BMS 302 MİKROKONTROLCÜLER

CCP (CAPTURE-COMPARE-PWM)

Prof.Dr. Mutlu AVCI

Çukurova Üniversitesi Biyomedikal Mühendisliği Bölümü

Bahar 2020

- CCP birimi 3 değişik modda çalışan yani 3 değişik modül içeren bir birimidir.
- Bu çalışma modları Yakalama (Capture), Karşılaştırma (Compare) ve Darbe Genişlik Modülasyonu (PWM) modudur.
- PIC16F877A'da iki adet CCP birimi bulunmaktadır. (CCP1 ve CCP2)
- Her CCP birimi 16 bitlik Capture ve Compare kaydedicisi olarak ve 10 bitlik PWM çevrimi kaydedicisi olarak çalışabilecek 16 bitlik kaydediciye sahiptir.

- CCP donanımı zamanlayıcı/sayıcı birimlerini kullanarak çalışır.
- Capture ve Compare modları Timer1'i, PWM modu ise Timer2'yi kullanır.
- CCP1 ucu RC2 (17 numaralı), CCP2 ucu RC1 (16 numaralı) dir.
- CCPx birimi CCP1CON ve CCP2CON isimli iki register ile kontrol edilir.

CAPTURE (YAKALAMA) MODU

- CCPx (x yerine 1 veya 2 gelir) uçlarından gelen sinyal ayarlamaya göre yakalanınca TMR1 registerinin içeriği 16 bitlik CCP_x registerine (CCP_x_HIGH:CCP_x_LOW register çiftinden oluşur) yazılır ve bir kesme oluşur.

- Sinyalin her düşen kenarında
- Sinyalin her yükselen kenarında
- Sinyalin her 4. yükselen kenarında
- Sinyalin her 16. yükselen kenarında

- CCP_x, CCP_x_HIGH ve CCP_x_LOW registerlerine doğruda değer atam yapılabilir.

ÖR:

```
CCP_1=600;
CCP_2_HIGH=0x45;
```

gibi

COMPARE (KARŞILAŞTIRMA) MODU

- CCP_x registerinin içeriğindeki değer ile TMR1 registerinin içeriğindeki değer sürekli kontrol edilir. Eşleşme olursa CCPx kesmesi meydana gelir.

- Bu işleme aşağıdaki ayarlardan biri eşlik eder:

- CCPx ucu lojik 1 olsun kesme meydana gelsin
- CCPx ucu lojik 0 olsun kesme meydana gelsin
- CPPx ucunun durumu değişmesin kesme meydana gelsin
- CCPx ucunun durumu değişmesin, kesme meydana gelsin ve Timer1 resetlensin

PWM (PULSE WIDTH MODULATION) MODU

- CCP birimi PWM modu, istenen CCPX ucundan istenen görev çevirimine (duty cycle – doluluk oranı) sahip PWM sinyali elde etmek için kullanılır.
- PWM birimi Timer2 zamanlayıcısını kullanır. PIC16F877A'da iki adet CCP modülü olduğundan iki adet de PWM çıkış ucu vardır.
- Bu uçlar RC1/CCP2 ve RC2/CCP1'dir.
- PWM sinyali; denetleyici osilatör frekansı, Timer2 zamanlayıcı biriminin PR2 değeri ve bölme oranı değeri ile belirlenir.

- Timer2 fonksiyonunda bulunan *postscaler* değerinin PWM sinyaline bir etkisi yoktur. PWM sinyalinin periyodu ve frekansı ;

TPWM = Tkomut x (Timer2 Bölme Oranı)x(PR2 değeri +1)

