

PIC enerjilendiğinde PORTB nin 0. biti 1 olacak

Akış diyagramı

- başla
- LIST=16F84
- PORTB yi temizle
- BANK1 e geç
- PORTB nin uçlarını çıkış olarak yönlendir
- BANK 0 a geç
- PORT B nin 0. bitini 1 yap
- SON

Program

```
LIST          P=16F84          ;pic tanıtması
CLRF          h'06'           ;PORTB çıkışlarını 0 yap
BSF          h'03', 5         ;BANK1 e geç
CLRF          h'86'           ;PORTB uçlarını çıkış yap
BCF          h'03', 5         ;BANK0 a geç
BSF          h'06', 0         ;PORTB 0. bitini 1 yap
END           ;programı bitir
```

#PIC enerjilendiğinde PORTA içeriğinin tersini PORTB de karşılık gelen biti 0 olacak

Akış diyagramı

- başla
- tanıt
- PORTB çıkış PORTA giriş
- PORTA oku
- PORTB gönder
- Son

Program

```
LIST          P=16F84
PORTA         EQU          h'05'
PORTB         EQU          h'06'
STATUS        EQU          h'03'
TRISA         EQU          h'85'
TRISB         EQU          h'86'
              CLRF         PORTB ;PORTB içeriğini 0 yap
              BSF         STATUS, 5 ;BANK1 e geç
              CLRF         TRISB   ;PORTB çıkış yap
              MOVLW        h'FF'   ;W registerine h'FF' yükle
              MOVWF        TRISA   ;PORTA yı giriş yap
              BCF         STATUS, 5 ;BANK0 geç
BASLA
              MOVF         PORTA, W ;PORTAyı W ye yaz
              MOVWF        PORTB ;durumu PORTB ye aktar
DONGU
              GOTO        DONGU
              END
```

#PORTA nın 1.biti 1 olduğunda PORTB nin 0. bitini 1 yapan program

Akış diyagramı

- başla
- tanıt
- PORTB sil
- PORTB çıkış, PORTA giriş
- PORTA bit 1, 0 mı?
- Hayır → tekrar test et. Evet → devam et
- PORTB 0. biti 1 yap
- Son

```

Program
LIST          P=16F84
PORTA EQU     h'05'
PORTB EQU     h'06'
STATUS EQU    h'03'
TRISA EQU     h'85'
TRISB EQU     h'86'
              CLRF      PORTB          ;PORTB temizle
              BSF       STATUS, 5      ;BANK1e geç
              CLRF      TRISB          ;PORTB uçlarını çıkış yap
              MOVLW     h'FF'          ;W registerine h'FF' yükle
              MOVWF     TRISA          ;PORTA uçlarını giriş yap
              BCF       STATUS, 5      ;BANK0 geç

TEST_PORTA
              BTFSC     PORTA, 1        ;PORTA 1. bitini test et
              GOTO      TEST_PORTA     ;0 değilse tekrar test et
              BSF       PORTB, 0        ;PORTB 0. biti 1 yap

DONGU
              GOTO      DONGU
              END

```

#PORTA 1. biti 10 defa 1 olunca PORTB 0. bitini 1 yapan program

Akış diyagramı

- başla
- tanıt
- PORTB temizle
- W registerine 10 sayısını ata
- W registerini sayaca yükle
- PORTA bit1 0 mı? Hayır→ tekrar sor evet→ devam et
- NOP×5
- Sayaç=sayaç-1
- Sayaç 0 mı? Hayır→PORTA bit1 0 mı sorusuna git evet→devam et
- PORTB 0. biti 1 yap
- Son

```

Program
LIST          P=16F84
              INCLUDE    "P16F84.INC"
SAYAC EQU     h'0C'          ;SAYAC registerinin adresi
              CLRF      PORTB
              BSF       STATUS, 5
              CLRF      TRISB
              MOVLW     h'FF'
              MOVWF     TRISA
              BCF       STATUS, 5

BASLA
              MOVLW     D'10'          ;W registerine 10 sayısını ata
              MOVWF     SAYAC          ;W registerini sayaca yükle

TEST
              BTFSC     PORTA, 1        ;PORTA 1. biti 1 mi
              GOTO      TEST            ;hayır; tekrar test et
              NOP                    ;1 sayıkl bekle
              "                ;"
              "                ;"
              "                ;"
              "                ;"
              DECFSZ    SAYAC, F        ;SAYAC=SAYAC-1, SAYAC=0 mı?
              GOTO      TEST            ;hayır, PORTA test et
              BSF       PORTB, 0        ;evet PORTB 0. bitini 1 yap
              END

```

#SUBLW komutu ile yapılan bir karşılaştırma sonucunda STATUS registerinin içeriğini görüntüleyen program

Akış diyagramı

- başla
- tanıt
- PORTB çıkış
- Test sayısı→W
- W=sabit-test sayısı
- STATUS→PORTB
- Sonsuz döngü
- Son

Program

```
LIST P=16F84
INCLUDE "P16F84.INC"
CLRF PORTB ;PORTB temizle
BSF STATUS, 5 ;BANK2 ye geç
CLRF TRISB ;PORTB çıkış yap
BCF STATUS, 5 ;BANK1 geç

BASLA
MOVLW h'04' ;W h'04' yükle
SUBLW h'05' ;W=h'05'-h'04'
MOVF STATUS, W ;STATUS W registerine yaz
MOVWF PORTB ;STATUS u çıkışa aktar

DONGU
GOTO DONGU
END
```

#SUBWF komutu kullanılarak oluşturulan döngü programı

Akış diyagramı

- başla
- tanıt
- PORTB çıkış
- SAYAC→h'00'
- SAYAC→PORTB
- SAYAC=SAYAC+1
- h'07'→W
- SAYAC-W→W
- STATUS bit2 1mi? Evet: end komutuna geç. Hayır:SAYAC→PORTB
- Son

Program

```
LIST P=16F84
INCLUDE "P16F84.INC"
SAYAC EQU h'0C' ;SAYAC registeri tanımla
CLRF PORTB ;PORTB temizle
BSF STATUS, 5 ;BANK2 ye geç
CLRF TRISB ;PORTB çıkış
BCF STATUS, 5 ;BANK1 geç

BASLA
CLRF PORTB ;PORTB sil
CLRF SAYAC ;SAYAC sil

TEKRAR
MOVF SAYAC, W ;SAYACı W registerine yükle
MOVWF PORTB ;W registerini PORTB gönder
INCF SAYAC, F ;SAYAC+1→SAYAC
MOVLW h'07' ;W registerine h'07' yükle
SUBWF SAYAC, W ;SAYAC-W→SAYAC
BTFSS STATUS, 2 ;STATUS bit2 1mi?
```

```

DONGU      GOTO      TEKRAR
           GOTO      DONGU
           END

```

#Zaman gecikme döngüsü kullanarak PORTB nin çıkışlarını belirli zaman aralıklarıyla değiştiren program

Akış diyagramı

- başla
- tanıt
- PORTB→çıkış
- PORTB→h'00'
- CALL gecikme
- PORTB→h'FF'
- CALL gecikme buradan PORTB→h'00' ye yönelen

Program

```

LIST      P=16F84
INCLUDE   "P16F84.INC"
SAYAC1    EQU      h'0C'
SAYAC2    EQU      h'0D'
          CLRF     PORTB
          BSF     STATUS, 5
          CLRF     TRISB
          BCF     STATUS, 5

TEKRAR
          MOVLW   h'00'
          MOVWF   PORTB
          CALL    GECIKME
          MOVLW   h'FF'
          MOVWF   PORTB
          CALL    GECIKME
          GOTO    TEKRAR

GECIKME   ;alt program başlangıcı
          MOVLW   h'FF'
          MOVWF   SAYAC1

DONGU1
          MOVLW   h'FF'
          MOVWF   SAYAC2

DONGU2
          DECFSZ  SAYAC2, F
          GOTO    DONGU2
          DECFSZ  SAYAC1, F
          GOTO    DONGU1
          RETURN
          END

```

#PORTA nın 1 numaralı bitine 10 defa 1 uygulanması ile PORTB nin tüm çıkışlarını 1 yapan program

Akış diyagramı

- başla
- tanıt
- PORTA giriş, PORTB çıkış
- PORTA 1. bit 0 mı? (A) Hayır→tekrar sor, evet→devam et
- MEM=MEM+1
- MEM→W
- W=h'10'-W
- STATUS 2. bit 0 mı? Evet→CALL GECIKME→A hayır→devam et
- PORTB→h'FF'
- Son

```

Program
LIST          P=16F84
INCLUDE       "P16F84.INC"
SAYAC1       EQU      h'0C'
SAYAC2       EQU      h'0D'
MEM          EQU      h'0E'
CLRF         PORTB    ;PORTB sıfırla
BSF          STATUS, 5 ;BANK2 geç
CLRF         TRISB    ;PORTA nın 1. biti giriş
BSF          TRISA, 1 ;PORTB ucları çıkış
BCF          STATUS, 5 ;BANK1e geç
CLRF         MEM      ;MEM registerini sıfırla

TEKRAR
BTFSC        PORTA, 1 ;PORTA nın 1. biti 0 mı?
GOTO         TEKRAR  ;hayır tekrar test et
INCF         MEM      ;evet, MEM=MEM+1
MOVF         MEM, W   ;MEM→W
SUBLW        d'10'    ;W=d'10'-w
BTFSC        STATUS, 2 ;STATUS un 2. biti 0 mı?
GOTO         YAK      ;hayır Z=1
CALL         GECIKME  ;evet buton arkının sönmesini bekle
GOTO         TEKRAR  ;butonu test için başa git

YAK
MOVLW        h'FF'    ;W→h'FF'
MOVWF        PORTB    ;PORTB deki tüm çıkışları 1 yap

DONGU
GOTO         DONGU
;=====; GECIKME ALT PROGRAMI=====
GECIKME
MOVLW        h'FF'
MOVWF        SAYAC1

DONGU1
MOVLW        h'FF'
MOVWF        SAYAC2

DONGU2
DECFSZ       SAYAC2, F
GOTO         DONGU2
DECFSZ       SAYAC1, F
GOTO         DONGU1
RETURN
END

```

#PORTB çıkışlarını sırayla kaydıran, işlemin bitmesi ile çıkışları 0 yapan program

Akış diyagramı

- başla
- tanıt
- PORTB çıkış
- h'01'→PORTB
- CALL GECIKME
- RLF PORTB
- STATUS 0. bit 1mi? Hayır→CALL GECIKME evet→Son
- Son

```

Program
LIST          P=16F84
INCLUDE       "P16F84.INC"
SAYAC1       EQU      h'0C'
SAYAC2       EQU      h'0D'
CLRF         PORTB    ;PORTB sıfırla
BCF          STATUS, 0 ;CF sıfırla

```

	BSF	STATUS, 5	;BANK1 geç
	CLRF	TRISB	,PORTB çıkış
	BCF	STATUS, 5	;BANK0 geç
	MOVLW	h'01'	;b'00000001' W ye yükle
TEKRAR	MOVWF	PORTB	;W registerini PORTB ye yükle
	CALL	GECIKME	;gecikme yap
	RLF	PORTB, F	;PORTB veriyi sola kaydır
	BTFSS	STATUS, 0	;CF 1mi?
DONGU	GOTO	TEKRAR	;hayır
	GOTO	DONGU	;evet sonsuz döngüye gir
GECIKME			;gecikme alt programı
	MOVLW	h'FF'	
	MOVWF	SAYAC1	
DONGU1			
	MOVLW	h'FF'	
	MOVWF	SAYAC2	
DONGU2			
	DECFSZ	SAYAC2, F	
	GOTO	DONGU2	
	DECFSZ	SAYAC1, F	
	GOTO	DONGU1	
	RETURN		
	END		

