Lubin, 2021

MikroC Kitapçık & Temrinleri

INCREASING THE DIGITAL QUALIFICATIONS IN DIGITAL TRANSFORMATION-INDUSTRY 4.0
KA202 ERASMUS+ PROJESİ

Co-funded by the
Erasmus+ Programme
of the European Union

ÖNSÖZ

Endüstri 4.0 sanayi devriminde mikrodenetleyicilerin rolü büyüktür. Bir çok işlem mikrodenetleyici kontrollü gerçekleşmektedir. Öğrencilerimizin mikrodenetleyicileri anlamaları ve öğrenmeleri önemlidir. Öğrencilerimizin bu konuda en büyük ihtiyaçlarından birisi de konuyla ilgili, teknik eğitime uygun uygulamalı kaynak kitap ihtiyacıdır.

IQinDIGIT KA202 proje ekibi olarak öğrencilerimizin bu ihtiyacını bir miktarda olsa karşılamak amacıyla bu kitapçık oluşturulmuştur. Mikrodenetleyici olarak MicroChip firmasına ait PIC temel alınmıştır. Başka mikrodenetleyicilere de uyum sağlanabilir. Mikrodenetleyiciyi programlamak için MikroElektronika firmasına ait MikroC yazılım dili seçilmiştir. Devre uygulamaları board üzerine kurulabileceği gibi Labcenter firmasına ait Proteus programında da simüle edilebilir. Böylece tüm işlemler bilgisayar ortamında yapılabilir. Eğitim sisteminde Dijital Dönüşüme bir katkı olarak görülebilir.

KA202 - Increasing Digital Qualification in the Digital Transformation- Industry4.0 isimli Erasmus+ projesi sebebiyle bir araya gelen öğretmenlerin bilgi ve tecrübelerini sizlere aktarmak bizi mutlu etmektedir. Kitabın tüm teknik personele yararlı olmasını diliyoruz.

EXTERNATO COOPERATIVO DA BENEDITA

IISS GALILEO GALILEI

ITO VAKSI SULEYMAN TAŞTEKIN MTAL

ZESPOL SZKOL NR 1

"Erasmus+ Programı kapsamında Avrupa Komisyonu tarafından desteklenmektedir. Ancak burada yer alan görüşlerden Avrupa Komisyonu ve Türkiye Ulusal Ajansı sorumlu tutulamaz."

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Co-funded by the Erasmus+ Programme of the European Union

INDEX

Index 2

MikroC 3

Yazım Kuralları 3

Veri Türleri 4

Operatörler 5

Kontrol İfadeleri 6

Döngü Yapıları 7

Fonksiyon ve Diziler 7

Önişlemci Komutları 8

MikroC Compiler and Proje Organizasyonu 9

İlk Projenizin Oluşturulması 11

Temrin 1 14

Temrin 2 17

Temrin 3 19

Temrin 4 23

Temrin 5 25

Referanslar 26

Katkı Sunanlar 27

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union

MikroC

MikroC

MikroC programlama dili kütüphanesi en zengin programlama olmakla birlikte;

- C tabanlı olması,
- Komut yapısının kolay, kullanışlı ve esnek olması,
- Mikrodenetleyici programcısı için gerekli olan yazılımı barındırması,
- Hemen hemen her konuda örnek devre şeması ve program kodları sunması,
- Mikrodenetleyici donanımlarına kolay ve esnek erişim imkanı vermesi, gibi üstün özellikleri vardır.

Emin olun, PIC programlamayı hiç bilmeyen bir kullanıcı bile, MikroC kütüphanesinde verilen örnekleri inceleyerek iyi bir PIC programcısı olabilir. Bu kitapta verilenn uygulamaların bir çoğu MikroC kütüphanesinden faydanılarak yapılmıştır. Tüm bu örnekleri dijital ortamda uygulamak için MikroC Pro ve/veya Proteus programları yeterlidir.

PIC mikroişlemciler için MikroC PRO'yu, www.mikroe.com/mikroc-pic internet sayfasından bilgisayarınıza indirebilirsiniz.

Ayrıca ilgili mikrodenetleyici ile devre simülasyonları yapmak için www.labcenter.com/simulation internet sayfasından Proteus programını bilgisayarınıza indirebilirsiniz.

Yazım Kuralları

MikroC dilinin kendine özgü yazım kuralları vardır. Bu kuralların dışına çıkıldığında MikroC derleyicisi hata verecektir. Bu kurallar;

- MikroC tarafından rezerve edilen anahtar kelimeler kullanıcı tarafından kullanılamaz.

  asm, code, switch, static, operator.

- Açıklamalar "//" işaretinden sonra yazılmalıdır. Örneğin

  //Erasmus+ KA202 Project

- Değişkenlere, sabitlere, fonksiyonlara isim verilirken İngiliz alfabesindeki bütün harfler büyük veya küçük harf olarak, rakamlar, "_", kullanılabilir. Ancak belirlenen ismin ilk karakteri ya bir harf ya da "_" olmalıdır. MikroC büyük küçük harf duyarlılığına sahiptir.

  Total3, Fuse, temp_t1, _user

- Tam sayılar, onluk, onaltılık, ikilik veya sekizlik sayı sistemlerinde yazılabilir. Onaltılık sayı sistemi "0x", ikilik sayı sistemi "0b", sekizlik sayı sistemi "0" ile başlamak zorundadır. Onluk sayı sitemi için bir ifade yazmamıza gerek yoktur. Ondalıklı sayılarda ondalıklı kısım "." ile devam eder.

  0xFE, 0b1101, 0131, -1.35

- Karakter ifadeleri tek tırnak arasına yazılır. String ifadeler çift tırmak içine yazılır.

  'b', 'M', "Hello World"

- Ekrana ekrana doğrudan yazılmayan, ama görüntülemeyle ilgili görevleri olan karakterlere boşluk karakterleri denir. "\" ile başlar.
  \n alt satıra geçer

Ayırıcı işaretler kullanılır;

[ ] Dizilerin uzunluklarını ve indisleri belirtmek için kullanılır.

char str[9] = "IQinDIGIT";

( ) Aritmetik işlemlerde gruplama,karşılaştırma deyimlerinde fonksiyon çağrılarında kullanılır.

Top = mal * (adet + fiyat) , if (m == 100) ++y; fonktop ( );

{ } Bir kod bloğunun başlangıcını ve sonunu belirtir.

if (m == 100) {

}

, Parametrelerin arasına, değişken veya dizi elemanlarının arasına konur.

; Sonlandırma işlemi için kullanılır. Aynı zamanda assembly kodlarında açıklama satırında kullanılır.

= Değişkenlere veya sabitlere değer atamak için kullanılır.

İşlemin bir derleyici eylemi olduğunu belirtmek için kullanılır.

#include <math.h>

Veri Türleri

MikroC programlama dilinde değişkenler, sabitler tanımlanırken türlerin belirtilmesi gerekir.

Const niteleyicisi sabit tanımlamak için kullanılır. Tanımlanan sabitin değeri aynı kalır.

const double HP = 0.735;

Aritmetik türleri belirleyen ifadeler void, char, int, float, ve double kelimeleridir. Bu kelimelerin önüne signed ve unsigned gelerek değişkenin negatif veya pozitif bir sayıyı ifade ettiğini belirtir. Ayrıca short, long kelimeleri değişkenin uzunluğunu belirtir.

Type	Byte	Limits
(unsigned) char	1	0…255
signed char	1	-128…127
(signed) short (int)	1	-128…127
unsigned short (int)	1	0…255
(signed) int	2	-32768…32767
unsigned (int)	2	0…65535
(signed) long (int)	4	-2147483648…2147483647

unsigned long (int)	4	0…4294967295
float	4	$-1.5*10^{45}$…$+3.4*10^{38}$
double	4	$-1.5*10^{45}$…$+3.4*10^{38}$
long double	4	$-1.5*10^{45}$…$+3.4*10^{38}$

Operatörler

Aynı kategorideki operatörlerin öncelik sıraları aynıdır. Aynı satırdaki aynı önceliğe sahip operatörlerde ise işlem sırası soldan sağadır. = *= /= %= += -= &= ^= <<= >>=, ! ++ -- + - * & (type) sizeof hariç. Tabi ki satırda herhangi bir parantez yoksa.