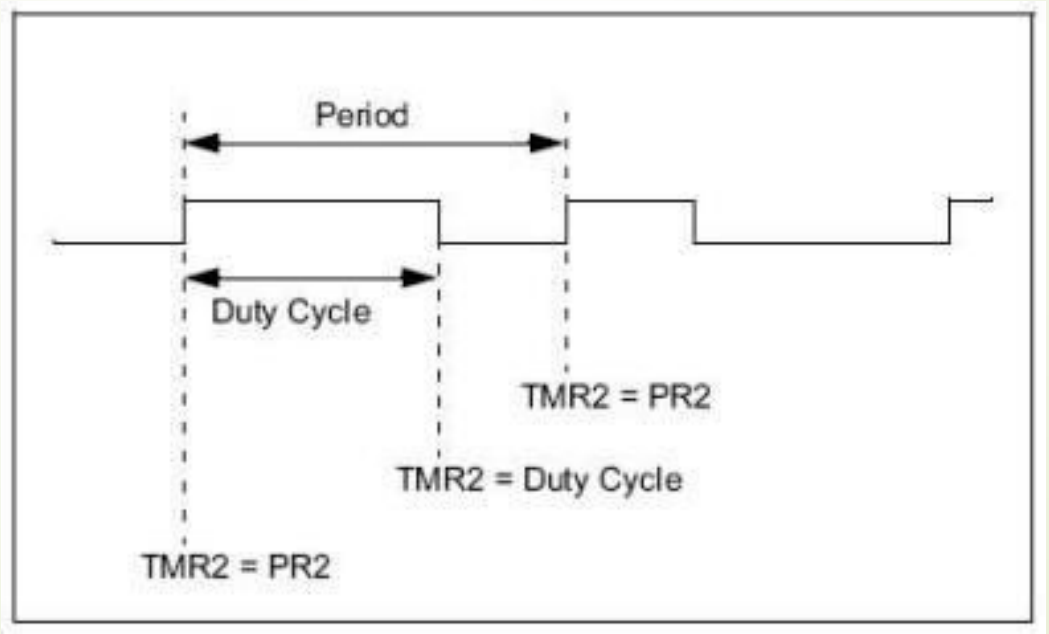
Tkomut(sn) = 1/fkomut(hz)

fkomut = Kontrolcü Osilatör Frekansı / 4 = fosc/4

fPWM = 1/ TPWM

şeklinde hesaplanır

- PWM ile oluşturulan bir dalga şekli:



- PWM dalga şeklinde görüldüğü gibi PWM sinyalinin bir görev çevrimi – doluluk oranı – (duty cycle) bölümü vardır. Bu bölüm PWM sinyalinde lojik 1 olan süredir.

- Bu görev çevirimi süresi PWM periyodundan uzun olamaz.

- Görev çevirimi periyodunun hesaplanmasında PWM görev çevirimi (duty cycle) değeri kullanılır. Bu değer en fazla 10 bitlik bir değer olabilir.

- Örneğin ;

#use delay (clock=4000000)

setup_timer_2(T2_DIV_BY_4, 120, 1);

komutları kullanılarak bir PWM sinyalinin frekansı ;

fkomut = fosc/4 = 1 Mhz

TPWM = 1/fkomut x (PR2+1) x (Timer2 bölme oranı) = 1/1 MHz x (120+1) x 4 = 0.484 ms

fPWM =1/ TPWM = 1/ 0.484 ms = 2.066 kHz

şeklinde hesaplanır.

SET_PWMX_DUTY( ) Fonksiyonu:

- PWM modunda oluşturulan sinyalin görev çevrim süresini (duty cycle) belirlemeye yarayan fonksiyondur. Kullanımı ;

set_pwmx_duty(değer);

set_pwm2_duty(200);

- X kısmına kullanılacak CCP modülü birimi numarası yazılır.

- Fonksiyondaki değer kısmına 8 veya 10 bitlik sabit veya değişken yazılabilir. Bu değer PWM görev çevirimi (duty cycle) çevrimi hesaplanmasında kullanılır.

Ör:

```
setup_timer_2(T2_DIV_BY_4,120,1);
set_pwm1_duty(30);  // %25 görev çevrimi
set_pwm1_duty(90); // %75 görev çevrimi
```

SETUP_CCPX ( ) Fonksiyonu

- Bu fonksiyon ile mikro kontrolcü içinde bulunan CCP modülü ile ilgili ayarlamalar yapılır. Kullanımı ;

setup_ccp1 (mod) ;

setup_ccp2(mod);

şeklindedir.

"mod" kısmına kullanılmak istenen moda göre sabit tanımlamalar yazılır.

