

ALT PROGRAMLAR

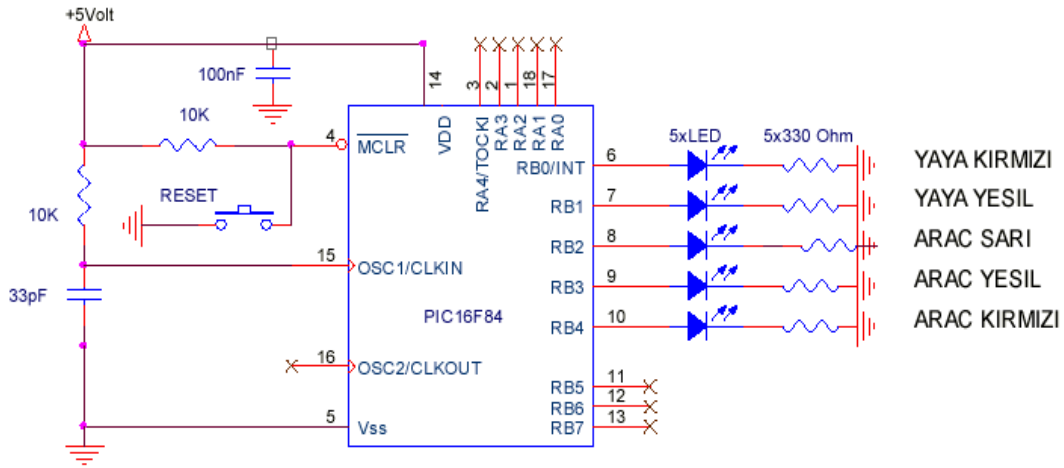
1.1.ALTPROGRAM NEDİR?

Programlamada döngü kadar etkili bir diğer kullanım şekli de alt programlardır. Bu sistemde işlemin birkaç yerinde lazım olan bir program parçasını tekrar tekrar yazmak yerine, bu bölümü bir kez ayrı bir program gibi yazıp, bu parçanın gerekli olduğu yerde programın bu kısmını çağırmak (ya da o kısma atlamak) metodu kullanılır. Bu sadece bir program yazımından kısaltma ile kurtulmak değildir. Bir tablodan istenilen verinin seçilmesi veya programın çok fazla hafıza isteyip işlemcinizin bunu karşılayamadığı durumlarda, alt programlar adeta bir can yeleğidir.

Alt program yazımının bir programı ne kadar kısaltacağını görebilmek için, döngüler kısmında çözülen trafik ışığı örneğini birde alt program kullanarak çözelim.

Örnek 1.1: Bir kavşaktaki trafik ışıklarının aşağıda verilen zaman ve sıra içerisinde çalışması isteniyor. Trafik ışığı olarak led kullanarak gerekli devreyi tasarlayınız ve programını yazınız.

Süre (Saniye)	Araç	Yaya
15	Yeşil	Kırmızı
5	Sarı	Kırmızı
25	Kırmızı	Yeşil
5	Sarı ve Kırmızı	Kırmızı



Şekil 1.1: Trafik ışığı sorusu devre şekli

Çözüm: Devre şekli üstte verilmiştir. Yine zamanlama için $200 \times 200 = 40000$ turluk bekleme döngüsünü, 1 saniye olarak kabul ediyoruz.

;Trafik ışığı problemi,

;Lambalar:RB0:Yaya-kırmızı, RB1:Yaya-Yeşil

;RB2:Araç-sarı, RB3:Araç-Yeşil, RB4:Araç-Kırmızı

;Süre ve durumlar

;15 sn Araç-Yeşil,Yaya-Kırmızı

;5 sn Araç-Sarı,Yaya-Kırmızı

;15 sn Araç-Kırmızı,Yaya-Yeşil

;15 sn Araç-Sarı-Kırmızı,Yaya-Kırmızı

LIST P=16F84

; Registerler

STATUS EQU 3h

PORTA EQU 5h

PORTB EQU 6h

TRISA EQU 5h

TRISB EQU 6h

; Değişkenler

ZD1 EQU 0Fh
ZD2 EQU 0Eh
zaman EQU 0Dh

X1 org 0h ; Power on
goto START ; 0000

START bsf STATUS,5 ; Page 1
movlw 0h ; 0000-0000 sayısını W registerine al
movwf TRISB ; PortB yi çıkış olarak ayarla
;TRISB=00000000

TOP bcf STATUS,5 ; Page 0
movlw 09h ; 0 0 0 0 1 0 0 1
movwf PORTB ; Araç Yeşil, Yaya Kırmızı
movlw 0Fh
movwf zaman
call BEKLE ; Bekle 15 saniye

movlw 05h ; 0 0 0 0 0 1 0 1
movwf PORTB ; Araç Sarı, Yaya Kırmızı
movlw 05h
movwf zaman
call BEKLE ; Bekle 5 Saniye

movlw 12h ; 0 0 0 1 0 0 1 0
movwf PORTB ; Araç Kırmızı, Yaya Yeşil
movlw 19h
movwf zaman
call BEKLE ; Bekle 25 saniye

```

        movlw    15h            ; 0 0 0 1 0 1 0 1
        movwf    PORTB          ; Araç Sarı-Kırmızı, Yaya Kırmızı
        movlw    05h
        movwf    zaman
        call     BEKLE          ; Bekle 5 Saniye

        goto     TOP            ; Tekrarla

```

```

;*****
;

```

```

; BEKLE alt programı

```

```

;*****
;

```

```

BEKLE    movlw    .200
          movwf    ZD1
D1        movlw    .200
          movwf    ZD2
D2        decfsz   ZD2,F
          goto     D2
          decfsz   ZD1,F
          goto     D1
          decfsz   zaman,F
          goto     BEKLE
          return

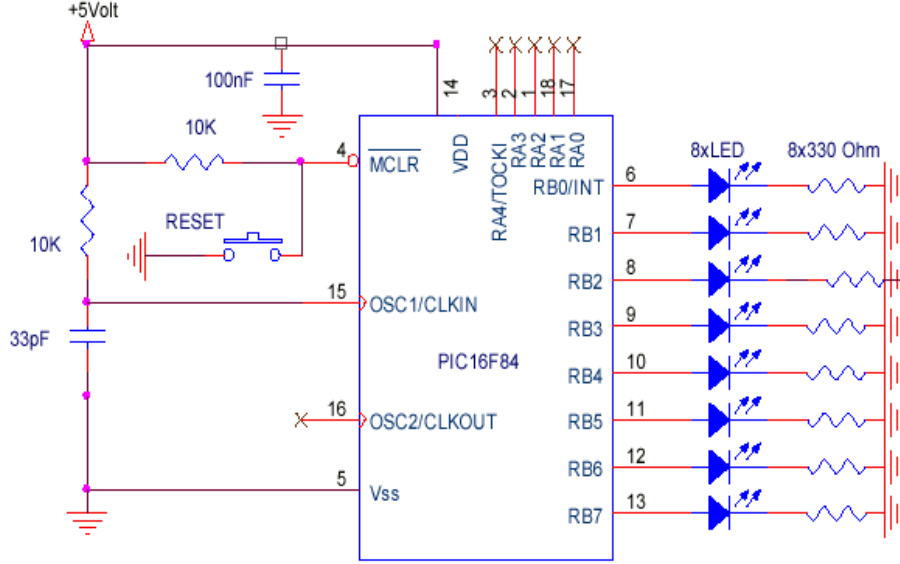
```

```

END

```

örnek 1-2: Port B'de bağlı olan 8 adet ledı yakıp söndüren bir flip,flop programı yazınız. Bu programı yazarken zaman gecikmesini alt programlar şeklinde yazınız?



Şekil 1.2: 8 adet led devresi (Yürüyen ışık devresi)

çözüm:

```

LIST      P=16F84
INCLUDE  "P16F84.INC"

S1      EQU    h'00'
S2      EQU    h'0D'

CLRFB    PORTB
BSF      STATUS,5
CLRFB    TRISB
BCF      STATUS,5

```

```

TEK      MOVLW    h'00'
          MOVWF    PORTB
          CALL     GECİKME
          MOVLW    h'FF'
          MOVWF    PORTB
          CALL     GECİKME
          GOTO     TEK

GECİKME  MOVLW    h'FF'      ;Alt program kısmı
          MOVWF    S1
DÖN1     MOVLW    h'FF'
          MOVWF    S2
DÖN2     DECFSZ   S2,F
          GOTO     DÖN2
          DECFSZ   S1,F
          GOTO     DÖN1
          RETURN   ;Altprogram sonu

END

```

Programın birinci kısmı port ayar işlemi yapmakta yani portB'nin tüm uçlarını çıkış olarak ayarlamaktadır.İkinci kısım ise porta b'00000000' bilgisi ile b'11111111' sayısını sırası ile göndererek port ucundaki ledlerin sönmesini ve yanmasını sağlamaktadır. Gecikme isimli son kısım ise porta yanma ya da sönme bilgisi gönderildikten sonra bekleme süresini ayarlamakta ve alt program şeklinde kullanılmaktadır.

soru : Aynı programı farklı mantıklar ve teknikler kullanarak yazınız?

Örnek 1.3: Port B'deki bağlı olan 8 adet ledi kayan ışık şeklinde çalıştıran programı yazınız. Program ledleri bir kez sırayla yaktıktan sonra tüm ledlerin sönük kalması şeklinde sona erecektir. (Şekil 1.2)

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

MOVLW h'01'
MOVWF PORTB
TEKRAR CALL BEKLE
RLF PORTB,F
BTFSS STATUS,0
GOTO TEKRAR
DUR GOTO DUR

BEKLE MOVLW h'FF'
MOVWF SAYAC1
DÖNGÜ1 MOVLW h'FF'
MOVWF SAYAC2
```

```

DÖNGÜ2  DECFSZ SAYAC2,F
        GOTO   DÖNGÜ2
        DECFSZ SAYAC1,F
        GOTO   DÖNGÜ1
        RETURN
        END

```

Soru= Programı işlemi sürekli yapacak hale getiriniz.

Örnek 1.4: Yukarıdaki örnekten faydalananarak sağa ve sola doğru kayan ışık (kara şimşek) programını yazınız? (Şekil 1.2)

```

                LIST P=16F84
                INCLUDE "P16F84.INC"
SAYAC1 EQU     h'0C'
SAYAC2 EQU     h'0D'
                CLRF  PORTB    ;PORTB=0
                BCF   STATUS,0 ; Carry=0
                BSF   STATUS,5   ;Portb=Çıkış
                CLRF  TRISB
                BCF   STATUS,5
                MOVLW h'01'
                MOVWF PORTB

SOL            CALL  BEKLE
                RLF   PORTB,F
                BTFSS PORTB,7
                GOTO  SOL

```



```

SAĞ    CALL    BEKLE
        RRF     PORTB,F
        BTFSS   PORTB,0
        GOTO    SAĞ
        GOTO    SOL
BEKLE   MOVLW   h'FF'
        MOVWF   SAYAC1
DÖNGÜ1 MOVLW   h'FF'
        MOVWF   SAYAC2
DÖNGÜ2 DECFSZ   SAYAC2,F
        GOTO    DÖNGÜ2
        DECFSZ   SAYAC1,F
        GOTO    DÖNGÜ1
        RETURN
        END

```

Örnek 1.5: Zeminden üst katlara yük taşıyan bir asansörün bakım sistemi için şöyle bir düzenek isteniyor. Asansörün yukarı çıkıp aşağı dönüşü 1 tur sayılmak kaydıyla 40.000 tur sonunda bakım zamanının geldiğine dair bir sarı ikaz lambasının yanması isteniyor. Eğer bakım yapılmadan çalışmaya devam eder ise 48.000 turdan sonra kırmızı ikaz lambasının yanması isteniyor. Şayet asansör bakım yapılmadan 2.000 tur daha çalıştırılırsa bu durumda asansörün çalışmasını engellemesi isteniyor. Bakım yapıldığında tur sayısının sıfırlanarak aynı işlemin tekrar devreye girmesi isteniyor. Bu işlem için gerekli devrenin prensip şemasını çizin ve programını yazınız.

(İpucu: Asansörün her bir turunu zeminde iken algılatma şeklinde düşünebilirsiniz.)

Çözüm: Burada esas olay asansörün iniş çıkış turlarını saydırmaktır. Normal sayıcı mantığıyla düşündüğünüzde bir değişkenle sayabileceğiniz en büyük sayı 255'tir. Bu şekilde düşündüğünüzde birler, onlar basamağı mantığı ile düşünüp 2

veya 4 deęişken kullanarak program yazılabilir ki biraz uzunca bir program olur. Ancak iç içe iki döngü ile tuş basmasını saydırırsak, bu durumda iç içe olmak üzere her ikisi de 200 turluk 2 döngü ile $200 \times 200 = 40000$ tur sonunda döngü tamamlanır. Sonunda 40000 tur sonunda yapılacak işlem yaptırılır. 8000 ve 2000 tur içinde aynı teknik kullanılabilir. Aşağıdaki çözüm bu mantığa göre tasarlanmıştır.

;Asansör koruma-bakım problemi

;RA0 ucunda asansor turlarını sayan buton sensor

;RB0 Sarı ışık ikaz

;RB1 Kırmızı ışık ikaz

;RB2 Asansor aktif/Pasif ucu, RB2=0 iken asansor çalışır durumda

	LIST	P=16F84	
STATUS	EQU	3h	
PORTA	EQU	5h	
PORTB	EQU	6h	
TRISA	EQU	5h	
TRISB	EQU	6h	
say	EQU	0Ch	;
D0	EQU	0Dh	; döngü degiskeni 0
D1	EQU	0Eh	; döngü degiskeni 1
D2	EQU	0Fh	; döngü degiskeni 2
X1	org	0h	; Power on
	goto	START	; 0000

START	bsf	STATUS,5	
	movlw	0h	
	movwf	TRISB	
	movlw	0fh	
	movwf	TRISA	
	bcf	STATUS,5	
	clrf	PORTB	
	movlw	.200	;Asansor 40000 tur calisti mi
	movwf	D1	;sorusu icin 200*200=40000
ZD1	movlw	.200	;turluk dongu icinde
	movwf	D2	;tus programi tekrarlanir
ZD2	call	tus	
	decfsz	D2,F	
	goto	ZD2	
	decfsz	D1,F	
	goto	ZD1	
	bsf	PORTB,0	;40000 tur olmustur SARI led yanar
	movlw	.40	;Asansor 8000 tur daha calisti mi?
	movwf	D1	;sorusu icin 40*200=8000
ZD3	movlw	.200	;turluk dongu icinde
	movwf	D2	;tus programi tekrarlanir
ZD4	call	tus	
	decfsz	D2,F	
	goto	ZD4	
	decfsz	D1,F	
	goto	ZD3	

```

        bcf      PORTB,0    ;48000 tur olmuştur SARI led söner
        bsf      PORTB,1    ;48000 tur olmuştur KIRMIZI led yanar

        movlw    .10        ;Asansor 2000 tur daha çalıştı mı?
        movwf    D1         ;sorusu için 10*200=2000
ZD5      movlw    .200       ;turluk dongu icinde
        movwf    D2         ;tus programı tekrarlanır
ZD6      call     tus
        decfsz   D2,F
        goto     ZD6
        decfsz   D1,F
        goto     ZD5

ZD7      bsf      PORTB,2    ;50000 tur olmuştur KIRMIZI led zaten
                                   ;yanıktır,
        goto     ZD7         ;Asansor duması için RB2 ucu 1 yapılır ve
                                   ;Bu konumdan çıkmak için RESET butonuna
                                   ;basılana kadar beklenir.

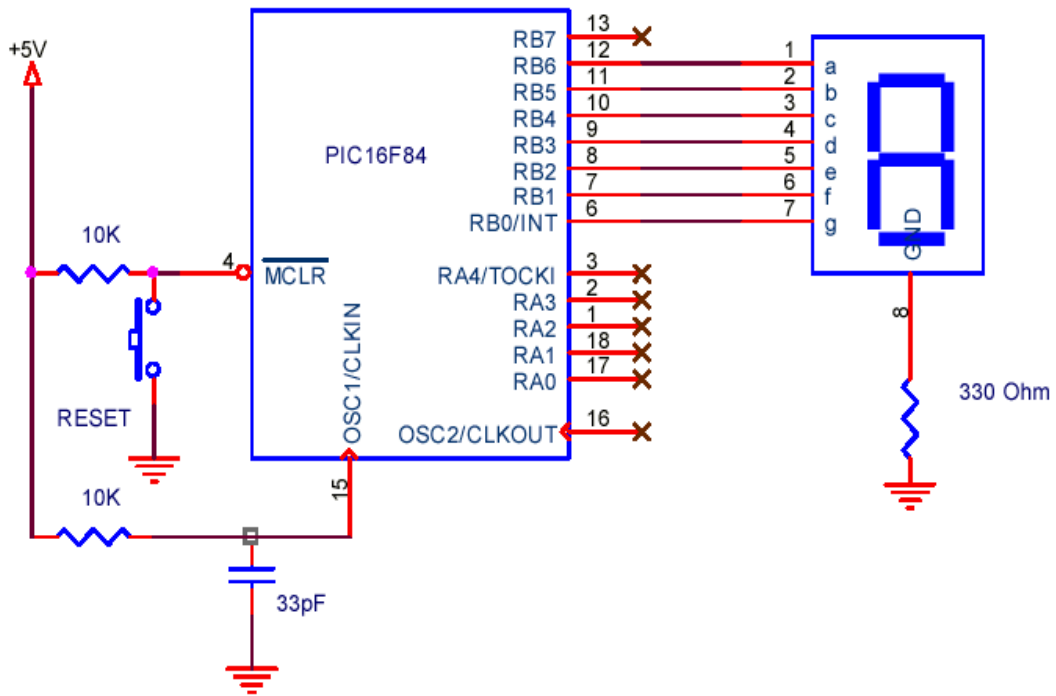
.*****
;
; TUS Altprogramı
.*****
;
tus      btfss    PORTA,0    ;Asansorun 1 turu için zemindeki
        goto     tus        ;butona basılıp bırakıldı mı? şeklinde
tus2     btfsc    PORTA,0    ;programla test etmek gerekir
        goto     tus2       ;tus ikilemesi gerekmez
        return
        END

```

1.2. ÇEVİRİM TABLOLARI

Bazı programlarda, bir tablodan belirli değerlerin seçtirilmesi, sıkça lazım olan bir durumdur. Bunun için CALL komutu RETLW komutu ile birlikte kullanılır. Sistemde CALL komutu ile bir alt program çağırılırken eğer W registerinde bir değer varsa ve bu değer PCL ile toplanırsa bu sistem bir index gibi kullanılır. Bu index ile atlanan adımda RETLW değer şeklinde bulunan komut satırındaki değer W registerine aktarılarak alt programdan çıkılır ve böylece ilgili indexteki değer tespit edilmiş olur. Bu işlemi bir örnekle görelim:

Örnek 1.6: PortB'ye bağlı olan seven segment displayı kullanarak, 0 ile F arasındaki sayıları beklemeler yaparak sayan programı tablo tekniğini kullanarak yazınız?



Şekil 1.3: Pic16f84 için tek display bağlantısı

```

LIST      P=16F84
INCLUDE  "P16F84.INC"

SAYAC1   EQU    h'0C'
SAYAC2   EQU    h'0D'
SAYAC    EQU    h'0E'


CLRF     PORTB      ;PORTB=0
BSF      STATUS,5   ;Portb=Çıkış
CLRF     TRISB
BCF      STATUS,5


BAŞLA    MOVLW    h'00'
          MOVWF    SAYAC
DÖNGÜ    MOVF     SAYAC,W
          ANDLW    B'00001111'
          CALL     CEV7SEG
          MOVWF    PORTB
          INCF     SAYAC,F
          CALL     BEKLE
          GOTO     DÖNGÜ


CEV7SEG  ADDWF    PCL,F
          RETLW    h'3F' ;0
          RETLW    h'06' ;1
          RETLW    h'5B' ;2
          RETLW    h'4F' ;3
          RETLW    h'66' ;4
          RETLW    h'6D' ;5
          RETLW    h'7D' ;6
          RETLW    h'07' ;7
          RETLW    h'7F' ;8

```

	RETLW	h'6F' ;9
	RETLW	h'77' ;A
	RETLW	h'7C' ;B
	RETLW	h'39' ;C
	RETLW	h'5E' ;D
	RETLW	h'79' ;E
	RETLW	h'71' ;F
BEKLE	MOVLW	h'FF'
	MOVWF	SAYAC1
DÖNGÜ1	MOVLW	h'FF'
	MOVWF	SAYAC2
DÖNGÜ2	DECFSZ	SAYAC2,F
	GOTO	DÖNGÜ2
	DECFSZ	SAYAC1,F
	GOTO	DÖNGÜ1
	RETURN	
	END	

INTERRUPT

2.1. INTERRUPT NEDİR:

Kelime anlamı olarak kesilim isteği anlamına gelen interrupt kelimesi, işlemcinin herhangi bir programı icra ederken, bu programa dışarıdan gelen bir sinyal sebebiyle ara verip, başka bir programa (interrupt alt programı) atlayıp, atladığı programı tamamladıktan sonra tekrar geri dönüp ilk kaldığı programdan devam etmesi işlemidir. Bu anlatımdan anlaşılacağı gibi interrupt bir alt program çalıştırma işleminin aynısı gibidir. Ancak, alt programın çağırılma şekli farklıdır. Bu fark ise normal alt programı çağırma işlemi CALL komutu ile yapılırken, interrupt alt programının ise, normal programın herhangi bir anında dışarıdan gelecek bir sinyal ile çağırılması olayıdır.

Uygulamada çok rastlanan bir örnekle konuyu açıklayalım: Bilgisayarınıza bağlı bir yazıcıdan 40 sayfalık bir yazının çıkarılması için yazdırma işlemi başlattınız ve sizde aynı zamanda bilgisayarda başka bir programı kullanıyorsunuz. Ancak, yazıcınızda diyelim ki 25 adet kağıt takılı olsun. Bu 25 kağıt yazıldıktan sonra, kalan 15 sayfa yazılamayacağı için bilgisayarınız sizin kullandığınız programa ara vererek, yazıcı ile ilgili bir mesajı ekrana çıkarır. İşte bu mesajın çıkarılma nedeni, yazıcınızın kağıt bitmesi sebebiyle interrupt istemiş olmasıdır. Mesajı ekrandan kaldırdıktan sonra eski kullandığınız programa tekrar kaldığınız yerden devam edebilirsiniz.

Interrupt alt programının yazılımı normal alt program ile aynıdır, ancak alt programın bitiş komutu RETFIE komutudur. Bu alt programın interrupt geldiğinde kullanılması için 3 işlemin program içerisinde yapılmış olması gerekir. Bu işlemler:

1. Interrupt alt programının adı, 04h adresine interrupt vektör adresi olarak tanıtılmış olmalıdır.

Örnek:

```
ORG 04h  
GOTO INTPRG
```


2. Interrupt ayarlarını sağlayan INTCON ve OPTION registerlerinin ayarlarının yapılmış olması gerekir. Bu registerlerin her bir bitinin bir anlamı vardır. Bu sebeple kullanılacak interrupt tipine göre ayarlanmış olmaları gerekir.

3. Kullanılan interrupt türüne göre interrupt geldiğinde INTCON registerinin kullanılan interrupt ile ilgili bir biti 1 olmaktadır. Interrupt alt programı tamamlandığında yeni interrupt gelmediği halde, tekrar interrupt alt programının çalışması istenmiyorsa, interrupt geldi anlamında 1 olan bitin alt program içerisinde tekrar 0 yapılması gerekir.

2.2. INTERRUPT ÇEŞİTLERİ:

Interruptları genel olarak ikiye ayırmak mümkündür:

1. Mask Edilemeyen Interruptlar: Bu tür interrupt'ta interruptun işlemci tarafından algılanması herhangi bir program mantığıyla engellenemez. Bu tür kesilime örnek olarak işlemcilerin reset butonlarını göstermek mümkündür ki; tam olarak yukarıda verdiğimiz tarife uymamakla birlikte reset tuşu programa ara verilip sistemin açılışa dönmesini sağlayan bir interrupt şeklindedir ve programla reset tuşu devre dışı bırakılamaz.

2. Mask Edilebilen Interruptlar: Bu tür interruptlar yukarıda verilen interrupt tarifine tam uyan interruptlardır. Bunlar istenirse öncelik sırasına dizilebileceği gibi, istenmezse hiç interrupt gelmemiş gibi programlanarak, yok farz edilebilirler. Bu işlemin nasıl olduğunu konunun ilerleyen detaylarında inceleyelim.

6.3. Pic16f84 için Interrupt çeşitleri:

Pic 16f84 işlemcisi için (MCLR=RESET hesaba katılmazsa) 3 tür interrupttan söz etmek mümkündür.

- 1) **RB0 Interrupt:** Bu tür interruptta interrupt alt programı, işlemcinin B portunun 0 numaralı (RB0) ucundan gelecek bir sinyale göre işleme girer. Bu sinyal alçalan kenar veya yükselen kenar şeklinde sürekli sinyal olabileceği gibi, bir puls sinyali de olabilir. Tek sinyalde interrupt kontrol edilmesi istenilen yerlerde kullanışlı olur.

- 2) TMR0 (timer) Interrupt: Bu tür interrupt, işlemcinin içerisinde programdan bağımsız çalışan bir sayıcının çalıştırılarak, sayıcının her 0'dan geçişte sisteme interrupt vermesi esasına göre çalışan çeşididir ki, burada sayıcı option registerdeki ayarlamalar sayesinde kendi içerisindeki osilatör sinyaline göre veya dışarıdan gelecek olan (RA4/T0CKI ucundan) sinyalleri sayma esasına göre programlanabilir. Zamana bağlı problemleri çözmede çok kullanışlı olur. Mesela saat, kronometre, vb...
- 3) RB4..RB7 uçlarındaki bilginin değişmesi esasına göre tasarlanmış bir interrupt çeşididir ki, iletişim ve güvenlik işlemlerinde kullanılması uygun olur.

Bu anlatılanlar ışığında INTCON register ve OPTION registerin içeriğine bakalım:

INTCON REGISTERİ:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	EEIE	T0IE	INTE	RBIE	TOIF	INTF	RBIF

Bit No	Adı	Açıklama
Bit 7	GIE	Tüm interrupt işlemlerini açma bayrağı 0: Tüm interruptlar disable 1: Tüm interruptlar enable
Bit 6	EEIE	EEPROM hafızaya yazma işlemlerini tamamlama interruptu 0: Disable 1: Enable
Bit 5	T0IE	TMR0 sayıcısı interruptunu aktif yapma bayrağı 0: Disable 1: Enable
Bit 4	INTE	Harici (RB0) interruptunu aktif yapma bayrağı 0: Disable (Harici interruptları geçersiz yapar- algılanmaz) 1: Enable (Harici interruptları geçerli yapar- algılanır)

Bit 3	RBIE	RB7..RB4 değişimi interruptunu aktif yapma bayrağı 0: Disable (RB7..RB4 değişimi algılanmaz) 1: Enable (RB7..RB4 değişimi interrupt oluşturur)
Bit 2	TOIF	TMR0 sayıcısı zaman aşımı bayrağı 0: Sayıcı normal saymaya devam ediyor 1: Sayıcı sayma süresini tamamlamış (FF-00 geçişi)
Bit 1	INTF	Harici (RB0) interrupt bayrağı 0: Harici interrupt algılanmadı 1: Harici interrupt algılandı
Bit 0	RBIF	RB7..RB4 değişimi interruptı bayrağı 0: (RB7..RB4) uclarında değişim yok 1: (RB7..RB4) uclarında değişimi var.

OPTION REGISTER

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit0
BPUR	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Bit No	Adı	Açıklama
Bit 7	RBPU	PORTB Pull Up Geçerli yapma biti 0: PORTB Pull Up'lar Geçersiz 1: PORTB Pull Up'lar Geçerli
Bit 6	INTEDG	Harici interrupt tipi seçme biti 0: RB0/INT ucundan alçalan kenar tetikleme 1: RB0/INT ucundan yükselen kenar tetikleme
Bit 5	T0CS	TMR0 sayıcısı sinyal kaynağı seçme biti 0: Dahili komut periyodu seçilir 1: Harici sinyal (RA4/T0CKI ucundan) periyodu seçilir
Bit 4	T0SE	TMR0 harici sinyal kaynağı kenar seçme biti 0: RA4/T0CKI ucundan alçalan kenar tetiklemesi 1: RA4/T0CKI ucundan yükselen kenar tetiklemesi
Bit 3	PSA	Frekans bölücü seçme biti 0: Frekans bölme sayısı TMR0 için geçerli 1: Frekans bölme sayısı WDT için geçerli
Bit 2	PS2	Frekans bölme sayısı biti 2 (Altındaki tabloya bakınız)
Bit 1	PS1	Frekans bölme sayısı biti 1 (Altındaki tabloya bakınız)
Bit 0	PS0	Frekans bölme sayısı biti 0 (Altındaki tabloya bakınız)

Frekans Bölme Sayısı (Bit2-Bit1-Bit0)	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

Yukarıdaki tablolar incelendiğinde INTCON registerinin 7 numaralı ucunun bu işte en önemli register olduğunu anlamak mümkündür. Çünkü bu uc 0 olduğunda tüm interrupt sinyalleri DISABLE yani algılanmaz olur. Yukarıda anlatılan her bir interrupt şeklinden kullanılacak olanı ayarlamak için birer uç (3-4-5) ve interrupt geldiğinde bunu belirten birer uç (0-1-2) olmak üzere 6 uc programlanabilen interrupt işlemlerine ayrılmıştır. Option register ise daha ziyade TMR0 ile ilgili ayarlar içindir. Option registerin 6 numaralı ucu ise RB0 ucunun alçalan ya da yükselen kenar şeklinde tasarlanması için kullanılmaktadır.

Şimdi interrupt ile ilgili örnekleri inceleyerek, bu işi daha detaylı öğrenmeye çalışalım.

Örnek 2.1: RB0 ucundan gelen sinyalleri sayacak ve ekrana gönderecek bir sayıcı programını interrupt kullanarak yazınız.

Çözüm : RB0 interrupt ile 0-9 sayıcı örneği
; (Display RB7-RB4 uçlarında 4056 ile bağlı)

```
list      p=16f84
include   "p16f84.inc"
say       equ    11h
tmp       equ    12h
```

```

org      00h
goto     basla
org      04h      ;Interrupt vektör adresi veriliyor
goto     saydır   ;Interrupt gelince saydır altprogramını
                    ; çalıştırır

basla     bsf      STATUS,5
          movlw    01h      ;RB0 interrupt kullanıldığından RB0 ucu giriş
          movwf    TRISB    ;B portunun diğer uçları çıkış
          movlw    b'01000000' ;RB0 ucundan yükselen kenar tetiklemesi
                                ;seçilmiş
          movwf    OPTION_REG
          bcf      STATUS,5

          movlw    b'10010000' ;Tüm interruptlar açık, RB0 interrupt aktif
                                ;RB0 interrupt algılaması 0
          movwf    INTCON
          clrf     say       ;Ekрана gidecek sayı 0 yapılmış

yaz       SWAPF    say,w     ;Display üst nibble'a bağlı olduğundan
          movwf    PORTB     ;ekrana gidecek bilgi swap edilerek alınmış
          goto     yaz

saydır    movwf    tmp       ;W'deki değer geçici hafızaya alınmalı
          bcf      INTCON,INTF ;RB0 interrupt algılamasını tekrar 0 yap
                                ;değilse tekrar int alt pr. Çalışır. Yeni gelecek
                                ;interrupt'un algılanması için 0 yapılmalı
          incf     say,1     ;bu kısım normal 10luk sayıcı programıdır
          movf     say,w
          sublw    .10
          btfss    STATUS,2

```

```

        goto    devam
        clrf    say
devam   movf    tmp,w          ;W'deki orijinal deęer geri yüklenmeli
        retfie
        end

```

Örnek 2.2: TMRO interruptunu kullanarak dahili sinyalin 256 peryodunu 1 kabul ederek belirli zaman aralıklarıyla sayan 99 sayıcı tasarlayınız.

Çözüm: TMRO interrupt kullanarak 99 sayıcı örneęi;
;Displaylar B portuna decoder ile baęlı

```

        list    p=16f84
        include "p16f84.inc"
say      equ    11h
tmp      equ    12h

        org     00h
        goto    basla
        org     04h
        goto    saydır

basla    bsf     STATUS,5
        clrf    TRISB
        movlw   b'00000111' ;dahili sinyal 1/256 oranı
        movwf   OPTION_REG
        bcf     STATUS,5

```

```

movlw    h'00'
movwf    TMR0        ;Timer 0 başlangıç sayısı 00
movlw    b'10100000' ;interruptlar açık, Timer 0 interrupt aktif

movwf    INTCON      ;Timer 0 zaman aşımı 0
clrf     say

yaz      movf    say,w
movwf    PORTB
goto    yaz

saydır   movwf    tmp        ;w'deki bilgi geçici değişkene yazılır
bcf      INTCON,T0IF ;Timer 0 zaman aşımını tekrar 0 yap
incf     say,1        ;değilse tekrar int alt pr. çalışır
movf     say,w
sublw    .10
btfss    STATUS,2
goto     devam
movf     say,w
andlw    h'F0'

addlw    10h
movwf    say
sublw    h'A0'
btfss    STATUS,2
goto     devam
clrf     say

```



```

devam    movlw    h'01'        ;TMR0 başlangıç sayısı 01h olarak ayarlanmış
        movwf    TMR0        ;Böylece 255 tur olunca ikaz vermesi sağlanır
        movf     tmp,w        ;W'deki orijinal değer geri yüklenir
        retfie
        end

```

Örnek 2.3: TMR0 interruptunu kullanarak desimal moda çalışan 0-99 sayıcı tasarlayınız.

Çözüm: TMRO interrupt kullanarak taramalı ekranda 99 sayıcı örneği
;Displaylar B portu üzerinde decoder ile paralel bağlı

```

        list      p=16f84
        include   "p16f84.inc"
say      equ      11h
tmp      equ      12h
        org       00h
        goto      basla
        org       04h
        goto      saydır
basla    bsf       STATUS,5
        clrf      TRISB
        movlw     0Ch
        movwf     TRISA
        movlw     b'00000111' ;dahili sinyal oranı 1/256
        movwf     OPTION_REG
        bcf       STATUS,5

```



```

movlw    h'00'
movwf    TMR0        ;Timer 0 başlangıç sayısı 00
movlw    b'10100000' ; Tüm interruptlar açık, TMR0 interrupt aktif
movwf    INTCON      ; TMR0 zaman aşımı 0 değerleri INTCON'a
                    ; yüklendi

clrf     say

yaz      movf        say,w
clrf     PORTA
bsf      PORTA,0
movwf    PORTB
SWAPF    say,w
clrf     PORTA
bsf      PORTA,1
movwf    PORTB
goto     yaz

saydır   movwf    tmp        ; W'deki değer geçici hafızaya alınmalı
bcf      INTCON,T0IF        ;Timer 0 zaman aşımını tekrar 0 yap
incf     say,1
movf     say,w
sublw    .10
btfss    STATUS,2
goto     devam
movf     say,w
andlw    h'F0'
addlw    10h
movwf    say
sublw    h'A0'
btfss    STATUS,2
goto     devam

```

```

        clrf      say

devam  movlw     h'01'      ;TMR0 başlangıç sayısı 01h
        movwf    TMR0
        movf     tmp,w      ;W'deki orijinal değeri geri yükle
        retfie
        end

```

Örnek 2.4 : TMR0 interruptunu kullanarak hex moda çalışan 0-FF sayıcı tasarlayınız.

Çözüm:

; TMR0 interrupt örneği
; TMR0 ile süre tutuluyor ve her bir süre sonunda
; B portuna 7 segment decoder ile bağlı displaylarda (0-FF) saydırılıyor.

```

        list     p=16f84
        include  "p16f84.inc"

        org     00h
        goto    basla
        org     04h
        goto    saydır

basla   bsf      STATUS,5
        clrf     TRISB
        movlw    b'00000000'    ;dahili sinyal 1/2 oranı
        movwf    OPTION_REG
        bcf      STATUS,5
        movlw    h'f9'

```

```

    movwf    TMR0        ;Timer 0 başlangıç sayısı 00
    movlw    b'10100000' ;interruptlar açık, Timer 0 interrupt aktif
                                ;Timer 0 zaman aşımı 0

    movwf    INTCON
    clrf     PORTB

dongu goto    dongu

saydır
    bcf      INTCON,T0IF  ;Timer 0 zaman aşımını tekrar 0 yap
                                ;değilse tekrar int alt pr. çalışır

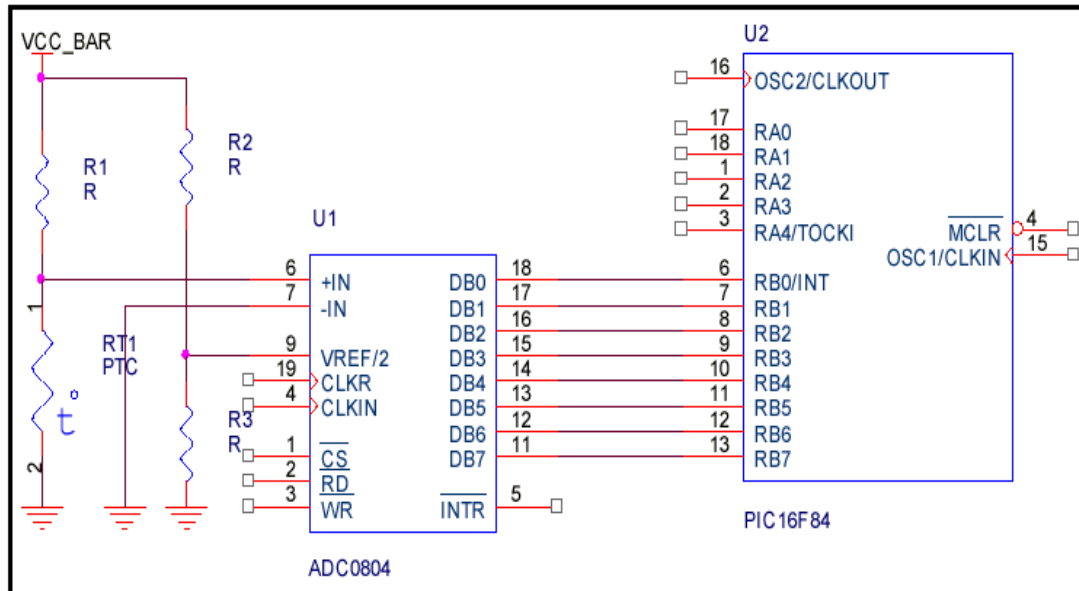
    incf     PORTB,f
    movlw    h'f0'        ;TMR0 başlangıç sayısı f0h
    movwf    TMR0
    retfie
end

```

ADC KULLANIMI

Mikrodenetleyici kullanırken en çok rastlanan husus, bir fiziksel büyüklüğün ölçülmesi ya da ayarlanması şeklindedir. Mesela ortam sıcaklığının ölçülmesi ya da ayarlanması, ortam ışıklandırılması vb... Bu gibi durumlarda ADC kullanılması gerekmektedir. Bazı PIC serisi işlemcilerde kendi içerisinde ADC mevcuttur, ancak biz burada 16f84 ile ADC kullanarak yapılabilecek işlemlere bakacağız. İşlemlerimizde ADC olarak ADC0804 elemanını kullanacağız.

8 bit ADC uygulamaları için, B portunun ADC çıkışlarına birebir olacak şekilde bağlandığını kabul edeceğiz. Böylece diğer işlemler için sadece A portunu kullanacağız. Tabii ki burada dışarıdan çoğullama kullanılarak çok farklı çözümler üretmenin mümkün olduğu unutulmamalıdır



PTC'nin istenilen sıcaklıktaki değerine yakın seçilirse sistemin çalışma hassasiyeti artar.

Burada "Sıcaklık derecesine göre, ADC çıkışındaki sayı kaç olur?" sorusunun cevabı, aşağıdaki örnekle verilebilir:

Örnek 3.1: PTC'nin 20°C 'deki direnci 5000 Ohm ise ve R1 direncinin değeri 10000 Ohm ise ADC çıkışındaki sayıyı bulunuz?

ADC girişindeki gerilim: $V = V_{cc} \cdot R_{PTC} / (R_1 + R_{PTC}) = 5.5 / (5 + 10) = 1,66 Volt$

1 sayı 19,53 milivolt ise

x sayı 1,66 Volt olur

şeklinde orantı kurulduğunda $x = 1,66 / 19,53 = 85$

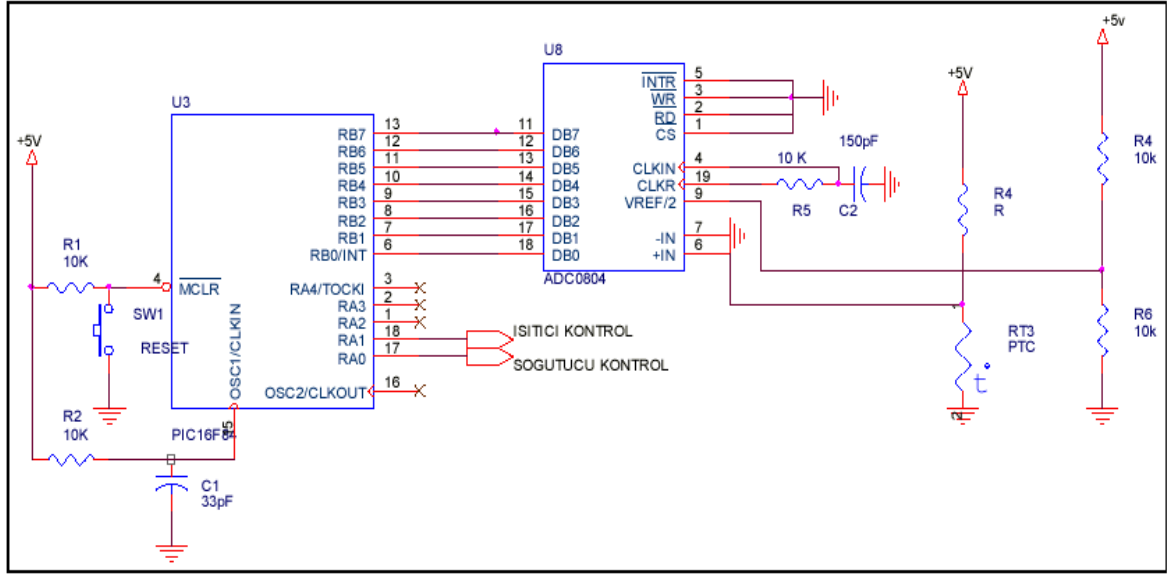
Böylece mesela 20 °C nin altındaki bir sıcaklıkta bir ısıtıcı devreye alınacaksa, ADC'nin çıkışı 85 sayısından küçük olduğu sürece ısıtıcı aktif edilir, ADC 85'ten büyük sayı veriyorsa ısıtıcı pasif duruma getirilir. Örnek problemlerde bunu görebiliriz.

örnek 3.2: Bir süpermarkette iç ortam sıcaklığının 22°C ile 25°C arasında ayarlanması isteniyor. Bu iş için sıcaklığı ölçmek amacıyla aşağıda değerleri verilen PTC kullanılmıştır. Gerekli devreyi çizin ve programını yazınız.

22 °C de PTC Direnci :22 K

25 °C de PTC Direnci :25 K

Çözüm : Burada, devrede bir harici ADC kullanılır ve PTC ile sabit bir direncin, gerilim bölücü olarak kullanılmasından faydalanılır. PTC direncinin değişimine göre de ADC'nin giriş voltajı (gerilim bölme kuralı ile) ile bu voltaja karşılık ADC'nin üreteceği sayı hesaplanır. Sonrasında problem, girişteki sayı X'den küçükse A işlemini yap, girişteki sayı Y'den büyükse B işlemini yap durumuna gelir.



Şekil 3.2: Pic16f84 ile ortam sıcaklığını kontrol devresi

- ;Market ısını 22 derece ile 25 derece arasında ayarlama işlemi
- ;B portu giriş yapıp ADC üzerinden sıcaklık bilgisi okunacak
- ;PTC ile gerilim bölücü direnç 25 K bağlanmış olup
- ;22 derecede 120 sayısı, 25 derecede 128 sayısı B portundan okunuyor
- ;RA0 Isıtıcı kontrol çıkışı
- ;RA1 Soğutucu kontrol çıkışı

```

LIST      P=16F84

STATUS   EQU    3h
PORTB    EQU    6h
TRISB    EQU    6h
PORTA    EQU    5h
TRISA    EQU    5h

X1        org    0h          ; Power on
          goto    START      ; 0000
  
```

```

START    bsf      STATUS,5
          movlw    h'FF'      ;B portu giriş
          movwf    TRISB
          movlw    00h        ;A portu çıkış
          movwf    TRISA
          bcf      STATUS,5

sil       clrf      PORTA

soguk     movf      PORTB,w
          sublw     .120
          btfss     STATUS,0
          goto      sıcak
          movlw     01h
          movwf     PORTA
          goto      soguk

sicak     clrf      PORTA
stest     movf      PORTB,w
          sublw     80h        ;desimal 128 sayısı
          btfsc     STATUS,0
          goto      sil
          movlw     02h
          movwf     PORTA
          goto      stest
          END

```

ÖRNEK PROBLEMLER:

Aşağıda işlenen tüm konuları içerisine alan çözülmüş örnekler verilmiştir. Bu örneklerde esas amaç, gerçek hayattaki problemlerin mikroişlemciye uyarlanmasını sağlamaktır. Ayrıca bu örnekler programlamanın değişik durumlarını öğretmesi açısından faydalı olacaktır.

ÖRNEK PROBLEM 1: Bir süpermarkette kapıların otomatik çalışması istenmektedir. Bu iş için kullanılacak optik sensörler görüş açısındaki cismi algılayınca 1, boşta iken 0 vermektedirler. Bir kapının giriş-çıkış şeklinde çalışması için gerekli devreyi tasarlayınız. Kapı ortasında kimsenin sıkışmaması için gerekli tedbiri alınız.

Çözüm: Burada en az 3 sensöre ihtiyaç olacaktır. Bunlar kapıda biri olduğunu algılayan sensör ile kapı tam açık ya da kapı tam kapalı şeklindeki değerleri veren limit switch şeklindeki kapı açık-kapalı sensörleridir. Bunlar olduğunda problem kapıda biri varsa ve kapı tam açık değilse kapı açma motorunu çalıştır. Kapıda kimse yoksa ve kapı kapalı değilse kapı kapatma motorunu çalıştır şeklinde düşünülür.

;Otomatik kapı sorusu çözümü

;RB0,RB1 uçlarında kapıda biri var işareti veren sensörler bağlı

;RB2 de kapı tam açık sensörü bağlı

;RB3 te kapı tam kapalı sensörü bağlı

;RA0 ucunda kapıyı açan motor bağlı

;RA1 ucunda kapıyı kapatan motor bağlı

	LIST	P=16F84
STATUS	EQU	3h
PORTA	EQU	5h
PORTB	EQU	6h

TRISA	EQU	5h	
TRISB	EQU	6h	
X1	org	0h	; Power on
	goto	START	; 0000
START	bsf	STATUS,5	
	movlw	h'FF'	;B portu giriş
	movwf	TRISB	
	movlw	00h	;A portu çıkış
	movwf	TRISA	
	bcf	STATUS,5	
sil	clrf	PORTA	
kontrol	movf	PORTB,w	
	andlw	03h	;sadece RB0 ve RB1 bilgilerini ayırmak için
	btfsc	STATUS,2	;kapıda biri yoksa ANDLW işleminin sonucu
			;0 dır
	goto	kapat	
ac	btfsc	PORTB,2	
	goto	sil	
	movlw	01h	
	movwf	PORTA	
	goto	kontrol	
kapat	btfsc	PORTB,3	
	goto	sil	
	movlw	02h	
	movwf	PORTA	
	goto	kontrol	
END			

ÖRNEK PROBLEM 2: Bir süpermarkette iç ortam aydınlığının 100 Lüks'ün üzerinde ayarlanması isteniyor. Bu iş için aydınlık derecesini ölçmek amacıyla aşağıda değerleri verilen LDR kullanılmıştır. Ayrıca aydınlatma lambaları iki ayrı grup altında birleştirilmiştir. Buna göre, ışık miktarı 80 Lüks'den az iken her 2 grup lambayı yanması, 80-100 Lüks arasında ise 1 grup lambanın yanması isteniyor. Eğer, ışık miktarı 100 Lüks'ün üzerinde ise lambaların sönmük olması isteniyor. Gerekli devreyi çizin ve programını yazınız.

80 Lüks'de LDR Direnci :100 K

90 Lüks'de LDR Direnci :90 K

100 Lüks'de LDR Direnci :80 K

Çözüm : Yine ADC gerektiren bir örnek. Burada aydınlığı ölçecek LDR ile bir gerilim bölücü direnç kullanarak ve kritik aydınlık değerlerinde ADC'nin üreteceği sayı hesaplanarak, büyük-küçük kıyaslamaları ile problem çözülür.

- ;Market aydınlığını 100 lüxe ayarlama işlemi
- ;B portu girilip ADC üzerinden aydınlık bilgisi okunacak
- ;LDR ile gerilim bölücü direnç 100 K bağlanmış olup
- ;80 Lüks de 128 sayısı, 100 Lüks de 113 sayısı B portundan okunuyor
- ;RA0 1. grup lamba kontrol çıkışı
- ;RA1 2. grup lamba kontrol çıkışı

LIST P=16F84

STATUS EQU	3h
PORTB EQU	6h
TRISB EQU	6h
PORTA EQU	5h
TRISA EQU	5h
da EQU	11h

```

X1      org      0h          ; Power on
        goto     START      ; 0000

START   bsf       STATUS,5
        movlw    h'FF'      ;B portu giriş
        movwf    TRISB
        movlw    00h        ;A portu çıkış
        movwf    TRISA
        bcf      STATUS,5

sil      clrf      da

siyah    movf     PORTB,w
        sublw    .128
        btfsc    STATUS,0
        goto     beyaz
        movlw    03h
        movwf    da
        goto     yaz

beyaz    movf     PORTB,w
        sublw    .113
        btfss    STATUS,0
        goto     gri
        clrf     da
        goto     yaz

gri      movlw    01h
        movwf    da

```

```

yaz      movf    da,w
          movwf   PORTA
          goto    sil

```

END

ÖRNEKPROBLEM 3: Bir cam üretim tesisinde, hareketli bant sistemi üzerindeki ürünlerin kalitesi 4 ayrı noktadaki optik sensörlerle kontrol ediliyor. Bu sensörlerin her biri malzeme normal ise 0, problemlili ise 1 veriyor. Her bir sensörün kendi yanında bir adet imha pistonu mevcut olup, herhangi bir sensörden hatalı imalat bilgisi gelirse, ona bağlı olan pistonun 1 mili saniye süreyle ileri gidip sonra tekrar geri gelmesi isteniyor. Bu işlem için gerekli devrenin prensip şemasını çiziniz ve programını yazınız.

çözüm:

```

;Cam üretim tesisi problemi
;RA0 ,RA1,RA2,RA3 uçları hatalı üretim sensörleri
;RB0, RB1,RB2, RB3 uçları piston kontrol çıkışları

```

