

# 9

## EEPROM BELLEĐE YAZMA ve OKUMA

- WRITE Komutu
  - READ Komutu
  - EEPROM Komutu
  - DATA Komutu
  - I2CREAD Komutu
  - I2CWRITE Komutu
- 
- 

### 9.1. WRITE

**Dizilimi:** *WRITE Adres, Deđer*

Chip üzeri EEPROM belleđi bulunan PIC'lerde belirlenen *adrese* bir *deđer* yazar. Bu komut sadece chip üzeri EEPROM belleđi bulunan PIC16F84, PIC16F628A, PIC16F877A gibi mikrodeneleyicilerde kullanılabilir.

WRITE komutu programın çalışması anında EEPROM alanına bir deđer yazmak isteniyorsa kullanılır. Eđer programlama anında deđer yazmak isteniyorsa DATA veya EEPROM komutları kullanılmalıdır.

Her bir WRITE komutunun bir mikrodeneleyicide çalışması için gereken süre 10 mS dir.

**Örnek:**

```
WRITE 5, B0 'EEPROM'un 5 adresine B0 içindeki deđerini  
yazar.
```

### 9.2. READ

**Dizilimi:** *READ Adres, Deđerşken*

Chip üzeri EEPROM'un belirtilen adresini okur ve sonucu deđerşken içerisinde saklar. Bu komut sadece chip üzeri EEPROM belleđi bulunan PIC16F84, PIC16F628A, PIC16F877A gibi mikrodeneleyicilerde kullanılabilir.

**Örnek:**

```
READ 5, B2 'EEPROM'un 5 adresindeki veriyi B2  
deđerşkeni içerisinde saklar.
```

**UYGULAMA 9.1:** *READ ve WRITE komutlarıyla word tipi verilerin chip üzeri EEPROM alanına yazdırılmasına örnek program.*Deneme: PIC PROG/DEKA 87x Deneme : PIC DEKA V1.1 

Programda EEPROM alanına yazılan word tipi verileri elde etmek için her bir adres numarasına 1000 eklenmiştir ve elde edilen veri EEPROM hücresine yazılmıştır. Word tipi veriler 2 byte'lık bir alan kaplayacağından WRITE ve READ komutlarıyla önce HIGHBYTE daha sonra LOWBYTE işlenmiştir.

Şekil 9.1'deki devreye göre RA0 butonuna basılınca PIC16F877A'nın dahili EEPROM alanındaki 0, 2, 4, 6, 8, 10, 12. adreslerine 1000, 1002, 1004, 1006, 1008, 1010, 1012 verilerini yazdırır. RA1 butonuna basılınca okuma moduna geçilir. Okuma modunda adreslerdeki verileri tek tek okumak için RA3 butonuna basılır. Her basışta bir sonraki adresteki veri okunarak LCD'de yazılır. RA2 butonuna basılınca EEPROM belleğe yazılmış olan veriler silinir.

```

'-----
'Programın adı: Uyg-9_1.bas
'Programın işlevi: WRITE komutu ile chip üzeri EEPROM alanına
' yazılan word tipi verileri READ komutu ile okuduktan sonra
' LCD'de gösterir.
'Tarih:28/07/2007   Uyarlama:1.00   Derleyici:PBP
'-----Tanımlamalar-----
DEFINE LCD_DREG  PORTB      ' RB0-->DB4 PortB.0
DEFINE LCD_DBIT  0         ' RB1-->DB5 PortB.1
                             ' RB2-->DB6 PortB.2
                             ' RB3-->DB7 PortB.3

DEFINE LCD_RSREG  PORTB
DEFINE LCD_RSBIT  4         ' RS   PortB.4
DEFINE LCD_EREGL  PORTB
DEFINE LCD_EBIT   5         ' E    PortB.5
DEFINE LCD_BITS   4         ' 4 bit ile iletişim.
DEFINE LCD_LINES  2         ' LCD satır sayısı.
B1  VAR  BYTE
W2  VAR  WORD
ADCON1 = 7
Pause 100                  ' LCD'nin açılmasını bekle.

Ana_prog:
  IF portA.0=1 then yaz    ' RA0'a basınca EEPROM'a yaz.
  IF portA.1=1 then oku   ' RA1'e basınca EEPROM'u oku.
  IF portA.2=1 then sil   ' RA3'e basınca EEPROM'a sil.
  goto Ana_prog
yaz:
  For B1 = 0 To 12 step 2  ' 2'şer adım ilerle, çünkü word
                          ' tipi veriler 2 byte yer kaplar.
    W2 = B1 + 1000        ' Adrese 1000 ekle.

