MİKROİŞLEMCİ NEDİR?
MİKRODENETLEYİCİ NEDİR?
MİKRODENETLEYİCİLER HAKKINDA GENEL BİLGİLER
PIC PROGRAMLAMA İÇİN NEYE İHTİYACIMIZ VAR

MİKROİŞLEMCİ NEDİR?

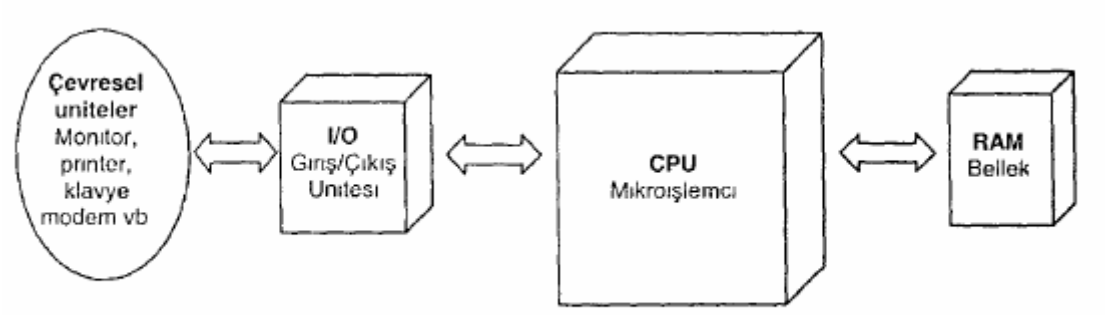
Günümüzde kullanılan bilgisayarların özelliklerindenbahsedilirken duyduğunuz 80386, 80486, Pentium-ll, Pentium-lll birer mikroişlemcidir (Microprocessor). Mikroişlemciler bilgisayar programlarının yapmak istediği tüm işlemleri yerine getirdiği için, çoğu zaman merkezi işlem ünitesi (CPU- Central Processing Unit) olarak da adlandırılır. PC adını verdiğimiz kişisel bilgisayarlarda kullanıldığı gibi, bilgisayarla kontrol edilen sanayi tezgahlarında ve ev aygıtlarında da kullanılabilmektedir. Bir mikroişlemci işlevini yerine getirebilmesi için aşağıdaki yardımcı elemanlara ihtiyaç duyar. Bunlar:

1. lnput (Giriş) ünitesi.

2. Output (Çıkış) ünitesi.

3. Memory (Bellek) ünitesi.

Bu üniteler CPU chip'inin dışında, bilgisayarın ana kartı üzerinde bir yerde farklı chiplerden veya elektronik elemanlardan oluşur. Aralarındaki iletişimi ise veri yolu (Data bus), adres yolu (Address bus) denilen iletim hatları yapar.

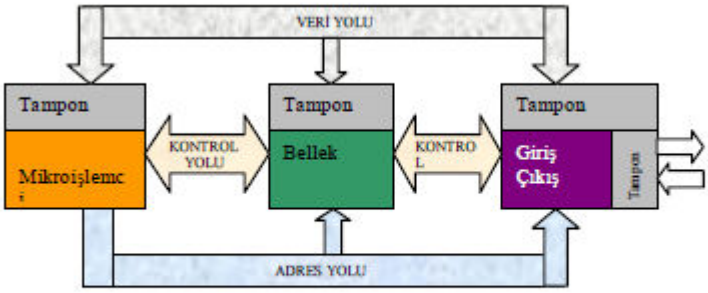


Bir mikroişlemci sisteminin temel bileşenlerinin blok diyagramı

Intel, Cyrix, AMD, Motorola mikroişlemci üreticilerden birkaçıdır, Günümüzde mikroişlemciler genellikle PC adını verdiğimiz kişisel bilgisayarlarda kullanılmaktadır.

Merkezi İşlemci Biriminde İletişim Yolları

Mikroişlemcide işlenmesi gereken komutları taşıyan hatlar yanında, işlenecek verileri taşıyan hatlar ve kesme işlemlerini kontrol eden sinyalleri taşıyan hatlar bulunur. İşlenecek verileri işlemciye yollamak veya işlenen verileri uygun olan birimlere aktarmak için aynı hatlardan faydalanılır. Tüm bu yollara iletişim yolları adı verilir.



Mikroişlemcili sistemde birimler arasında iletişimi sağlayan yollar

1. **Veri Yolu**

Merkezi işlem biriminden bellek ve giriş / çıkış birimlerine veri göndermede ya da bu birimlerden işlemciye veri aktarmada kullanılan hatlar, veri yolu olarak isimlendirilir. Veri yolu genişliği, mikroişlemcinin yapısı, mikroişlemci kaydedici genişliği ve kullanılan kelime uzunluğu ile doğrudan ilişkilidir. 8-bitlik mikroişlemcilerde veri yolu 8 hattı içerirken, 16-bitlik işlemcilerde 16 hattı içerir. Mikroişlemciye işlenmek üzere iletilen veriler veri yolu üzerinden iletildiği ya da mikroişlemcide işlenen veriler veri yolu üzerinden ilgili birimlere yollandığı için, veri yolunda iki yönlü iletişim mümkün olmaktadır.
Bellekte bulunan ve CPU tarafından işlenmesi istenilen veriler, veri yolu üzerinden iletilir.

2. **Adres Yolu**

Verinin alınacağı (okunacağı) veya verinin gönderileceği (yazılacağı) adres bölgesini temsil eden bilgilerin taşınmasında kullanılan hatlar, adres yolu olarak isimlendirilir. Adres yolu, tek yönlüdür ve paralel iletişim sağlayacak yapıdadır.
CPU'da işlenen verilerin, bellekte saklanması veya diğer elemanlara gönderilmesi gerekebilir. Bu durumda, verinin saklanacağı veya gönderileceği yerin adresi, mikroişlemci içerisindeki PC yardımı ile adres yolu üzerine yerleştirilir. Yerleştirilen bilginin temsil ettiği adres bölgesi dahili bellekte olabileceği gibi, harici bellekte de olabilir. Yerleştirilen bilginin kodu çözülerek ilgili adres bölgesi bulunur ve bulunan adres bölgesindeki veri, veri yoluna konur. Yapılan bu işlemlerin düzgün ve kontrollü olarak gerçekleştirilmesinden, zamanlama ve kontrol birimleri görevlidir.

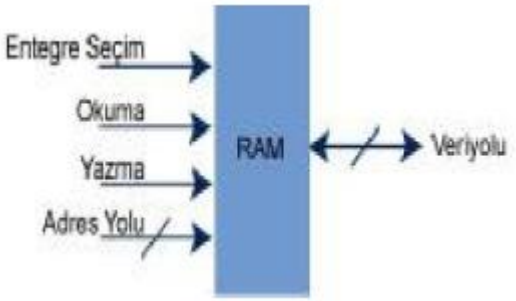
3. **Kontrol Yolu**

Mikroişlemcili sistemde bulunan birimler arasındaki ilişkiyi düzenleyen sinyallerin iletilmesi amacıyla kullanılan hatlar **kontrol yolu** olarak adlandırılır. Her bir mikroişlemciye ait komut kümesi ve belirli amaçlar için kullanılan sinyallerin farklı olması sebebiyle, her mikroişlemcide farklı sayıda hattı içeren kontrol yolu bulunabilir.

Bellek

1. **RAM Bellekler**

Mikroişlemcinin çalışması esnasında her türlü değişkenin üzerinde yer aldığı ve geçici işlemlerin yapıldığı birimi RAM belleklerdir. Özel bir sıra takip etmeden herhangi bir adrese erişildiği için rastgele erişimli bellek (Random Access Memory) – RAM olarak isimlendirilir. Ayrıca yığın olarak adlandırılan ve mikroişlemci programlarının çalıştırılması esnasında çeşitli alt-programlar kullanıldıkça geri dönüş adreslerinin, içeriklerinin değişmesinin istenmediği kaydedici içeriklerinin saklandığı bellek bölgeside yine RAM'da birimlerinde yer alır.



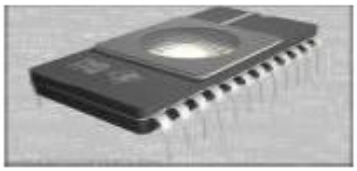
RAM tipi entegreler hem yazmada hem okumada kullanıldıklarından CPU, bu entegreleri kontrol ederken okuma R (Okuma) ve W (yazma) sinyalleri göndermesi gerekir

2. ROM Bellekler

Yalnız okunabilen birimlere ROM (Read Only Memory) bellekler denir. Bu bellek elemanlarının en büyük özelliği enerjisi kesildiğinde içindeki bilgilerin silinmemesidir. ROM belleklere bilgiler üretim aşamasında yüklenir. Kullanıcıların bellek içindeki bilgileri değiştirmesi mümkün değildir.

3. Programlanabilir ROM Bellek (PROM)

PROM'lar bir kez programlanabilir. Bu bellek elemanı entegre şeklindedir. Kaydedilen bilgiler enerji kesildiğinde silinmez. Üzerine program kodlarını veya verileri yazmak için PROM programlayıcı cihazlara ihtiyaç vardır. Bu bellek elemanının yapısında küçük sigorta telleri bulunur. Bellek hücrelerinde, hepsi sağlam durumda bulunan sigortalar "1"i temsil eder. Yazılacak olan bilginin bit düzeninde "0"lara karşılık gelen hücredeki sigorta, küçük bir elektrik akımı ile aktarılır. Bu şekilde PROM programlanır.



Şekil 1.15: Bir EPROM Bellek

4. Silinebilir Programlanabilir ROM Bellek (EPROM)

EPROM'lar bellek hücrelerine elektrik sinyali uygulanarak programlama işlemi yapılır. Kaydedilen bilgiler enerji kesildiğinde silinmez. EPROM içindeki programın silinmemesi için cam pencereli kısım ışık geçirmeyen bantla örtülmelidir. Eprom belleğe yeniden yazma işlemi yapmak için EPROM üzerindeki bant kaldırılıp ultraviyole altında belirli bir süre tutmak gerekir. Bu şekilde içindeki bilgiler silinebilir. (Şekil 1.15) Böylece tekrar programlanabilir hâle gelen ürün tekrar tekrar farklı programların denenmesi ve cihazın çalıştırılması için kullanılabilir. Silme işlemi esnasında belirli şartlara dikkat edilmemesi (gereğinden fazla süre UV ışığa maruz kalmak, yüksek ışık şiddetine sahip UV ampul kullanmak gibi) hâlinde silinebilme ömrü kısalan entegreler bir süre sonra kullanılamaz (silinemez) hâle gelmektedir.

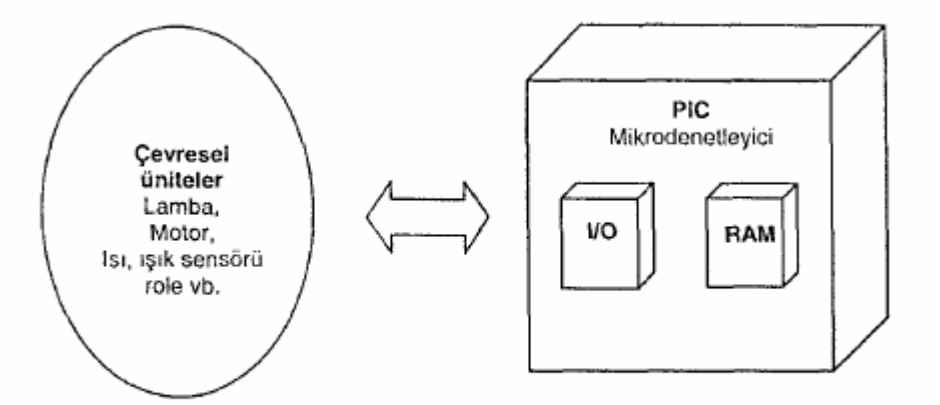
5. Elektriksel Yolla Değiştirilebilir ROM Bellek (EEPROM)

Üzerindeki bilgiler, elektriksel olarak yazılabilen ve silinebilen bellek elemanlarıdır. EEPROM' u besleyen enerji kesildiğinde üzerindeki bilgiler kaybolmaz. EEPROM'daki bilgilerin silinmesi ve yazılması için özel silme ve yazma cihazlarına gerek yoktur. Programlayıcılar üzerinden gönderilen elektriksel sinyalle programlanır. EEPROM'la aynı özellikleri taşıyan fakat yapısal olarak farklı ve daha hızlı olan, elektriksel olarak değiştirilebilir ROM'lara flash bellek denir.

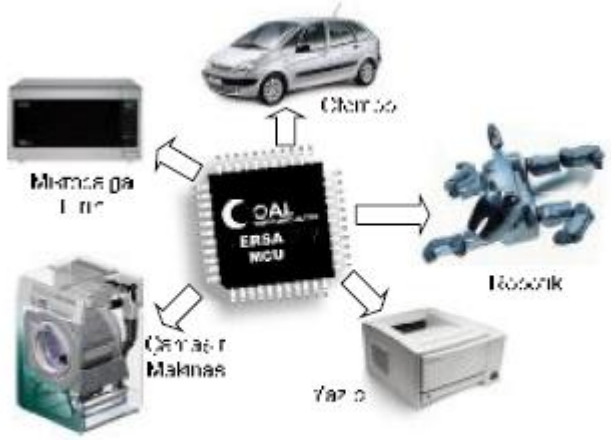
MİKRODENETLEYİCİ NEDİR?

Bir bilgisayar içerisinde bulunması gereken temel bileşenlerden RAM, I/O ünitesinin tek bir chip içerisinde üretilmiş biçimine mikrodenetleyici (Microcontroller) denir. Bilgisayar teknolojisi gerektiren uygulamalarda kullanılmak üzere tasarlanmış olan mikrodenetleyiciler, mikroişlemcilere göre çok daha basit ve ucuzdur. Günümüz mikro denetleyicileri otomobillerde, kameralarda, cep telefonlarında, fax-modem cihazlarında, fotokopi, radyo, TV, bazı oyuncaklar gibi sayılamayacak kadar pek çok
alanda kullanılmaktadır.

Günümüz mikrodenetleyicileri birçok chip üreticisi tarafından üretilmektedir. Her firma ürettiği chip'e farklı isimler vermektedir. Örneğin Microchip firması ürettiklerine PIC adını verirken, Intel'in ürettiği ve 1980'lerin başında piyasaya sürdüğü 8051, bazen MCS-51 olarak da adlandırılır.



Bir mikrodenetleyici sisteminin temel bileşenlerinin blok diyagramı



Şekil 1.16: Mikrodenetleyicilerin kullanım alanları

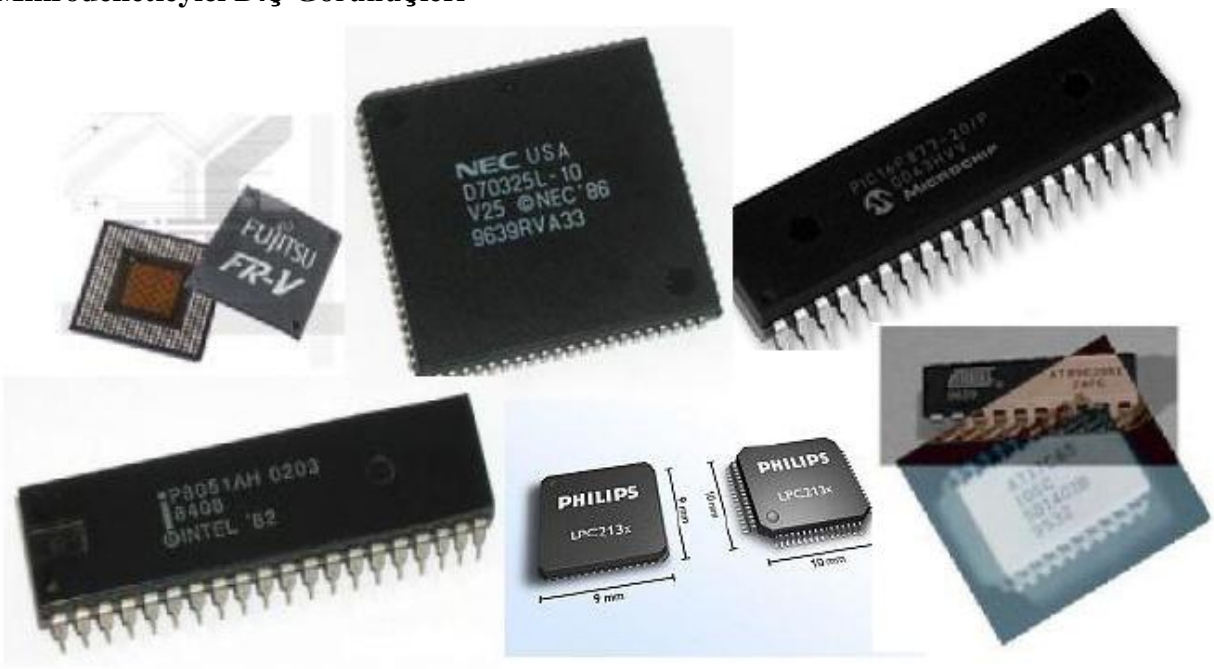
Neden Mikroişlemci Değil de Mikrodenetleyici Kullanılıyor?

Mikro işlemci ile kontrol edilecek bir sistemi kurmak için en azından şu üniteler bulunmalıdır; CPU, RAM, I/O ve bu ünitelerin arasındaki veri alış verişini kurmak için DATA BUS (data yolu) gerekmektedir. Elbette bu üniteleri yerleştirmek için baskılı devreyi de unutmamak gerekmektedir. Mikrodenetleyici ile kontrol edilecek sistemde ise yukarda saydığımız ünitelerin yerine geçecek tek bir chip (Mikrodenetleyici) ve bir de devre kartı kullanmak yetecektir. Tek chip kullanarak elektronik çözümler üretmenin maliyetinin daha düşük olacağı kesindir. Ayrıca da kullanım ve programlama kolaylığı da ikinci bir avantajıdır. İşte yukarda saydığımız nedenlerden dolayı son zamanlarda bilgisayar kontrolü gerektiren elektronik uygulamalarda mikrodenetleyici kullanmaya eğilimin artmasının haklılığını ortaya koyuyor.

MİKRODENETLEYİCİLER HAKKINDA GENEL BİLGİLER

Neredeyse her mikroişlemci (CPU) üreticisinin ürettiği birkaç mikrodenetleyicisi bulunmaktadır. Bu denetleyicilerin mimarileri arasında çok küçük farklar olmasına rağmen aşağı yukarı aynı işlemleri yapabilmektedirler. Her firma ürettiği chipe bir isim ve özelliklerini birbirinden ayırmak için de parça numarası vermektedir. Örneğin Microchip ürettiklerine PIC adını, parça numarası olarak da 12C508, 16C84, 16F84, 16C711 gibi kodlamalar verir.Intel ise ürettiği mikrodenetleyicilere MCS-51 ailesi adını vermektedir. Genel olarak bu adla anılan mikrodenetleyici ailesinde farklı özellikleri bulunan ürünleri birbirinden ayırt etmek için parça numarası olarak da 8031AH, 8051AH, 8751AHP, 8052AH, 80C51 FA gibi kodlamalar kullanılmaktadır.

Mikrodenetleyici Dış Görünüşleri



Şekil 1.17:Mikrodenetleyici çeşitleri

Neden PIC?

Bilgisayar denetimi gerektiren bir uygulamayı geliştirirken seçilecek mikrodenetleyicinin ilk olarak tüm isteklerinizi yerine getirip getirmeyeceğine, daha sonra da maliyetinin düşüklüğüne bakmalısınız. Ayrıca, yapacağınız uygulamanın devresini kurmadan önce seçtiğiniz mikrodenetleyicinin desteklediği bir yazılım üzerinde simülasyonunu yapıp yapamayacağınızı da dikkate almalısınız.

Yukarda saydığımız özellikleri göz önüne aldığımızda Microchip'in ürettiği PIC'leri kullanmak en akılcı bir yol olduğunu görülmektedir. İşte, bu kitapta PIC'leri ele alınmamızın nedenlerini şöyle sıralayabiliriz.

- Yazılımın Microchip'ten veya internetten parasız olarak elde edilebilmesi.
- Çok geniş bir kullanıcı kitlesinin bulunması.
- PIC'lerin çok kolaylıkla ve ucuz olarak elde edilebilmesi.
- Elektronikle hobi olarak uğraşanların bile kullanabildikleri basit elemanları kullanarak yapılan donanımla programlanabilmesi.
- Çok basit reset, clock sinyali ve güç devreleri gerektirmeleri.

PIC, adını İngilizce'deki Peripheral Interface Controller cümlesindeki kelimelerin baş harflerinden almış olan bir mikrodenetleyicidir. Eğer bu cümleyi Türkçe'ye çevirirsek, çevresel üniteleri denetleyici arabirim gibi bir anlam çıkacaktır. PIC gerçekten de çevresel üniteler adı verilen lamba, motor, role, ısı ve ışık sensörü gibi 1/0 elemanların denetimini çok hızlı olarak yapabilecek şekilde dizayn edilmiş bir chip'tir. RISC mimarisi adı verilen bir yöntem kullanılarak üretildiklerinden bir PIC'i programlamak için kullanılacak olan komutlar oldukça basit ve sayı olarak da azdır. 1980'lerin başından itibaren uygulanan bir tasarım yöntemi olan RISC (Reduced lnstruction Set Computer) mimarisindeki temel düşünce, daha basit ve daha az komut kullanılmasıdır. Örneğin P1C16F84 microdenetleyicisi toplam 35 komut kullanılarak programlanabilmektedir.

Neden P1C16F84 ?

Bu kitapta programlanması ve örnek uygulamaları verilen P1Cm 16F84 serisi olmasının en önemli nedeni: P1C16F84 (veya PICİ6F84A) mikrodenetleyicisinin program belleğinin flash teknolojisi ile üretilmiş olmasıdır.

Flash memory teknolojisi ile üretilen bir belleğe yüklenen program, chip'e uygulanan enerji kesilse bile silinmez. Yine bu tip bir belleğe istenirse yeniden yazılabilir. Flash bellekler bu özellikleri ile EEPROM bellekler ile aynı görünmektedirler. Gerçekten de Flash ile EEPROM bellek aynı şeylerdir. Ancak bazı üreticiler tarafından EEPROM belleğe Flash ROM da denilmektedir.

Flash belleğe sahip olan P1C16F84i programlayıp ve deneylerde kullandıktan sonra, silip yeniden program yazmak PIC ile yeni çalışmaya başlayanlar için büyük kolaylıktır. Böylece işe yeni başlayanlar yaptıkları programlama hataları nedeniyle chip'i atmak zorunda kalmayacaklardır. Gerçi EPROM program memorisi olan chip'lere de yeniden yazmak mümkündür ama, bu durumda bir EPROM silici cihazına ihtiyaç vardır. Bir silici cihaz bulunsa bile programı bellekten silmek için en azından 10-15 dk beklemek zorunda kalınacaktır. işte P1C16F84ün bu özelliği mikrodenetleyici kullanmaya yeni başlayanlar için ideal bir seçenektir.

P1C16F84'ü seçmemizin ikinci nedeni de, programlama donanımının çok ucuz ve kullanışlı olması ve hatta çoğu meraklı elektronik kullanıcı tarafından bile üretilebilmesidir. Kitabın Ekler bölümünde adresini verdiğimiz firmanın ürettiği programlayıcı donanımı ve yazılımı ödemeli olarak istenebilmesi Türkiye'deki kullanıcılar için çok büyük bir avantajdır.

P1C16F84'ü programlamak için öğrendiğiniz her şeyi diğer PIC 16/17 mikrodenetleyicilerinin uygulamalarında da 'kullanabilmeniz, yapılan seçimin doğruluğunu göstermektedir.

PIC Programlamak İçin Gerekli Donanımlar

- PC bilgisayar
- Bir metin editörünün kullanılmasını bilmek
- PIC assembler programı
- PIC programlayıcı donanımı
- PIC programlayıcı yazılımı
- Programlanmış PIC'in çalışmasını görmek için PIC deneme kartı

Assembly program kodlarını kolayca yazmak, doğru ve hızlı bir şekilde PIC' in program belleğine göndermek için, bilgisayara ihtiyaç vardır. Bir metin editörü kullanarak yazılan program kodları derlendikten sonra PIC'e gönderilmesi gerekir. Program kodlarının PIC'e yazdırma işlemi paralel veya seri porta bağlanan PIC programlama kartı ile yapılır. Bu işleri yapabilmek için gereken donanımlar; görsel bir işletim sistemi (Windows, Linux), basit bir editör (Edit, Notpad, Word gibi) 1GHz CPU, 256 MB RAM, 40 GB sabit disk ve CDROM sürücüsü olmalıdır.

Metin Editörü

Assembly dili komutlarını yazıp bir metin dosyası oluşturmak için EDİT veya NotPad gibi bir editörü kullanabilmeniz gerekir. İsterseniz ASM uzantılı metin dosyalarınızı yazabileceğiniz PFE editörünü de kullanabilirsiniz. Bu editörün hem DOS hem de WİNDOVVS altında çalışan versiyonlan bulunmaktadır ve PIC konusunda destek veren bir internet sitesinden alınmıştır. Ekler bölümünde adını verdiğimiz firma da bu programı disket içerisinde sunmaktadır.

Assembler Programı

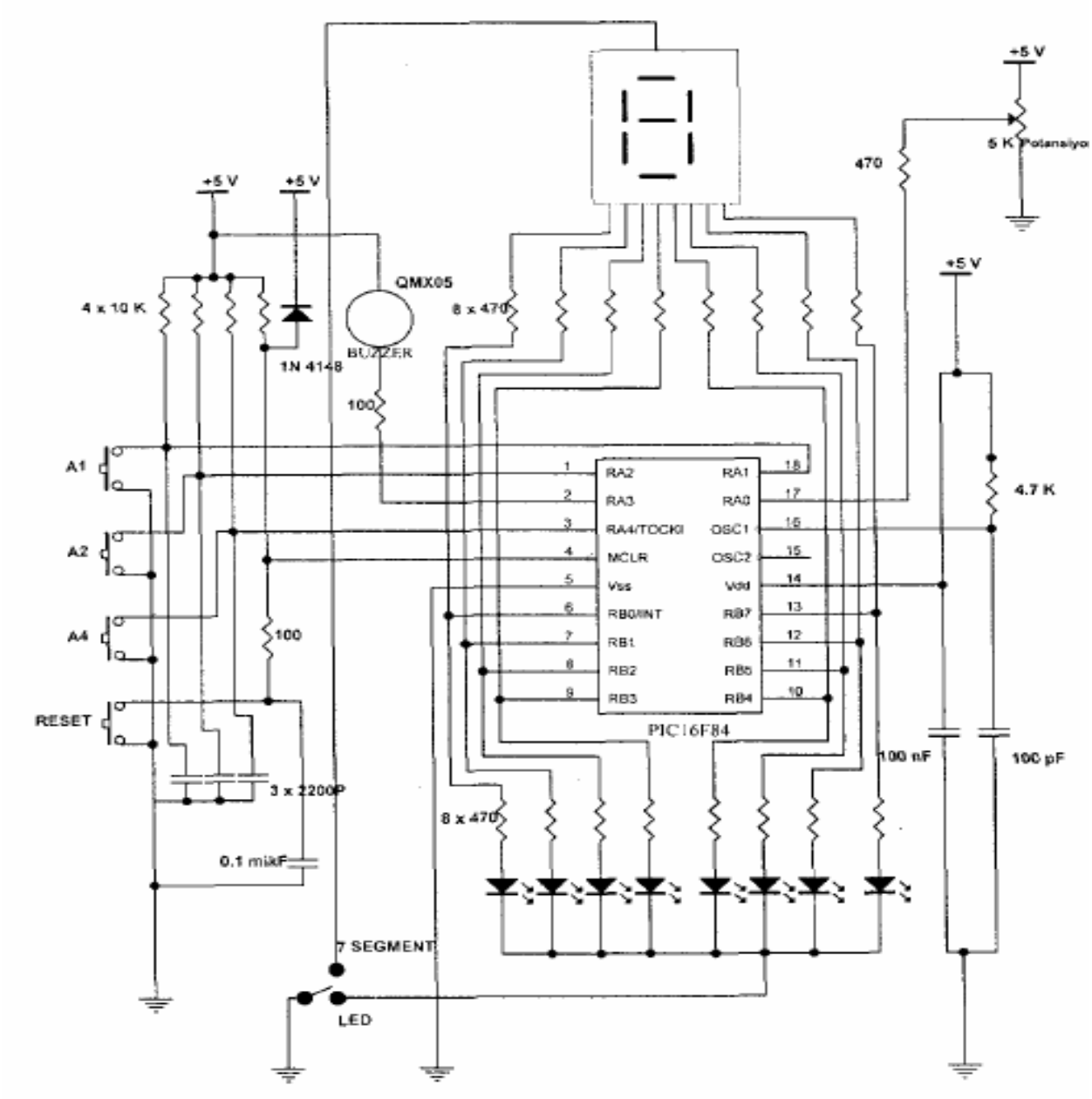
PIC Assembly dili adı verilen ve toplam 35 komuttan oluşan programlama dilini bu kitapta öğreneceksiniz. Bu komutları basit bir editörde yazabiliyoruz. Ancak, İngilizce'deki bazı kelimelerin kısaltmasından oluşan bu dilin komutlarını P1C'in anlayabileceği makine diline çeviren bir programa ihtiyacımız vardır. Bu programa assembler adını veriyoruz. Text dosyası biçiminde kaydedilmiş olan assembly dili komutlarını makine diline çeviren MPASM'nin hem DOS altında hem de WINDOWS altında çalışan versiyonu bulunmaktadır.

PIC Programlayıcı Yazılımı

MPASM tarafından derlenerek makine diline dönüştürülmüş assembly programı kodlarının PIC'e yazdırılmasında kullanılan bir programa gereksinim vardır. Programlayıcı yazılımları, PIC'i programlamak için kullanılan elektronik karta bağımlıdır

Programlanmış PIC'i Deneme Kartı

Programladığınız PIC'İ breadboard üzerinde kendi kurduğunuz devre de deneyebileceğiniz gibi şekilde görülen özel bir deneme kartı üzerinde de deneyebilirsiniz. Başlangıç ve orta düzey PIC programlayıcılara hitap etmek üzere geliştirilen bu kart üzerinde deneme yapmak, breadbord üzerinde devre kurmaktan çok daha kolaydır. PIC'in port çıkışlarındaki sinyalleri izlemek amacıyla 8 tane LED, bir tane de 7 segmentli LED yerleştirilmiştir. Kart üzerindeki, butonlar, potansiyometre ve LED'ler aracılığıyla farklı şekilde programlanan PIC'lerin kolayca denenmesini sağlanır. Bu kartın yapısı ve kullanılması hakkında Ekler bölümünde geniş bilgi bulacaksınız.



PIC DONANIM ÖZELLİKLERİ

PIC ÇEŞİTLER
PIC'LERİN GÖRÜNÜŞÜ
PIC BELLEK ÇEŞİTLERİ

PIC ÇEŞİTLERİ

Microchip ürettiği mikro denetleyicileri 4 farklı gruba (Genellikle aile diye adlandırılır.) ayırarak isimlendirmiştir. PIC ailelerine isim verilirken kelime boyu (Word lenght) göz önüne alınmıştır. Şimdi kelime boyunun ne anlama geldiğine bakalım. Microprocessor veya mikro denetleyiciler kendi içlerindeki dahili veri saklama alanları olan registerleri arasındaki veri alış verişini farklı sayıdaki bitlerle yaparlar. Örneğin 8088 mikro işlemcisi chip içerisindeki veri

alış verişini 16-bit ile yaparken, Pentium işlemcileri 32-bit'lik verilerle iletişim kurarlar. Bir CPU veya MCU'nun dahili veri yolu uzunluğuna kelime boyu denir.

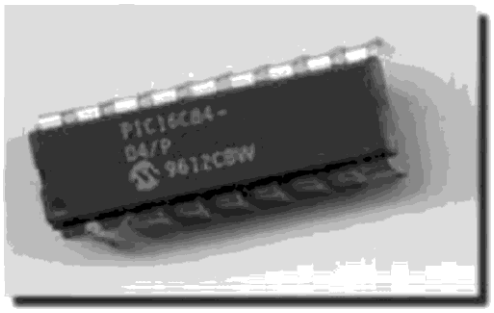
Microchip PICleri 12/14/16 bit'lik kelime boylarında üretmektedir ve buna göre aşağıdaki aile isimlerini vermektedir.