```

          LIST      P=16F84
STATUS EQU        3h
PORTB EQU         6h
TRISB EQU         6h
PORTA EQU         5h
TRISA EQU         5h
D0      equ       11h          ;Zaman bekletme için değişken

X1      org       0h          ; Power on
          goto     START      ; 0000

```

```

START    bsf        STATUS,5
          movlw     00h          ;RB7,RB6,RB3,RB2,RB1,RB0 çıkış
          movwf     TRISB
          movlw     0Fh          ;A portu giriş
          movwf     TRISA
          bcf        STATUS,5
sil       clrf       PORTB

devam    movf       PORTA,w
          andlw     0Fh
          movwf     PORTB
          call      BEKLE
          goto      sil

;*****
;
; BEKLE 1 milisaniye Altprogramı
;*****
;
BEKLE    movlw     .40
          movwf     D0
ZD       decfsz    D0,1
          goto      ZD
          return
          END

```

ÖRNEKPROBLEM 4: Bir asansör sisteminin fazla yük konusundaki problemleri çözme konusu şu şekilde ayarlanıyor. Eğer asansördeki yük 3000 kilogramdan az ise tek motor devreye girerek asansörü çalıştırıyor. Eğer yük 3000-5000 kilogram arasında ise iki adet motor ile sistemin çalışması sağlanıyor. Eğer yük 5000 kilogramdan fazla ise, bu durumda asansörün hareket etmemesi ve aşırı yük alarminin devreye girmesi isteniyor. Bu problemin çözümü için kullanılan basınç sensörü hiç yük yokken değeri 1000 ohm olup elemanın direnç değeri her 1

kilogramda 1 ohm artmaktadır. Bu işlem için gerekli devrenin prensip şemasını çizin ve programını yazınız.

çözüm:

;Asansör için aşırı yük koruma işlemi
;B portu giriş yapıp ADC üzerinden ağırlık bilgisi okunacak
;Ağırlık sensörü ile gerilim bölücü direnç 6000 Ohm bağlanmış olup
;3000 kg da 102 sayısı, 5000 kg da 128 sayısı B portundan okunuyor
;RA0 1. motor kontrol çıkışı
;RA1 2. motor kontrol çıkışı
;RA3 alarm çıkışı

	LIST	P=16F84	
STATUS	EQU	3h	
PORTB	EQU	6h	
TRISB	EQU	6h	
PORTA	EQU	5h	
TRISA	EQU	5h	
da	EQU	11h	
X1	org	0h	; Power on
	goto	START	; 0000
START	bsf	STATUS,5	
	movlw	h'FF'	;B portu giriş
	movwf	TRISB	
	movlw	00h	;A portu çıkış
	movwf	TRISA	
	bcf	STATUS,5	

sil	clrf	da
agır	movf	PORTB,w
	sublw	.128
	btfsc	STATUS,0
	goto	orta
	movlw	04h
	movwf	da
	goto	yaz
orta	movf	PORTB,w
	sublw	.102
	btfss	STATUS,0
	goto	l1
	movlw	01h
	movwf	da
	goto	yaz
l1	movlw	03h
	movwf	da
yaz	movf	da,w
	movwf	PORTA
	goto	sil
	END	

ÖRNEKPROBLEM 5: 60 saniye süre içerisinde, istenilen süreye ayarlanabilen ve ayarlanan süreden birer saniye aralıklarla geri sayarak süre tamamlandığında bir zili çalan kronometre (zaman rolesi) yapılmak isteniyor. Bu işlem için gerekli devreyi tasarlayınız ve programını yazınız.

çözüm:

;RA0 ucu ayar butonu her basışta 1 geri saydırır
;RA1 ucu zaman başlatma butonudur, basılınca sistem
;geri sayar ve diğer buton iptal olur
;Süre 0 olunca RA2 ucundaki zil çıkışı 1 olur
;B portunda 2 adet 4056 ile iki display bağlıdır

```
                LIST      P=16F84
STATUS EQU      3h
PORTB EQU      6h
TRISB EQU      6h
PORTA EQU      5h
TRISA EQU      5h
sayi EQU      0Ch      ; PORTB'ye gönderilecek sayı
D1 EQU      0Eh      ; BEKLEME SAYACI 1
D2 EQU      0Fh      ; BEKLEME SAYACI 2

X1      org      0h      ; Power on
        goto     START   ; 0000
START   bsf      STATUS,5
        movlw    0h
        movwf    TRISB
        movlw    03h
        movwf    TRISA
        bcf      STATUS,5

        movlw    60H
        movwf    sayi
TOP      movf     sayi,W
        movwf    PORTB
```


ayar	btfsc	PORTA,0
	goto	eksil
ates	btfss	PORTA,1
	goto	ayar
say	movf	sayi,w
	movwf	PORTB
	call	BEKLE
	decf	sayi,1
	movf	sayi,W
	andlw	0Fh
	sublw	0Fh
	btfss	STATUS,2
	goto	sifir
onluk	decf	sayi,1
	decf	sayi,1
	decf	sayi,1
	decf	sayi,1
	decf	sayi,1
	decf	sayi,1
	goto	say
sifir	movf	sayi,W
	sublw	0h
	btfss	STATUS,2
	goto	say
dur	bsf	PORTA,2
	goto	dur

```

eksil    decf    sayi,1
         movf    sayi,W
         andlw   0Fh
         sublw   0Fh
         btfss   STATUS,2
         goto    sifir2

```

```

onluk2   decf    sayi,1
         decf    sayi,1
         decf    sayi,1
         decf    sayi,1
         decf    sayi,1
         decf    sayi,1
         goto    yaz

```

```

sifir2   movf    sayi,W
         sublw   0h
         btfss   STATUS,2
         goto    yaz
         goto    TOP

```

```

yaz      movf    sayi,w
         movwf   PORTB
         goto    ayar

```

```

;*****
;

```

```

; BEKLETME ALT PROGRAMI

```

```

;*****
;

```

```

BEKLE    movlw   .200      ;200*200 lük iki döngü yaklaşık 1 saniye
         movwf   D1        ;kabul edilecektir.

```

```

ZD1      movlw    .200
          movwf    D2
ZD2      decfsz   D2,1
          goto     ZD2
          decfsz   D1,1
          goto     ZD1
          return

```

END

ÖRNEKPROBLEM 6: Dönen platform şeklindeki bir alış-veriş reyonunda, kontrol sisteminin reyondaki malın çok azalması veya normalden fazla olması problemleri çözme konusu şu şekilde ayarlanıyor. Reyondaki yük 300 kilogramdan az ise tek motor devreye girerek reyonun dönmesini sağlıyor; eğer yük 300-500 kilogram arasında ise iki adet motor ile sistemin çalışması sağlanıyor. Şayet yük 500 kilogramdan fazla ise bu durumda sistemin hareket etmemesi isteniyor ve yük 700 kilogramdan fazla ya da 100 kilogramdan az ise yine sistemin çalışmayıp bir ikaz alarminin devreye girmesi isteniyor. Bu problemin çözümü için kullanılan basınç sensörü hiç yük yokken değeri 100 ohm olup elemanın direnç değeri her 1 kilogramda 1 ohm artmaktadır. Bu işlem için gerekli devrenin prensip şemasını çizin ve programını yazınız.

çözüm:

- ;Dönen platform için aşırı yük koruma işlemi
- ;B portu giriş yapıp ADC üzerinden ağırlık bilgisi okunacak
- ;Ağırlık sensörü ile gerilim bölücü direnç 600 Ohm bağlanmış olup
- ;300 kg da 102 sayısı, 500 kg da 128 sayısı
- ;700 kg da 146 sayısı B portundan okunuyor
- ;RA0 1. motor kontrol çıkışı
- ;RA1 2. motor kontrol çıkışı
- ;RA3 alarm çıkışı

```

LIST      P=16F84

STATUS    EQU    3h
PORTB     EQU    6h
TRISB     EQU    6h
PORTA     EQU    5h
TRISA     EQU    5h
da        EQU    11h


X1        org    0h          ; Power on
          goto   START      ; 0000


START     bsf     STATUS,5
          movlw   h'FF'      ;B portu giriş
          movwf   TRISB
          movlw   00h        ;A portu çıkış
          movwf   TRISA
          bcf     STATUS,5


sil       movlw   04
          movwf   da


hafif     movf    PORTB,w
          sublw   .102
          btfss   STATUS,0
          goto    orta
          movlw   01h
          movwf   da
          goto    yaz

```

orta	movf	PORTB,w
	sublw	.128
	btfss	STATUS,0
	goto	agir
	movlw	03h
	movwf	da
	goto	yaz
agir	movf	PORTB,w
	sublw	.146
	btfss	STATUS,0
	goto	yaz
	movlw	00h
	movwf	da
yaz	movf	da,w
	movwf	PORTA
	goto	sil

END

ÖRNEKPROBLEM 7: Bir pic16f84 buton şeklinde çalışan bir algılayıcının önünden geçen parçaları sayan 0-F sayıcı şeklinde çalışıyor. Ayrıca sayıcı olarak çalışan bu pic16f84'ün 11h adresindeki sayıyı (bu sayı sayıcının ekranda gösterdiği sayıdır) bir butona basılınca diğer bir pic16f84'ün 12h adresineaktarılmasını sağlıyor. Diğer pic16f84 ise birinciden gelen sayıyı alıyor ve ekranda gösteriyor . Bu sistem için gerekli devreyi çiziniz ve her iki pic16f84'te olması gereken pogramları yazınız. (İpucu: Sayıcı kısmını, haberleşme kısmını ve sayıları ekranda gösterme kısmını ayrı alt programlar şeklinde tasarlayınız.)

çözüm:

;Pic1 programı
;Pic ler arası interrupt kullanarak iletişim örneği
;1 nolu pic RB1 ucundan gelen sinyalleri sayıyor
;ve RB2 ucundaki butona basılınca o andaki sayıyı
;RB4-7 üzerinden 2. PIC'e gönderiyor.
;RA0-3 üzerinde 4056 ile display bağlı

```
list      p=16f84
include   "p16f84.inc"

say       equ    11h
tmp       equ    12h
tmp2      equ    13h


org       00h
goto      basla
org       04h
goto      intprg


basla     bsf     STATUS,5
          clrf    TRISA
          movlw   b'00000111'
          movwf   TRISB
          movlw   b'01000000'      ;Yükselen kenar tetiklemeli RB0
          movwf   OPTION_REG
          bcf     STATUS,5


          movlw   b'10010000'      ;interruptlar açık, RB0 interrupt aktif
          movwf   INTCON


sil       clrf    say
          clrf    tmp
```

```

sayar    movf    say,w
          movwf   PORTA
tus      btfss   PORTB,1
          goto    tus2
          incf    say,1
          movf    say,w
          sublw   0Fh
          btfss   STATUS,2
          goto    sayar
          goto    sil

tus2     btfss   PORTB,2
          goto    sayar
          swapf   say,w
          movwf   PORTB
          bsf     PORTB,3
          bsf     tmp,0
bekle    btfsc   tmp,0
          goto    bekle
          clrf    PORTB
          goto    sayar

intprg   movwf   tmp2
          clrf    tmp
          bcf     INTCON,INTF
          movf    tmp2,w
          retfie
          end

```

;2 nolu pic programı

;PIC ler arası interrupt kullanarak iletişim örneği

;1 nolu PIC RB1 ucundan gelen sinyalleri sayıyor

;ve RB2 ucundaki butona basılınca o andaki sayıyı

;RB4-7 üzerinden 2. PIC'e gönderiyor.

;RA0-3 üzerinde 4056 ile display bağlı

```
list      p=16f84
include   "p16f84.inc"

say       equ    11h
tmp       equ    12h


org       00h
goto      basla
org       04h
goto      oku


basla     bsf     STATUS,5
          clrf    TRISA
          movlw   b'11110001'
          movwf   TRISB
          movlw   b'01000000'      ;Yükselen kenar tetiklemeli RB0
          movwf   OPTION_REG
          bcf     STATUS,5


          movlw   b'10010000'      ;interruptlar açık, RB0 interrupt aktif
          movwf   INTCON


sil       clrf    say
goster    movf    say,w
          movwf   PORTA
          goto    goster
```



```

oku      movwf    tmp
          bcf      INTCON,INTF
          SWAPF    PORTB,w
          andlw    0Fh
          movwf    say
          bsf      PORTB,3
          movf     tmp,w
          clrf     PORTB
          retfie

          end

```

Örnek Sorular: Aşağıda kendinizi geliştirebilmeniz için örnek birkaç soru verilmiştir.

Örnek Soru 1: 7 segment display kullanarak bir display sistemi tasarlayınız ve bu sistemde istenilen bir kelimeyi ekranda gösterecek programı yazınız?

Örnek Soru 2: Bir basketbol maçı için skorbord tasarlayalım. Öyle ki iki adet butondan birincisi her basışta ekrandaki sayıyı 1 artıracak. İkincisi her basışta sayıyı 1 azaltacak. Üçüncü bir buton ise sistemi başlangıca döndürecek olsun. Ayrıca artırma ve azaltma butonları basıldığı anda değil biraz basılı tutulunca artırma ve azaltma işlemini yerine getirsin. Başa alma butonu ise dokunulduğu anda sistemi başa alsın.

Örnek Soru 3: Bir elektronik termometre tasarlayınız ve uygun programı yazınız?

Örnek Soru 4: Bir ohmmetre tasarlayınız ve uygun programı yazınız?

Örnek Soru 5: Bir şifreli kilit sistemi tasarlayınız?

UYGULAMALAR:

Kitabın bu bölümünde daha önceki bölümlerde verilen örnekler ve problemlere ek olarak değişik uygulamalar verilmiştir. Bu uygulamalar sınıf içi uygulamalara uygun olacak şekilde seçilmiştir. Bu bölümde, öğrencinin programlama mantığının iyi derecede geliştirilmesinin yanında, mekanik uygulama yaparak pratik bilgi ve becerisinin gelişmesi, devre tasarlayabilir hale gelmesi, ayrıca zaman döngüleri ile tablo mantığının kullanılması ve interrupt içeren programlar yazma-çalıştırma ve deneme becerisi kazanması amaçlanmaktadır. Böylece gerçek hayattaki problemlere çözüm geliştirmede daha verimli olunması sağlanacaktır.

Bu çalışmaların tamamında deney adımları aynı şekildedir. İlk adım, programın bilgisayara yazılması, ikinci adım bilgisayarda denenmesi, üçüncü adım programın mikrodenetleyiciye kaydedilmesi ve son adım programın mekanik devrede denenmesidir. Şimdi bu adımları sırasıyla inceleyelim:

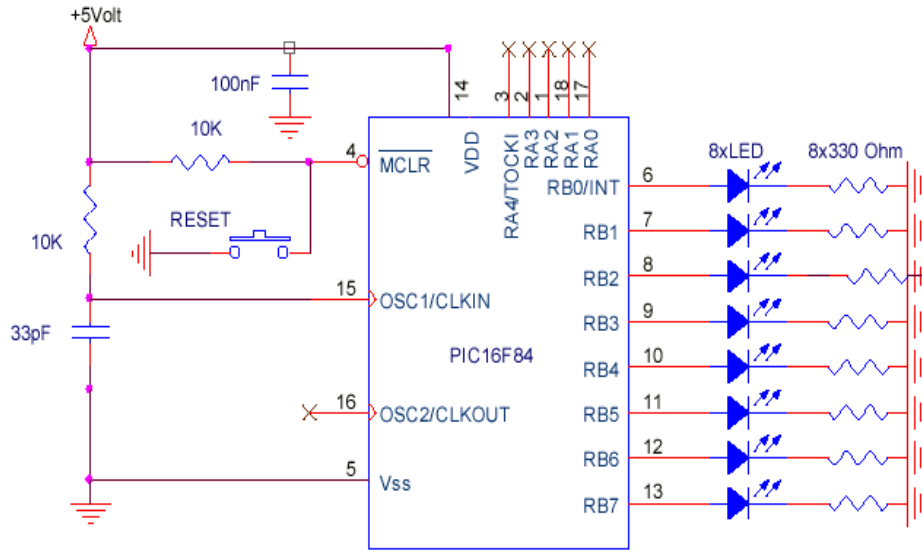
UYGULAMA ADIMLARI

- 1) Deney için gerekli ASM dosyasını MPLAB programında yazınız. Programınızı ISIM.ASM şeklinde isim vererek kaydediniz.
- 2) Aynı isimle proje oluşturarak programınızı çalışır hale getiriniz.. (Bakınız: MPLAB kullanımı)
- 3) Programınızı MPLAB yardımıyla deneyiniz. Programın içerisinde döngü işlemi varsa süreleri kısaltarak deneyiniz. Programın testi bittikten sonra döngü süreleri ile ilgili rakamları olması gereken değerlerle değiştirip yeniden BUILD ALL işlemi yapınız.

- 4) PICALLW programını çalıştırınız. FILE menüsünden OPEN FILE seçeneği ile daha önce çalıştırdığınız ISIM.ASM dosyasının ISIM.HEX dosyasını çağırınız. Üstteki pencerelerde entegre adının ve programlayıcı adının doğru seçilmiş olmasına dikkat ediniz.
- 5) CONFIG mönüsünden RC OSİLATÖR tipini seçiniz. WDT kullanılmıyor ise WDT seçeneğinin yanındaki tik işaretini kaldırınız. (Bakınız: Picallw kullanımı)
- 6) Programlama setinizin bağlantılarını yapınız. (Programlama kablosunu Printer portuna takınız. Güç bağlantılarını yapınız.
- 7) Programlama setine güç uygulayınız. Setin yeşil ışığı yanıyor ise set hazırdır. Picallw programı üzerindeki PROGRAM butonuna tıklayarak PIC'i programlayınız. Eğer hata mesajı almış iseniz ders öğretmenine müracaat ediniz. Eğer hata mesajı yok ise PIC normal programlanmış demektir.
- 8) Programlanmış PIC'i programlama setinden çıkararak deney için hazırlanmış devreye takınız.
- 9) Deney devresinin güç bağlantılarını dikkatlice yapınız. **(+5 Volt ve 0 Volt)**
- 10) Sisteme güç vererek çalışmasını izleyiniz.
- 11) Önemli not: Deneylerin başlangıcında kullanılacak uygulama devresinin yapısı kısaca verilmiştir. Buradaki bazı uygulamalarda, kullanacağınız uygulama devresine uymayan kısımlar vardır. Bu durumları program içerisinde gerekli değişiklikleri yaparak düzenleyiniz.

Burada başlangıçta MPLAB ve Picallw programlarının kullanılmasını bilmek gerektiği açıktır. Bu programların kullanımı ile ilgili bilgiler ek.1 ve ek.2 bölümlerinde verilmiştir.

DENEY 1: Yürüyen Işık



Şekil U.1:Yürüyen ışık devresi

;8 ışıklı yürüyen ışık programı

;8 ışıklı yürüyen ışık programı

;ledler 330 ohmluk dirençlerle port b'ye bağlanacak

;10K,33pf osilatör bağlantısı ile her bir ışık yaklaşık 1 sn süreyle yanar.

;PIC 16F84

;WATCHDOG DEVRE DIŞI

list p=16f84

status equ 03

portb equ 06 ; port adresi

trisb equ 86h ;

s1 equ 0Ch

s2 equ 0Dh

giden equ 0Eh

```

        org     0                ; reset vektörü
        goto    init

init     clrf    portb
        bsf     status,5
        clrf    trisb
        bcf     status,5

main     movlw   01h
        movwf   giden
tekrar   movf    giden,0
        movwf   portb
        call    bekle
        rlf     giden,1
        goto    tekrar

bekle    movlw   .210
        movwf   s1
a1        movlw   .210
        movwf   s2
d2        decfsz  s2,1
        goto    d2
        decfsz  s1,1
        goto    a1
        return

end

```

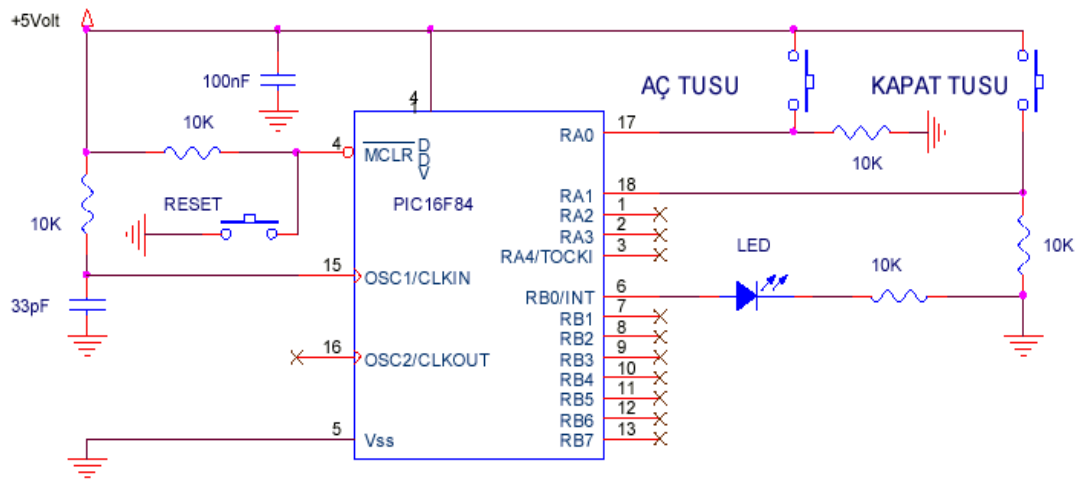
```

bekle    movlw    .210
          movwf    s1
a1        movlw    .210
          movwf    s2
d2        decfsz   s2,1
          goto     d2
          decfsz   s1,1
          goto     a1
          return

end

```

DENEY 2: 2 butonla 1 ledi yak-söndür deneyi



Şekil U.2: 2 buton, 1 led bağlantısı devresi

İki butonla bir ledi yak-söndür programı

```

;PROGRAM    AC_KAPA.ASM
;PIC        16F84
;WATCHDOG   DEVRE DIŞI
;FONKSİYON   AC butonuna basıldığında LED yanar,
;            KAPA butonuna basıldığında LED söner.

```

```
;          PORTA'nın 0 nolu ucu AÇ butonuna bağlıdır.
;          PORTA'nın 1 nolu ucu KAPA butonuna bağlıdır.
;          PORTB'nin 0 nolu ucu LED'e bağlıdır.
```

```
list p=16f84
```

```
status      equ      03
porta       equ      05      ; port adresi
portb       equ      06      ; port adresi
trisa       equ      85h      ;
trisb       equ      86h      ;
#define ac   porta,0      ; A portunun 1.bitine bağlı buton
#define kapa porta,1      ; A portunun 2.bitine bağlı buton
#define led  portb,0      ; B portunun 0.bitine bağlı LED
```

```
org 0      ; reset vektörü
goto init
```

```
,***** ana program burada başlıyor *****
```

```
init      clrf      portb      ; Port B'yi sıfırlayarak başla.
          bsf        status,5   ; 1.yazmaç sayfasını seç.
          movlw      0fh
          movwf      trisa      ; A portunu giriş yap.
          clrf      trisb      ; B portunu çıkış yap.
          bcf        status,5   ; 0.yazmaç sayfasına dön.

main      btfsc      ac          ; AC butonuna basılıp basılmadığını kontrol et,
          ; basılmışsa sonraki adımı atla.
          goto      main      ; Basılana kadar döngüye gir.
```

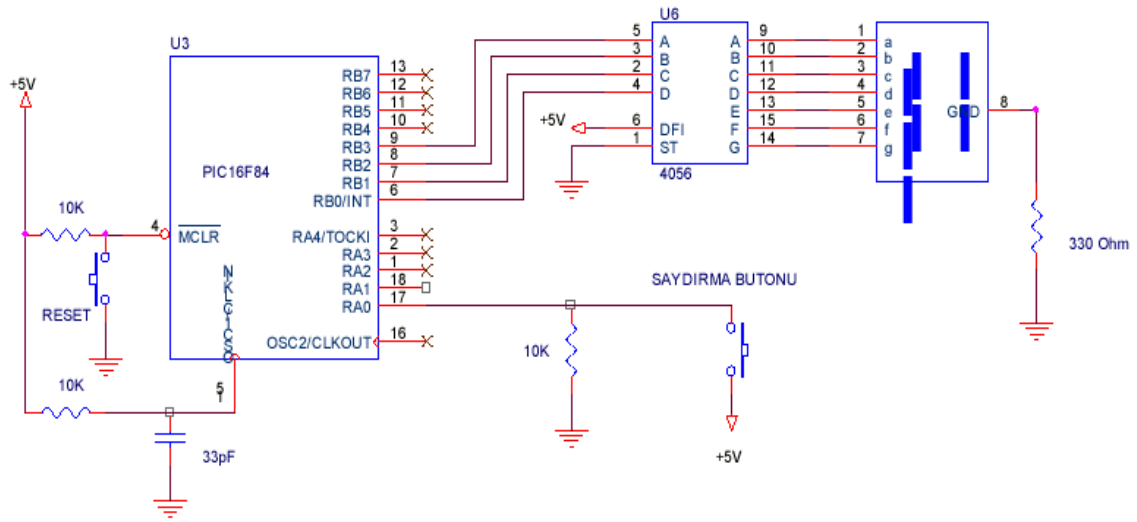
```

        bsf      led      ; LED'i yak.
test    btfss    kapa     ; KAPA butonuna basılıp basılmadığını kontrol et,
        ; basılmışsa sonraki adımı atla.

        goto     test     ; Basılana kadar döngüye gir.
        bcf      led      ; LED'i söndür.
        goto     main     ; Başa dön.
        end        ; Programın sonu.

```

DENEY 3 : 1 butonla tek bit sayıcı



Şekil U.3: 7 segment decoder kullanarak 1 buton 1 display bağlantısı

; 7-segment göstergede başçek buton kullanılarak
 ; gerçekleştirilen 0-9 arası sayıcı.
 ; B portuna bağlı seven segmen decoder ile display sürülmektedir.
 ;; Program RA0 butonuna her basışta 0-9 arası yukarı sayar

LIST P=16F84