#karaşımsek devresi programı

Akış diyagramı

- başla
- tanıt
- PORTB çıkış
- h'01'→PORTB
- CALL GECIKME(1)
- RLF PORTB
- PORTB 7. bit 1 mi? Hayır→CALL GECIKME(1) evet→devam
- CALL GECIKME(2)
- RRF PORTB
- PORTB 0. bit 1mi? Hayır→CALL GECIKME(2) evet→CALL GECIKME(1)

Program

	LIST	P16F84	
	INCLUDE	"P16F84.INC"	
SAYAC1	EQU	h'0C'	
SAYAC2	EQU	h'0D'	
	CLRF	PORTB	
	BCF	STATUS, 0	;CF sıfırla
	BSF	STATUS, 5	
	CLRF	TRISB	
	BCF	STATUS, 5	
	MOVLW	h'01'	;b'00000001'→ W
	MOVWF	PORTB	;W→PORTB
SOL			
	CALL	GECIKME	;gecikme yap
	RLF	PORTB, F	;PORTB verisini Sola kaydır
	BTFSS	PORTB, 7	;PORTB 7. bit 1 mi?
	GOTO	SOL	;hayır sola kaydır
SAG			
	CALL	GECIKME	;gecikme yap
	RRF	PORTB, F	;PORTB veriyi sağa kaydır
	BTFSS	PORTB, 0	;PORTB 0. bit 1 mi?
	GOTO	SAG	;evet sağa kaydır

```

GECIKME      GOTO      SOL                      ;evet sola kaydır
                                           ;gecikme alt programı
                                           MOVLW      h'FF'
                                           MOVWF      SAYAC1
DONGU1
                                           MOVLW      h'FF'
                                           MOVWF      SAYAC2
DONGU2
DECFSZ       SAYAC2, F
GOTO         DONGU2
DECFSZ       SAYAC1, F
GOTO         DONGU1
RETURN
END

```

#PORTB çıkışlarını ilk dört daha sonra son dört bitini 1 yapan program

Akış diyagramı

- başla
- tanıt
- PORTB çıkış
- h'0F'→PORTB
- COMF PORTB
- CALL GECIKME buradan tekrar COMF PORTB ye dön

Program

```

LIST         P=16F84
INCLUDE      "P16F84.INC"
SAYAC1      EQU      h'0C'
SAYAC2      EQU      h'0D'
CLRF        PORTB
BSF         STATUS, 5
CLRF        TRISB
BCF         STATUS, 5
MOVLW      h'0F'          ;b'00001111'→ W
MOVWF      PORTB         ;W→PORTB
TERSLE
COMF        PORTB, F     ;PORTB verisini tersle
CALL        GECIKME     ;gecikme yap
GOTO        TERSLE
GECIKME
MOVLW      h'FF'
MOVWF      SAYAC1
DONGU1
MOVLW      h'FF'
MOVWF      SAYAC2
DONGU2
DECFSZ     SAYAC2, F
GOTO       DONGU2
DECFSZ     SAYAC1, F
GOTO       DONGU1
RETURN
END

```

#PORTA nın 1. ve 2. bitlerinin her ikisinde 1 olması durumunda PORTB nin 0. bitini 1 yapan program

Akış diyagramı

- başla
- tanıt
- PORTA giriş
- PORTB çıkış
- W→b'00011001'
- PORTA 1. ve 2. bit 1 mi?
- STATUS 2. bit 1 mi? Hayır→W→b'00011001' evet→devam

- W→h'01'
- W→PORTB
- Son

Program

```

LIST          P=16F84
INCLUDE      "P16F84.INC"
CLRF        PORTB           ;PORTB sil
BSF         STATUS, 5      ;BANK1 geç
MOVLW      h'FF'          ;h'FF'→W
MOVWF      TRISA           ;PORTA giriş
CLRF        TRISB          ;PORTB çıkış
BCF         STATUS, 5      ;BANK0 geç

TEST_PORTA
MOVLW      b'00011001'     ;W ye yükle
XORWF      PORTA, W        ;PORTA XOR W→W
BTFS       STATUS, 2       ;Z flag 1 mi?
GOTO       TEST_PORTA     ;hayır PORTA tekrar test et

YAK
MOVLW      h'01'          ;evet h'01'→w
MOVWF      PORTB          ;PORTB 0. bit 1 yap

DONGU
GOTO       DONGU

```

#W registerdeki h'5A' sayısı ile h'53' sayısını toplayıp sonucu PORTB ye aktaran program

Akış diyagramı

- başla
- tanıt
- PORTB çıkış
- h'5A'→W
- W→W+h'53'
- PORTB→W
- Son

Program

```

LIST          P=16F84
INCLUDE      "P16F84.INC"
CLRF        PORTB
BSF         STATUS, 5
CLRF        TRISB
BCF         STATUS, 5
MOLW      h'5A'
ADDLW      h'53'
MOVWF      PORTB

DONGU
GOTO       DONGU
END

```

#h'61A3' ve h'2EE0' sayılarını toplayan program. Program alt byteı PORTB ye aktarır, üst byte toplamını görmek için A1 butonuna basılır

2 byte lık h'61A3' sayısına A, h'2EE0' sayısına da B dersek bu sayılar 1 bytelik veri depolayabilen 2 tane file register kullanmamız gerekir.

Programda kullanılan file registerlerin adları:

```

A=61A3      61→AH          B=2EE0      2E→BH
            A3→AL          E0→BL

```

Program

```

LIST          P=16F84
INCLUDE      "P16F84.INC"
CLRF        PORTB          ;PORTB sil
BSF         STATUS, 5      ;BANK1 geç

```

```

        CLRFB          TRISB          ;PORTB çıkış
        MOVLW         h'FF'          ;registre sayıyı yükle
        BCF           STATUS, 5      ;BANK0 geç
AL      EQU          h'0C'          ;AL registeri adresi
AH      EQU          h'0D'
BL      EQU          h'0E'
BH      EQU          h'0F'
BASLA

        MOVLW         h'A3'
        MOVWF        AL
        MOVLW         h'61'
        MOVWF        AH
        MOVLW         h'E0'
        MOVWF        BL
        MOVLW         h'2E'
        MOVWF        BH

TOPLA

        MOVF          AL, F          ;AL→W
        ADDWF         BL, F          ;BL=BL+W(AL) alt byte toplamı
        BTFSC        STATUS, 0      ;CF=1?
        INCF          BH, F          ;evet, BH=BH+1
        MOVF          AH, W
        ADDWF         BH, F          ;BH=BH+W(AH) üst byte toplamı
ALT_BYTE_GOSTER
        MOVF          BL, W
        MOVWF        PORTB          ;alt byte toplamını göster

TEST_A1

        BTFSC        PORTA, 1        ;A1 butonuna basılı mı
        GOTO         TEST_A1        ;hayır, tekrar test et

UST_BYTE_GOSTER
        MOVF          BH, W
        MOVWF        PORTB          ;üst byte toplamını göster

DONGU

        GOTO         DONGU
        END

```

#PORTB registeri içerisindeki h'5A' sayısından W registeri içerisindeki h'53' sayısını çıkaran, sonucu PORTB ye aktaran program

Akış diyagramı

- başla
- tanıt
- PORTB çıkış
- h'5A'→W
- W→PORTB
- h'53'→W
- PORTB-W→PORTB
- Son

Program

```

LIST      P=16F84
INCLUDE   "P16F84.INC"
        CLRFB          PORTB
        BSF           STATUS, 5
        CLRFB          TRISB
        BCF           STATUS, 5
        MOVLW         h'5A'
        MOVWF        PORTB
        MOVLW         h'53'
        SUBWF        PORTB, F

DONGU

```

```
GOTO DONGU
END
```

#h'0004' sayısından h'0001' sayısını çıkaran program. Program çalıştığında çıkarma sonucunun alt bytei PORTB de görülür. Üst bytei görmek için A1 butonuna basılır

```
A=0004      00→AH      B=0001      00→BH
            04→AL      01→BL

LIST        P=16F84
INCLUDE     "P16F84.INC"
            CLRF      PORTB
            BSF      STATUS, 5
            CLRF      TRISB
            MOVLW    h'FF'
            MOVWF    TRISA
            BCF      STATUS, 5
AL          EQU      h'0C'      ;AL registeri adresi
AH          EQU      h'0D'
BL          EQU      h'0E'
BH          EQU      h'0F'
BASLA
            MOVLW    h'04'
            MOVWF    AL
            MOVLW    h'00'
            MOVWF    AH
            MOVLW    h'01'
            MOVWF    BL
            MOVLW    h'00'
            MOVWF    BH
CIKAR
            MOVF     BL, W      ;BL→W
            SUBWF   AL, F      ;AL=AL-W(BL) alt byte sonucu
            BTFSS   STATUS, 0   ;CF=0?
            DECF    AH, F      ;evet, AH=AH-1
            MOVF    BH, W      ;hayır, BH→W
            SUBWF   AH, F      ;BH=BH+W(AH) üst byte toplamı
ALT_BYTE_GOSTER
            MOVF     AL, W
            MOVWF   PORTB      ;alt byte sonucunu göster
TEST_A1
            BTFSC   PORTA, 1   ;A1 butonuna basılı mı
            GOTO    TEST_A1    ;hayır, tekrar test et
            MOVF    AH, W
            MOVWF   PORTB      ;üst byte sonucunu göster
DONGU
            GOTO    DONGU
END
```

#7 segment display üzerinde 5 sayısını gösteren program

Program

```
LIST        P=16F84
INCLUDE     "P16F84.INC"
            CLRF      PORTB      ;PORTB sil
            BSF      STATUS, 5   ;BANK1 geç
            CLRF      TRISB      ;PORTB çıkış
            BCF      STATUS, 5   ;BANK0 geç
BASLA
            MOVLW    h'05'      ;sayıyı registre yükle
            CALL     CEV_TAB     ;alt prog geç
            MOVWF    PORTB      ;içeriği PORTB ye aktar
```

```

DONGU
    GOTO      DONGU
CEV_TAB
    ADDWF    PLC, F ;PC alt 8 biti içerisine W yi yükle ve 5. satırdaki RETLW komutuna dallan
    RETLW
    RETLW    h'3F'
    RETLW    h'06'
    RETLW    h'5B'
    RETLW    h'4F'
    RETLW    h'66'
    RETLW    h'6D' ;sayıyı registere yükle
    RETLW    h'7D'
    RETLW    h'07'
    RETLW    h'7F'
    RETLW    h'6F'
    RETLW    h'77'
    RETLW    h'7C'
    RETLW    h'39'
    RETLW    h'5E'
    RETLW    h'79'
    RETLW    h'71'
    RETLW    h'80'
    END

