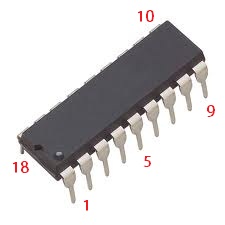
**Microdenetleyiciler Dersi Ders Notu**

**Mikrodenetleyici**

Günlük hayatta bir çok amaca yönelik entegre üretilmiştir. Bu entegreler belli amaçlar doğrultusunda kullanılabilmektedir. Entegreler gerekli çalışma şartları yerine getirildiğinde istenilen işleri yapan küçültülmüş elektronik devrelerdir. Büyük devreler küçük entegrelerin içine sığdırılabilmektedir. Entegrelerin belli bacakları belli işlevler için kullanılır entegre bilgileri üretici firmalar tarafından entegre birlikte yayınlanan **datasheet(bilgi sayfaları)’**lerde kullanıcılara sunulur. Entegrelerin çalıştırılabilmesi için yerine getirilmesi gereken çalışma şartları bulunmaktadır. Hemen hemen her entegre bir besleme gerilimine ihtiyaç duyar, bunun dışında da entegrelerin kendilerine özgü çalışma şartları olabilir. Her entegrenin bir ismi olur, isim genellikle harfler ve rakamlardan oluşur(LM317T, LM7805, NE555,PIC16fF628A... gibi). Entegrelerin çalışma şartları ve hangi bacağın hangi işlem için kullanılacağı datasheetlerinden öğrenilebilir. Entegre datasheetleri Google’a “entegre ismi datasheet” yazılarak aranabilir. Örneğin 16f628a datasheetinin arama motoruyla aranması için “16f628a datasheet” yazılmalıdır.

Entegre bacakları belli kurallara göre sıralandırılır. Entegreler tasarlanırken 1 numaraları bacaklarının ne tarafta bulunduğu birinci ve sonunucu bacağın bulunduğu tarafta çentik bulunur. Bu çentik ilk ve son bacağın bulunduğu tarafı gösterir. Ayrıca 1 numaralı bacağın hemen yanına genellikle birde nokta bulunur. Bu nokta da bir numaralı bacağı gösterir. Bacakların numaraları tespit edilirken 1 numaralı uçtan başlanarak sırayla sayılır. Kenar sonuna gelindiğinde **U** harfi yapacak şekilde entegre bacakları numaralandırılmaya devam edilir.

Kısaca **Mikrodenetleyiciler programlanabilen entegrelere denir**. Klasik entegrelerle mikrodenetleyici entegrelerinin farkı, mikrodenetleyici entegrelerinin programlanabilmesidir. Çok karmaşık büyük devrelerin yapabileceği işlemleri çoğu zaman çok küçük mikrodenetleycili devreler yapabilir. Bir çok firmanın ürettiği programlanabilen entegreler bulunmaktadır. Biz MikroChip firması tarafından üretilen PIC entegrelerini programlama ve kullanmayı öğreneceğiz. PIC kelimesi Peripheral Interface Controller(Çevresel Arayüz Denetleyicisi) kelimelerinin baş harflerinin birleştirilmesi ile oluşturulmuştur.

PIC entegreleri besleme,reset ve osilatör gibi çalışma şartları sağlandığında içine yüklenen program doğrultusunda çalışan entegrelerdir. Modellerine göre giriş çıkış uçları bulunur bu uçlardan entegre içine bilgi girişi ve entegreden bilgi çıkışı yapılabilir. Temel Prensip olarak aynı şekilde çalıştırılabilen PIC modellerinden biz 16F628A entegresini kullanacağız.

Not: Ders notunun tamamında kullanılan mikrodenetleyiciden bahsedilirken mikrodenetleyici, programlanabilir entegre, entegre ya da PIC isimleri kullanılacaktır.

**Neden Mikrodenetleyiciler**

Mikrodenetleyicilerin en önemli avantajları;

1. Az yer kaplar.
2. Az enerji harcar.
3. Maliyeti düşüktür.

Mikrodenetleyicili devrelerin normal devrelere karşı üstünlükleri;

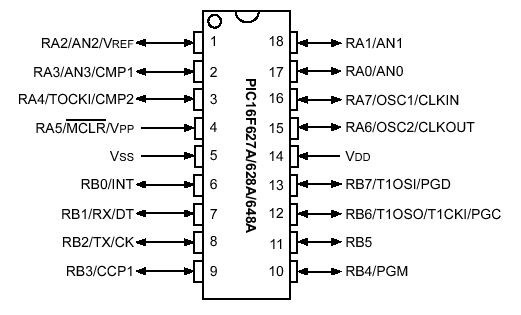
1. Devrede değişiklik yapılmadan tekrar tekrar programlanabilirdir.
2. Çok daha küçük devrelerle,j daha az devre elemanı kullanılarak çok karmaşık işlemler yapılabilir.
3. Mikrodenetleyicili devreleri yapmak daha az zaman ve para gerektirir.

**Günümüzde Mikrodenetleyicilerin kullanıldığı yerlerden bazıları**

1. Asansörler
2. Fotoğraf makinesi
3. Işıklı reklam tabelaları
4. Televizyon
5. Buzdolabı, Çamaşır makinesi, Bulaşık makinesi
6. Otomatik garaj kapıları
7. Sıramatik
8. Modem
9. Robot
10. Klima

**PIC 16F628A**

Bu pic entegresi 18 bacaklıdır. 18 bacaktan 2 tanesi besleme olarak kullanılırken, geriye kalan 16 uç giriş ve çıkış olarak kullanılabilen uçları oluşturur. PIC 16F628A entegresi bacak bağlantıları aşağıdaki şekilde belirtilmiştir.



* 16f628a entegresi 18 bacaklıdır.
* Entegre şekildeki gibi tutulduğunda ortadaki iki bacak besleme uçlarıdır(5,14).
* Beseleme uçlarının üstünde kalan uçlar RA uçları; altında kalan uçlar ise RB uçlarıdır.

Burada ortada bulunan VSS(-) ve VDD(+5V\*\*\*) bacakları **besleme gerilimi**nin uygulanacağı bacakları, diğer bacaklarda 8 bitlik A Portunu(RA0-RA7) ve 8 bitlik B portunu(RB0-RB7) oluşturmaktadır. Burada RA ve RB uçlarının yanında / ile gösterilen uçlar opsiyonel olarak kullanılabilecek uçlardır. İlk etapta besleme uçları(VDD,VSS), osilatör uçları(OSC1,OSC2), reset ucu(MCLR) A ve B portu dışındaki uçlar haricindeki özellikler kullanılmayacaktır. Bizim ilk olarak bilmemiz gereken VSS ve VDD uçlarının besleme uçları olduğu ve bu uçların altında kalan uçların B portu uçlarını, üstünde kalan uçların A portu uçları olduğudur.

**PIC16F628A Entegresinin Çalıştırılabilmesi Yapılması Gerekenler**

1. Besleme Gerilimi Sağlanması
2. Osilatör devresi bağlanması(\*)
3. Reset ucu(MCLR) bağlanması(\*\*)
4. Program yazılması ve yüklenmesi

\*\* Dahili osilatörü bulunan entegrelerde osilatör devresi zorunlu değildir, isteğe bağlı olarak kurulur.

\*Reset ucu kullanılmak istenmezse reset bağlantısı yapmak zorunlu değildir, programla iptal edilip devresi yapılmayabilir.

**Besleme Gerilimi**

PIC16F628A entegresi nin besleme gerilimi 5 Volttur\*\*\*. Bu gerilimin üstünde gerilim uygulamak entegreye verir. Kesinlikle çok kısa süreli de olsa besleme uçlarına 5 Voltun üzerinde gerilim **uygulanmamalıdır**.

\*\*\* Mikrodenetleyicinin besleme gerilimi kullanılan osilatör frekansı ve mikrodenetleyicinin modeline göre 2V ve 5V arasında olabilir. Biz genellikle çalışma gerilimini 5V olarak alacağız.

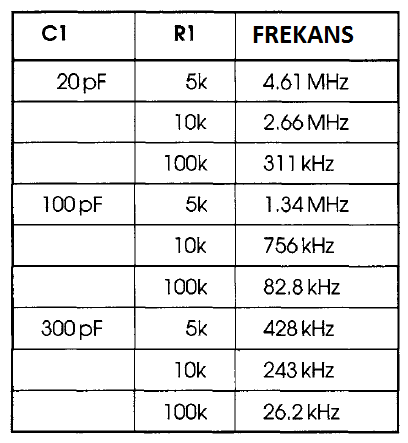
**Osilatör Devresi**

Osilatör PIC entegresinin içindeki program komutlarını işleme hızını belirleyen titreşimleri gönderen devreye denir. Osilatör mikrodenetleyicinin hızını ve hassasiyetini belirler. Dijital mantığına göre çalışan entegreler Osilatöre ihtiyaç duyar. Çok çeşitli osilatör devreleri bulunmaktadır. En basit osilatör devresi basit Flip- Flop devresidir. PIC’li devreler de genellikle Kristal osilatör kullanılır. Yaygın olarak iki çeşit kristal devresi kullanılır. Bunlar direnç ve kondansatör kullanılarak yapılan RC osilatör ve Kristal ve kondansatör kullanılarak yapılan Kristal osilatördür. Saat gibi yüksek hassasiyet gerektiren devrelerde kristal osilatör kullanılırken. Zaman hassasiyetinin çok önemli olmadığı yerlerde kristal osilatöre göre daha ucuz olan RC osilatör kullanılır.

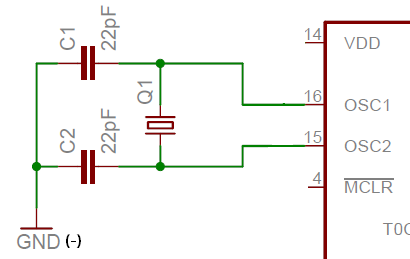
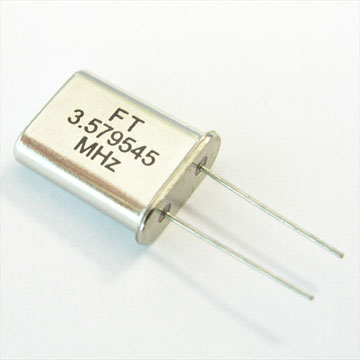
**Not:** Osilatör frekansı arttıkça pic entegresi tarafından harcanan enerjide artar. Bu yüzden çok hassas zamanlama ve hız gerektirmeyen işlemlerin yapılacağı devrelerde osilatör frekansının düşük tutulması daha az enerji tüketilmesini sağlar. Örneğin dijital saatlerde genellikle osilatör frekansı 32.768 kHz’dir.

**Not:** Bazı pic entegrelerinde dahili kristal osilatör bulunmaktadır. Böyle mikrodenetleyiciler osilatör devresi bağlanmadan da çalıştırılabilir. Örneğin 16f628a entegresinde dahili 4MHz osilatör devresi vardır. İstenirse 16f628a entegreli devreler osilatördevresi bağlanmadan çalıştırılabilir. Dahili osilatörün kullanılabilmesi için Project menüsünden Edit Project sekmesinden osilatör türü(osilatör selection) “intosc” şeklinde seçilmelidir.

**RC Osilatör**

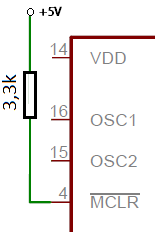
Bir direnç ve bir kondansatörden oluşur. Devrenin bağlantı şekli aşağıdaki gibidir. Osilatörün frekansı direnç ve kondansatörün değerlerine bağlıdır. Yandaki tabloda R1 ve C1 değerlerine karşılık elde edilen frekans değerleri belirtilmiştir. Örneğin 4.6 MHz ‘lik bir frekans isteniyorsa C1 kondansatörü yerine 20pF, R1 direnci yerine 5k’lık direnç bağlanmalıdır. RC osilatör zamanlama olarak hassasiyet gerektirmeyen devrelerde kullanılır.

**Kristal Osilatör**

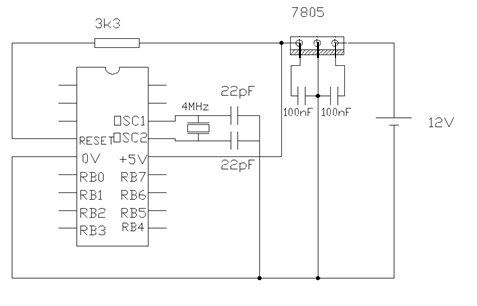
****Kristal osilatör 2 adet kutupsuz kondansatör ve bir adet kristal kullanılarak yapılır. Osilatörün iki bacağı 16F628A entegresinin 15 ve 16 nolu bacaklarına bağlanır. Burada Kristal frekansı devrenin Osilatör frekansını belirler. Yukarıda kristalin resmi gözükmektedir. Kristalin çalışma frekansı üzerinde yazar. Örneğin yukarıdaki kristalin frekansı 3,5 MHz’dir.

Not: Kristal osilatörlerin frekansları düştükçe parazitlerden etkilenme oranları artar. Parazitli ortamlarda çalışacak devrelerin yüksek frekanslı osilatörlerin kullanması faydalıdır.

**Reset(MCLR) Ucunun Bağlanması**

Reset ucu program çalıştırıldıktan sonra programın ilk satıra döndürülmesi için kullanılan uçtur. Reset ucuna 5V uygulandığında program normal olarak çalışır. Bu gerilim sıfır Volta ayarlandığında reset işlemi gerçekleşmiş olur. Yani Programın normal olarak çalışmasını sürdürmek için bu uca 5v’luk gerilim uygulamak gerekmektedir. Reset ucu MikroC programından aktif edildiği halde bu gerilim bağlanmazsa devre kararsız çalışır. Reset ucu 16f628A entegresinde 4 nolu bacaktadır.

\*\*\* Reset butonu kullanılmayacaksa reset özelliği MikroC programından Project🡪Edit Project🡪 RA5/MCLR/Vpp Pin Function = Disabled şeklinde değiştirilerek yandaki bağlantıya da gerek kalmadan iptal edilebilir. Yani bu ayar yapılırsa devrenin MCLR ucuna gerilim uygulama gerekliliği ortadan kalkar.



Şekil: Temel Çalışma şartları Sağlanmış PIC Devresi

**PIC Mikrodenetleyicisi İçin Program Yazılması**

**MikroC Program Geliştirme Aracı**

Mikrodenetleyicilerle iletişim kurabilmek için kullanılan ortak dile program denir. Programlar komutlardan oluşur. Her firmanın mikrodenetleyicileri için ayrı ayrı komutlar bulunur. Pic mikrodenetleyicisinin 30’u aşkın komutu bulunmaktadır. Bu komutlar ile Mikrodenetleyici ile iletişim kurabiliriz. Yaptırmayı istediğimiz işleri mikrodenetleyicinin komutları ile ona anlatabiliriz. Pic Mikrodenetleyicileri Assembler komutları denen komutları kullanır. Ancak bu komuları kullanarak program yazmak yeni bir kullanıcı için oldukça zordur. Bunun yerine daha az komut kullanarak daha karmaşık işlemlerin daha kolay bir şekilde yapıldığı Pascal gibi C dili gibi, BASIC dili gibi diller mevcuttur. Biz bu dillerden C dili altyapısına uygun olarak hazırlanmış MikroC programlama dilini kullanacağız. C programlama dili belli standartlarla belli işleri yaptırmak için kullanılır komut sayısı asembler diline göre çok daha azdır. MikroC dilinde Program yazmak daha kolay ve çabuk öğrenilir. MikroC programının en önemli özelliklerinden bir tanesi, kullanımın, kurulumunun çok basit ve kolay olmasıdır. Programı <http://meykitap.com/ders/download.aspx> adresinden indirip kurabilirsiniz.

Mikrodenetleyici için program oluşturma ve entegreye yükleme işlemi aşağıdaki basamaklar izlenerek yapılır.

1. Proje oluşturmak
2. Program yazmak
3. Programı derlemek
4. Derlenmiş programı(hex dosyasını) PIC mikrodenetleyicisine yüklemek

**MikroC’de Yeni Proje Oluşturmak**

1. MikroC programı açılır.
2. **Project** 🡪 New **Project** seçilir
3. Açılan pencerede **Next** butonuna basılır ve ilerlenir.
4. Açılan pencerede;
5. “**Project Name**” yazan kutuya yeni oluşturulan projenin ismi
6. **Project Folder** kutusuna projenin kaydedileceği yer(**Browse** yazan butonla masa üstü seçilirse projenin bulunması kolay olur)
7. **Device Name** yazan kutudan, kullanılacak Pic entegresi seçilir. Entegrede kullanılacak osilatör frekansı **Device Clock** yazan kutuya girilir(Çoğunlukla 4 MHz olur).

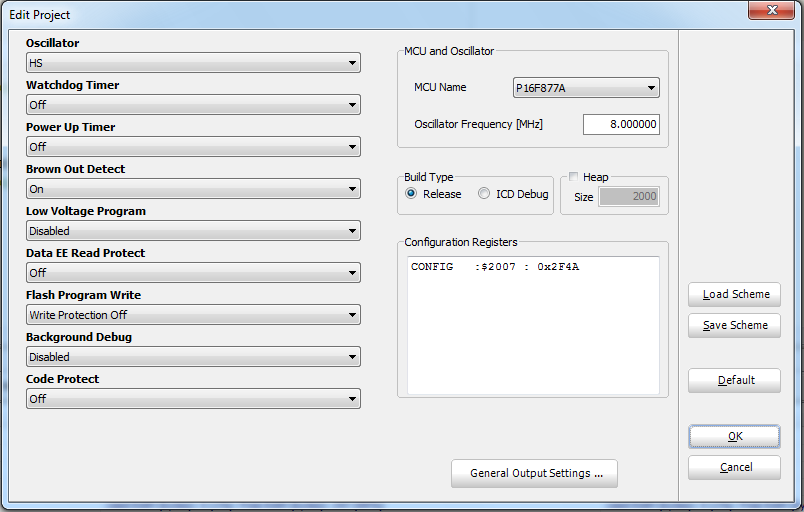
**Not:** Çok sık yapılan hatalardan bir tanesi b şıkkında belirtilen “Project Folder” yani projenin kaydedileceği yerin belirtilmemesidir. Bu seçim boş bırakıldığında yeni oluşturulan proje programın kurulum dizinine kaydedilmektedir. Bu da oluşturulan projenin çoğunlukla yeni kullanıcılar tarafından bulunamamasına neden olur. Project Folder kısmı **kesinlikle** boş geçilmemelidir.

