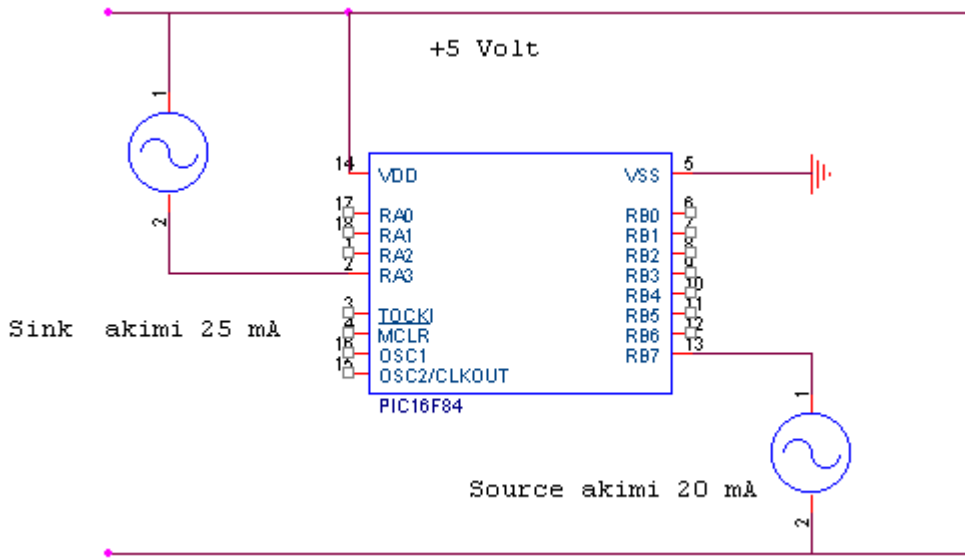


16F84'ün çektiği akım saat hızına bağlıdır. Saat hızı arttıkça çekilen akım artar. 4 Mhz saat hızında çekilen akım 2 mA kadardır. Eğer pic sleep modda ise bu akım 40 mikroamper'e kadar düşer.

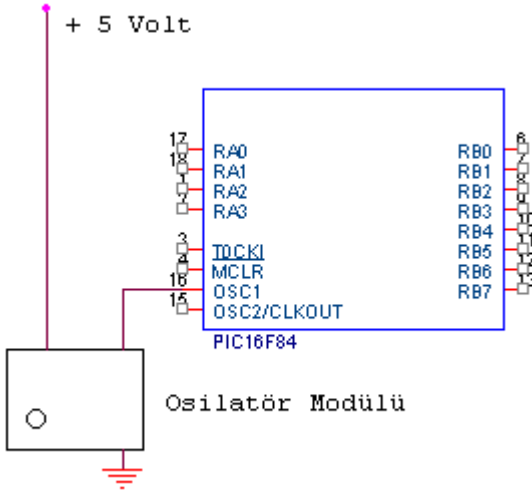
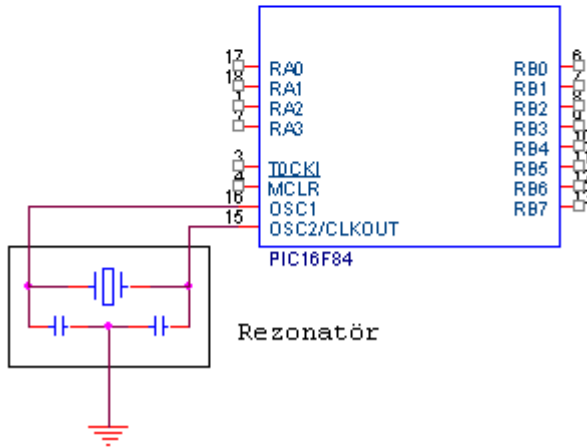
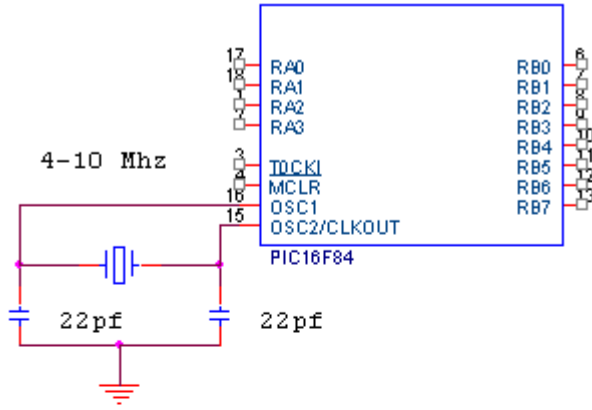
CMOS entegrelerde giriş uçlarını +5 Volt'a bağlamak gerekir. 16F84 de bir CMOS entegredir, fakat burada pic içinde I/O pinleri özel şekilde bağlanmışlardır ve program aracılığı ile giriş veya çıkış haline getirilir. Giriş pozisyonunda bu uçlar pull-up konumundadır, yani pic içinde bir şekilde +5 volta bir direnç üzerinden bağlı gibidir. Burada pull-up dirençleri 50 K kadardır.



16F84'ün çalıştırılmasında en çok kullanılan yöntemler :

- 1-) Xtal osilatör
- 2-) Seramik rezonatör
- 3-) Osilatör Modülü

3. şıkta bahsedilen osilatör modülleri sıcaklık değişimlerine karşı da korunmuş oldukları için çok kararlı çalışırlar.



Bir 16F84'de + gerilim ile GND uçları arasında bir adet 0.1 uf kondansatör konması, istenmeyen bazı gerilim dalgalanmalarını önler.

16 nolu baktan girilen osilatör sinyali, pic içinde 4'e bölünür ve frekansın 1/4'ü 15 nolu baktan alınabilir. Bu 4'e bölünmüş saat frekansının karşılığı olan periyota "instruction cycle" yani komut süresi denir. Bu bir komutun işlenmesi için gereken

zamandır.16F84'de bu 4 Mhz de 1 mikrosaniye 10 Mhz de ise 0.4 mikrosaniyedir. Bu zaman, programlama esnasında çok önem arzeder, bu komut sürelerinin toplamı ile zamanlar hesaplanır.

Dışarıdan bağlanan osilatörler ile pic daha yavaş saat hızlarında da çalıştırılabilir. Bu nun için kullanılan uç RA4 /TOCKI pinidir. Burası çıkış olarak RA4, giriş olarak ta TOCKI girişidir (Timer Zero Clock Input).

Kristal osilatör kullanıldığında 16F84'e 22 pf kondansatör bağlanır. Seramik rezonatör için bir özellik yoktur.