```

STATUS EQU    3h
PORTA  EQU    5h
PORTB   EQU    6h
TRISA   EQU    5h
TRISB   EQU    6h

```



```

say    EQU    0Ch    ; counter to turn on the pins on PortB
D0     EQU    0Dh    ; delay counter 0
D1     EQU    0Eh    ; delay counter 1
D2     EQU    0Fh    ; delay counter 2

```

```

X1     org    0h      ; Power on
      goto   START    ; 0000

```

```

START  bsf    STATUS,5
      movlw  0h
      movwf  TRISB
      movlw  0fh
      movwf  TRISA
      bcf    STATUS,5

```

```

TOP1   movlw  00H
      movwf  say

```

```

TOP2   movf   say,W
      movwf  PORTB
      call  TUS

```

```

      incf   say,1
      movf   say,w
      sublw  0Ah
      btfss  STATUS,2
      goto   TOP2
      goto   TOP1

```

```

.*****
;

;TUS Subrutine
.*****
;

TUS    btfss    PORTA,0
        goto    TUS
        call    DELAY

BIRAK  btfsc    PORTA,0
        goto    BIRAK
        return

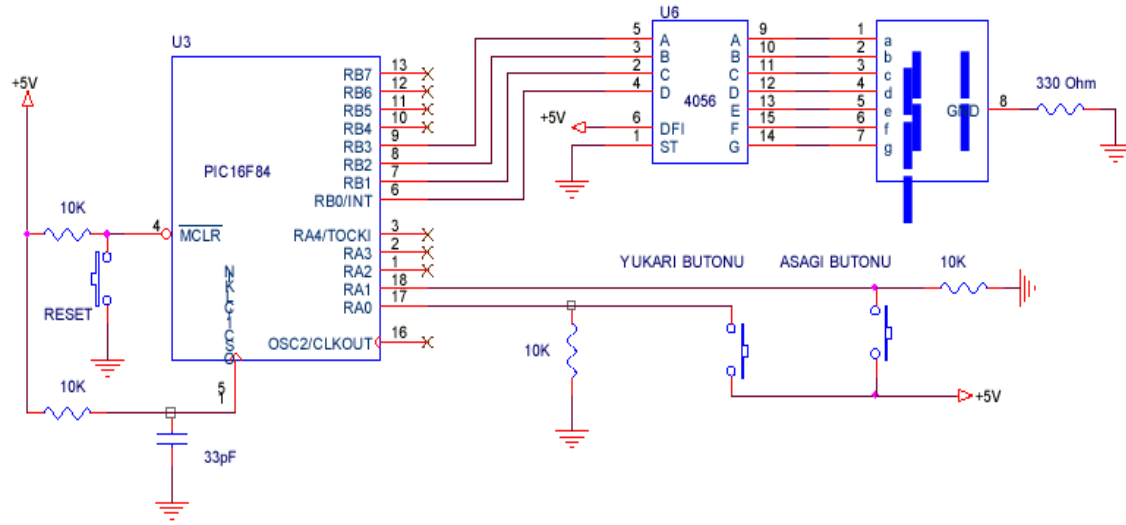
.*****
;

; DELAY Subroutine
.*****
;

DELAY  movlw    .2
        movwf   D0
ZD0    movlw    .200
        movwf   D1
ZD1    decfsz   D1,F
        goto    ZD1
        decfsz   D0,F
        goto    ZD0
        retlw   00
        END

```

DENEY 4: Tek bit yukarı aşağı sayıcı



Şekil U.4: 7 Segment decoder kullanarak 2 buton, 1 display bağlantısı

;0-9 yukarı-aşağı sayıcı

;Program buton sayıcı RA0 butonuna her basışta yukarı sayar

;RA1 butonuna basınca aşağı sayar (0-9 Arası)

;Yukarı ve asagi tuslarına beraber basılırsa yukarı sayar

```
LIST      P=16F84
STATUS EQU 3h
PORTA EQU 5h
PORTB EQU 6h
TRISA EQU 5h
TRISB EQU 6h

say EQU 0Ch ; counter to turn on the pins on PortB
D0 EQU 0Dh ; delay counter 0
D1 EQU 0Eh ; delay counter 1
D2 EQU 0Fh ; delay counter 2
```

```

X1      org      0h          ; Power on
        goto     START      ; 0000

START   bsf       STATUS,5
        movlw    0h
        movwf    TRISB
        movlw    0fh
        movwf    TRISA
        bcf      STATUS,5
TOP1    movlw     00H
        movwf    say
TOP2    movf      say,W
        movwf    PORTB

Y_TEST  btfss     PORTA,0
        goto     A_TEST
        call     Y_TUS
A_TEST  btfss     PORTA,1
        goto     Y_TEST
        call     A_TUS
        GOTO     Y_TEST

;*****
;
;YUKARI SAY TUSU Subrutine
Y_TUS   btfss     PORTA,0
        goto     Y_TUS
        call     DELAY
Y_BIRAK btfsc     PORTA,0
        goto     Y_BIRAK

```

```

        incf    say,1
        movf    say,w
        sublw   0Ah
        btfss   STATUS,2
        goto    S_CIKIS
        movlw   0h
        movwf   say
S_CIKIS movf    say,W
        movwf   PORTB
        return

.*****
;
;ASAGI SAY TUSU Subroutine
A_TUS   btfss   PORTA,1
        goto    A_TUS
        call    DELAY
A_BIRAK btfsc   PORTA,1
        goto    A_BIRAK

        decf    say,1
        movf    say,w
        sublw   h'FF'
        btfss   STATUS,2
        goto    DEVAM
        movlw   09h
        movwf   say
DEVAM   movf    say,W
        movwf   PORTB
        return

.*****
;
; DELAY Subroutine
.*****
;

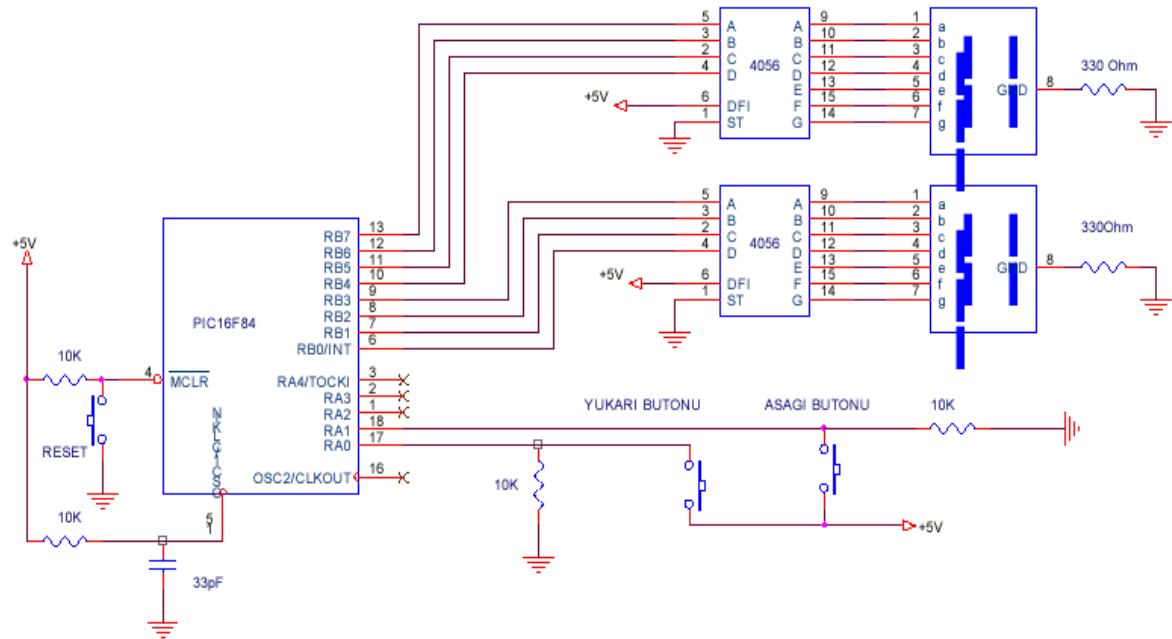
```

```

DELAY    movlw    .2
          movwf    D0
ZD0       movlw    .2
          movwf    D1
ZD1       decfsz   D1,F
          goto     ZD1
          decfsz   D0,F
          goto     ZD0
          retlw    00
          END

```

DENEY 5: 2 bit ileri-geri sayıcı



Şekil u.5: 2 buton, 2 display bağlantısı

;0-99 yukarı-aşağı butonlu sayıcı

;Program buton sayıcı RA0 butonuna her basışta yukarı sayar

;RA1 butonuna basınca aşağı sayar (0-99 Arası)

Yukarı ve aşağı tuslarına beraber basılırsa yukarı sayar

;Displaylar B portunda 7 segment decoder ile bağlanmıştır.

LIST P=16f84

```
STATUS      EQU      3h
PORTA EQU      5h
PORTB EQU      6h
TRISA  EQU      5h
TRISB  EQU      6h

say      EQU      11h      ; counter to turn on the pins on PortB
D0       EQU      12h      ; delay counter 0
D1       EQU      13h      ; delay counter 1
D2       EQU      14h      ; delay counter 2
TEMP     EQU      15h      ; Geçici register

X1       org      0h      ; Power on
         goto     START   ; 0000

START    bsf      STATUS,5
         movlw    0h
         movwf    TRISB
         movlw    0fh
         movwf    TRISA
         bcf      STATUS,5

TOP1     movlw    00H
         movwf    say
TOP2     movf     say,W
         movwf    PORTB
Y_TEST   btfss    PORTA,0
         goto     A_TEST
         call     Y_TUS
```

```

A_TEST  btfss    PORTA,1
        goto     Y_TEST
        call     A_TUS
        GOTO     Y_TEST

.*****
;
;YUKARI SAY TUSU Subrutine
.*****
;
Y_TUS   btfss    PORTA,0
        goto     Y_TUS
        call     DELAY
Y_BIRAK btfsc     PORTA,0
        goto     Y_BIRAK

        incf     say,1
        movf     say,w
        andlw    .15
        sublw    .10
        btfss    STATUS,2
        goto     S_CIKIS

        movf     say,W
        andlw    h'f0'
        addlw    .16
        movwf    say
        andlw    h'f0'
        sublw    h'A0'
        btfss    STATUS,2
        goto     S_CIKIS
        movlw    00h
        movwf    say

```



```

S_CIKIS  movf      say,W
          movwf     PORTB
          return

.*****
,

;ASAGI SAY TUSU Subrutine
.*****
,

A_TUS    btfss     PORTA,1
          goto      A_TUS
          call      DELAY
A_BIRAK  btfsc     PORTA,1
          goto      A_BIRAK

          decf      say,1
          movf      say,w

          andlw     .15
          sublw     .15
          btfss     STATUS,2
          goto      DEVAM

          movf      say,W
          andlw     h'f9'
          movwf     say

          andlw     h'f0'
          sublw     h'F0'
          btfss     STATUS,2
          goto      DEVAM
          movlw     99h
          movwf     say

```

```

DEVAM  movf      say,W
        movwf    PORTB
        return

```

```

;*****
;

```

```

; DELAY Subroutine

```

```

;*****
;

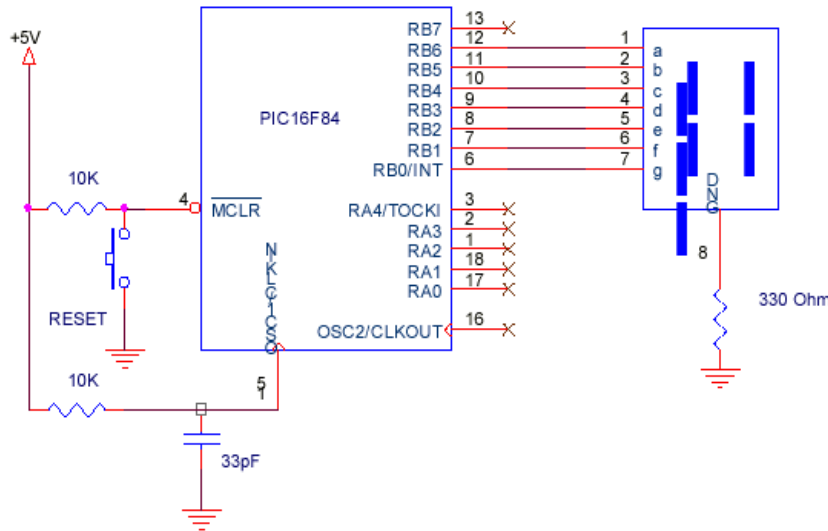
```

```

DELAY  movlw     .2
        movwf    D0
ZD0    movlw     .2
        movwf    D1
ZD1    decfsz    D1,F
        goto     ZD1
        decfsz    D0,F
        goto     ZD0
        retlw    00
END

```

DENEY 6 : Tek display ile decoder kullanmadan yapılan sayıcı



Şekil U.6: Tek display bağlantısı

;Tablo mantığı kullanarak 0-9 sayıcı yapımı

; Tablo programı örneği

```
list      p=16f84
include "p16f84.inc"

sayı      equ      11h
org       00h
goto      basla

basla     bsf       STATUS,5
          clrf      TRISB
          bcf       STATUS,5
t2        movlw    00h
          movwf     sayı

t1        movf     sayı,w
          call      tablo
          movwf     PORTB
          call      bekle
          incf      sayı,1
          movf     sayı,w
          sublw    0ah
          btfss    STATUS,2
          goto     t1
          goto     t2

bekle     nop
          nop
          return
```

```

tablo    addwf    PCL,f
         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'

```

```

.*****
,

```

```

; DELAY Subroutine

```

```

.*****
,

```

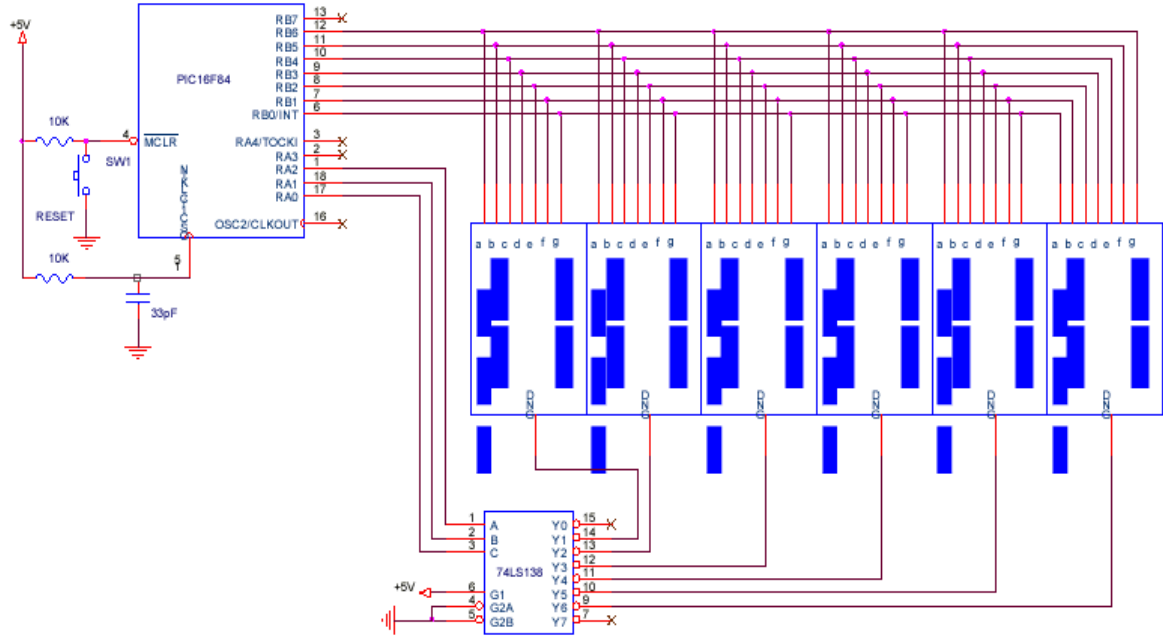
```

DELAY    movlw    .2
         movwf    D0
ZD0       movlw    .2
         movwf    D1
ZD1       decfsz   D1,F
         goto     ZD1
         decfsz   D0,F
         goto     ZD0
         retlw    00
         END

```

DENEY 7: 6 display ile sabit yazı yazma

; SABİT YAZI YAZDIRMA



Şekil U.7: 6'lı taramalı display bağlantısı

;Bu program taramalı çalışan 6 display üzerinde sabit bir yazıyı yazar.

;Display bağlantısı:

;a=RB0

;b=RB1

;c=RB2

;d=RB3

;e=RB4

;f=RB5

;g=RB6

;Sıralama d1,d2,d3,d4,d5,d6

;Select uçları RA2,RA1,RA0 üzerine bağlanmış 3 to 8 mux ile yapılmaktadır.

;Multiplexerin Y0 çıkışı boş bırakılmıştır.

;Diğer uçlar sırayla d1..d6 ya bağlanmış ve Y7 boş bırakılmıştır.

;Örnek data : -OGUZ- şeklindedir.

LIST P=16f84

; Registerler

STATUS	EQU	3h
PORTA	EQU	5h
PORTB	EQU	6h
TRISA	EQU	5h
TRISB	EQU	6h

; Değişkenler

D1	EQU	11H
D2	EQU	12H
D3	EQU	13H
D4	EQU	14H
D5	EQU	15H
D6	EQU	16H

X1	org	0h
	goto	START

START	bsf	STATUS,5	; Bank 1
	clrf	TRISB	; PortB çıkış
	clrf	TRISA	; PortA çıkış
	bcf	STATUS,5	; Bank 0

hazır	movlw	40h	;1. harf datasını D1 adresine gönder
	movwf	D1	
	movlw	3fh	;2. harf datasını D2 adresine gönder
	movwf	D2	

```

movlw 7dh          ;3. harf datasını D3 adresine gönder
movwf D3

movlw 3eh          ;4. harf datasını D4 adresine gönder
movwf D4

movlw 5bh          ;5. harf datasını D5 adresine gönder
movwf D5

movlw 40h          ;6. harf datasını D6 adresine gönder
movwf D6

tt    call    yazar    ;Yazı yazma alt programını çağır
      goto    tt      ;Yazma işlemini sürekli yap

yazar  movf    D1,w     ;1. datayı 1. displayda göster
      clrf    PORTA
      movwf   PORTB
      movlw   01h
      movwf   PORTA

      movf    D2,w     ;2. datayı 2. displayda göster
      clrf    PORTA
      movwf   PORTB
      movlw   02h
      movwf   PORTA

      movf    D3,w     ;3. datayı 3. displayda göster
      clrf    PORTA
      movwf   PORTB
      movlw   03h
      movwf   PORTA

```

```

movf    D4,w      ;4. datayı 4. displayda göster
clrf    PORTA
movwf   PORTB
movlw   04h
movwf   PORTA

movf    D5,w      ;5. datayı 5. displayda göster
clrf    PORTA
movwf   PORTB
movlw   05h
movwf   PORTA

movf    D6,w      ;6. datayı 6. displayda göster
clrf    PORTA
movwf   PORTB
movlw   06h
movwf   PORTA
return

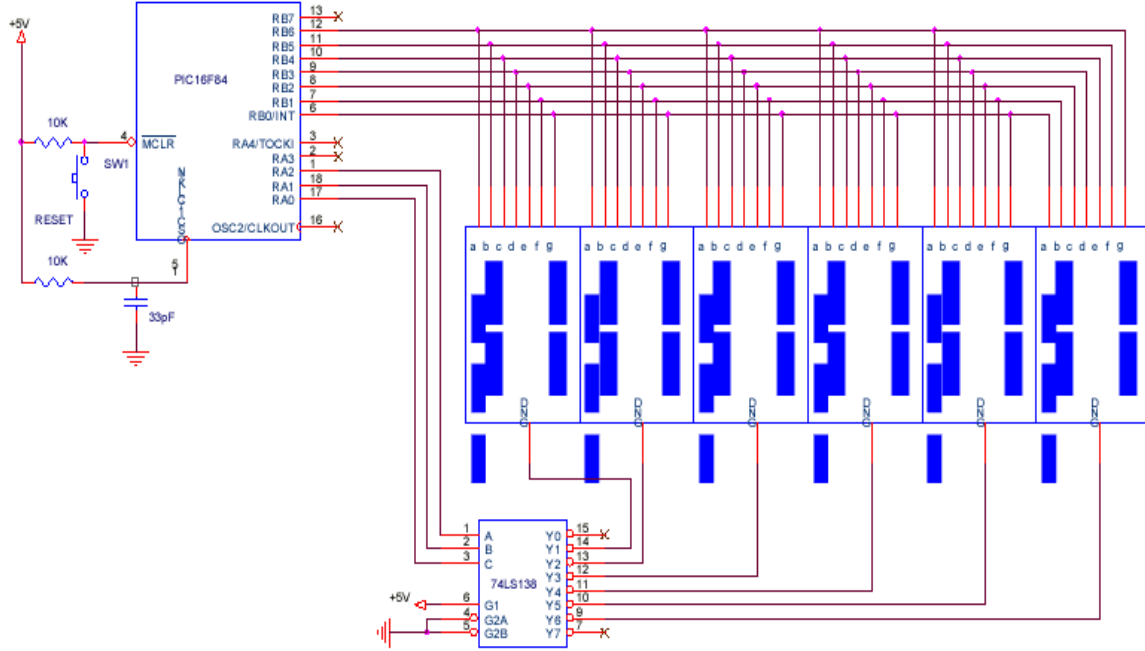
end

```

Bu örnekteki yazı yazma modelinde harflerin sabit olması sebebiyle ek bir bilgiye ihtiyaç olmaksızın istediğimiz yazıyı ekrana çıkartmak mümkün olmaktadır. Ancak dataların değiştiği programlarda bu işlem şu ana kadar öğrendiğimiz komutlarla pek pratik olmaz. Bu sebeple çok fazla sayıdadatanın sırasıyla porta gönderilmesi gibi durumlarda tablo mantığını kullanmak gerekir. Tablo mantığında veriler birbirini izleyen adreslere sırası ile yerleştirilir. Sonra bir indeks mantığıyla istenilen adresteki data çağrılmış olur. Burada indeks oluşturmak için datanın kaçınıcı sırada olduğu bilineceğinden, bu sıra sayısını bir değişkene aktarıp, tablonun başlangıcına atladıktan sonra, bu aktarılan değeri PCL (program Counter) registerindeki sayı ile

topladığımızda bizi istenilen datanın adresine aktarır. Buradan RETLW komutu ile istediğimiz datayı alıp programa geri dönmek mümkün olacaktır. Bunu aşağıdaki kayan yazı örneği ile deneyelim.

DENEY 8: 6 display ile kayan yazı deneyi



Şekil U.8: 6'lı taramalı display bağlantısı

;Display Örneği 2 KAYAN YAZI YAZDIRMA

;Bu program taramalı çalışan 6 display üzerinde sabit bir yazıyı kaydırarak yazar.

;Display bağlantısı:

;a=RB0

;b=RB1

;c=RB2

;d=RB3

;e=RB4

;f=RB5

;g=RB6

;Sıralama d1,d2,d3,d4,d5,d6

;Select uçları RA2,RA1,RA0 üzeine bağlanmış 3 to 8 mux ile yapılmaktadır.

;Multiplexerin Y0 çıkışı boş bırakılmıştır.

;Diğer uçlar sırayla d1..d6 ya bağlanmış ve Y7 boş bırakılmıştır.

;Örnek data : -bAhAr GELdl hOSGELdl- şeklindedir.

LIST P=16f84

; Registerler

STATUS EQU 3h

PORTA EQU 5h

PORTB EQU 6h

TRISA EQU 5h

TRISB EQU 6h

PCL EQU 02h

; Değişkenler

ZD1 EQU 0Fh

ZD2 EQU 0Eh

h_ad EQU 17h

ilk EQU 18h

;Display değişkenleri adresleri

D1 EQU 11H

D2 EQU 12H

D3 EQU 13H

D4 EQU 14H

D5 EQU 15H

D6 EQU 16H

X1 org 00h

goto START

```

START  bsf    STATUS,5    ; Bank 1
        clrf   TRISB      ; PortB çıkış
        clrf   TRISA      ; PortA çıkış
        bcf    STATUS,5    ; Bank 0

tekrar movlw   .30        ;Harf adedini tespit et
        movwf  h_ad

        movlw  00h        ;İlk data adresi 0 olacak
        movwf  ilk

hazır  movf    ilk,w      ;data adresindeki harfi almak için index ayarla
        call   tablo      ;tablodan harfi seç
        movwf  D1         ;ilgili display adresine gönder
        incf   ilk,1      ;Indexi 1 artır

        movf    ilk,w      ;Aynı işlemi 2. display için tekrarla
        call   tablo
        movwf  D2
        incf   ilk,1

        movf    ilk,w      ;Aynı işlemi 3. display için tekrarla
        call   tablo
        movwf  D3
        incf   ilk,1

        movf    ilk,w      ;Aynı işlemi 4. display için tekrarla
        call   tablo
        movwf  D4
        incf   ilk,1

```

	movf	ilk,w	;Aynı işlemi 5. display için tekrarla
	call	tablo	
	movwf	D5	
	incf	ilk,1	
	movf	ilk,w	;Aynı işlemi 6. display için tekrarla
	call	tablo	
	movwf	D6	
	decf	ilk,1	;Bir sonraki tur için indexi ayarla (4 azalt)
	decf	ilk,1	
	decf	ilk,1	
	decf	ilk,1	
	movlw	.1	;Bir turun ekrandaki süresini ayarla
	movwf	ZD1	
t2	movlw	.200	
	movwf	ZD2	
t1	call	yazar	;Ekran yazma programını döngü süresince tekrar
			;tekrar çağır
	decfsz	ZD2,1	
	goto	t1	
	decfsz	ZD1,1	
	goto	t2	
	decfsz	h_ad,1	;Mesajın tamam olup olmadığını kontrol et
	goto	hazır	
	goto	tekrar	;Mesaj tamam ise baştan başla

;Display adreslerindeki dataları ekrana yazdıran alt program

