beti®

Mikrodenetleyici Eğitim Seti Kullanım ve Deney Kitabı

Dünyada en çok kullanılan MikroC Derleyici kodları ile

NİSAN-2023

MİKRODENETLEYİCİ UYGULAMA VE GELİŞTİRME SETİ

KULLANIM VE DENEY KİTABI

Yazarlar:

Dr. F. Zeynep KÖKSAL
Ph.D. in EEE, ODTÜ/1990

Emrah İŞSEVER
G.Ü.T.E.F./1998

Oğuz ŞENGÖZ
B.Sc. EEE, ODTÜ/2014

Son Güncelleme
Mehmet Ali DAL
Elektronik Teknik Öğretmeni

BETİ ELEKTRONİK SAN. ve TİC. LTD. ŞTİ.
NİSAN 2023

Editör

Nazlı Elif KÖKSAL
MS in Economics
Université Paris I-Panthéon-Sorbonne/2002

İsteme Adresi:
Beti Elektronik San. ve Tic. Ltd. Şti.
Cevizlidere Mah. 1244 Sokak No:8/B Çankaya / Ankara
Tel: 0 312 2221800
info@beti.com.tr

on-line sipariş www.elektrovadi.com

ÖNSÖZ

Türkiyemizi, Türk mühendislerini, teknikerlerini, teknisyenlerini, Teknik Üniversite, Meslek Yüksek Okulu, Teknik Lise ve Meslek Lisesi öğrencilerini, özetle Elektronik sahasında Teknolojik Gelişme heyecanını bizimle birlikte takip eden tüm meslektaşlarımızı Teknolojik Eğitim Setleri ile tanıştırmaya devam ediyoruz. 2001 yılında satışa sunduğumuz Microchip'in PIC Mikrodenetleyicilerini baz alan PICuse/84'den sonra şimdi de Beti MCU isimli özgün setimiz; çeşitli uygulama kartları, C Compiler Yazılımı ve dokümantasyonu ile ülkemiz Teknik eğitim ve öğretiminin kullanımına sunulmaktadır.

Herkes tarafından tescillenen Mikrodenetleyici Eğitim Setleri konusundaki uzmanlığımız, bizleri yeni konu başlıklarına yöneltmekte ve "İleri Teknoloji Eğitim Setleri" imalatı konusundaki misyonumuzu geliştirerek devam ettirmemize motive ettirmektedir. Bilimsel ve akademik yaklaşımımızla; tüm eğitim setlerimizde pahalı ve sofistike sistemler yerine, ucuz, kullanımı kolay ve öğrenci-sevecen donanım ve yazılım geliştirme alt yapısı oluşturmaktayız. Bu kurguya haiz setlerimizin kullanımının yaygınlaşması ülkemizdeki teknik eğitimin toplam kalitesini global boyutlara taşıyacaktır. Bizi izlemeye devam ediniz.
Saygılarımızla

Dr. F. Zeynep KÖKSAL

İÇİNDEKİLER

DENEY 1. MİKRODENETLEYİCİDE VERİ ÇIKIŞI UYGULAMASI

AMAÇ:

1) Mikrodenetleyici'nin portlarını öğretmek,
2) Mikrodenetleyici'ye veri alışverişini öğretmek.

GEREKLİ MALZEMELER:

1) EasyPIC7 Kartı

Giriş:

Giriş/Çıkış'lar (G/Ç), Mikrodenetleyici'nin dış dünya ile ilişkisini sağlayan, giriş ve çıkış şeklinde ayarlanabilen bağlantı pinleridir. G/Ç'ler çoğunlukla Mikrodenetleyici'nin iletişim kurmasına, kontrol etmesine veya bilgi okuması amacıyla kullanılır. PIC18F45K22 Mikrodenetleyici'sinde 35 adet G/Ç pini mevcuttur. Bu pinlerinin adları RA0-RA7, RB0-RB7, RC0-RC7, RD0-RD7, RE0-RE3 şeklindedir. Bu 36 pinin 35 adedi giriş ya da çıkış olarak kullanılabilirken, RE3 pini sadece giriş olarak kullanılabilir. Bu sebeple "PIC18F45K22 Mikrodenetleyici'si 35 adet G/Ç'den oluşur" denilir. PIC içerisinde TRIS adı verilen kaydedici sayesinde portların giriş veya çıkış olup olmadığı belirlenir.

TRIS Kaydedicisi

```
TRISB=0;      //B portunun tüm pinlerini çıkış olarak ayarlar

TRISA=0x0F; //A portunun ilk 4-bit'ini (RA0, RA1, RA2, RA3) giriş olarak,
              //son 4 bit'ini (RA4, RA5, RA6, RA7) çıkış olarak ayarlar.

TRISC=1;      //RC1 pinini giriş olarak ayarlar.

TRISB.B5=1; //RB5 pinini giriş olarak ayarlar.

TRISD.B2=0;  //RD2 pinini çıkış olarak ayarlar.
```

Çıkış olarak ayarlanmış olan bir porta veya pine "0" veya "1" değerlerini göndermek için kullanılacak kaydedici "LAT" isimli kaydedicidir. LAT kaydedicisi ile bir pine 0 değeri gönderildiğinde o pin GND seviyesine (örn. 0V) çekilirken, 1 değeri gönderildiğinde o pin VCC seviyesine (örn. 5V) çekilir.

LAT Kaydedicisi

```
LATB=0;      //B portunun tüm pinlerine 0 değeri gönder.

LATA=0x0F;  //A portunun ilk 4-bit'ine (RA0, RA1, RA2, RA3) 0,
              //son 4 bit'ine (RA4, RA5, RA6, RA7) 1 değeri gönder.

LATC=1;      //RC1 pinine 1 gönder.

LATB.B5=1;  //RB5 pinine 1 gönder.

LATD.B2=0;  //RD2 pinine 0 gönder.
```

Güç kaynağından giriş portuna doğru akan akıma SINK akımı, çıkış portundan toprağa doğru akan akıma SOURCE akımı denir. Bu 25mA'lık akımlar bir LED'i direk sürebilirler. Bu akımlar röle, optik izolatör, transistör, mosfet, triyak ve yükselteç devreleri gibi elemanlarla daha yüksek akımları ve gerilimleri kontrol edebilirler.

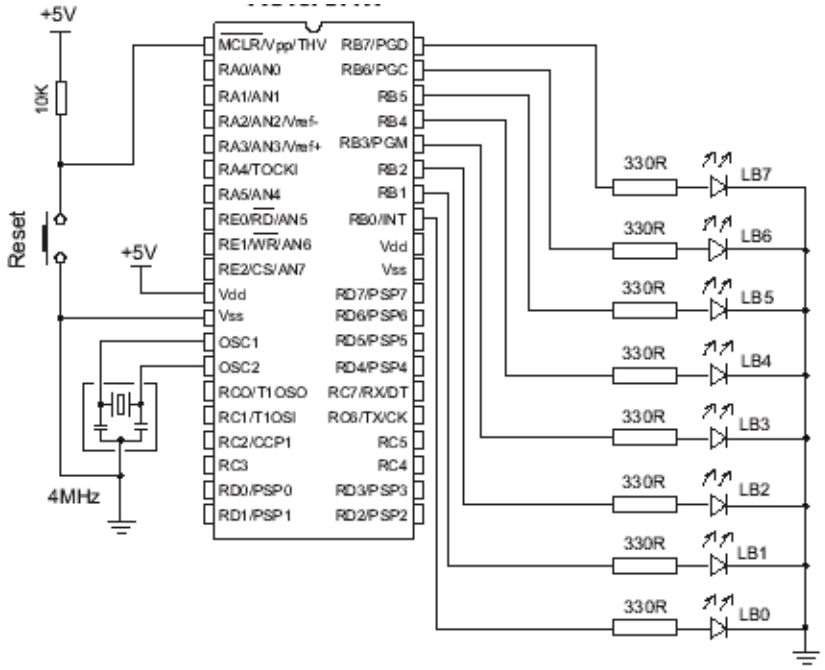
Bu uygulamada PORTB'nin bütün pinleri çıkış olarak ayarlanacak ve 1 saniye aralıklarla bütün ledler yakılıp söndürülecektir. Bu işlemi yapabilmek için öncelikle TRISB kaydedicisinde PORTB'nin bütün pinleri çıkış olarak ayarlanmalıdır. Daha sonra LAT kaydedicisi kullanılarak PORTB'nin pinlerine 1 saniye aralıklarla 1 ve 0 değerleri gönderilerek işlem tamamlanır.

Örnek program:

```
//************************
//Proje ismi   : Deney1
//Amacı        : PortB'yi çıkış olarak ayarlayarak pinlerin dijital değerlerini kontrol etmek.
//Test konfigürasyonu
//MCU          : 18F45K22
//Osilatör     : 32Mhz
//
//SW3.2     => ON,
//SW3.1, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8=>OFF
//SW1       =>OFF
//SW2       =>OFF
//SW4       =>OFF
//
//ADC INPUT J16 ataçı boşta olmalı.
//************************

void main()
{
 TRISB = 0b00000000;     // Port B çıkış olarak ayarlandı
 while(1)                // Sonsuz döngü
 {
  LATB=0;                // B portunun bütün pinlerine 0 değeri gönderiliyor.
  delay_ms(1000);        // 1 saniye bekleniyor(1000 mili saniye).
  LATB=0b11111111;       // B portunun bütün pinlerine 1 gönderiliyor(5V).
  delay_ms(1000);        // 1 saniye bekleniyor(1000 mili saniye).
 }
}
```

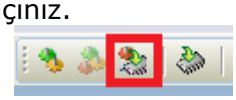
Yukarıdaki program mikroC ile yazılıp derlendikten sonra EasyPIC7 sistemine yüklendiğinde PORTB'ye bağlı ledlerin 1 er saniye aralıklarla yanıp sönmesi gözlemlenir.

.



Şekil 1.1 Çıkış Uygulaması Referans Devre Şeması

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 1-MİKRODENETLEYİCİDE VERİ ÇIKIŞI UYGULAMASI" içerisindeki "Deney1.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Ledlerin yanıp sönmesini gözlemleyiniz.

- **PORTB'ye bağlı ledlerden ilk önce ilk 4 tanesini, daha sonra son 4 tanesini birer saniye aralıklarla yakıp söndüren programı yazınız.**
- **Örnek kodlarda verilen işlemi aynı anda PORTB, PORTC ve PORTD üzerinde gerçekleştiriniz.**

DENEY 2. LED'Lİ YÜRÜYEN IŞIK UYGULAMASI

AMAÇ

1) Mikrodenetleyicinin portlarının çıkış olarak seçilmesini öğretmek,
2) Mikrodenetleyicide gecikme ve döngüleri öğretmek.

GEREKLİ MALZEME

1) EasyPIC v7 Kartı.

Giriş:

Bir önceki deneyde yapmış olduğumuz uygulamada belli bir portu çıkış olarak seçip o porttaki LED'i yakarak çıkış uygulamasını görmüştük. Bu uygulamada ise çıkıştaki bir bilginin kaydırılması ve belli bir süre bekletilmesi ile ilgili çalışma yaparak LED'li yürüyen ışık uygulamasını göreceğiz.

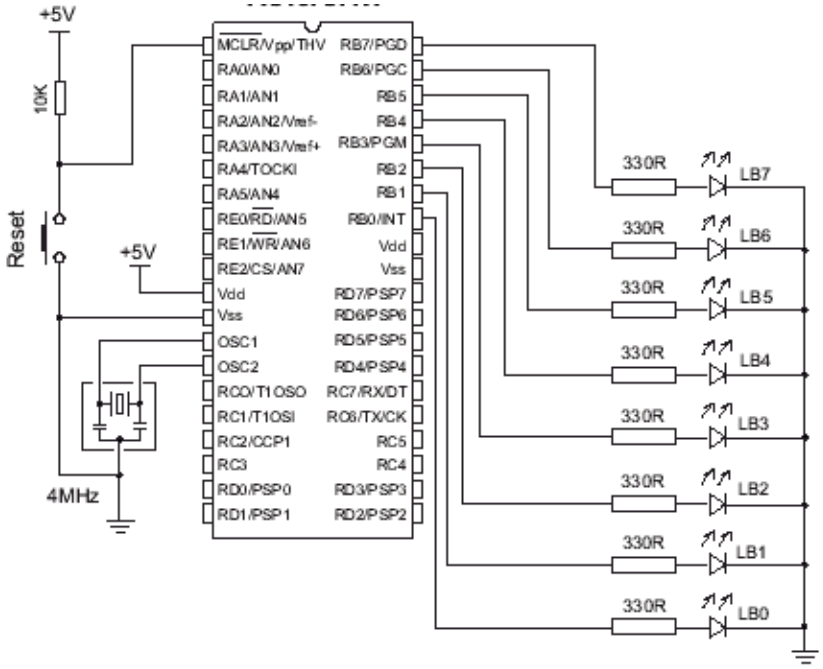
MikroC'de bit kaydırma işlemi sadece >> ve << işaretleri ile yapılır.

>>	sağa kaydırma işlemi için kullanılır.
<<	sola kaydırma işlemi için kullanılır.

Bit kaydırma işleminin nasıl yapıldığı aşağıdaki örnek ile açıklanmıştır.

```
char TA;        // TA isminde char tipinde bir değişken tanımlanıyor.
char TB;        // TB isminde char tipinde bir değişken tanımlanıyor.
char TC;        // TC isminde char tipinde bir değişken tanımlanıyor.

void main ()
{
TA = 0x0F               //TA ya 00001111 bilgisi atanır
TB = 0x81               //TB ye 10000001 bilgisi atanır
TC = TA >> 4;           //TC 00000000 olur. (TAnın 4-bit sağa kaydırılmış durumu)
TC = TB << 4;           //TC 00010000 olur. (TA nın 4-bit sola kaydırılmış durumu)
}
```

Programın içerisinde gecikme yaptırılması için kullanılan komut da "delay" dir. Kullanımı **delay_ms(1000)** şeklindedir. Parantez içindeki değer milisaniye olarak gecikmenin süresini belirlemektedir.



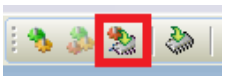
Şekil 2.1 LED kaydırma uygulaması referans şeması

Örnek program:

```
//************************
//Proje ismi   : Deney2-Led_kaydirma
//Amacı        : Belirlediğimiz bir datanın bitlerini
//               kaydırarak değişimi D portunda gözlemlemek
//Test konfigürasyonu
//MCU          : 18F45K22
//Osilatör     : 32Mhz
//
//SW3.4     => ON
//SW3.1, 3.2,3.3, 3.5, 3.6, 3.7, 3.8=>OFF
//SW1       =>OFF
//SW2       =>OFF
//SW4       =>OFF
//
//ADC INPUT J16 ataçı boşta olmalı.
//************************

unsigned int test;      // "test" isminde işaretsiz tamsayı tipinde değişken tanımlanıyor.
int i;                  // "i" isminde tamsayı tipinde bir değişken tanımlanıyor.
void main() {
TRISD=0;                // Port D çıkış olarak ayarlandı
test=0b10100000;        // test değişkenine 0b10100000 değeri yükleniyor.
```

```
while(1)              // sonsuz döngü
{
 for(i=0;i<9;i++)     // 9 kez sayacak döngü.
 {
  PORTD=test;         // PORTD'ye test değişkeni gönderiliyor.
  test=test>>1;       // bitler sağa kaydırılıyor
  delay_ms(500);      // 500 milisaniye bekleniyor.
 }
}
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 2_LED'Lİ YÜRÜYEN IŞIK UYGULAMASI" içerisindeki "Deney2.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Ledlerin sağa doğru kayışını gözlemleyiniz.

• Uygulamada ledler sadece 1 sefer sağa doğru kaydıktan sonra program bitmektedir. Kodlara ledler'in hiç durmadan kayma işlemini tekrar etmesi için gereken eklemeyi yapınız.
• Ledleri önce sağa, sonra sola kayacak şekilde çalışacak kodu yazınız.

DENEY 3. MİKRODENETLEYİCİDE VERİ GİRİŞ-ÇIKIŞ UYGULAMASI

AMAÇ

1) Mikrodenetleyici'nin portlarını öğretmek,
2) Mikrodenetleyici'ye veri alışverişini öğretmek.

GEREKLİ MALZEME

1) EasyPIC v7 Kartı
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti,

Giriş:

Bir önceki deneyde yapmış olduğumuz uygulamada belli bir portu çıkış olarak seçip o porttaki LED'i yakarak çıkış uygulamasını görmüştük. Bu uygulamada ise mikrodenetleyicinin girişine uygulanan bir bilgiye göre çıkıştan bir bilginin alınmasını göreceğiz.

TRIS Kaydedicisi

TRISB=0 //B portunun tüm pinlerini çıkış olarak ayarlar

TRISA=0x0F //A portunun ilk 4-bit'ini (RA0, RA1, RA2, RA3) giriş olarak ayarlar

TRISC=0xFF //C portunun tüm pinlerini giriş olarak ayarlar

TRIS kaydedicisiyle istenilen pinler giriş ve çıkış olarak ayarlandıktan sonra giriş olan pinlerden veriler "PORT" kaydedicisi ile okunurken, çıkış olarak ayarlanan pinlere "LAT" kaydedicisi ile veri gönderilir. Ayrıca eğer giriş olarak ayarlanan pinlerin analog fonksiyonları da bulunuyor ise bu pinlerin bağlı oldukları porta ait "ANSEL" kaydedicisi ile dijital giriş olarak ayarlanması gerekmektedir.

ANSEL Kaydedicisi

18F45K22 mikrodenetleyicisinde bulunan bütün portlarda analog fonksiyonlara sahip pinler de bulunmaktadır. Dolayısı ile bütün portlara ait ANSEL kaydedicileri vardır. ANSEL kaydedicisindeki bir biti "0" yaptığımızda o kaydediciye denk gelen pinin dijital giriş olarak kullanılmasına izin verilir, "1" yaptığımızda ise o kaydediciye denk gelen pinin analog giriş özelliği aktif edilir.

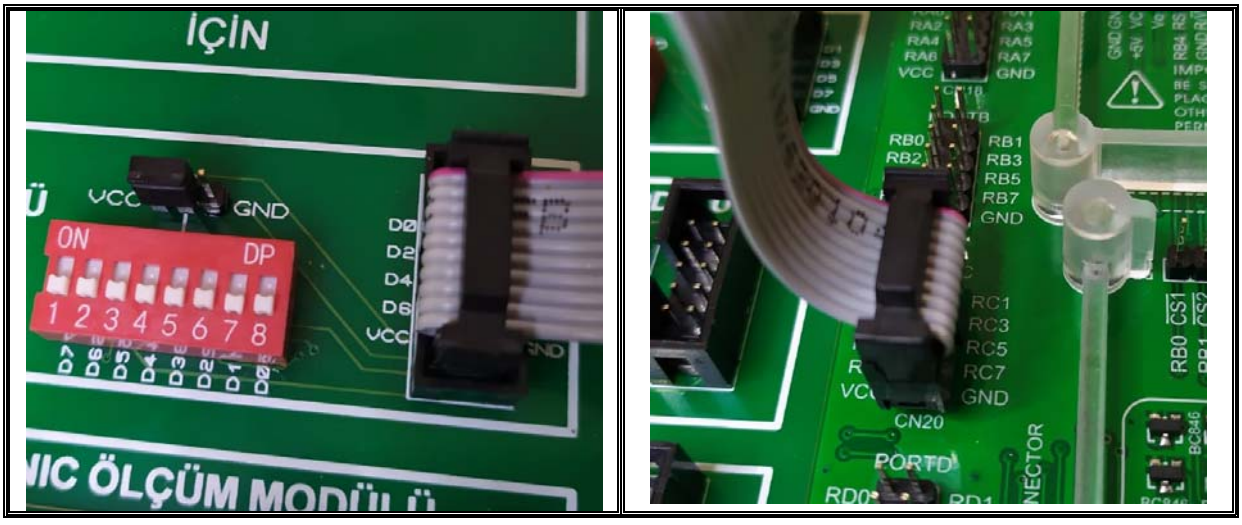
Örnek program:

```
//************************
//Proje ismi        : Deney3-Veri_Giris
//Amacı             : Belirlediğimiz bir datanın bitlerini
//                    kaydırarak değişimi D portunda gözlemlemek
//Test konfigürasyonu
//MCU               : 18F45K22
//Osilatör          : 32Mhz
```

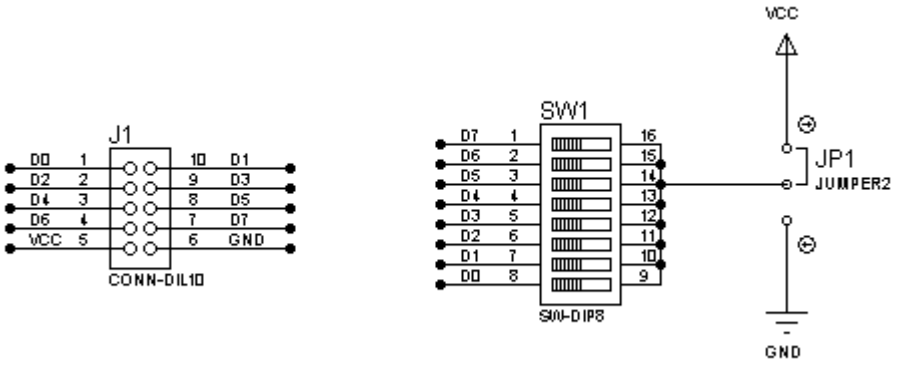
```
//
//PortC=>Pull-down
//SW3.4     => ON
//SW3.1, 3.2,3.3, 3.5, 3.6, 3.7, 3.8=>OFF
//SW1       =>OFF
//SW2       =>OFF
//SW4       =>OFF
//
//ADC INPUT J16 ataçı boşta olmalı.
//Giriş modülü ataçı VCC tarafında.
//**********************

void main()
{
  ANSELC = 0;         // Port C dijital olarak ayarlandı
  TRISC = 0xFF;       // Port C giriş olarak ayarlandı
  TRISD = 0x00;       // Port D çıkış olarak ayarlandı
  LATD=0X00;

  while(1)
  {
   LATD=PORTC;
  }
}
```

Yukarıdaki program mikroC ile yazılıp derlendikten sonra EasyPIC7 sistemine yüklendiğinde C portu butonları input, D portu LED'leri output olarak atanır. C portundaki herhangi bir butona basıldığında D portunda ona karşılık gelen LED yanar. Örneğin C portunun 3. Butonuna basıldığında, D portunun 3. LED'i yanar.

Bu uygulamayı BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki GİRİŞ MODÜLÜ'nü kullanarak çalıştırabilirsiniz. GİRİŞ MODÜLÜ'nü flat kablo ile EasyPIC7'nin C portuna bağlamanız yeterlidir. Çünkü programda C portu giriş olarak ayarlanmıştır.



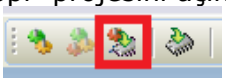
BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki GİRİŞ MODÜLÜ kullanılırken EASYPIC v7 ile bağlantı sağlayacak olan flat kablonun konektörleri yukarıdaki fotoğraflarda görüldüğü şekilde takılmalıdır.



Şekil1.1 GİRİŞ MODÜLÜ Devre Şeması

GİRİŞ MODÜLÜ'nün şemasında görüldüğü gibi, giriş bilgisi DIP anahtar kullanılarak sağlanmaktadır. Burada, JP1 jumperının VCC konumunda olmasına dikkat edilmelidir. Bu sayede Giriş Modülündeki DIP Anahtarının ON pozisyonuna getirilen kanalının, ilgili pinine lojik 1 (Vcc) bağlantısı sağlanır. Yani yukarıdaki programımızda C portu input olarak ayarlandığından DIP anahtarında 1. kanalı ON pozisyonuna getirirsek C portunu 7. pinine lojik 1 bilgisi iletmiş oluruz. Bu durumda PORTD'nin 7. Ledi yanacaktır.

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 3_MİKRODENETLEYİCİDE VERİ GİRİŞ-ÇIKIŞ UYGULAMASI" içerisindeki "Deney3.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. EasyPIC v7 kartının PORTC bölümündeki butonlara basarak PORTD bölümündeki ledlerin değişimini gözlemleyiniz.
7. EasyPIC v7'nin sol tarafında bulunan PORTC pinleri ile beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki Giriş Modülü'ne ait pinleri ı flat kablo aracılığı ile fotoğraflarda görüldüğü gibi yapınız.
8. Giriş modülündeki DIP Switch anahtarlarının konumlarını değiştirerek PORTD bölümündeki ledlerin durumlarını gözlemleyiniz.

• Giriş Modülündeki değişiklikleri aynı anda PORTB ve PORTD'ye bağlı ledlerde gösteren programı yazınız.

DENEY 4. 7-PARÇALI LED GÖSTERGE KONTROL DENEYİ

AMAÇ

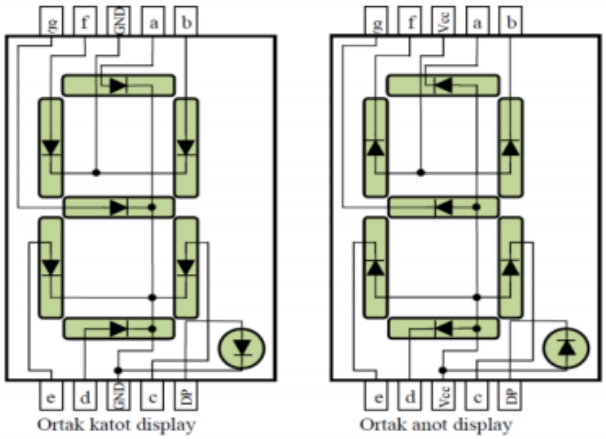
1) Mikrodenetleyicinin portlarını kullanarak 7-segmentli display kullanılmının öğrenilmesi.

GEREKLİ MALZEME

1) EasyPIC v7 Kartı

Giriş:

Seven segment LED display, ortak anotlu veya ortak katotlu olabilen ve 7 adet LED içeren bir displaydir. Bu LED'lerden bazılarının yakılması durumunda displayde bir görüntü ortaya çıkar. Display bir PIC mikrodenetleyici yardımı ile sürülür. PIC'in RB0 dan RB7'ye kadar olan 8-bit'i bu iş için ayrılır. Genelde RB0 segmenti A LED'ine gider ve RB7 nokta olarak kullanılır. Bir 7-segmentte saat yönünde giderek ve tepeden başlayarak LED ler: A, B, C, D, E, F, G olarak adlandırılır. Segmentin üst ve altında orta uçlar ortak uç olarak kullanılır (ortak katot veya ortak anod).



Şekil 4.1 7-Segmentli display yapısı

PORTB'den gelen bilgilerin, klasik bağlantı yapılmış olan bir displayde oluşturduğu şekil, PORTB'ye aktarılan bilgiye bağlıdır. RB0 dan RB7'ye kadar olan bit'lerin HIGH (1) veya LOW (0) olmasına göre, bu çıkışların bağlandığı LED'ler bir görüntü oluşturur.

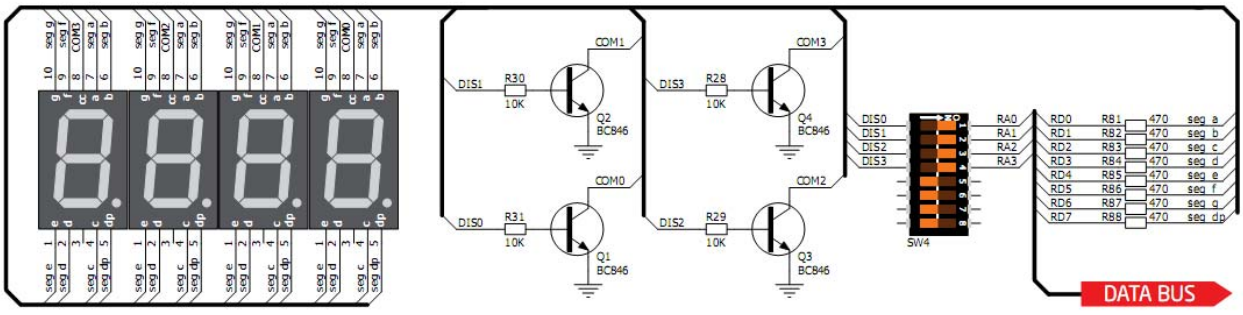
Örneğin segmentte 0 olması için, A, B, C, D, E, F LED'lerinin HIGH olması gerekir. Bu LED'lerin geldiği PIC pinleri RD0 – RD6 pinleridir. O halde bu bit'ler 1 olmalıdır, yani binary (ikilik) olarak PORTD'ye "00111111" atanmalıdır. Bu atamayı ikilik olarak yapabileceğimiz gibi decimal veya hexadecimal olarak da yapabiliriz, yukarıdaki ikilik sayının karşılığı decimal "63", hexadecimal "3F" tir. Yani decimal 63 sayısı ikilik olarak ifade edildiğinde, karşılığı B "00111111"dir. Bu da PORTD'de "0" gösterir. Sonuçta PORTD'ye atanan Hex. Veya Dec. sayının ikilik karşılığıdır.

DECIMAL	HEXADECIMAL	BINARY	SEVEN SEGMENT
63	3F	00111111	0 Okunur
6	06	00000110	1 Okunur
91	5B	01011011	2 Okunur
79	4F	01001111	3 Okunur
102	66	01100100	4 Okunur
109	6D	01101101	5 Okunur
125	7D	01111101	6 Okunur
7	07	00000111	7 Okunur
127	7F	01111111	8 Okunur
111	6F	01101111	9 Okunur

Tablo 4.1 7-Segmentli Display'de Rakamlara Karşılık Gelen Kodlar

Tablo 4.1'de görülen bir decimal veya hexadecimal sayıyı PORTD'ye atadığınız zaman seven segmentte karşılığı olan rakamı görürsünüz. Bu işlem için çevrim tabloları kullanılır. Çevrim tablosuna hexadecimal veya decimal olarak yazdığımız sayı, ikilik karşılığına çevrilerek pic portlarından displaye aktarılır ve bir görüntü oluşur.

EasyPIC7 setinde 7-segment bağlantı şeması Şekil 4.2'de verilmiştir. 4 adet displayin ortak katotları RA0, RA1, RA2 ve RA3 pinleri ile kontrol edilmiştir. PORTD pinleri ise segmentleri kontrol etmektedir. 4 adet display'den hangisini çalıştırmak istersek onun ortak katodundaki transistörü sürerek display seçimi yapılmış olur.



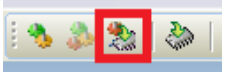
Şekil 4.2 EasyPIC7'de 7-Segment Bağlantı Şeması

Bu uygulamada 7 segment displayde 0'dan 9'a kadar olan rakamlar birer saniye aralıklarla gösterilecektir.

Örnek program:

```
//***********************
//Proje ismi       : Deney3-Veri_Giris
//Amacı            : Belirlediğimiz bir datanın bitlerini
//                   kaydırarak değişimi D portunda gözlemlemek
//Test konfigürasyonu
//MCU              : 18F45K22
//Osilatör         : 32Mhz
//
//
//SW1      =>OFF
//SW2      =>OFF
//SW3      =>OFF
//SW4.1, 4.2, 4.3, 4.4  => ON
//SW4.5, 4.6, 4.7, 4.8  => OFF
//***********************

//Rakam görüntülerinin hafızada tutulduğu tablo dizisi.
char tablo[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
int i=0;                    // "i" isminde tamsayı tipinde bir değişken tanımlanıyor.
void main()
{
 TRISA=0;                   // PORTA'nın tamamı çıkış olarak ayarlanıyor.
 TRISD=0;                   // PORTD'nin tamamı çıkış olarak ayarlanıyor.
 LATA=1;                    // EN sağdaki 7 segment display aktif ediliyor.
 LATD=0;                    // Bütün segmentler söndürülüyor.
 while(1)                   // Sonsuz döngü
 {
  for(i=0; i<10; i++)       // 0'dan 10'a kadar sayan döngü.
  {
   LATD=tablo[i];           // PORTD'ye rakamları gösterecek değerler sırasıyla gönderiliyor.
   delay_ms(1000);          // 1 saniye bekleniyor.
  }
 }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 4-7-PARÇALI LED GÖSTERGE KONTROL DENEYİ" içerisindeki "Deney4.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.

6. EasyPIC v7 kartı üzerindeki 7 segment displayi gözlemleyiniz.

- **7 Segment display'de 0'dan başlayıp 9999'a kadar sayan bir sayıcının kodlarını , araştırma yaparak yazınız.**

DENEY 5. LCD MODÜL KONTROL DENEYİ

AMAÇ

1) Mikrodenetleyici kullanarak LCD nin nasıl çalıştırıldığını öğretmek
2) MikroC Derleyicisinde LCD komutlarının öğretilmesi

GEREKLİ MALZEME

1) EasyPIC v7 Kartı
2) 2x16 LCD

Giriş :

LCD (**L**iquid **C**ristal **D**isplay) yani Sıvı Kristal Ekran elektrikle kutuplanan sıvının ışığı tekfazlı geçirmesi ve önüne eklenen bir kutuplanma filtresi ile gözle görülebilmesi ilkesine dayanan bir görüntü teknolojisidir. LCD 1x16 , 2x16, 2x20, 4x16 gibi değişik ebatlarda olabilir. Bu ebatlar LCD ekranın satır ve sütun sayılarını belirtmektedir. Özellikle mikrodenetleyici ile yapılan uygulamalarda kullanıcı referansı için ayarlanması gereken değerler için ya da ölçümü yapılması gereken değerlerin kullanıcıya aktarılması gibi amaçlar için kullanılan görüntü arabirimidir.

EasyPIC v7 setinde bir çok LCD uygulaması yapmak mümkündür. Sette 2x16 (2 Satır, 16 Karakter) LCD kullanılmıştır. PIC18F45K22'nin B Portu çıkış olarak kullanılarak bağlanmıştır. LCD üzerinde 8 adet Data Portu, 3 adet te Kontrol Portu bulunmaktadır.

Karakter LCD ekranlar ister 8 bit, ister 4 bit data pini kullanılarak kontrol edilebilmektedir. 8 bit kontrol modunda veriler 2 kat daha hızlı iletilir. EasyPIC v7 kartında bulunan 2x16 karakter LCD4 bit data modunda kullanılmaktadır.

Data Pinleri : D0,D1,D2,D3,D4,D5,D6,D7
Control Pinleri : R/W , RS , E

Pic Mikrodenetleyicisi ile bu data ve kontrol pinleri doğrudan bağlanabilir. Şekil 5.1 de bu bağlantı şekli görülmektedir. LCD nin 3. Pinine (Vee) bağlı olan gerilim bölücü devre LCD deki görüntünün parlaklığını ayarlamak için kullanılmaktadır.

Karakter LCD Kontrolü için kullanılan MikroC Komutları :

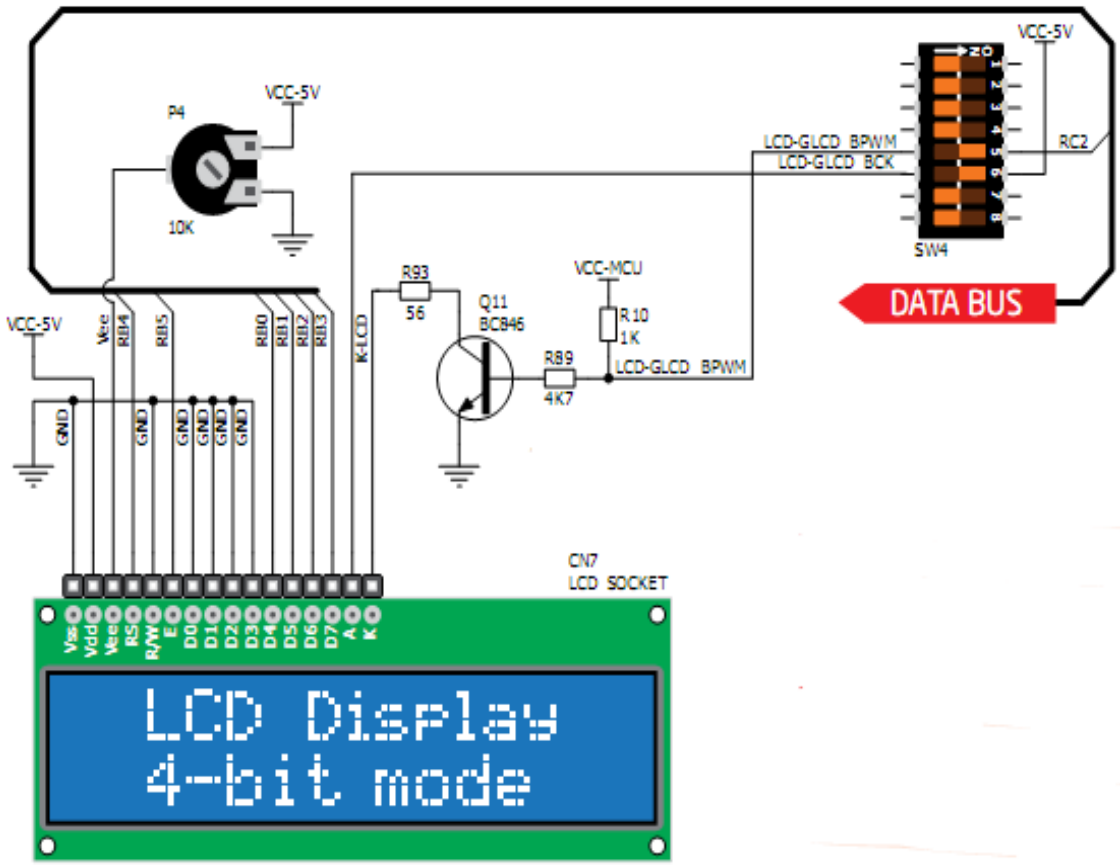
Komutlar	**Açıklama**
Lcd_First_Row	kursörü 1. sıraya götür
Lcd_Second_Row	kursörü 2. sıraya götür
Lcd_Third_Row	kursörü 3. sıraya götür
Lcd_Fourth_Row	kursörü 4. sıraya götür
Lcd_Clear	Ekranı Temizle
Lcd_Return_Home	kursörüçıkış pozisyonunda tut
Lcd_Cursor_Off	kursörü kapat
Lcd_Underline_On	altını çizme yi aç
Lcd_Blink_Cursor_On	yanıp sönmeyi aç
Lcd_Move_Cursor_Left	LCD nin RAM ındaki bilgiyi değiştirmeden kursörü sola kaydır
Lcd_Move_Cursor_Right	LCD nin RAM ındaki bilgiyi değiştirmeden kursörü sağa kaydır
Lcd_Turn_On	LCD yi aç

Lcd_Turn_Off	LCD yi kapat
Lcd_Shift_Left	LCD nin RAM ındaki bilgiyi değiştirmeden kursörü sola değiştir
Lcd_Shift_Right	LCD nin RAM ındaki bilgiyi değiştirmeden kursörü sağa değiştir

MikroC Derleyici yazılımının LCD için kullanılan komutlarının dışında bir takım procedure ve function'lar kullanılmaktadır. Bunlar;

LCD_Init	: LCD yi portla ilişkilendirir.
LCD_Out	: LCD ye seçilmiş satır ve sütuna yazı yazdırır.
LCD_Out_Cp	:Kursör pozisyonundan itibaren yazı yazdırır.
LCD_Chr	: LCD ye seçilmiş satır ve sütuna karakter yazdırır.
LCD_Chr_Cp	: Kursör pozisyonundan itibaren karakter yazdırır.
LCD_Cmd	: LCD ye komut gönder

Bu procedure ve functionların kullanım şekilleri, örnekler içerisinde açıklamaları ile birlikte verilmiştir.



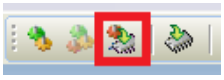
Şekil 5.1 EasyPIC v7 üzerindeki LCD bağlantı şeması

Örnek Program :

```
//************************
//Proje ismi        : Deney5-Lcd Modül Kontrol Deneyi
//Amacı             : LCD modülde metin görüntülemek
// Test konfigürasyonu      :
//MCU               : 18F45K22
//Osilatör          : 32Mhz
//
//SW4.6     => ON
//
//************************

// Lcdmodül bağlantıları
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// Lcdmodül bağlantıları bitis


char i;                             //Döngü değişkeni.
void main()
{
 Lcd_Init();                         // LCD ilklendirme fonksiyonu
 Lcd_Cmd(_LCD_CLEAR);                // LCD ekranını temizler.
 Lcd_Cmd(_LCD_CURSOR_OFF); // Cursor ekranda gösterilmez.
 LCD_Out(1,3,"Beti Bilisim");       // txt1'deki veriyi LCD ekranının
                                     // 1. satırına 3. karakterden başlayacak şekilde yazar.
 delay_ms(2000);                     // 2s beklenir.

 for(i=0; i<2; i++)                  // ekrandaki yazı 2 kere sola kaydırılır.
 {
  Lcd_Cmd(_LCD_SHIFT_LEFT);  // sola kaydırır.
  delay_ms(600);
 }

 while(1)                            //sonsuz döngü
 {
  for(i=0; i<4; i++)                 // ekrandaki yazı 4 kere sağa kaydırılır.
  {
    Lcd_Cmd(_LCD_SHIFT_RIGHT);
    delay_ms(200);
  }
```

```
 for(i=0; i<4; i++)                    // ekrandaki yazı 4 kere sola kaydırılır.
   {
    Lcd_Cmd(_LCD_SHIFT_LEFT);
    delay_ms(400);
   }
 }
}
```

Yöntem:

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 5-LCD MODÜL KONTROL DENEYİ" içerisindeki "Deney5.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. EasyPIC v7 kartı üzerindeki LCD'yi gözlemleyiniz.

