PIC MCU ile UYGULAMALAR

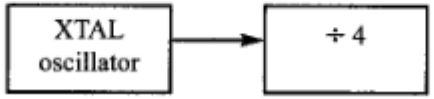
- Gecikme Programları
- TMRO Gecikmesi
- 7 Segment Göstergeler
- Sayaç Örnekleri
- Trafik Sinyalizasyonu
- ADC-DAC Uygulamaları
- Kesmeler ve Uygulamaları
- Tuş Takımı (Keypad) Uygulamaları
- Paralel LCD Uygulamaları
- Seri LCD Uygulamaları
- Step Motorlar
- DC Motorlar
- Servo Motorlar

Gecikme Programları

Örnek 1: a) 10 MHz b) 16MHz c) 4 MHz saat frekansına sahip PIC MCU için bir komutu işleme süresi kaç sn dir. (Farzedinki frekans ön bölücü değerleri yoktur.

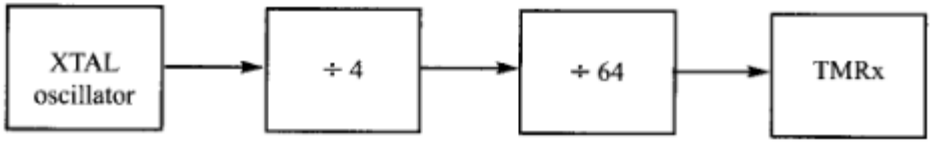
Çözüm:

(a) 1/4 × 10 MHz = 2.5 MHz and T = 1/2.5 MHz = 0.4 µs
(b) 1/4 × 16 MHz = 4 MHz and T = 1/4 MHz = 0.25 µs
(c) 1/4 × 4 MHz = 1 MHz and T = 1/1 MHz = 1 µs



Örnek 2: a) 10 MHz b) 16MHz saat frekansına sahip PIC MCU için bir komutu işleme işleme süresi kaç sn dir. (Farzedinki frekans ön bölücü değeri 1/64)

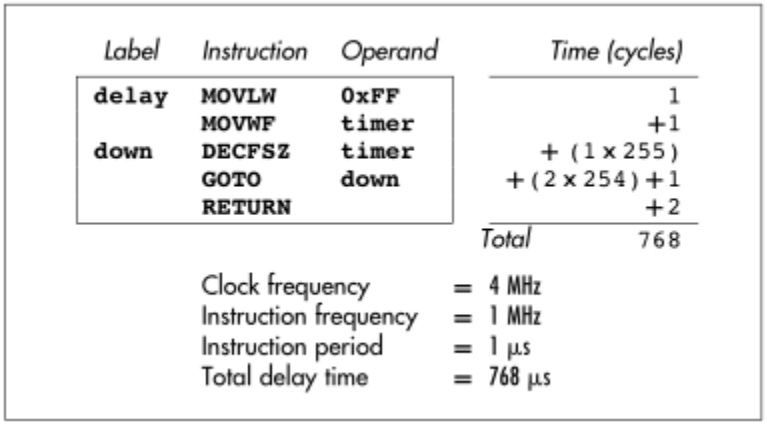
Çözüm:



(a) 1/4 × 10 MHz = 2.5 MHz and 1/64 × 2.5 MHz = 39062.5 Hz due to 1:64 prescaler and T = 1/39062.5Hz = 25.6 µs

(b) 1/4 × 16 MHz = 4 MHz and 1/64 × 4 MHz = 62500 Hz due to prescaler and T = 1/62500 Hz = 16 µs

Gecikme Programları



Label	Instruction	Operand	Time (cycles)
delay	MOVLW	0xFF	1
	MOVWF	timer	+1
down	DECFSZ	timer	+ (1 x 255)
	GOTO	down	+ (2 x 254) + 1
	RETURN		+2
			Total 768

Clock frequency = 4 MHz
Instruction frequency = 1 MHz
Instruction period = 1 µs
Total delay time = 768 µs

Örnek 2: Tek bir döngü ile yaklaşık 1000 saykıllık bir gecikme sağlayacak programı yazınız.

Çözüm: Toplam Gecikme= Döngü dışındakiler + SAYAC* Döngü içindekiler

= 1+1+ (249*4) + 2=1000 saykıl

Tabii 1000 saykıllık gecikme programının ne kadar sürelik bekleme sağlayacağı PIC mikro denetleyicinin çalışma frekansına bağlıdır. 10 MHz lik PIC MCU için toplam gecikme; 1000*0.1µS*4=400µS lik bir gecikme sağlayacaktır.

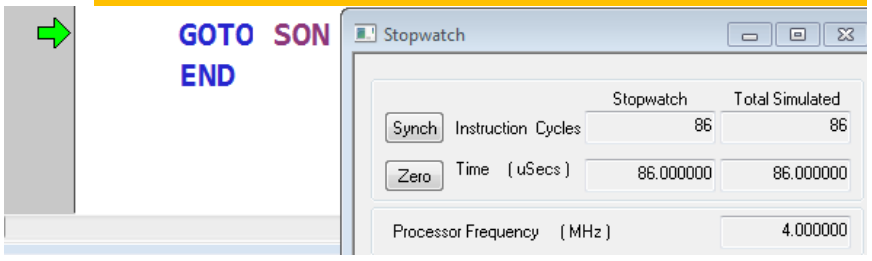
```
        MOVLW d'250'  ;1 saykıl
        MOVWF SAYAC   ;+1 saykıl
DON
        NOP           ;+1*250
        DECFSZ SAYAC, F ;+1*250 +(1)
        GOTO DON      ;+2*249
        RETURN        ;+2
```

TG=250+250+2*249+3+2= 1003 saykıl

1) Aşağıdaki program kaç saat saykılında çalışır?
Ve 4MHz lik bir MCU için çalışma süresini hesaplayınız?

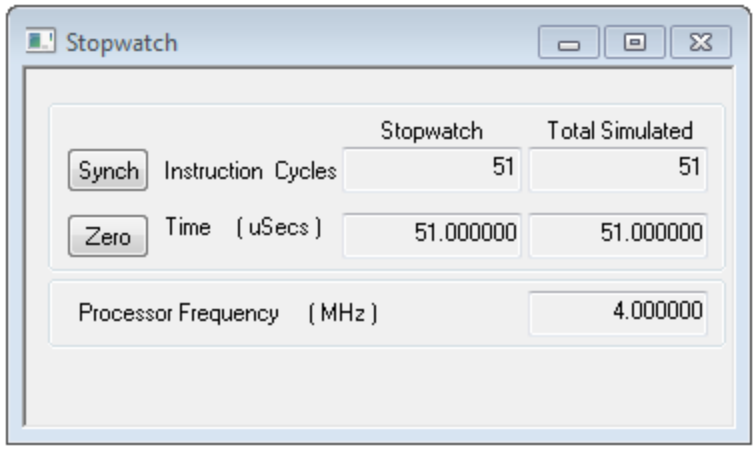
```
;O dan 9 a ARTAN SAYICI
        LIST P=16F877
        #INCLUDE<P16F877.INC>
        SAYAC   EQU 20H          ;16F84 İÇİN 0CH OLACAK
1       CLRF    PORTB            ;PORTB temizlenir
1       BSF     STATUS, 5        ;BANK1'e geçilir
1       CLRF    TRISB            ;PORTB nin tüm uçları çıkış olacaktır
1       BCF     STATUS, 5        ;BANK0'a geçilir

1       CLRF SAYAC
DON:
        MOVF SAYAC,W        1
        MOVWF PORTB         1
        INCF SAYAC,F        1
        MOVLW .10           1
        SUBWF SAYAC,W       1
        BTFSS STATUS,Z      1
        GOTO DON            2
SON
        GOTO SON
        END
```

TG= DonguİIci*Sayac+DonguDisi=
=8*10+5+1=86 saykıl



ÖRNEK 3: PORTB ye bağlı LEDleri sırası ile LSB- MSB taraflarını yakan programdaki BEKLE alt programı ne kadarlık bir gecikme sağlar?

```
    LIST P=16F877
    #INCLUDE<P16F877.INC>
    SAYAC   EQU 20H      ;16F84 İÇİN 0CH OLACAK
    CLRF    PORTB        ;PORTB temizlenir
    BSF     STATUS, 5    ;BANK1'e geçilir
    CLRF    TRISB        ;PORTB nin tüm uçları çıkış olacak
    BCF     STATUS, 5    ;BANK0'a geçilir
BASLA
    MOVLW h'0F'
    MOVWF PORTB
    CALL BEKLE
    SWAPF PORTB,F
    CALL BEKLE
BEKLE
    MOVLW .10     ;1 saykıl
    MOVWF SAYAC   ;1 saykıl
DON
    NOP           ;1*10
    DECFSZ SAYAC, F ;1*10
    GOTO DON      ;2*9
    RETURN
    END
```



İç içe Döngüler ile Yapılan Gecikme

Tek bir döngü ile genelde istediğimiz zaman gecikmesini elde edemeyebiliriz. Çünkü bir kaydedici içine yazabileceğimiz en büyük değer ondalık olarak 255 tir. Bu da 255 ten fazla tekrar yaptıramayacağımız anlamına gelir. Bu durumda iç içe döngüler kullanarak bu sayıyı çok çok büyütebiliriz. İç içe döngüler kullandığımızda hem tekrarlanan komut sayısı artacağından döngünün bir adımının harcadığı süre uzayacak, hem de içi içe döngüler sebebiyle iki (veya daha fazla) döngünün çarpımı kadar sayıda tekrar olabilecektir.

Örnek 2. İç içe döngülü bir gecikme alt programı ;

Bu programda yaklaşık

TG= 3*SAYAC1*SAYAC2

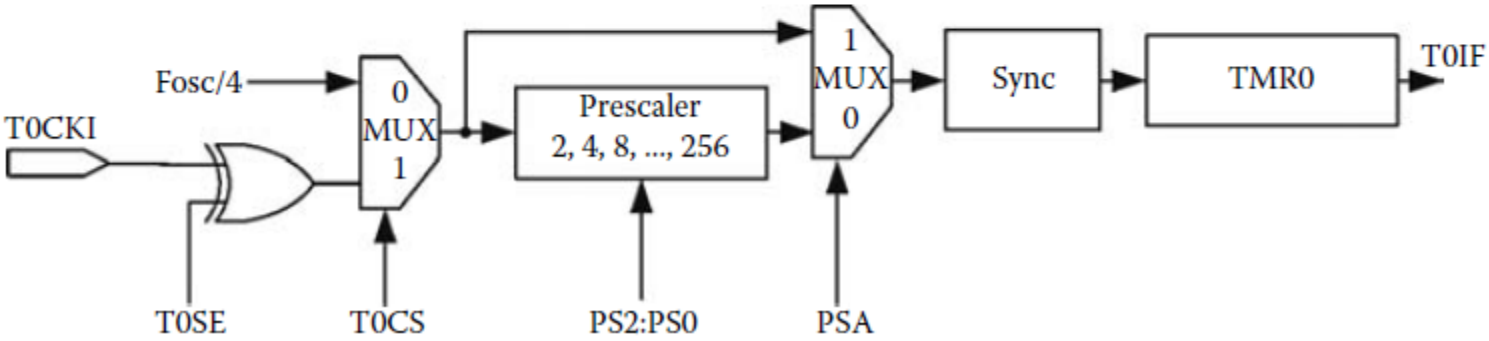
TG = 3*255*255 kadardır.

MHz lik dahili saat saykılında bu süre yaklaşık 195µS lik bir gecikme sağlar.

Gecikme		
	MOVLW d'255'	*1*
	MOVWF SAYAC1	*1*
DON1		
	MOVLW d'255'	*1*SAYAC1*
	MOVWF SAYAC2	*1*SAYAC1*
DON2		
	DECFSZ SAYAC2, F	*1*SAYAC1*SAYAC2*
	GOTO DON2	*2*SAYAC1*SAYAC2*
	DECFSZ SAYAC1, F	*1*SAYAC1*
	GOTO DON1	*2*SAYAC1*
	RETURN	*2*

Orta Ölçekli PIC TIMER Yapısı

Timer	Size	Prescaler Division	Postscaler Division	SFR for Count Number	Overflow
Timer0	8 bits	2, 4, …, 256	NO	TMR0	bit T0IF in OPTION
Timer1	16 bits	1, 2, 4, 8	NO	TMR1H, TMR1L	bit TMR1IF in PIR1
Timer2	8 bits	1, 4, 8	1, 2, …, 16	TMR2	bit TMR2IF in PIR2

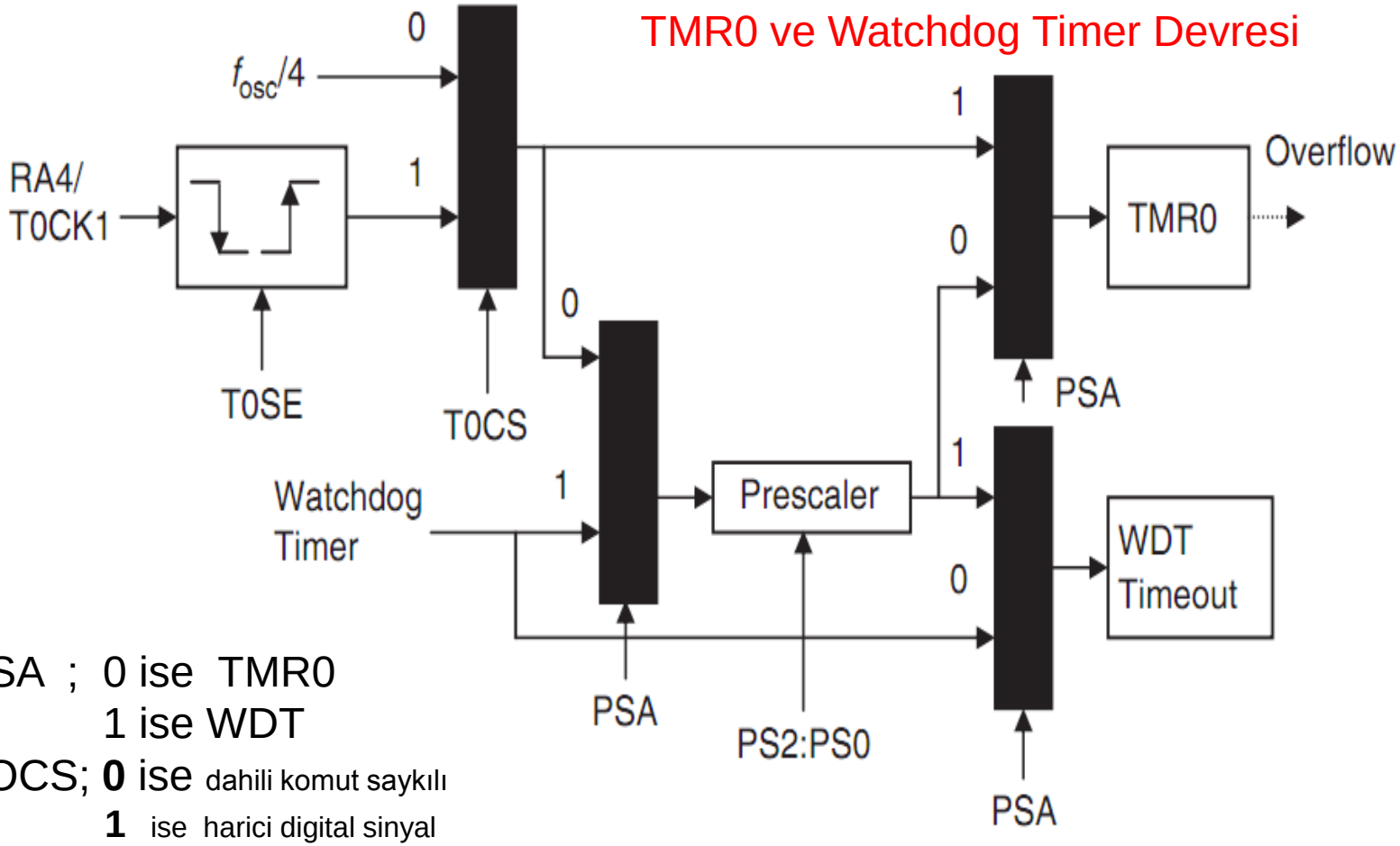


PSA ; 0 ise TMR0
1 ise WDT

TOCS; **0** ise dahili komut saykılı
1 ise harici digital sinyal

Zamanlayıcı kullanan Gecikme Prog;

TMR0 ve Watchdog Timer Devresi



PSA ; 0 ise TMR0
1 ise WDT

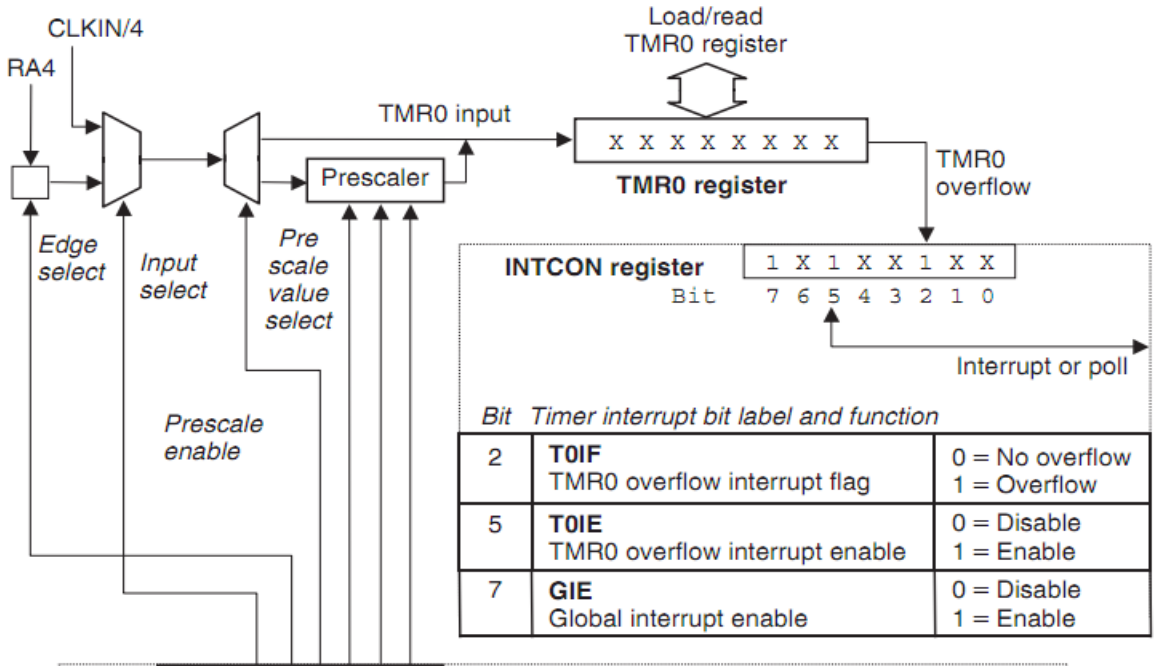
TOCS; **0** ise dahili komut saykılı
1 ise harici digital sinyal

OPTION

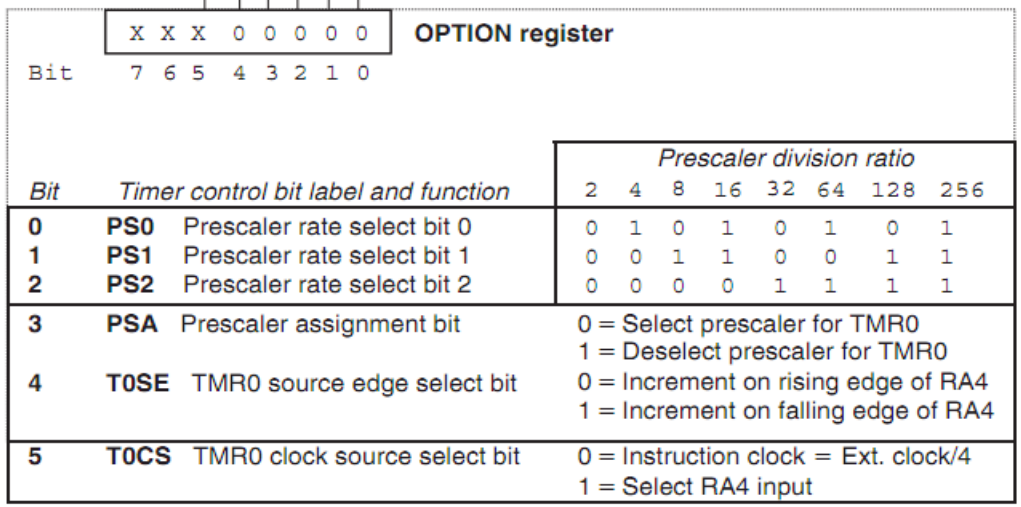
7	6	5	4	3	2	1	0
RBPU#	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

INTCON

7	6	5	4	3	2	1	0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIIF



Bit	Timer interrupt bit label and function	
2	**T0IF** TMR0 overflow interrupt flag	0 = No overflow 1 = Overflow
5	**T0IE** TMR0 overflow interrupt enable	0 = Disable 1 = Enable
7	**GIE** Global interrupt enable	0 = Disable 1 = Enable



Bit	Timer control bit label and function		Prescaler division ratio 2	4	8	16	32	64	128	256
0	**PS0**	Prescaler rate select bit 0	0	1	0	1	0	1	0	1
1	**PS1**	Prescaler rate select bit 1	0	0	1	1	0	0	1	1
2	**PS2**	Prescaler rate select bit 2	0	0	0	0	1	1	1	1
3	**PSA**	Prescaler assignment bit	0 = Select prescaler for TMR0 1 = Deselect prescaler for TMR0							
4	**T0SE**	TMR0 source edge select bit	0 = Increment on rising edge of RA4 1 = Increment on falling edge of RA4							
5	**T0CS**	TMR0 clock source select bit	0 = Instruction clock = Ext. clock/4 1 = Select RA4 input							

MOVLW b'11010111'; *TMR0, dahili sinyal kaynağı ve prescaler:111 seç*
MOVWF OPTION_REG

TMR0 ile Gecikme

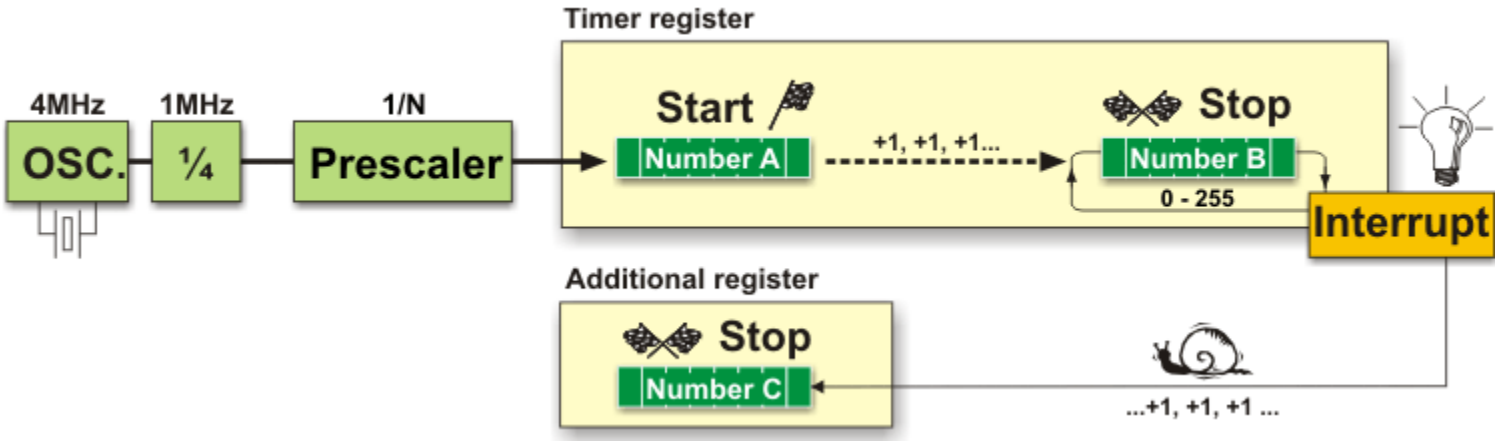
	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (x)
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

TMR0 sayıcısının FF (255) den 00 'a geçmesi TMR0 kesmesine se bep olur ve bu kesme sonucunda INTCON kesme kaydedicisinin 2. (T0IF) biti 1 değerini alır. Bu kesmeyi kullanabilmek için daha önc esinde **INTCON** kaydedicisinin TOIE bitinin 1 yapılarak kesmeye izin verilmesi gerektiği unutulmamalıdır.

Kesme gecikmesi (Overflow time)= 4 *TOSC * Prescaler *(256 – TMR0 başlangıç değeri)

- Bu formülden TMR0 başlangıç değeri de çekilebilir. O zaman

TMR0 = 256 – (Gecikme zamanı)/(4 *TOSC* Prescaler)



TMR0 ile Gecikme

	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (x)
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

TMR0 hem yazılabilir, hem okunabilir bir sayıcıdır. OPTION kaydedicisi ile belirlenebilen frekans bölme seçeneği vardır. Saymaya ana programda, alt programlarda ve kesme alt programlarında da devam eder. Bu bir avantajdır. En önemli özelliği ise, saydığı değer FFh s ayısından 00h sayısına geçerken oluşan taşmada, INTCON kaydedicisinde T0IF bayrağı 1 değerini alır ve bu değer kullanılarak bir kesme alt programı çalıştırılabilir. Öncesinde TOIE biti «1 « yapılarak TMRO da taşma olması halinde kesmeye izin verilmesi sağlanmalıdır.

- **Kesme gecikmesi (Overflow time)= 4 *Tosc * Prescaler *(256 – TMR0 başlangıç değeri)**
- Bu formülden TMR0 başlangıç değeri de çekilebilir. O zaman

TMR0 = 256 – (Gecikme zamanı)/(4 *Tosc* Prescaler)

TMR0 Gecikme Alt programı

Örnek 4:

Osilatör frekansı 4MHz olan bir PIC için, OPTION kaydedicisindeki PS2, PS1, PS0 bitlerinin değerleri sırasıyla b'111' şeklindedir. TMR0 zamanlayıcısının sıfır(0) değerinden saymaya başladığı kabulü ile bu PIC kaç saniye sonra TMR0 tarafından bir kesme oluşturur?