```
yazar  movf   D1,w           ;d1 adresindeki datayı al
        clrf   PORTA        ;PortA'yı sil
        movwf  PORTB        ;d1 datasını portB'ye gönder
        movlw  01h          ;A portundan 1. displayı seç
        movwf  PORTA

        movf   D2,w           ;Aynı işlemi ikinci display için tekrarla
        clrf   PORTA
        movwf  PORTB
        movlw  02h
        movwf  PORTA

        movf   D3,w           ;Aynı işlemi üçüncü display için tekrarla
        clrf   PORTA
        movwf  PORTB
        movlw  03h
        movwf  PORTA

        movf   D4,w           ;Aynı işlemi dördüncü display için tekrarla
        clrf   PORTA
        movwf  PORTB
        movlw  04h
        movwf  PORTA

        movf   D5,w           ;Aynı işlemi beşinci display için tekrarla
        clrf   PORTA
        movwf  PORTB
        movlw  05h
        movwf  PORTA
```

```

movf    D6,w           ;Aynı işlemi altıncı display için tekrarla
clrf    PORTA
movwf   PORTB
movlw   06h
movwf   PORTA
return

```

;Mesaj datalarını tutan alt program

```

tablo  addwf  PCL,1      ;Mesaj harfler tablosu
        retlw  00h        ;İlk 5 data 00 (boşluk)
        retlw  00h
        retlw  00h
        retlw  00h
        retlw  00h
        retlw  7ch        ;Gerçek data başlangıç adresi
        retlw  77h
        retlw  74h
        retlw  77h
        retlw  50h
        retlw  00h
        retlw  7dh
        retlw  79h
        retlw  38h
        retlw  5eh
        retlw  06h
        retlw  00h
        retlw  74h
        retlw  3fh
        retlw  6dh
        retlw  7dh
        retlw  79h

```