• LCD'de 0-9999'a kadar sayan bir program yazmak için gerekli araştırmayı yapınız ve bu işlemi gerçekleştiren kodu yazınız.

DENEY 6. SAYISAL/ÖRNEKSEL ÇEVİRİCİ (DIJITAL-TO-ANALOG CONVERTER, DAC) DENEYİ

AMAÇ

1) Sayısal/Örneksel Çevirici devresi (DAC) ile Mikrodenetleyici entegresinin iletişimini sağlamak,
2) Microchip firmasının MCP4921; 12-bit Seri DAC entegresinin nasıl programlanacağını göstermek ve Mikrodenetleyici/DAC arabirimi yardımıyla sayısal-örneksel dönüşümler yapmak,
3) Mikrodenetleyici kontrollü bir DAC ile Rampa, Üçgen, Kare Dalga sinyallerinin üretilebileceğini göstermek

GEREKLİ MALZEME

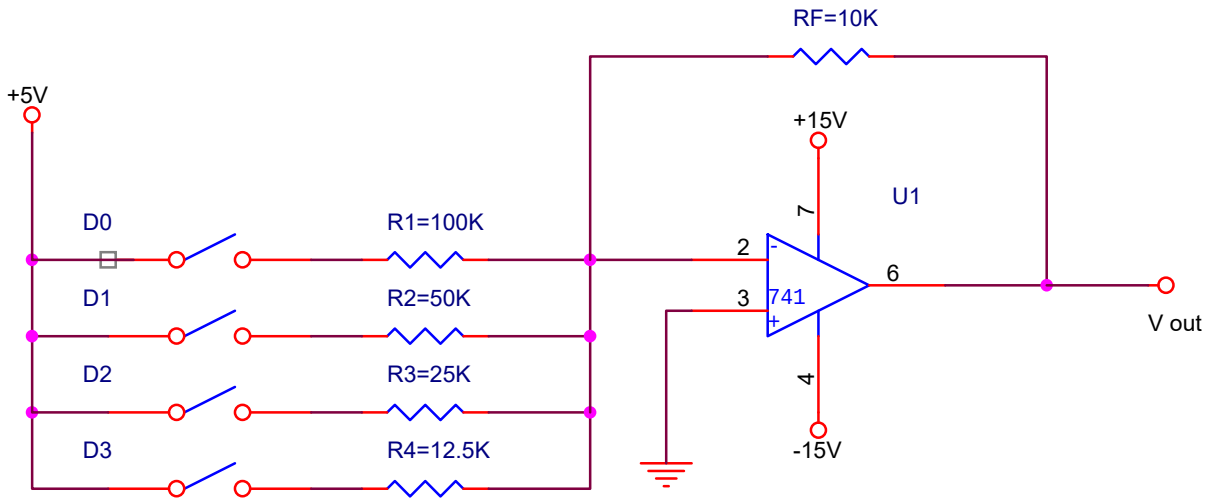
1) EasyPIC7 Kartı,
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti,
3) Osiloskop.

Giriş

Bu deneyde, Sayısal/Örneksel (S/Ö) Çevirici'nin çalışma prensiplerini, teknik özelliklerini ve özellikli uygulama parametrelerini ele alacağız. Daha sonra, Mikrodenetleyici portlarından S/Ö Çevirici'nin programlanmış işlevleri yerine getirebilmesi için nasıl programlandığını basitçe örnekleyeceğiz. S/Ö Çevirici devresi çıkışında elde edilen sinyalleri izlemek için osiloskop kullanacağız.

Sayısal/Örneksel dönüşüm teknikleri:

İkilik Ağırlıklı Direnç Girişli S/Ö Çevirici



Şekil 6.1. İkilik Ağırlıklı Direnç Girişli S/Ö Çevirici

Şekil 6.1'de 4-bit İkilik Ağırlıklı Direnç Girişli S/Ö Çevirici'nin devresi gösterilmiştir. Girişlerine bağlı anahtarlar yardımı ile ikilik nibble; (4-bit), sayısal veri uygulanan devrenin işlemsel yükselteci çıkışından, bu sayısal veri ile orantılı örneksel voltaj seviyesi elde edilir. Şekil 6.1'deki devre 4 girişli basit bir işlem yükselteci toplama devresidir.

İşlemsel yükselteçler ile ilgili bilinmesi gereken iki adet kural vardır;
1- İşlemsel Yükseltecin girişlerinden akım akmaz.
2- İşlemsel yükselteç girişlerindeki voltajlar birbirine eşittir.

Devreden görüldüğü gibi, işlemsel yükseltecin evirici olmayan girişi toprağa bağlanmıştır, yani sürekli 0 Volt'ta durmaktadır. İşlem yükseltecinin çıkışından evirici çıkışa bağlı bir geri-besleme vardır; bu nedenle işlem yükselteci bu girişi sürekli 0 Volt'ta tutmaya çalışacaktır. Genel olarak böyle bir devrede evirici giriş "Hayali Toprak" (Virtual Ground) veya "Toplama Noktası" (Summing Point) olarak isimlendirilir.

4 anahtardan hiçbiri kapalı değilse, dirençlerin hiçbirinden ve dolayısıyla da geri besleme direncinden (R_f), akım geçmeyecek ve işlem yükselteci evirici giriş ile aynı voltajda, yani 0 Volt olacaktır.

Şimdi **D0** anahtarının kapatıldığını düşünelim: Böylece R1 direncinin bir ucuna +5 Volt uygulanmış olur. Ohm kanunu R1 direncinden 5V/100kΩ=0.05 mA pozitif akım geçtiğini söyler. Evirici girişten işlemsel yükselteç içerisine herhangi bir akım akmayacağından bu akım R_f üzerinden akar ve R_f üzerindeki voltaj 0.05 mAx10 kΩ=0.5 V olur. R_f'in bir ucu 0 Volt olduğundan işlem yükselteci çıkışındaki voltaj -0.5 V olarak bulunur.

Özetle **D0** anahtarının kapatılması S/Ö Çevirici'nin çıkışında -0.5 V üretir.

Şimdi **D0** anahtarını kapatıp **D1** anahtarını açalım: R2 direnci R1 direncinin yarı değerinde olduğundan R2 direncinden 0 Volt'a ve oradan R_f' doğru akan akım bir önceki akımın iki katı, yani 0.1 mA olacaktır. Bu nedenle de işlem yükselteci çıkışındaki voltaj –(0.1 mAx10 kΩ)=-1 Volt olur.

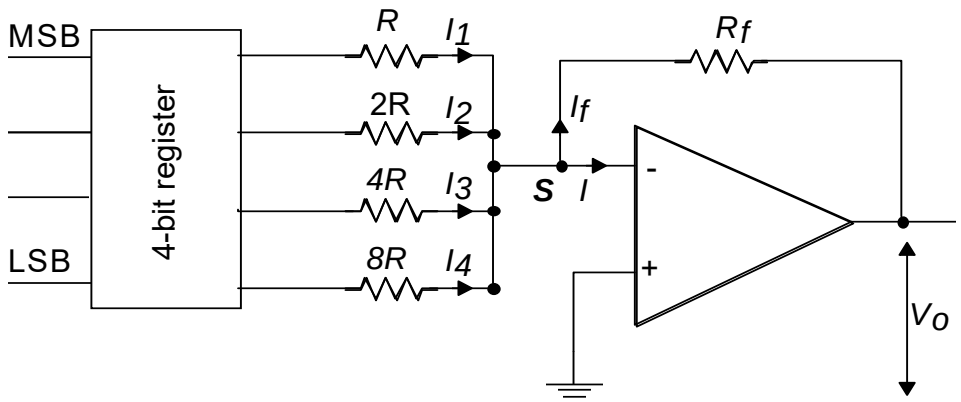
Özetle **D1** anahtarının kapatılması S/Ö Çevirici'nin çıkışında **D0** anahtarının kapatılması ile oluşan voltajın iki katını üretir.

Şimdi **D1** anahtarını kapatıp **D2** anahtarını açalım: R3 direncinden geçen 5V/25kΩ=0.2mA pozitif akım R_f üzerinden işlem yükselteci çıkışına akarak, çıkışta -2 Volt oluşturur. Benzer şekilde **D2** anahtarı açılıp **D3** anahtarı kapatılırsa, R4 direncinden geçen 5V/12.5kΩ=0.4mA pozitif akım R_f üzerinden işlem yükselteci çıkışına akarak, çıkışta -4 Volt oluşturur. Eğer **D2** ve **D3** anahtarlarının ikisi de kapatılırsa R3 direncinden geçen 0.2mA ile R4 direncinden geçen 0.4 mA toplama noktasında 0.6 mA oluşturur, bu akım da R_f üzerinden işlem yükselteci çıkışında -6 Volt üretir.

Özetle çıkıştaki voltaj kapatılan anahtarların oluşturduğu akımla orantılıdır. 4 akım değeri; R1, R2, R3, R4 dirençlerinin ağırlıkları ile aynen orantılıdır. Bu nedenle çıkışta oluşan voltaj girişlere uygulanan ikilik sayısal değer ile orantılıdır. **D0** anahtarı, en küçük akım değerini oluşturduğundan, **"En Az Ağırlıklı Bit" (LeastSignificant Bit, LSB), D3** anahtarı ise, en büyük akım değerini oluşturduğundan, **"En Fazla Ağırlıklı Bit" (MostSignificant Bit, MSB)** olarak adlandırılır.

Bu örnekteki S/Ö Çevirici devresinde 4 giriş olduğundan, sayısal kelimenin 0000 ile 1111 arasında 16 değişik kombinasyonu olabilir. 0000 girişi çıkıştaki en küçük gerilimi; 0 Volt'u üretirken, 1111 girişi çıkıştaki en büyük voltajı; 7.5Volt'u üretir.

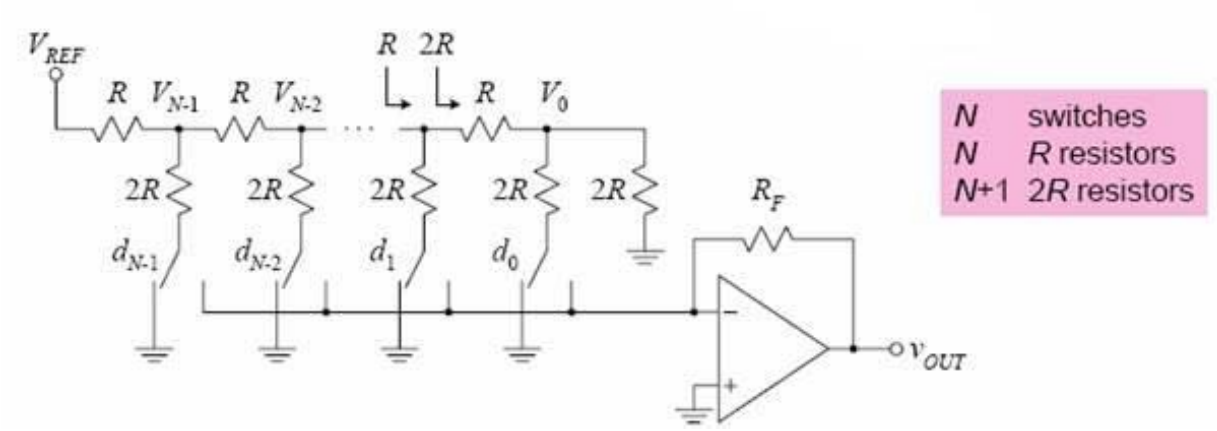
Bu devrenin sayısal girişleri anahtarları açıp kapamak yerine Şekil 6.2'de gösterildiği gibi bir 4-bit ikilik sayıcının çıkışlarından elde edilebilir.



Şekil 6.2. İkilik Sayıcı ile Sürülen İkilik Ağırlıklı Direnç Girişli S/Ö Çevirici

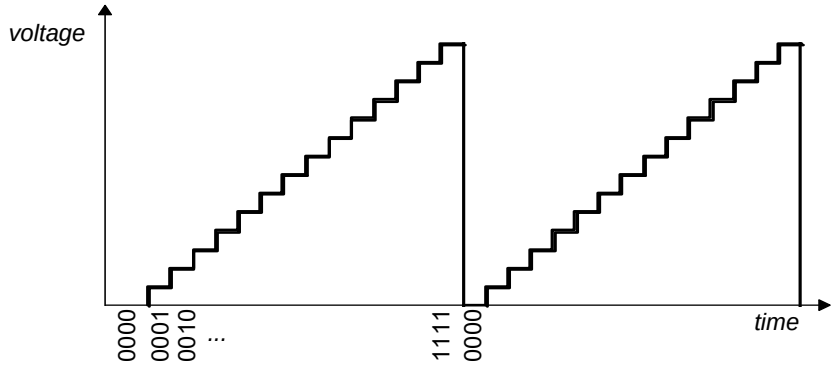
Bu örnekteki İkilik Ağırlıklı Direnç Girişli S/Ö Çevirici devrelerin en büyük dezavantajı kararlılığının ve doğruluğunun direnç değerlerine ve bunların sıcaklıkla değişimine bağlı olmasıdır. Ayrıca her bir direnç, bir öncekinden iki kat daha büyük olduğundan çözünürlüğü yüksek S/Ö Çevirici devrelerde çok büyük direnç değerlerinin kullanılmasını gerektirecektir.

Bu dezavantajdan dolayı tasarımda kullandığımız S/Ö Çevirici devrelerin çoğunda Şekil 6.3'de gösterilenR/2R devresi kullanılmaktadır.



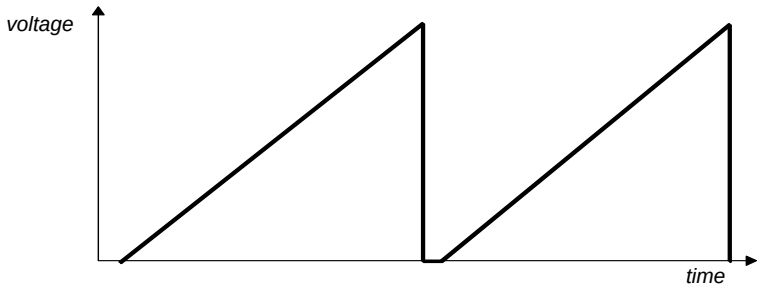
Şekil 6.3. R/2R direnç devreli S/Ö Çevirici

Şekil 6.2'de gösterilen İkilik Sayıcı ile Sürülen İkilik Ağırlıklı Direnç Girişli S/Ö Çevirici devresindeki sayıcı bir saat sinyali ile saydırılırsa, işlem yükselteci çıkışında Şekil 6.4'de gösterilen "Merdiven Dalga" (Stair-Case) sinyali elde edilir.



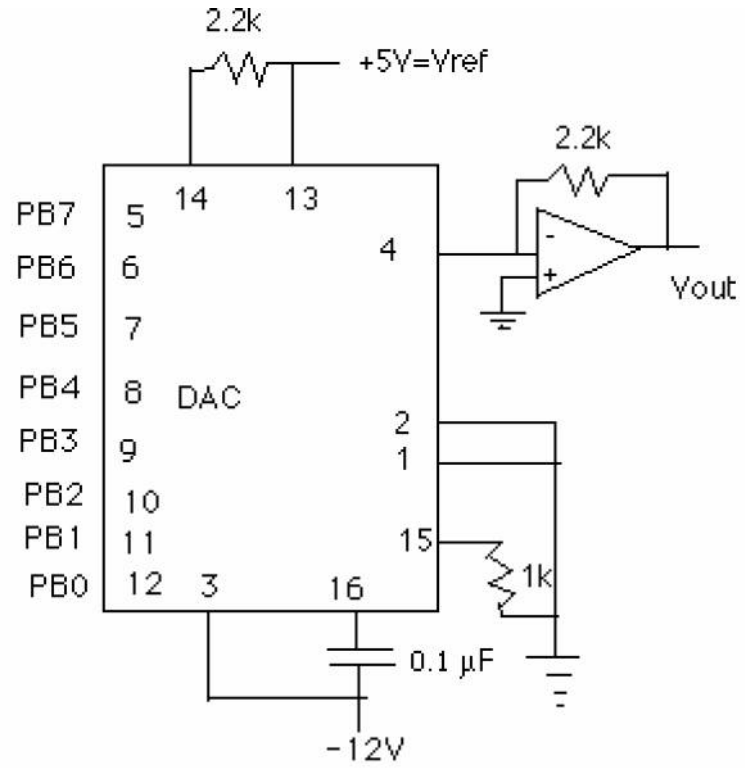
Şekil 6.4. S/Ö Çevirici Çıkışındaki Gerçek Dalga Şekli

Şekil 6.4' den öngörüleceği gibi, D/A Çevirici devresinde ne kadar fazla bit kullanılırsa, adımlar o kadar fazla ve küçük olur, dolayısıyla da çıkış gerilimi o kadar fazla örneksel sinyale yaklaşır (Şekil 6.5).



Şekil 6.5. S/Ö Çevirici çıkışında beklenen testere-dişi sinyal

Burada gösterdiğimiz gibi S/Ö Çevirici'nin kalbi ikilik ağırlıklı akım kaynakları ile bunları açıp-kapayan ikilik girişlerdir. Pratikte bütün bunlar tek bir yonga üzerindeki S/Ö Çevirici entegreleri içerisinde geliştirilmiştir.



Şekil 6.6 MC1408 8-bit S/Ö Çevirici

D/A Çevirici'lerin performans parametreleri:

1. Çözünürlük (Resolution)

Çözünürlük, çıkış geriliminin kaç değişik seviyeden oluştuğunu gösterir ve bit sayısı ile orantılıdır. Şekil 6.5'de verilen 8 girişli S/Ö Çevirici'nin 2^8=256 değişik çıkış seviyesi olabilir, bu nedenle de çözünürlüğü 1/256 dır. Bit sayısı ne kadar arttırılırsa çözünürlük o kadar artar, örneğin 10-bit S/Ö Çevirici'nin çözünürlüğü 1/1024, EasyPIC7 Setinde kullandığımız 12-bit MCP4921 Seri S/Ö Çevirici entegresinin çözünürlüğü ise 1/4096 dır.

2. Tam-Skala Çıkış Voltajı (Full-ScaleOutputVoltage)

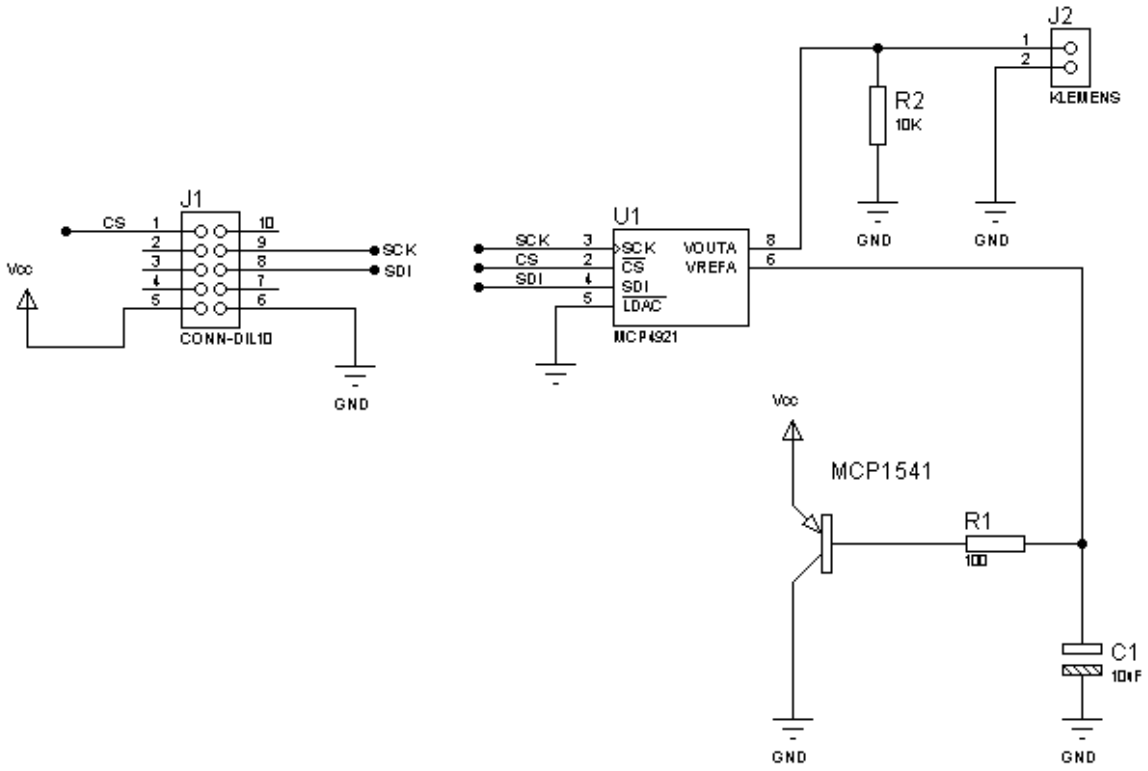
Önceki bölümde gösterdiğimiz gibi,S/Ö Çevirici'nin Tam-Skala Çıkış Voltajı referans akımı ve akım-voltaj çevirici işlem yükseltecinin geri-besleme direnci ile belirlenir. Çoğu S/Ö Çevirici'ler genellikle +5 Volt ile +10 Volt arasındaTam-Skala Çıkış Voltajı'na sahiptir. Bazı entegrelerde ise Tam-Skala Çıkış Voltajı ±5 Volt'dur. Yine önceki bölümden S/Ö Çevirici'nin gerçek maksimum voltaj seviyesi, Tam-Skala Çıkış Voltajı'ndan 1 LSB küçüktür. Örnek verirsek, 10-bit çözünürlüğü ve +5 Volt Tam-Skala Çıkış Voltajı olan bir S/Ö Çevirici düşünelim. Bu entegrenin en küçük adımı 5 Volt/1024= 0.0049Volt'dur. Dolayısıyla S/Ö Çevirici'nin gerçek maksimum Tam-Skala Çıkış Voltajı 5 V-0.0049 V= 4.9951 Volt'dur.

3. Doğruluk (Accuracy)

Doğruluk parametresi,S/Ö Çevirici'nin çıkışındaki örneksel voltajın gerçek değerine ne kadar yakın olduğunun göstergesidir. İdeal olarak S/Ö Çevirici'nin maksimum hatası ± ½ LSB'dir. Hatanın bu değerden büyük olması durumunda çıkış voltajı beklenen değerin bir üstündeki veya bir altındaki seviyeye yaklaşacaktır. Birçok yarı-iletken üreticisi firma S/Ö Çevirici'nin hata parametresini "± ½ LSB veya daha küçük" olarak vermektedir.

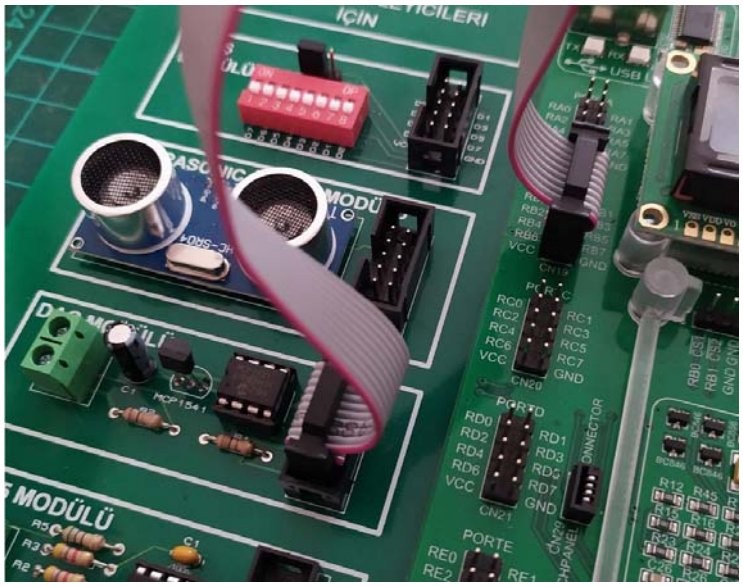
DAC Deneyi:

Microchip firmasına ait olan MCP4921 entegresi kullanılarak yaptığımız DAC devresi aşağıdaki şekilde görülmektedir. MCP4921 12bit çözünürlüğe sahip bir DAC entegresidir. Bu sayede 4096 farklı seviyede voltaj üretebilir.



Şekil 6.7 Beti MCU Seti üzerindeki DAC devre şeması

Bu uygulamayı BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki DAC MODÜLÜ'nü kullanarak çalıştırabilirsiniz. DAC MODÜLÜ'nü flat kablo ile EasyPIC7'nin B portuna bağlamanız yeterlidir. Çünkü programda B portu SPI iletişim için ayarlanmıştır.

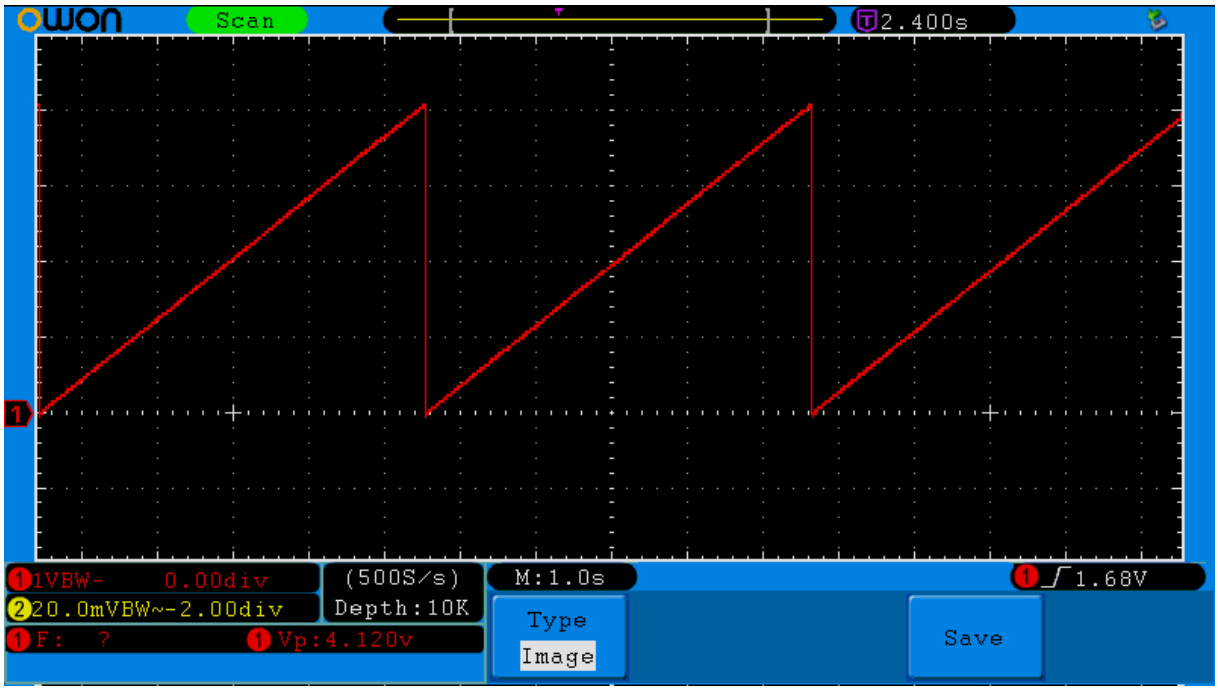


BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki DAC MODÜLÜ kullanılırken EASYPIC v7 ile bağlantı sağlayacak olan flat kablonun konektörleri yukarıdaki fotoğrata görüldüğü şekilde takılmalıdır.

Örnek program:

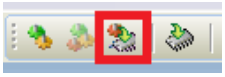
```
//************************
// Proje ismi       : Deney6-DAC Deneyi
// Amacı            : MCU içinde saydırdığımız dijital değeri
//                    sürekli arttırarak elde ettiğimiz analog
//                    voltajdan rampa sinyali elde etmek.
// Test konfigürasyonu       :
//----------------------------

//MCU              : 18F45K22
//Osilatör         : 32Mhz
//
// SW1=>OFF
// SW2=>OFF
// SW4=>OFF
// SW3=>OFF
//
// Library Manager ==> "Software_SPI" Seçilmeli.
//
//************************

// DAC modül bağlantıları
sbit SoftSpi_CLK at LATB3_bit;
sbit SoftSpi_SDI at LATB4_bit;
sbit SoftSpi_SDO at LATB5_bit;
```

```
sbit SoftSpi_CLK_Direction at TRISB3_bit;
sbit SoftSpi_SDI_Direction at TRISB4_bit;
sbit SoftSpi_SDO_Direction at TRISB5_bit;
sbit ChipSelect at LATB.B0;
// DAC modül bağlantılarının sonu.

unsigned int value, sayac=0;

void DAC_Output(unsigned int valueDAC)
{
 char temp;
 ChipSelect = 0;                    //  MCP4921 iletişimi aktifleştirir.
 //  High Byte gönderilir.
 temp = (valueDAC>> 8) & 0x0F;    //  valueDAC [11..8] -->temp [3..0] 'a yazılır.
 temp |= 0x30;                     // DAC ayarları yapılır.
 Soft_SPI_Write(temp);             // SPI ile gönderilir.

// LowByte gönderilir.
 temp = valueDAC;                  //  valueDAC[7..0] -->temp[7..0]  'a yazılır.
 Soft_SPI_Write(temp);             // SPI ile gönderilir.

 ChipSelect = 1;                   //   MCP4921 iletişimi durdurulur.
}

void main()
{
 TRISB = 0x00;                  // Configure PORTB as output
 ChipSelect=1;                  // MCP4921 pasif
 Soft_SPI_Init();               // SPI başlangıç ayarları

 value = 2048;              // DAC'ınbaşlangüç değeri.(analog olarak Vmax/2 verir)
                            //4096-->Vmax,  0000-->Vmin
 while (1)
 {
  if (sayac <4096) sayac++;
  else sayac = 0;
  value=sayac;

  DAC_Output(value);      // DAC'adata gönderilir.
  delay_us(100);
 }
}
```



Şekil6.8 Ürettiğimiz rampa dalgası

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 6 -SAYISALÖRNEKSEL ÇEVİRİCİ DENEYİ" içerisindeki "Deney6.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Osiloskop ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki DAC Modülünün çıkışındaki sinyali osiloskop vasıtası ile gözlemleyiniz.

• DAC modülünün çıkışından bir sinüs sinyali elde etmek için gerekli kodu yazınız.

DENEY 7. ANALOG SICAKLIK ÖLÇÜM UYGULAMASI

AMAÇ

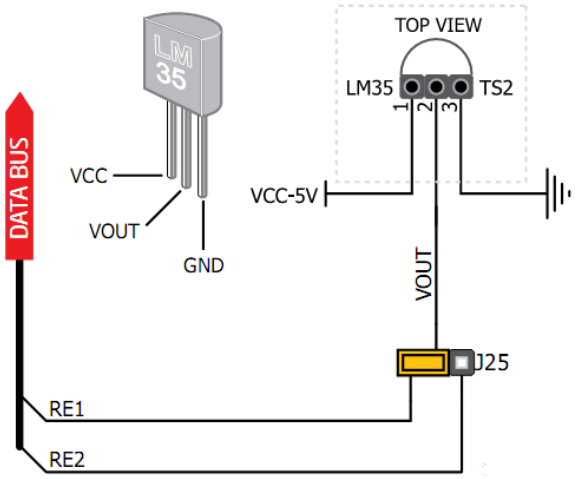
1) PIC18F45K22 kullanarak sıcaklık ölçümünün öğretilmesi.
2) EasyPIC v7 setinde LM35 kullanılarak analog giriş yaptırılması.
3) Sıcaklığın LCD'de görüntülenmesi.

GEREKLİ MALZEME

1) EasyPIC v7 Kartı.
2) LM35 Sıcaklık sensörü.
3) LCD modül.

Giriş :

Sıcaklık ölçümü, genelde endüstriyel ve medikal alanlarda uygulama sahası çok olan bir uygulamadır. Sıcaklığın mikrodenetleyici tarafından alınıp yazılan çeşitli programlarla çıkışlarına bağlanan LCD ekranda izlenmesi, sıcaklığın belli bir aralıkta sabit tutulması ya da algılanan sıcaklıkla başka çıkışların kontrol edilmesi gibi birçok uygulama sahası mevcuttur.



Şekil 7.1 Easy PIC v7 Üzerindeki Analog Sıcaklık Ölçüm Devresi

Bu uygulamada LM35 sıcaklık sensöründen ölçülen voltaj, sıcaklık bilgisine dönüştürülecek ve LCD ekranda gösterilecektir.

LM35 sıcaklık sensörü çıkışından, her 1°C için 10mV voltaj çıkartmaktadır. Örneğin 50°C sıcaklık içim LM35 çıkışından 500mV voltaj ölçülecektir.

18F45K22 mikrodenetleyicisinin RE1 pini aynı zamanda ADC biriminin 6. Kanalı görevini de üstlenmektedir. LM35 sıcaklık sensörü bu pine EasyPIC V7 kartı üzerindeki J25 ataçı vasıtasıyla RE1 pinine bağlanmıştır.

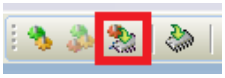
18F45K22 mkrodenetleyicisi 10bit çözünürlüğünde bir ADC birimine sahiptir. 0-5V arası 1024 parçaya bölünmüştür. Bu sebeple ADC biriminden gelen değer 0-1024 arasında bir değer olacaktır. Buradan yola çıkarak önce ölçülen voltaj milivolt cinsinden hesaplanacak, daha sonra voltaj değeri sıcaklık değerine çevrilerek LCD ekranda gösterilecektir.

