# DENEY 1-A: PIC 16F877 İLE MİKRODENETLEYİCİDE VERİ ÇIKIŞI UYGULAMASI

**AMAÇ:**

1- Mikrodenetleyici'nin portlarını öğrenmek
2- Mikrodenetleyici ile veri alışverişini öğrenmek

**GEREKLİ MALZEMELER:**

1- EasyPIC 7 uygulama seti

**Giriş:**

Giriş/Çıkış'lar (G/Ç veya I/O), Mİkrodenetleyici'nin dış dünya ile ilişkisini sağlayan, girdi ve çıktı şeklinde ayarlanabilen bağlantı pinleridir. G/Ç'lar çoğunlukla Mikrodenetleyici'nin iletişim kurmasına, bir şeyleri kontrol etmesine veya bilgi okumasına izin verir. PIC16F877A Mikrodenetleyici'sinde 33 adet G/Ç pini mevcuttur. Bu pinlerin adları RA0-RA5, RB0-RB7, RC0-RC7, RD0-RD7, RE0-RE2 şeklindedir. Bu 33 pinin her biri giriş ya da çıkış olarak kullanılabildiğinden "PIC16F877A Mikrodenetleyici'si 33 adet G/Ç'dan oluşur" denilir.

PIC içerisinde bulunan ve TRIS adı verilen kaydedici sayesinde portların durumu giriş veya çıkış olarak tanımlanabilir.

**TRIS Kaydedicisi:**

```
set_tris_b(0x00);    // B portunun tüm pinlerini çıkış olarak ayarlar

set_tris_a(0x0F);    // A portunun ilk 4 bitini (RA0, RA1, RA2, RA3) giriş olarak
                     // ayarlar
set_tris_c(0xFF);    // C portunun tüm pinlerini giriş olarak ayarlar
```

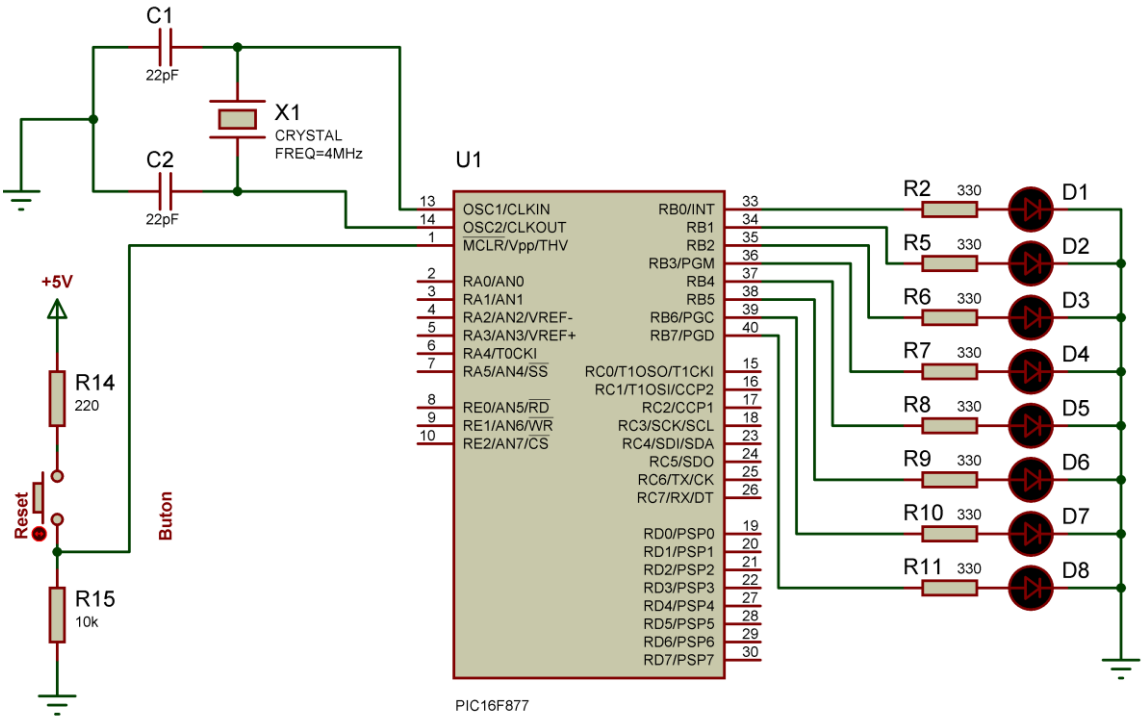
Güç kaynağından çıkış portuna doğru akan akıma SINK akımı, çıkış portundan toprağa doğru akan akıma ise SOURCE akımı denir. Bu 20-25 mA'lik akımlar bir LED'i direk sürebilirler. Röle, optik izolatör, triyak ve yükselteç devreleri bu küçük akımlar ile sürülebilir. Böylece bu elemanlarla daha yüksek akım ve gerilimler kontrol edilebilir.