**Not:** Device Name yazan kutuya programlanacak entegrenin hatalı girilmesi sıkça yapılan bir hatadır programın çalışmamasına ya da hatalı çalışmasına neden olur. Projelerimizde çoğunlukla 16F628A kullanacağımızı daha önce belirtmiştik. Yapacağımız hemen hemen bütün projelerde burdan 16F628A’yı seçmeliyiz.

1. Bilgiler girildikten sonra 3 defa **Next** butonuna ve en sonunda **Finish** butonuna basılır, proje oluşturma işlemi bitmiştir.

**MicroC’de daha önce oluşturulmuş projede değişiklikler yapmak(Edit Project Menüsü)**

Çalışılan projenin özelliklerinde değişiklik yapılmak istendiğinde Project 🡪Edit Project Penceresi kullanılır buradan projenin bir çok özelliği değiştirilebilir.

****

**Oscillator:** Kullanılan osilatörün türü ve frekansına göre ayarlanması gereken seçenektir. Buradaki seçenekler aşağıdaki gibidir.

1. RC: RC osilatör kullanıldığı zaman seçilmelidir.
2. LS: 32.768 kHz’lik kristal osilatör kullanılacağı zaman seçilmelidir.
3. XT: 8 MHz’ye kadar olan kristaller kullanılacağı zaman seçilmelidir.
4. HS: 8 MHz’den yüksek frekanslı kristaller kullanılacağı zaman seçilmelidir.
5. ***INTOSC:Osilatör devresi kurulmadan, mikrodenetleyici içindeki osilatör kullanılacağı zaman seçilmelidir***.

**Low Voltage Program:** Bu özellik RB4 bacağının normal olarak kullanılabilmesi için “Disabled” şeklinde ayarlanmalıdır.

**Code Protect:**  Bu seçenek entegreye yazılmış programın okunamamasını sağlar ve aynı zamanda da programlanmış entegrenin tekrar programlanmasını da engeller. Bu yüzden kesinlikle “On” durumuna getirilmemelidir. Aksi taktirde entegre tekrar programlanamaz duruma gelir.

**MCU Name:** Buradan kullanılacak mikrodenetleyici entegresi seçilmelidir.

**Oscillator Frequency:** Osilatör frekansı girilmelidir.

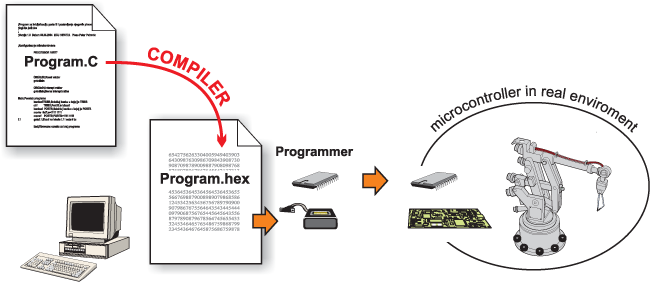
**Uzantı nedir**

Kaydedilen her dosyanın bir uzantısı bulunur. Uzantılar dosyanın içeriği hakkında bilgi sahibi olmamızı sağlar. Örneğin word’te yazılmış dosyaların uzantısı doc ya da docx, resim dosylarının uzantısı jpg olabilmektedir. Uzantı dosya isminde noktadan sonraki kısma denir. Örneğin **resim.jpg** dosyasının uzantısı **jpg**’dir. **Yazi.doc** dosyasının uzantısı da **doc**’tur.

Not: Windows işletim sisteminin bilinen dosya uzantılarını gizle şeklinde bir seçeneği vardır. Eğer bu seçenek seçiliyse bazı dosyaların uzantıları windows tarafından gizlenir. Uzantısı gizlenmiş bir dosyanın uzantısı dosya üzerine sağ tıklanıp, özellikler seçilerek, çıkan ekranda dosya türü kısmından öğrenilebilir.

**Mikrodenetleyicinin programlanması aşamaları**

1. MikroC’de proje oluşturulması
2. MikroC’de program yazılması (.c)
3. Programın derlenmesi(C uzantılı dosyadan hex uzantılı dosyanın oluşturulması)
4. Hex dosyasının PIC entegresine yüklenmesi



**MikroC’de Program yazmak**

Yeni proje oluşturulduğunda MikroC programı yeni bir sayfa açar bu sayfada

Void main()

{

}

şeklinde bir kaç satır kod yazılı halde çıkar. Yukardaki şekilde yazılmış fonksiyona en temel MikroC fonksiyonu denir. En temel mikroC fonksiyon her programda mutlaka bulunacağı için hazır olarak yazılmıştır. Program yazma işlemi ileriki derslerde öğreneceğimiz komutlar kullanılarak iki dalgalı”{}” parantez arasına yazılır. Bu yazıları yazdığımız dosyanın uzantısı .C uzantılıdır. Buraya yazdığımız komutlar mikroC komutlarıdır. Bu komutların entegreye yüklenebilecek duruma getirilmesi için derleme işlemine ihtiyaç duyulur. Derleme işlemi sonucunda .hex uzantılı dosya oluşturulur. Bu dosya entegrenin içine yüklenecek bilgilerin bulunduğu dosyadır.

**Yazılan Programın Derlenmesi(hex dosyası oluşturmak)**

MikroC’de yazılan kodların Pic entegresine doğrudan yüklenemez. Öncelikle bu kodların mikrodenetleyici diline dönüştürülmesi gereklidir. Yazılan kodların **hex**(örnek: Myproject.hex) uzantılı dosyaya dönüştürülmesine **DERLEME(Build)** işlemi denir. Derleme işlemi yapılmadan önce .hex uzantılı dosya oluşturulmaz. Derleme işlemi MikroC’de program yazıldıktan sonra **Build** menüsünden **Build** seçilerek gerçekleştirilir. C’de yazılan her değişiklikten sonra build işlemi yapılmalıdır ki .hex uzantılı dosya güncellensin. Mikrodenetleyici entegresine yüklenecek dosya derleme işlemi sonucunda oluşan hex uzantılı dosyadır.

**Hex Dosyasının PIC Entegresine Yüklenmesi**

Hex dosyasının PIC entegresinin belleğine aktarılması işlemidir. Bu işlemin yapılması için PIC entegresi ve bilgisayar arasında iletişim kurulması gerekmektedir. Bu iletişim programlama devreleri ile gerçekleştirilir. Programlama devreleri bilgisayar programları ile kullanılırlar. 10larca çeşit programlama devresi vardır ve her programlama devresinin bir de programı vardır. Programlama devrelerinin amacı sadece hex dosyalarının PIC entegresine yüklenmesidir. Hex dosyasının PIC entegresine yüklenmesi işlemi

1. Programlama devresine programlanacak entegre takılarak devrenin bilgisayara bağlanması
2. Programlama devresi programının bilgisayarda çalıştırılması
3. Derleme sonrası oluşan hex uzantılı dosyanın entegreye yükleme programı yardımıyla programlanacak entegreye yüklenmesi şeklinde gerçekleştirilir. PIC entegresine program yüklendikten sonra istenilen devreye bağlanarak çalıştırılır.



Programlama Devreleri

Bilgisayarda oluşturulan hex dosyalarının Pic mikrodenetleyicisine yüklenmesi için kullanılan devreler dir. 3 temel ögeden oluşurlar. Bunlar;

1. Programlama devresi: Programlanacak entegrenin takılacağı devredir.
2. Bağlantı kablosu: Programlama devresi ve bilgisayar arasında veri transferinin yapılacağı bağlantı kablosudur.
3. Bilgisayar programı: Bilgisayarda bulunan hex dosyasının programlama devresine aktarılmasını sağlayacak bilgisayar programıdır.

**Mikro Burn Programının Bilgisayara Kurulması**

Entegrelerimize yazdığımız programlara yükleyebilmek için Program yükleme kartımızı bilgisayarımıza tanıtmak gerekmektedir. Bu işlemi aşağıdaki adımları sırayla yaparak gerçekleştiririz.

* <http://temelelektronik.info/ders/download/microburn.zip> adresinden indireceğimiz sıkıştırılmış dosyayı bilgisayarımızın herhangi bir yerine winrar( <http://temelelektronik.info/ders/download/wrar401tr.exe> ) ya da winzip gibi programlarla açarız.
* Sıkıştırılmış dosyaları açtığımız klasörden programlayıcı kartımız takılı değilken “PL2303\_Prolific\_DriverInstaller\_v1.7.0.exe” dosyasını çalıştırırız ve kurulumu gerçekleştiririz.
* Programlayıcımızı bilgisayarımıza bağlarız. Kurulum işlemi tamamlanmış olur.

**Micro Burn Programı ile Mikrodenetleyiciye Program Yüklenmesi**

Kurulum işlemi gerçekleştirildikten sonra mikro burn programının bulunduğu klasörden “microbrn.exe” programın çalıştırırız. Açılan programda port ayarı yapmamız gerekiyor. Port ayarı aşağıdaki adımlar izlenerek yapılır.

* Masaüstünden “bilgisayarım” simgesine sağ tılanarak “özellikler” seçilir. Açılan pencereden aygıt yöneticisine tıklanır. “Bağlantı noktaları” ‘nın üzerine tıklanır ve seçenekler açılır. Burada Prolific….. şeklinde gözüken maddenin en sağındaki COMX yazısının sonundaki port numarası öğrenilir.
* Bu port numarası Mikro Burn programında file menüsünden port seçeneği seçildikten sonra çıkan kutuya yazılır ve “ok” butonuna basılır.
* Hangi entegre için program yüklenecekse entegre listesinden seçilir.
* “Load” butonuna basılarak bilgisayardan “hex” dosyası seçilir.
* Entegre sağ üst köşedeki resimdeki gözüktüğü gibi programlayıcıya yerleştirilerek mandala basılır.
* Program butonuna basılarak “hex” dosyası entegreye yazılır.

\*\*\*Program yükleme yapıldıktan sonra yükleme problemsiz gerçekleşse bile hata veriyor. Bu hata alınmadan program yüklenmek isteniyorsa. “Program” butonuna basılmadan önce “fuses” butonuna basılıp açılan mesaj kutusunda hiçbir işlem seçilmeden kapatılmalı ve “program” butonuna basılarak programlama işlemi gerçekleştirilmelidir.

**MikroC programının oluşturuduğu dosya türleri**

MikroC programı bir çok dosya oluşturmaktadır. Bunlardan en önemli 3 tanesi aşağıdaki gibidir bu uzantıların ne işe yaradığını bilmek bizim için yeterli olacaktır.

**.C uzantılı dosya:** MikroC’de ekrana kullanıcı tarafından yazılan program satırlarının bulunduğu dosyadır.

**.hex uzantılı dosya:**MikroC’de siyah ekrana yazılan satırların build menüsünden build seçeneği seçilerek derlendikten sonra oluşturulan dosyadır. Bu dosya Pic entegresinin içine yüklenen dosyadır. .c uzantılı dosyada değişiklik yapıldıktan sonra build işlemi gerçekleştirilmeden değişiklikler hex uzantılı dosyaya kaydedilmez.

**.mcppi:** pic proje dosyası uzantısıdır. Bu dosya MikroC programı kapatıldıktan sonra projenin yeninden açılması gerektğinde kullanılır.

Kaydedilmiş bir projenin yeniden açılması işlemi;

1. Mcppi uzantılı dosyaya çift tıklanarak , ya da
2. MikroC programı açıldıktan sonra “Project” menüsünden “Open Project” seçeneği seçildikten sonra açılan pencereden Mcppi uzantılı dosya seçilerek kaydedilmiş proje tekrar açılabilir.

**Not:** MikroC ile hazırlanmış bir projenin Flaş diske kaydedilebilmesi için ya da farklı bir bilgisayarda çalıştırılabilmesi için MikroC tarafından oluşturulmuş c,hex,mcppi uzantılı dosyaların yanı sıra MikroC tarafından oluşturulmuş diğer tüm dosyaların da taşınması gerekmektedir. Herhangi bir proje oluşturlup derlendikten sonra projeye ait 16 dosya oluşturulmaktadır. Bu projenin farklı bir bilgisayarda çalıştırılabilmesi için tüm bu dosyaların tamamının taşınması gerekmektedir.

**Bit Byte Word terimleri**

Bit:Bit en küçük bilgisayar verisine verilen isimdir. 1 ya da 0 olabilir. Örnek olarak: 11 🡪 2 bitlik bir bilgisayar verisidir. 111110010🡪 9 bitlik bilgisayar verisidir.

Byte: 8 bitlik bilgisayar verisine Byte denir.

Örnek:

10101010 🡪 1 byte ya da 8 bit

1010101010101010 🡪 2 byte ya da 16 bit

Word: 16 bitlik ya da 2 byte’lık bilgisayar verisine 1 word denir.

Örnek:

1010101010101010 🡪 1 word, 2 byte, 16 bit

10101010101010101010101010101010🡪 2 word, 4 byte, 32 bit

**MikroC’de noktalı virgül “;” kullanımı**

Program yazılırken komutların birbirinden ayrılması için kullanılır. Her komuttan sonra noktalı virgül kullanılır. Noktalı virgül kendisinden önceki komutun bittiği analamına gelir.Noktalı virgülün kullanılması önceki komut bitti noktalı virgülden sonra yeni komut yazacağım anlamına da gelir.

**MikroC’de Değer atamak(= atama operatörü)**

= işareti atama işleminin gerçekleştirilmesi için kullanılır**. Bu** işaretin sağındaki değer solundaki yere atanır.

Örnek:

Portb = 15; //b portuna = işaretinin sağıkdaki değer portb komutuna atanmıştır.

**MikroC sayıların ifade edilmesi**

Sayısal değerler 10 tabanında ya da 2 tabanında yazılabilir. Yazılışları farklı olmasına rağmen kullanımından herhangi bir fark yoktur.

İkili sayılar kullanılırken ikili sayıdan önce “**0b**” öneki eklenmelidir. Aksi taktirde bilgisayar bu sayıyı 10 tabanında kabul edecektir.

Örnekler:

1, 2 , 233, 123, 333,454 on tabanındanki sayıları integer olarak ifade edilir.

**0b**00011110, **0b**01010010,**0b**11100011 iki tabanındaki sayılari ifade eder.

Not:

Portb=255;

Portb=0b11111111;

Yukardaki iki satır arasında yapılan işlem olarak bir fark yoktur çünkü 0b11111111 sayısı on tabanında 255’e eşittir.

**MikroC’de açıklama satırları(//)**

Program yazılırken istenilen komut satırından sonra “//” işareti konularak programla ilgili açıklama eklenebilir. // işaretinden sonra yazılan kısımlar sadece program yazan kullanıcının yazdığı programı daha kolay anlayabilmesi için kullanılır. // işaretinden sonra yazılan yazılar Pic entegresinin çalışmasında bir değişikliğe yol açmaz.

**En Temel MikroC fonksiyonu**

Void main()

{

}

Her yazılan programda mutlaka en temel MikroC fonksiyonu bulunur. Proje ilk olarak oluşturulduğunda en temel MikroC fonksiyonu MikroC programı tarafından yazılmış halde sayfa açılır. Burada dalgalı parantezlerin arasına yazılan program kodları, PIC entegresine besleme gerilimi verildiğinde ilk olarak işlenecek kodlardır. Dalgalı parantezlerden biri veya ikisi silinirse program çalışamaz. Program dalgalı parantezlerin dışına yazılırsa çalışmaz.

Parantez Türleri

MikroC dilinde bir kaç çeşit parantez kullanılır bunlardan en yaygın olanları dalgalı parantez, ve normal parantezdir.

Dalgalı Parantez{}

Dalgalı parantez “Alt Gr” tuşu basılıyken 7 ve 9 rakamlarına basılarak ekrana yazılır. Dalgalı parantezler bir etki söz konusu olduğu zaman kullanılır. Örneğin yukardaki en temel mikroC fonksiyonunda en temel mikroC fonksiyonunun etkisi “{“ dalgalı parantezin açıldığı noktadan sonra başlar ve dalgalı parantezin kapatıldığı noktaya kadar devam eder. Bu prensim dalgalı parantezin kullanıldığı her yerde bu şekildedir. Yani dalgalı parantez kullanılmışsa bir etki söz konusudur.

Normal Parantez()

Normal Parantez “Shift” tuşuna basılıyken 8 ve 9 rakamlarına basılarak ekrana yazılır. Normal parantezler bir bilgi girişi yapılacağı zaman kullanılırlar. En temel mikroC fonksiyonunda bilgi girişi olmadığı için parantezi içi boştur. Normal parantezle Dalgalı parantez çoğunlukla karıştırılır. Bunu kolayca ayırabilmek için; “Normal parantez birden fazla satır bilgi bulundurmaz, dalgalı parantez ise çoğunlukla bir satırdan fazla bilgi barındırır” cümlesi kullanılabilir.

Örnek: Delay\_ms(1000); //Burada 1000 sayısının fonksiyona girişi gerçekleştirilmiştir.

Not: Parantezler kesinlikle birbirinin yerine kullanılamaz. Açılan parantez sayısı kapatılan parantez sayısına eşit değilse program çalışmaz. Parantezlerin yeri programın çalışma şeklini etkiler.

**Örnek:**

1. void main()

{ //parantezin içi boş olduğu için parantez “}”’den

} //sonraki satırlar çalıştırılmaz.

Trisb=0;

Portb=255;

1. void main()

{ //Parantezden sonraki iki satır işlendikten sonra

Trisb=0; // program sonlanır. Programda hata yoktur.

Portb=255;

}

1. void main()

{

Trisb=0; //açılan parantez kapatılmadığı için program derlenirken

Portb=255; //hata verir. Açılan parantez sayısı kapatılan parantez sayısına

//eşit olmalıdır.

Yukarıda parantezlerin kullanımı ile ilgili örnekler ve sık yapılan hatalar verilmiştir. A ve b örneğinde aynı komutlar kullanılmasına rağmen parantezlerin bulunduğu yerlerin farklı olması nedeniyle farklı sonuçlar alınmıştır. C şıkkında ise dalgalı parantez kapatılmadığı için program hata vermiştir, a şıkkında ise program en temel mikroC fonksiyonunun dışına yazıldığı için çalışıtırlmamıştır.