```

#0-F arasında saydıran program

Program

```

    LIST      P=16F84
    INCLUDE   "P16F84.INC"
SAYAC1     EQU    h'0C'
SAYAC2     EQU    h'0D'
SAYAC      EQU    h'0E'
           CLRF   PORTB
           BSF    STATUS, 5
           CLRF   TRISB
           BCF    STATUS, 5
BASLA
           MOVLW  h'00' ;b'00001111' sayısını W ye yükle
           MOVWF  SAYAC ;W→SAYAC
DONGU
           MOVF   SAYAC, W
           ANDLW  B'00001111' ;W nin üst 4 bitini sıfırla
           CALL   CEV7SEG
           MOVWF  PORTB
           INCF   SAYAC, F ;SAYAC+1→W
           CALL   GECIKME
           GOTO   DONGU
CE7SEG
    ADDWF    PLC, F ;W(SAYAC)→PCL
    RETLW    h'3F'
    RETLW    h'06'
    RETLW    h'5B'
    RETLW    h'4F'
    RETLW    h'66'
    RETLW    h'6D'
    RETLW    h'7D'
    RETLW    h'07'
    RETLW    h'7F'
    RETLW    h'6F'
    RETLW    h'77'
    RETLW    h'7C'

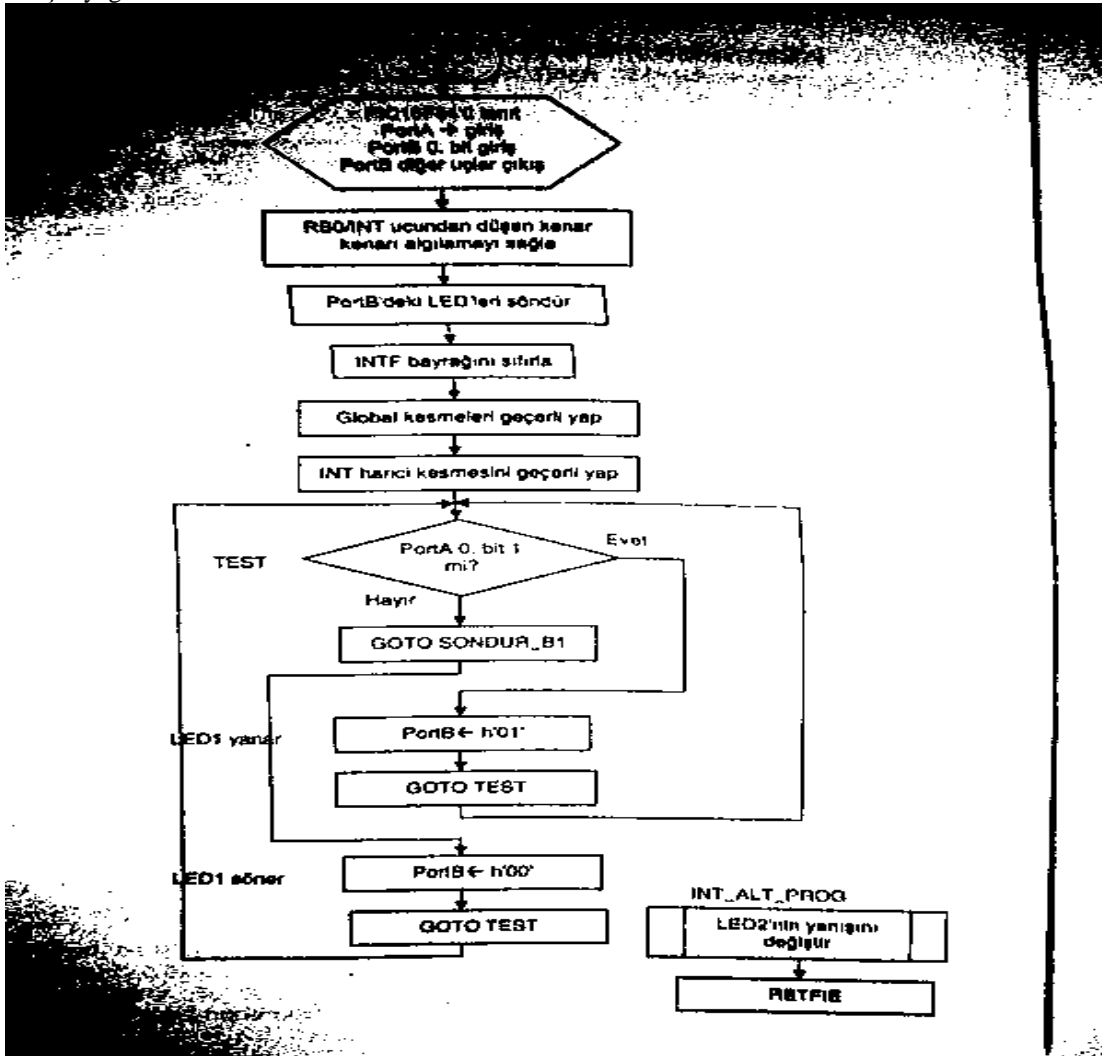
```

```

RETLW    h'39'
RETLW    h'5E'
RETLW    h'79'
RETLW    h'71'
RETLW    h'80'
GECIKME
MOVLW    h'FF'
MOVWF    SAYAC1
DONGU1
MOVLW    h'FF'
MOVWF    SAYAC2
DONGU2
DECFSZ   SAYAC2, F
GOTO     DONGU2
DECFSZ   SAYAC1, F
GOTO     DONGU1
RETURN
END

```

#RB0/INT ucundan girilen sinyal ile kesme oluşturulmasına örnektir. PORTA'nın 1. bitine bağlı butonun basılı olup olmadığını gösteren program. Çıktılar RB1 ve RB2 çıkışlarındandır. A1 butonuna basıldığı anda RB1 çıkışında oluşan 1 programın devamlı olarak çalıştığını gösterir. RB0/INT ucundan bir sinyal girerek kesme oluşturmak için bu girişe bir buton bağlanmıştır. Butona basıldığında kesme oluşur ve kesme alt programı çalışarak RB2 çıkışında 1 oluşturur. Bu da ana program çalışırken harici bir kesme meydana geldiğinde programın buna tepki verdiğini gösterir. (girişlerin her ikisinde normalde 1) Akış diyagramı



```

Program
LIST          P=16F84
INCLUDE       "P16F84.INC"
ORG          h'000'
GOTO         BASLA          ;ana program başlangıcı
ORG          h'004'
GOTO         INT_ALT_PROG  ;kesme alt programı başlangıcı
BASLA
MOVLW        h'FF'          ;sayıyı W ye yükle
MOVWF        TRISA         ;PORTA tüm uçlar giriş
MOVLW        h'00000001'    ;h'01'→W
MOVWF        TRISB         ;PORTB 0.bit giriş
MOVLW        h'10111111'    ;b'10111111'→W düşen kenar
MOVWF        OPTION_REG    ;W→OPTION
CLRF         PORTB         ;PORTB sil
BCF          INTCON, 1      ;INTF bayrağını sil kesme sinyaline hazırla
BSF          INTCON, 7      ;global kesme geçerli
BSF          INTCON, 4      ;RB0/INT kesmesi geçerli
TEST
BTFS        PORTA, 0        ;PORTA 1. biti test et
GOTO        LOGIC0_RB1
LOGIC1_RB1
BSF         PORTB, 1        ;PORTB 1. bit 1 yap
GOTO        TEST
LOGIC0_RB1
BCF         PORTB, 1        ;PORTB 1. biti 0 yap
GOTO        TEST
INT_ALT_PROG
BCF         INTCON, 1        ;INTF bayrağı sil
MOVLW        b'00000100'    ;terslenecek biti W ye yükle
XORWF        PORTB, F        ;RB2 yi tersle
RETFIE
END

```

Programdaki

```

MOVLW      b'10111111'
MOVWF      OPTION_REG

```

komutları ile RB0 ucundan girilen sinyalin düşen kenarında kesme oluşması sağlanmıştır.

OPTION register 6. bit INTEDG =0 düşen kenarda kesme

INTCON registerindeki bayrakların kurulması için

```

BCF      INTCON, 1
BSF      INTCON, 7
BSF      INTCON, 4

```

INTCON register GIE (7. bit) tüm kesme işlemleri aktif bayrağı, INTE (4. bit) harici kesmeyi aktif yapma bayrağı, INTF (1. bit) harici kesme bayrağı

Kesme olayının meydana geldiğini gösteren ve PORTB nin 2. bitinin durumunu değiştirmek için aşağıdaki komutlar kullanılmıştır:

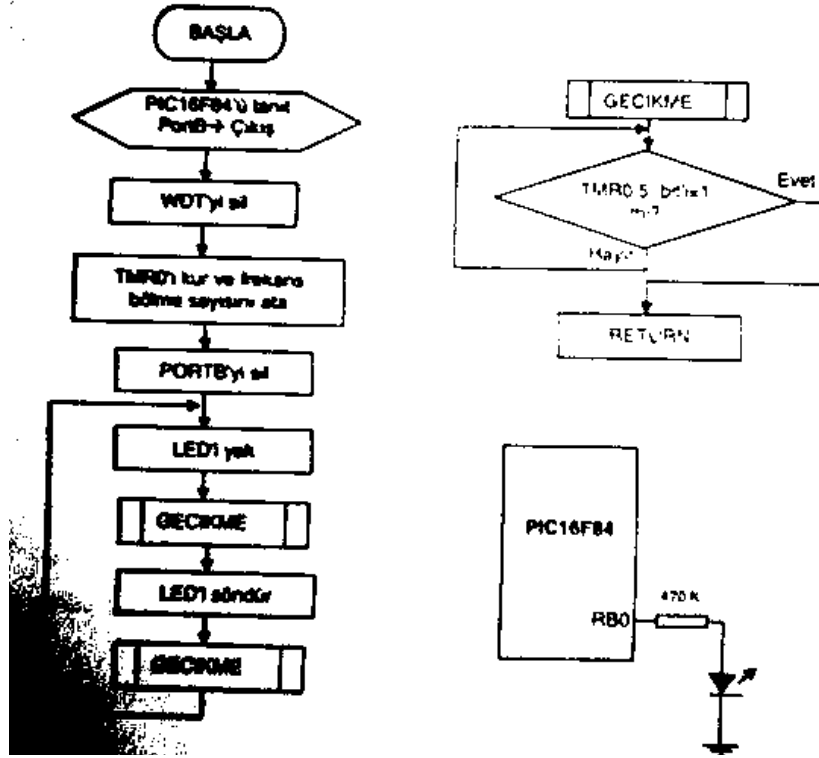
```

MOVLW      b'00000100'
XORWF      PORTB, F

```

	00000100	W register
RB2 1 ise	00000100	PORTB
XOR	00000000	RB2 0 oldu
RB2 0 ise	00000100	W register
	00000000	PORTB
XOR	00000100	RB2 1 oldu

#TMR0 sayıcısı ile PORTB nin 0. bitinde kare dalga üreten program
Akış diyagramı



Program

```

LIST          P=16F84
INCLUDE      "P16F84.INC"
CLRF        PORTB          ;PORTB sil
BSF         STATUS, 5      ;BANK1 geç
CLRF        TRISB         ;PORTB çıkış
BCF         STATUS, 5      ;BANK0 geç

BASLA
CLRWDI      ;frekans bölme işlemine hazırla
BSF         STATUS, 5      ;BANK1
MOVLW      b'11010111'    ;TMR0 ı frekans bölme değeri ve sinyal kaynağını seç
MOVWF      OPTION_REG     ;OPTION registere yaz
CLRF        PORTB

YAK
BSF         PORTB, 0       ;LEDi yak
CALL       GECIKME

SONDUR
BCF         PORTB, 0       ;LED söndür
CALL       GECIKME
GOTO      YAK

GECIKME
CLRF        TMR0          ;TMR0 ı h'00' a kur

TEST_BIT
BTFSS      TMR0, 5        ;TMR0 5. bit 1 mi?
GOTO      TEST_BIT       ;hayır 5. bit tekrar test et
RETURN

END
    
```