```
retlw    38h
retlw    5eh
retlw    06h        ;Dataların sonu
retlw    00h        ;Sonunda 6 adet boşluk
retlw    00h
retlw    00h
retlw    00h
retlw    00h
retlw    00h
end
```

EK.1. MPLAB KULLANIMI

Bu bölümde MPLAB Ver 3.31.00 versiyonunun çalıştırılması ve kullanımı ana hatlarıyla anlatılacaktır. MPLAB programının diğer versiyonları da bu versiyon ile menü açısından benzerlik göstermektedir. Bu sebeple bu bölüm iyi anlaşıldığında, programın diğer versiyonları da kullanılabilir. MPLAB programı diskette veya CD ile alındığında kendi kurulum programını çalıştırdığınızda (Bu versiyon için MPL33100.exe) tüm sorulara olumlu yanıt verdiğinizde kendisini:

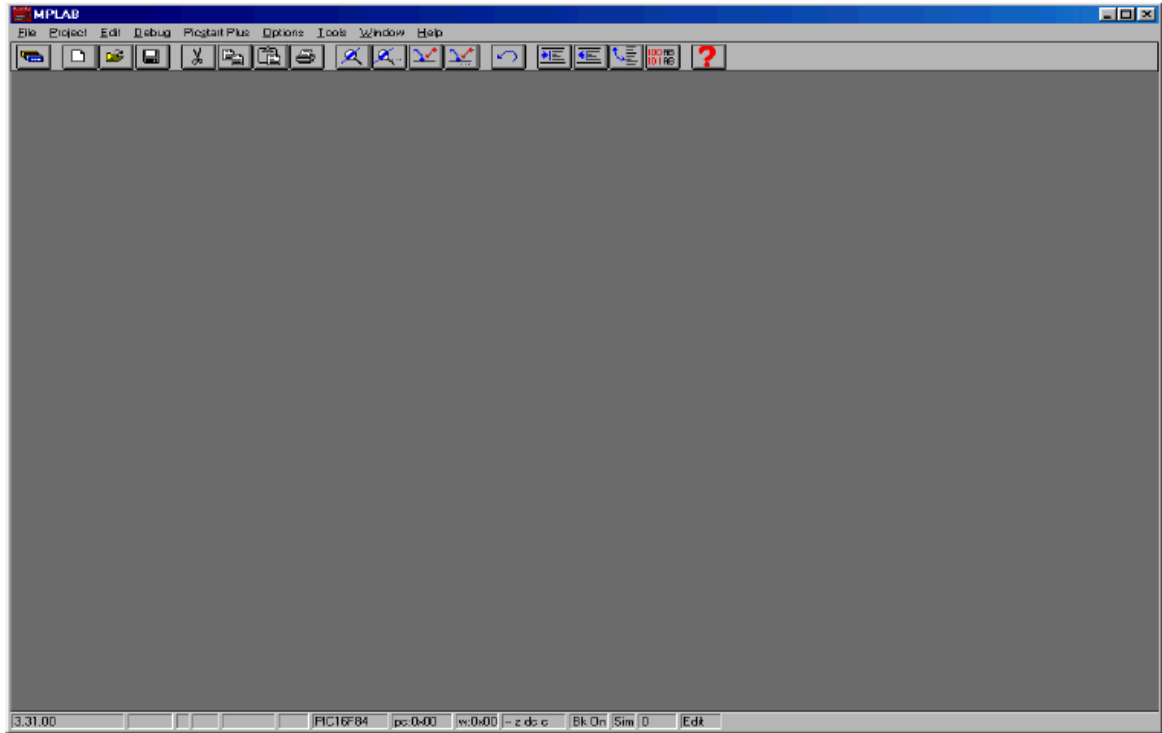
C:\PROGRAM FILES\MPLAB adında bir klasör içerisine kurar ve başlat içerisinde de kendisine Microchip MPLAB adında yer açar. Programı çalıştırmak için:

Başlat/Programlar/Microchip Mplab/MPLAB seçilir.



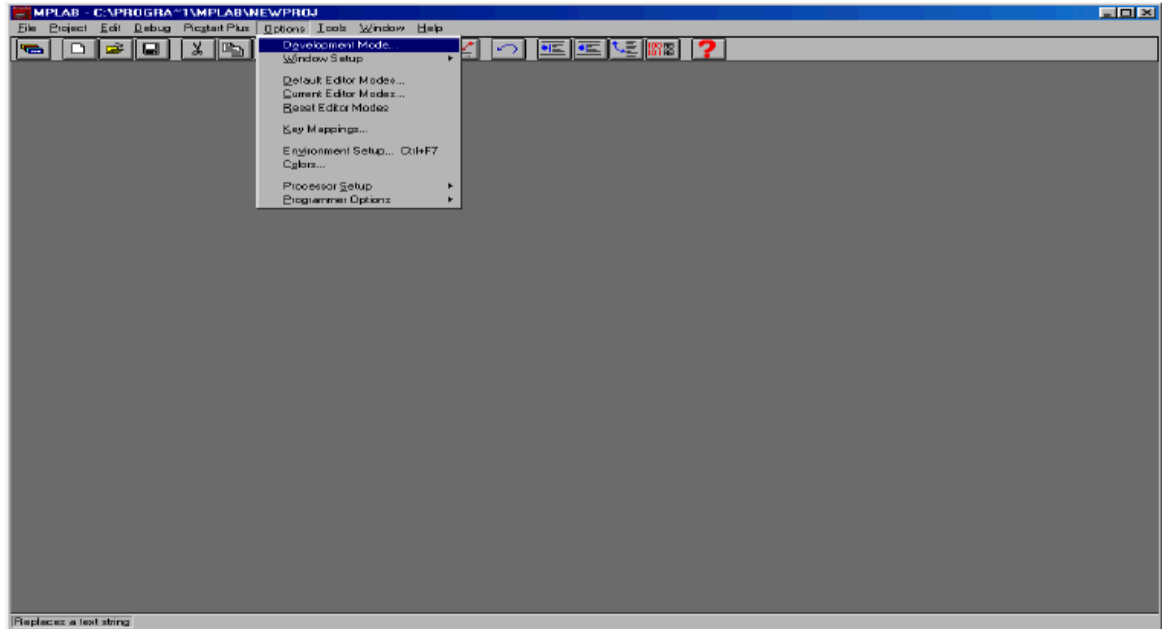
Şekil.1:MPLAB çalıştırılması

Bu seçim yapıldığında aşağıdaki ekran karşımıza gelir.



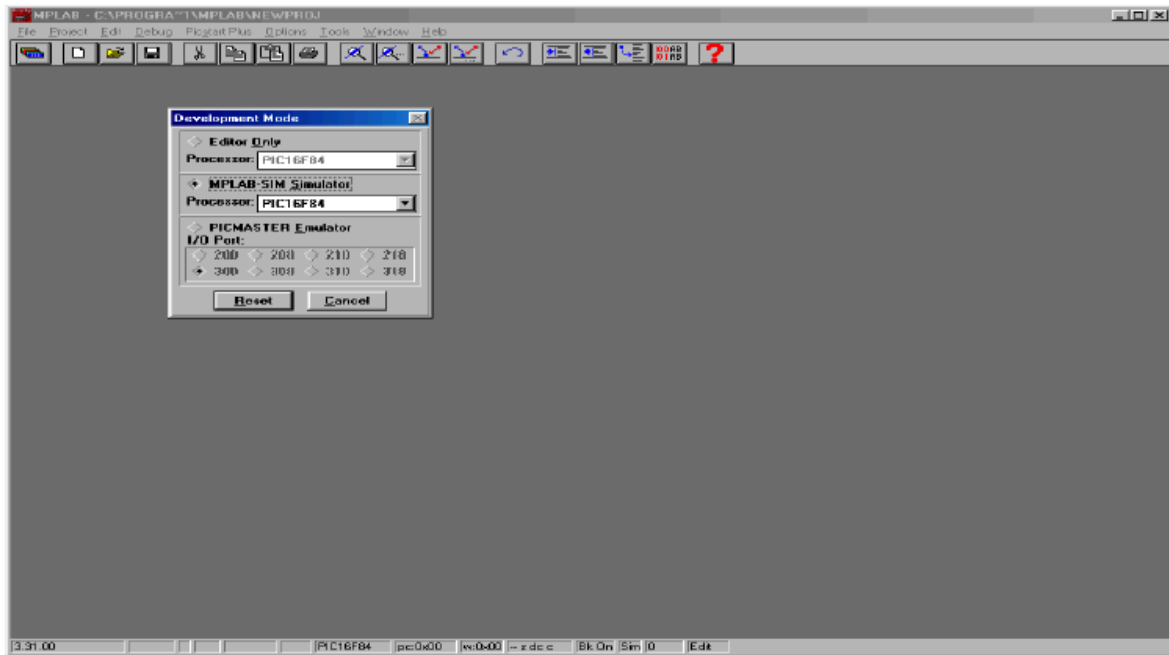
Şekil.2:MPLAB açılış sayfası

Programı ilk kurduğunuzda yapılması gereken 2 adet ayar vardır. Bu ayarlar kullanacağınız işlemciyi tanıtmak ki; biz pic16f84 tanıtacağız. Bunun için; Options/Development Mode seçilir.



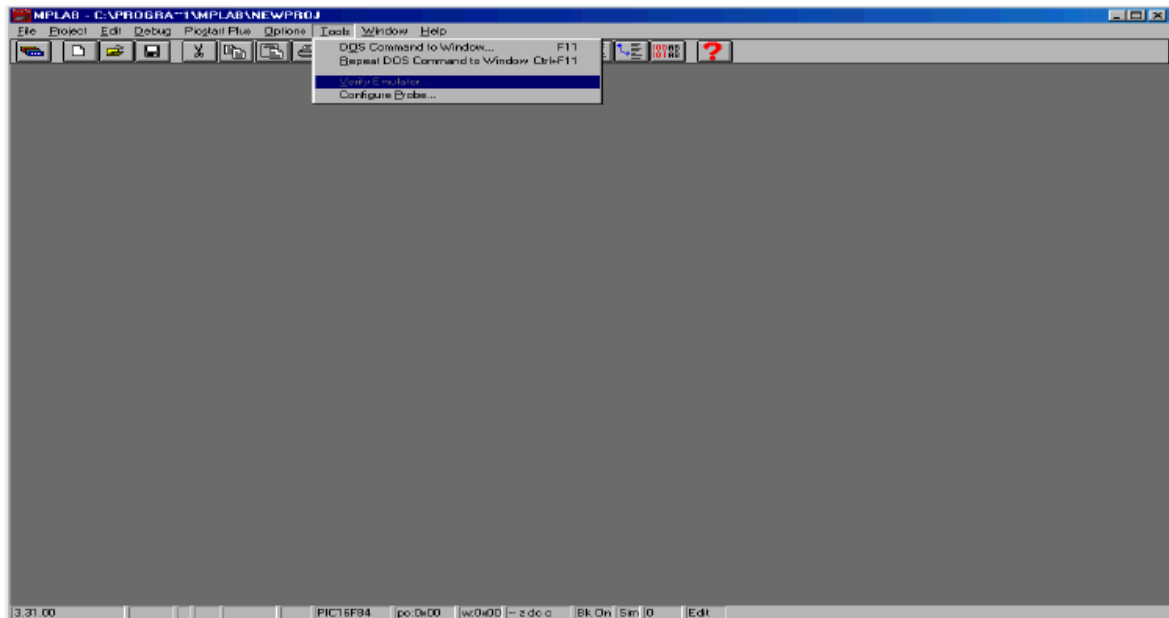
Şekil.3:MPLAB'ın ilk çalıştırılması

Gelen ekrandan kullanacağınız işlemciyi belirleyiniz (Bizim örneğimizde Pic16f84)



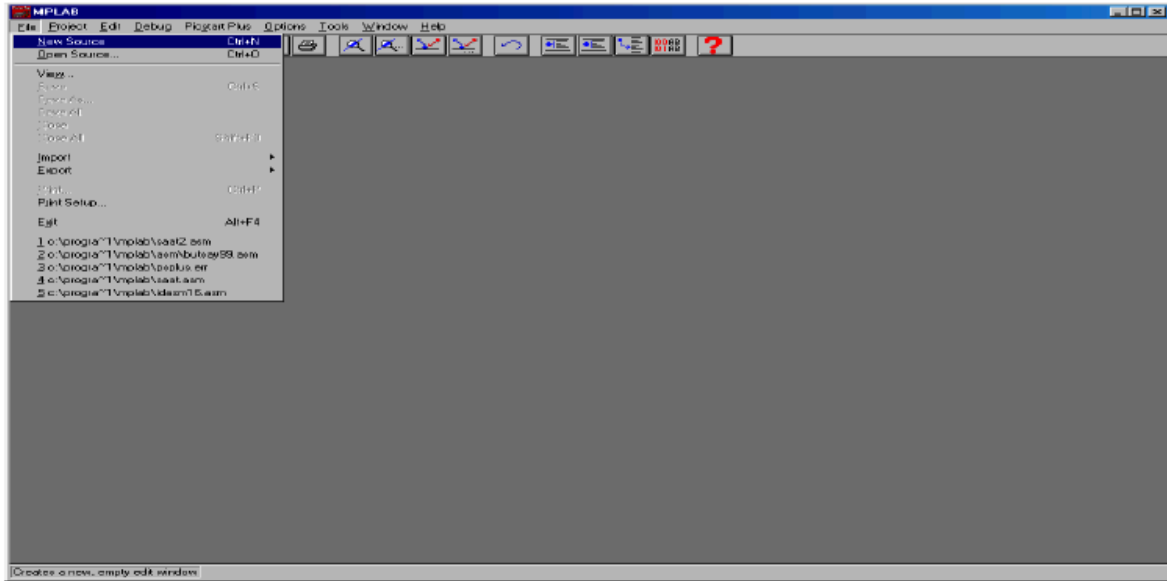
Şekil.4:MPLAB'ın ilk çalıştırılması

Yapılması gereken diğer ayar ise Tools/Verify Emulator işlemidir ki; burada bu seçildikten sonra gelen mesajlara olumlu (YES/OK) cevaplar vererek arada gelecek olan üçlü menüden SIMULATOR seçeneği seçildikten sonra, yine gelen mesajlara olumlu cevaplar verilerek işlem tamamlanır. Bu işlemi yapmak bize yazacağımız programı simulasyon modunda çalıştırma imkanı verir.



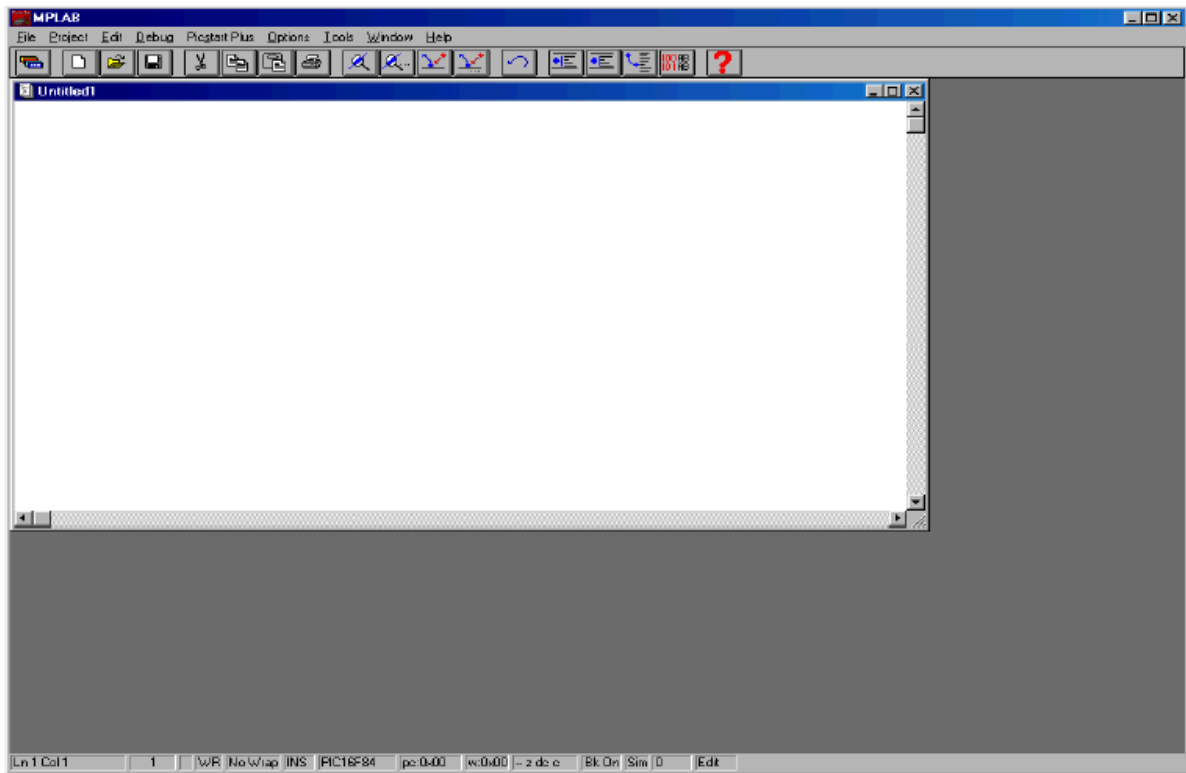
Şekil.5:MPLAB'ın ilk çalıştırılması

Yeni programınızı yazabilmek için boş bir sayfa açmalısınız. Bunun için;
+File/New Source seçiniz.



Şekil.6:MPLAB'da yeni sayfa açılması

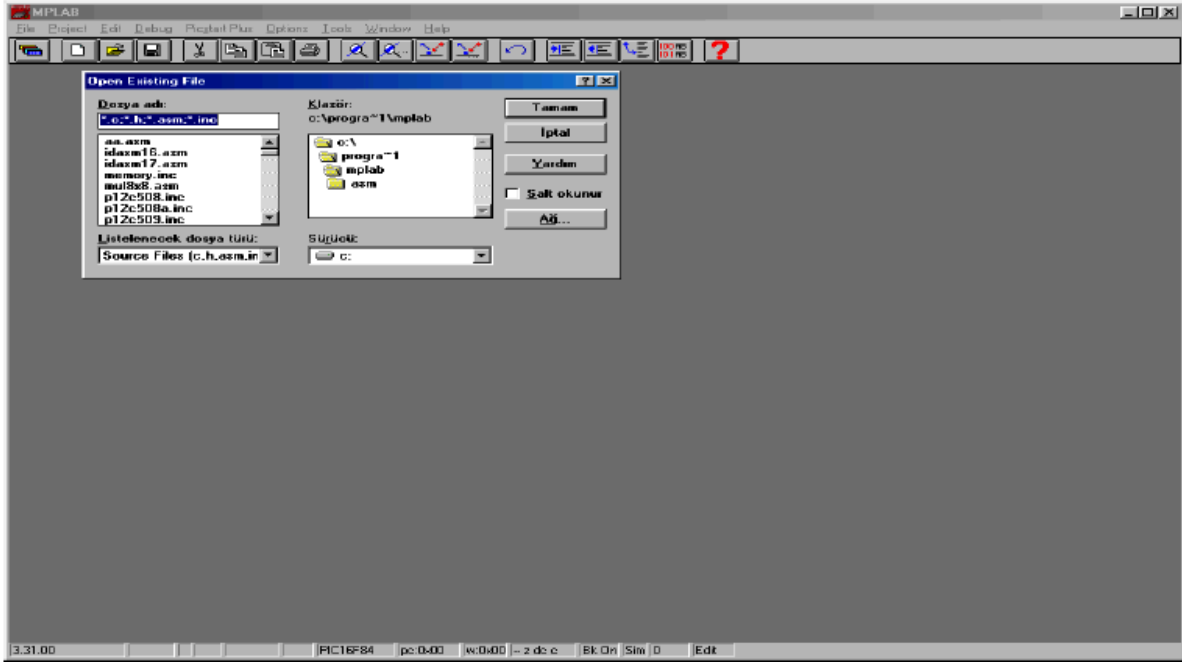
Bu işlemten sonra aşağıdaki boş sayfa karşınıza gelir ve programınızı buraya yazabilirsiniz.



Şekil.7:MPLAB'da yeni sayfa açılması

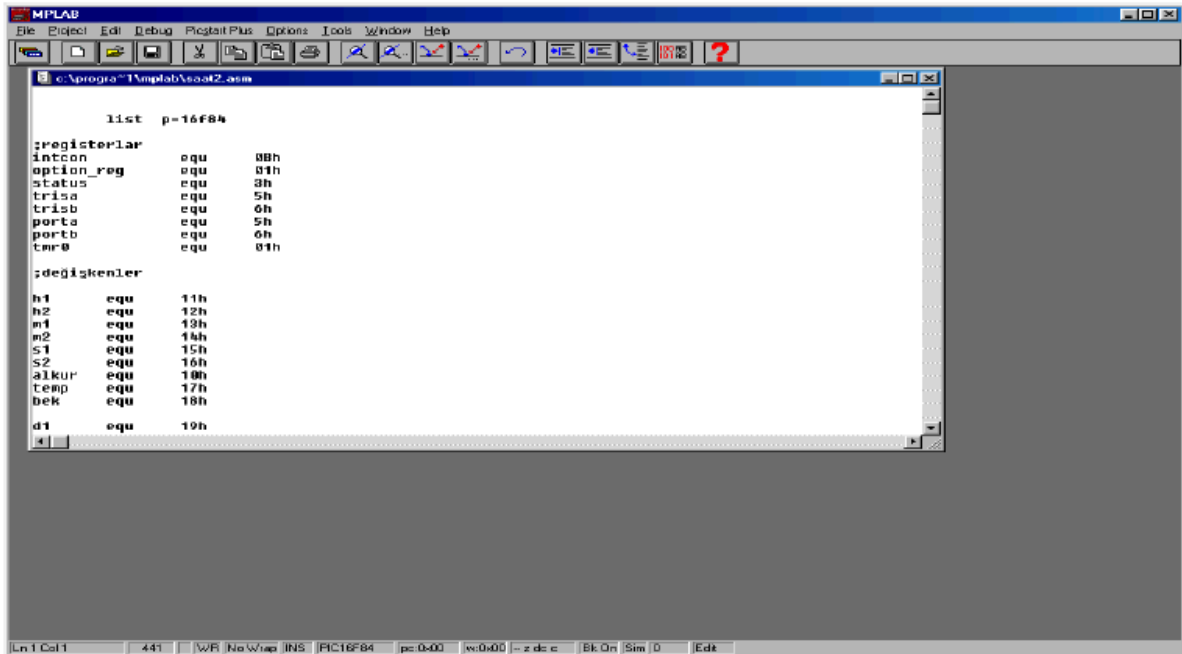
Eğer daha önceden yazıp kaydettiğiniz bir dosyayı açacaksanız;

File/Open Source seçiniz ve gelen ekrandan klasör ve dosya isimlerini seçerek dosyanızı ekrana getirebilirsiniz.



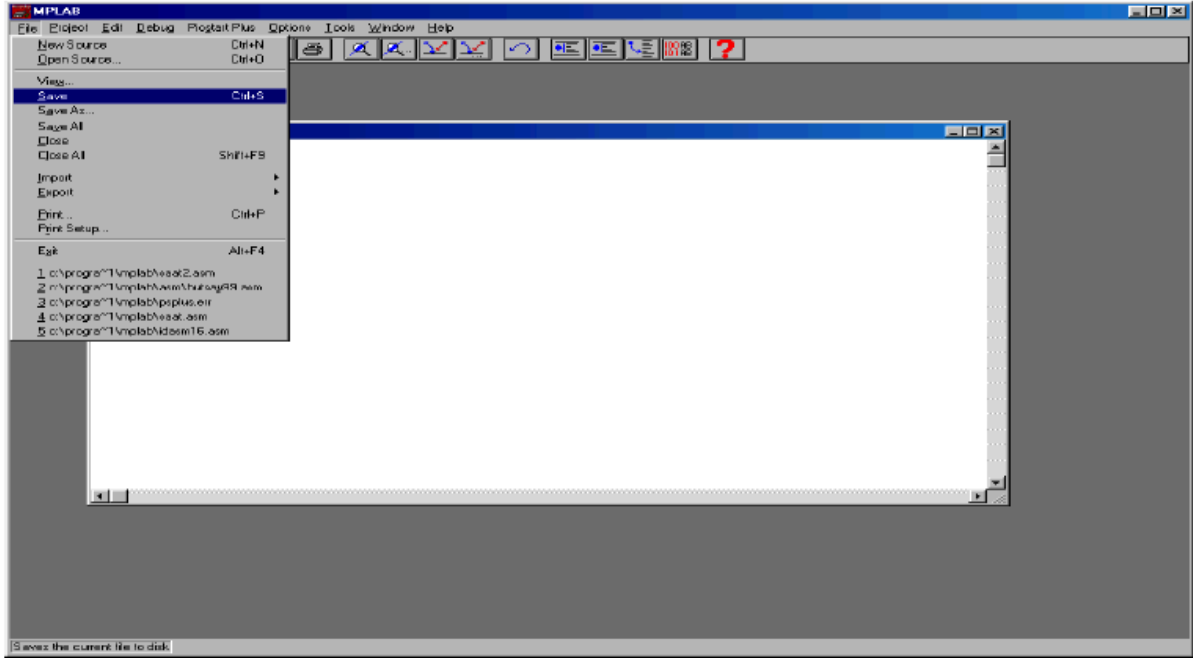
Şekil.8:MPLAB'da önceden yazılmış programın açılması

Eski dosyanızı çağırdığınızda veya yeni yazdığınız programın yazımını tamamladığınızda aşağıdaki şekilde bir görüntü oluşacaktır.



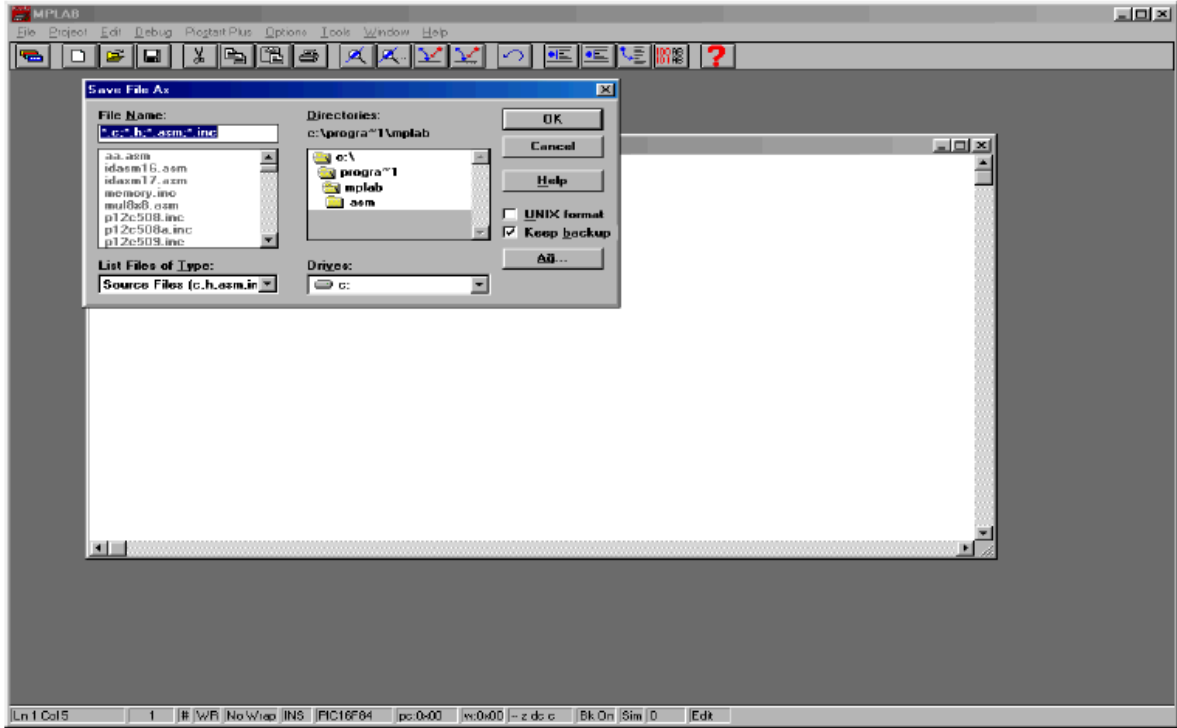
Şekil.9:MPLAB'da program yazımı

Ancak programı yeni yazdıysanız başlık satırında isim yerine UNTITLED yazısı görünecektir. Programınızı yeni yazdıysanız isim vererek kaydetmelisiniz. Kayıt işlemi için File/Save seçeneğini seçiniz.



Şekil.10:MPLAB’da yazılan programın kaydedilmesi

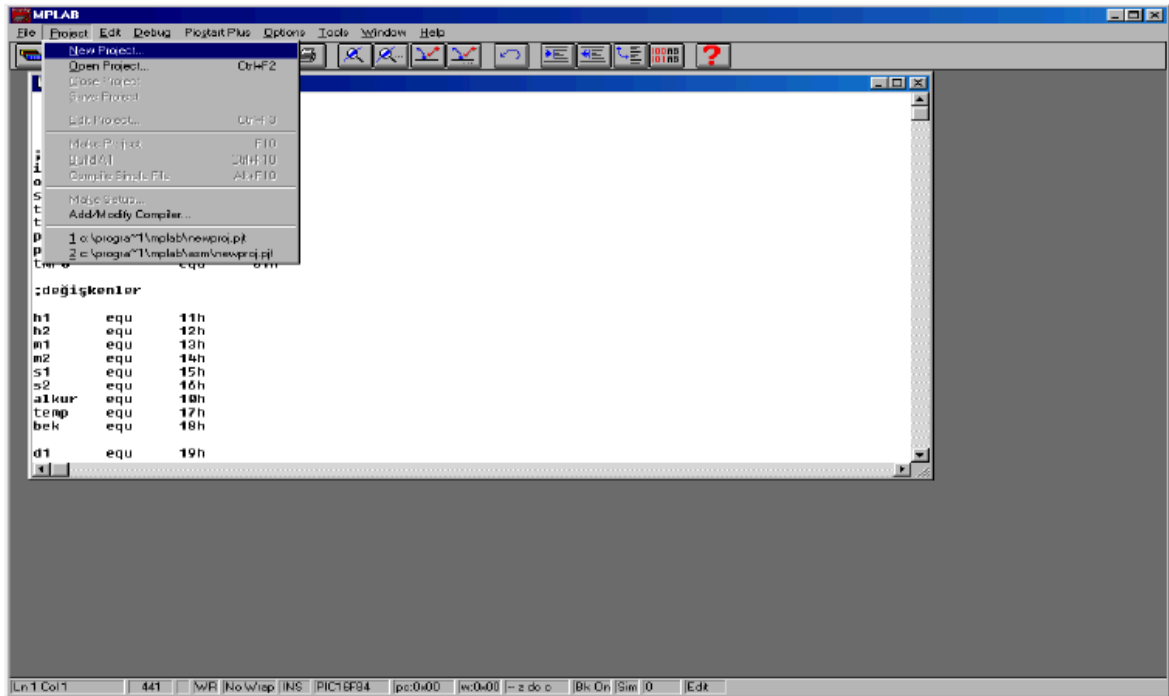
Gelen ekranda seçili klasör, sizin istediğiniz klasör değilse, Windows işletim sisteminin kullanım metoduyla, seçili klasörü istediğiniz şekilde değiştirebilirsiniz. Ancak daha sonra seçtiğiniz klasörün isminin lazım olacağını unutmayınız. File Name kısmına ise istediğiniz ismi veriniz; ancak uzantı ismi olarak asm vermeyi unutmayınız. Örnek SAAT.ASM gibi...



Şekil.11:MPLAB’da yazılan programın kaydedilmesi

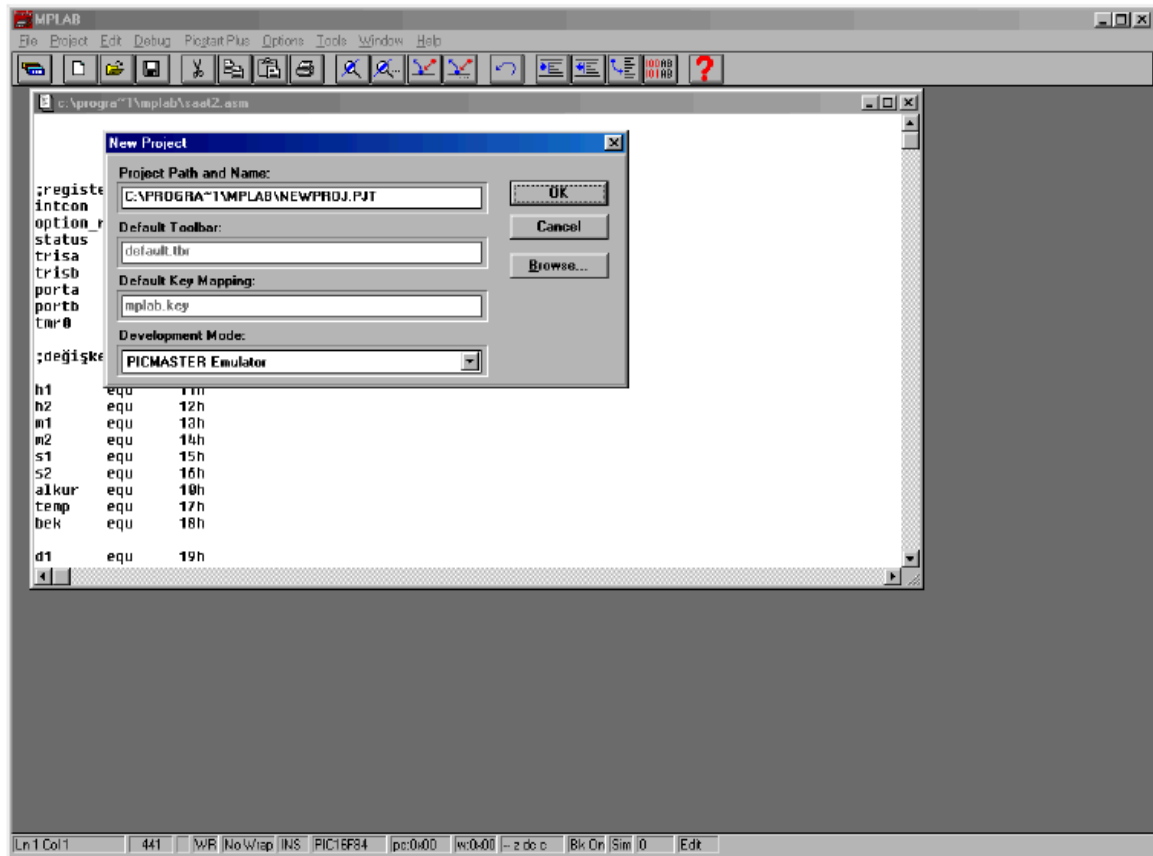
Dosya oluşturduktan sonra sıra proje oluşturma aşamasına gelecektir. Proje oluşturma aşamasında ise ilk iş;

Project/New Project seçeneğini seçmektir.



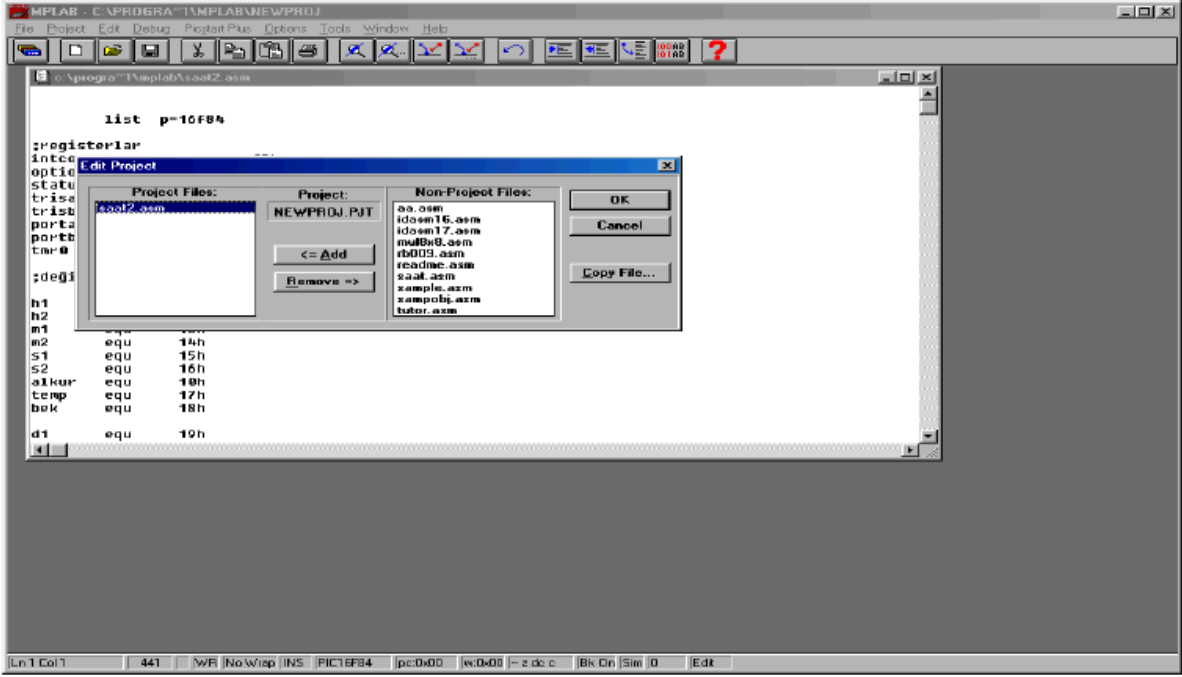
Şekil.12:MPLAB’da proje oluşturulması

Bu seçim yapılırken proje ismi vermek için aşağıdaki ekran gelir. Burada NEWPROJ.PJT ismi kendiliğinden verilir. İstenilirse proje ismi değiştirilir. Bu değiştirme yapılırken ileride sorun yaşamamak için, proje adına PJT uzantısının yazılması ve klasör adının değiştirilmemesine özen gösterilmelidir.



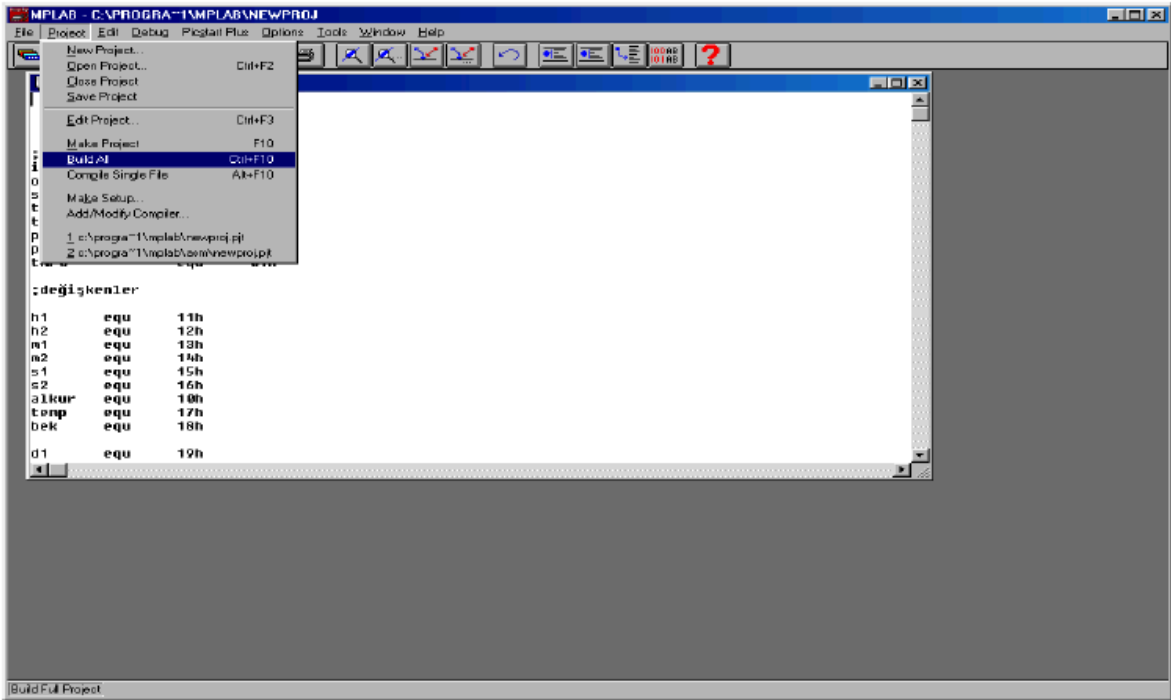
Şekil.13:MPLAB'da proje oluşturulması

Bu aşamadan sonra Edit Project ekranı gelir. Burada iki ayrı bölüm vardır. Project Files ve Non Project Files alanları. Burada Project Files alanına projede olmasını istediğimiz dosyanın adını diğer taraftan çift tıklayarak veya seçip add seçeneğini tıklayarak aktarırız. Bu işlemden sonra tamam (Ok) diyerek işlemi tamamlamış oluruz. Proje içerisindeki dosya adı ile ilgili bir sorun olursa Project/Edit Project seçeneği ile bu bölüme tekrar gelebiliriz.



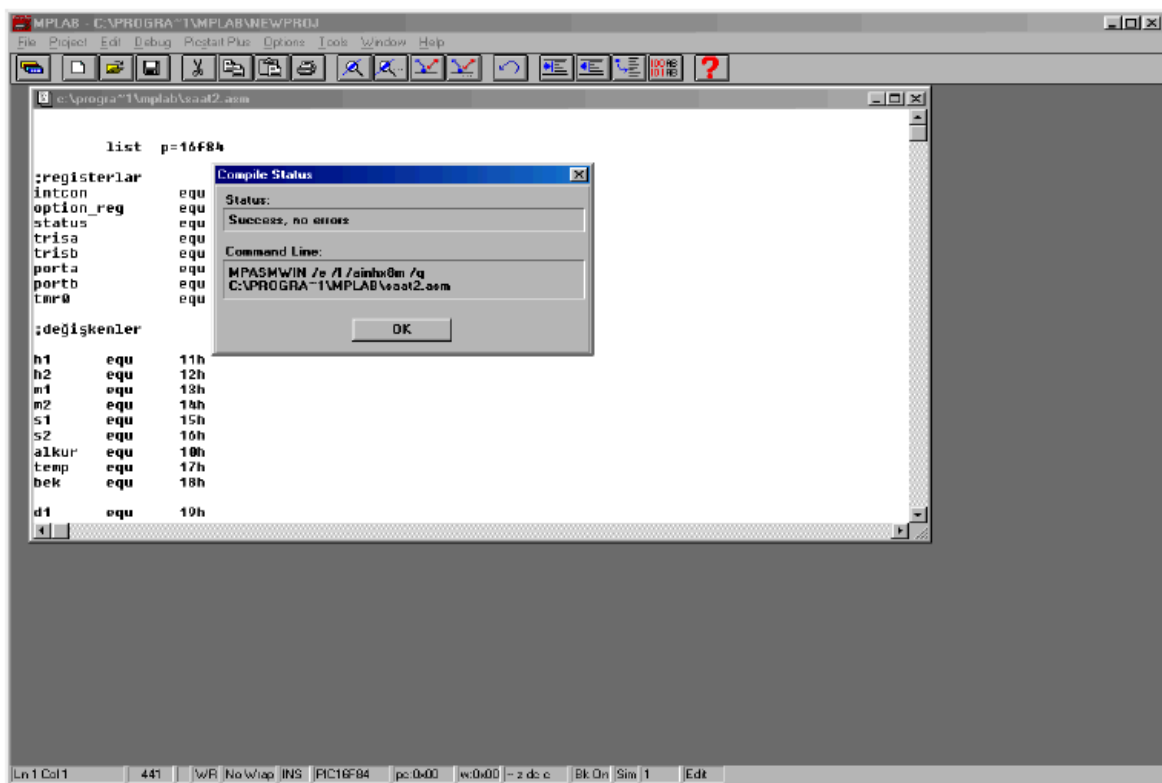
Şekil.14:MPLAB’da proje oluşturulması

