRUE
```

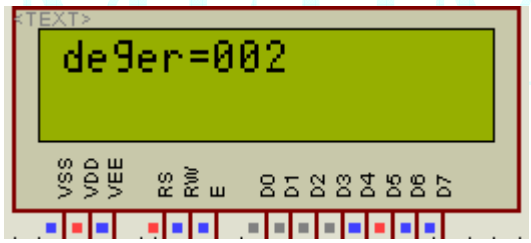
```
#define use_portb_lcd TRUE
```

// işareti kaldırıldığı zaman yukarıdaki şekilde renk değişimi olur.

LCD 2 : LCD Displayde bir değişkenin yazdırılması

Uygulama devresi bir önceki uygulama ile aynı olacaktır.

```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
#include<lcd.c>
char n=0;
void main()
{
lcd_init();

while(1)
{
n++;
if(n>255) n=0;
lcd_gotoxy(1,1); // Bu satır ile 1. satır 1. sütundan başlanarak yazılır.
printf(LCD_PUTC, "deger=%03u",n);  // 03 u ile 3 basamaklı sayı ve boş basamakta 0

                                   //görüntülenecek şekilde yazdırılır.

delay_ms(200);
}
}
```

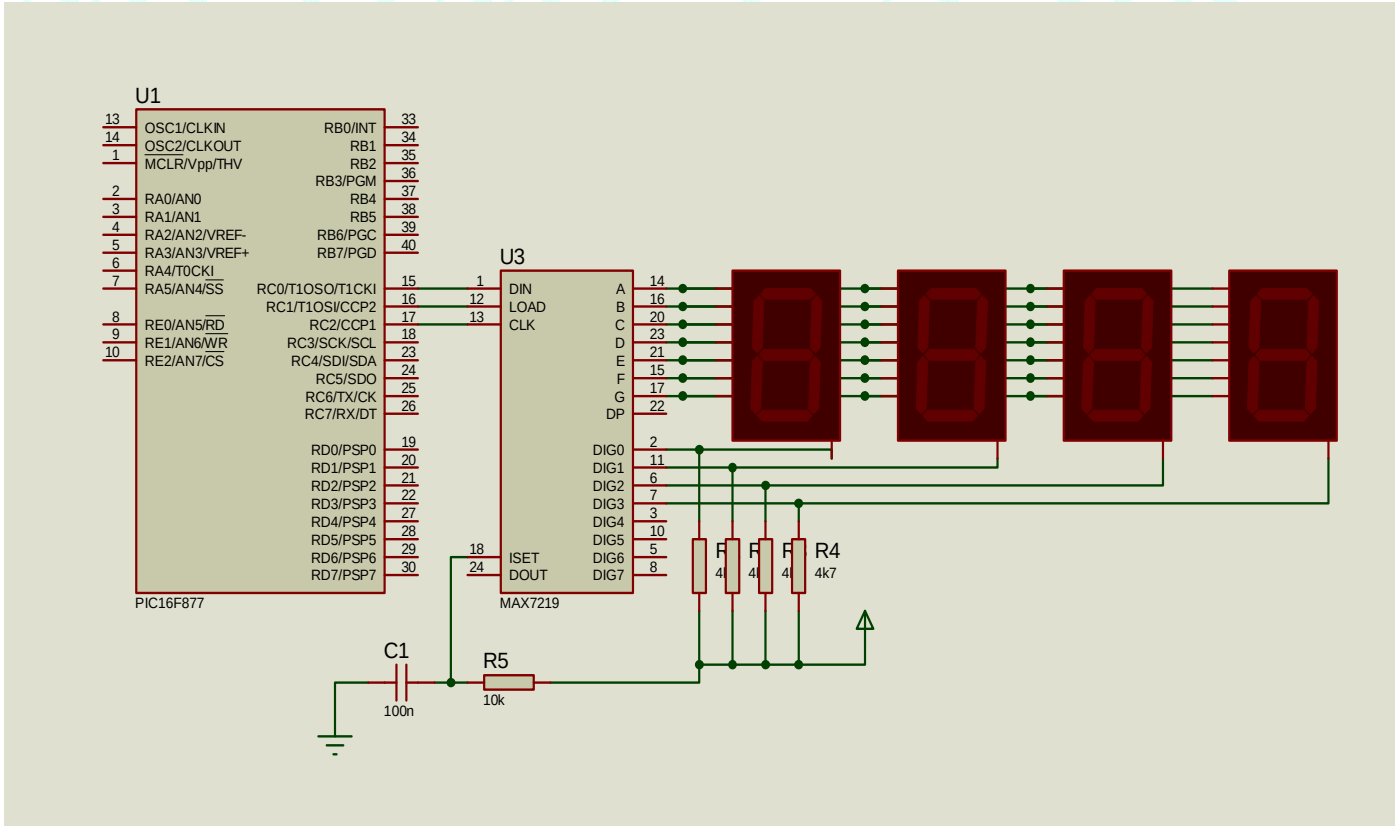
Ekran görüntüsü bu şekilde olmaktadır. Sayı 0 dan başlayarak sayar ve 255 ten sonra tekrar 0 a döner. Burada yazdırılan n değeri yerine başka bir yerden alınan değişken değeri de yazdırılabilir. (Örnek ADC girişindeki bir bilgi)

LCD 3: 2x16 lık displayde sağdan sola doğru kayan yazı uygulaması.

Devre yukarıdaki devre ile aynı olacaktır.