PIC16C5XX ailesi 12-bit kelime boyu,
PIC16CXXX ailesi 14-bit kelime boyu,
PIC17CXXX ailesi 16-bit kelime boyu,
PIC12CXXX ailesi 12-bit/1 4-bit kelime boyuna sahiptir.

Bir CPU veya MCUnun chip dışındaki harici ünitelerle veri alışverişini kaç bit ile yapıyorsa buna veri yolu bit sayısı denir. PIC'ler farklı kelime boylarında üretilmelerine rağmen harici veri yolu tüm PIC ailesinde 8-bittir. Yani bir P10, 1/0 portu aracılığı ile çevresel ünitelerle veri alışverişi yaparken 8-bit'lik veri yolu kullanır.

PIC programlayıcıları program kodlarını yazarken bir komutun kaç bit'lik bir kelime boyundan oluştuğuyla pek fazla ilgilenmezler. Seçilen bir chip'i programlarken uyulması gereken kuralları ve o chip'le ilgili özelliklerin bilinmesi yeterlidir. Bu özellikler PIC'in bellek miktarı, IO portu sayısı, A/D dönüştürücüye sahip olup olmadığı, kesme (interrupt) fonksiyonlarının bulunup bulunmadığı, bellek tipinin ne olduğu (Flash, EPROM,EEPROM vb.) gibi bilgilerdir. Bu özelliklerin en son değişikliklerini içeren güncel ve tam bir listesine microchip'in kataloglarından ulaşmak mümkündür.

PIC'LERIN DIŞ GÖRÜNÜŞÜ



PIC'ler çok farklı ambalajlarla piyasaya sunulmaktadırlar. Bu kitapta program örnekleri verilen PIC 16F84, 1 8-pinli DIP tipinde ambalajlanmış olanıdır. Standart DIP (Dual In-Line Package) olarak ambalajlanmış olan PIC'ler 0.3 inç genişliğindedir. Her iki tarafında 0.1 inç aralıklarla yerleştirilmiş 9 pin bulunmaktadır. P1C16F84 şekil-a da görülmektedir.Bu örneklerin dışında farklı pin sayısına sahip başka DITP ambalajlı PIC'ler de bulunmaktadır. Geniş bilgi için Microchip'in kataloglarına bakmanızı öneririz.

PIC BELLEK ÇEŞİTLERİ

Farklı özellikte program belleği bulunan PIC'ler Microchip firması tarafından piyasaya sürülmektedir. Bunlar:

- Silinebilir ve programlanabilir bellek (Erasable Programmable Memory-EPROM).
- Elektriksel olarak silinebilir ve programlanabilir bellek (Electrically Erasable Programmable Memory- EEPROM). FLASH bellek olarak da adlandırılır.
- Sadece okunabilir bellek (Read-OnIy Memory-ROM).

Her bir bellek tipinin kullanılacağı uygulamaya göre avantajları ve dezavantajları vardır. Bu avantajlar; fiyat, hız, defalarca kullanmaya yatkınlık gibi faktörlerdir.
EPROM bellek hücrelerine elektrik sinyali uygulayarak kayıt yapılır. EPROM üzerindeki enerji kesilse bile bu program bellekte kalır. Ancak silip yeniden başka bir program yazmak için ultra-viole ışını altında belirli bir süre tutmak gerekir. Bu işlemler EPROM silici denilen özel aygıtlarla yapılır. EPROM bellekli PIC'ler iki farklı ambalajlı olarak bulunmaktadır:

- Seramik ambalajlı ve cam pencereli olan tip, silinebilir olan tiptir.
- Plastik ambalajlı ve penceresiz olan tipler ise silinemez (OTP) tiptir.

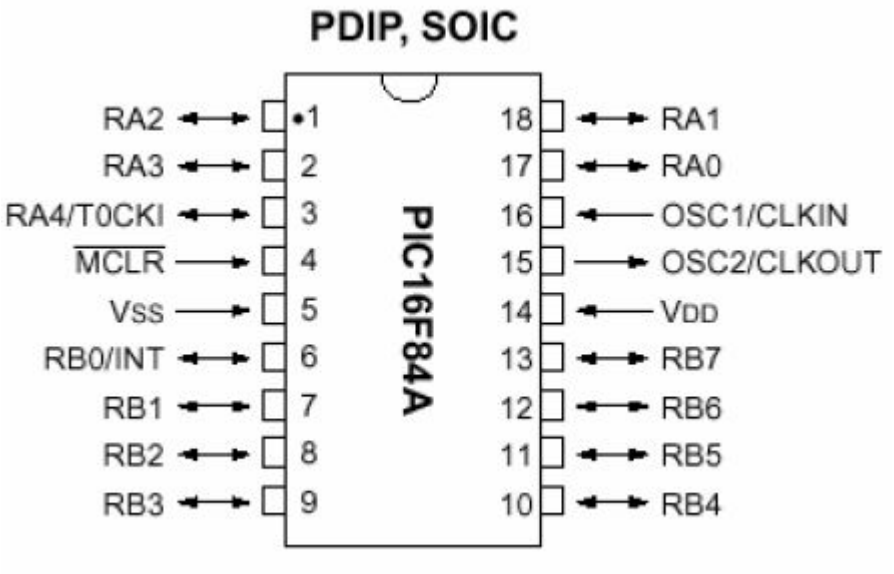
Seramik ambalajlı ve pencereli olan bellek içerisindeki programın silinmemesi için pencere üzerine ışık geçirmeyen bir bant yapıştırılır. Ultra-viole ışığı ile silinmesi istenildiğinde bu pencere açılır ve silici aygıt içerisinde belirli bir süre bekletilir. Plastik ambalajlı EPROM'lar ise programlandıktan sonra silinmesi mümkün değildir ve fiyatı silinebilen tipe göre oldukça ucuzdur. Silinemeyen tipe OTP (One Time Programmable - Bir defa programlanabilir) olarak adlandırılır.

EEPROM belleği bulunan bir PIC içerisine program yazmak için PIC programlayıcı vasıtasıyla elektriksel sinyal gönderilir. EEPROM üzerindeki enerji kesilse bile bu program bellekte kalır. Programı silmek veya farklı yeni bir program yazmak istendiğinde PIC programlayıcıdan elektriksel sinyal gönderilir. Bu tip belleğe sahip olan PIC'ler genellikle uygulama geliştirme amacıyla kullanılırlar. Microchip bu tip belleğe çoğu zaman FLASH bellek olarak da adlandırmaktadır. Fiyatları silinemeyen tiplere göre biraz pahalıdır. Bellek erişim hızları ise EPROM ve ROM'lara göre daha yavaştır. PIC 16C84 ve PIC 16F84'ler bu tip program belleğine sahiptir.

ROM program belleğine sahip PIC'lerin programları fabrikasyon olarak yazılırlar. EPROM ve EEPROM eşdeğerlerine nazaran fiyatları oldukça düşüktür. Ancak fiyatların düşüklüğünden dolayı gelen avantaj bazen çok pahalıya da mal olabilir. ROM bellekli PIC programlarının fabrikasyon olarak yazılması nedeniyle PIC'in elde edilme süresi uzundur. Programda oluşabilecek bir hatanın PIC'e program yazıldıktan sonra tespit edilmesi, eldeki tüm PIC'lerin atılmasına da neden olabilir. Bu tip PIC'ler çok miktarda üretilecek bir ürünün maliyetini düşürmek amacıyla seçilir. Program hataları giderilemediği için uygulama geliştirmek için uygun değildir. Microchip, ROM program bellekli PIC'lere parça numarası verirken "CR" (PIC 16CR62, PIC 16CR84 gibi) harfleri kullanılır.

P1C16F84'ÜN PIN GÖRÜNÜŞÜ

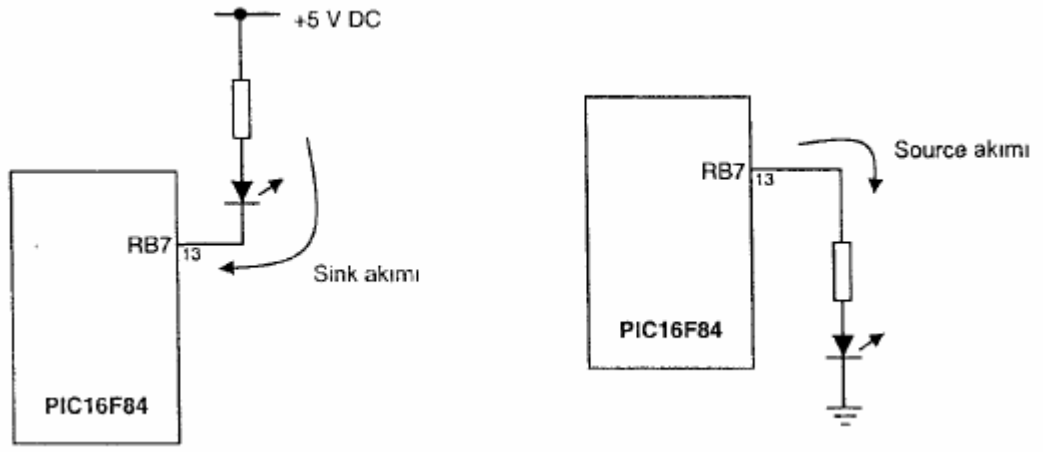
BESLEME GERİLİMİ
CLOCK UÇLARI VE CLOCK OSİLATÖRÜ ÇEŞİTLERİ



CMOS teknolojisi ile üretilmiş olan P1C16F84 çok az enerji harcar. Flash belleğe sahip olması nedeniyle clock girişine uygulanan sinyal kesildiğinde registerleri içerisindeki veri aynen kalır. Clock sinyali tekrar verildiğinde PIC içerisindeki program kaldığı yerden itibaren çalışmaya başlar. RAO-RA3 pinleri ve RBO-RB7 pinleri 1/0 portlarıdır. Bu portlardan girilen dijital sinyaller vasıtasıyla PIC içerisinde çalışan programa veri girilmiş olur. Program verilen değerlendirerek portları kullanmak suretiyle dış ortama dijital sinyaller gönderir. Dış ortama gönderilen bu sinyallerin akımı yeterli olmadığı durumda yükselteç devreleri (röle, transistör v.s) ile yükseltilerek kumanda edilecek cihaza uygulanır. Portların maksimum sink ve source akımları aşağıda verilmiştir. Bu akımlar genellikle bir LED sürmek için yeterli olduğundan, bu kitapta verilen uygulama devrelerinde herhangi bir yükseltme işlemi yapılmamıştır.

	IO pini
Sink akımı	25 ma
Source akımı	20 mA

Hatırlatma amacıyla sink ve source akımlarının ne olduğundan bahsedelim. Sink akımı, gerilim kaynağından çıkış potuna doğru akan akıma, source akımı ise 1/0 pininden GND ucuna doğru akan akıma denir.



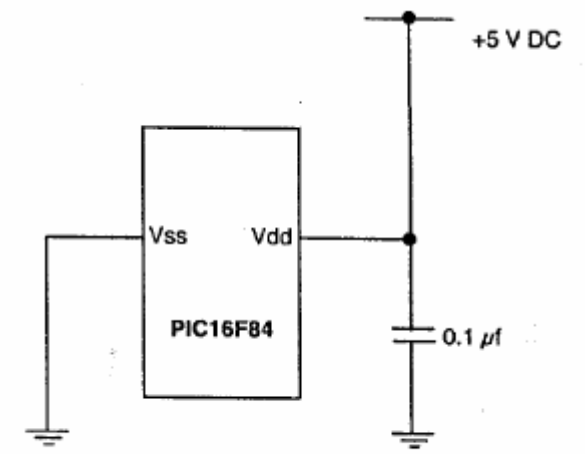
P1C16F84'ün çektiği akım, besleme gerilimine, clock girişine uygulanan sinyalin frekansına ve hC pinlerindeki yüke bağlı olarak değişir. Tipik olarak 4 MHz'lik clock frekansında çektiği akım 2 mA' kadardır. Bu akım uyuma modunda (Sleep mode) yaklaşık olarak 40 AA' e düşer. Bilindiği gibi CMOS entegrelerdeki giriş uçları muhakkak bir yere bağlanır.Bu nedenle kullanılmayan tüm girişler besleme geriliminin +5V luk ucuna bağlanmalıdır.

P1C16C84 ve P1C16F84 özellikleri tamamen aynısı olan PIC'lerdir. Her ikisi de FLASH belleğe sahip olmalarına rağmen Microchip ilk ürettiği EEPROM bellekli PICler " C harfi (C harfi CMOS'dan gelmektedir.) ile tanımlarken, son zamanlarda ürettiği EEPROM bellekli PIC'leri "E" harfi (FLASH) ile tanımlamaktadır.

Bizim bu kitapta örneklerini verdiğimiz programlar için her iki PIC de kullanılabilir. PICI6F84A ile P1C16F84 arasında da herhangi bir fark yoktur. PIC'i tanımlayan bu harf ve rakamlardan sonra yazılan 1O/P, 04/P clock girişine uygulanacak maksimum frekansı belirtir. Örneğin 10 MHze kadar frekanslarda P1C16F84-10/P kullanılırken, 4MHz'e kadar frekanslarda PIC16F84-04/P kullanılabilir.

BESLEME GERİLİMİ

P1C besleme gerilimi 5 ve 14 numaralı pinlerden uygulanır. 5 numaralı Vdd ucu +5 V'a, 14 numaralı Vss ucu da toprağa bağlanır. PIC'e ilk defa enerji verildiği anda meydana gelebilecek gerilim dalgalanmaları nedeniyle istenmeyen arızaları önlemek amacıyla Vdd ile Vss arasına 0.1 F lık bir dekuplaj kondansatörü bağlamak gerekir. PIC'ler CMOS teknolojisi ile üretildiklerinden çok geniş besleme gerilimi aralığında (2 - 6 V) çalışmalarına rağmen 5 V luk gerilim deneyler için ideal bir değerdir.



PIC16F84'e besleme geriliminin bağlanması

CLOCK UÇLARI ve CLOCK OSİLATÖRÜ ÇEŞİTLERİ

PIC belleğinde bulunan program komutlarının çalıştırılması için bir kare dalga sinyale ihtiyaç vardır. Bu sinyale clock sinyali denilir ve Türkçede "klok" olarak okunur. P1C16F84'ün clock sinyal girişi için kullanılan iki ucu vardır. Bunlar OSCİ(16 pin) ve OSC2 (15pin) uçlarıdır. Bu uçlara farklı tipte osilatörlerden elde edilen clock sinyalleri uygulanabilir. Clock osilatör tipleri şunlarıdır:

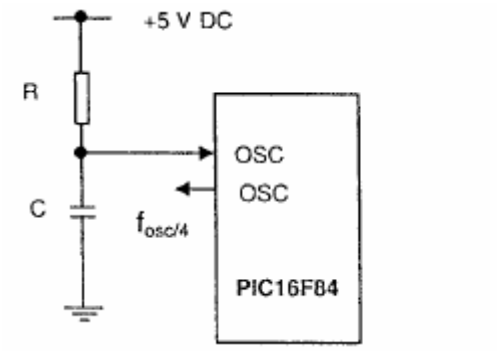
RC - Direnç/kondansatör (Resistor/Capacitor).
XT - Kristal veya seramik resonatör (Xtal).
HS - Yüksek hızlı kristal veya seramik resonatör (High Speed).
LP - Düşük frekanslı kristal (Low Power).

Seçilecek olan osilatör tipi P1C kontrol ettiği devrenin hız gereksinimine bağlı olarak seçilir. Aşağıdaki tablo hangi osilatör tipinin hangi frekans sınırları içerisinde kullanılabileceğini gösterir.

Osilatör tipi	Frekans sınırı
RC	0-4MHz
LP	5—200KHz
XT	100 KHz—4MHz
HS (-04)	4 MHz
HS(-10)	4—10MHz
HS (-20)	4 — 20 MHz

PIC'e bağlanan clock osilatörünün tipi programlama esnasında P10 içerisinde bulunan konfigürasyon bitlerine yazılmalıdır. Osilatör tipini belirten kodları (RC, XT, HS ve LP) kullanarak konfigürasyon bitlerinin nasıl yazılacağı programlama örnekleri verilirken detaylı olarak incelenecektir.

RC clock osilatörü, P1C kontrol ettiği elektronik devredeki zamanlamanın çok hassas olması gerekmediği durumda kullanılır. Belirlenen değerden yaklaşık %20 sapma gösterebilirler. Bir direnç ve kondansatörden oluşan bu osilatörün maliyeti oldukça düşüktür. OSCİ ucundan uygulanan clock frekansı R ve C değerlerine bağlıdır. Şekilde RC osilatörün clock girişine bağlanışı ve çeşitli R, C değerlerinde elde edilen osilatör frekansları örnek olarak verilmiştir.

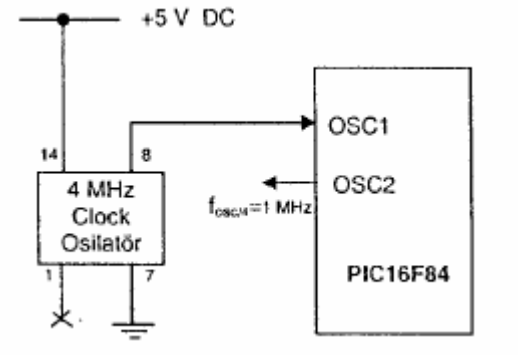


R	C	f_{osc} (yaklaşık)
10 K	20 pF	625 KHz
10 K	220 pF	80 KHz
10 K	0,1 µF	85 KHz

RC osilatörün PIC'e bağlantısı ve örnek R, C değerleri

OSC1 ucundan uygulanan harici clock frekansının 1/4'ü OSC2 ucunda görülür. Bu clock frekansı istenirse devrede kullanılan diğer bir elemanı sürmek için kullanılabilir.

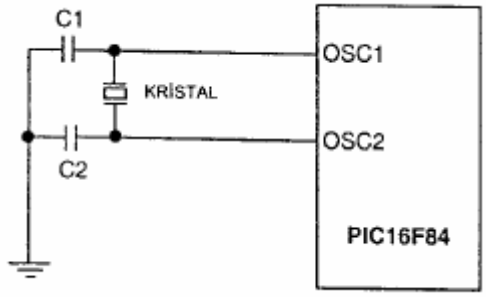
Kristal kontrollü clock osilatörleri zamanlamanın çok hassas olması gerektiğinde kullanılır. Bu tip clock osilatörleri metal bir kutu görünümündedir ve şekilde görülmektedir. Bu tip osilatörlere kondansatör bağlantısı gerekmez. PIC assembly programlama dili ile yazılan zaman geciktirme (Time delay) döngülerinde yapılacak hesaplamaları kolaylaştırmak için genellikle 4 MHz'lik kristal clock osilatörleri kullanılması tavsiye edilir. Bu durumda harici clock frekansı (OSC1) 4'e bölündüğünde, dahili clock frekansı 1 MHz olur (OSC2). Çoğu PIC assembly komutu bir komut saykılı süresinde (dahili clock) çalıştığından, bir komutun işlevini gerçekleştirme süresi 1 mikro saniye olur. Bu süre ise deneysel çalışmalar için oldukça uygundur.



Kristal clock osilatörün PIC'e bağlantısı

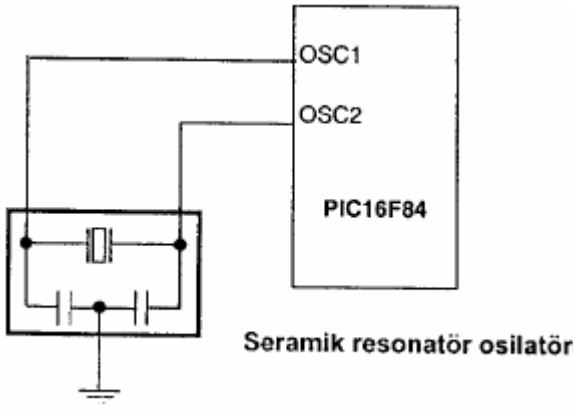
Kristal ve kondansatör kullanılarak yapılan osilatörler de zamanlamanın önemli olduğu yerlerde kullanılır. Kristal osilatörlerin kullanıldığı devrelerde kristale bağlanacak kondansatörün seçimine özen göstermek gerekir. Aşağıda hangi frekansta kaç (µF lık kondansatör kullanılacağını gösteren tablo görülmektedir.

OSİLATÖR TİPİ	FREKANS	KONDANSATÖR
LP	32KHz	33—68pF
	200KHz	15—47pF
	100KHz	47—lOOpF
XT	500 KHz	20 — 68 pF
	1MHz	15—68pF
	2MHz	15—47pF
	4MHz	15—33pF
HS	8MHz	15—47pF
	20MHz	15—47pF



Seçilen kondansatör değerlerinin yukarıdaki değerlerden yüksek olması, elde edilen kare dalgaların bozuk olmasına ve PIC çalışmamasına neden olur. C1 ve C2 kondansatörlerin değerleri birbirine eşit olmalıdır.

Seramik resonatörler, içerisinde kondansatörleri hazır bulunan osilatörlerdir. Fiyatları ucuz ve hassastırlar (+/ %1.3). Küçük bir seramik kondansatöre benzeyen resonatörlerin üç ucu vardır. Bu uçlardan ortadaki toprağa, diğer iki ucu da 0501 ve 05C2 uçlarına bağlanırlar. Hangi ucun OSC' ye bağlanacağı önemli değildir, her ikisi de bağlanabilir.

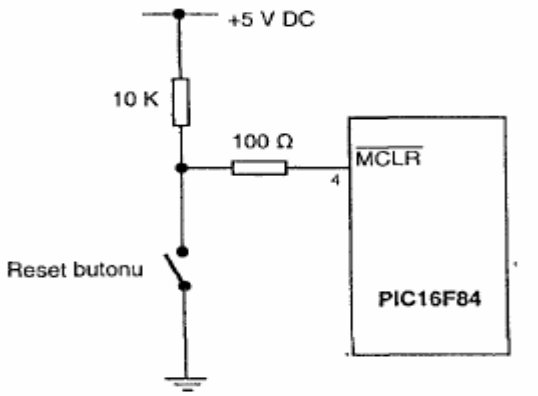


Seramik resonatör osilatörünün PICe bağlantısı

RESET UÇLARI VE RESET DEVRESİ

P1C16F84ün besleme uçlarına gerilim uygulandığı anda bellekteki programın başlangıç adresinden itibaren çalışmasını sağlayan bir reset devresi vardır. Bu reset devresi P10 içerisindedir ve "Power-on-reset" denir.

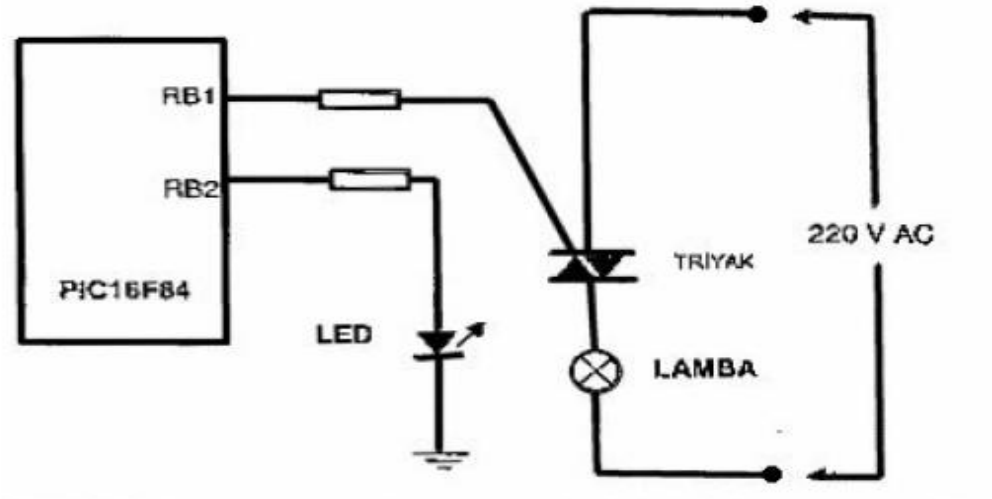
MCLR ucu ise kullanıcının programı kesip, kasti olarak başlangıca döndürebilmesi için kullanılır. P10m 4 numaralı MCLR ucuna uygulanan gerilim 0 V olunca programın çalışması başlangıç adresine döner. Programın
ilk adresten itibaren tekrar çalışabilmesi için reset ucuna uygulanan gerilimin +5 V olması gerekir. Bir buton aracılığı ile reset işlemini yapan devre şekilde görülmektedir.



PIC16F84'ün reset devresi

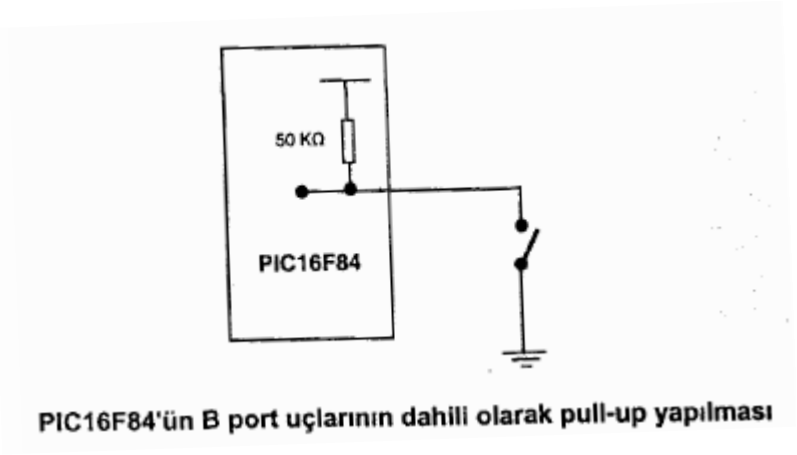
IO PORTLARI

P1C16F84ün 13 adet giriş/çıkış portu vardır. Bunlardan 5 tanesine A portu (RAO-RA4), 8 tanesine de B portu (RBO-RB7) denir. 13 portun her biri giriş ya da çıkış olarak kullanılabilir. P10 içerisinde adına TRIS denilen özel bir data yönlendirme registeri vardır. Bu register aracılığı ile portların giriş/çıkış yönlendirmesinin nasıl yapılacağı hakkında detaylı bilgiyi programlama konusunda bulacaksınız. I/O portlarından geçebilecek 25 mA lik bir sink akımı veya 20 mA lik source akımı LED'leri doğrudan sürebilir. Bu akımlar aynı zamanda LCD, lojik entegre ve hatta 220 V luk şehir şebekesine bağlı bir lambayı kontrol eden triyakı bile tetiklemeye yeterlidir. Çıkış akımı yetmediği durumda yükselteç devreleri kullanarak daha yüksek akımlara kumanda etmek mümkün olabilir.



PIC16F84'ün portlarıyla kontrol edilen 220 V luk lamba ve LED

B portunun 8 ucu, PIC içerisinde dahili olarak 50 KΩ luk dirençlerle pulI-up yapılmış gibi etki gösterir. Bu durum Şekilde temsili olarak görülmektedir PIC'in içerisinde gerçekte bir pulI-up direnci değil, farklı bir mantıksal devre vardır.



PIC16F84'ün B port uçlarının dahili olarak pull-up yapılması

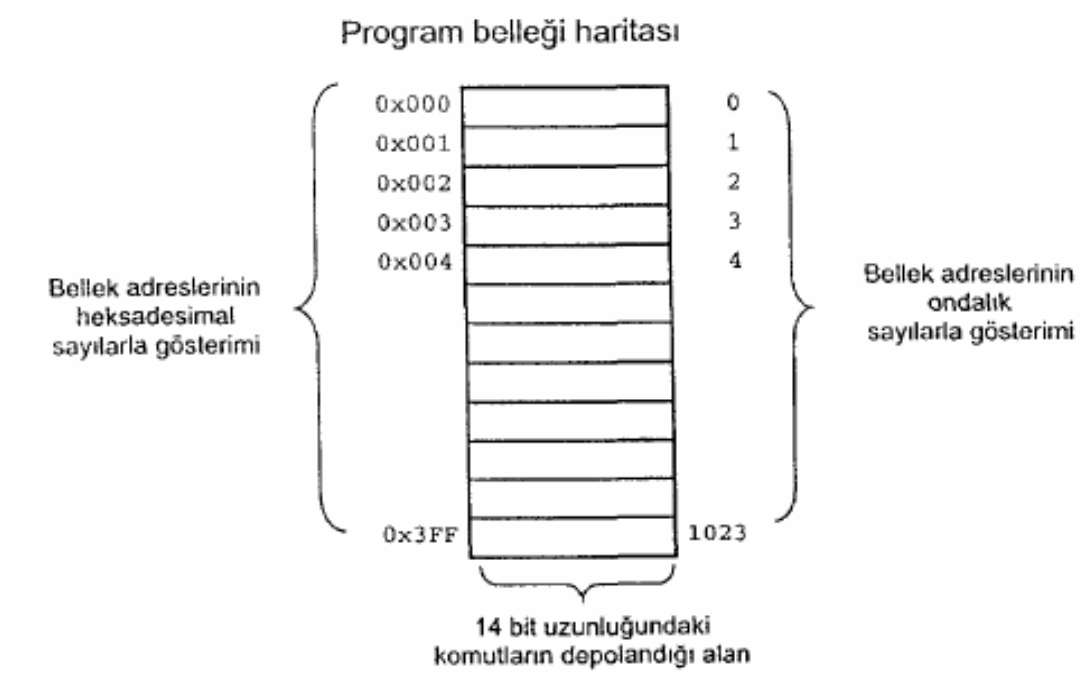
Bu 8 pulI-up direncinin tamamı option register içerisindeki yazılım aracığıyla iptal (Disable) edilebilir veya geçerli (Enable) kılınabilir. Port uçlarından herhangi birisi çıkış olarak yönlendirildiğinde o uçtaki pulI-up direnci otomatik olarak iptal olur. PIC'e enerji verildiğinde (Power-on-reset) ise tüm pull-up'lar iptal edilir

PIC 16F84 'ÜN BELLEĞİ

P1C16F84 mikrodenetleyicisinin belleği, program ve RAM belleği olmak üzere iki ayrı bellek bloğundan oluşur. Microchip'in kataloglarında PIC denetleyicilerin RISC işlemci olarak tanıtılmasının nedeni de budur. Çünkü Harvard Mimarisi ile üretilen RISC işlemcilerde program belleği ile data belleği birbirinden aynıdır. Oysa PC'lerde kullanılan çoğu mikroişlemci mimarisinde böyle bir ayrım yoktur. Bu da demektir ki; burada ayrıntısına girmeye gerek görmediğimiz nedenlerden dolayı mikroişlemciler, mikrodenetleyicilere göre komut işlemede daha yavaştırlar.

Program Belleği

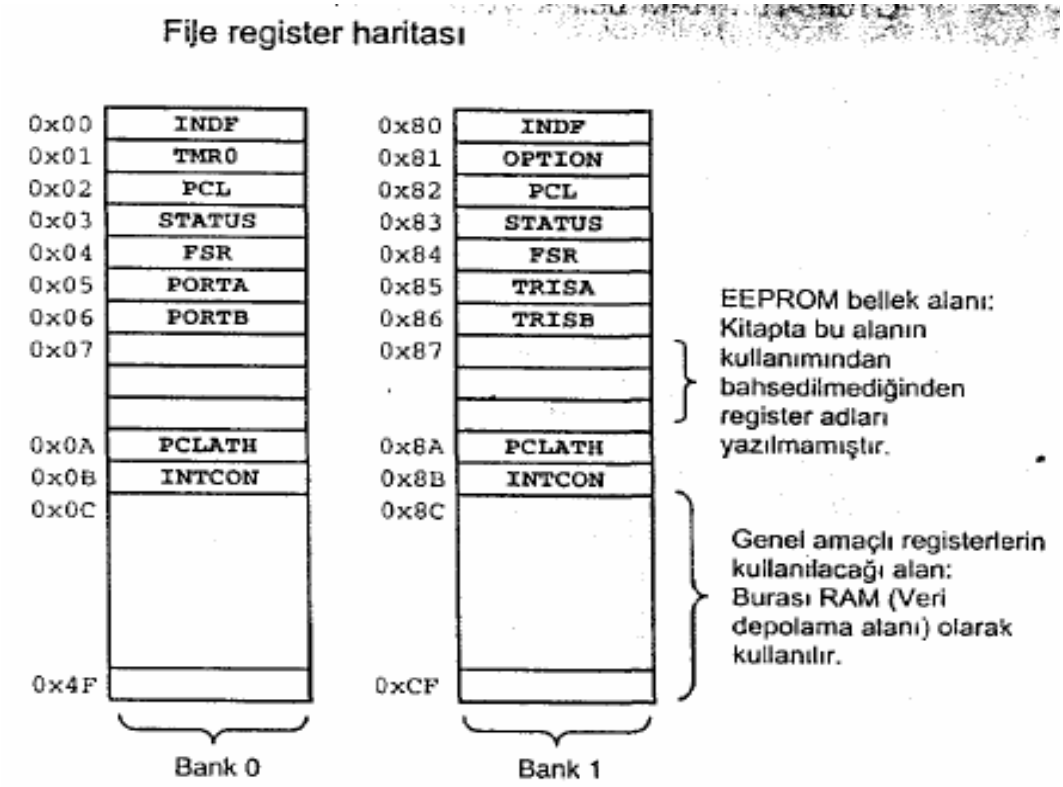
P1C16F84'ün 1 Kbyte'lık program belleği vardır. Her bir bellek hücresi içerisine 14 bit uzunluğundaki program komutları saklanır. Program belleği flash (elektriksel olarak yazılıp silinebilir.) olmasına rağmen, programın çalışması esnasında sadece okunabilir.