Programda OPTION registere yüklenen verinin anlamı
11010111

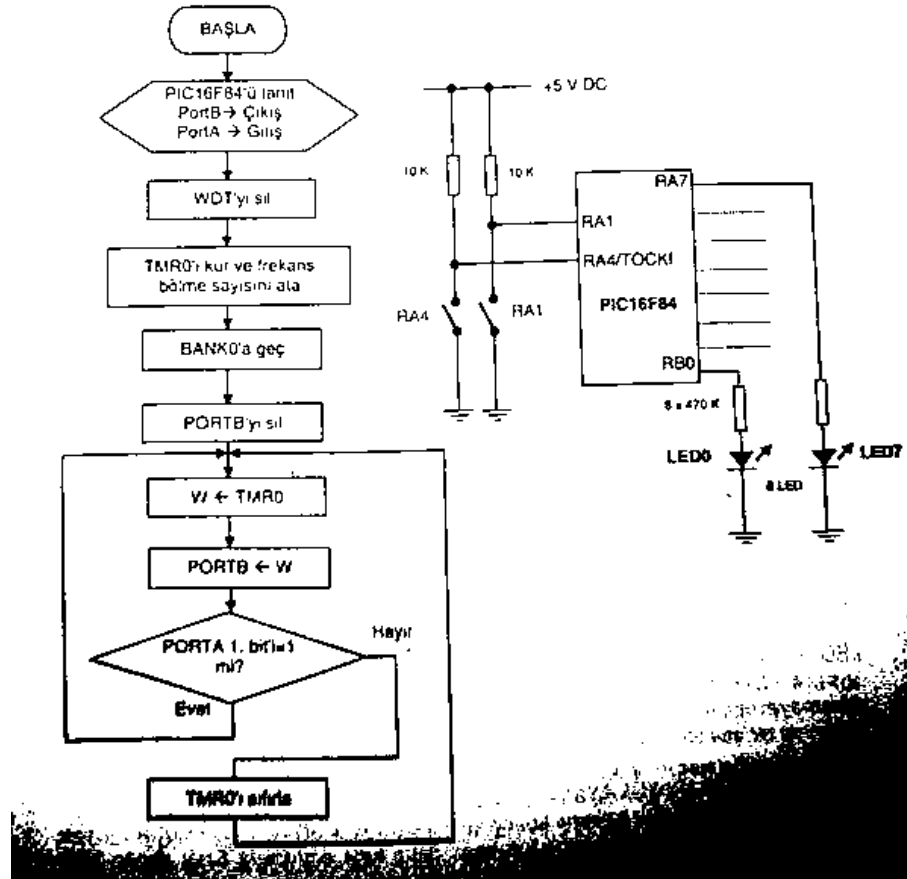
baştan başlayarak özetlersek

- PORTB pull-up lar geçersiz (kullanılmıyor)
- Harici kesme yükselen kenar (kullanılmıyor)
- TMR0 sinyal kaynağı dahili komut sayıklı

- TMR0 sayı artması harici sinyalin (RA4/TOCKI) düşen kenarında (kullanılmıyor)
- Frekans bölme değeri atanır
- Son üçlü frekans bölme oranı

#TMR0 sinyal kaynağı olarak harici giriş (RA4/TOCKI) kullanılması. PORTA nın 3. bitine bağlı butona (RA4) basıldığında PORTB de binary olarak artan sayıları gösterir. RA1 e basıldığında TMR0 registerini sıfırlar ve saymaya sıfırdan tekrar başlar.

Akış diyagramı



Program

```

LIST          P=16F84
INCLUDE      "P16F84.INC"
CLRF        PORTB          ;PORTB sil
BSF         STATUS, 5     ;BANK1 geç
CLRF        TRISB         ;PORTB çıkış
MOVLW      h'FF'
MOVWF      TRISA          ;PORTB uçları giriş
BCF         STATUS, 5     ;BANK0

BASLA
CLRF        PORTB          ;PORTB sil
CLRF        TMR0          ;TMR0 ve frekans bölme değeri sil
CLRWDI     ;WDT sil
BSF         STATUS, 5     ;BANK1
MOVLW      b'00101111'   ;frekans bölme değeri ata
MOVWF     OPTION_REG     ;OPTION registre yaz
BCF         STATUS, 5     ;BANK0
CLRF        PORTB

DONGU
MOVF       TMR0, W
MOVWF     PORTB
BTSS      PORTA, 1       ;PORTA 1. bit 0 mı?

```

```
CLRF      TMR0      ;evet TMR0 sıfırla
GOTO     DONGU
END
```

#TMR0 sayıcı kesmesi. PORTB nin binary olarak artan sayıları göstermesini sağlar. Sayma aralıklarındaki duruşlar için TMR0 zamanlayıcısı kullanılmıştır.

Akış diyagramı

Ana program

- başla
- tanıt
- h'04'→ORG
- GOTO LED_YAK
- PORTB çıkış
- b'00000111'→OPTION_REG
- h'00'→TMR0
- b'10100000'→INTCON
- PORTB sil

Alt program

- LED_YAK
- T0IF bayrağı sil
- PORTB+1→PORTB
- h'00'→TMR0
- RETFIE

Program

```
LIST      P=16F84
INCLUDE   "P16F84.INC"
ORG       h'00'
GOTO     BASLA
ORG       h'04'
GOTO     LED_YAK      ;kesme alt programına git
BASLA
BSF      STATUS, 5      ;BANK1
CLRF     TRISB          ;PORTB çıkış
MOVLW   b'00000111'    ;sayıyı W ye yükle
MOVWF   OPTION_REG     ;W→OPTION_REG
BCF     STATUS, 5      ;BANK0
MOVLW   h'00'
MOVWF   TMR0
MOVLW   b'10100000'    ;1→GIE, 1→T0IE, 0→T0IF
MOVWF   INTCON
CLRF    PORTB
DONGU
GOTO    DONGU
LED_YAK
BCF     INTCON, T0IF    ;0→T0IF
INCF   PORTB, F        ;PORTB+1→PORTB
MOVLW   h'00'
MOVWF   TMR0
RETFIE
END
```

#PORTB üzerinde binary artan sayıları gösteren program. WDT ye atanan değere göre sayma devam ederken PORTB içerisindeki veri h'FF' e ulaşmadan WDT zaman aşım sinyali nedeniyle PORTB yi tekrar başlatır.

Program

```
LIST          P=16F84
INCLUDE      "P16F84.INC"
SAYAC1      EQU    h'0C'
SAYAC2      EQU    h'0D'
BASLA

BSF         STATUS, 5           ;BANK1
MOVLW      b'00000111'        ;WDT seç
MOVWF      OPTION_REG         ;W→OPTION_REG
CLRF       TRISB
BCF        STATUS, 5           ;BANK0

SONDUR

CLRF       PORTB

YAK

CALL       GECIKME
INCF      PORTB, 1             ;PORTB+1→PORTB
GOTO      YAK

GECIKME

MOVLW     h'4F'
MOVWF     SAYAC1

DONGU1

CLRF      SAYAC2

DONGU2

DECFSZ    SAYAC2, 1
GOTO     DONGU2
DECFSZ    SAYAC1, 1
GOTO     DONGU1
RETURN
END
```

#PORTB nın 0. bitine bağlı voltmetrede 2.5V luk gerilim üreten program.

Dijital sinyal çıkış geriliminin %50 si olacağından

IS değişkeni→h'80'

BEK değişkeni→h'80' olarak atanır

Akış diyagramı

Ana program

- başla
- tanımlama
- PORTB çıkış
- IS süresi
- BEK süresi
- PORTB 0. bit 1 yap
- IS→W
- CALL GECIKME
- PORTB 0. bit 0 yap
- BEK→W
- CALL GECIKME buradan PORTB 0. bitinin 1 yapa gider

Alt program

- CALL GECIKME
- W→SAYAC
- SAYAC-1→SAYAC
- SAYAC 0 mı? Hayır→bir üstteki komut evet→ana programa dön
- RETURN

Program	LIST	P=16F84	
	INCLUDE	"P16F84.INC"	
IS	EQU	h'0C'	
BEK	EQU	h'0D'	
SAYAC	EQU	h'0E'	
	ORG	h'00'	
BASLA			
	BSF	STATUS, 5	
	CLRF	TRISB	
	BCF	STATUS, 5	
	CLRF	PORTB	;CS ucu 0
	MOVLW	h'80'	
	MOVWF	IS	
	MOVLW	h'80'	
	MOVWF	BEK	
TEKRAR			
	BSF	PORTB, 0	;PORTB 0. bit 1
	MOVF	IS, W	;IS→W iş süresini al
	CALL	GECIKME	
	BCF	PORTB, 0	;PORTB 0. bit 0
	MOVF	BEK, W	;BEK→W bekleme süresini al
	CALL	GECIKME	
	GOTO	TEKRAR	
GECIKME			
	MOVWF	SAYAC	;IS yada BEK içerisindekini SAYAC a yaz
DONGU			
	DECFSZ	SAYAC, F	
	GOTO	DONGU	
	RETURN		
	END		

Tegin Yücel MAYADAĞLI

File register haritası

Bank 0	Bank 1
0x00 INDF	0x80 INDF
0x01 TNF0	0x81 OPTION
0x02 PCL	0x82 PCL
0x03 STATUS	0x83 STATUS
0x04 FSR	0x84 FSR
0x05 PORT A	0x85 TRISA
0x06 PORT B	0x86 TRISB
0x07	0x87

EEPROM bellek alanı (kullanılmayacak)

0x0A PCLATH	0x8A PCLATH
0x0B INTCON	0x8B INTCON
0x0C	0x8C

genel amaçlı registerlerin kullanılacağı alan (veri depolama alanı olarak kullanılır)

0x4F	0xCF
------	------

Bank değiştirme

Bir banktan diğerine geçmek için STATUS register denilen özel registerin 5. ve 6. bitinin durumunu değiştirmek gerekir.

00→bank 0

01→bank 1

16F84 ün sadece 2 bankı bulunduğundan sadece 5. bitin değerini değiştirmek yeterlidir. 6. bit daima 0 olmalıdır. Her iki bit pic enerjilendiğinde 0 dir. Ayrıca reset girişlei ile bu bitler 0 yapılır. Pic enerjilendiğinde doğrudan bank 0 seçilmiş olur.

STATUS registerinin 5. bitini 1 yapmak için BSF, 0 yapmak için BCF kullanılır.

BSF h'03', 5 → bank 1 seçilir

BCF h'03', 5 → bank 0 seçilir

Portların I/O olarak yönlendirilmesi

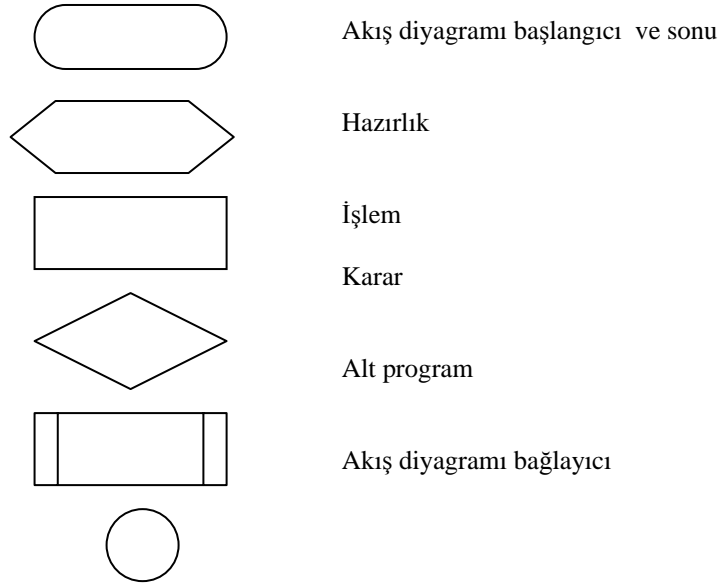
PortA y1 → TRISA registeri

PortB y1 → TRISB registeri yönlendirir.

PortA/B nin hangi biti giriş yapılma isteniyorsa TRISA/B içerisine o bite karşılık gelen bit 1 yapılır, çıkış olarak yönlendirilmek istenen bit içinse 0 yazılır.