16F84'ün 13 ucunu da çıkış olarak kullanabiliriz. Bu çıkışlar A ve B diye ikiye ayrılır A çıkışları 5 adettir RA0-RA1-RA2-RA3-RA4 diye adlandırılır. B çıkışları ise 8 adettir ve bunlarda RB0-RB1-RB2-RB3-RB4-RB5-RB6-RB7 diye adlandırılırlar. Bu çıkışlar da bir tek RA4 yani TOCKI ucu diğerlerinden farklıdır ve bu uç açık kollektör özelliği gösterir, bu nedenle çıkış olarak kullanıldığında, 10 K lık bir direnç le + 5 V a asılır.

RB0 ucu giriş olarak ayarlanırsa, aynı zamanda "external interrupt" girişidir. Yine B port'un 4-5-6-7 çıkışları (RB4-RB5-RB6-RB7) internal interrupt alma özelliğindedir.

Pic power-on reset ile yani pic'e enerji verildiğinde, başlangıç adresinden başlar. Bu durumda tüm pull-uplar iptal edilmiş olur ve pic portları çıkış pozisyonundadır. Programlama ile port çıkışları giriş veya çıkış olarak yönlendirilir, bunun için "option" register içinde de disable veya enable seçenekleri vardır. Power-on reset ile disable olan pic de tüm portlar çıkıştır.

Pic'i başlangıç adresinden başlatan 3 mekanizma vardır bunlar :

- 1-) Power-on reset
- 2-) MCLR reset (Master Clear Reset) kullanıcı tarafından programı başa döndürmeye yarar.
- 3-) Watchdog timer (İptal edilmediğinde programı belli zaman aralıkları ile başa döndürür.

Bu portların giriş veya çıkış olarak yönlendirilmesi TRIS register ile sağlanır, veya Write register vasıtası ile aktarılan komutlar yardımı ile port lar tek tek seçilebilir.

Pic 16F84'de iki hafıza blok'u mevcuttur. Bunlar

- 1- Program memory blok
- 2- Data memory blok

Program memory blok 14 bit kelime boyunda ve 1 Kbyte kapasitesindedir. 13 bit ile adreslenir, program bus 14 bittir. Buraya program olarak yazıp Pic içine attığımız hex dosyası bilgileri yazılır, gerekirse silinir ve tekrar yazılır.

Data memory alanı RAM ve EEPROM olmak üzere iki tiptir. RAM alanına programın çalışması sırasında işlenen bilgiler yazılır. Data memory alanında SFR ve GPR registerler bulunur.

Pic 16F84'ün en önemli registerleri şekilde görülmektedir. 68 GPR alanında bizim tanımladığımız: sayaç, timer vs... gibi registerler yazılır.

File Address		File Address	
00h	Indirect addr. ⁽¹⁾	Indirect addr. ⁽¹⁾	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h			87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2 ⁽¹⁾	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch			8Ch
	68 General Purpose registers (SRAM)	Mapped (accesses) in Bank 0	
4Fh			CFh
50h			D0h
7Fh			FFh
	Bank 0	Bank 1	

☐ Unimplemented data memory location; read as '0'.
 Note 1: Not a physical register.

Write Register: 16F84 içinde RAM bellek alanında, görülmeyen, direk ulaşılmayan, adresi olmayan, geçici bir depolama alanı vardır, burası "Write Register" dir. Bu registerler içine yazılan bilgiler başka registerlere aktarılır ve write register bu aktarma işleri için geçici aktarma alanı olarak kullanılır. Aritmetik ve atama işlemleri bu sayede yapılır.

Register adreslerinin bazılarını bilmek programlama işlemleri için önemlidir. Inc. dosyası kullanıldığında, register tanımları yapmaya gerek yoktur fakat temel ve en çok kullanılan registerlerin adreslerini ezbere bilmek gereklidir.

Örneğin:

```
PORTA EQU H'05'
PORTB EQU H'06'
STATUS EQU H'03'
TRISA EQU H'85'
TRISB EQU H'86'
```

Gibi registerler en temel registerlerdir, programlamada ilerledikçe diğer register adresleri de otomatikman bellekte kalır veya inc.dosyası kullanımı alışkanlığı ile tamamen bellekten silinir.

Aşağıda ileride gerekli olacak register ve onların bit kısaltmaları verilmiştir. Programlama esnasında, register bitleri, kısaltma olarak da yazılabilir. Inc. dosyalarında bunlar tanımlanmıştır.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								----	----
01h	TMR0	8-bit real-time clock/counter								xxxx	xxxx
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000	0000
03h	STATUS ⁽²⁾	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	000z quuu
04h	FSR	Indirect data memory address pointer 0								xxxx	xxxx
05h	PORTA	—	—	—	RA4/TOCKI	RA3	RA2	RA1	RA0	---x	xxxx
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx	xxxx
07h		Unimplemented location, read as '0'								----	----
08h	EEDATA	EEPROM data register								xxxx	xxxx
09h	EEADR	EEPROM address register								xxxx	xxxx
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC ⁽¹⁾				---	0000	
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000	000x
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								----	----
81h	OPTION	RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111	1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000	0000
83h	STATUS ⁽²⁾	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	000z quuu
84h	FSR	Indirect data memory address pointer 0								xxxx	xxxx
85h	TRISA	—	—	—	PORTA data direction register				---	1111	
86h	TRISB	PORTB data direction register								1111	1111
87h		Unimplemented location, read as '0'								----	----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---	0 x000
89h	EECON2	EEPROM control register 2 (not a physical register)								----	----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC ⁽¹⁾				---	0000	
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000	000x

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.

2: The TO and PD status bits in the STATUS register are not affected by a MCLR reset.

3: Other (non power-up) resets include: external reset through MCLR and the Watchdog Timer Reset.

Pic programlamak için gereken temel şeyler şunlardır.

- 1-) Pic programlamak için pic Assembly dilini bilmek ve program yazarak bunu hex dosyası haline getirmek
- 2-) Pic programlayacak donanıma sahip olmak, bunlar bilgisayar ve gerekli yazılımlardır.

Assembler veya Compiler:

MS-DOS/Edit
WINDOWS/notepad
MPLAB/Pfe

Gibi bir text editöründe Pic Assembly dili ile yazılmış programı derlemeye yani HEX file haline getirmeye yarayan programdır. Bu iş için biz Microchip firmasının MPLAB içinde gelen MPASM programını kullanacağız.

Pic Assembly: Pic'in yapmasını istediğimiz işlemleri, pic'in anlaması için, kısa komutlardan oluşturulmuş bir dildir. Bu dil ile yazılan program ASM dosyası olarak kaydedilir daha sonra MPASM ile HEX dosyası haline getirilip, pic içine yazılır.