NOT: Yukarıda PIC16F84'ün program bellek haritasındaki adresler gösterilirken sol tarafta heksadesimal notasyon kullanılmıştır. PIC programlama esnasında da bellek adresleri bu şekilde yazılır. Ox)(X heksadesimal notasyonunda X'ler 0—F arasındaki herhangi bir sayıyı, "Ox" ise bu sayıların heksadesimal olduğunu belirtir. Örneğin 0x0F, heksadesimal OF sayısı demektir. Ox3FF ise 3FF heksadesimal sayısını gösterir. P1C16F84'ün program belleğine 14 bit uzunluğunda toplam 1024 tane komut yazılabilir. Bellek haritasında son bellek adresinin 0x3FF=1023 gösterilmesinin nedeni, adresin O'dan başlamasındandır. Adres l'den başlasaydı son adres 0x400=1024 olacaktı.

RAM Bellek

P1C16F84'ün OxOO—Ox4F adres aralığında ayrılmış olan RAM belleği vardır. Bu bellek içerisindeki file registerleri içerisine yerleştirilen veriler PIC CPU'sunun çalışmasını kontrol ederler. File registerlerin bellek uzunluğu 8 bittir. File register adı verilen özel veri alanlarının dışında kalan diğer bellek alanları, normal RAM bellek olarak kullanılırlar. Yani bu alanlarda programda içerisindeki değişkenler için kullanılır.



P1C16F84ün RAM belleği iki sayfadan (Bank'tan) meydana gelir. BankO daki registerlerin adresleri OxOO—Ox4F arasında, bank1 deki registerlerin adresleri de 0x80—OxCF arasındadır. Toplam 80 tane file register olmasına rağmen programlamaya yeni başlayanlar bankl'de 80, bank 2'de 80 olmak üzere 160 register alanı olduğunu sanırlar. Oysa, dikkat edilirse bazı özel amaçlı registerler her iki bankta da görülür.

Bir banktaki registeri kullanabilmek için o banka geçmek gerekir. Bank değiştirme işlemi programlama örnekleri verilirken detaylı olarak işlenecektir. Yalnız burada bir şeyden daha bahsetmek gerekir. Bazı özel registerlerin her iki bank'ta da görülmesinin nedeni, bank değiştirme işlemine gerek duyulmaksızın kullanılabilmesi içindir.

ASSEMBLER NEDİR ?

PIC ASSEMBLY DİLİ NEDİR?
PIC ASSEMBLY DİLİ YAZIM KURALLARİ
PIC ASSEMBLY KONUTLARININ YAZILIŞ BİÇİMİ
SAYI VE KARAKTERLERİN YAZILIŞ BİÇİMİ
PIC ASSEMBLY KOMUTLARI

ASSEMBLER NEDİR?

Assembler, bir text editöründe assembly dili kurallarına göre yazılmış olan komutları PIC'in anlayabileceği hexadesimal kodlara çeviren (derleyen) bir programdır. Microchip firmasının hazırladığı MPASM bu işi yapan assembler programıdır. Assemblere çoğu zaman compilerde (derleyici) denilir.

PIC ASSEMBLY DİLİ NEDİR?

Assembly dili, bir PICe yaptırılması istenen işlerin belirli kurallara göre yazılmış komutlar dizisidir. Assembly dili komutları İngilizce dilindeki bazı kısaltmalardan meydana gelir. Bu kısaltmalar genellikle bir komutun çalışmasını ifade eden cümlenin baş harflerinden oluşur. Böylece elde edilen komut, bellekte tutulması kolay bir hale getirilmiştir.
Örneğin:

BTFSC (Bit Test F Skip if Clear) - File registerdeki bit'i test et, eğer sıfırsa bir sonraki komutu atla, anlamında kullanılan ingilizce cümlenin kısaltmasıdır.

PIC ASSEMBLY DİLİ YAZIM KURALLARI

PIC assembly programlarının yazılması için kullanılan text editörlerinden l.bölümde bahsetmiştik. Bu editörler Windows altında çalışan NOTPAD veya DOS altında çalışan EDIT en uygunlarıdır. Bunların dışında printer kontrol komutları içermeyen ve ASCII kodunda dosya üretebilen herhangi bir editör de kullanılabilir. MPLAB kullanıldığında ayrıca bir editör kullanmaya gerek yoktur. Çünkü MPLAB'ın içinde hem bir text editörü hem de MPASM bulunmaktadır.

MPASM assembler programının yazılan komutları doğru olarak algılayıp, PIC'in anlayabileceği heksadesimal kodlara dönüştürebilmesi için şu bilgiler program içinde özel formatta yazılması gerekir:

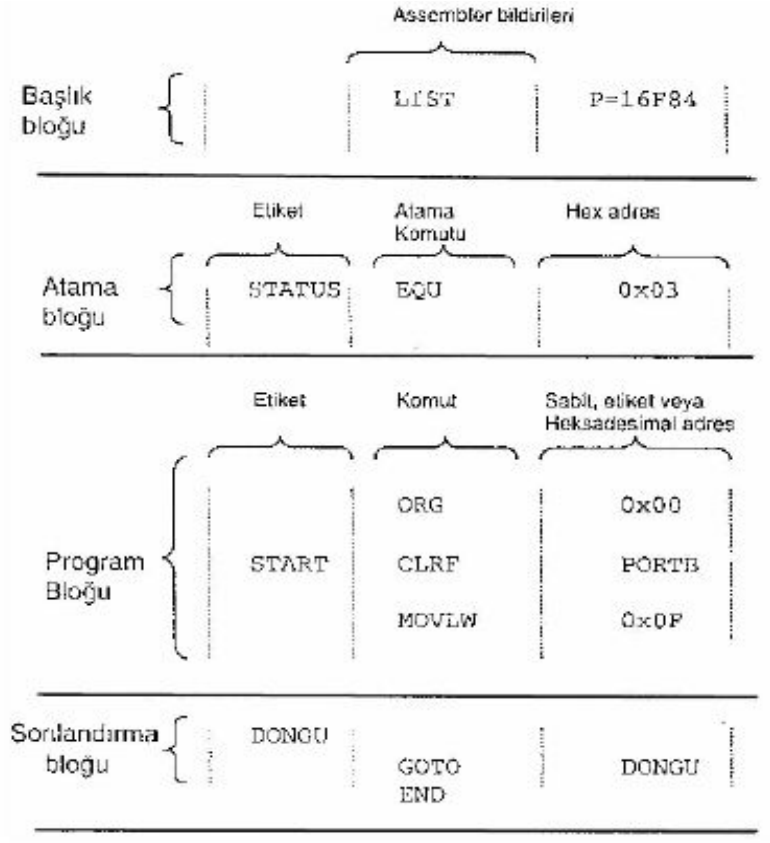
- Komutların hangi PIC16XXX için yazıldığı,
- Programın bellekteki hangi adresten başlayacağı,
- Komutların ve etiketlerin neler olduğu,
- Programın bitiş yeri,

Noktalı Virgül (;)

Baş tarafına (;) konulan satır, assembler tarafından hexadesimal kodlara dönüştürülmez. Bu satırlar programın geliştirilmesi esnasında hatırlatıcı açıklamaların yazılmasında kullanılır. Örneğin CLRF ile başlayan satırda "portB'nin içini sıfırla" cümlesi, CIJRF komutunun ne iş yaptığını açıklar. Programın bölümlerini birbirinden ayırmak için (-————— veya =====) çizgileri kullanmak, programı görsel olarak daha okunur hale getirdiği gibi bu çizgiler arasına uyarılar ve açıklamalar da yazılabilir.

Girintiler ve Program Bölümleri

Text editörlerinde birbirinden farklı uzunlukta girintiler veren TAB özelliği vardır. Bu özellikten yararlanarak assembly komutları üç kolona bölünerek yazılır. Bir assembly programı temel olarak dört bölüme ayrılır. Bunlar: Başlık, atama, program ve sonuç bölümleridir.



Aşağıdaki programda program bölümlerine dikkat ediniz.

```
            LIST        P=16F84
STATUS      EQU         0x03
PORTB       EQU         0x06
TRISB       EQU         0x86

START
            CLRF        PORTB
            BSF         STATUS,5
            CLRF        TRISB
            BCF         STATUS,5
            MOVLW       OxOF
            MOVWF       PORTB
DONGU

            GOTO        DONGU
            END
```

Başlık

Programın en başındaki bilgilere başlık bölümü denilir.

```
PICTEST1. ASM  ====================  1 O/0312000======
LIST            P=16F84
```

Başlık bölümünde program dosyasının adı ve hazırlandığı tarih, istenirse hazırlayanın adı da yazılabilir. İlk satır, bir açıklama satırıdır ve assembler tarafından derlenmez.

LIST P=16F84 satırı, programın hangi PIC için yazıldığını belirtir. LIST bir compiler bildirisidir. Yani compileri yönlendiren bir komuttur ve yegane kullanış amacı yeri burasıdır.

Etiketler

PIC belleğindeki bir adresin atandığı, hatırlamayı kolaylaştıran kısaltmalardan meydana gelen sembolik isimlere etiket denilir. Örneğin PORTB etiketi, P1C16F84'ün file register belleğindeki B portunun bulunduğu adresi temsil eden etikettir. Etiketler program içerisinde 1. kolona yazılır.

Etiket tanımlarken uyulması gereken kurallar şunlardır:

- Etiketler 1. kolona yazılmalıdır.
- Etiketler bir harfle veya alt çizgi(_) ile başlamalıdır.
- Etiketler içerisinde Türkçe karakterler kullanılamaz.
- Etiketler bir assembly komutundan oluşamaz.
- Etiketlerin içerisinde alt çizgi, rakam, soru işareti bulunabilir.
- Etiketler en fazla 31 karakter uzunluğunda olabilir.
- Etiketlerde büyük/küçük hart duyarlılığı vardır. ("Döngü" olarak tanımlanmış bir etiketi program içerisinde "döngü" yazarak kullanılamaz.)

Atama deyimi (EQU)

EOU deyimi P1C16F84'ün belleğindeki bir hexadesimal adresi belirlenen bir etikete atamak için kullanılır.

Sonlandırma Bloğu

P1C16F84'ün duraklama (halt) komutu yoktur. Programı belirli bir yerde duraklatmak için bazen sonsuz döngü kullanılır.

```
DONGU
GOTO  DONGU
 END
```

Yukarıdaki sonsuz döngüde DONGU etiketine assembler otomatik olarak bir adres verir. GOTO DONGU komutu ise program akışını devamlı olarak aynı adrese gönderir. Bu durumda program belirlenen adreste duraklatılmış olur.

END deyimi ise program komutlarının sona erdiğini assembler'a bildirir. Her program sonunda END deyimi muhakkak kullanılmalıdır. Aksi halde program devam derlenirken dosya sonunun belirtilmediğini belirten bir hata mesajı verecektir.

PIC ASSEMBLY KOMUTLARININ YAZILIŞ BİÇİMİ

P1C16F84ün toplam 35 tane komutu vardır. Bu komutların yazılış biçimini üç grupta toplayabiliriz.

1. Byte-yönlendirmeli komutlar.
2. Bit-yönlendirmeli komutlar.
3. Sabit işleyen komutlar.
4. Kontrol komutları.

SAYI VE KARAKTERLERİN YAZILIŞ BİÇİMİ

PIC assembly komutlarında sayılar heksadesimal, binary veya desimal formda kullanılabilir. Değişik kaynaklarda kullanılan sayı ve karakter gösteriliş biçimleriyle karşılaştığınızda bunları okuyabilmeniz için aşağıda örnekler verilmiştir.

Heksadesimal sayılar

Heksadesimal sayılar Ox, 0 veya h' harfleriyle başlamalıdır. Örneğin, STATUS registerine 03 adresini atamak için aşağıda gösterilen yazılış biçimleri kullanılabilir.

```
STATUS  EQU     0X03
        EQU     3
        EQU     03
        EQU     03h
        EQU     h' 03' (Bu kitapta örneklerde kullanılacak
```

h' FF'
Eğer FF heksadesimal sayısını aşağıdaki gibi kullanmaya kalkarsanız çalışmaz

```
MOVLW   FP
MOVLW   FFh
```

Çünkü kural olarak heksadesimal sayılar muhakkak '0' veya "h" harfi ile başlamalıdır.

Binary Sayılar

Binary sayılar b harfi ile başlamalıdır. Örneğin 00001010 binary sayısını W registeri içerisine yüklemek için aşağıdaki gibi yazılmalıdır.

```
MOVLW   b'00001010'
```

Desimal sayılar

Desimal sayıların başına d harfi konularak tırnak içerisinde yazılırlar. Örneğin 15 desimal sayısı W registeri içerisine yüklemek için aşağıdaki gibi yazılmalıdır.

```
MOVLW     d'15'
MOVLW     d'255'
```

ASCII Karakterler

Genellikle RETLVV komutu ile birlikte kullanılan ASCII karakterler tırnak içerisine alınarak aşağıdaki gibi yazılırlar.

```
RETLW     'A'
RETLW     'T'
```

PIC ASSEMBLY KOMUTLARI

Yer Değiştirme veya Yükleme Komutları			
Komut ve Örnek		İngilizce tanımı	Türkçe açıklaması
MOVLW k		Move Literal to W	K sabit değerini W registerine yükler.
MOVLW h '0F'			W<-OP
MOVF f,d		Move f	f registerinin içeriğini W veya f'e yükler.
MOVF TEST,0			d=0 W←TEST d=1 TEST←TEST
MOVMF f		Move W to f	W registerin içeriğini f registerine yükler.
MOVWF PORTA			PORTA←W
Register İçeriğini Değiştirme Komutları			
CLRF	F	Clear f	f registerinin içeriğini siler (sıfırlar).
CLRF	TRISA		TRISA←00000000
CLRW		Clear W	W registerin içeriğini siler (sıfırlar).
CLRW			W←00000000

COMF	F,d	Complement f	F registerinin içindeki sayı terslenir. Yani tüm 1'ler 0, 0'lar 1 olur. Sonuç W veya f registerine yüklenir.
COMF	SAY, 0		SAY= 0011011 ise, d=0 W←11001010 d=1 olsaydı SAY←11001010
DECF	f,d	Decrement f	F registerinin içerisindeki sayıyı "1" eksiltir. Registerin içeriği h'00' ise, "1" eksiltildiğinde h'FF' olur. Sonuç W veya f registerine yazılır.
DECF	GIT,1		GIT=h' 2C" ise 2C-1=2B d=1 GIT←2B d=0 olsaydı W←2B
INCF	f,d	Increment f	F registerinin içerisindeki sayıyı "1" artırır. Registerin içeriği h' FF' ise, "1" arttırıldığında h' 00' olur. Sonuç W veya f registerine yazılır.
INCF	GIT, 0		GIT=h' 2C' ise 2C+1=2D d=0 W←2D d=1 olsaydı GIT←2D
BCF	f,b	Bit Clear f	f registerinin içerisindeki sayının b.ninci bitini sıfırlar.
BCF	PORTB,5		PORTB = b'11111111' ise, PORTB←b' 11011111'
BSF	f,b	Bit Set f	F registerinin içerisinde sayının b.ninci bitini 1 yapar.
BSF	PORTA,3		PORTA = b' 00000000' ise, PORTA←b' 00001000'
RLF	f,d	Rotate Left f	f registeri içerisindeki sayıyı bir pozisyon sola kaydırır. Registerden taşarak Carry bayrağına yazılan bit, LSB'ye yazılır. Sonuç W veya f registerine yüklenir.
RLF	KAY,0		
RRF	F,d	Rotate Right f	F registeri içersindeki sayıyı bir pozisyon sağa kaydırır. Registerden taşarak Carry bayrağına yazılan bit, MSB'ye yazılır. Sonuç W veya f registerine yüklenir.
RKF	KAY,1		
SWAPF	f,d	Swap nibbles inf	f registerinin içerisindeki ilk dört bit ile son dört biti yer değiştirir. Sonuç W veya f registerine yüklenir.
SWAPF	DEG,1		DEG = b' 00101111' ise, d=1 olduğundan DEG←11110010

			d=0 olsaydı W←11110010'
Program Akışını Kontrol Etme Komutları			
GOTO	k	Go to address	Program akışı k adresine dallanır.
GOTO	DOMGU		Program, DÖNGÜ etiketinin yazıldığı yere dallanır ve buradan itibaren devam eder.
CALL	k	Call subroutine	Program akışı k etiketinin bulunduğu yerdeki alt programa dallanır.
CALL	TIMER		Program TIMER etiketinin yazıldığı alt program satırlarının başlangıcına dallanır ve buradan itibaren devam eder.
KETÜRN		Return from subrouitine	Alt program komutlarının en sonuna yazılan bu komut, program akışını ana programa geri döndürür.
RETLW		Return with Literal in W	Program akışını alt programdan ana programa döndürür ve W registerine k sabitini yükler.
RETLW	H'2F'		Alt programdan ana programa döndürür ve W registerine 2F yüklenir.
RETFIE		Return From Interrupt	Program akışını interrupt alt programından ana programa döndürür.
BTFSC	f,b	Bit Test F, Skip if Clear	F registerinin b.inci bit'ini test eder. Eğer bu bit "0"sa program akışı bir sonraki komuta geçer.
BTFSC	PORTA,2		
BTFSS	f,b	Bit Test F, Skip if Set	F registerinin b.inci bit'ini test eder. Eğer bu bit "1"se program akışı bir sonraki komuta geçer.
BTFSS	PORTA,0		
DECFSZ	f,d	Decrement f, Skip if Zero	F registerinin içeriğini "1" azaltır. Register içeriği 0'sa bir sonraki komuta atlar. Sonuç W veya f registere yazılır.
DECFSZ	SAYAC,!		SAYAC=h' 2F' ise 2F-1=2E d=0 olsaydı W←h' 2E' d=1 olduğundan SAYAC←h'2E'
INCFSZ	F,d	Increment f, Skip if Zero	F registerinin içeriğini "1" arttırır. register içeriği "0"sa bir sonraki komuta atlar. Sonuç W veya f registere yazılır.
INCFSZ	SAYAC, 1		SAYAC=h' 2F' ise 2F+1=30 d=1 SAYAC←h' 30' d=0 W←h'30"

Mikrodenetleyici Kontrol Komutları			
CLRWDT		Clear VVatchdog Timer	Watchdog timer'ı sıfırlar. Aynca vvatchdog timer'ın prescalar değerini de sıfırlar. Status bitlerinden TO ve PD'yi "1"yapar.
SLEEP		Go into standby mode	Mikrodenetleyiciyi uyuma moduna geçirerek güç harcamasını azaltır. Microdenetleyici uyuma modundan reset, watchdog timer ve TOCKI girişi vasıtasıyla çıkar.
Mantıksal Komutlar			
ANDLW	K	AND Literal with W	W registerin içeriği ile k sabitine AND işlemini uygular. Sonuç W registerine yazılır.
ANDLW	b' 0110001'		W = b' 10011101' ise, b' 00110001' sabitin değeri b' 00010001' AND işlemi sonucu W←b' 00010001'
ANDWF	f,d	AND W with f	W registeri ile file register içeriğine AND işlemini uygular. Sonuç W veya f registerine yazılır.
ANDWF	TEST,1		W=b' 11111111' ise, TEST=b'11011110" ise, b' 11011110' AND işlemi sonucu d=0 ise W←b' 11011110' d=1 olduğundan TEST←b' 11011110 '
IORLW	k	Inclusive OR Literal with W	W registerin içeriği ile k sabitine OR işlemini uygular. Sonuç W registerine yazılır.
IORLW	B-00101000-		W =b' 10000100' ise b' 00101000' sabitin değeri b' 10101100' OR sonucu , W ←b' 10101100'
IORWF	f,d		W registeri içeriği ile file registerin içeriğine OR işlemini uygular. Sonuç W veya f registerine yazılır.
IORWF	TEST,1		W = b' 10000100" ise, TEST = b' 00101000' ise, b' 10101100 ' OR sonucu d=0 ise W←b' 10101100" d=1 olursa TEST←b' 0101100 '
XORLW	k	Exclusive OR Literal withW	W registerin içeriği ile k sabitine XOR işlemini uygular. Sonuç W registerine yazılır.
XORLW	b' 00101000'		W = b ' 00000000 ' ise b' 00101000' sabitin değeri b' 11010111" XOR sonucu W←b' 11010111'
XORWF<"	*P.S*- • ..• . -	Exclusive OR W with f	W register ile file register içeriğine XOR işlemini uygular. Sonuç W veya f registerine yazılır.

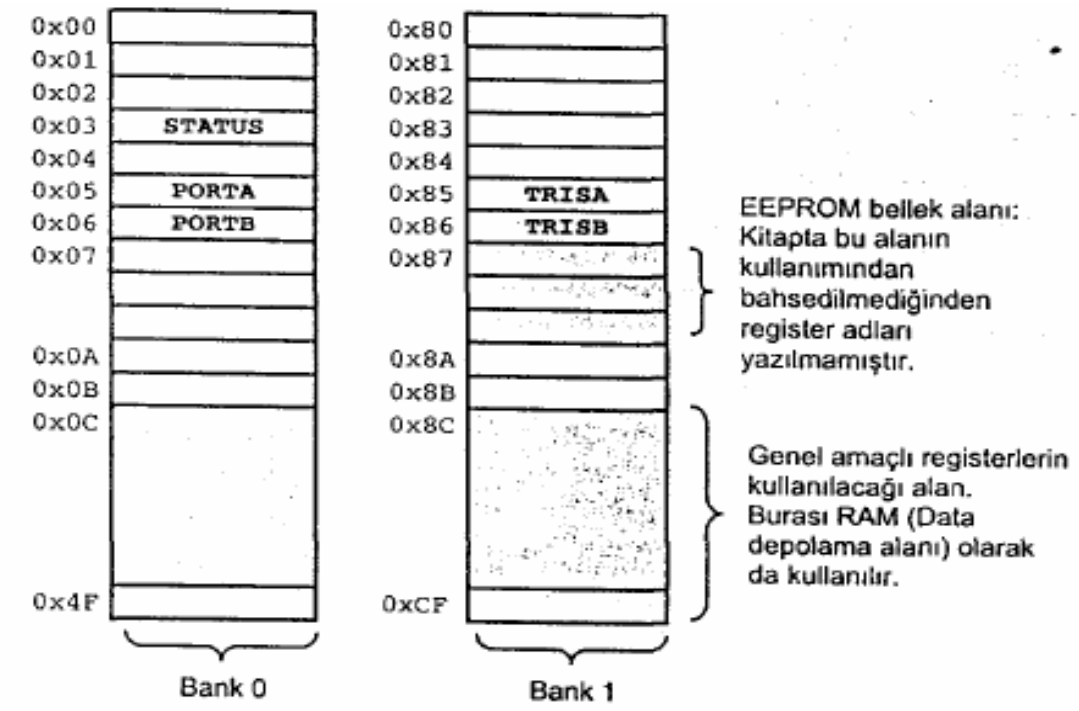
XORWF	TEST,1		W = b' 00000000' ise, TEST = b' 00101000' ise, b' 11010111" XOR sonucu d=0 ise W←b' 11010111" d=1 olursa TEST←b' 1101011'l'
Aritmetik İşlem Komutları			
ADDWF	f,d	Add W with f	W registerinin içeriğini f registeri ile toplar. Sonuç W veya f registerine yazılır.
ADDWF	TOPLA,0		W=h' 2A' ise, TOPLA=h ' 31' ise, h' 2A'+h' 31'=b' 5B' d=1 ise TOPLA←h' 5B' d=0 olduğundan W←h' 5B'
ADDLW	k	Add Literal andW	W registerinin içeriğini k sabit değeri ile toplar. Sonuç W registerine yazılır.
ADDLW	H' 2F'		W=h' B0' ise, h' B0'+h' 2F'=h' DF' W←h' DF'
SUBLN	k	Subtract W from Literal	K sabit değerinden W registeri içeriğini çıkarır. Sonuç W registerine yazılır.
SUBLW	H' 90'		W=h' 83' ise h' 90'-h' 83' = h' 07' W←h' 07'
SUBWF	f,d	Subtract W from file register	f registerinin içeriğinden W registerinin içeriğini çıkarır. Sonuç W veya f registerine yazılır.
SUBWF	CIK,1		W=h' 83' ise, CIK=h' 90' ise, h' 90'-h' 83'=h' 07' d=0 ise W←h' 07' d=1 olduğundan CIK←h'07'
İşlem Yapmayan Komut			
NOP		No Operation	Bir komut saykılı süresince hiçbir işlem yapmayan bir komuttur. Bir dahili komut süresinde çalışır. Bu nedenle zaman geciktirme işlemlerinde kullanılır.

İLK PROGRAMINIZ

AKIŞ DİYAGRAMI SEMBOLLERİ
AKIŞ DİYAGRAMININ ÇİZİLMESİ
ASSEMBLY PROGRAM KOMUTLARININ YAZILMASI
PROGRAMLARIN DERLENMESİ (MPASM)

Bu ünitede, basit bir programı ele alıp, bunun akış diyagramının nasıl çizildiğini, program komutlarının nasıl yazılarak derlendiğini daha sonra da bu programın PICe nasıl yazıldığını göstereceğiz.

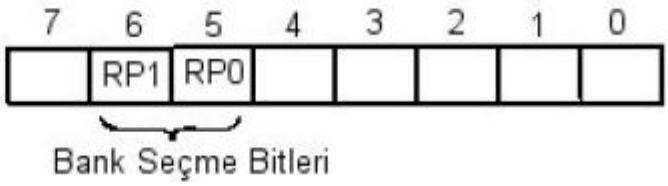
Program, PICe enerji verince portB'nin 0. bit'ine bağlı olan bir LED'i direkt olarak yakacaktır. Programa başlamadan önce kullanacağımız özel file registerlerin adreslerini vereceğiz. Daha sonra da STATUS registerini kullanarak bank değiştirme işleminin nasıl yapıldığını göreceğiz.



Bank Değiştirme

Bir bank'tan diğer banka geçmek için STATUS register denilen özel registerin 5. ve 6. bitinin durumunu değiştirmek gerekir.

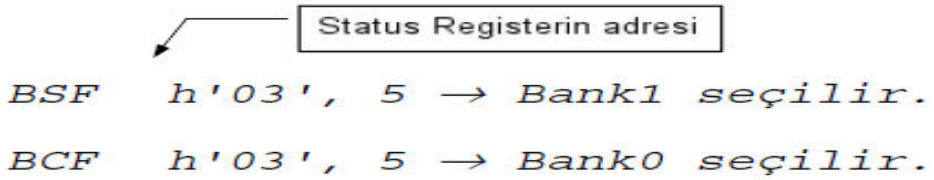
STATUS REGİSTER



00— Bank 0
01— Bank 1
10— Bank 2
11— Bank 3

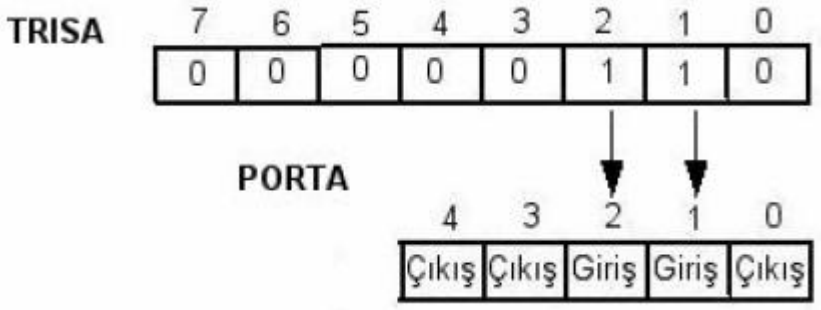
P1C16F84'ün sadece 2 bank'ı bulunduğundan, sadece 5. bit'in değerini değiştirmek yeterlidir. 6. bit'in değeri daima "0" olmalıdır. Zaten PIC'e enerji verildiğinde (Power-on reset) 5. ve 6. bit'in değeri•"O"dır. Bu bit'ler aynı zamanda diğer reset girişleri (MCLR ucundan yapılan harici reset ve Watchdog timer reset) yapıldığında da "0" olur. Yani PIC'i çalıştırmak için enerji verildiğinde direkt olarak bankO seçilmiş olur. Bu durumda bank değiştirme işlemine gerek duyulmaksızın bank0'daki registerler kullanılabilir.

STATUS registerinin 5. bit'ini "1" yapmak için BSF komutu, "0" yapmak için de BCF komutu kullanılır. Aşağıda bank değiştirmek için kullanılan komutlar görülmektedir.

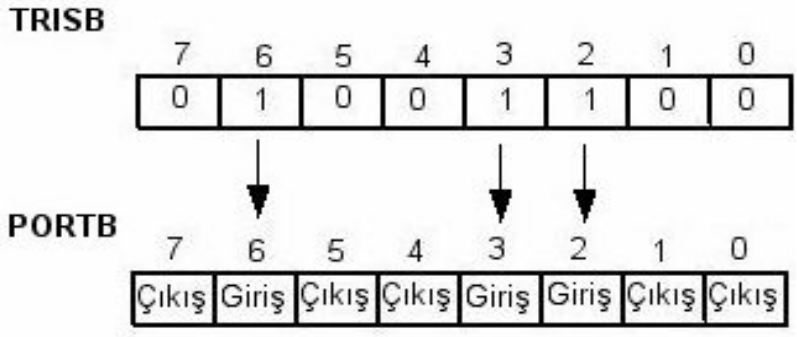


Port'ların Giriş veya Çıkış Olarak Yönlendirilmesi

PortA ve PortB'nin uçlarına bağlı bulunan bir 1/0 elemanını kullanabilmek için portları giriş veya çıkış olarak yönlendirmek gerekir.
PortA'yı —>TRISA registeri,
PortB'vi —>TRISB reqisteri yönlendirir.
PortA/PortB'nin hangi bit'i giriş yapılmak isteniyorsa, TRISA/TRISB içerisinde o bit'e karşılık gelen bit "1" yapılır. Çıkış olarak yönlendirilmek istenen bit'ler için de TRISA/TRISB içerisine "0" yazılır.
TRISA 1— PortA —> Giriş
TRISB 0 — PortB —> Çıkış
Bu söylediklerimizi bir örnek vererek şematik olarak gösterelim. Örneğin, PortA'nın 1. ve 2. bitleri giriş, diğer bitlerim çıkış olarak yönlendirelim.



PortA'nın sadece 5 ucu bulunduğundan, TRISA registerinin ilk 5 biti içerisine yazılan veriler PORTA'yı yönlendirir. Diğer bit'lerin 0 veya 1 olmasının hiçbir önemi yoktur.
PortB'nin 2, 3, 6. bitlerini giriş diğerlerini çıkış olarak yönlendirmek için de aşağıdaki konfigürasyon yapılır.