```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)

#include<lcd.c>
char x=16;
void main()
{
lcd_init();

while(1)
{
x--;
lcd_gotoxy(x,1);
lcd_putc("\ MEHMET\ ");
delay_ms(200);
if(x= =-6) x=16;
}
}
```

MAX 7219 1: MAX 7219 7 segment display tarayıcı entegresi uygulaması (İstenilen segmente istenilen değer yazdırma)



```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,NOBROWNOUT
#use delay(clock=20000000)
#include <max7219.c>


void main( )
{

init_7219( );

while(true){

send_16bit(1,1); // 1. display e 1 rakamını yazdırır.
send_16bit(2,2); // 2. display e 2 rakamını yazdırır.
send_16bit(3,3); // 3. display e 3 rakamını yazdırır.
send_16bit(4,4); // 4. display e 4 rakamını yazdırır.

}
}
```

Bu programı yazarken max 7219 driver dosyası gerekmektedir.
Bu dosya max7219.c isimli olarak driver klasörü içerisine kaydedilmelidr.


```
//////////////////////////////////////
Bu alanda haberleşmede PIC in hangi uçları kullanılacaksa o tanımlanır.
//////////////////////////////////////
#define da pin_c0
#define cs pin_c1
#define clk pin_c2
//////////////////////////////////////

void send_16bit(byte address,byte data)
{
byte i;
#bit carry=0x03.0;
carry=0;
output_low(cs);
for(i=0;i<8;i++)
{
if((address & 0b10000000)==0)
output_low(da);
else
output_high(da);
rotate_left(&address,1);

output_high(clk);
delay_us(50);
output_low(clk);
delay_us(50);
}
```

```
for(i=0;i<8;i++)
{
if((data & 0b10000000)==0)
output_low(da);
else
output_high(da);

rotate_left(&data,1);
output_high(clk);
delay_us(50);
output_low(clk);
delay_us(50);
}
output_high(cs);
}

void init_7219()
{
send_16bit(0x09,0xff); //decode modu
delay_us(100);
send_16bit(0x0a,0x0f); //parlaklık ayarı( 0-f arası yapılabilir.)
delay_us(100);