- PWM modu için mod kısmına, ***CCP_PWM*** yazılır.
- CCP birimini devre dışı yapmak için ***CCP_OFF*** yazılır.
- CAPTURE modu için:

CCP_CAPTURE_FE

CCP_CAPTURE_RE

CCP_CAPTURE_DIV_4

CCP_CAPTURE_DIV_16

- COMPARE Modu için:

CCP_COMPARE_SET_ON_MATCH

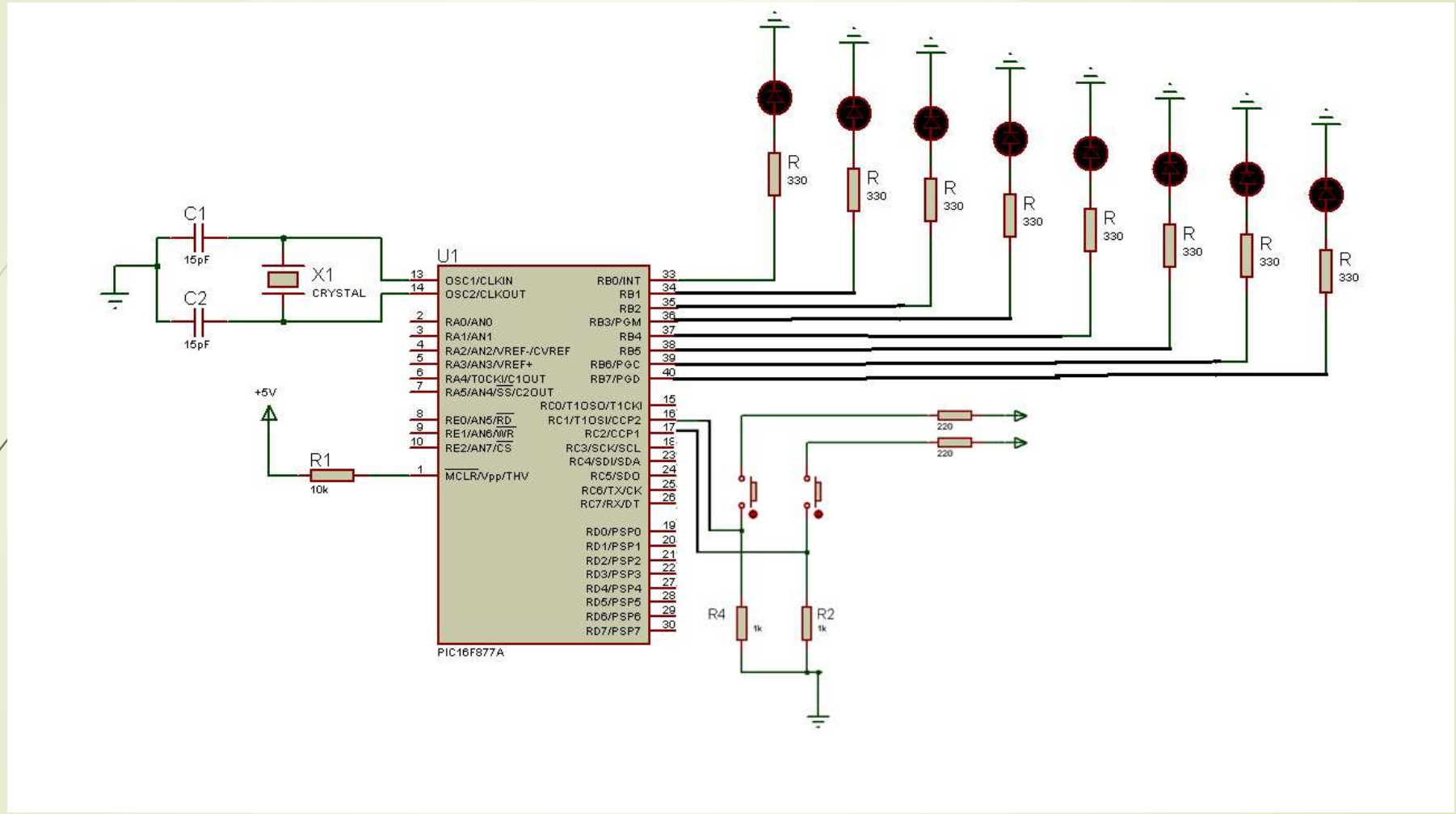
CCP_COMPARE_CLR_ON_MATCH

CCP_COMPARE_INT

CCP_COMPARE_RESET_TIMER

Örnek:

CCP1 ucuna uygulanan işaretin her 4. yükselen kenarında bir sayaç değişkeninin değerini 1 arttıran, CCP2 ucuna uygulanan işaretin her yükselen kenar değerinde sayaç değişkeninin değerini bir azaltan ve sayaç değişkeninin değerini B portuna bağlı LED'ler ile gösteren devre için gerekli CCS C program kodlarını yazınız.