Bir pic'i programlamak için yapmamız gerekenler sırası ile şöyledir:

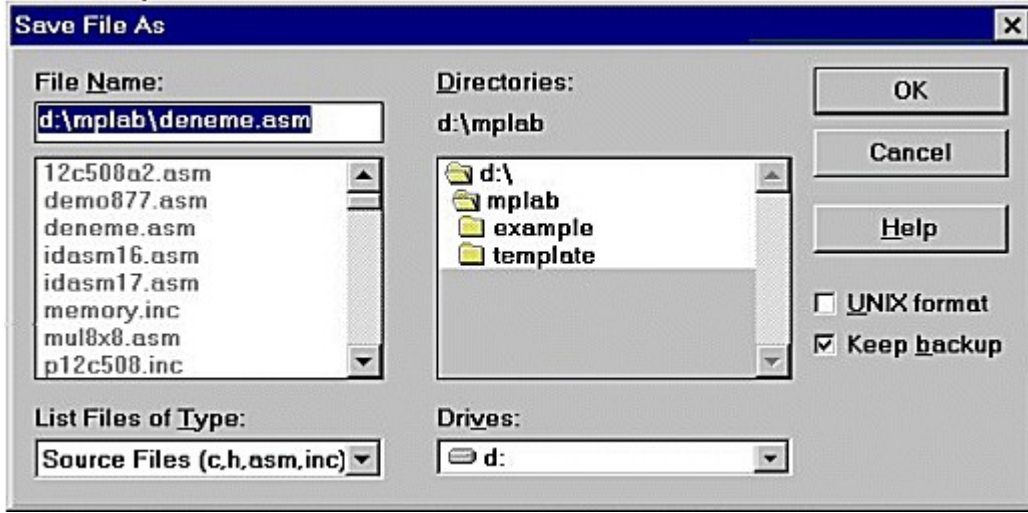
- 1-) Bilgisayarımıza gerekli Software'i yüklemek
- 2-) Pic Assembly dilinin genel kurallarını öğrenmek
- 3-) 16F84'ün 35 komut setini öğrenmek
- 4-) Program yazıp ASM olarak kaydetmek
- 5-) Programı hatasız yazıp Compile edebilmek
- 6-) Compile edilip HEX haline gelmiş dosyayı EASYPIC ile pic'e aktarmak.

SOFTWARE: Microchip firmasının sitesinden ücretsiz olarak indirilen program MPLAB v5.20, file manager içinde MP520ful 10.659 Kb Application dosyası şeklinde görülür. Bu dosyanın yeni versiyonunun adı ve boyu değişik olabilir. Burada zip file olarak verilen dosya budur ve internetten indirebilirsiniz.

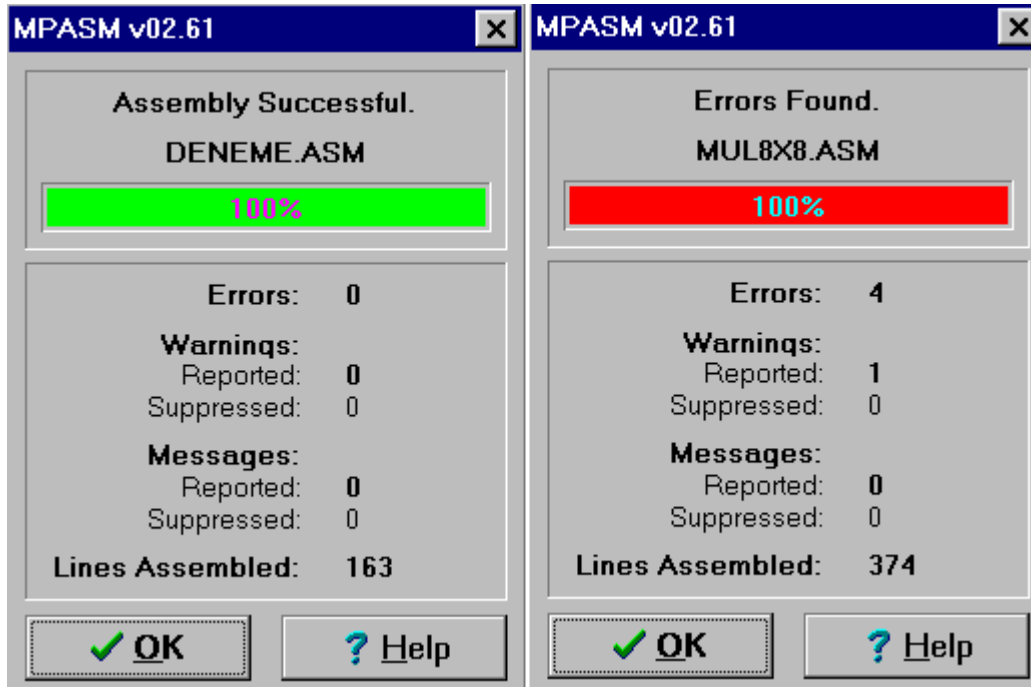
Program kendisini C /Program Files/MPLAB /altında install eder. Burada oluşan Mpasmwin ve Mplab application dosyalarını kısayol olarak masa üstüne taşıyınız.



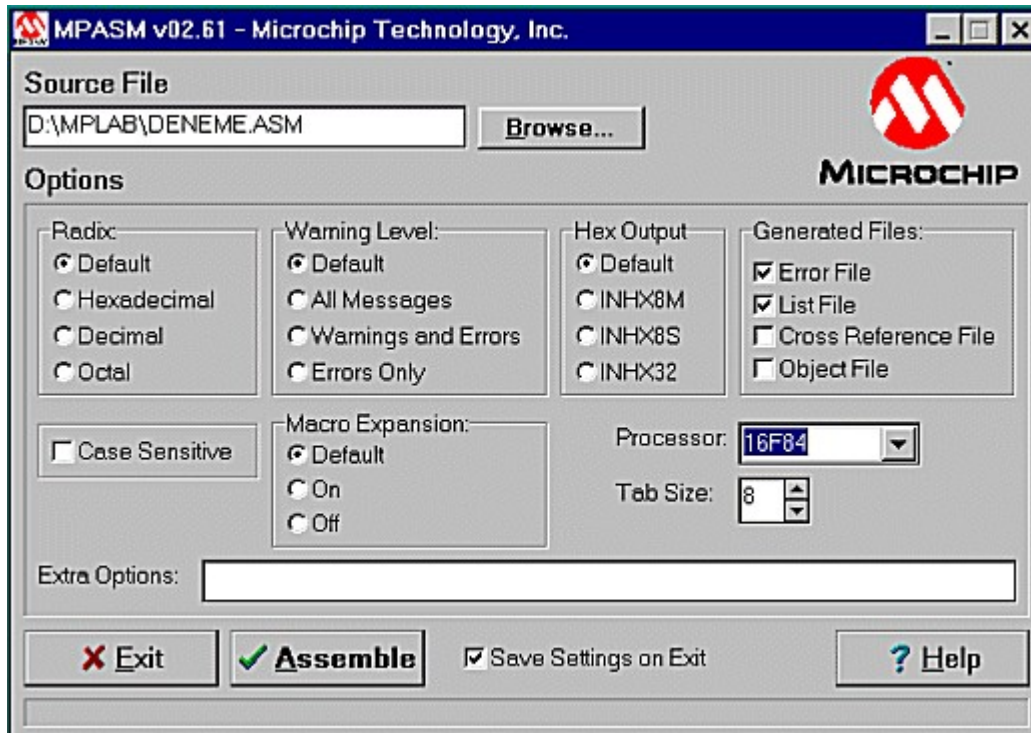
Masaüstünden MPLAB'ı tıklayarak File /new altından yeni bir sayfa açınız. Bu sayfaya artık pic assembly komutlarını kullanarak programınızı yazabilirsiniz. Bu programı sonu ASM olacak şekilde kaydetmeniz gerektiğini tekrar söyleyelim. Programdan