Çözüm:

4 MHz saat frekansı ise peryot , TOSC =1/fosc =0.25 µs

PS2:PS0 = "111" olduğundan Prescaler= 1:256

Kesme gecikmesi = 4 *TOSC * Prescaler *(256 – TMR0 başlangıç değeri)

Kesme gecikmesi (Overflow time) = 4 *0.25 µs * 256 *(256 – 0)

= 65536 µs =65.536ms

```
MOVLW b'11010111' ;TMR0, DAHİLİ SİNYAL,1:256
MOVWF OPTION_REG
…
GECIKME
    CLRF TMR0
DON
    BTFSS INTCON, TOIF
    GOTO DON
    BCF INTCON, TOIF
    RETURN
```

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

TMR0 Gecikme Alt programı

```
CLRF      TRISB
MOVLW b‘00000101’
OPTION ; for internal timer/64
CLRF PORTB ; Clear Port B (LEDs Off)
next:
CLRF TMR0 ; clear timer register
BCF INTCON,T0IF ; clear timeout flag
check
BTFSS INTCON,T0IF ; wait for next tir
GOTO check ; by polling timeout flag
INCF PORTB ; Increment LED Count
GOTO next ; repeat forever...
END ;Terminate source code
```

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

İkili (binary) olarak artırma ve azaltma işlemi uygulamaları

Örnek 5: 4 Bitlik Binary(ikili) Geri Sayıcı(15-0)

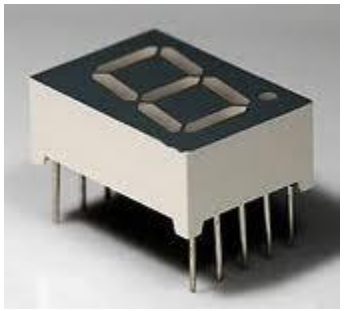
```
            LIST P=16F84A
            #INCLUDE<P16F84A.INC>
            CLRF        PORTB       ;PORTB temizlenir
            BSF         STATUS, 5   ;BANK1'e geçilir
            MOVLW b'11010111'       ;TMR0, DAHİLİ SİNYAL,1:256
            MOVWF OPTION_REG
            MOVLW       h'0F'
            MOVWF       TRISA       ;PORTA nın tüm uçları giriş
            CLRF        TRISB       ;PORTB nin tüm uçları çıkış olacaktır
            BCF         STATUS, 5   ;BANK0'a geçilir
BASLA
            MOVLW d'16'
            MOVWF PORTB
TEST:
            BTFSC PORTA,0
            GOTO TEST
            CALL BEKLE ;65,53ms
            CALL BEKLE ;65,53+65,53=131ms lik gecikme
            DECFSZ PORTB,F
            GOTO TEST
            GOTO BASLA              ;Başa dön

BEKLE:      ;gecikme alt programı
            CLRF TMR0
DON
            BTFSS INTCON,T0IF
            GOTO DON
            BCF INTCON,T0IF
            RETURN
            END
```

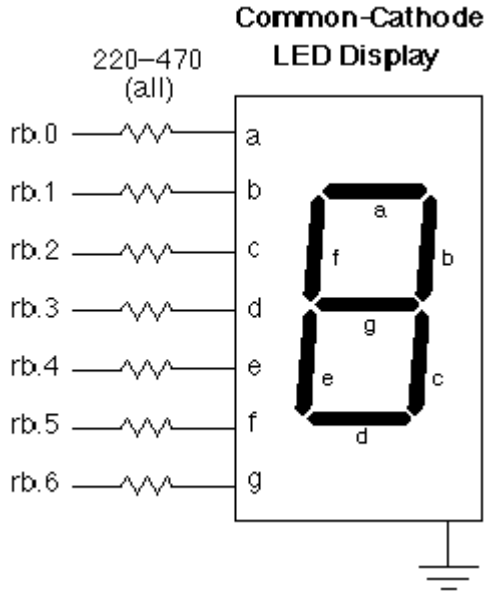
PORTB ye bağlı ledlerde binary sayım:

0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	1	1	3
0	0	0	0	0	1	0	0	4
0	0	0	0	0	1	0	1	5
0	0	0	0	0	1	1	0	6
0	0	0	0	0	1	1	1	7
0	0	0	0	1	0	0	0	8
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
1	1	1	1	1	1	1	1	255

Çevrim Tabloları Ve 7 Segment Display Uygulaması

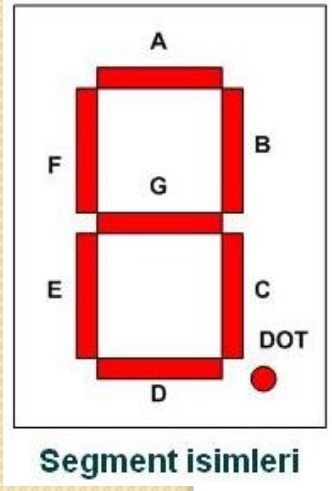
Çevrim / Bakış tabloları ile bir kodu başka bir koda dönüştürmek için kullanılırlar. Örneğin PIC mikro denetleyiciyi portlarına bağlı 7 Segment display / gösterge üzerinde hexadecimal (onaltılık tabandaki) sayıları göstermek, sıcaklık dönüşümü(derece-fahrenayt gibi) yapma, sinus, kosinus alma gibi işlemlerde dönüşüm/çevrim tabloları kullanılır. 7 Segment display kodlaması için aşağıdaki tablo kullanılır.



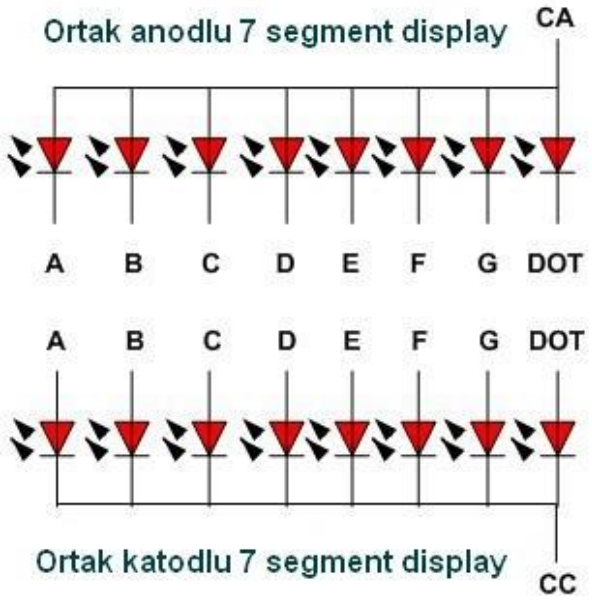


Çevrim Tabloları Ve 7 Segment Display Uygulaması

7 segment displaylerin içerisinde aslında 8 adet led bulunmaktadır. Her bir segment bu ledler ile oluşturulmuştur ve göstergenin hangi segmentinin yanmasını istiyor isek o ledi yakarız. 7 segment displayler ortak anot ve ortak katotlu olmak üzere iki farklı şekilde bulunurlar.

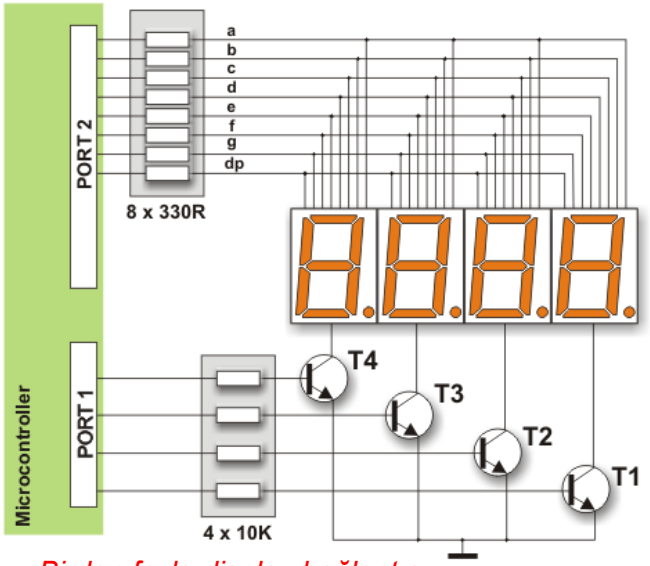


Segment isimleri

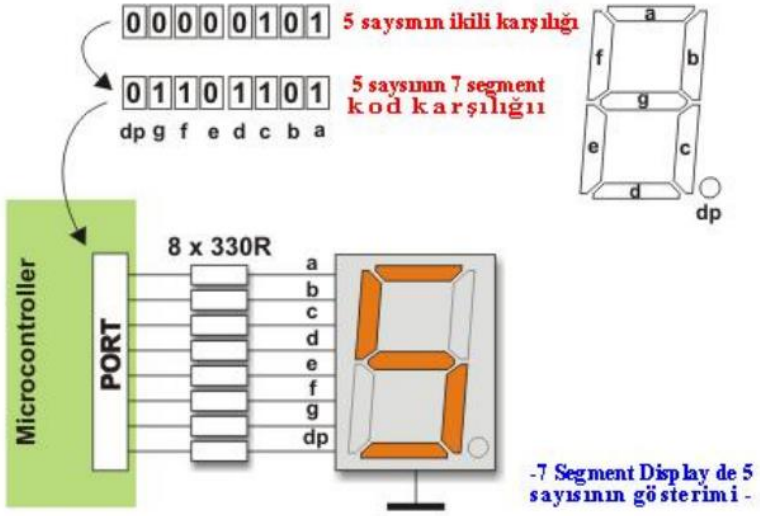


. G F E D C B A	
0 →	b' 0 0 1 1 1 1 1 1'
1 →	b' 0 0 0 0 0 1 1 0'
2 →	b' 0 1 0 1 1 0 1 1'
3 →	b' 0 1 0 0 1 1 1 1'
4 →	b' 0 1 1 0 0 1 1 0'
5 →	b' 0 1 1 0 1 1 0 1'
6 →	b' 0 1 1 1 1 1 0 1'
7 →	b' 0 0 0 0 0 1 1 1'
8 →	b' 0 1 1 1 1 1 1 1'
9 →	b' 0 1 1 0 1 1 1 1'
A→	b' 0 1 1 1 0 1 1 1'
b →	b' 0 1 1 1 1 1 0 0'
C→	b' 0 0 1 1 1 0 0 1'
d →	b' 0 1 0 1 1 1 1 0'
E→	b' 0 1 1 1 1 0 0 1'
F →	b' 0 1 1 1 0 0 0 1'

Çevrim tablosunda uygun kodu seçmek için **program sayıcıyı** (PCL-Program Counter), seçilen kodu ana programa göndermek için de **RETLW** komutunu kullanırız



Birden fazla display bağlantısı



Çevrim tablosundaki verilere sıralı olarak erişerek PCL'ye yani o anki adrese istediğimiz sayıyı ekleyerek istediğimiz adrese / elemana ulaşırız. PCL'nin o anki değerine ADDWF ile istediğimiz sayıyı ekleriz.

Kullanım Şekli**: ADDWF PCL, F**

İstediğimiz değeri geri döndürecek komut ise **RETLW** (RETLW **h'3F' gibi)** dir.

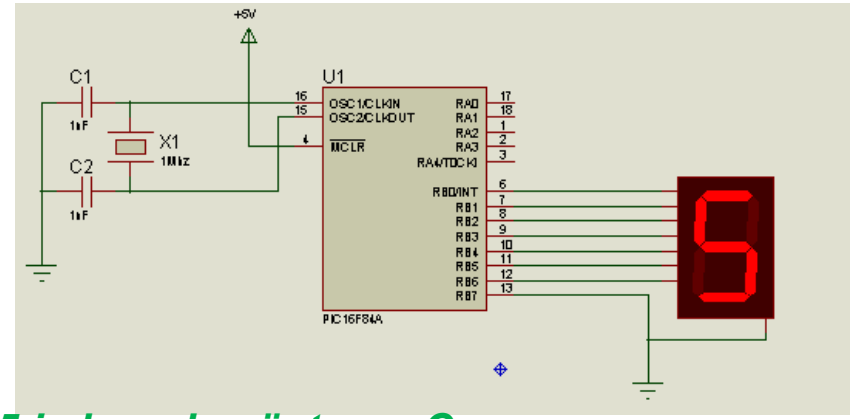
RETLW komutu ile alt alta yazılan sayı değerleri **dt** komutu yanyana yazılabilir.

DIZI

ADDWF PCL, F
dt h'3F', h'06', h'5b',….

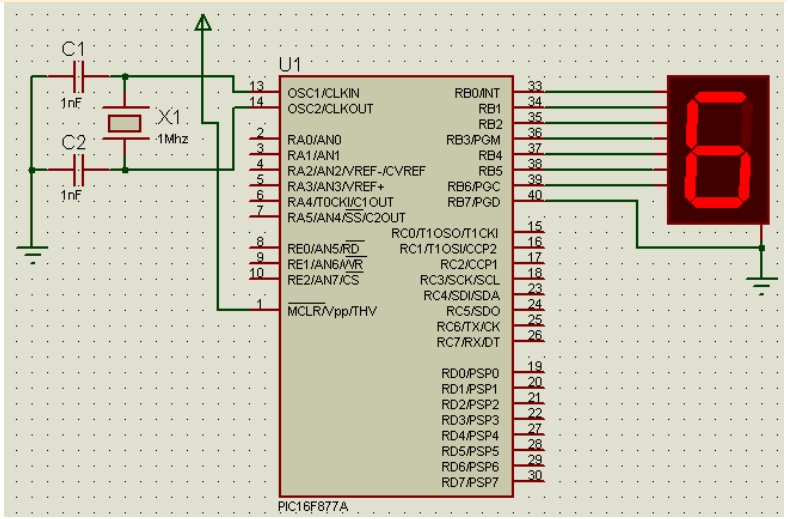
Örnek 6: 7 segment display de 5 sayısını gösteren programı yazınız.

```
        LIST P=16F84
        #INCLUDE<P16F84.INC>
        BSF STATUS,5
        CLRF TRISB
        BCF STATUS,5
        CLRF PORTB
basla
        MOVLW h'05'             ;W kaydedicisine h'05' değerini yükle
        CALL DIZI
        MOVWF PORTB             ;W içeriğini PORTB ye aktar
DON
        GOTO DON
DIZI
        ADDWF PCL, F            ;W içeriğini PCL ye aktar
        RETLW b'00111111'       ;W ya 0 değeri yüklendi
        RETLW b'00000110'       ;W ya 1 değeri yüklendi
        RETLW b'01011011'       ;W ya 2 değeri yüklendi
        RETLW b'01001111'       ;W ya 3 değeri yüklendi
        RETLW b'01100110'       ;W ya 4 değeri yüklendi
        RETLW b'01101101'       ;W ya 5 değeri yüklendi
        END
```




```
// 5 i ekranda gösteren C programı
#include <htc.h>
const unsigned char
dizi[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D};
void main(void)   // Ana fonksiyon alanı
{

  TRISB=0x00;   // PORTB çıkıs ol. yönlendir
  PORTB=0x00;   // PORTB'nin tüm ucları çıkış
  for(;;)   // Sonsuz döngüye giriliyor
  {
  PORTB=dizi[5];  // 7 segment değeri alınıyor
  }
}
```

Örnek 6: 7 segment display de 6 sayısını gösteren programı yazınız.




```
LIST P=16F877
#INCLUDE<P16F877.INC>
CLRF PORTB
BSF STATUS,5 ; BANKSEL TRISB
CLRF TRISB
BCF STATUS,5 ; BANKSEL PORTB

BASLA

MOVLW .6
CALL DIZI
MOVWF PORTB

DON

GOTO DON

DIZI

ADDWF PCL,F
RETLW b'00111111'      ;W ya 0 değeri yüklendi
RETLW b'00000110'      ;W ya 1 değeri yüklendi
RETLW b'01011011'      ;W ya 2 değeri yüklendi
RETLW b'01001111'      ;W ya 3 değeri yüklendi
RETLW b'01100110'      ;W ya 4 değeri yüklendi
RETLW b'01101101'      ;W ya 5 değeri yüklendi
RETLW b'01111101'      ;W ya 6 değeri yüklendi
RETLW b'00000111'      ;W ya 7 değeri yüklendi
RETLW b'01111111'      ;W ya 8 değeri yüklendi
RETLW b'01101111'      ;W ya 9 değeri yüklendi
END
```

Örnek 7: 0 dan 9 kadar olan sayıları PORB uçlarına bağlı 7 segment display'de gösteren programı gerçekleştiriniz.

```
        LIST P=16F84A
        #INCLUDE <P16F84A.INC>
        SAYAC1 EQU h'0D'
        BSF     STATUS,5        ;BANK1 e geçiş yap
        CLRF TRISB              ;PORTB nin tüm uçları çıkış seçildi
        BCF     STATUS,5        ;BANK0 a geçiş yap
        CLRF    PORTB           ;PORTB yi temizle
Basla
        MOVLW   h'00'           ;W kaydedicisine h'00' değerini yükle
        MOVWF   SAYAC1
DON     MOVF    SAYAC1,W
        CALL    DIZI
        MOVWF   PORTB           ; W içeriğini PORTB ye aktar
        INCF    SAYAC1,F        ; SAYAC1 değerini artır
        GOTO    DON
DIZI
        ADDWF PCL, F            ;W içeriğini PCL ye aktar
        RETLW b'00111111'       ;W ya 0 değeri yüklendi
        RETLW b'00000110'       ;W ya 1 değeri yüklendi
        RETLW b'01011011'       ;W ya 2 değeri yüklendi
        RETLW b'01001111'       ;W ya 3 değeri yüklendi
        RETLW b'01100110'       ;W ya 4 değeri yüklendi
        RETLW b'01101101'       ;W ya 5 değeri yüklendi
        RETLW b'01111101'       ;W ya 6 değeri yüklendi
        RETLW b'00000111'       ;W ya 7 değeri yüklendi
        RETLW b'01111111'       ;W ya 8 değeri yüklendi
        RETLW b'01101111'       ;W ya 9 değeri yüklendi
        END
```

Örnek 8: **0 dan 9 kadar olan sayıları PORB uçlarına bağlı 7 segment display'de gösteren programı C dili ile gerçekleştiriniz.**

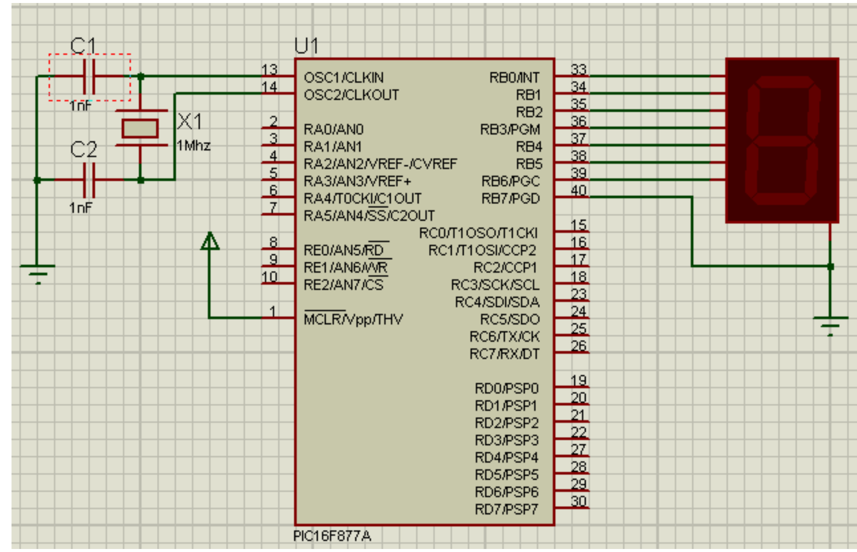
```
// 0 dan 9 a da kadar sayan program
#include <htc.h>
const unsigned char
dizi[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
void bekle(){
   for (int i=0;i<10000;i++) ;
}

void main(void)   // Ana fonksiyon alanı
{
  char i;   // Herhangi bir değisken tanımlanıyor
  TRISB=0x00;   // PORTB çıkıs olarak yönlendiriliyor
  PORTB=0x00;   // PORTB'nin tüm çıkısları sıfırlanıyor
  for(;;)   // Sonsuz döngüye giriliyor
  {
  PORTB=dizi[i];  // Seven segment değerleri alınıyor
   i++;    // i bir artırılıyor
   bekle();
    if(i>9)   // Eğer sayı 9'dan büyük ise 0'a dön
   i=0;   // Deği□ ken 0 yapılıyor
  }
}
```

Örnek 9: 9 Dan 0 A Geri Sayıcı

```
        LIST P=16F84
        INCLUDE "P16F84.INC"
SAYAC   EQU h'0C'
        CLRF PORTB
        CLRF SAYAC
        BSF STATUS,5
        CLRF TRISB
        BCF STATUS,5
BASLA
        INCF SAYAC,F
        MOVF SAYAC,W
        SUBLW d'10'                ;W=10-W
        BTFSS STATUS,Z
        GOTO DISPLAY
        MOVLW h'00'
        MOVWF SAYAC
        GOTO DISPLAY
        GOTO BASLA
DISPLAY
        CALL DIZI
        MOVWF PORTB
        GOTO BASLA
DIZI
        ADDWF PCL,F
        RETLW b'00111111';0GFEDCBA
        RETLW b'00000110'
        RETLW b'01011011'
        RETLW b'01001111'
        RETLW b'01100110'
        RETLW b'01101101'
        RETLW b'01111101'
        RETLW b'00000111';7
        RETLW b'01111111'
        RETLW b'01101111'
        END
```

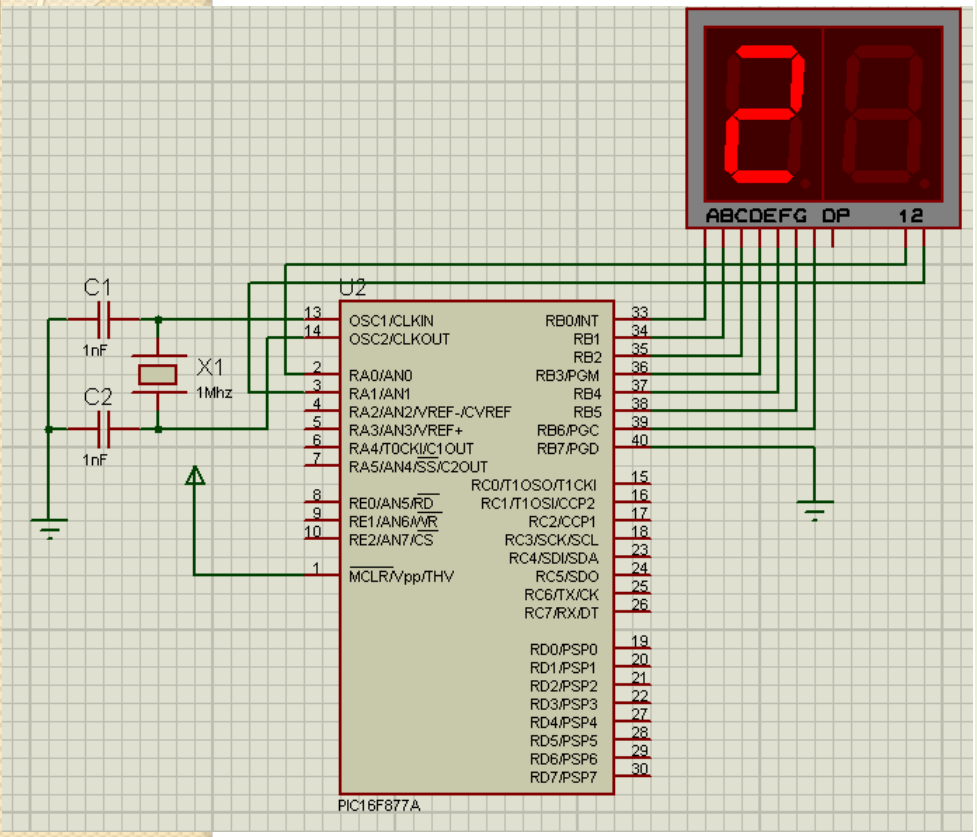
```
// 9 dan 0'a geri sayıcı C kodu
#include <htc.h>
const unsigned char
dizi[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x
7F,0x6F};
void bekle(){
   for (int i=0;i<10000;i++) ;
}
void main(void)   // Ana fonksiyon alanı
{
  char j=9;   // Herhangi bir değisken tanımlanıyor
  TRISB=0x00;   // PORTB çıkıs olarak yönlendiriliyor
  PORTB=0x00;
  for(;;)   // Sonsuz döngüye giriliyor
  {
  PORTB=dizi[j];  // 7 segment değerleri alınıyor
   j--;
   bekle();
    if(j<0)
                {
                 j=9;
                }
  }
}
```

Uygulama-13: 0 dan 9 kadar olan sayıları PORB uçlarına bağlı 7 segment display'de gösteren programı XC8 ile gerçekleştiriniz.