C1
15pF
C2
15pF
X1
CRYSTAL
U1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
PIC16F877A
+5V
R1
10k
R
330
220
R4
1k
R2
1k

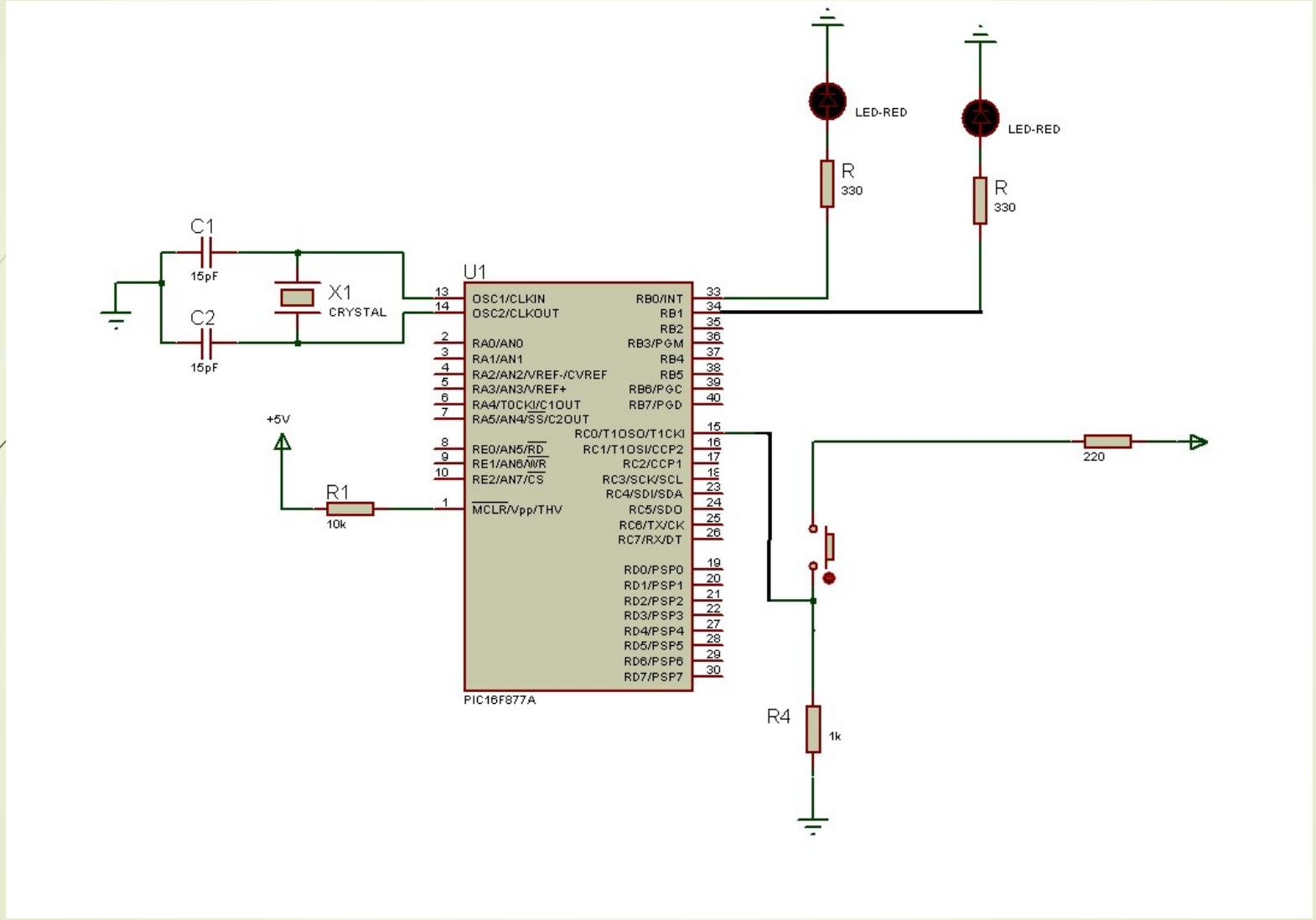
```
#include <prog.h>
#use delay(clock=4000000)
#use fast_io(c)
int8 Sayac=0;
#INT_CCP1
void yakala1(){
Sayac++;
output_B(Sayac);
}
#INT_CCP2
void yakala2(){
Sayac--;
if (Sayac<=0) Sayac=0;
output_B(Sayac);}
```

```
void main(){
Set_tris_C(0x07);
output_B(0x00);

setup_CCP1(CCP_CAPTUE_DIV_4);
Setup_CCP2(CCP_CAPTURE_RE);
CCP_1=0;
CCP_2=0;
enable_interrupts(INT_CCP1);
enable_interrupts(INT_CCP2);
enable_interrupts(GLOBAL);
while(1);
}
```

