

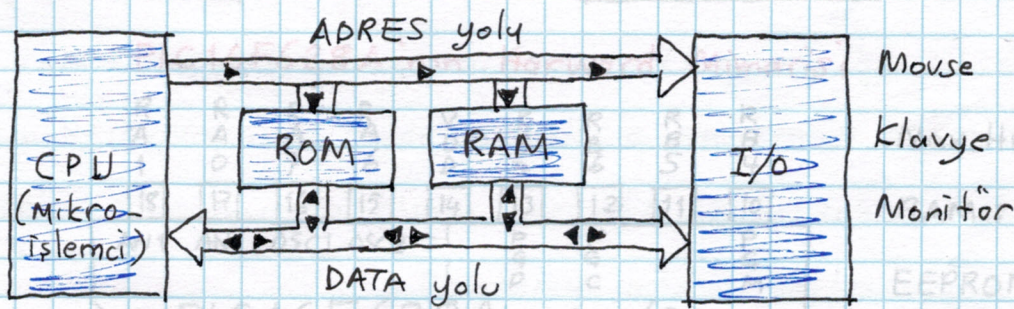
MİKRODENETLEYİCİLER (Mikro Kontrol Devreleri Persi)

MODÜL 1 : Mikrodenetleyici Programlama

1- Mikrodenetleyici ve Programı :

Mikroişlemci : Kullanıcılar tarafından programlanabilen, tek bir kılıf içerisinde (Entegre halde, gömülü) Merkezi İşlem Birimlerine (CPU) (Central Processing Unit) yada Mikroişlemci denir.

Mikroişlemciler (CPU) ile birlikte RAM (Random Access Memory) ve ROM (Read Only Memory) ile Giriş-Çıkış (I/O) birimleriyle Adres yolu ve Data yolu ile haberleşme yaparak çalışırlar. Aşağıdaki şekilde mikroişlemci (CPU) ile çalışan mikrobilgisayarın blok diyagramı verilmiştir.



Mikroişlemcinin iç ve dış birimlerle haberleşmesi ADRES ve DATA yolları aracılığıyla yapılır. ADRES yolu ROM, RAM belleklerdeki adreslere ve I/O Input/Output birimlerine ulaşmayı sağlar. DATA yolu ise mikroişlemci ile ROM, RAM belleklerle, I/O birimleri arasında bilgi paylaşımını sağlar.

Mikrokontrolcü, Mikrokontroller : Mikroişlemcinin (CPU) diğer yardımcı devrelerle birlikte tek bir kılıf içerisinde (ROM, RAM, Portlar) entegre edilmiş haline mikrokontrolcü yada mikrokontroller adı verilir.

Günümüzde PIC adı verilen (Peripheral Interface Controller) serisi mikrodnetleyiciler çeşitli kılıflar içerisinde üretilerek, farklı donanımsal özellikleri ve farklı pin yapıları ile piyasadan temin edilebilmektedir. Bizler bu derste hem kolay bulunabilmesi, hem ucuz oluşu nedeniyle PIC16F628A mikrokontroller entegresini kullanacağız. Bu entegre 8 bitlik bir işlemciye sahip olup, içerisinde hazır kullanılabilen 4MHz ve 32kHz'lik osilatörler vardır.

1.2. Mikrodenetleyicinin donanım özellikleri : (16F628A)

RISC Mimarisinin 16F628A'ya kazandırdığı özellikler şunlardır.

- 1- 35 komut ile kontrol etme.
- 2- Çalışma hızı max. 20 MHz (HS-OSC) içinde 4MHz ve 32KHz'lik OSC.
- 3- DC 200ns (nanosaniye) komut periyodu
- 4- 14 bit uzunlukta komutlar
- 5- 8 bit uzunluğunda veri yolu (Data)
- 6- 15 tane özel donanım Registeri (Donanım kullanım bayrakları)
- 7- 8 seviyeli derin donanım stackleri
- 8- Doğrudan - dolaylı ve izafi adreslenebilme
- 9- 1000 kere program yazıp - silme özelliğine sahip Flash bellek
- 10- Programı enerji kesildikten sonra 10 yıl saklayabilme.

CPU : İçerisinde PC - Program Counter (Program Sayıcı), ALU (Aritmetik Logic Unit) bulunan ve diğer birimlerle çalışmayı organize eden ve flash bellekteki programı çalıştıran birimdir. (Mikrodenetleyicinin beynidir.)

Program Memory (Flash) : Programın içinde saklandığı ve defalarca silinip yazılabilen Flash bellektir. 16F628A'nın Flash kapasitesi (2Kx14 Byte)'tır.

Veri Belleği (Data Memory) : PIC'in elektrigi kesilse bile silinmesi istenmeyen bilgilerin saklandığı bellektir. 16F628A'nın Veri belleği (128x8 Bit)'tir.

RAM : Programın çalışması esnasında geçici olarak saklanması gereken verileri (Toplama, Çarpma gibi işlemlerde) kullanılan bellektir. Genel amaçlı saklayıcı (GPR) denilen bu bellek, programın çalışması esnasında değişkenleri saklamak için kullanılır. RAM belleğin içinde özel amaçlı saklayıcı (SPR) adı verilen saklayıcılar da vardır.