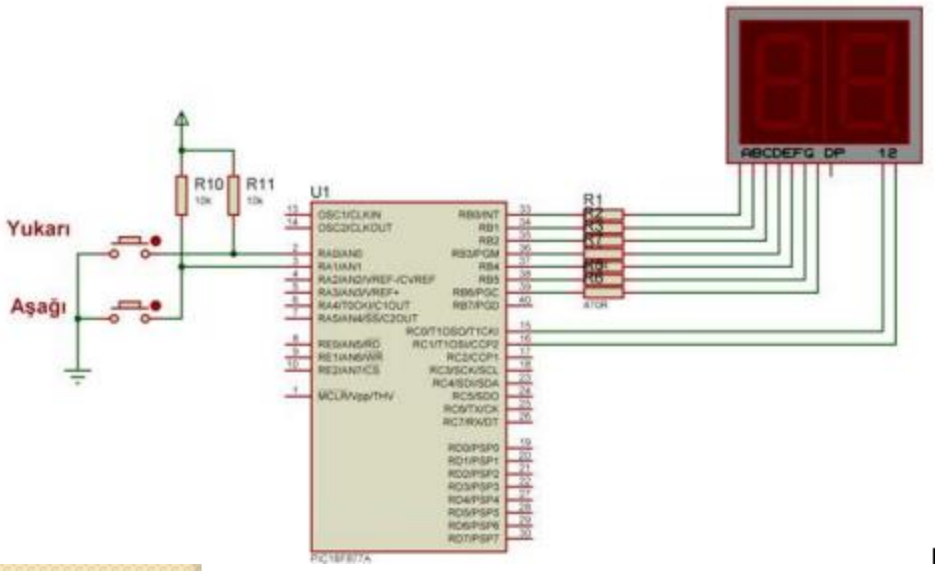

```c
#include <xc.h>
bekle ()
{
    for (int a=0;a<10000;a++);
}
//ortak katotlu display
unsigned char display[]={0x3F,0x06, 0x5B, 0x4F, 0x66,
                         0x6D, 0x7D, 0x07, 0x7F, 0x6F};
void main() {
    TRISB = 0;                          // portb cıkıs
    while (1)
      for (int i = 0; i<=9; i++) {   // 0 dan 9 say
            PORTB = display[i];
            bekle();
      }
}
```

Uygulama 8: 0-99 arası sayıcı




```
#include <xc.h>
#include "delay.h"
int A[]={0x3F,0x06,0x5B,0x4F,0x66,
         0x6D,0X7D,0x07,0x7F,0x6F};
void goster(char i)
{
PORTA=0x02;      // RA1 aktif
PORTB=A[i/10];   // Onlar basamağı
DelayMs(50);     // Bekle
PORTA=0x01;      // RA0 aktif
PORTB=A[i%10];   // Birler basamağı
DelayMs(50);
}
void main()
{
unsigned char i=0;
TRISB=0;
TRISA=0;
PORTA=0;
PORTB=0;
for (;;)
{
          i++;
          if (i>99)
          i=0;
          goster(i);
} }
```

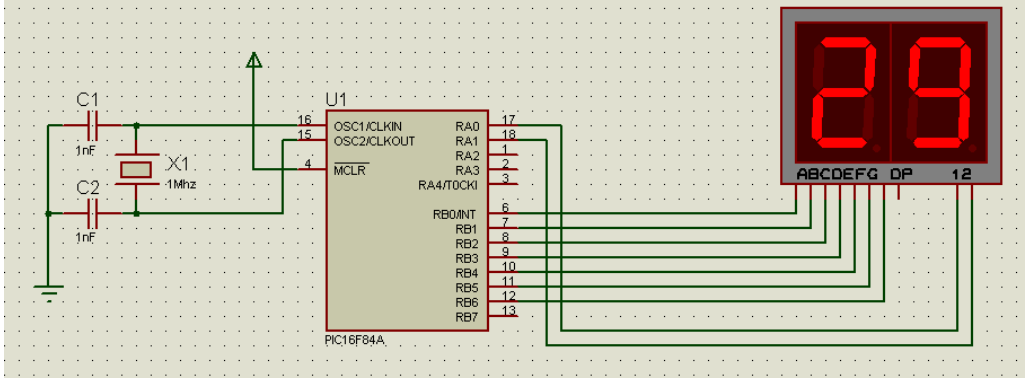
Örnek 11: 00-99 a İleri Sayıcı (Tarama Yöntemi ile)




```
#include <htc.h>
#include "delay.h"   // Gecikme kütüphanesi

const unsigned char
segment[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0
x7F,0x6F};

void sayi_goster(char i)  // Sayıları gösteren fonksiyon
{
 PORTC=0x01;    // PORTC'de 2 değeri gönderiliyor
 PORTB=segment[i/10];  // i'nin 10'a bölümü
  DelayMs(5);    // 5ms bekleniyor
  PORTC=0x02;    // PORTC'de 1 değeri gönderiliyor
  PORTB=segment[i%10];  // i'nin 10'a bölümünden kalan
  DelayMs(5);    // 5ms bekleniyor
}
```

```
void main(void)    // Ana proram
{
  int i;    // Herhangi bir değisken tanımlanıyor
  ADCON1=0x07;   // PORTA dijital yapılıyor
  TRISA=0x03;    // PORTA'nın ilk iki pini giris
  TRISB=0x00;    // PORTB çıkıs olarak yönlendiriliyor
  TRISC=0x00;    // PORTC çıkıs yapılıyor
  PORTA=0x00;    // PORTA'nin tüm çıkısları sıfırlanıyor
  PORTB=0x00;    // PORTB'nin tüm çıkısları sıfırlanıyor
  PORTC=0x00;    // PORTC'nin tüm çıkısları sıfırlanıyor
  for(;;)    // Sonsuz döngüye giriliyor
  {
   if(RA0==0)   // RA0'pini 0 mı?
   {
    while(!RA0); // Buton bırakıldı mı diye bakılıyor
     i++;   // Degisken artırılıyor
     if(i>99)  // Eğer degisken 99'dan büyükse 0 oluyor
     i=0;
   }
     else if(RA1==0)   // RA1'pini 0 mı?
    {
    while(!RA1); // Buton bırakıldı mı diye bakılıyor
    i--;   // Değisken azaltılıyor
     if(i<0)  // Eğer değisken 0'dan küçükse 99 oluyor
     i=99;
   }
   sayi_goster(i);  // O anki sayı gösteriliyor
   }
}
```

Örnek 10: 00-99 a İleri Sayıcı (Timer ile)




```
// 00 dan 99'a ileri sayıcı
#include <htc.h>
#include "delay.h"

unsigned char sayac=0;
unsigned char cnt=0;
/*void bekle(){
   for (int i=0;i<10000;i++) ;
}*/
static void interrupt yeni_olay(void){
cnt++;
if (cnt==10)
{
cnt=0;
sayac++;
}
T0IF=0;//TMR0 bayrağını temizle
TMR0=0;
}

void main(void)   // Ana fonksiyon alanı
{
  const unsigned char
dizi[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
  unsigned char msd,lsd;
  TRISB=0;
  TRISA=0;
                    T0CS=0;//TMR0 kaynağı
                    PSA=0;            // ön bölücü
                    PS0=1;
                    PS1=1;
                    PS2=1;
                    TMR0=0;
                    T0IE=1;
                    T0IF=0;
                    GIE=1;            // Genel kesme izni veriliyor

//programın devamı
for(;;)   // Sonsuz döngüye giriliyor
  {
                    msd=sayac/10; //onlar hanesi
                    lsd=sayac-10*msd; //birler hanesi
                    PORTA=1;      //birleri seç
                    PORTB=dizi[lsd];  // birler hanesini göster
                    DelayMs(10);
                    PORTA=2;      //onları seç
                    PORTB=dizi[msd];  // birler hanesini göster
                    DelayMs(10);

  }
}
```

Uygulama-3: 0 dan 9 kadar olan sayıları PORB uçlarına bağlı 7 segment display'de gösteren programı Assembly ile gerçekleştiriniz.

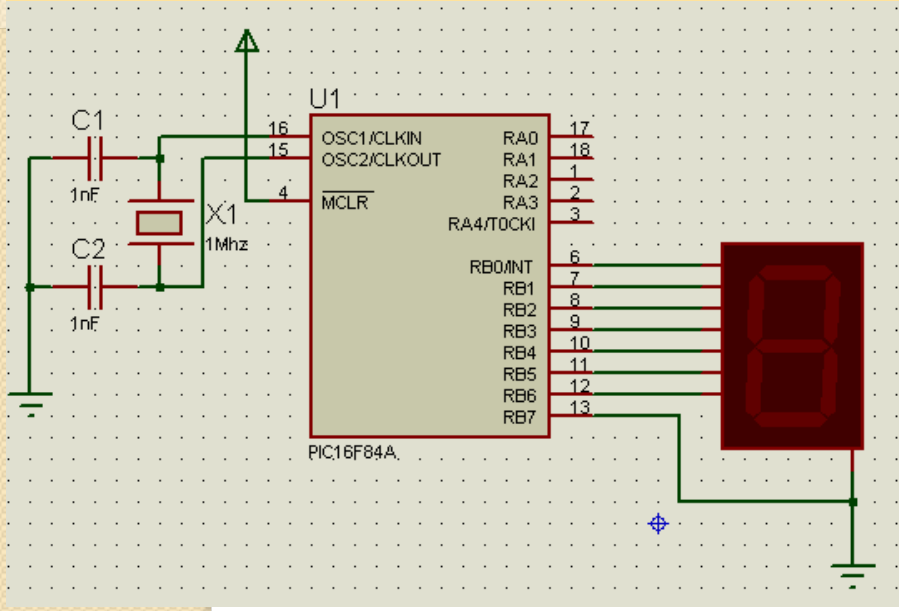
```
            LIST P=16F84A
            #INCLUDE <P16F84A.INC>
            SAYAC1 EQU h'0D'
            BSF     STATUS,5        ;BANK1 e geçiş yap
            CLRF TRISB              ;PORTB nin tüm uçları çıkış seçildi
            BCF     STATUS,5        ;BANK0 a geçiş yap
            CLRF    PORTB           ;PORTB yi temizle
Basla
            MOVLW   h'00'           ;W kaydedicisine h'00' değerini yükle
            MOVWF   SAYAC1
DON         MOVF    SAYAC1,W
            CALL    DIZI
            MOVWF   PORTB           ; W içeriğini PORTB ye aktar
            INCF    SAYAC1,F        ; SAYAC1 değerini artır
            GOTO    DON
DIZI
            ADDWF PCL, F            ;W içeriğini PCL ye aktar
            RETLW b'00111111'       ;W ya 0 değeri yüklendi
            RETLW b'00000110'       ;W ya 1 değeri yüklendi
            RETLW b'01011011'       ;W ya 2 değeri yüklendi
            RETLW b'01001111'       ;W ya 3 değeri yüklendi
            RETLW b'01100110'       ;W ya 4 değeri yüklendi
            RETLW b'01101101'       ;W ya 5 değeri yüklendi
            RETLW b'01111101'       ;W ya 6 değeri yüklendi
            RETLW b'00000111'       ;W ya 7 değeri yüklendi
            RETLW b'01111111'       ;W ya 8 değeri yüklendi
            RETLW b'01101111'       ;W ya 9 değeri yüklendi
            END
```

Uygulama-4: 0 dan 9 kadar olan sayıları PORB uçlarına bağlı 7 segment display'de gösteren timer gecikmeli (ileri sayıcı) programı gerçekleştiriniz.

```
        LIST P=16F877
        #INCLUDE<P16F877.INC>
SAYAC   EQU h'20'
        CLRF PORTB
        BSF STATUS,5 ; BANKSEL TRISB
        CLRF TRISB
        MOVLW b'11010111'
        MOVWF OPTION_REG
        BCF STATUS,5 ; BANKSEL PORTB
BASLA
        MOVLW .0
        MOVWF SAYAC
DON
        MOVF SAYAC,W
        CALL DIZI
        CALL GECIKME
        MOVWF PORTB
        INCF SAYAC,F
        CALL GECIKME
        GOTO DON
```

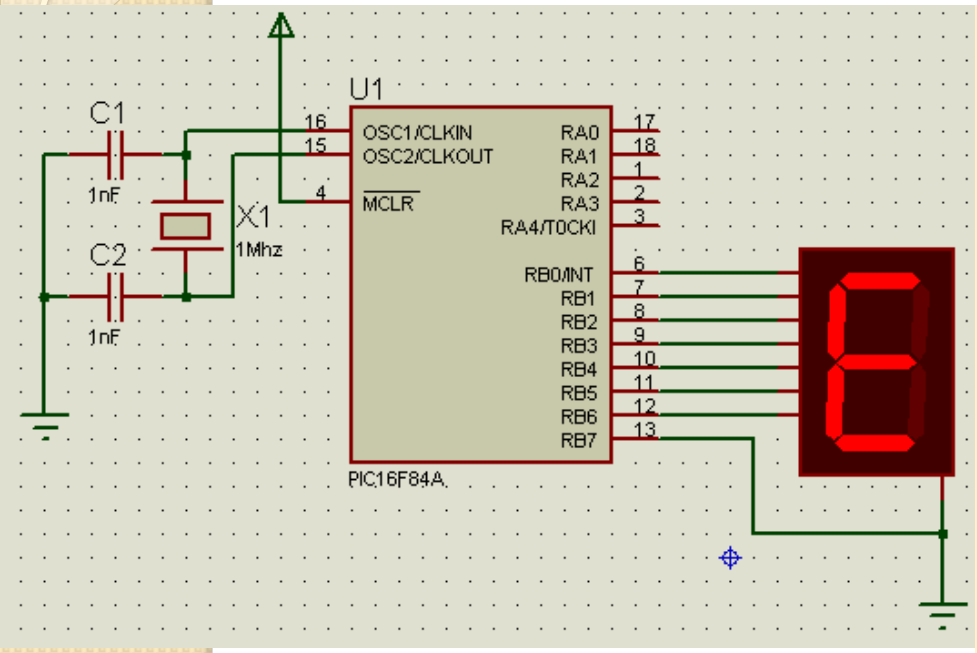
```
DIZI
        ADDWF PCL,F
        RETLW b'00111111'
        RETLW b'00000110'
        RETLW b'01011011'
        RETLW b'01001111‘
        RETLW b'01100110'
        RETLW b'01101101'
        RETLW b'01111101'
        RETLW b'00000111'
        RETLW b'01111111'
        RETLW b'01101111‘
GECIKME
        CLRF TMR0
DON1
        BTFSS INTCON, T0IF
        GOTO DON1
        BCF INTCON, T0IF
        RETURN
        END
```

Uygulama 5: 0 dan F İleri Sayıcı Gecikmeli




```
        LIST P=16F84
        INCLUDE "P16F84.INC"
SAYAC   EQU h'0C'
SAYAC2  EQU h'0D'
SAYAC3  EQU h'0E'
        CLRF PORTB
        BSF STATUS,5
        CLRF TRISB
        BCF STATUS,5
        CLRF SAYAC              ;sayac=0
DON
        MOVF SAYAC,W
        CALL DIZI
        MOVWF PORTB
        CALL BEKLE
        INCF SAYAC,F            ;sayac++
        GOTO DON
BEKLE
        MOVLW h'FF'
        MOVWF SAYAC2
DON1
        MOVLW h'FF'
        MOVWF SAYAC3
DON2
        DECFSZ SAYAC3,F
        GOTO DON2
        DECFSZ SAYAC2,F
        GOTO DON1
        RETURN
DIZI
        ADDWF PCL,F
        dt h'3F',h'06',h'5B',h'4F',h'66',h'6D',
        dt h'7D',h'07',h'7F',h'6F',h'77',h'7C',
        dt h'39',h'5E',h'79',h'71'
        END
```

Uygulama 7: F den 0 a kadar Geri Sayıcı




```
                LIST P=16F84
                INCLUDE "P16F84.INC"
SAYAC           EQU h'0C'
                CLRF PORTB
                CLRF SAYAC
                BSF STATUS,5
                CLRF TRISB
                BCF STATUS,5
BASLA
                INCF SAYAC,F
                MOVF SAYAC,W
                SUBLW d'16'  ;W=10-W
                BTFSS STATUS,Z
                GOTO DISPLAY
                MOVLW h'00'
                MOVWF SAYAC
                GOTO DISPLAY
                GOTO BASLA
DISPLAY
                CALL DIZI
                MOVWF PORTB
                GOTO BASLA
DIZI
                ADDWF PCL,F
                dt h'3F',h'06',h'5B',h'4F',h'66',h'6D',
                dt h'7D',h'07',h'7F',h'6F',h'77',h'7C',
                dt h'39',h'5E',h'79',h'71'
                END
```

Uygulama 8: Trafik Işığı Program Parçası

```
BASLA   CLRW
        MOVWF   DURUM
DON
        CALL    SINYAL      ; DURUMU DEĞİŞTİR.
        MOVWF   PORTB       ; SINYAL DEĞERINI PORTB DE GOSTER
        INCF    DURUM,W     ; DURUMU BİR ARTIR, SONUCU W YA YAZ.
        ANDLW   0X03        ; MAKSIMUM 3 'E KADAR ARTIR.
        MOVWF   DURUM       ; W İÇERİĞİNİ DURUM DEĞİŞKENİ
        CALL    GECIKME     ; BEKLE :-)
        GOTO    DON
SINYAL
        MOVF    DURUM,W     ; DURUMU W YA TAŞI.
        ADDWF   PCL,F
        RETLW   0X41        ; DURUM==0 İSE YEŞİL VE KIRMIZI(RB6,RB0)
        RETLW   0X23        ; DURUM==1 İSE  SARI VE KIRMIZI/SARI (RB5, RB0/RB1)
        RETLW   0X14        ; DURUM==2 İSE  KIRMIZI VE YEŞİL (RB4,RB2)
        RETLW   0X32        ; DURUM==3  İSE KIRMIZI/SARI VE SARI (RB4/RB5, RB1
```

K
S
Y

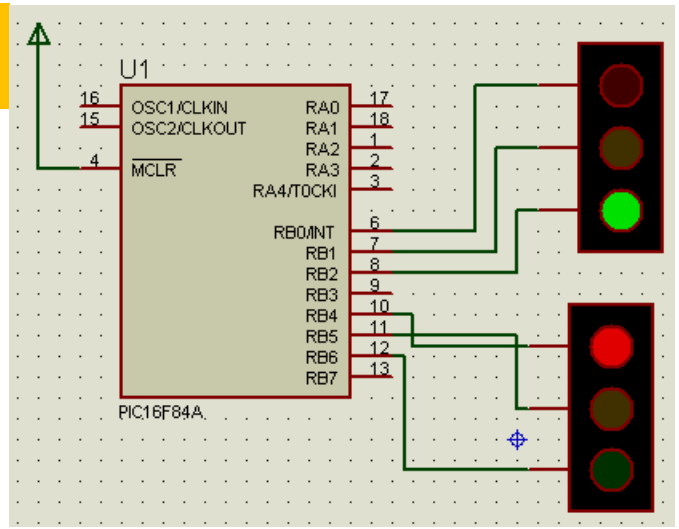
K
S
Y

	B7	B6	B5	B4	B3	B2	B1	B0
	0	Y	S	K	0	Y	S	K
41		I						I
23			I				I	I
14				I		I		
32			I	I			I	

Örnek Uygulama 8: Trafik Işığı Programı Tamamı

```
    LIST P=16F84
    INCLUDE "P16F84.INC"
DURUM EQU h'0C'
    CLRF PORTB
    BSF STATUS,5
    CLRF TRISB
    BCF STATUS,5
BASLA
    CLRW
    MOVWF    DURUM
DON
    CALL    SINYAL  ; DURUMU DEĞİŞTİR.
    MOVWF   PORTB   ; SINYAL DEĞERINI PORTB DE GOSTER
    INCF    DURUM,W  ; DURUMU BİR ARTIR, SONUCU W YA YAZ.
    ANDLW   0X03     ; MAKSIMUM 3 'E KADAR ARTIR.
    MOVWF   DURUM    ; W İÇERİĞİNİ DURUM DEĞİŞKENİNE AKTAR
    NOP              ; bekle alt programı konmalı
    GOTO    DON
SINYAL
    MOVF    DURUM,W    ; DURUMU W YA TAŞI.
    ADDWF   PCL,F
    RETLW   0X41       ; DURUM==0 İSE YEŞİL VE KIRMIZI(RB6,RB0)
    RETLW   0X23       ; DURUM==1 İSE  SARI VE KIRMIZI/SARI (RB5, RB0/RB1)
    RETLW   0X14       ; DURUM==2 İSE  KIRMIZI VE YEŞİL (RB4,RB2)
    RETLW   0X32       ; DURUM==3 İSE KIRMIZI/SARI VE SARI (RB4/RB5, RB1
    END
```

```
#include <xc.h>
void main()
{
unsigned char A[]={0X41, 0X23, 0X14, 0X32 };
TRISB=0;
PORTB=0;
while(1)
{
for (int i=0; i<4;i++)
{
  PORTB=A[i];
} } }
```



	B7	B6	B5	B4	B3	B2	B1	B0
	0	Y	S	K	0	Y	S	K
41		I						I
23			I				I	I
14				I		I		
32			I	I			I	

Kesmeler

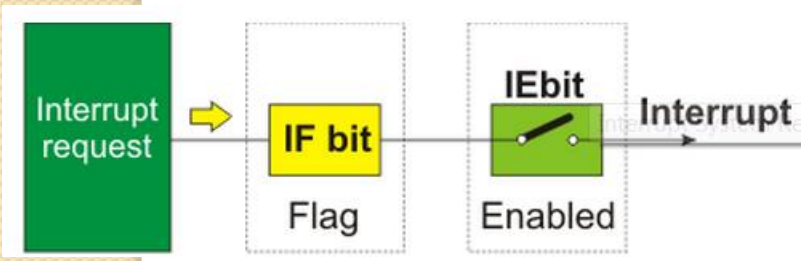
Kesme (Interrupt), mikro denetleyicinin gerçekleştirdiği işleme bakmaksı zın belirli durumların/olayların olması durumunda isteklere / olaylara ce vap verilmesini sağlayan mekanizmadır. Bu mekanizma, mikro denetleyi ci ile çevre birimleri arasındaki bağlantıları oluşturması ve ilişkileri düzenl emesi nedeniyle çok önemli bir yere sahiptir.

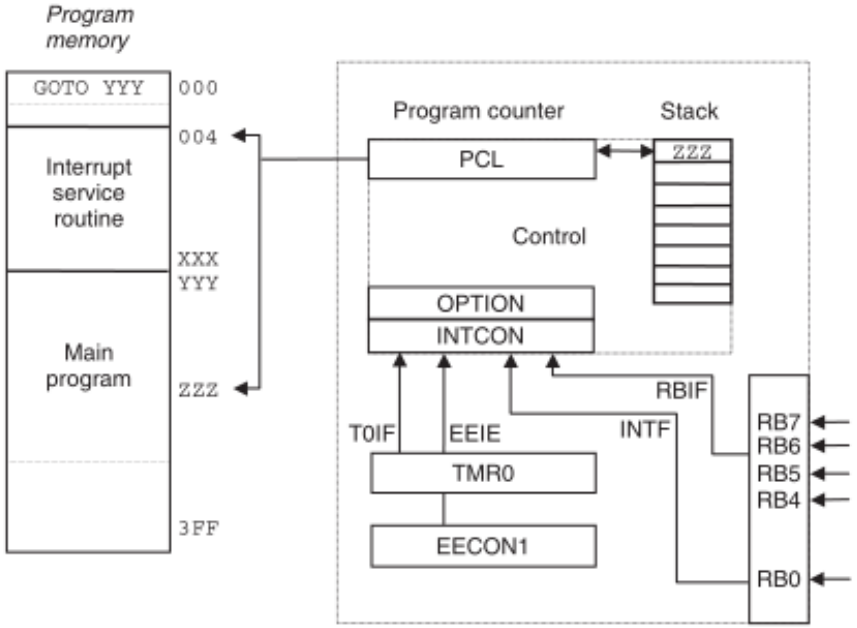
Oluşan her kesme programı ile programın normal işlenme süreci değiştirilerek program durdurulur ve kesme ile ilgili rutin/altprogram gerçekleştirildikten sonra ana programın işlenmesi kalınan noktadan devam edilir

PIC16F84 mikro denetleyicisi dört farklı kaynaktan kesme alabilir. Bunlar;

Interrupt Source	Enabled by	Completion Status
External interrupt from INT	INTE = 1	INTF = 1
TMR0 interrupt	T0IE = 1	T0IF = 1
RB4–RB7 state change	RBIE = 1	RBIF = 1
EEPROM write complete	EEIE = 1	–

Kesmeler





Interrupt control bit functions

	Bit	Label	Function	Settings
INTCON	0	**RBIF**	Port B (4:7) Interrupt flag	0 = No change 1 = Bit change detected
	1	**INTF**	RB0 Interrupt flag	0 = No interrupt 1 = Interrupt detected
	2	**T0IF**	TMR0 overflow Interrupt flag	0 = No overflow 1 = Overflow detected
	3	**RBIE**	Port B (4:7) Interrupt enable	0 = Disabled 1 = Enabled
	4	**INTE**	RB0 Interrupt enable	0 = Disabled 1 = Enabled
	5	**T0IE**	TMR0 overflow Interrupt enable	0 = Disabled 1 = Enabled
	6	**EEIE**	EEPROM write complete interrupt enable flag	0 = Disabled 1 = Enabled
	7	**GIE**	Global interrupt enable	0 = Disabled 1 = Enabled
OPTION	6	**INTEDG**	RB0 interrupt Active edge select	0 = Falling edge 1 = Rising edge

RB0/INT Pini Harici Kesme Örneği

RB0/INT pini harici kesmesi kenar tetiklemelidir. Yani bu uçtaki sinyalin 1 den 0'a veya 0 dan 1'e geçişi kesmeye sebep olur. Kesmenin yükselen kenarda mı yoksa düşen kenarda mı gerçekleşeceğine programcı karar verir. Bunun için OPTION_REG kaydedicisinin INTEDG biti kullanılır. INTEDG biti 1 ise kesme yükselen kenarda, 0 ise düşen kenarda gerçekleşir.

INT kesmesini kullanabilmek için INTCON kaydedicisinin INTE biti 1 yapılarak kesmeye izin verilmelidir. INT kesmesi oluştuğunda INTCON kaydedicisinin INTF biti 1 olur.

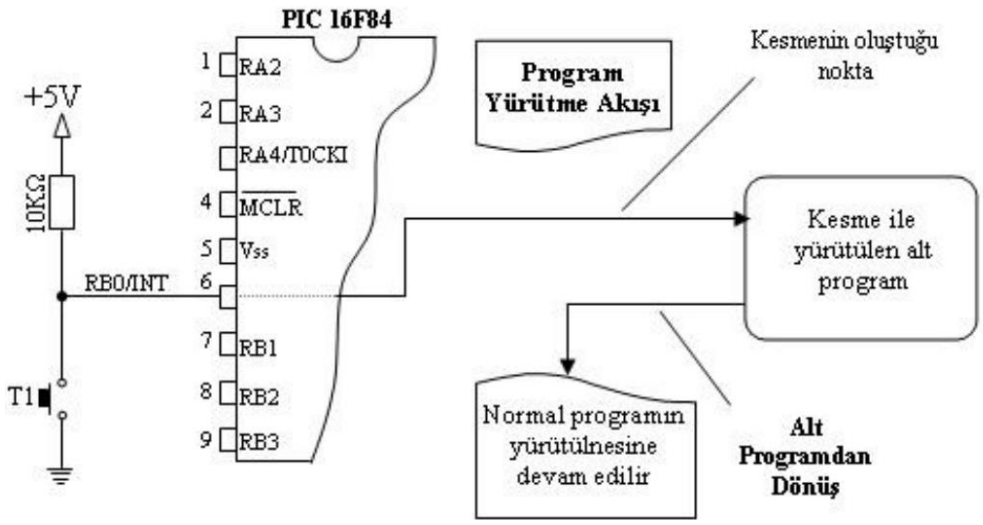
Programda kesme alt programı içerisinde INTF=0 yapılmalıdır.

OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0



Uygulama-12:

PORTB nin RB0/INT ucundan gelen bir kesme gerçekleşince kesme alt programında PORTA ya bağlı LED'leri yakan programı yazalım

Program Algoritması:

- RB0/INT ucunu giriş olarak seç ,
- OPTION_REG kaydedicisinin INTEDG biti ile düşen veya yükselen kenar tetiklemesini seç ,
- INTCON kaydedicisinin GIE ve INTE bitleri 1 yapılarak kesmeye izin verilir,
- Kesme oluşup, program kesme alt programına dallandığında INTF bitini 0 yap.

```
        ;==KESME1.ASM ===
        LIST P=16F84            ;PIC16F84 seçildi
        #INCLUDE<P16F84.INC>    ;PIC16F84.INC dosyasını programa dahil et
        ORG H'00'               ;ANA PROGRAM '00' adresindeki
        GOTO BASLA              ;BASLA etiketinden başlıyor
        ORG H'04'               ;KESME alt programı '04' adresindeki
        GOTO KESME              ;KESME etiketinden başlıyor
BASLA
        BSF STATUS,5            ;BANK1 e geçiş yap
        MOVLW H'FF'
        MOVWF TRISB             ;PORTB giriş,
        CLRF TRISA              ;PORTA çıkış seçildi
        BCF OPTION_REG,6        ;RB0/INT ucunu düşen kenar tetiklemeli seç

        BCF STATUS,5            ;BANK0 a geçiş yap
        CLRF PORTA              ;PORTA ya bağlı ledleri söndür
        BSF INTCON, INTE        ;INT kesmesine izin ver
        BSF INTCON, GIE         ;Tüm kesmelere izin ver
DON
        NOP
        GOTO DON                ;Ledleri sürekli sönük tut
KESME
        BCF INTCON, INTF        ;PORTB INT kesmesine ait biti/bayrağı sıfır yap
        MOVLW H'FF'
        MOVWF PORTA             ; PORTA ya bağlı ledleri yak
        RETFIE                  ;Kesme alt programından ana programa dön
        END                     ;Programı sonlandır
```

PORTB Değişim Kesmesi Örneği

PORTB nin 4 ve 7. bitlerinde (RB4-RB7) bitlerinde bir değişim meydana gelmesi PORTB değişim kesmesine sebep olur. Bu kesmeyi aktif hale getirmek için INTCON kaydedicisinin RBIE bitinin 1 yapılması gerekir. PORTB değişim kesmesi oluştuğunda RBIF=1 olur ve program kesme alt programına dallanır. Program RBIF bitini otomatik sıfırlayamadığı için kesme alt programı içerisinde RBIF=0 yapılır.

UYGULAMA - 9 : PORTB nin RB4-RB7 uçlarına bağlı butonlardan bir veya bir kaçına basıldığında PORTA'nın ilk 4 bitini yakan program.

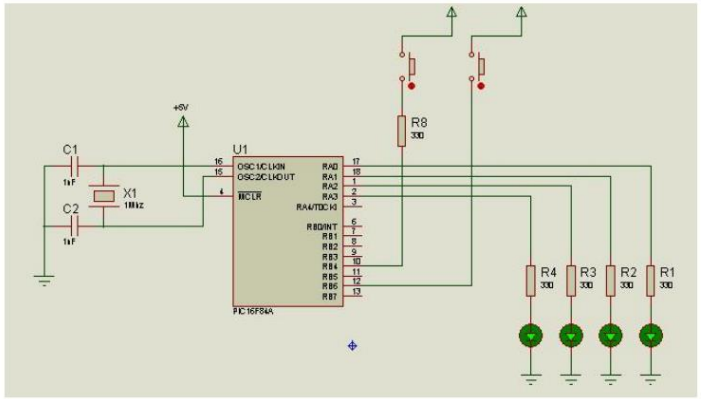
Program Algoritması:

- PORTB nin 4, 5, 6 ve 7. Bitlerini giriş olarak seç ,
- INTCON kaydedicisinin GIE ve RBIE bitini 1 yaparak PORTB değişim kesmesine izin ver.
- Kesme oluşup, program kesme alt programına dallandığında RBIF bitini 0 yap

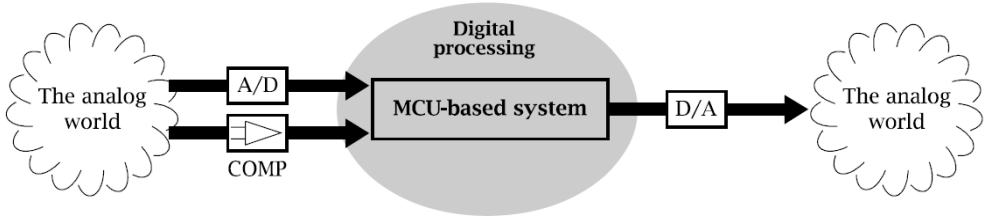
Uygulama-13: PORTB nin RB4-RB7 uçlarından gelen bir kesme gerçekleşince kesme alt programında PORTA ya bağlı LED'leri yakan programı yazalım

```
                LIST P=16F84
                INCLUDE "P16F84.INC"
                CBLOCK H'0C'
                SAYAC1,SAYAC2
                ENDC
                ORG 0X00
                GOTO BASLA
                ORG 0X04
                GOTO KESME
BASLA
                BSF     STATUS,5    ;BANK1 e geçiş yap
                CLRF TRISA          ;PORTA çıkış seçildi
                MOVLW 0XFF
                MOVWF TRISB         ;PORTB GIRIŞ
                BCF     STATUS,5    ;BANK0 a geçiş yap
                CLRF    PORTA       ;PORTA ya bağlı ledleri söndür
                BSF     INTCON,RBIE ;RB değişim kesmesine izin ver
                BSF     INTCON,GIE  ;Tüm kesmelere izin ver
                GOTO BASLA
KESME
                MOVLW 0X0F
                MOVWF PORTA
                CLRF INTCON
                CALL BEKLE
                RETFIE
BEKLE
                MOVLW 0X05
                MOVWF SAYAC1
DON1
                MOVLW 0X06
                MOVWF SAYAC2
DON2
                DECFSZ SAYAC2,F
                GOTO DON1
                DECFSZ SAYAC1,F
                GOTO DON2
                RETURN
                END
```

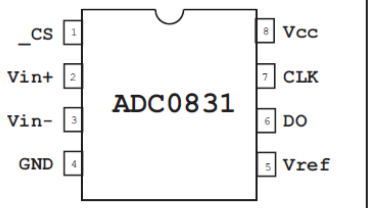
Devre Şeması:



ADC-DAC

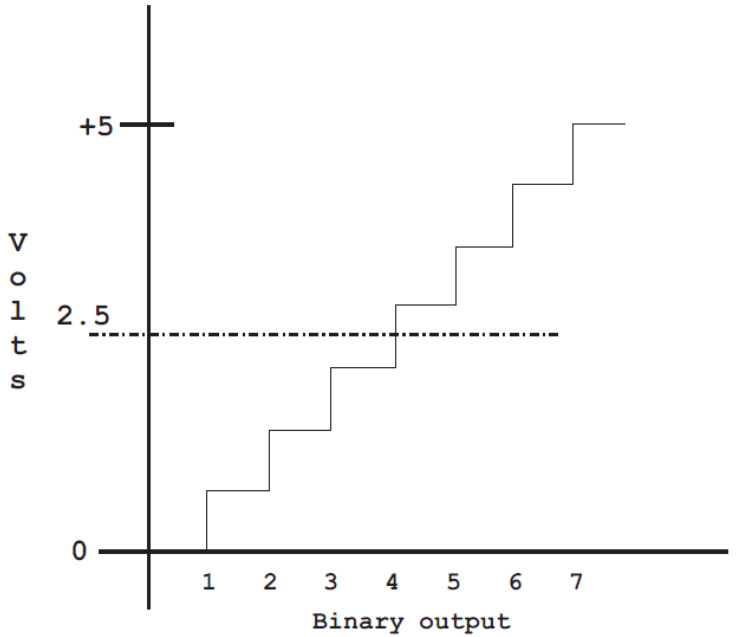


ADC birimi yoksa ADC0831 entegresini kullanarak dönüştürme yapabilirsiniz.




```
ADC0831 PINOUT

 _CS -  Chip Select (active low)
Vin+ - Analog voltage input +
Vin- - Analog voltage input -
 GND - Ground
Vref - Voltage reference
  DO - Data out
 CLK - Clock signal
 Vcc - +5V power
```

ADC0831, Üç kontrol hattı, DO (data out), CLK (clock), _CS (chip seçim) kullanır.

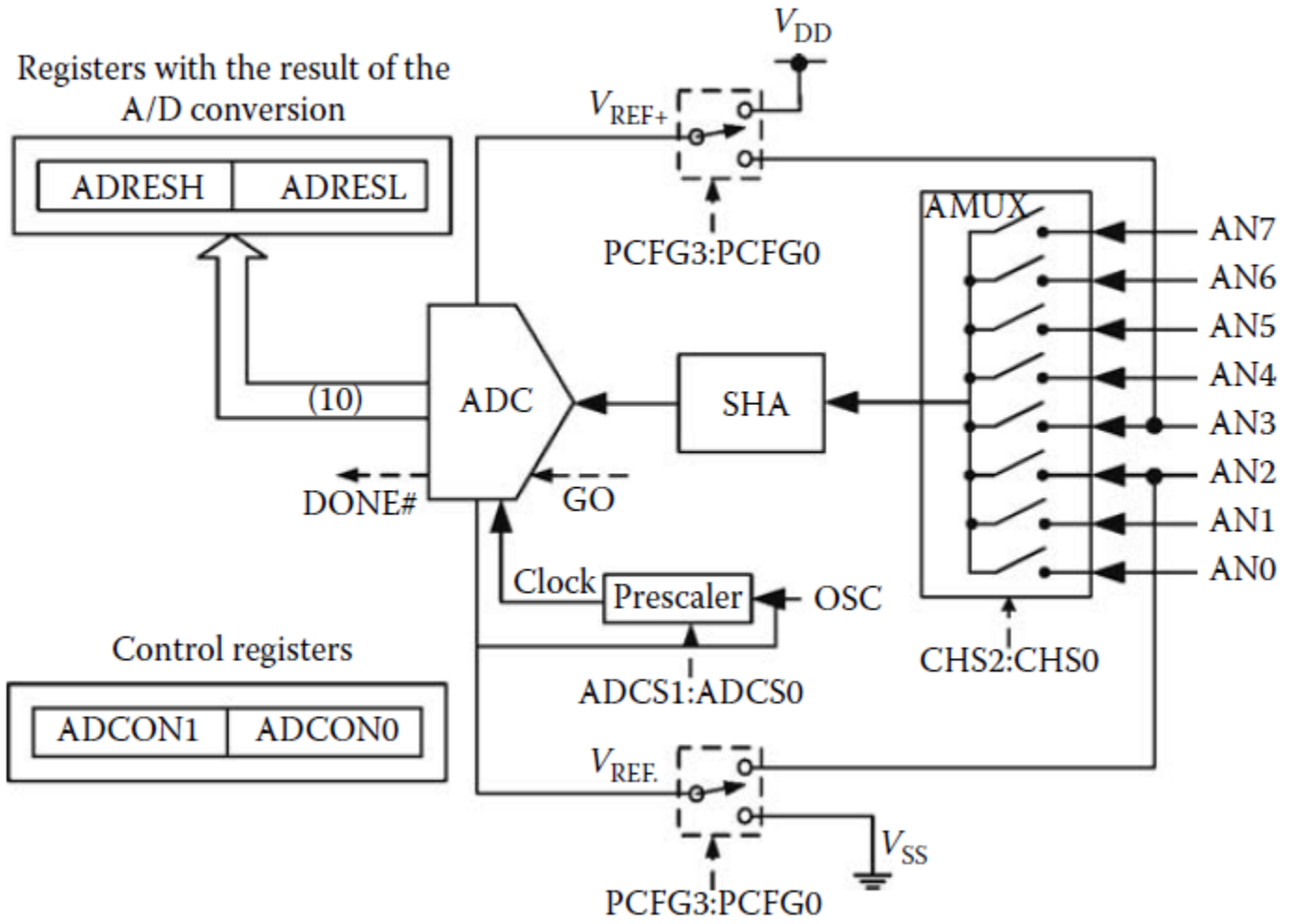


Veya
Vin=Vo*adım buyuklugu

Adım buyuklugu=5/2^10=4.88

$$voltage\ resolution = \frac{5}{255} = 0.01960\,volts = 19.60mV$$

Orta Ölçekli PIC ADC Yapısı

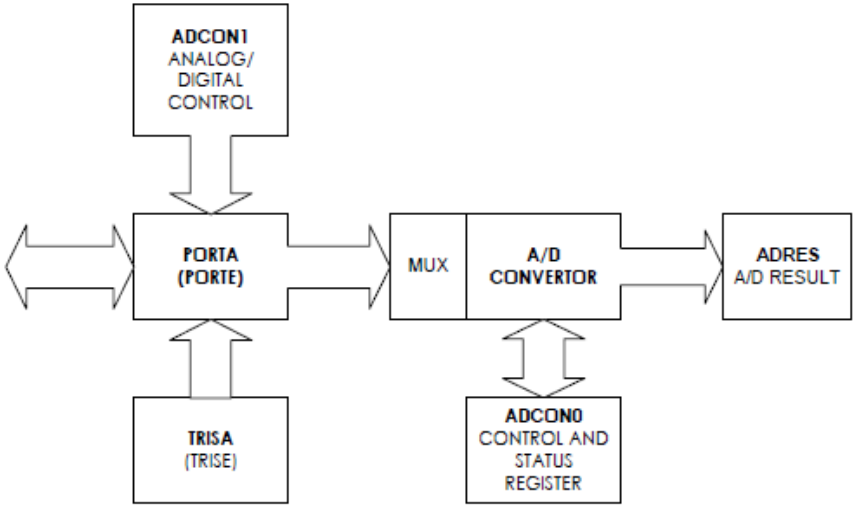


PIC16f877 ADC yapısı

CHS2, CHS1, CHS0 : Kanal seçme bitleridir.

000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock



GO/DONE : ADC çevrimini başlatan bittir, çevrim bitince 0 olur.
ADON : ADC birimini açan bittir.
ADCS1, ADCS0 : Frekans kaynağı seçim bitleridir

ADFM: ADRES kaydedicisi formatını belirler
ADCS2 : Yardımcı frekans seçme biti.
PCFG3, PCFG2, PCFG1, PCFG0 : Portların görevlerini ayarlayan bitler.

ADCON1 REGISTER (ADDRESS 9Fh)

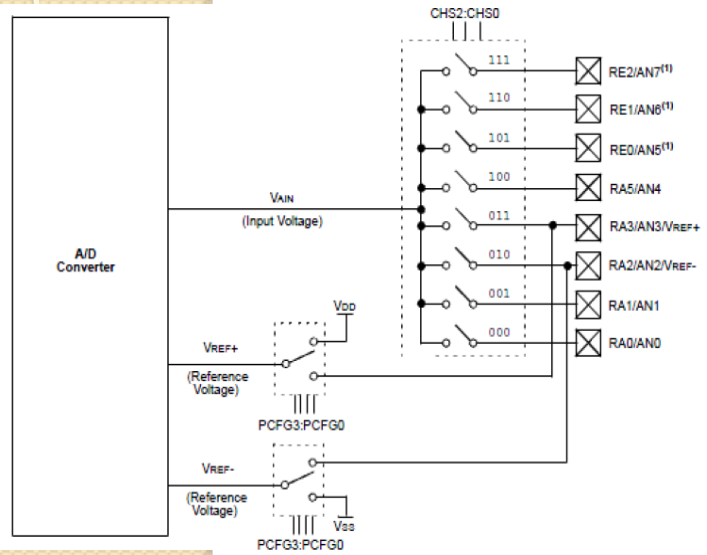
R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

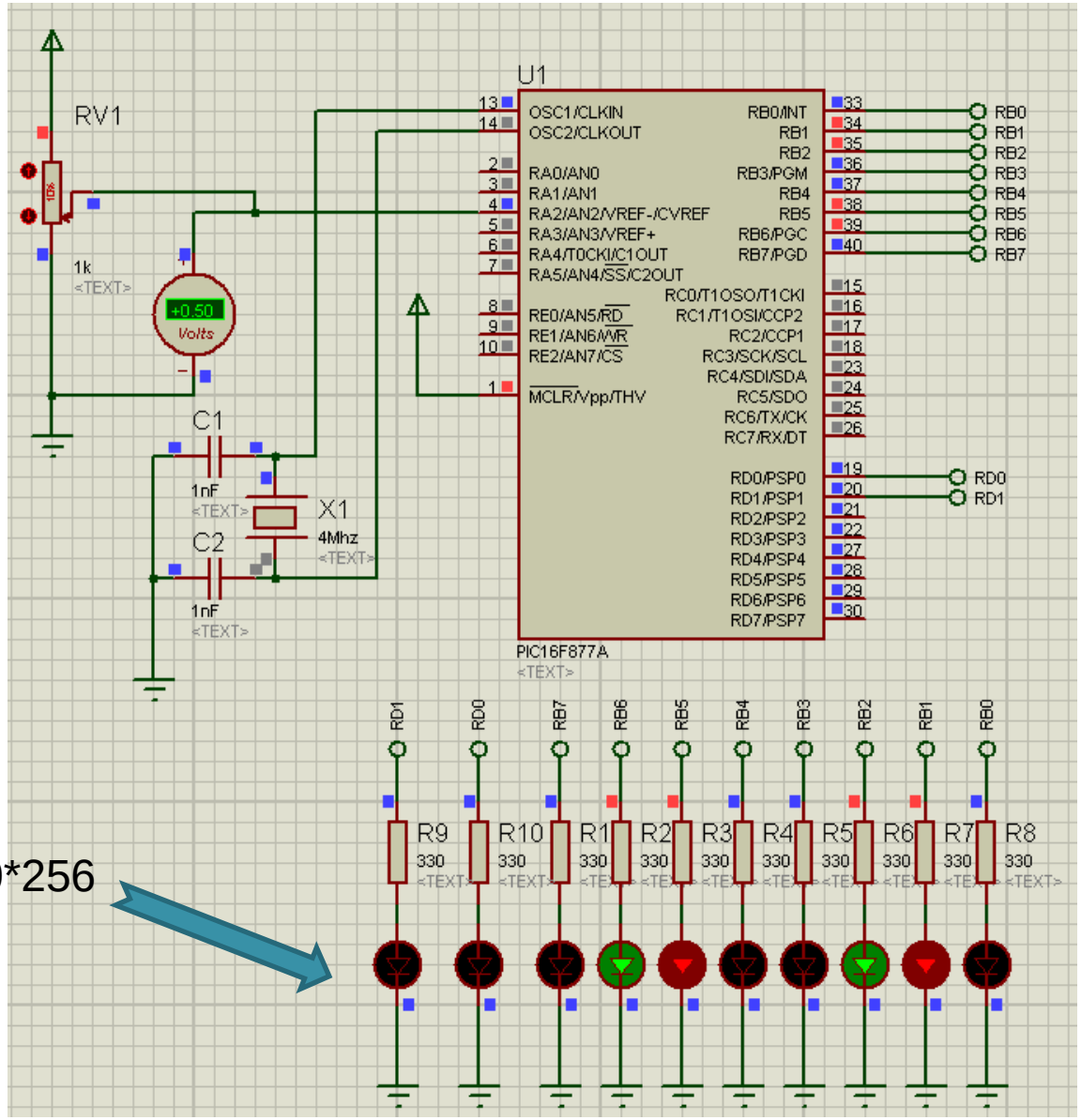
ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

AD Çevirici





Vo= Vin*4/5.0*256

AD Çevirici

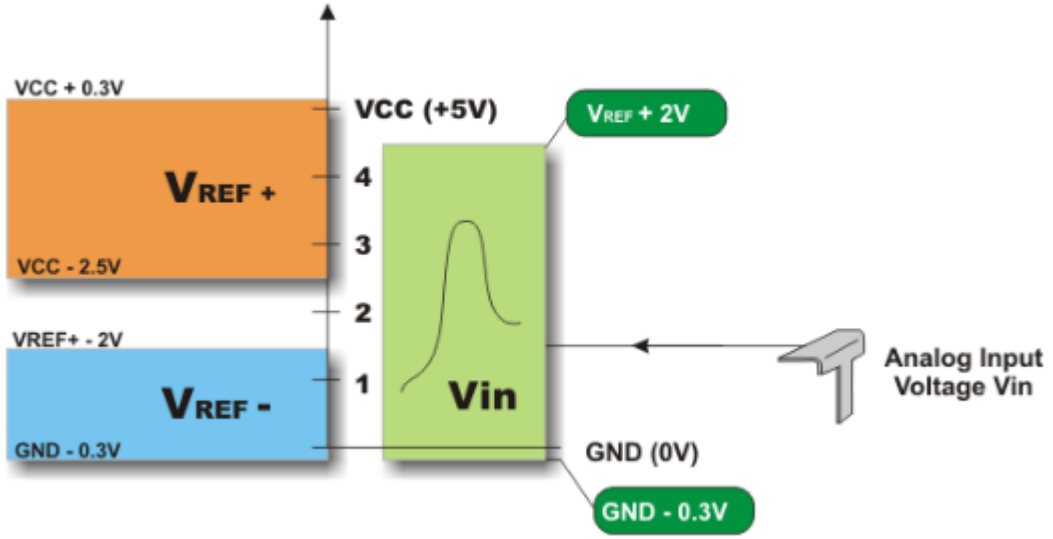
```
#include <xc.h>
void main() {
TRISA = 0xFF; // PORTA input
TRISD = 0x00; // portd çıkış
TRISB = 0; // PORTB çıkış
PORTC=0;
PORTD=0;
ADCON1=0b11000000; // AN ler analog, ADFM=1,ADcs2=1
ADCON0=0b00010001; //fosc/4, AN2, GODONE=0,
ADON=1
for(;;){
ADCON0bits.GO=1; // çevrime başla
while (ADCON0bits.nDONE); //bitene kadar bekle
PORTB = ADRESL; // ilk 8 bit PORTB e gönder
PORTD = ADRESH; //Son iki biti (MSB leri) PORTD e gönder
}
}
```

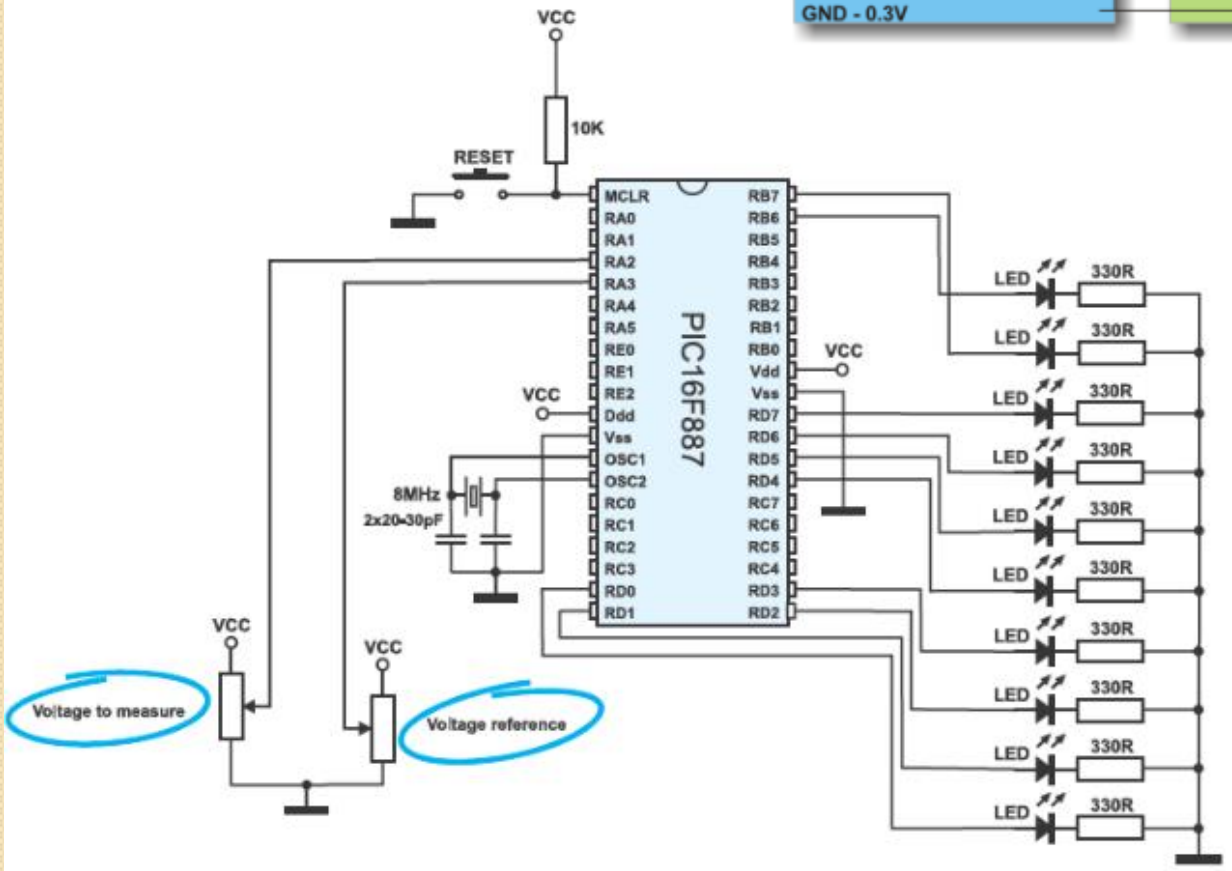
Micro C Kodu

```
unsigned int temp_res;

void main() {
  ANSEL = 0x0C;          // Pins AN2 and AN3 are configured as analog
  TRISA = 0xFF;          // All port A pins are configured as inputs
  ANSELH = 0;            // Rest of pins is configured as digital
  TRISB = 0x3F;          // Port B pins RB7 and RB6 are configured as
                         // outputs
  TRISD = 0;             // All port D pins are configured as outputs
  ADCON1.F4 = 1 ;        // Voltage reference is brought to the RA3 pin.

  do {
    temp_res = ADC_Read(2); // Result of A/D conversion is copied to temp_res
    PORTD = temp_res;     // 8 LSBs are moved to port D
    PORTB = temp_res >> 2;  // 2 MSBs are moved to bits RB6 and RB7
  } while(1);             // Endless loop
}
```