**Port Nedir**

Port entegrenin giriş çıkış bacaklarına verilen isimdir. PIC entegresini büyük bir alışveriş merkezine benzetirsek portları da kapılara benzetebiliriz. PIC entegresi portları giriş ya da çıkış olarak kullanılabilir. 16f628a entegresinde A portu ve B portu olmak üzere 2 adet port bulunmaktadır. Her port 8’er bacak(bit)’tan oluşmaktadır. A portu 8 bacaklıdır ve RA0 ile RA7 arasındaki 8 bacak A portunu oluşturur A portunun ilk beş portundan sonrası(RA5-RA7) genellikle kullanılmaz . B portu da 8 bacaklıdır, RB0-RB7 B portunu oluşturur. 16f628a entegresini bir alışveriş merkezi olarak düşünürsek bu alışveriş merkezinin 16 adet kapısı bulunmaktadır. Bu kapılar bizim isteğimize göre giriş ya da çıkış olarak kullanılabilmektedir. A ve B portlarının giriş ya da çıkış olacağını programın başında belirtmemiz gerekir. Bu işlemi belirtmek için TRIS komutu kullanılır. A ya da B portu denilince akla 8 adet uç gelmelidir. Örneğin A portu denilince RA0-RA7 uçları gelmelidir.

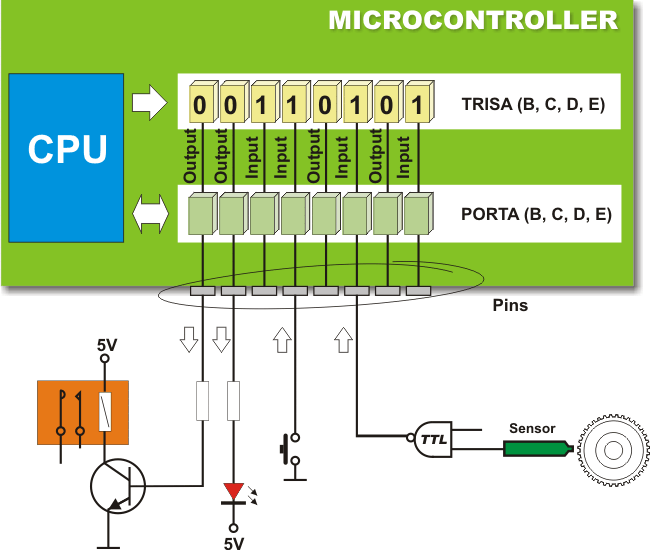
**Uçların giriş(input) ya da çıkış (output) olarak kullanılması**

Bir entegre ucunun giriş olarak kullanılması demek dışarıdan buraya bilgi girişinin yapılması demektir. Örneğin entegre bacağına bağlı bir butonumuz var ve butonun basılı olması durumu bizim için giriş bilgisi demektir. Mesela RB0 ucuna bir buton bağladık, bu butona basıldığında entegrenin bir lambayı yakması çekildiği durumda da söndürmesini istiyoruz. Bu durumda butonun yaptığı görev RB0 ucuna giriş bilgisi sağlamaktır. Bu durumda RB0 giriş olarak ayarlanmalıdır.

İnput 🡪**1**nput: Giriş

Output🡪 **0**utput: Çıkış

Bir entegre ucunun çıkış olarak kullanılması demek dışarıya bilgi çıkışı yapılması demektir, yukarıdaki örnekte düğmeye basıldığında lambanın yakılmasını istiyorduk, burada lambanın yakılması için lambamıza PIC entegremizden bilgi çıkışı yapmamız gerekmektedir. Bu işlemi PIC entegresinin bir ucunu çıkış olarak kullanarak gerçekleştirebiliriz. BU durumda entegre içinden entegre dışına gerilim uygulama söz konusu olur bu işleme PIC entegresi ucunun çıkış olarak kullanılması denir. Örnek RB0 ucuna bağlı led’in yakılması ve söndürülmesi işlemi bir çıkış işlemidir.



**Dijital Mantığı (Dijital 1 ve Dijital 0)**

Pic 16f628a entegresi ve diğer Pic Entegreleri dijital elektronik prensibine göre çalışmaktadır. Bu yüzden giriş ve çıkışlarda sadece dijital 1 ve dijital 0 değerleri kullanılabilmektedir. Dijital 1 değeri 5 Voltla, dijital 0 değeri de 0 Voltla belirtilmektedir. Örnek olarak RB0 ucunu dijital 1’e ayarlmak demek RB0 ucunu 5Volt gerilme ayarlamak demektir. Dijital 0’a ayarlamak ta 0 Volta ayarlamak demektir.

**MikroC’de komut yazımı**

MikroC komutları en temel MikroC fonksiyonunda dalgalı parantezlerin içine yazılır. Komutlar yazılırken her komuttan sonra diğer komuta geçerken “;” noktalı virgül işareti konur bu işaretin konuluşunun amacı komutları birbirinden ayırmaktır. Programlar işlenirken komutlar satır satır işlenir. Yani bir komut bittikten sonra diğer komuta geçilir.

**MikroC Komutları**

1. **Tris komutu:** 16f628a entegresinde A ve B portlarını giriş ya da çıkış olarak ayarlamak için kullanılır. Tris komutu hangi port için kullanılacak o portun harfi Tris kelimesinden sonra getirilir. Örnek A portunu giriş ya da çıkış olarak kullanmak için Trisa, B portu için TrisB komutu kullanılır. Burada Tris komutunun alabileceği iki değer vardır. Bunlar 0 ve 1’dir. Tris komutuna 0 değeri atanırsa port çıkış, 1 değeri atanırsa port giriş olarak kullanılır.

Örnek:

Trisa=0b00000000; // A portu(RA uçları)na 0 değeri atandığı için port çıkış olarak ayarlandı

Trisb=0b11111111; // B portu (RB uçları)na 1 değeri atandığı için port giriş olarak ayarlandı.

Trisb=0b00001111; //B portuuçlarından ilk dördü giriş son dördü çıkış olarak ayarlandı

\*\*\*\* burada 0 ve 1 sayılarının giriş mi çıkış mı için olduğu karıştırılırsa in ve out kelimelerinin baş harfi hatırlama işlemini kolaylaştırır. (**i**n,**1**N-**O**ut,**0**ut) 1 in yani giriş kelimesinin baş harfini 0’da out yani çıkış kelimesinin baş harfini oluşturur.

1. **Port komutu:** A ve B portlarına değer atamak ya da değer okumak için kullanılır. Porta ya da portb komutu A ya da B portu bacaklarındaki dijital bilgileri ifade eder. Port komutu da Tris komutu gibi hangi port için kullanılıyorsa o portun harfini sonuna alır. Porta ve Portb gibi. Port komutu kullanıldığında port komutundan sonra kullanılan = işaretinin sağındaki değer istenilen porta gönderilir. Ancak bu gönderme işleminden önce gönderilecek sayı entegre içinde iki tabanına dönüştürülür.

Örnek: **porta=25;** //Porta komutuna 25 atandığı için 25’in ikili karşılığı olan 11001 sayısı A portuna gönderilir. Bu durumda gönderilen sayının en sağındaki bit Ra0 ucuna olmak üzere sırayla Ra1’den Ra7’ye kadar dağıtılır. Dağıtım işleminden sonra değeri 1 olan bacaklar 5V değerine ayarlanırken , 0 olan bacaklar 0V’a ayarlanır. Yani yukardaki örnekte Ra0,Ra3, Ra4 uçları 5V’a ayarlanırken diğer bacaklar 0 V a ayarlanır.

**Portb=45;** //45’in ikili karşılığı 101101 olduğu için RB0,RB2,RB3,RB5 uçları Dijital 1 durumuna yani 5 Volta ayarlanır. Diğer uçlar dijital 0 yani 0 Volt olarak kalır.

**Portb=255;** //255’in ikili karşılığı 11111111 olduğu için RB0 ile RB7 arası tüm uçlar dijital 1 durumuna yani 5 Volta ayarlanır.

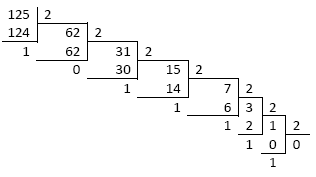
**\*\*Portb=0b11111111;** komutu **Portb=255;** komutu ile aynı işlemi yapar. Sadece bir tanesinde B portuna atanacak değer iki tabanında diğerinden on tabanında girilmiştir.

Komutun kullanılacağı yere göre istersek sayıyı on tabanında istersek iki tabanında girebiliriz. Sadece sayı iki tabanında girilirken başına 0b ekini eklemeye dikkat etmemiz gerekir. Bu eki eklemezsek sayının on tabanında yazılmış olduğu kabul edilir.

\*\*\*\*On Tabanındaki Sayıların İki Tabanına Dönüştürülmesi

MikroC programında kullanılan sayılar on tabanında ya da başka tabanlarda olabilir. On tabanında yazılan sayıların çıkış olarak bir porta gönderilmesi ya da dışardan okunabilmesi durumunda on tabanındaki sayıların portlara gönderilmesi ve dışardan okunması işlemi iki sayı tabanında yapılır. On tabanındaki sayıların ikili tabana dönüştürülmesi işlemi on tabanındaki sayının sürekli olarak bölümü 0 çıkana kadar 2’ye bölünmesi ve en sonunda bölme işlemlerinin kalanlarının tersten yazılması şeklinde yapılır. Bu işlem pratik olarak windows hesap makinesi bilgisayar programcısı modunda kullanılarak ta yapılabilir.

Örnek: On tabanındaki 125 sayısını 2 tabanına dönüştürülmesi işlemi. Sonuç iki tabanında 1111101’ dir. Yani 10 tabanındaki 125 sayısı ve iki tabanındaki 1111101 sayısı birbirine eşittir.



Şekil : 10 tabanında yazılmış sayıları iki tabanına dönüşümü

Ödev: Aşağıdaki sayıları ikili sayılara dönüştürün.

1. (255)10=(......)2
2. (144)10=(......)2
3. (55)10=(......)2
4. (88)10=(......)2
5. (13)10=(......)2
6. (7)10=(......)2
7. (68)10=(......)2
8. (44)10=(......)2
9. (33)10=(......)2
10. (17)10=(......)2

Not: İkili sayılarla ilgili bilginizi <http://forums.cisco.com/CertCom/game/binary_game.swf> adresindeki oyunu oynayarak pekiştirebilirsiniz

1. **Delay\_ms() Fonksiyonu:** Bekleme fonksiyonudur. Parantez içine yazılan rakam kadar milisaniye programı hiç birşey yapmadan bekletir.

Örnek: Delay\_ms(100); // Programı 100 milisaniye bekletir.

Delay\_ms(1000); // Programı 1000 milisaniye yani 1 saniye bekletir.

Delay\_ms(500); //Programı 500 milisaniye yani yarım saniye bekletir.

**Örnek MikroC uygulaması:** B portuna 45 sayısını verip 1 saniye bekledikten sonra B portuna 0 sayısını veren program

Void main() //Entemel MikroC fonksiyonu

{

Trisb=0; // B portu yani RB0-RB7 uçları çıkış olarak ayarlandı.

Portb=45; //B portuna 45 sayısının ikili karşılığı gönderildi

Delay\_ms(1000); //1000 milisaniye yani 1 saniye beklendi

Portb=0; //B portuna 0 sayısı gönderildi.

}

**Not:** Her komutun farklı satıra yazılma zorunluluğu yoktur istenirse komutlar arka arkaya da yazılabilir. Ancak yazılan programda hata olması gibi durumlarda programın anlaşılırlığını artırmak için her komut ayrı satıra yazılır.

Örneğin;

Trisb=0;portb=45; satırı ile aşağıdaki satırlar arasında yapılan işlem bakımından fark yoktur.

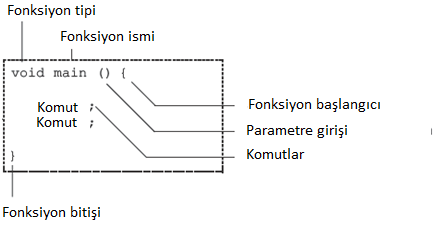
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Trisb=0;

Portb=45;

**Fonksiyonlar**

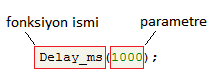
Program içerisinden birden fazla kullanılacak komutlar kümesinin isimlendirilmesine denir. Fonksiyonlar bir defa oluşturulduktan sonra kendi isimleri ile kullanılırlar. Örneğin 5 satırlık bir komut kümesini bir fonksiyon haline getirip bu fonksiyona fonksiyon1 ismini verirsek programda fonksiyon1 yazdığımız her yerde bu beş satır çalıştırılır. Fonksiyonlar aşağıdaki gibi oluşturulur. Fonksiyonlar ikiye ayrılırlar. Bunlar hazır fonksiyonlar ve kullanıcı tanımlı fonksiyonlardır. Hazır fonksiyonlar daha önceden yazılmış fonksiyonlardır. Örneğin Delay\_ms(), hazır fonksiyon için iyi bir örnektir. Bu fonksiyonları kullanmak için önceden tanımlamamıza gerek yoktur. Kullanıc tanımlı fonksiyonlar ise bizim tarafımızdan oluşturulan fonksiyonlardır. Biz daha çok hazır fonksiyonların kullanışı ile ilgili uygulamalar yapacağız.

Fonksiyonlar aşağıdaki kısımlardan oluşurlar.

1. Fonksiyon tipi
2. Fonksiyon ismi
3. Parametre girişi
4. Fonksiyon başlangıcı
5. Komutlar
6. Fonksiyon bitişi

**Fonksiyonların kullanılması**

Fonksiyonlar kullanılırken isim ve parametreden oluşan 2 kısım yazılarak çalıştırılır. En yaygın kullanılan fonksiyon Delay\_ms(); fonksiyonudur. Bun fonksiyondan yola çıkarsak “Delay\_ms” bu fonksiyonun ismidir. “()” parantezin içine de parametre yazılır. Delay\_ms fonksiyonunun parametresi bekleme fonksiyonunun ne kadar süre bekleyeceğini belirtir. Parametreler parantezin içine girilir. Her fonksiyonda parametre olacak diye bir kural yoktur. Bazı fonksiyonlar parametresiz de kullanılabilir.



**Operatörler**

Operatörler belli işlerin yapılmasında kullanılan kısaltma sembollerdir.

**Atama Operatörü(=)**

= işareti atama işleminin gerçekleştirilmesi için kullanılır**. Bu** işaretin sağındaki değer solundaki yere atanır.

Örnek:

Portb = 15; //b portuna = işaretinin sağıkdaki değer portb komutuna atanmıştır.

**İlişkisel Operatörler(==,!=,>,<,>=,<=)**

Doğru ya da yanlış olan cümleciklere önerme denir. Mesela 5 sayısı 2 sayısına eşittir önermesi yanlış bir önermeyken, 7 sayısı 5 sayısından büyüktür doğru bir önermedir. İlişkisel operatörler bilgisayar dilinde önermeler yazmak için kullanılan sembollerdir. İlişkisel operatörler kullanılarak yazılan önermeler doğruysa önerme 1 değerini alır, yanlışsa önerme 0 değerini alır.

**== (Eşit)Operatörü**

Sağına ve soluna yazılan değerleri karşılaştırır, eğer değerler eşitse 1, değilse 0 değeri sonuç olur.

Örnek:

3==2; //işlemin osnucu 3, 2’ye eşit olmadığı için 0’dır. Yani 3,2’ye eşit olmadığı için sonuç yanlış olur.

2==2; //2, 2’ye eşit olduğu için sonuç 1’dir.

**Eşit Değil (!=) Operatörü**

Sağına ve soluna yazılan değerleri karşılaştırıp, eşit olmadığında 1, eşit olduğunda 0 sonucunu alan operatördür.

Örnek:

1!=2; // 1 sayısı 2’ye eşit değildir ifadesi doğru olduğu için sonuç 1 değerini alır.

1!=1; //1 sayısı 1 sayısına eşit değildir ifadesi yanlış olduğu için sonuç 0’dır.

Not: Ünlem işareti önüne getirildiği operatörü tersine çevirir. Örneğin “=” ifadesi eşit iken “!=” ifadesi eşit değil anlamı vermektedir. Bu mantık “!<,!>,” ifadeleri için de geçerlidir.

1!<2; //1 küçük değildir 2 ifadesi yanlış oldğu için sonuç 0’dır.

**Büyük(>) Operatörü**

Sağına ve soluna yazılan değerlerden soldaki değerin sağdaki değerden büyük olması durumunda 1 aksi taktirde 0 döndüren operatördür.

**Küçük(<) Operatörü**

Sağına ve soluna yazılan değerlerden soldaki değerin sağdaki değerden küçük olması durumunda 1 aksi taktirde 0 döndüren operatördür.

**Büyük(>=) Operatörü**

Sağına ve soluna yazılan değerlerden soldaki değerin sağdaki değerden büyük olması veya eşit olması durumunda 1 aksi taktirde 0 döndüren operatördür.

**Küçük(<=) Operatörü**

Sağına ve soluna yazılan değerlerden soldaki değerin sağdaki değerden küçük olması veya eşit olması durumunda 1 aksi taktirde 0 döndüren operatördür.

Örnekler;

100==100; // sayılar eşit olduğu için sonuç 1

20==21; //sayılar eşit olmadığı için sonuç 0

12>11; //soldaki sayı büyük olduğu için sonuç 1

12<11; //soldaki sayı küçük olmadığı için sonuç 0

11<=11; //soldaki sayı eşit olduğu için sonuç 1

11<=12; //soldaki sayı küçük olduğu için sonuç 1

**Sonsuz For(için) Döngüsü**

Belli bir mantıksal işlemin sonucunun doğru olduğu durumlarda sürekli olarak bir dizi komutun işlenmesi istendiğinde kullanılan döngüdür.

for(*;;*)

{

Program komutları //Dalgalı parantezlerin arasına yazılan komutlar dalgalı parantezin içindeki Program komutları // önermenin doğru olması durumunda sürekli olarak işlenir. Komutlar işlenip Program komutları //}’e gelindiğinde parantezin içi tekrar kontrol edilir ve önerme hala 1 // sonucunu döndürüyorsa döngünün ilk satırına tekrar dönülür ve { dalgalı // parantezden sonraki satırlar tekrar tekrar işlenir. () parantez içindeki // önermenin sonucunun 0 dönmesi durumunda döngüden çıkılır.

}

Örnek Uygulama: Sürekli olarak bir saniye B Portunun tüm bitlerini 1 yapan sonra 1 saniye tüm bitleri 0 yapan örnek program. Enerji verldiği sürece bir saniye tüm çıkışlar 1 yapılır, 1 saniye 0 yapılır.