Örnek:

Timer1 ucundan uygulanan her 5 saat darbesi için RB0 ucuna bağlı LED'in, her 10 saat darbesi için RB1 ucuna bağlı LED'in durum değiştirdiği CCS C programını yazın.


C1
15pF
C2
15pF
X1
CRYSTAL
U1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
PIC16F877A
+5V
R1
10k
LED-RED
R
330
LED-RED
R
330
220
R4
1k

```
#include<prog.h>
#use delay(clock=4000000)
#use fast_io(C)

#INT_CCP1
void karsilastir1(){
output_toggle(pin_B0);}

#INT_CCP2
void karsilastir2(){
set_timer1(0);
output_toggle(pin_B1);}
```

```
void main(){
set_tris_C(0x01);
output_B(0);
setup_CCP1(CCP_COMPARE_INT);
Setp_CCP2(CCP_COMPARE_INT);

setup_timer_1(T1_EXTERNAL_SYNC | T1_DIV_BY_1);
CCP_1=5;
CCP_2=10;
set_timer1(0);
enable_interrupts(INT_CCP1);
enable_interrupts(INT_CCP2);
enable_interrupts(GLOBAL);
while(1);}
```

Örnek:

RA0 ve RA1 uçlarına bağlı girişler ile görev çevrimi her basış için %1 arttırılıp azaltılabilen, başlangıçta CCP1 çıkışının görev çevrimi %20, CCP2 ucunun görev çevrimi %50 olan, 50Hz frekanslı PWM çıkışı üreten devrenin CCS C programını yazın.

f = 50 Hz ise T = 1/f = 20 ms'dir. Frekans düşük, clock frekansı da düşük olmalıdır.

%1 artış azalış için PR2'nin 99 veya 199 olarak kullanımı uygundur.

En yüksek TMR2 bölme oranı 16 kullanılırsa

$T_{PWM} = T_{komut} * (PR2+1)*(TMR2\_bölme\_oranı)$

$20m = T_{komut} * 100 * 16$

$f_{komut} = 1 / T_{komut}$ = 80 kHz bulunur.

$f_{OSC} = 4 * f_{komut}$ = 320 kHz

PIC'i 320 kHz frekanslı bir kristal osilatörle sürmemiz gereklidir.

```
#include<prog.h>
#use delay(clock=320000)
#use fast_io(C)
int8 k1=20,k2=50;
void main(){
set_tris_C(0);
setup_CCP1(CCP_PWM);
setup_CCP2(CCP_PWM);
setup_timer_2(T2_DIV_BY_16,99,1);
set_pwm1_duty(k1);
set_pwm2_duty(k2);
```

```
while(1){
if(input(pin_A0)){
while(input(pin_A0));
k1=k1+1;
k2=k2+1;
if(k1>=99) k1=99;
if(k2>=99) k2=99;
Set_pwm1_duty(k1);
set_pwm2_duty(k2);
}
```

```
if(input(pin_A1)){
while(input(pin_A1));

k1=k1-1;
k2=k2-1;
İf (k1<=1) k1=1;
İf (k2<=1) k2=1;
Set_pwm1_duty(k1);
set_pwm2_duty(k2);
}
}
```