send_16bit(0x0b,0x03); //tarama yapılan display adeti  0x03 olursa 4 adet displayi tarar
delay_us(100);
send_16bit(0x0c,0x01); //display i kapatmak için kullanılır.(0x00 kapalı)
delay_us(100);
send_16bit(0x0f,0x00); //displayleri test etmek içindir. İlk önce tarama yapılır.  (0x01 test taraması
yapar.)
delay_us(100);
}
```


Decode modu:

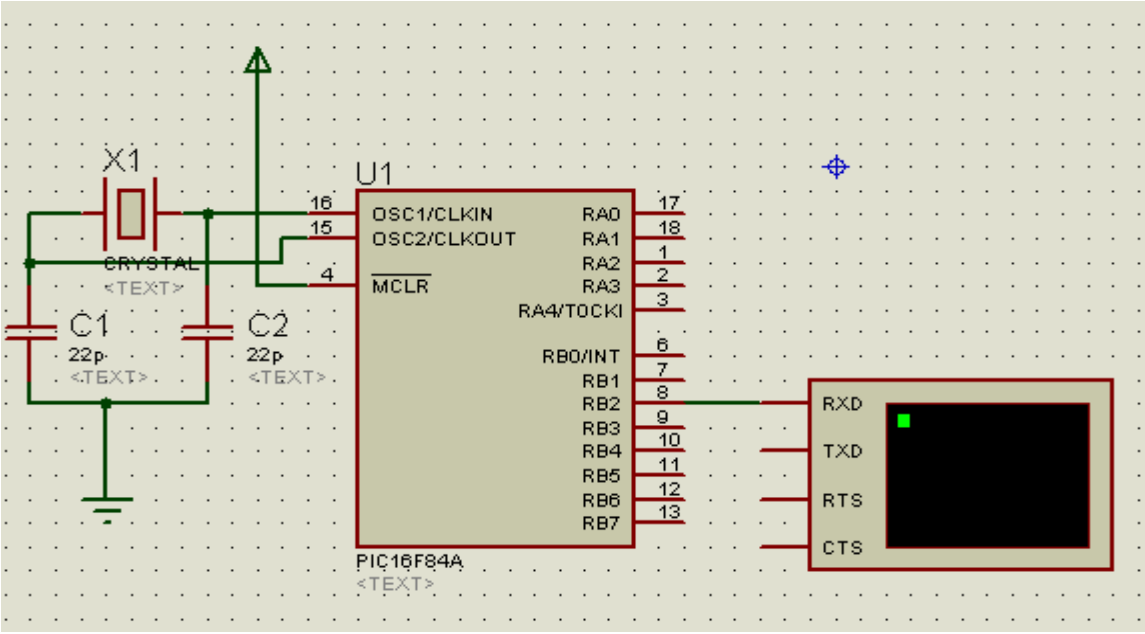
Bu modde 0x09 nolu register e gönderilen bilgiye göre kod çözme işlemi yapılır. Örneğin FF yazıldığında tüm displayler kod çözülmüş olur. Yani biz herhangi bir displaye 2 bilgisi gönderirsek bu bilgi 2 olarak gözükür. Eğer kod çözme aktif edilmezse FF yerine 01 yazılır ise 0. displayde kod çözülür. Diğerlerinde ise tek bitlik bilgi gönderilmiş olunur.

DECODE MODU	**REGISTER DATA**								**HEX KOD**
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	
Kod çözülmez	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0x00**
Sadece 1 displayde çözülür	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0x01**
İlk 4 display de çözülür	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0x0F**
Tüm displaylerde kod çözülür.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0xFF**

Font Modu:

Displayde Görünen	Binary Karşılığı				Desimal Karşılığı	Displayde nokta göstermek için
0	0	0	0	0	0	128+0=128
1	0	0	0	1	1	128+1=129
2	0	0	1	0	2	128+2=130
3	0	0	1	1	3	128+3=131
4	0	1	0	0	4	128+4=132
5	0	1	0	1	5	128+5=133
6	1	1	0	0	6	128+6=132
7	0	1	1	1	7	128+7=135
8	1	0	0	0	8	128+8=136
9	1	0	0	1	9	128+9=137
-	1	0	1	0	10	128+10=138
E	1	0	1	1	11	128+11=139
H	1	1	0	0	12	128+12=140
L	1	1	0	1	13	128+13=141
P	1	1	1	0	14	128+14=142
Boşluk	1	1	1	1	15	128+15=143

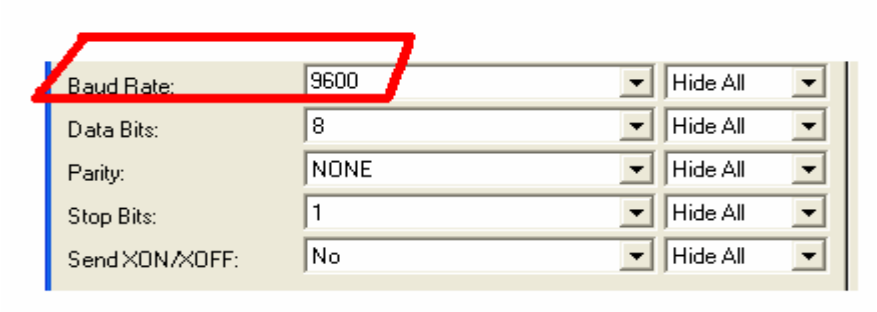
Seri İletişim 1: PIC 16F84 ile seri bir bilginin gönderilmesi.