```

```

    Write B1, W2.HIGHBYTE ' Word'ün high byte'ını yaz.
    Write B1+1, W2.LOWBYTE ' Sonraki adrese low byte'ı yaz.
Next B1
goto Ana_prog
oku:
For B1 = 0 To 12 step 2 ' Word tipi veri okumak için
\ 2'şer adım ilerle.

bekle:
IF portA.3 = 0 then bekle 'RA3'e basınca sonraki adresi oku
  Pause 100 ' Buton arkını söndür.
  Read B1, W2.HIGHBYTE ' High byte'ı oku.
  Read B1+1, W2.LOWBYTE ' Low byte'ı oku.
  LCDOUT $FE, 1, $C0, #W2 ' Veriyi LCD'de göster.
Next B1
goto Ana_prog
sil:
For B1 = 0 To 12 step 2 ' Word tipi veri okumak için
\ 2'şer adım ilerle.

W2=0
Write B1, W2.HIGHBYTE ' Word'ün high byte'ını yaz.
Write B1+1, W2.LOWBYTE ' Sonraki adrese low byte'ı yaz.
Next B1
GoTo Ana_prog ' İşlemi devamlı yap.
End

```

### Programın çalışması:

Ana program içerisinde RA0, RA1 ve RA2 butonları kontrol edilmektedir. RA0 butonuna basıldığında program yaz etiketine dallanarak for-next döngüsündeki B1 adres değişkenini ilk önce 0'a kurar. Programda bu adrese yazılacak veri  $W2=B1+1000$  ifadesi ile tespit edilmektedir. EEPROM'un 0. adresine yazılacak veri  $W2=1000$ 'dir. (Elbette bu veri programın yaptığı işe göre farklı veriler olabilir. Burada amacımız sadece EEPROM alanına veri yazdırmayı öğretmek olduğu için ayrıntılardan kaçınmak amacıyla basit bir formülle elde edilen sayılar girilmiştir.)

EEPROM alanına Write B1, W2.HIGHBYTE ve Write B1+1, W2.LOWBYTE komutlarıyla yazılan ilk veri EEPROM bellekte aşağıdaki gibi yer alır.

B1		Adres	Veri
0	B1	\$00	10
0	B1 + 1	\$01	00

For-next döngüsünde B1=2 değerini alınca bu defa yeni adreslere yazılan verileri de  $W2=1001$  olur ve bellekte aşağıdaki gibi yer alır.

B1		Adres	Veri
0	B1	\$00	10
0	B1 + 1	\$01	00
2	B1	\$02	10
2	B1 + 1	\$03	01



## KENDİNİZ UYGULAYINIZ

Programın çalışması esnasında klavyeden girilen verileri EEPROM belleğin istenen adresine yazdıran programı aşağıda verilen bilgilere göre yapınız:

1. Program başlayınca EEPROM'un 0. adresinden itibaren yazmaya hazır olsun. Her veri girişinden sonra bu adres otomatik olarak 1 arttırılsın.
2. 4x4 klavye \$0 ile \$F arasında sayılar üretsin. Klavyeden bir tuşa basılınca (0 hariç) bu değer bir değişken içerisinde saklansın.
3. Klavyeden 0 sayısı girildiğinde (Bu tuş ENTER tuşu olarak kabul edilsin) değişken içerisindeki değer o anda hangi adres aktif ise o adrese yazılsın.
4. Klavyeden F sayısı girildiğinde de EEPROM'a yazılan veriler 0.5 sn aralıklarla LCD ekranda gösterilsin.

## 9.3. EEPROM

**Dizilim:** `EEPROM { Adres, } [ Sabit {,Sabit...} ]`

Chip üzeri EEPROM alanına programlama esnasında veri yazmak için kullanılır. *Adres*, sabitin yazılacağı yeri gösterir. Eğer *adres* yazılmazsa ilk EEPROM komutu 0 adresinden itibaren yazmaya başlar. Takip eden komutlar ise bu adresten itibaren yazmaya devam eder.

*Sabit*, sayısal veya karakter sabit olabilir. Sayısal sabitlerin sadece alt byte'ı saklanır. Karakter sabitlerin ise ASCII karşılıkları olan byte'lar ardı ardına yazılır.

EEPROM komutu sadece chip üzeri EEPROM alanı bulunan PIC16F84, PIC16C84 ve PIC16F877 gibi mikrodenetleyicilerde çalışır. Mikrodenetleyicinin enerjisi kesilse bile EEPROM alanındaki veriler aynen kalır.

EEPROM komutuyla sadece mikrodenetleyici programlanırken veri yazılabilir. Eğer programın çalışması esnasında veri (sabit) yazmak istenirse WRITE komutunu kullanmak gerekir.