çıkmadan Project/build ile dosyayı HEX haline getirebilirsiniz ve ASM dosyası ile aynı yerde, aynı isimde HEX dosyanız açılmış olur. Bu işlem sırasında yazdığınız program doğru ise, başarılı olduğuna ilişkin, hatalı ise hatalı olduğuna dair bir mesaj verilir. MPLAB içinde yazıp ASM olarak kaydettiğiniz dosyayı ,masa üstünden MPASM'ı açarak ve dosyanızı "source file name" penceresinden bularak compile edebilirsiniz. Bu sırada başarılı ise yeşil, başarısız ise kırmızı uyarı işareti verilir.

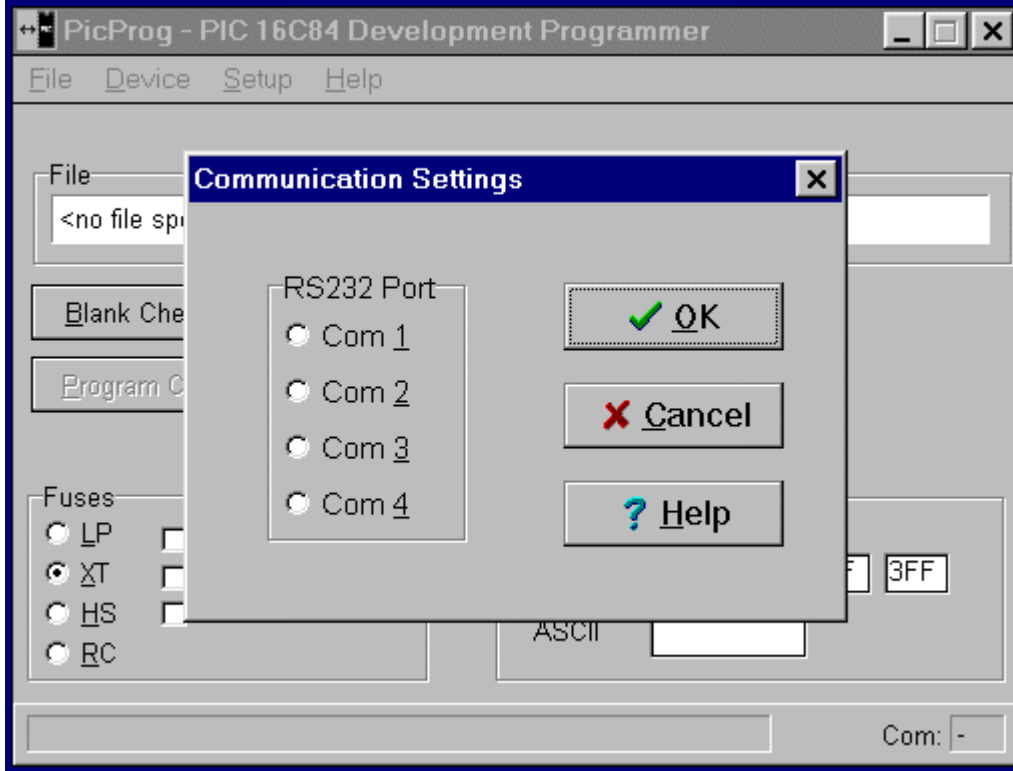


Başarısız ise , oluşturulmuş bulunan err dosyasını editörde açarak,hatanızın nerede olduğunu görebilirsiniz.Bu iş için şekilde görülen dosyalar içinde error file kısmının işaretli olması gerekir.

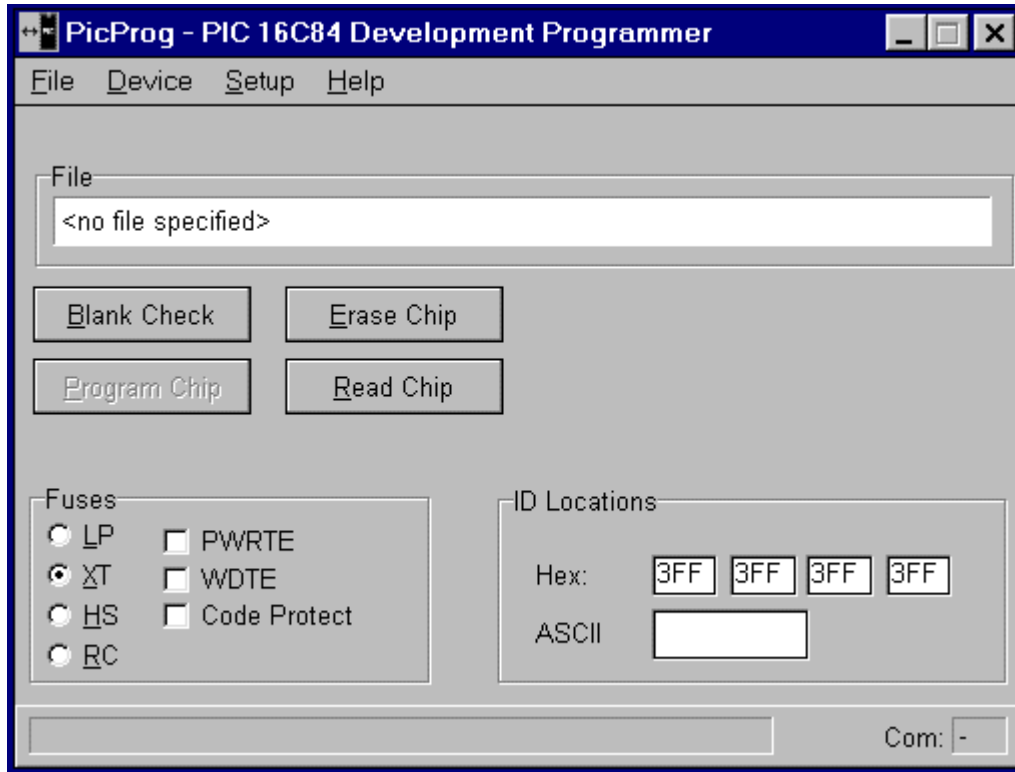


Başarı ile compile edilmiş dosyayı, pic'e aktarmak için kullanılan program pigprog dur, 16f84 için başka programlar da mevcuttur. Programlayıcılar da paralel veya seri porttan çalışabilirler. Biz burada pigprog ve seri porttan çalışan Easypic programlayıcısını kullanacağız.