PORTA ve PORTB : Mikrodenetleyicinin dış Dünya'ya açılan elektronik kapısıdır.

PIC portları PORTA, PORTB, PORTC gibi isimlendirilir. Her port 8 bitten oluşur.

PIC'ten dışarıya bilgi gönderilecek ise çıkış, PIC'e dışarıdan bilgi gelecek ise giriş olarak yönlendirilir. PIC 16F628A'da PORTA'nın RA5, RA6 ve RA7 pinleri

giriş veya çıkış olarak kullanılmaz. RA5 ucu +5V'a bağlanarak kullanılmalıdır.

RA5 = pin 4 RA6 = pin 15 RA7 = pin 16

3- Tanımlayıcı kelimeler (Türk isimleri): Değişkenler, sabitler, fonksiyonlar gibi bazı ifadeler isim verilirken bazı kuralları vardır. Bu isimlerde Türkçe karakter kullanılamaz. 0'dan 9'a kadar rakamlar değişken isimlerinde kullanılabilir. Alt çizgi "_" karakteri belirtilen isimler arasında kullanılabilir. Micro C dilinde büyük harf ile küçük harf arasında duyarlılık vardır. Örneğin Toplam ile TOPLAM kelimeleri farklıdır.

Örneğin: Toplam3 geçerli %yükseklik Geçersizdir. 5Toplam geçersizdir. Çünkü ilk karakter sayı olamaz.

4- Tam Sayılar: Tam sayılar Pecimal (Onlu), Oktal (Sekizli), İkili (Binary) veya Hexadecimal (Onaltılı) sistemlerde yazılabilir. Micro C derleyicisi bu sayı sistemlerinin hepsini kullanabilir.

Onaltılık sayılar yazılacaksa başına "0x" ifadesi yazılır.

Onluk sayılar yazılacaksa herhangi bir ifade olmadan direkt yazılır.

Sekizli sayılar yazılacaksa "0" ile başlayarak yazılır.

İkili sayılar yazılacaksa "0b" ile başlayarak yazılır.

5- Ondalık Sayılar: Ondalık sayılara, kayan noktalı sayı da denilir. Kayan noktalı sayılar, sayının tam kısmı, ^{sonra} nokta ve ^{sonra} noktadan gelen ondalık kısımdan oluşur. Örneğin -1.23, 2e-5, 3.14 gibi yazılır.

6- Karakter ifadeler: Karakter ifadeleri tek tırnak arasına yazılan ASCII karakterden oluşan ifadelerdir. Örneğin 'a', 'G', 'k' gibi.

7- String ifadeler: String ifadeler; ASCII karakterlerden oluşan, çift tırnak işaretlerinin arasına yazılan ifadelerdir. Micro C derleyicisi, çift tırnakla başlayan ve çift tırnakla sonlanan ifadeleri string olarak algılar. Örneğin "Merhaba Dünya" 13 karakter uzunluğunda string bir ifadedir.

8- Bosluk (Kaçış) Karakterleri: Harfler ve sayıların dışında bir takım özel karakterler de char tipi ile kullanılabilir. Bunlara boşluk karakterleri denilir.

"\" karakteri ile başlayan ekrana doğrudan yazılmayan fakat görüntülenmeye ilgili görevleri olan karakterlerdir. Örneğin \n karakterinin görevi yeni bir satıra geçmektir. \r karakteri satır başı yapar.

9- Tam Ayırıcı İşaretler: Micro C'de kullanılan ayırıcı işaretler şunlardır.

a- [] köşeli parantez

b- () parantez

c- , virgöl

d- ; noktalı virgöl

e- : iki nokta üst üste

f- . nokta

g- * yıldız

h- = eşittir

i- # diyez şimdi bunları açıklayalım.

10-a [] köşeli parantez: Dizilerin uzunluklarını ve indekslerini belirtmek için kullanılır. Örneğin `char str[6] = "microC";` ile `str` dizisinin içerisinde `microC` ifadesi atanır. `ch = str[3];` yazmak dizinin 4 elemanı `ch` olur, demektir.

10-b () parantez: Aritmetik işlemlerde gruplandırma için, karşılaştırma işlemlerinde, fonksiyon çağırma ve bildirimlerinde "(" parantez kullanılır. Örneğin `d = c * (a + b);` işleminde önce parantez içindeki işlem yapılır. `if (i == 1) ++x;` karşılaştırma işleminde `i == 1` ise `x` değerini arttırır.

10-c , virgöl: Fonksiyon parametrelerinin arasına, aynı türdeki değişkenler tanımlanırken veya dizi elemanlarının arasına ayırıcı olarak virgöl işareti konulur. Örneğin `char i, j, k;` `i, j` ve `k` değişkenleri tanımlanmıştır. `LCD_Out(1, 1, txt);` yazılarak LCD çıkışında 1. satır 1. sütuna `txt` değişkeni yazdırılır.

10-d ; Noktalı virgöl: Her yazılan komutun sonlandırılmasında noktalı virgöl kullanılır. Noktalı virgöl ile sonlandırılmayan komutlar hata verir. Örneğin `x = a + b;` doğru yazılmış bir komut iken `x = a + b` yazılır ise hata verir.