TRISA/B → 1/0 → PortA/B → giriş/çıkış

Akış diyagramları



Konfigürasyon bitlerinin yazılması

Aşağıdaki koşulları belirlemede kullanılır

- osilatör tipi belirlemek
- watchdog timer on-off yapmak
- power-on reset on-off yapmak
- kod korumayı on-off yapmak (yazılan programın okunmasını engellemek)

yazılacak tanım kodları

- kod koruma var → `_CP_ON`
- kod koruma yok → `_CP_OFF`
- power-on reset var → `_PWRTE_ON`
- watchdog timer devrede → `_WDT_ON`
- low power osc → `_LP_OSC`
- xtal osc → `_XT_OSC`
- high speed osc → `_HS_OSC`
- RC osc → `_RC_OSC`

Buna göre program şöyle yazılmaya başlanır

```
LIST          P=16F84
INCLUDE      "P16F84.INC"
_CONFIG      _CP_OFF & _WDT_OFF & _PWRTE_OFF
işlem...
```

VERİ TRANSFERİ VE KARAR İŞLEMLERİ

W registeri kullanımı

Veri transferi

W register RAM içerisindeki file registerden bağımsızdır. Registerler arasında veri transferi yapmak için kullanılır.

PORTA içeriğini PORTB ye yazmak

```
MOVF        PORTA, W      ; PORTA içeriğini W registerine taşı
MOVWF       PORTB        ; W registeri içeriğini PORTB ye gönder
```

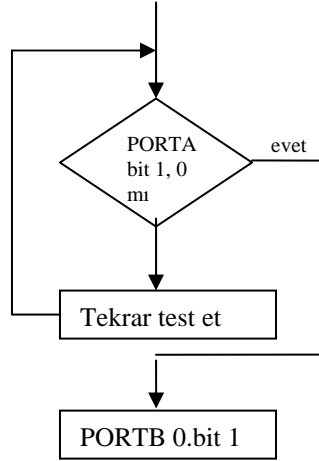
PORTB nin ilk 4 bitini 1 yapmak

```
MOVLW      H'0F'         ; W registerine H'0F' yükle ( 0F ilk 4 hanenin binary karşılığı )
MOVWF      PORTB        ; W registeri içeriğini PORTB ye gönder
```

Bit test ederek karar vermek

BTFSC veya BTFSS komutları ile register içerisindeki herhangi bir bit test edilebilir.
BTFSC → register deki biti test et eğer 0 ise bir sonraki komuta atla.
BTFSS → registerdeki biti test et eğer 1 ise bir sonraki komuta atla.

BTFSC (register), (test edilecek bit 0-7)



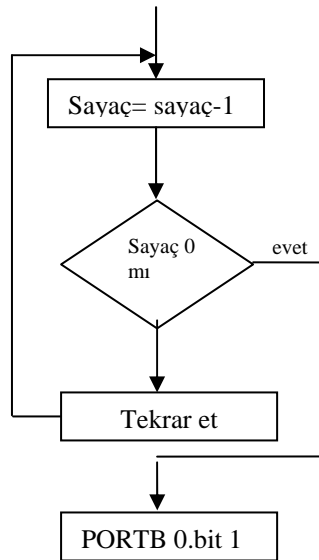
```
TEST_PORTA
BTFSC PORTA, 1
GOTO TEST_PORTA
BSF PORTB, 0
```

DÖNGÜ DÜZENLEMEK

Sayaç kullanarak döngü düzenlemek

Bazı işlemlerin önceden belirlenen sayıda tekrarlanması durumunda register sayaç olarak kullanılır. Her işlem tekrarında sayaç bir azalır.

DECFSZ (sayaç) (0veyaW-1veyaF→W ise sayaçtan 1 çıkarılır ve sonuç W ye yazılır- F ise sayaçtan 1 çıkarılır ve sonuç sayaç içerisine yazılır)



```
TEKRAR
DECFSZ SAYAC, F
GOTO TEKRAR
BSF PORTB, 0
```

Karşılaştırma yaparak döngü düzenlemek

Bazı işlemler önceden belirlenen sayıda tekrarlanması gerekir. Bu durumda bir register sayaç olarak kullanılır. Her bir işlem tekrarında sayaç bir artırılır. Sayaç sayısı işlemin tekrar sayısına ulaştığında döngü sona erer ve başka komuta geçilir.

TEKRAR	CLRF	SAYAC	;sayaç temizle
	INCF	SAYAC, F	;sayacı 1 artır
	MOVLW	h'07'	;W registerine h'07' yükle
	SUBWF	SAYAC, W	;sayaçtan W yi çıkar
	BTFSC	STATUS, 2	;STATUS 2. biti 1mi?
	GOTO	TEKRAR	
DONGU	GOTO	DONGU	
	END		

Döngünün tekrar sayısı W registeri içerirse yüklenen sayı ile belirlenir. SAYAC değişkeni içerisindeki sayı döngünün her tekrarında bir artırılır ve SUBWF komutu ile SAYAC tan W nin içeriği çıkarılır. Çıkarma işlemi neticesinde STATUS registerinin 0. ve 2. bitindeki değer etkilenir. Bu bitler BTFSC komutuyla test edilir. Test sonucunda istenen değere ulaşıncaya döngüye son verilir. STATUS registeri 0. bit CARRY FLAG 2. bit ZERO FLAG olarak adlandırılır.

SUBWF komutu

Çıkarma yapan bir komuttur. W registerinin içeriğini file registerden çıkarır ve sonucu W registerine yazar. SUBWF (file register), W yada F olabilir. W çıkarma sonucunu W registerine yazdırır F ise file registre.

Z ve C flag içeriği çıkarma sonucu şöyle etkilenir.

File register > W Z=0 C=1

File register = W Z=1 C=1

File register < W Z=0 C=0

1) F=W ise

MOVLW	h'50'	
MOVWF	MEM	
SUBWF	MEM, W	
BTFSC	STATUS, 2	;W=MEM ise program GOTO ile devam eder, > veya < ise GOTO yu atlar
GOTO	XXX	
BSF	PORTB, 0	

2) F<W ise

MOVLW	h'40'	
MOVWF	MEM	
MOVLW	h'50'	
SUBWF	MEM, W	
BTFSC	STATUS, 0	;W=veya> ise program GOTO ile devam eder, < ise GOTO yu atlar
GOTO	XXX	
BSF	PORTB, 0	

3) F>W ise

MOVLW	h'60'	
MOVWF	MEM	
MOVLW	h'50'	
SUBWF	MEM, W	
BTFSC	STATUS, 0	;W=veya< ise program GOTO ile devam eder, > ise GOTO yu atlar
GOTO	XXX	
BSF	PORTB, 0	

SUBLW komutu

Çıkarma yapan komuttur. Sabit L içerisinde W registeri içeriğini çıkarır, sonucu W registerine yazar.

Çıkarma sonucunda W ve sabitin değerleri flagların değerini değiştirir.

File register>W Z=0 C=1
File register=W Z=1 C=1
File register<W Z=0 C=0

1) sabit=W ise

```
MOVLW    h'50'  
SUBLW    h'50'  
BTFSC    STATUS, 2    ;sabit=W ise BSF komutuna geçilir  
GOTO     XXX  
BSF      PORTB, 0
```

2) sabit<W

```
MOVLW    h'60'  
SUBLW    h'50'  
BTFSC    STATUS, 0    ;sabit<W ise BSF komutuna atlanır  
GOTO     XXX  
BSF      PORTB, 0
```

3) sabit>W

```
MOVLW    h'40'  
SUBLW    h'50'  
BTFSC    STATUS, 0    ;sabit>W ise BSF komutuna atlanır  
GOTO     XXX  
BSF      PORTB, 0
```

Status register

STATUS registerinin içeriği bir çok komutun çalışması neticesinde değişir. STATUS registerinin Z, DC veya C bitleri etkileyen bir komut kullanıldığında bu bitlere yazma işlemi yapılamaz. Bu bitlerin 1 veya 0 olan durumu kullanılan PIC e bağlıdır. Bank seçmek için kullanılan RP0 ve RP1 bitlerine yazmak mümkün olduğu halde, TO ve PD bitlerine yazma işlemi yasaklanmıştır. Bu nedenle bir STATUS registerini etkileyen komutun çalıştırılması sonucunda etkilenen bitler beklenenden farklı olabilir.

STATUS register bitleri:

Bit 7: **IRP**:register bank select bit

16F84 te kullanılmaz 0 olarak kalmalıdır

Bit 6-5: **RP1:RP0**: register bank select bit

00=bank0 (h'00'-h'FF')

01=bank1 (h'80'-h'FF')

10=bank2 (h'100'-h'17F')

11=bank3 (h'180'-h'1FF')

16F84 te sadece RP0 kullanılır, RP1 0 olarak kalmalıdır

Bit 4: **TO**: time out bit

1=PIC enerjilendiğinde, CLRWDT ve SLEEP komutu çalışınca

0=WDT zamanlayıcısında zaman dolduğunda

Bit 3: **PD**: power down bit

1=PIC enerjilendiğinde ve CLRWDT komutu çalıştığında

0=SLEEP komutu çalışınca

Bit 2: **Z**: zero bit

1=aritmetik veya mantıksal komutun sonucu 0 olduğunda

0= " " " " " " 0 olmadığına

Bit 1: **DC**: digit carry/borrow bit

ADDLW ve ADDWF komutları kullanıldığında oluşan taşma ve ödünç alma olduğunda

1=alt dört bitin 4. bitinde taşma meydana geldiğinde

0= " " " " " " gelmediğinde

Bit 0: **C**: carry/borrow bit

ADDLW ve ADDWF komutları kullanıldığında oluşan taşma ve ödünç alma olduğunda

1=en soldaki 7. bitte taşma olduğunda

0=en soldaki 7. bitte taşma olmadığına

NOT: RLF ve RFF komutları çalıştığında en sol bit veya en sağ bit değeri carry bitine yüklenir.

ZAMAN GECİKTİRME VE ALT PROGRAMLAR

Zaman geciktirme döngüleri

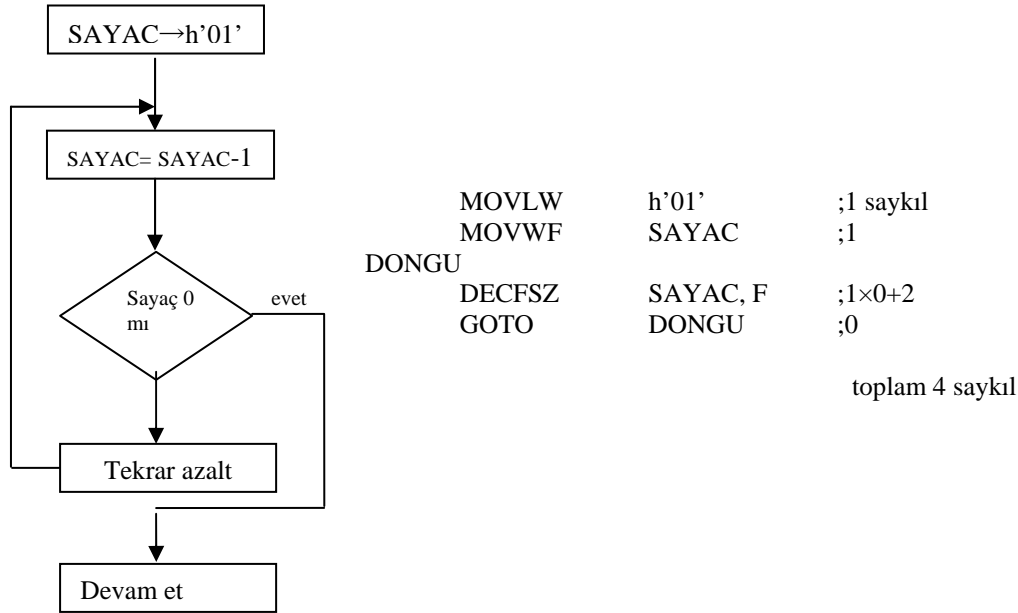
Dahili komut saykılı

PIC girişine uygulanan osc frekansı 4 e bölünür ve bir komutun icra süresi bu frekansın bir saykılı süresindedir. Fakat PIC16F84 komutlarından 10 tanesi hariç diğerleri bir komut saykılı süresince çalışır. Bu komutlar:

Komut	Komut saykılı
GOTO	2
CALL	2
RETURN	2
Program sayıcıya (PC) veri yazan komutlar	2
DECFSZ	1 (register içerisindeki sayı 1 ise) 2 (register içerisindeki sayı 0 ise)
RETLW	2
RETFIE	2
INCFSSZ	1 (register içerisindeki sayı 1 ise) 2 (register içerisindeki sayı 0 ise)
BTFSC	1 (test edilen bit 1 ise) 2 (test edilen bit 0 ise)
BTFSS	1 (test edilen bit 0 ise) 2 (test edilen bit 1 ise)

Tek döngü ile min zaman geciktirme

SAYAC registeri içerisine h'01' yüklenir, DECFSZ komutu ile tekrar sayısından her defasında 1 çıkarılır. Çıkarma sonucu 0 olduğunda döngü sona erdirilir.