Kullandığımız toplama, çıkarma, çarpma gibi işlemleri gerçekleştiren operatörlere aritmetik operatörler denir.

Operatör	**İşlem**
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Mod Alma
++	Bir arttırma
--	Bir eksiltme

İfadeler arasında mantıksal karşılaştırma işlemleri yapan ve sonucunda TRUE veya FALSE değerlerini döndüren operatörlere karşılaştırma operatörleri denir.

Operatör	**İfadesi**
==	Eşittir
!=	Eşit değildir
>	Büyüktür
<	Küçüktür
>=	Büyük eşittir
<=	Küçük eşittir

Bir değişkene değer aktarılması işleminde kullanılan operatörlere atama operatörleri denir. En sık kullanılanı "=" operatörüdür. Karşılaştırma operatörü == ile karıştırmamalıdır.

Operatör	**Anlamı**
=	Basit Atama
+=	Toplama Ataması
-=	Çıkarma Ataması
*=	Çarpma Ataması

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union

/=	Bölme Ataması
%=	Mod Ataması

Bir veya birden fazla karşılaştırma yaparak sonucu TRUE veya FALSE olarak döndüren operatörlere mantıksal operatörler denir.

Operatör	**Anlamı**
&&	Mantıksal VE işlemi
||	Mantıksal VEYA işlemi
!	Mantıksal DEĞİL işlemi

Ayrıca << sola kaydırma ve >> sağa kaydırma operatörleri de oldukça fazla kullanılmaktadır.

Kontrol İfadeleri

Kontrol ifadeleri, program akışı esnasında verilen değerleri karşılaştıran ve bu değerlere göre programın işlenmesini sağlayan yapılardır.

If - Else

If – else bloğu karşılaştırma yapar. Bu karşılaştırmanın sonucuna göre programın dallanmasını sağlar.

```
if ( koşul )
        Koşul doğru olduğunda çalıştırılacak satır
```

If tek başına kullanılabileceği gibi Else ile beraberde kullanılabilir.

```
if ( koşul ) {
        Koşul doğru olduğunda çalıştırılacak satır
}
else {
        Koşul yanlış olduğunda çalıştırılacak satır
}
if ( pos > 0 ) pos --; else pos = 3 ;
```

Switch - Case

Switch ifadesi bir deyimin sabit bir tamsayı ile eşleşip eşleşmediğini kontrol eder ve eşleşmenin olduğu yerden programın akışını sağlar. Eşleşme sağlandığında kontrol ifadesini bitirmek için break komutu kullanılır. Hiçbir durumun eşleşmemesi durumunda default deyimi içindeki komutlar çalıştırılacaktır.

```
Switch (deyim) {
        case sabit0 : ifadeler ; break;
        case sabit1 : ifadeler ; break;
        case sabit2 : ifadeler ; break;
```

```
        default : ifadeler ;
{
```

Döngü Yapıları

Program yazarken bazı kodların birden fazla çalıştırılmasına gereksinim duyarız. Döngü yapıları bu kodların birden fazla çalıştırılması için kullanılan yapılardır. 3 tür döngü bulunmaktadır.

While

While döngüsü kendisine verilen koşul doğru olduğu sürece döngüyü çalıştırır. Döngüye girilmeden önce koşul kontrol edilir, TRUE ise döngü çalıştırılır, yanlış ise döngüden sonraki satırdan devam edilir.

```
while ( koşul ifadesi ) {
        çalıştırılacak komut satırları;
}
```

Do-While

do-while döngüsü while döngüsünün bir çeşididir. Tek farkı koşulun doğruluğu döngü sonunda yapılır. Yani koşul yanlış olsa dahi bir kere döngü bloğu çalıştırılır.

```
do  {
        çalıştırılacak komut satırları;
}  while ( koşul );
```

For

En çok kullanılan döngü türüdür. Önceki döngü türlerinde döngü sayısı için bir sayaç değişkeni kullanırız. Bu sayacı arttırarak veya azaltarak döngü koşulunu sürekli kontrol ederiz. Oysaki bu işlem For döngüsünde oldukça basittir.

```
for ( sayacın başlangıç değeri; koşul ifadesi; sayacın artış miktarı ) {
        çalıştırılacak komut satırları ;
}
```

Bir döngü içerisinde break deyimine rastlanıldığında döngünün dışına çıkılır. continue ifadesi döngü başlangıcına dönmekte kullanılır.

```
while ( .. ) {                  do  {                   for ( .. ; .. ; .. ) {
        ....                            …                       …
        ....                            …                       …
}                               while ( .. );           }
```

Fonksiyon ve Diziler

Fonksiyonlar, giriş parametrelerine göre bazı görevleri gerçekleştiren alt program parçacıklarıdır. Bir işlemi program içerisinde birden fazla yapacaksak, bu durumda aynı kodları tekrar yazmaktansa, bu işlem bir fonksiyona çevrilebilir. Böylece ihtiyacımız olduğunda çağırabiliriz. Fonksiyon içinde işlemler yapıldıktan sonra

return komutuyla ana programa bir değer döndürülür. C'de ana program main() fonksiyon bloğunun içine yazılır. Diğer fonksiyonlar bu ana programın içinden çağrılarak kullanılır.

```
Fonksiyonun geriye döndüreceği tür    Fonksiyonun adı  (parametre listesi);

void main() {

        int temp_change ( int x,   int y ) {

                …

                …

         return …

        }

}
```

Bir fonksiyon tanımlanırken, bu fonksiyon içerisinde oluşan sonucun (return ile) geriye döndürülemeyeceği durumlarda Void kullanılır.

```
 Void pressure_change (char pressure) {

        Lcd_Out_Cp (" Pressure : ");

        Lcd_Out_Cp (pressure);

}
```

Diziler, C dilinde sıkça kullanılan yöntemlerden biridir. Diziler sayesinde aynı türdeki birden fazla veriyi bir değişken dizisinin içine depolayabiliriz.

```
int dizi_adı [indeks] = { Dizinin elemanları aralarına virgül konarak yazılır }

unsigned short sigortalar [9] = { 6, 10, 16, 20, 25, 32, 40, 50, 63 }
```

Dizilerin ilk elemanının indeks değeri her zaman için "0" dır. sigortalar[0] ile 6 değerine, sigortalar[8] ile 63 değerine ulaşabiliriz.