10-e: İki nokta üst üste: Etiket tanımlamalarında iki nokta üst üste ayırma işareti kullanılır. Örneğin;

Basla: x=0;

if(a=x) { PORTB = 0x06; } else { PORTB = 0x00; }

goto Basla; // Burada Basla etiketi iki nokta üst üste ile birlikte kullanılmıştır.

10-f: Nokta .: Mikrodenetleyicide kullanılan bir register (kaydedici) içindeki bir bit alanına ulaşmak için kullanılır. Örneğin TRISB.F0=0; yazımında TRISB'den sonra . kullanılmıştır. Böylece RBO biti 0 olarak seçilir.

10-g: * Yıldız: Yıldız işareti "*" göstericileri tanımlarken kullanılır. Göstericiler dolaylı adreslemede kullanılır. Örneğin int *veri_pt; yazımında tamsayı tipinde bir verinin adresini saklanır.

10-h: Eşittir =: Sabitlere ya da değişkenlere değer atamak için eşittir "=" işareti kullanılır. Örneğin int x=5; yazımında x değişkenine 5 değeri atanır.

10-I: Diyez #: Ön işlemci komutlarında veya tanımlama yapmak için diyez işareti "#" kullanılır. Örneğin #include <math.h> yazımında # işareti ile ön işlemci komutu kullanılır. #define buton = PORTB.F0 yazımında ise buton değişkeninin PORTB'nin RBO pinine bağlı olduğu açıklanır.

Depolama Sınıfları: Bir değişken veya fonksiyonun C programı içindeki ömrünü depolama sınıfları belirler. Micro C programında kullanılan depolama sınıfların auto, register, static ve extern gibi türleri vardır. Şimdi bunları açıklayalım.

a - Auto: Auto; veri depolama sınıfı, değişkenin lokal bir değişken olduğunu belirtir. Örneğin Auto olarak tanımlanmış bir değişkene, oluşturulduğu fonksiyon veya kod bloğunun içinden erişilebilir.

b - Register: Register veri depolama sınıfı, RAM yerine bir kaydediciye depolanan, bir lokal değişkeni tanımlamak için

kullanılır. Register C dili için geçerli bir kuraldır.

c- Static : Static olarak tanımlanan bir değişkeni, program çalışmaya başlar başlamaz hafızaya yerleşir ve program sonlanana kadar hafızadan silinmez. Örneğin;

```
void f() { // f fonksiyonu her çağrılışında a=1 olur.  
    int a = 1; // f fonksiyonu ilk çağrıldığında b=1 olur.  
    static int b = 1 // static int ile b=1 olur. Devamlı  
    a++; // olarak b artarken, a=1 kalır. Artmaz.  
    b++; }
```

d- Extern : Global değişkenler ve fonksiyonlar varsayılan olarak extern tanımlanır. Bir değişken global ise program çalıştığı andan itibaren bellekte tutulur.

TİP NİTELEYİCİLERİ : Micro C'de oluşturulan bir değişken için ek özellikler katan tamamlayıcılara tip niteleyicisi denir. Bu tip niteleyicileri Const ve Volatile olarak belirtilir. Şimdi bunları açıklayalım.

a- Const : Const niteleyicisi bir değişkenin önünde kullanılırsa, o değişkenin programın çalışması esnasında değiştirilemeyeceği anlamına gelir. Örneğin, Const int a=10; yazımı a tam sayı değişkeninin 10 olacağını ve program esnasında sabit kalacağı anlamına gelir.

b- Volatile : Volatile niteleyicisi, bir değişkenin kullanıcı müdahalesiyle yada kullanıcı müdahalesi olmadan değiştirilebileceği anlamına gelir.

Başlangıç değeri atama : Micro C'de oluşturulan bir değişkene program başlarken istenilen değeri atamaya, başlangıç değeri atama denir. Örneğin; int i=12; yazım (i) başlangıç değeri olarak i değişkeni 12 olarak atanır.

MİCRO C'DE VERİ TÜRLERİ : Micro C dilinde değişkenler ve sabitler tanımlanırken kullanılacak verinin türünün belirtilmesi

gerektir. Örneğin Micro C'de tamsayı olarak bir veri tanımlanırken "int" ifadesiyle tanımlanır. Şimdi bu veri türlerini açıklayalım.

1. Aritmetik türler: Micro C dilinde aritmetik türleri belirleyen ifadeler void, char, int, float ve double kelimeleridir. Bu ifadelerin önüne "signed" ifadesi eklenirse işaretli olduğunu, önüne "unsigned" ifadesi eklenirse işaretsiz olduğunu gösterir. Eğer "short" ifadesi eklenirse değişkenin kısa olacağını, "long" ifadesi eklenirse değişkenin uzun olacağını gösterir.

a- Tamsayı değişkenler: Micro C'de tamsayı değişkenler, int ve char ifadeleriyle gösterilir. Aşağıdaki tabloda Tamsayı olarak tanımlanabilecek değişkenlerin alabileceği max. ve min. değerleri gösterilmiştir.