Örnek Program :

```
//***********************
// Proje ismi       : Deney7-Analog Sıcaklık Ölçüm Uygulaması.
// Amacı            : MCU içinde saydırdığımız dijital değeri
//                    sürekli arttırarak elde ettiğimiz analog
//                    voltajdan rampa sinyali elde etmek.
// Test konfigürasyonu       :
//---------------------------
//MCU               : 18F45K22
//Osilatör          : 32Mhz
//
// SW1=>OFF
// SW2=>OFF
// SW4.6 => ON
// SW3=>OFF
//
// 5V - güç kaynağı için, güç Jumper'ini
// (J5) 5V pozisyonuna getiriniz. (referans = 5V)
// 3.3V - güç kaynağı için, güç Jumper'ini
// (J5) 3.3V pozisyonuna getiriniz. (referans = 3.3V)
//
// Library Manager ==> "ADC", "Lcd", "Conversions" Seçilmeli.
//
//***********************

// LCD modul bağlantıları
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// LCD modul bağlantıları

char txt[15];

void main()
{
 Lcd_Init();                    // LCD kurulumu yapılıyor.
```

```
 Lcd_Cmd(_LCD_CLEAR);                 // LCD temizleniyor.
 Lcd_Cmd(_LCD_CURSOR_OFF);            // LCD imleci kapatılıyor.
 Lcd_Out(1,1,"Sicaklik=");            // LCD ekrana "Sicaklik=.....°C" yazdırılıyor.
 Lcd_Chr(1,15,223);
 Lcd_Chr_CP('C');

 ADC_Init();                          // ADC birimi aktif ediliyor.

 while(1)
 {
  unsigned int adcDegeri=ADC_Read(6);      // RE1 pininden ADC değeri okunuyor (0-1024)
  float miliVolt=adcDegeri*(5000.0/1023);  // Ölçülen ADC değeri miliVolt'a çevriliyor.
  float sicaklik=miliVolt/10;              // miliVolt değeri sıcaklık değerine
//dönüştürülüyor.
  FloatToStr_FixLen(sicaklik,txt,5);       // Sıcaklık sayısal değeri ekranda gösterilmek için
                                           // yazıya (karakterlere) dönüştürülüyor.
  Lcd_Out(1,10,txt);                       // Stringe dönüştürülen sıcaklık ekranda
//gösteriliyor.
  delay_ms(100);
 }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 6 -SAYISALÖRNEKSEL ÇEVİRİCİ DENEYİ" içerisindeki "Deney6.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Parmağınızla LM35 sıcaklık sensörüne dokunarak sıcaklığın değiştiğini LCD ekranda gözlemleyiniz.

• ADC sinyalini dijital olarak filtreleme yöntemlerini araştırınız. Bubble Sort ve Medyan filtreleme tekniklerini inceleyeiniz.

DENEY 8. STEP (ADIMLI) MOTOR KONTROLU DENEYİ

AMAÇ

1) Adımlı-motorun kontrolünü öğretmek,
2) Bir Adımlı-motorun maksimum dönme hızını göstermek.

GEREKLİ MALZEME

1) EasyPIC v7 Kartı,
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti,

Giriş:

Bu uygulamada, Adımlı-motorların temel özelliklerini kullanarak, PIC 18F45K22 Mikrodenetleyici'si ile bir Adımlı motorun dönme yönünün ve dönüş hızının nasıl kontrol edilebileceği incelenecektir.

Adımlı-motorun Çalışması:

Senkron motorların tek çalışma modları vardır ki bu da "sürekli dönme"dir. Adımlı-motorlar ise küçük ve hassas bir açı (veya adım) kadar döndürülebilir ve orada durdurulabilir. Her adımı attırabilmek için motor sargılarına sayısal akım darbesi uygulamak gerekir. Böyle (n) darbe peş peşe uygulandığında, motor (n) adım döner ve durur. Adım açısı motor tasarımına bağlı olup, genelde 1,8 ile 30 derece arasında değişmektedir. Bu deneyde kullanılan motorların adım açısı 7,5 derecedir ki bu da bize motorun 48 adımda bir tur tamamlayacağını gösterir. Akım darbeleri motor sargılarına kontrol transistörleri ile uygulanır. Darbelerin uygulama sırası, motorun dönme yönünü, uygulama sıklığı ise motorun dönme hızını tayin eder.

Standart adım açıları;

1.8° - 200 adımda bir tur,
3.75° - 96 adımda bir tur,
7.5° - 48 adımda bir tur,
15° - 24 adımda bir tur.

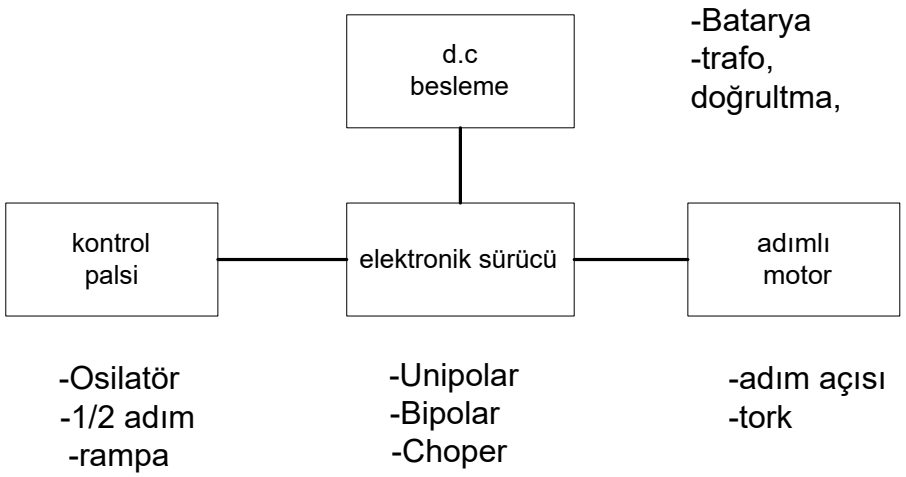
Adım sayısı ile adım açısının çarpımı motor milinin hareket açısını verir. Örneğin 7.5°'lik 6 adım, motora 45°'lik hareket verir.

Adımlı-motorlar genellikle sayısal kontrol işlemlerinde, elektronik cihazların kesin ve hızlı hareket etmeleri gereken durumlarda kullanılır. Örneğin:

- Manyetik teyp sürücüleri,
- Tele teyp ve şerit yazıcılar,
- Kamera iris kontrolü,
- Plotter'lerin ayarlanmasında, artan grafik kayıtçıları ve değişken hızlı grafik sürücüleri,
- Medikal aletler; kan örnekleyici, akciğer analizörleri, diyaliz pompaları,
- Sıvı yakıt kontrolü, valf kontrolü,
- Taksimetreler, kart okuyucular, üretim bandı pals sayıcıları, kantarlarda ve etiketleme sistemlerinde,
- Sayısal/Örneksel Çevirici'lerde ve uzaktan kontrollü cihazlarda,
- CNC, Lazer Kesim Cihazları ve 3D yazıcılarda…

Bu örneklerin hepsinde ortak nokta harekettin kontrolüdür. Hareket ve/veya pozisyon kontrolünün gerekli olduğu her yerde step motor kullanılabilir. Bu, genelde daha avantajlıdır.

Uygulamada kullanılacak Adımlı-motor, sürücü devre ve motor karakteristiklerine göre seçilir. Aşağıdaki şekil Adımlı-motor sisteminin dört önemli öğesini göstermektedir.



Şekil 8.1 Genel olarak Adımlı-motorun sürücü blok şeması

Çalışma voltajı	12 volt
Tipik faz akımı	175 mA
Adım açısı	7.5 derece
Max. hız	40 adım/san.
Tork	20 mNm
Ağırlık	170 gr.

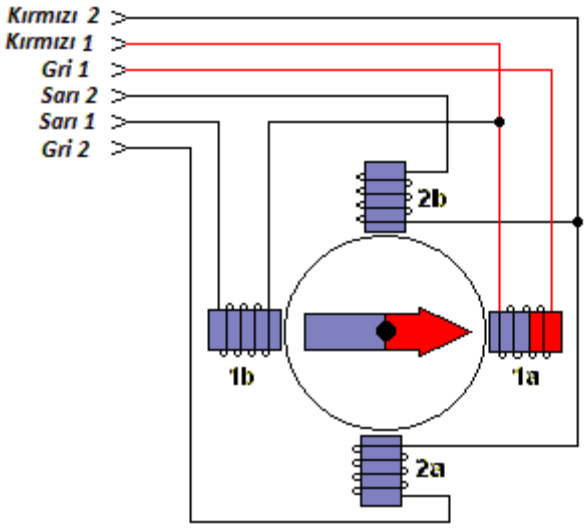
Tablo 8.1 MB11 kodlu Adımlı-motor özellikleri

Adımlı-motorların dönen kısmı (rotor) sabit mıknatıstan yapılmaktadır. Duran kısmında (stator) ise belirli aralıklarla yerleştirilmiş elektromıknatıslar bulunmaktadır. Elektromıknatısın içerisinden geçen akımın yönüne göre N-S kutuplarının yönü de değişir. Bir Adımlı-motorun döndürülmesi için belli bir sırayla bu elektromıknatısların enerjilenmesini sağlayan gerilimler motor uçlarından uygulanır. Böylece sabit mıknatıs, duran kısmın enerjilenen kutupları tarafından yönlendirilir.

	Sargı 1	Sargı 2	Sargı 3	Sargı 4	
Saat yönü ↓	1	1	0	0	Saat Yönünün Tersi ↑
	0	1	1	0	
	0	1	0	1	
	1	0	0	1	

Tablo 8.2 MB11 seri kodlu Adımlı-motorun adım dizesi

Bu deneyde kullanılan ve Tablo 8.1 ve 8.2'de özellikleri verilen Adımlı-motor "kalıcı mıknatıslı, bipolar Adımlı-motor" tipi olarak bilinir ve yapısı Şekil 8.2'de gösterilmiştir.

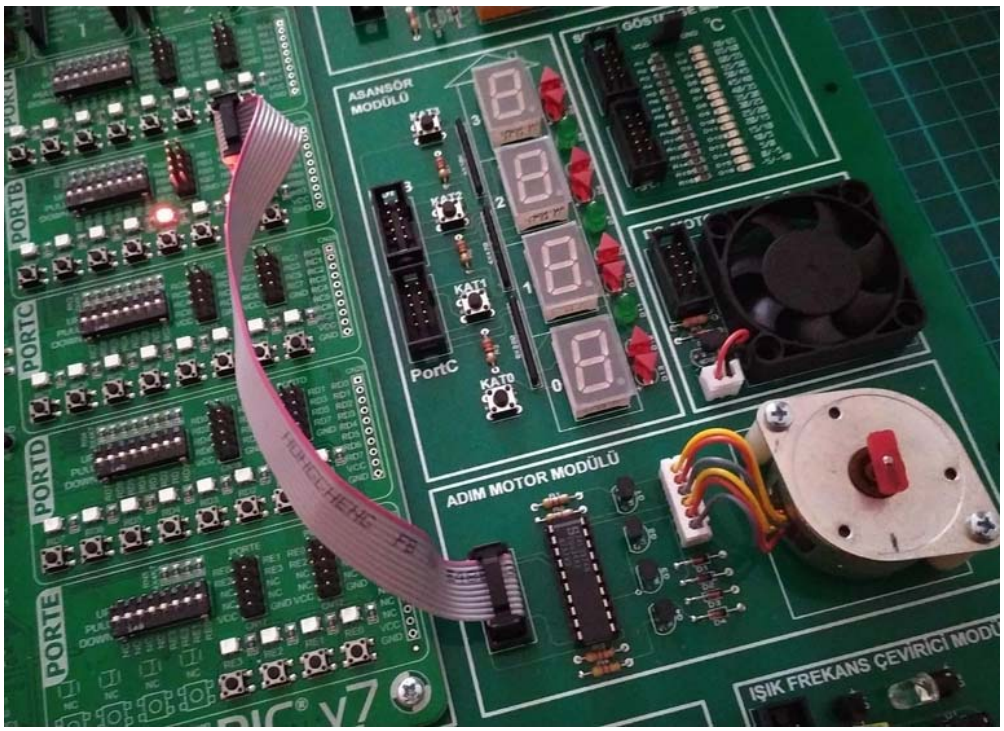


Şekil 8.2 Kalıcı mıknatıslı, paralel sargılı bipolar Adımlı-motor

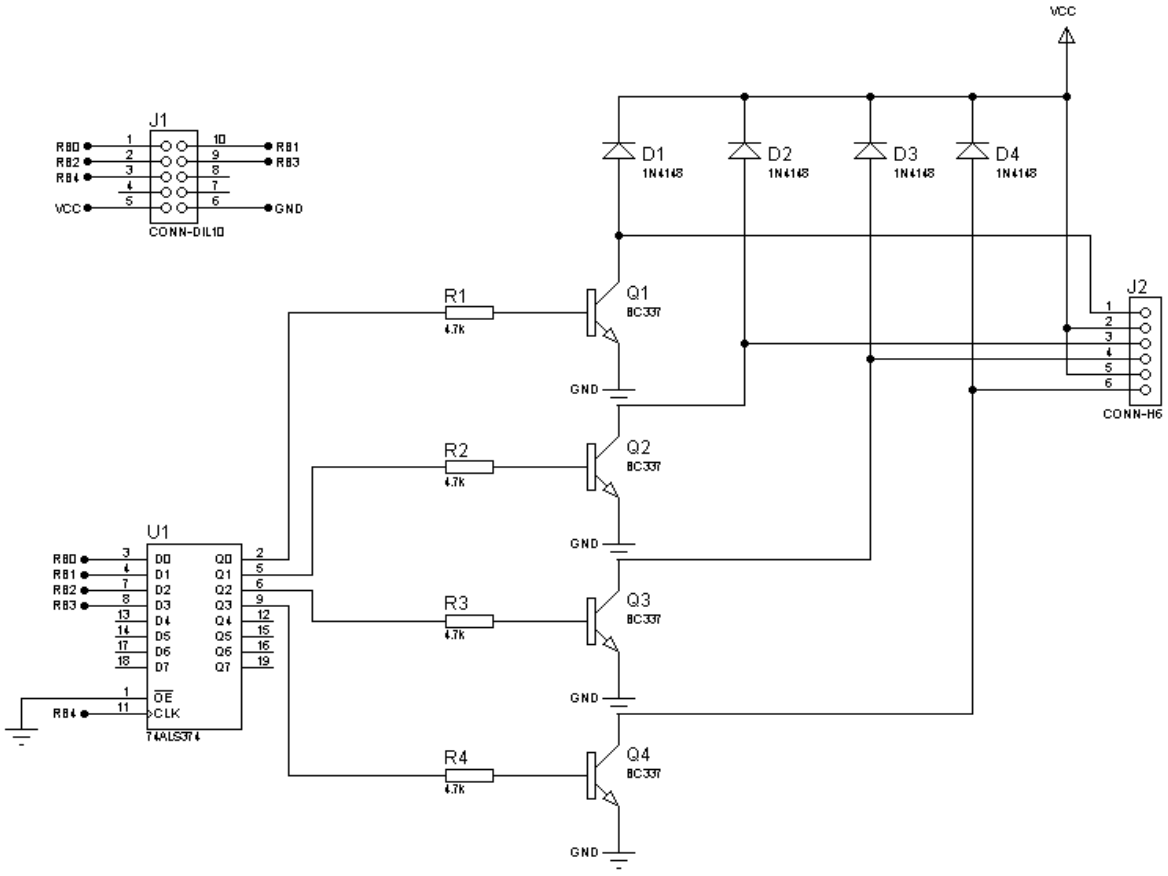
Basit yapısı ve düşük fiyat özelliği yüzünden bu tür motorlar endüstriyel olmayan uygulamalarda yaygın olarak kullanılmaktadır; bilgisayar yazıcıları, disket sürücüler v.b. Bipolar motorların dönüş hızı düşük olmasına karşın tork (güç) parametreleri yüksektir. Piyasadaki 8-uçlu motorlar seri veya paralel bağlantıya imkan verir. Bizim kullandığımız motor, 6-uçlu olup seri bağlantı için düzenlenmiştir. PIC Mikrodenetleyici'nin çıkış uçlarını Adımlı-motorun çıkış kablolarına direk bağlamak doğru değildir. Çünkü PIC18F45K22'nin çıkış pinlerinin 20-25 mA sink-source kapasitesi vardır. Adımlı-motorun çekeceği akım ise Tablo 8.1'den görüleceği gibi 175 mA civarındadır. Bundan dolayı PIC18F45K22 ile Adımlı-motor sürmek istenilirse Şekil 8.3'deki gibi transistörlü sürücü devre kullanılmalı ve akım kazancı sağlanmalıdır.

Aşağıdaki Tablo 8.3'te sadece Adımlı-motorun sargılarına sinyal vererek sabit hızda ve yönde dönmeyi sağlayacak bir program görülmektedir. Bu programda sargılara sırasıyla Tablo 8.2'de verilen ikilik değerler belirli bir gecikme aralığı ile uygulanarak motorun dönmesi sağlanmıştır. Bu gecikme aralığı, bir sargıya verilen sinyal ile diğer sargıya verilen sinyal arasında geçen süre olduğundan, girilen gecikme aralığı aynı zamanda motorun dönüş hızını belirlemiş olacaktır.

Bu uygulamada adım motoru tam tur ileri - tam tur geri, yarım tur ileri – yarım tur geri, çeyrek tur ileri – çeyrek tur geri şeklinde çalıştırılmıştır.



EasyPIC v7 kartı ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki Adım Motor Kontrolü arasındaki bağlantı flat kablo ile yukarıdaki fotoğraftaki gibi olmalıdır.



Şekil 8.3 Beti MCU Seti üzerinde bulunan Adımlı Motor Uygulama Devre Şeması

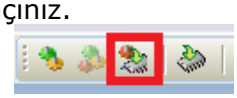
Örnek program

```
//***********************
//Proje ismi     : DENEY 8- STEP (ADIMLI) MOTOR KONTROLU DENEYİ
//Amacı          : B Portunu çıkış olarak ayarlayıp, adım
//                 motoru sürmek için gerekli verileri
//                 göndererek adım motorun  yönünü ve hareketini
//                 kontrol etmek.
// Test konfigürasyonu       :
//---------------------------
// MCU           : 18F45K22
// Osilatör      : 32Mhz
//
// SW1=>OFF
// SW2=>OFF
// SW4=>OFF
// SW3.2=>ON, diğerleri OFF
//
//***********************


//Step motor bağlantıları
sbit sari1    at LATB.B0;
sbit gri1     at LATB.B1;
sbit sari2    at LATB.B2;
sbit gri2     at LATB.B3;
sbit clk      at LATB.B4;

unsigned int adimSayac=0;          // adım sayan ve kaldığı yerei tutan değişken

// ileriye doğru adım atan fonksiyon.
void turAtIleri(unsigned int turSayisi)    // Fonksiyon kaç tur atılacağını
{                                           // değer olarak alır ve o kadar adım atar.
 unsigned int sayac=0;                      // Fonksiyon içerisindeki tur sayısını tutacak değişken.
 while (sayac<turSayisi)                   // istenilen tur sayısına erişilene kadar adım atan
döngü.
 {
  sayac++;
  adimSayac++;
  adimSayac%=4;
  switch(adimSayac)                        // sırasıyla sargılara gerekli akımlar uygulanıyor.
  {
   case 0: sari1=0; gri1=1; sari2=0; gri2=1; break;
   case 1: sari1=1; gri1=0; sari2=0; gri2=1; break;
   case 2: sari1=1; gri1=0; sari2=1; gri2=0; break;
```

```
   case 3: sari1=0; gri1=1; sari2=1; gri2=0; break;
   default: break;
  }
  clk=1;                                  // 74LS374 entegresine clock uygulanarak
  delay_ms(200);                          // girişindeki veri çıkışa aktarılıyor.
  clk=0;
 }
}

// geriye doru adım atan fonksiyon.
void turAtGeri(unsigned int turSayisi)    // Fonksiyon kaç tur atılacağını
{                                         // değer olarak alır ve o kadar adım atar.
 unsigned int sayac=0;                    // Fonksiyon içerisindeki tur sayısını tutacak değişken.
 while (sayac<turSayisi)                  // istenilen tur sayısına erişilene kadar adım atan
döngü.
 {
  sayac++;
  adimSayac--;
  adimSayac%=4;
  switch(adimSayac)                       // sırasıyla sargılara gerekli akımlar uygulanıyor.
  {
   case 0: sari1=0; gri1=1; sari2=0; gri2=1; break;
   case 1: sari1=1; gri1=0; sari2=0; gri2=1; break;
   case 2: sari1=1; gri1=0; sari2=1; gri2=0; break;
   case 3: sari1=0; gri1=1; sari2=1; gri2=0; break;
   default: break;
  }
  clk=1;                                  // 74LS374 entegresine clock uygulanarak
  delay_ms(200);                          // girişindeki veri çıkışa aktarılıyor.
  clk=0;
 }
}

// Ana fonksiyon
void main()
{
char i=0;
TRISB=0;                                  // B portu çıkış olarak ayarlandı.
LATB=0;                                   // B portunun pinlerine 0 gönderildi.
turAtIleri(48);                           // ileri doğru 1 tam tur atar.
delay_ms(2000);                           // 2 saniye bekle.
turAtGeri(48);                            // geriye doğru 1 tam tur atar.
delay_ms(2000);

turAtIleri(24);                           // ileri ve geriye doğru yarım tur atar.
delay_ms(2000);
turAtGeri(24);
```

```
delay_ms(2000);

turAtIleri(12);                    // ileri ve geri doğru çeyrek tur atar.
delay_ms(2000);
turAtGeri(12);
delay_ms(2000);
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 7-ANALOG SICAKLIK ÖLÇÜM UYGULAMASI" içerisindeki "Deney7.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki Adım Motor Modülü'nü gözlemleyiniz.

- **Adım motorun hızını değiştirmek için kodda ne gibi bir değişiklik yapmamız gerektiğini düşününüz.**
- **Adım motoru daha hassas döndürmek için neler yapılmalıdır?**
- **"Microstepping" kavramını araştırınız.**

DENEY NO 9. TUŞ TAKIMI UYGULAMASI

AMAÇ

1) Tuş takımının kullanımını anlamak.
2) LCD modülde tuşlara basarak, basılan tuşları LCD ekranda göstermek.

GEREKLİ MALZEMELER

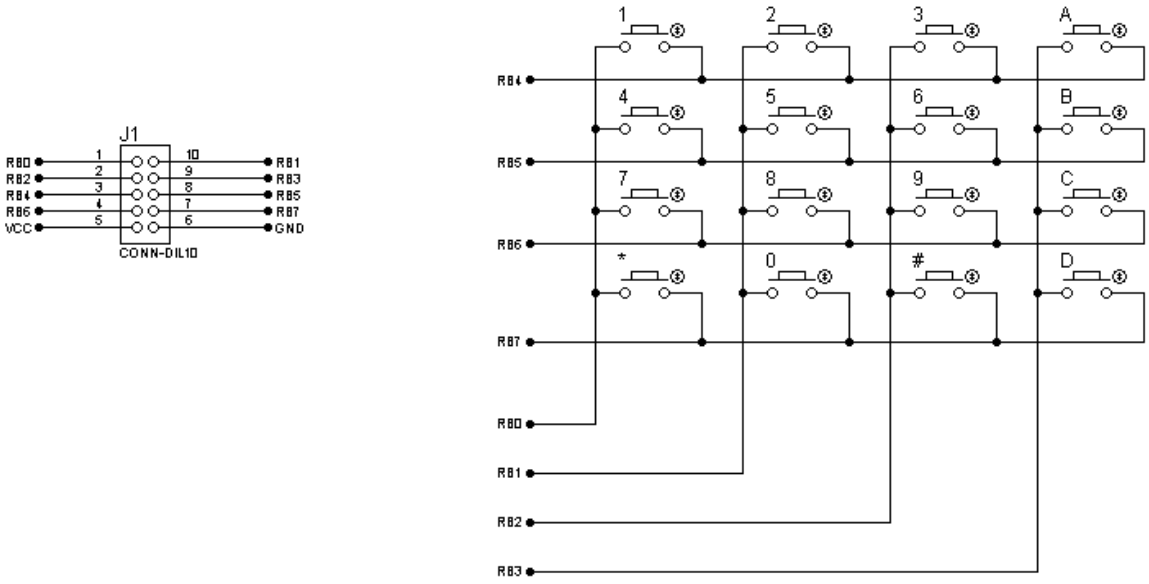
1) EasyPIC v7 Kartı
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti
3) LCD modül

Giriş

Bu uygulamamızda MikroC derleyicisinin "KeyPad" kütüphanesinden faydalanacağız. Bu kütüphanenin fonksiyonları ve işlevleri aşağıdakigibidir.

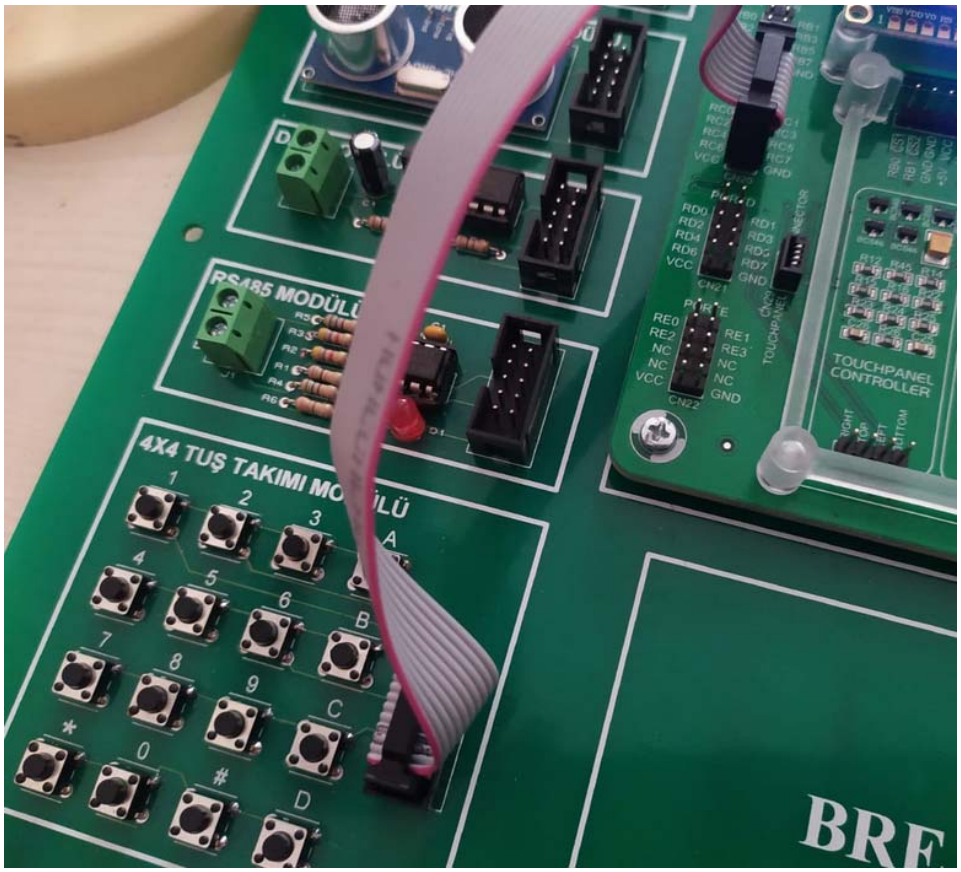
- **Keypad_Init**: Tuştakımının bağlı olduğu portu hazırlar.
- **Keypad_Key_Press**: Basılan tuşun değerini basıldığı anda verir.
- **Keypad_Key_Click**: Basılan tuşun değerini bırakıldığında verir.

Bu fonksiyonlar ile Şekil 9.1'deki şemadaki bağlantılar baz alınarak kullanılabilir. Bu kütüphane ile kullanılacak tuş takımının boyutu 4x4 olmalıdır.



Şekil 9.1 Tuş Takımı Devre Şeması

Bu uygulamada tuş takımından basılan tuşlar ve kaçar kez basıldıkları LCD'de gösterilmektedir.



EasyPIC v7 kartı ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki 4x4 Tuş Takımı Modülü arasındaki bağlantı yukarıdaki fotoğraftaki gibi olmalıdır.

Örnek program

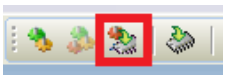
```
//************************
//Proje ismi        : DENEY NO 9. TUŞ TAKIMI UYGULAMASI
//Amacı             : 4x4 tuş takımından hangi
//                    tuşun kaç kere basıldığını
//                    LCD ekranda gözlemlemek.
//Test konfigürasyonu       :
//-----------------------------------
// MCU              : 18F45K22
// Osilatör         : 32 Mhz
//
// SW3.3     => ON, Diğerleri OFF
//
// EASYPIC v7 kartındaki PORTC modülündeki anahtarlar
// PULL_DOWN konumuna alınmalıdır.
//
// Library Manager Sekmesinden Conversions, Keypad4x4 ve Lcd kütüphaneleri
// işaretlenmelidir.
//
//************************
```

```
unsigned short tus_kodu, sayac, tus_kodu_gecmis = 0;
char txt[6];

// TusTakımı  modül bağlantıları
char keypadPort at PortC;


// Lcd Modül Bağlantıları
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// Lcd Modül Bağlantıları biter


void main()
{
 sayac = 0;                                  // sayacı sıfırlar
 ANSELB=0;
 ANSELC=0;
 Keypad_Init();                              // tus takımı başlangıç ayarları

 Lcd_Init();                                 // LCD ilklendirme fonksiyonu
 Lcd_Cmd(_LCD_CLEAR);                        // LCD ekranını temizler.
 Lcd_Cmd(_LCD_CURSOR_OFF);                   // Cursor ekranda gösteilmez.

 LCD_Out(1, 1, "Tus   :");                   // LCD ekranına yazılır.
 LCD_Out(2, 1, "Sayac :");

 do
  {
    tus_kodu = 0;                           // tus_kodu sıfırlanır.
    do                                      // Tuşa basılması beklenir.
    {
     tus_kodu = Keypad_Key_Click();         // tus kodunun değerini alır.
    }while (!tus_kodu);

  // LCD ekranında görüntülemek için ASCII'ye çevirilecek.
```

```
    switch (tus_kodu)
    {
      case  1: tus_kodu = 49; break; // 1
      case  2: tus_kodu = 50; break; // 2
      case  3: tus_kodu = 51; break; // 3
      case  4: tus_kodu = 65; break; // A
      case  5: tus_kodu = 52; break; // 4
      case  6: tus_kodu = 53; break; // 5
      case  7: tus_kodu = 54; break; // 6
      case  8: tus_kodu = 66; break; // B
      case  9: tus_kodu = 55; break; // 7
      case 10: tus_kodu = 56; break; // 8
      case 11: tus_kodu = 57; break; // 9
      case 12: tus_kodu = 67; break; // C
      case 13: tus_kodu = 42; break; // *
      case 14: tus_kodu = 48; break; // 0
      case 15: tus_kodu = 35; break; // #
      case 16: tus_kodu = 68; break; // D
    }

    if (tus_kodu != tus_kodu_gecmis)
    {// basılan tus bir önceki basılandan farklı ise
      sayac = 1;
      tus_kodu_gecmis = tus_kodu;
    }
    else
    {                                   // basılan tus bir önceki basılandan aynı ise
      sayac++;
    }

    Lcd_Chr(1, 14, tus_kodu);           // basılan tus LCD ekranında gösterilir.

    if (sayac == 255)
    {// sayac 255'e ulaşırsa
     sayac = 0;
     LCD_Out(2, 10, "   ");
    }

    WordToStr(sayac, txt);              // sayacdeğeri  string  tipine çevirilir.
    LCD_Out(2, 10, txt);                // ve LCD üzerinde görüntülenir.
   }while (1);
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.

3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 9-TUŞ TAKIMI UYGULAMASI" içerisindeki "Deney9.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki 4x4 Tuş Takımı Modülü üzerindeki tuşlara basarak LCD'deki değişimleri gözlemleyiniz.

• 4 Haneli bir şifre belirleyerek tuş takımından bu şifre doğru sıralamda girildiğinde LCD ekranda "Şifre doğru" mesajı gösteren bir program yazınız.

DENEY 10. UART SERİ HABERLEŞME DENEYİ

AMAÇ

1) Seri haberleşme yöntemlerinden birisi olan UART protokolünü tanıtmak, donanım ve yazılım gereksinimlerini göstermek,
2) EasyPICv7-PC arasında çift yönlü veri transferini örneklemek.

GEREKLİ MALZEMELER

1) EasyPICv7 Kartı,
2) Kişisel Bilgisayar (PC),
3) USB A/B Kablosu

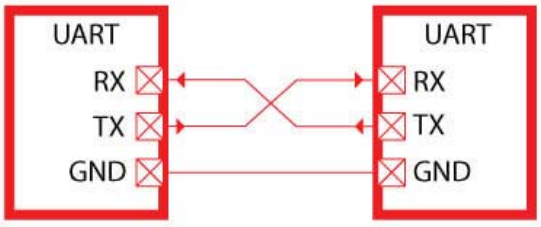
Giriş:

Basitliğinden ve paralel iletişime göre daha az donanım gereksiniminden dolayı, seri iletişim günümüz elektronik endüstrisinde en fazla kullanılan yöntemdir. Seri yöntemler arasında en popüler olan ise UART protokolüdür. Açılımı; universal asynchronous receiver- transmitter olan uart haberleşmenin amacı da verileri seri bir şekilde iletmek ve de almaktır. UART, bunun yanında 2 telli olmaktadır. Bu konuda seri veriler Tx, yani verici ve Rx- alıcı pinleri tarafından yönetilmektedir. UART, verilerini asenkron iletmekte, bunun sonucunda da veri gönderme ve alma süreçlerinde hiçbir saat sinyali ilişkilendirilemez.

UART Özellikleri:

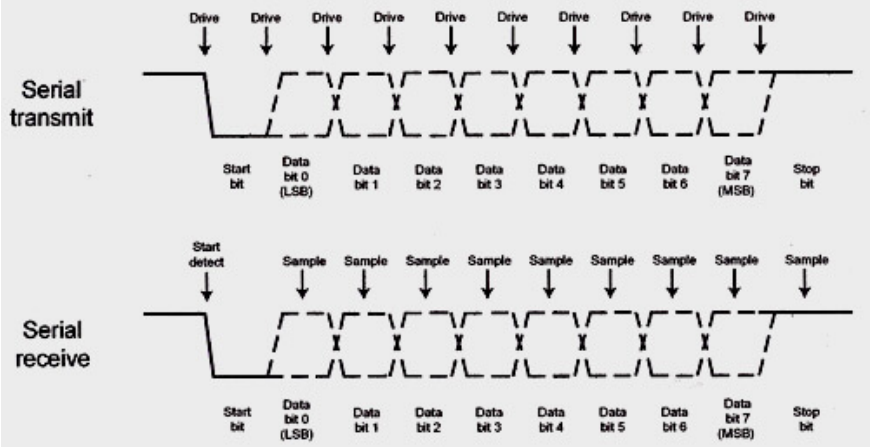
UART haberleşmesi gerçekleştirirken ilk başta baudrate yani veri taşıma hızının ayarlanması gerekmektedir. Bunun yanında veri taşıma hızı çok farklı aralıklarda da olmaktadır. Fakat piyasada çok sık bir şekilde kullanılan baudrate'ler; 4800- 9600- 57600-115200 ve mikro işlemciler için fazla tercih edilmese de; 921600 genel olarak çok fazla hızlı işlem gereksinimi olan yerlerde kullanılmaktadır.

Genel olarak, haberleşme işlemi bir başlangıç bitinden hemen sonra data bitleri, sonrasında da parity biti ve en sonda da bitiş biti gönderilip, sonlandırılır. Bu işlem süreçlerinde data uzunluğu ve parity biti opsiyonel bir şekilde değişkenlik gösterebilmektedir.



Şekil 10.1: İki cihazın UART iletişim protokolünde haberleşirken kullanılan bağlantıları

Seri kanalda gönderilen seri darbe dizisi sadece veri bitlerinden oluşmaz, veri bitlerine bazı formatlama bitleri eklenir ki alıcı sistem bu bitler arasındaki 8-bit veri bilgisini çözebilir. (Şekil 10.2)



Şekil 10.2 UART iletişim formatı

Şimdi bu seri sinyal formatındaki tanımlamaları verelim:

Baud Hızı (Baud Rate): Seri iletişim yapan iki cihaz arasındaki verinin hangi hızla gönderildiğini belirler; birimi bit/saniyedir, örneğin 9600 baud=9600 bit/saniye.

Başlangıç Biti (Start bit): Seri veri paketinin başlangıcını belirleyen işaretleme bit'idir; sinyal olarak HIGH (yüksek) seviyeden LOW (düşük) seviyeye geçiş olarak tannımalnabilir

Veri bitleri (Data Bits): Veri bitleri "ters döndürülmüş ve geriye" doğru gönderilir, yani daha önce de belirttiğimiz gibi negatif mantık uygulanır ve gönderme sırası En Düşük Ağırlıklı Bit'den (LSB), En Yüksek Ağırlıklı Bit'e (MSB) doğrudur.

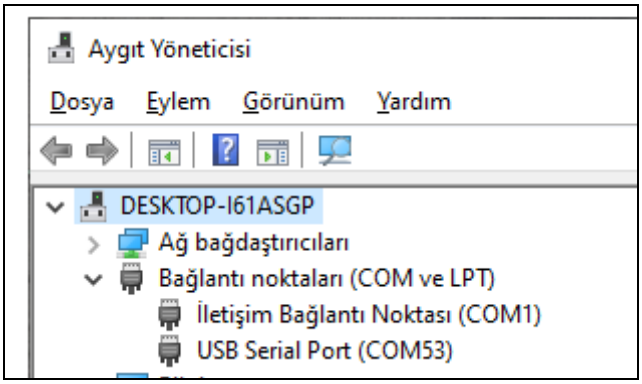
Eşlik biti (Parity bit): Kullanımı opsiyoneldir ve veri bitlerinde olduğu gibi negatif mantık konvansiyonundadır. Kullanım amacı seri kanalda oluşacak hataları kontrol etmektir; öncelikle iletişimde "çift" veya "tek" parite seçeneklerinden hangisinin uygulanacağına karar verilir. Örneğin "tek" parite seçeneği kullanılsın, bu durumda veri gönderen sistem veri ve parite bitlerinde toplam 1'lerin sayısı "tek" olacak şekilde parite bitini set eder.

Durdurma Bitleri (Stop bits): Seri paketin son karakterleri 1, 1.5, veya 2 stop bit'dir. Bu bitler daima pozitif voltaj seviyesindedir. Eğer daha başka paket transfer edilmiyor ise hat pozitif voltaj seviyesinde (SİNYAL) bekleyerek kalır.

UART Deneyi:

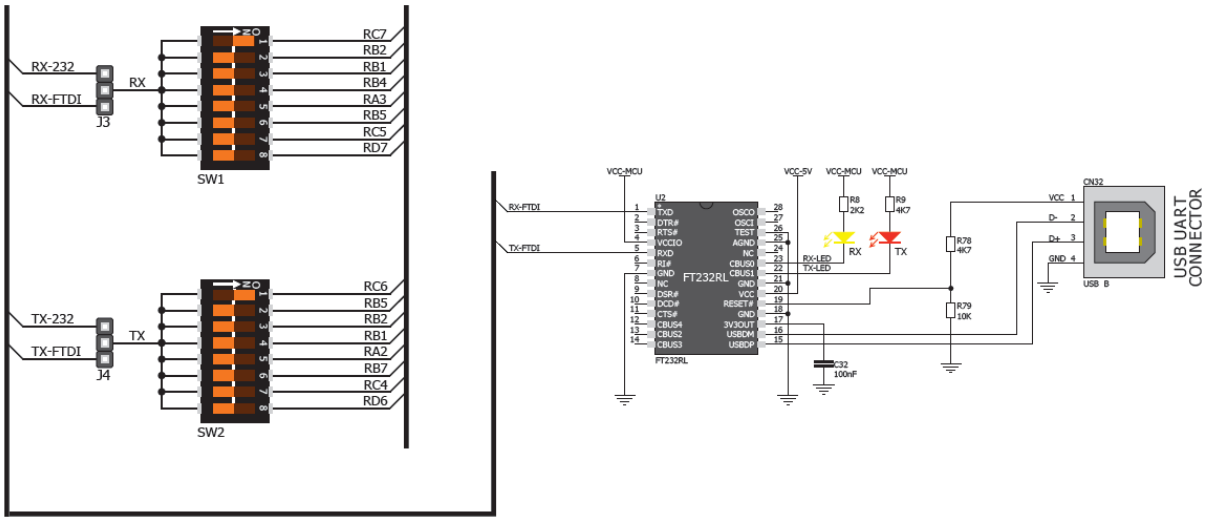
Easy PICv7 kartı üzerindeki UART uygulamasının devre şeması Şekil 10.3'te görüldüğü gibidir. Şemada gördüğünüz J3 ve J4 jumperları, PC ile bağlantının RS232 konektörü ya da USB konektörü ile mi yapılacağının seçildiği jumperdır. Bu uygulamamızda USB UART kısmında konumlandırılacaktır. SW1 ve SW2, FT232RL entegresinin RX ve TX pinlerinin Mikrodenetleyicinin hangi pinleri ile iletişim sağlayacağının seçildiği switchlerdir. SW1.1 ON konumuna getirilerek FT232RL entegresinin RX pini MCU'nun RC7, SW2.1 ON konumuna getirilerek FT232RL entegresinin TX pini MCU'nun RC6 pinleri ile ilişkilendirilmiştir.

Bu uygulamada FT232RL isimli UART/USB dönüştürücü çipi kullanılarak PC üzerinde bir sanal COM port oluşturulur. Bu COM port kullanılarak PC-Mikrodenetleyici arasında çift yönlü veri iletişimi sağlanabilir. Deney için kulanılacak PC'de bulunan Aygıt Yöneticisi penceresinde aşağıdakine benzer bir USB Serial Port görülmelidir.



Bu işlem için FT232RL çipine ait sürücünün önceden PC'ye yüklenmesi gerekmektedir.

Bu uygulamada mikroC pro for Pro derleyicisi ile birlikte sunulan ve "Tools" sekmesi altında bulunan "USART Terminal" aracı kullanılabilir. Bu araç sayesinde mikrodenetleyiciden gönderilen veriler PC ekranında görülebilir. Uygulamada PC'den gönderilecek veriler EasyPIC v7 kartı üzerindeki LCD'de gösterilecek, EasyPIC v7 kartı üzerinde bulunan PORTD kısmındaki basılan butonlar ise USART Terminal aracında gösterilecektir.



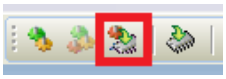
Şekil 10.3

Örnek program:

```
//************************
//Proje ismi        : DENEY NO 10. UART SERİ HABERLEŞME DENEYİ
//Amacı             : PC<->MCU arası seri iletişim uygulaması yapmak
//Test konfigürasyonu       :
//-----------------------------------
// MCU              : 18F45K22
// Osilatör         : 32 Mhz
//
// SW1.1     => ON, Diğerleri OFF
// SW2.1     => ON, Diğerleri OFF
// SW3.4     => ON, Diğerleri OFF
// SW4.6     => ON, Diğerleri OFF
//
// J3 ve J4 ataçları USB UART konumuna alınmalıdır.
// J12 jumperları I/O konumunda olmalıdır.
// J17 VCC konumunda olmalıdır.
//
// Library Manager Sekmesinden UART ve Lcd kütüphaneleri işaretlenmelidir.
//
//************************

// Lcd Modül Bağlantıları
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// Lcd Modül Bağlantıları biter


unsigned int lcdSayac=0;
char gelenVeri;
char portDurum;

void main()
{
```

```
  UART1_Init(115200);                  // UART iletişim 115000 bps olarak ayarlanıyor.

  ANSELC=0;                            // PORTC dijital olarak ayarlanıyor.

  ANSELD=0;                            // PORTD dijital olarak ayarlanıyor.
  TRISD=0xFF;                          // PORTD dijital giriş olarak ayarlanıyor.

  Lcd_Init();                          // LCD ilklendirme fonksiyonu
  Lcd_Cmd(_LCD_CLEAR);                 // LCD ekranını temizler.
  Lcd_Cmd(_LCD_CURSOR_OFF);            // Cursor ekranda gösterilmez.


  while(1)
  {
   // UART üzerinden veri geldiyse veriyi al ve LCD'de göster.
   if(UART1_Data_Ready())
   {
    lcdSayac++;
    gelenVeri= UART1_Read();
    if(lcdSayac<17) Lcd_Chr(1,lcdSayac,gelenVeri);
    if(lcdSayac>=17 && lcdSayac<33) Lcd_Chr(2,lcdSayac-16,gelenVeri);
    if(lcdSayac==33)
    {
     lcdSayac=0;
     Lcd_Cmd(_LCD_CLEAR);
    }
   }

   portDurum=PORTD;
   switch(portDurum)
   {
    case 1: UART1_Write_Text(" Basilan Tus: RD0\r\n"); while(PORTD==1); break;
    case 2: UART1_Write_Text(" Basilan Tus: RD1\r\n"); while(PORTD==2); break;
    case 4: UART1_Write_Text(" Basilan Tus: RD2\r\n"); while(PORTD==4); break;
    case 8: UART1_Write_Text(" Basilan Tus: RD3\r\n"); while(PORTD==8); break;
    case 16: UART1_Write_Text(" Basilan Tus: RD4\r\n"); while(PORTD==16); break;
    case 32: UART1_Write_Text(" Basilan Tus: RD5\r\n"); while(PORTD==32); break;
    case 64: UART1_Write_Text(" Basilan Tus: RD6\r\n"); while(PORTD==64); break;
    case 128: UART1_Write_Text(" Basilan Tus: RD7\r\n"); while(PORTD==128); break;
    default: break;
   }
  }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 10-UART SERİ HABERLEŞME DENEYİ" içerisindeki "Deney10.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. MikroC pro for PIC Derleyicisi içerisindeki Tools sekmesinden USART Terminal aracını çalıştırarak 115200 baud rate ayarında kartla bağlantıyı sağlayınız.
7. USART Terminal aracının Send kısmından karakterler göndererek EasyPIC v7 üzerindeki LCD'yi gözlemleyiniz.
8. EasyPIC v7 kartı üzerindeki PORT kısmındaki tuşlara basarak USART Terminal aracının Receive kısmındaki değişilikleri gözlemleyiniz.

- **Kablosuz veri iletişiminde UART kullanımını araştırınız.**

DENEY 11. RÖLE KONTROL DENEYİ

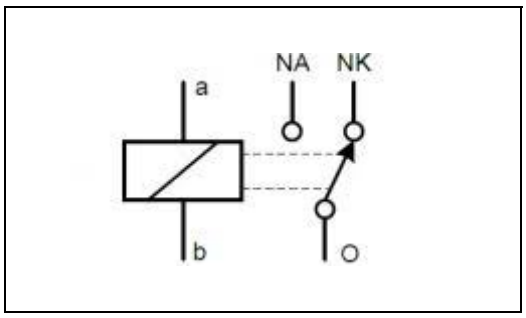
AMAÇ

1) Röle elemanını tanımak, röle kullanımının gereklerini anlamak,
2) EasyPIC v7 anakartı üzerinde PIC18F45K22 Port'larından röle sürülmesini örneklemek.