(1: Sağa dayalı, 0: Sola dayalı) ADFM biti

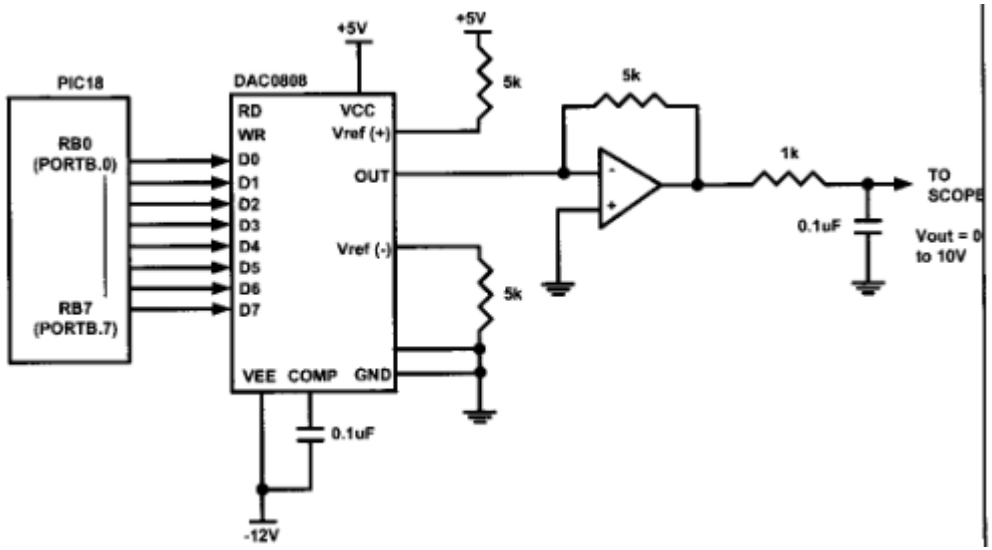
Soru:


VCC + 0.3V
VCC (+5V)
VREF + 2V
VREF +
VCC - 2.5V
4
3
2
VREF+ - 2V
VREF -
1
Vin
Analog Input
Voltage Vin
GND - 0.3V
GND (0V)
GND - 0.3V


VCC
10K
RESET
MCLR
RA0
RA1
RA2
RA3
RA4
RA5
RE0
RE1
RE2
Ddd
Vss
OSC1
OSC2
RC0
RC1
RC2
RC3
RD0
RD1
PIC16F887
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
Vdd
Vss
RD7
RD6
RD5
RD4
RC7
RC6
RC5
RC4
RD3
RD2
8MHz
2x20-30pF
LED
330R
Voltage to measure
Voltage reference

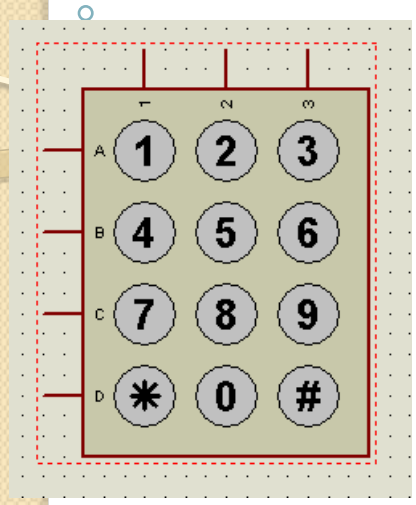
DAC Uygulaması

```
#include <pic.h>
const unsigned char
deger[]={128,192,238,255,
238,192,128,64,17,0,17,64,128};

void main()
{
unsigned char x;
TRISB=0;
while(1)
{
 for (x=0;x<12;x++)
         PORTB= deger[x];
}
}
```



Angle θ (degrees)	Sin θ	V_{out} (Voltage Magnitude) 5 V + (5 V × sin θ)	Values Sent to DAC (decimal) (Voltage Mag. × 25.6)
0	0	5	128
30	0.5	7.5	192
60	0.866	9.33	238
90	1.0	10	255
120	0.866	9.33	238
150	0.5	7.5	192
180	0	5	128
210	–0.5	2.5	64
240	–0.866	0.669	17
270	–1.0	0	0
300	–0.866	0.669	17
330	–0.5	2.5	64
360	0	5	128

TUŞ TAKIMI UYGULAMASI

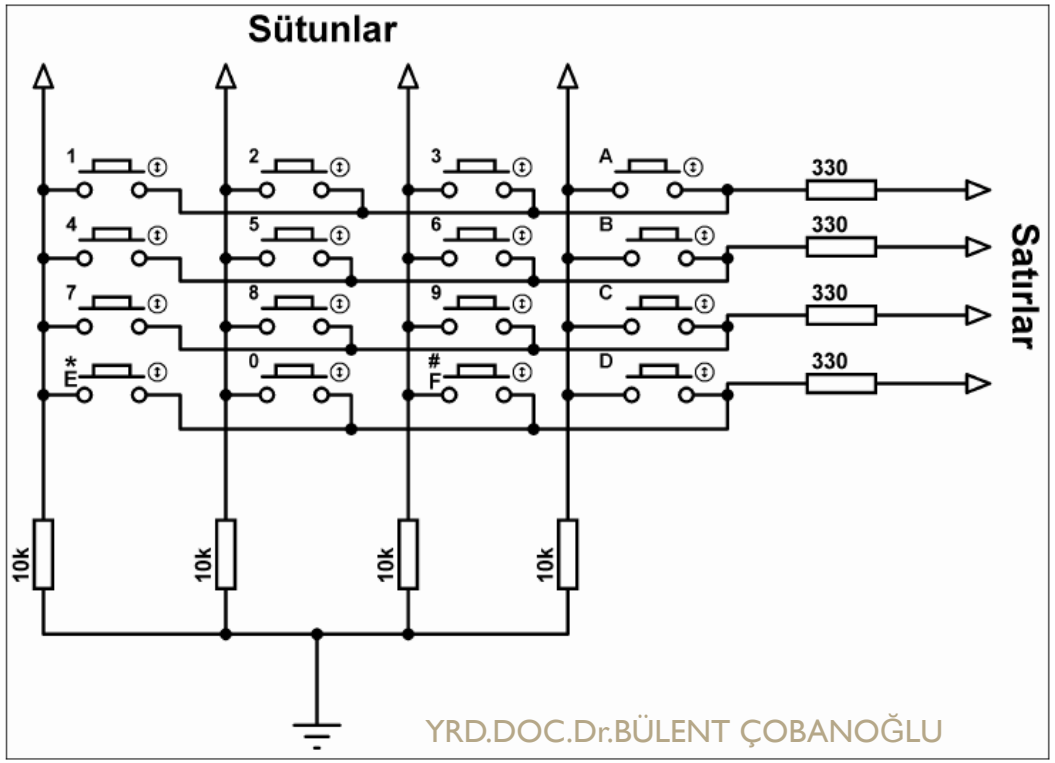


Kontrol sistemlerinde dış dünyadan insanlar tarafından veri girişleri genellikle tuş takımı (keypad-klavye) ile yapılır. Tuş takımı butonlarla gerçekleştirilebileceği gibi çeşitli hazır tuş takımları piyasada bulunmaktadır. Örneğin 4x3'lük bir keypad, 4 satır ve 3 sutundan oluşan bir tuş takımıdır.

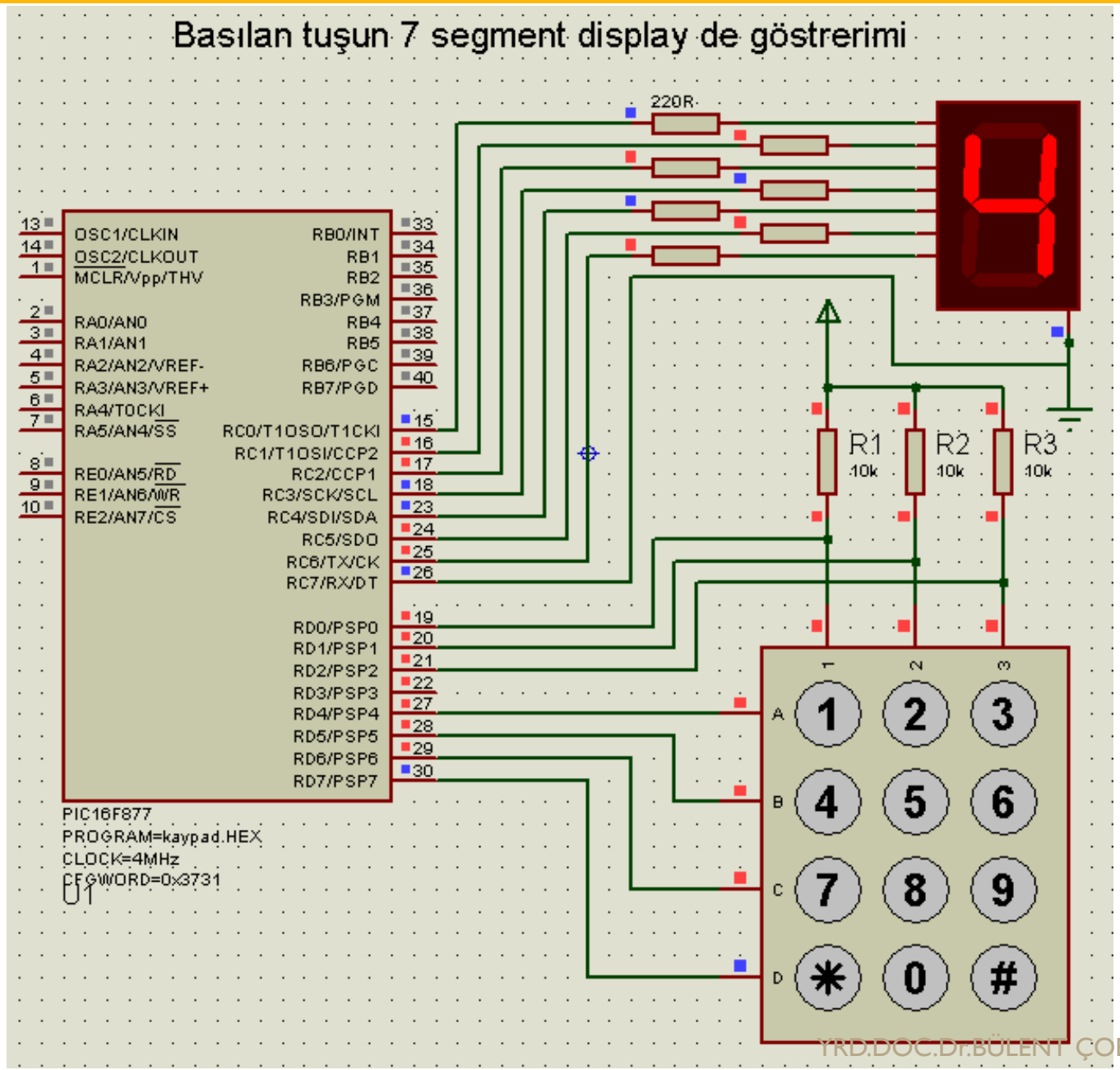
TUŞ TAKIMI (KEYPAD)

Dışarıdan bilgi girişi amacıyla yaygın olarak kullanılan birimlerden biri olan tuş takımı, sütun hatları ve satır hatları olarak düzenlenmiş iletken yolların bir buton ile kısa devre edilmesinden ibarettir. Bağlantımıza göre sütunlarda lojik-0 (GND-şase) vardır. Hangi tuşa basıldığını anlamak için önce satırlardan biri lojik-1 diğerleri lojik-0 yapılır. Sonra sütunlar okunur, hangi giriş lojik-1 ise o satıra ait sütundaki tuşa basılmış demektir. İstenen tuşa hangi değerin verileceği programcıya aittir.

Şekil'de butonlarla yapılmış 4x4 tuş takımı görülmektedir. Butonların bir ucu satır kısmına, bir ucu da sütun kısmına bağlıdır. Denetleyici ile tarama yapılırken **satırlar çıkış, sütunlar ise giriş olarak tanımlanır.**

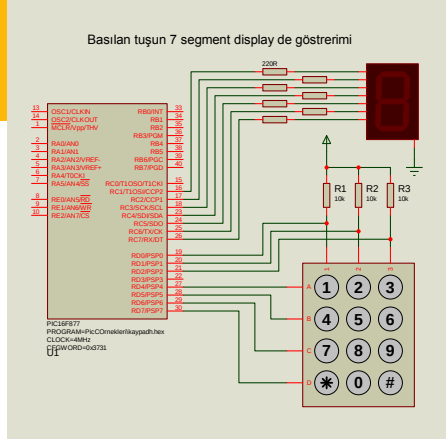


KEYPAD UYGULAMASI

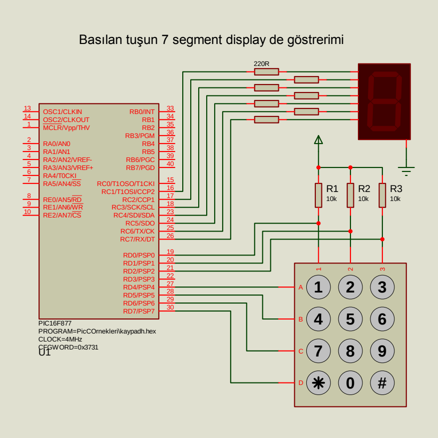


KEYPAD UYGULAMASI: ASSEMBLY KODU (1/2)

```
1         LIST P=16F877
2         #INCLUDE<P16F877.INC>
3         Sayac    EQU 0x20          ; Sayac degiskeni
4       ; PORTLARIN HAZIRLANMASI.....................................
5         BSF STATUS,5
6         CLRF     TRISC            ; PORTC CIKIS
7         MOVLW    B'00000111'      ; Keypad
8         MOVWF    TRISD            ; PORTD SUTUNLAR GIRIS, SATIRLAR CIKIS
9         BCF STATUS,5              ; BANK DEGISTIR
10        CLRF     PORTC
11        GOTO     ana      ; ANA PROGRAMA GIT
12      ; Keypad satırlarını kontrol et ..............................
13      row
14        INCF     Sayac            ; Sayac degeri
15        BTFSS    PORTD,0          ; ilk sutun
16        GOTO     TusBas           ; basılı tuş bulundu, çık
17
18        INCF     Sayac            ; tekrarla
19        BTFSS    PORTD,1          ; ikinci sutun
20        GOTO     TusBas           ; basılı tuş bulundu
21
22        INCF     Sayac            ; tekrarla
23        BTFSS    PORTD,2          ; ucuncu sutun
24        GOTO     TusBas           ; basılı tuş bulundu
25        GOTO     next             ; sonraki satıra git
26      ; Keypadi tara...............................................
27      tara
28        CLRF     Sayac            ; sayacı sıfırla
29        BSF      STATUS,0         ; Elde bitini birle
30        BCF      PORTD,4          ; ilk satırı sec
31      newrow
32        GOTO     row              ; satırı kontrol et
33      next
34        BSF      PORTD,3          ; Satırlar ayarlanıyor
35        RLF      PORTD            ; Bir sonraki satırı seç
36        BTFSC    STATUS,0         ; Elde biti sıfır mı?
37        GOTO     newrow           ; değilse, sonraki satıra git
38        GOTO     tara             ; evetse, tekrar tara
39
40      TusBas   RETURN             ; Sayac değeri ile birlikte çık
```



KEYPAD UYGULAMASI: ASSEMBLY KODU (2/2)



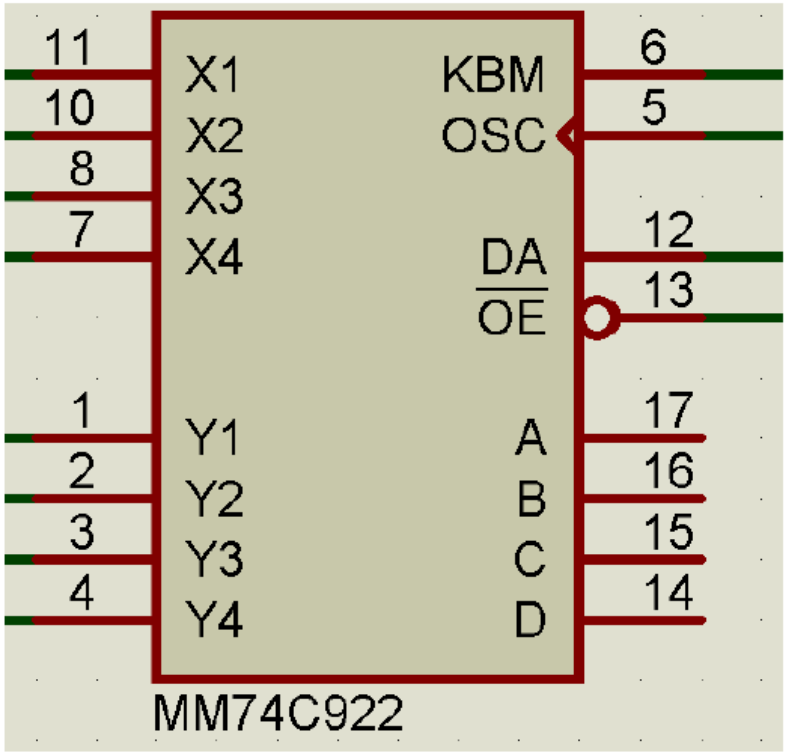

```
; Display kod tablosu.......................................
tablo
    MOVF    Sayac,W      ; Sayac değerini al (W ya aktar)
    ADDWF   PCL          ; ve tabloda ilgili değere atla
    NOP                  ; tabloya gir
    RETLW   b'00000110'  ; kod '1'
    RETLW   b'01011011'  ; kod '2'
    RETLW   b'01001111'  ; kod '3'
    RETLW   b'01100110'  ; kod '4'
    RETLW   b'01101101'  ; kod '5'
    RETLW   b'01111101'  ; kod '6'
    RETLW   b'00000111'  ; kod '7'
    RETLW   b'01111111'  ; kod '8'
    RETLW   b'01101111'  ; kod '9'
    RETLW   B'11110110'  ; kod '*'
    RETLW   b'00111111'  ; kod '0'
    RETLW   B'10010010'  ; kod '#'
; DISPLAYDE GOSTER...............................
goster
    CALL    tablo        ; display kod tablosundan kodu al
    MOVWF   PORTC        ; ve onu goster
    RETURN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; keypad oku & display de göster....
ana
    MOVLW   0XFF
    MOVWF   PORTD
    CALL    tara         ; tuş değerini al
    CALL    goster       ; ve onu göster
    GOTO    ana          ; ve tekrarla
    END
```

KEYPAD UYGULAMASI: PIC C KODU

```
#include <htc.h>
#include "delay.h" // Gecikme fonksiyonu

#define sut1 RD0 // sut1 ifadesi RD0 ifadesine esitle
#define sut2 RD1 // sut2 ifadesi RD1 ifadesine esitle
#define sut3 RD2 // sut3 ifadesi RD2 ifadesine esitle
#define sut4 RD3 // sut3 ifadesi RD3 ifadesine esitle
#define sat1 RD4 // sat1 ifadesi RD4 ifadesine esitle
#define sat2 RD5 // sat2 ifadesi RD5 ifadesine esitle
#define sat3 RD6 // sat3 ifadesi RD6 ifadesine esitle
#define sat4 RD7 // sat4 ifadesi RD7 ifadesine esitle

char keypad_oku(void) // Fonksiyon ismi
{
  TRISD=0b00000111;
  static char tus;
  PORTD=0xFF;
  sat1=0; // 1. satır lojik-0 yapılıyor
  if(sut1==0) // 1. sütun okunuyor
  {
  DelayMs(20);
  tus=0;
  }
  if(sut2==0) // 2. sütun okunuyor
  {
  DelayMs(20);
  tus=1;
  }
  if(sut3==0) // 3. sütun okunuyor
  {
  DelayMs(20);
  tus=2;
  }

  sat1=1; // 1. satır lojik-1 yapılıyor
  sat2=0; // 2. satır lojik-0 yapılıyor
  if(sut1==0) // 1. sütun okunuyor
  {
  DelayMs(20);
  tus=3;
  }
  if(sut2==0) // 2. sütun okunuyor
  {
  DelayMs(20);
  tus=4;
  }
  if(sut3==0) // 3. sütun okunuyor
  {
  DelayMs(20);
  tus=5;
  }

  sat2=1; // 2. satır lojik-1 yapılıyor
  sat3=0; // 3. satır lojik-0 yapılıyor
  if(sut1==0) // 1. sütun okunuyor
  {
  DelayMs(20);
  tus=6;
  }
  if(sut2==0) // 2. sütun okunuyor
  {
  DelayMs(20);
  tus=7;
  }
  if(sut3==0) // 3. sütun okunuyor
  {
  DelayMs(20);
  tus=8;
  }

  sat3=1; // 3. satır lojik-1 yapılıyor
  sat4=0; // 4. satır lojik-0 yapılıyor
  if(sut1==0) // 1. sütun okunuyor
  {
  DelayMs(20);
  tus=9;
  }
  if(sut2==0) // 2. sütun okunuyor
  {
  DelayMs(20);
  tus=10;
  }
  if(sut3==0) // 3. sütun okunuyor
  {
  DelayMs(20);
  tus=11;
  }
  return tus; // Fonksiyon "tus" değeri ile geri döner
  }

void sonuc_goster(char tus_no)
{
TRISC=0; //PORTB Çıkıs
char rakam[]={0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x64,0x3F,0x71};
PORTC=rakam[tus_no];
DelayMs(5);
}
void main(void) // Ana fonksiyon alanı
{
  for(;;)
  {
    sonuc_goster(keypad_oku());
  }
}
```

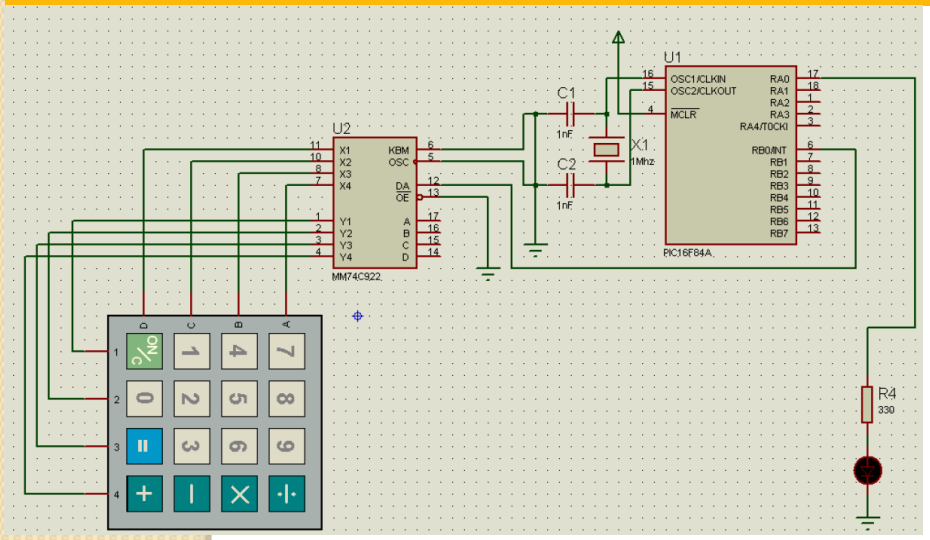
KEYPAD Entegresi

74C922 klavye entegresi



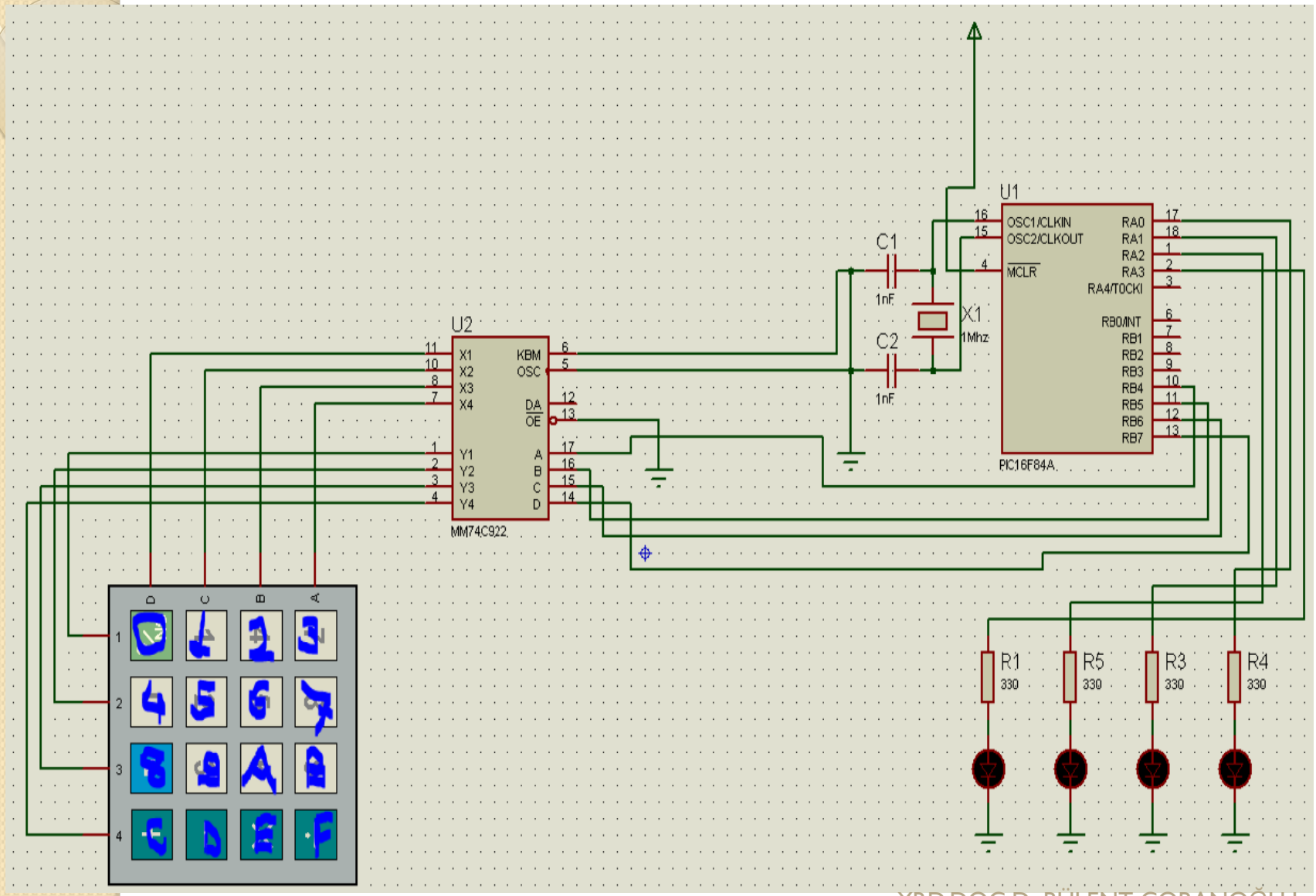
PINLER	AÇIKLAMASI
X1-X4	SUTUNLAR
Y1-Y4	SATIRLAR
A,B,C,D	DATA ÇIKIŞLARI
DA	DATA MEVCUT UCU; 1: DATA VAR, 0: DATA YOK
OE	ÇIKIŞA IZIN VER; 0: BASILAN TUŞU GOSTER 1: BASILAN TUŞU GÖSTERME
OSC	OSİLATOR
KBM	ARK SÖNDÜRME
Vcc	+3-15 V ÇALIŞMA GERİLİMİ
GND	ŞASE