	Tür (Type)	Boyut (Byte)	Sınırlar (min - max)
char	<u>(unsigned) char</u>	1 Byte (8bit)	0 ... 255
signed char	<u>signed char</u>	1 Byte (8bit)	-128 ... 127
short	<u>(signed) short (int)</u>	2 Byte (16bit)	-32768 ... 32767
unsigned short	<u>unsigned short (int)</u>	2 Byte (16bit)	0 ... 65535
int	<u>(signed) int</u>	4 Byte (32bit)	-2147483648 ... 2147483647
unsigned int	<u>unsigned (int)</u>	4 Byte (32bit)	0 ... 4294967295
long	<u>(signed) long (int)</u>	8 Byte (64bit)	-9223372036854775808 ... 9223372036854775807
	<u>(unsigned) long (int)</u>	8 Byte (64bit)	0 ... 18446744073709551615

b- Ondalık sayı değişkenler: Önünde "float" ve "double" ifadeleri bulunan değişken isimleri Micro C derleyicisinde ondalık sayı olarak değerlendirilir. Bunlar float için 4 Byte olup -1.5×10^{45} ile $+3.4 \times 10^{38}$ arasında olup double için de aynı şekilde -1.5×10^{45} ile $+3.4 \times 10^{38}$ arasında değer alabilir.

2 - Void Türü : Void ifadesi, herhangi bir değerin olmadığını belirten özel bir ifadedir. Özellikle fonksiyonların tanımlanmasında void'den sonra istediğimiz bir isim vererek fonksiyon tanımlarız. Örneğin; sıcaklık değişkeni 32 olsun.

```
void sıcaklik-yaz (char sıcaklik) {  
    LCD-out-Cp ("Sıcaklık:"); // Ekranı Sıcaklık yazdır  
    LCD-out-Cp (sıcaklik); // Ekranı sıcaklık değerini yazdır  
    LCD-out-Cp (223); // Derece işareti'nin kodunu yazdır  
    LCD-out-Cp ('C'); // Ekranı C işareti karakteri yazdır  
}
```

Ekranı **Sıcaklık : 32 °C** şeklinde fonksiyon yazdırma yapar

GÖSTERİCİLER : Gösterici, bir değişken, dizi gibi bilgilerin hafızadaki adresini gösteren değişkendir. Bu değişkene doğrudan erişilebilirken, bu hafıza alanının adresini tutan başka bir hafıza alanıdır. Göstericinin tanımlanması değişkenin tanımlanmasına benzetilebilir beraber önüne "*" işareti konulur.

Örneğin; unsigned int *p; yazımında işaretsiz tam sayı olarak gösterici tanımlanmış olur.

YAPILAR : Yapılar yardımıyla aynı veya farklı türdeki değişkenler (int, char, double vb) bir araya getirilerek kendi yapı türümüzü oluşturabiliriz. Örneğin; struct öğrenci-Yapı {

```
    char isim [25];  
    char soyisim [25];  
    int yas;  
    int numara; };
```

Yazılımında char isim [25]; ile 25 tane dizi isim değişkeni altında tanımlanmıştır. Yapı içindeki elemanlara doğrudan erişim ya da dolaylı erişim metodları ile ulaşabiliriz. Doğrudan erişim için "." operatörü kullanılır. Örneğin öğrenci-Yapı.isim yazımıyla yapı içindeki isim değişkenine erişebiliriz. "->" operatörü ise gösterici olarak

belirlenen bir yapıya erişim sağlar.

DİZİLER: Micro C'de diziler, aynı türdeki birden fazla veriyi bir değişken dizi ismi içinde toplayan, depolayan yapıdır. Örneğin ; `int dizi_adi [indeks] =` yazımında `int` dizideki değişkenlerin tamsayı olduğunu, dizinin adının "`dizi_adi`" olduğunu ve `indeks` ise dizinin bir elemanı olduğunu belirtir.

Örneğin `int vektor1 [10];` // 10 tane tamsayı olan vektor1 dizi
`int aci [5] = {0,30,45,60,90};` // 5 tane tamsayı açısı dizi.

FONKSİYONLAR: Fonksiyonlar, giriş parametrelerine göre bazı görevleri gerçekleştiren alt program parçacıklarına denir. Örneğin program yazarken bir işlemi birden fazla tekrarlatmak gerekebilir. Fonksiyonlar yardımıyla bir kod parçacığını ana program (main) içinde sürekli tekrarlamamış oluruz. Micro C dilinde hazır kütüphanelerdeki fonksiyonlar kullanılabildiği gibi kendi fonksiyonlarımızı yazarak kütüphane oluşturabiliriz.