### Örnek:

```
EEPROM ["ABCDEF"]  '0 adresinden itibaren 65,
                  '66, 67, 68, 69, 70 verilerini yazar.
EEPROM 5, [10, 20, 30]  '5 adresinden itibaren 10,
                       20, 30 verilerini yazar.
```

## 9.4. DATA

**Dizilimi:** `{ etiket } DATA { @ Adres, } Sabit {,Sabit...}`

Mikrodenetleyiciyi programlama esnasında chip üzerindeki EEPROM belleğe *sabit* değerler yazmak için kullanılır. Eğer *adres* yazılmazsa ilk DATA komutu 0

adresinden başlar ve takip eden DATA komutları bu adresten itibaren sabitleri yazmaya devam eder. Eğer adres için bir değer girilmişse, sabit değerlerin girileceği adresin başlangıç adresi belirtilmiş olur. İstenirse EEPROM'un başlangıç adresi olarak *etiket* (iki nokta üst üste ":" içermeyen) kullanılabilir.

*Sabit'* ler sayısal veya karakter sabit olabilir. Sayısal sabitlerin önüne word tipi olduğunu belirleyen WORD kelimesi konulmadığı sürece sadece sayının alt byte'ı kaydedilir. String sabitlerdeki her bir harfin ASCII kodu karşılığı olan sayı bir byte olarak kaydedilir.

DATA komutu sadece chip üzeri EEPROM belleği bulunan PIC16F84 ve PIC16F877A mikrodenetleyicilerinde kullanılır. EEPROM bellek içerisindeki data enerji kesildiğinde silinmez.

DATA komutuyla sadece mikrodenetleyici programlanırken veri yazılabilir. Eğer programın çalışması esnasında veri (sabit) yazmak istenirse WRITE komutunu kullanmak gerekir.

EEPROM komutu ile aralarında yazılış formatı bakımından fark olsa bile benzer işlemleri yaparlar. PBP'da böyle benzer iki komutun bulunmasının nedeni, Basic Stamp 2 kullanıcılarının alışageldiği DATA komutunu kullanabilmelerini sağlamak amacıyla yapılan bir uyumluluk kaygısındanadır. Aralarındaki tek fark DATA komutunda word tipi veriler varken EEPROM komutunda bulunmamasıdır.

#### Örnek:

```
DATA @5, 10, 20, "A" \ 5 adresinden itibaren 10, 20
                    \ sayılarını ve A harfinin ASCII kodu
                    \ karşılığını EEPROM'a kaydeder.

DATA word $1234 \ İlk iki byte'a $34, ikinci iki
                \ byte'a $12 kaydeder.

DATA $1234 \ İlk byte'a $4, ikinciye $3
           \ üçüncüye $2, dördüncüye $1
           \ kaydeder.

DATA (4), 0 (10) \ 4 adres atlar ve 10 tane 0
                \ kaydeder.
```

#### UYGULAMA 9.2 : EEPROM komutuyla dahili EEPROM belleğe programlama esnasında veri yazmaya örnek program.

Deneme: PIC PROG/DEKA 87x

Deneme : PIC DEKA V1.1

Bu programda PIC programlama kartı ile PIC programlanırken chip üzeri EEPROM belleğe veriler yazdırılmıştır. Bu veriler 0..4 adreslerine yazdırılan "ALTAS" karakterleri ve 5..9 adreslerine yazdırılan (75, 73, 84, 65, 80) sayısal sabitleridir. Program çalışınca 0. adresten itibaren okunan veriler 0.5 sn aralıklarla okunarak