Pigprog küçük bir dosyadır bu dosyayı uygun bir klasör altına koyabilirsiniz, bu program 331 Kb lık bir application dosyasıdır. Bu dosyayı da masa üstüne taşıyınız. Elinizde bir HEX file varsa pigprog penceresinden o file'ı bulunuz ve önce Com port ayarını (bu ayar bir kez yapılırsa hep aynı kalır) yapınız. Daha sonra gerekli ayarları yaparak pic'i programlayınız.



Com port genelde bilgisayarınızın konfigürasyonuna bağlı olarak Com1 veya Com 2 dir. Konu ile ilgili bilgi daha detaylı olarak ileride tekrar verilecektir. Bizim şimdi esas konumuz program yapmayı öğrenmek olduğu için bu konuyu burada noktıyoruz.



Pigprog program penceresinden hex file bulunur. xt işaretlenir ve program butonuna basılır. Programlama hardware'i ne ileride tekrar döneceğiz.

Pic Assemblby genel kuralları:

Genel kurallara geçmeden önce son olarak şu konuya da dikkatinizi çekelim. Bir program yazıldıktan sonra compiler'in bunu tanınması için ASM olarak kaydedilen bu dosya, compiler'in yani mpasmwin'in bulunduğu direktörde olmalıdır. Eğer program yazımında inc dosyası kullanılmış ise o da burada olmalıdır. Inc dosyası program içinde yazılı font ile klasör altında bulunmalıdır, aksi halde compiler tarafından tanınmaz.

PROGRAM FILES / MPLAB/

Pigprog
mplab
mpasmwin
deneme.asm
deneme.hex
deneme. Err
deneme.lst
16f84.inc

gibi bir düzen ortaya çıkar.

Noktalı Virgöl:

; bir program yazılırken derlenmeyen satırların başına konur , bizi veya programı inceleyen bilgilendirecek bir bilgiyi ; işaretinden sonra yazarız ,yine bu işareten sonra çeşitli şekiller ve süslemeler yapabiliriz.

```
;=====
; Bir deneme programıdır TA2FR
;=====
; -----00000000000000000000-----
```

program içine bir satır yazıldıktan sonra satır yanına açıklayıcı bilgi (;) işaretinden sonra yazılır.

```
SAYAC EQU h'0D' ; h'0D ' adresinde sayac diye bir register tanımla
        BSF STATUS,5 ; bank 1 ' e geç
```

Burada da compiler noktalı virgül den sonraki açıklamaları görmez, bu sebeple buraya istediğimiz açıklamaları yazabiliriz.

AKIŞ DİYAGRAMLARI :

Pic Assembler ile bir program yazmadan evvel, pic'in hangi adımlarla programı işleyeceğini planlamamız gerekmektedir.Bu planlama işlemi akış diyagramları ile yapılır.Akış diyagramı işlenecek komutların sırasıdır.Uzun ve karmaşık programlarda, akış diyagramları, hangi seviyeden programın ayrılacağını ve hangi komutları yerine getireceğini, sonra tekrar nereye döneceğini göstermesi nedeni ile faydalıdır. Akış diyagramlarının kendine has sembolleri de vardır.

Bir program yazılırken 3 blok halinde yazılır.Bu bloklar:

1-) Etiket

2-) Komut (Equate)

3-) Adres

bloklarıdır.

ETİKET : Etiketler iki tiptir. Birinci tipi adres atanmış etiketlerdir, ikinci tipi ise adres atanmayan etiketlerdir. Tüm etiketler birinci blok 'a yazılır. Adres atanmış etiketler EQUATE komutu ile kullanılır veya INC. Dosyası sayesinde tanımlanırlar. Bu etiketler sayesinde tanımlanan registerler, program içinde tekrar tanımlamaya gerek kalmadan kullanılır

PORTB EQU 0X06

STATUS EQU 0X03

Burada PORTB ve STATUS Etikettir. EQU komutu ile tanımlanan bu registerler, daha sonra kullanıldığında, hangi adreste olduğu otomatik olarak pic tarafından bilinir. Bir diğer etiket çeşidi ise adres atanmayan etiketlerdir. Adres atanmayan etiketler, pic 'in çalışması sırasında döneceği program seviyesini göstermekte kullanılır. Bu etiketler programcı tarafından yaratılırlar.

BASLA

START

DÖNGÜ

Gibi etiketler bu tip etiketlerdendir. Bir kısım adres atanmış etiket de, programcı tarafından yaratılır.

SAYAC

Etiket gibi etiketler bu çeşit etikettir. Adres atanmış bu etiketler RAM alanına yazılır. Bu etiketleri INC. Dosyası içine ilave ederek Equate tanımlarını yapmadan INC. dosyası ile de kullanılabiliriz.

ETİKET KURALLARI :

- 1-) Etiketler birinci sütuna yazılır.
- 2-) Etiketler de İngilizce harf karakterleri kullanılır.
- 3-) Assembly komutundan etiket olmaz
- 4-) Etiket en fazla 31 karakter olabilir.
- 5-) Etiket ile program içinde kullanımı aynı karakter özellikleri göstermelidir.
- 6-) Etiket Harf veya alt çizgi ile başlar.
- 7-) Etiket içinde alt çizgi, rakam, soru işareti bulunabilir.

ETİKET	KOMUT	ADRES
STATUS	EQU	H'03'
TRISB	EQU	H'86'
SAYAC1	EQU	H'0D'
START		
	CLRF	PORTB
	BSF	STATUS,5
	GOTO	START

Burada start etiketi programcı tarafından yaratılmıştır ve bir adresi yoktur, bu etikete adres Assembler tarafından atanır ve bunu biz bilemeyiz.

Bir program yazılırken küçük veya büyük harf seçimi programcıya kalmıştır. Kullanılan karakterler etiket ve adres kısmında aynı olduktan sonra problem çıkmaz. Etiket te PORTB yazıp adres kısmında Portb yazamayız. Bu anlatılanlar EQU komutu kullanıldığında ve register tanımlarının programcı tarafından yapıldığında geçerlidir. Eğer INC. Dosyası kullanılacaksa dosya içi karakterler ile programcının yazı karakterleri aynı olmalıdır. Yani INC. dosyası içinde bir etiket büyük harf ile yazılı ise, programcı program içinde bu etiket kullanmak gerektiğinde büyük harf kullanmalıdır.