```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
```

```
#use rs232(baud=9600, xmit=PIN_B2,rcv=PIN_B1)

void main()
{

while(1)
{
putc('M');
delay_ms(500);
}

}
```

Bu uygulamada PIC 16F84 ten seri bir bilginin gönderilmesi sağlanmıştır. Proteus simulasyonunda virtual terminal seçildiğinde ve gerekli ayarlamalar yapıldığında 500mS aralıklarla M karakteri gönderilecektir. Burada dikkat edilmesi gereken husus virtual terminal ayarlarının doğru yapılması gerekmektedir.



C programında baud hızı olarak ne belirtilmişse burada da aynı hız ayarlanmalıdır. Ayrıca dat biti parite biti gibi ayarlamalar program içerisinde yazılmışsa burada doğru olarak tanımlanmalıdır.

RS232 KOMUTLARI:

#use rs232 (***seçenekler)***

Seçenekler

BAUD=x	Baud hızı
XMIT=pin	Bilgi gönderim pini
RCV=pin	Bilgi alım pini
FORCE_SW	Yazılımla farklı 2 UART pini tanımlarken kullanılır.(Donanımsal olarak USART destekleyen PIC lerde kullanılabilir)
BRGH1OK	Farklı baud oranlarına izin vermek için kullanılır.
ENABLE=pin	RS 485 iletişimde kullanılır.
INVERT	Seri bilginin polaritesini değiştirmek için kullanılır.(Max 232 da data+ data – uçlarını yer değiştirmek için)
PARITY=X	X Değişkeni N,E ve O harfleri kullanılır (NOT, EVEN, ODD)
BITS =X	X değişkeni 5-9 arası bit değeri olabilir.

#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

RS232 haberleşme protokolünde haberleşme hızı baud, bilgi gönderme ucu xmit ve bilgi alma ucu rcv belirlenmelidir.

Baud hızı olarak 300-9600 arası bir değer verilebilir.

getc( ) , getch( ), getchar( ) komutları:

rs232 Haberleşme protokolünde rcv ucundan gelen bilgileri almak için kullanılır.

value= getc( ) şeklinde kullanılır. rcv ucundan seri olarak gelen bilgi value isimli değişkene atanır.

putc (cdata), putchar (cdata) Komutları:

rs232 Seri iletişimde bilgi göndermek için kullanılır. Bilgi char tipli veri olarak gönderilir.

gets (*string*) Komutu:

String tipli verileri almak için kullanılır.

printf (*string*) Komutu:

Seri olarak bilgi gönderiminde kullanılır.

printf("Merhaba");

printf("RTCC Degeri =%2x\n\r",get_rtcc());

printf("%2u %X %4X\n\r",x,y,z);

printf(LCD_PUTC, "n=%u",n);

şeklinde kullanımları vardır.

Örnek Format:

Özellik	Değer=0x12	Değer=0xfe
%03u	018	254
%u	18	254
%2u	18	*
%5	18	254
%d	18	-2
%x	12	fe
%X	12	FE
%4X	0012	00FE
%3.1w	1.8	25.4

c	Character
s	String veya character
u	Unsigned int
d	Signed int
Lu	Long unsigned int
Ld	Long signed int
x	Hex int (küçük karakter)
X	Hex int (büyük karakter)
Lx	Hex long int (küçük karakter)
LX	Hex long int (büyük karakter)
f	Float with truncated decimal
g	Float (tam sayıdan fazlası silinmiş)
e	Float (tam sayıya yuvarlanmış)

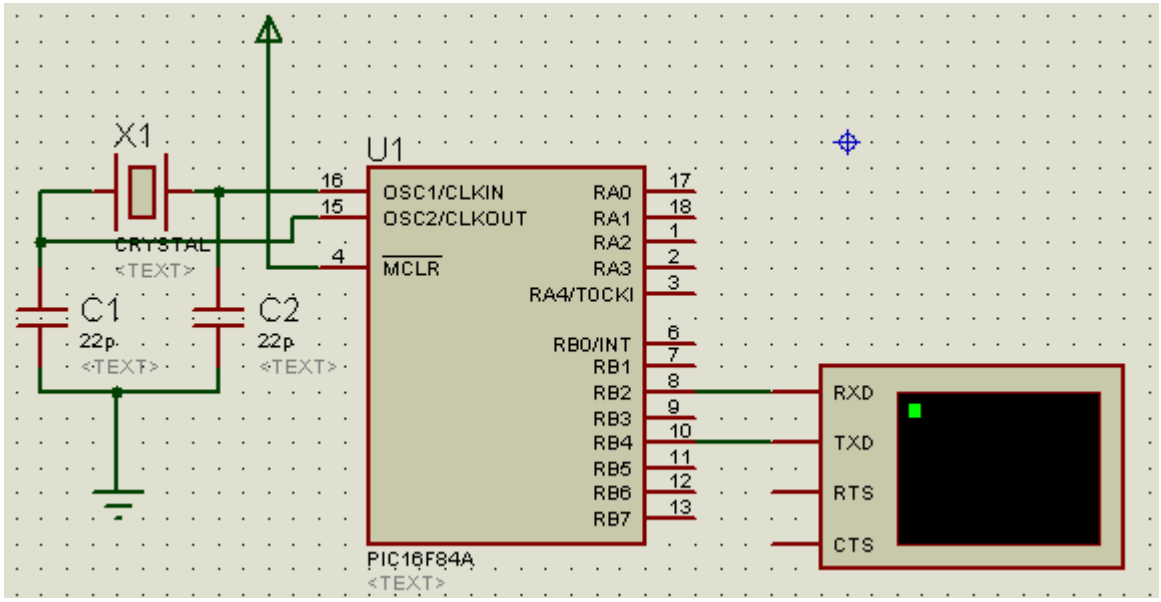
set_uart_speed (*baud*, [*stream*]) Komutu:

100-115200 arası baud hızı ile uart haberleşme hızı ayarlanabilir.

set_uart_speed(2400);

Şeklinde kullanılır.

Seri İletişim 2: Seri bilgi gönderilmesi ve alınması




```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
```

```
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_B2,rcv=PIN_B4)
int a;
void main()
{

while(1)
{
a=getc();
delay_ms(500);
putc(a);
}

}
```

Bu uygulamada ise seri olarak gelen bilgi 500mS bekletilerek yine seri olarak gönderilmektedir. Virtual terminalden girilen karakter gecikmeli olarak terminal ekranında görüntülenecektir. Seri olarak alınan bilgiye göre işlem yapılabilir.

Seri İletişim 3: Seri olarak alınan bilgi a karakteri ise seri olarak M karakteri gönderen, a değil ise X karakteri gönderen program.

Uygulama devresi aynı devredir.

```
#include <16F84.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_B2,rcv=PIN_B4)
int deg;
```