LCD ekranda gösterilmektedir. Okunup gösterme işlemi enerji kesilene kadar sürekli devam etmektedir.

```

-----
'Programın adı: Uyg_9_2.bas
'Programın işlevi: EEPROM komutu ile PIC'i programlarken chip
'üzeri EEPROM alanına veri yazdırır ve daha sonra bu verileri
'READ komutu ile okuduktan sonra seri LCD'de gösterir.
'Tarih:30/07/2007   Uyarlama:1.00   Derleyici:PBP
-----
DEFINE LCD_DREG      PORTB      ' RB0-->DB4 PortB.0
DEFINE LCD_DBIT      0          ' RB1-->DB5 PortB.1
                                ' RB2-->DB6 PortB.2
                                ' RB3-->DB7 PortB.3

DEFINE LCD_RSREG     PORTB
DEFINE LCD_RSBIT     4          ' RS   PortB.4
DEFINE LCD_EREG      PORTB
DEFINE LCD_EBIT      5          ' E   PortB.5
DEFINE LCD_BITS      4          ' 4 bit ile iletişim
DEFINE LCD_LINES     2          ' LCD satır sayısı

B1 var byte
B2 var byte
ADCON1 = 7

EEPROM ["ALTAS"]      ' EEPROM[0..4] = 65, 76, 84, 65, 83
EEPROM 5, [75,73,84,65,80] ' EEPROM[5..9] = 75, 73, 84, 65, 80

Dongu:
  LCDOUT $FE, 1      ' LCD'yi sil kursör satır başına.
  For B1 = 0 To 4    ' READ komutunun kullanımı.
    Pause 500        ' Buton arkını söndür.
    Read B1,B2       ' EEPROM içeriğini oku.
    LCDOUT $FE, $06, B2 ' Veriyi LCD'de göster.
  Next B1
  LCDOUT $FE, $C0    ' Kursör bir alt satırın başına.

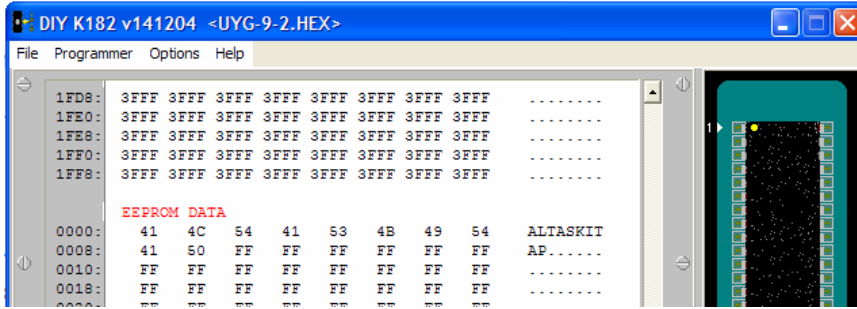
  For B1 = 5 To 9   ' READ komutunun kullanımı.
    Pause 500        ' Buton arkını söndür.
    Read B1,B2       ' EEPROM içeriğini oku.
    LCDOUT $FE, $06, B2 ' Veriyi LCD'de göster.
  Next B1
  Goto Dongu        ' İşlemi devamlı yap.
-----

```

### Programın Çalışması:

Programı programlama kartınızla PIC16F877A'ya yüklerken kullandığınız yazılımda (Bizim kullandığımız "micropro") EEPROM DATA alanına yazılacak verilerin

görüntüleneceği kısım bulunur. Uyg\_9\_2.HEX adlı dosyanızı yükleyip, EEPROM bölgesine yazılacak olan verilerin HEX kodunu ve ASCII kodunu inceleyiniz. Aşağıdaki ekranda görülen verileri siz de göreceksiniz.



**Şekil 9.2 :** MikroPro yazılımında EEPROM bölgesini görmek.

EEPROM'a yazılacak olan verilerin yeri programı derlerken belirlenir. Bu nedenle programlama esnasında PIC'e yazılır. Programımızda EEPROM ["ALTAS"] komutuyla "ALTAS" karakterlerinin ASCII kodu karşılığı olan sayılar ve EEPROM 5, [75,73,84,65,80] komutuyla da "KITAP" karakterlerinin ASCII kodu karşılıkları doğrudan EEPROM'a yazdırılmıştır.

Dongu program parçasında for-next döngüsünde 0..4. adreslerdeki veriler LCD ekrana gönderildikten sonra, ikinci for-next döngünde 5..9. adreslerdeki veriler 0.5 sn aralıklarla LCD'ye gönderilmiştir. Enerji kesilene kadar EEPROM'a bir defa yazılmış olan veriler devamlı olarak okunur ve LCD'de gösterilir.

#### İşlem Basamakları:

1. Programı PIC16F877A'ya yazdırınız. Programı şekil 9.1'deki devre üzerinde veya PIC PROG/DEKA 87X üzerinde doğrudan deneyebilirsiniz.
2. Devreye enerji vererek çalıştırınız.
3. LCD'nin birinci satırında "ALTAS" kelimesini oluşturan harflerin tek tek görülmesinden sonra ikinci satırda "KITAP" kelimesinin harflerinin 0.5 sn aralıklarla ekrana geldiğini görünüz.
4. Enerjiyi kesip, tekrar veriniz. Program yukarıda yukarıdaki işlemleri baştan itibaren yapacaktır.

## 9.5. I2CREAD

**Dizilimi:** *I2CREAD Data\_pini, Clock\_pini, Kontrol\_kodu, {Adres},*  
*[Değişken{,Değişken...}] {Etiket}*

*Kontrol\_kodu* ve isteğe bağlı adres byte'larını *clock\_pini*'nden ve *data\_pini*'inden gönderir. Gönderilen bu verilere göre okunan veri byte'larını *değişken* içerisine kaydeder. *Clock\_pini* ve *data\_pini* 0-15 arasında sabit bir sayı veya 0-15 sayılarını içeren bir değişken (örn. B0) ya da pin adı (örn. PORTA.0) olabilir.