Örneğin `int fonksiyon-adi (int x, int y) { ..... return }`
`void main () {`
`int ara-deger; // Fonksiyondan gelen değer`
`ara-deger = fonksiyon-adi (3,5); // x=3`
`// y=5 değerini ara değer olarak tutar.`

OPERATÖRLER: Micro C'de operatörler (matematiksel ve mantıksal işlemler için 15 farklı kategoride operatörün işlem önceliği vardır)

Öncelik sırası	OPERATÖRLER	Aynı satırda önceliği
1	<code>?</code>	soldan sağa
2	<code>= * = / = % = + = - = & = ^ = = << = >> =</code>	sagdan sola
3	<code>? :</code>	sagdan sola
4	<code> </code>	soldan sağa
5	<code>& &</code>	soldan sağa
6	<code>!</code>	soldan sağa
7	<code>^</code>	soldan sağa
8	<code>&</code>	soldan sağa
9	<code>== !=</code>	soldan sağa
10	<code><< = >> =</code>	soldan sağa
11	<code><< >></code>	soldan sağa
12	<code>+ -</code>	soldan sağa
13	<code>* / %</code>	soldan sağa
14	<code>! ~ ++ -- + - * & (type) sizeof</code>	sagdan sola
15	<code>() [] . -></code>	soldan sağa

Karşılaştırma operatörleri: Karşılaştırma operatörleri, ifadeler arasında mantıksal karşılaştırma işlemleri yaparak sonuçta logic 0 veya logic 1 değerini döndüren operatörlerdir. Aşağıda verilen tabloda karşılaştırma operatörleri ve anlamları görülmüyor.

Operatör	Anlamı	Öncelik sırası
==	Eşittir	9
!=	Eşit değil	9
<	Küçüktür	10
>	Büyüktür	10
<=	Küçük eşit	10
>=	Büyük eşit	10

Atama Operatörleri: Bir değişkene değer aktarma işlemine atama işlemi denir. En çok kullanılan atama operatörü eşittir "=" operatörüdür. Aşağıdaki tabloda eşittir operatörü ile birlikte kullanılan atama operatörleri verilmiştir.

Operatör	Açıklama	Öncelik sırası.
=	Atama operatörü	2
+=	Toplama ataması	2
-=	Çıkarma ataması	2
*=	Çarpma ataması	2
/=	Bölme ataması	2
%=	Mod ataması.	2

Mantıksal Operatörler: Mantıksal operatörler bir veya birden fazla karşılaştırma yaparak sonucu lojik 0 veya lojik 1 olarak verirler.

Operatör	Açıklama	Öncelik sırası.
&&	Mantıksal VE işlemi	5
	Mantıksal VEYA işlemi	4
!	Mantıksal DEĞİL işlemi	14

Diğer Operatörler: Bazı operatörler daha önce açıkladığımız operatörlere dahil edilmez. Bunlardan birincisi "()" operatörüdür. Bu operatör parantez açma ve kapama işaretleri, işlem sırasını değiştirme işlemi yanında fonksiyon çağırma operatörü olarak kullanılır. Örneğin main fonksiyonu yanındaki parantez () onun bir fonksiyon olduğunu belirtir. Bunlardan ikincisi "[]" operatörüdür. Bu operatör aynı zamanda dizi operatörüdür. Bunun yanında göstericilerde kullanılan $\rightarrow$ ve $\cdot$ operatörleri de bu grupta sayabiliriz.

Deyimler: Deyimler; sabit, değişken, fonksiyon ve operatörlerin bir araya gelmesinden oluşur. Örneğin `carpan * carp()` yazımı bir deyimdir. Aynı şekilde `temp = x/y` 'de bir deyim olarak sayılır.

İfadeler: ifadeler (statement), bir programın içindeki işlemleri gösterir. C dilinde ifadelerin sonunda ; (noktalı virgül) ile sonlandırıcı bulunmalıdır. Örneğin `if (x == y) a = x * 3;` bir ifadedir. Anlamı eğer x, y'ye eşit ise a'nın değerine x'i 3 ile çarparak çıkan sonucu ata demektir.

Micro C'de kullanılan ifadeler 5 gruba ayrılır. Bunlar;

1. Etiket ifadeleri : Örneğin `başla :`
2. Deyim ifadeleri : Örneğin `int sayi = 0 ;`
3. Kontrol ifadeleri : Örneğin `if sayi >= 10 sayi = 0 ;`
4. Döngü ifadeleri : Örneğin `while (1)`
5. İç içe blok ifadeler : Örneğin `{ { } }`

Kontrol Yapıları: Kontrol yapıları, program içinde kendilerine verilen değerleri karşılaştırıp bu değerlere göre programın işlenmesini sağlayan yapılardır. Bu kontrol yapıları 2 farklı komutla yapılırlar. Bunlar; 1. if else kontrol yapısı.

2. switch case kontrol yapısı'dır.

1 - if - else kontrol yapısı : if kelimesinin türkçe karşılığı eğer demektir. Anlamından anlaşılacağı gibi if - else bloğu karşılaştırma ve bu karşılaştırma sonucuna göre programın istenilen kod parçasına dallanmasını sağlayan kontrol yapısıdır. if deyiminde karşılaştırılan ifade doğru ise program if bloğu içinde çalışmaya devam eder. Sayet ifade yanlış ise if bloğuna girmeden bloğun altından devam eder.

Örn1: if (koşul) koşul doğru olduğunda çalıştırılacak satır
Yukarıdaki ifadede if içindeki koşul doğru olduğunda çalıştırılacak satır yazılmıştır. Else yazılmamıştır. 2. örnekte else yazılmıştır.

Örn2: if (koşul) { koşul doğru olduğunda çalıştırılacak satır; }
else { koşul yanlış ise program buradaki komutlara girer }

Örn3: if (koşul1) koşul1 doğru ise çalıştırılacak satır.
else if (koşul2) koşul2 doğru ise çalıştırılacak satır.
if (koşul3) koşul3 doğru ise çalıştırılacak satır.
else (koşul3 yanlış ise bu satır çalıştırılır.)
else (koşul2 yanlış ise bu satır çalıştırılır.)