BİNARY ve HEXADECİMAL SAYILAR :

Bilgisayar sistemlerinde sık kullanılan ikili (binary) ve 16 lı (hexadecimal) sayı sistemlerine de kısaca göz atalım:

0 ve 1 kullanarak sayıları ifade etmek mümkün müdür ? Günlük hayatta en çok kullanılan 10 tabanına göre aritmetiktir. Burada 10 sembol ile sayılar oluşturulur bunlar :

1 2 3 4 5 6 7 8 9 0 sembolleridir. Bu semboller ile ifade edilen bir çokluğu yalnızca iki sembolle veya 16 sembollü bir sistemle de ifade edebiliriz. İki sembol söz konusu ise Binary 16 sembol söz konusu ise Hexadecimal aritmetik ten bahsedilir. Binary sistemin sembolleri

0 1

Hexadecimal sistem sembolleri ise:

0 1 2 3 4 5 6 7 8 9 A B C D E F dir.

10 lu sistemdeki bir sayıyı binary sayı sisteminde yazmak için, bize kolaylık sağlayan şu tabloyu göz önünde tutalım.

$$2^0 = 1$$

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8$$

$$2^4 = 16$$

$$2^5 = 32$$

$$2^6 = 64$$

$$2^7 = 128$$

$$2^8 = 256$$

10 lu sistemdeki bir sayıyı binary 'e çevirirken, busayıyı oluşturan ,ikinin katları şeklinde yazılabilen sayıları büyükten küçüğe doğru toplarsınız, burada ikinin her üs sayısı kullanılmalıdır.Bir üs şeklinde ifade edilemiyorsa 0X şeklinde yazılır.Örnek :

255 sayısını iki tabanına göre yazalım.

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 sayılarının toplamı olarak yazılır bu da:

$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$ olarak yazılabilir.

Burada dikkat edilmesi gereken nokta kımızı olarak gösterilen 2 nin tüm üs lerinin sıra ile kullanılma zorunluğudur, bu sıralama büyükten küçüğe doğru olacaktır.Çarpım dan evvel olan katsayılar ikili sistemde yalnız 0 veya 1 olabilir,burada hepsi birdir.Sonuç olarak da 255 sayısının binary karşılığı,bastaki katsayılar yanyana konarak bulunur.

11111111 (8 tane 1 dir).

ÖRNEK 2 :

129 sayısını aynı şekilde yazalım :

128 + 1 dir

Burada 128 den başlayarak ikinin üs 'sü şeklinde tüm üsleri kullanarak 129 sayısını elde edelim.

$$1X2^7 + 0X2^6 + 0X2^5 + 0X2^4 + 0X2^3 + 0X2^2 + 0X2^1 + 1X2^0$$

Binary karşılığı" 10000001"olarakbulunur.Bu şekilde 8 bit ile 256 ya kadar olan sayıları binary olarak yazabiliriz bunun yetmediği hallerde 12 bitle $2^{12} = 4096$ veya 16 bitle $2^{16} = 65536$ ya kadar sayıları ifade edebiliriz.

HEXADECİMAL SİSTEM :

16 Sembol setinden oluşan bir sistemdir.Elemanları:

0 1 2 3 4 5 6 7 8 9 A B C D E F dir.

Burada decimal 10 sayısına A karşılık olarak gelmektedir, F de 15'e karşılık olur.0 ile birlikte 16 sembol içeren bu sayı sistemi hexadecimal sayı sistemidir,bu singeler 12 li sayı sisteminde $X=10$ ve $\Sigma=11$ dır.

Bir sayının Hexadecimal karşılığını bulmak için, binary de yaptığımız işlemi yaparız.

$$16^0=1$$

$$16^1=16$$

$$16^2=256$$

$$16^3=4096$$

$$16^4=65536$$

$$16^5=1.048.576$$

Binary sistemde 8 basamaklı 255 sayısını, hexadecimal sistemde iki basamak ile elde ederiz.

256 sayısı hexadecimal olarak

$1X16^2 + 0X16^1 + 0X16^0$ demektir"100 "olarak yazılır.

255 sayısı için

$15X16^1 + 15X16^0$ burada 15. karakter F olduğu için " 255 "decimal " FF "Hexadecimal

olur.

EQUATES : EQU komutu program yazımının başında INC. dosyası kullanmadığımızda registerler adreslerinin tanımı için kullanılır.EQU

komutu ile register'in bir sabitten olusan adresi tanımlanır veya bizim yarattığımız bir file register için bir adres atanmasında kullanılır.

Bir sabit veya bir isim Equ komutu ile tanımlanabilir.

MAX EQU3; 3 rakamı nın adı MAX olsun,-bit yönlendirmeli komutla kullanıldığında bu sabit 3.bit'e karşılık olur-.

FLAGSEQUH'09'; h'09' da flgs diye bir file register olsun

BSFFLAGS,MAX; flags registerinin 3.bitini 1 yap

SABİTLER : Bir register adresini tanımlamada kullanılan bir haxadecimal sayıdır.En çok MOVLW komutu ile kullanılır ve bir sabiti Write registere yazar.Sabitlerle kullanılan komutlar ileride verilecektir.Desimal sayılar da sabit olarak kullanılır.

ORG : Program başlangıç adresini göstermek için veya interrupt alt program başlangıcını göstermek için kullanılır.

END : Program komutlarının sona erdiğini Assembler'e gösterir,olmazsa derlemede hata mesajı verilir.

SAYILAR :

Hexadecimal sayılar birkaç şekilde yazılabilir

0X06

6

06

06h

h'06'

Biz program yazarken**H'FF'** şeklindeki ifadeyi kullanacağız.

Binary sayılar**B** ile ifade edilirler**B '11111111'** tarzında yazılırlar.

Decimal sayılar **D** ile ifade edilirler**D'255'** şeklinde yazılırlar.

Pic 16F84'ün RAM belleğinin iki bank'tan ibaret olduğunu söylemiştik. Bunlar Bank 0 ve Bank 1'dir, değişik piclerde başka banklar da mevcuttur. Bir banktaki registerleri kullanmak için o bank'a geçmek gerekmektedir. Bank değiştirme işlemleri STATUS registeri vasıtasıyla yapılır.

STATUS REGISTER

Status register 8 bittir. Bu register'in 5. bitinin değiştirilmesi ile 16F84 de bank değiştirme işlemi gerçekleştirilir. 16F84'de Bank bitleri Status registerin 5. ve 6. Bitidir, 6. bit 0 kalmalıdır. Yalnız 5. bit'in değiştirilmesi bank değiştirmek için yeterlidir. Tüm reset hallerinde 5. ve 6. bit 00 olur ve Pic bank 0'da bulunur.