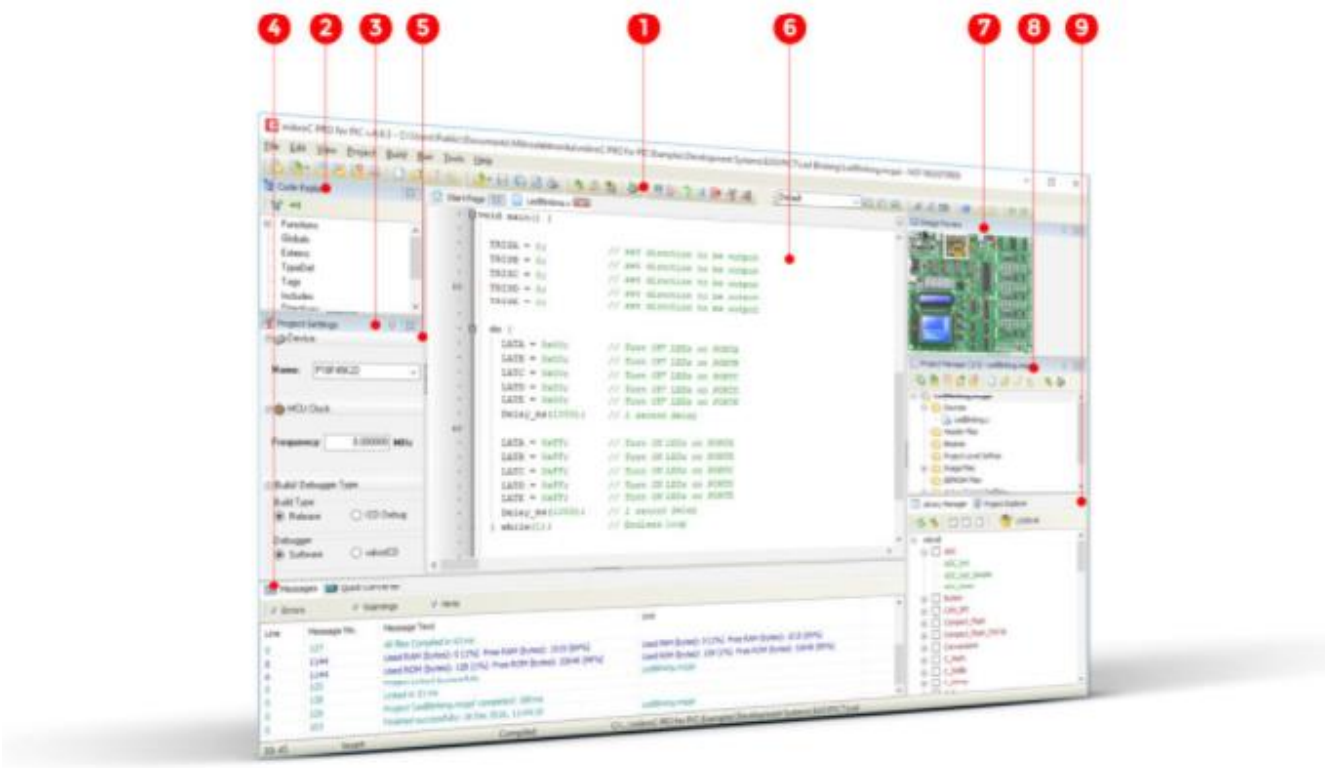
Önişlemci Komutları

Önişlemci derleme işleminin bir parçasıdır. Derlemeye başlamadan önce önişlemci komutlar çalıştırılırlar. # işareti ile başlayan satırlar önişlemci komutlardır. Önişlemci komutları kullanarak programımıza dışarıdan bir dosya ekleyebiliriz. Bunun için #include komutu kullanılır. Ayrıca önişlemci kullanarak fonksiyonlar gibi makrolar tanımlayıp programlarda kullanabiliriz. Makro oluşturmak için #define önişlemci komutlarını kullanırız.

```
#include <file_name>

#include "built_in.h"

#define   servo1    portd.rd0
```

MikroC Compiler and Projec Organizasyonu

MikroC Pro derleyicisi MikroElektronika firması tarafından üretilmektedir. MikroC Pro yalnızca derleyici olmamakla beraber; miktodenetleyici kodlarının yazılabildiği bir alan, bu kodlarla ilgili her türlü bilgiyi alabileceğiniz pencereler, yardımcı programlar (USART Terminal, HID Terminal, GLCD Bitmap Editor vb) ve çok zengin bir kütüphaneyi bizlere sunmaktadır. MikroC kütüphanesi hem öğretici, hem de kullanımı kolaydır.



1. Ana Araç Çubuğu

Varsayılan düzende, Ana Araç Çubuğunun ilk bölümü projenin oluşturulmasına, düzenlenmesine ve silinmesine ayrılmıştır. İkinci bölüm dosya eklemek içindir.

Üçüncüsü, kaydetme seçenekleri ve yazdırma ile ilgilidir. Dördüncüsü, projenizi oluşturmak ve programlayıcıyı başlatmak içindir.

Beşincisi USART terminalini, EEPROM editör aracını ve diğer seçenekleri yönetmek içindir. Altıncısı düzen özellikleri içindir.

Yedinci, montaj, listeler ve istatistikler içindir. Sekiz, Yardım dosyası ve örnekler klasörü içindir.

Dokuzuncu ve son bölüm, Kod Düzenleyici'deki adımlarınızı yeniden izlemek için Geçmiş'tir.

2. Kod Gezgini

Kod Gezgini, ekranın sol üst köşesinde bulunur. Açtığınız projede fonksiyonların, web bağlantılarının ve aktif yorumların bir listesini görebilirsiniz.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union

3. Proje Ayarları

Bu bölümde kullandığınız cihazın adı, MCU saatinin frekansı ve Build/Debugger türleri bulunur.

MCU saatinin frekansı, mikrodenetleyicinin hızını belirler.

4. Mesajlar

Derleme sırasında hatalarla karşılaşılması durumunda, derleyici bunları Mesaj kutusuna bildirir ve bir hex dosyası oluşturmaz.

Derleyici ayrıca uyarılar da bildirir, ancak bunlar çıktıyı etkilemez; yalnızca hatalar bir altıgen oluşturulmasına müdahale edebilir.

5. Hızlı Dönüştürücü

Bir formdan diğerine çeviri yapmayı kolaylaştırır. Örneğin, bir ondalık sayıdan ikili sayıya.

6. Kod düzenleyici

Kod Düzenleyici, yaygın yazım hataları için ayarlanabilir Sözdizimi Vurgulama, Kod Katlama, Kod Yardımcısı, Parametre Yardımcısı, Otomatik Düzeltme özelliklerine sahiptir.

ve Kod Şablonları (Otomatik Tamamlama).

7. Görüntü Önizleme

Görüntü Önizleme kutusu, varsayılan olarak ekranın sağ üst köşesinde bulunur. Kodun yorumlar kısmında eklemiş olduğunuz aktif linklerin resimlerini gösterir.

8. Proje Yöneticisi

Proje Yöneticisi, birden çok projeyi yönetmenize izin veren bir IDE özelliğidir. Her projede kaynak ve başlık dosyalarını gösterir.

Aynı anda birkaç proje açık olabilir, ancak aynı anda sadece biri aktif olabilir.

Bir projeyi aktif modda ayarlamak için Proje Yöneticisinde istediğiniz projeye çift tıklamanız gerekir.

9. Kütüphane Yöneticisi

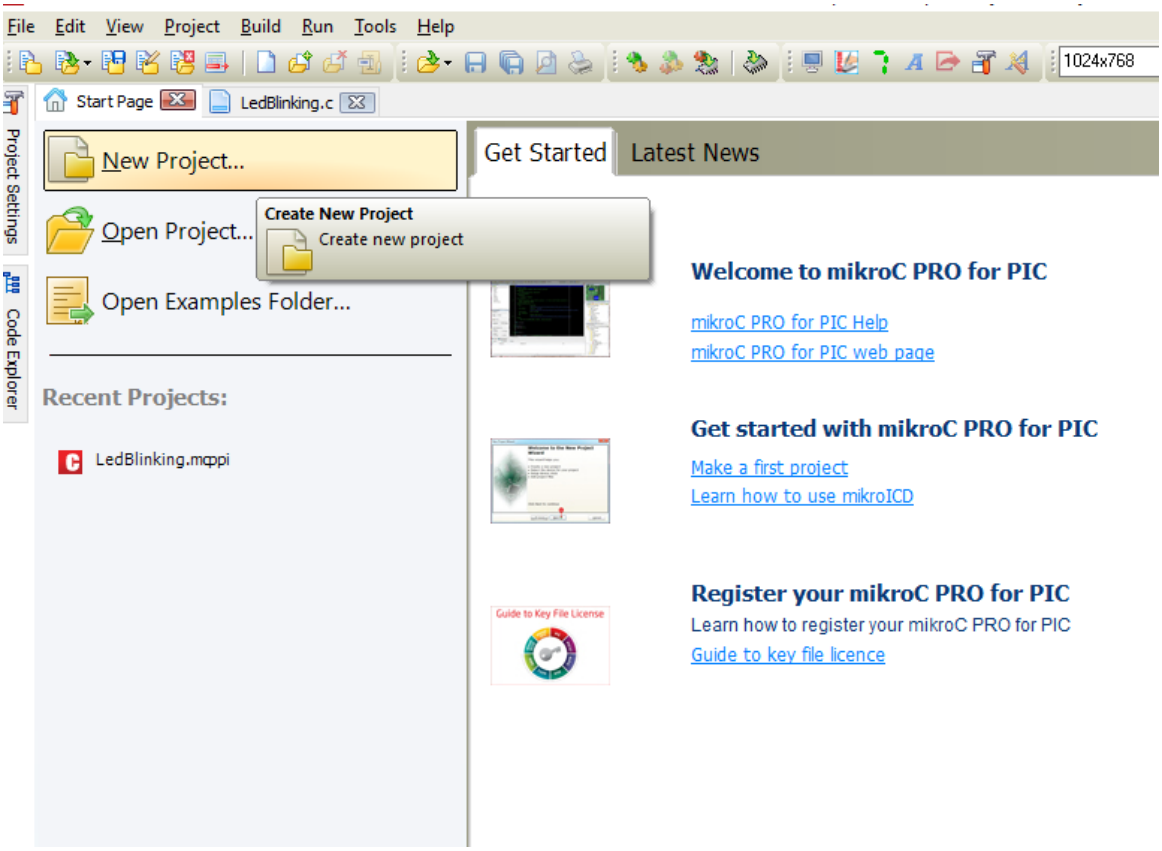
Kütüphane Yöneticisi, kütüphanelerle basit ve kolay bir şekilde çalışmanıza izin verir. Kitaplık Yöneticisi penceresi tüm kitaplıkları listeler.