2 - switch - case kontrol yapısı : Switch ifadesi bir deyim - in sabit bir tamsayı ile eşleşip eşleşmediğini kontrol ederek eşleşmenin olduğu yerden programın devam etmesini sağlayan kontrol yapısıdır.

Örnek: switch (deyim) {
 case sabit deyim 1 :
 ifadeler; break;
 case sabit deyim 2 :
 ifadeler; break;
 default :
 ifadeler;
switch (PORTB) { // Switch - case kontrol yapısı
 case 0 : Lo(); break; // iç içe if ifadesi yerine
 case 1 : Hi(); break; // kullanılabilir. PORTB = 0 ise
 default : // Lo(); fonksiyonu. PORTB = 1 ise Hi(): çalıştırılır

Döngü Yapıları: Micro C'de bazı kod parçacıklarının birden fazla çalıştırılması gerektiğinde döngü yapıları kullanılır. C dilinde 3 tür döngü yapısı vardır. Bunlar;

1. **for döngüsü**
2. **while döngüsü**
3. **do-while döngüsü**

Şimdi bu döngü yapılarını açıklayacağız.

1. For döngüsü: Micro C'de program yazarken en çok kullanılan döngü türü for döngüsüdür. Diğer döngü ifadelerine göre daha esnek ve güçlüdür. For döngüsünün yazım şekli aşağıdaki gibidir.

```
for (sayacın başlangıç değeri ; koşul ; sayacın artış miktarı ) {  
    çalıştırılacak satırlar  
    çalıştırılacak satırlar  
}
```

For döngüsüne girmeden önce ilk olarak sayacın başlangıç değeri değişkene atanır. Bu işlem her döngü başlangıcında bir kereye mahsus yapılır. Koşul ifadesi döngüye ilk giriste kontrol edilir. Koşul sağlanırsa yani (true=1) doğru ise döngüye girilir. Koşul yanlış ise (false=0) olana kadar döngü çalışmaya devam eder. 3. parametre ise sayacın artış miktarıdır. Bu ise döngüdeki değişkenlerin artış miktarını belirler.

Örneğin ; `for (sayac = 0 ; sayac < 10 ; sayac++) s += a[i] * b[i];`

Bu örnekte sayac değişkeninin değeri 0'a eşitlenir. Sayac değeri 10'dan küçük olduğundan sayac değeri 1 arttırılır.

For döngülerinde { } blok parantezleri birden fazla satır ile komutlar kullanılacak ise yazılır. Bir satır ile yazılan for ifadelerinde blok parantezi zorunlu değildir. For döngüsünden sonra iki defa ";" ifadesi kullanıldığında bunun anlamı for döngüsünün while(1) döngüsü gibi devamlı olan bir döngü olarak kullanılacağını belirtir. (for ; ;)

2- **while döngüsü:** while döngüsü kendisine verilen koşul doğru olduğu sürece döngüyü çalıştırır. Döngüye girmeden önce koşulun doğruluğu kontrol edilir. Koşul doğru (true, 1) ise döngü çalıştırılır.

Örnek: while (koşul ifadesi) {
çalıştırılacak satırlar.
çalıştırılacak satırlar.
}

while döngüsü, döngünün koşul ifadesinin sağlanması bitene kadar devam eder. Programlarımızın devamlı çalışması için çoğunlukla sonsuz döngüye ihtiyaç vardır. Bunun için while döngüsü kullanılır. While döngüsünde koşul ifadesine (1) veya (true) bilgisi girilir. Böylece sonsuz döngü elde edilir { } parantezleri arasında yazılan ifadeler yukarıdan aşağıya çalıştırıldıktan sonra tekrar başa döner.

3- **Do-while döngüsü:** do-while döngüsü while döngüsünün bir çeşididir. While döngüsünde koşulun doğru olup olmadığının kontrolü döngü başında yapılır, koşul doğru olmadığında döngüye girilmez. Do-while döngüsünde ise koşulun doğruluğunun kontrolü sonunda yapılır. Yani koşul yanlış (false=0) olsa bile en az bir kere döngü bloğundaki kodlar çalıştırılır.

Örnek: do {
çalıştırılacak satırlar.
çalıştırılacak satırlar.
} while (koşul);

Do-while döngüsünde kontrol işlemi döngü sonunda olduğu için kontrol işlemi sağlanmasa bile (koşul) do-while döngüsü en az bir kere çalıştırılır. While(1) döngüsü içinde olursa devamlı olarak çalıştırılır. Örneğin while(1) {

(while(1) döngüsü içinde kullanılan do-while döngüsü.
do { portb = display[sayac++];
delay_ms(250);
} while (sayac < 10);
sayac = 0;

Ön işlemci Komutları: Ön işlemci, derleme işleminin bir parçasıdır.

Yazılan program derlenmeye başlamadan önce ön işlemci komutları çalıştırılır ve çalışan sisteme özgü olan görevleri yerine getirir.