GEREKLİ MALZEME

1) EasyPIC v7 Kartı,
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti,

Giriş:



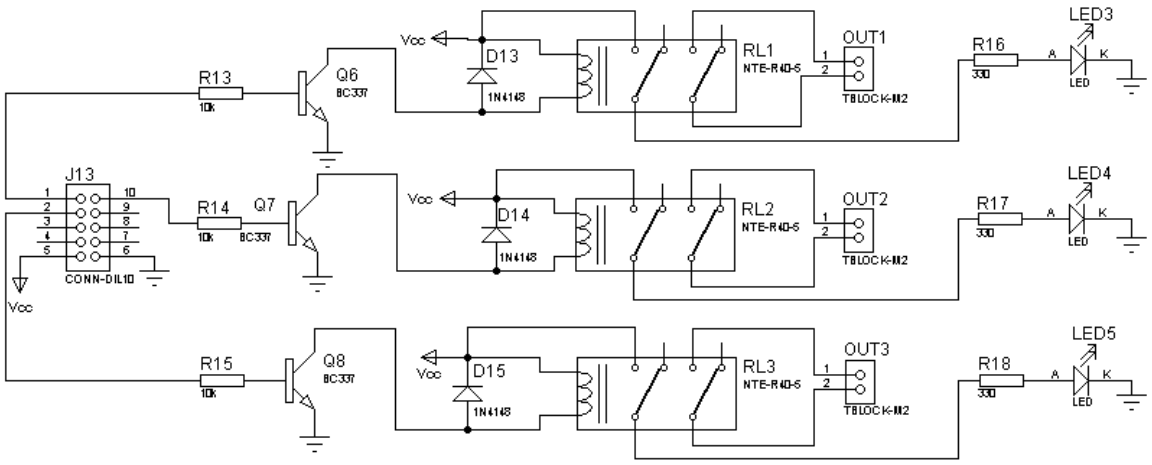
Şekil 11.1 Röle Sembolü

Röle elektriksel sinyali mekanik harekete dönüştüren elektromekanik bir elemandır. Metal bir çekirdek üzerine yalıtılmış telden sarılmış bir sargı ile bir veya birden fazla kontağı olan metal armatürden oluşur. Sargıya DC bir güç uygulandığında sargıdan akım geçer bu akımın oluşturduğu manyetik alan armatürün kapalı kontağını açar, açık kontağını kapar. Röleye bağlı güç kapatıldığında bobindeki manyetik alan çöker ve ters yönde kuvvetli bir voltaj oluşturur ki bu ters voltaj da röleye bağlı transistorü bozabilir. Bu nedenle röle sargılarına ters besleme konumunda bir yarı-iletken diyot bağlanarak oluşan yüksek voltajın tekrar bobin üzerine doğru akıtılarak sönümlenmesi sağlanır.

PIC18F45K22 portlarının maksimum akım sürme kapasitesi 25 mA'dir. Ancak röle sargısı yaklaşık 100 mA çeker. Bu nedenle araya bir transistör bağlanarak akım kazancı yükseltilir; röle transistorün kollektör bacağına bağlanır. Transistör beyz ucuna mantık (1) seviyesi uygulandığında, transistör doyuma girer ve röle bobinin üzerinden akım geçmesi sağlanmış olur. Transistör beyzine bağlanan direnç rölenin beyz akımını sınırlamak içindir. Bu nedenle RB0'dan çıkan mantık (1) seviyesindeki +5V röleyi tetikler.

Röle ayrıca bir optik-kuplaj entegresi ile de sürülebilir. Optik-kuplaj devresi hem akım kazancı sağlar ve hem de dış bağlantı devresini Mikrodenetleyici'den izole eder. Genellikle yüksek akım optik-kuplaj entegrelerinin çıkışlarında "Darlington Transistör" bağlantısı vardır; bu nedenle bu entegreler yüksek akım sağlayabilirler.

Optik-kuplaj entegresi bağlantısı genellikle Mikrodenetleyici'nin yüksek güçte bir yük sürme gereği olduğu durumlarda gerekir; örneğin motor veya ısıtıcı gibi yükler voltaj kararsızlığından dolayı Mikrodenetleyici'yi riskli duruma sokar. Bizim deneyimizde uyarılan rölenin kontakları kart tipi klemens'lere taşınmış ve yük konusunda kullanıcıya esneklik tanınmıştır. En basit uygulamayla röle kontakları 5 Voltluk bir LED'i sürüp yakabilir.



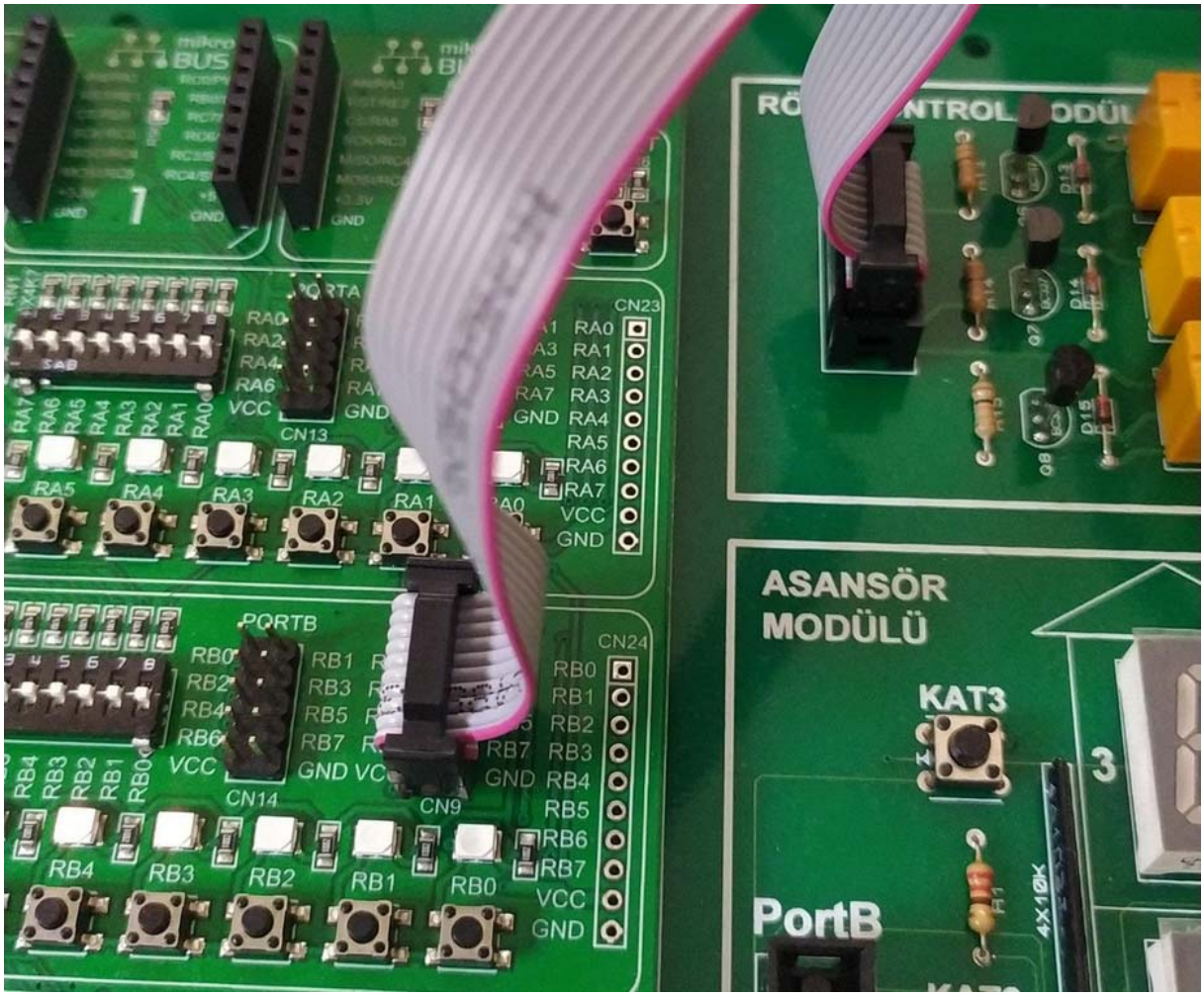
Şekil 11.2 Beti MCU Uygulama Setinde Kullanılan Röle Kartının Devre Şeması

Bu uygulamada EASYPIC v7 kartının üzerindeki PORTD kısmındaki RD0 RD1 ve RD2 pinlerine bağlı butonlara basıldığında BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki Röle Kontrol Modülü'ndeki röleler tetiklenecektir.

Örnek program:

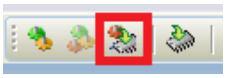
```
//***********************
//Proje ismi     : RÖLE KONTROL DENEYİ
//Amacı          : Giriş olarak ayarlanan D portuna,
//                 butonlar ile girilen veriyle,
//                 çıkış olarak ayarlanan B
//                 portundaki röleleri kontrol etmek.
//Test konfigürasyonu       :
//-----------------------------------
// MCU           : 18F45K22
// Osilatör      : 32Mhz
//
// J17 VCC konumunda olmalıdır.
// SW3.2,3.4     => ON, Diğerleri OFF
//
//***********************

void main()
{
  ANSELB = 0;              // Port B dijital olarak ayarlandı
```

```
  ANSELD = 0;             // Port D dijital olarak ayarlandı

  TRISD=0x07;           // PORTD'nin LSB 3 biti giriş olarak ayarlandı.
  LATD=0;               // PORTD'nin çıkış olarak ayarlanan pinlerine 0 gönderildi.
  TRISB=0xF8;           // PORTB'nin LSB 3 biti çıkış olarak ayarlandı.
  LATB=0;               // PORTB'nin çıkış olarak ayarlanan pinlerine 0 gönderildi.

  // PORTB'nin LSB 3 bitindeki butonlara basıldığında PORTD'nin
  // LSB 3 bitindeki röleleri çektiren döngü.
  while(1)
  {
   LATB.B0=PORTD.B0;
   LATB.B1=PORTD.B1;
   LATB.B2=PORTD.B2;
  }
}
```



EasyPIC v7 kartı ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki PORTB bölümüyle Röle Kontrol Modülü arasındaki bağlantı yukarıdaki fotoğraftaki gibi olmalıdır.

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 11-RÖLE KONTROL DENEYİ" içerisindeki "Deney11.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. EasyPIC v7 kartı üzerindeki PORTD kısmında bulunan RD0, RD1 ve RD2 butonlarına basarak Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki Röle Kontrol Modülündeki röleleri gözlemleyiniz.

- **Kendinizin belirleyeceği bir butona basıldıktan 5sn sonra çalışan (röle kontağının çekmesi) ve 15 sn. sonra duran (röle kontağının bırakması) uygulamasını yazıp set üzerinde deneyiniz.**

DENEY 12. DARBE GENİŞLİK MODÜLASYONU (PWM) UYGULAMASI

AMAÇ

1) Darbe Genişlik Modülasyonu (PWM) kavram ve gereksinimini öğretmek,
2) PIC18F45K22 Mikrodenetleyici'nin PWM çıkışlarını ve özelliklerini tanımak,
3) PWM uygulamasının kullanım alanlarını örneklemek.

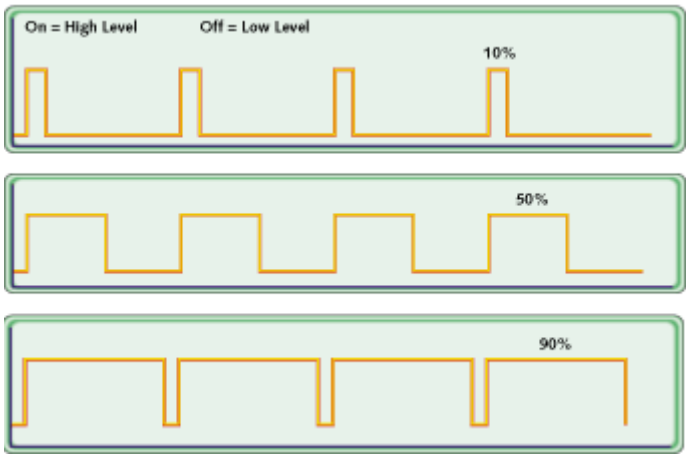
GEREKLİ MALZEME

1) EasyPICv7 Kartı,
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti

Giriş:

Darbe Genlik Modülasyonu (Pulse Width Modulation); dijital sistem çıkışları ile analog devrelerin ve elemanların kontrolünde kullanılan güçlü bir tekniktir. PWM yöntemi; ölçümden iletişime ve güç kontrolünden güç dönüşümüne kadar çok sayıda uygulamada kullanılmaktadır.

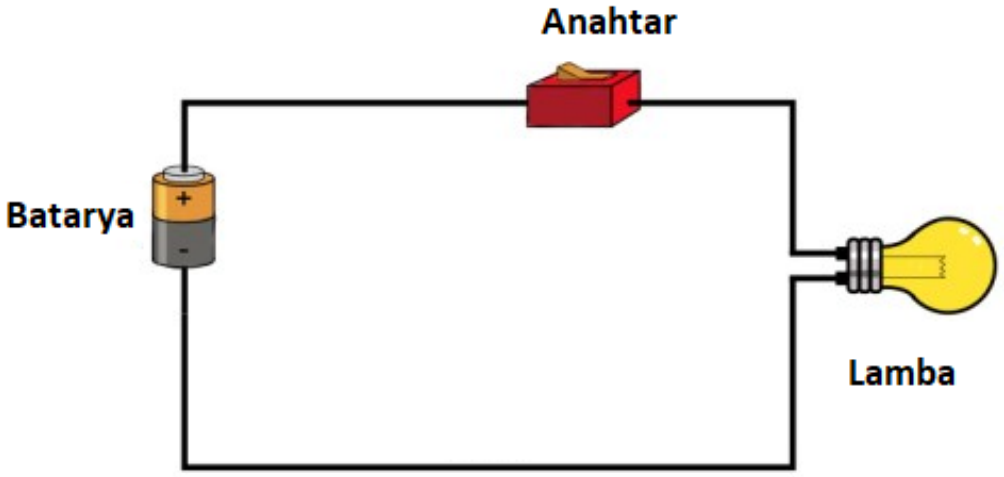
Şekil 12.1'de üç ayrı PWM sinyali gösterilmiştir. Şekil 12.1a'da; görev saykılı (Duty Cycle) %10 olan bir PWM çıkış gösterilmiştir. Yani bu örnekte sinyal periyodun %10 diliminde "On", %90 diliminde ise "Off"dur. Şekil 12.1b'de ve 1c'de PWM çıkışları %50 ve %90 duty cycle olan PWM sinyalleri örneklenmiştir. Bu 3 değişik PWM çıkışları 3 ayrı örneksel sinyal seviyesini, tam gücün %,10, %,50 ve %90 değerine kodlamaktadır. Örneğin, örneksel voltajın tam değeri 9 V ise, %10 duty cycle, 0.9 V örneksel sinyal seviyesini üretir.



Şekil 12.1 Değişik Duty Cycle oranlarında PWM sinyalleri

Şekil 12.2'de PWM ile sürülebilen, çok basit bir örneksel devre gösterilmiştir. Devrede, 9 V batarya Flaman'lı bir ampulü yakmaktadır. Eğer anahtarı 50 ms süre ile kapatırsak, ampul bu süre zarfında 9 V alacaktır. Daha sonra 50 ms süre ile anahtarı açarsak, ampul bu sürede 0 V alacaktır. Bu işlemi saniyede 10 kere tekrarlarsak, ampul

sanki bir 4.5 V bataryaya bağlanmış kadar parlak yanacaktır. (%50X9 V=4.5 V). Bu uygulamada duty cycle %50, modülasyon frekansı ise 10 Hz'dir.



Şekil 12.2. Basit bir PWM devresi

Pratikte endüktif ve kapasitif yükler, 10 Hz'den çok daha yüksek modülasyon frekansı gerektirir. Düşünelim ki bir önceki örneğimizde ampul 5 saniye yansın, 5 saniye sönsün, tekrar 5 saniye yansın. Bu durumda da duty cycle %50dir, ancak ampul ilk 5 saniye süresince parlak yanacak, ikinci 5 saniye süresince ise sönük kalacaktır. Ampulün 4.5 volt bir güç görebilmesi için, frekansın yükün reaksiyon zamanına göre hızlı olması gerekir. İstenen dimmer seviyesine ulaşmak (ancak ampul sürekli yanık olacak) için, modülasyon frekansını arttırmak gerekir. Bu temel özellik diğer tüm PWM uygulamaları için de geçerlidir. Genelde birçok uygulama için modülasyon frekansı 1 kHz'den 200 kHz'e kadar değişmektedir.

İletişim ve Denetleme:

PWM metodunun en önemli avantajlarından birisi, sinyalin işlemciden kontrol edilen sisteme kadar dijital kalmasıdır; bu nedenle dijital /analog dönüşüm kesinlikle gerekmez. Sinyali dijital tutmak gürültü etkilerini de minimize etmek demektir. Gürültünün bir dijital sinyali etkileyebilmesi için mantık-1 seviyesini, mantık-0 seviyesine, veya mantık-0 seviyesini, mantık-1 seviyesine dönüştürecek kadar güçlü olması gerekir.

Gürültüye karşı bağışıklık özelliği PWM uygulamasının analog denetime karşı tercih edilme sebeplerinin en önemlilerinden biridir. Bu sebeple PWM, bazen iletişim (Telekomünikasyon) alanında da kullanılır. Analog sinyalden PWM'na dönüşüm iletişim kanallarının dramatik olarak uzatılabilmesine imkan yaratmaktadır. Kanalın alım (receiver) ucunda uygun seçilmiş bir RC (Rezistans-Kondansatör) veya LC (İndüktör-Kondansatör) devresi yüksek frekanslı kare dalga biçimindeki modülatör sinyalini süzer ve sinyali analog forma geri döndürür.

PWM'nun çok değişik uygulama alanları mevcuttur. Bu kitapta PWM ile DC Motor kontrolü üzerinde duracağız.

DC Motor (Fan) Kontrolü

Birçok kişi DC Motor veya Fan kontrolünü örneksel olarak yapmaktadır; bu yöntemde motoru yavaşlatmak için motora uygulanan voltajı veya motordan geçen akımı azaltan reosta veya potansiyometre kullanılmaktadır. Ancak yük (Motor veya Fan) güçlendikçe kontrol eden reostanın da hacimli ve güçlü olması, dolayısıyla da fiziksel olarak büyük olması gerekir.

Örnek: Bilgisayarda kullanılan 80 mm fan, 12 Volt'da 0.8 Amper çekmektedir.

Fan'ın rezistif bir yük olduğunu kabul edersek, 6 Volt'ta, en fazla güç reostadan yayılacaktır.

Fan'ın çekeceği akım 6 / 12 * 0.8 = 0.4A

Diğer 6 Volt'u taşıyan reostadan, 0.4A geçtiğine göre:

- Güç = I*V
- Güç = 0.4 * 6 = 2.4 watt'tır.

Bu hesaba göre en az 10 watt'lık bir reosta kullanılmalıdır, zira reostaların güç katsayısı direnç sargısının tümü için geçerlidir. 10W'lık bir reosta da ya tel sargılıdır veya seramik dirençlidir, bu nedenle de hem fiziksel olarak büyük ve hem de pahalıdır.

Bu çarpıcı örnekle de gösterdiğimiz gibi PWM, analog kontrole göre çok fazla üstünlükleri olan bir denetim metodudur.

PWM metodunda fan 12 Volt genlikli darbelerle sürülecek; darbelerin genişliği Fan'ın dönüş hızını belirleyecektir. Geniş darbeler hızı arttıracak, dar darbeler ise hızı azaltacaktır.

Donanımsal Denetleyiciler

Birçok Mikrodenetleyici entegresinin yonga üzerinde konuşlandırılmış PWM devreleri vardır ki bu da pratik uygulamaları çok kolaylaştırmaktadır. Örneğin, Microchip'in PIC18F45K22 entegresi iki PWM, 3 adet de güçlendirilmiş PWM çıkış sağlamaktadır. Her çıkışın on-zamanı ve periyodu programlanabilmektedir. Duty Cycle, on-zamanın periyoda oranıdır; modülasyon frekansı ise periyodun tersidir. Mikrodenetleyici teknik veri dokümanları PWM işlemini başlatmak için yazılım kodunun aşağıda gösterilen işlevleri yapması gerektiğini söylemektedir:

- Modülasyon kare dalgasını üretecek yonga üstündeki zamanlayıcı/sayıcı devresinin on-zamanını ayarla,
- PWM kontrol kayıtçısındaki on-zamanını ayarla,
- Genel amaçlı kapı pinlerinden biri olan PWM çıkışının yönünü ayarla
- Zamanlayıcıyı başlat,
- PWM denetleyiciyi aktif yap.

Genellikle spesifik PWM denetleyicilerinin programlama detayları değişse de, temel felsefe hep aynıdır.

MikroC derleyicisinin PWM kütüphanesi PWM metodunun uygulanmasını kolaylaştırmaktadır. PWM kütüphanesi PWM uygulamaları için bize 4 adet fonksiyon sunar:

- PWM1_Init
- PWM1_Set_Duty
- PWM1_Start
- PWM1_Stop

PWM1_Init: PWM modülü 0 Duty Cycle ile başlatır. PWM1_Init(frekans) şeklinde girilen frekans değeri PWM dalgamızın frekansını belirler.

```
PWM1_Init(5000); // 5KHz PWM oluşturulur.
```

PWM1_Set_Duty: PWM dalgasının Duty Cycle'ını ayarlar.

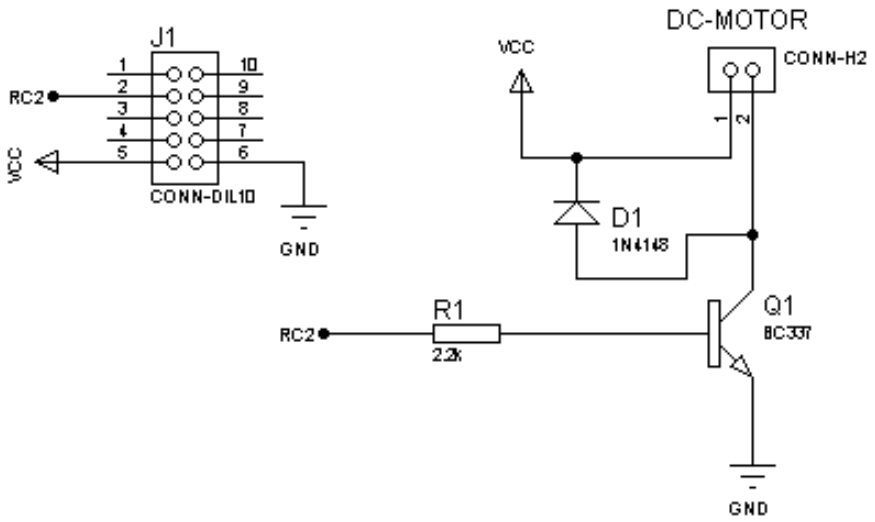
PWM1_Set_Duty((yüzde*255)/100) şeklinde girilen yüzde değeri Duty Cycle değerine denktir. Bu fonksiyonun alabileceği en büyük değer 255 tir.

```
PWM1_Set_Duty(192); // %75 Duty Cycle ile PWM üretir.
```

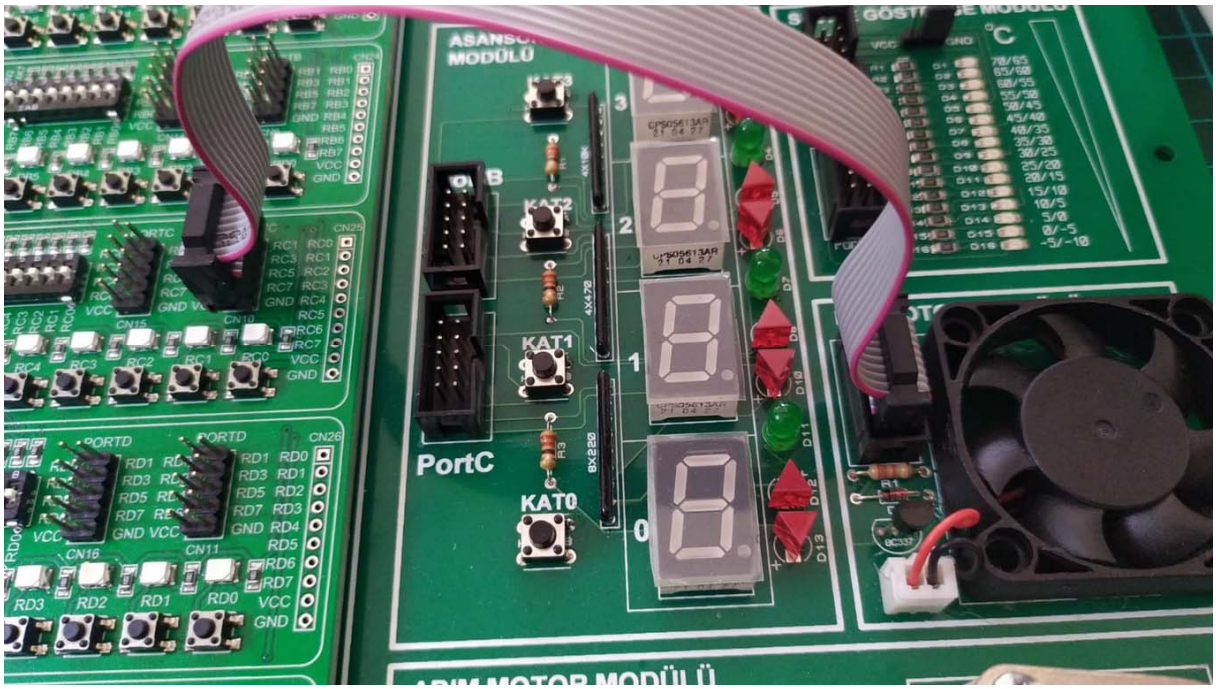
PWM1_Start: PWM dalgasını başlatır.

PWM1_Stop: PWM dalgasını durdurur.

PIC18F45K22 entegresi gibi birden fazla PWM kanalı olan entegreler için, kullanılmak istenen kanal, fonksiyonlarda PWM1, PWM2, PWM3 şeklinde belirtilmelidir. PWM kütüphanesi kullanmadan da PWM dalganızı üretebilirsiniz. Örnek MikroC kodu aşağıdadır.



Şekil 12.3 BETI MCU Seti üerindeki DC Motor devre şeması



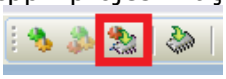
EasyPIC v7 üzerindeki PORTC bölümüyle ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki DC Motor Modülü arasındaki bağlantı yukarıdaki fotoğraftaki gibi olmalıdır.

ÖrnekProgram

```
//************************
// Proje ismi        : DARBE GENİŞLİK MODÜLASYONU (PWM) UYGULAMASI
// Amacı             : Çıkış olarak ayarlanan C portundan
//                   verilen pwm dalgasının Duty Cycleoranı
//                   sürekli arttırarak, değişimi DC Motor'da gözlemlemek.
//Test konfigürasyonu
//MCU               : 18F45K22
//Osilatör          : 32Mhz
//j12 jumperları I/O konumunda olmalıdır.
// j17          => VCC
// SW3.3     =>ON, Diğerleri OFF
//
//************************

void main()
{
  ANSELC =  0;              // PortC dijital olarak ayarlandı
  PWM1_Init(2000);          // PWM1 Frekansı 5KHz olarak ayarlandı.
```

```
  PWM1_Set_Duty(0);     // PWM1 Duty Cycle Oranı %0 olarak ayarlandı.
  PWM1_Start();
  // 2 saniyede bir Duty Cycle azaltılarak motor hızı da azaltılıyor.
  while(1)
  {
    PWM1_Set_Duty(255);
    delay_ms(2000);
    PWM1_Set_Duty(200);
    delay_ms(2000);
    PWM1_Set_Duty(150);
    delay_ms(2000);
    PWM1_Set_Duty(100);
    delay_ms(2000);
    PWM1_Set_Duty(50);
    delay_ms(2000);
    PWM1_Set_Duty(0);
    delay_ms(2000);
  }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 12-DARBE GENİŞLİK MODÜLASYONU (PWM) UYGULAMASI" içerisindeki "Deney12.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki DC MOTOR MODÜLÜ bölümündeki motoru gözlemleyiniz.

- **PWM sinyali kullanarak bir hoparlörden nasıl ses sinyali alınabileceğini araştırınız.**

DENEY 13. GERÇEK ZAMAN SAATİ (RTC)

AMAÇ

1) Gerçek Zaman Saati (RTC) kavram ve gereksinimini öğretmek,
2) Uygulamalarda Gerçek Zaman Saati'nin (RTC) nasıl set edildiğini ve bu bilginin nasıl görüntüle bileneceğini göstermek.

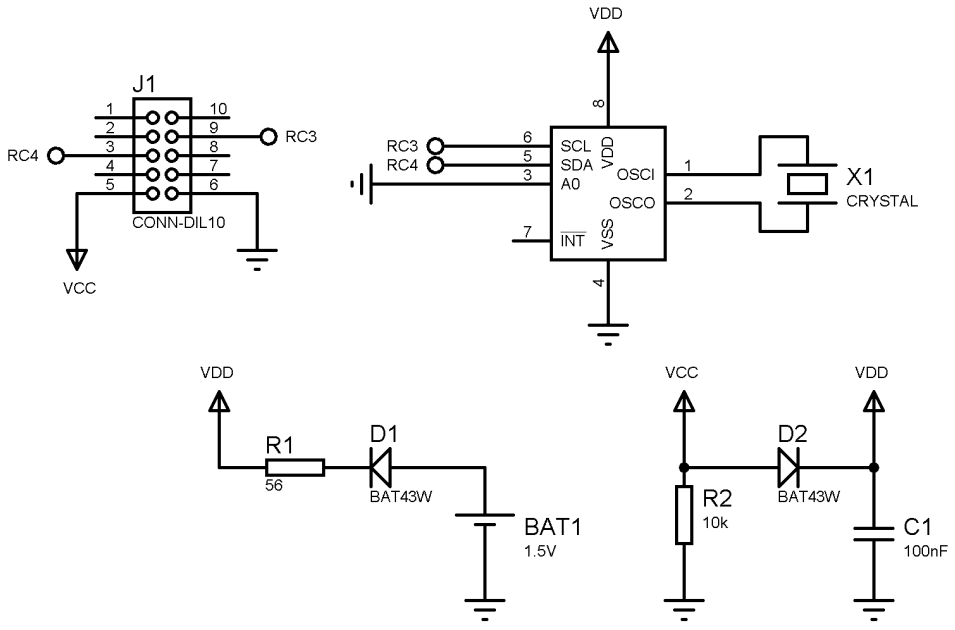
GEREKLİ MALZEMELER

1) EasyPIC v7 Kartı,
2) Beti Mikrodenetleyici Uygulama ve Geliştirme Seti
3) LCD Modül.

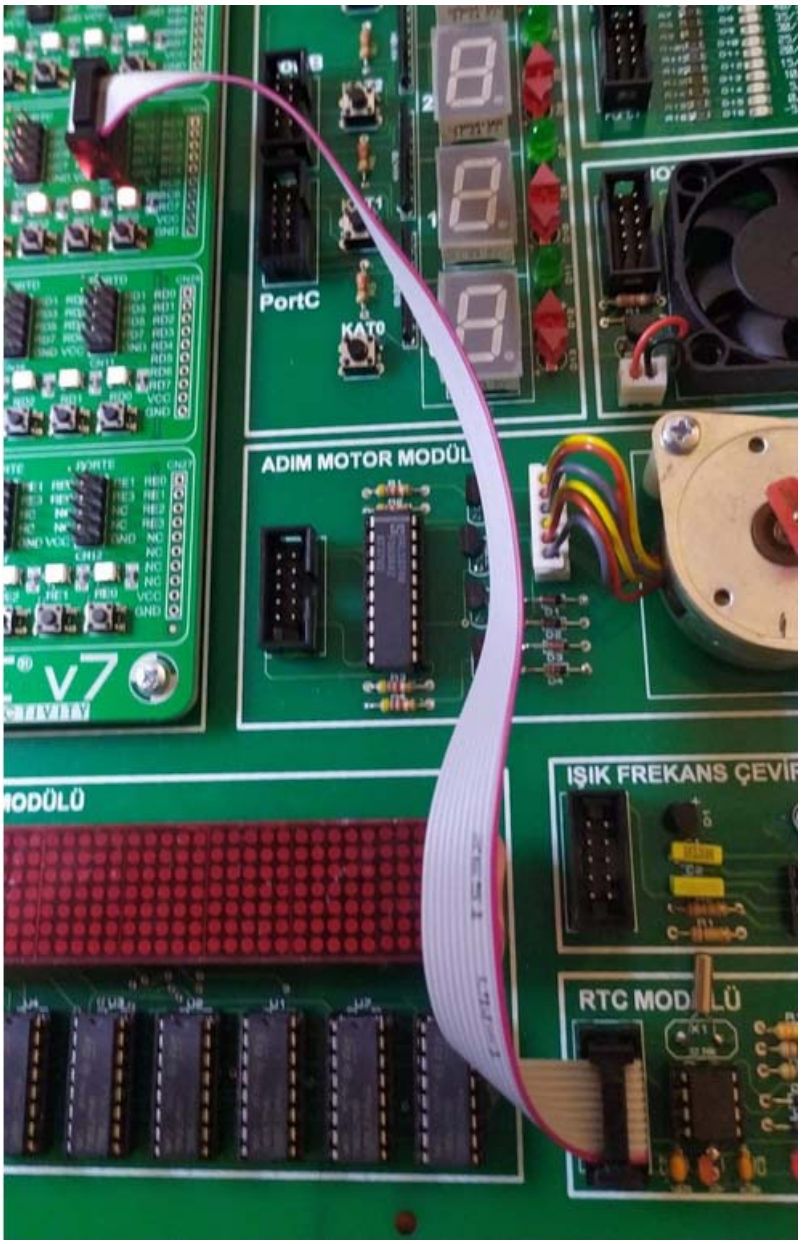
Giriş

Birçok projede gerçek zaman parametresini bilmek çok fazla önem taşır. Örneğin sıcaklık, nem ve atmosfer basıncının günde iki kere ölçüldüğü meteorolojik bir gözlemi ele alalım; bu ölçümlerin ölçümün yapıldığı tarih ve saat bilgisini gerçek saate uygun ve doğru olarak verilmesi gerekir, aksi halde ölçümler bilimsel bir özellik taşımaz. Gerçek Zaman Saati kullanmadığımız sürece, Mikrodenetleyici' nin dahili saat devresi ile ürettiği saniye, dakika, saat, gün, ay, yıl birimlerini kullanmak durumunda kalırız. Bir mahzur da dahili saat devresinin hatasının bir yıl boyunca ulaşacağı birikmiş hata miktarıdır. Geliştirilen özel amaçlı entegre devrelerin bile; örneğin bu uygulamada kullandığımız PhilipsinPCF8583 entegresinin bile gecikme hataları olabilir.

Bu nedenlerle en uygun uygulama özel entegreyi kendi güç kaynağı ile birlikte kullanmak olur ki bu durumda ölçülen zaman gerçek zamana eşit olur. PCF8583 entegresinin zaman doğruluğu (Accuracy) 1 saniye/gün'dür. Dolayısıyla bir yıllık sapma 6 dakikayı geçmez ki, bu da birçok gerçek zaman uygulamasının tolere ettiği bir hata büyüklüğüdür.



Şekil 13.1 Beti MCU Seti üzerindeki kullanılan RTC devre şeması



EasyPIC v7 üzerindeki PORTC bölümüyle ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki RTC MODÜLÜ arasındaki bağlantı yukarıdaki fotoğraftaki gibi olmalıdır.

Örnek Program

```
//************************
//Proje ismi        : GERÇEK ZAMAN SAATİ (RTC)
//Amacı             : PCF8583 RTC entegresine saymaya başlayacağı
//                    saat ve tarih bilgisini I2C seri haberleşme
//                    protokolü ile gönderdikten sonra saat ve
//                    tarihi LCD ekranda gözlemlemek.
//Test konfigürasyonu       :
```

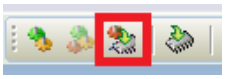
```
//-----------------------------
//MCU           : 18F45K22
//Osilatör      : 8Mhz
//
// SW4.6     => ON
//
// PortC      =>Pull-Up
// j12 jumperları I/O konumunda olmalıdır.
//***********************

char salise, saniye, dakika, saat, gun, ay;  // tarih ve zaman değişkenleri
unsigned int sene;
unsigned int modul_adres, sene_simdiki=2023; // RTC modülnün bağlı olduğu adres ve yıl bilgisi

char deneme_sayar=0;

// Lcdmoduleconnections
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// EndLcdmoduleconnections


void i2c_yaz(char veri,int adres)   //i2c hattı kullanılarak ilgili datayı ilgili adrese yazar.
{
  I2C1_Start();                 //  I2C1 başlangıç sinyali gönderilir.
  I2C1_Wr(modul_adres);     //  ( entegre adresi + 'W')
  I2C1_Wr(adres);             // EEPROM'un işlem yapılacak adresi
  I2C1_Wr(veri);              // adrese yazılacak değer
  I2C1_Stop();                // I2C1 durdurma sinyali gönderilir.


Delay_ms(100);   //100ms bekleme.
}
   //-----------------
void Baslangic_yapilandirmasi() {

sene_simdiki  = 2023;
modul_adres   = 0xA0;
```

```
I2C1_Init(50000);                       //  Soft I2C bağlantısı

Lcd_Init();                             // LCD ilklendirme fonksiyonu
Lcd_Cmd(_LCD_CLEAR);                    // LCD ekranını temizler.
Lcd_Cmd(_LCD_CURSOR_OFF);               // Cursor ekranda gösteilmez.

LCD_Out(1,1,"Tarih:");                  // LCD hazırlanır.
Lcd_Chr(1,9,'/');
Lcd_Chr(1,12,'/');
LCD_Out(2,1,"Zaman:");
Lcd_Chr(2,9,':');
Lcd_Chr(2,12,':');
}

//----------------- rtc_yapilandirma  bu yapilandirmalar için pcf8583 datasheet'iniincleyeniz.
void rtc_yapilandirma() {               // rtc'nin başlangıç zamanını ayarlar.

char yazilacak_data, yazilacak_adres;

yazilacak_data=0x80;                    //saymayı durdur.
yazilacak_adres=0x00;                   //control and status register.
 i2c_yaz(yazilacak_data,yazilacak_adres);

 //salise
yazilacak_data=0x00;   //
yazilacak_adres=0x01;  //
 i2c_yaz(yazilacak_data,yazilacak_adres);

 //saniye
yazilacak_data=0x50;   //
yazilacak_adres=0x02;  //
 i2c_yaz(yazilacak_data,yazilacak_adres);

 //dakika
yazilacak_data=0x16;    //
yazilacak_adres=0x03;   //
 i2c_yaz(yazilacak_data,yazilacak_adres);

 //saat
yazilacak_data=0x15;    //
yazilacak_adres=0x04;   //

 i2c_yaz(yazilacak_data,yazilacak_adres);

 //yıl ve gunyapilandirmasi
yazilacak_data=0x15;   //
yazilacak_adres=0x05;  //
 i2c_yaz(yazilacak_data,yazilacak_adres);

 //ay ve gunyapilandirmasi
yazilacak_data=0xb2;   //
```

```
yazilacak_adres=0x06;//
  i2c_yaz(yazilacak_data,yazilacak_adres);
  //yapılandırma tamamlandı!!!


  //yapılandırma tamamlnadıktan sonra sayma tekrar başlatılacak.
yazilacak_data=0x00;   //saymayı tekrar başlatır.
yazilacak_adres=0x00;//controlandstatusregister.
  i2c_yaz(yazilacak_data,yazilacak_adres);

}
//---- PCF8583 RTC 'den tarih ve zaman bilgisi okunur.
void RTC_oku_PCF8583() {
char okunan_data;

  I2C1_Start();                  //  I2C1 başlangıç sinyali gönderilir.
  I2C1_Wr(modul_adres);          //  ( entegre adresi + 'W')   yazma yapılacak.
  I2C1_Wr(2);                    // EEPROM'un işlem yapılacak adresi (1.byte'tan başlanıyor.)
  I2C1_Stop();                   // I2C1 durdurma sinyali gönderilir.