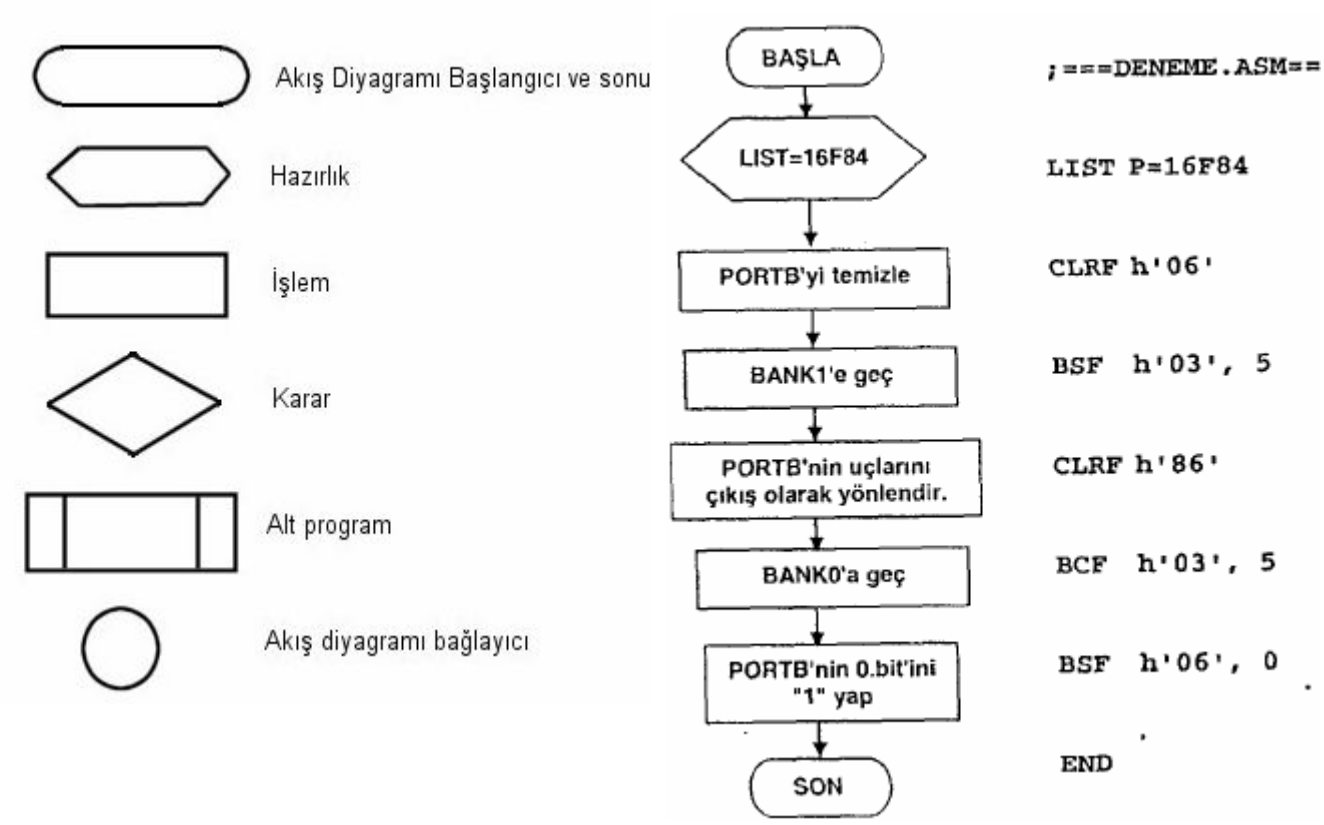


AKIŞ DİYAGRAMI SEMBOLLERİ

Tüm programlama dillerinde olduğu gibi assembly dili programlarını yazmadan önce akış diyagramı çizmek iyi bir alışkanlıktır. Kısa programlarda, genellikle karar işlemlerinin olmadığı programlarda çoğu zaman akış programı çizmeye gerek duyulmaz. Ancak uzun ve karmaşık mantık işlemlerinin yoğun olduğu programları yazarken direkt olarak komutları yazmaya başlamak kısa bir süre sonra içinden çıkılmaz hale getirebilir.

Bu nedenle programları yazmadan önce akış diyagramını çizerek, komutların hangi sıraya göre yazılacağını görmek için görsel bir düşünme ortamı yaratılmış olunur.

Programlama dillerinde ortak olarak kullanılan akış sembolleri vardır. Bu sembollerin neler olduğunu ve nerelerde kullanılacağını aşağıda göreceksiniz.



Akış diyagramları çizilirken semboller içerisine yazılan açıklamalar çoğu zaman yukarda olduğu gibi çok fazla ayrıntılı olması gerekmez. Yapılacak işlemler yerine hangi komut kullanılacaksa sembol içerisine o komut direkt olarak da yazılabilir. Akış diyagramı çizmenin amacı, karmaşık mantıksal işlemler gerektiren programları yazarken düşünme kolaylığı sağlamaktır.

Bu nedenle akış diyagramları çizerken kesin kurallara uymaya özen göstermeniz gerekmez. Semboller içerisine yazacağınız hatırlatıcı birkaç kelime ya da işaret çoğu zaman yeterli olabilir.

ASSEMBLY PROGRAM KOMUTLARININ YAZILMASI

Akış diyagramı çizildikten sonra, işlemleri gerçekleştirecek olan assembly komutları yazılır. Assembly komutları ASCII kodunda dosya üreten bir editörde hazırlanması gerekir. Bu editörler, DOS altında çalışan EDIT, WINDOWS altında çalışan NOTPAD olabilir. Kullanmaya alışık olduğunuz herhangi bir editör de bu iş için uygundur. Biz bu yaptığımız örneklerde MPLAB programını kullanacağız.

===DENEME .ASM====27/04/2000=======

```
LIST  P=16F84        ;PIC16F84'ü  MPASM'ye  tanıt
CLRF  h'06'          ;PORTB'ye bağlı LED'leri  söndür
BSF   h'03' ,   5    ;BANK1'e geç
CLRF  h'86'          ;PORTB'nin  tüm uçlarını çıkış yap
BCF   h'03' ,   5    ;BANKO'a geç
BSF   h'06' ,   0    ;PORTB'nin  O.bit'indeki  LED'i yak
END                  ;Program komutlarının  sonu
```

ilk programımızın komutlarını yazarken daha sonraki çalışmalarınızda hatırlatıcı olması için yukarda görüldüğü gibi yanlarına açıklamalar yazabilirsiniz.

Şimdi bu programı MPASM'yi kullanarak derlemeden önce, aynı programı daha anlaşılır ve daha kısa yazılış biçimlerine bir göz atalım.

Dikkat ederseniz komutları yazarken kullandığımız register ve portların RAM bellekteki adreslerini belirttik. Örneğin;

CLRF h'06' PORTB'nin RAM bellekteki adresi

BSF h'03', 5 STATUS registerin RAM bellekteki adresi

Atama (EQU) Komutu Kullanarak Program Yazmak

Şimdi programı EQU komutunu kullanarak register adlarını etiket sütununa, adreslerini de adres sütununa yazarak programı yeniden düzenleyelim.

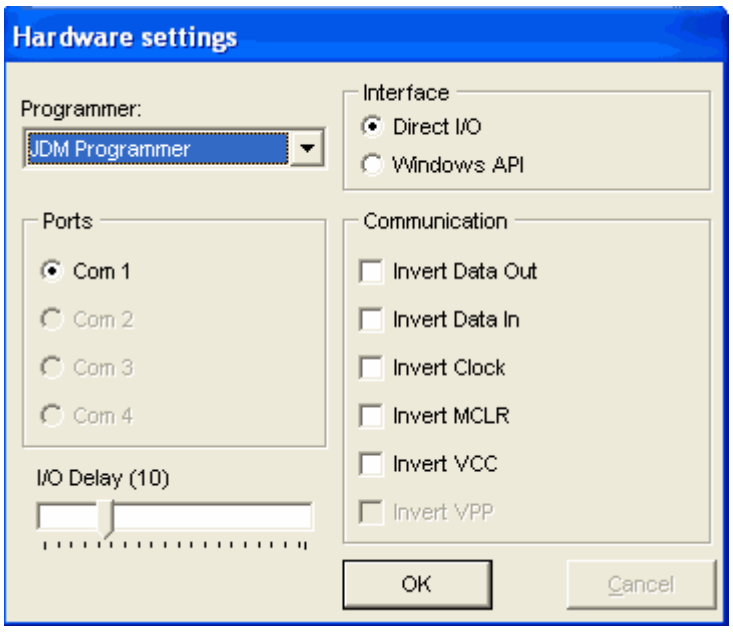
Programı editörünüzde yazdıktan sonra DENEME.ASM adıyla kaydediniz.

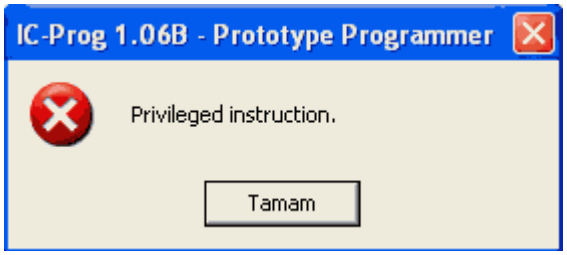
```
;===DENEME.ASM====27/04/2000=======
      LIST  P=16F84      ;PIC16F84'ün MPASM'ye tanıt
PORTB EQU   h'06'
STATUSEQU   h'03'
TRISB EQU   h'86"
      CLRF  PORTB        ;PORTB'ye bağlı LED'leri söndür
      BSF   STATUS, 5    ;BANKl'e geç
      CLRF  TRISB        ;PORTB'nin uçlarını. çıkış yap
      BCF   STATUS, 5    ;BANKO'a geç
      BSF   PORTB, 0 ,   ;PORTB'nin O.bitindeki LED'i yak
      END                ;Program komutlarının sonu
```

ICPROG Kurulumu, Genel Ayarlar ve Kullanımı:

Program dosyaları klasöre çıkarttıktan sonra **icprog.exe** dosyasını çalıştırın ilk defa kullanılıyorsa uyarı,hata mesajları görünebilir

burada gerekli olan donanım ayarlarını yapmanızı istiyor **OK** butonuna tıklayıp bu mesajı geçiyoruz sonrasında **hardware settings** bölümü gelecek

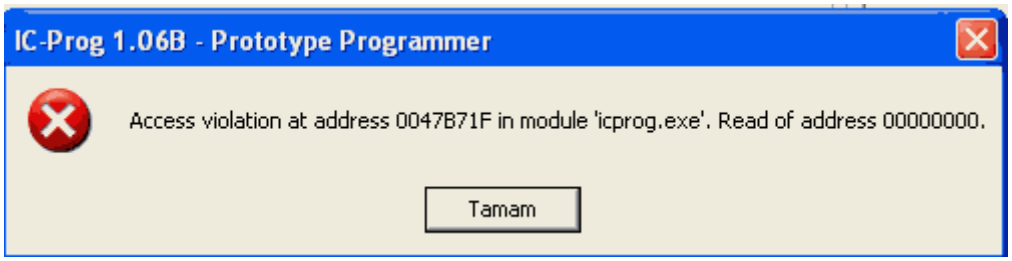




Burada hiç bir değişiklik yapmaya gerek yok. ama burada **I/O Delay** ve **Communication** bölümlerinde farklı ana kartların elektriksel özelliklerinden dolayı zamanlama problemleri programlamada hata meydana gelebiliyormuş Bu durumda **I/O delay** kaydırma çubuğunu ayarlayarak kullanılan PIC için en uygun **I/O** gecikme zamanının seçilmesi gerekli

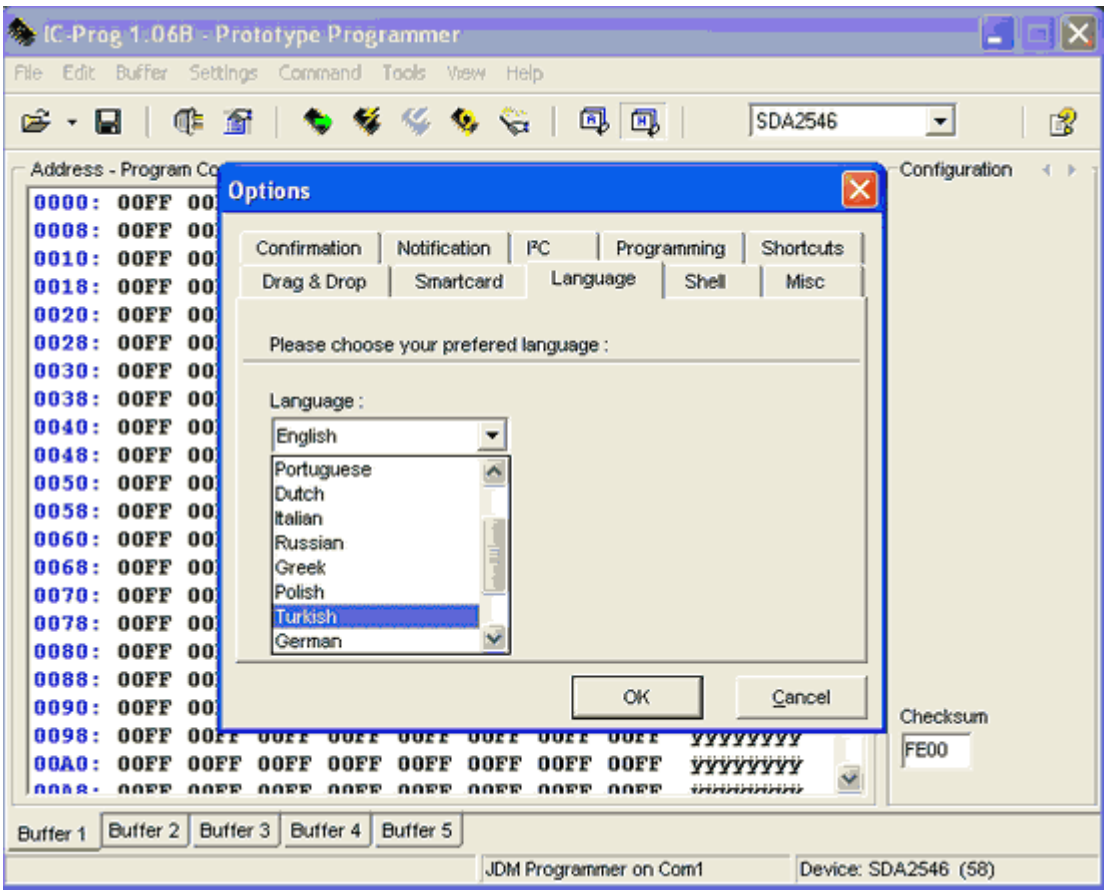
şimdiye kadar programı 3 farklı bilgisayarda kullandım bu ayarlamayı hiç yapmadım

OK butonuna tıklayıp bu mesajıda geçtikten sonra hata mesajları gelebilir örnekler aşağıda

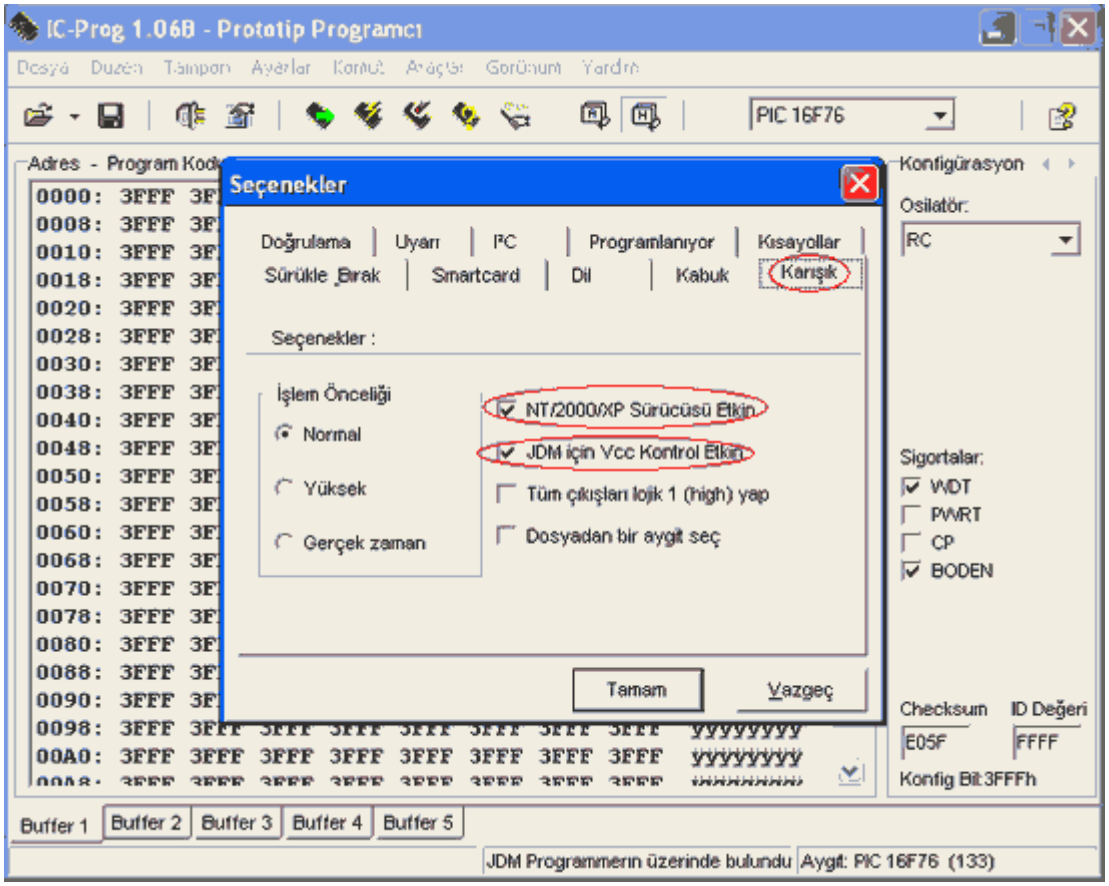


tamam butonuna tıklayarak bu hata mesajlarını geçiyoruz programın ana ekranı geldiğinde üst bölümde **Settings** yazısına tıklıyoruz oradan **Options** gelen pencerede **Language** bölümünden **Türkçe** dil seçimini yapıyoruz

IC-PROG Dil Ayarları

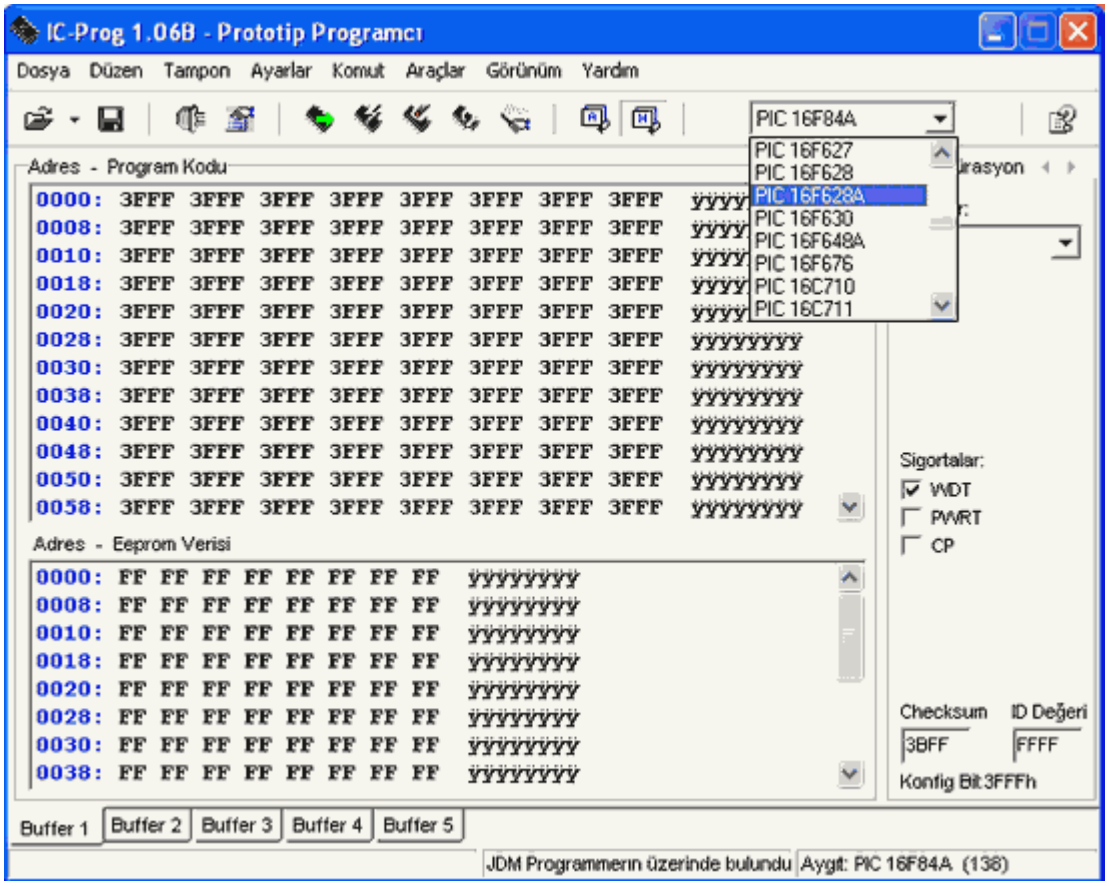


OK diyoruz artık program Türkçe son olarak aynı işlemleri tekrar yapıyoruz bu sefer **Karışık** bölümünden XP işletim sistemi için gerekli ayarları yapıyoruz.**NT/2000/XP Sürücüsü Etkin** ve **JDM İçin Vcc kontrol Etkin** seçeneklerini işaretleyip tamam diyoruz bu kadar ana ayarlar bitti şimdi programı kapatalım tekrar açalım hata mesajı gelebilir sürücü yüklenemedi vb. bu mesajları geçelim.

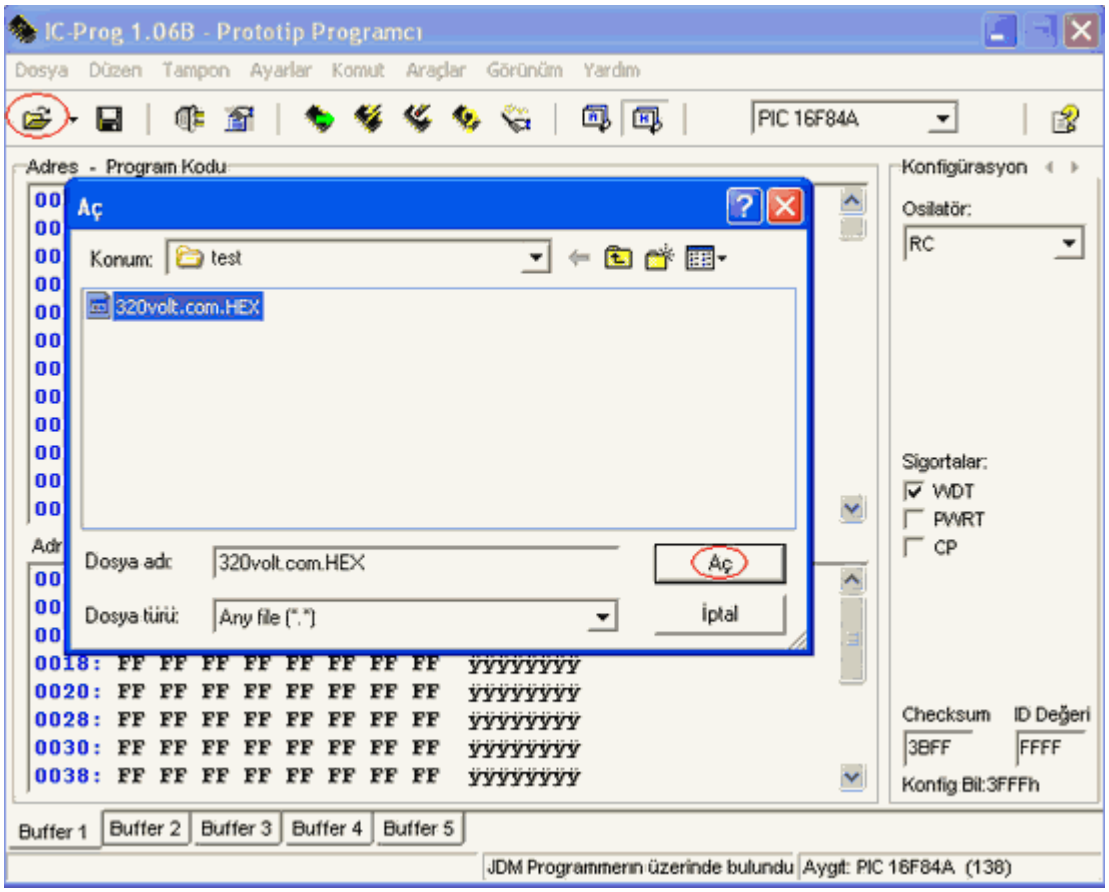


IC-PROG ile HEX Kodunun yazdırılması

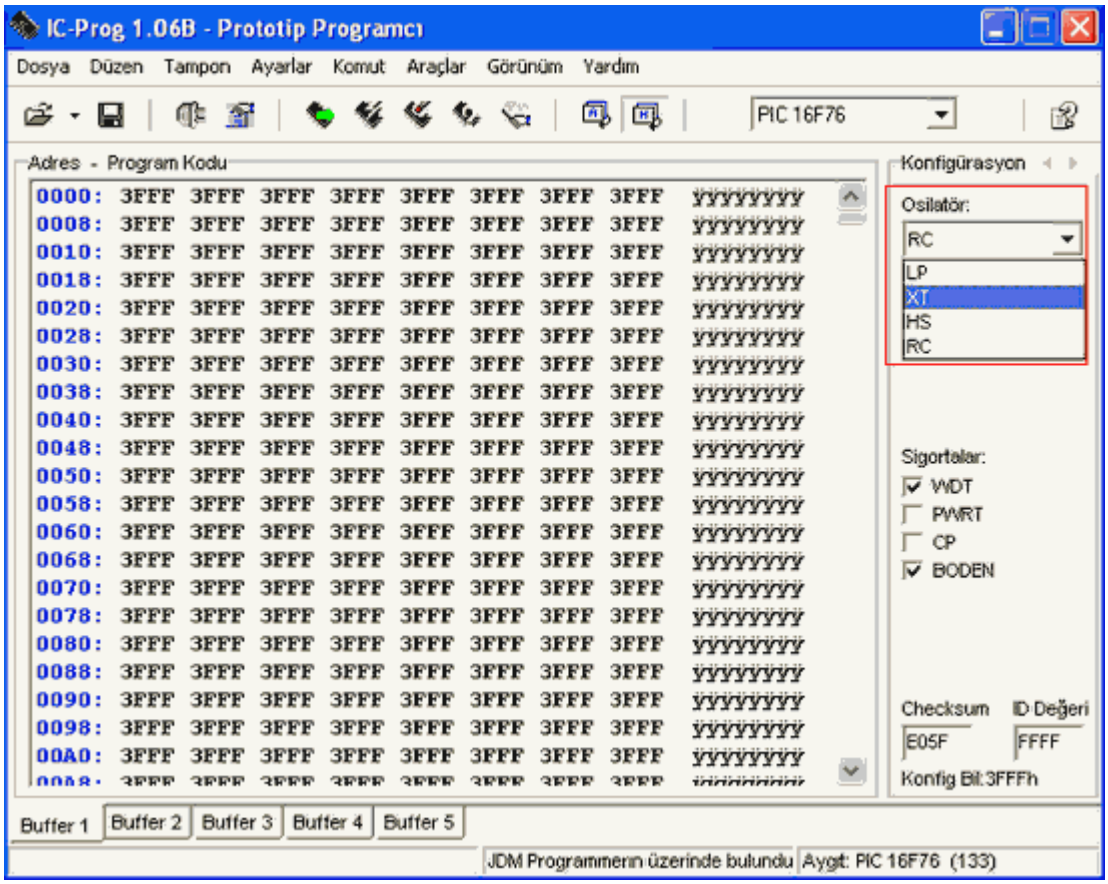
Programı yeniden başlattıktan sonra açılan ana pencerede sağ üt bölümden Programlanacak PIC Entegresini seçiyoruz



Sonrasında sol üst köşedeki klasör simgesine tıklıyoruz **HEX** dosyamızı yüklüyoruz



Son olarak Kullanılan **Osilatör** tipini seçiyoruz genelde devrelerde **XT** (kristal) kullanılır



Son olarak Tümünü Programla butonuna tıklıyoruz .

Program Menülerinin Açıklanması

- **Aç**: Yüklenilmesi istenen dosyayı seçmemizi sağlar.
- **Yeni Adla Kaydet:** Seçilmiş programı yeni adla kaydetmemizi sağlar.
- **Donanım:** Donanım ayarları yapılır.
- **Seçenekler:** Program ayarları yapılır.

-  **Tümünü Oku:** Mikrodenetleyici içindeki programı ekranda gösterir.

-  **Tümünü Programla:** Seçilmiş olan *.hex uzantılı dosyaları mikrodenetleyiciye yükler.

-  **Tümünü Sil:** Mikrodenetleyici içinde yüklü olan programı siler.

- **Doğrula:** Mikrodenetleyiciye yüklenen programla kaynak programı karşılaştırır.
- **Assembler Görünüm:** Seçilmiş programın ekranda assembler modunda görünmesini sağlar.
- **Hex Görünüm:** Seçilmiş programın ekranda hex modunda görünmesini sağlar.

Dikkat : Osilatör seçim menüsünün altında Sigortalar diye bir bölüm var orda **CP** (kod koruma) seçili olursa PIC tek kullanımlık olur tekrar yazım yapamazsınız

PIC PROGRAMLAMA KARTI UYGULAMASI

ALANI : ELEKTRİK ELEKTRONİK TEKNOLOJİSİ
DALI : GÜVENLİK SİSTEMLERİ

ADI SOYADI :

SINIFI / NO :

DEĞERLENDİRME

İŞLEM BASAMAKLARI	MONTAJ	İŞ ALIŞKANLIKLARI	SÜRE

MİKRODENETLEYİCİ PROGRAMLAMA KARTI

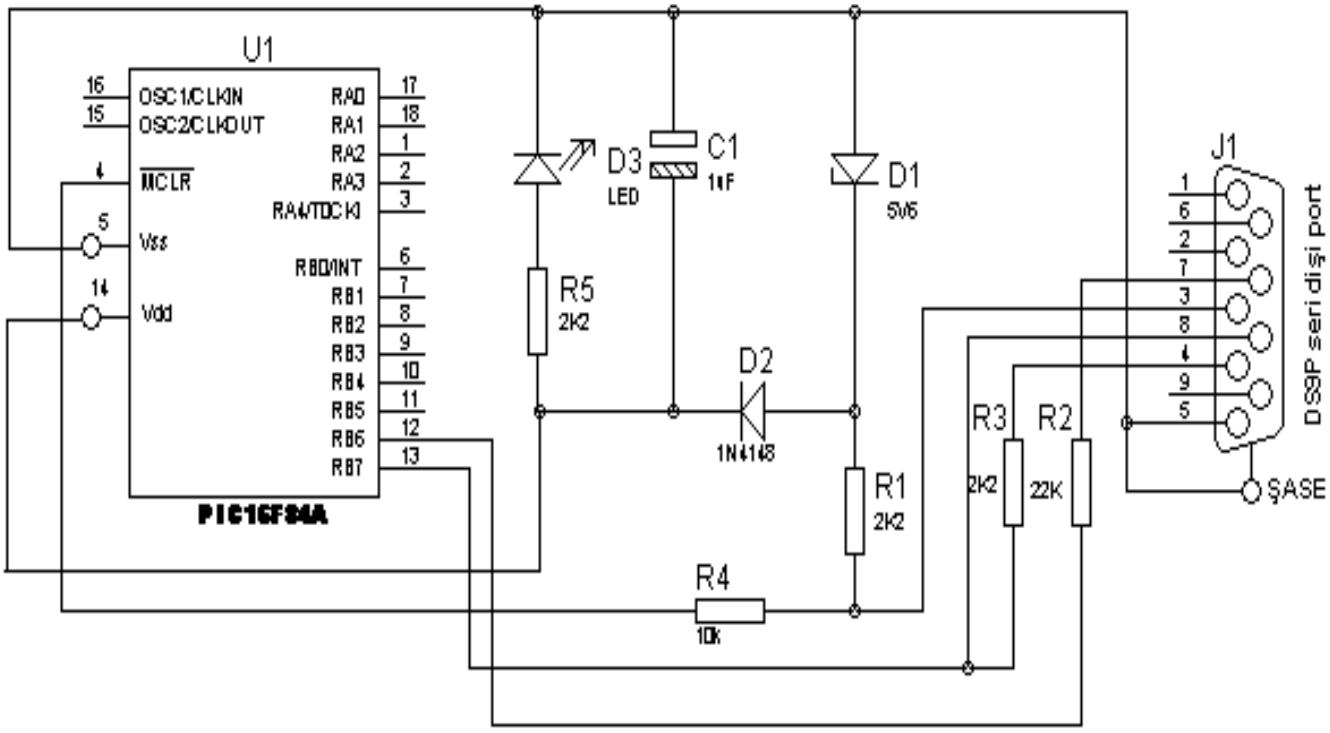
Mikrodenetleyici programlama kartı, mikrodenetleyicinin istenilen şekilde çalışabilmesi için yazılan programı mikrodenetleyiciye yükleyen karttır. Yapılan kartın özelliğine göre bilgisayarın seri veya paralel portuna bağlanabilir. Bizim yapacağımız kart seri iletişim yapan mikrodenetleyici programlama kartıdır.