Tek döngü ile max zaman geciktirme

SAYAC registerine yüklenen sayı h'FF' olursa toplam 766 saykılılık zaman gecikmesi elde edilir.

Komut saykıl sayısının bulunması

N registre atanan sayı olursa bir döngünü 3N saykılılık bir geciktirme oluşturur.

N sayısının bulunması

$$N = GS \times f / 12$$

N → registre atanan decimal sayı hex e dönüştürülerek yazılmalı

GS → gecikme süresi

f → PICe uygulanan harici frekans

Alt programlar

Gerektiğinde alt program adı CALL komutundan sonra yazılarak çağırılır. RETURN komutu ile ana programa dönülür. Alt programın çalıştırılmaya başladığı anda programın tekrar geri döneceği adres bilinmesi gerekir. STACK register bu adresin yazıldığı özel bir registerdir. Alt programlardan geri dönülürken en son yazılan adres ilk önce işlem görür.

- ana programın ilk komutundan itibaren program çalışmaya başlar.
- program akışı CALL_ALTPROG ile alt prog adlı alt programa geçer
- ana alt programdan çıkıldığı andaki adres STACK registere yazılır
- alt program komutları çalışır
- RETURN alt program komutlarının bittiğini gösterir. Program akışını ana programda kaldığı yerdeki adrese gönderir
- STACK registerde yazılı bulunan ayrılma adresine alınır
- Ana program kaldığı yerden itibaren çalışmaya başlar ve END komutu ile sona erer

BİT KAYDIRMA VE MANTIKSAL İŞLEM KOMUTLARI

Sola kaydırma

RLF komutu ile belirlenen bir file register içerisindeki bitlerin pozisyonu her defasında bir sola kaydırmak için kullanılır. Register içerisindeki bitler sola kaydıldığında MSB biti STATUS registerde bulunan carry flag içerisine yazılır. Carry flag içeriği ise registerlerin LSB bitine yazılır.

RLF (file register) (d) destination W veya F W ise sonuç W ye F ise F e yazılır

MEM adındaki bir file registerin içeriği h'86' ise RLF komutu çalıştırıldığında MEM registerinin içeriği

MOVLW h'86'
MOVWF MEM
RLF MEM, F

H'86' = b'1000110'

1CF	1MSB	0	0	0	0	1	1	0LSB
-----	------	---	---	---	---	---	---	------

Kaydırma sonucu MEM registeri

0CF	0MSB	0	0	0	1	1	0	1LSB
-----	------	---	---	---	---	---	---	------

Sağa kaydırma

RRF komutu ile belirlenen bir file register içerisindeki bitlerin pozisyonu her defasında bir sağa kaydırmak için kullanılır. Register içerisindeki bitler sağa kaydıldığında LSB biti STATUS registerde bulunan carry flag içerisine yazılır. Carry flag içeriği ise registerlerin MSB bitine yazılır.

RRF (file register) (d) destination W veya F W ise sonuç W ye F ise F e yazılır

MEM adındaki bir file registerin içeriği h'49' ise RRF komutu çalıştırıldığında MEM registerinin içeriği

MOVLW h'49'
MOVWF MEM
RRF MEM, F

H'49' = b'01001001'

0MSB	1	0	0	1	0	0	1	1CF
------	---	---	---	---	---	---	---	-----

Kaydırma sonucu MEM registeri

1	0	1	0	0	1	0	0	0
---	---	---	---	---	---	---	---	---

COMF ve SWAPF komutları

COMF komutu seçilen file register içerisindeki bitleri tersine çevirir yani 1 ler 0, 0 lar 1 olur

COMF (file register) (d) W veya F

MOVLW h'0F'
MOVWF MEM
COMF MEM, F

SWAPF komutu seçilen bir file register içerisindeki verinin ilk dört biti ile son dört biti yer değiştirir.

SWAPF (file register) (d)

MOVLW h'C6'
MOVWF MEM
SWAPF MEM, F

Mantıksal işlem komutları

ANDLW

W register içeriğini istenen bir sabit veri ile AND ler elde edilen sonucu W registerine yazar. Bu komut W register içerisindeki bir veya birkaç bitin değerini 0 yapmak için kullanılır.

ANDLW (sabit)

1 ile AND lenen bitlerin değeri değişmez

0 ile AND lenen bitlerin değeri 0 olur.(maskeleye)

W register içerisindeki b'11010111' ise 2. ve 7. biti 0 yapmak için b'01111011' sabiti ile AND lenmek gerekir.

```
MOVLW    b'11010111'  
ANDLW    b'01111011'  
MOVWF    PORTB
```

DONGU

```
GOTO     DONGU
```

Sonuçta W register içeriği şöyledir b'01010011'

ANDWF

Seçilen bir file register içeriğini AND ler sonuç W veya F registeri içerisine yazılır.

ANDWF (file register) (d) W veya F olabilir

PORTA içeriği b'00101100' olsun W registeri içeriği ise b'11011111' verisini yerleştirerek 5. bitin içeriğini 0 yapan ve PORTB ye gönderen komutlar.

```
MOVLW    b'11011111'  
ANDWF    PORTA, W  
MOVWF    PORTB
```

DONGU

```
GOTO     DONGU
```

Sonuçta PORTA içeriği şöyledir PORTA'00001100'

IORLW

W registerinin içeriğini istenen bir sabit veri ile OR lar.

IORLW (sabit)

W register içerisindeki veri b'00000011' ise bu verinin 6. ve 7. bitindeki verileri 1 yapmak için b'11000000' sabiti ile OR lenmek gerekir.

```
MOVLW    b'00000011'  
IORLW    b'11000000'  
MOVWF    PORTB
```

Sonuçta PORTB içeriği şöyledir PORTB'11000011'

IORWF

Seçilen bir file register ile W registerinin içeriğini OR lar sonucu W registerine yada file registre yazar.

IORWF (file register) (d)

XORLW

W register içeriğini sabit bir veri ile XOR lar, elde edilen sonucu W register içerisine yazar.

XORLW (sabit)

```
MOVLW    b'11110000'  
XORLW    b'10000001'  
MOVWF    PORTB
```

Sonuçta PORTB içeriği şöyledir PORTB'01110001'

XORWF

Seçilen bir file register ile w registerinin içeriğini XORlar sonuç W veya F register içerisine yazılır.

XORWF (file register) (d)

Bir bytelık iki veriyi karşılaştırmak

W register içindeki bir bytelık veriyi istenen bir bytelık sabit veri ile aynı olup olmadığını test etmek amacıyla XORLW komutu kullanılır. Bir file register içerisindeki veriyi W register içindeki veri ile karşılaştırmak içinde XORWF komutu kullanılır. Veriler aynı ise XORlama sonucunda Z flag 1, veriler aynı değil ise Z flag 0 olur. PORTA registeri içerisindeki verinin b'00000110' verisi ile aynı olup olmadığını kontrol eden program

```

        MOVLW      b'000001100'
TEST_PORTA
        XORWF     PORTA, F
        BTFSS    STATUS, 2
        GOTO     TEST_PORTA
DEVAM
        MOVF     PORTA
        MOVWF    PORTB
        .
        .
        .

```

Bir bytelık veriyi 0 ile karşılaştırmak

W register içerisindeki bir bytelık verinin 0 olup olmadığını test etmek için 0 sabit verisi ile ORlanır. Bu işlem için IORLW kullanılır.

Bir file register içerisindeki verinin 0 olup olmadığını kontrol etmek için W register içerisine 0 verisi atandıktan sonra bu ikisi ORlanır. Bu işlem için IORWF komutu kullanılır.

OR sunucunda karşılaştırılan byte lar aynı ise yani içerikleri 0 ise Z flag 1 olur

W register içeriğinin 0 olup olmadığını kontrol eden komutların yazılışı

```

        MOVLW      b'00000000'
TEST_W
        IORLW     b'00000000'
        BTFSS    STATUS, 2
        GOTO     TEST_W
DEVAM

```

ARİTMETİK İŞLEMLER

Aritmetik işlemler için gerekli komutlar:

```

ADDLW      sabiti W registerden çıkarır
ADDWF     W registerle F registeri toplar
SUBLW     sabitten W registeri çıkarır
SUBWF     F registerden W registeri çıkarır
RLF       F register içerisindeki bitleri bir sola kaydırır
RRF       F register içerisindeki bitleri bir sağa kaydırır

```

0-255 sayıları arasındaki 8 bitlik toplamada carry flag elde edilen sonucun 8 bitin dışına taşıp taşmadığını gösterir.

Eğer CF=0 ise sonuç 8 bitten fazla değildir

CF=1 ise sonuç 8 bitten fazladır yani taşma vardır

0-255 sayıları arasındaki 8 bitlik çıkarmada CF elde edilen sonucun negatif veya pozitif olduğunu gösterir eğer CF=1 ise sonuç pozitif

CF=0 ise sonuç negatiftir

8 bit toplama

Bir bytelik iki sayı iki şekilde toplanabilir. İlki W register sabit bir sayı ile toplanır (ADDLW) sonuç W registerine yazılır. İkincisi W registerle bir file register içeriği toplanır (ADDWF) sonuç W veya F e yazılır.

$h'02' + h'FD' = h'FF'$ CF=0

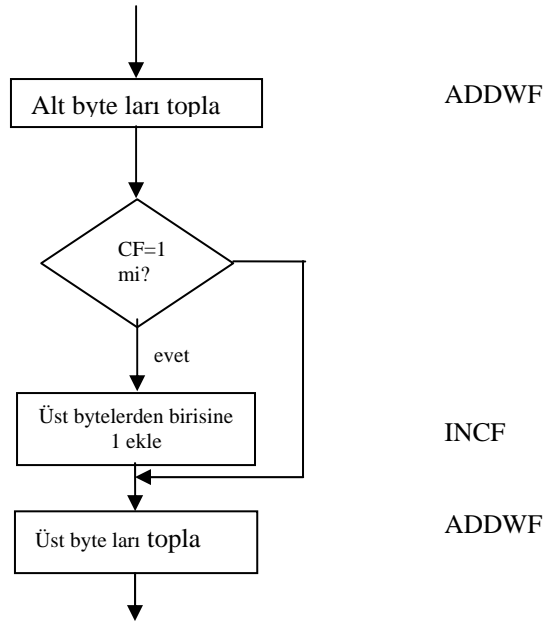
$h'03' + h'FD' = h'00'$ CF=1

16 bit toplama

Toplama işleminde kullanılan sayılar 1byte= $h'FF' = d'256'$ dan daha büyük ise sayı iki yada daha fazla byte ile gösterilir. $h'01FD'$ sayısını iki parçaya bölerek $h'01'$ için A registeri, $h'FD'$ için B registeri kullanarak işlemleri yaparız.