Void main() //En temel MikroC fonksiyonu

{

Trisb=0; // B portu çıkış olarak ayarlandı

for(;;) //While döngüsü == operatörünün sağındaki ve solundaki ifade eşit

{ //olduğu sürece (1==1 sonucu her zaman 1’dir) devam edecek döngü

Portb=255; // B portuna 255 sayısının ikili karşılığı olan 11111111 sayısının //gönderilmesi

Delay\_ms(1000); //Bir saniye beklenmesi

Portb=0; //B portunun tüm bitlerinin sıfırlanması

Delay\_ms(0); //Bir saniye beklenmesi

} //Bu paranteze gelinince program () parantez içini kontrol eder sonuç //1 se döngünün başına dönülüp devam edilir aksi taktirde döngüden //çıkılır. 1, 1’e eşit olduğu için döngüden çıkılamaz.

}

for döngüsünde () içine “;;” yazılması durumunda *for* döngüsü sonsuz devam eder. Yani enerji verildiği sürece bu döngüden çıkılamaz. Sonsuz *for* döngüsü program yazılırken çok sık kullanılır. For döngüsünün dalgalı parantezlerinin içine yazılmış komutlar sürekli olarak, entegreye enerji verilmeye devam edildiği sürece tekrarlanır.

For döngüsünüz istenilen sayıda işlem yapıp çıkan bir versiyonuda mevcuttur. İstenilen sayıda işlem yapıp döngüden çıkan for döngüsüne ilerleyen sayfalarda yer verilecektir.

\*\* Sonsuz **for** döngüsü

for(;;) // sonsuz döngü

{

}

**Proteus Isis’te Deneme Devresi Oluşturulması**

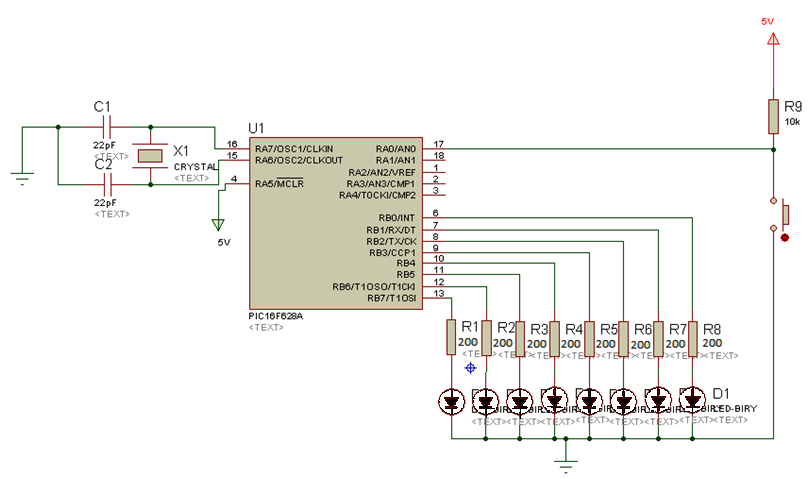
MikroC programında yazılan programların farklı elemanların bulunduğu deneme devrelerinde denenmesi mümkün olmaktadır. Ancak programın öğrenme aşamasında sık sık ve farklı devrelerde uygulanması fayda sağlayacaktır. Devre yapımı zaman ve maliyet gerektiren bir işlem olduğu için MikroC’de yazılacak programların ilk aşamada Elektronik Devre Simülasyon programı olan Proteus Isis programında istenilen devreler kurularak denenmesi bize zaman ve maliyet açısından fayda sağlayacaktır. Proteus Isiste program denemesi yaparken dikkat etmemiz gereken bazı noktalar vardır.

1. Isis programında bulunan Pic entegrelerine besleme gerilimi otomatik olarak verilmektedir.
2. Isis programına osilatör bağlanmadığı durumlarda da program sağlıklı bir şekilde çalışmasına rağmen aynı programlar gerçek hayatta osilatöre ihtiyaç duyar.
3. Isis programında Pic16f628A entegresinin 4 numaralı bacağına 5 Volt bağlanmadığı durumda da program çalışmaya devam ederken, gerçek devrelerde 4 nolu bacağa gerilim uygulanmadığında sürekli reset algıladığı için devre kararsız çalışır.

Proteus programı <http://www.meykitap.com/ders/download.aspx> adresinden indirilip kurulabilir.

**Deneme Devresi 1**

Temel fonksiyonların deneme işleminin yapılacağı isis deneme devremiz aşağıdaki gibidir.



Şekil İsis Temel Deneme Devresi

Not: Deneme devresi oluşturulduktan sonra hex dosyasının entegreye yüklenmesi işlemi aşağıdaki adımlar izlenerek yapılır.

Not: dikkat edilirse ledlerden önce 200 ohm’luk direnç bağlanmıştır, bunun nedeni entegrenin uçlarından dışarıya 5V verilmesidir, bilindiği gibi Ledlerin çalışma gerilimi 2V civarindadir. Direnç bağlanmadan Ledlerin bağlanması durumunda, ledler patlayacağı için; direnç bağlanması zorunludur.

1. Entegre üzerine sağ tıklanır
2. Çıkan menüden “Edit Properties” seçilir
3. Açılan pencereden “Program File” yazan satırda bulunan klasör resmine tıklanır
4. Açılan pencereden hex dosyası seçilir.
5. “Processor Clock Frequency” satırınada kullanılan kristal osilatör frekansı girilir.

Örnek Uygulama: İsis Temel Deneme Devresinde B portuna birer saniye aralarla sırasıyla 0,2,4,8,16,32,64,128 sayılarını gönderen programı yazın

void main() //En temel mikroC fonksiyonu

{ // Fonksiyon parantezle başlatılıyor

trisb=0; // B portu çıkış olarak ayarlanıyor

portb=0; // B portuna 0 sayısı gönderiliyor

delay\_ms(1000); // Bir saniye hiç bir işlem yapılmadan bekleniyor

portb=2; // B portuna 2 sayısı gönderiliyor

delay\_ms(1000); // Bir saniye hiç bir işlem yapılmadan bekleniyor

portb=4;

delay\_ms(1000);

portb=8;

delay\_ms(1000);

portb=16;

delay\_ms(1000);

portb=32;

delay\_ms(1000);

portb=64;

delay\_ms(1000);

portb=128;

delay\_ms(1000);

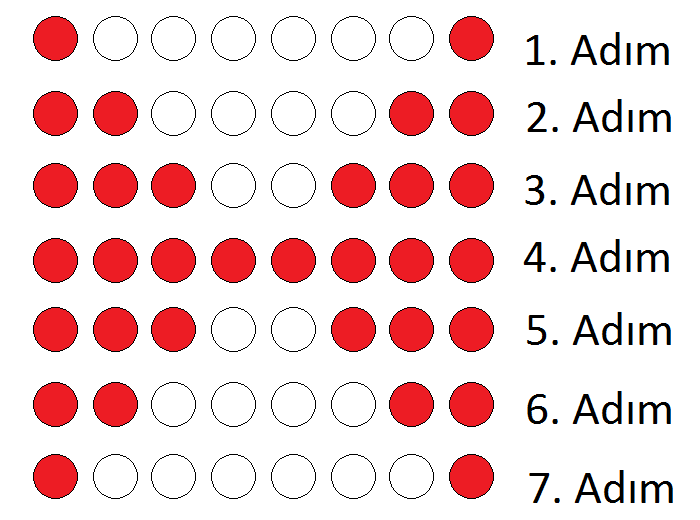
} // Parantezin kapatılmasıyla programdan çıkılıyor

Not: Bu program çalıştırıldığında ledlerin tek tek yandığı görülecektir. Yanan ledler b portuna gönderilen sayıların iki tabanına göre yanar. Örneğin 64 sayısı b portuna gönderildiğinde öncelikle ikili sayıya dönüştürülür sonra çıkışa yönlendirilir. 64 sayısının ikili karşılığı “01000000” olduğu için sadece soldan ikinci led yanacaktır. Ledlerin yanma durumlarını 1’ler sönme durumlarını 0’lar ifade eder.

Ödev: B portuna ikişer saniye aralıklarla sırasıyla 0,1,3,7,15,31,63,127,255 sayılarını gönderip ledlerin yanışını kontrol edin. Program bittiğinde hangi ledler sürekli yanıyor? Neden?

**Sayıların İkili Olarak Girilmesi**

Sayılar normal olarak girildiğinde MikroC programı sayıların 10 tabanında olduğunu anlar. Sayıları dilersek ikili tabanda kullanabiliriz. Bunu yapmak için sayı girişlerinden önce “0b” önekini ikili sayılardan önce girmemiz gereklidir. Örneğin “0b00001111” şeklinde belirtilmiş bir ifade ikili olarak 00001111 ikili sayısını belirtir. İkili sayıların “0b” öneki ile başlaması ifade edilen sayının iki tabanında ifade edildiğini belirtir.Sayıların ikili olarak girilmesi ve 10 tabanında girilmesi arasında şekil dışında hiçbir fark yoktur yani “0b11111111” ve 255 arasında değer olarak bir fark yoktur. Sadece yazım kolaylığı ve kolay anlaşılması bakımından hangisi uygunsa o yazım şekli kullanılır. Zaten 255’in ikili karşılığı 11111111’dir

**Örnek Uygulama(Perde uygulaması):** Yandaki şekile göre İsis temel deneme devresini programlayın(bekleme süresi 1 saniye). İlk deney devresinde kullanılabilecek basit bir led uygulaması.

void main() //En temel MikroC fonksiyonu

{

Trisb=0; //B portu çıkış olarak ayarlandı

for(;;) //sonsuz döngü

{

Portb=0b00000000; //B portuna ikili 00000000 sayısı gönderiliyor.

Delay\_ms(50); //50 milisaniye bekle

Portb=0b10000001;

Delay\_ms(50);

Portb=0b11000011;

Delay\_ms(50);

Portb=0b11100111;

Delay\_ms(50);

Portb=0b11111111;

Delay\_ms(50);

Portb=0b11100111;

Delay\_ms(50);

Portb=0b11000011;

Delay\_ms(50);

Portb=0b10000001;

Delay\_ms(50);

Portb=0b00000000;

Delay\_ms(50);

}

}

**Not:** Bu çalışmada RB4 çıkışı yani 10 numaralı bacağa bağlı lambanın yanması gerektiği halde yanmadığı görülür. Bu durumun nedeni MikroC programında varsayılan ayar olarak Düşük Voltaj Programlama (Low Voltage Programming ) özelliğinin aktive edilmiş olmasıdır(sadece eski mikroc programlarında). Bu özellik aktif olduğunda RB4 bacağı kullanılamaz. Bu durumun düzeltilmesi için Project menüsünden Edit Project’e tıklayarak, çıkan proje menüsünde Düşük Voltaj Programlama (Low Voltage Programming ) özelliğinin İptal (Disable) edilmesi gerekmektedir. Edit project menüsünden yapılan tüm değişiklerin programa yansıtılması için tekrar derlenmesi gerekmektedir.

**Port komutu ile bit bazında değer atması yapılması**

Porta ya da Portb komutu ile portlara istediğimiz değerleri atayabiliyoruz. Ancak yapmak istediğimiz işlem sadece bir bit’i yani A ya da B portunun sadece bir bitini değiştirmekse o zaman diğer bit değerini değiştirmeden sadece istediğimiz bit değerini değiştirebilmemiz faydalı olacaktır. Bu işlemi gerçekleştirebilmek için porta ya da portb ifadesinden sonra “.b[bit numarası]” şeklinde yazarak istediğimiz bit’in değerini ayarlayabilir ya da okuyabiliriz. Mesela porta.b1 ifadesi A portunun 2. Bitini yani RA1 ucunu ifade etmektedir. Bu şekilde sadece istediğimiz bacaktan bilgi okuya bilir ya da değer atması gerçekleştirebiliriz.

\*\*\*Önemli not: Burada b**x** yerine bacak numarasıda yazılabilir. Örneğin RA1 bacağının değeri 1 yapılmak istendiğinde

porta.b1=1; ya da porta.ra1=1; yazılabilir. İşlem olarak yapılan işlem aynıdır.

Örnek Uygulama: Bir defa sağdan sola kayan tek led uygulaması

void main() //En temel mikroC fonksiyonu

{

trisb=0; //B portu çıkış olarak ayarlandı

portb=0; //b portu sıfırlandı

portb.b0=1; //b portunun ilk biti RA0’ın değeri 1’e ayarlandı

Delay\_ms(1000); //bir saniye bekleme

portb.b0=0; //b portunun ilk biti sıfırlandı.

portb.b1=1; //b portunun ikinci biti RA1’in değeri 1’e ayarlandı

Delay\_ms(1000);

portb.b1=0;

portb.b2=1;

Delay\_ms(1000);

portb.b2=0;

portb.b3=1;

Delay\_ms(1000);

portb.b3=0;

portb.b4=1;

Delay\_ms(1000);

portb.b4=0;

portb.b5=1;

Delay\_ms(1000);

portb.b5=0;

portb.b6=1;

Delay\_ms(1000);

portb.b6=0;

portb.b7=1;

Delay\_ms(1000);

portb.b7=0;

}

**Sbit Komutu ile Bacaklara İsim Verilmesi**

**İs**tenirse bacakların isimleri programın başında sbit komutu ile değiştirilebilir örneğin rb0 ucuna bir adet led bağlı olsun. Biz bu ledi bir saniye aralıklarla yakıp söndürmek isteyelim. Programı aşağıdaki şekilde yazabiliriz.

**Sbit** “verilmek istenen isim” **at** “bacağın gerçek ismi”;

**sbit** Led **at** portb.b0; //portb’nin 0 numaralı Bacağına **Led** ismini verdik.

Void main()

{

Trisb=0;

For(;;)

{

Led=0; //portbnin 0 numaralı bacağına 0 volt uyguladık

Delay\_ms(1000);

Led=1; //portbnin 0 numaralı bacağına 5 volt uyguladık

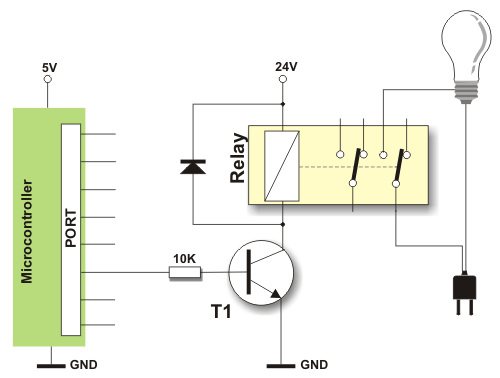
Delay\_ms(1000);

}

}

**Mikrodenetleyici Çıkışına Röle Bağlanması**

Mikrodenetleyici çıkışıyla 5 V’tan yüksek gerilim ile çalışan cihazlar kontrol edilmek istendiğinde bu tür cihazlar transistör ve röle kullanılarak sürülmelidir. Bu durumda aşağıdaki devre bağlantı şekli kullanılabilir.



Ödev: Yukardaki devreyi isis’te kurarak lambayı 5’er saniye aralıklarla yakıp söndüren programı yazın.

**Aritmetik Operatörler(+,-,\*,/,%)**

Bu operatörler sağına ve soluna yazılan değerlerle matematiksel işlemlerin yapılmasını sağlar.

Örnekler:

3+3 //Toplama işlemi yapılmasını sağlar sonuç 6’dır

4-1 //çıkarma işlemi yaıpılmasını sağlar sonuç 3’tür

3\*3 //çarpma işlemi yapılmasını sağlar sonuç 9 dur

4/2 //bölme işlemi yapılmasını sağlar sonuç 2 dir. Bölme işleminin sonucunun virgüllü çıkması durumunda virgülden sonraki kısmı ihmal eder. Örneğin 7/2 işleminin sonucu gerçek hayatta 3,5 olmasına rağmen 7/2 işleminin sonucu 3 olarak çıkar.

7%2 //% operatörü bölme işlemi sonucunda çıkan kalan sayıyı bulur. Yani sonuç 7’nin ‘2 ye bölünmesinden kalan 1’dir

Örnekler:

Portb=52%10; // Burada Portb ye 52’nin 10’a bölünmesinden çıkan 2 kalanı atanır.

Portb=4\*5; //Burada Portb ye 4\*5 işleminin sonucu olan 20 sayısı atanır.

**Örnek Uygulama:** Deneme Devresi 1 için ikili sayıcı uygulaması. Sıfırdan başlayarak her saniye ikili sayı çıkışını bir artıran program

void main() //En temel MikroC fonksiyonu

{

trisb=0; //B portu 0’a yani çıkışa ayarlanıyor.

portb=0; //B portunun değeri 0’a ayarlanıyor.

for(;;) //Sonsuz döngü

{

portb=portb+1; //B portunun değeri, B portunun değerinin bir fazlasına ayarlanıyor.

delay\_ms(1000); //1000 milisaniye(1 saniye) yani bir saniye bekleme

} //while döngüsünün başına dönülür.

}

**Bitsel Operatörler(<<,>>,&,|)**

Bitsel operatörler bir sayının bitlerinin modifiye edilmesine(değiştirilmesine) yarar. Genellikle ikili sayılarla kullanılırlar.

**<< sola kaydırma operatörü:** Bir sayının bitlerini istenilen adım kadar sola kaydıran operatördür. “<” işareti [z harfinin solundaki tuş]a basılarak ekrana yazılır.

0b00000001<<3; //işleminin sonucu 0b00001000 sayısıdır.

**>> sağa kaydırma operatörü:** Bir sayının bitlerini istenilen adım kadar sağa kaydıran operatördür. “>” işareti [shift + z harfinin solundaki tuş]a basılarak ekrana yazılır.

0b00010100>>4; //işleminin sonucu 0b00000001 sayısıdır yani sayıda bulunan 1’ler 4 adım sağa kaydırılır.

Not: Kaydırma işleminde alan dışına çıkan sayılar kaybolur.