**Uygulama Programı 1:**

```
#include <16f877.h>                 // Kullanılacak denetleyicinin başlık
                                    // dosyası tanıtılıyor
#fuses HS, NOWDT, NOPROTECT         // Denetleyici konfigürasyon ayarları

#use delay(clock=4000000)           // Gecikme fonksiyonu için kullanılan
                                    // osilatör frekansı belirtiliyor
#use fast_io(b)                     // Port yönlendirme komutları B portu için
                                    // geçerli
```

```
                    /*** ANA PROGRAM ***/
void main()
        {
          setup_adc_ports(NO_ANALOGS);
          setup_adc(ADC_OFF);
          setup_psp(PSP_DISABLED);
          setup_spi(SPI_SS_DISABLED);
          setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
          setup_timer_1(T1_DISABLED);
          setup_timer_2(T2_DISABLED,0,1);

          set_tris_b(0x00);        // B portunun tüm pinlerini çıkış olarak ayarla
          output_b(0x00);          // B portunun çıkışlarını temizle

         while(1)
          {
          output_b(0x01);          // RB0'a bağlı LED'i yak
          delay_ms(500);           // 500 ms gecikme yap
          output_b(0x05);          // RB0 ve RB2'ye bağlı LED'leri yak
          delay_ms(500);           // 500 ms gecikme yap
          output_b(0x15);          // RB0, RB2 ve RB4'e bağlı LED'leri yak
          delay_ms(500);           // 500 ms gecikme yap
          output_b(0x55);          // RB0, RB2, RB4 ve RB6'ya bağlı LED'leri yak
          }
        }
```






**Şekil1.1** PIC16F877 RB portu çıkışları

Yukarıdaki program kodu CCS-C ile yazılıp derlendikten sonra EasyPIC7 uygulama setine yüklendiğinde; RB0, RB2, RB4 ve RB6 pinlerine bağlı LED'lerin sırasıyla yandığı ve sürekli olarak bu işlemi sonsuz döngüde tekrar ettiği görülmektedir.

**Uygulama Programı 2:**

```
#include <16f877.h>                  // Kullanılacak denetleyicinin başlık dosyası
                                     // tanıtılıyor
#fuses HS, NOWDT, NOPROTECT // Denetleyici konfigürasyon ayarları

#use delay(clock=4000000)            // Gecikme fonksiyonu için kullanılan
                                     // osilatör frekansı belirtiliyor
#use fast_io(b)                      // Port yönlendirme komutları B portu için geçerli

int i;                               // integer "i" değişkeni tanımlaması

                                     /*** ANA PROGRAM ***/

void main()
{
 setup_adc_ports(NO_ANALOGS);
 setup_adc(ADC_OFF);
 setup_psp(PSP_DISABLED);
 setup_spi(SPI_SS_DISABLED);
 setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
 setup_timer_1(T1_DISABLED);
 setup_timer_2(T2_DISABLED,0,1);

  set_tris_b(0x00);  // B portunun tüm pinlerini çıkış olarak ayarla
  output_b(0x00);   // B portunun çıkışlarını temizle

        while(1)
        {
                for(i=0;i<10;i++)
                        {
                          output_b(0xFF);          // PortB'ye bağlı LED'leri yak
                          delay_ms(200);           // Gecikme
                          output_b(0x00);          // PortB'ye bağlı LED'leri söndür
                          delay_ms(200);           // Gecikme
                        }
                output_b(0xF0);          // PortB'ye bağlı LED'leri yak
                delay_ms(500);           // Gecikme
                output_b(0x0F);          // PortB'ye bağlı LED'leri yak
                delay_ms(500);           // Gecikme
        }
}
```

Yukarıdaki program kodu CCS-C ile yazılıp derlendikten sonra EasyPIC7 uygulama setine yüklendiğinde; önce PortB'ye bağlı LED'lerin tamamının 10 defa 200 ms aralıklarla yanıp söndüğü, daha sonra ilk dört bitin yandığı, 500 ms sonra ise diğer dört bitin yandığı ve programın tekrar sonsuz döngüde baştan başladığı görülmektedir.

**Yöntem:**

1. EasyPIC 7 üzerinde J17 jumper'ını VCC konumuna ve SW3.2 Switch'ini ON olarak ayarlayınız.
2. EasyPIC 7 kartını, USB kablo ile bilgisayarınıza bağlayın.
3. Yazdığınız programa ait oluşturduğunuz "HEX" dosyasını "mikroProg Suite for PIC" programınına yükleyiniz.
4. mikroProg Suite for PIC programının WRITE komutunu kullanarak bu dosyayı EasyPIC7 anakartı üzerindeki Mikrodenetleyici'ye transfer ediniz. Transfer ve doğrulama (Verify) işlemi biter bitmez program işlemeye başlayacaktır.

**Ödev:**

1. B portuna bağlı 8 tane LED'in önce 500 ms aralıklarla ilk 4 tanesini sonra son 4 tanesini 15 defa yakıp söndüren, ardından B portuna bağlı 8 tane LED'lerin 1000 ms aralıklarla önce 0x55 sonra 0xAA yüklenerek 5 defa yakıp söndüren programı yazınız.

2. Programı sonsuz döngü kullanarak ve sonsuz döngü kullanmadan ayrı ayrı deneyiniz.

3. B portuna bağlı 8 tane LED'in 500 ms aralıklarla ilk olarak "0X01" sonrasında "0X03", "0X07", "0X0F", "0X1F", "0X3F", "0X7F", "0XFF",şeklinde 15 defa yakıp söndüren, ardından aynı işlemi yanma düzeni ters olacak şekilde yapabilen programı yazınız.