```
void main()
{

while(1)
{
deg=getc();
delay_ms(500);
if(deg=='a') putc('M');
else putc('X');
}

}
```

ADC 1: Analog Dijital Dönüştürücü Kullanımı (Analog uçtan okunan değerin LCD de gösterilmesi)

Öncelikle ADC komutlarından ve nasıl kullanılacağından bahsedelim.

ADC Komutları ve ayarlamaları:

setup_adc(mod); Bu komut ile analog-digital konvertör ayarlaması yapılır.

Mod olarak

ADC_OFF : ADC Kapalı
ADC_CLOCK_INTERNAL : Dahili osilatör
ADC_CLOCK_DIV_32 : Fosc/32
ADC_CLOCK_DIV_8 : Fosc/8
ADC_CLOCK_DIV_2 : Fosc/2

Kullanılabilir.

setup_adc_ports(Değer); Bu komut ile ADC giriş ucu seçilmektedir.

Değer olarak
NO ANALOG
ALL ANALOG
A0…A5
E0..E2
Şeklinde mikrodenetleyici türüne göre seçim yapılabilir.

NO_ANALOGS
ALL_ANALOG
AN0_AN1_AN2_AN4_AN5_AN6_AN7_VSS_VREF 1
AN0_AN1_AN2_AN3_AN4
AN0_AN1_AN2_AN4_VSS_VREF
AN0_AN1_AN3
AN0_AN1_VSS_VREF
AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF
AN0_AN1_AN2_AN3_AN4_AN5
AN0_AN1_AN2_AN4_AN5_VSS_VREF