  I2C1_Start();                   // I2C1 başlatma sinyali gönderilir.
  I2C1_Wr(modul_adres+1);        //  ( entegre adresi + 'R')   okuma yapılacak.

okunan_data = I2C1_Rd(1);                    // 2. byte okunur. (saniye)
  saniye = ((okunan_data& 0xF0) >> 4)*10 + (okunan_data& 0x0F);       // saniye çevirimi

okunan_data = I2C1_Rd(1);                    // 3. byte okunur. (dakika)
  dakika =  ((okunan_data& 0xF0) >> 4)*10 + (okunan_data& 0x0F);     // dakika çevirimi

okunan_data = I2C1_Rd(1);                    // 4. byte okunur. (saat)
  saat =   ((okunan_data& 0xF0) >> 4)*10 + (okunan_data& 0x0F);       // saat çevirimi

okunan_data = I2C1_Rd(1);                   // 5. byte okunur. (sene ve gun(rakamla))
  sene = sene_simdiki + ((okunan_data& 0xC0) >> 6);                   // sene çevirimi
gun = ((okunan_data& 0x30) >> 4)*10 + (okunan_data& 0x0F);            // gün çevirimi

okunan_data = I2C1_Rd(0);                    // 6. byte okunur. (ay ve gun(adıyla))
  ay =   ((okunan_data& 0x10) >> 4)*10 + (okunan_data& 0x0F);         // ay çevirimi

  I2C1_Stop();                  // I2C1 durdurma sinyali gönderilir.
}

//-------------------- LCD üzerinde görüntüleme
void LCD_goruntuleme() {

Lcd_Chr(1, 7, (gun / 10)   + 48);
Lcd_Chr(1, 8, (gun % 10)   + 48);
Lcd_Chr(1,10, (ay / 10)  + 48);
Lcd_Chr(1,11, (ay % 10)  + 48);
Lcd_Chr(1,13, ((sene / 1000) % 10) + 48);
Lcd_Chr(1,14, ((sene / 100)  % 10) + 48);
```

```
Lcd_Chr(1,15, ((sene / 10)   % 10) + 48);
Lcd_Chr(1,16, (sene % 10)        + 48);

Lcd_Chr(2, 7, (saat / 10)   + 48);
Lcd_Chr(2, 8, (saat % 10)   + 48);
Lcd_Chr(2,10, (dakika / 10) + 48);
Lcd_Chr(2,11, (dakika % 10) + 48);
Lcd_Chr(2,13, (saniye / 10) + 48);
Lcd_Chr(2,14, (saniye % 10) + 48);
}

//-----------
void main() {

  ANSELB = 0;
  ANSELC = 0;


Baslangic_yapilandirmasi();
LCD_goruntuleme();
Delay_ms(2000);

//aşağıdaki comment'i kaldırırsanız rtc_yapilandirma fonksiyonu
//içindeki değerler rtc entegresinin ilgili kayıtçılarına yüklenir.
// böylece istediğiniz değere entegrenizi kurmuş olursunuz.

// comment kalkarsa kod her çalışmaya başladığında
// rtc_yapilandirma fonksiyonu işleyeceği için buradaki değerlerden başlar.
// bu yüzden entegerenizi bir kere dogru degere kurduktan sonra bu satiri comment'leyiniz.
//-------------------------------------------------------------------------------
rtc_yapilandirma();   // rtc'nin başlangıç zamanını ayarlar.

while (1)                // sonsuz döngü

{
   RTC_oku_PCF8583();          //RTC'den veri okuma

LCD_goruntuleme();
Delay_ms(1000);                //1sn bekleme

if (deneme_sayar==0)
deneme_sayar=1;
else
deneme_sayar=0;

   Lcd_Chr(2,16, deneme_sayar+48);
}
}
```

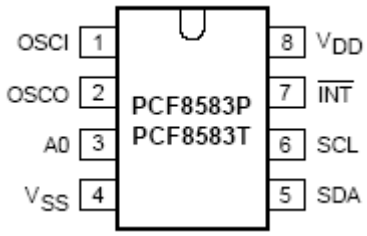
Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 13-GERÇEK ZAMAN SAATİ (RTC)" içerisindeki "Deney13.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. EasyPIC v7 kartı üzerindeki LCD'de Tarih ve saati gözlemleyiniz.

- **PCF8583'e kayıtlı tarih ve saati bulunduğunuz gün ve saate güncellemek için kodlarda gerekli değişikliği yapınız.**

PCF8583 Gerçek Zaman Saat Entegresi (RTC) ve I2C Seri Protokolü Hakkında Genel Bilgiler

PCF8583 8-bitX256 byte şeklinde düzenlenmiş bir 2048-bit statik CMOS RAM yapısını baz alan saat/takvim entegresidir. Entegre, adres ve veri iletişimini iki-hatlı ve iki-yönlü I2C bus üzerinden yapar. Yonga üzerindeki byte adres kayıtçısı her okuma/yazma işleminden sonra otomatik olarak arttırılır. Entegrenin (3) No'lu bacağı A0'dır ve bus'a iki cihazın birden ilave donanım gerektirmeden bağlanmasını sağlar.
Yonga üzerindeki 32.768 kHz osilatör devresi ve RAM'in ilk 8 byte'ı sayıcının saat/takvim işlemleri için kullanılır. Sonraki 8 byte ya alarm kayıtçısı olarak kullanılır veya alarm kullanılmayan uygulamalarda ise serbest RAM bölgesidir. Kalan 240 byte ise tamamen serbest RAM bölgesidir.



SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

Şekil 13.2 PCF8583 Gerçek Zaman Saati (RTC) Entegresi

PCF8583 RTC entegresi saat veya hadise sayıcısı olmak üzere iki modda çalışır, entegrenin hangi modda ve hangi parametrelere bağlı olarak çalışacağını 00 adresindeki okunabilen/yazılabilen Durum/Kontrol kayıtçısı belirler.

MSB							LSB
7	6	5	4	3	2	1	0
Sayımdurdurmabayrağı 0: say 1:dur	Son sayımbayrağı 0: say 1:son sayımı tut	Fonksiyonmodu 00:32,768 kHz saat 01:50Hz saat 10:sayıcı 11:test		Maske bayrağı 0:05 ve 06 adreslerimaskesizokunur 1:Tarih veaydirekokunur	Alarm 0:kapalı 1:açık	Alarm Bayrağı	Sayıcıbayrağı

Tablo13.3 PCF8583 RTC EntegresiKontrol/DurumKayıtçısı

Saat sayımmodu		Adres	Hadisesayımmodu	
Kontrol/Durum		00	Kontrol/Durum	
1/10saniye	1/100 saniye	01	D1	D0
10 saniye	1 saniye	02	D3	D2
10 dakika	1 dakika	03	D5	D4
10 saat	1 saat	04	Kullanılmıyor	
10 gün	1 gün	05	Kullanılmıyor	
10 ay	1 ay	06	Kullanılmıyor	
Zamanlayıcı 10 gün	Zamanlayıcı 1 gün	07	Zamanlayıcı T1	Zamanlayıcı T0
Alarm kontrol		08	Alarm kontrol	
Alarm 1/10 saniye		09	D1	D0
Alarm saniye		0A	D3	D2
Alarm dakika		0B	D5	D4
Alarm saat		0C	Kullanılmıyor	
Alarm tarih		0D	Kullanılmıyor	
Alarm ay		0E	Kullanılmıyor	
Alarm Zamanlayıcısı		0F	Alarm Zamanlayıcısı	
Serbest RAM			Serbest RAM	

Tablo13.4 PCF8583 RTC EntegresiKayıtçıFonksiyonları

MSB							LSB
7	6	5	4	3	2	1	0
Format 0: 24 saatformatı 1:12 saatformatı	AM/PM bayrağı 0: AM 1:PM	Saat OnlarHanesi (0-2 İkilik)		Saat BirlerHanesi (BCD Format)			

Tablo13.5 PCF8583 RTC Entegresi Saat Sayıcısı (Hafıza Adresi:04H)

MSB							LSB
7	6	5	4	3	2	1	0
Yıl (0-3 İkilik)		Gün OnlarHanesi (0-3 İkilik)		Gün BirlerHanesi (BCD Format)			

Tablo13.6 PCF8583 RTC EntegresiYıl/TarihSayıcısı (Hafıza Adresi:05H)

MSB							LSB
7	6	5	4	3	2	1	0
HaftanınGünleri (0-6 İkilik)			AyOnlarHanesi (0-2 İkilik)	AyBirlerHanesi (BCD Format)			

Tablo13.7 PCF8583 RTC EntegresiHaftanınGünleri/AySayıcısı (Hafıza Adresi:06H)

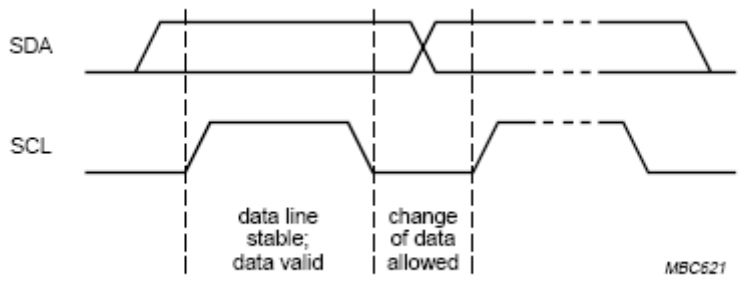
Kayıtçıların verdiğimiz açıklamalı işlevlerini gözönüne alarak Tablo13.1 de verilenRtc_write programının komutlarındaki değişkenlerle hangi tarih ve saat bilgisinin PCF8583'e yazdırıldığını irdeleyin.

I2C Seri İletişimHaberleşmeProtokolü

I2C protokolü; bu ara birime sahip modül veya entegre devreleri iki-tel üzerinden iki-yönlü haberleştirebilen bir arabirim standardıdır. Bu iki hattan biri verinin seri olarak gönderildiği (SDA) Verihattı, diğeri de Saat sinyalinin gönderildiği (SCL) Saat hattıdır. Her iki hat da pull-up dirençleri ile pozitif güç kaynağına bağlanmalıdır. Veri iletişimi sadece hat meşgul değilken (not busy) yapılabilir.

Bit transferi

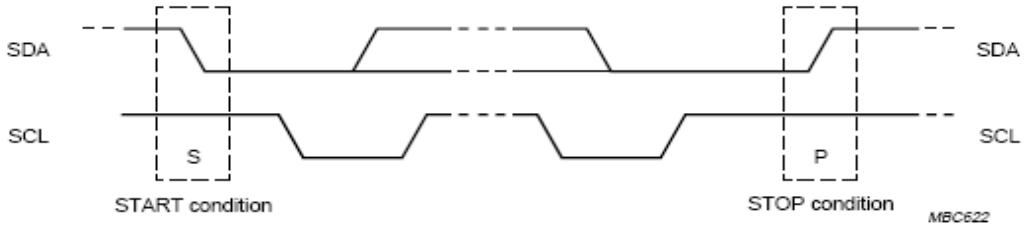
Her bir saat darbesi süresince 1 bit transfer edilir. Saat sinyali mantık (1) (HIGH) iken SDA hattındaki veri kararlı olmalıdır.



Şekil 13.3 I2C üzerinde bit transferi

Başla-Dur (Start-Stop) Şartları

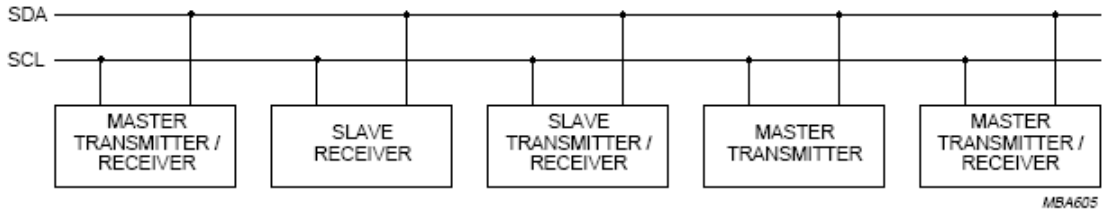
Hem verihattı SDA ve hem desaathattı SCL; hat meşgul değilken mantık(1) (HIGH) seviyesinde durur. SCL HIGH iken SDA'nın HIGH-LOW geçişi Başla (Start) (S) işlemini, SCL HIGH iken SDA'nın LOW-HIGH geçişi Dur (Stop) (P) işlemini belirtir. (Bakınız Şekil13.4)



Şekil 13.4 I2C üzerinde Start-Stop işlemleri

SistemKonfigürasyonu

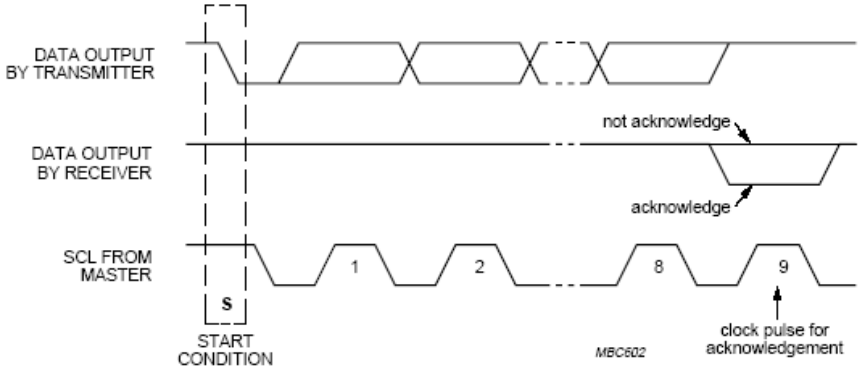
Bus üzerinde mesaj üreten Gönderici (Transmitter), mesajı alan da Alıcı (Receiver) durumundadır. Mesajı kontrol eden sistem (Şef) Master, Master'ın kontrol ettiği birimler de Hizmetli (Slave) konumundadır.



Şekil 13.5 I2C Bus Sistem Konfigürasyonu

Selamlaşma

Başla ve Dur zamanları arasında Göndericiden Alıcıya gönderilen veri byte'ı sayısı sınırsızdır. 8-bit'lik her veri byte'ı sonunda 1-bit Selamlaşma biti vardır.



Şekil 13.6 I2C Bus Üzerinde Selamlaşma

DENEY 14. OPTİK ASANSÖR SİMÜLASYON UYGULAMASI

AMAÇ

1) Optik asansör simülasyonunu öğrenmek

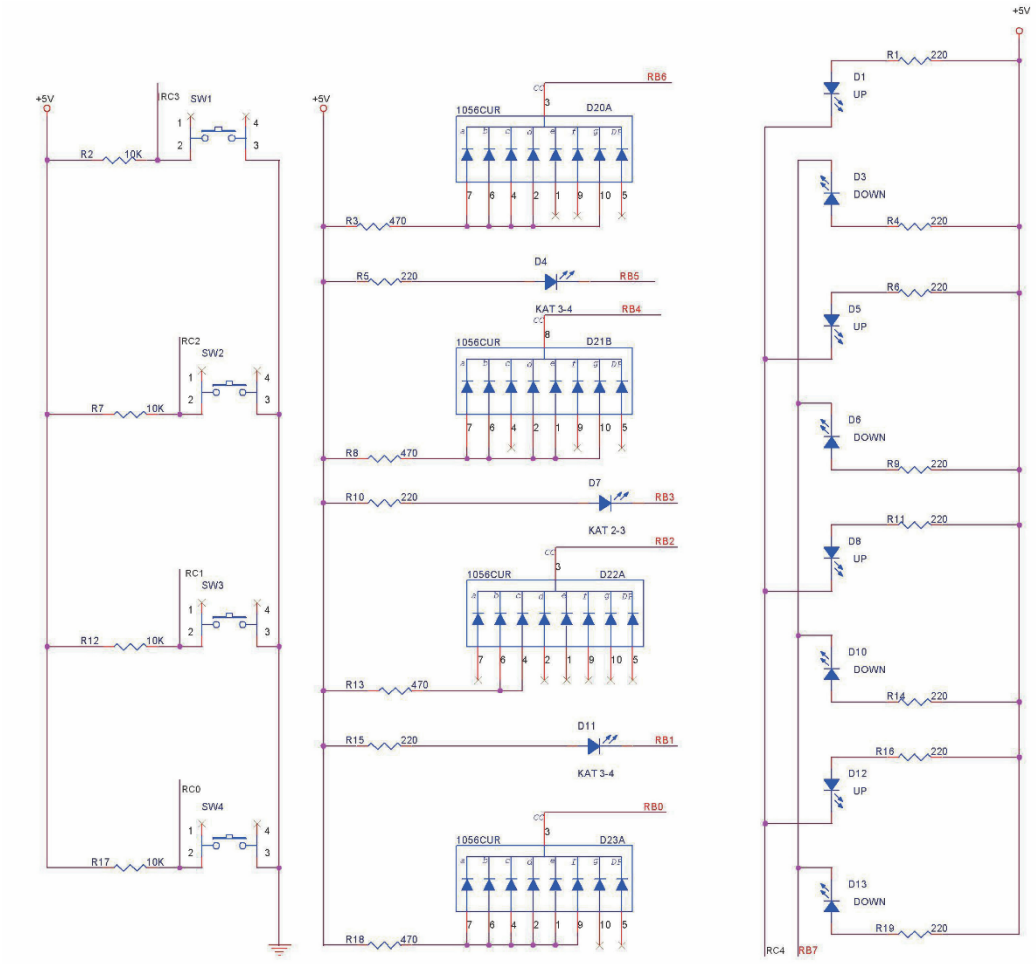
GEREKLİ MALZEME

1) EasyPIC v7 Kartı,
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti,

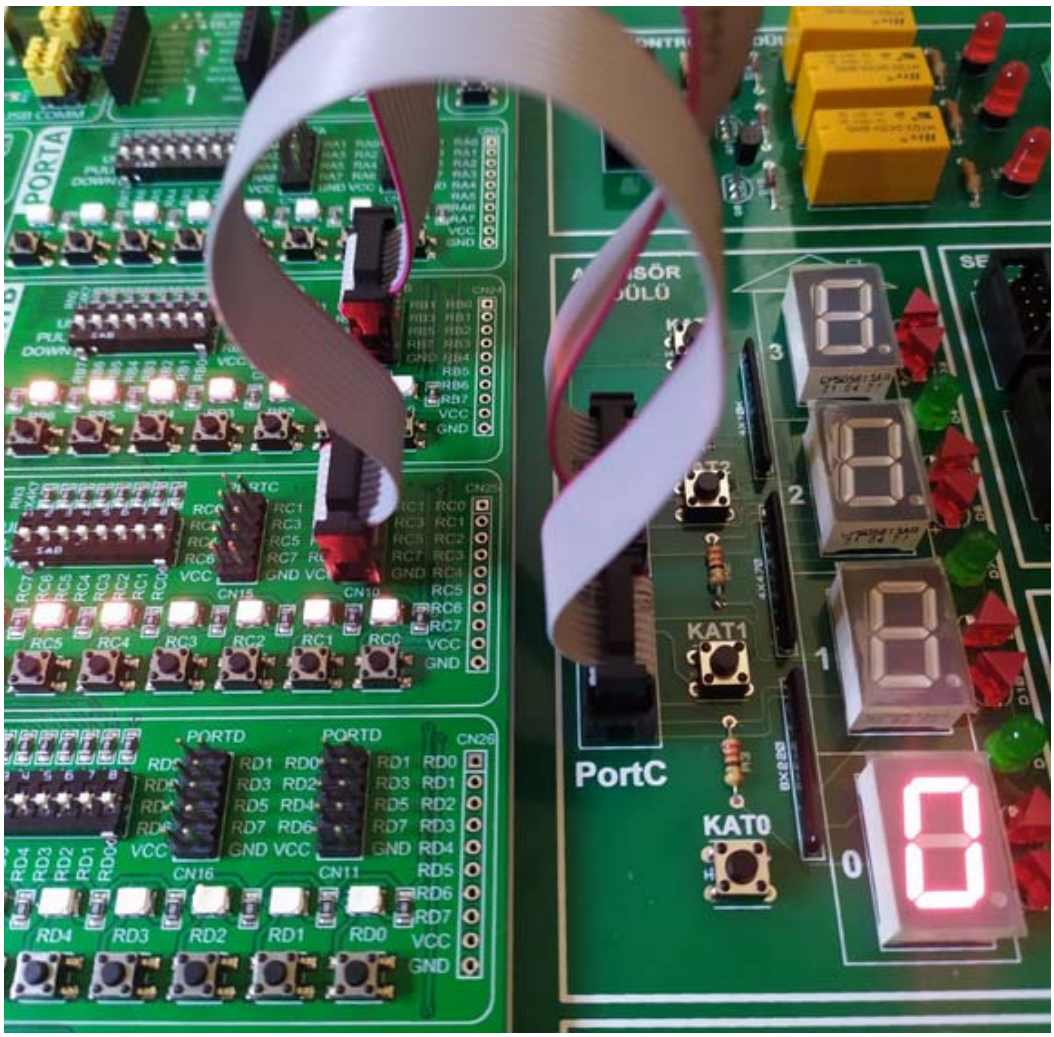
Giriş

Bu deneyde mekanik asansörün optik asansör olarak simüle edilmesi incelenecektir. Deneyde kat butonlarından birine bastığımızda o kata ait yedi parçalı gösterge ve asansörün yönüne göre de aşağı ya da yukarı ledleri yanacaktadır. Asansörün katlar arasındaki geçişini ise kırmızı ledler göstermektedir.

Port/B'nin RB0, RB2, RB4 ve RB6 pinleri katları gösteren yedi parçalı göstergelere RB1, RB3 ve RB5 pinleri kat arası ledlerine bağlanmıştır. PortC'nin RC0, RC1, RC2 ve RC3 pinleri kat butonlarına bağlıdır.



Şekil 14.1 Optik Asansör Uygulaması Devre Şeması



EasyPIC v7 üzerindeki PORTC ve PORTB bölümleri ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki ASANSÖR MODÜLÜ arasındaki bağlantı yukarıdaki fotoğraftaki gibi olmalıdır.

Örnek Uygulama

```
//************************
//Proje ismi      : Asansör Uygulaması
//Amacı           : Gerçek bir asansörün 7 parça
//                  gösterge ve LED'lerle elektronik
//                  bir simulasyonunu gözlemlemek.
//Test konfigürasyonu   :
//-------------------------
//MCU             : 18F45K22
//Osilatör        : 32Mhz
//
// j17      => VCC
// j12 jumperları I/O konumunda olmalıdır.
// SW3.2, 3.3 =>ON, DİĞERLERİ OFF
//
//************************
```

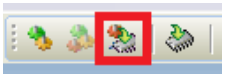
```
//rc0 kat0 btn
//rc1 kat1 btn
//rc2 kat2 btn
//rc3 kat3 btn



int bulunulan_kat=0;

// Asansötlerin bulunduğu katları gösteren fonksiyon
void kati_goster()
{
 if (bulunulan_kat==0) // KAT 0
 {
  LATB.b0=0; //kat 0
  LATB.b1=1; //kat 0-1
  LATB.b2=1; //kat 1
  LATB.b3=1; //kat 1-2
  LATB.b4=1;  //kat 2
  LATB.b5=1;  //kat 2-3
  LATB.b6=1;  //kat 3
 }

 if (bulunulan_kat==1)  // KAT 0-1 ARASI
 {
  LATB.b0=0; //kat 0
  LATB.b1=0; //kat 0-1
  LATB.b2=1; //kat 1
  LATB.b3=1; //kat 1-2
  LATB.b4=1;  //kat 2
  LATB.b5=1;  //kat 2-3
  LATB.b6=1;  //kat 3
 }

 if (bulunulan_kat==2)  //KAT 1
 {
  LATB.b0=1;   //kat 0
  LATB.b1=1;   //kat 0-1
  LATB.b2=0;   //kat 1
  LATB.b3=1;   //kat 1-2
  LATB.b4=1;   //kat 2
  LATB.b5=1;   //kat 2-3
  portb.b6=1;  //kat 3
 }

 if (bulunulan_kat==3) //KAT 1-2 ARASI
 {
  LATB.b0=1; //kat 0
  LATB.b1=1; //kat 0-1
  LATB.b2=0; //kat 1
```

```
  LATB.b3=0; //kat 1-2
  LATB.b4=1;  //kat 2
  LATB.b5=1;  //kat 2-3
  LATB.b6=1;  //kat 3
 }

 if (bulunulan_kat==4) //KAT 2
 {
  LATB.b0=1; //kat 0
  LATB.b1=1; //kat 0-1
  LATB.b2=1; //kat 1
  LATB.b3=1; //kat 1-2
  LATB.b4=0;  //kat 2
  LATB.b5=1;  //kat 2-3
  LATB.b6=1;  //kat 3

 }


 if (bulunulan_kat==5) //KAT 2-3 ARASI
 {
  LATB.b0=1; //kat 0
  LATB.b1=1; //kat 0-1
  LATB.b2=1; //kat 1
  LATB.b3=1; //kat 1-2
  LATB.b4=0;  //kat 2
  LATB.b5=0;  //kat 2-3
  LATB.b6=1;  //kat 3
 }

 if (bulunulan_kat==6) // KAT 3
 {
  LATB.b0=1; //kat 0
  LATB.b1=1; //kat 0-1
  LATB.b2=1; //kat 1
  LATB.b3=1; //kat 1-2
  LATB.b4=1;  //kat 2
  LATB.b5=1;  //kat 2-3
  LATB.b6=0;  //kat 3
 }
}


void kat_hareket (int gidilecek_kat)
{
 if(bulunulan_kat>gidilecek_kat )          //aşağı geliyor.
 {
  LATB.b7=0;                               //aşağı geliyor.
  while(bulunulan_kat!=gidilecek_kat)
  {
   kati_goster();
```

```
   delay_ms(1000);
   bulunulan_kat--;
  }
  kati_goster();
  delay_ms(1000);
  LATB.b7=1;
 }
 else
 {
  LATC.b4=0;                    //yukarı geliyor.
  while(bulunulan_kat!=gidilecek_kat)
  {
   kati_goster();
   delay_ms(1000);
   bulunulan_kat++;
  }
  kati_goster();
  delay_ms(1000);
  LATC.b4=1;
 }
}

void main()
{
 ANSELB = 0;    // PORTB dijital olarak ayarlandı.
 ANSELC = 0;    // PORTC dijital olarak ayarlandı.

 trisc=0x0f;     // PORTC'nin LSB 4 biti giriş, diğerleri çıkış olarak ayarlandı.
 trisb=0x00;     // PORTB çıkış olarak ayarlandı.

 latb=0xff;
 latc=0xff;
 latb=0xff;
 latc=0xff;

 kati_goster();

 while(1)
 {
  if (portc.b0==0)                    // Zemin kata baıldı
  {
   kat_hareket(0);
  }

  if (portc.b1==0)                    //1. Kata basıldı
  {
   kat_hareket(2);
  }

  if (portc.b2==0)                    //2. kata basıldı
  {
```

```
    kat_hareket(4);
   }

   if (portc.b3==0)                    //3. kata basıldı
   {
    kat_hareket(6);
   }
  }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 14-ASANSÖR DENEYİiçerisindeki "Deney14.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki Asansör Modülü üzerindeki kat butonlarına basarak 7 parçalı göstergeleri gözlemleyiniz.

- **Asansör sistemlerinde asansörün kata geldiğini tespit eden sensörleri araştırınız.**

DENEY 15. SICAKLIK DEĞERİNİN BARGRAPH VE LCD MODÜLDE GÖSTERİLMESİ

AMAÇ

1) DS1820'den tektel (onewire) üzerinden sıcaklık değerinin okunması
2) Ölçülebilir herhangi bir değerin bargraph yardımıyla gösterilmesi

GEREKLİ MALZEME

1) EasyPIC v7 Mikrodenetleyici Eğitim Sistemi,
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti,
3) LCD modül

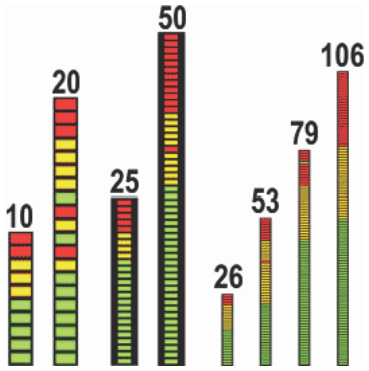
Giriş:

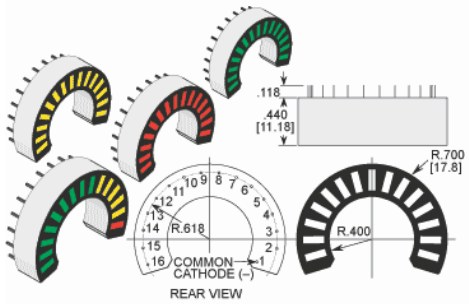
Bargraph (çubuk grafik) nedir?

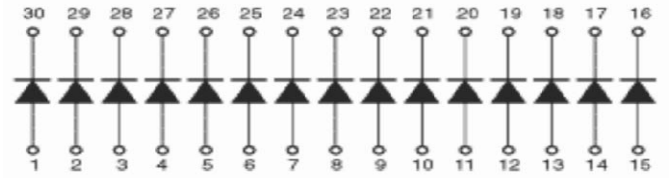
Bargraph ister yatay ister dikey olarak ölçülmüş olan değerleri göstermemize yarayan LED dizilerinden oluşmuş elektronik bileşenlerdir. Led bargraphlar birçok alanda kullanılmaktadırlar. Özellikle otomasyon, otomotive ve düşük maliyet gerektiren ölçüm cihazlarında özellikle mukayese maksatlı olarak görselliği sağlamak amacıyla kullanılırlar.

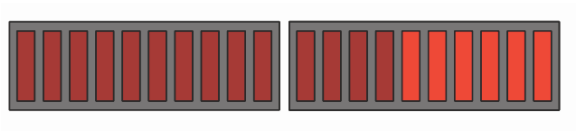
Bir LED bargraph kullanarak hız, basınç, sıcaklık, nem, voltaj değeri, akım değeri kısacası herhangi bir LCD ekranda gösterebildiğiniz tüm fiziksel ölçüm değerlerini LED bargraph'tan gözlemlemeniz mümkündür. Çoğu kez LED bargraph kullanımı dijital göstergelerden daha anlamlı olmaktadır. Mesela bir arabanın hız göstergesinde dijital hız göstergesi olmasının yanında LED bargraph'ın bulunması sürücü açısından çok anlamlıdır.

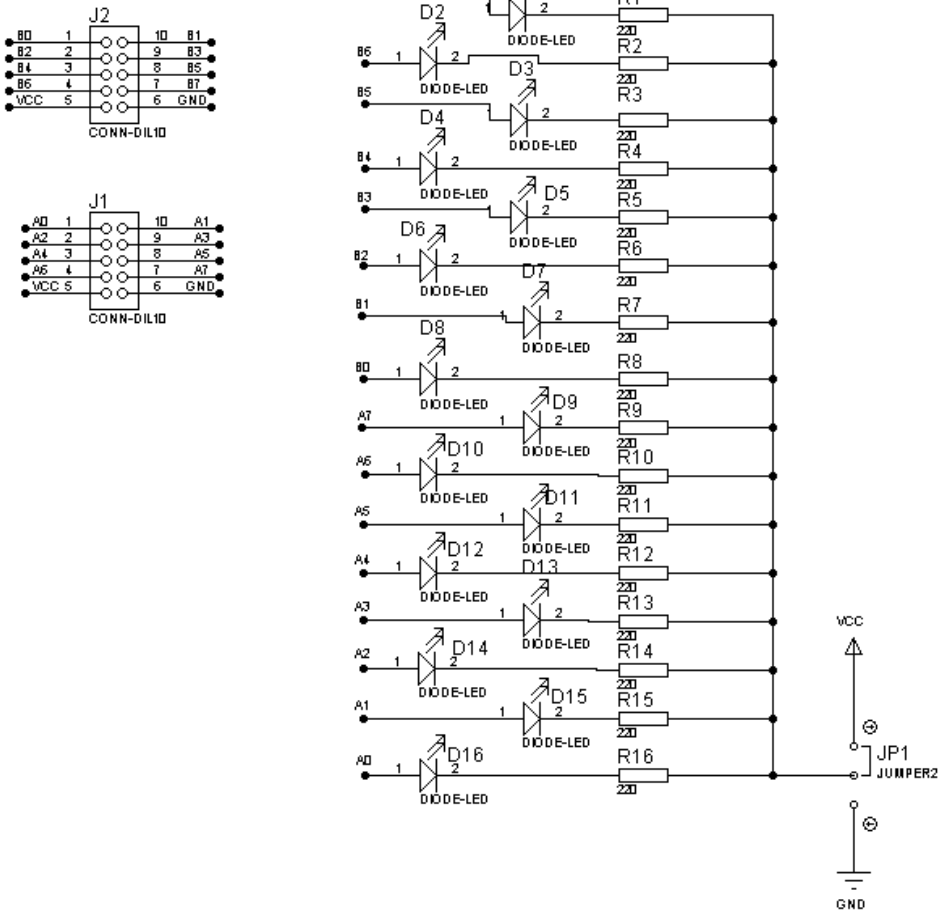
Genel yapıları LED dizileri şeklinde olan bargraphlar farklı çözünürlüklere (led sayısına) sahip çubuk, at nalı veya yarım at nalı şeklinde farklı biçimlere sahiptirler.







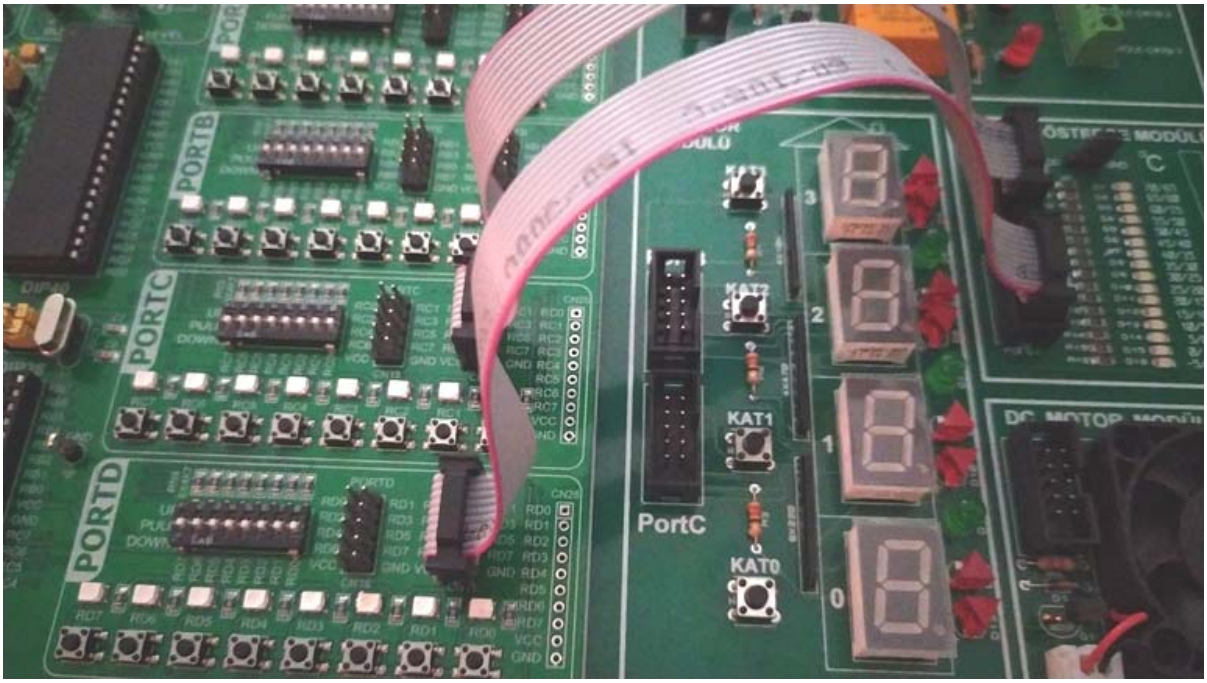




Şekil 15.1 Beti Mcu Setinde Kullanilan Seviye Göstergesinin Devre Şeması

Bu uygulamada DS18B20 ismindeki sensörden sıcaklık bilgisi okunacak ve hem EasyPIC v7 kartında bulunan LCD'de hem de Beti Mikrodenetleyici Uygulama ve Geliştirme Setinde bulunan LED Gösterge Modülünde gösterilecektir.

DS18B20 sıcaklık sensörü One Wire (Tek tel) iletişim protokolünde haberleşmektedir. Bu protokolde once mikrodenetleyiciden sensöre ayar ve okunmak istenilen adres bilgileri gönderilir daha sonar sensor aynı hat üzerinden mikrodenetleyiciye istenilen verileri gönderir. Yani iletişim tek hat üzerinden çift yönlü gerçekleşmektedir. One Wire protokolünde tek hat üzerine birden fazla one wire destekleyen çip bağlanabilmekte ve her biriyle farklı adresleri olduğu için birbirlerinden bağımsız bir şekilde haberleşilebilmektedir.



EasyPIC v7 üzerindeki PORTC ve PORTD bölümleri ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki LED GÖSTERGE MODÜLÜ arasındaki bağlantı yukarıdaki fotoğraftaki gibi olmalıdır.