Micro C dilinde # (diyez) işareti ile başlayan satırlar ön işlemci komutlarını ifade ederler. Aşağıdaki tabloda standart ön işlemci komutları görülmektedir.

#(null directive)	#error	#ifndef
#define	#endif	#include
#elif	#if	#line
#else	#ifdef	#undef

Programımızın başlangıcında, programımıza bir dosya eklemek için #include komutu kullanılır. Bu dosyalar genellikle fonksiyon prototiplerinin tutulduğu header (başlık) dosyalarını, yada Micro C kütüphanesinde bulunmayan bir elektronik devre elemanı için yazdığımız kayıtlı sürücü dosyalarını programımıza ekleyebiliriz.

Örneğin #include <dosya adı>

#include "C:\pic-projelerim\test.h"

Yukarıdaki yazım Micro C içine eklenirse C diskindeki (HDD) pic-projelerim klasörü içindeki test.h dosyasını programın içine ekler.

Şartlı derleme komutları #if, #elif, #else ve #endif ön işlemci komutlarıdır.

Örneğin

#if sabit-ifade-1

çalıştırılacak satır

#elif sabit-ifade-2

çalıştırılacak satır

#elif sabit-ifade-n

çalıştırılacak satır

#else

çalıştırılacak satır

#endif.

By örnekteki ön işlemci

komutlarının kullanımı if

kontrol yapısına benzer.

Sadece bir tane #else

kullanılabilir. #endif ile

şartlı derleme sonlandırılır.

Daha önceden #define ön işlemci komutu ile tanımlanan bir ifade

varsa #ifdef ve #ifndef önislemleri komutlarıyla tanımlanma-
nın yapıp yapılmadığı kontrol edilebilir.

#ifdef (tanımlayıcı)

Derlenecek satırlar

#endif

Eğer bu ifade tanımlanmışsa

#ifdef önislemleri komutunun

altındaki satırlar çalıştırılır.

#endif önislemleri komutuyla

şartlı derleme işlemi sonlandırılır.

Micro C ile PIC16F628A Portlarının ayarlanması:

16F628A veya 16F88 mikrodenetleyicilerinin 8 tane A portu, 8 tane B portu olmak üzere giriş/çıkış olarak atanabilecek 16 tane pini vardır. PIC mikrodenetleyicilerde bulunan özel amaçlı Tris kaydedicileri kullanılarak bu pinler giriş-çıkış olarak ayarlanabilir. Port A'yı giriş-çıkış olarak ayarlamak için TRISA kaydedicisi (register) kullanılır. Port B'yi giriş ve çıkış kullanabilmek için de TRISB kaydedicisi kullanılır.

Lojik 1 yüklenen bitlere (pinlere) karşılık gelen pin **GİRİŞ**,
Lojik 0 yüklenen bitlere (pinlere) karşılık gelen pin **ÇIKIŞ** olarak belirlenir. Örneğin B portunu kontrol eden TRISB özel amaçlı registerine **0b00000000**; (Binary sayı) yüklenirse B portunun tüm pinleri **ÇIKIŞ** olarak belirlenmiş olur.

Eğer **TRISB = 0x00**; olarak yazılırsa da Hexadecimal sayı olarak 0 (sıfır) yüklenerek tüm pinleri **ÇIKIŞ** olur.

Aynı şekilde TRISA registeri de Port A'yı giriş-çıkış yapar. Örneğin RA0 ve RA1 bitlerini **GİRİŞ** yapmak istersek TRISA özel amaçlı registerine **0b00000011** (Binary sayı) yüklenmelidir. Tris B veya Tris A registerlerinde 0. bit (RA0) en sağdan başlayarak yazılır. Bu nedenle bu örnekteki RA0 ve RA1 en sağdaki iki bite karşılık gelir.

TRISA = 0x03; olarak yazılırsa da Hexadecimal sayı olarak 3'e karşılık RA0 ve RA1 bitleri Lojik 1 yapılmış olur.

8 bit sayıcı (Uygulama)

İşlem Basamakları:

1. Micro C programını bilgisayara kur ve çalıştır,
2. New Project'i seç. Adını (Name) sayıcı olarak ver,
3. Açılan kod penceresine kodları yaz.
4. Project → Edit project MCU 16F628A'yı seç.
5. Oscillator Selection I/O Function'un INT'ini seç.
6. Oscillator Frequency 4.000.000 seç.
7. Oluşan sayıcı.hex dosyasını bir klasöre kaydet.
8. Proteus (Ish-Ares) 8.0'da aşağıdaki devreyi oluştur.
9. Bu devredeki PIC16F628A entegresine sağ tıklayarak oluşturduğun programı (sayıcı.hex) entegreye yükle
10. Play (simülasyon) başlat "▶" tusuna basarak test et.