AN0_AN1_AN4_AN5_VREF_VREF
AN0_AN1_AN4_VREF_VREF
AN0_AN1_VREF_VREF
AN0
AN0_VREF_VREF
ANALOG_RA3_REF
A_ANALOG
A_ANALOG_RA3_REF
RA0_RA1_RA3_ANALOG
RA0_RA1_ANALOG_RA3_REF
ANALOG_RA3_RA2_REF
ANALOG_NOT_RE1_RE2
ANALOG_NOT_RE1_RE2_REF_RA3
ANALOG_NOT_RE1_RE2_REF_RA3_RA2
A_ANALOG_RA3_RA2_REF
RA0_RA1_ANALOG_RA3_RA2_REF
RA0_ANALOG
RA0_ANALOG_RA3_RA2_REF

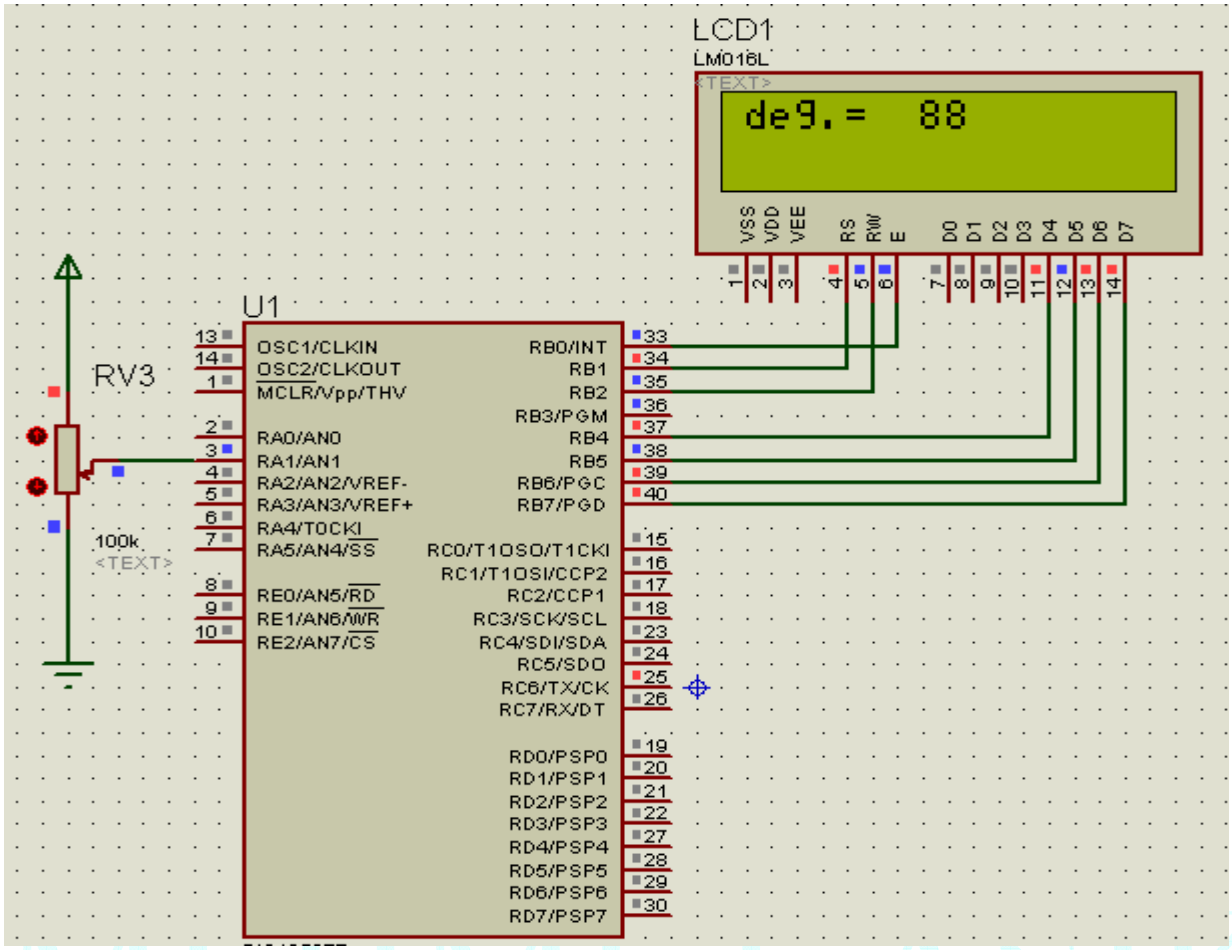
Değerleri kullanılabilir (PIC 16F877için bu değerler kullanılabilir.)
set_adc_channel(kanal); Bu komut ile analog giriş olarak kullanılacak olan uç seçilir.

set_adc_channel(1);

setup_adc_ports( ) ALL_ANALOG seçilmişse aşağıdaki değere bakılarak giriş kanalı seçilmelidir.
A0 A1 A2 A3 A5 E0 E1 E2

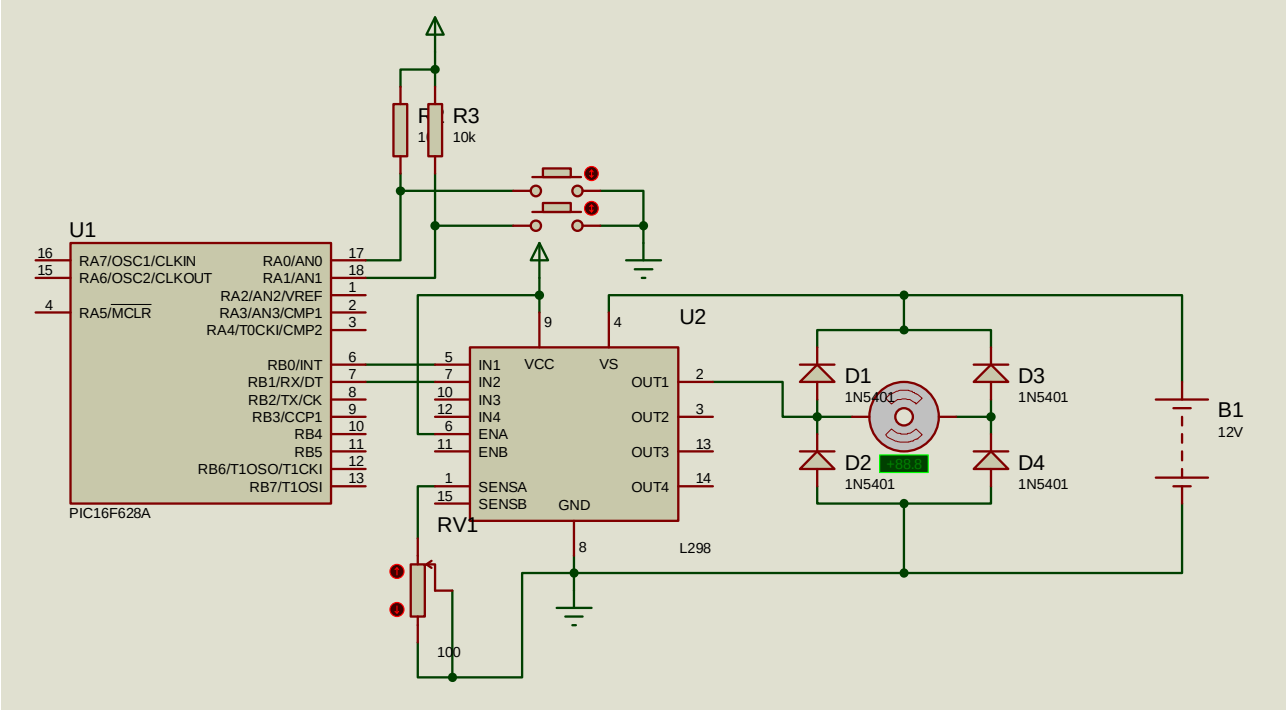
#device ADC=10 **// 10 Bit değer okumak için ayarlanmalıdır.**

Değer okumak içinse
a = read_adc();
a gibi bir değişkene dijitale dönüştürülmüş analog bilgi kaydedilir.



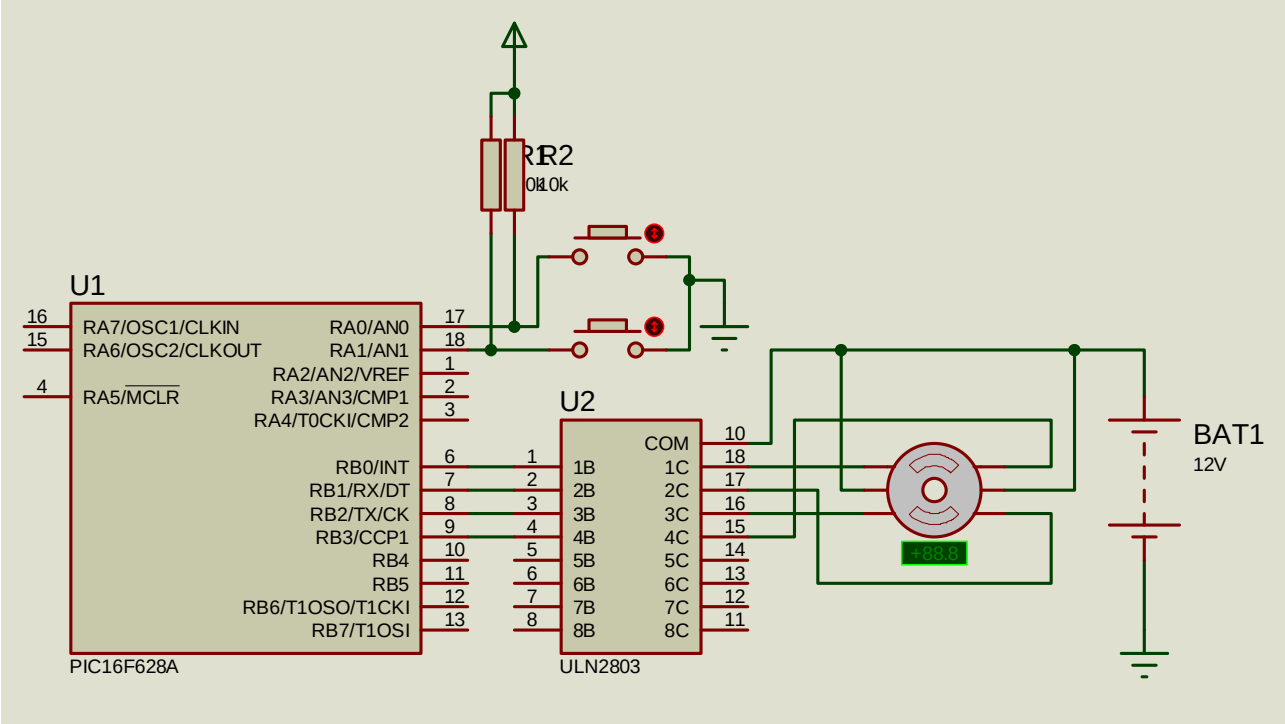