İstenilen kitaplık, kitaplık adının yanındaki onay kutusu seçilerek projeye eklenir.

Tüm kitaplık işlevlerinin kullanılabilir olması için Tümünü Kontrol Et düğmesine basmanız yeterlidir, hepsi bu kadar.

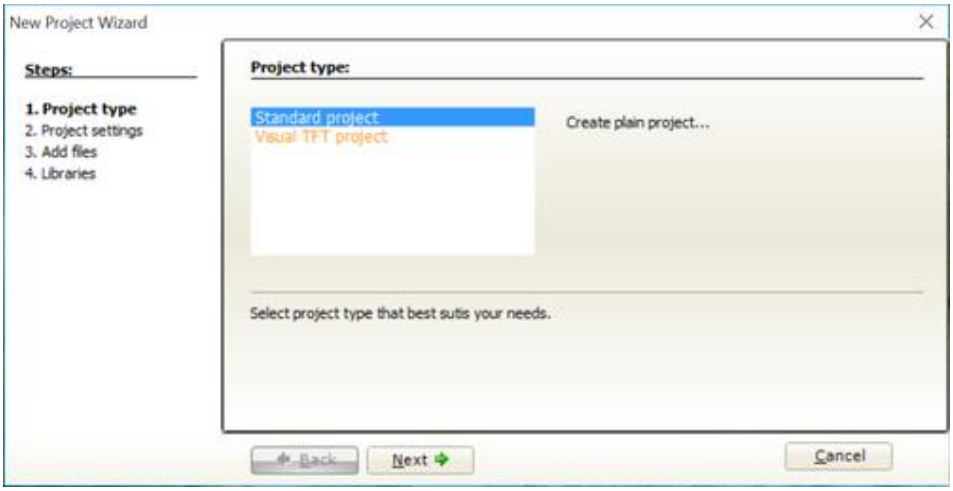
İlk Projenizin Oluşturulması

Bu bölümde, derleyicide ilk projenizi nasıl oluşturacağınızı ve derleyip sonuçları nasıl test edeceğinizi göstereceğiz. Öncelikle MikroC Pro derleyicisini bilgisayarınıza kurmuş olmalısınız. Derleyiciyi açtığınızda Project menüsünden New Project seçeneğini seçin.

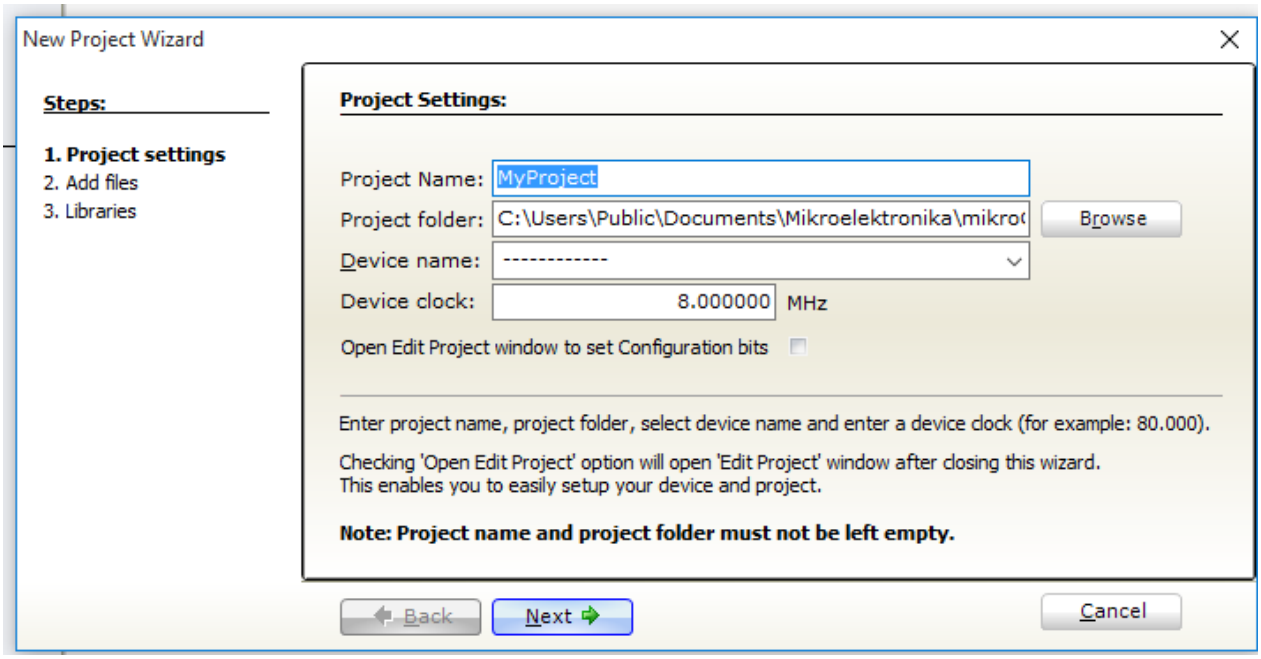


Yeni Proje Sihirbazı penceresi görünecek ve projenizi oluşturma sürecinde size kolayca rehberlik edecektir.

PIC, dsPIC veya FT90x. derleyicilerden birini kullanıyorsanız, Visual TFT projesi mi yoksa Standart proje mi oluşturacağınızı seçmenizi isteyen yeni bir pencere açılacaktır.



İlk yapmamız gereken genel proje bilgilerini belirtmektir. Hedef mikrodenetleyiciyi, çalışma frekansını ve projenizin adını seçin.



Derleyici, girdiğiniz bilgilere göre dahili ayarları yapacaktır. Başlangıçta sizin için varsayılan yapılandırma önerilir.

Yeni projenizi saklamak için önerilen yolu kullanmak istemiyorsanız, hedef klasörü değiştirebilirsiniz.

Üzerinde çalışacağınız projenin türüne göre projenin adını girin. Cihaz saatini ayarlayın. Saat hızı, hedef donanımınıza bağlıdır. Mikrodenetleyicinin üzerinde çalıştığı tam saati belirttiğinizden emin olun.

Diğer pencerede, eğer kullanacağımız elektronik parçalar kütüphanelerde bulunmuyorsa (nadir durum), Add butonuna basarak dosyaları projemize dahil edebiliriz.

Bu adım, projenize tüm kitaplıkları dahil etmek isteyip istemediğinizi hızlı bir şekilde ayarlamanıza olanak tanır.

Tüm kütüphaneler dahil olsa bile, herhangi bir bellek tüketmezler.

bunlar açıkça kodunuzun içinden kullanılır. Tüm kitaplıkları dahil etmenin ana avantajı, kodunuzda hemen kullanabileceğiniz 500'den fazla işleve sahip olmanızdır. Bu işlevler Code Assistant [CTRL+Space]'den görülebilir. Varsayılan yapılandırma "Tümünü Dahil Et" seçeneğidir.

Bitirdiğinizde Bitir düğmesine tıklayın.