6. bit 0 5.bit 0 ise Bank 0
6. bit 0 5.bit 1 ise Bank 1 olur

Status register bitlerini değiştirmek için kullanılan komut BSF ve BCF komutlarıdır.

```
BSF STATUS,RP0
BSF STATUS,5
BSF H'03',RP0
BSF H'03',5
```

Komutları ile 5.bit 1 yapılır ve bank 1'e geçilir. 1 yapılan 5. biti 0 yapmak için ise BCF komutunu kullanırız.

```
BCF STATUS,RP0
BCF STATUS,5
BCF H'03',RP0
BCF H'03',5
```

"STATUS , RP0" "H03', RP0" ifadelerinde ki STATUS register adresi ve RP0 INC. Dosyasında tanımlıdır. INC dostası kullanılmaz ise bunlar EQU komutu ile tanımlanmalıdır. RP0 yerine, bit değeri olan 5 kullanıldığında bunu ayrıca tanımlamak gerekmez.

PORTLARIN GİRİŞ / ÇIKIŞ YAPILMASI

Portların giriş veya çıkış olarak yönlendirilmesi için TRIS registeri kullanılır. PORTA TRISA ile PORTB TRISB registeri ile yönlendirilir. TRIS registeri 8 bittir. Her bir bit bir I/O kapısına karşılıktır. PORTB nin 8 I/O kapısı vardır, fakat PORTA da 5 I/O kapısı mevcuttur bu nedenle PORTA için TRIS registerinin ilk 5 bit'i kullanılır, diğer bitler 0 olarak kalır.

PORT A ve PORTB kapıları ya tüm olarak veya tek tek bit olarak giriş, çıkış yapılabilir.

TRIS register bitleri :

1 ise kapı GİRİŞ
0 ise kapı ÇIKIŞ olarak yönlendirilir

PORTA çıkışlarını kontrol eden TRISA register bitleri, aşağıdaki gibi ise çıkışlar şu anlama gelir:

BİT	7	6	5	4	3	2	1	0
DEĞER	-	-	-	1	0	1	0	1
SONUÇ	-	-	-	G	Ç	G	Ç	G

PORTB çıkışlarını kontrol eden TRISB register bitleri, aşağıdaki gibi ise çıkışlar şu anlama gelir.

BİT	7	6	5	4	3	2	1	0
DEĞER	1	1	0	1	0	1	0	1
SONUÇ	G	G	Ç	G	Ç	G	Ç	G

PORTA ve PORTB çıkışlarının hepsini birden çıkış yapmak için kullanılan komut

"CLRF" komutudur.

CLRF TRISA tüm PortA kapılarını 0 yapar sonuçta tüm kapılar çıkış olur.

CLRF TRISB tüm PortB kapılarını 0 yapar sonuçta tüm kapılar çıkış olur.

PORT A veya PORTB çıkışlarının tümünü farklı giriş veya çıkış olarak yönlendirmek istersek karşılık gelen TRISA ve TRISB register bitlerinin tümüne farklı 0 veya 1 değeri vermemiz gerekir. Böylece o bitlere denk gelen kapılar giriş veya çıkış olarak yönlendirilir. Bunun için kullanılan komutlar şöyledir:

MOVLW B'11010101' ; write registere değerleri yaz

MOVWF TRISB ; write register değerlerini trisb ye yaz

BSF

BCF

Gibi komutlar da kullanılabilir. Bu komutlarla istenen bitleri tek tek 0 veya 1 yapabiliriz.

BSF (bit set file) TRISA veya TRISB nin ilgili bitini 1 yaparak karşılık olan A veya B port kapısını giriş yapar.

BCF (bit clear file) TRISA veya TRISB nin ilgili bitini 0 yaparak karşılık olan A veya B port kapılarını çıkış yapar.

BSF TRISA, 0 ; trisa'nın ilk biti 1 olur ve porta ilk bit giriş olur.
BCF TRISB, 1 ; trisb'nin 2. biti 0 olur ve portb nin ikinci biti çıkış olur.

P1c 16F84 komut seti (instruction set) 35 komuttan oluşmaktadır. Bu komutlar değişik şekilde sınıflandırılabilir. Komut setini öğrenmeden önce komutlarla kullanılan bazı terimleri görelim.

k bir sabit veya bir etiket'i ifade eder. Sabit hexadecimal, decimal, binary olabilir. Bir sabit şu şekilde olabilir.

H'FF'
B'11111111'
D'255'

k tanımladığımız bir etiket de olabilir.

GOTO SAYAC

d desimal veya destination ifadesidir. Destination sabitin yazıldığı yerdir. Write register veya file register olabilir.

d 0 = W write register

d 1 = F file register

Biz 0 ve 1 ifadeleri yerine W ve F ifadelerini kullanacağız. INC dosyası içinde W ve F adreslenmiştir fakat bu dosya kullanılmazsa bile MPASM W ve F kullanılmasına izin verir.

b binary ve bit ifadesidir. Bit oriented komutlarla kullanılır. bir desimal sayıdır ve bir registerin ilgili bitini ifade eder.

BSF STATUS , 5 **KOMUTLAR**

1. Yalnız kullanılan komutlar
2. **k** ile kullanılan komutlar
3. **f** ile kullanılan komutlar
4. **f+bit** ile kullanılan komutlar
5. **f+d** ile kullanılan komutlar

1. byte oriented
2. bit oriented
3. literal instructions
4. control instructions

1. data tanımlayan ve yer değiştiren komutlar
2. register muhtevasını değiştiren komutlar
3. program akışını kontrol eden komutlar
4. mikrokontrolleri kontrol eden komutlar
5. logic komutlar
6. aritmetik komutlar
7. boş komut

yalnız	k ile	f ile	f+b ile	f+d
CLRW	MOVLW	MOVWF	BCF	MOVF
RETURN	RETLW	CLRF	BSF	COMF
CLRWDI	ANDLW	TRIS	BTFSC	DECF
OPTION	IORLW		BTFSS	INCF
SLEEP	XORLW			RLF
RETFIE	ADDLW			RRF
NOP	SUBLW			DECFSZ
	GOTO			INCFSZ
	CALL			ANDWF
				IORWF
				XORWF
				ADDWF
				SWAPF
				SUBWF