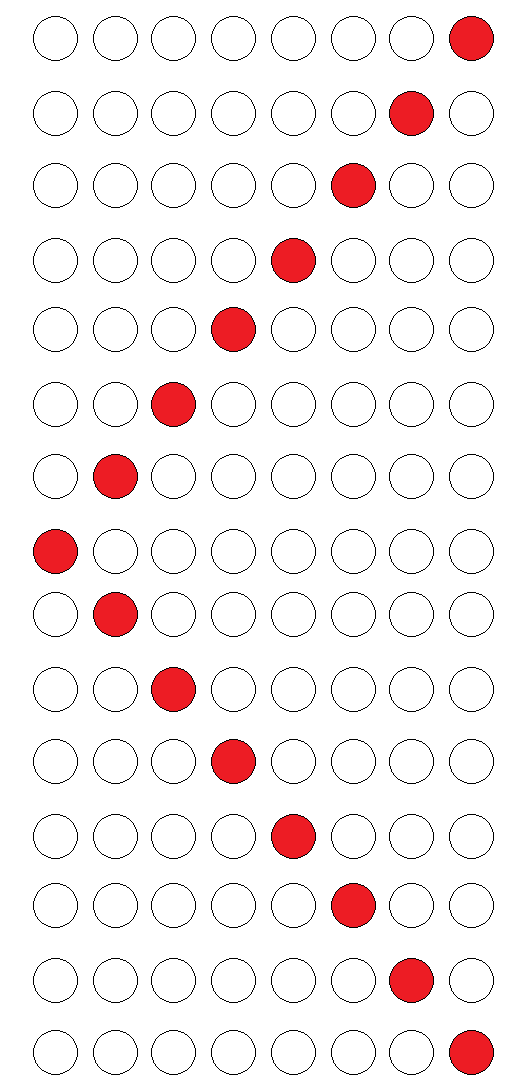
0b00000100>>4; //işleminin sonucu 0b00000000 sayısıdır yani sayıda bulunan 1 , 4 adım sağa kayıp alan dışına çıkmıştır.

**& ve(and kapısı) operatörü:** Operatörün sağına ve soluna yazılan sayılara mantıksal ve işlemi yapar. & işareti [shift + 6 rakamı]a basılarak ekrana yazılabilir.

0b00011111&0b11100010; // işleminin sonucu 0b00000010 sayısıdır iki sayının her biti ayrı ayrı mantıksal ve işlemine tutulmuş çıkan sonuç o bitlerin yerine yazılmıştır.

**| veya (OR kapısı) operatörü:** Operatörün sağına ve soluna yazılan sayılara mantıksal veya(OR) işlemi yapar. | işareti [alt gr + z harfinin solundaki tuş]a basılarak ekrana yazılabilir.

0b00011111 | 0b11100010; // işleminin sonucu 0b11111111 sayısıdır. Yani iki sayının her biti ayrı ayrı mantıksal veya işlemine tutulmuş çıkan sonuç o bitlerin yerine yazılmıştır.

**Örnek:** İsis temel deneme devresini yandaki karaşimşek şekline göre sağa ve sola kaydırma operatörünü kullanarak programlayın. Yapılan işlemi sonsuz(while) döngü kullanarak sürekli hale getirin(bekleme süresi 300 milisaniye).

void main()

{

trisb=0; //b portu çıkış olarak ayarlandı

portb=0b00000001; //b portunun değeri 00000001 şeklinde ayarlandı

for(;;) //sonsuz döngü

{ //Sonsuz döngü başlangıcı

portb=portb<<1; //b portundaki ikili sayı bir adım sola kaydırılıyor.B portu değeri=00000010

delay\_ms(100); //100 ms bekle

portb=portb<<1; //b portundaki ikili sayı bir adım sola kaydırılıyor. B portu değeri=00000100

delay\_ms(100); //100 ms bekle

portb=portb<<1; //b portundaki ikili sayı bir adım sola kaydırılıyor. B portu değeri=00001000

delay\_ms(100); //100 ms bekle

portb=portb<<1; //b portundaki ikili sayı bir adım sola kaydırılıyor. B portu değeri=00010000

delay\_ms(100); //100 ms bekle

portb=portb<<1; //b portundaki ikili sayı bir adım sola kaydırılıyor. B portu değeri=00100000

delay\_ms(100); //100 ms bekle

portb=portb<<1; //b portundaki ikili sayı bir adım sola kaydırılıyor. B portu değeri=01000000

delay\_ms(100); //100 ms bekle

portb=portb<<1; //b portundaki ikili sayı bir adım sola kaydırılıyor. B portu değeri=10000000

delay\_ms(100); //100 ms bekle

portb=portb>>1; //b portundaki ikili sayı bir adım sağa kaydırılıyor. B portu değeri=01000000

delay\_ms(100); //100 ms bekle

portb=portb>>1; //b portundaki ikili sayı bir adım sağa kaydırılıyor. B portu değeri=00100000

delay\_ms(100); //100 ms bekle

portb=portb>>1; //b portundaki ikili sayı bir adım sağa kaydırılıyor. B portu değeri=00010000

delay\_ms(100); //100 ms bekle

portb=portb>>1; //b portundaki ikili sayı bir adım sağa kaydırılıyor. B portu değeri=00001000

delay\_ms(100); //100 ms bekle

portb=portb>>1; //b portundaki ikili sayı bir adım sağa kaydırılıyor. B portu değeri=00000100

delay\_ms(100); //100 ms bekle

portb=portb>>1; //b portundaki ikili sayı bir adım sağa kaydırılıyor. B portu değeri=00000010

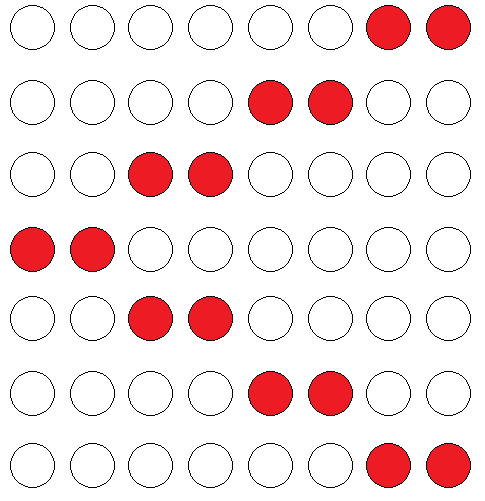
delay\_ms(100); //100 ms bekle

portb=portb>>1; //b portundaki ikili sayı bir adım sağa kaydırılıyor. B portu değeri=00000001

delay\_ms(100); //100 ms bekle

}

}



Ödev: İsis temel deneme devresinde yandaki animasyonun programını yazın(bekleme süresi 300 ms).

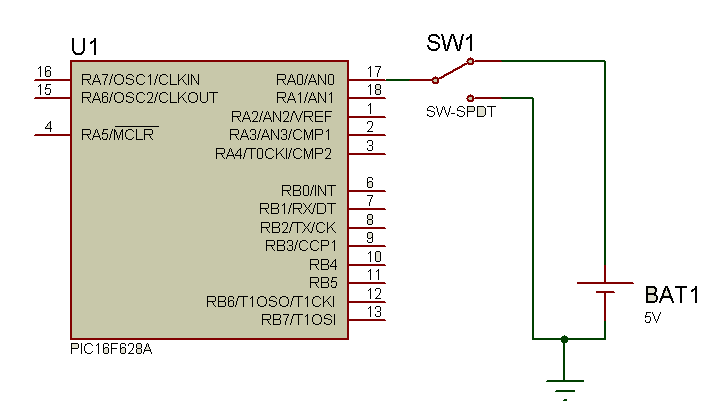
**Pic Entegresi portlarının giriş olarak kullanılması**

Entegre portlarının giriş ve çıkış olabileceğini daha önceki konularda işlemiştik. Entegre bacaklarının giriş ve çıkış olarak kullanılmasını bir marketin giriş ve çıkış kapılarına benzetebiliriz. Müşteriler eğer dışardan geliyorsa giriş yapılıyor. Müşteriler marketten ayrılıyorsa bu durumda çıkış yapılıyor demektir. Bu durum Pic Entegresi için entegre bacaklarına entegre içinden dışarı doğru gerilim veriliyorsa çıkış, entegre dışından entegre içene doğru gerilim uygulanıyorsa giriş olur. Entegre bacaklarının giriş olarak kullanılması durumu dışardan giriş bacağına gerilim uygulamakla olur. Eğer giriş ucuna 5 Volt gerilim uygulanırsa içerden dijital 1 olarak okunur, 0 Volt uygulanması durumunda da dijital 0 olarak okunur. Entegre giriş bacağına Deneme Devresi 1’deki gibi buton bağlamak en basit giriş uygulamasıdır.

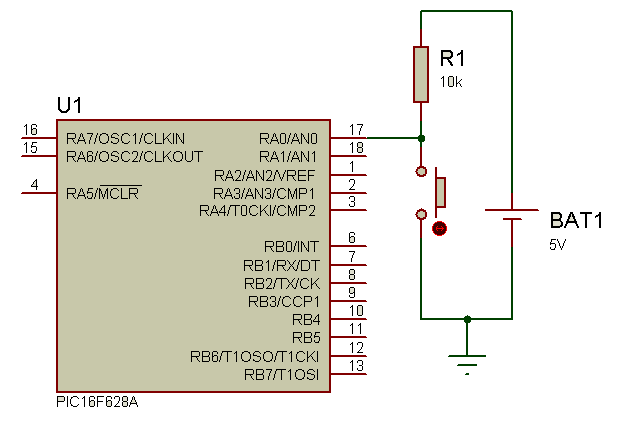
Örneğin B portu için RB0, RB1, RB3 bacaklarına 5 V diğer bacaklarına 0 V uygulansın, bu durumda entegrenin B portundan okunan değer ikili olarak 00001011 sayısı olacaktır. Bununda onlu karşılığı 11 sayısı olacaktır.

Örnek Buton Bağlantı Şekli

Entegre bacaklarından bir tanesine giriş amaçlı buton bağlanacağı zaman buton basıldığında entegre bacağına verilen dijital bilginin değişmesi gerekir. Yani butona basılmadığı durumda entegre bacağına 0 uygulanıyorsa, buton basılıyken 1 uygulanmalıdır. Aşağıdaki devrede butona basılmadığı durumda 5V(dijital 1) dirençler üzerinden uygulanırken butona basıldığında 0 V(dijital 0) 100 K’lık direnç üzerinden uygulanmaktadır. En basit olarak bir uca buton bağlanması aşağıdaki şekilde gerçekleştirilebilir. Anahtar üst konumuna getirildiğinde RA0 Bacağı pilin artı ucuna temas edeceği için RA0 ucuna 1 uygulanmış olur, tersi durumda ise RA0 bacağı pilin eksi ucuna temas edeceği için RA0 ucuna 0 uygulanmış olur.



Yukardaki bağlantı şeklinde iki konumlu anahtar kullanmak gereklidir. Bundan biraz daha pratik olan bağlantı şekli aşağıdaki gibidir. Bu bağlantı şeklinde butona basılı değilken RA0 ucuna 10K lık direnç üzerinden 5V gerilim uygulanır yani, RA0 ucuna 1 uygulanır. Butona basıldığı durumda ise, RA0 ucuna pilin eksi ucu bağlanır yani, RA0 ucuna 0 uygulanır. Yani butona basılmazken RA0’a 1, basılınca 0 uygulanmaktadır.

****

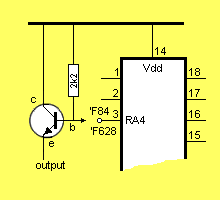
**Önemli Hatırlatmalar**

**\***RA2 ucu mikroC’de ayar yapmadan GİRİŞ olarak kullanılamaz.

*(programın başına tris komutlarından önce ya da sonra cmcon=0b00000111 yazılmalıdır)*

\*RA5 çıkış olarak kullanılamaz.

\* RA4 open kollektör devre düzeniyle çıkış olarak kullanılabilir.



Ra4’ün çıkış olarak kullanılması

**Örnek Uygulama:** RA0’dan girilen değeri RB0’a atayan program. Butona basılınca RB0’a bağlı ledi söndüren program.

void main()

{

Trisb=0; //B portu çıkış olarak ayarlanıyor

Trisa=1; //A portu giriş olarak ayarlanıyor

portb=0; //B portuna sıfır değeri gönderiliyor

for(;;) //Sonsuz döngü RA0 girişinin sürekli kontrol edilmesi için

{

Portb.b0=porta.b0; //RB0’a RA0’dan girilen atanıyor

}

}

**Dönem Değerlendirme soruları**

1. Aşağıdaki program kodlarının yapacağı işleri yanlarına yazın.
   1. Trisb=0;
   2. Portb=0b000111;
   3. While(1){ }
   4. For(i=0;i<100;i++){ }
   5. For( ; ; )
   6. Void main() { }
   7. Delay\_ms(275);
   8. Portb=255;
   9. Portb=1;
   10. Portb=255-254;
   11. Portb=51\*5;
   12. Portb=5/5;
2. **Aşağıdaki soruları yanlarına cevaplayın.**
   1. 16f628akaç bacaklıdır.
   2. 16f628a entegresi besleme uçları hangi numaralı bacaklardır?
   3. 16f628a zorunlu çalışma şartları kaç tanedir? Nelerdir?
   4. 16f628a çalışma gerilimi kaç volttur? Kaç volttan fazla gerilim uygulamak entegrenin yanmasına neden olur.
   5. B portu kaç bacaklıdır. Bu bacaklar sırasıyla entegrenin hangi bacaklarıdır.
   6. A portu kaç bacaklıdır. Bu bacaklar sırasyıla entegrenin hangi bacaklarıdır.
   7. Rb0 ucuna bir adet led bağlamak istesek en az kaç ohmluk direnç bağlamamız gerekir.
   8. C uzantılı dosyanın içinde ne bulunur?
   9. Proje oluşturulduğunda ilk defa hex uzantılı dosya ne zaman oluşturulur.
   10. 16f628a entegresi dahili osilatör frekansı kaç MHz dir?
   11. Portb=127; kodu b portuna bağlı ledlerden kaç tanesini yakar.
   12. Edit project menüsünden **Low voltage programming** seçeneği **disabled** yapılmazsa ne olur.
   13. Osilatör devresi yapıladığı halde osilatör XT olarak ayarlanırsa ne olur?
   14. Reset direnci bağlanmadığı halde MCLR seçeneği disabled yapılmazsa ne olur?
3. Aşağıdaki sayıları istenilen tabana dönüştürün.
   1. (35)10=(......)2
   2. (100)2=(......)10
   3. (70)10=(......)2
   4. (1100)2=(......)10
   5. (105)10=(......)2
   6. (10101)2=(......)10
   7. (140)10=(......)2
   8. (10101010)2=(......)10
4. Aşağıda istenilen programları yazın.
   1. Rb0 ucuna bağlı tek ledi bir defa bir saniye ara ile yakıp söndürecek program kodunu yazın
   2. B portuna bağlı tüm ledleri yakacak tek satırlık program kodunu b portuna on tabanında sayı girerek yazın.
   3. Rb0 ucuna bağlı tek bir ledi sürekli olarak yarım saniye aralıklarla yakıp söndürecek programını yazın
   4. B portuna bağlı tüm ledleri yarım saniye aralıklarla 20 defa yakıp söndürecek program kodunu yazın.
   5. Polis çakarı devresinin program kodlarını temel deneme devresine göre for döngüsü kullanarak yazın.
   6. B portuna bağlanmış ortak katot displaye numaranızın son hanesindeki rakamı yazdıracak program kodunu yazın.
   7. Entegreye enerji verdikten on saniye sonra rb0 ucuna bağlı ledi yakıp, bir dakika yanık kaldıktan sonra söndürecek program kodunu yazın.
   8. Portb=127; kodu ile aynı işlemi yapacak kodu on tabanındaki sayıyı iki tabanında girerek yazın.
   9. İki tane for döngüsü iç içe yazılırsa içteki for döngüsünün dalgalı parantezinin içindeki kodlar kaç defa işlenir.
   10. B portuna bağlı ledleri bir saniyede 500 defa yakıp söndürecek program kodlarını yazın.
5. Aşağıdaki devreleri çizin.
   1. 16f628a entegresi beslemesine verilecek gerilimi 5 voltta sınırlayacak regüle devresini çizin.
   2. 16f628a entegresinin rb0 nolu bacağından bir 220 voltluk lamba yakıp söndürülmek isteniyor. Gerekli röleli kontrol devresini çizin.
   3. 5 voltluk pille 16f628a entegresi çalıştırılmak isteniyor. Entegrenin minimum çalışma şartlarını sağlayacak devre şemasını çizin.
   4. Entegrenin rb0 ucundan 6 adet led beslenmek isteniyor. Devre şemasını basitçe çizin.

**Değişkenler**

Bilgisayar programlarında türü, ismi ve değeri olan değer tutucu bellek bölümlerine değişken denir. Değişkenlerin değerleri program akışı esnasında okunabilir ya da değişitirilebilir. Değişkenlerin kullanılabilmesi için öncelikle tanımlanması gerekir. Tanımlama işleminden sonra değişken ismi ile çağrılır. Değişkenler bilgisayar programları için vazgeçilmezdir.

**Değişkenlerin tanımlanması**

Değişkenler En temel MikroC fonksiyonundan önce tanımlanır. Tanımlama işlemi

Tür isim;

Şeklinde olur. Değişkenler bir çok türde olabilir. Ancak Mikrodenetleyicilerin programlarında genellikle Tür sayısal değişken olan “**int**” değişkenidir. Int değişkenleri 1, 255, 5654, -4567 gibi sayısal değerlerin hafızada tutulması için kullanılır. Yani int sayısal değişken türüdür.

Yeni int değişkeni tanımlandığı zaman ilk değeri otomatik olarak “0” olur. Bu değer istenirse tanımlama esnasında

Tür isim=değer;

Şeklinde de verilebilir. İlk değer verilmeyen değişkenlerin değeri “0”dır. Tanımlama işlemi gerçekleştikten sonra değişkenler En temel mikroC fonksiyonunun dalgalı parantezi içerisinde kullanılabilir.

Örnek 1:

int yakit=100; //ilk değerli tanımlama

void main()

{

//değişken burada tanımlı, yani kullanılabilir

}

Örnek 2:

int yakit; //ilk değersiz tanımlama

Void main()

{

yakit=75; //değişkenin çağırılması ve değer verilmesi

}

**Değişkene isim verilmesi**

Değişkenler isimlendirilirken ingiliz alfabesindeki harfler ya da “\_” karakteri ile başlayacak şekilde isimlendirilir. Değişken isimleri içerisinde ç,ö,ü,ğ,ş gibi ingiliz alfabesinde bulunmayan karakterler kullanılamaz.

**Örnek değişken isimleri:**

Yakit,Yas,\_yasin,\_not,\_maas,Gerilim,Fiyat,\_fiyat,Fiyat2 şeklinde yazılmış değişken isimleri olabilir.