Artık ekranda kodumuzu yazacağımız pencere görülmektedir. Yazdığımız programı direkt mikrodenetleyiciye yükleyemeyiz. Derleyip, .hex kodunu oluşturmalıyız. Bunun için bazı kısayollar önerilir:

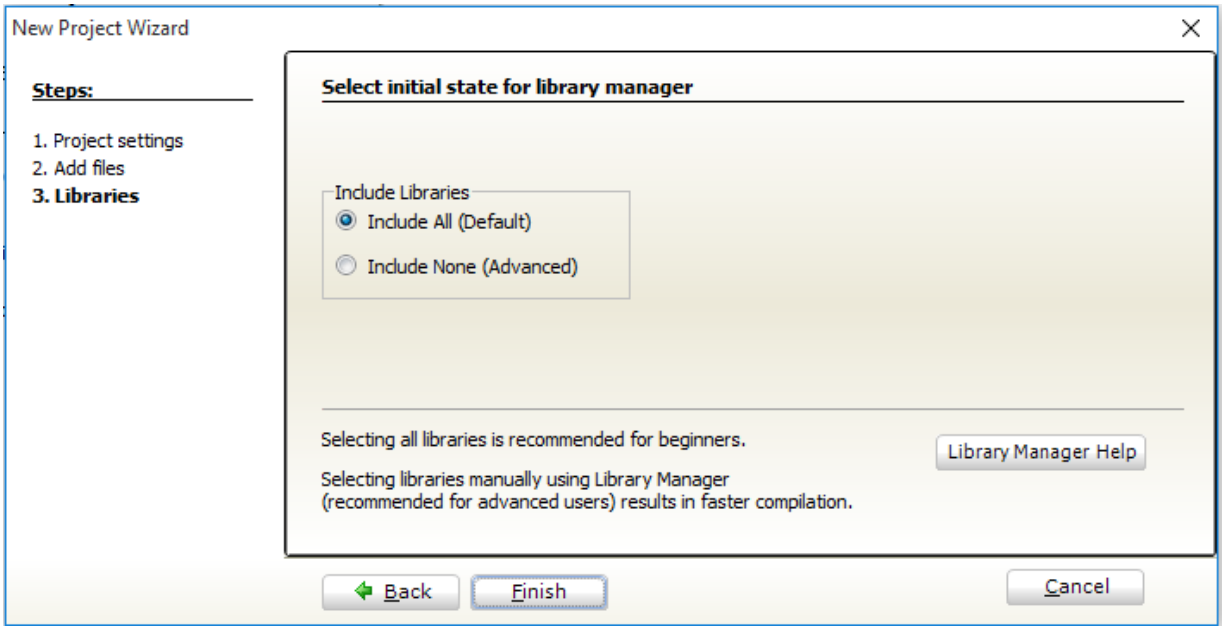
Ctrl+Alt+G to Kod oluşturma

Ctrl + Shift + S Projeyi kaydetme

Ctrl + F9 Kodları derleme veya Project Menüsü altında Build seçeneği

Ctrl + F11 Kodları derleme ve MCU'ya atma

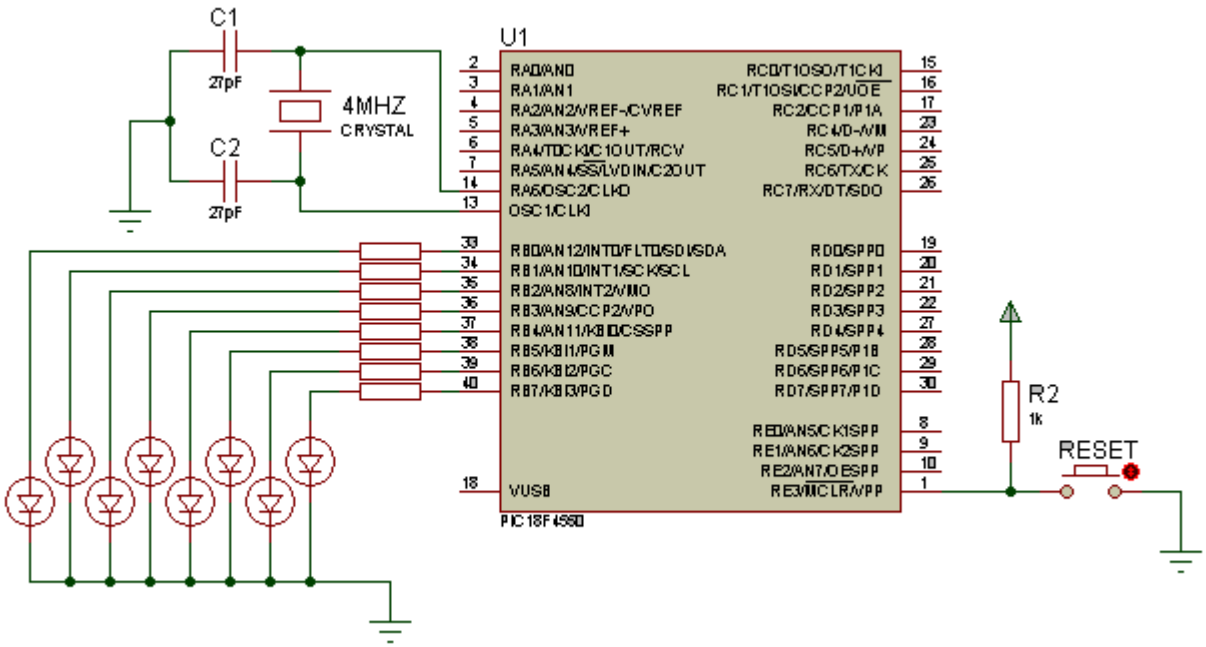
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union



Derleme işlemi sonrasında derleyici projenin oluşturulduğu klasör içerisine .asm, .hex, .mcl ve .lst uzantılı dosyaları oluşturur.

TEMRİN 1

Bu uygulamada PIC18F4550 mikrodenetleyicisinin portlarını çıkış olarak kullanacağız.



Uygulamanın devre şeması şekilde verilmiştir. Lütfen Proteus programında devreyi kurunuz. MikroC programında programı derlemeden önce konfigürasyon ayarların da osilatör seçimini yapmayı unutmayınız. Devrede osilatör olarak kristal kullandığımız için Project- Edit Project seçeneğini çalıştırarak karşımıza gelen pencereden, “XT oscillator” seçilmelidir.

Uygulama 1

B portunun RB0 pinine bağlı bulunan bir LED diyotun program çalışmaya başladıktan 5sn sonra yakılması.

```
void main () {
        trisb.rb0 = 0 ;
        portb.rb0 = 0 ;
        delay_ms (5000) ;
        portb.rb0 = 1 ;
}
```

Uygulama 2

B portunun RB0 pinine bağlı bulunan bir LED diyotun 1 sn aralıklarla yakılıp, söndürülmesi.

```
void main () {
        trisb.rb0 = 0 ;
        portb.rb0 = 0 ;
        while (1)
        {
```

```
                portb.rb0 = 0 ;
                delay_ms (1000) ;
                portb.rb0 = 1 ;
                delay_ms (1000) ;
        }
}
```

Uygulama 3

PORTB portuna bağlı bulunan 8 LED diyotun 1 sn aralıklarla yakılıp, söndürülmesi.

```
void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;

        trisb = 0 ;
        portb = 0 ;
        while (1)
        {
                portb = ~portb ;
                delay_ms (1000) ;
        }
}
```

Uygulama 4

PORTB portuna bağlı bulunan 8 LED diyot ile 0,5sn aralıklarla 255 sayısına kadar binary olarak yukarı ve aşağı sayıcı uygulaması.

```
void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        trisb = 0 ;
        portb = 0 ;
        while (1)
        {
                do {
                        portb++ ;
                        delay_ms (500) ;
                } while ( portb < 255 ) ;
                delay_ms (2000) ;
                 do {
                        portb-- ;
                        delay_ms (500) ;
                } while ( portb > 0 ) ;
                delay_ms (2000) ;
        }
}
```