Örnek program

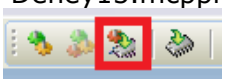
```
//************************
//Proje ismi      :DENEY15-SICAKLIK DEĞERİNİN BARGRAPH VE LCD MODÜLDE GÖSTERİLMESİ
//Amacı           :DS1820 dijital sıcaklık sensöründen
//                 alınan dijital sıcaklık verisine göre,
//                 Bargraph'da sıcaklık seviyesini,
//                 LCD ekranda sıcaklık değerini gözlemlemek.
//Test konfigürasyonu             :
//-------------------------
// MCU            : 18F45K22
// Osilatör       : 32Mhz
//
```

```
// SW3.3, 3.4     => ON
// SW4.6          => ON
// j12 jumperları I/O konumunda olmalıdır.
// DS18B20 jumper => RE2
//Gösterge Modlülü Jumper Vcc Konumunda Olmalı.
//***********************


// LCD mödül bağlantıları
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// LCD modül bağlantılarının sonu


char txt[6];
float sicaklik;
unsigned int onbinler,binler,yuzler,onlar,birler;

// Sıcaklığı LCD'de ve Bargraphta gösteren fonksiyon.
void sicakligiGoster()
{
 //Virgüllü değerden kurtulmak için sicaklik
 //değişkeni 1000 ile çarpılıyor.
 unsigned int sicaklikTamsayi=sicaklik*1000;

 //LCD'de gösterebilmek için basamak değerleri hesaplanıyor.
 onbinler=sicaklikTamsayi/10000;
 binler=(sicaklikTamsayi/1000)%10;
 yuzler=(sicaklikTamsayi/100)%10;
 onlar=(sicaklikTamsayi/10)%10;
 birler=sicaklikTamsayi%10;

 // Basamak değerleri 48 eklenerek karaktere dönüştürülüyor.
 // (0'ın ASCII karşılığı 48'dir.)
 txt[0]=onbinler+48;
 txt[1]=binler+48;
 txt[2]='.';
 txt[3]=yuzler+48;
 txt[4]=onlar+48;
 txt[5]=birler+48;
```

```
txt[6]=0;

// Sıcaklık LCD'de gösteriliyor.
lcd_out(2,5,txt);
lcd_chr_cp(223); //derece işareti
lcd_chr_cp('C');

// Bargraphta sıcaklığa göre yanacak ledler ayarlanıyor.
if (sicaklik >70 )
   {
    latc=0xff;
    latd=0xff;
   }
else if (sicaklik >65 )  //  65-70
   {
    latc=0xff;
    latd=0xff;
   }
else if (sicaklik >60 )   // 65 -60
   {latc=0x7f;
    latd=0xff;}
else if (sicaklik >55 )   // 60 -55
   {latc=0x3f;
    latd=0xff;}
else if (sicaklik >50 )   // 55 -50
   {latc=0x1f;
    latd=0xff;}
else if (sicaklik >45 )    // 50 -45
   {latc=0x0f;
    latd=0xff;}
else if (sicaklik >40 )   // 45 -40
   {latc=0x07;
    latd=0xff;}
else if (sicaklik >35 )   // 40 -35
   {latc=0x03;
    latd=0xff;}
else if (sicaklik >30 )    // 35 -30
   {latc=0x01;
    latd=0xff;}
else if (sicaklik >25 )     // 30 -25
   {latc=0x00;
    latd=0xff;}
else if (sicaklik >20 )      // 25 -20
   {latc=0x00;
    latd=0x7f;}
else if (sicaklik >15 )      // 20 -15
   {latc=0x00;
    latd=0x3f;}
else if (sicaklik >10 )
   {latc=0x00;
    latd=0x1f;}
```

```
 else if (sicaklik >5 )
    {latc=0x00;
     latd=0x0f;}
 else if (sicaklik >0 )
    {
     latc=0x00;
     latd=0x07;
    }

}

void main()
{
  int tempmsb, templsb;
  ANSELE = 0;                    // PORTE dijital olarak ayarlanıyor.
  TRISE.B2 = 1;                  // RE2 pini giriş olarak ayarlanıyor.(DS18B20)

  ANSELC = 0;                    // PORTC dijital olarak ayarlanıyor.
  ANSELD = 0;                    // PORTd dijital olarak ayarlanıyor.

  trisc=0x00;                    // PORTC çıkış olarak ayarlanıyor (BARGRAPH)
  trisd=0x00;                    // PORTD çıkış olarak ayarlanıyor (BARGRAPH)
  LATC=0X00;                     // BARGRAPH ledleri söndürülüyor.
  LATD=0X00;

  Lcd_Init();                              // LCD kuruluyor.
  Lcd_Cmd(_LCD_CLEAR);                     // LCD temizleniyor.
  Lcd_Cmd(_LCD_CURSOR_OFF);                // LCD imleci kapatılıyor.
  Lcd_Out(1, 4, " Sicaklik:   ");

  do
  {
  Ow_Reset(&PORTE, 2);                 // Onewire reset sinyali gönderiliyor.
  Ow_Write(&PORTE, 2, 0xCC);           // SKIP_ROM Komutu
  Ow_Write(&PORTE, 2, 0x44);           // CONVERT_T Komutu

  Delay_ms(750);   //Dönüşüm için bekleniyor.

  Ow_Reset(&PORTE, 2);
  Ow_Write(&PORTE, 2, 0xCC);           // SKIP_ROM Komutu
  Ow_Write(&PORTE, 2, 0xBE);           // READ_SCRATCHPAD Komutu

  templsb =  Ow_Read(&PORTE, 2);           // lsb'yi oku
  tempmsb =  Ow_Read(&PORTE, 2);           //msb'yi oku

  sicaklik=((tempmsb<<8)+templsb)*0.0625;     // sıcaklığı hesapla

  sicakligiGoster();

  }while (1);
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 15-SICAKLIK DEĞERİNİN BARGRAPH VE LCD MODÜLDE GÖSTERİLMESİ" içerisindeki "Deney15.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki Asansör Modülü üzerindeki DS18B20 sıcaklık sensörünü ısıtarak LCD'de ve LED GÖSTERGE MODÜLÜNDE'ki değişimleri gözlemleyiniz.

- **One Wire iletişim protokolünü kullanan güncel cihazları ve çipleri araştırınız.**

DENEY 16.KAYAN YAZI UYGULAMASI (5×7 DOT MATRİX)

AMAÇ

1) Mikrodenetleyicinin çıkışlarından aldığımız veriye göre 5x7 DotMatrix'lerde istediğimiz şekli veya yazıyı oluşturmak.

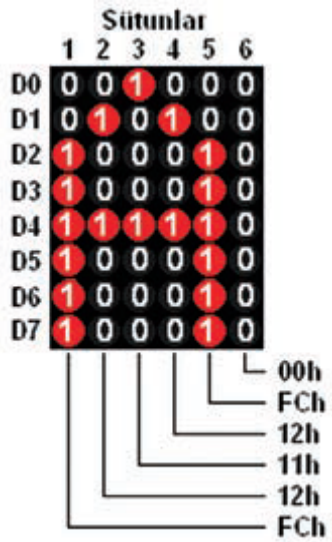
GEREKLİ MALZEME

1) EasyPIC7 Kartı,
2) Beti PIC Mikrodenetleyici Seti

GİRİŞ

Karakter oluşturma:

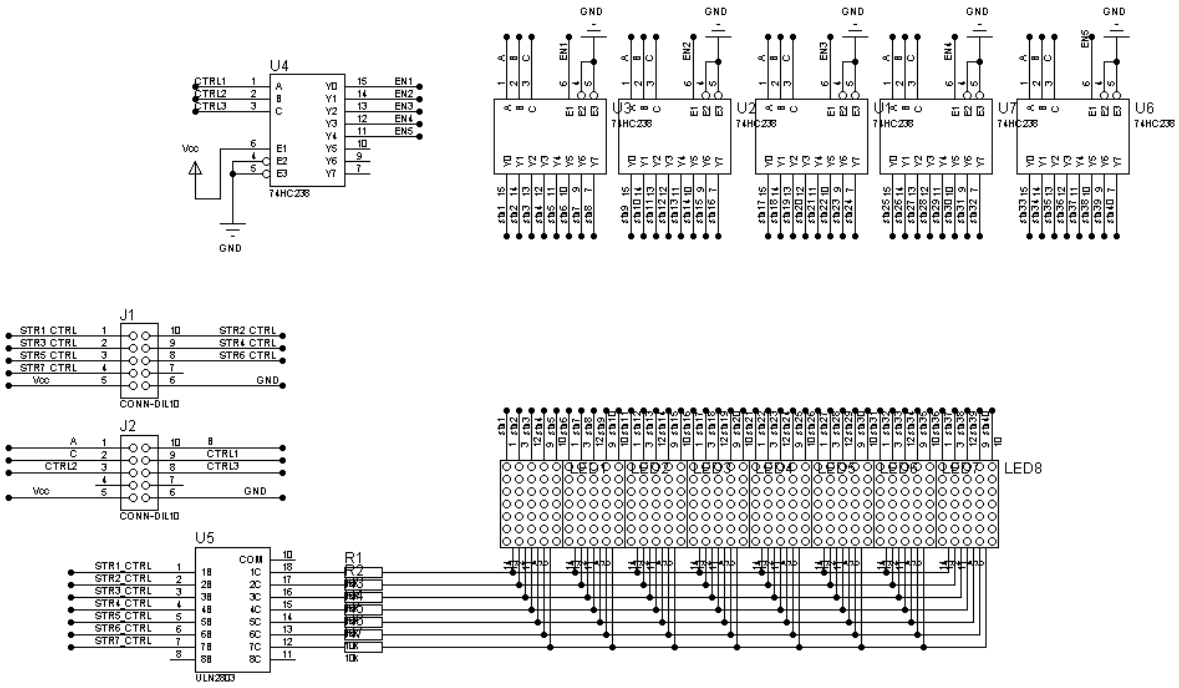
Matris display üzerinde harf, rakam ve diğer karakterleri görüntüleyebilmek için öncelikle bir karakter tablosu oluşturmak gerekir. Bu tabloda, karakterin kaç satırdan ve kaç sütundan oluşacağı, hangi LED'lerin ışık yayacağı belirlenir. Aşağıdaki figürde A harfi için karakter oluşturma mantığı görülüyor. Kırmızı renkli noktalara 1, diğerlerine 0 yazılarak karaktere ait satır verileri elde edilir. Örneğin, A harfi 6 sütun, 8 satırdan oluşur ve matris displayde bu karakteri görüntüleyebilmek için sırasıyla 0xFC, 0x12, 0x11, 0x12, 0xFC, 0x00 verilerinin satırlara iletilmesi gerekir. 16'lık tabanda (hexadesimal) yazılmış bu verileri matris displayin satırlarına iletmenin en uygun yolu bir mikro denetleyici kullanmaktır. Mikrodenetleyicinin çıkış portuna ait pinler, birer direnç üzerinden matris displayin satır uçlarına bağlanırsa, karakter tablosundaki hexadesimal değerlere göre LED'leri yakmak mümkün olur.



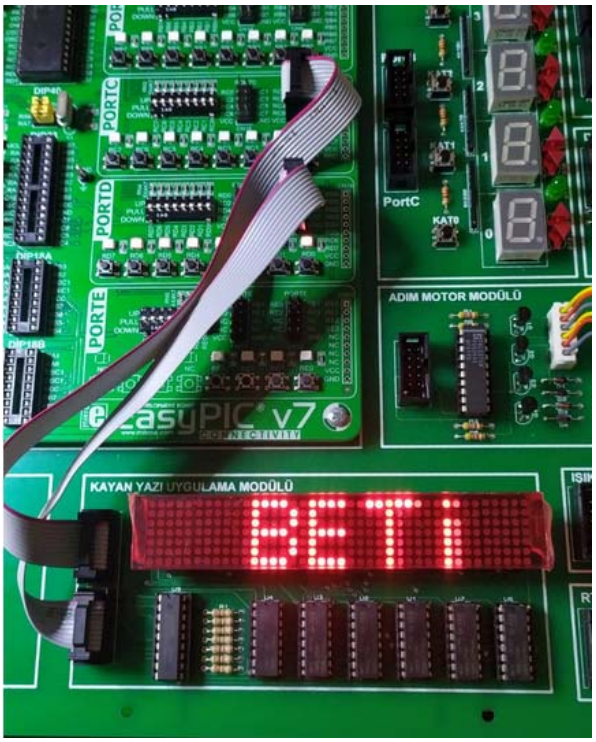
Karakteri kaydırma:

Karakterleri oluşturduktan sonra kaydırmak için oluşturduğumuz karakterin sütunlarını belirli zaman aralıklarıyla istediğimiz yöne kaydırmamız gerekmektedir. Bütün

karakterlerin bütün sütunlarına aynı işlemi uyguladığımız zaman sanki yazımız kayıyormuş gibi görünecektir.



Şekil 16.1 Beti MCU Uygulama Seti Üzerindeki Kayan Yazı Modülü Devre Şeması



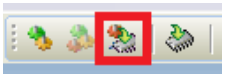
EasyPIC v7 üzerindeki PORTC ve PORTD bölümleri ile Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki KAYAN YAZI UYGULAMA MODÜLÜ arasındaki bağlantı yukarıdaki fotoğraftaki gibi olmalıdır.

Örnek program

```
//***********************
//Proje ismi      : DENEY 16-KAYAN YAZI UYGULAMASI (5×7 DOT MATRİX)
//Amacı           : Çıkış olarak ayarlanan PortB ve PortC den
//                  kayan yazı modülüne gönderilen veriye göre
//                  DotMatrix'de istediğimiz yazıyı kaydırmak.
//Test konfigürasyonu   :
//---------------------------
//MCU             : 18F45K22
//Osilatör        : 32Mhz
//j12 jumperları I/O konumunda olmalıdır.
//SW3.1, 3.2 => ON
//PortC     => Kayan yazı modülü üst soket
//PortD     => Kayan yazı modülü alt soket
//***********************

unsigned const char metin[]=
{
0xFF,0x49,0x49,0x49,0x36,0x00, // B
0xFF,0x49,0x49,0x49,0x00,0x00, // E
0x01,0x01,0x7F,0x01,0x01,0x00, // T
0x00,0x00,0x7d,0x00,0x00,0x00, // i
0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00
};

unsigned char gecici_dizi[40];
unsigned char i,j;
unsigned char toplam_sutun=40;
int kayma,deger;


void main()
{
  ANSELC = 0;                   // PORTC dijital olarak ayarlanıyor.
  ANSELD = 0;                   // PORTD dijital olarak ayarlanıyor.

  TRISC=0x00;                   // PORTC çıkış olarak ayarlanıyor.
  TRISD=0x00;                   // PORTD çıkış olarak ayarlanıyor.
  LATC=0x00;                    // PORTC çıkışları sıfırlanıyor.
  LATD=0x00;                    // PORTD çıkışları sıfırlanıyor

  //Port ayarlama işlemleri

  while(1)
  { //Ana döngü
    //Geçici diziyi sıfırla
    for(i=0;i<40;i++)
    {
```

```
    gecici_dizi[i]=0X00; //Dizi elemanları başlangıçta 0
   }

   //Kaydırma işlemleri
   for(kayma=-39;kayma<=toplam_sutun;kayma++)
   {
   //Metni 40 sütunluk parçalara böl
    for(i=0;i<=39;i++)
    {
     deger=i+kayma;
     if(deger<0)gecici_dizi[i]=0X00;                //metin girişi
     if(deger>=0&&deger<=toplam_sutun)
     gecici_dizi[i]=metin[deger];
     if(deger>toplam_sutun)gecici_dizi[i]=0X00; //metin çıkışı
    }
    //Tarama işlemleri
    for(j=0;j<20;j++)
    {                          //Aynı görüntüyü 40 kez tekrarla
     for(i=0;i<40;i++)
     {                         //Geçici diziyi görüntüle
      LATD=gecici_dizi[i];  // Veriyi PortB'ye gönder
      LATC=i;               // ilgili sütun'u seç
      Delay_Us(50);         // 200 mikrosaniye bekle
     }
    }
   }
  }   // işlemleri tekrarla
}     // Program sonu
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY16-KAYAN YAZI UYGULAMASI" içerisindeki "Deney16.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki KAYAN YAZI UYGULAMA MODÜLÜ'nde "BETİ" yazısının kaymasını gözlemleyiniz.

- **Örnek programı inceleyerek, seçeceğiniz kısa bir kelimeyi kaydırmak için gerekli değişikliği yaparak uygulayınız.**

DENEY 17. IŞIK FREKANS DÖNÜŞTÜRÜCÜ DENEYİ

AMAÇ

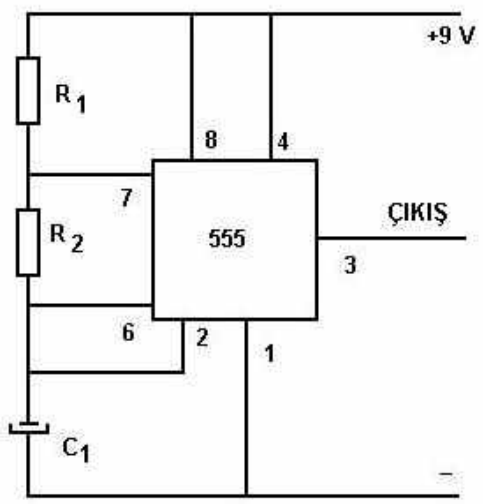
1) 555 timerentegresi ile kare dalga üretmek
2) Ürettiğimiz kare dalgayı infraredile kablosuz bir şekilde optik alıcıya göndermek.
3) Gönderilen kare dalganın frekansını ölçüp LCD modülde gözlemlemek.

GEREKLİ MALZEME

1) EasyPIC v7 Kartı.
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti.
3) LCD modül.

Giriş:

555 Entegre Devresi monostable multivibratör (asimetrik kare dalga osilatör), astable multivibratör (simetrik kare dalga osilatör) olarak kullanılabilmektedir. Aşağıda gösterilen devre şemasına göre, kapasitörün dolup boşalmasına bağlı olarak 3 numaralı pinden kare dalga çıkışı alınır.



Şekil 17.1 555 devre şeması

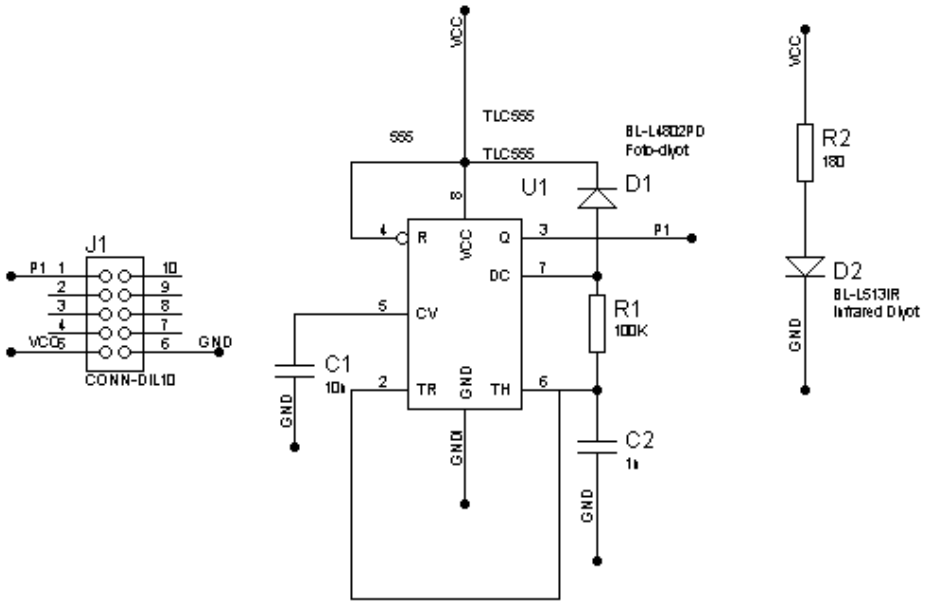
Bu devrede frekansı R1 R2 ve C1 belirler. 3 nolu pin ise frekans çıkış ucudur.

Devrenin Periyodu
T=0,693*(R1+2.R2).C1 dir.

Frekans ise "F=1/T" formülü kullanılarak bulunur.

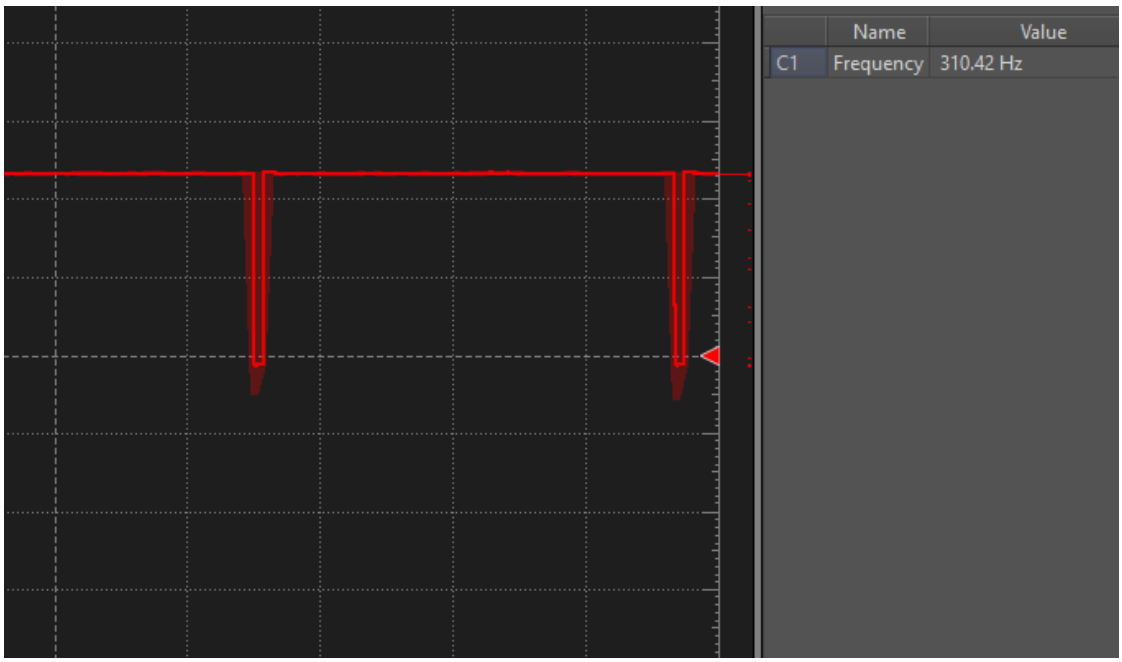
Birimler
T = Periyot (Saniye)
F = Frekans (1/Saniye) (yada saykıl/saniye)
R1= Direnç (OHM)
R2= Direnç (OHM)
C1=kondansatör (Farad)

Bu uygulamada kullanılan devrenin şeması Şekil 17.2'de görülebilir. Şemaya dikkat edilirse R1 direnci yerine bir fotodiyot konulmuştur.

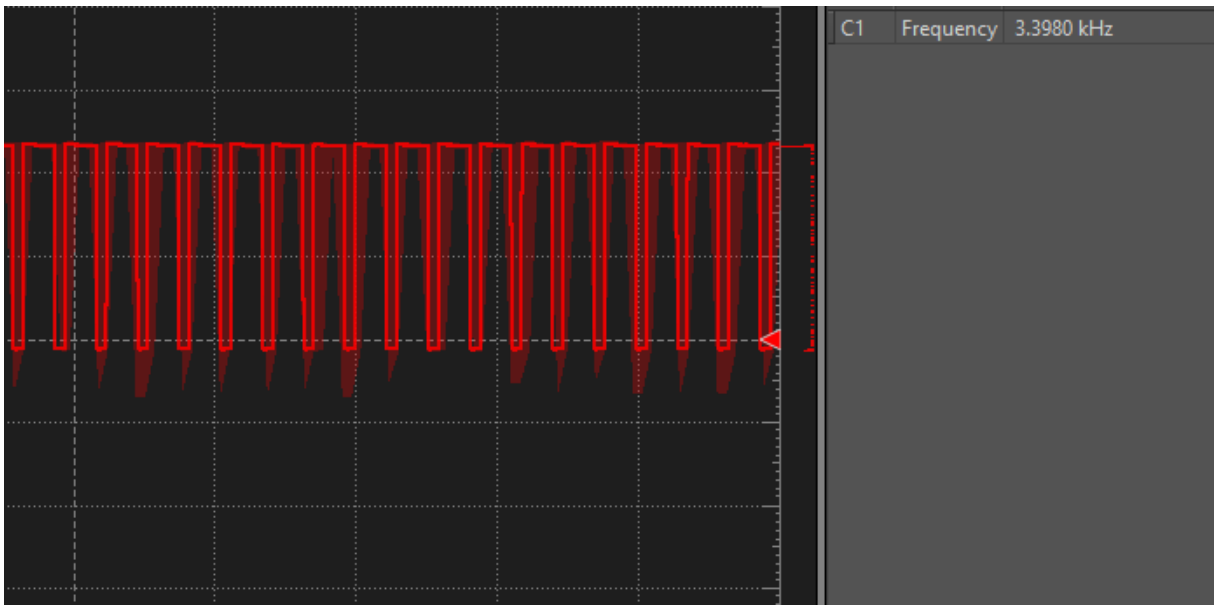


Şekil 17.2 IŞIK FREKANS ÇEVİRİCİ MODÜLÜ Devre Şeması

Fotodiyotun yönüne dikkat edilecek olursa, devreye ters bağlantı yapıldığı görülebilir. Bu bağlantı şeklinde fotodiyot üzerine düşen ışık miktarıyla doğru orantılı olarak, ters yönde iletkenliği değişmektedir. Yani fotodiyot bir anlamda ışığa bağlı direnç gibi çalışmaktadır. Bu sayede fotodiyot üzerine düşen ışık miktarı arttıkça direnci azalmakta ve 555 çıkış frekansı artmaktadır. Foto diyot üzerine düşen ışık miktarı azaldığında ise 555 çıkış frekansı azalır.



Şekil 17.3 Foto diyot üzerinde ışık miktarı azken 555 entegresinin çıkış sinyali



Şekil 17.4 Foto diyot üzerinde ışık miktarı fazlayken 555 entegresinin çıkış sinyali

Işık-Frekans Dönüştürücü Deneyi

Deneyde Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerinde bulunan IŞIK FREKANS ÇEVİRİCİ MODÜLÜ kullanılmıştır. Bu modül üzerinde bir adet Infrared (kızılötesi) led kullanılmış ve 555 entegresine bağlanmış olan fotodiyota yönlendirilmiştir. IR led ve foto diyot arasına opak bir cisim yaklaştırılarak fotodiyot üzerine düşen kızılötesi ışık miktarı azaltıldıkça EasyPIC v7 kartı üzerindeki LCD'de 555 entegresinin çıkış frekansının değişimi görülebilmektedir.

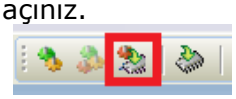
Örnek Uygulama

```
//************************
//Proje ismi       : DENEY 17-IŞIK FREKANS DÖNÜŞTÜRÜCÜ DENEYİ
//Amacı            : NE555 entegresiyle oluşturduğumuz kare
//                 dalgayı IR LED ile fotodiyoda yollayarak
//                 kare dalganın frekansını ölçüp LCD
//                 ekranda gözlemlemek.
//Test konfigürasyonu   :
//-----------------------------
//MCU              : 18F45K22
//Osilatör         : 32Mhz
//
//SW3.3     => ON
//SW4.6     => ON
// j12 jumperları I/O konumunda olmalıdır.
// Library Manager Sekmesinden Lcd aktif edilmelidir.
//************************

// LCD Bağlantıları
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
```

```
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// LCD Bağlantılarının Bitişi

int sayac,okunan=0;
int okunan_yenilendi=0;
int timerSayac=0;

void Timer1Kurulum(){
 T1CON          = 0x31;
 TMR1IF_bit = 0;
 TMR1H          = 0x3C;
 TMR1L          = 0xB0;
 TMR1IE_bit = 1;
 INTCON = 0xC0;
}

void main() {

 ANSELB = 0;        // PORTB Dijital olarak ayarlandı.
 ANSELC = 0;        // PORTC Dijital olarak ayarlandı.

 TRISC=0xff;        // PORTC giriş olarak ayarlandı.

 Lcd_Init();        // LCD kurulumu yapıldı

 Lcd_Cmd(_LCD_CLEAR);               // LCD ekranını temizler.
 Lcd_Cmd(_LCD_CURSOR_OFF);      // Cursor ekranda gösterilmez.
 LCD_Out(1,1,"Frekans:");
 LCD_OUT(2,6,"Hz");

 Timer1Kurulum();           // Timer1 50ms'de bir kesme oluşturacak şekilde kuruluyor.

 while(1)
 {
  while(PORTC==0);          // Buradaki "while" işlemleri 0'dan 1'e geçişleri
  sayac++;                  // her seferinde sadecd 1 kez saymak için kullanılmıştır.
  while(PORTC==1);

  if (okunan_yenilendi)
  {
   //(+48)'ler sayıları ASCII formatına çevirmek için eklenmiş.
   Lcd_Chr(2,1, ((okunan / 10000)  % 10) + 48);   //datanın on binler basamağı.
```

```
    Lcd_Chr(2,2, ((okunan / 1000)  % 10) + 48);    //datanın binler basamağı.
    Lcd_Chr(2,3, ((okunan / 100)   % 10) + 48);    //datanın yüzler basamağı.
    Lcd_Chr(2,4, ((okunan / 10)    % 10) + 48);    //datanın onlar basamağı.
    Lcd_Chr(2,5, (okunan % 10)+ 48);               //datanın birler basamağı.
    okunan_yenilendi=0;
   }
  }

}

// Kesme fonksiyonu.
void Interrupt(){
  if (TMR1IF_bit){
    TMR1IF_bit = 0;
    TMR1H          = 0x3C;
  TMR1L            = 0xB0;
    timerSayac++;
    if(timerSayac==20)     // Kesme 20 kez oluşmuşsa yani 1 saniye tamamlanmışsa..
    {
     okunan=sayac;
     okunan_yenilendi=1;
     sayac=0;
     timerSayac=0;
    }
  }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 17-IŞIK FREKANS DÖNÜŞTÜRÜCÜ DENEYİ" içerisindeki "Deney17.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki IŞIK FREKANS ÇEVİRİCİ MODÜLÜ'ndeki IR led ve Foto diyot arasına bir kağıt parçası veya bir opak cisim yakınlaştırarak LCD'de ki frekans değişimlerini gözlemleyiniz.

- **Frekans ölçmek için 18F45K22 mikrodenetleyicisinde bulunan "Capture" modülünün nasıl kullanılacağını araştırınız.**

DENEY. 18 GRAFİK LCD UYGULAMASI

AMAÇ

1) Grafik LCD görüntüleme biriminin özelliklerini öğrenmek.
2) MikroC GLCD kütüphanesini kullanarak, Grafik LCD'de istediğimiz görüntüyü elde etmek.

GEREKLİ MALZEME

1) EasyPIC7 Kartı,
2) GLCD modül,

Giriş

MikroC pro for PIC günümüzde gittikçe yaygınlaşan ve fiyatları ucuzlayan Grafik LCD modülleri için kapsamlı ve kolay kullanışlı bir kütüphane sağlar. Geçmiş deneylerde kullandığımız karakter LCD görüntüleme biriminde sadece alfanumerik karakter görüntülemek mümkün iken, "nokta (pixel) matris" yapısında geliştirilen GLCD modüllerinde; çubuk grafikler, x-y fonksiyonları gibi bit-eşlem formunda veri şeklindeki her türlü şekli görüntülemek mümkündür. Çok değişik çözünürlükte GLCD birimleri geliştirişmişse de en yaygın olarak kullanılan 128x64 çözünürlüklü olanlardır.

Bu konuda daha fazla bilgi edinmek isterseniz kullandığımız Grafik LCD'nin denetleyici entegresinin özelliklerini inceleyiniz. Bu deneyde kullandığımız PGM12864B GLCD birimi Samsung firmasının ürünü olan S6B0108 denetleyicisini kullanmaktadır.

PGM12864B kodlu tek renk (mono) GLCD'nin teknik özelliklerine ulaşmak için www.elektrovadi.com sitemizi ziyaret ediniz.

Giriş bölümünde de söylediğimiz gibi MikroC pro for PIC, GLCD modüllerin kullanımını kolaylaştıran kapsamlı bir kütüphane sunar. Bu kapsamlı kütüphanenin komutları aşağıdaki gibidir:

1. **Basit Komutlar**

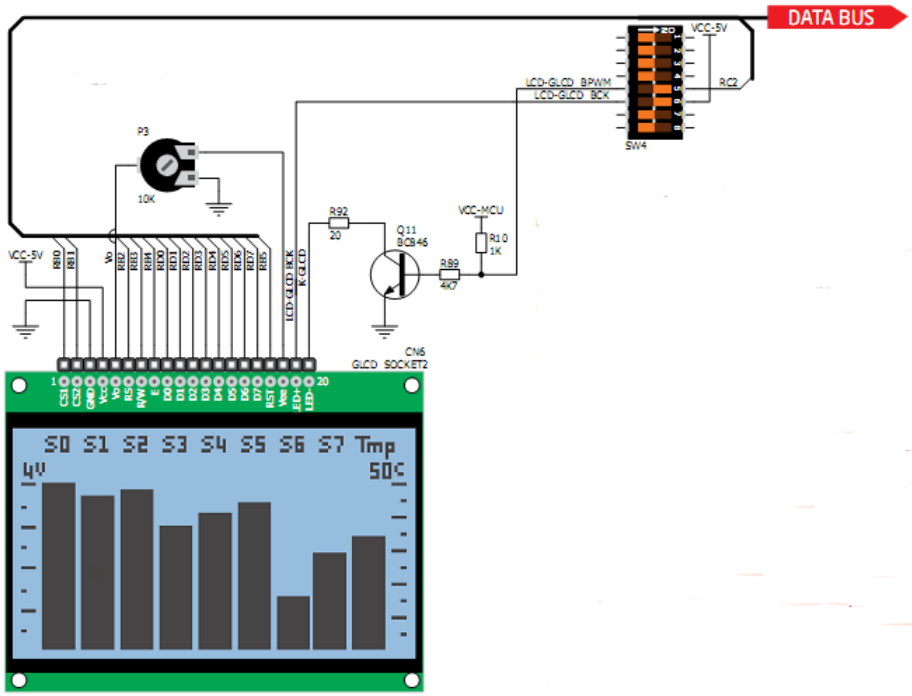
- Glcd_Init
- Glcd_Set_Side
- Glcd_Set_X
- Glcd_Set_Page
- Glcd_Read_Data
- Glcd_Write_Data
- Glcd_Set_Ext_Buffer

2. **Gelişmiş Komutlar**

- Glcd_Fill
- Glcd_Dot
- Glcd_Line
- Glcd_V_Line
- Glcd_H_Line
- Glcd_Rectangle
- Glcd_Rectangle_Round_Edges

- Glcd_Rectangle_Round_Edges_Fill
- Glcd_Box
- Glcd_Circle
- Glcd_Circle_Fill
- Glcd_Set_Font
- Glcd_Set_Font_Adv
- Glcd_Set_Ext_Font_Adv
- Glcd_Write_Char
- Glcd_Write_Char_Adv
- Glcd_Write_Text
- Glcd_Write_Text_Adv
- Glcd_Write_Const_Text_Adv
- Glcd_Image
- Glcd_Ext_Image
- Glcd_PartialImage
- Glcd_Ext_PartialImage

GLCD de görüntülemek istediğiniz özel şekilleri de yine MikroC pro for PIC'in sağladığı GLCD Bitmap Editor ile C dosyasına çevirip projelerinize ekleyebilirsiniz.



Şekil 18.3 Easy PIC v7 de kullanılan Grafik LCD uygulama şeması

Örnek Uygulama

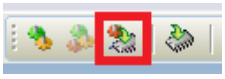
```
//************************
//Proje ismi      : DENEY-18 GRAFİK LCD UYGULAMASI
//Amacı           : GLCD kütüphanesini kullanarak,
//                  ekranda istediğimiz görüntüyü elde etmek.
//Test konfigürasyonu   :
//-----------------------------
//MCU             : 18F45K22
```

```
//Osilatör          : 32Mh
//
//SW4.6      => ON
//
// Library Manager Sekmesinden GLCD aktif edilmelidir.
//***********************

//resmimiz-------------------------------------------------
const code char resim[1024] = {
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 239, 199, 195, 195,   1,   1,   1,   1,   1,
1,   1,   1,   0,   0,   0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 127, 127,  63,  63,  63,  31,
15,  15,  15,   7,   7,   7,   7, 127, 255, 255, 255, 255, 255, 255, 255, 255, 255, 227, 195, 193, 193, 193,
193, 193, 192, 192, 192, 192, 225, 241, 251, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,   0,   0,   0,   0,   0,
0,   0,   0,   0,   0,   0, 131, 195, 195, 193, 192, 128,   0,   0,   0,   0,   0,   0,   0,   1,   3,   7,  15,  31, 255,
255, 255, 127,  63,  15,   7,   3,   3,   3,   1,   1,   0,  96, 248, 248, 248, 248, 240,   0,   0,   0,   1,   1,   3,
3,   7,  15,  63, 255, 241, 240, 240, 240, 240,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 224, 240, 240,
240, 240, 240, 240, 248, 255, 255, 195, 195, 131,   3,   3,   1,   1,   1,   1,   0,   0,   0,   0,   0, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,   0,   0,   0,   0,   0,
0,   0,   0,   0,   0,   0, 255, 255, 255, 255, 255, 255, 254,   0,   0,   0,   0,   0,   0,   0,   0,   0, 192, 224, 255,
255, 251, 128,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 120, 120, 248, 248, 248, 248, 248, 248, 248, 248,
120, 120,  56, 120, 248, 255, 255, 255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  63, 255,
255, 255, 255, 255, 127, 255, 255, 255, 255, 255, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 251, 248, 248, 248,
248, 248, 248, 240, 240, 224, 224, 224, 224, 227, 227, 227, 227, 225, 224, 224, 224, 240, 240, 240,
248, 252, 254, 255, 255, 255, 255, 255, 255, 255, 255, 252, 248, 248, 240, 240, 240, 224, 224, 224,
224, 224, 224, 224, 224, 224, 224, 224, 240, 240, 240, 248, 248, 252, 255, 255, 255, 255, 255, 255,
255, 248, 248, 240, 240, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 240, 248, 248, 254,
255, 255, 243, 225, 224,  32,  32, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 225, 243, 255,
255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,   3,   1,   1, 217, 217, 217,
217, 217, 217, 255, 255,   1,   1,   3, 255, 255, 255, 255, 255, 255, 255,   3,   1, 153, 217, 217, 217, 217,
217, 255, 255, 255,   1,   1,  63, 159, 207,   7,  35, 121, 253, 255, 255, 253, 249, 249, 249,   1, 249, 249,
249, 249, 255, 255,   1,   1, 185, 185, 185,  57,  25, 195, 231, 255, 255,  15,   3, 241, 249, 249, 253,
249, 249, 227,   3,   7, 255, 255,   1,   1, 193, 199,  31,  63, 127,  23,   1, 255, 255, 255,   1,   1, 255, 255,
3,   1,   1, 159, 207,   7,  51, 121, 249, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 252, 248, 248, 251, 251,
251, 251, 251, 251, 255, 255, 248, 248, 248, 251, 251, 251, 251, 251, 255, 255, 248, 248, 249, 251,
251, 251, 251, 251, 255, 255, 255, 248, 248, 255, 255, 255, 255, 254, 248, 249, 251, 255, 255, 255,
```

```
255, 255, 248, 255, 255, 255, 255, 255, 255, 248, 248, 255, 255, 255, 255, 252, 248, 251, 251, 255,
254, 252, 252, 249, 249, 251, 251, 249, 252, 252, 254, 255, 255, 248, 252, 255, 255, 255, 254, 248,
248, 248, 255, 255, 255, 248, 248, 255, 255, 248, 248, 248, 255, 255, 255, 252, 248, 248, 251, 255,
255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255
};
//------------------------------------------------------------

// Glcdmoduleconnections
char GLCD_DataPort at PORTD;

sbit GLCD_CS1 at LATB1_bit;
sbit GLCD_CS2 at LATB0_bit;
sbit GLCD_RS  at LATB2_bit;
sbit GLCD_RW  at LATB3_bit;
sbit GLCD_EN  at LATB4_bit;
sbit GLCD_RST at LATB5_bit;

sbit GLCD_CS1_Direction at TRISB0_bit;
sbit GLCD_CS2_Direction at TRISB1_bit;
sbit GLCD_RS_Direction  at TRISB2_bit;
sbit GLCD_RW_Direction  at TRISB3_bit;
sbit GLCD_EN_Direction  at TRISB4_bit;
sbit GLCD_RST_Direction at TRISB5_bit;
// EndGlcdmoduleconnections

void main()
{
  unsigned short sayac;
  char i=0;

  ANSELB=0;             // PORTB dijital olarak ayarlanıyor.
  ANSELD=0;             // PORTD dijital olarak ayarlanıyor.

  Glcd_Init();                      // GLCD yi başlat
  Glcd_Fill(0x00);                  // GLCD yi temizle

  while(1)
  {
   int i;
   Glcd_Image(resim);                // resmi çiz
   delay_ms(2000);
   Glcd_Fill(0x00);                  // GLCD'yi temizle

   Glcd_Box(62,40,124,56,1);        // Kutu çiz
```

```
  Glcd_Rectangle(5,5,84,35,1);          // Dörtgen çiz
  Glcd_Line(0, 0, 127, 63, 1);          // Çizgi çiz
  delay_ms(2000);

  for(i = 5; i<60; i+=5 ){              // Yatay ve dikey çizgiler çiziliyor.
    Delay_ms(250);
    Glcd_V_Line(2, 54, i, 1);
    Glcd_H_Line(2, 120, i, 1);
  }
  delay_ms(2000);

  Glcd_Fill(0x00);                      // GLCD'yi temizle

  for (i=1; i<=10; i++){                // Çemberler çiziliyor.
    Glcd_Circle(63,32, 3*i, 1);
  }
  delay_ms(2000);;

  Glcd_Box(12,20, 70,57, 2);            // Kutu çiziliyor.
  delay_ms(2000);
 }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 18-GRAFİK LCD UYGULAMASI" içerisindeki "Deney18.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. EasyPIC v7 kartı üzerindeki GLCD'yi gözlemleyiniz.

- **Yukarıdaki kodlarda değişiklik yaparak kendi belirlediğiniz tek renkli bir bmp resmini GLCD üzerinde gösteriniz.**

DENEY.19 RS485 (Recommended Standart 485) Haberleşmesi

AMAÇ

1) RS485 protokolünün yapısının incelenmesi
2) Bu protokol üzerinden haberleşme uygulamasının gerçekleştirilmesi

GEREKLİ MALZEME

1) EasyPIC v7 Kartı (2 adet)
2) BETI MCU Uygulama ve Geliştirme Seti (2 adet)

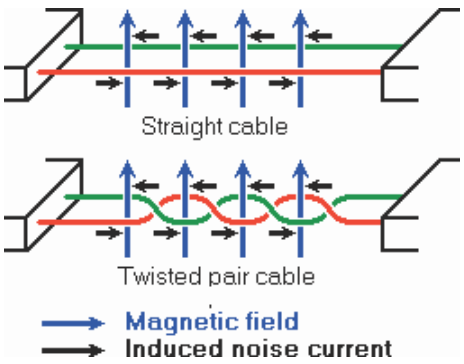
Giriş

TIA/EIA-485 veya eski adıyla RS-485; 1200 m'ye kadar kablo uzunluğuna izin veren, çok noktalı, yarı çift yönlü (Half duplex), seri iletişim veriyolu standardıdır. Tek bir kabloya birden fazla cihazın bağlanmasına olanak sağlayan bir sistemdir. RS-232 standardının uzun mesafelisi olarak düsünülebilir. RS-232, 5 metreye kadar kablo uzunluklarının desteklerken, EIA-485'te bu uzunluk çok daha fazladır. İletişim hızı kullanılacak kablo uzunluğu ve türüne göre değişkendir. Bağlantılarda, üreticiye ve adres yolu türüne (Bus) bağlı olarak çeşitli kablolar kullanılabilir.

RS485'in Özellikleri:

RS232'deki eksikliklerden biri sinyal iletim hattı üzerindeki gürültüleri engelleyemiyor ol- masıdır. Alıcı ve verici verinin voltajını karşılaştırırlar ve sonrasında ise bir genel sıfır hattı (zero line) ile hatlar arasında handshake (el sıkışma) işlemini gerçekleştirirler. Toprak düzeyindeki bir kayma (shift) çok feci sonuçlar doğurabilir. Bu nedenden dolayı RS232 nin tetikleme voltajı daima +/-3V'un üzerindedir. Gürültü kolayca ortaya çıkacak ve maksimum iletim mesafesi ve maksimum veri aktarım hızı düşecektir.

RS485'te bu durumun tersine genel sıfır hattı yoktur. Toprak düzeyindeki birkaç voltluk değişme alıcı ve vericide herhangi bir soruna neden olmayacaktır. RS485 sinyalleri kayan bir yapıya sahip olup sig+ ve sig- hatları üzerinden iletileceklerdir. RS485 alıcısı, sinyal hattı üzerindeki voltaj düzeylerinin mutlak değerlerini almak yerine, her iki hat arasındaki voltaj düzeylerinin farkını almaktadır. Eğer sig+ ve sig- veri iletim hatları birbirleri ile sarmal bir yapıya dönüştürülürlerse maksimum verim alınacaktır.