PL Devices

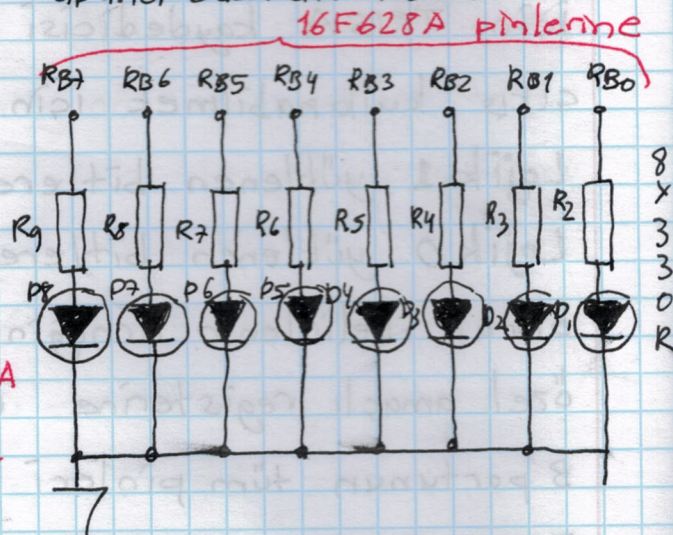
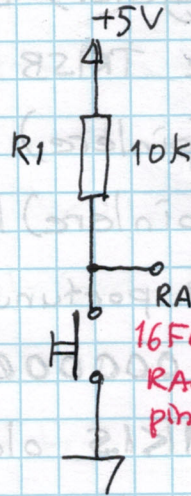
BUTTON

PIC16F628A

RES

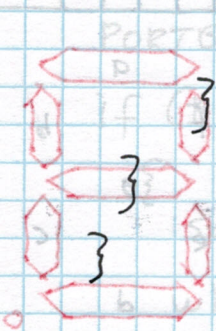
MINRES330K

LED-RED



Kodlar: (20070709.1) 10:02 0-0 10:02 F

```
void main() { // Ana program
    CMCON = 7; // Karşılaştırmacılar kapalı
    TRISA = 0x03; // PORTA.0 ve 1. giriş
    PORTA = 0x00; // PORTA'yı temizle
    TRISB = 0x00; // PORTB'nin hepsi çıkış
    PORTB = 0x00; // PORTB'yi temizle
    while(1) { // Sonsuz döngü
        int sayı; // Sayı değişkeni tanımlandı max 255
        if(PORTA.F0 == 0)
        {
            for(sayı = 0; sayı < 255; sayı++) // 0'dan 255'e sayı.
            {
                PORTB = sayı; // Sayı değerini PORTB'ye gönder
                delay_ms(500); // 500 ms bekle
            }
        }
    }
}
```

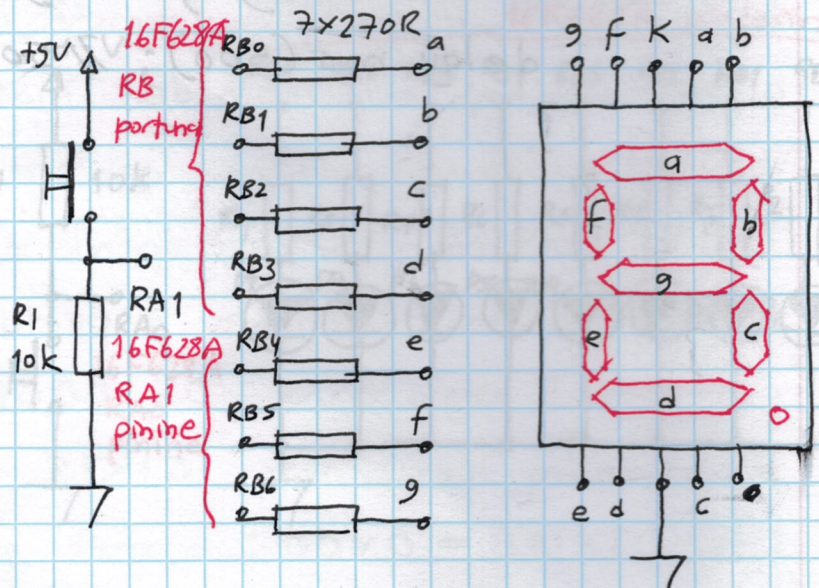


AKTIF
10:02
10:02
10:02
10:02

Kodlar:

you might find

- $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ 0 & 1 \end{pmatrix}$



Kodlar :

```
#define BUTON PORTA.RA1
unsigned short sayac = 9; // Sayac değeri 9 yapılır (ilk değer)
char seven-segment[10] = { 0x3F, 0x06,
0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07,
0x7F, 0x6F }; // 0'dan 9'a kadar 7 segment için Hex. kodları
void main() { // Ana program
    CMCON = 7; // Karşılaştırıcılar kapalı.
    TRISA = 0x03; // PORTA 0 ve 1 girişi yapıldı
    PORTA = 0x00; // PORTA temizlendi.
    TRISB = 0x00; // PORTB'nin hepsi çıkış
    PORTB = 0x00; // PORTB temizlendi.
    BUTON = 0; // PortA.2 sıfır yapıldı.
    while (1) { // Sonsuz döngü
        PORTB = seven-segment[sayac]; // PORTB'ye 10'luk dizi eşitlenir.
        if (BUTON) // Eğer butona basılırsa (ilk anda 9 yazar.
        {
            sayac--; // Sayac değerini
            while (BUTON); // Eğer hala butona basılı iken
        }
        if (sayac == 255) sayac = 9; // Eğer sayac 0'dan küçük
        } // olursa yani 255 olursa o zaman tekrar 9 olacak.
    }
```