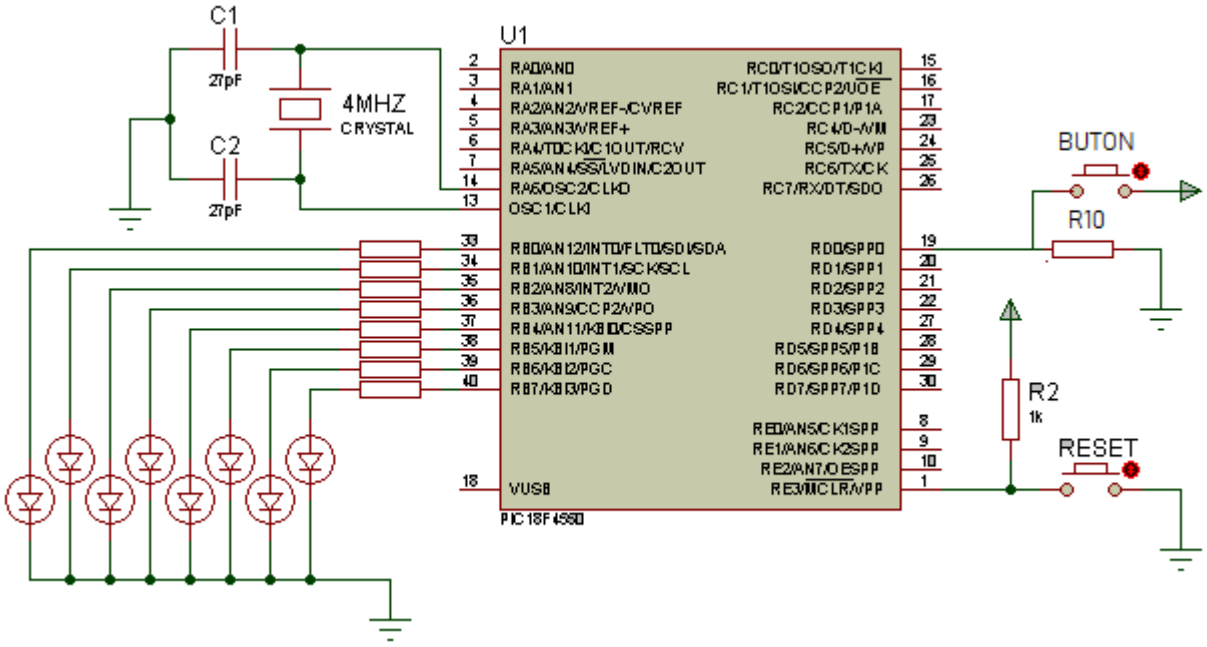
Uygulama 5

PORTB portuna bağlı bulunan 8 LED diyodun sırayla yanıp sönmesiyle oluşturulan uygulama.

```
void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        trisb = 0 ;
        portb = 0 × 01 ;
        while (1)
        {
                while ( portb < 0 × 80 )  {
                        delay_ms (200) ;
                        portb = portb << 1 ;
                }
                while ( portb > 0 × 01 ) {
                        delay_ms (200) ;
                        portb = portb >> 1 ;
                }
        }
}
```

TEMRİN 2

Bu uygulamada PIC18F4550 mikrodenetleyicisinin portlarını giriş olarak kullanacağız.



D portunun RD0 pinine bağlanmış olan BUTON ile RB portuna bağlanmış LED diyotların kontrolü sağlanmıştır. RD0 butonuna bağlı bulunan buton bir pull-down direnci ile basıldığında Lojik-1 verecek şekilde tasarlanmıştır. Programda B portunun tamamı çıkış, D portunun RD0 pini ise giriş olarak yönlendirilmiştir.

Uygulama 1

Butona basıldığında PORTB portuna bağlı bulunan 8 LED diyoda durum değiştiren uygulama.

```
void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        trisb = 0 × 00 ;
        portb = 0 × 0F ;
        trisd.rd0 = 1 ;
        portd.rd0 = 0 ;
        while (1)
        {
                if ( portd.rd0 )  {
                        portb = ~portb ;
                        while (portd.rd0 ) ;
                }
        }
}
```

Uygulama 2

Butona her basıldığında PORTB'nin değeri bir artırılmaktadır. Yani binary yukarı sayıcı uygulaması.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union

```
void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        trisb = 0 ;
        portb = 0 ;
        trisd.rd0 = 1 ;
        portd.rd0 = 0 ;
        while (1)
        {
                if ( portd.rd0 )  {
                        portb++ ;
                        while (portd.rd0 ) ;
                }
        }
}
```

Uygulama 3

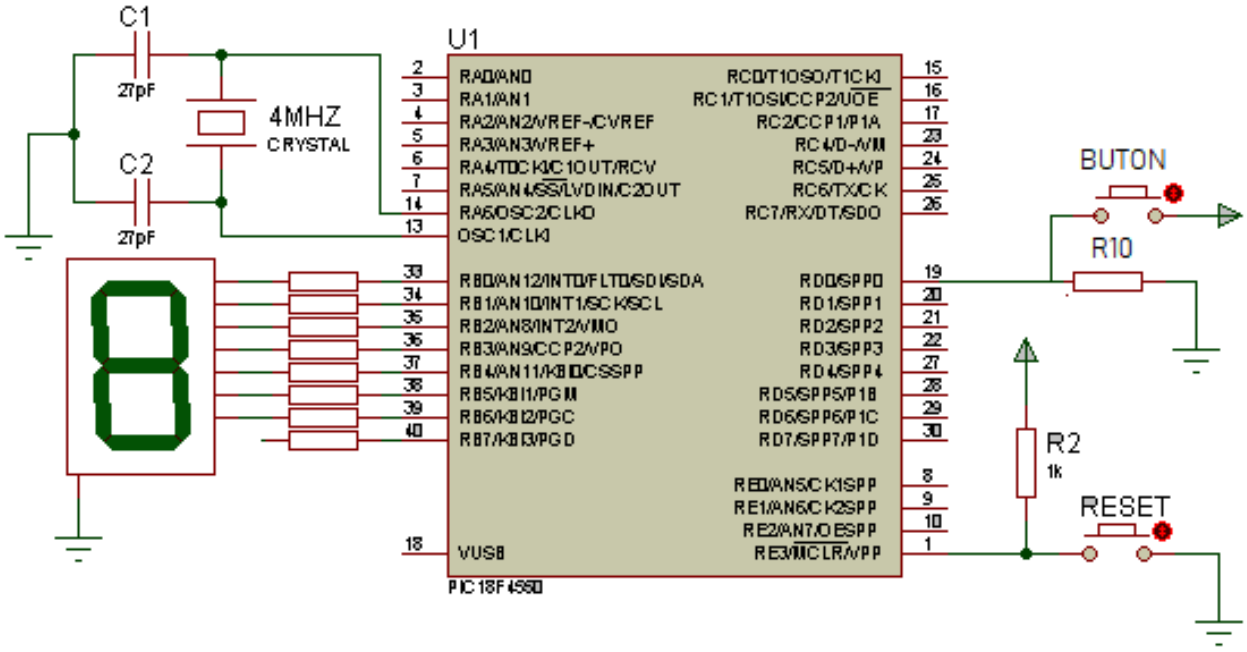
PORTB'ye 1 değeri yüklenmiş ve butona her basıldığında PORTB' deki değer bir sola kaydıran uygulama.

```
void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        trisb = 0 ;
        portb = 0 ;
        trisd.rd0 = 1 ;
        portd.rd0 = 0 ;
        while (1)
        {
                if ( portd.rd0 )  {
                        portb <<= 1 ;
                        while (portd.rd0 ) ;
                }
        }
}
```

while (portd.rd0) ; satırı ile de, RD0 pini lojik 1 olduğu sürece boş döngü oluşturulmaktadır. Bunun nedeni ise 200 ms olmasındandır. Böylece kontak sıçraması engellenmektedir.

TEMRİN 3

Bu uygulamada 7 Segment Display ile 0-9 sayıcı uygulaması yapacağız.



Bu uygulamada ortak katotlu 7 segment display PORTB'ye bağlanmıştır. Pinlerin bağlantısı aşağıdadır.