Soru: 74C922 klavye entegresi ile RBO/INT harici kesmesi kullanılarak tuş takımının herhangi bir tuşuna basılı ise alarm veren program




```
        LIST P=16F84
        INCLUDE "P16F84.INC"
        ORG H'00'
        CLRF PORTA
        GOTO BASA
        ORG H'04'
        GOTO KLAVYE
BASA
        BSF STATUS,5 ;BANK1 e geçiş yap
        CLRF TRISA ;PORTA çıkış seçildi
        MOVLW b'11000000‘ ; --> KENAR
        MOVWF OPTION_REG
        BCF STATUS,5 ;BANK0 a geçiş yap
        BSF INTCON,INTE ;RB0/INT kesmesi aktif
        BSF INTCON,GIE ;Tüm kesmelere izin ver
TEKRAR
        NOP
        BCF PORTA,0
        GOTO TEKRAR
        KLAVYE
        BCF INTCON,INTF
        BSF PORTA,0
        NOP
        RETFIE
        END
```

Soru: 74C922 klavye entegresi ile RB4-RB7 değişim kesmesi kullanılarak basılan tuşu 4 adet led üzerinde ikili gösteren program



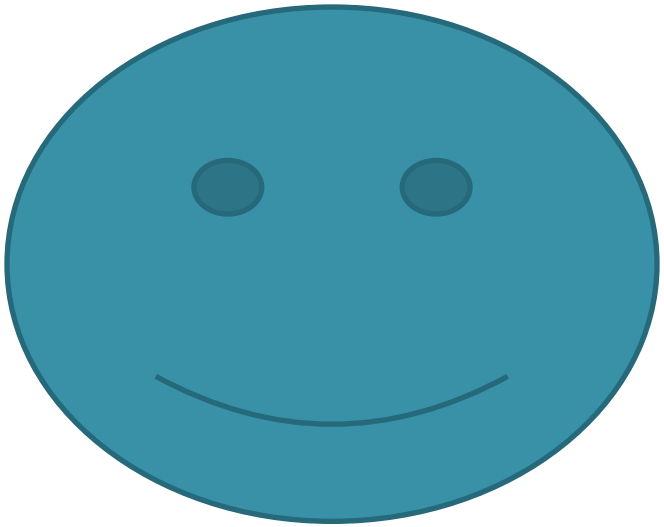
Soru: 74C922 klavye entegresi ile RB4-RB7 değişim kesmesi kullanılarak basılan tuşu 4 adet led üzerinde ikili gösteren program

//C Kodu

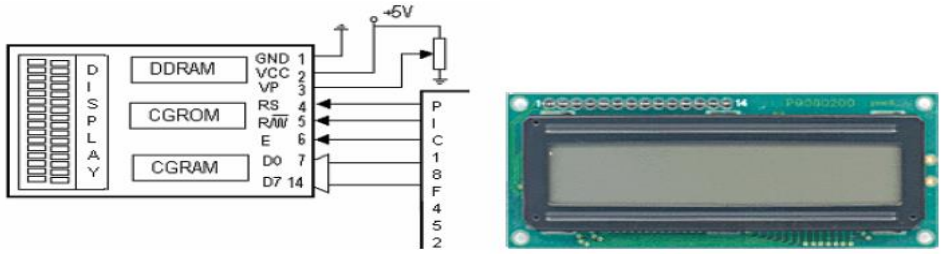
```
#include <xc.h>
void interrupt klavye()
{
RBIF=0;
#asm
SWAPF PORTB,W
ANDLW 0x0F
MOVWF PORTA
#endasm
}
void main()
{
        TRISA=0;
        TRISB=0xFF;
        PORTA=0;
        RBIE=1;
        GIE=1;
        while(1);
}
```

```
    ;Assembly kodu
    LIST P=16F84
    INCLUDE "P16F84.INC"
    ORG H'00'
    GOTO BASA
    ORG H'04'
    GOTO KLAVYE
BASA
    BSF STATUS,5 ;BANK1 e geçiş yap
    CLRF TRISA ;PORTA çıkış seçildi
    MOVLW 0XFF
    MOVWF TRISB ;PORTB GIRIŞ
    BCF STATUS,5 ;BANK0 a geçiş yap
    CLRF PORTA
    BSF INTCON,RBIE ;RB kesmesi aktif
    BSF INTCON,GIE ;Tüm kesmeler aktif
TEKRAR
    NOP
    GOTO TEKRAR
KLAVYE
    BCF INTCON,RBIF
    SWAPF PORTB,W
    ANDLW 0X0F
    MOVWF PORTA
    RETFIE
    END
```

Çalışma Sorusu: Keypade basılı tuşu PIC16F877 nin PORTC sine bağlı 7 segment display de gösteren programı 74C922 klavye entegresini kullanarak yazınız?



LCD UYGULAMASI



a) b)

Şekil 2. a) LCD içyapısı b) LCD gösterge

- LCD (Liquid Crystal Display) göstergeli mikro denetleyici uygulamaları ile hayatımızın her alanında (cep telefonları, fotokopi makineleri, otomobiller, kameralar, oyuncaklar, güvenlik sistemleri gibi) karşılaşılmaktadır.
- Karakter tabanlı dot matrix LCD (paralel/seri) ve grafik LCD olmak üzere iki çeşit LCD vardır.
- LED gösterge ile sadece sayısal değerler ve sınırlı sayıdaki karakterler gösterilebilmektedir. Buna karşılık LCD göstergeler ile her türlü yazı ve sayısal değeri göstermek mümkündür. LCD'ler çeşitli firmalar tarafından üretilmesine rağmen kontrolleri standartlaşmıştır.
- Tüm LCD göstergelerde **yetki (Enable), oku yaz (R/W), ve kaydedici seçim (RS) uçları ile veri giriş hatları** vardır.
- Bağlantı şekillerine göre LCD göstergeler seri ve paralel olarak sınıflandırılmaktadırlar. Paralel LCD'ler ucuz oluşları nedeniyle çok yaygın olarak kullanılmaktadır.

LCD Kontrol İşlemleri

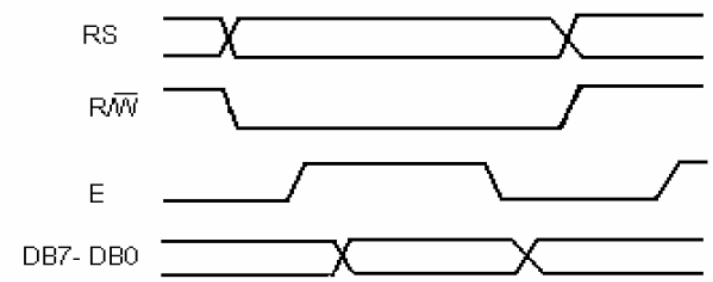
LCD göstergeye gönderilen veri ya bir komut kodu veya bir karakterdir. Şekil 3. de LCD göstergeye yazma işlemine ait zamanlama diyagramı görülmektedir. RS ucu düşük seviyeye çekilirse yapılacak işlem bir kontrol işlemidir. Eğer yüksek seviyede tutulursa gönderilen bir karakterdir. LCD'ye her 8 bitlik veri, önce yüksek değerlikli 4 - bit, sonra düşük değerlikli 4 - bit olmak üzere iki defada gönderilir. LCD ekrana veriler ASCII karakter kodları gönderilerek gösterilirler. Mesela ekrana 0 göstermek için, sıfırın ASCII kodu olan 48'i göndermek gerekir. RW=0 ise LCD ye yazma, RW=1 ise LCD den okuma işlemi yapılıyordur.

LCD ekrana veri yazmak için aşağıdaki adımlar izlenir;

- Veri, veri yoluna konulur,
- RS ucu lojik 1 yapılarak, yazma işleminin komut olmadığı belirtilir,
- RW ucu lojik 0 yapılır,
- E ucuna lojik "1–0" şeklinde bir saat (clock) darbesi verilir.

LCD ekrana komut yazmak için ise aşağıdaki adımlar izlenir;

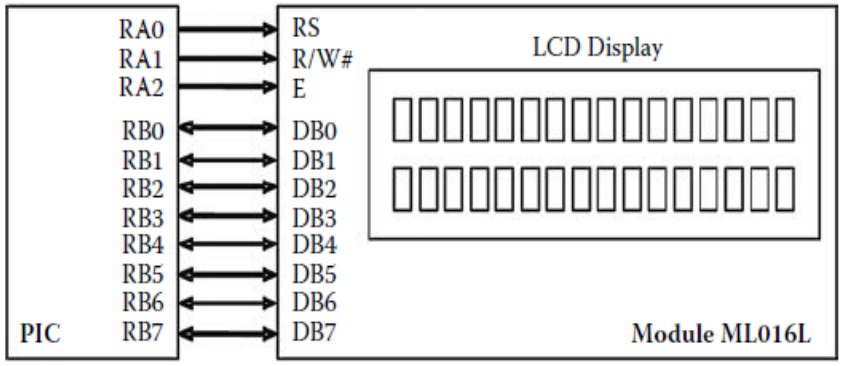
- Komut, veri yoluna konulur,
- RS ucu lojik 0 yapılarak, yazma işleminin komut olduğu bildirilir,
- RW ucu lojik 0 yapılır,
- E ucuna lojik "1–0" şeklinde bir saat (clock) darbesi verilir.

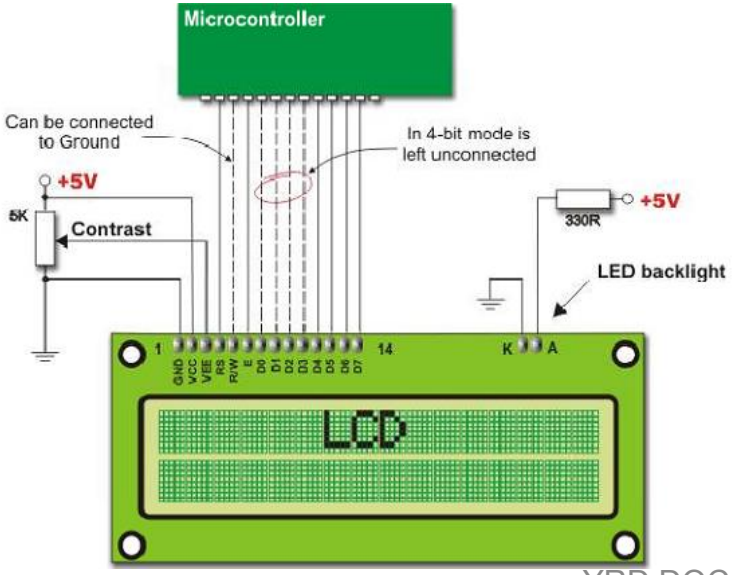


Şekil 3. LCD göstergeye yazma işlemine ait zamanlama diyagramı [3].

LCD Bacak Bağlantıları

LCD dış dünyaya 14 pinlik bir konnektör ile bağlanır. Tablo'da LCD nin pin (bacak) numaraları ve her pinin görevi verilmiştir.





Pin No	Pin ismi	Fonksiyon	Açıklama
1	Vss	Toprak/Şase	GND
2	Vdd	Kaynak / Power	+ 5 V
3	Vee	Kontrast /Parlaklık	(-2) 0 - 5 V
4	RS	Komut/Veri Seçici	0: Komut, 1: Veri
5	R/W	Oku/Yaz	0:LCD'ye yaz, 1: LCD den oku
6	E	Enable /Etkinleştirme	LCD'ye veri gönderme için aktif yapılır. E bacağının lojik 1 den lojik 0' a geçişi ile LCD'ye veri transfer olur. Bacağın lojik 0'dan lojik 1'e geçmesi ile LCD'den durum okunabilir.
7	D0	I/O	Data LSB
8	D1	I/O	Data
9	D2	I/O	Data
10	D3	I/O	Data
11	D4	I/O	Data
12	D5	I/O	Data
13	D6	I/O	Data
14	D7	I/O	Data MSB

Karakter gösterimi

- Karakter LCD'lerin oluşturabileceği her bir karakter ise karakter LCD'nin özel CGROM hafızasına kaydedilmişlerdir. ASCII karakter uyumu olan karakterlerin listesi şekil'de görülebilmektedir.
- Şekilde görüleceği üzere CGROM'un ilk 8 karakterlik (0x00..0x0F) kısmı boştur ve yazılabilirdir. Bu kullanıcıya tabloda olmayan karakterleri (CGRAM ile) kendisinin tanımlamasına imkan tanır.

Şekil 40 – LCD Karakter Tablosu

LCD

Table 12-4: List of LCD Instructions

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (Max)
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter	1.64 ms
Return Home	0	0	0	0	0	0	0	0	1	-	Sets DD RAM address 0 as address counter. Also returns display being shifted to original position. DD RAM contents remain unchanged.	1.64 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display. These operations are performed during data write and read.	40 µs
Display On/ Off Control	0	0	0	0	0	0	1	D	C	B	Sets On/Off of entire display (D), cursor On/Off (C), and blink of cursor position character (B).	40 µs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Moves cursor and shifts display without changing DD RAM contents.	40 µs
Function Set	0	0	0	0	1	DL	N	F	-	-	Sets interface data length (DL), number of display lines (L), and character font (F).	40 µs
Set CG RAM Address	0	0	0	1	AGC						Sets CG RAM address. CG RAM data is sent and received after this setting.	40 µs
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address. DD RAM data is sent and received after this setting.	40 µs
Read Busy Flag & Address	0	1	BF	AC							Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	40 µs
Write Data CG or DD RAM	1	0	Write Data								Writes data into DD or CG RAM.	40 µs
Read Data CG or DD RAM	1	1	Read Data								Reads data from DD or CG RAM.	40 µs

Notes:
1. Execution times are maximum times when fcp or fosc is 250 kHz.
2. Execution time changes when frequency changes.
(e.g., when fcp or fosc is 270 kHz: 40 µs × 250 / 270 = 37 µs.)

Table 12-1: Pin Descriptions for LCD

Pin	Symbol	I/O	Description
1	V_{SS}	--	Ground
2	V_{CC}	--	+5 V power supply
3	V_{EE}	--	Power supply to control contrast
4	RS	I	RS = 0 to select command register, RS = 1 to select data register
5	R/W	I	R/W = 0 for write, R/W = 1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

Table 12-2: LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 matrix

Note: This table is extracted from Table 12-4.

LCD nin kullanıma hazır hale getirilmesi

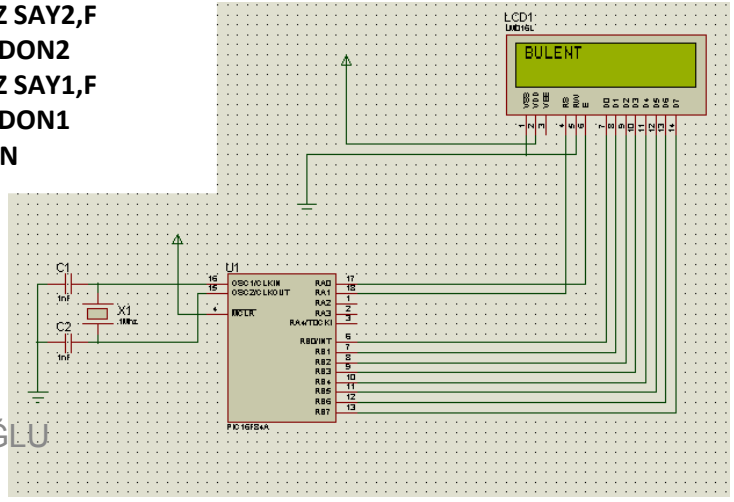
COMMAND	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
Clear display	0	0	0	0	0	0	0	0	0	1
Cursor home	0	0	0	0	0	0	0	0	1	x
Entry mode set	0	0	0	0	0	0	0	1	I/D	S
Display on/off control	0	0	0	0	0	0	1	D	U	B
Cursor/Display Shift	0	0	0	0	0	1	D/C	R/L	x	x
Function set	0	0	0	0	1	DL	N	F	x	x
Set CGRAM address	0	0	0	1	CGRAM address					
Set DDRAM address	0	0	1	DDRAM address						
Read "BUSY" flag (BF)	0	1	BF	DDRAM address						
Write to CGRAM or DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Read from CGRAM or DDRAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0

```
I/D 1 = Increment (by 1)          R/L 1 = Shift right
    0 = Decrement (by 1)              0 = Shift left

S 1 = Display shift on            DL 1 = 8-bit interface
  0 = Display shift off              0 = 4-bit interface

D 1 = Display on                  N 1 = Display in two lines
  0 = Display off                   0 = Display in one line

U 1 = Cursor on                   F 1 = Character format 5x10 dots
  0 = Cursor off                    0 = Character format 5x7 dots

B 1 = Cursor blink on             D/C 1 = Display shift
  0 = Cursor blink off                0 = Cursor shift
```

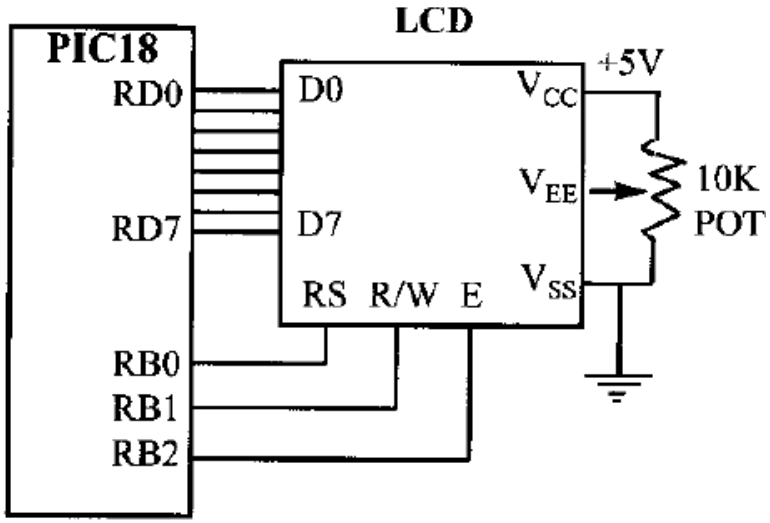
1. Display is cleared.
2. Mode
   DL = 1 - Communication through 8-bit interface
   N = 0 - Messages are displayed in one line
   F = 0 - Character font 5 x 8 dots
3. Display/Cursor on/off
   D = 0 - Display off
   U = 0 - Cursor off
   B = 0 - Cursor blink off
4. Character entry
   ID = 1 Displayed addresses are automatically incremented by 1
   S = 0 Display shift off

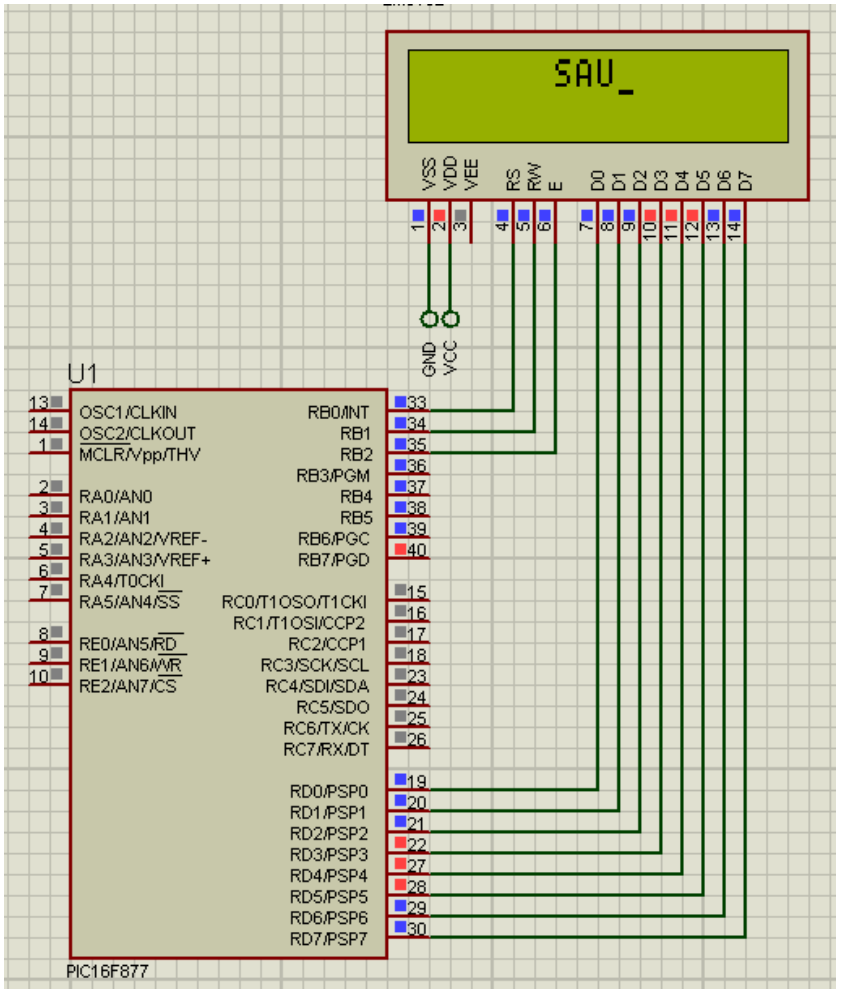
PIC- Assembly: 8 bitlik LCD ye yazı yazmak

```
        LIST P=16F84
        INCLUDE "P16F84.INC«

        CBLOCK H'0C'
        SAY1,SAY2
        ENDC

        CLRF PORTB
        BSF STATUS,5
        CLRF TRISA
        CLRF TRISB
        BCF STATUS,5
BASLA
        MOVLW 0X01; DISPLAY TEMIZLE
        CALL        KOMUTYAZ
        MOVLW 0X30 ;8 BITLIK BAGLANTI, 1 SATIR
        CALL        KOMUTYAZ
        MOVLW 0X0C  ;EKRANI AÇ,KURSORU KAPAT, YANIP SONME YOK
        CALL        KOMUTYAZ
        CALL        SATIRYAZ
DUR
        GOTO DUR

KOMUTYAZ
        BCF PORTA,1    ;RS=0 İLE KOMUT YAZ
        MOVWF PORTB
        BSF PORTA,0    ;E=1
        CALL BEKLE
        BCF PORTA,0    ;E=0
        RETURN
DATAYAZ
        BSF PORTA,1    ;RS=1 İLE VERI YAZ
        MOVWF PORTB
        BSF PORTA,0    ;E=1
        CALL BEKLE
        BCF PORTA,0    ;E=0
        RETURN
SATIRYAZ

        MOVLW 'B'
        CALL DATAYAZ
        MOVLW 'U'
        CALL DATAYAZ
        MOVLW 'L'
        CALL DATAYAZ
        MOVLW 'E'
        CALL DATAYAZ
        MOVLW 'N'
        CALL DATAYAZ
        MOVLW 'T'
        CALL DATAYAZ
        RETURN
BEKLE

        MOVLW H'FF'
        MOVWF SAY1
DON1

        MOVLW H'FF'
        MOVWF SAY2
DON2

        DECFSZ SAY2,F
        GOTO DON2
        DECFSZ SAY1,F
        GOTO DON1
        RETURN
        END
```



Devre Şeması



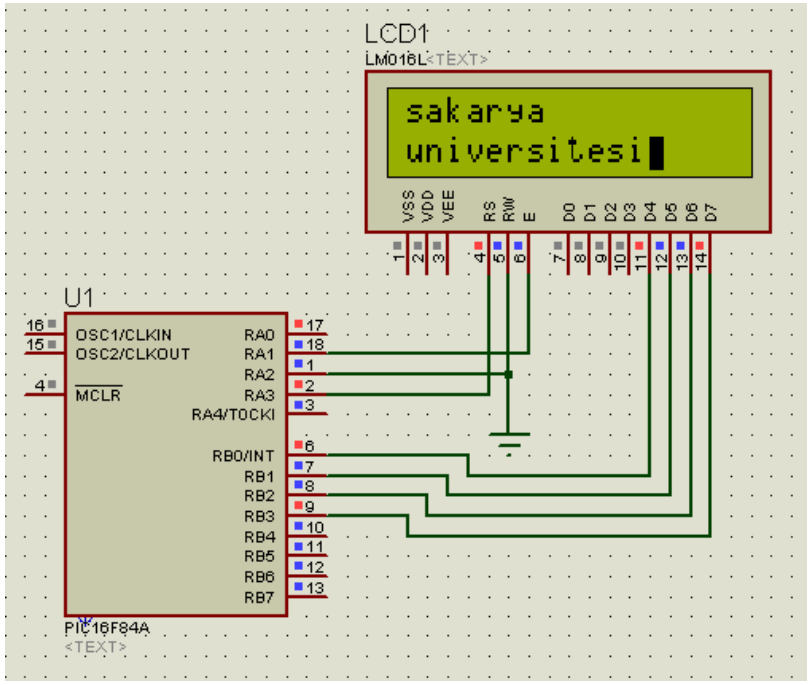


PIC- C: 8 bitlik LCD ye yazı yazmak

```
#include <xc.h>
#define RS RB0
#define RW RB1
#define E RB2
void lcdKomut (unsigned char);
void lcdVeri (char);
void bekle(unsigned int);

void main()
{
TRISD=0;
TRISB=0;
E=0;
bekle(250);
lcdKomut(0x38);//2 satır 5*7 matrix
bekle(250);
lcdKomut(0x0E); //display ve kursor açık
bekle(15);
lcdKomut(0x01);//Ekranı temizle
bekle(15);
lcdKomut(0x06);//Kursor sağa kaysın
bekle(15);
lcdKomut(0x86);//1. satır,6.sıra
bekle(15);
lcdVeri('S');
bekle(15);
lcdVeri('A');
bekle(15);
lcdVeri('U');
}
```

```
void lcdKomut(unsigned char k)
{
PORTD=k;
RS=0;
RW=0;
E=1;
bekle(1);
E=0;
}

void lcdVeri(char veri)
{
PORTD=veri;
RS=1;
RW=0;
E=1;
bekle(1);
E=0;
}

void bekle(unsigned int sn)
{
unsigned int i,j;
for (i=0; i<sn; i++)
 for (j=0; j<135; j++);
}
```