Mikrodenetleyici programlama kartı D sub 9P (seri dişi port) üzerinden iletişim yapmaktadır. Devre enerjisi seri port üzerinden verilmektedir. Haricî enerji uygulamaya gerek yoktur. PIC mikrodenetleyiciler 2V-5,5V arasında çalışmaktadır. Mikrodenetleyicinin en uygun çalışma gerilimi 5V'tur.

Şekil 2.1: Mikrodenetleyici programlama ve deneme kartı görünüşleri

Mikrodenetleyiciye (16F84) derlenmiş yani makine diline çevrilmiş program kodlarının yüklemek için aracı program kullanmak gerekir. Bu konuda firmaların ürettiği; Micropro, MPlab, PicEQ, Propic, ICProg….gibi bir çok program mevcuttur. Bunlardan birini tercih ederek kodları mikrodenetleyiciye yükleyebilirsiniz.

PIC 16F84 PROGRAMLAMA KARTI AÇIK ŞEMASI



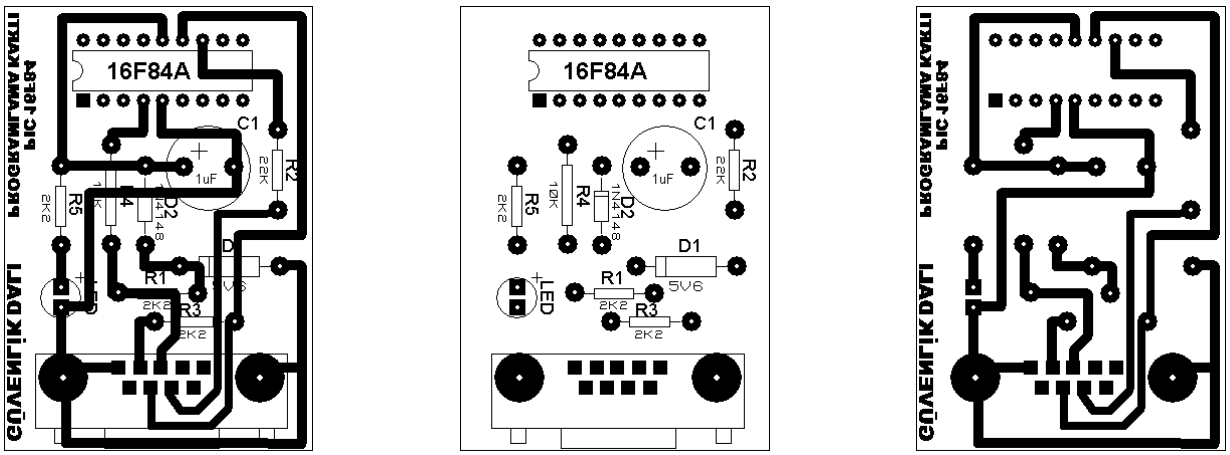
PIC 16F84 Programlama Kartı Malzeme Listesi

R1,R3,R5= 2,2 K	R2= 22 K	R4= 10 K	C1= 1 µF
D1(zener)= 5,6 V	D2= 1N4148	D3= Led diyot	
J1= DS9P(Seri dişi port)	5X6 cm Bakır plaket	U1= 18 Pin IC soket + PIC16F84	

NOT: Her öğrenci bu uygulamayı yaparken yanında iş önlüğü, el aletleri, havya, baskı devre kalemi lehim, ölçü aleti, ince zımpara, 1mm matkap ucu bulundurması gerekir.

PIC16F84 Programlama Kartı Baskı Devre Üstten Görünüşleri

Malzemeler + Yollar **Malzemeler** **Yollar**



Devre Elemanlarını Baskı Devre Üzerine Monte Etme

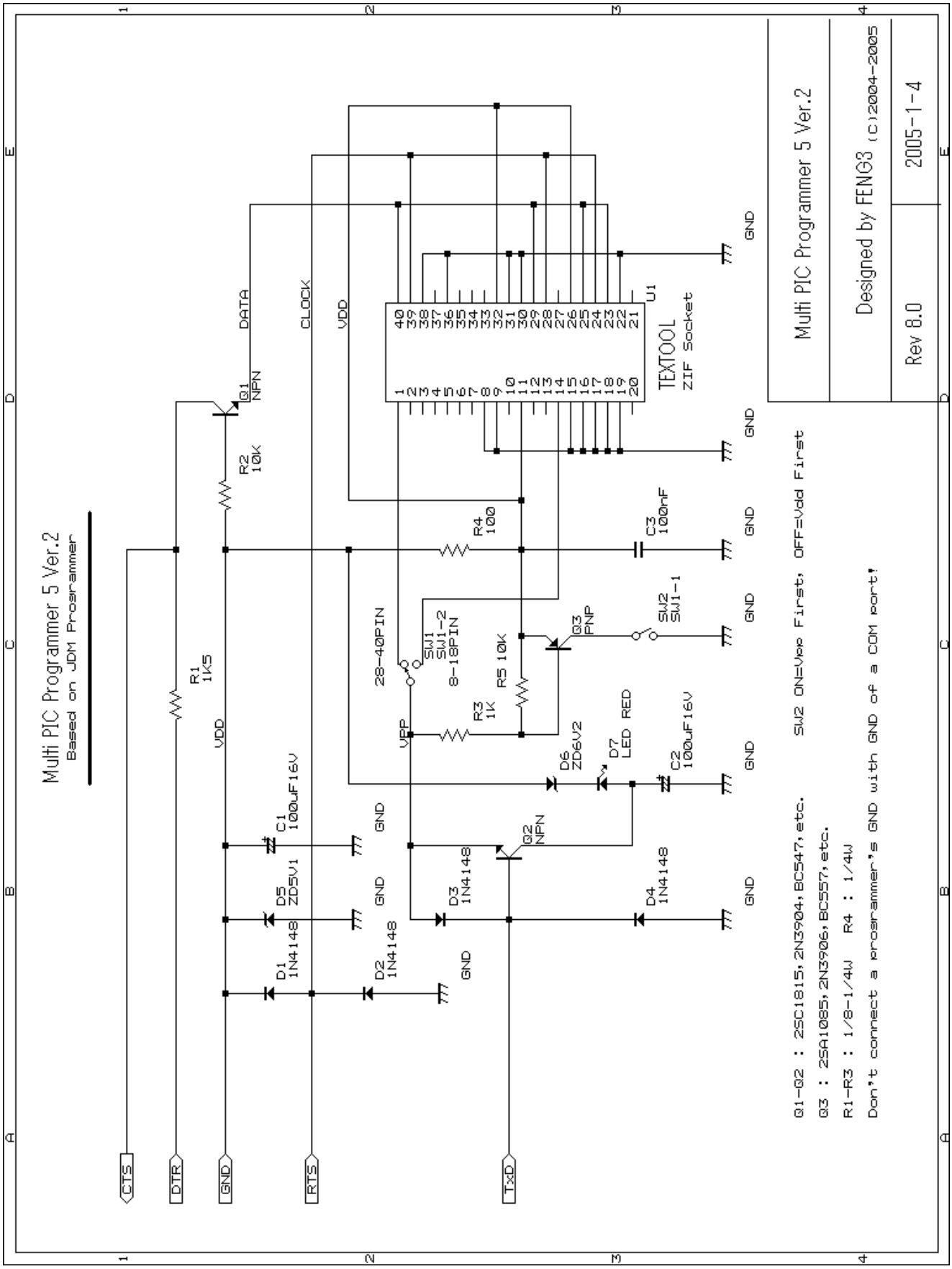
İşlem Basamakları

- Bakır plaket üzerindeki bağlantı yollarını test ediniz.
- Malzemelerinizin sağlamlık kontrolünü yapınız.
- Bağlantı noktalarına göre,malzemelerinizin ayak uzunluklarını belirleyiniz.
- Havyanızı uygun sıcaklığa getiriniz.
- Malzemelerinizi üst görünüş şemasına göre bakır plaket üzerine lehimleyiniz.

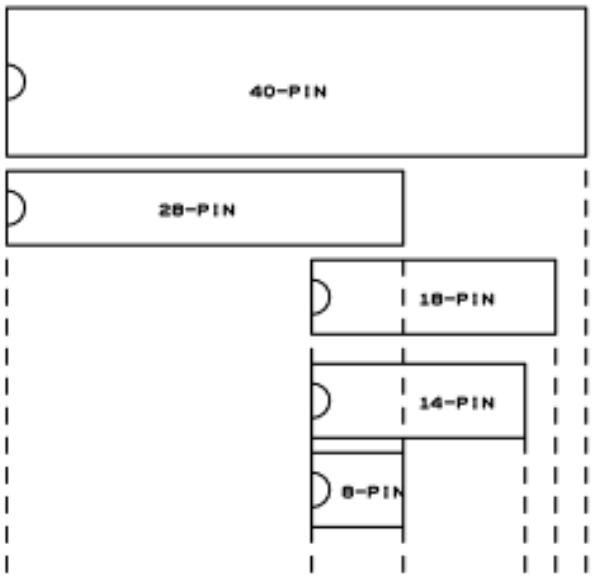
Öneriler

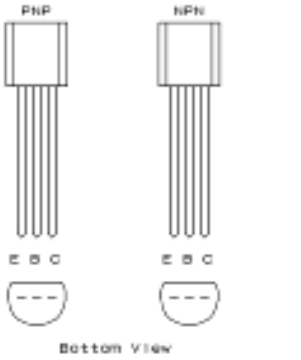
- Bağlantı yollarını şemadan takip ederek en uç noktalarıyla irtibatlı olup olmadığına bakabilirsiniz.
- AVO metre ile sağlamlık kontrolü yapılabilecek elamanları ölçebilirsiniz.
- Elemanların bağlantılarını dik veya yatay yapabilirsiniz.
- Elektronik malzemelerin montajı yapılırken 30 W'lık havya kullanabilirsiniz.
- Entegrelerin sıcaktan etkilenmesini önlemek için yalnız soketleri lehimleyebilirsiniz.

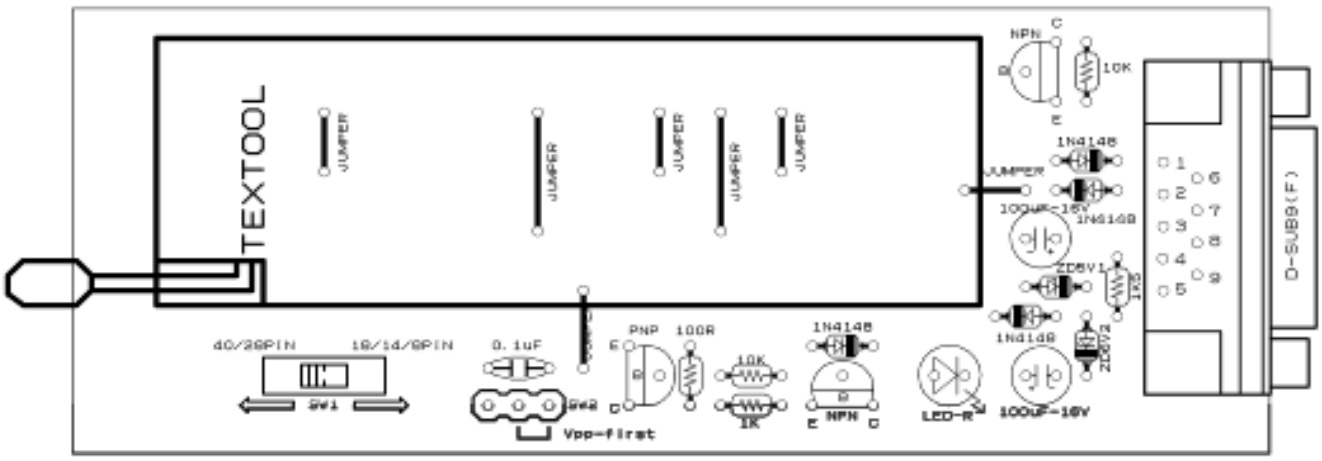
ÇOKLU PROGRAMLAMA KARTI AÇIK ŞEMASI



# Multi PIC Programmer
# Version5.2
Rev8.0(2005-1-4)





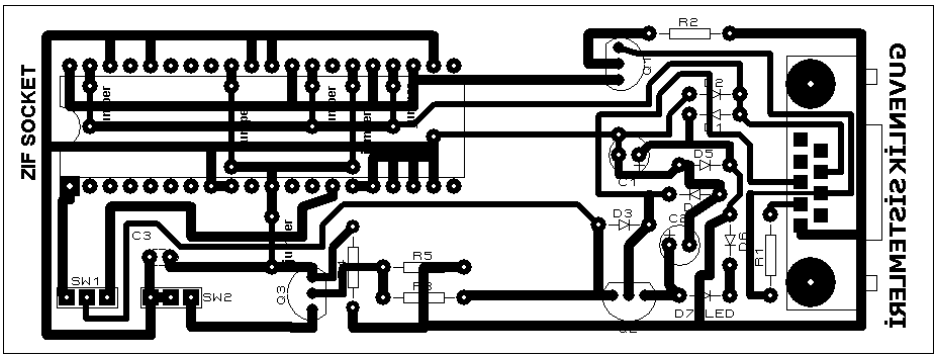


Çoklu PIC Programlama Kartı Malzeme Listesi

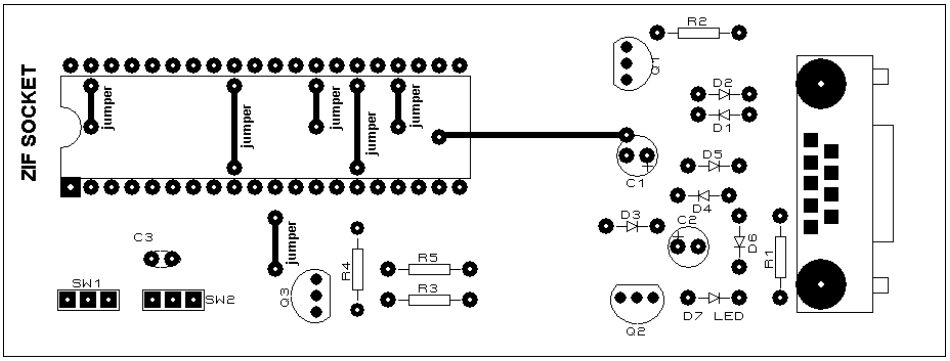
R1= 1K5	R2= 10 K	R3= 1 K	R4= 100 Ω
C1,C2= 100 µf 16V	C3= 100 nF	D1,D2,D3,D4,= 1N4148	
D5(zener)= 5V1	D6(zener)= 6V2	D7= Led diyot	
Q1,Q2= BC547	Q3= BC557	DS9P(Seri dişi port)	
ZIF SOCKET 40' lı	5X10 cm Bakır plaket		

Çoklu PIC Programlama Baskı Devre Üstten Görünüşleri

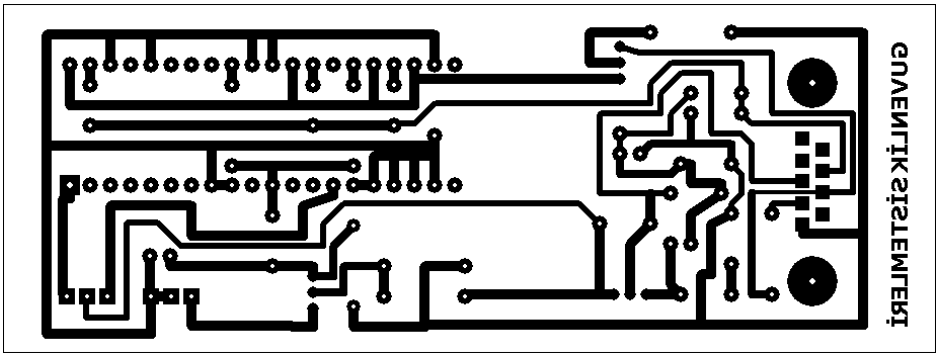
Malzemeler + Yollar



Malzemeler



Yollar



DEĞERLENDİRME ÖLÇÜTLERİ

1. Programlama kartının şemasına göre baskı devresini tekniğe uygun çıkarttınız mı?
2. Kart için gerekli malzemeleri doğru ve eksiksiz tespit ettiniz mi?
3. Devre elemanlarının sağlamlık kontrolünü yaptınız mı?
4. Kart üzerinde bulunan elemanların yerleşimini ve montajını tekniğine uygun olarak yaptınız mı?
5. Kart üzerine yerleştirdiğin devre elemanlarının lehimlemesini tekniğe uygun olarak yaptınız mı?
6. Devre elemanlarının montajı tamamlanan kartın denemesini yaptınız mı?

PROGRAMLARIN DERLENMESİ (MPASM VEYA MPLAB)

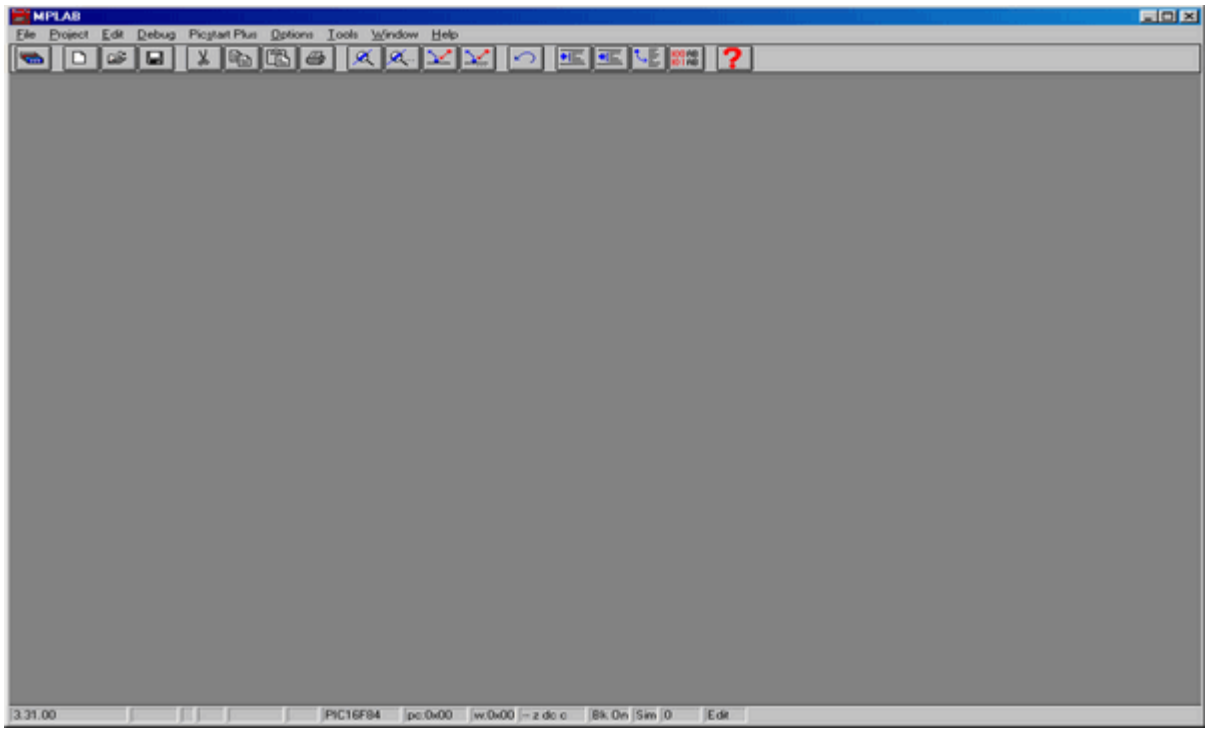
*.ASM uzantılı kaydedilen kaynak programı derlemek için gerekir.

MPLAB Kullanımı

Bu bölümde MPLAB Ver 3.31.00 versiyonunun çalıştırılması ve kullanımı ana hatlarıyla anlatılacaktır. MPLAB programının diğer versiyonları da bu versiyon ile menü açısından benzerlik göstermektedir. Bu sebeple bu bölüm iyi anlaşıldığında, programın diğer versiyonları da kullanılabilir. Programı çalıştırmak için: Başlat/Programlar/Microchip Mplab/MPLAB seçilir.

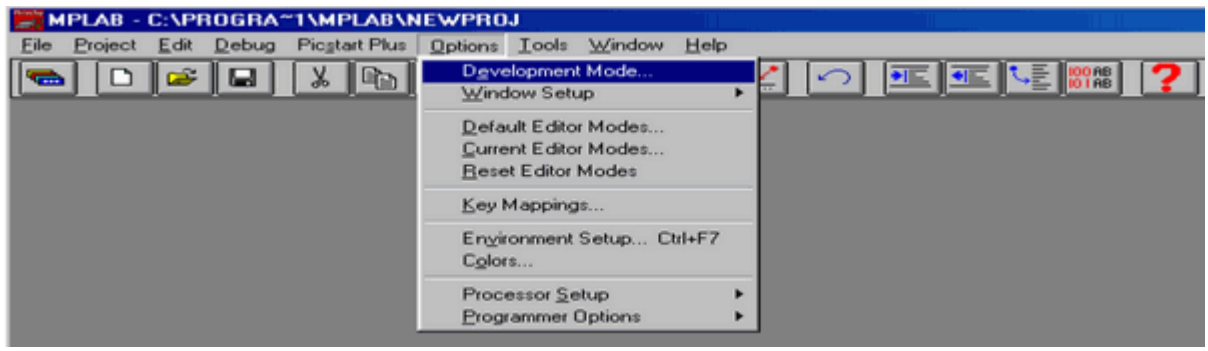
Bu seçim yapıldığında aşağıdaki ekran karşımıza gelir.

MPLAB açılış sayfası

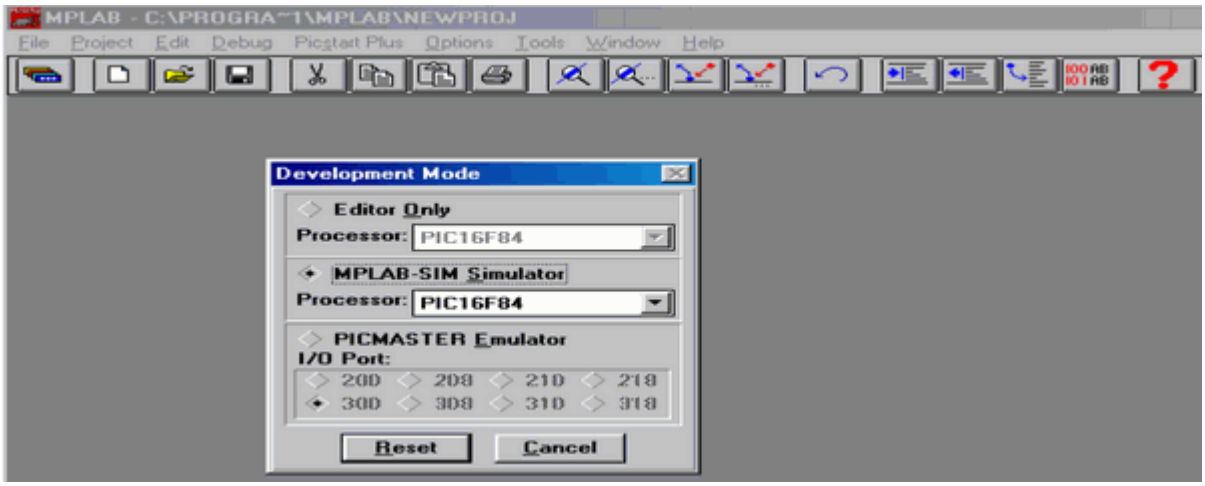


Programı ilk kurduğunuzda yapılması gereken 2 adet ayar vardır. Bu ayarlar kullanacağınız işlemciyi tanıtmak ki; biz pic16f84 tanıtacağız. Bunun için; Options/Development Mode seçilir.

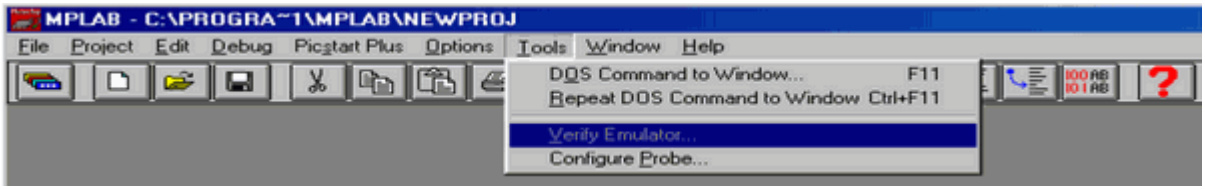
MPLAB'ın ilk çalıştırılması



Gelen ekrandan kullanacağınız işlemciyi belirleyiniz (Bizim örneğimizde Pic16f84)

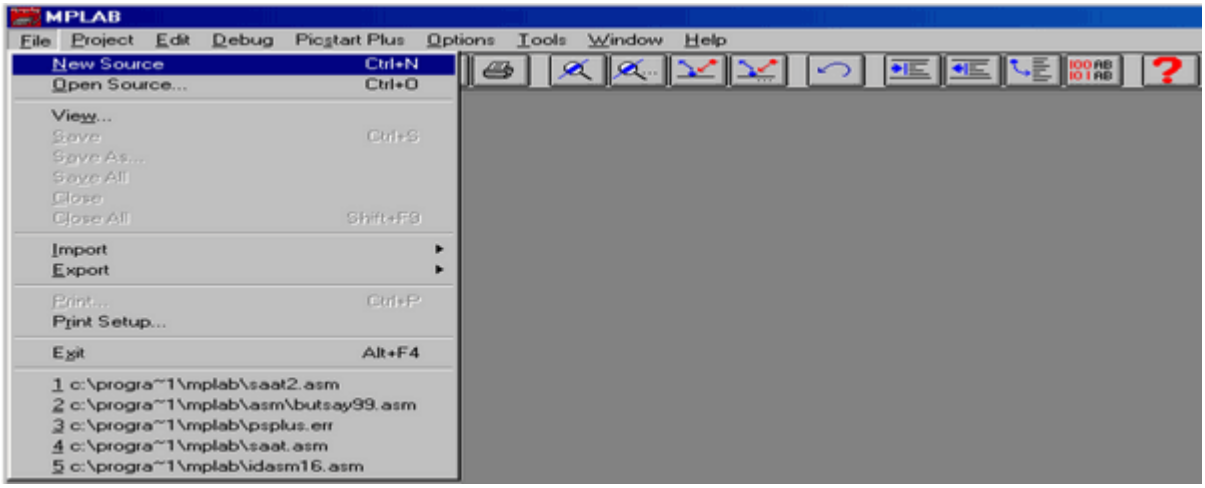


Yapılması gereken diğer ayar ise Tools/Verify Emulator işlemidir ki; burada bu seçildikten sonra gelen mesajlara olumlu (YES/OK) cevaplar vererek arada gelecek olan üçlü menüden SIMULATOR seçeneği seçildikten sonra, yine gelen mesajlara olumlu cevaplar verilerek işlem tamamlanır. Bu işlemi yapmak bize yazacağımız programı simulasyon modunda çalıştırma imkanı verir.



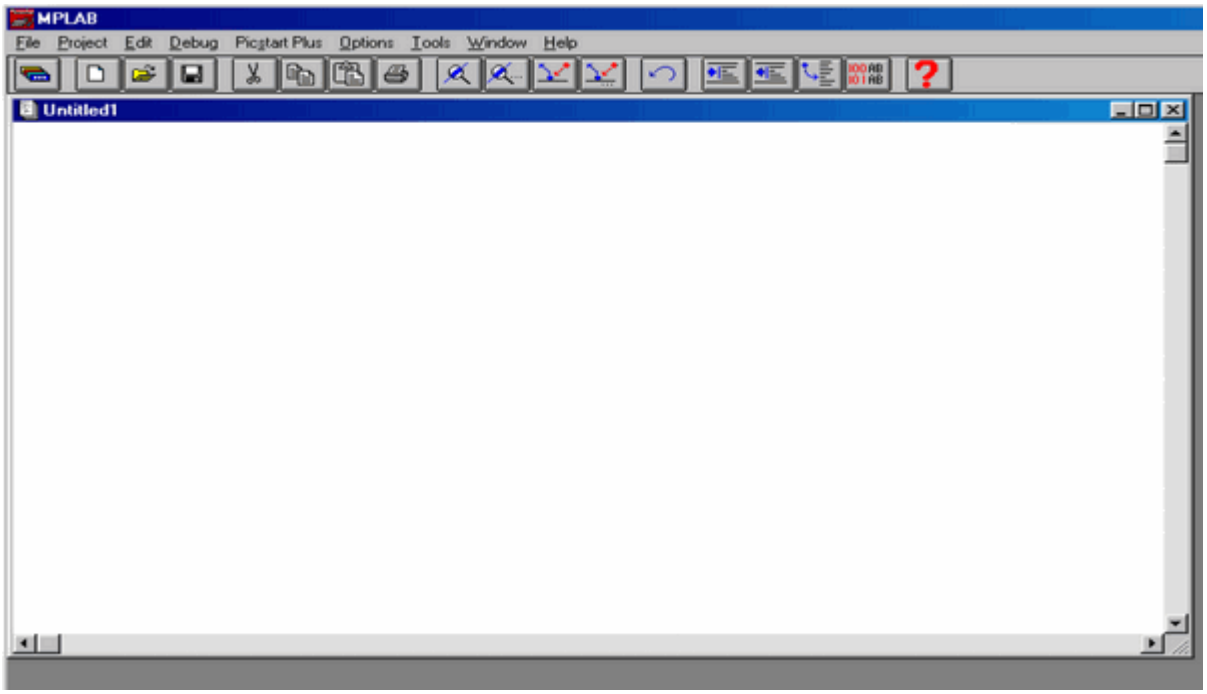
Yeni programınızı yazabilmek için boş bir sayfa açmalısınız. Bunun için; File/New Source seçiniz.

MPLAB'da yeni sayfa açılması



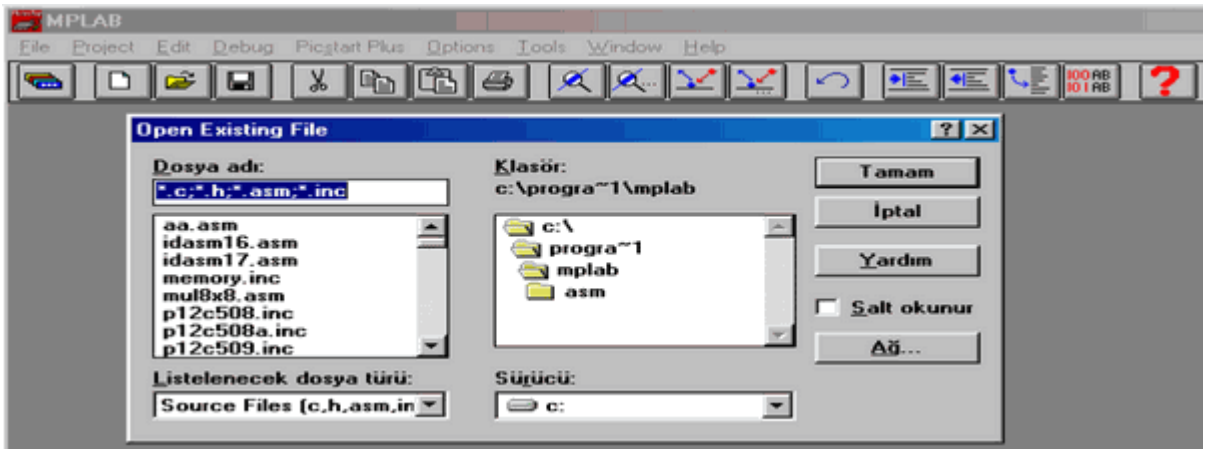
Bu işlemden sonra aşağıdaki boş sayfa karşınıza gelir ve programınızı buraya yazabilirsiniz.