NUMBER	null	g	f	e	d	c	b	a	BIN	HEX
	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0		
0	0	0	1	1	1	1	1	1	0b00111111	0x3F
1	0	0	0	0	0	1	1	0	0b00000110	0x06
2	0	1	0	1	1	0	1	1	0b01011011	0x5B
3	0	1	0	0	1	1	1	1	0b01001111	0x4F
4	0	1	1	0	0	1	1	0	0b01100110	0x66
5	0	1	1	0	1	1	0	1	0b01101101	0x6D
6	0	1	1	1	1	1	0	1	0b01111101	0x7D
7	0	0	0	0	0	1	1	1	0b00000111	0x07
8	0	1	1	1	1	1	1	1	0b01111111	0x7F
9	0	1	1	0	1	1	1	1	0b01101111	0x6F

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union

Uygulama 1

Tablodaki veriler MikroC programında; dizi şeklinde kullanılacak, belirlenen bir süre içerisinde ve ardı ardına PORTB'ye gönderilecektir. const char display [10] satırı ile displaye gönderilecek olan veriler hafızada sabit karakter tipinde dizi olarak tanımlanmıştır.

```
const char display [10] = {     0b00111111,
                                0b00000110,
                                0b01011011,
                                0b01001111,
                                0b01100110,
                                0b01101101,
                                0b01111101,
                                0b00000111,
                                0b01111111,
                                0b01101111   } ;
unsigned short i = 0;

void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        trisb = 0 ;
        portb = 1 ;
        while (1)
        {
                portb = display [ i ] ;
                i++ ;
                delay_ms (500 ) ;
                if ( i > 9 ) i = 0 ;
        }
}
```

Uygulama 2

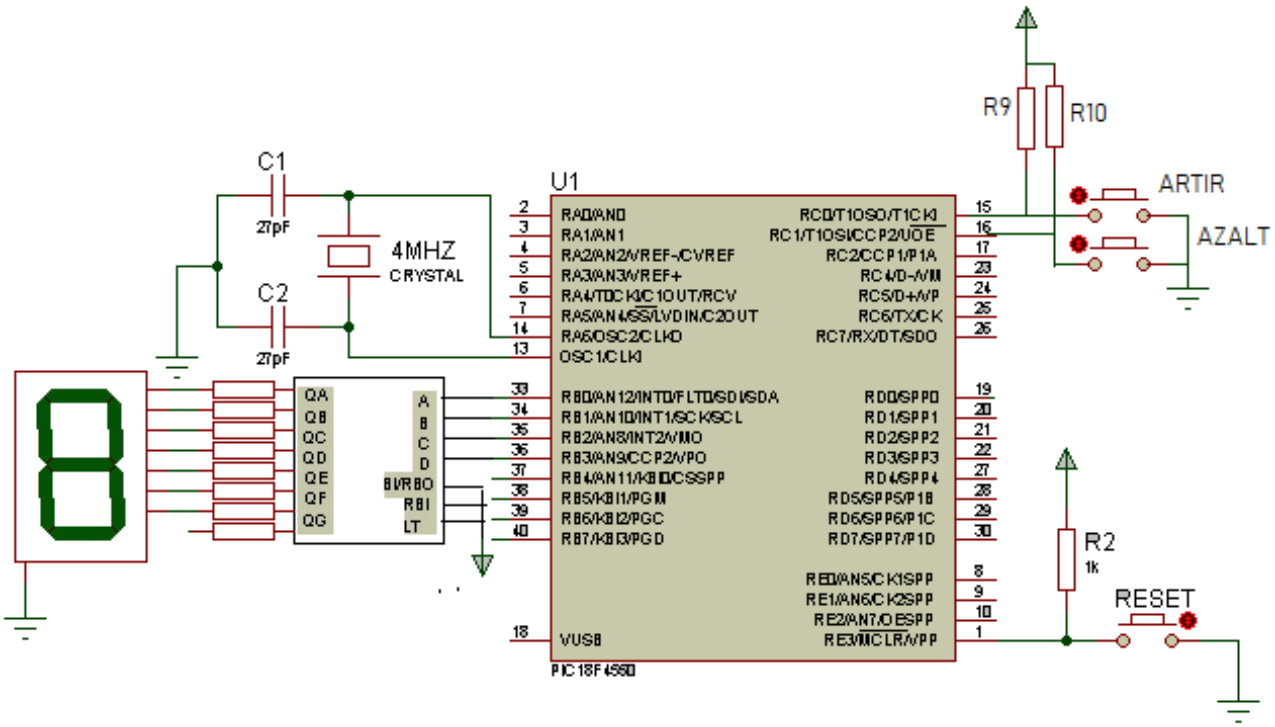
0-9 yukarı ve ardından 9-0 arası aşağı sayıcı uygulaması.

```
const char display [10] = { 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F } ;
unsigned short i = 9;

void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        trisb = 0 ;
        portb = 1 ;
        while (1)
        {
                for ( i = 0; i < 10 ; i++ )
                {
                        portb = display [ i ] ;
```

```
                delay_ms (500 ) ;
        }
        for ( i = 10; i > 0 ; i-- )
        {
                portb = display [ i - 1] ;
                delay_ms (500 ) ;
        }

    }
}
```

Uygulama 3

Bu uygulamada C portunun RC0 ile RC1 pinlerine bağlanan butonlar ile, B portuna bağlı bulunan 7448 ortak katot display sürücü entegresi sayesinde 7 segment display aşağı-yukarı sayıcı olarak kullanılacaktır. 7448 entegresi ile mikrodenetleyicinin daha az pini kullanılmaktadır. Ayrıca MikroC kodları daha basit haldedir.

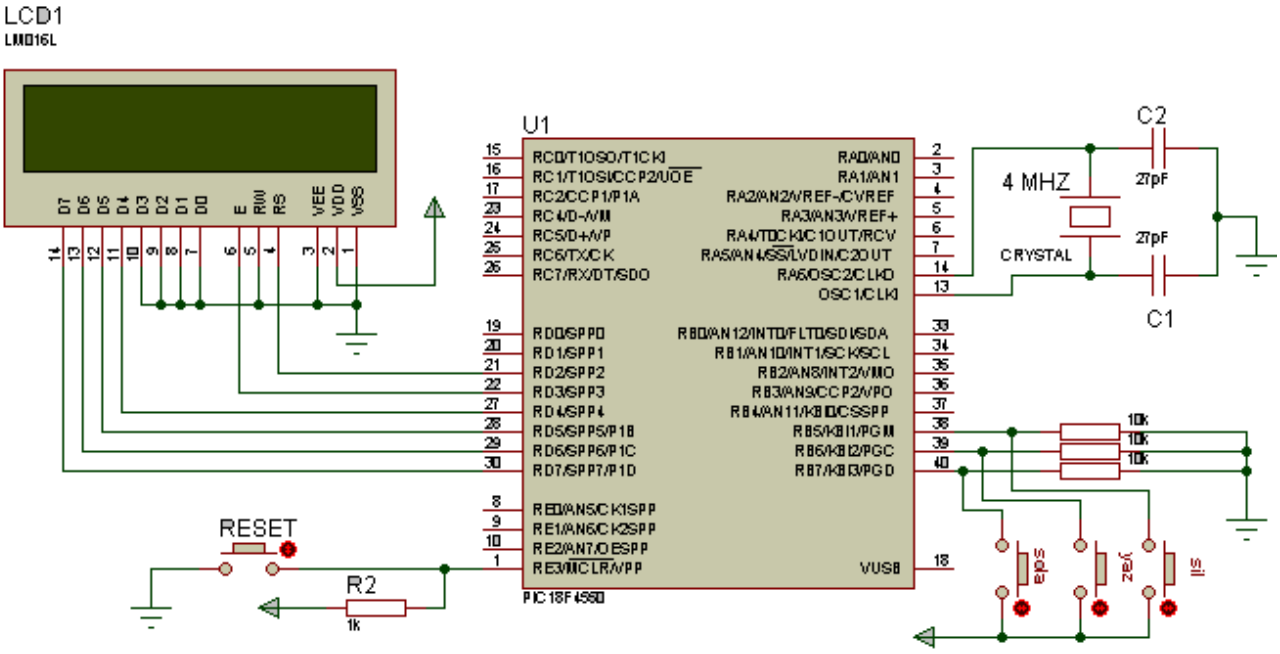



```
#define artir     PORTC.RC0
#define azalt     PORTC.RC1
short  display = 0 ;

void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        trisb = 0 ;
        portb = 0 ;
        trisc = 0 ×03 ;
        portc = 0 ;
```

```
        while ( 1)
        {
                if  (!artir)
                {
                        display++ ;
                        while (!artir) ;
                }
                if (!azalt)
                {
                        display-- ;
                        while (!azalt) ;
                }
                if ( (display > 9 ) | (display < 0) ) display  = 0 ;
                portb = display ;
        }
}
```

TEMRİN 4

Bu uygulamada piyasada oldukça bol kullanılan HD44780 işlemcili LCD uygulaması yapacağız.