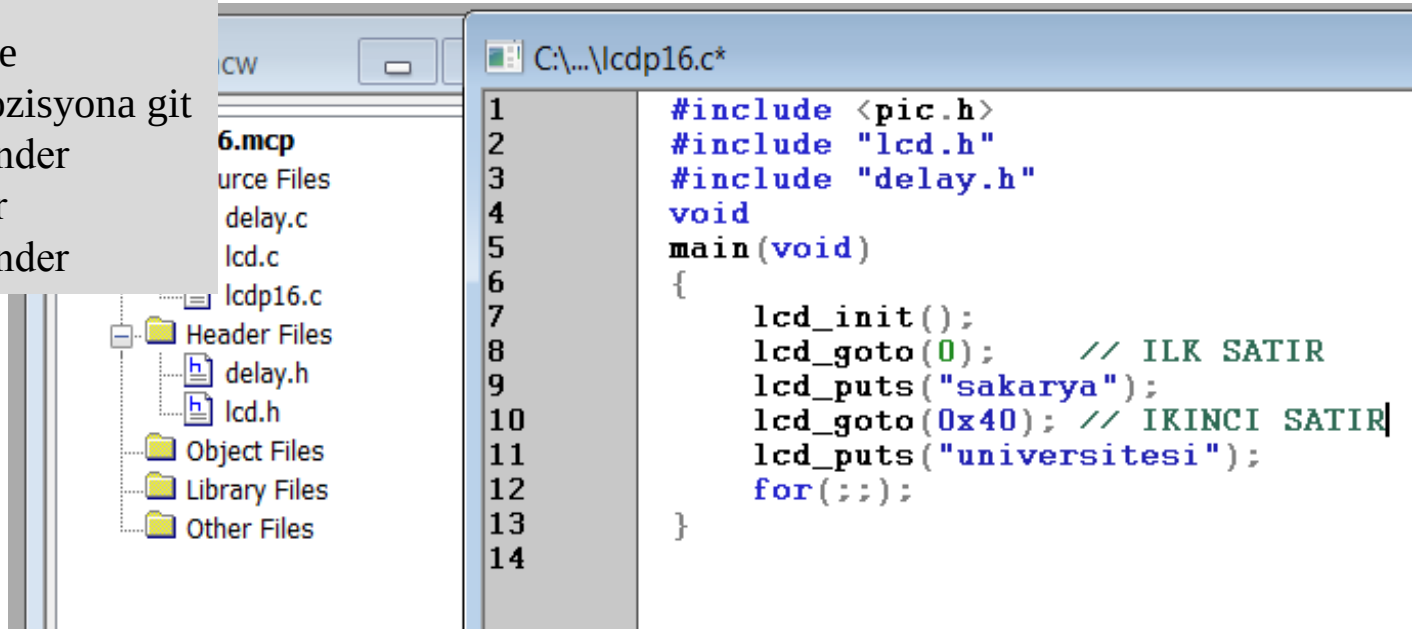
PIC- C: Hazır fonksiyon kullanarak 4 bitlik LCD ye yazı yazmak



Kod Değişikliği:
Hitech klasörünün altındaki 'LCDemo' klasörü içerisindeki «lcd.c» dosyasındaki PORTD yi PORTB, TRISD'yi de TRISB ye şeklinde değiştirmeliyiz.

lcd.h içindeki fonksiyonlar;

lcd_init(): lcd yi hazırla
lcd_clear(): lcd yi temizle
lcd_goto(p): belirtilen pozisyona git
lcd_write(): Karakter gönder
lcd_puts(): String gönder
lcd_putch(): karakter gönder

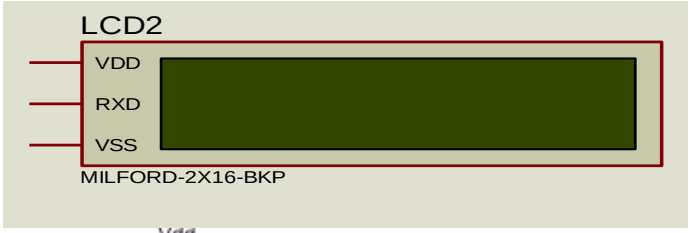


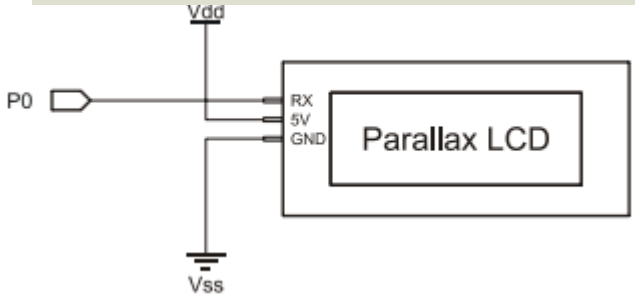
Seri LCD

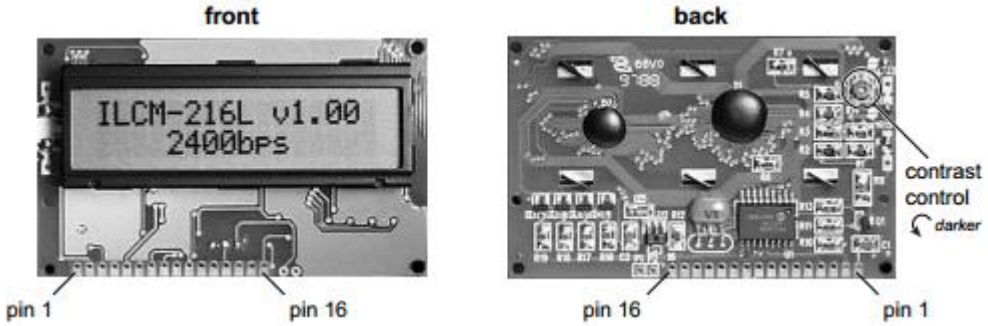
MCU ya seri olarak sadece bir kablo ile bağlanırlar ve normal olarak RS232 seri iletişim protokolüne göre çalışırlar. Fiyatları paralel LCD ye göre daha pahalıdır.

Table 4.14 Pin configuration of ILM-216

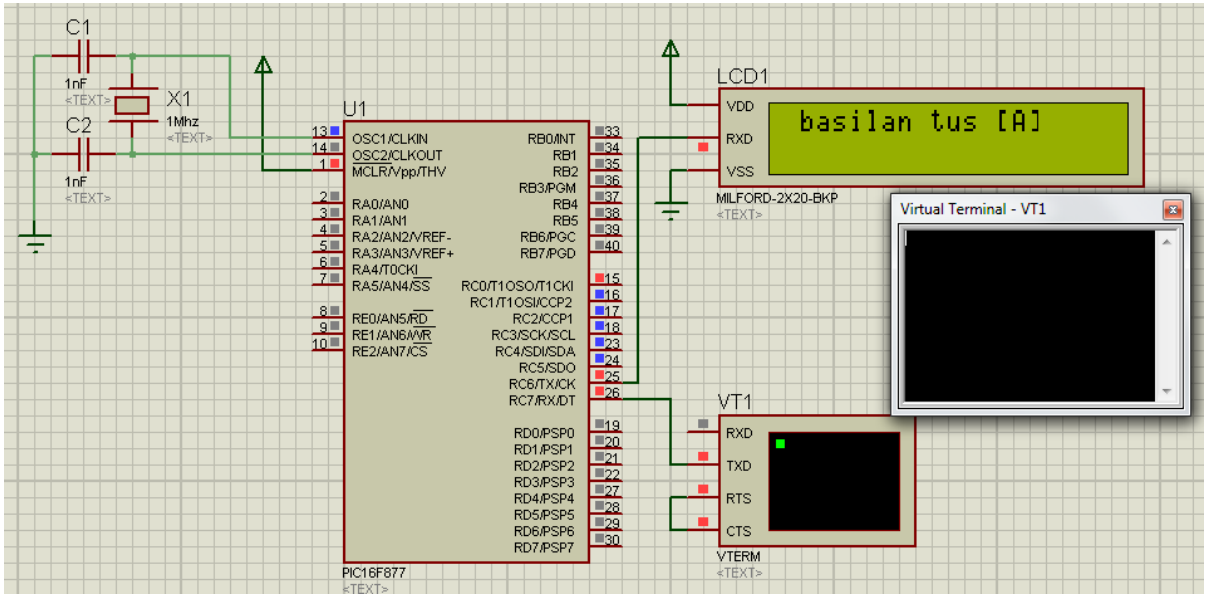
Pin No	Function
1	Ground
2	+5 V
3	Serial in
4	Serial out
5	Bell
6	Ground
7	Config/test
8	9600 baud
9	Switch 1
10	Switch 1 ground
11	Switch 2
12	Switch 2 ground
13	Switch 3
14	Switch 3 ground
15	Switch 4
16	Switch 4 ground







Basılan tuşu seri LCD de gösteren uygulama




```
#include <stdio.h>
#include <htc.h>
void main(void){
    unsigned char input;

    INTCON=0;   //kesmeler iptal
    printf("\rbir tusa bas bekle:\n");
    while(1){
        input = getch();     // kullanıcıdan cevap bekle
        printf("\rbasilan tus [%c]",input); // yansıt
    }
}
```

MOTOR TİPLERİ

- **DC MOTOR:**

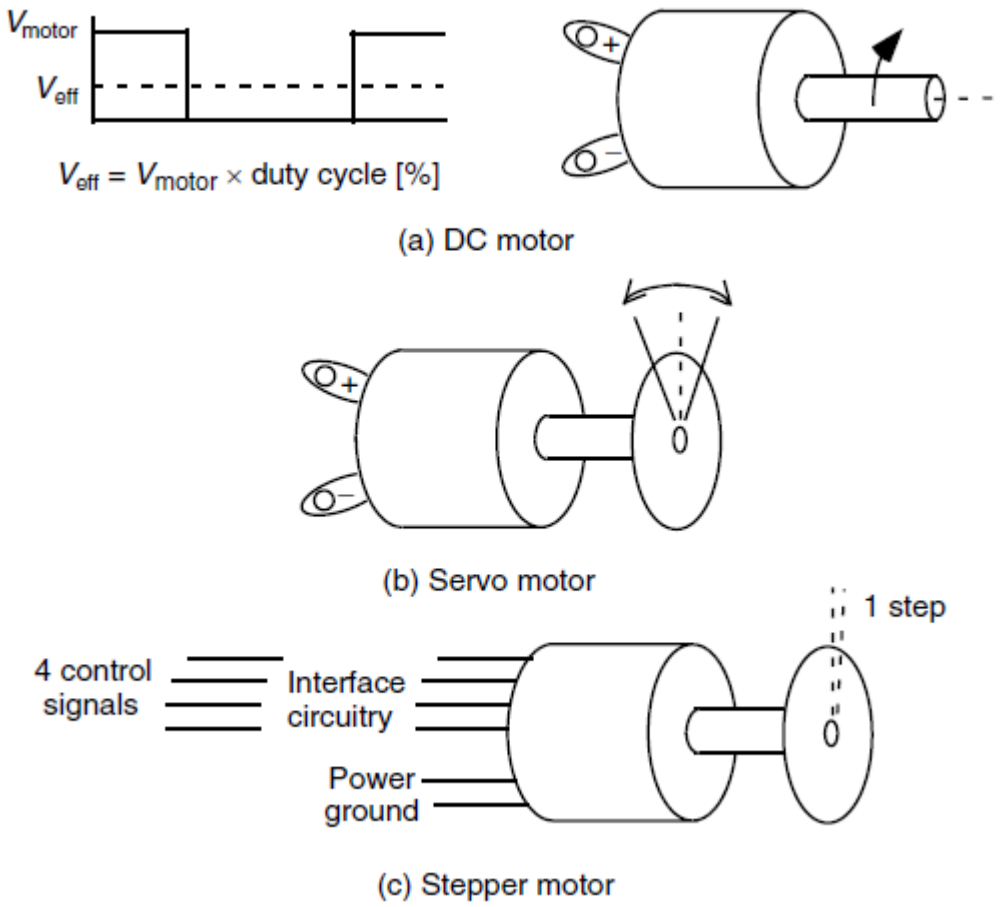
Kutuplarına uygulanan DC gerilim motorun dönmesini sağlar, kutup uçları ters çevrildiğinde motor dönüş yönü de değişir.

- **SERVO MOTOR:**

Verilen duty cycle periyotlarına göre açısal bir dönüş yapar.

- **STEP MOTOR**:

Girişlerine uygulanan lojik sinyallere karşılık analog dönme hareketi yaparlar. Her bir pals, rotoru bir adım döndürmektedir.



Step motor

Step motorlar, diğer motorlardan farklı olarak motor bobinlerine uygulanan elektrik palsi (pulse) ile döndürülürler. Her bir pals, rotoru bir adım döndürmektedir. (Adımın derece değeri, motorun adım sayısına bağlıdır) bu palsler, "adım" olarak adlandırıldıklarından bu motor türüne adım (step) motor denir.

Adım motorları (Step Motors), girişlerine uygulanan lojik sinyallere karşılık analog dönme hareketi yapan fırçasız, sabit mıknatıs kutuplu DC motorlardır. Sabit mıknatıslı kutuplar hareketli kısımda yer alır. DC gerilimin uygulandığı sargıların bulunduğu kısım "stator", dönen kısım ise "rotor" olarak isimlendirilir.

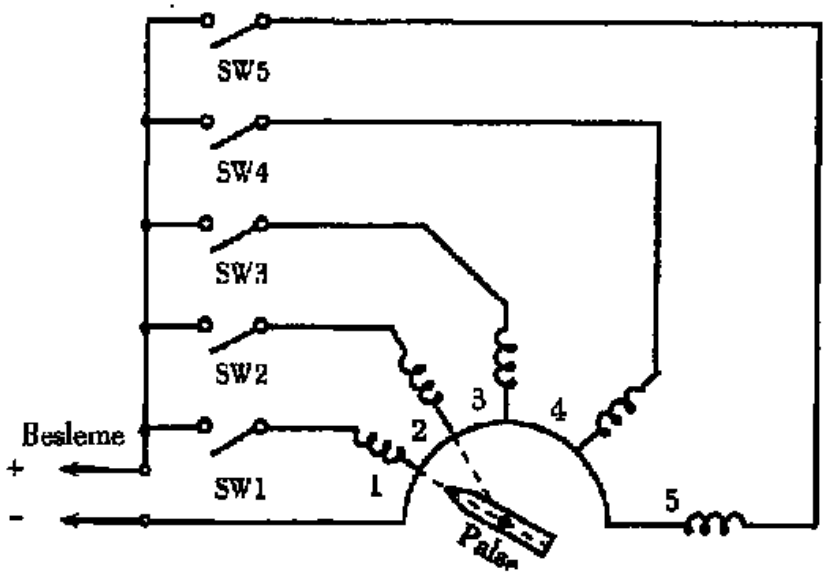
Step motorlar, bir turdaki adım sayısı ile anılırlar. Örneğin 300 adımlık bir step motorun bir tam dönüşünde (tur) 300 adım bulunur. **Zaten ismini de buradan alır.** Buna göre bir adımın açısı; 360 / 300 = 1,2 derecedir. Bu değer, step motorun hassasiyetinin bir ölçüsüdür. **Her bir adımda motorun döneceği açı derecesi motorun üzerinde yazar. Örneğin 7.5 derecelik (360º/48) step motor, bir tam turda 48 adım atar.**

Bir devirdeki adım sayısı yükseldikçe step motorun hassasiyeti yükselmektedir ancak buna paralel olarak maliyeti de artmaktadır. Step motorlar yarım adım modunda çalıştırıldıklarında hassasiyetleri daha da artar. Örneğin 300 adım / tur değerindeki bir step motor, yarım adım modunda çalıştırıldığında tur başına 600 adım yapacaktır. Bu da 1,2 dereceye oranla daha hassas olan 0,6 derecelik bir adım açısı anlamına gelir.

ÇALIŞMA PRENSİBİ /Kullanım Yerleri

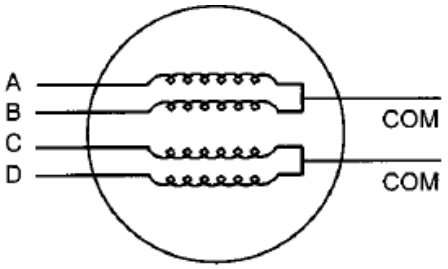
- **Step motora giriş palsi uygulandığı zaman belli bir miktar döner ve durur. Bu dönme miktarı motorun yapısına göre belli bir açı ile sınırlandırılmıştır. Step motorda rotorun dönmesi girişe uygulanan pals adedine bağlı olarak değişir. Girişe tek bir pals verildiğinde rotor tek bir adım hareket eder ve durur. Daha fazla pals uygulanınca pals adedi kadar adım hareket eder. Bütün step motorlarının çalışma prensibi bu şekildedir.**

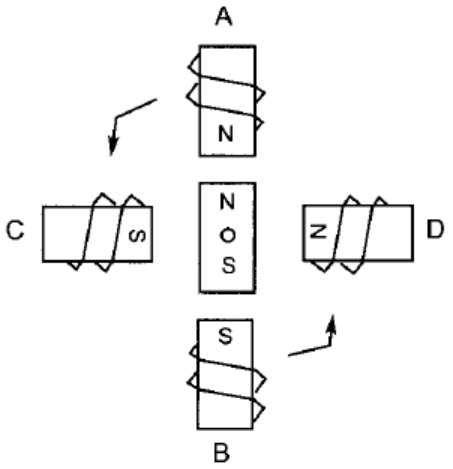
- Kullanım yerleri: Bant sürücüler, imalat tezgahları, printer (yazıcı), disket sürücüleri, teyp sürücüleri, hafıza işlemlerinde, tıbbi cihazlarda, makine tezgahlarında, dikiş makinelerinde, kameralarda, taksimetrelerde, kart okuyucularında, ayar ve kontrol tekniğinde, uzaktan kumanda göstergelerinde kullanılır.
- Sonuç olarak step motorlar; her türlü kontrol edilmiş hareket veya pozisyon gerekli olan yerlerde, dijital bilgileri mekanik harekete çeviren bir transduser olarak görev yapar.

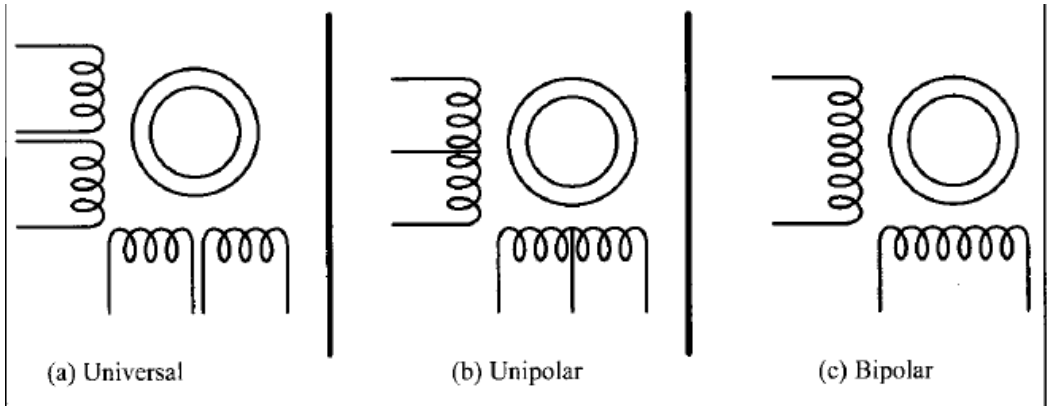


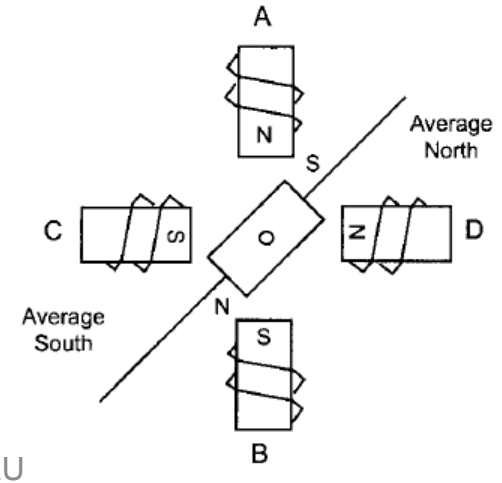
Step motorların sürülmesi

- Bir step motor, tek fazlı, 2 faz tam adımlı ve 2 faz yarım adımlı olmak üzere farklı şekillerde sürülebilir. Bazı kaynaklarda unipolar step motorlar tek fazlı, bipolar motorlar ise 2 fazlı olarak adlandırılabilir. Bipolar iki yönlü beslenen anlamına gelir ve Bipolar step motor, iki yönde de akım akabilen motor demektir.









1 fazlı sürüm

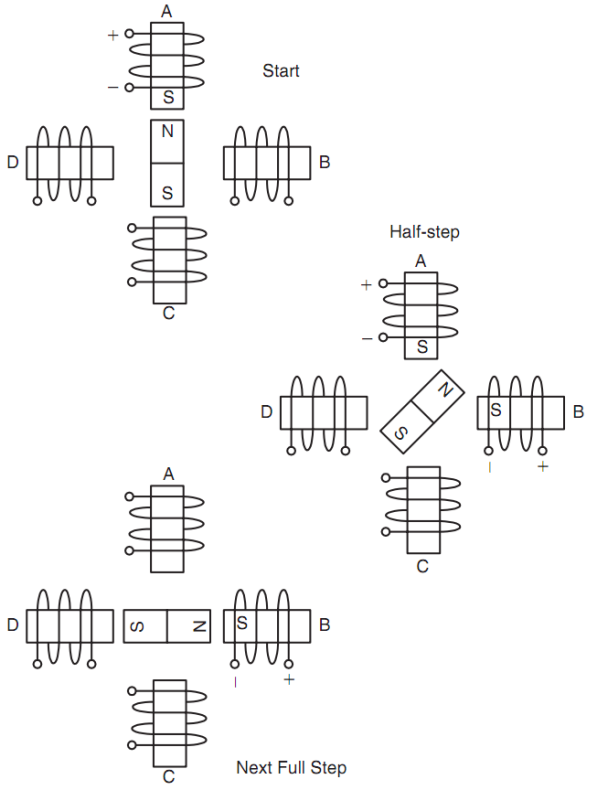
A	B	C	D
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

2 fazlı TAM adımlı sürüm

A	B	C	D
1	1	0	0
0	1	1	0
0	0	1	1
1	0	0	1

2 fazlı YARIM adımlı sürüm

A	B	C	D
1	0	0	0
1	1	0	0
0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	1
0	0	0	1
1	0	0	1



Normal / Dalga Sıralı 4 adım

Clockwise ↓

Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

↑ Counter-clockwise

Clockwise ↓

Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

↑ Counter-clockwise

Table 17-4: Stepper Motor Step Angles

Step Angle	Steps per Revolution
0.72	500
1.8	200
2.0	180
2.5	144
5.0	72
7.5	48
15	24

Soru: 2 dereceli ve 4 adımlı bir step motoru 80 derece hareket ettirmek için kaç sıralı adım yaptırmamız gerekir?

Cevap: ?

Yarım adımlı

Clockwise

Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	0	0	0
3	1	1	0	0
4	0	1	0	0
5	0	1	1	0
6	0	0	1	0
7	0	0	1	1
8	0	0	0	1

Counter-clockwise

Table 17-4: Stepper Motor Step Angles

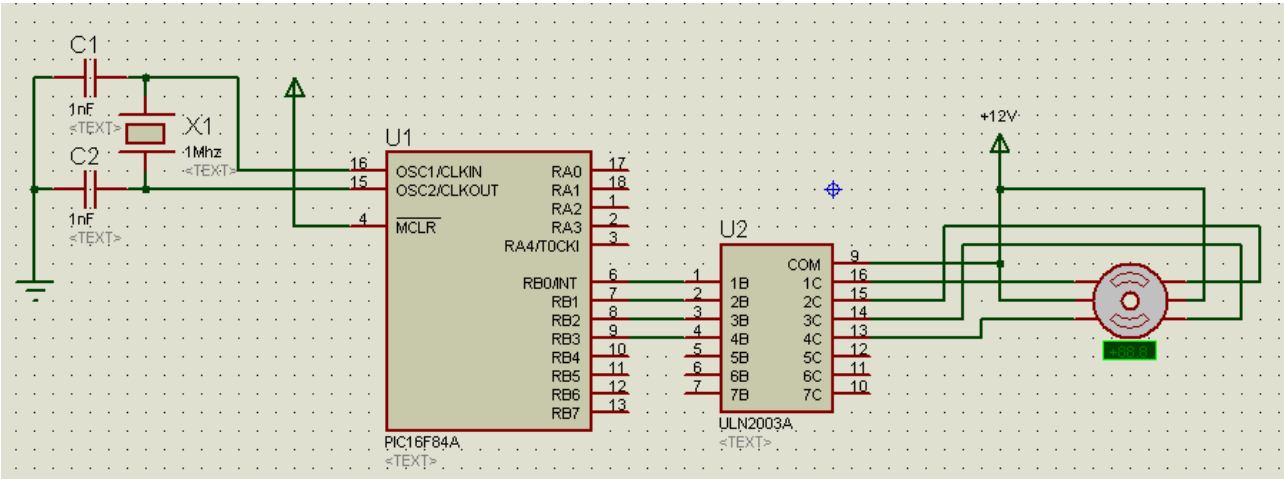
Step Angle	Steps per Revolution
0.72	500
1.8	200
2.0	180
2.5	144
5.0	72
7.5	48
15	24

Soru: 2 dereceli ve 4 adımlı bir step motoru 80 derece hareket ettirmek için kaç sıralı adım yaptırmamız gerekir?

Cevap: ?