**Hatalı değişken isimleri:**

Yakıt //değişken isimleri ı,ç,İ,ğ gibi karakterler barındıramaz

Yaş // değişken isimleri ş,ç,İ,ğ gibi karakterler barındıramaz

-yasin //değiken isimleri “\_” karakteri dışındaki sembollerle başlayamaz yada içinde bulunduramaz

.not // değiken isimleri “\_” karakteri dışındaki sembollerle başlayamaz yada içinde bulunduramaz

2dost // değişken isimlerinin ilk harfi rakam olamaz ancak diğer harfler rakam olabilir.

**Değişkenlere değer verilmesi ve değişken değerinin okunması**

Değişkenler tanımlandıktan sonra değerleri değiştirilebilir. Değişkenlerin isminin değiştirilmesi “=” atama operatörü ile yapılır. Değer verilmesi işlemi herhangi bir portta değer verilmesi ya da porttan değer okunması işlemine çok benzerdir. Değişkene değer verileceği zaman değişken atama operatörünün soluna değer sağına yazılır. Değişkenin değeri okunacağı zaman değişkenin kendisi atama operatörünün sağına, değerin aktarılacağı yer ise atama operatörünün soluna yazılır.

**Örnek Uygulama:**

int Yakit=75; //İlk değeri 75 olan Yakit isimli int değişkeni oluşturuldu.

void main()

{

Yakit=74; //Yakit değişkeninin değeri 74 yapıldı

Yakit=73; // Yakit değişkeninin değeri 73 yapıldı

Yakit=72; //Yakit değişkeninin değeri 72 yapıldı

Portb=Yakit; //Yakit değişkeninin değeri B portuna atandı.

} //Programdan çıkıldı. B Portu çıkışı 72 olarak kalır.

Son durumda B portuna gönderilen sayı değeri 72’dir. Son durumda Yakit değişkeninin değeri de 72’dir.

**if Sorgusu**

Program akışı esnasında bazi değerlerin büyüklüğüne ya da küçüklüğüne göre farklı işlemler yaptırılması söz konusu olabilir. Bu durumda if sorgusu denilen kalıp kullanılır. İf sorgusu İngilizcedeki Eğerle başlayan cümlelere kalıp olarak benzer.

Eğer **yağmur yağıyor**sa **şemsiye kullan**

Eğer **hava soğuk**sa **palto giy**

Eğer **karnın aç**sa **yemek ye**

Gibi günlük hayattan örneklere benzer. Yukardaki örneklerde bir durum karşısında ne yapılacağı bildirilmiştir. Mesela ilk örnekte yağmur yağıyorsa şemsiye kullan, yağmıyorsa şemsiye kullanma denilmiştir. Yani şemsiyenin kullanılma durumu yağmurun yağmasına bağlanmıştır. İşte if sorgusuda tıpkı günlük hayattaki bu cümlelere benzer. İf sorguları da tıpkı günlük hayattaki örnekler gibi 2 kısımdan oluşur. Birtanesi yapılacak iş bir taneside bu yapılacak işin ne zaman yapılacağını bildiren önerme kısmıdır. Bildiğiniz üzere önerme; doğruluğu ya da yanlışlığı ispatlanmamış durumdur. Önerme doğruysa 1 değerini yanlışsa 0 değerini dönderir. If sorgusuna uyarlarsak; eğer önerme doğruysa , yapılacak iş yapılır; yanlışsa işlem yapılmadan dalgalı parantezler atlanır.

if(önerme) //önerme doğruysa dalgalı parantez içi yapılır yoksa yapılmaz.

{

Yapılacak iş

}

Örnek:

int yakit=75; //ilk değeri 75 olan yakit değişkeni tanımlanıyor

Void main()

{

Trisb=0; // B portu çıkış olarak ayarlanıyor

if(yakit>50) //Yakıt değişkeninin değeri 50’den büyük olduğu için önerme sonucu 1

{

Portb=255; // önerme sonucu 1 olduğu durumda B portuna 255 değeri gönderiliyor.

}

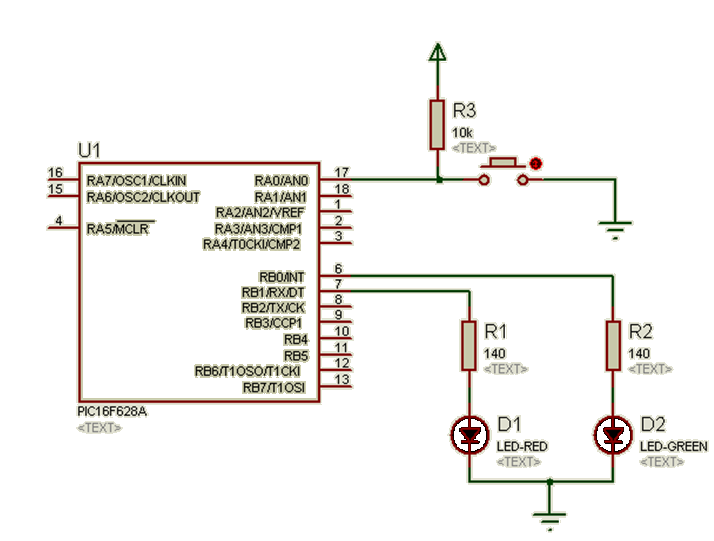
}

Yukardaki örnekte yakıt değişkeninin değerinin 50’den büyük olması durumuna göre B portuna 255 sayısı gönderilmiştir. Burada yakıt değişkeninin 50’den küçük olduğu durumlarda if sorgusunun dalgalı parantezleri içindeki komutlar işlenmez. Yalnızca yakıt değişkeni 50’den büyükse if sorgusunun önerme kısmının değeri 1 olur. Yani yakit değişkeninin değeri 43 olsaydı B portuna 255 sayısının gönderildiği satırlar işlenmeyecekti.

**Örnek Uygulama:**

Ra0 bacağına bağlı bir butona basıldığında RB0 a bağlı yeşil bir lambayı yakan, Ra0 bacağına bağlı butona basılmadığında RB1’e bağlı kırmızı lambayı yakan programı yazın.

Deney devresi:



Devrenin üst kısmında direnç ve butondan oluşan bağlantı şekline Pullup bağlantı şekli denir bu durumda butona basılmadığı durumda 10k’lık direnç üzerinde 5V Ra0 ucuna verilir. Yani Ra0’ın değeri dijital 1 olur. Butona basıldığında ise Ra0’a 0 Volt uygulanır yani. Ra0’ın değeri dijital 0 olur. Örnek uygulamamızda butona basılma ve basılmama durumuna göre B portuna bağlı yeşil ya da kırmızı renkli ledleri yakmamız gerekmektedir.

Program kodu:

void main()

{

Trisa=0b00000001; //Ra0 giriş olarak ayarlandı

Trisb=0b00000000; //RB0 ve RB1 çıkış olarak ayarlandı

if(porta.b0==1) //Ra0’ın değeri 1 mi değilmi kontrolü

{

Portb=0b00000010; //if önermesi sonucu 1 ise yapılacak işlem

}

if(porta.b0==0) // Ra0’ın değeri 0 mı değilmi kontrolü

{

Portb=0b00000001; //if önermesi sonucu 0 ise yapılacak işlem

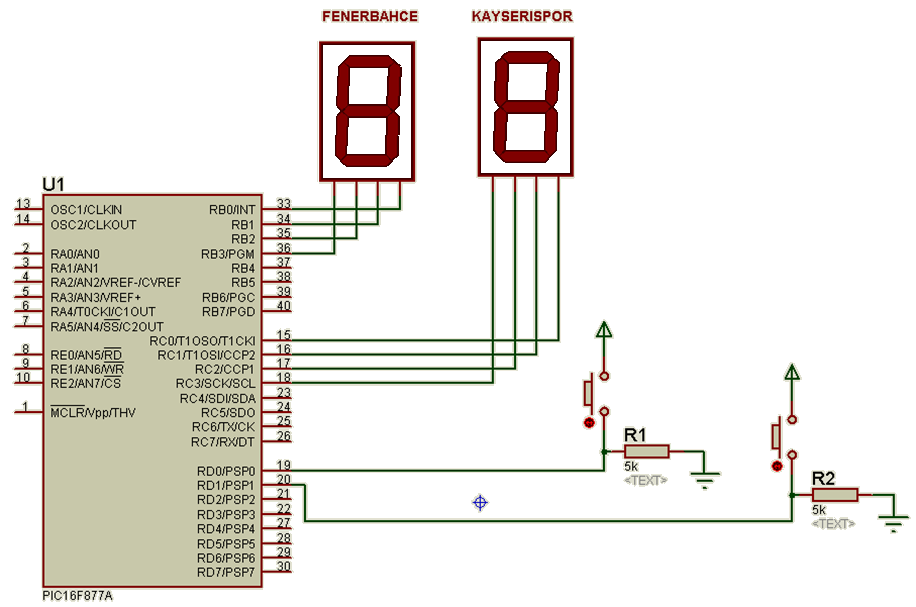
}

}

Not: bu çalışmada anlık değer değişimi kontrolü yapılmamaktadır. Yani Ra0’a bağlı butonun durumu sürekli olarak kontrol edilmez. Devreye enerji verildiği anda butonun durumuna göre program bir defa işletilir ve o duruma göre lambalar yanar. Ra0’ın durumu sürekli olarak kontrol edilmek istendiğinde if sorguları sonsuz bir while döngüsü içine yazılır. Bu sayede sürekli olarak butonun durumuna göre lambalar yakılır.

16F877A’lı Örnek Skorbord Uygulaması:

2 buton ve iki adet 7 segment BCD kullanarak skorbord yapınız.



Not: ilk durumda karşılaşma başladığında gösterge 0 – 0’ı gösterecek Ra0’a bağlı butona 1 saniye basıldığında soldaki takımın; Ra1’e bağlı butona 1 saniye basıldığında da sağdaki takımın skoru bir puan artırılacak.

**Program Kodları:**

int takim1=0; //ilk takımın skorunu tutmak için oluşturulan değişken

int takim2=0; //ikinci takımın skorunu tutmak için oluşturulan değişken

void main()

{

trisd=0b00000011; //RD0 ve RD1 giriş olarak ayarlanıyor

trisb=0; //B portu çıkış olarak ayarlanıyor

trisc=0; //C portu çıkış olarak ayarlanıyor

portb=0; //B ve C portu değerleri sıfırlanıyor

portc=0;

for(;;) // Sonsuz döngü

{

if(portd.b0==1) //RD0a bağlı butona basılı olma durumu sorgulanıyor

{

takim1=takim1+1; //RB0’a bağlı butona basılmışsa takim1 değişkeninin değeri bir artırılıyor

}

if(portd.b1==1) //RD1e bağlı butona basılı olma durumu sorgulanıyor

{

takim2=takim2+1; //RB1’a bağlı butona basılmışsa takim2 değişkeninin değeri bir artırılıyor

}

portb=takim1; //B portuna bağlı display’e takim1 değişkeninin değeri gönderiliyor

portc=takim2; //C portuna bağlı display’e takim2 değişkeninin değeri gönderiliyor

delay\_ms(1000); //Bir saniye bekleme. Butona basılı tutulduğunda skor üçer beşer artmasın diye bekletiliyor

}

}

**Not:** İf sorgusu kullanılırken sadece iki durum varsa ve her iki durum için farklı işlem yapılması gerekiyorsa **else(yoksa)** cümlesi if sorgusundan sonra eklenir. Uygulanışı aşağıdaki gibidir.

Aşağıdaki kodların çalışma şekli: RB0, 1’e eşitse if’ten sonraki dalgalı parantezlerin arasını işle, Yoksa else’ten sonraki dalgalı parantezlerin arasını işler.

Daha önce verdiğimiz örnekleri biraz değiştirirsek.

**Yağmur yağıyorsa** şemsiye al, **yoksa** alma.

**Hava soğuksa** palto giy, **yoksa** giyme.

if(portb.rb0==1)

{

//RB0 bacağının değeri 1 ise işlenecek program kodları

}

else

{

//RB0 bacağının değeri 1 değilse işlenecek program kodları

}

**Ödev:** Deneme devresi 1’de butona basılı olduğu durumda tümledlerin yanmasını, butona basılı değilken tüm ledlerin sönmesini sağlayan programı if ve else kullanarak yazın.

**For Döngüsü**

Bilgisayar programlarında bazı işlemlerin belli sayıda yapılması gereklidir böyle durumlarda For döngüsü kullanılır. For döngüsünün while döngüsünden farkı , döngünün bir değişkene otomatik olarak bağlıdır. For döngüsünde bir değişken, bu değişkenin durumuna bağlı olan bir şart, döngü her işlendiğinde bu değişkene yapılacak işlem ve yapılacak işlem bulunur. For döngüsünde kullanılacak değişkenin en temel mikroC fonksiyonundan önce diğer değişkenler gibi tanımlanması gerekmektedir. Kalıp aşağıdaki gibidir.

int sayac;

void main()

{

for(sayac=0;sayac<100;sayac=sayac+1) //0-100 arasında 100 defa parantez içindeki kodlar işlenecek

{

Portb=sayac; //For döngüsünde yapılacak iş

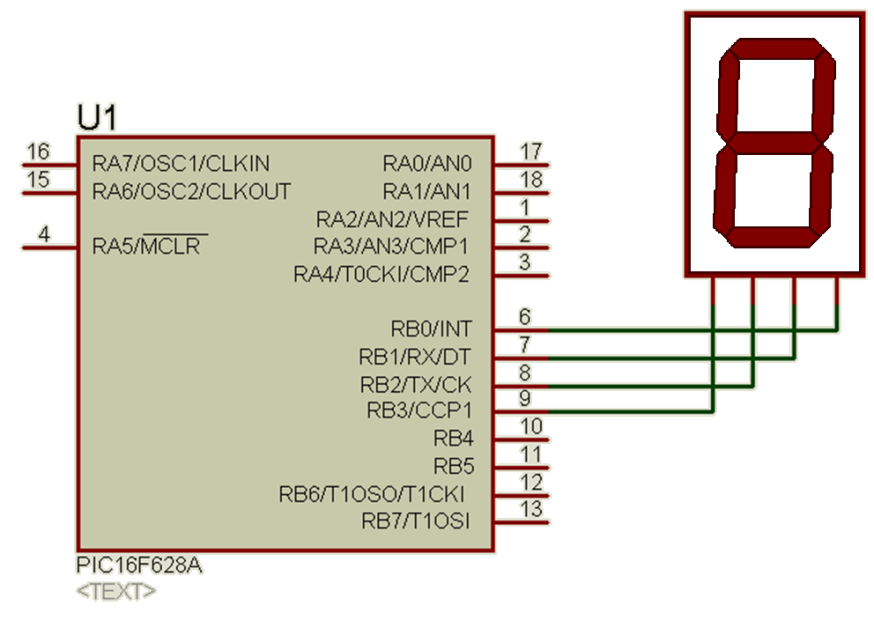
}

}

Burada sayac isimli değişkenin 0’dan 100’e kadar olan her bir değeri için for döngüsünün dalgalı parantezi içindeki komut satırları işlenecektir. Yani dalgalı parantez içindeki komutların her işlenişinde sayac değeri bir artırılacak. Bu işlem sayac değeri 100’den büyük olana kadar yani for şartı sağlanmayana kadar devam edeceketir.

**Not:** For döngüsünde normal parantezin içine sadece 2 noktalı virgül konursa For döngüsü sonsuz döngü olur.

**Örnek Uygulama:** 0- 9 arası sayıcı.



**Program Kodları:**

int sayac; //sayac değişkeni oluşturuluyor

void main() {

trisb=0; // B portu çıkış olarak ayarlanıyor

portb=0; //B portu sıfırlanıyor

for(sayac=0;sayac<9;sayac=sayac+1) //Döngü her işlendiğinde sayac değeri bir artırılyor

{

Delay\_ms(1000); //bir saniye bekle

portb=portb+1; //B portunun değerini bir artırarak B portuna gönder

}

}

**Not:** Burada sayac değişkeninin 0 ile 9 arasındaki her değeri için dalgalı parantez içindeki komutlar işlenmektedir. Yani 0-9 arasında dalgalı parantezler arasındaki işlem 9 defa yapılmaktadır.

**Not:** Herhangi bir döngüden çıkılmak istendiğinde break komutu kullanılır. Break komutu devam eden döngüden çıkılmasını sağlar.

Örnek:

int i=0;

void main()

{

trisb=0;

for(i=0;i<100;i=i+1)

{

Portb=i;

if(i==50)

{

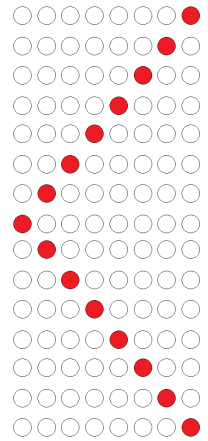
Break;

}

}

}

Açıklama: For döngüsündeki i değişkeninin değeri 50 olunca if sorgusu çalıştırılır ve for döngüsünden çıkılır. Yani işlem 100 defa yapılacakken break komutu nedeniyle 50 defa yapıldıktan sonra döngü dışına çıkılır. Aynı işlem sonsuz while döngüsünden çıkmak için de kullanılabilir.

**Ödev:** İsis temel deneme devresini kullanarak yandaki karaşimşek animasyonuna göre sağa, sola kaydırma operatörü ve for döngüsü kullanarak programlayın. Yapılan işlemi sonsuz(for) döngü kullanarak sürekli hale getirin.

**Artırma (++) ve Azaltma (--) Operatörü**

Bir değişkenin değeri bir artırılacağı ya da bir azaltılacağı zaman kullanılır.

Degisken1++; //Degisken1 isimli değişken değeri bir artırılmıştır.

Degisken1--; //Değişken1 isimli değişken değeri bir azaltılmıştır.

Portb++; //B portuna gönderilen sayı değeri bir artırılmıştır.

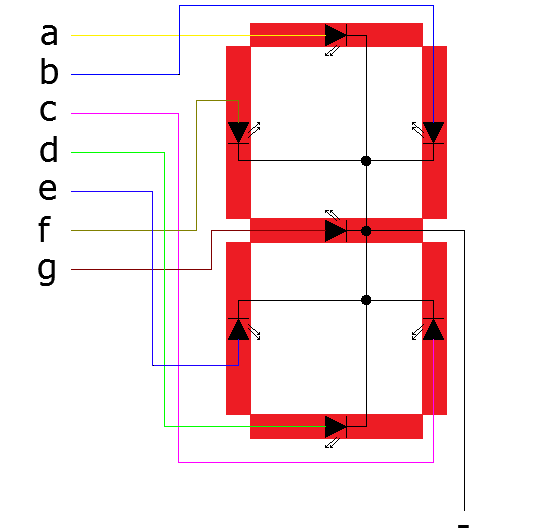
Aslında Degisken1++; ile Degisken1=Degisken1+1; ifadesi birbirine eşittir.