Şemada görülen bağlantılar MikroC kodlarında düzgün tanımlanmalıdır. Giriş-çıkışlara dikkat edilmelidir. Ayrıca bu programda sonsuz döngü kullanılmamıştır. Çünkü, LCD'ye gönderilen veriler, LCD RAM'inde enerji olduğu sürece kalır. Derleme esnasında LCD kütüphanesini seçmeyi unutmayın.

```
#define sil     portb.rb5
#define yaz     portb.rb6
#define sola    portb.rb7

sbit LCD_RS at RD2_bit ;
sbit LCD_EN at RD3_bit ;
sbit LCD_D4 at RD4_bit ;
sbit LCD_D5 at RD5_bit ;
sbit LCD_D6 at RD6_bit ;
sbit LCD_D7 at RD7_bit ;
sbit LCD_RS_Direction at TRISD2_bit ;
sbit LCD_EN_Direction at TRISD3_bit ;
sbit LCD_D4_Direction at TRISD4_bit ;
sbit LCD_D5_Direction at TRISD5_bit ;
sbit LCD_D6_Direction at TRISD6_bit ;
sbit LCD_D7_Direction at TRISD7_bit ;
void main () {
        ADCON1 l= 0×0F ;
```

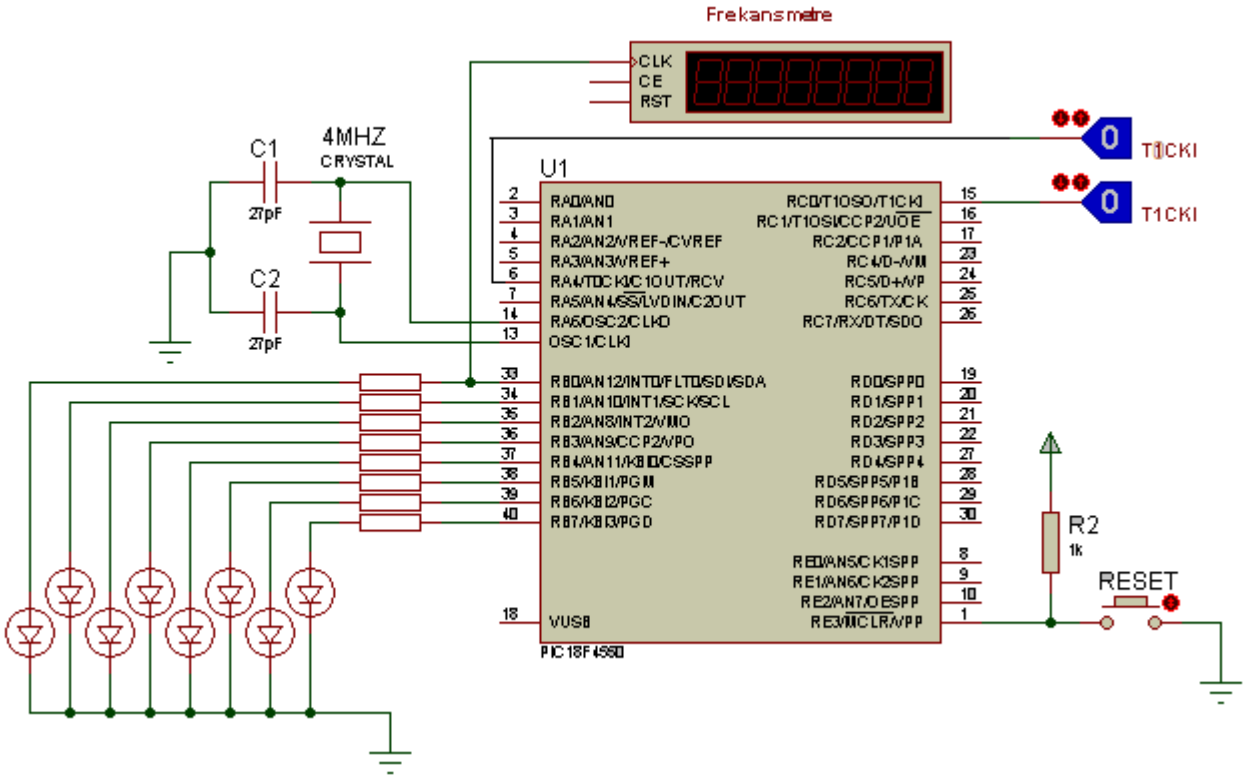
```
        CMCON |= 0×07 ;
        Lcd_Init () ;
        Lcd_Cmd (_LCD_CURSOR_OFF) ;
        Lcd_Cmd(_LCD_CLEAR) ;
        Lcd_Out (1, 3,  “Erasmus+”) ;
        Lcd_Out (2,  1, “IQinDIGIT”= ;
        For ( ; ; )
        {
                if (sil) LCD_Cmd ( _LCD_CLEAR) ;
                if (yaz) {
                        Lcd_Out (1,  2,  “Merhaba” ) ;
                        Lcd_Out (2,  2,  “ Dunya” ) ;
                }
                if (sola) {
                        LCD_Cmd (_LCD_SHIFT_RIGHT) ;
                        while (sola) ;
                }
        }
}
```

TEMRİN 5

Mikrodenetleyicilerin en önemli özelliği dahili donanım zamanlayıcılarının olmasıdır. Bunlara TIMER denir. TIMER zamanlayıcı veya sayıcı olarak kullanılabilir. Mikrokontrolcü modeline göre TIMER sayıları değişebilir. Her TIMER ortaya çıkan durumlara karşı KESME üretebilir. Bizim kullandığımız mikrokontrolcüde 4 adet TIMER vardır. Biz burada Timer0 modülünü kullanacağız. Bu modül 8/16 bit olarak çalışabilir. 8 bit yazılımla prescaler, saat kaynağı seçimi, harici clock sinyali kenar seçimi, kesme oluşturma gibi özellikleri vardır. Timer0 donanım birimi işlemlerini yapılandırmak için T0CON kaydedicisine sahiptir.

Uygulama 1

Timer0'ın zamanlayıcı olarak kullanılması uygulamasıdır. 8 bit olarak kullanılacaktır. Timer0 her 5017,6 µs'debir kesme okuşturacak şekilde ayarlanmıştır. 20 Mhz'lik kristal osilatör kullanılmaktadır. Kesme zamanı prescaler oranına ve TMR0L'a yüklenen ön değere bağlıdır.




```
unsigned cnt;
void interrupt () {
        if (TMR0 IF_bit) {
                cnt++;
                TMR0L = 158;
                TMR0 IF_bit = 0 ;
        }
}
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union

```
void main () {
        ADCON1 l= 0×0F ;
        CMCON l= 7 ;
        cnt = 0 ;
        TRISB = 0 ;
        PORTB = 0×FF ;
        T0CON = 0×C7 ;
        INTCON = 0×C0 ;
        TMR0IE_bit = 1 ;
        do {
                if (cnt >= 100 ) {
                PORTB = ~PORTB ;
                cnt = 0 ;
                }
        } while (1) ;
}
```

$f_{command}$= Microcontroller Oscillator Frekansı / 4 = 20 Mhz / 4 = 5 Mhz

$T_{command} = 1 / f_{komut} = 1 / 5Mhz = 0,2 \times 10^{-6} = 0,2\ \mu s$

Kesme Zamanı = $T_{command}$ × Prescaler rate ×(256 – TMR0 default value)
= 0,2 µs × 256 × (256 -158)
= 51.2 µs ×98
= 5017,6 µs

PORTB Konum Değiştirme Zamanı = Kesme Zamanı × 100
= 5,0176ms × 100
= 501,76 ms

Kaynakçalar:

MikroC PRO Complier Quick Start Guide

MikroC and PIC18F4550, Hikmet ŞAHİN, K.Serkan DEDEOĞLU

https://www.microchip.com/

https://www.mikroe.com/

https://www.labcenter.com

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the Erasmus+ Programme of the European Union

KATKI SUNANLAR

Samuel BRANCO

Michal CIERNIAK

Oktay CELTIK

Hakan ESER

Antonio FERREIRA

Antonio GAGLIOSTRO

Alessandro LOMBINO

Genesio MINICHIELLO

Ozay OZCAN

Grzegorz STOZEK

Co-funded by the Erasmus+ Programme of the European Union

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Co-funded by the
Erasmus+ Programme
of the European Union