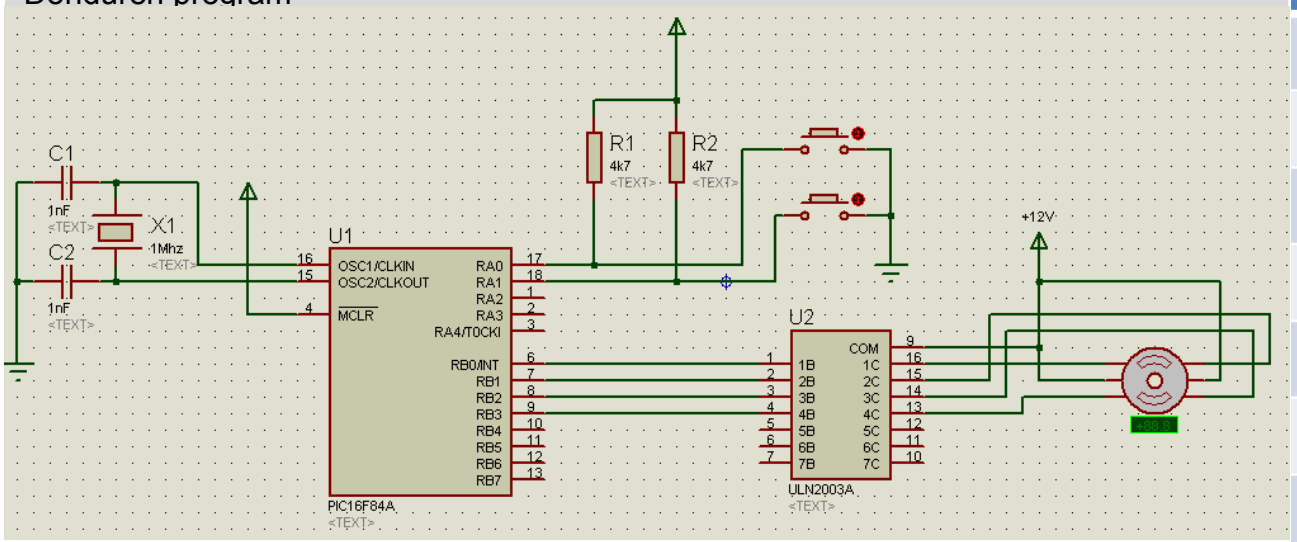
Step motoru yarım adım döndüren program

A(B4)	B(B3)	(B2)	D(B1)
0	0	1	1
0	0	1	0
0	1	1	0
0	1	0	0
1	1	0	0
1	0	0	0
1	0	0	1
0	0	0	1




```
#include<pic.h> //<htc.h>
void bekle()
{for (int i=0; i<=10000;i++);}
void main()
{
TRISB=0;
PORTB=0;
unsigned char dizi[]={0b0011,0b0010,0b0110,0b0100,0b1100,0b1000,0b1001,0b0001};
while(1)
{
    for(int i=0; i<=7; i++)
    {
        PORTB=dizi[i];
        bekle();
    }
}}
```

Step motoru RA0 a basınca sağa RA1 e basınca sola
Döndüren program



A(B4)	B(B3)	(B2)	D(B1)
0	0	1	1
0	0	1	0
0	1	1	0
0	1	0	0
1	1	0	0
1	0	0	0
1	0	0	1
0	0	0	1

```
#include <htc.h>
#include "delay.h"
// __CONFIG(0x3F31);
void main(void)
{
char i=0;
PORTA=0x00;    // Portlar sıfırlanıyor
PORTB=0x00;
TRISA=0x03;    // RA0, RA1 giris
TRISB=0x00;    // PORTB çıkıs
// Yarım adım için dizi tanımlanıyor
char y_adim[]={0b0011,0b0010,0b0110,0b0100,0b1100,0b1000,0b1001,0b0001};
while(1)
  {
        while(!RA0){ //sağ
            for (int i=0; i<=7; i++)
            {
                PORTB=y_adim[i];
                DelayMs(20);
            }
        }
        while(!RA1){ //sol
            for (int i=7; i>=0; i--)
            {
                PORTB=y_adim[i];
                DelayMs(20);
            }
         }
} }
```

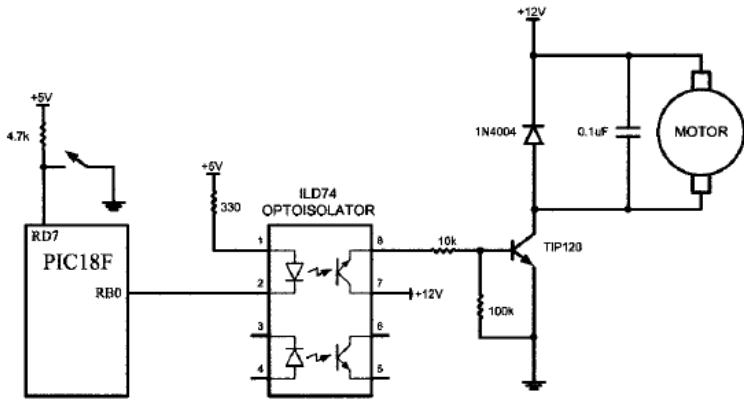
2 tur sola 4 tur sağa dönen step motor

```
    LIST P=16F84
    INCLUDE "P16F84.INC"
    CBLOCK H'0C'
    SAY1,SAY2,TUR,STEP
    ENDC
    CLRF PORTB
    BSF STATUS,5
    CLRF TRISB
    BCF STATUS,5
    MOVLW H'FF'
    MOVWF STEP
    MOVLW .2
    MOVWF TUR
SOL
    INCF STEP,F
    MOVF STEP,W
    ANDLW b'00000111'
    CALL DIZI
    MOVWF PORTB
    CALL BEKLE

    DECFSZ TUR,F
    GOTO SOL
    MOVLW .4
    MOVWF TUR
SAG
    DECF STEP,F
    MOVF STEP,W
    ANDLW b'00000111'
    CALL DIZI
    MOVWF PORTB
    CALL BEKLE

    DECFSZ TUR,F
    GOTO SAG
DON
    GOTO DON
```

```
DIZI
    ADDWF PCL,F
    RETLW b'00000001'
    RETLW B'00001001'
    RETLW B'00001000'
    RETLW B'00001100'
    RETLW B'00000100'
    RETLW B'00000110'
    RETLW B'00000010'
    RETLW B'00000011'
BEKLE
    MOVLW H'FF'
    MOVWF SAY1
DON1
    MOVLW H'FF'
    MOVWF SAY2
DON2
    DECFSZ SAY2,F
    GOTO DON2
    DECFSZ SAY1,F
    GOTO DON1
    RETURN
    END
```

YRD.DOC.Dr.BÜLENT
ÇOBANOĞLU

Soru: RD7 ye bağlı anahtara göre saat yönüne (anahtar==0) veya tersi (anahtar==1) yönde hareket eden uygulamayı yapınız




```
#define SW PORTDbits.RD7
void MSDelay(int ms);
void main()
  {
    TRISD=0x80;        //RD7 as input pin
    TRISB=0x0;         //PORTB as output
    while(1)
      {
        if(SW == 0)
          {
          PORTB = 0x66;
          MSDelay(100);
          PORTB = 0xCC;
          MSDelay(100);
          PORTB = 0x99;
          MSDelay(100);
          PORTB = 0x33;
          MSDelay(100);
          }
        else
          {
          PORTB = 0x66;
          MSDelay(100);
          PORTB = 0x33;
          MSDelay(100);
          PORTB = 0x99;
          MSDelay(100);
          PORTB = 0xCC;
          MSDelay(100);
          }
      }
  }

void MSDelay(unsigned int value)
  {
    unsigned int x, y;
    for(x=0;x<1275;x++)
      for(y=0;y<value;y++);
  }
```

DC MOTOR

DC Motor, doğru akım elektrik enerjisini mekanik enerjiye dönüştüren elektrik makinesidir. Herhangi bir iletkene doğru akım tatbik edildiğinde iletken, sabit bir manyetik alan oluşturur.

DC motorlar, **statorda oluşturulan sabit manyetik alanın rotorda oluşturulan sabit manyetik alanı itmesi ve çekmesi prensibine göre çalışır**. Statorda kuzey-güney ekseninde oluşan sabit manyetik alana karşı, rotorda bu eksenden belli bir açıda kayık olarak yerleştirilen sargıda ikinci bir sabit manyetik alan oluşturulur.

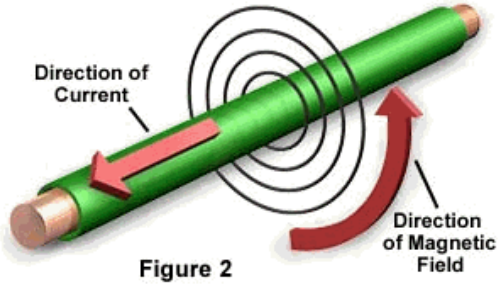
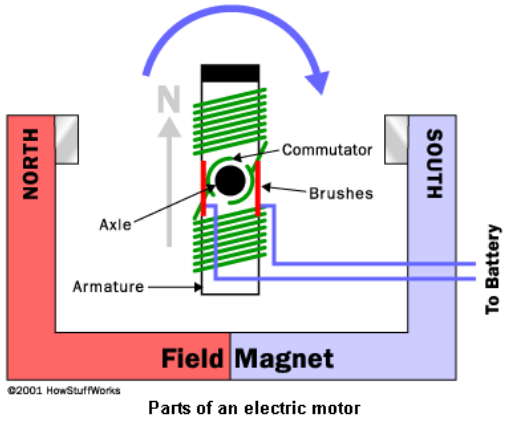
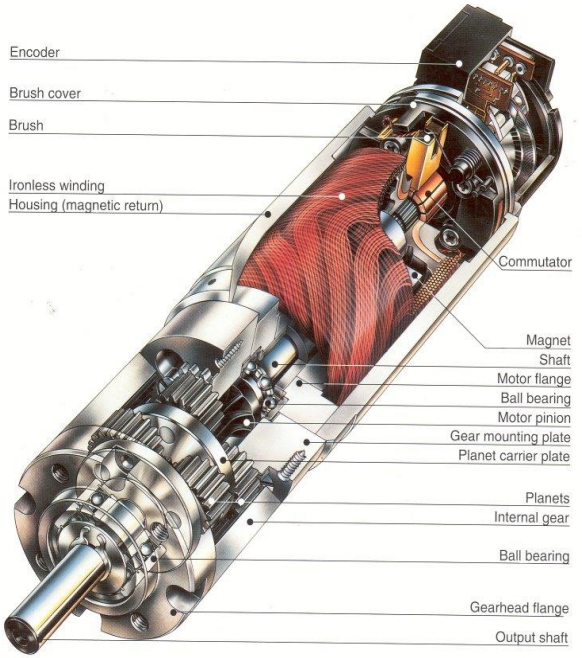


Figure 2

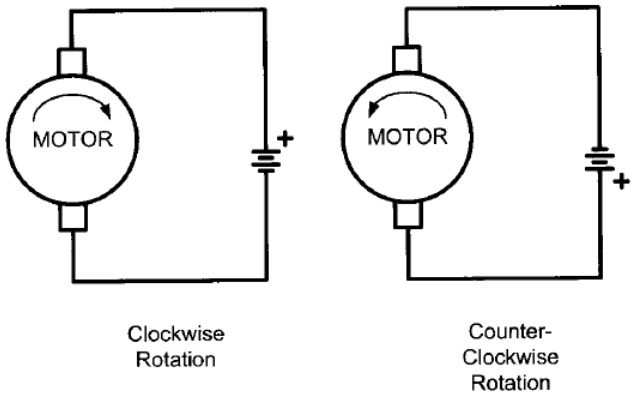


Parts of an electric motor

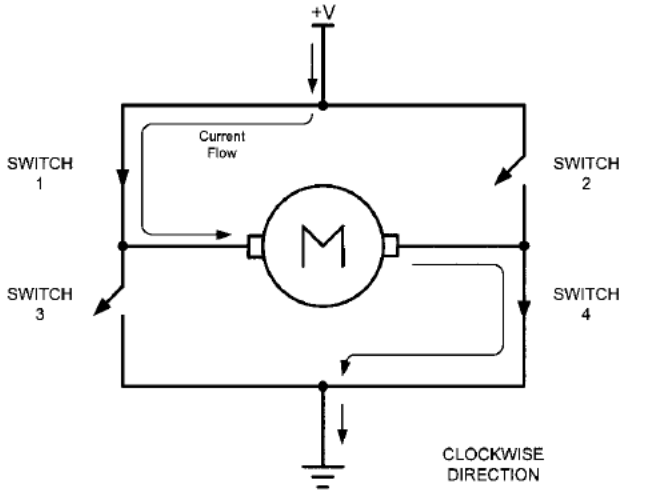


DC MOTOR

Step motorun aksine DC motorun hareketi süreklidir. Çoğu anakart üzerindeki küçük fanlarda, CPU soğutucuları üzerinde DC motor vardır.
Küçük boyutları ve maliyetlerinin düşüklüğü tercih edilme sebepleridir. Kutuplarına uygulanan DC gerilim motorun dönmesini sağlar, kutup uçları ters çevrildiğinde motor dönüş yönü de değişir.
Dönüş hızı doğrudan uçlarına uygulanan DC gerilimin büyüklüğü ile doğru orantılıdır. Fakat uygulamalarda genellikle PWM yöntemi ile dönüş hızı ayarlanır.



Motor Operation	SW1	SW2	SW3	SW4
Off	Open	Open	Open	Open
Clockwise	Closed	Open	Open	Closed
Counterclockwise	Open	Closed	Closed	Open
Invalid	Closed	Closed	Closed	Closed



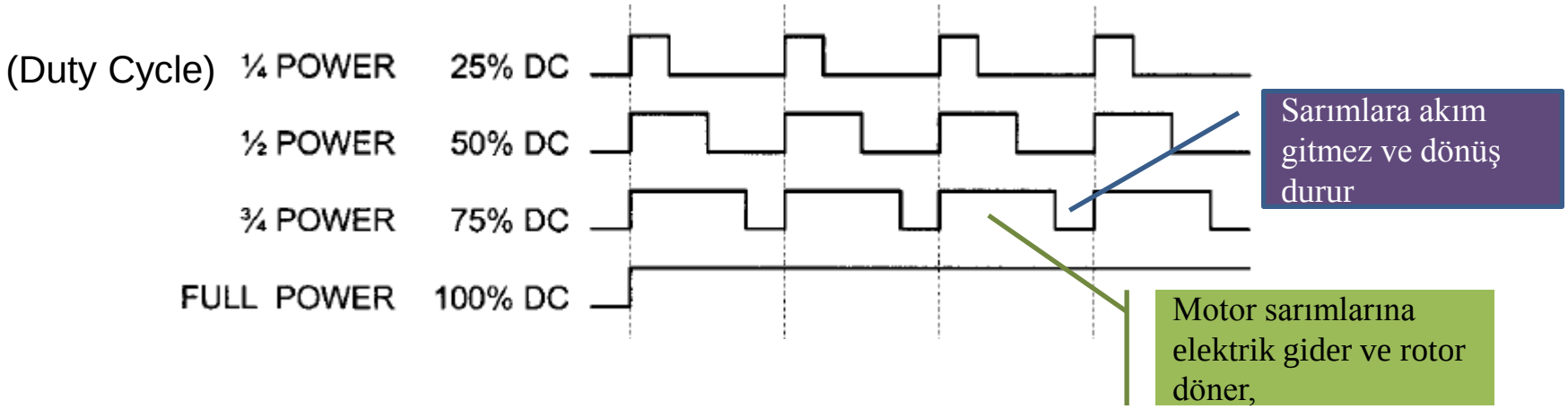
17-15. H-Bridge Motor Clockwise Configuration

DC MOTOR

DC motorun hızı üç faktöre bağlıdır:

- Yüke,
- Voltaja,
- Akıma

PWM (Pals Genişlik Modülasyonu) ile, ayarlanabilir bir frekans ve periyotta sayısal sinyal üretilebilir. Üretilen dalganın genişliği değiştikçe DC motor hızı da değişir, genişlik artarsa hız da artar



Donanımsal Olarak PWM Programlama

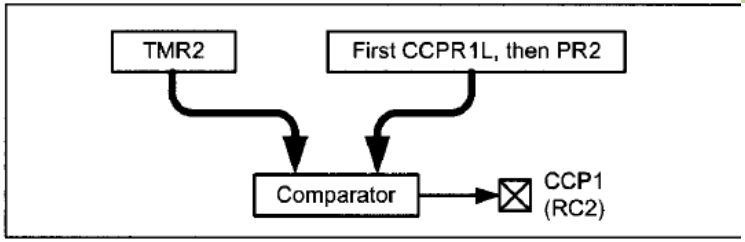


Figure 15-16. TMR2 and PR2 Role in Creating the Duty Cycle

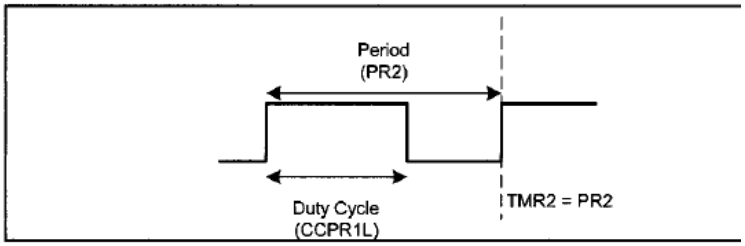


Figure 15-17. TMR2 Relation to CCPR1L and PR2 in PWM

TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
D6						D0

D7 Not used

TOUTPS3:TOUTPS0 D6-D3 Timer 2 Output Postscale Select bits
- 0 0 0 0 = 1:1 Postscale value
- 0 0 0 1 = 1:2 Postscale value
- 0 0 1 0 = 1:3 Postscale value
- 0 0 1 1 = 1:4 Postscale value
- 1 1 1 0 = 1:15 Postscale value
- 1 1 1 1 = 1:16 Postscale value

TMR2ON D2 Timer 2 ON and OFF Control bit
- 1 = Enable (start) Timer2
- 0 = Stop Timer2

T2CKPS1:T2CKPS0 D1–D0 Timer2 Clock Prescale Select bits
- 0 0 = Prescale is 1.
- 0 1 = Prescale is 4.
- 1 x = Prescale is 16.

Figure 15-14. T2CON (Timer2 Control) Register

						TMR2IF	TMR1IF

TMR2IF Timer 2 Interrupt overflow Flag bit
- 0 = TMR2 value is not equal to PR2 register.
- 1 = TMR2 value is equal to PR2 register.

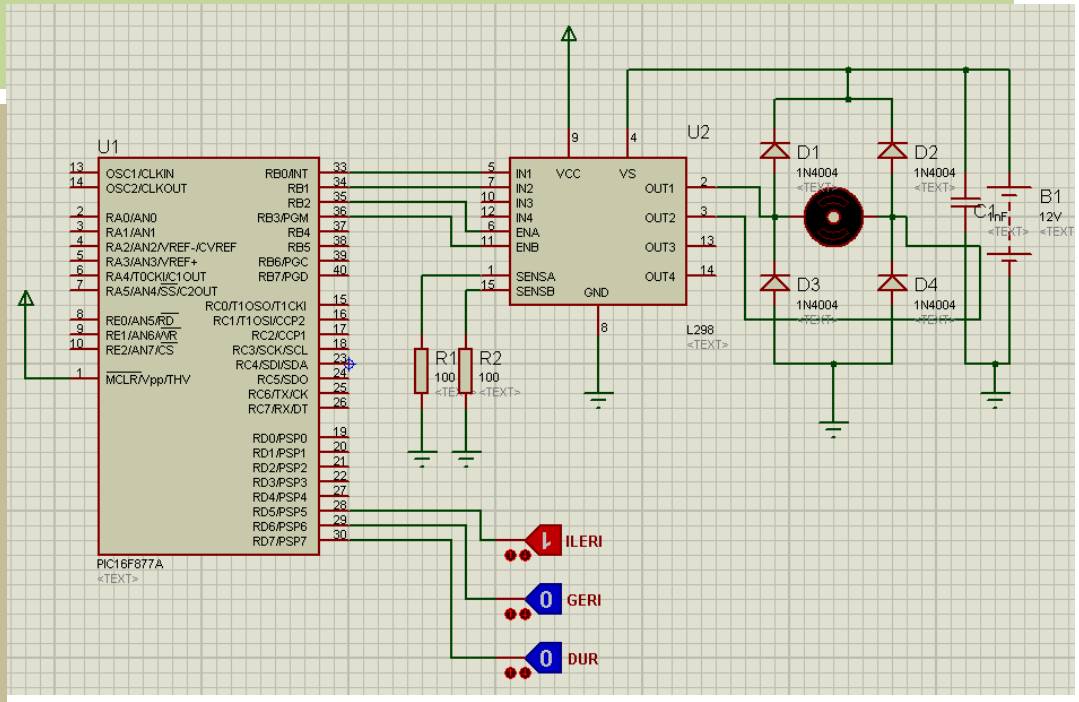
The location of the TMRxIF in the PIR register can vary in future products.

Figure 15-15. PIR1 (Peripheral Interrupt Flag Register 1) Has the TMR2IF Flag

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to CCPR1L for the higher 8 bits.
3. Set the CCP pin as an output.
4. Using the T2CON register, set the prescale value. See Figure 15-14.
5. Clear the TMR2 register.
6. Configure the CCP1CON register for PWM and set DC1B2:DC1B1 bits for the decimal portion of the duty cycle.
7. Start Timer2.

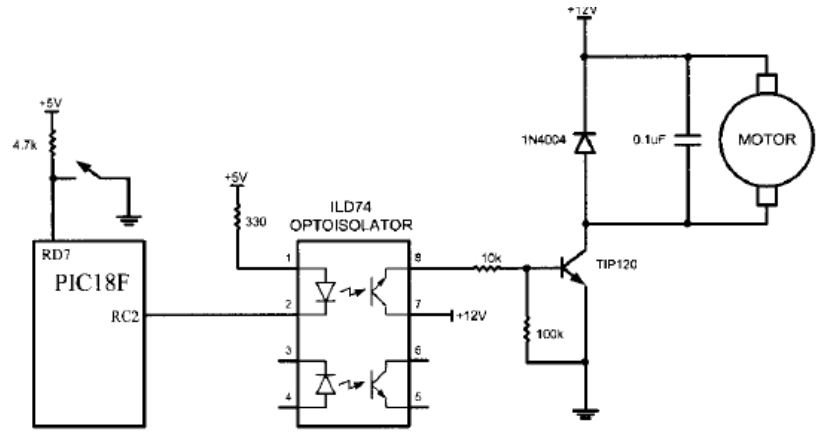
DC MOTOR

```
#include <xc.h>
void main(void)
{
PORTD=0x00;
PORTB=0x00;
TRISD=0xFF; // GIRIS
TRISB=0x00; // PORTB çıkıs
RB2=1; // A köprüsü seçiliyor
RB3=0;
for(;;)
{
if(RD5) //ileri
{
RB0=1;
RB1=0;
}
if(RD6) // Geri
{
RB0=0;
RB1=1;
}
if(RD7) // Dur
{
RB0=0;
RB1=0;
}
}
}
```



DC MOTOR

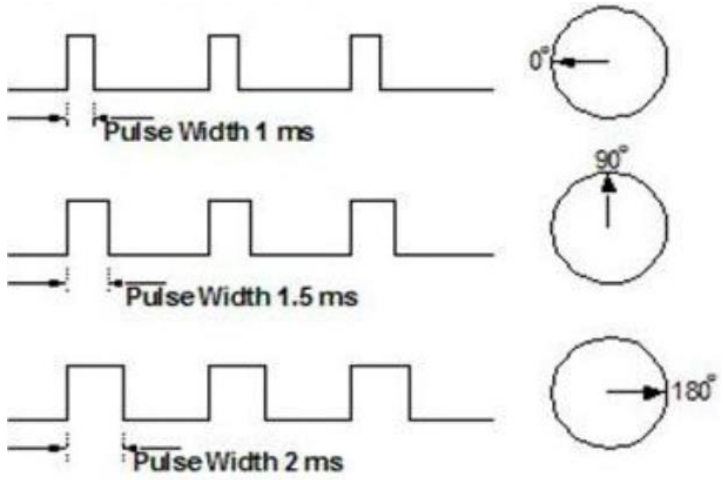
CCP ile DC Motor Kontrolü




```
#include <xc.h>
//#define btn RD7;
void bekle(int);
void main()
{
          TRISD=0x80;
          TRISB=0xFD;
          while(1) {
          if ( RD7 == 1) //%25 DC
                    {
                    RB1=1;
                    bekle(25);
                    RB1=0;
                    bekle(75);
                    }
          else
                    { //%50 DC
                    RB1=1;
                    bekle(50);
                    RB1=0;
                    bekle(50);
                    }
          }
}
void bekle(int sn)
{
unsigned int i,j;
for (int i=0; i<=1275;i++)
          for (int j=0; i<=sn;j++);
}
```

Servo MOTOR

Servo motorlar genel itibari ile içerlerinde bir DC motor ve şaft konum bilgisi devresi barındırırlar. Bu devre şaftın kaç derece döndüğünün algılanmasında kullanılır .
Servo motorlar 20ms periyotlu, 1ms'den 2ms'e kadar değişen duty cycle'lı PWM sinyali ile sürülürler. Verilen duty cycle periyotlarına göre servo motorun 0-180 derece aralığı arasında alacakları değerler yanda gösterilmiştir.
Burada sinyal verilirken örneğin 1,5ms ile 1ms arası 90'a bölünerek istenilen açıya gidilebilir.
Model uçak, araba ve robotik uygulamalarda yaygın olarak kullanılır



```
#include <xc.h>
#include "delay.h"
void main()
{
  TRISB =0x00; // PORTB cikis
  TRISD=0xFF;
  while(1)
  {
  if(RD5)//0 Derece Dön
  {
   for(;;){
   RB0 = 1;
   DelayUs(250);DelayUs(250);
   DelayUs(250);DelayUs(250);//1ms bekle
   RB0 = 0;
   DelayMs(19);
   if(RD6 | RD7) break;
  }         }
 if(RD6) //90 Derece Dön
  {
   for(;;){
   RB0 = 1;
   DelayUs(250);DelayUs(250);
   DelayUs(250);DelayUs(250);
   DelayUs(250);DelayUs(250);
   RB0 = 0;
   DelayMs(18);
   DelayUs(250);DelayUs(250);
   if(RD5 | RD7) break;
   } }
  if(RD7) //180 Derece Dön
  {
   for(;;){
   RB0 = 1;
   DelayMs(2);
   RB0 = 0;
   DelayMs(18);
   if(RD5 | RD6) break;
   } }
}}
```

Servo MOTOR