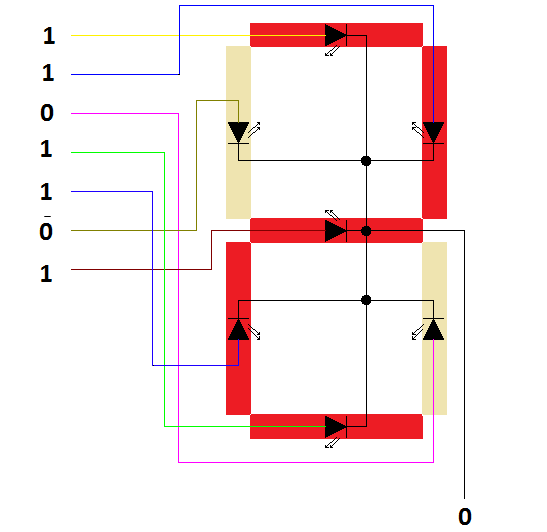
Degisken1--; ile Degisken1=Degisken1-1; ifadesi birbirine eşittir.

Artırma ve azaltma operatörleri for döngülerinde sıklıkla kullanılır.

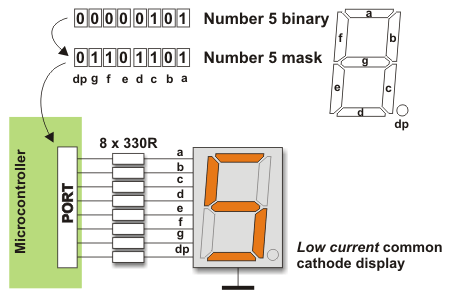
Ödev: 0-9 arası sayıcı devresini 0- 9, 9-0 arasında sürekli olarak sayacak şekilde programlayın.

**7 Segment Displayler**

İsmindende anlaşıldığı üzere 7 led’in rakam oluşturacak şekilde bağlanması sonucu oluşturulur. Kullanış şekli 7 adet normal led kullanımına benzer. Displayler çalışma şekline göre Ortak Katot ve Ortak Anod olmak üzere ikiye ayrılır. Biz çalışmalarımızda daha kolay anlaşılır olduğu için ortak katot olanları kullanacağız. Ortak Katot displaylerde, 7 Ledin eksi uçları birleştirilmiştir. Ledlerin artı uçları display’in dışına çıkarılmıştır. Yani 7 segment display’de en az 8 bacak olmalıdır. Displaylerde genellikle 8 bacaktan daha fazla bacak bulunur. Bunun nedeni eksi ucun birden fazla uçla çıkarılmasıdır. Günlük hayatta 7 segment display kullanırken eksi ucu bağlarken, öncelikle eksi uçları tespit etmek ve bizim için kullanımı en kolay olanı kullanmamız gereklidir. Displaylerin artı uçları ise yukardaki şekilde gözüktüğü gibi her ledin artı ucundan alınarak display dışına çıkarılmıştır.

Displaylerde bulunan her çizgiye bir segment denir. Segmentler en üstteki segment “a” harfi olmak üzere sırasıyla a,b,c,d,e,f,g şeklinde harflerle isimlendirilir. Display’in mikrodenetleyicinin B portuna örnek bağlantı şekli aşağıda gösterilmiştir. Displaylerin çalışması yakılacak segmentlerin artı uçlarına gerilim uygulanması ile olur. Rakam yazdırabilmek için o rakamın yazılması için yakılacak segment bacaklarına gerilim uygulanmalıdır. Örneğin 5 rakamını yazdırabilmek için a,f,g,c ve d uçlarına gerilim vermemiz gerekmektedir. Yani bu uçlara bağlı mikrodenetleyicinin B portu uçları olan rb0, rb2, rb3, rb5, rb6 uçlarını 1 diğer uçları 0 yapmamız gerekmektedir. Kod olarak “Portb=0b00000111;” program içinde kullandığımız zaman display’e 5 sayısını göndermiş oluruz. Her rakam için bu işlemin yapılması işlemine 7 segment kodlama denir.

Hangi rakamız yazdırmak için hangi sayının B portuna gönderilmesi gerektiğini her seferinde hesaplamamak için MikroC’de Tools menüsünden “Seven Segment Editör” kullanabiliriz.

****

**Switch Cümlesi**

Bir değişkenin farklı değerlerine göre farklı işlemlerin yapılacağı durumlarda kullanılan programlama kalıbıdır. Switch cümlesi bir değişken tarafından yönetilir. Değişkenin aldığı her değer için farklı durumlar oluşur. Her durum için farklı komutlar çalıştırılır. Örneğin 7 Segment bir displaye sayı gönderileceği zaman değişkenin her değerinin on tabanındaki sayıdan 7 segment kodu çözülerek displaye gönderilmesi gerekir böyle durumlarda switch cümlesi kullanılır. Değişkenin alabileceği tüm değerler düşünülmeli ve her değer için çalıştırılacak komutlar yazılmaldır.

**Örnek:**

int sayi; //degisken isimli değişken tanımlanıyor

void main()

{

switch(sayi) //switch cümlesini yönetecek değişken sayi olarak belirtiliyor

{

Case 1:

Portb=1; //sayi degiskeninin 1 olmasi durumunda yapılacak işlem

Break;

Case 2:

Portb=2; //sayi degiskeninin 2olmasi durumunda yapılacak işlem

Break;

Case 3:

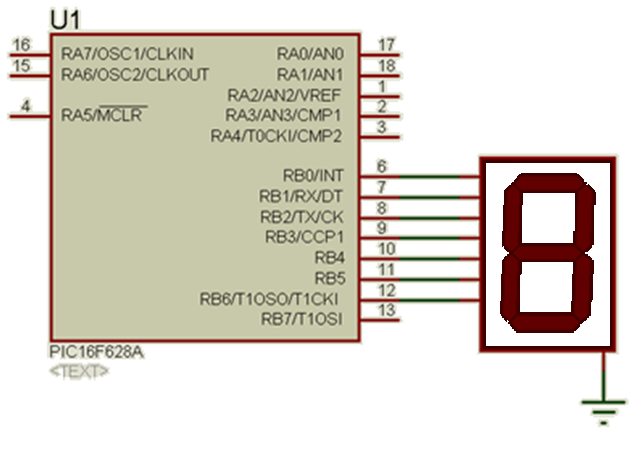
Portb=3; //sayi degiskeninin 3 olmasi durumunda yapılacak işlem

Break;

}

}

**Örnek Program:** 0-9 arasında sayan program devresi aşağıdadır. Devre10 tabanında yazılmış sayıdan 7 Segment display için kod dönüştürmesi yapmaktadır.



Program kodları:

int sayac; // sayac değişkeni tanımlanıyor

void main()

{

trisb=0; // B portu çıkış olarak ayarlanıyor

portb=0; //B portunun değeri sıfırlanıyor

for(sayac=0;sayac<10;sayac=sayac+1)

{ //Sayac degiskeninin 0’dan ona kadar her degeri için islenecek kodlar için for döngüsü

Delay\_ms(1000); //Bir saniye bekleme

switch(sayac) //switch cümlesi sayac degiskeni ile yönetiliyor

{

case 0: //sayac degiskeni 0 olunca işlenecek satır

Portb=63; //0 sayısının 7 segment karşılığı olan sayı

break;

case 1: //sayac degiskeni 1 olunca işlenecek satır

Portb=48; //1 sayısının 7 segment karşılığı olan sayı

break;

case 2: //sayac degiskeni 2 olunca işlenecek satır

Portb=91; //2 sayısının 7 segment karşılığı olan sayı

break;

case 3: //sayac degiskeni 3 olunca işlenecek satır

Portb=79; //3 sayısının 7 segment karşılığı olan sayı

break;

case 4: //sayac degiskeni 4 olunca işlenecek satır

Portb=102; //4 sayısının 7 segment karşılığı olan sayı

break;

case 5: //sayac degiskeni 5 olunca işlenecek satır

Portb=109; //5 sayısının 7 segment karşılığı olan sayı

break;

case 6: //sayac degiskeni 6 olunca işlenecek satır

Portb=125; //6 sayısının 7 segment karşılığı olan sayı

break;

case 7: //sayac degiskeni 7 olunca işlenecek satır

Portb=7; //7 sayısının 7 segment karşılığı olan sayı

break;

case 8: //sayac degiskeni 8 olunca işlenecek satır

Portb=127; //8 sayısının 7 segment karşılığı olan sayı

break;

case 9: //sayac degiskeni 9 olunca işlenecek satır

Portb=111; //9 sayısının 7 segment karşılığı olan sayı

break;

}

}

}

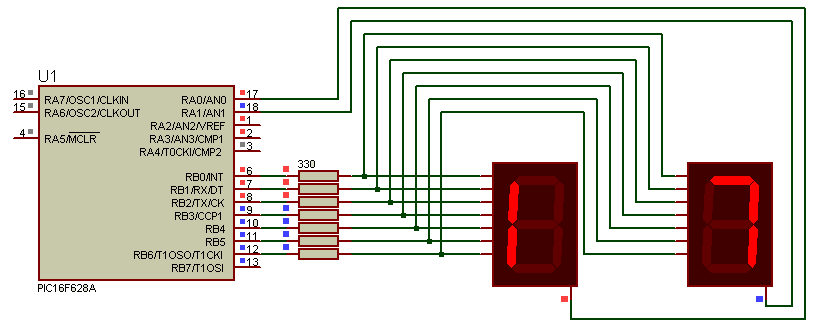
**Not:** for döngüsünde sayac degiskeninin 0’dan 10’a kadar olan tüm değerleri için işlem yapılacağı için for döngüsünün her farklı işlenmesinde switch cümlesine gelen sayac değeri de farklı olacaktır. Sayac değişkeninin her farklı değeri için de farklı kodlar işlenecek. Yani program 7 segment kodlayıcı gibi çalışacaktır.

**Ödev:** Yukardaki programı **sürekli olarak** 0-9, 9-0 arasında yukarı aşağı sayacak şekilde değiştirin.

**Ödev:** Bir display ve bir butonlu dijital zar uygulamasını gerçekleştirin.

**İki ya da daha fazla haneli sayıların 7 Segment Display’de gösterilmesi**

Bir display’e bir rakam yazılabilmesi daha önce en az 8 bacağın kullanılması gerektiğini söylemiştik. Bu mantığa göre iki display kullanabilmek için 16, 4 display kullanabilmek için 32 adet bacak kullanmamız gerekir ki bu basit bir saat yaptığımızı düşünürsek çok büyük 40 bacaklı mikrodenetleyiciler kullanmamızı gerektirir. Günlük hayatta birden fazla display kullanılan uygulamalarda bu işlem bu şekilde yapılmamaktadır. Displaylerin anod uçları paralel olarak bağlanır (7 uç) ve display’e gönderilecek sayılar teker teker sırayla gönderilir. Bu yapılan işleme **tarama işlemi** denir. Yani 17 sayısı 2 display’e yazdırılacaksa önce soldaki displaye 1 sayısı gönderilir, bu sırada sağdaki display sönük olarak tutulur. Daha sonra 7 sayısı sağdaki displaye gönderilir ve soldaki display sönük olarak tutulur. Bu işlem sürekli olarak tekrarlandığında iki display’de 17 sayısı sürekli olarak yanıyormuş gibi gözükür. Peki sayı gönderdiğimiz display dışındaki displayi söndürme işlemini nasıl gerçekleştirebiliriz. Bu işlem ortak katot ucu kullanılarak yapılır. Her displayin ortak katot ucu A portunun kullanılmayan bir ucuna bağlanır. Örneğin aşağıdaki devrede soldaki displayin ortak ucu RA0’a, sağdaki displayin ortak ucu RA1 ucuna bağlanmıştır. 1 sayısını B portundan gönderdiğimizde soldaki displayin çalışmasını sağdaki displayin sönmesini istiyoruz, bu durumda ilk displayi yakmak için RA0 ucunu 0, ikinci displayi söndürmek için RA1 ucunu 1 yapmamız gerekiyor. 7 rakamını sağdaki displaye yazdırmak istediğimiz durumda da B portuna 7 rakamını gönderip sağdaki displayi yakmak için RA1 ucunadan 0, RA0 ucundan 1 göndermemiz gerekiyor. Yani yakmak istediğimiz displayin ortak ucuna dijital 0; söndürmek istediğimiz displayin ortak ucuna da dijital 1 uyguluyoruz.



Örnek Uygulama: 17 rakamını yukardaki devrede gösterecek program kodlarını yazın.

void main()

{

trisa=0b00000000;

trisb=0b00000000;

for(;;)

{

porta=0b11111110;

portb=48;

delay\_ms(20);

porta=0b11111101;

portb=7;

delay\_ms(20);

}

}

**Ödev:** 4 display kullanarak 1453 sayısını yazdıran programı ve devre simülasyonununu oluşturun

**Ödev:** İki display ve bir butonlu dijital zar uygulamasını gerçekleştirin. Butona basıldığında iki displayde iki zardan gelecek sonuç gözükmelidir.

**Ödev:** 0-99 arası sayıcı uygulamasını gerçekleştirin.

**MicroC’de bellek türleri**

1. **ROM:** Sadece okunabilen bellek türüdür. Yazdığımız hex dosyası bu belleğe kaydedilir ve buradan çalıştırılır. Bu belleği biz değer kaydetmek amacıyla kullanamayız.
2. **RAM:** Mikro denetleyicinin kendisinin kullandığı bellek türüdür. Bu belleğin özelliği elektrik kesintilerinden tamamıyla silinir. Örneğin PORTB’ye ve TRISB’ye gönderilen değer RAM’de kaydedilir.
3. **EEPROM:** Elektrik kesintilerinden etkilenmeyen bellek türüdür. Elektrik kesintisinde kaybolmamasını istediğimiz bilgileri **program esnasında** bu belleğe kaydedebiliriz. EEPROM kapasitesi her mikrodenetleyici için farklıdır. 16f628a entegresinin EEPROM kapasitesi 128 byte’tır.

**EEPROM’a Değer Kaydedilmesi**

EEPROM\_WRITE fonksiyonu ile istenilen adrese istenilen bilgi kaydedilir. Fonksiyon kalıbı aşağıdaki gibidir.

EEPROM\_WRITE(adres,deger);

Örnek:

EEPROM\_WRITE(0x02,128); //EEPROM’un 0x02 numaralı adresine 128 sayısı kaydedilmiştir.

**EEPROM’dan Değer Okunması**

EEPROM\_READ fonksiyonu kullanılarak istenilen EEPROM adresindeki değerler okunabilir. Fonksiyonun kalıbı aşağıdaki gibidir.

EEPROM\_READ(adres);

Örnek:

Portb=EEPROM\_READ(0x02); //EEPROM’un 0x02 numaralı adresindeki değeri B portuna gönder.

Örnek Uygulama : Deneme devresi 1 kullanılarak bir sayının önce EEPROM’a yazıldığı sonra yazıldığı EEPROM adresinden okunup B portuna gönderildiği program.

void main()

{

trisb=0;

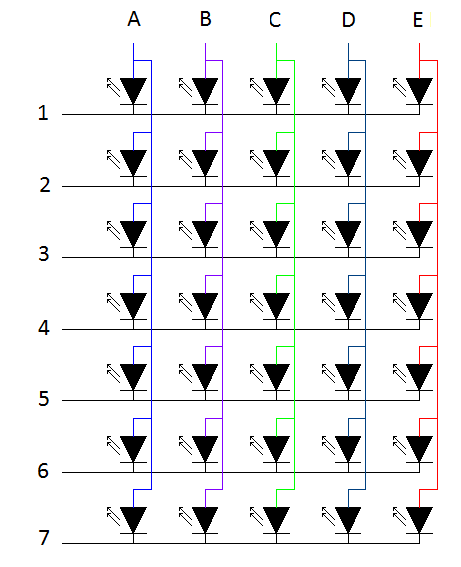
portb=0;

EEPROM\_write(1,255); //1 nolu EEPROM adresine 255 sayısı kaydediliyor.

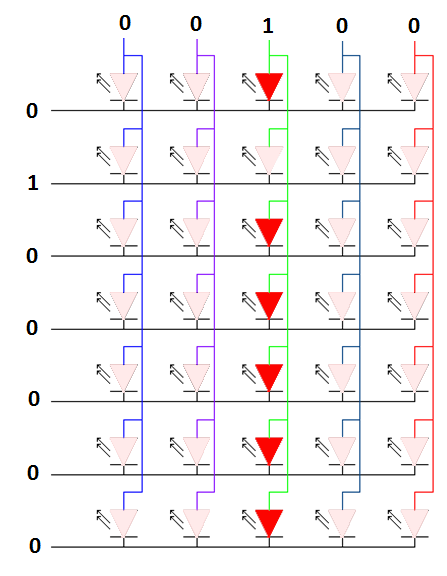
portb=EEPROM\_READ(1) ; //1 nolu EEPROM adresindeki değer okunup B portuna gönderiliyor

}

**Dot Matris Kullanımı**

Dot Matrix(Nokta Matris) belli sayıda led’den oluşan ve çoğunlukla ışıklı yazı yazmak için kullanılan devre elemanıdır. 5x7 ve 8x8 olmak üzere iki çeşit dot matris vardır. 5x7 dot matris demek toplamda 5 sütün ve 7 satır toplam 35 led’den oluşan elemandır. Dot matrisler 7 Segment displaylere benzer prensipte çalıştırılır. Dot matriste bulunan belli noktaları yakabilmek için; dot matris uçlarına gerekli gerilimleri vermek gerekmektedir. Bir satırı aktive etmek için o satıra 0V, bir sütunu aktive etmek içinde o sütüna 2V gerilim vermemiz gerekmektedir. Örneğin C sütunundan en üstteki led’i yakmak istediğimizde C bacağına 2V diğer bacaklara 0V, 1 numaralı satıra 0V diğer satırlara 2V uygulamamız gerekir. Bu durumda sadece C1 ledi yanacaktır.