- toplanacak iki sayının ilk önce alt byte ları toplanır
- alt byte lerde taşma varsa üst bytelardan birisine 1 eklenir
- üst bytelar toplanır

işlemin akış diyagramı:



8 Bit çıkarma

SUBLW h'02' ;sayıdan W registerini çıkar

SUBWF MEM, F;MEM den W registerini çıkar

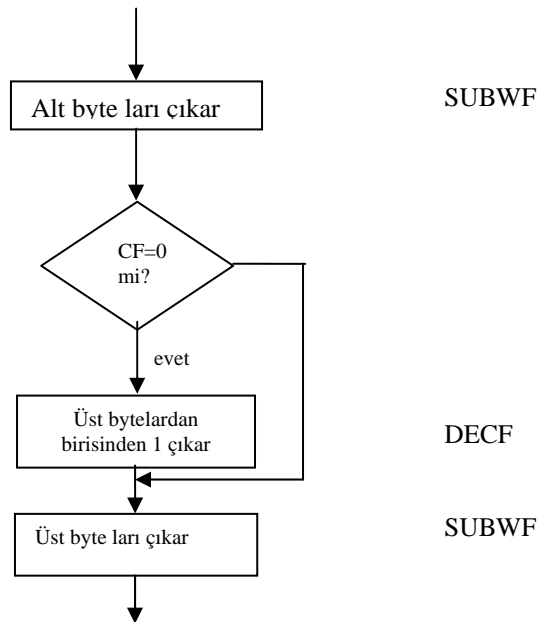
Küçük sayıdan büyük sayı çıkarılırsa CF=0, büyük sayıdan küçük sayı çıkarılırsa veya sayılar birbirlerine eşit ise CF=1 olur.

16 Bit çıkarma

- alt byteler birbirlerinden çıkarılır

- ödünç alma var ise CF=0 sa üst bytelardan birisinden 1 çıkarılır

- eğer ödünç alma yok ise CF=1 se üst byteların çıkarması yapılır



ÇEVİRİM TABLOLARI

Çevrilecek kod	Çevrilen 7seg k. PORTB	7 seg uçlarındaki veri	7 segde görülecek sayı
h'00'	h'3F'	00111111	0
h'01'	h'06'	00000110	1
h'02'	h'5B'	01011011	2
h'03'	h'4F'	01001111	3
h'04'	h'66'	01100110	4
h'05'	h'6D'	01101101	5
h'06'	h'7D'	01111101	6
h'07'	h'07'	00000111	7
h'08'	h'7F'	01111111	8
h'09'	h'6F'	01101111	9
h'0A'	h'77'	01110111	A
h'0B'	h'7C'	01111100	b
h'0C'	h'39'	00111001	C
h'0D'	h'5E'	01011110	d
h'0E'	h'79'	01111001	E
h'0F'	h'71'	01110001	F
Nokta	h'80'	10000000	.

Program counter

13 bitlik bir program sayıcısı vardır. GOTO ve CALL komutlarıyla kullanılan 11 bitlik adresler 2KB lik program belleği bulunan PIC leri adreslemek için yeterlidir. (PIC16F84 te 1KBlik program belleği vardır) PIC programlarında program sayıcı PC kodu ile kullanılır. Program sayıcının alt 8 bitine PCL üst 5 bitine PCH adı verilir. Program sayıcının üst 5 bitini doğrudan okumak ve yazmak mümkün değildir. Ancak PCLATH özel registeri ile veri yüklenir. Bu registerde 5 bittir.

PCLATH içeriği PIC enerjilendiğinde b'00000' dır. PCH komutu kullanılmadığı sürece bu registerle herhangi bir işlemiz yoktur. Fakat çalışmada soruna neden olmaması için 4. ve 3. bitlerinin içerikleri 0 lanmalıdır.

RETLW

Ana programa dönüş için kullanılır. RETURN komutundan farklı olarak W registerie sabit bir sayı yükler.

RETLW h'3F' ;W ye verilen sayıyı yükle ve ana programa dön

KESMELER (INTERRUPTS)

Port girişlerinden veya donanım içerisindeki bir sayıcıdan gelen sinyal nedeniyle belleğinde çalışmakta olan programın kesilmesi olayıdır. Programın kesildiği andan itibaren önceden hazırlanan bir alt program çalışır. Alt program işlevini bitirdikten sonra ana program kaldığı yerden itibaren tekrar çalışmasına devam eder. Neticede kesme ana programın çalışmasını duraklatır fakat işlevini devam ettirmesini engellemez. Alt programlarla arasındaki fark normal alt program çağırma, program içerisinde yazılan komutlar vasıtasıyla yapılır. Kesme alt programlarının çağırılması ise donanımda oluşan değişiklikler yapar. Kesme meydana geldiğinde o anda çalışmakta olan komutun çalışması tamamlanır daha sonra program akışı PIC program belleğinin h'0004' adresine atlar ve bu adresteki komutu çalıştırır. PIC kesme alt programı çalıştıktan sonra ana programda hangi adrese geri döneceğini o anda çalışan komutun adresini STACK registerine kaydeder. Alt program işlevini tamamladığında ana program akışı bu adresten itibaren devam eder. Kesme alt programından ana programa dönüş komutu RETFIE komutu ile yapılır.

Kesme alt programında meydana gelen olaylar:

- kesme olayı meydana geldiğinde STACK registerinin olduğu adrese h'23F' atla
- ana programın kaldığı adresi STACK registere yaz
- h'04' adresindeki komutu çalıştır. Bu komut kesme alt programını çalıştır
- kesme alt programının olduğu adrese atla
- kesme alt programı çalıştır. Alt programın en son komutu RETFIE dir
- STACK registerin olduğu yere git
- Ana programa dönüş adresini al
- Ana programın kesildiği yerdeki adresten bir sonraki adrese git ve devam et

INTCON registeri

RAM bellekte h'0B' adresinde bulunan özel registerlerden bir tanesidir. Tüm kesme işlemlerinin kontrolü bu register aracılığı ile yapılır.

PORTB deki deęişiklięi algılamak için bu porttaki son deęer RB4-RB7 uçlarından okunan veri ile karşılaştırılır. Eski ve yeni okunan veriler OR lanır. Farklılık varsa RBIF bayraęı (INTCON registerinin 0. biti) 1 olur. PORTB kesmesi şöyle silinebilir:

- RBIE biti (INTCON 3.bit) silinmek suretiyle
- PORTB yi okuduktan sonra RBIF bitini silmek suretiyle

Kesme alt programlarının düzenlenmesi

Tüm kesme işlemlerinin aktif yapma bayraęı (GIE)

Bir kesme olayının meydana gelmesi esnasında INTCON registerinin 7. biti 0 olur. Bu işlem yeni bir kesmenin program akışını bozmaması için otomatik olarak yapılır. Kesme alt programı çalışmasını RETFIE komutu ile sona erdirip, ana programa döndüğü anda sonraki kesmelerin geçerli olabilmesi için tekrara otomatik olarak 1 yapılır.

Kesme olayında meydana gelen olaylar:

- INTCON GIE=1
- Kesme olayı
- INTCON GIE=0
- PC içerisindeki adres STACK registerine kaydedilir
- h'004' adresine atlar
- kesme olayı başlar alt program çalışır
- kesme bayraęı kontrol edilir
- kesme olayı biter (RETFIE)
- program akışı ana programa geçer
- INTCON GIE=1

Kesme esnasında W ve STATUS registeri saklamak

Eđer W ve STATUS registerinin içerięi korunmak isteniyorsa gerekli komutlar kesme alt programının içerisinde yer almalıdır. Kesme olduğunda yapılması gereken işlemler:

- W registeri deęişkene yükle
- STATUS registeri deęişkene yükle
- Kesme işlemi gerçekleştir
- STATUS geri yükle
- W geri yükle
- Kesme alt programından dön (RETFIE)

Bu işlemleri yapacak komutlar:

```
ORG          h'004'
GOTO         KESME_ALT_PROG
.
.
KESME_ALT_PROG
1:  MOVWF   SAKLA_W           ;W registerini yükle
2:  SWAPF   STATUS, W         ;STATUS içerięini SWAP yap
3:  MOVWF   SAKLA_S           ;STATUS içerięini sakla
.
.
;kesme alt programı komutları
.
.
4:  SWAPF   SAKLA_S, W        ;STATUS içerięini yeniden SWAP yap
5:  MOVWF   STATUS            ;STATUS registere yeniden yükle
6:  SWAPF   SAKLA_W, F        ;W içerięini SWAP yap
7:  SWAPF   SAKLA_W, W        ;W yi yeniden SWAP yap ve W ye yaz
RETFIE
```

SWAPF komutu bir register içerięini başka bir registere STATUS registerini deęiştirmeden yüklemek için kullanılır. Bu komutla alt 4 bit ile üst 4 bitin yeri deęişir. Bu nedenle alt programdan geri dönülmeden önce tekrar SWAPF komutu eski haline getirilmesi gerekir.

Bu işlem sırasında MOVF komutu STATUS registerindeki Z bayraęının içerięini deęiştirdiğinden zero flag durumunu kontrol ederek işlem yapan bir programda hatalara neden olabilir.

W registeri içeriği b'00001111'
STATUS registeri içeriği ise b'00110100' olsun ve yukarıdaki programı inceleyelim:

```
          00001111      W register
          00110100      STATUS register
KESME_ALT_PROG
1:        00001111      SAKLA_W
2:        01000011      W register
3:        01000011      SAKLA_S
; kesme alt programı komutları
4:        00110100      W register
5:        00110100      STATUS
6:        11110000      SAKLA_W
7:        00001111      W register
```

Kesme alt programı nereye yazılmalı

İlk komut h'004' adresine yazılmalıdır.

Kesme kullanan programların düzenlenmesi aşağıdaki gibi olmalıdır:

```
          ORG   h'000'
          GOTO  BASLA           ;ana program başlangıcı

          ORG   h'004'
          GOTO  INT_ALT_PROG    ;kesme alt programı başlangıcı
;
BASLA
    Ana program komutları
    .
    .
INT_ALT_PROG
    Kesme alt programı komutları
    .
    .
RETFIE
```

Kesme gecikmesi

Bir kesme oluştuğunda alt program başlamadan önce 3-4 komut sayıklı süresince gecikme meydana gelir. Zamanın önemli olduğu uygulamalarda göz önüne alınmalıdır.

DONANIM SAYICILARI

Donanım sayıcı/zamanlayıcı

PORT çıkışlarına gönderilen sinyaller arasında bir gecikme istediğimiz zaman bu gecikmeyi sağlaması için bir alt program yapılır bu alt programa software timer denir. Fakat bu işlemi PIC donanımı içerisinde yapan hardware timer bulunmaktadır. Genelde h'00' dan başlayıp içerisindeki birer birer artıran bir file registerdir. Sayma aralıkları nedeniyle gecikme olmaktadır. PIC de iki tip timer bulunmaktadır. TMR0 olarak adlandırılan 8 bit sayıcı ilkidir. RAM belleğin h'01' adresinde olan özel bir registerdir. İkincisi ise watchdog timer WDT adı verilen zamanlayıcıdır.