```
#include <16F877.h>
#device ADC=10  // 10 Bit değer okumak için ayarlanmalıdır.
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
#include<lcd.c>
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

int32 n;
void main()
{
lcd_init();

setup_adc(ADC_CLOCK_DIV_2 ); // clock değeri Fosc/2 olarak seçilmiştir.
SETUP_ADC_PORTS( ALL_ANALOG );// Tüm uçlar analog olarak ayarlanmştır.
set_adc_channel( 1 ); // Giriş ucu olarak 1 nolu uç (RA1) seçilmiştir.


while(1)
{

n=read_adc();
lcd_gotoxy(1,1);
printf(LCD_PUTC, "deg.=%4lu",n); // lcd de yazdırmak için long unsigned olarak yazdırmalıdır.

}
}
```

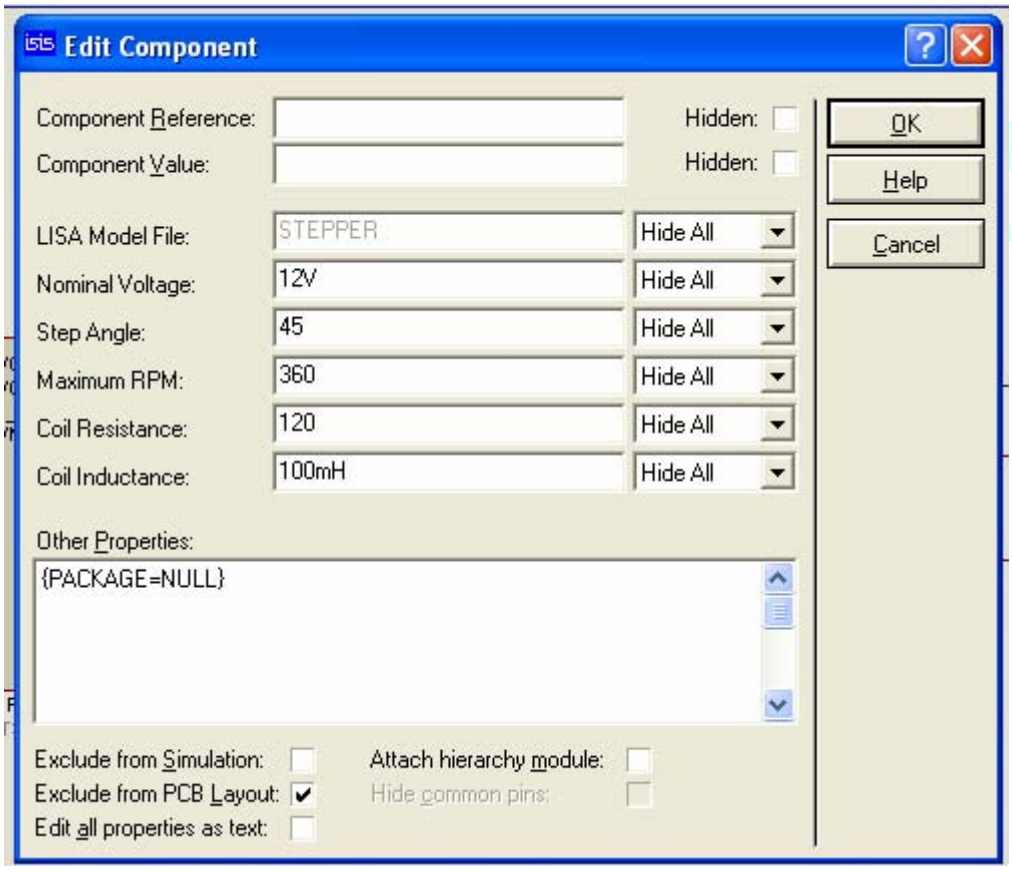
Motor 1: (DC Motor): Butonlara basılarak motor yön kontrolü, potansiyometre ile de hız kontrolü yapılabilmektedir.




```
#include <16F628.h>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
void main( ){

while(true){
if(!input(PIN_A0)) output_low(PIN_B0);
else output_high(PIN_B0);
if(!input(PIN_A1)) output_low(PIN_B1);
else output_high(PIN_B1);

}
}
```

Motor 2: (Step Motor): Butonlara basarak step motor her ili yönde de kontrol edilebilir.





Step angle adım açısıdır. Bu açı yazılan programa göre değiştirilmelidir. Yani step motor 4 adımda tam tur döndürülüyorsa step angle 90 olmalıdır. 8 Adımda Döndürülecekse 45 dersece olarak ayarlanmalıdır. Ayrıca motorun besleme gerilimi 12V, 9V ya da farklı gerilim olabilir. Uygulamada buna dikkat edilmelidir. Motor uçlarının tespitinde her step motorun farklı olabileceği düşünülerek sabit uç bulunmalı ve ona göre besleme gerilimi verilmelidir.

Sürücü olarak ULN 2803 entegresi kullanılmıştır. Her bir uçtan 500mA akım çekilebilir. Kullanılan step motorun çekeceği akıma göre sürücü entegre seçilmelidir. Step motorun uçları asla doğrudan PIC e bağlanmamalıdır. Aksi takdirde PIC bozulur.