MPLAB'da yeni sayfa açılması



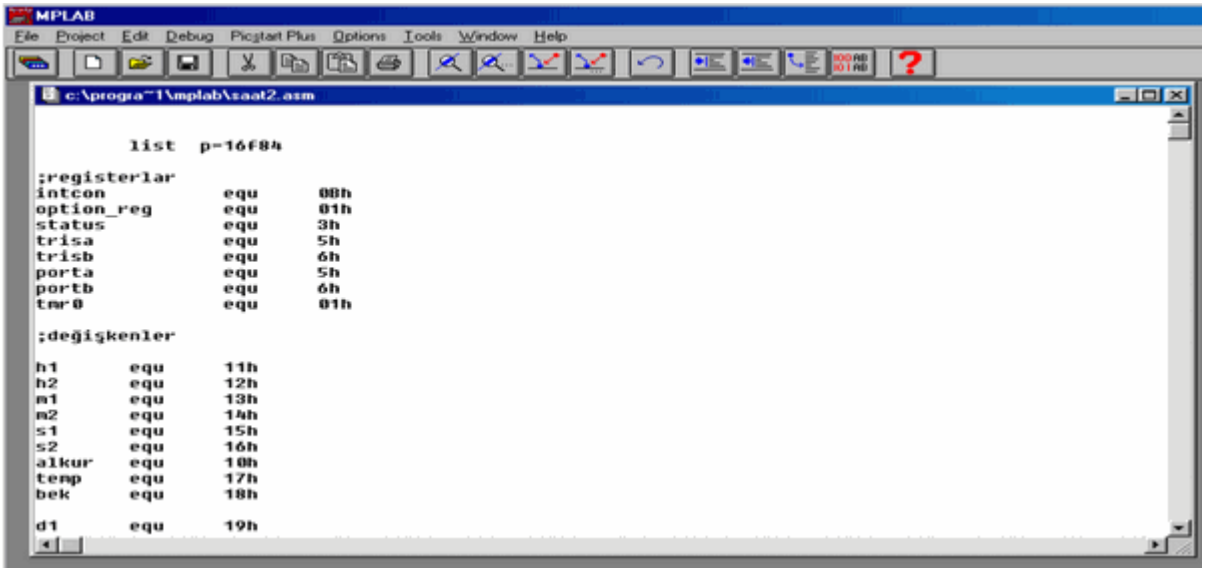
Eğer daha önceden yazıp kaydettiğiniz bir dosyayı açacaksanız;
File/Open Source seçiniz ve gelen ekrandan klasör ve dosya isimlerini seçerek dosyanızı ekrana getirebilirsiniz.

MPLAB'da önceden yazılmış programın açılması



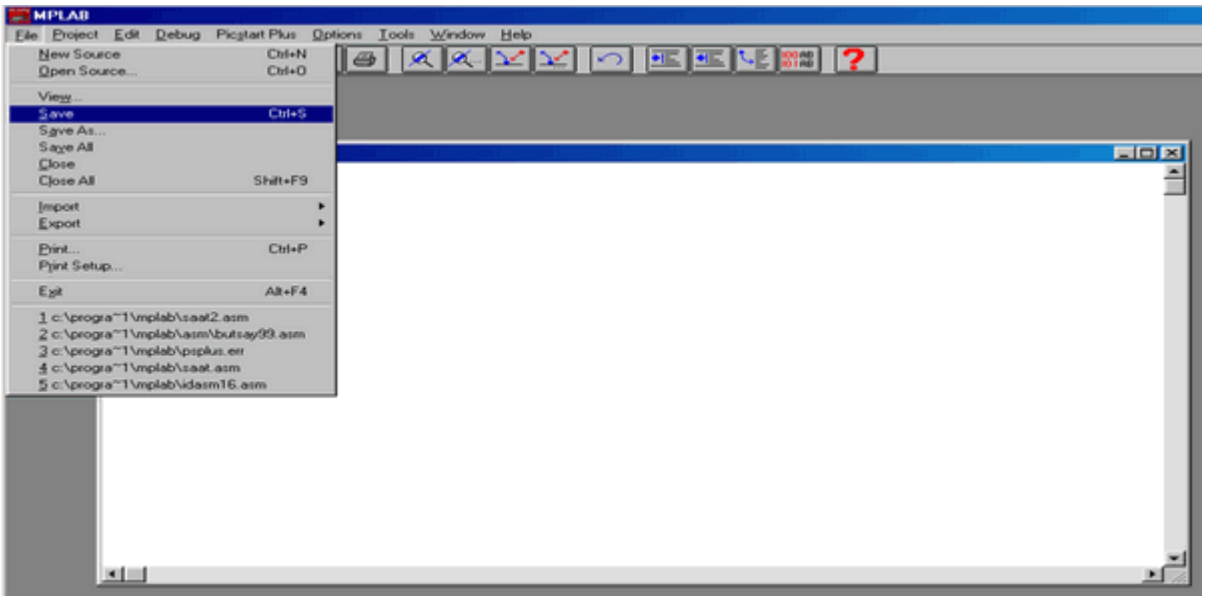
Eski dosyanızı çağırdığınızda veya yeni yazdığınız programın yazımını tamamladığınızda aşağıdaki şekilde bir görüntü oluşacaktır.

MPLAB'da program yazımı

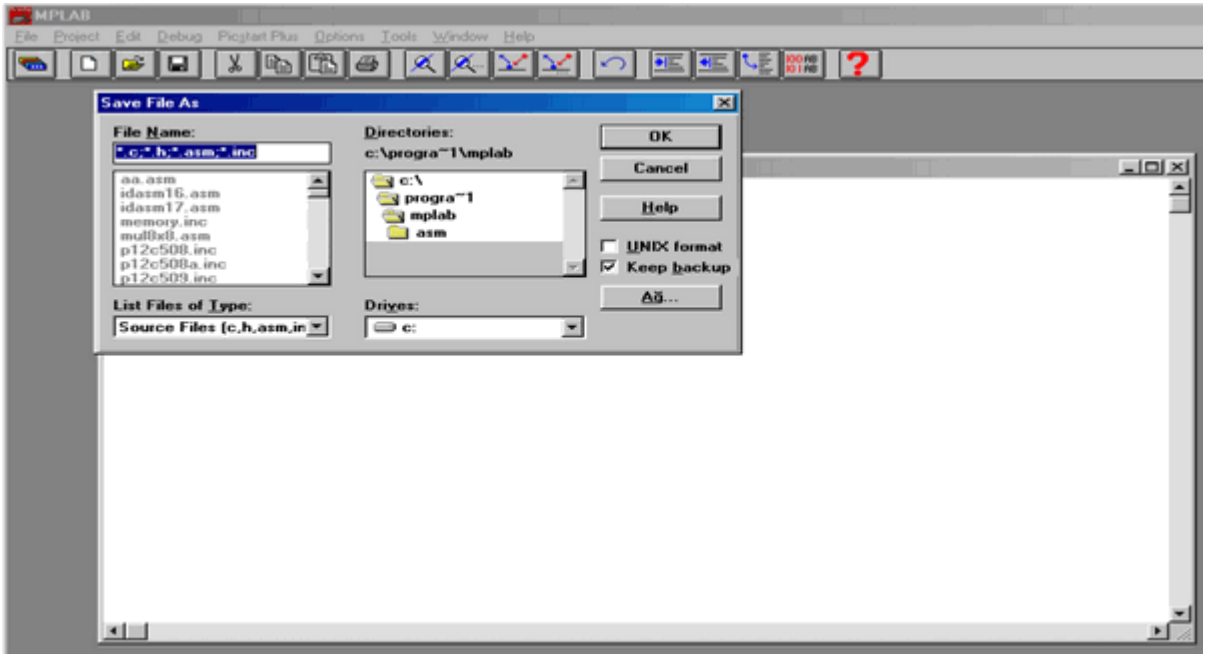


Ancak programı yeni yazdıysanız başlık satırında isim yerine UNTITLED yazısı görünecektir. Programınızı yeni yazdıysanız isim vererek kaydetmelisiniz. Kayıt işlemi için File/Save seçeneğini seçiniz.

MPLAB'da yazılan programın kaydedilmesi

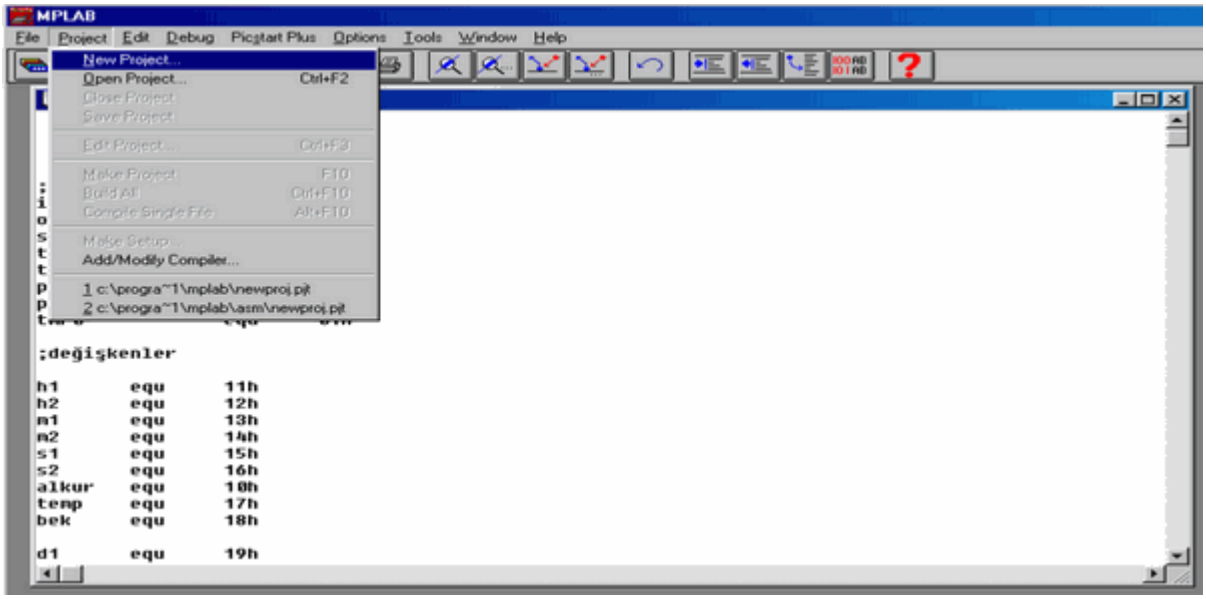


Gelen ekranda seçili klasör, sizin istediğiniz klasör değilse, Windows işletim sisteminin kullanım metoduyla, seçili klasörü istediğiniz şekilde değiştirebilirsiniz. Ancak daha sonra seçtiğiniz klasörün isminin lazım olacağını unutmayınız. File Name kısmına ise istediğiniz ismi veriniz; ancak uzantı ismi olarak asm vermeyi unutmayınız. Örnek SAAT.ASM gibi…

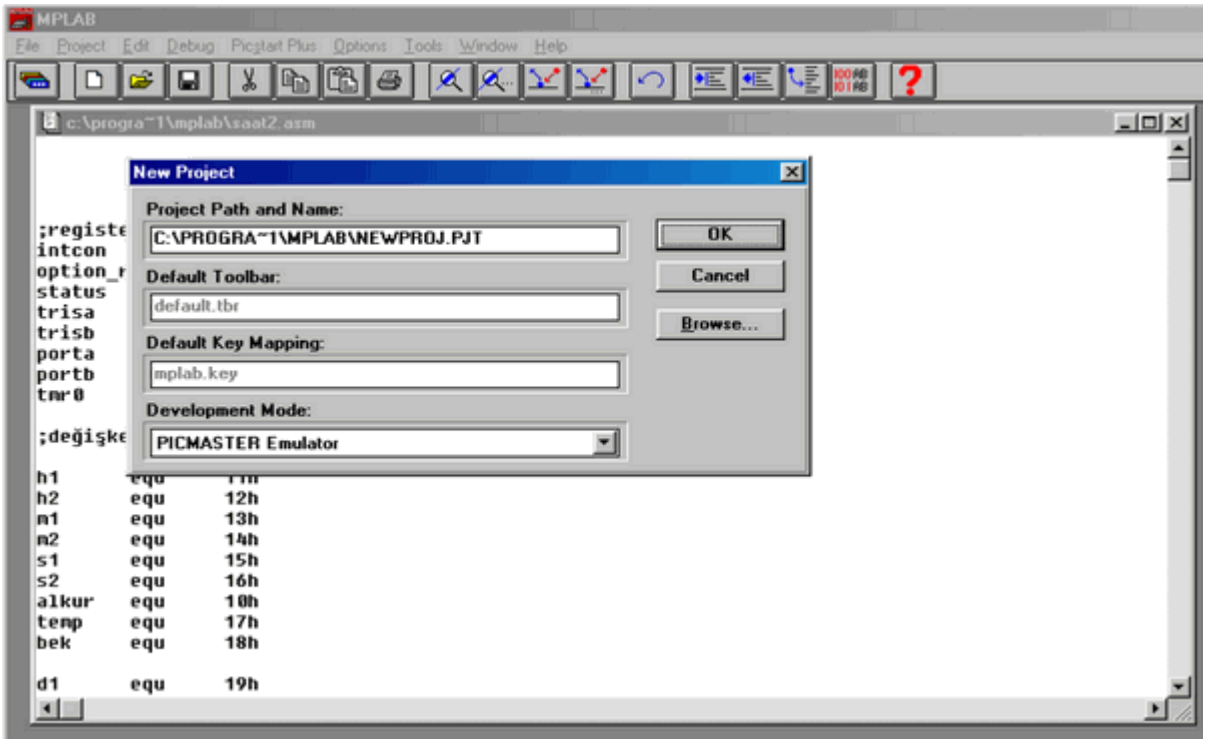


Dosya oluşturduktan sonra sıra proje oluşturma aşamasına gelecektir. Proje oluşturma aşamasında ise ilk iş;
Project/New Project seçeneğini seçmektir.

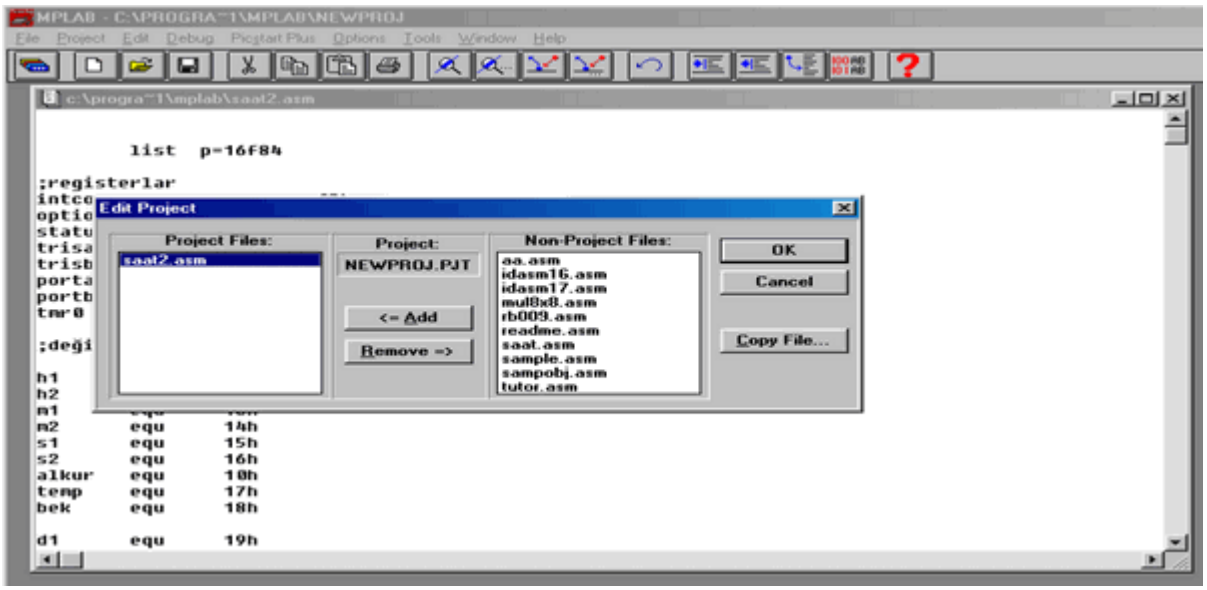
MPLAB'da proje oluşturulması



Bu seçim yapılınca proje ismi vermek için aşağıdaki ekran gelir. Burada NEWPROJ.PJT ismi kendiliğinden verilir. İstenilirse proje ismi değiştirilir. Bu değiştirme yapılırken ileride sorun yaşamamak için, proje adına PJT uzantısının yazılması ve klasör adının değiştirilmemesine özen gösterilmelidir.

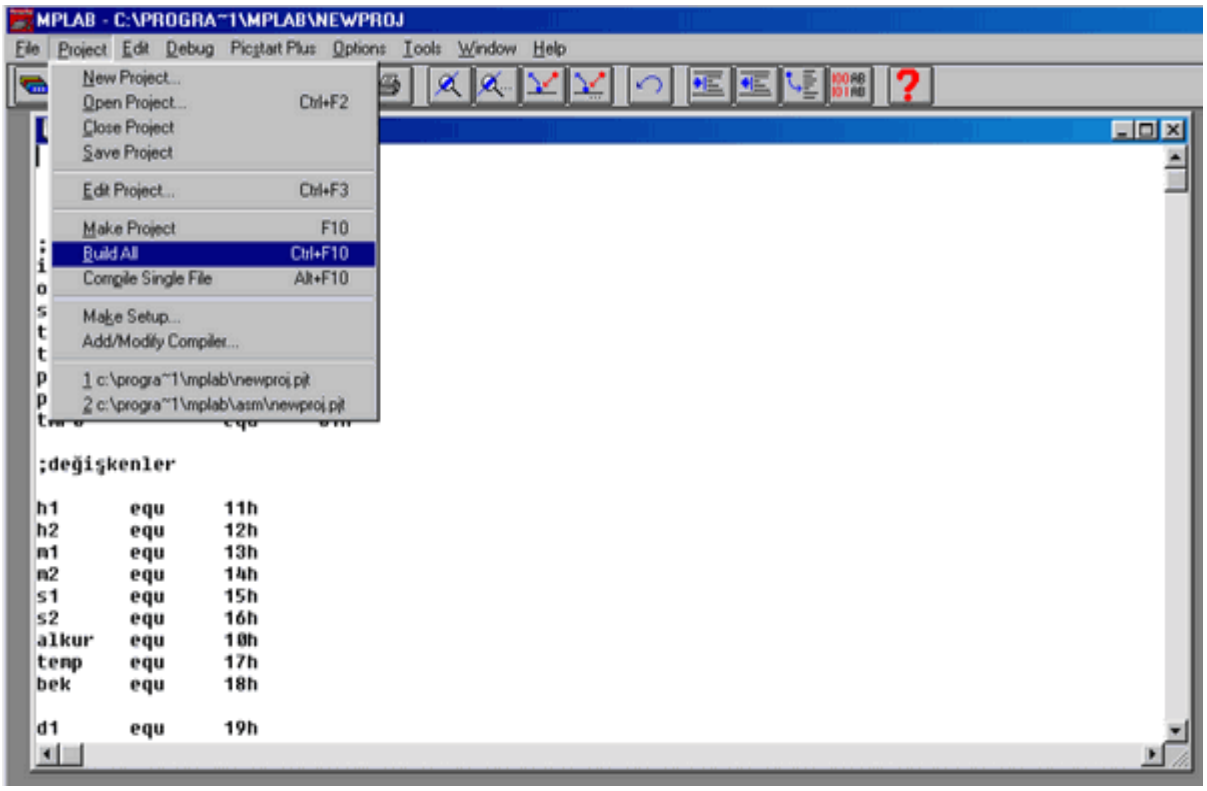


Bu aşamadan sonra Edit Project ekranı gelir. Burada iki ayrı bölüm vardır. Project Files ve Non Project Files alanları. Burada Project Files alanına projede olmasını istediğimiz dosyanın adını diğer taraftan çift tıklayarak veya seçip add seçeneğini tıklayarak aktarırız. Bu işlemden sonra tamam (Ok) diyerek işlemi tamamlamış oluruz. Proje içerisindeki dosya adı ile ilgili bir sorun olursa Project/Edit Project seçeneği ile bu bölüme tekrar gelebiliriz.

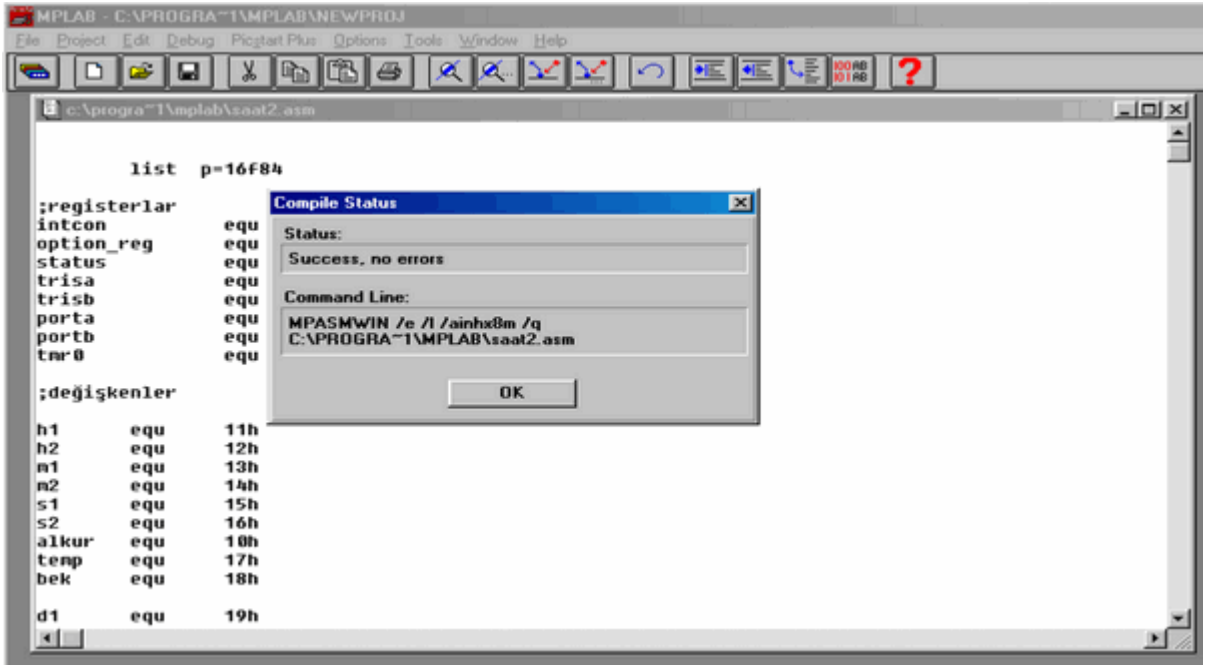


Şimdi sıra yazdığımız programın derlenmesine gelmiştir. Derleme esnasında hatalar belirlenip bize mesaj olarak verilir. Ayrıca uzantısı “hex” olan bir dosya ile de yazılı programın çalışır kodları üretilir. Bu işlem için;
Project/Build All seçilir.

MPLAB'da oluşturulan projenin derlenmesi



Bu seçimden sonra ekranda hareketli band olan bir pencere gelir ve derlemenin yapıldığını gösterir. İşlem bittiğinde aşağıdakine benzer bir pencere gelir. Eğer şekildeki gibi Success No Error mesajı varsa programda yazım hatası yok demektir ve program denemeye hazır demektir. Eğer There are Errors şeklinde bir mesaj gelirse hatalar uzantısı "err" olan bir dosya da kayıtlıdır. Bu dosyayı açarak hataların nerelerden kaynaklandığına bakar ve düzeltiriz. Bu hatalar dosya adı, satır numarası ve hata cinsi şeklinde satır satır belirtilir. Burada satır numarasının kaç olduğunu ekranın altındaki durum satırından takip ederek hataları düzeltme yoluna gidilir.



Programda hata yoksa artık sıra deneme işlemine gelmiştir

PIC 16F84
DENEY KARTI
UYGULAMASI

2010-2011

ALANI : ELEKTRİK ELEKTRONİK TEKNOLOJİSİ
DALI : GÜVENLİK SİSTEMLERİ

ADI SOYADI :
SINIFI / NO :

DEĞERLENDİRME

İŞLEM BASAMAKLARI	MONTAJ	İŞ ALIŞKANLIKLARI	SÜRE

MİKRODENETLEYİCİ DENEME KARTI

Programladığınız PIC breadboard üzerinde kendi kurduğunuz devre de deneyebileceğiniz gibi sekilde görülen özel bir deneme kartı üzerinde de deneyebilirsiniz. Baslangıç ve orta düzey PIC programlayıcılara hitap etmek üzere gelistirilen bu kart üzerinde deneme yapmak, breadbord üzerinde devre kurmaktan çok daha kolaydır. PIC'in port çıkıslarındaki sinyalleri izlemek amacıyla 8 tane LED, bir tane de 7 segmentli LED yerlestirilmistir. Kart üzerindeki, butonlar, LED'ler aracılığıyla farklı şekilde programlanan PIC'lerin kolayca denenmesini sağlanır.

Mikrodenetleyici deneme kartı, programlanmış mikrodenetleyicinin çalışmasının gözlemlendiği karta deneme kartı denir. Deneme kartlarında giriş olarak push buton ve anolog giriş için potansiyometre seçilebilir. Çıkış değerlerini görmek için led, display ve lcd ekran kullanılabilir.

PIC 16F84 Deneme Kartı Malzeme Listesi

1. IC1= PIC 16F84
2. 18 Pin IC soket (2 adet)
3. 7805 Regüle Entegresi
4. Ortak katotlu display
5. S0…S5 Push buton (6 adet)
6. Led (9 Adet)
7. R1..R8, R15= 220Ω (9 Adet)
8. R9…R14= 10 KΩ (6 adet)
9. D2= 1N4148
10. C1,C2= 22 pF (2 adet)
11. C3,C4 = 10 µF 25 Volt (2 adet)
12. Kristal = 4 Mhz
13. D1=1N4001
14. 10 x 8 cm Bakır plaket

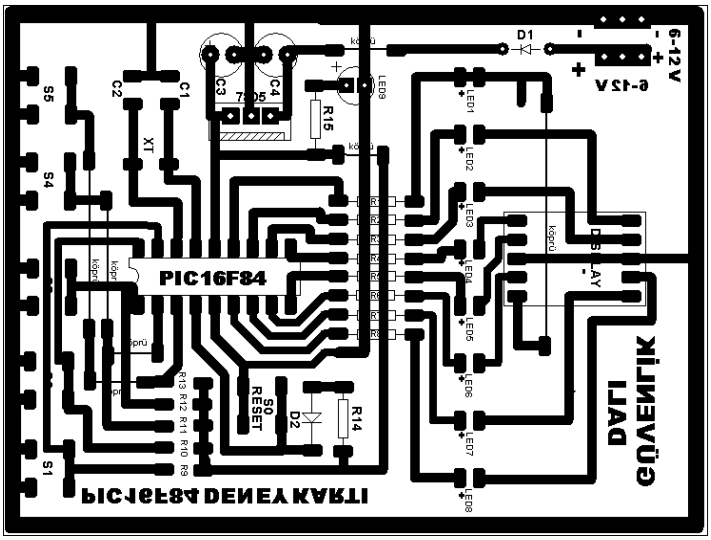
NOT: Her öğrenci bu uygulamayı yaparken yanında iş önlüğü, el aletleri, havya, baskı devre kalemi lehim, ölçü aleti, ince zımpara, 1mm matkap ucu bulundurması gerekir.

Devre Elemanlarını Baskı Devre Üzerine Monte Etme

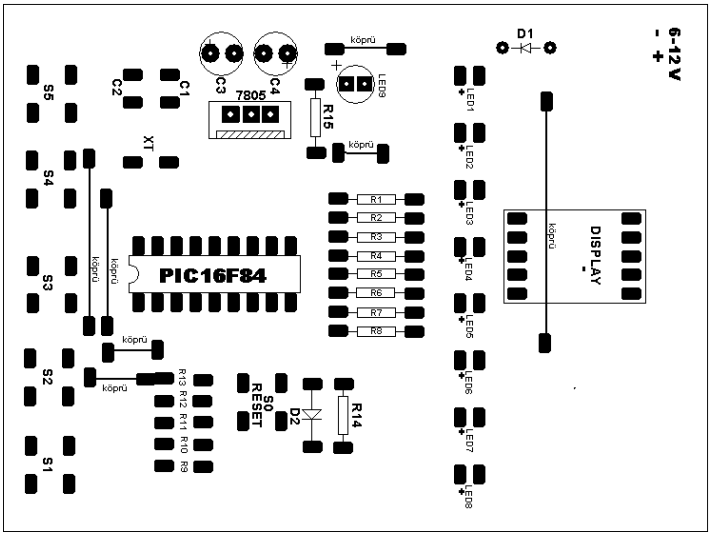
İşlem Basamakları	Öneriler
• Bakır plaket üzerindeki bağlantı yollarını test ediniz.	• Bağlantı yollarını şemadan takip ederek en uç noktalarıyla irtibatlı olup olmadığına bakabilirsiniz.
• Malzemelerinizin sağlamlık kontrolünü yapınız.	• AVO metre ile sağlamlık kontrolü yapılabilecek elamanları ölçebilirsiniz.
• Bağlantı noktalarına göre,malzemelerinizin ayak uzunluklarını belirleyiniz.	• Elemanların bağlantılarını dik veya yatay yapabilirsiniz.
• Havyanızı uygun sıcaklığa getiriniz.	• Elektronik malzemelerin montajı yapılırken 30 W'lık havya kullanabilirsiniz.
• Malzemelerinizi üst görünüş şemasına göre bakır plaket üzerine lehimleyiniz.	• Entegrelerin sıcaktan etkilenmesini önlemek için yalnız soketleri lehimleyebilirsiniz.