TMR0 sayıcı/zamanlayıcı

RAM belleğin h'01' adresinde bulunur. TMR0 programlanan bir sayıcıdır. Saymaya istenilen bir komuttan başlatılabilir, herhangi bir anda da içeriği sıfırlanabilir. Özellikleri:

- 8 bit sayıcıdır
- yazılabilir ve okunabilir
- programlanabilen frekans bölme değeri
- harici veya dahili clock ile sayı artışı
- harici olarak düşen veya yükselen kenar tetiklemesi
- sayıcı değeri artan yöndedir. (h'00', h'01'h'FF')

- TMR0 değeri h'FF' den h'00' a gelince ilgili flag 1 yaparak kesme oluşturur

Ana program veya kesme alt programları çalışırken sayma işlemini durdurmaz. Sayma işlemi devam ederken h'FF' den h'00' a geçişte meydana gelen taşma INTCON registerinin 2. bitinde (T0IF) 1 olarak görülür. Bu bayrak kontrol edilerek kesme oluşturulabilir.

OPTION register

RAM belleğin 1. BANK ındaki h'81' adresinde bulunan özel registerdir. Kesme sinyalinin tetikleme kenarını , PORTB nin pull-up yapılma durumunu, TMR0 veya WDT yi seçme bayrağını kontrol eden 8 bitlik registerdir.

RBPU(7.bit)	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0(0.bit)
--------------------	---------------	-------------	-------------	------------	------------	------------	-------------------

PS0, PS1, PS2: frekans bölme sayısı

Frekans bölme sayısı	TMR0 oranı	WDT oranı
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

PSA: frekans bölücü seçme biti

0=bölme sayısı TMR0 için geçerli

1=WDT için geçerli

T0SE: TMR0 sinyal kaynağı kenar seçme biti

0=RA4/T0CKI ucundan düşen kenar tetiklemesi

1=yükselen kenar tetiklemesi

T0CS: TMR0 sinyal kaynağı seçme biti

0=dahili komut saykılı seçilir

1=harici sinyal (RA4/T0CKI ucu)

INTEDG: harici kesme sinyali kenar seçme biti

0=RB0/INT ucundan düşen kenarda tetikleme

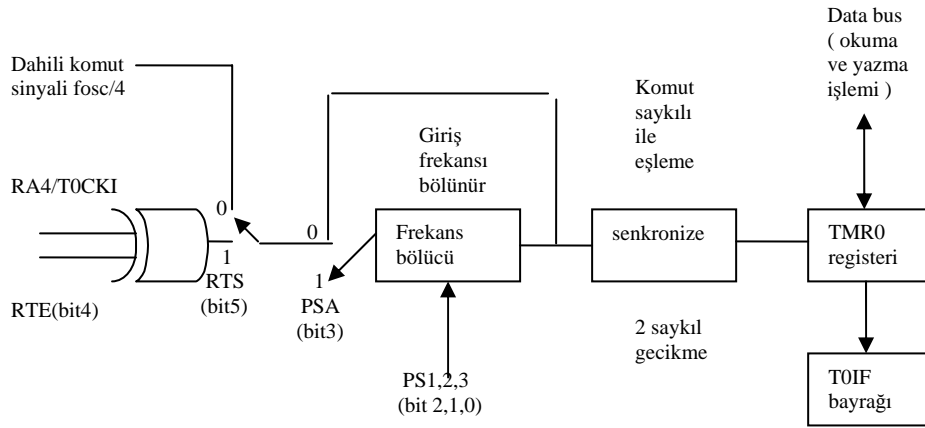
1=yükselen kenarda tetikleme

RBPU: PORTB pull-up geçerli yapma biti

0=PORTB uçlarındaki pull-up lar iptal edilir

1=geçerli yapılı

TMR0 sayıcı özellikleri



sayıcı içindeki sayının artırılması için gerekli clock sinyali iki farklı kaynaktan sağlanabilir.

- 1) dahili komut sinyali
- 2) harici sinyal (RA4/T0CKI)

- harici sinyal PORTA nın 3. bitinden (RA4/T0CKI)
- TMR0 in sinyal kaynağını seçmek için OPTION registerinin 5. biti (TOCS) kullanılır
- Sinyal kaynağından gelen sinyal direkt olarak TMR0 besleyebildiği gibi frekans bölücü aracılığı ile de beslenebilir
- Frekans bölme sayısı TRM0 veya WDT ye OPTION registerinin 3. biti (PSA) ile yönlendirilir. Her iki sayıcının da bölme oranları birbirlerinden farklıdır dikkat!
- OPTION registerinin 0,1,2. bitleri kullanılarak 8 farklı frekans bölme değeri seçilebilir. TMR0 sayıcısını tetikleyecek olan sinyal direkt olarak sinyal kaynağından sağlanmak isteniyorsa (frekans bölücüyü bypass ederek) frekans bölme değeri WDT ye atanır. Bu işlem PSA bitile yapılır
- Frekans bölme değeri TMR0 a atandığında TMR0 a yazmak için kullanılan tüm komutlar frekans bölme değerini siler
- Frekans bölme değeri kullanılmadan direkt olarak harici sinyal kullanılırsa dahili komut sinyali ile senkronizasyonu sağlamak için 2 saykılılık bir gecikme sağlanır
- Harici sinyal uygulandığında TMR0 sayıcısının içindeki sayının hangi kenarda artacağını belirlemek için OPTION registerinin 4. biti (T0SE) aracılığı ile belirlenir.
- TMR0 sayıcısından çıkış sinyalleri aşağıdaki durumlarda oluşur:
 - 1) RAM belleğin h'01' adresindeki TMR0 registeri okununca
 - 2) Sayıcının h'FF' den h'00' a geçişinde meydana gelen taşma sinyalinin INTCON registerinin 2. bitine (TOIF bayrağı) 1 yazılması anında
- TMR0 daki sayıları artıran sayıcıya uygulanan sinyallerdir
- Sayılar harici sinyal aracılığı ile artırıldığında istenirse bu sinyaller sayılabilir. Bu durumda TMR0 sinyal sayıcı olarak kullanılmış olur

Frekans bölme sayısının kullanılması

Frekans bölme sayısı	TMR0 oranı	WDT oranı
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 ve WDT oranı

TMR0 oranı 1/2 ise 2 komut saykılında bir defa üst sayıya geçmiş olur.

Frekans bölme sayısı b'000' seçilirse TMR0 ın sayma aralığı süresi ne olur?

Fosc=4 MHz ise Tkomut=1µs olur

TMR0 dahili komut saykılının 2 saykılında 1 defa artar o zaman:

TMR0 sayma aralığı süresi=Tkomut×TMR0oranı

=1µs×2=2µs buna göre TMR0 saymaya başladığında ilk sayı h'00' ise bu sayı h'FF'e

gelince kesme sinyali oluşacaktır. Bu sayı ondalık olarak 256 eder. Öyleyse TMR0 kaç saniyede kesme sinyali vereceği şöyle hesaplanır:

kesme gecikmesi=TMR0 sayma aralığı×256

=2µs×256=512µs şeklinde hesaplanır.

Frekans bölme sayısının atanması

TMR0 sayısını direkt olarak sinyal kaynağından beslemek için frekans bölme sayısını WDT ye atamaktır.

Frekans bölme değeri WDT ye atandığında TMR0 a yazmak için kullanılan CLRF, BSF, MOVWF gibi komutlar frekans bölme değerini sıfırlayarak yeni bir değer için hazırlar. Frekans bölme değeri TMR0 dan WDT ye veya tersi atama işlemi yapılırken frekans bölme sayısının sıfırlanması nedeniyle PIC çalışma esnasında istenmeyen resetler oluşabilir. Bunları önlemek için aşağıdaki program muhakkak kullanılmalıdır.

#TMR0 dan frekans bölme değeri atama:

```
BCF          STATUS, 5          ;BANK0 geç
CLRF         TMR0              ;TMR0 ve frekans bölme silinir
BSF         STATUS            ;BANK1 geç
CLRWDW      ;WDT sil
MOVLW      b'xxxx1xxx'       ;yeni frekans bölme değeri seç
MOVWF      OPTION_REG        ;OPTION registere yaz
BCF         STATUS, 5          ;BANK0 geç
```

#WDT den TMR0 a frekans bölme değeri atama:

```
CLRWDW      ;WDT yi ve frekans bölme değerini sil
BSF         STATUS, 5          ;BANK1 geç
MOVLW      b'xxxx0xxx'       ;TMR0 ı yeni frekans bölme değerini ve yeni sinyal kaynağı seç
MOVWF      OPTION_REG        ;OPTION registere yaz
BDF         STATUS, RP0        ;BANK0 geç
```

TMR0 sayıcısının kullanılması

TMR0 sayıcısının kurulması

- yukarıda verilen komutlar ile frekans bölme değerinin atanmasıyla
- OPTION registerin bitlerine uygun veri göndererek

TMR0 sayıcısının başlatılması

- sayıcıya bir sayı yazmak sayma işlemini başlatır

Saymaya başlama sayısının değiştirilmesi

- sayıcının h'00' dan değil de başka bir sayıdan başlatmak için TMR0 registerine istenen sayının atanması yeterlidir.

TMR0 ın çalıştığı nasıl anlaşılır

- bu registeri herhangi bir anda okuyarak
- herhangi bir bitini test ederek
- h'FF' den h'00' a geçişte oluşan kesme sinyalinin test ederek. Kesme anında INTCON registerinin 2. biti (TOIF bayrağı) 1 olur
- TMR0 çalışması esnasında PIC yüklendiği diğer işleri yapmaya devam eder

Neler yapılmadığı sürece TMR0 çalışmasına devam eder

- silinmediği sürece (CLRF TRM0)
- içerisine sayı yazılmadığı sürece (MOVLW h'05', MOVWF TMR0)
- PIC resetlenmediği sürece

Örnek program parçası:

```
WDT_KONT
    BTFSS     STATUS, 4    ;TO biti 1 mi?
    CALL     XXX
    MOVLW    h'FF'        ;hayır program akışına devam et
    MOVWF    TRISA
    .
    .
```

WDT uygulamasının kapatılması PIC programlayıcının ara yüzünden yapılır. Ancak bu program içerisinde bir konfigürasyon satırı oluşturularak da yapılabilir. Komut program esnasında disable yapılamaz sadece CLRWDT ile resetlenebilir. Bu durumda h'00' dan itibaren tekrar saymaya başlar sayma süresi bitince STATUS registerinin TO bayrağını 0 yapar.

SLEEP komutunun kullanılması

Komut genellikle PIC programının devamlı olarak çalışması gerekmeyen durumlarda kullanılır. Bu komut ile program olduğu yerde durur ve tüm özel registerler ve RAM bilgileri saklanır. SLEEP modundan çıkma WDT sinyali ve harici kesme sinyalleri vasıtasıyla yapılabilir.

D/A VE A/D ÇEVİRME İŞLEMLERİ

PWM metodu

Çıkış geriliminin bir periyot içerisindeki iş ve bekleme süresi ayarlanarak çıkış gerilim ortalama değeri 0-5V arasında değiştirilebilir.

İş ve bekleme süresi tespiti

RC osc ile beslenen PIC in RB0 ucundan 0-5V arasında değişen gerilim elde etmek için iş ve bekleme süresini belirleyen program şöyledir:

```
GECIKME
    MOVWF    SAYAC
DONGU
    DECFSZ   SAYAC, F
    GOTO     DONGU
    RETURN
```

Şeklinde. Örneğin 2.5 V lik çıkış için %50 iş süresi %50 bekleme süresi gereklidir. Bu durumda sayac registerine atanacak sayılar:

Register içerisindeki sayı max h'FF' olacağından bu da 256 döngü demektir.

$256 \times 50 = 128 \rightarrow h'80$ iş süresi ve bekleme süresi için

Tegin Yücel MAYADAĞLI