Şimdi sıra yazdığımız programın derlenmesine gelmiştir. Derleme esnasında hatalar belirlenip bize mesaj olarak verilir. Ayrıca uzantısı “hex” olan bir dosya ile de yazılı programın çalışır kodları üretilir. Bu işlem için;
Project/Build All seçilir.



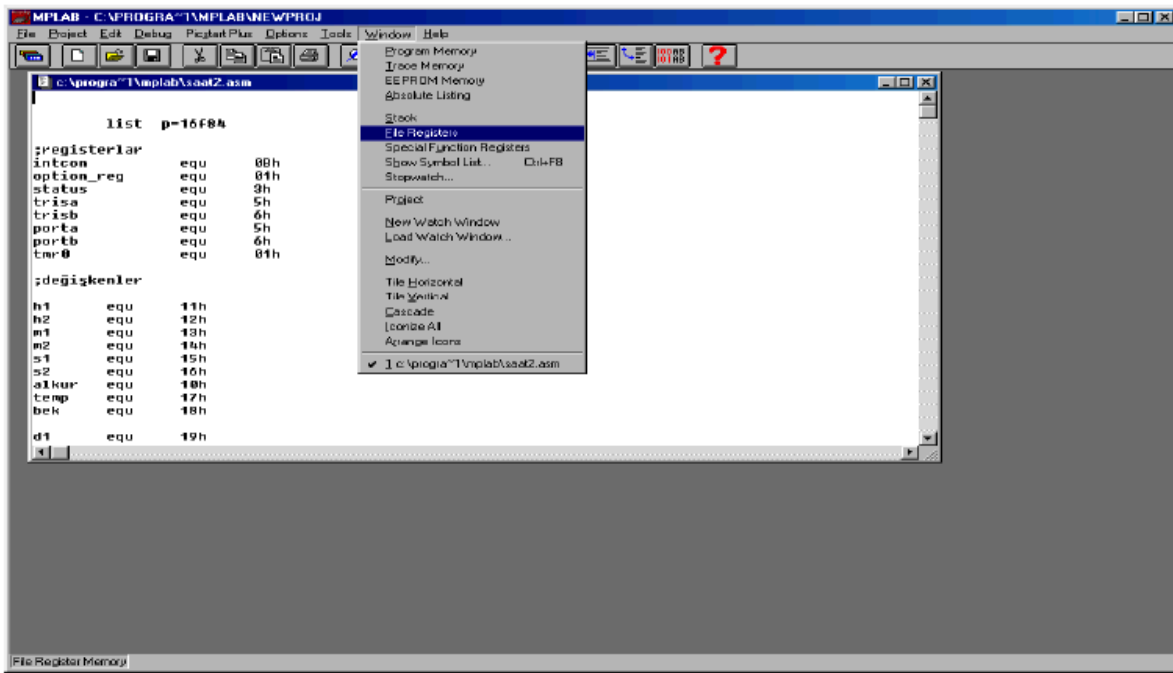
Şekil.15:MPLAB’da oluşturulan projenin derlenmesi

Bu seçimden sonra ekranda hareketli band olan bir pencere gelir ve derlemenin yapıldığını gösterir. İşlem bittiğinde aşağıdakine benzer bir pencere gelir. Eğer şekildeki gibi Success No Error mesajı varsa programda yazım hatası yok demektir ve program denemeye hazır demektir. Eğer There are Errors şeklinde bir mesaj gelirse hatalar uzantısı “err” olan bir dosya da kayıtlıdır. Bu dosyayı açarak hataların nerelerden kaynaklandığına bakar ve düzeltiriz. Bu hatalar dosya adı, satır numarası ve hata cinsi şeklinde satır satır belirtilir. Burada satır numarasının kaç olduğunu ekranın altındaki durum satırından takip ederek hataları düzeltme yoluna gidilir.



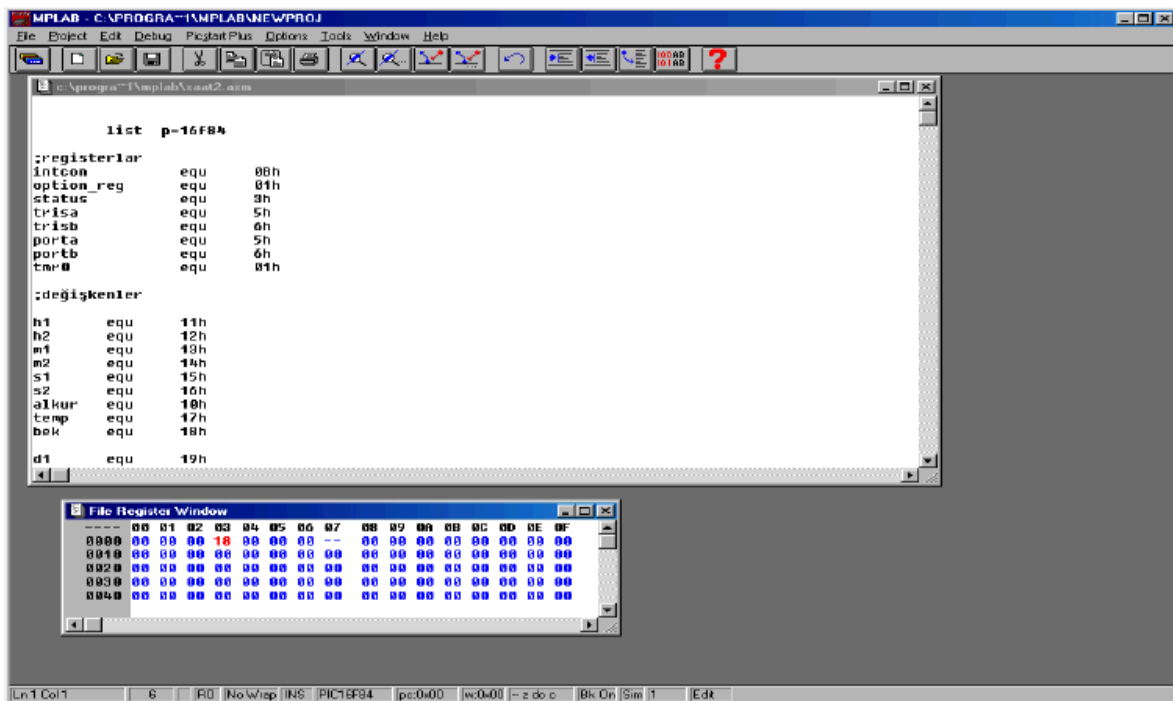
Şekil.16:MPLAB’da oluşturulan projenin derlenmesi

Programda hata yoksa artık sıra deneme işlemine gelmiştir. Deneme işlemi için; programda buton işlemi yoksa iki adım yeterlidir. Bunlardan birincisi; Windows/File Registers seçeneğini seçmektir.



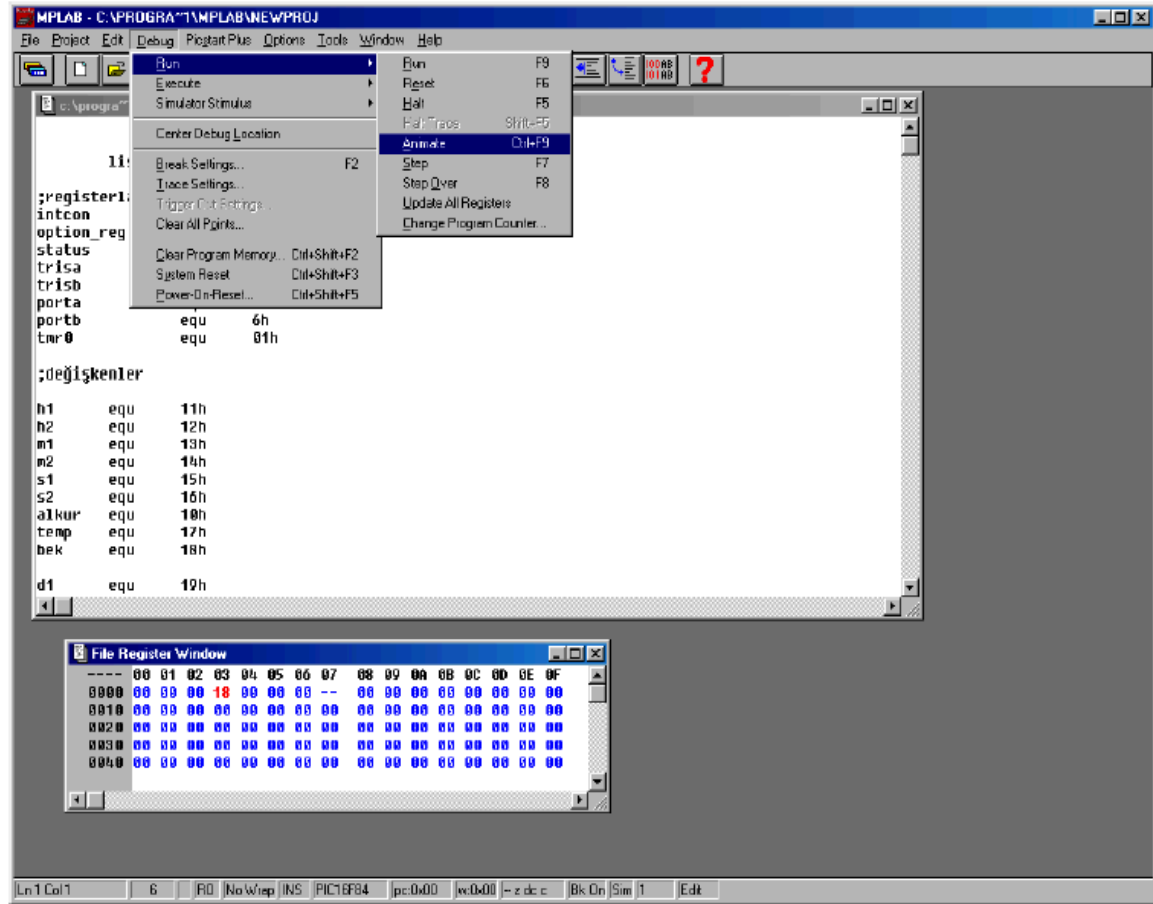
Şekil.17:MPLAB’da oluşturulan projenin simülasyon modunda çalıştırılması

Ekrana yeni gelen pencerede RAM adresleri ve içerikleri verilmiştir. Burada örnek olarak 0000 numaralı satırdaki 05 adresi A portu, 06 adresi B portu dur. Programı çalıştırdığımızda buradaki değişimlere bakarak A ve B portundaki bilgi değişimlerine bakarak programın çalışma şekli hakkında fikir yürütebiliriz.



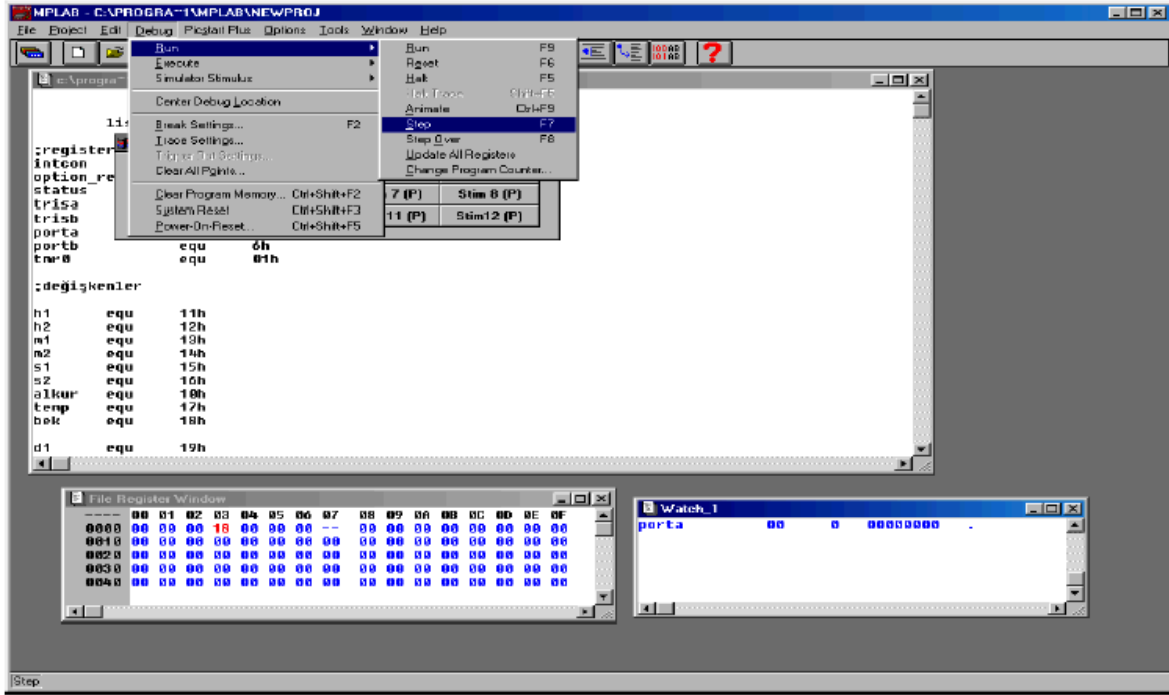
Programı çalıştırmak için;

Debug/Run/Animate seçeneği seçilir ve ekrandaki değişimler izlenerek programın doğruluğu hakkında fikir yürütülür. Burada animate modu programın yavaş bir hızda çalıştırılması işlemidir. Yani bilgisayarınız programı rahat inceleyebilmeniz için normalin 5000-10000 katı bir yavaşlıkta programı sürekli moda çalıştırır.



Şekil.19:MPLAB’da oluşturulan projenin simülasyon modunda çalıştırılması

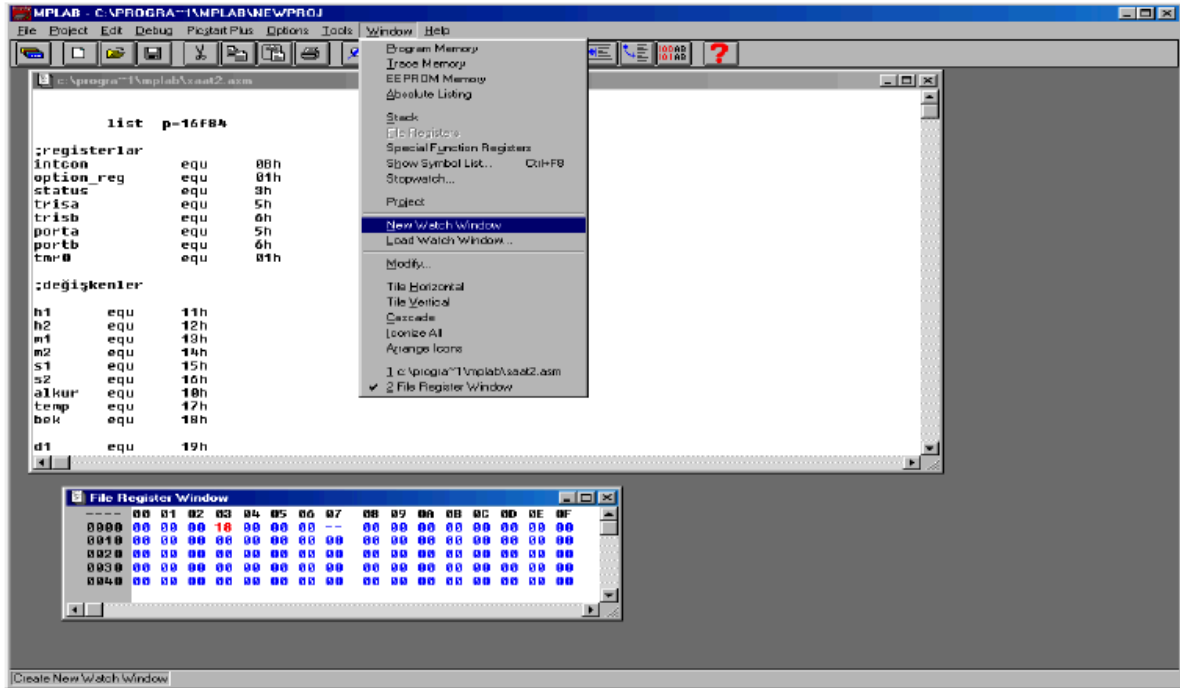
Eğer program adım adım çalıştırılmak isteniyorsa, Debug/Run/Step seçilir veya F7 tuşuna her basışta 1 komut çalışması sağlanır.



Şekil.20:MPLAB'da oluşturulan projenin simülasyon modunda çalıştırılması

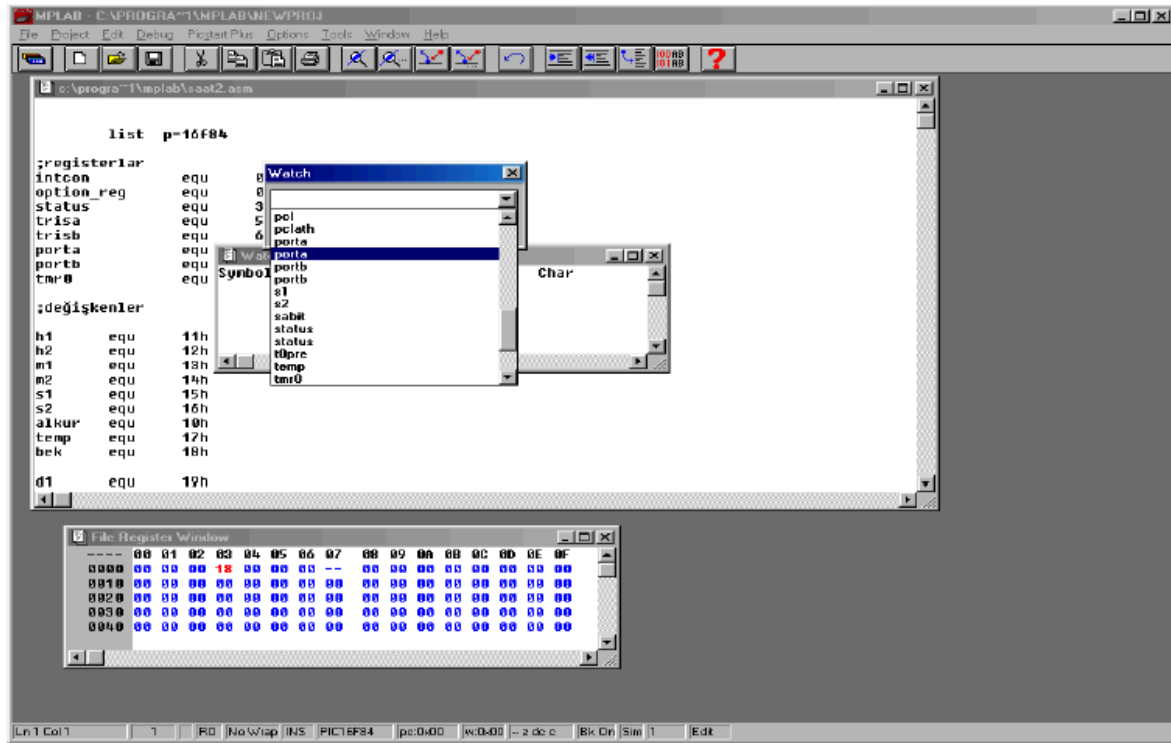
Eğer programın çalışması esnasında sadece bir iki yerdeki değişim izlenmek isteniyorsa

Windows/New Watch Window seçilir.



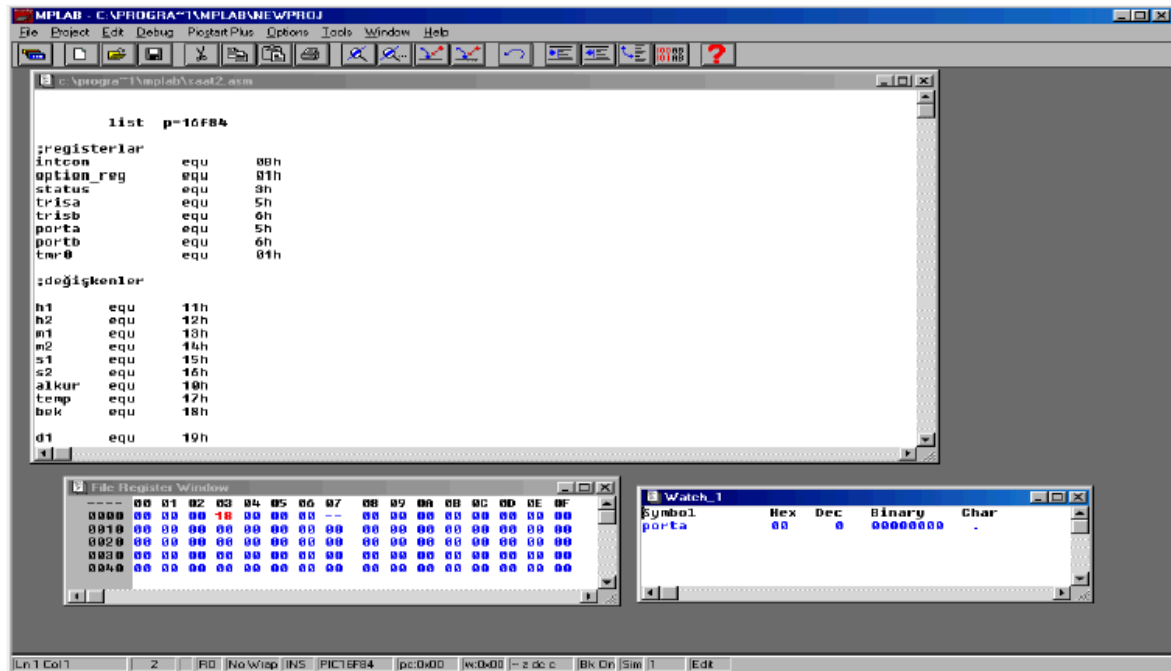
Şekil.21:MPLAB'da oluşturulan projenin simülasyon modunda çalıştırılması

Çıkan ekranda istenilen register seçilir. Örnekte PortA seçilmektedir.



Şekil.22:MPLAB’da oluşturulan projenin simülasyon modunda çalıştırılması

Seçim tamamlandığında Watch_1 şeklinde gelen ekranda A portundaki sayı ikilik, onluk ve onaltılık sistemde gösterilir.

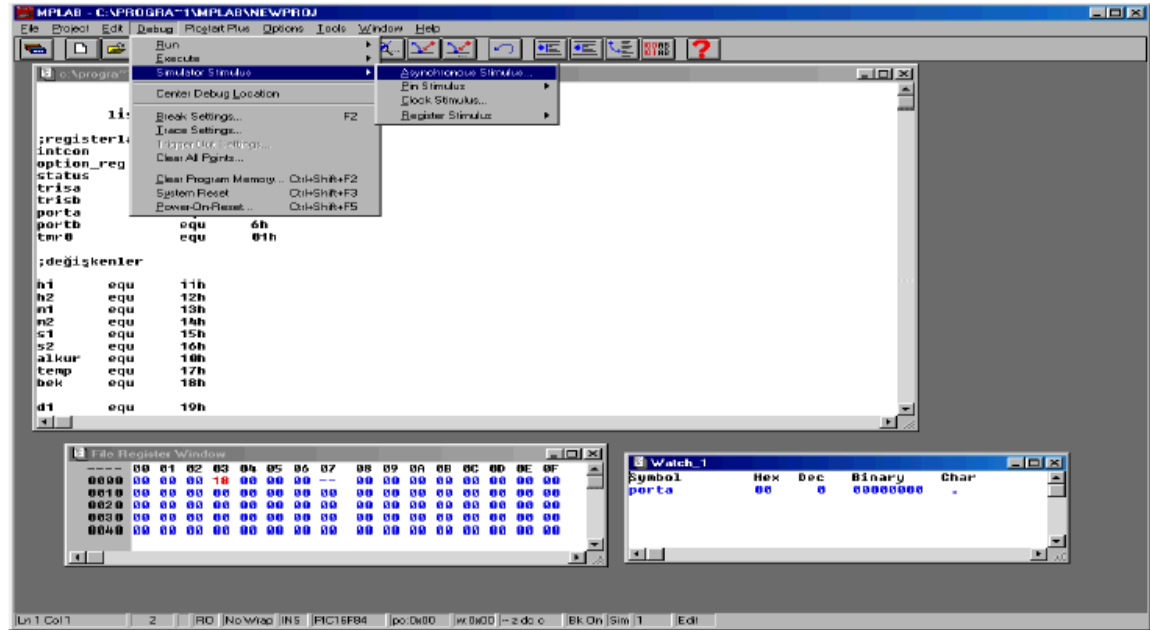


Şekil.23:MPLAB’da oluşturulan projenin simülasyon modunda çalıştırılması

Eğer sistemde dışarıdan buton veya anahtar bağlantısı ile bilgi girişi varsa ve bu bilgi girişi için programın vereceği tepki izlenmek isteniyorsa buton tarifi yapılması gerekir.

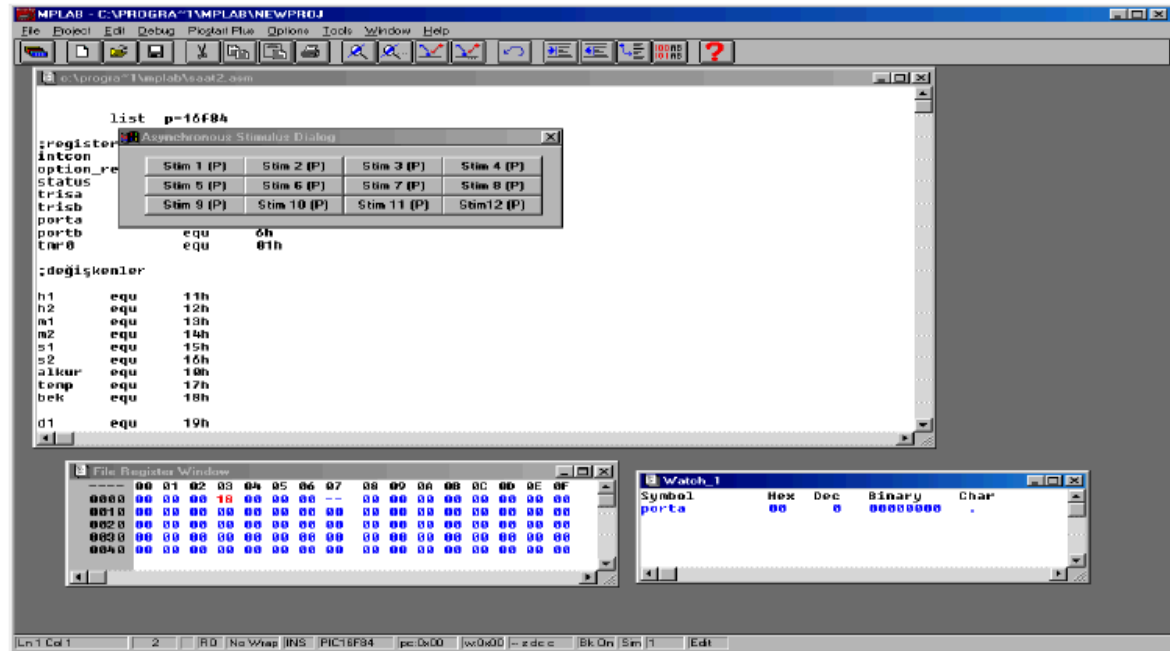
Bunun için;

Debug/Smilator Stimulus/Asynchronous Stimulus seçeneği seçilir.



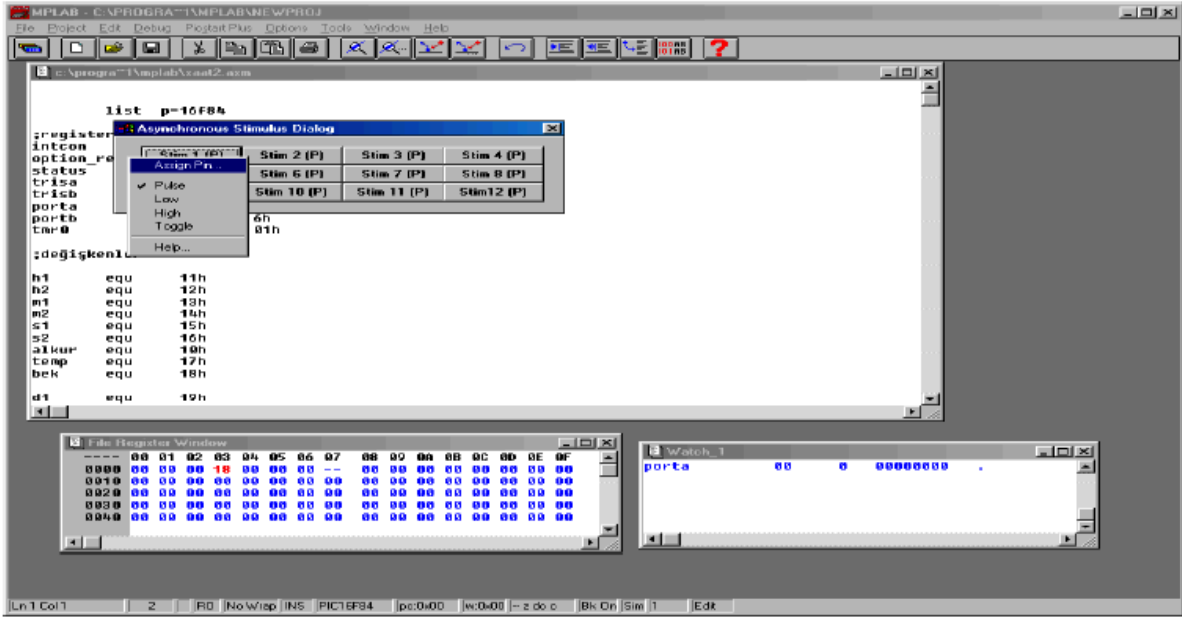
Şekil.24:MPLAB’da oluşturulan projenin simülasyon modunda çalıştırılması

Yeni gelen pencerede 12 adet buton mevcuttur ve bu butonların her birini işlemcinizin bir ucuna bağlı olarak gösterebilirsiniz.

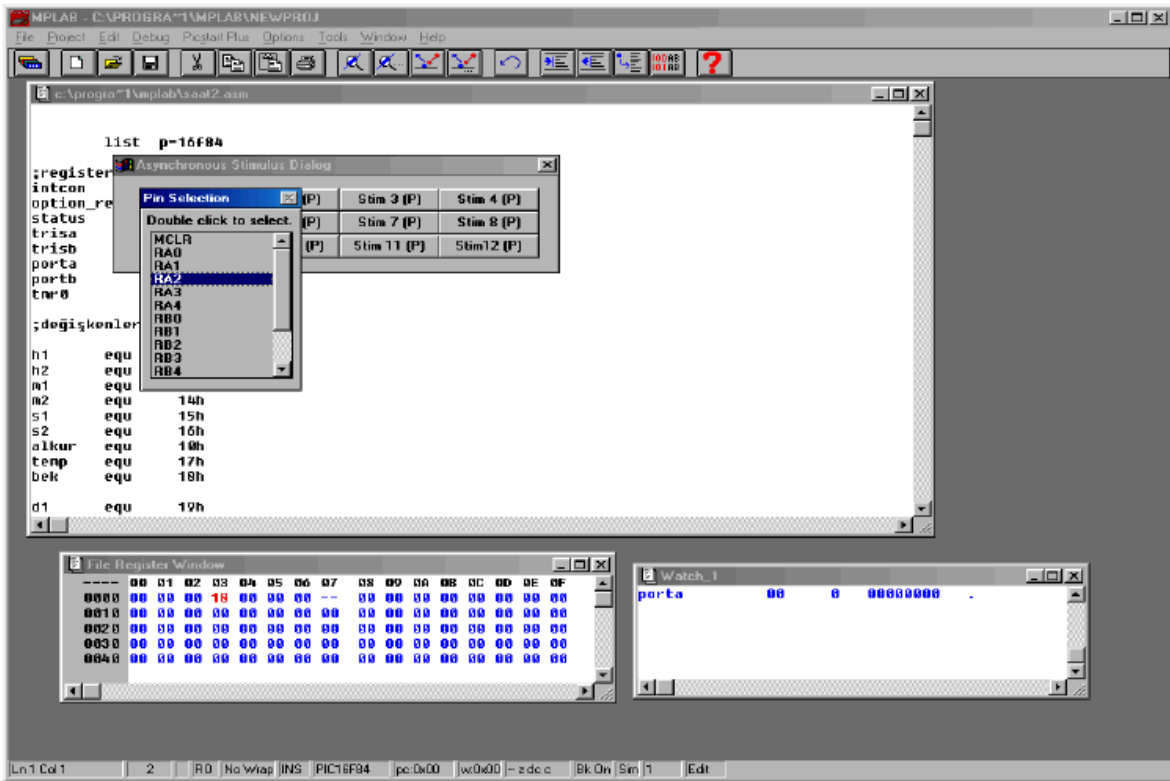


Şekil.25:MPLAB’da oluşturulan projenin simülasyon modunda çalıştırılması

Biz örnek olarak RA2 ucuna Toggle moda bir buton tarifi yapalım. Burada interrupt algılamalı denemeler haricinde genel olarak toggle mod kullanılır. Bu işlem için Stim1 butonuna sağ tuş ile tıklayalım:



Şekil.26:MPLAB’da oluşturulan projenin simulasyon modunda çalıştırılması
Gelen pencereden Assign Pin seçelim ve yine gelen pencerede RA2’yi çift tıklayalım.



Şekil.27:MPLAB’da oluşturulan projenin simulasyon modunda çalıştırılması

The screenshot shows the MPLAB IDE interface. The main window displays assembly code for a PIC16F84. A dialog box titled "Asynchronous Stimulus Dialog" is open, showing a table of stimulus values for various pins. The "Pulse" tab is selected, and the "Pulse" checkbox is checked. The "File Register Window" shows the values of various registers, and the "Watch Window" shows the value of the "porta" register.

Stim 2 (P)	Stim 3 (P)	Stim 4 (P)	Stim 6 (P)	Stim 7 (P)	Stim 8 (P)	Stim 10 (P)	Stim 11 (P)	Stim 12 (P)