PIC16F84 Deneme Kartı Baskı Devre Üstten Görünüşleri

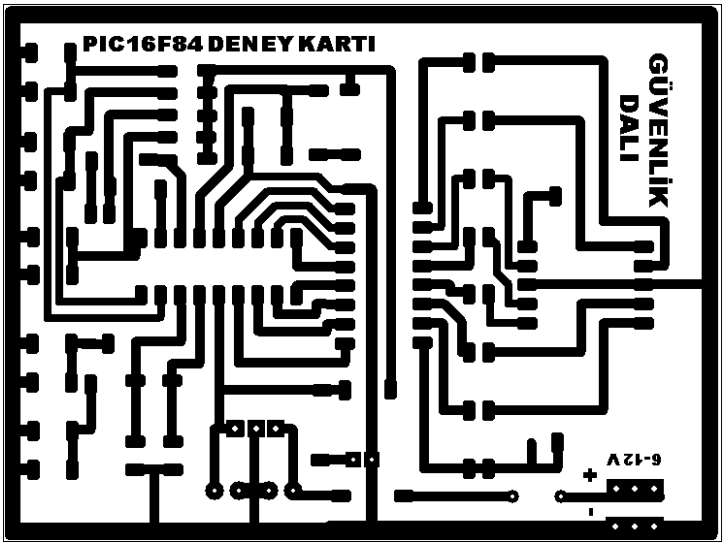
Malzemeler + Yollar (üstten görünüş)



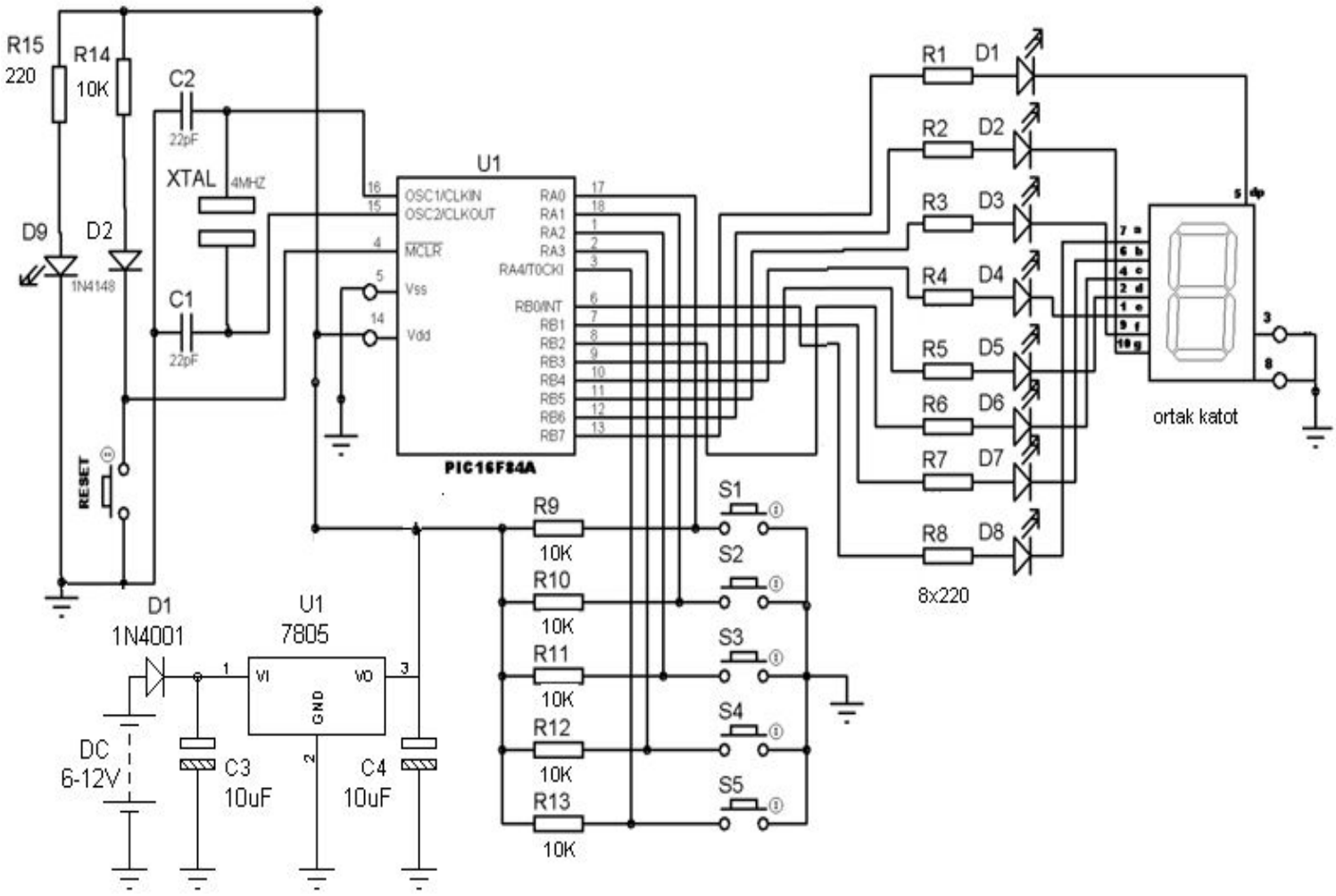
Malzemeler (üstten görünüş)



Yollar (alttangörünüş)



PIC 16F84 DENEME KARTI AÇIK ŞEMASI



DEĞERLENDİRME ÖLÇÜTLERİ

7. Deneme kartının şemasına göre baskı devresini tekniğe uygun çıkarttınız mı?

8. Kart için gerekli malzemeleri doğru ve eksiksiz tespit ettiniz mi?

9. Devre elemanlarının sağlamlık kontrolünü yaptınız mı?

10. Kart üzerinde bulunan elemanların yerleşimini ve montajını tekniğine uygun olarak yaptınız mı?

11. Kart üzerine yerleştirdiğin devre elemanlarının lehimlemesini tekniğe uygun olarak Yaptınız mı?

12. Devre elemanlarının montajı tamamlanan kartın denemesini yaptınız mı?

MPASM 'NIN ÜRETTİĞİ DİĞER DOSYALAR

Editör ile hazırladığınız kaynak program (DENEME.ASM) derlendikten sonra .HEX uzantılı bir dosya elde edildiğini gördük. MPASM bu dosyanın dışında (LST, ERR, COD, HXI-l, HXL, XRF) uzantılı dosyaları da oluşturarak yine aynı klasör içerisine kaydeder. Bu dosyaların ismi kaynak program ile aynıdır. Çoğu zaman kullanmak zorunda kalacağımız LST ve ERR dosyaları, assembly komutlarını yazarken yapılabilecek yazım hatalarını bulmakta çok yararlıdır.

.LST Dosyası

.LST dosyasını kullandığınız editörden açabilirsiniz. Bu dosya iki sayfadan oluşmuştur. llksayfada (PAGE 1);

- Komutun program belleği veya RAMdaki adresi (LOC),
- Komutların hexadesimal karşılıkları (OBJECT),
- Kaynak program ve satır numaralan (LINE SOURCE TEXT) verilir. İkinci sayfada ise (PAGE 2);
- Programda kullanılan etiketler (SYMBOL TABLE) ve adresleri (VALUE),
- Bellek kullanım haritası (X'ler kullanılan alanı, -' ler kullanılmayan alanı gösterir.)
- Kullanılan ve boş kalan alanın miktarı (DENEME.LST dosyasında kullanılan alan=5, kullanılmayan alan=1 019),

- Hata (Error) sayısı, uyarı (Warning) sayısı verilir.

.ERR Dosyası

Bu dosyayı da kullandığınız editörden açabilirsiniz. Eğer assembly komutlarının yazılışında bir hata yaptıysanız, hatalı satır numarası ve hatanın ne olduğu yazılıdır. Hatasız programlarda bu dosyanın içi boştur.

INCLUDE DOSYALARI

Assembly programlarını yazarken kullanılacak olan registerlerin adreslerini tanımlama bölümünde kullanmak, programı daha anlaşılır hale getirmektedir. Aslında P1C16F84 mikrodenetleyicisinin RAM belleğindeki özel registerlerin adresleri sabittir. öyleyse adresleri sabit olan registerleri her programı yazarken yeniden tanımlamak gereksiz görülmektedir. Microchip bu durumu göz önüne alarak INCLUDE dosyalarını kullanıma sunmuştur.

Header file denilen bu dosyaları kullanmak suretiyle programları yazarken her defasında register adreslerini tanımlama zorunluluğu kaldırılmıştır. MPASM ile derlenebilecek her PIC için ayrı bir INCLUDE dosyası bulunmaktadır Şimdi DENEME.ASM kaynak dosyasını INCLUDE dosyası kullanarak yeniden yazalım. Kodların daha da azaldığını göreceksiniz.

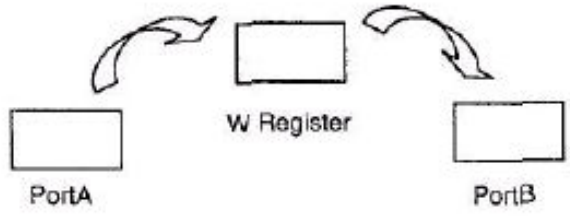
```
;===DEMEME.ASM====27/04/2000======
        LIST        P=16F84          ;PIC16F84'ün MPASM'ye tanıt.
        INCLUDE     'P16F84.INC'
        CLRF        PORTB            ;PORTB'ye bağlı LED'leri söndür
        BSF         STATUS, RP0      ;BANKl'e geç
        CLRF        TRISB            ;PORTB'nin uçlarını çıkış yap
        BCF         STATÜS, RP0      ;BANKO'a geç
        BSF         PORTB, 0         ;PORTB'nin 0.bitindeki LED'i yak
        END                          ;Program komutlarının sonu
```

VERİ TRANSFERİ ve KARAR İŞLEMLERİ

W REGİSTERİNİN KULLANIMI (MOVLW, MOVWF KOMUTLARI)
BİT TEST EDEREK KARAR VERME (BTFSC, BTFSS>

W REGİSTERİN KULLANIMI (MOVLW, MOVWF KOMUTLARI)

W register, RAM içerisindeki file registerlerden bağımsız bulunmaktadır. Registerler arasında veri transferi yapmak için kullanılır. örneğin, PortA içerisindeki veriyi PortB içerisine transfer etmek için aşağıdaki komutları yazmak gerekir.




```
MOVF PORTA, W     ;PortA'nın içeriğini W registere tası

MOVWF PORTB       ;W registerin içeriğini PortB'ye gönder
```

PortB'ye bağlı olan 8 LED'in ilk dört tanesini yakacak olan veriyi göndermek için de aşağıdaki komutlar yazılmalıdır.

```
MOVLW H‘0F’       ;W registerine H OH yükle

MOVWF PORTB       ;W registeri içeriğini PortB'ye gönder
```

Eğer bir register içerisine H'00' verisi gönderilmek istenirse, o registerin içerisini CLRF komutu ile silmek daha pratik bir yöntemdir.

```
CLRF PORTB ;PortB'nin içeriğini sıfırla
```

PROGRAM-1) PIC'e enerji verildiğinde PortA'nın uçlarına bağlı butonlardan hangisi basılı tutulursa, PortB'de o butona karşılık gelen LED'i söndüren program.

```
;==PROGl.ASM====14/05/2000=======
                LIST   P=16F84
PORTA           EQU h'05"
PORTB           EQU h'06'
STATUS          EQÜ h'03'
TRISA           EOÜ h'85"
TRISB           EQU h'86'
CLRF            PORTB               ;PORTB'ye bağlı LED'leri söndür
BSF             STATUS, 5           ;BANKl'e geç
CLRF            TRISB               ;PORTB'nin uçlarını çıkış yap
MOVLW           h'FF'               ;W registere b'FF' yükle
MOVWF           TRISA               ;PortA'nın uçlarını. giriş yap
BCF             STATUS, 5           ;BANK='a geç
BASLA
MOVF            PORTA,W             ;PortA'yı oku, sonucu W'ye yaz
MOVWF           PORTB               ;Butonların durumunu PortB'de göster
DONGU
GOTO DONGU                          ;Sonsuz döngü ve program sonu
END
```

PROGRAM-2) A portunun uçlarına bağlı butonlara basıldığı sürece, B portunda o butona karşılık gelen LEDi söndüren program.

Bu programda A portundaki RAO-RA4 uçlarının devamlı olarak okunup B portuna gönderilmesi gerekiyor. Bunu devamlı olarak yapabilmek için programı "BASLA" etiketine gönderip, sonsuz döngüyü buraya kurmak gerekiyor.
Programın PROG1 .ASM ile arasındaki tek fark GOTO komutunun yazıldığı satır olacaktır.

PROG1.ASM' yi editörünüze yükleyerek GOTO ile başlayan satırı aşağıdaki gibi değiştiriniz. Yeni programı PROG2.ASM adıyla kaydediniz.

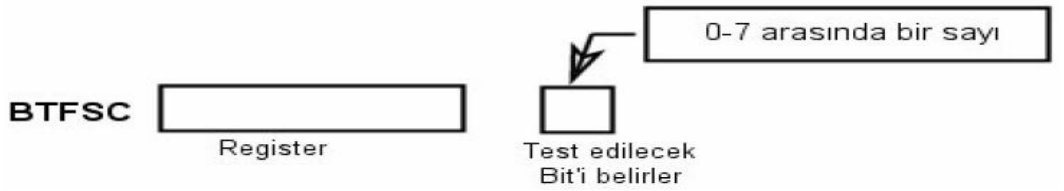
```
GOTO            BASLA           ;Okumayı tekrarla
```

BİT TEST EDEREK KARAR VERMEK (BTFSC, BTFSS)

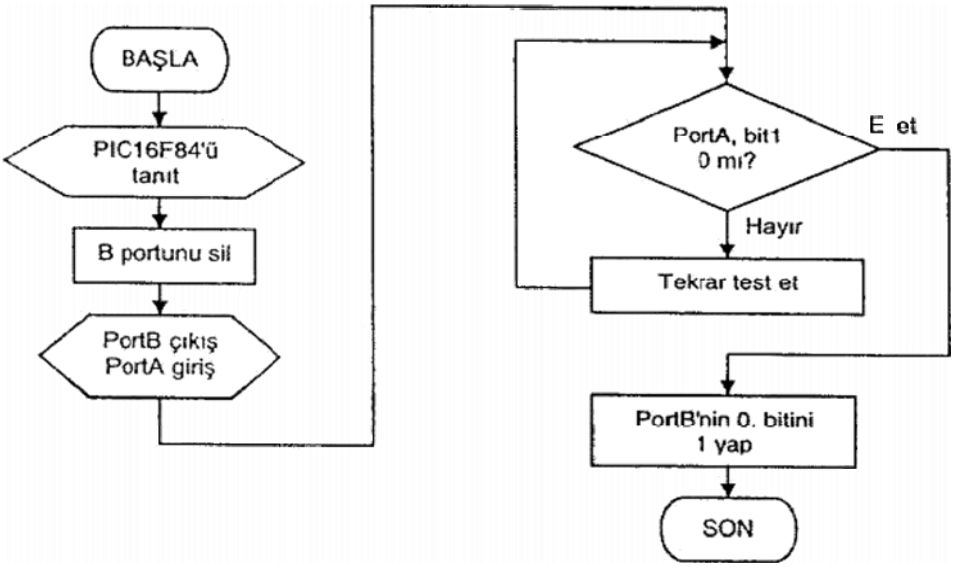
Bir register içerisindeki herhangi bir bit BTFSC veya BTFSS komutları ile test edilebilir. Bu test sonucuna göre program akışı istenilen komuta dallandırılabilir.

BTFSC komutu İngilizce'deki " Bit Test File register Skip if Clear", komutunun kısaltmasıdır. Türkçe'deki karşılığı "Registerdeki bit'i test et, eğer "O"sa bir sonraki komuta atla" biçimindedir.

BTFSS komutu ise "Bit Test file register Skip if Set" cümlesinin kısaltmasıdır. Türkçe karşılığı ise "Registerdeki bit'i test et, eğer "l"se bir sonraki komuta atla"dır.



PROGRAM-3) PortB'nin 0. bitine bağlı LED'i, A portunun 1. bitindeki butona basınca yakan program.




```
                        ;===PROG3.ASM====22/05/2000=======
            LIST P=16F84
PORTA       EQU h'05"
PORTB       EQU h'06'
STATUS      EQU h'03'
TRISA       EQU h'85'
TRISB       EQU h'86'
        CLRF    PORTB       ;PORTB'ye bağlı LED'leri söndür
        BSF     STATUS, 5   ;BANKl'e geç
        CLRF    TRISB       ;PORTB'nin uçlarına. çıkış yap
        MOVLW   h'FF'       ;W registere h'FF' yükle
        MOVWF   TRISA       ;PortA'nın uçlarını giriş yap
        BCF     STATUS, 5   ;BANKO'a geç
TEST_PORTA
        BTFSC   PORTA, 1    ;A portunun 1. bitini test et
        GOTO    TEST_PORTA  ;0 değilse tekrar test et
        BSF     PORTB, 0    ;B portunun 0. Bitini l(Bet) yap
DONGU
        GOTO DONGU
        END
```

PROGRAM-4) A portunun 2. bit'indeki butona basınca B portuna bağlı tüm LED'leri yakan program.

Bu programın PROG3.ASM'den tek farkı portB'ye gönderilecek olan verinin farklı b'00000001' yerine b'111111111') olmasıdır. BCF veya BSF komutlarıyla sadece bir bit'i 0(clear) veya 1(set) yapabiliyoruz. Bu nedenle bu komutun kullanıldığı yerde değişiklik yapmamız gerekiyor.

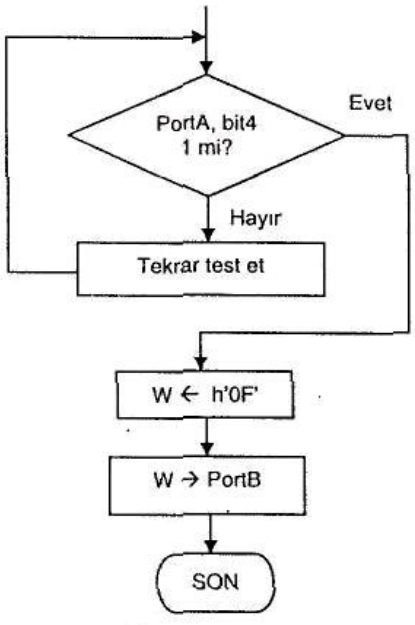
İşlem Basamakları

1. PROGS.ASM'yi editörünüze yükleyip "TEST_PORTA" etiketinden sonra yazılanları aşağıdaki gibi değiştiriniz. Yeni programı PROG4.ASM olarak kaydediniz.

```
TEST_PORTA
            BTFSC       PORTA, 2        ;PortA'nın 2. Biti 0(clear) mı.?
            GOTO        TEST_PORTA      ;Tekrar test et
            MOVLW       H'FF'           ;W registere b'11111111' yükle
            MOVWF       PORTB           ;W registeri portB'ye gönder
DONGU
            GOTO DONGU
END
```

PROGRAM-5) A portunun 4. bitine bağlı olan buton basılı tutularak PIC'e enerji verildiğinde, B portundaki LEDleri sönük tutan, butondan el çekildiğinde ilk dört LED'i yakan program.

Bu programın PROG4.ASMden farkı BTFSC komutu yerine BTFSS komutunu kullanmak zorunda oluşumuzdur. Çünkü A portunun 4. bitindeki butonu basılı tutuğumuzda buradaki veri O(clear)dır. Butondan elimizi çektiğimizde veri l(set)tir. Butona basılı durumdayken LED'lerin sönük kalabilmesini sağlamak için BTFSS komutuyla program akışı TEST_PORTA'etiketine gönderilmelidir. Buna göre akış diyagramının sadece değişen bölümünü yeniden çizelim.




```
TEST_PORTA
BTFSS       PORTA, 4    ;PortA'nın 4.bit'i l(set) mi?
GOTO        TEST_PORTA  ;Hayır, tekrar test et
MOVLV       H'0F'       ;Wregistere b'00001111'yükle
MOVWF       PORTB       ;W registeri portB'ye gönder
DONGU
GOTO DONGU
END
```

DÖNGÜ DÜZENLEMEK

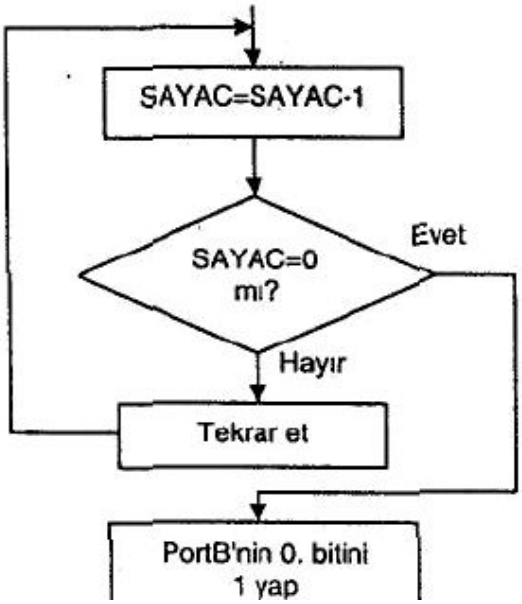
SAYAÇ KULLANARAK DÖNGÜ KULLANMAK (DECFSZ KOMUTU)
KARŞILAŞTIRMA İLE DÖNGÜ DÜZENLEMEK
(SUBLW, SUBWF KOMUTLARI)
STATUSREGİSTER

SAYAÇ KULLANARAK DÖNGÜ KULLANMAK (DECFSZ)

Bazı işlemlerin önceden belirlenen sayıda tekrarlanması gerekebilir. Bu durumda bir register sayaç olarak kullanılır. ilk önce sayaç içerisine işlemlerin tekrar sayısını belirleyen sayı yüklenir. Daha sonra her bir işlem tekrarında sayaç içerisindeki değer bir azaltılır. Azaltma işlemi DECFSZ komutu ile yapılır. Sayaç içerisindeki sayı
0 (sıfır) olunca döngü biter ve program ya bitirilir ya da başka bir komuta dallandırılarak devam eder.

DECFSZ komutu İngilizcedekı " Decrement file register Skip it Zero" dur. Türkçe'ye çevirirsek şu anlam çıkar; Registerden "1" çıkart, eğer sonuç "O sa bir sonraki komuta atla.

```
TEKRAR
        DECFSZ   SAYAC, F
        GOTO     TEKRAR
        BSF      PORTB, 0
```



ZAMAN GECİKTİRME ve ALT PROGRAMLAR

ZAMAN GECİKTİRME DÖNGÜLERİ
ALT PROGRAMLAR

ZAMAN GECİKTİRME DÖNGÜLERİ

Bazı işlemlerin yapılması esnasında belirli bir zaman hiçbir şey yapılmadan beklenmesi gerekebilir. Böyle bir beklemenin gerekliliği 7. bölümde görülmüştü. Bir butonun basıldığında/bırakıldığında kontakların birbirine teması esnasında ark meydana geliyordu. Bu arkı önlemek için zaman geciktirme döngüsü gerekiyordu.

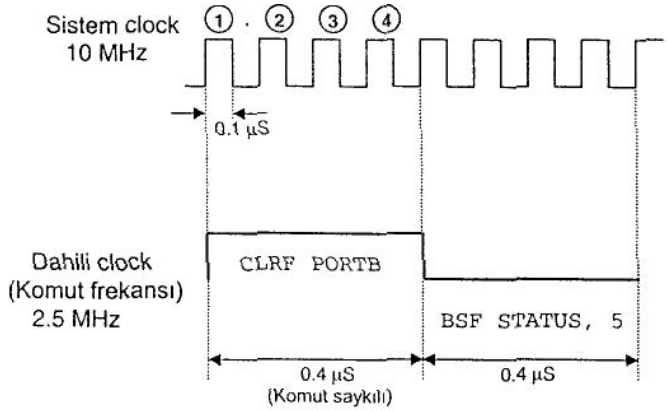
Zaman geciktirme işlemlerini yazılım döngüleri kullanarak yapabildiğimiz gibi donanımın (P1C16F84) bize sunduğu özel geciktirme olanaklarını kullanarak da yapabiliriz. Donanım geciktirmelerinin (TMRO ve WDT) nasıl kullanıldığını daha sonraki bölümlerde ele

alınacaktır.Bir zaman geciktirme döngüsünde, gecikme zamanını tespit etmek içir komutların icra süresi göz önüne alınır. RC osilatör kullanılan PIC devrelerinde bir komutun icra süresini hassas olarak saptamak mümkün değildir. Ancak kristal veya seramik rezonatör kullanılan devrelerde çok hassas gecikme döngüleri elde etmek mümkündür.

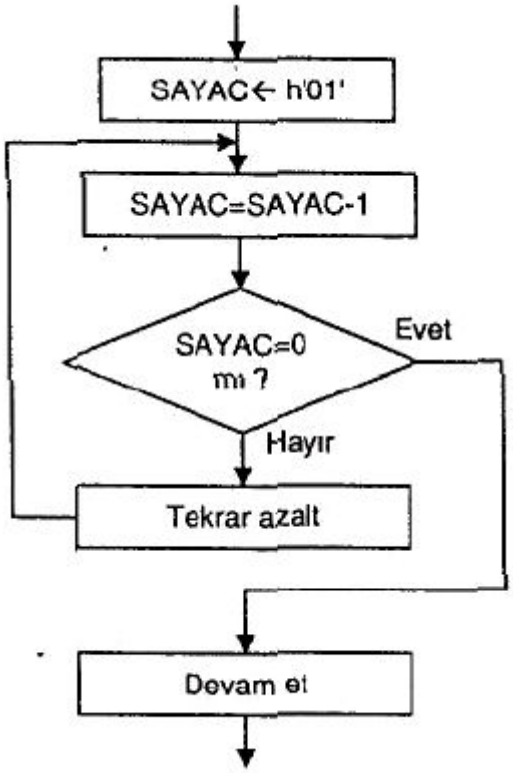
Dahili Komut Saykılı

P1C16F84 mikrodenetleyicisinin girişine uygulanan osilatör frekansı 4e bölünür. Bir komutun icra süresi bu frekansın bir saykılı süresindedir.
P1C16F84'ün osilatör girişine 4 MHz'lik bir kristal osilatör frekans uygulanırsa, P1C16F84 bu frekansı kendi içerisinde 4e böldüğünde, 1 MHz'lik bir dahili frekans elde edilir. Bu frekansın bir saykılı (T=1/f=1/1 MHz=1 mikroS)bir komutun icra süresidir.

10 MHz'lik frekansın bir saykılı T=1/f =1/106 Sn=0.1 mikroS eder. Dahili komut saykılı ise 4x0.1 mikroS eder.



	Komut	
	MOVLW	H'FF'
	MOVWF	SAYAC
DONGU		
	DECFSZ	SAYAC, F
	GOTO	DONGU



P1C16F84 mikrodenetleyicisinin komutlarından 10 tanesi hariç diğerleri 1 komut saykılı süresinde çalışır.

Tek Döngü ile Minimum Zaman Geciktirme

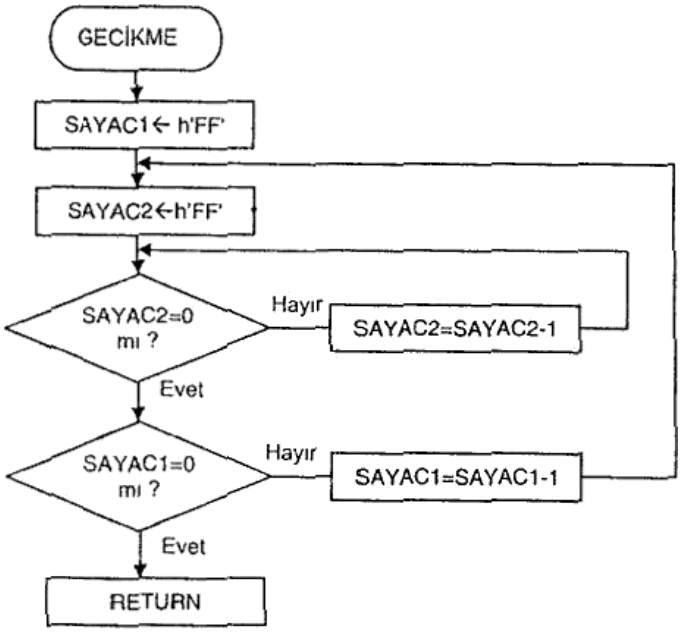
Bir gecikme döngüsü hazırlanırken sayaç olarak kullanılacak bir file register tanımlanır. Bu register içerisine döngünün tekrar sayısı yüklendikten sonra DECFSZ komutu ile tekrar sayısından her defasında 1 çıkartılır. Çıkarma sonucu 0 olunca döngü sona erdirilir.

Tek Döngü ile Maksimum Zaman Geciktirme

SAYAC registeri içerisine yüklediğimiz sayıyı h'FF' yaparsak yukarıda akış diyagramını çizdiğimiz gecikme döngüsünden maksimum gecikmeyi sağlarız. SAYAC registeri içerisine yüklediğimiz sayı N=h'FF'= d'255' dir.

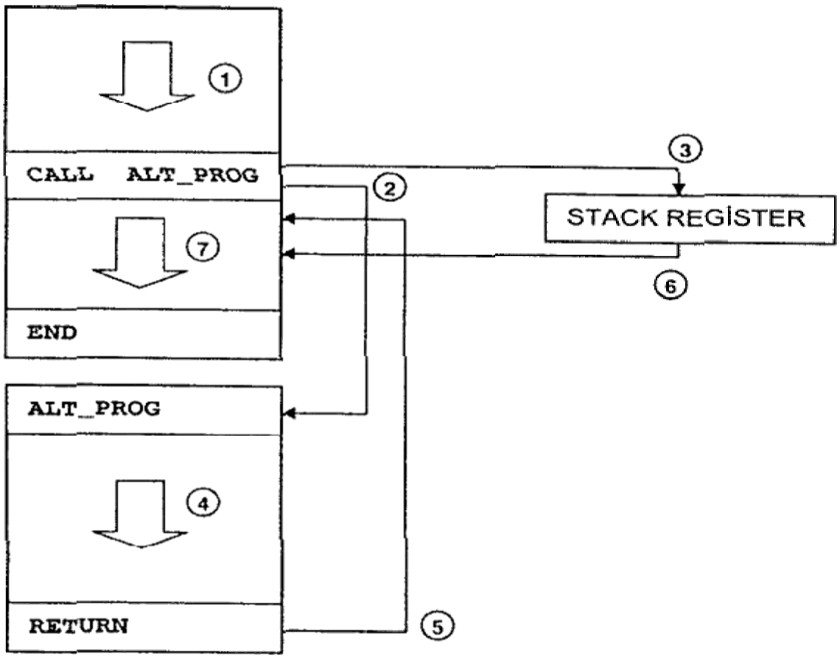
Çift Döngülü Zaman Geciktirme

Maksimum gecikme yapan tek döngüde oluşturulan 766 saykıllık süre az gelebilir. Bu durumda iç içe iki veya daha fazla döngü kullanılabilir. Şimdi bu kitaptaki çoğu zaman geciktirme işlemlerinde kullanacağımız çift döngünün akış diyagramını ve programı yapalım.



ALT PROGRAMLAR

Program içerisinde defalarca tekrar edilmesi gereken program parçaları olabilir. Bu durumda aynı program komutlarını her defasında yazmak hem zor olacak, hem de bellekte çok fazla yer kaplayacaktır. Bu sorunu gidermek amacıyla: alt programlar bir defa yazılır, kullanılması gerektiğinde alt programın adı CALL komutundan sonra yazılarak çağrılır. Bu andan itibaren programın akışı alt programa geçer. Alt program komutları bittiğinde en sonuna yazılan RETURN komutu vasıtasıyla ana programa dönülür ve kaldığı yerden devam eder.
Bir alt programın çalışmasını şematik olarak aşağıdaki gibi gösterebiliriz.



CALL ALT_PROG komutuyla alt program çağrılarak çalıştırılmaya başlandığı anda, programın tekrar geri döneceği adres bilinmesi gerekir. İste STACK REGİSTER bu adresin yazıldığı özel bir registerdir. Programcının bu registere adresin nasıl yazıldığı ile ilgilenmesi gerekmez. Çünkü assembler programı bu işlemi otomatik olarak yapar. Eğer çağrılan alt program içerisinden başka bir alt program daha çağrıldıysa bu adresler stack register içerisinde ardı ardına yazılır. Bu nedenle registere stack (yığın) adı verilmiştir. Alt programlardan RETURN komutuyla geri dönülürken en son yazılan adres ilk önce işlem görür. RETURN komutuyla geri dönülürken en son yazılan adres ilk önce işlem görür. Şimdi yukarıda verdiğimiz örnekteki işlem sırasını açılayalım: Ana programın ilk komutundan itibaren program çalışmaya başlar.