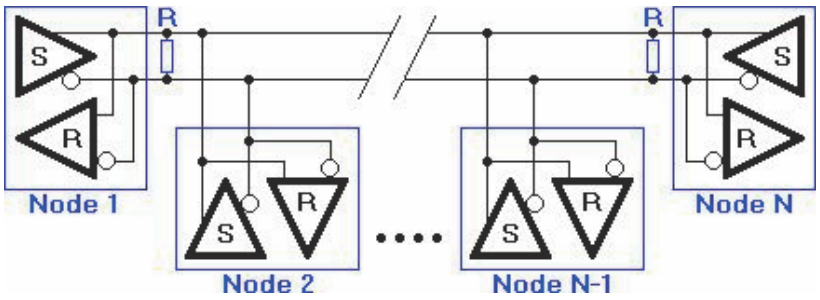
Şekil 19.1 Doğru bağlantılı ve Sarmal yapılı kablolarda gürültü oluşumu

Daha düşük gürültü ile veri aktarımı yapmak için sarmal yapıya sahip olan STP (shielded twisted pair) ve FTP (foiled twisted pair) ağ kabloları kullanılarak sağlanabilinir. RS485 te veri iletimi daha uzak mesafelere yapılabilir. Yaklaşık 1200 m ye kadar veri iletebilmek mümkündür. Kafanızda şekillenmesi maksadıyla RS485'in diğer seri haberleşme protokolleri ile karşılaştırılması aşağıdaki tabloda verilmiştir

	RS232	RS423	RS422	RS485
Differential	no	no	yes	yes
Max number of drivers Max number of receivers	1 1	1 10	1 10	32 32
Modes of operation	half duplex full duplex	half duplex	half duplex	half duplex
Network topology	point-to-point	multidrop	multidrop	multipoint
Max distance (acc. standard)	15 m	1200 m	1200 m	1200 m
Max speed at 12 m Max speed at 1200 m	20 kbs (1 kbs)	100 kbs 1 kbs	10 Mbs 100 kbs	35 Mbs 100 kbs
Max slew rate	30 V/µs	adjustable	n/a	n/a
Receiver input resistance	3..7 kΩ	□ 4 kΩ	□ 4 kΩ	□ 12 kΩ
Driver load impedance	3..7 kΩ	□ 450 Ω	100 Ω	54 Ω
Receiver input sensitivity	±3 V	±200 mV	±200 mV	±200 mV
Receiver input range	±15 V	±12 V	±10 V	–7..12 V
Max driver output voltage	±25 V	±6 V	±6 V	–7..12 V
Min driver output voltage (with load)	±5 V	±3.6 V	±2.0 V	±1.5 V

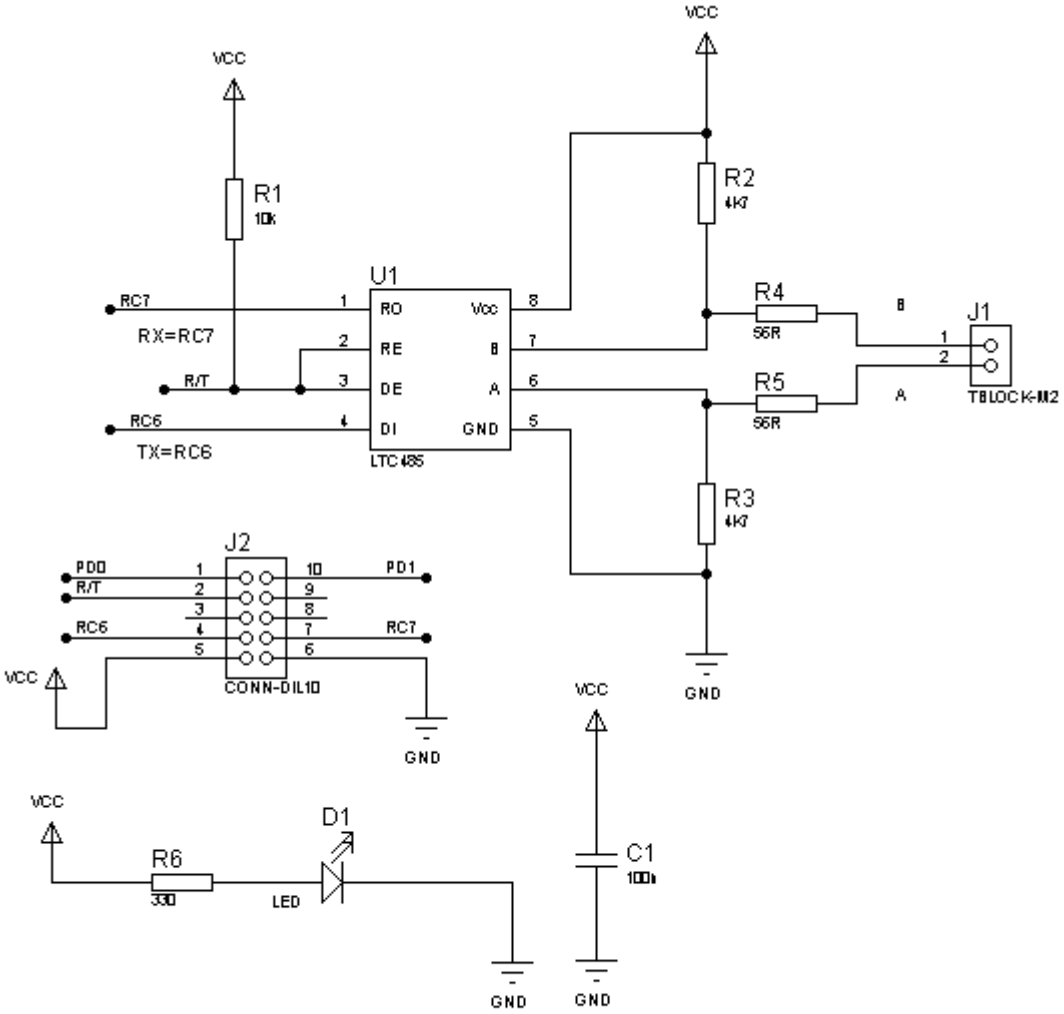
RS485 AĞ TOPOLOJİSİ:

Muhtemelen RS485 in sahip olduğu ağ topolojisi onu, veri aktarımı ve kontrol uygula- malarında, diğer seri haberleşme protokollerinden üstün tutmaktadır. RS485 çoklu alıcı/verici arasında ağ-çalışması yapabilmeyi sağlayan bir arayüzdür. RS485 alıcısı 12 kOhm luk bir giriş empedansı ile birlikte kullanılırsa yaklaşık 32 adet kadar aygıt ile ağ kurma imkanı tanımaktadır. Bu giriş empedansını yüksek tutarsanız 256 ya kadar farklı aygıtı haberleştirebilirsiniz. Ayrıca RS485 tekrarlayıcıları (repeater) kullanarak bu sayıyı binlere çıkarabilir ve haberleşme mesafesini km'lere çıkarabilirsiniz. Ayrıca özel zeki ağ donanımları gerektirmeyen, sadece yazılımsal olarak oluşturulan uygulamalarda çoğu özellik RS232 ile aynıdır.



Şekil 19.2 RS485 ağ topolojisi

RS485 DENEYİ



Şekil 19.3 Beti MCU Seti üzerinde kullanılan RS485 Devre şeması

Bu uygulamada Master olarak tanımlanan mikrodenetleyicinin PORTb pinlerine bağlı butonlardan gönderilen değerler Slave olarak tanımlanan mikrodenetleyiciye gönderilecektir. Slave mikrodenetleyiciye gönderilen değerler PORTD'ye bağlı ledlerde gösterilecek ve daha sonra bit bazında terslenerek Master mikrodenetleyiciye geri gönderilecektir. Master mikrodenetleyici de kendisine geri gelen terslenmiş veriyi PORTD'ye bağlı ledlerde gösterecektir.

MikroC RS485 kütüphanesinde veri yapısı 7 bayttan oluşur ve aşağıdaki gibidir.

- `data_tampn dizisi`: 7 byte uzunluğunda bir veri paketidir.
- `data[0..2]`: Bu baytlar veriyi taşıyan baytlardır.
- `data_buffer[3]`: Kaç bayt veri geldiğini gösteren bayttır.
- `data_buffer[4]`: Mesaj geldiğinde bu bayt 255 değerini alır.
- `data_buffer[5]`: Eğer iletişimde hata oluşursa 255 değerini alır.
- `data_buffer[6]`: Veriyi gönderen Slave cihazın adresini taşır.

EasyPIC v7 kartının PORTC pinleriyle Beti Mikrodenetleyici Uygulama ve Gelştirme Seti üzerindeki flat kablo bağlantısı yukarıdaki fotoğrafta görüldüğü gibi yapılmalıdır.

Örnek Uygulama (Master aygıt için)

```
//************************
//Proje ismi        : Deney 19- RS485 Uygulaması
//Amacı             : RS485  protokolünün yapısının incelenmesi
//Test konfigürasyonu       :
//------------------------
//MCU               : 18F45K22
//Osilatör          : 32Mh
//
//SW3.2, 3.4  => ON, DİGERLERİ => OFF
//j12 jumperları I/O konumunda olmalıdır.
//J17 => VCC
//
//Library Manager Sekmesinden RS485 ve UART aktif edilmelidir.
//************************

char dat[7];              // Mesaj alıp göndermek için kullanılan tampon dizisi.

sbit  rs485_rxtx_pin at RC2_bit;                    // veri alıp gönderme pini ayarlanıyor.
sbit  rs485_rxtx_pin_direction at TRISC2_bit;   // veri alıp gönderme pininin yön kaydedicisi
belirtiliyor.
unsigned int hataSayac=0;
//Kesme altprogramı
```

```
void interrupt()
{
  RS485Master_Receive(dat);// Veri geldiğinde bu kesme fonksiyonu çalışıp veriyi "dat" dizisine atar.
}

void main()
{
  ANSELB = 0;          // PORTB dijital olarak ayarlanıyor.
  ANSELC = 0;          // PORTC dijital olarak ayarlanıyor.
  ANSELD = 0;          // PORTD dijital olarak ayarlanıyor.

  TRISB = 0xFF;        // PORTB giriş olarak ayarlanıyor.
  TRISD = 0;           // PORTD çıkış olarak ayarlanıyor.
  LATD  = 0;           // PORTD pinleri 0 lanıyor.

  C1ON_bit = 0;        // Disable comparators
  C2ON_bit = 0;

  UART1_Init(9600);     // UART1 kuruluyor.
  Delay_ms(100);
  RS485Master_Init();   //Mikrodenetleyici Master olarak ayarlanıyor.
  dat[0] = 0;           // "dat" tampon sizisi içeriği sıfırlanıyor.
  dat[1] = 0;
  dat[2] = 0;
  dat[3] = 0;
  dat[4] = 0;
  dat[5] = 0;
  dat[6] = 0;

  RCIE_bit = 1;          // UART1 veri alma kesmesi aktif ediliyor.
  TXIE_bit = 0;          // UART1 veri gönderme kesmesi pasif ediliyor.
  PEIE_bit = 1;          // Çevresel kesmeler aktif ediliyor.
  GIE_bit = 1;           // Bütün kesmelere izin veriliyor.

  RS485Master_Send(dat,1,160);  // 160 adresli mikrodenetleyiciye dat dizisi 1 byte veri ile
gönderiliyor.
  while (1)
  {
    hataSayac++;
    if (dat[5]) {          // Eğer bir hata meydana gelirse
      PORTD = 0xAA;        // Bu durumu PORTD'ye 0xAA verisi yükleyerek göster.
      dat[5]=0;
    }

    if (dat[4])            // Eğer veri alma işlemi hatasız gerçekleştiyse.
    {
      hataSayac = 0;                  // hata sayacını sıfırla.
      dat[4] = 0;                     // Veri alma haber verisi temizleniyor.
      LATD = dat[0];                  // Gelen veri PORTD'de gösteriliyor.
      dat[0]=PORTB;                   // dat dizisinin 0. elemanına PORTB'den okunan değer yükleniyor.
      RS485Master_Send(dat,1,160);  // 160 adresli mikrodenetleyiciye dat dizisi 1 byte veri ile
```

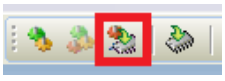
```
//gönderiliyor.
   }

   if(hataSayac > 10000)                // Eğer 10000 çevrim veri iletişimi olmazsa
   {
     hataSayac = 0;                     // hataSayac değişkennin sıfırla ve
     RS485Master_Send(dat,1,160);       // 160 adresli slave cihaza veri gönder.
   }
 }
}
```

Örnek Uygulama (Slave aygıt için)

```
//************************
//Proje ismi       : Deney 19- RS485 Uygulaması
//Amacı            : RS485  protokolünün yapısının incelenmesi
//Test konfigürasyonu      :
//-------------------------
//MCU              : 18F45K22
//Osilatör         : 32Mh
//
//3.4  => ON, DİGERLERİ => OFF
//
//
//Library Manager Sekmesinden RS485 ve UART aktif edilmelidir.
//************************

char dat[7];              // Mesaj alıp göndermek için kullanılan tampon dizisi.

sbit  rs485_rxtx_pin at RC2_bit;                  // veri alıp gönderme pini ayarlanıyor.
sbit  rs485_rxtx_pin_direction at TRISC2_bit;   // veri alıp gönderme pininin yön kaydedicisi
belirtiliyor.

//Kesme altprogramı
void interrupt() {
 RS485Slave_Receive(dat); // Veri geldiğinde bu kesme fonksiyonu çalışıp veriyi "dat" dizisine atar.
}

void main() {

  ANSELC = 0;            // PORTC dijital olarak ayarlanıyor.
  ANSELD = 0;           // PORTD dijital olarak ayarlanıyor.

  TRISD = 0;            // PORTD çıkış olarak ayarlanıyor.
  LATD  = 0;            // PORTD pinleri 0 lanıyor.

  UART1_Init(9600);               // UART1 kuruluyor.
  Delay_ms(100);
  RS485Slave_Init(160);           // Mikrodenetleyici Slave olarak 160 adresiyle ayarlanıyor.
```

```
dat[0] = 0;   // "dat" tampon sizisi içeriği sıfırlanıyor.
dat[1] = 0;
dat[2] = 0;
dat[3] = 0;
dat[4] = 0;
dat[5] = 0;
dat[6] = 0;

RCIE_bit = 1;              // UART1 veri alma kesmesi aktif ediliyor.
TXIE_bit = 0;              // UART1 veri gönderme kesmesi pasif ediliyor.
PEIE_bit = 1;              // Çevresel kesmeler aktif ediliyor.
GIE_bit = 1;               // Bütün kesmelere izin veriliyor.

while (1)
{
  if (dat[5]) {            // Eğer bir hata ile karşılaşılırsa
    LATD = 0xAA;           // PORTD'ye 0xAA verisi yükleniyor.
    dat[5] = 0;            // hata verisi temizleniyor.
  }

  if (dat[4])              // Eğer veri alma işlemi hatasız gerçekleştiyse.
  {
    dat[4] = 0;            // Veri alma haber verisi temizleniyor.
    LATD = dat[0];         // Gelen veri PORTD'de gösteriliyor.
    Delay_ms(1);
    dat[0]=~dat[0];                // Veri bit bazında tersleniyor.
    RS485Slave_Send(dat,1);        // Terslenen veri Master'a gönderiliyor.
  }
 }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 19-RS485 (Recommended Standart 485) Haberleşmesi" içerisindeki "Deney19_Master.mcppi" projesini açınız ve master olarak kullanılacak sete yükleyiniz. "Deneyler" klasöründe bulunan "DENEY 19-RS485 (Recommended Standart 485) Haberleşmesi" içerisindeki "Deney19_Slave.mcppi" projesini açınız ve slave olarak kullanılacak sete yükleyiniz.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Master cihazda PORTB bölümündeki butonlara basarak Slave cihazın PORTD kısmındaki ledlerdeki değişimleri gözlemleyiniz.

- **3 Farklı deney setini birbirine bağlayarak, Master cihazdan farklı iki Slave cihaza ayrı ayrı veri göndermeyi deneyiniz.**

DENEY.20 DOKUNMATİK PANEL UYGULAMASI

AMAÇ

1) Dirençli dokunmatik panelin nasıl çalıştığını öğrenmek.
2) Dokunmatik paneli, Grafik LCD modüle yapıştırarak, dokunmatik buton uygulaması geliştirmek.

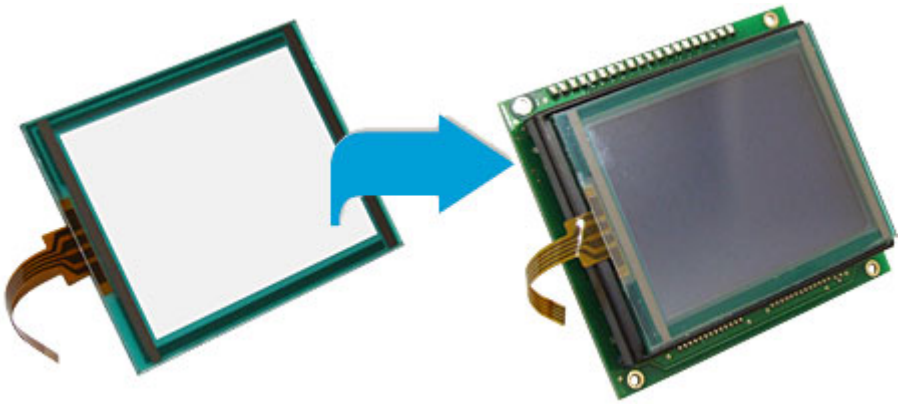
GEREKLİ MALZEME

1) EasyPICv7 Kartı,
2) GLCD modül,
3) Dokunmatik panel,

Giriş:

Dokunmatik Panel

Dokunmatik panel, Grafik LCD modülün üzerine yerleştirilen, kendinden yapışkanlı saydam bir paneldir. Basınca çok hassas yapısı sayesinde, en küçük dokunuşlarda bile panelin çıkış sinyalinde değişmeler olabilir. Birkaç çeşit dokunmatik panel çeşidi vardır. Bunların en basiti ve bizim de uygulamamızda kullanacağımız çeşit, dirençli dokunmatik paneldir.

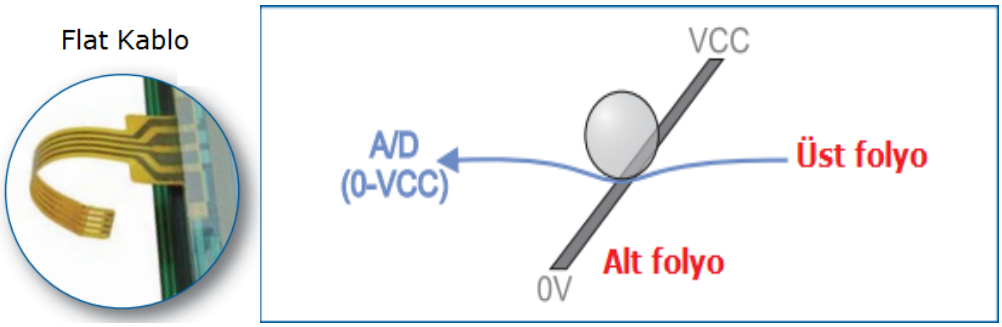


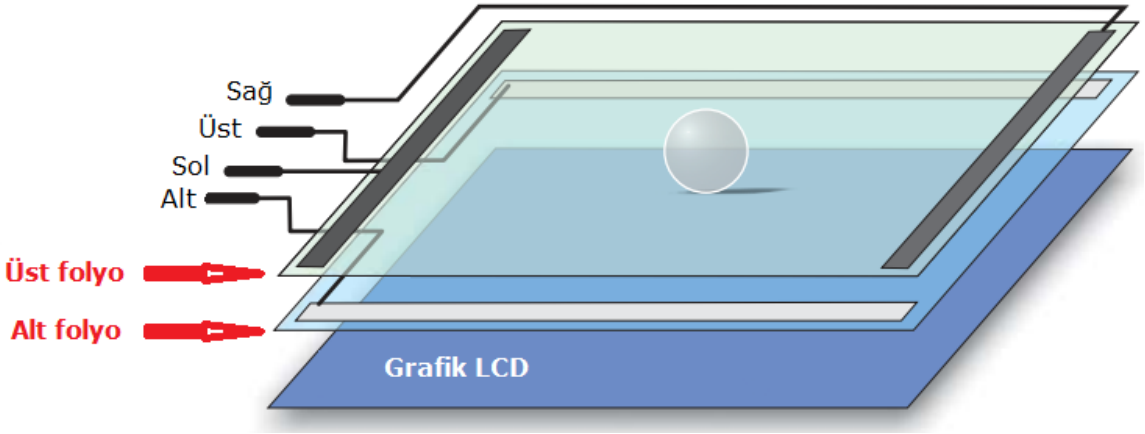
Şekil 10.1 Dokunmatik Panelin 128x64 LCD ile kullanımı

Dokunmatik Panelin Çalışma Prensibi

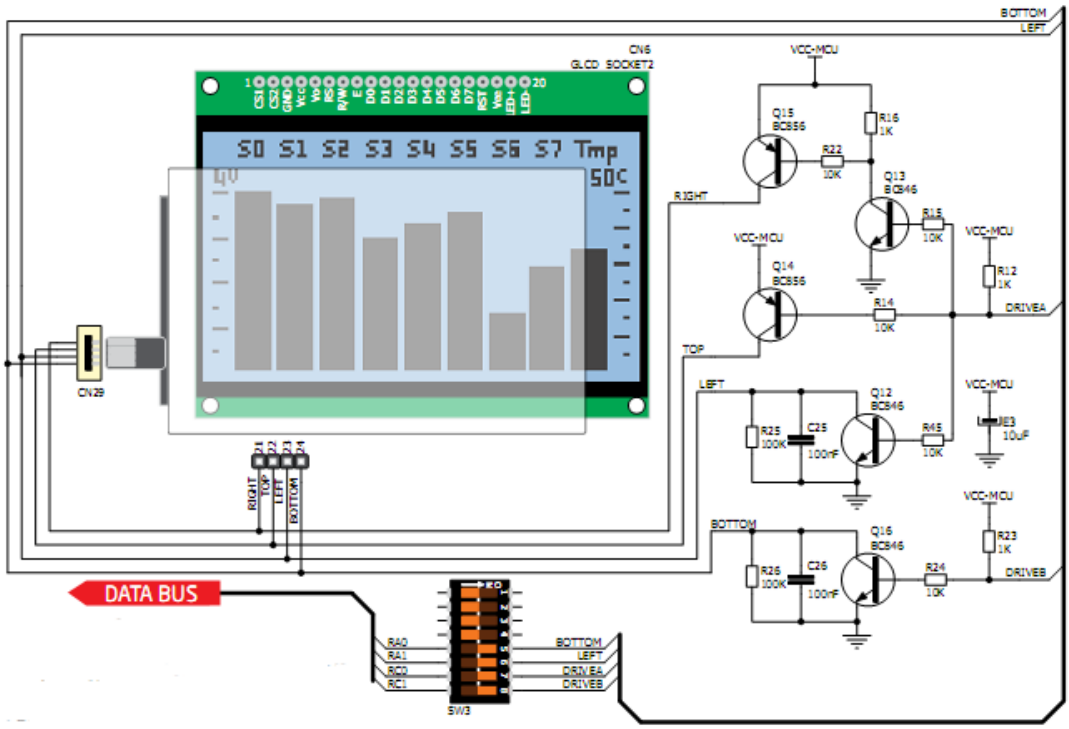
Dirençli bir dokunmatik panelin yapısını sandviçe benzetebiliriz. Panel, iç tarafları dirençli bir özelliğe sahip iki adet folyodan oluşur. Bu dirençler genelde 1KΩ'u geçmez. Bu folyoların dış tarafları, panelin flat kablosuna bağlıdır. Dokunmatik panelin neresine basıldığına karar verme işlemi iki aşamadan oluşur. İlk aşama dokunulan noktanın X eksenindeki koordinatının belirlenmesi, ikinci aşama dokunulan noktanın Y eksenindeki koordinatının belirlenmesi şeklindedir. X eksenindeki koordinatın belirlenmesi, voltajın bölünmesi prensibine dayanır. Üstteki folyonun sol tarafı GND'ye, sağ tarafı güç kaynağına bağlanır. Panele dokunduğumuzda, üst katmandaki folyo üzerindeki direnç noktalarıyla devreyi tamamlar ve voltajı bölmüş oluruz. Bölünen voltaj okunur, elde

edilen veri işlenerek X eksenindeki koordinatımız belirlenmiş olur. Y eksenindeki koordinatın belirlenmesi de aynı şekilde gerçekleşir. Alt katmandaki folyonun alt tarafı GND'ye, üst tarafı güç kaynağına bağlanır. Panele dokunduğumuzda, alt katmandaki direnç noktalarıyla devreyi tamamlar, voltajı bölmüş oluruz. Bölünmüş voltaj okunarak Y eksenindeki koordinat da belirlenmiş olur.





Şekil 20.2 Dokunmatik Panel Çalışma Prensibi



Şekil 20.3 Easy PIC7 Üzerindeki Dokunmatik Panel Devre Şeması

Bu uygulamada GLCD ekranında çizdirilen kutulara dokunulduğunda , PORTE'nin 0 ve 1. Pinlerine bağlı ledler yakılıp söndürülmektedir.

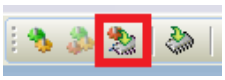
Örnek Uygulama

```
//************************
//Proje ismi        : dokunmatik
//Amacı             : Grafik LCD modülün üzerine Dokunmatik Panel
//                  yapıştırarak basit bir dokunmatik
//                  buton uygulaması yapmak.
//Test konfigürasyonu       :
//-------------------------
//MCU               : 18F45K22
//Osilatör        : 32Mhz
//
//SW4.6      => ON
//SW3.1, 3.5, 3.6, 3.7, 3.8=> ON
//j12 jumperları I/O konumunda olmalıdır.
// Library Manager Sekmesinden ADC ve Glcd kütüphanelerini aktif ediniz.
//
//************************

unsigned int x_coord, y_coord, x_coord128, y_coord64;
```

```
// Glcdmodul bağlantıları
char GLCD_DataPort at PORTD;
sbit GLCD_CS1 at LATB1_bit;
sbit GLCD_CS2 at LATB0_bit;
sbit GLCD_RS  at LATB2_bit;
sbit GLCD_RW  at LATB3_bit;
sbit GLCD_EN  at LATB4_bit;
sbit GLCD_RST at LATB5_bit;

sbit GLCD_CS1_Direction at TRISB1_bit;
sbit GLCD_CS2_Direction at TRISB0_bit;
sbit GLCD_RS_Direction  at TRISB2_bit;
sbit GLCD_RW_Direction  at TRISB3_bit;
sbit GLCD_EN_Direction  at TRISB4_bit;
sbit GLCD_RST_Direction at TRISB5_bit;

// Glcdmodul bağlantıları

unsigned int GetX()
{
  // X koordinatının okunması
  LATC.F0 = 1;
  LATC.F1 = 0;
  Delay_ms(5);
  return ADC_read(0);
}
unsigned int GetY()
{
  // Y koordinatının okunması
  LATC.F0 = 0;
  LATC.F1 = 1;
  Delay_ms(5);
  return ADC_read(1);
}

void main() {
  ANSELA = 0x03;              // RA0 ve RA1 pini analog olarak ayarlandı
  ANSELB = 0;                 // Port B dijital olarak ayarlandı
  ANSELC = 0;                 // Port C dijital olarak ayarlandı
  ANSELD = 0;                 // Port D dijital olarak ayarlandı
  ANSELE = 0;                 // Port E dijital olarak ayarlandı

  TRISA = 0x03;
  LATA = 0x00;

  TRISC = 0 ;
  LATC = 0 ;

  TRISE=0;
  LATE=0;
  Glcd_Init();                // Glcd başladı
```

```
  Glcd_Fill(0);                 // Glcd Temizlendi

  Glcd_Rectangle(8,16,60,48,1);     // butonlar çizildi
  Glcd_Rectangle(68,16,120,48,1);
  Glcd_Box(10,18,58,46,1);
  Glcd_Box(70,18,118,46,1);

  while (1)
  {
    x_coord = GetX();
    y_coord = GetY();                 // RA1 ve RA0 pinlerinden alınan analog
    x_coord128 = (x_coord /8);          // veri Grafik LCD'deki piksel bilgisine
    y_coord64 = 64 -(y_coord /16);       // dönüştürüldü

    // Eğer ilk buton seçilirse
    if ((x_coord128 >= 10) && (x_coord128 <= 58) && (y_coord64 >= 18) && (y_coord64 <= 46))
    {
      if(PORTE.F1 == 0)
      {
       LATE.F1 = 1;                       // RC7 LED'ini yak
       Glcd_Circle_Fill(34, 32, 10, 0);
      }
      else
      {
       LATE.F1 = 0;                       // RC7 LED'ini södür
       Glcd_Circle_Fill(34, 32, 10, 1);
      }
    }                                       // Eğer ikinci buton seçilirse
    if ((x_coord128 >= 70) && (x_coord128 <= 118) && (y_coord64 >= 18) && (y_coord64 <= 46))
    {
     if(PORTE.F0 == 0)
     {
      LATE.F0 = 1;                    // RC6 LED'ini yak
      Glcd_Circle_Fill(94, 32, 10, 0);
     }
     else
     {
      LATE.F0 = 0;                    // RC6 LED'ini söndür
      Glcd_Circle_Fill(94, 32, 10, 1);
     }
    }
    Delay_ms(100);
  }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 20-DOKUNMATİK PANEL UYGULAMASI.doc" içerisindeki "Deney20.mcppi" projesini açınız.
4.
5. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
6. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
7. EasyPIC v7 kartı üzerindeki GLCD'yi gözlemleyiniz.

- **PORTE pinine bağlı 4 adet farklı ledi, GLCD üzerindeki 4 ayrı kutuya dokunarak yakıp söndüren programı yazınız.**

DENEY.21 SES ÜRETECİ DENEYİ

AMAÇ

PIC18F45K22 çıkışlarını kullanarak ses üretmek.

GEREKLİ MALZEME

EasyPICv7 Kartı.

Giriş:

Darbe Genişlik Modülasyonu (PWM); sayısal system çıkışları ile örneksel devrelerin ve elemanların kontrolünde kullanılan güçlü bir tekniktir. Bu yöntem ile ilgili detaylı bilgi Deney14'te detail olarak anlatılmıştır. Mikrodenetleyici'nin bir çıkışına bağlı olan buzzer sayesinde bir porttan üretilen PWM sinyali sayesinde ses üretilmiş olur.

MikroC for PIC PWM kütüphanesi

MikroC pro for PIC, kullanımı çok kolay ve geniş kapsamlı bir kütüphane sunar. Bu kütüphane sayesinde PWM dalganızın frekansını ve Duty Cycle'ını (Görev saykılı) kolayca değiştirebilirsiniz. MikroC pro for PIC PWM kütüphanesinin fonksiyonları şunlardır:

- PWM1_Init
- PWM1_Set_Duty
- PWM1_Start
- PWM1_Stop

PWM1_Init: PWM modülünü 0 Duty Cycle ile başlatır. PWM1_Init(frekans) şeklinde girilen frekans değeri PWM dalgamızın frekansını belirler.

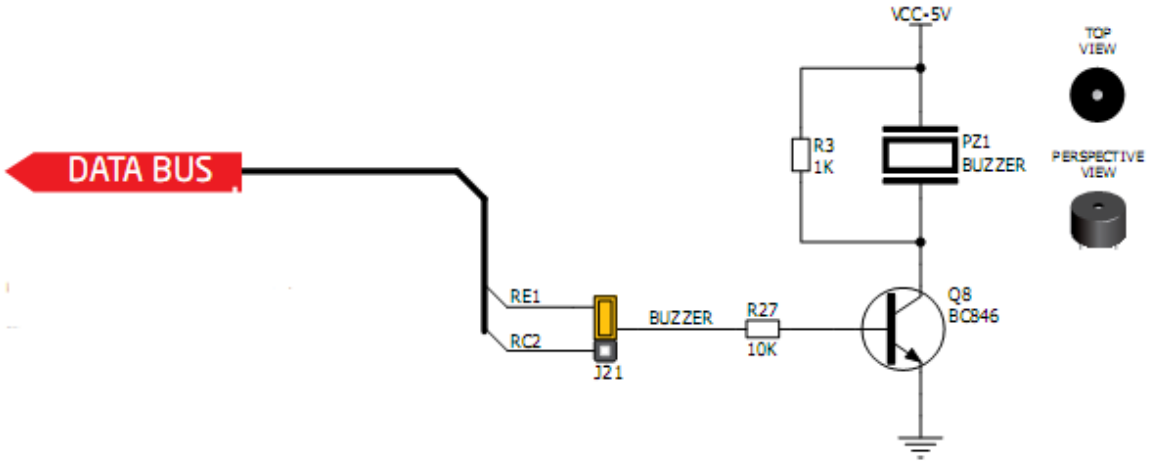
```
PWM1_Init(5000); // 5KHz PWM oluşturulur.
```

PWM1_Set_Duty: PWM dalgasının Duty Cycle oranını ayarlar. PWM1_Set_Duty((yüzde*255)/100) şeklinde girilen yüzde değeri Duty Cycle değerine denktir.

```
PWM1_Set_Duty(192); // %75 Duty Cycle ile PWM üretir.
```

PWM1_Start: PWM dalgasını başlatır.

PWM1_Stop: PWM dalgasını durdurur.



Şekil 21.1 EasyPIC v7 üzerindeki ses üreteci ile ilgili devre şeması.

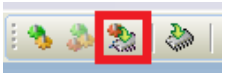
Bu uygulamada EasyPIC v7 üzerindeki aktif buzzer kullanılarak farklı PWM frekanslarında %50 Duty Cycle oranında farklı sesler üretilmektedir.

Örnek program:

```
//************************
//Projeismi       : DENEY 21: SES ÜRETECİ DENEYİ
//Amacı           : PWM dalgasıkullanarak, buzzer'danfarklıtonlareldeetmek.
//Test konfigürasyonu   :
//------------------------
//MCU             : 18F45K22
//Osilatör        : 8Mhz
// J12 jumperları I/O konumunda olmalıdır.
// J21      => RC2
//
// Library Manager Sekmesinden PWM12 aktif edilmelidir.
//************************
void ses_1()
{
PWM1_Init(500);      // frekans 500Hz
PWM1_Start();
PWM1_Set_Duty (126);
delay_ms(500);

}

void ses_2()
{
PWM1_Init(1500);     // frekans 1500Hz
PWM1_Start();
PWM1_Set_Duty (126);
delay_ms(500);
}

void ses_3()
{
```

```
PWM1_Init(2500);     // frekans 2500Hz
PWM1_Start();
PWM1_Set_Duty (126);
delay_ms(500);

}

void ses_4()
{
PWM1_Init(3500);      // frekans 3500Hz
PWM1_Start();
PWM1_Set_Duty (126);
delay_ms(500);

}

void main() {
while(1)
 {
   ses_1();
   ses_2();
   ses_3();
   ses_4();
 }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 21-SES ÜRETECİ DENEYİ" içerisindeki "Deney21.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. EasyPIC v7 kartı üzerindeki Buzzer'ın ürettiği sesleri dinleyiniz.

- **Farklı notalarda seslerin frekanslarını araştırarak bir müzik parçasını çaldıracak kodu yazınız.**

DENEY. 22 ANALOG/DİJİTAL ÇEVİRİCİ (ANALOG-TO-DIGITAL CONVERTER, ADC) DENEYİ

AMAÇ

1) Analog bilginin Dijital'a (A/D) dönüştürülmesinin öğrenilmesi,
2) EasyPIC v7 setinde on board A/D input girişlerinin kullanılması,

GEREKLİ MALZEME

Easy PIC7 Kartı

Giriş:

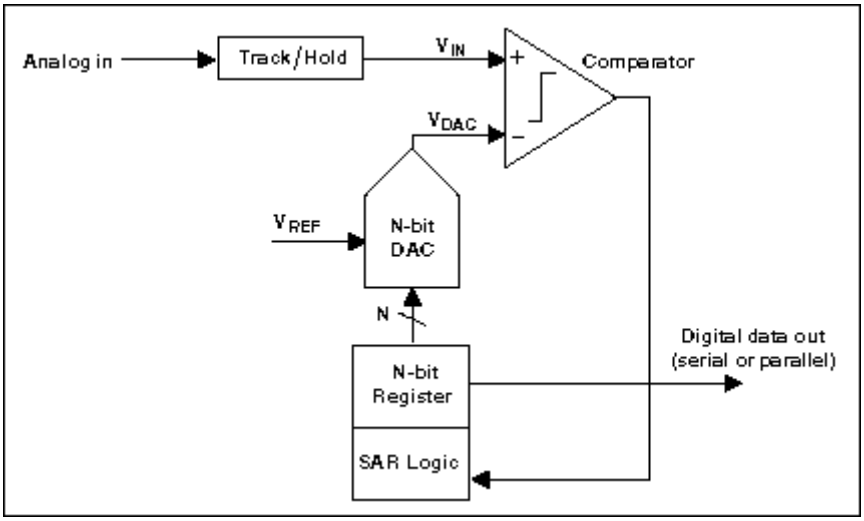
PIC18F45K22 entegresinin yongası üzerinde 10-bit Ardışık Yaklaşım Tipi (Successive Approximation Register, SAR) bir A/D Çevirici devresi vardır. Öncelikle bu tip A/D Çevirici'lerin çalışma prensiplerini inceleyelim.

Ardışık Yaklaşım Tipi ADC Devreleri

İki Eğimli (Dual-Wilkinson) A/D Çevirici devrelerinde analog değerin dijitala dönüşüm zamanı ($t_{dönüşüm}$) analog değer ile orantılıdır, yani örneklenmiş değerin DC karşılığı ne kadar büyükse $t_{dönüşüm}$ de o kadar büyüktür. Daha hızlı bir dönüşüm ARDIŞIK YAKLAŞIM TİPİ (Successive Approximation Register-SAR) denen ve Logaritmik bir prensibe dayanan A/D Çevirici ile elde edilebilir.

Bugün piyasada bulunan SAR A/D Çevirici devreleri orta-yüksek çözünürlük gerektiren uygulamalar için çözüm üretebilmektedir. Çeşitli Yarı-İletken üreticilerinin 8-18-bit çözünürlükte ve 5 Milyon Örnek/Saniye dönüşüm hızına kadar performans sağlayabilen A/D Çevirici entegreleri piyasada bulunabilmektedir.

Bir kimyacının ikilik düzende ağırlıklarla çalışan terazisi ile ölçüm yapması gibi, SAR A/D Çevirici bilinmeyen analog değeri hassas ikilik ağırlıklarla, (en büyük ağırlık değeri ile başlayarak) mukayese eder, analog değere sığan ağırlığın katsayısını 1, sığmayanınkini 0 olarak işaretler ve bir sonraki ağırlık mukayesesine geçer. Dönüşüm sonunda (End Of Conversion-EOC), SAR Kayıtçısı'nın içeriği analog değere karşılık gelen dijital değerdir. SAR A/D Çevirici'nin blok şeması Şekil 21.1'de gösterilmiştir.



Şekil.21.1 SAR A/D Çevirici blok şeması

Örnek: A/D Çevirici devremizin Analog Giriş Skalası 0-10 Volt olsun. A/D Çevirici'miz de 4-bit olsun;

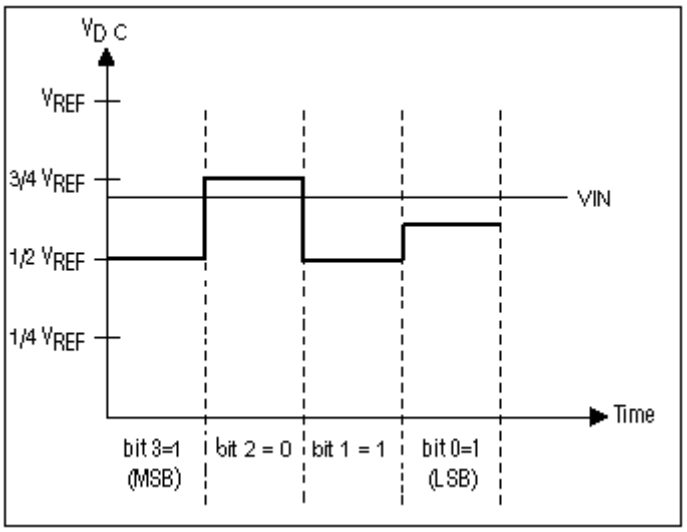
4-bit --→ 2^4=16 Kanal --→ Çözünürlüğümüz 10V/16=0,625 Volt= 625 Mili volt'tur.

İkilik Ağırlıklarımız ise:

$10V/2^1$=5 V, $10V/2^2$=2,5 V, $10V/2^3$=1,25 V, $10V/2^4$=0,625 V'tur.

Farz edelim V_{DC}=7 Volt analog değerimiz olsun. Bu değeri elimizdeki A/D Çevirici ile aşağıdaki yolla dönüştürebiliriz:

V_{DC}= 7 V= **1**X 5V + **0** X 2,5 V + **1** X 1,25 V + **1**X 0,625 V= 6,875 Volt.



Şekil.21.2 SAR İşlemi (4-bit A/D Çevirici örneği için)

a) Örnekte gördüğümüz gibi V_{DC} 'yi önce V_{Ref}/2= 5 Volt ile mukayese ediyoruz, V_{IN} > 5 V olduğu için çarpan **(1)** olarak kayıtçıya giriliyor,
b) Daha sonra ikinci ağırlık olan 2,5 Voltu ekliyoruz, fakat V_{IN}= 7V<5V+2,5V =7,5V, bu nedenle çarpan **(0)** olarak kayıtçıya giriliyor, ve 2,5 V ağırlık çıkarılıyor,
c) Mukayese voltajımız tekrar 5 V, bu ağırlığa 3. ağırlık olan 1,25 V ekliyoruz,V_{IN}= 7V>5V+1,25V =6,25V, bu nedenle çarpan **(1)** olarak kayıtçıya giriliyor,
d) Son olarak, son ağırlığımız olan 0,625 V ekliyoruz ve mukayese ediyoruz; V_{IN}=7V>5V+1,25V+0,625V =6,875V, bu nedenle çarpan **(1)** olarak kayıtçıya giriliyor,

Böylece 7 V AnalogGirişimiz $(1011)_2$= 11 numaralı kanala oturmuş oluyor.

Gördük ki 2^4=16 kanal SAR A/D Çevirici'de **4** mukayese yapılıyor, genelleme yaparsak 2^m kanal SAR A/D Çevirici'de de **m** mukayese yapılır.

Örnek:
M=10 ise, 10 mukayese ile 2^{10}=1024 kanal A/D Çevirici kullanmış oluruz, her mukayese için geçen süre 0,5 μ saniye ise $t_{dönüşüm}$= 5 μ saniye olacaktır.

Özetle;

Olumlu: SAR A/D Çevirici'ler, aynı kanal sayısına haiz Wilkinson A/D Çevirici'lerden daha hızlıdır. Ayrıca $t_{dönüşüm}$ Analog değere bağlı olmaksızın tüm değerler için aynıdır.

Olumsuz: Ayrı devrelerden oluşan SAR A/D Çevirici'lerde, "Diferansiyel Doğrusalsızlık" kötüdür.

MikroC pro for PIC ADC kütüphanesi

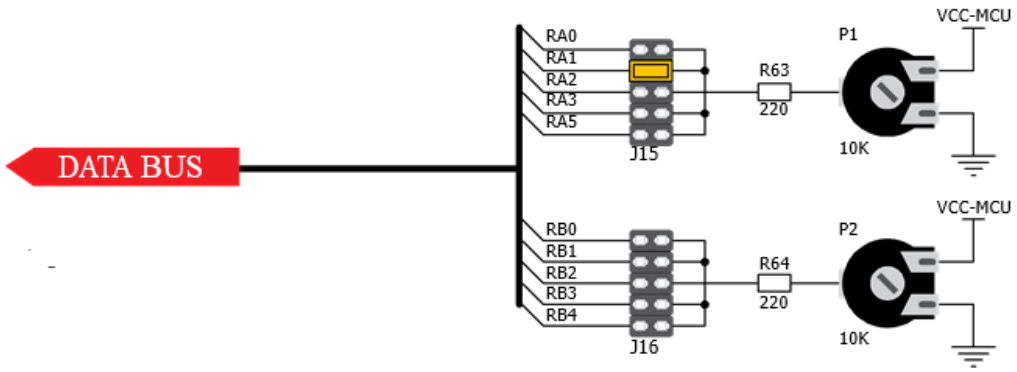
MikroC pro for PIC, kullanımı çok kolay bir ADC kütüphanesi sunar. Bu kütüphane sayesinde seçtiğiniz kanaldan verdiğiniz analog değerin dijital karşılığını kolaylıkla elde edebilirsiniz.

MikroC pro for PIC ADC kütüphanesinin fonksiyonları şunlardır.

- ADC_Init
- ADC_Get_Sample
- ADC_Read

ADC Deneyi

Easy PIC7 kartı üzerinde ADC uygulaması için iki adet trimpot bulunmaktadır. Bunlar PIC18F45K22'nin analog girişlerine 0-5 volt analog voltaj girişin sağlamak içindir. Aşağıda yapılan uygulamada Mikrodenetleyiciye giren analog bilgiye göre çıkış 10-bit binary olarak gösterilmiştir.



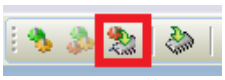
Şekil 21.3Easy PIC7 üzerindeki ADC giriş için bulunan ayarlı dirençlerin devre şeması

Örnek program:

```
//***********************
//Proje ismi       : DENEY. 22 ANALOG/DİJİTAL ÇEVİRİCİ
//(ANALOG-TO-DIGITAL CONVERTER, ADC) DENEYİ
//Amacı            : EasyPıc7 kartı üzerindeki ADC potu ile
//                   RA1 pinine verilen analog değer dijital
//                   bir değere çevirilir ve elde edilen
//                   dijital değer ledlerde gösterilir.
//Test konfigürasyonu   :
//------------------------
//MCU              : 18F45K22
//Osilatör         : 32Mhz
//j12 jumperları I/O konumunda olmalıdır.
//j17            => VCC
//SW3.3, 3.4 => ON, DİĞERLERİ => OFF
//J15       =>RA1
//***********************

unsigned int adc_rd;

void main()
{
 ANSELA = 0x02;        // RA1 pini analog olarak ayarlandı
 ANSELC = 0;           // PortC dijital olarak ayarlandı
 ANSELD = 0;           // PortD dijital olarak ayarlandı

 TRISA = 0x02;         // RA1 pini giriş olarak ayarlandı
 TRISC = 0x00;         // PortC çıkış olarak ayarlandı
 TRISD = 0x00;         // PortD çıkış olarak ayarlandı

 while (1)
 {
  adc_rd = ADC_Read(1);   // 1. kanaldan ADC değeri alınır
  LATC = adc_rd;          // PortC LED'lerinde adc_rd[7..0] gösterilir
  LATD = adc_rd>>8;       // PortD LED'lerinde adc_rd[9..8] gösterilir
 }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 22-ANALOG-DİJİTAL ÇEVİRİCİ" içerisindeki "Deney22.mcppi" projesini açınız.

4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. EasyPIC v7 kartı üzerindeki P1 potansiyometresini çevirerek PORTC ve PORTD bölümlerindeki ledlerdeki değişimleri gözlemleyiniz.

- **Ledlerin bazılarının sönük yandığını gözlemlemiş olmanız gerekir. Bunun sebebini ve önlemek için nelewr yapılması gerektiğini araştırınız.**

DENEY 23. EEPROM UYGULAMASI

AMAÇ

1) Seri EEPROM entegresi'nin nasıl çalıştığını göstermek,
2) I2C seri haberleşme protokolünün nasıl çalıştığını göstermek,

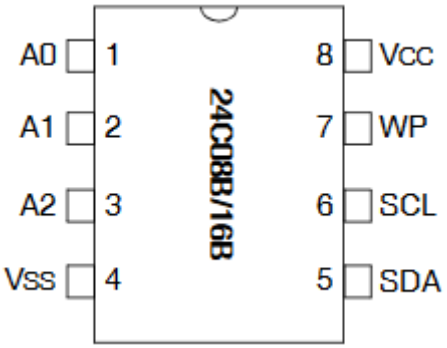
GEREKLİ MALZEME

EasyPIC7 Kartı,

Giriş

Bilindiği gibi Mikrodenetleyici tarafından saklanması gereken Veri (Değişkenler) Mikrodenetleyici güç kaynağı çalıştığı sürece RAM Bellek'de saklanır. Güç kesildiği anda RAM Bellek'deki tüm bilgiler kaybolur. Bu nedenle daha sonra tekrar kullanılması gereken verilerin sürekli veri saklayabilen (Permanent) Bellekler'de saklanması gerekir. Bu amaçla kullanılabilecek bellek entegre türlerinden biri ; Seri EEPROM'dur. EEPROM şu kelimelerin baş harflerinin kısaltılmışıdır: ***E****lectrically* ***E****rasable and* ***P****rogrammable* ***R****ead* ***O****nly* ***M****emory*. Seriden kasıt EEPROM'un mikrodenetleyici ile seri haberleşmesi; yani I2C, SPI, microwire gibi seri protokollerden birini kullanması anlamına gelir. Bu uygulamada 24Cxx ailesinden 24C08 entegresi kullanılmış ve iki hat üzerinden I2C protokolü ile Mikrodenetleyici ile haberleştirilmiştir. (Şekil 23.2)

24C08 8192 Bit= 8K bit Seri EEPROM'dur ve pin numaraları ve sinyalleri Şekil 16.1'de gösterilmiştir. Burada A0, A1 ve A2 adres hatları, **SCL** Seri Saat Sinyali, **SDL** ise Seri Veri Sinyalidir. WP Yazma Koruması (Write Protect) olup +5 Volt'a bağlanırsa tüm veriyi koruma altına alır, okuma/yazma yapılamaz.



Şekil 23.1 24C08 8KBit Seri EEPROM

EEPROM Mikrodenetleyici'ye SCL ve SDA hatları ile bağlanmıştır. SCL hattı saat hattı olup, SDA hattı üzerinden gönderilen veri'nin senkronizasyonu için kullanılmaktadır. SCL ve SDA hatları üzerinden gönderilen veri'nin frekansı 1MHz'e kadar çıkabilir. MikroC for PIC komutlarından I2C1_Rd ve I2C1_Wr Seri EEPROM'u okuma ve yazma işlemini çok basite indirgemektedir. EEPROM'u okuma ve yazma işlemi için entegrenin oturduğu adres gerekmektedir ki uygulamamızda bu adres $A0'dir. EEPROM adresi aşağıdaki gibi oluşturulmaktadır:

1	0	1	0	B2	B1	B0

Buradaki B2 B1 B0 değerleri Blok numaralarını seçmek için kullanılır. 24C08 eepromunda 4 adet 256x8 bitlik blok bulunmaktadır.

I2C HABERLEŞME PROTOKOLÜ

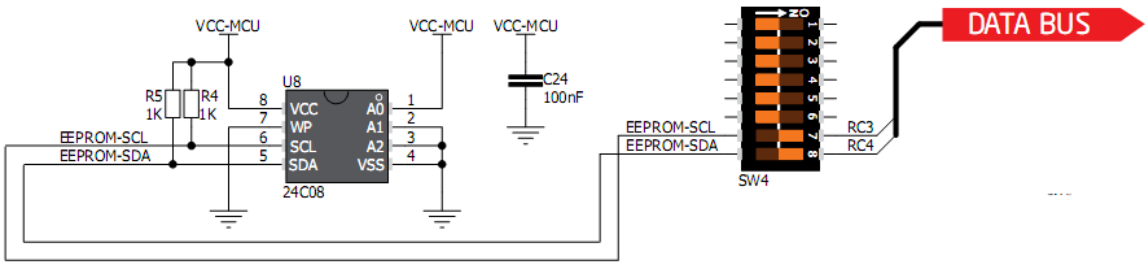
I2C seri haberleşme protokolü, SPI RS232 gibi haberleşme protokollerinden biridir. Uygulaması kolay, düşük bant genişliğine sahip, kısa-mesafe için ideal bir haberleşme protokolüdür. İçerisinde adresleme planı bulunduğu için, I2C ile birden fazla cihaz haberleşebilir.

I2C protokolünde SCL ve SDA isminde iki temel hat bulunur. SCL (SerialCLock) veri senkronizasyonu için kullanılan clock darbeleri hattı, SDA (SerialDAta) ise veri hattıdır.

I2C protokolünde veri alışverişi Startkomutuyla başlar ve Stopkomutuyla biter. Bu protokolde bilgi veya komut taşıyan veri alışverişi saat darbesinin lojik 0 (low) seviyelerinde gerçekleşir. Cihazlar arasında haberleşmenin başladığını veya tamamlandığını belirten Startve Stopdurum komutları ise sadece SCL hattının yani saat sinyalinin lojik 1(high) olduğu durumlarda gönderilir. SCL hattı lojik yüksek iken SDA hattında lojik1'den lojik0'a geçiş start komutu anlamına gelir ve veri transferinin başlayacağını bildirir. Benzer şekilde SCL hattı lojik yüksek iken SDA hattında lojik0'dan lojik1'e geçiş stop(bitir) komutu anlamına gelir ve haberleşen iki cihaz arasındaki haberleşmenin tamamlandığını bildirir.

I2C protokolünde 'Master' denilen ana bir kontrol birimi cihazı ve 'Slave' denilen cihazlar vardır. 'Master' cihazlar için genelde mikrodenetleyiciler kullanılır. Bu protokole göre, I2C veri yoluna bağlı her bir 'Slave' durumundaki cihazın, en fazla 7 bitten oluşan, kendisine ait bir adresi vardır. Bu cihaz bir LCD, EEPROM, mikrodenetleyici veya herhangi bir sayısal sistem olabilir. Veri yolundaki haberleşme iki yönlü de olabilir. SDA hattında eğer bu adres bilgisi master tarafından gönderilmişse, o adrese ait slave tarafından SDA hattına ACK yani 'alındı' anlamına gelen bir cevap gönderilir. Bu aşamadan sonra 'Slave' birimi kullanılmaya hazırdır.

EEPROM DENEYİ



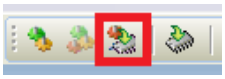
Şekil 23.2Easy PIC7 üzerindeki EEPROM devresi

Örnek program:

```
//***********************
//Proje ismi        : Deney-23 EEPROM UYGULAMASI
//Amacı             : EasyPıc7 kartı üzerindeki EEPROM
//                  entegresine veri yazıp, yazılan
//                  veriyi okuyarak, okunan verinin
//                  PortB LED'lerinde gözlemlenmesi.
//Test konfigürasyonu    :
//---------------------------------
//MCU               : 18F45K22
//Osilatör          : 32Mhz
//J12 jumperları I/O konumunda olmalıdır.
//SW3.2             => ON
//SW4.7, 4.8        => ON
//Library Manager Sekmesinden I2C aktif edilmelidir.
//***********************


void main(){
  ANSELB = 0;                    // PORTB dijital olarak ayarlandı
  ANSELC = 0;                    // PORTC dijital olarak ayarlandı

  TRISB = 0;                     // PORTB çıkış olarak ayarlandı
  LATB = 0;

  I2C1_Init(100000);              // I2C haberleşmeyi hazırla
  I2C1_Start();                   // I2C haberleşmeyi başlat
  I2C1_Wr(0xA0);                  // I2C ile yaz
  I2C1_Wr(2);                     // EEPROM adresine I2C ile yaz
  I2C1_Wr(0xAA);                  // veriyi I2C ile yaz
  I2C1_Stop();                    // I2C haberleşmeyi durdur

  Delay_100ms();

  I2C1_Start();                   // I2C haberleşmeyi başlat
  I2C1_Wr(0xA0);                  // I2C ile yaz
  I2C1_Wr(2);                     // EEPROM adresine I2C ile yaz
  I2C1_Repeated_Start();
  I2C1_Wr(0xA1);
  LATB = I2C1_Rd(0);              // okunan veriyi PORTB LED'lerinde gözlemle
  I2C1_Stop();                    // I2C haberleşmeyi durdur
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.

3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 23-EEPROM UYGULAMASI" içerisindeki "Deney23.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. EasyPIC v7 kartı üzerindeki PORTB bölümünden ledlerin aldığı değeri gözlemleyiniz.

- **PORTB'deki ledleri farklı şekillderde yakmak için kodlarda gerekli değişikliği yapınız.**
- **EEPROMUN fark adreslerine farklı değerler yükleyerek 1'er saniye aralıklarla okuyup PORTB'de gösteriniz.**

DENEY 24. USB HABERLEŞMESİ UYGULAMASI

DENEYİN AMACI:

1) USB haberleşme protokolünün yapısını incelenmesi
2) Bu protokol üzerinden haberleşme uygulamasının gerçekleştirilmesi

GEREKLİMALZEME:

EasyPIC7 Kartı,

GİRİŞ:

USB nedir?

UniversalSerial Bus (USB) Evrensel seri yolu, bilgisayar ve telekominikasyon endüstrisinde geliştirilmiş, iletişim standartlarında yeni bir bağlantı şeklidir. Amacı, geleneksel seri ve paralel portların yerini almak ve seri iletişimi evrenselleştirmektir.

USB, bağlandığı alete güç verir. Son sürümü USB 80Gbps'dir. 80 GByte/sn'lik aktarım hızı vardır. USB, PC' leri birbirine hızlı bağlamak için kullanılmaz, ama, küçük çapta şebeke ortamı kurmak için bazı ürünler çıkarılmıştır. USB, ana bilgisayar ile çevre birimleri arasında güçlü, bilgisayar çalışırken takıp çıkartabileceğiniz, "gerçek plug-and-play" arayüzü sağlar.

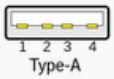
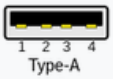
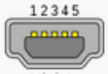
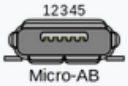
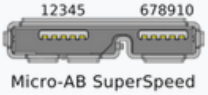
USB ilkel çevre bağlantı teknolojisini geliştirmek için icat edilmiştir. Avantajları şunlardır:

*Bilgisayarı kapatmanız gerekmez (Önemli bir avantaj)
*Kasayı açmanız gerekmez
*Kart takma yok
*Çakışma yok
*Kilitlenme yok
*Sürücü yüklemeye gerek kalmaz (bazılarında bir seferlik yükleme yapılır)
*Bazı aygıtlar voltajını buradan alır
*Performansı 80 GByte/sn'ye kadar ulaşabilir.
*İhtiyaç duyuldukça sürücüler, otomatik olarak yüklenir veya bırakılır.

USB Çeşitleri, Hızları ve kablo bağlantı özellikleri:

Günümüzde USB 80GB/sn hızlara ulaşabilmektedir.

Bir USB kablosu, dört iletkenden oluşur: iki adet veri yolu (kablo) gücü için ve iki adet de farklı sinyal çifti için. Pin tanımlamaları aşağıdaki şekildedir:

Standard	USB 1.0 1996	USB 1.1 1998	USB 2.0 2001	USB 2.0 Revised	USB 3.0 2008	USB 3.1 2013	USB 3.2 2017	USB4 2019	USB4 V2 2022
Maximum transfer rate	1.5 Mbit/s	12 Mbps	480 Mbps		5 Gbps	10 Gbps	20 Gbps	40 Gbps	80 Gbps
Type A connector	 Type-A 1.0 - 1.1		 Type-A 2.0		 Type-A SuperSpeed		Deprecated		
Type B connector	 Type-B				 Type-B SuperSpeed		Deprecated		
Mini-A connector	—	 Mini-A			Deprecated				
Mini-B connector	—	 Mini-B			Deprecated				
Mini-AB connector	—		 Mini-AB		Deprecated				
Micro-A connector	—		 Micro-A		 Micro-A SuperSpeed		Deprecated		
Micro-B connector	—		 Micro-B		 Micro-B SuperSpeed		Deprecated		
Micro-AB connector	—		 Micro-AB		 Micro-AB SuperSpeed		Deprecated		
Type C connector	Backwards compatibility only		 (Enlarged to show detail)						

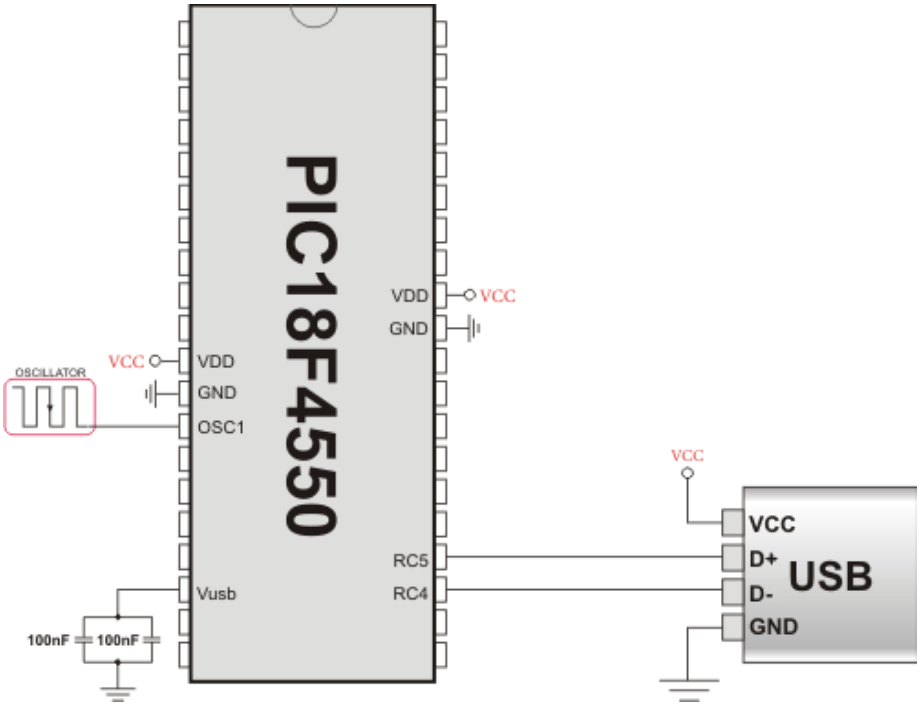
Genelde veri iletişimi için yereli olan USB pinleri aşağıda gösterilmiştir;

Pin 1. VSUB: Veri yolu gücü, kaynakta +5V Pin

Pin 2 ve 3. D- ve D+: Farklı sinyal çifti

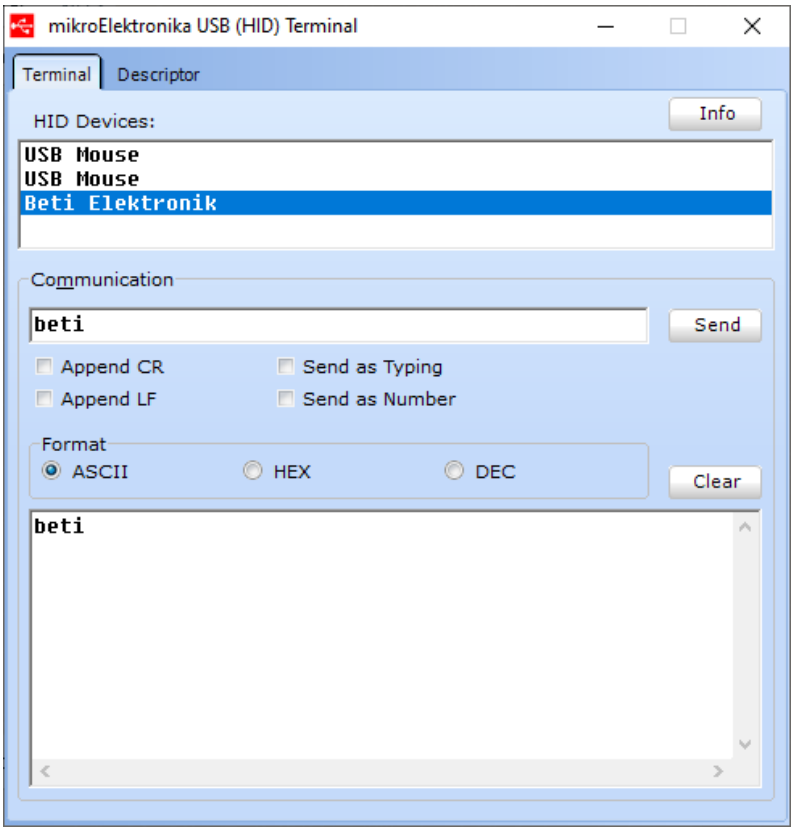
Pin 4. GND: Toprak

USB kablolarıda, tam hızlı kablolar ve düşük hızlı kablolar olmak üzere ikiye ayrılır. Tam hızlı kablo, kıvrımlı sinyal çifti ve dört kabloyu sarmak için çevresel koruyucu gerektirir. Tam hızlı USB kablosu için gerekli asgari marka bilgisi, "USB SHIELDED…" ibaresini, USB spesifikasyonları nedeniyle bulundurmalıdır. Düşük hızlı kablo ise koruma, kıvrımlı sinyal çifti iletkenleri ve özel marka bilgisi gerektirmez.



Şekil 24.1 Uygulamada kullanılan devre şeması.

Bu uygulamada 18F4550 mikrodenetleyicisi MikroC pro for PIC derleyicisi ile birlikte sunulan USB HID (Human Interface Device- İnsan Arayüz Cihazı) kütüphanesi ile kullanılacaktır. Bu şekilde ayarlanan bir USB cihazı herhangi bir sürücü yüklemesine gerek duymaksızın bilgisayara bağlanabilir ve bilgisayar ile haberleşebilir.



Şekil 24.2 : HID Terminal uygulaması ekran görüntüsü.

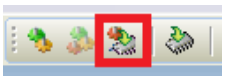
MikroC pro For PIC derleyicisinin "Tools" sekmesi altından "HID Terminal" uygulaması açılarak PIC18F4550 mikrodenetleyicisi ile haberleşme uygulaması test edilebilir. Tek seferde 64 byte uzunluğunda veri paketleri ile iletişim sağlanmaktadır. Uygulamada PC'den 18F4550'ye HID terminalden gönderilen karakterler 18F4550'den PC'ye geri gönderilmektedir.

Örnek program:

```
//************************
//Proje ismi       : DENEY24-USB HABERLEŞMESİ UYGULAMASI
//Amacı            : Kişisel bilgisayar ve EasyPıc7 arasında USB bağlantısı kurmak.
//Test konfigürasyonu        :
//---------------------------------
//MCU              : 18F4550
//Osilatör         : 48Mhz
//
// j12      =>  3 adet jumper USB konumunda olmalı.
//
// Library Manager Sekmesinden  USB aktif edilmeli.
//
// EasyPIC v7 karı üzerindeki USB Comm modülü kullanılarak haberleştirme sağlanmalıdır.
//************************

unsigned char okumaTamponu[64] absolute 0x500;   // Tamponlar USB RAM bölgesinde
//tanımlanmalıdır.
unsigned char yazmaTamponu[64] absolute 0x540;
char cnt;
```

```
char kk;

void interrupt(){
  USB_Interrupt_Proc();              // USB iletişim işlemleri Kesme altında yapılmaktadır.
}

void main()
{
 ADCON1 = 0x0F;                      // Bütün ortlar dijital olarak ayarlanıyor.
 CMCON  = 7;                         // Komparatörler kapatılıyor.
 HID_Enable(&okumaTamponu,&yazmaTamponu);    // HID iletişim aktif ediliyor.

 while(1)
 {
  if(HID_Read())                     // Eğer veri geldiyse
  {
   for(cnt=0;cnt<64;cnt++)           // okuma tamponundaki verileri yazma tamponuna aktar.
   {
    yazmaTamponu[cnt]=okumaTamponu[cnt];
   }
   HID_Write(&yazmaTamponu,64);      // PC'ye yazma tamponundaki verileri gönder.
  }
 }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 24-USB HABERLEŞMESİ UYGULAMASI" içerisindeki "Deney24.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F4550 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. HIF Terminal uygulamasından 18F4550'ye veri göndererek gelen cevabı gözlemleyiniz.

- **18F4550 mikrodenetleyicisine ait PORTB'deki butonlarun durumunu HID termianlde gösterecek kodu yazınız.**

DENEY 25. ULTRASONİK MESAFE SENSÖRÜ UYGULAMASI

AMAÇ

1) Ultrasonik mesafe sensörünün çalışma prensibini öğretmek,
2) PIC18F45K22 Mikrodenetleyici ile Ultrasonik mesafe sensörü uygulamasını tanıtmak,

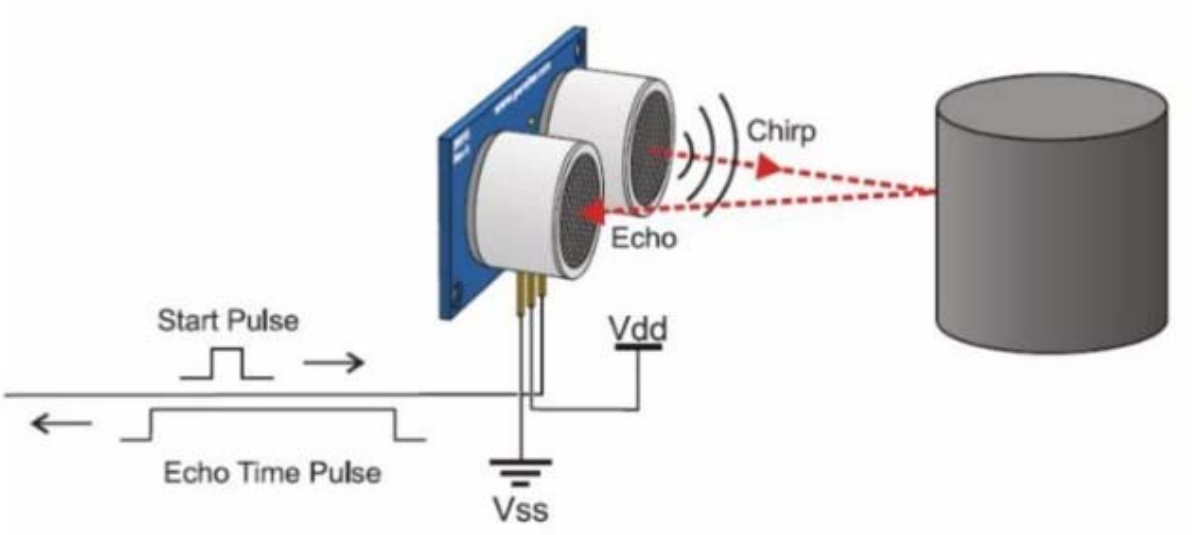
GEREKLİ MALZEME

1) EasyPIC v7 Kartı,
2) BETİ Mikrodenetleyici Uygulama ve Geliştirme Seti

Giriş

Uzaklık ölçümünde birçok teknik, prensip ve sensör kullanılmasıyla beraber en yaygın kullanımlardan biri de ultrasonik mesafe sensörleridir. Ultrasonik kelimesi ses üstü anlamına gelir ve insan kulağının duyabileceğinden daha yüksek frekanslardaki seslerin kullanılması prensibine dayanır. Mesafe ölçümünde yüksek frekanslı ses dalgalarının kullanılma sebebi bu dalgaların havada oldukça düzgün ve doğrusal yayılmaları, çarptıkları sert yüzeylerden kolay ve düzgün yansımalarıdır.

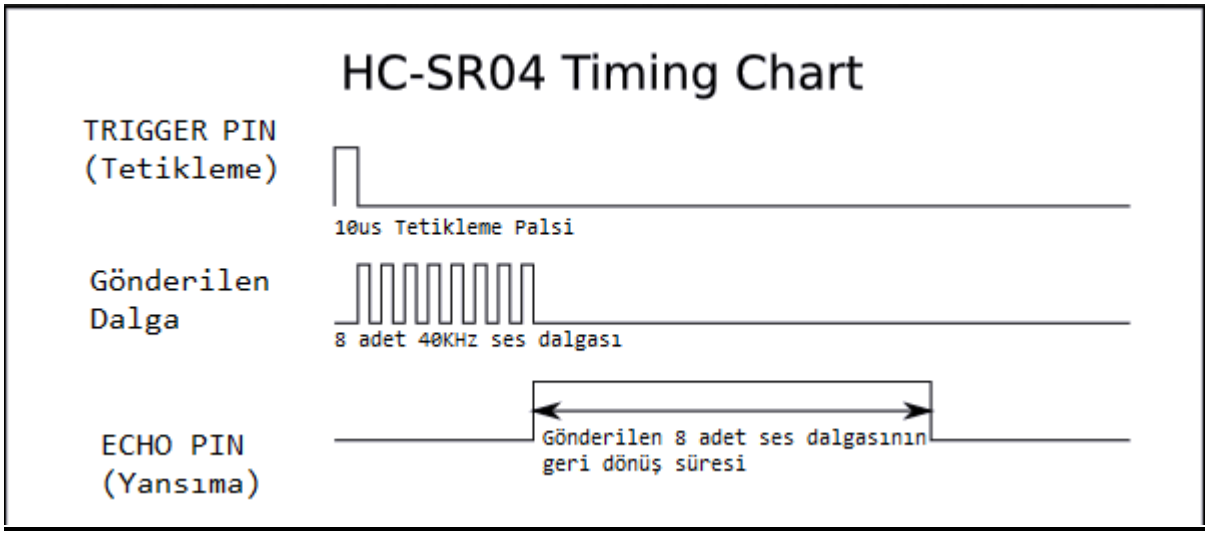
Mesafe Ölçme Prensibi



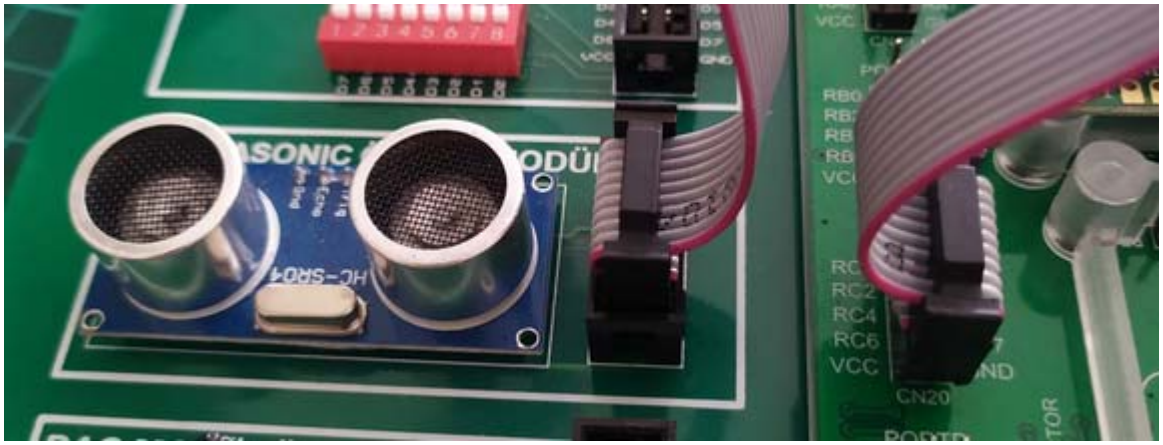
Ultrasonik sensörlerin algılama mesafesi kullanılan ortam koşuluna göre 30 metreye kadar varabilir. Ultrasonik sensörlerde iki adet transducer bulunur. Bunlardan biri ultrasonik ses dalgasını üreten ultrasonik hoparlör, diğeri de yansıyan ultrasonik ses dalgasını alan ultrasonik mikrofondur. Ultrasonik ses dalgasının havada aldığı yol ise ultrasonik ses dalgasının sensörden çıkma anı ile mikrofonla alınması arasındaki zaman farkının sesin havadaki hızı ile çarpımıyla bulunur. Ses dalgası cisme çarpıp geri döndüğü için, ses dalgasının havada aldığı yol, cisim ile sensör arasındaki mesafenin iki katıdır. Zaman ve hız çarpımımızı ikiye böldüğümüz zaman mesafemizi ölçmüş olacağız.

HC-SR04 Ultrasonic Mesafe Sensörü Protokolü

Kullandığımız HC-SR04 Ultrasonic Mesafe Sensörünün Vcc-Trig-Echo-Gnd isimli 4 adet pini bulunmaktadır. Sensöre veri girişini tetikleme pini olan Trig isimli pinden yapacağız. Sensörden gelen veriyi ise Echo isimli pinden alacağız. Öncelikle mikrodenetleyicimizin sensörün TRIG pinine bağlı olan pinini çıkış pini(output) olarak ayarlayıp 10 mikrosaniyelik bir tetik sinyali göndererek sensör üzerindeki devreyi başlatıyoruz.



Sensör üzerindeki devre sayesinde ultrasonic hoparlörden 40KHz frekansta ultrasonic bir ses dalgası yollanıyor. Tetik sinyali yollandıktan sonra sensöre bağlı olan Echo pinini okuyup bu pininin ne kadar süre HIGH seviyesinde kaldığını mikrodenetleyicimizin TIMER1 modülü sayesinde hesaplıyoruz. Timer1 modülü 1us'de bir saymak için hazırlanmıştır.



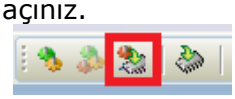
Ultrasonik Ölçüm Modülü ile EasyPIC v7 kartı arasındaki flat kablo bağlantısı yukarıdaki fotoğrafta görülebilir.

ÖrnekProgram

```
//***********************
// Proje ismi       :ultrasonic
//Amacı             :Ultrasonic sensörü başlatmak için
//                  tetik sinyali gönderip, sinyal pininin
//                  HIGH seviyesinde kaldığı süreyi
//                  ölçüp mesafeye ölçümü yapmak
//Test konfigürasyonu          :
//------------------------------
//MCU             : 18F45K22
//Osilatör        : 32Mhz
//
//SW 4.6 =>ON
//J12 jumperları I/O konumunda olmalıdır.
//Library Manager sekmesinden Conversions ve Lcd kütüphaneleri aktif edilmelidir.
//***********************

#define ECHO PORTC.B1
#define TRIGGER LATC.B0

// Lcd modül Bağlantıları
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// Lcd modül Bağlantıları bitişi


unsigned int sayac_TMR1=0;
char txt[7];
unsigned int toplamSure=0,mesafe_mm=0,Sure_us=0;
//---------------------------
void InitTimer1()
{
 T1CON            = 0x30; //32MHz clock, 1:8 prescaler.
 TMR1H = 0;
 TMR1L = 0;
}
//------------------------------
int MesafeOlc()
{
```

```
 long toplam=0,temp=0;
 char i,sayac;
 for(i=0;i<64;i++)  // 64 adet ölçüm yapılıp ortalaması alınacak
 {
  TRIGGER=1;      // tetikleme palsi
  Delay_us(10);   // (10us)
  TRIGGER=0;     // gönder
  while(!ECHO);           // ECHO "H" olana kadar bekle
  T1CON.TMR1ON=1;  // Timer1'i başlat
  while(ECHO);           // ECHO "L" olana kadar bekle
  T1CON.TMR1ON=0;  // Timer1'i durdur.
  sayac_TMR1=(TMR1H<<8)+TMR1L;         // Timer1 değerini int değişkene aktar
  toplam+=((long)sayac_TMR1*34)/200;    // mesafe hesapla ve topplam değişkeninde topla
  TMR1H=0; TMR1L=0;                     // Timer1'i sıfırla
  sayac_TMR1=0;                        // Timer'i aktardığımız us sayacını sıfırla
  Delay_ms(5);                         // 5 ms bekle
 }
 return (toplam>>6);                   // ortalamayı fonksiyon dışına taşı
}
//------------

void Kurulum()
{
 ANSELB=0;
 ANSELC=0;

 TRISC.B0=0;
 TRISC.B1=1;

 InitTimer1();
 Lcd_Init();
 Lcd_Cmd(_LCD_CURSOR_OFF);
}
//--------------------
void main()
{
 Kurulum();
 Lcd_Out(1,3,"Mesafe Olcer");
 Lcd_Out(2,1,"Beti Elektronik");
 Delay_ms(2000);
 Lcd_Cmd(_LCD_CLEAR);
 Lcd_Out(1,3,"Mesafe Olcer");
 while(1)
 {
  mesafe_mm=MesafeOlc();
  WordToStr(mesafe_mm,txt);

  Lcd_Out(2,1,txt);
  Lcd_Out_CP(" mm");
 }
}
```

Yöntem

1. EasyPIC v7 kartını USB kablo ile bilgisayara bağlayınız.
2. Bilgisayarda mikroC Pro for PIC derleyicisini çalıştırınız.
3. "Project" sekmesi altından "Open Project" seçeneğini kullanarak, "Deneyler" klasöründe bulunan "DENEY 25-_ULTRASONİK MESAFE SENSÖRÜ UYGULAMASI" içerisindeki "Deney25.mcppi" projesini açınız.
4. "Build" sekmesinden "Build +Program"  seçeneği ile kodları derleyerek EasyPIC v7 kartı üzerindeki PIC18F45K22 mikrodenetleyicisine yükleyiniz.
5. Bu aşamada mikroProg Suite for PIC programı açılacak ve derlenen kodlar mikrodenetleyici'ye transfer edilecektir. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.
6. Beti Mikrodenetleyici Uygulama ve Geliştirme Seti üzerindeki Ultrasnik Ölçüm modülüne elinize yaklaştırıp uzaklaştırarak LCD'de mesafe ölçümünü gözlemleyiniz.

- **Ölçülen mesafe azaldıkça buzzer ın ötme aralığının sıklaştığı, mesafe arttıkça buzzer ötme aralıklarının genişlediği bir kod yazınız.**

ELEKTROVADİ®

beti®

BETİ Elektronik San. ve Tic Ltd.Şti.

Cevizlidere Mah. 1244 Sokak No:8/B
Çankaya Ankara
Tel: 0312 2221800 Fax: 0312 2221808
info@beti.com.tr