C sütünunda bulunan ledlerin hepsini yakmak istesek, Yani Büyük I harfi yazdırmak için C bacağına 2V diğer bacaklara 0V ve tüm satırlarada 0V uygulamamız gerekir.

Dot matrislerde bulunan ledlerin çalışma gerimleri çoğunlukla 2 V olmakla birlikte değişik gerilim değerlerinde dot matrisler vardır. Bilindiği gibi PIC portundan alınan çıkışlar 5 V ve 0V’tur bu durumda dotmatrisin sütunları ile PIC bacakları arasına uygun direnç bağlanmalıdır(2V için 150-300 Ω’luk dirençler kullanılabilir).

Dot matrisler PIC entegresine bağlanırken satırlar ve sütünlar ayrı porta bağlanır. Aşağıdaki şekilde 5x7 Dot matrisin 16f877 entegresine bağlantı şekli gösterilmiştir.

Dot matrislerle harf yazabilmek için yazılacak harflerin sütunlara bölünerek, her sütunun ayrı ayrı gönderilmesi gerekmektedir. Örneğin tek sütün’da yazılabilecek “İ” harfini Dot Matrisimizde yazabilmek için Dot matris bacaklarına aşağıdaki şekildeki gibi 1 ve 0’ları uygulamamız gerekmektedir.

16F628A ile yapılmış aşağıdaki devrede 5x7 dot matrisin bağlantı şekli gösterilmiştir. Bu bağlantı 1 sütüna gönderilecek bilgi B portundan o sütunun nerde gösterileceği de A portundan kontrol edilmektedir. İ harfinin dot matriste gösterilebilmesi için A portunun değeri “0b00000100”, B portunun değeri ise “0b10000010” olmalıdır. Hangi sütunun yakılacağı A portundan o sütuna 1 göndererek, o sütunda hangi ledlerin yakılacağı ise yakılacak ledlere 0 gönerilerek sağlanmaktadır.

**Örnek Uygulama:** Aşağıdaki devrede için “İ” harfini yazdıran programı yazın.

void main()

{

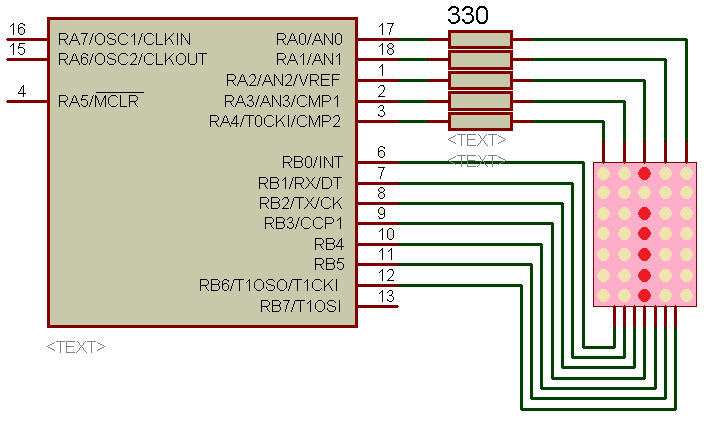
trisb=0;

trisa=0;

porta=0b00000100; //ortadaki sütün aktif hale getiriliryor

portb=0b10000010; //İ harfi için gerekli olan dikey ledler tek tek yakılıyor. 1’ler

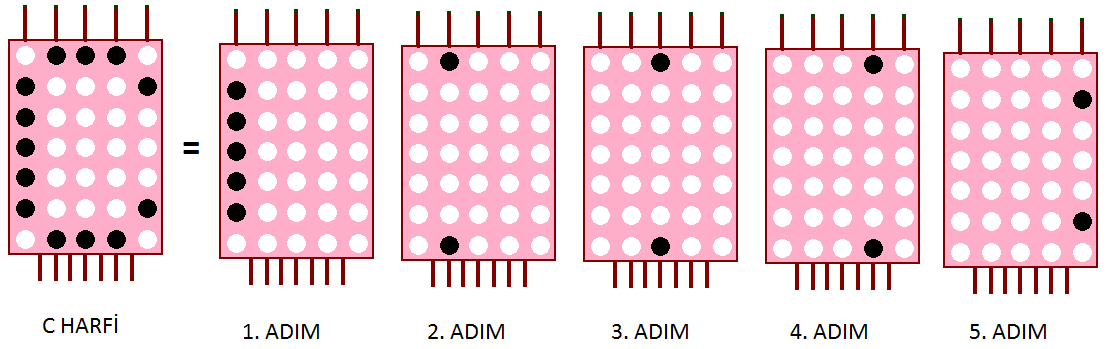
} //sönük ledleri, 0’lar yanık ledleri gösteriyor.



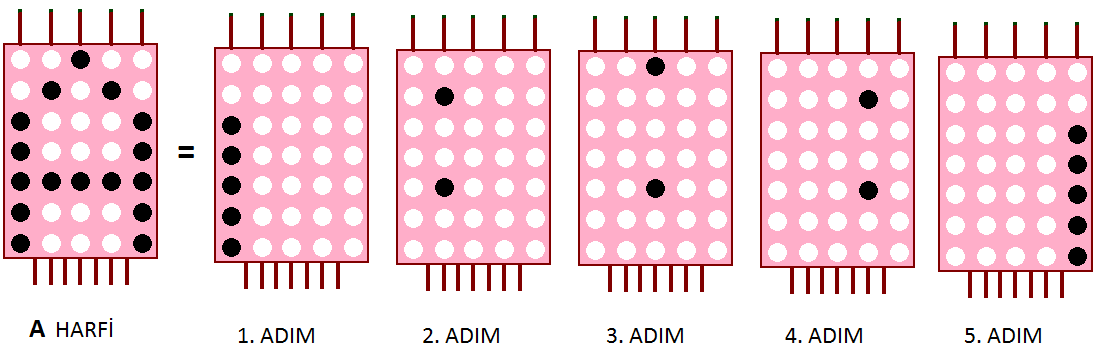
**Not:** A portu ve Dotmatris arasına bağlanan dirençler dotmatrise fazla gerilim uygulanmasını engellemek için bağlanmıştır.

Bir harfin tek sütundan oluştuğu durumlarda dot matrislere harfler bu şekilde kısa kodlarla gönderilebilir. Ancak bir harfin birden fazla sütunda gösterileceği durumlarda tarama mantığı kullanılması gerekmektedir. Tarama mantığı kullanılırken önce harf sütünlara bölünür, sonra bu sütünlar tek tek dot matrise tek tek sırayla gönderilir. Bu işlem hızlı olarak yapıldığında harf sürekli olarak yanıyormuş gibi gözükür.

Örnek 1: C harfinin dot matriste gösterilmesi.



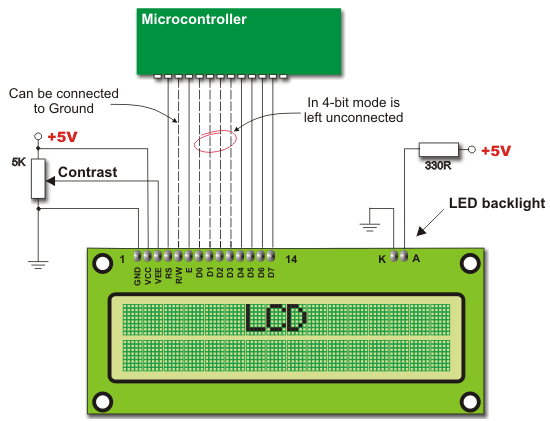
Örnek 2: A harfinin dot matriste gösterilmesi



Örneklerde de görüldüğü gibi birden fazla sütundan oluşan harflerin yazılmasında harflerin her sütünu sırayla, hızlı bir şekilde tek tek dot matrise gönderilir. 1-5 arası adımlar hızlı bir çekilde sürekli olarak tekrarlanır, bu durumda harf sürekli olarak yanıyormuş gibi gözükür.

Örnek Uygulama: Yukardaki devrede A harfini yazdıran programı hazırlayın.

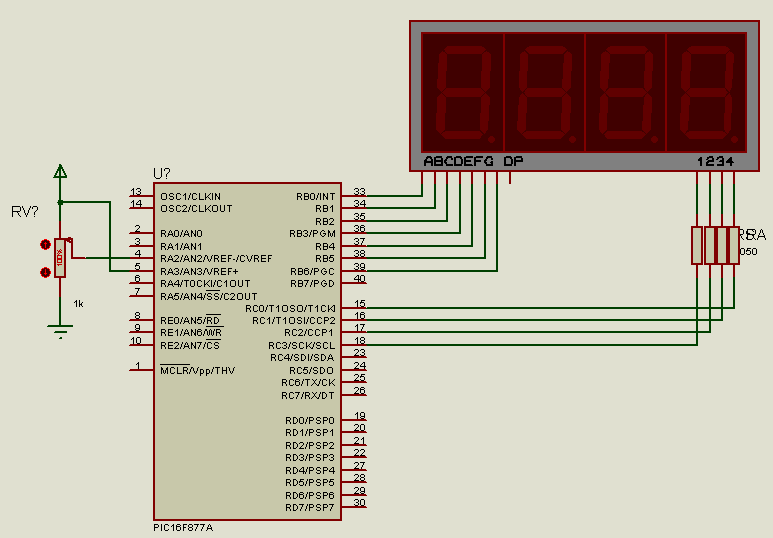
**Karakter LCD Display Kullanımı**



**16f877a ile Analog Dijital Converter Kullanımı**

Analog girişine uygulanan gerilim değerini sayısal değerlere dönüştüren devrelere Analog Dijital Converter devreleri denir. ADC(Analog Dijital Converter) özelliği bulunan mikrodenetleyiciler kullanılarak gerilim değerleri dijitale çevrilebilir. Aşağıdaki devrede gerilim bölücü dirençle RA2 ucuna uygulanan gerilim değeri ayarlanabilmektedir. ADC devrelerinin kullanımında maksimum gerilim değerini belirten Vref gerilimi ucu da bağlanmalıdır. Bu devrede Vref ucu +5 V olan kaynağa bağlanmıştır. Yani bu durumda ra2 ucuna 5V uyguladığımız zaman displaylerimizde 1024=(11111111)2 değeri gösterilecek; ra2 ucuna 0V uyguladığımız zaman 0=(0000000000)2 değeri displayden gözükecektir. Bunların arasındaki gerilim değerleri içinde 0’la 1024 arası hangi değere denk geliyorsa o değer gösterilecektir.

\*\* Dikkat edilmesi gereken bir husus VREF gerilimini kesinlikle 5V ‘un üzerinde gerilim uygulamamaktır. Aksi taktirde mikrodenetleyicimiz yanar.

****

Yukardaki devre için örnek program kodları aşağıda verilmiştir(Simülasyon devresi ve program <http://temelelektronik.info/devre.aspx?devre=496> adresinden indirilebilir).

int adc1=0;

void bportuna(int i) //B portuna ilgili sayılar gönderiliyor

{

switch(i)

{

case 0:portd=63 ; portb=63 ;

break;

case 1: portd= 6; portb= 6;

break;

case 2: portd= 91; portb= 91;

break;

case 3: portd= 79; portb= 79;

break;

case 4: portd= 102; portb= 102;

break;

case 5: portd=109; portb=109;

break;

case 6: portd= 125; portb= 125;

break;

case 7: portd= 7; portb= 7;

break;

case 8: portd= 127; portb= 127;

break;

case 9: portd=111; portb=111;

break;

case 10: portd=191; portb=191;

break;

}

}

void main()

{

AdresL = 0x0C; // AN2 ve AN3 analog giriş olarak ayarlandı

TRISA = 0xFF; // A portu giriş olarak ayarlandı

AdresH = 0; // Tüm uçlar Dijital olarak ayarlarlandı

ADCON1.F0 = 1 ; // RA3 ucu Vref olarak ayarlandı

trisc=0;

trisb=0;

trisd=0;

portc=255;

for(;;)

{

adc1= ADC\_Read(2);

bportuna(adc1%10); //birler basamağındaki sayı displaye gönderiliyor

portc=0b00001110;

delay\_ms(30);

bportuna((adc1%100)/10); //onlar basamağındaki sayı displaye gönderiliyor

portc=0b00001101;

delay\_ms(30);

bportuna(adc1/100); //yüzler basamağındaki sayı displaye gönderiliyor

portc=0b00001011;

delay\_ms(30);

bportuna(adc1/1000); //binler basamağındaki sayı displaye gönderiliyor

portc=0b00000111;

delay\_ms(30);

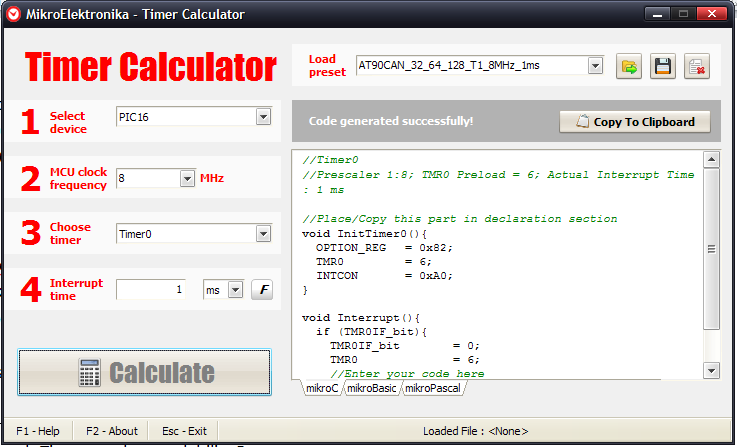
}

}

**Mikrodenetleyicilerde Timer Kullanımı**

Bir işin periyodik olarak yaptırılması için kullanılan mikrodenetleyici donanımına Timer(zamanlayıcı) denir. Örneğin bir saatin saniye sayacını her saniye bir artırmak istediğimizde bir saniyelik bir zamanlayıcıya ihtiyacımız vardır. Yani Timer kullanmamız gerekmektedir. Mikrodenetleyicilerde Timer kullanmak çok zahmetli bir iştir. Timer donanımının program yazarken ayarlanması gerekir.

Bu zahmetli işlemi kolaylaştıran MikroC programının “Timer Calculator” isimli yan programı vardır. Bu program kullanılarak Timer ayarları yapılabilir. Bu program <http://www.temelelektronik.info/download.aspx> adresinden indirilebilir.



Program indirildikten sonra “Select Device” kısmından PIC16 seçilir. Kullanılan Osilatör frekansı “MCU clock Frequency” kısmına girilir. Hangi timer kullanılacaksa o “Choose timer” kısmından seçilir. Daha sonra “Interrrupt time” kısmına ayarlanmak istenen timer frekansının çarpılarak bulunabileceği bir süre girilir. Örneğin 1 Saniye 1000x 1milisaniye işlemi yapılarak bulunabilir. Eğer timer süremizin 1 saniye olmasını istiyorsak Interrupt time kısmına 1 milisaniye girebiliriz.

Gerekli girişler yapıldıktan sonra “Calculate” butonuna basılır. Program tarafından üretilen kod MikroC programının başlangıcına yapıştırılır.

Pic16, 20Mhz, Timer1, 1ms seçenekleri ile üretilmiş örnek kod aşağıdadır.

**void** InitTimer1(){

T1CON = 0x01;

TMR1IF\_bit = 0;

TMR1H = 0xEC;

TMR1L = 0x78;

TMR1IE\_bit = 1;

INTCON = 0xC0;

}

**void** Interrupt(){

**if** (TMR1IF\_bit){

TMR1IF\_bit = 0;

TMR1H = 0xEC;

TMR1L = 0x78;

*//Enter your code here*

}

}

Burada InitTimer1() fonksiyonu entegre çalıştırıldığı anda çalıştırılmalıdır. Bu fonksiyon main() fonksiyonunun içine ilk satırlarda eklenmelidir.

Örnek:

**void** InitTimer1(){

T1CON = 0x01;

TMR1IF\_bit = 0;

TMR1H = 0xEC;

TMR1L = 0x78;

TMR1IE\_bit = 1;

INTCON = 0xC0;

}

**void** Interrupt(){

**if** (TMR1IF\_bit){

TMR1IF\_bit = 0;

TMR1H = 0xEC;

TMR1L = 0x78;

*//Enter your code here*

}

}

void main()

{

InitTimer1();

}

InitTimer1(); fonksiyonu main() fonksiyonu içine yazılmazsa Timer çalışmaya başlamaz. InitTimer1() fonksiyonunun sadece bir defa çalıştırılması yeterlidir.

Bu işlem yapıldıktan sonra Interrupt() fonksiyonu her bir milisaniyede çalıştırılacaktır. Bu aşamadan sonra Timer’ımızın süresi olan 1milisaniyeden istediğimiz süre olan 1 saniyeyi bulmamız gerekmektedir. 1 saniye 1milisaniyenin binde biridir. Yani bin defa timer’ımız Interrupt() fonksiyonunu çalıştırdığında bizim saniyemizi bir artırmamız gerekmektedir. Bu işlemi değişken kullanıp Interrupt() fonksiyonu her çalıştırıldığında değişkeni artırarak, değişkenin değeri 1000 olduğunda istediğimiz işi yaptırabiliriz. Yani 1000 x Interrupt()= 1 saniye.

Bu işlem için “sayac” ve “saniye” isminde iki adet değişken kullanalım. sayac değişkenini Interrupt() fonksiyonu her çalıştığında saniye değişkenini de sayaç değişkeninin değeri 1000 olduğunda artırabiliriz.

Bu ekleyeceğimiz kodları Interrupt() fonksiyonu içindeki “*//Enter your code here*” yazan yere eklememiz gerekiyor.

Bu işlemleri yaptığımızda programımız aşağıdaki gibi gözükecek.

int sayaç=0;

int saniye=0;

**void** InitTimer1(){

T1CON = 0x01;

TMR1IF\_bit = 0;

TMR1H = 0xEC;

TMR1L = 0x78;

TMR1IE\_bit = 1;

INTCON = 0xC0;

}

**void** Interrupt(){

**if** (TMR1IF\_bit){

TMR1IF\_bit = 0;

TMR1H = 0xEC;

TMR1L = 0x78;

sayac++;

if(sayac==1000)

{

saniye++;

sayac=0;

}

}

}

void main()

{

InitTimer1();

}

Bu işlemleri yaptıktan sonra saniye değişkenimizi programımızın istediğimiz yerinde rahatlıkla kullanabiliriz.

Timer kullanılarak yapılmış Dijital saat uygulamasının devresini ve programını <http://www.temelelektronik.info/devre.aspx?devre=581> adresinden bulabilirsiniz.