PROGRAM-6) Zaman gecikme döngüsü kullanarak PortBye bağlı ola tüm LEDleri belirli zaman aralıklarıyla yakıp-söndüren program.

```
;===PROG6.ASM======
                LIST        P=16F84
                INCLUDE     "P16F84.INC"
SAYACl          EQU         h'0C'
SAYAC2          EQU         h'OD'
                CLRF        PORTB
                BSF         STATUS, 5
                CLRF        TRISB
                BCF         STATUS, 5
TEKRAR
                MOVLW        h'00'
                MOVWF       PORTB
CALL            GECIKME
                MOVLW       h'FF'
                MOVWF       PORTB
CALL            GECIKME
                GOTO        TEKRAR
GECIKME                     ;Alt program başlangıcı
                MOVLW       h'FF'
                MOVWF       SAYACl
DONGUl
                MOVLW       h'FF'
                MOVWF       SAYAC2
DONGU2
                DECFSZ      SAYAC2, F
                GOTO        DONGU2
                DECFSZ      SAYACl, F
                GOTO        DONGUl
                RETURN
END
```

PIC 16F84
MİKRODENETLEYİCİSİ İLE
TRAFİK LAMBASI
UYGULAMASI

ALANI : ELEKTRİK ELEKTRONİK TEKNOLOJİSİ
DALI : GÜVENLİK SİSTEMLERİ

ADI SOYADI :

SINIFI / NO :

DEĞERLENDİRME

İŞLEM BASAMAKLARI	MONTAJ	İŞ ALIŞKANLIKLARI	SÜRE

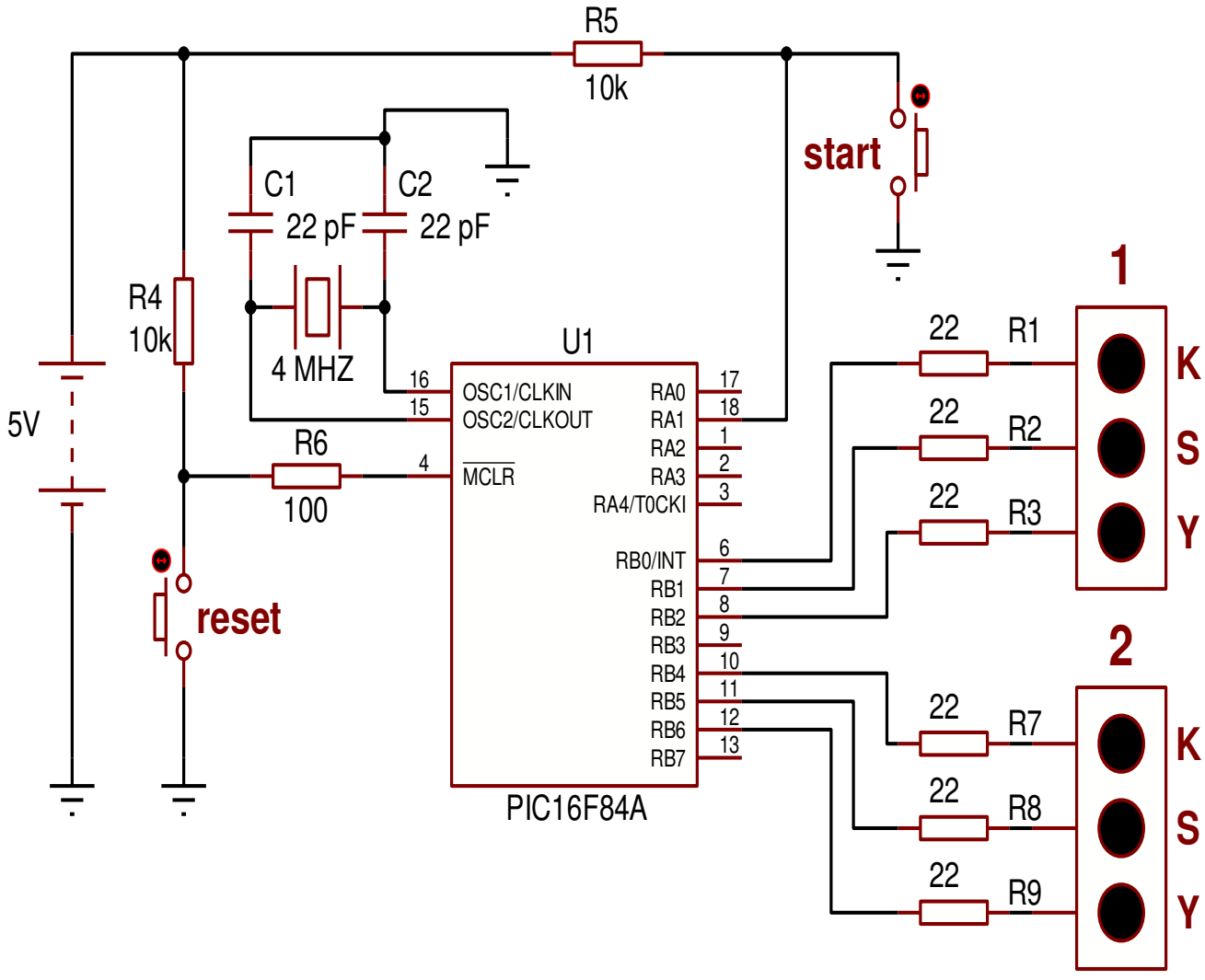
PIC16F84 İLE TRAFİK LAMBASI UYGULAMASI

PIC 16F84 ile yapılan trafik lambası uygulama devresinde kırmızı, yeşil ve sarı olmak üzere 2 ayrı kavşak olarak toplam 6 adet LED bulunmaktadır. Kırmızı ve yeşil ledler 15sn., sarı ledler ise 1sn yanık kalacaktır. Devre ilk çalıştırıldığında tüm ledler sönük durumda olacaktır. Sistem START butonu ile harekete geçirilecek ve devrenin çalışması istenilen anda RESET butonu ile kesilecektir. Ledler kırmızı, sarı, yeşil sırasına göre yanacak ve tekrar başa dönecektir.Kavşakta bir yöne geçiş varken diğer yöne geçiş olmayacaktır.

PIC'in A portu giriş, B portu çıkış olarak kullanılmaktadır. Portlardan hangisinin giriş ve çıkış olacağı kullanıcıya bağlıdır.Bu devreyi dijital entegreleri kullanarak da basit bir şekilde yapabiliriz. Fakat ledlerin yanık kalma sürelerini ayarlamak işin karmaşık kısmı oluşturmaktadır. Bu durumda devreye PIC girer. PIC'e yazdığımız programdaki küçük değişikliklerle ledlerin yanık kalma sürelerini artırıp azaltabiliriz. Bu süreleri değiştirmek için gecikme döngüleri yazılır.Tek döngü ile yapılan programın süresi çok kısa olduğundan en az iki döngülü programlar kullanılır.

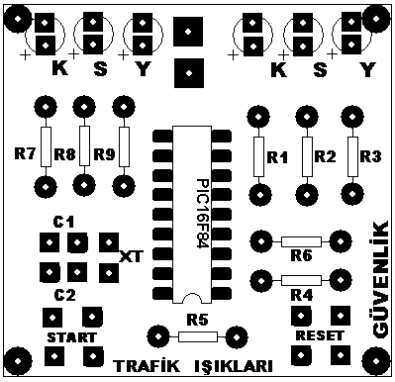
Devrede PIC'in besleme bacakları gösterilmemiştir. 14 nolu bacak pozitif kaynak (Vcc ya da Vdd), 5 nolu bacak ise toprak (GND ya da Vss)'tır.

DEVRE AÇIK ŞEMASI



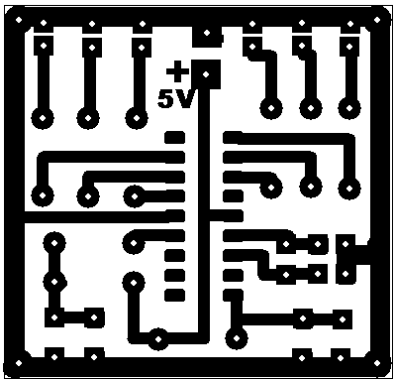
Malzeme katı

Üstten Görünüşü

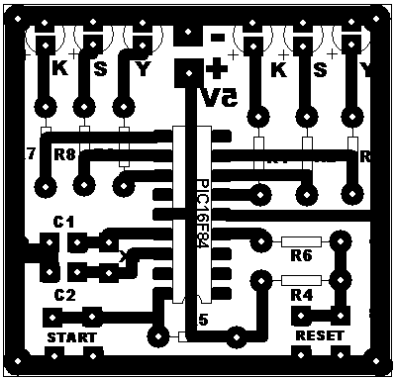


Bakırlı yol görünüşü

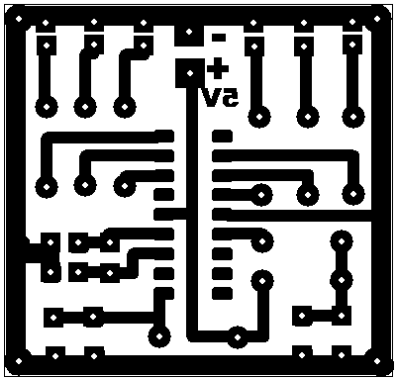
Alttan Görünüş



Üstten Görünüş tüm katlar



Ütülenecek Görüntü



Devrenin Malzemeleri

PIC16F84	18 pin entegre soketi	4 MHz kristal
2 adet kırmızı led	2 adet sarı led	2 adet yeşil led
C1 = C2 = 22pf	R1,R2 ,R3,R7,R8,R9 = 22 Ω	2 adet push buton
R4 = R5 = 10 KΩ	R6 = 100Ω	5cm x 5 cm bakırlı pertinaks

NOT: Her öğrenci bu uygulamayı yaparken yanında iş önlüğü, el aletleri, havya, baskı devre kalemi lehim, ölçü aleti, ince zımpara, 1mm matkap ucu bulundurması gerekir.

```
;========TRAFİK LAMBASI UYGULAMA PROGRAMI======S_2005=======
LIST    P=16F84
INCLUDE "P16F84.INC"
SAYAC1          EQU H'0C'       ; Gecikme alt programlarında kullanılan değişken
SAYAC2          EQU H'0D'       ; Gecikme alt programlarında kullanılan değişken
SAYAC3          EQU H'0E'       ; Gecikme alt programlarında kullanılan değişken
; Portları Ayarla……
CLRF PORTB              ; PORTB yi temizle
BSF STATUS,5                    ; BANK1 e geç
CLRF TRISB              ; PORTB çıkış
MOVLW H'FF'             ; W <-- H'FF'
MOVWF TRISA                     ; PORTA giriş
BCF STATUS,5                    ; BANK0 a geç
;Start butonuna basılıncaya kadar bekle……
BUTON
BTFSC PORTA,1                   ; PORTA'nın 1. biti 0 mı?
GOTO BUTON              ; Hayır, tekrar test et
TRAFIK
MOVLW H'41'             ; W <-- B'001000001'
MOVWF PORTB                     ; 1.kavşak için Kırmızı ledi yak.2.kavşak için yeşil ledi yak
CALL GECIKME1                   ;1 5sn. bekle
MOVLW H'23'             ; W <-- B'00100011'
MOVWF PORTB                     ; 1.kavşak için Kırmızı ve sarı ledi yak.2.kavşak için sarı ledi yak
CALL GECIKME2                   ; 1sn. bekle
MOVLW H'14'             ; W <-- B'00010100'
MOVWF PORTB                     ; 1.kavşak için yeşil ledi yak.2.kavşak için kırmızı ledi yak
CALL GECIKME1                   ; 15n. bekle
MOVLW H'32'             ; W <-- B'00110010'
MOVWF PORTB                     ; 1.kavşak için sarı ledi yak.2.kavşak için kırmızı ve sarı ledi yak
CALL GECIKME2                   ; 1sn. bekle
GOTO TRAFIK             ; TRAFIK etiketine git.
;=============GECİKME ALT PROGRAMLARI=====================
GECIKME1                ; 15  sn'lik gecikme alt programı
MOVLW d'60'             ; W<--D'60'
MOVWF SAYAC1                    ; SAYAC1 <-- W
G1
MOVLW d'255'            ; W<--D'255'
MOVWF SAYAC2                    ; SAYAC2 <-- W
G2
MOVLW d'255'            ; W<--D'255'
MOVWF SAYAC3                    ; SAYAC3 <-- W
G3
DECFSZ SAYAC3,F         ; Sayac3 bir azalt ve sıfır mı? kontrol et
GOTO G3                 ; Hayır G3'e git
DECFSZ SAYAC2,F         ; Evet. Sayac2 bir azalt ve sıfır mı?
GOTO G2                 ; Hayır G2'ye git
DECFSZ SAYAC1,F         ; Evet. Sayac1 bir azalt ve sıfır mı?
GOTO G1                 ; Hayır G1'e git
RETURN
GECIKME2                ; 1sn'lik gecikme alt programı
MOVLW d'5'              ; W<--D'5'
MOVWF SAYAC1                    ; SAYAC1 <-- W
D1
MOVLW d'255'            ; W<--D'255'
MOVWF SAYAC2                    ; SAYAC1 <-- W
D2
MOVLW d'255'            ; W<--D'255'
MOVWF SAYAC3                    ; SAYAC1 <-- W
D3
DECFSZ SAYAC3,F         ; Sayac3 bir azalt ve sıfır mı? bak
GOTO D3                 ; Hayır D3'e git
DECFSZ SAYAC2,F         ; Sayac2 bir azalt ve sıfır mı?
GOTO D2                 ; Hayır D2'ye git
DECFSZ SAYAC1,F         ; Sayac1 bir azalt ve sıfır mı?
GOTO D1                 ; Hayır D1'e git
RETURN
```

BİT KAYDIRMA KOMUTLARI

SOLA KAYDIRMA (RLF)
SAĞA KAYDIRMA (RRF)

SOLA KAYDIRMA (RLF)

RLF komutu, belirlenen bir file register içerisindeki bit'lerin pozisyonunu her defasında bir sola kaydırmak için kullanılır. Register içerisindeki bit'ler sola kaydığında MSB biti, STATUS registerde bulunan carry flag içerisine yazılır. Carry flag içeriği ise registerin LSB bit'ine yazılır.

NOT: MSB (Most Significiant Bit- Değerliği en yüksek bit, yani en soldaki bit.)
LSB (Least Significiant Bit- Değerliği en düşük bit, yani en sağdaki bit.)

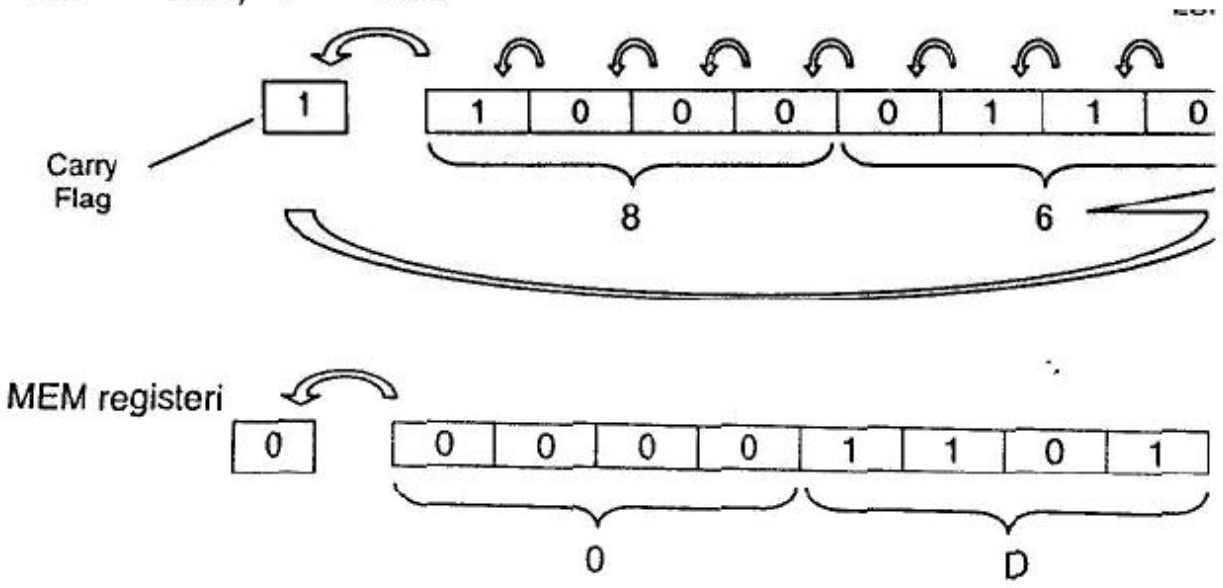
RLF komutunun yazılışı aşağıdaki gibidir.

RLF File Register Destination (Gideceği yer) W veya F

Destination W ise kaydırma sonucunda elde edilen bit paterni W registere, F ise file registere yazılır. örneğin MEM adındaki bir file register içeriği h'86" ise RLF komutu çalıştırıldığında MEM registerinin içeriğinin ne olduğunu şematik olarak gösterelim.

H86= b'10000110'

```
MOVLW  h'86'
MOVWF  MEM
RLF    MEM, F     MSB
```



SAĞA KAYDIRMA

RRF komutu, belirlenen bir file register içerisindeki bit'lerin pozisyonunu her defasında bir sağa kaydırmak için kullanılır. Register içerisindeki bit'ler sağa kaydığında LSB bit'i, STATUS registerde bulunan carry flag içerisine yazılır. Carry flag içeriği ise registerin MSB bit'ine yazılır.

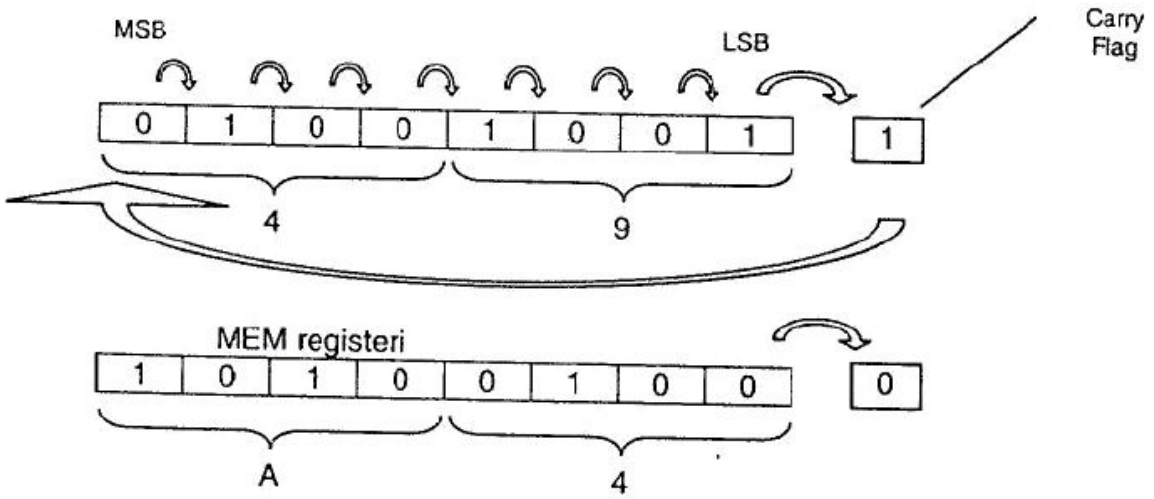
RRF komutunun yazılışı aşağıdaki gibidir.

RRF | File Register | d | Destination (Gideceği yer) W veya F

Destination W ise kaydırma sonucunda elde edilen bit paterni W registere, F ise file registere yazılır. Örneğin MEM adındaki bir file register içeriği H'49' ise RLF komutu çalıştırıldığında MEM registerinin içeriğinin ne olduğunu şematik olarak gösterelim.

H'49'= b'01001001' dir.

```
MOVLW   h'49'
MOVWF   MEM
RKF     MEM, F
```

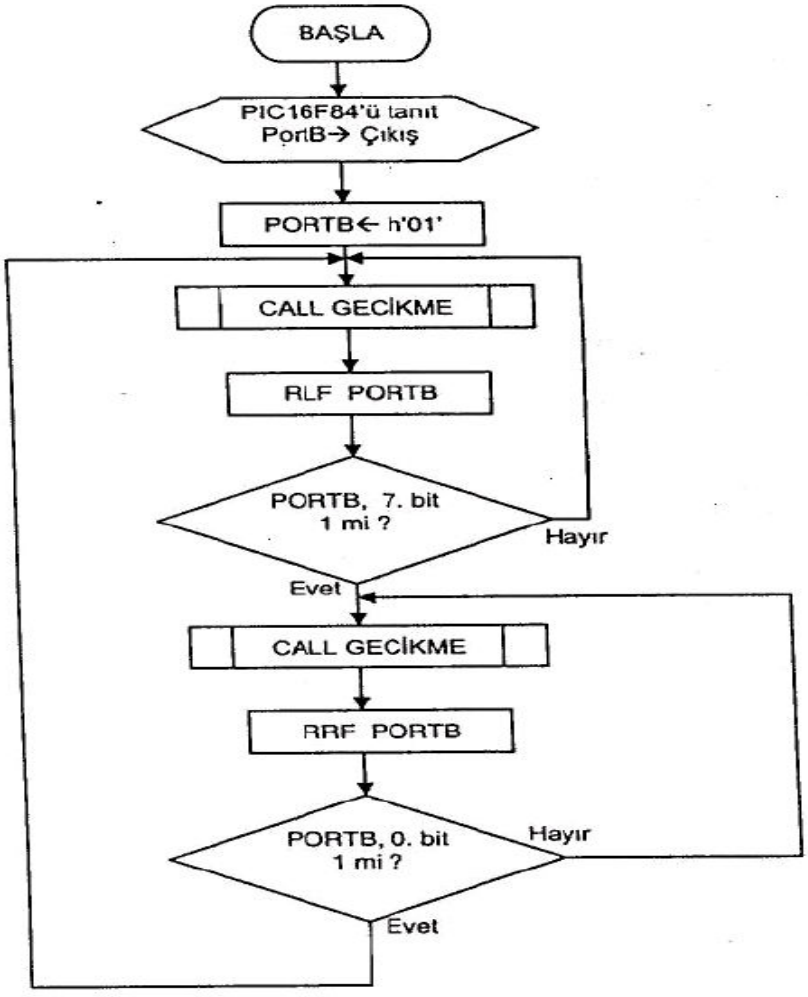


PROGRAM-13) PortB'ye bağlı olan 8 LED üzerinde bir LED'in yanışını sağa-sola kaydıran ve bu işlemi devamlı olarak yaptıran program. (Bu programın adını çok eskiden ünlü bir TV dizisindeki otomobilden alarak "Karaşimşek" diyoruz. Bu otomobilin de önündeki LED'Ierin yanışı sağa-sola kayarak yanıyordu, .);:Bu nedenle programa kısaca karaşimşek programı diyebilirsiniz.

```
;===PROG13.ASM====
                LIST        P=16F84
                INCLUDE     "P16F84.INC"
SAYAC1          EQU         h'0C'
SAYAC2          EQU         h'0D'
                CLRF        PORTB
                BCF         STATUS, 0     ;Carry flag'ı sıfırla
                BSF         STATUS, 5
                CLRF        TRISB
                BCF         STATUS, 5
                MOVLW       h'0l'         ;b'0000000l' sayısını W'ye yükle
                MOVWF       PORTB         ;W registerini PortB'ye yükle
SOL             CALL        GECIKME       ;Gecikme yap
```

```
            RLF         PORTB, F    ;PortB'deki veriyi sola kaydır.
            BTFSS       PORTB, 7    ;PortB 7. Bit 1 mi?
            GOTO        SOL         ;Hayır, sola kaydır.

SAG         CALL GECIKME            ;Gecikme yap
            RRF          PORTB, F   ;PortB'deki veriyi sağa kaydır.
            BTFSS       PORTB, 0    ;PortB 0. Bit 1 mi?
            GOTO        SAG         ;Hayır, sağa kaydır
            GOTO        SOL         ;Evet, sola kaydır.
            GECIKME                 ;Gecikme alt programı
            MOVLW        h'FF'
            MOVWF       SAYACl
DONGUl
            MOVLW        h'FF'
            MOVWF       SAYAC2
DONGU2
            DECFSZ      SAYAC2, F
            GOTO        DONGU2
            DECFSZ      SAYACl, F
            GOTO         DONGUl
            RETURN
            END
```



PIC 16F84
MİKRODENETLEYİCİSİ İLE
LED ŞOVU
UYGULAMASI

ALANI : ELEKTRİK ELEKTRONİK TEKNOLOJİSİ
DALI : GÜVENLİK SİSTEMLERİ

ADI SOYADI :

SINIFI / NO :

DEĞERLENDİRME

İŞLEM BASAMAKLARI	MONTAJ	İŞ ALIŞKANLIKLARI	SÜRE

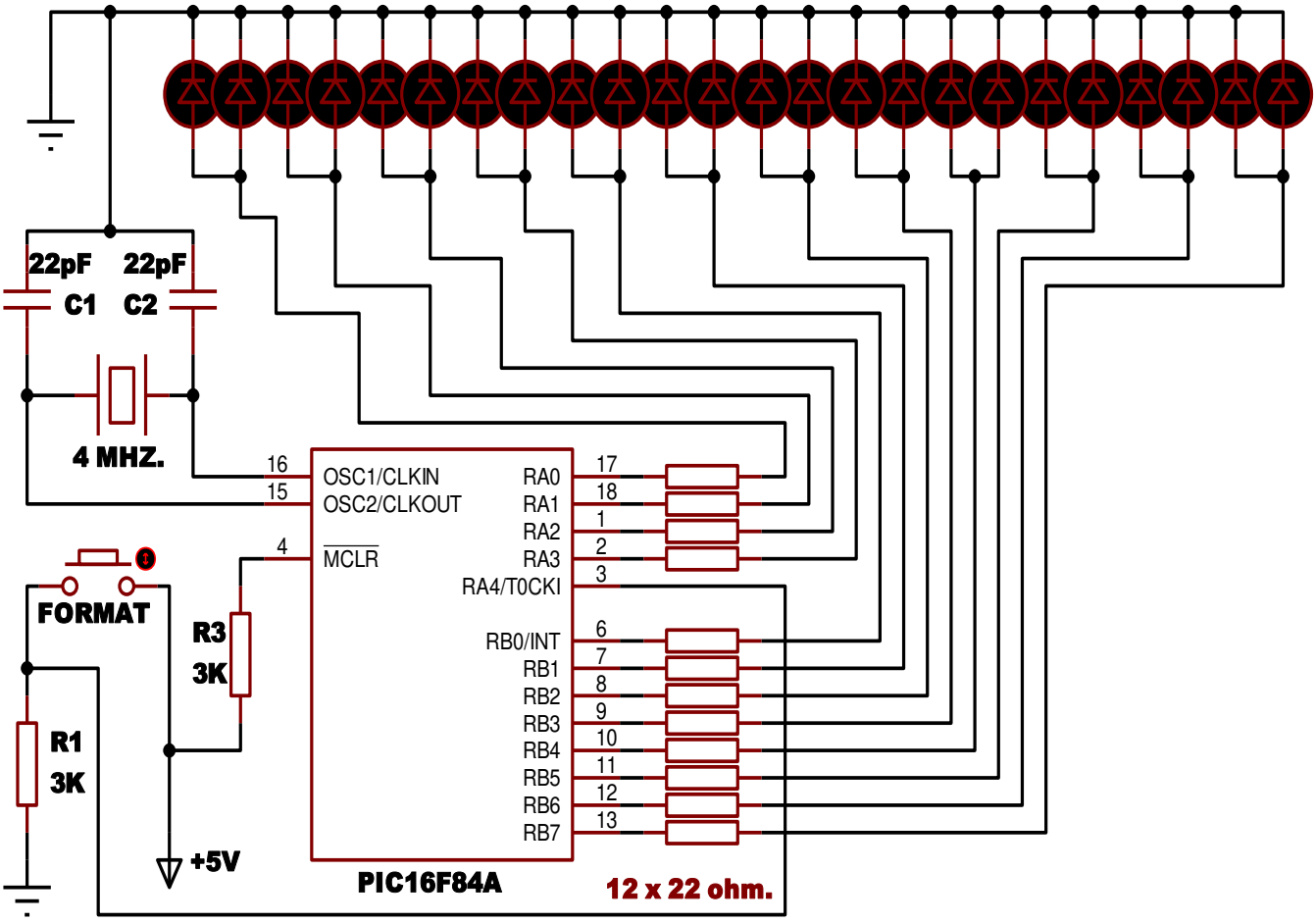
PIC16F84 İLE LED ŞOVU UYGULAMASI

PIC 16F84 ile yapılan LED ŞOVU uygulama devresinde 2'şerli toplam 24 adet LED bulunmaktadır. Devreye ilk enerji uygulandığında onikili olarak ard arda polis ışığı şeklinde ledler yanma gösterecektir. Format butonuna her basıldığında ledlerin yanma şekli değişecektir. Toplam 24 farklı led yanış şekli vardır. En son formatta tüm yanışlar sırasıyla tekrarlanır. Enerjinin kesilip tekrar verilmesiyle çalışma şekli en başa dönecektir.

PIC'in bir portu giriş, oniki portu çıkış olarak kullanılmaktadır. Ledlerin yanık kalma sürelerini ve yanış formatını düzenlemek için ASM dosyasında düzenleme yaparak tekrar hex kodlu dosya yaratmak gerekir.

Devre 5 Volt ile çalışmaktadır. Gerekirse 7805 regüle entegresi kullanılarak farklı gerilim değerlerinde de kullanılabilir. Devrede PIC'in besleme bacakları gösterilmemiştir. 14 nolu bacak pozitif kaynak (Vcc ya da Vdd), 5 nolu bacak ise toprak (GND ya da Vss)'tır.

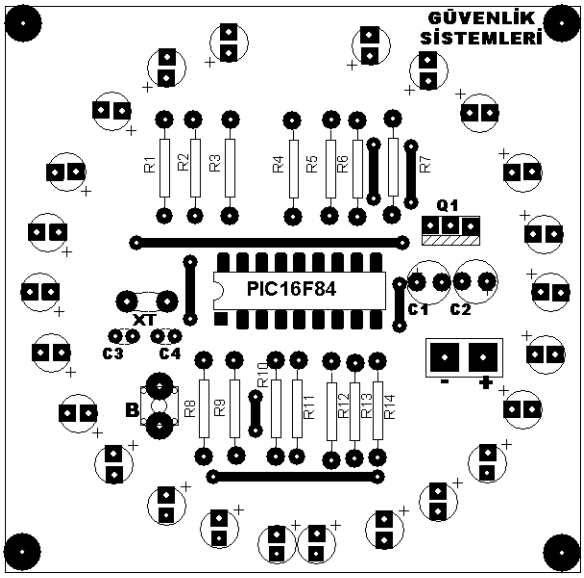
DEVRE AÇIK ŞEMASI



NOT: Her öğrenci bu uygulamayı yaparken yanında iş önlüğü, el aletleri, havya, baskı devre kalemi lehim, ölçü aleti, ince zımpara, 1mm matkap ucu bulundurması gerekir.

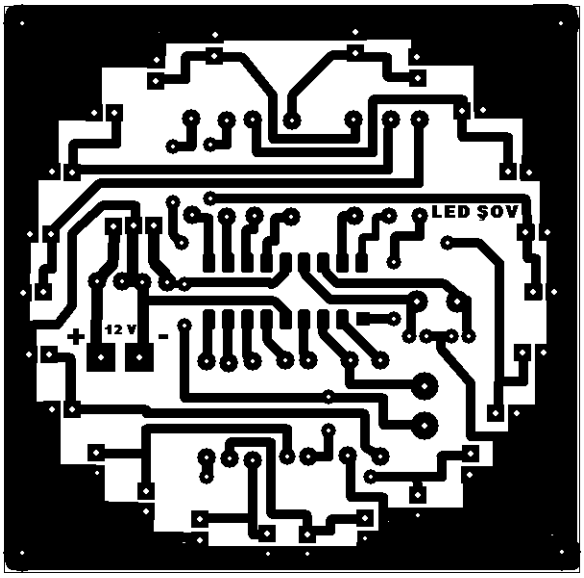
Malzeme katı

Üstten Görünüşü

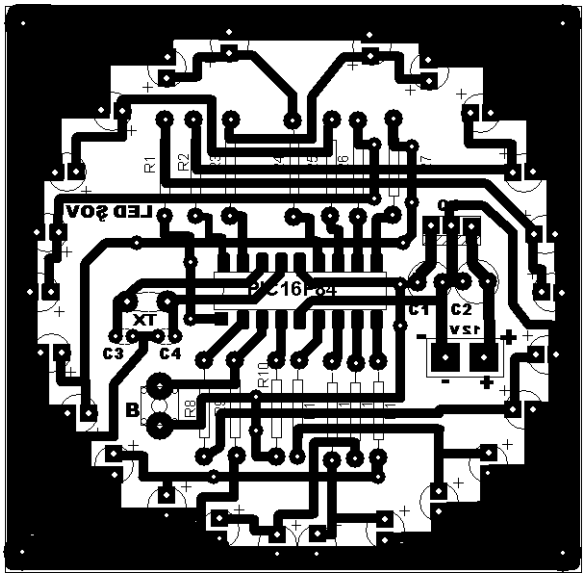


Bakırlı yol görünüşü

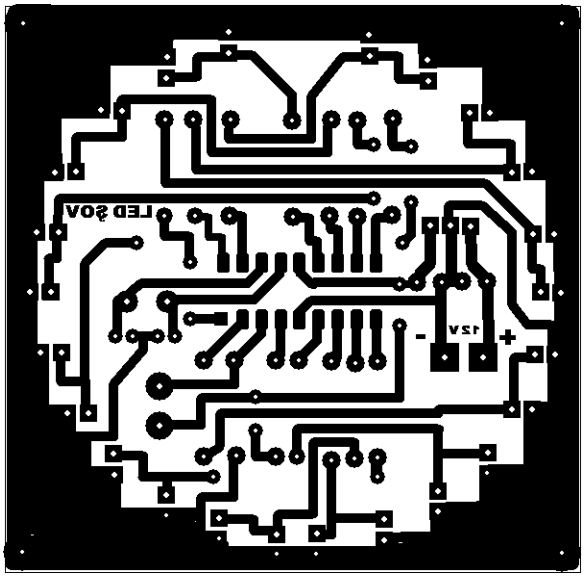
Alttan Görünüş



Üstten Görünüş tüm katlar



Ütülenecek Görüntü



Devrenin Malzemeleri

PIC16F84	18 pin entegre soketi	4 MHz kristal
C1,C2 = 10 µf 25 V	C3,C4 = 22 pf	24 Adet Led
R9 ,R10 =3 KΩ	R1, R2,R3,R4,R5,R6,R7,R8,R11,R12,R13,R14 = 22 Ω	
1 adet push buton	7,5 cm x 7,5 cm bakırlı pertinaks	