```
#include <16F628.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
int a;


BYTE const poz8[9] = {     0b0000,
                           0b1000,
                           0b1001,
                           0b0001,
                           0b0101,
                           0b0100,
                           0b0110,
                           0b0010,
                           0b1010};



void main( ) {
a=0;
while(true)
{

if(!input(PIN_A0))
{
 if(a>8) a=0;
a++;
 output_b(poz8[a]);
 delay_ms(50);


}

if(!input(PIN_A1))
{
if(a<1) a=9;
a--;
 output_b(poz8[a]);
 delay_ms(50);

}
}
}
```

GİRİŞ/ÇIKIŞ (INPUT/OUTPUT) KOMUTLARI:

output_low (*pin*);

Belirtilen pin i 0(sıfır) yapar.

Örnek: output_low(PIN_B7); // B Portunun 7. bitini 0 yapar.

output_high (*pin*);

Belirtilen pin i 1 yapar.

Örnek: output_high(PIN_B7); // B Portunun 7. bitini 1 yapar

output_bit (*pin*, *değer*);

Belirtilen pini 1 ya da 0 yapar.

output_a (*değer*);

A portuna istenilen bir değeri göndermek için kullanılır.

output_b( ); output_c( ); output_d( )

Şeklinde diğer portlarda çıkış olarak yönlendirilir ve değer gönderilebilir.

Bu komut kullanıldığında port hem çıkış olarak yönlendirilir hem de porta belirtilen değer gönderilir.

SPI KOMUTLARI

setup_spi (***mode***) Komutu:

SPI Haberleşmesi yapılması için SPI kurulumu yapılması gerekmektedir. mode içi yazılabilecekler;

- SPI_MASTER, SPI_SLAVE, SPI_SS_DISABLED
- SPI_L_TO_H, SPI_H_TO_L
- SPI_CLK_DIV_4, SPI_CLK_DIV_16,
- SPI_CLK_DIV_64, SPI_CLK_T2

SPI Haberleşmede 3 veya 4 uç kullanılabilir. Bunlar DATA IN, DATA OUT, CLK, SELECT uçlarıdır. Sadece bilgi okunacaksa DATA OUT kullanılmayabilir.

spi_read (***data***) Komutu:

a=spi_read(adres)

SPI ile okunan değer bir değişkene aktarılmalıdır. Okunmak istenen adres belirtilmelidir.

SPI_WRITE (***value***) Komutu:

spi_write(0xa); Kullanılan eeprom a göre belirlenen bir değerdir.

spi_write(address); Bilgi yazılmak istenen adres bilgisidir.

spi_write(data); Yazılmak istenen veri değeridir.

I2C KOMUTLARI
#use i2c(master, sda=PIN_C4, scl=PIN_C3)

I2C Haberleşmesi kullanılacaksa yukarıdaki tanımlama yapılmış olmalıdır. Programda CCS derleyicisi içerisinde bulunan INCLUDE dosya kullanılarak program yazımı sade ve kolay olur. sda= PIN_C4 ile harici EEPROM un sda ucunun PIC te hangi uca bağlanacağı tanımlanır. scl=PIN_C3 ile scl ucunun bağlanacağı PIC ucu tanımlanmış olunur.

i2c_start();

i2c_write(0xa0);

i2c_write(address);

data=i2c_read(0);

i2c_stop();

Komutları kullanılmaktadır. INCLUDE dosyalardan faydalanılabilir.

Örnek: 2416.c include dosyası kullanılacak olursa read_ext_eeprom komutu bir değişkene eşitlenir ve harici EEPROM dan veri okunur.

write_ext_eeprom(adres,data) komutu kullanılarak istenilen bir adrese bilgi yazdırılabilir.

Harici EEPROM uygulamasında dikkat edilmesi gereken bir husus sda ve scl uçlarına pull-up dirençlerinin bağlanması gerektiğidir. (4,7Kohm luk bir direnç +5V ucuna bağlanmalıdır)

Program Yazım Formatı:

Program Komutu	**Açıklama**
#if defined(__PCM__)	Mikrodenetleyici grubunu belirlemek içindir.(12Bit, 14Bit) Kullanılması zorunlu değildir.
#include <16F628.h>	Mikrodenetleyici çeşitini belirtmek için kullanılır.
#fuses NOWDT,NOPROTECT,INTRC_IO,NOMCLR	Mikrodenetleyicide sigorta ayarlamaları için kullanılır. (Kod koruma, osilatör tipi gibi)
#use delay(clock=4000000)	Devrede kullanılacak osilatör frekansı belirtilmelidir. Örnekte 4000000Hz = 4MHz osilatör

	frekansı seçilmiştir.
#use rs232(baud=9600, xmit=PIN_B2, rcv=PIN_B1)	RS232 haberleşmesi kullanılacak ise TX, RX uçları belirlenir ve bilgi akış hızı belirlenir.)

void main( )
{
 	
}
	Ana programın olduğu yerdir.
	{ } parantezi içerisine PIC mikrodenetleyiciye yaptırılmak istenen işlemlerin komutları yazılır.

Ana program parantezinden önce varsa değişken tanımlamaları yapılmalıdır. Ayrıca kullanılan eleman varsa (LCD, KEYBOARD gibi) onların dosyaları tanımlanmalıdır.