Mnemonic, Operands	Description	Cycles	14-Bit Opcode				Status Affected	Notes
			MSb		LSb			
ADDWF f, d	Add W and f	1	00	0111	ffff	ffff	C,DC,Z	1,2
ANDWF f, d	AND W with f	1	00	0101	ffff	ffff	Z	1,2
CLRF f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRWF -	Clear W	1	00	0001	0000	0011	Z	
COMF f, d	Complement f	1	00	1001	ffff	ffff	Z	1,2
DECf f, d	Decrement f	1	00	0011	ffff	ffff	Z	1,2
DECFSZ f, d	Decrement f, Skip if 0	1(2)	00	1011	ffff	ffff	None	1,2,3
INCF f, d	Increment f	1	00	1010	ffff	ffff	Z	1,2
INCFSZ f, d	Increment f, Skip if 0	1(2)	00	1111	ffff	ffff	None	1,2,3
IORWF f, d	Inclusive OR W with f	1	00	0100	ffff	ffff	Z	1,2
MOVF f, d	Move f	1	00	1000	ffff	ffff	Z	1,2
MOVWF f	Move W to f	1	00	0000	1fff	ffff	None	
NOP -	No Operation	1	00	0000	0xx0	0000	None	
RLF f, d	Rotate left f through carry	1	00	1101	ffff	ffff	C	1,2
RRF f, d	Rotate right f through carry	1	00	1100	ffff	ffff	C	1,2
SUBWF f, d	Subtract W from f	1	00	0010	ffff	ffff	C,DC,Z	1,2
SWAPF f, d	Swap nibbles in f	1	00	1110	ffff	ffff	None	1,2
XORWF f, d	Exclusive OR W with f	1	00	0110	ffff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF f, b	Bit Clear f	1	01	00bb	bfff	ffff	None	1,2
BSF f, b	Bit Set f	1	01	01bb	bfff	ffff	None	1,2
BTFSC f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff	None	3
BTFSS f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff	None	3
LITERAL AND CONTROL OPERATIONS								
ADDLW k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT -	Clear Watchdog Timer	1	00	0000	0110	0100	TO,PD	
GOTO k	Go to address	2	10	1kkk	kkkk	kkkk	None	
IORLW k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW k	Move literal to W	1	11	00xx	kkkk	kkkk	None	
RETFIE -	Return from interrupt	2	00	0000	0000	1001	None	
RETLW k	Return with literal in W	2	11	01xx	kkkk	kkkk	None	
RETURN -	Return from subroutine	2	00	0000	0000	1000	None	
SLEEP -	Go into standby mode	1	00	0000	0110	0011	TO,PD	
SUBLW k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

INCLUDE DOSYASI

Bir program yazılırken tüm registerleri tek tek tanımlamak oldukça zordur,bu nedenle include dosyası denilen bu tanımların içinde bulunduğu ve assembler tarafından tanınan bir dosya kullanırız.Assembler tarafından tanınması için program girişinde tanımlanır.MPLAB kullanıldığında içindeki metin editörü PFE kullanılır burada yazılan program ASM uzantılı olarak kaydedilir.Bu klasörde INCLUDE dosyası da bulunuyorsa MPASM dosyayı derlerken,INCLUDE dosyası içindeki bilgileri okur ve derlemeyi yapar.DOS editoründe yazılan bir program içinde tanımlanan INCLUDE dosyası ile klasör içindeki dosya fontları birbirine uymaz ise programlamada hata gösterir.genelde de bu hata olur.halbuki MPLAB . kendi editoründe yazılan programda harf hatası yoksa dosyayı tanır.

STATUS REGISTER:

Bank 0 ve Bank 1 de 03H ve 83H adreslerinde bulunan 8 bit bir register olan STATUS register bulunur. Status register , aritmetik logic ünitenin durumunu,reset halini ve bank seçimini belirleyen bitler içerir.Status register bir komut hedefi olduğunda Z,DC,C bitleri etkilenebilir ve daha sonra bu bitlere tekrar yazılmaz.Bu bitler duruma göre 0 veya 1 olurlar.TO ve PD bitleri yazılamaz bitlerdir.

CLRF STATUS komutu ile ilk 3 bit 0 olur TO ve PD değişmez,Z 1 olur sonuç ' 000DD1DD' olur,burada D =değişmez anlamındadır.Durum değişikliği olmasın istendiğinde ; BCF,BSF,SWAPF,MOVWF komutları kullanılabilir,bu komutlarla STATUS register bitleri etkilenebilir.

D değişmez anlamındadır W = Write R = Read

Bit nosu	7	6	5	4	3	2	1	0
Oku-Yaz	R-W	R-WR	R-W	Read	Read	R-W	R-W	R-W
Bit adı	IRP	RP1	RP0	TO	PD	Z	DC	C
CLRF	0	0	0	D	D	1	D	D
Power-on Reset	0	0	0	1	1	B	B	B
Bit özelliği	İndirekt adreslemede kullanılan register bank seçimi biti 16F84 de 0 dir.	Direkt adreslemede kullanılan register bank seçim biri16F84 de 0'dir.	Direkt adreslemede kullanılan register bank seçim biri16F84 de bank 0 da 0 bank 1 de 1 dir.	Time - out bitPower-up CLRWD T - SLEEP Komutu sonrası 1WD T timeOut olduğunda ise 0 dir.	Power-Down bitPower-up ve CLRWD T Sonrası 1 SLEEP komutu ile 0 dir.	Zero bitAritmetik ve lojik operasyon sonrası sonuç 0 ise 1Sıfır değilse 0 dir	Digit Carry bitTaşma ödünç bitişlem sonrası 4.bitteTaşma varsa 1taşma yoksa 0 dir	Carry bit Taşma ödünç bitişlem sonrası 7.bitteTaşma varsa 1taşma yoksa 0 dir

0 ve 1 . bitler C ve DC bitleri ADDLW ADDWF RLF RRF gibi komutları ilgilendirir.Bunlarla ilgili daha geniş bilgi ileride verilecektir.

SUBWF : File register ve Destination ile (F,D) kullanılır File Registerden ,write registeri çıkarır ve sonucu file veya write registerine yazar.

SUBWF = FILE REGISTER - WRITE REGISTER W veya F'e yazılır.

Sonuca göre Status register içinde bulunan Z ve C flaglarının durumu değişir.

Sonuç	Z Flag	C Flag
-------	--------	--------

Pozitif	0	1
0	1	1
Negatif	0	0

SUBLW : Bir sabit ile kullanılan komuttur. Bir sabitten write register içeriğini çıkarır ve sonucu WRITE registere yazar.

SUBLW = SABİT - WRITE REGISTER W registere yazılır

Bu sabit binary, hexadecimal veya decimal olabilir,sabitten daha evvel write registere yazılan değeri çıkardığımızda,çıkan sonuca göre Z ve C flag durumu değişir.

Sonuç	Z Flag	C Flag
Pozitif	0	1
0	1	1
Negatif	0	0

Bu komutların kullanılması ile flag durum değişikliğini kontrol ederek,programda kullanırız.