```

register
intcon
option _rc
status
trisa
trisb
porta
portb
trio
;digkey1
b1 equ 11h
b2 equ 12h
b3 equ 13h
m1 equ 14h
m2 equ 15h
s1 equ 16h
s2 equ 17h
alku equ 18h
temp equ 19h
bck equ 1Ah
d1 equ 19h

```

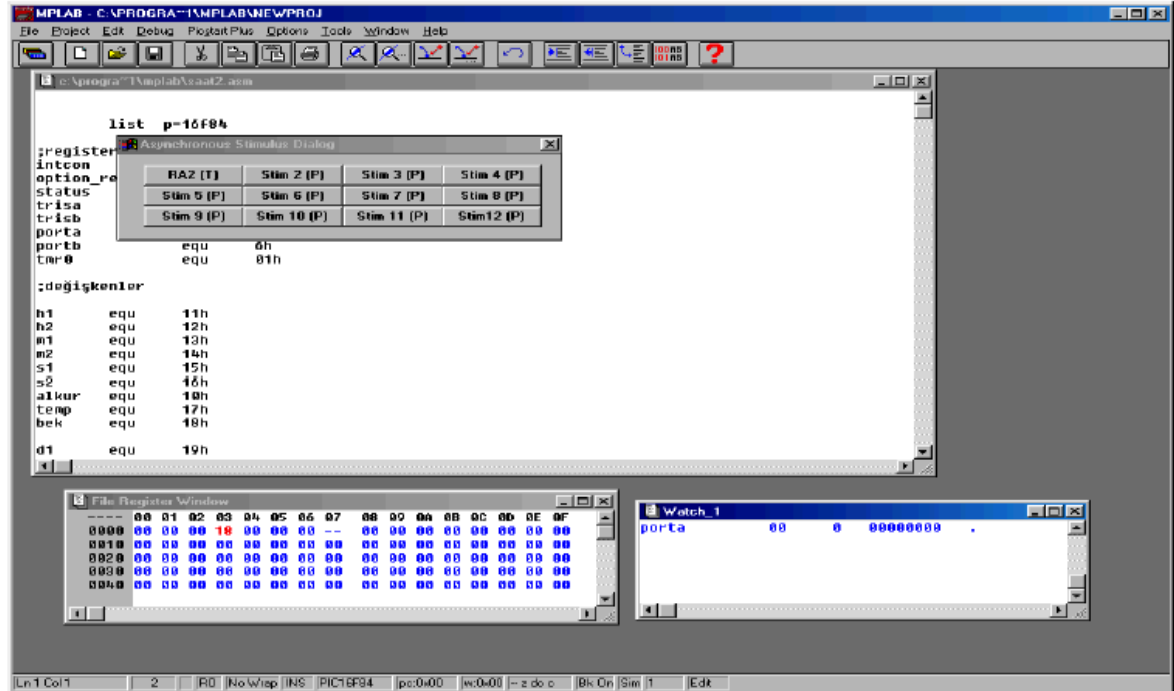
File Register Window

File	Register	Value
0000	00	00 00 00 00 00 00 00 00
0001	01	00 00 00 00 00 00 00 00
0002	02	00 00 00 00 00 00 00 00
0003	03	00 00 00 00 00 00 00 00
0004	04	00 00 00 00 00 00 00 00

Watch Window

File	Register	Value
0000	00	00 00 00 00 00 00 00 00
0001	01	00 00 00 00 00 00 00 00
0002	02	00 00 00 00 00 00 00 00
0003	03	00 00 00 00 00 00 00 00
0004	04	00 00 00 00 00 00 00 00

Artık işlem tamamlanmıştır. Başlangıçta üzerinde Stim1(P) yazan tuşta artık RA2(T) yazmaktadır. Bunun anlamı program animate edilirken bu tuşa her tıkladığımızda RA2 ucundaki bilgi durum değiştirecektir, yani bir 1 olacak bir 0 olacaktır.

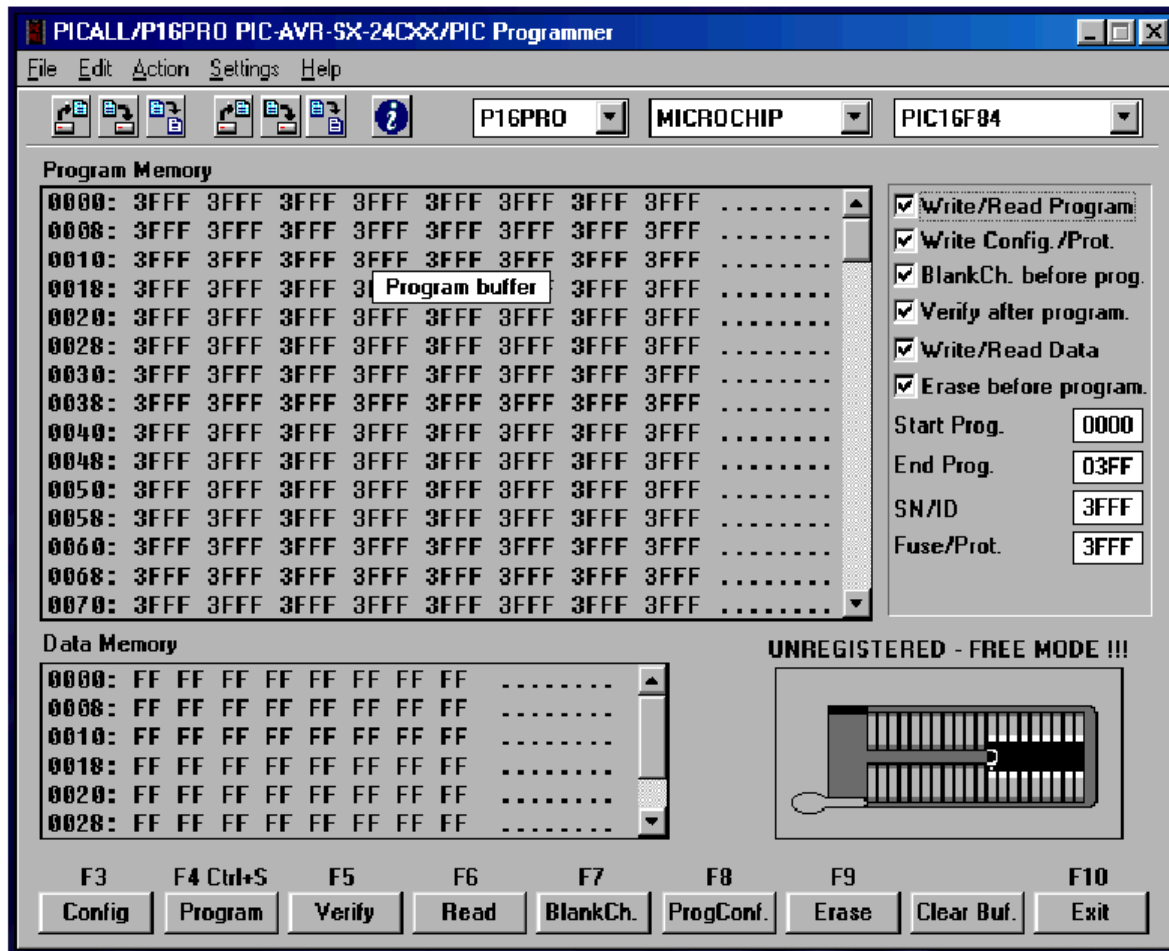


95

EK.2. PICALLW KULLANIMI

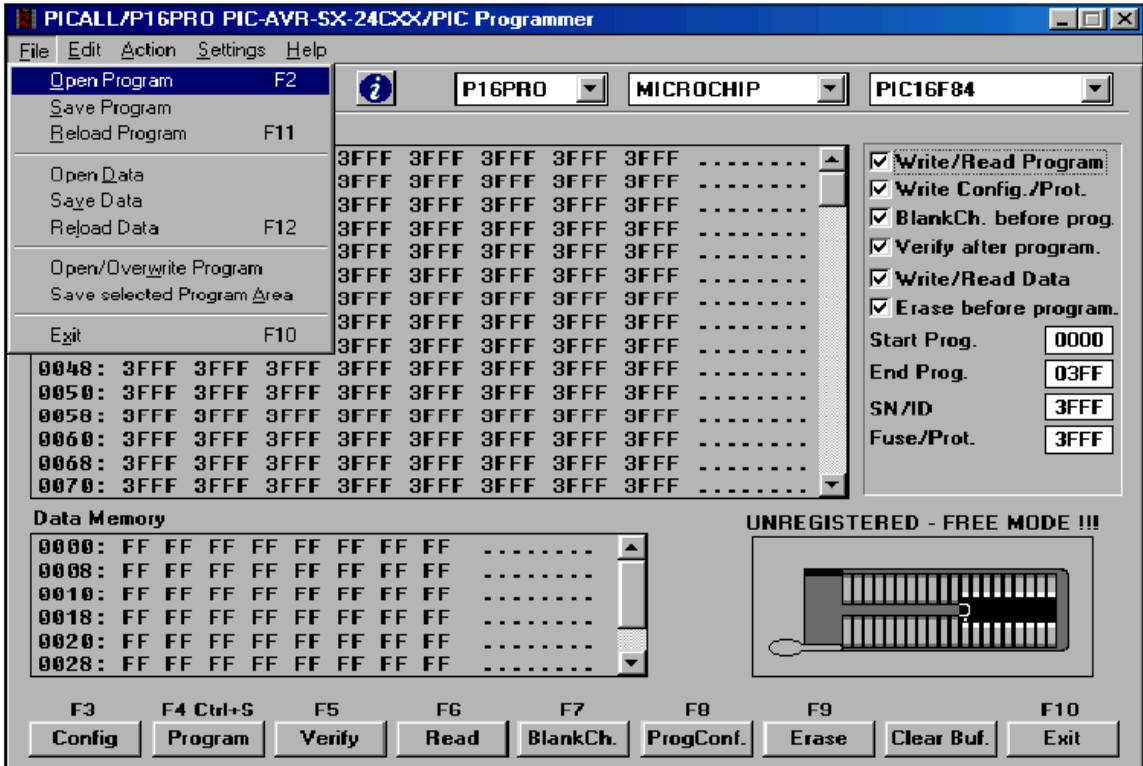
MPLAB'da deneyip hazır ettiğiniz programı mikrodeneleyicinize yüklemek için elinizde mevcut ise MPLAB ile uyumlu mikrochip ürünü bir yükleyici (Picstart gibi) kullanabilirsiniz. Bu mevcut değilse internetten de devresini indirebileceğiniz P16PRO devresini kurarak PICALLW programı ve bu set yardımıyla MPLAB'da hazırladığınız programı mikrodeneleyicinize yükleyebilirsiniz.

İlk iş picallw.exe programını çift tıklayarak çalıştırınız.



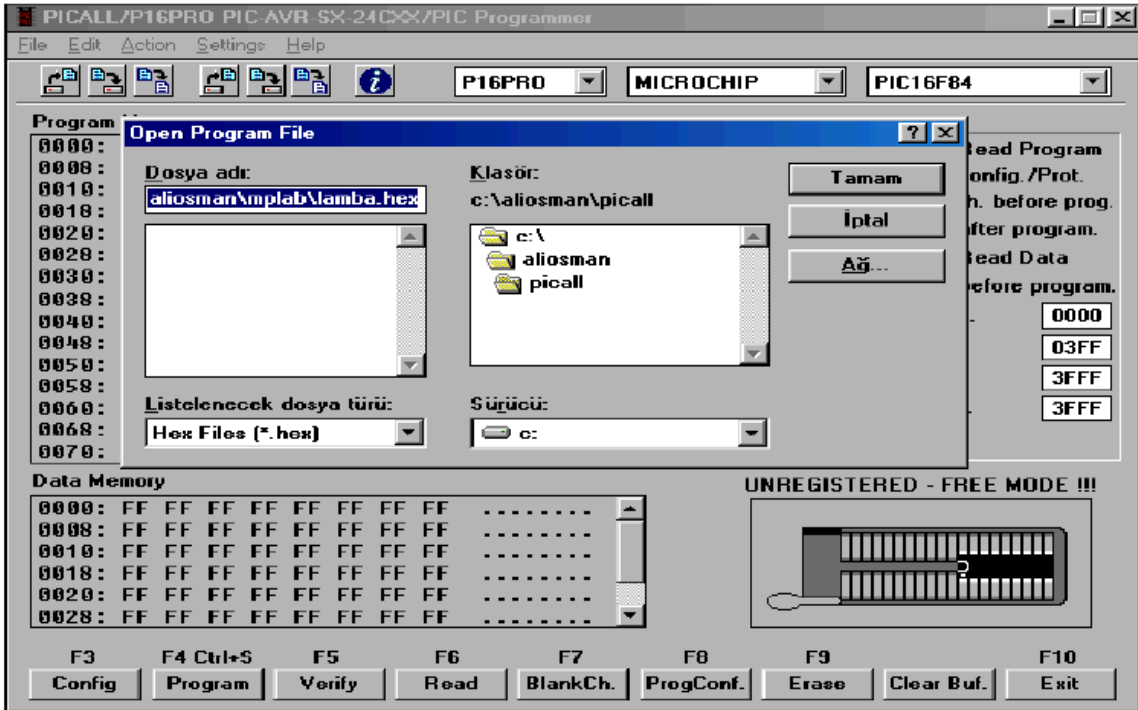
Şekil 1. PICALLW programı açılış ekranı

Sonra File/Open program seçeneğini kullanarak hazırladığınız programın uzantısı hex olan kaynak kodları dosyasını açınız.



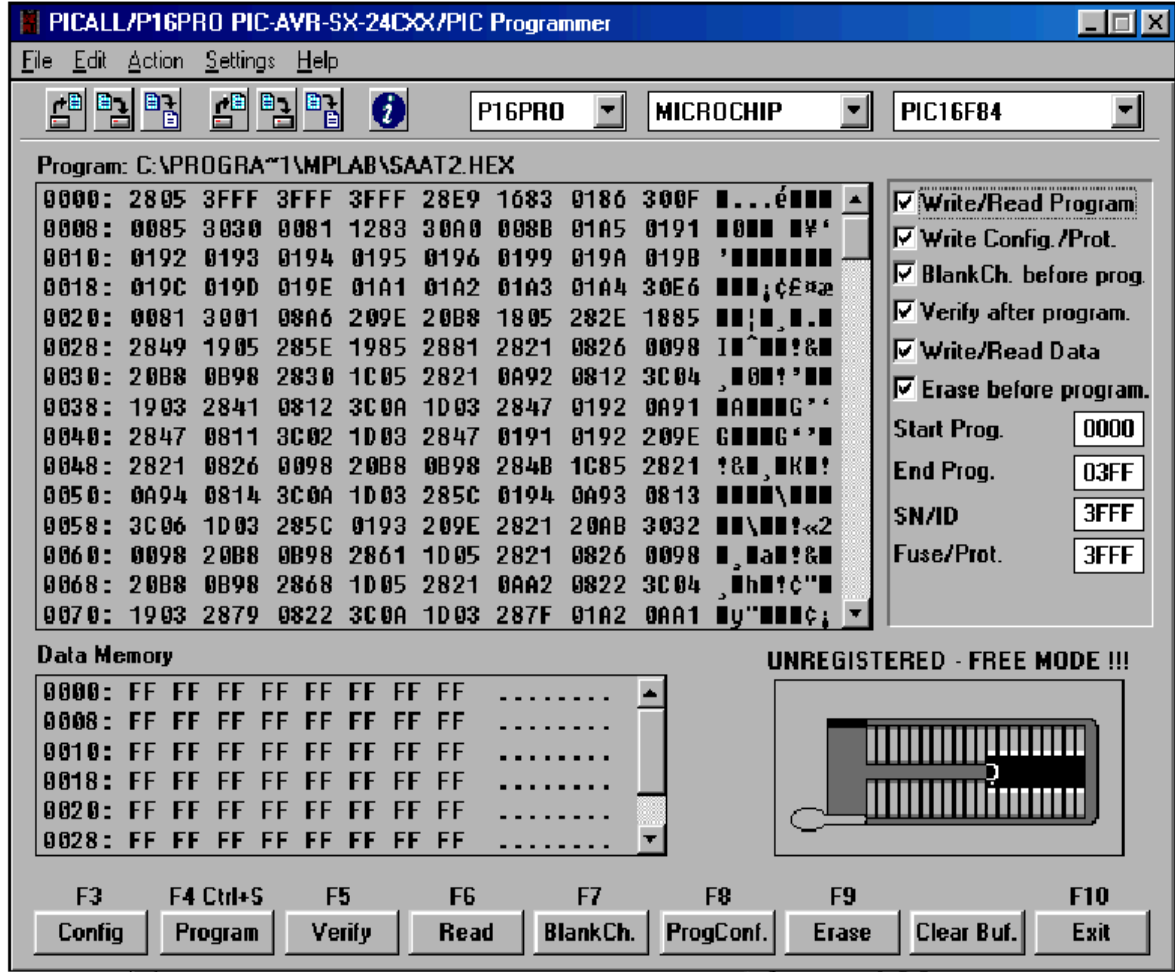
Şekil 2: Picallw programında dosya açılması

Open program seçildiğinde çıkan ekranda başka program adı görülebilir, kendi programınızı bulup seçiniz.



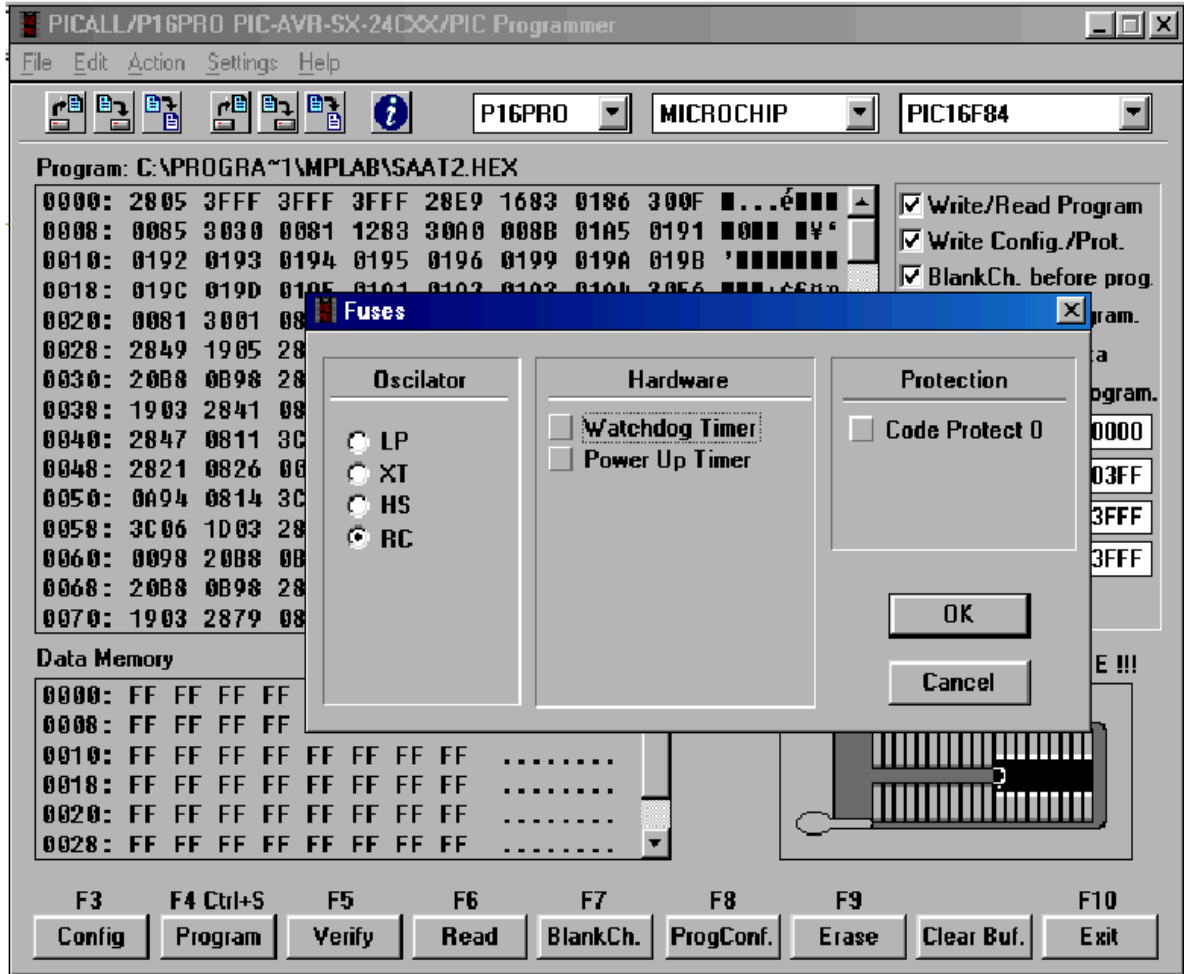
Şekil 3: Picallw programında dosya açılması

Dosyayı bulup Tamam seçeneğini seçtiğinizde ekran aşağıdakine benzer şekilde değişir. Görüntü program komutlarının durumuna göre farklılık gösterebilir.



Şekil 4: Picallw programında dosya açılması

Şimdi P16PRO setinizin bilgisayarla iletişimini kurması için setinizin data kablosunu bilgisayarınızın printer portuna takınız. Setinizin güç bağlantısını yapıp devreye güç uygulayınız. Set üzerindeki yeşil ışık yanıyor ise her şey yolundadır. Yüklemeye geçmeden önce son yapmanız gereken işlem konfigürasyon işlemidir. Bu işlem için programdaki CONFIG butonunu tıklayınız.



Şekil 5: Picallw programında konfigürasyonun hazırlanması

Çıkan ekranda osilatör tipini, watchdog timer ve power up timer seçeneklerini programınıza ve devrenize uygun olacak şekilde seçiniz. Eğer pic üzerindeki programın başkaları tarafından okunmasını istemiyorsanız protection kısmındaki Code protect 0 kısmını seçebilirsiniz. Deneme çalışmalarında bu kısmı seçmeyiniz.

Artık programınız yüklenmeye hazırdır ve şimdi ekrandaki programınız üzerindeki program butonunu tıklayınız. Eğer bir problem yoksa ekranda bazı sayılar görünecek ve sonunda "Device Was Successfully Programmed in 10 Seconds" (Alet 10 saniyede başarıyla programlandı) mesajını alacaksınız. Böylece yazdığınız program artık mikrodenetleyicinize kaydolmuştur. Artık setinizi kapatıp işlemcinizi buradan çıkarak uygulama devresinde deneyebilirsiniz.

EK-3: P16F84.INC DOSYASININ İÇERİĞİ

Bu bölümde hem .inc uzantılı dosyalara örnek olması açısından, hem de pic16f84 ile ilgili bazı bilgileri içermesi açısından p16f84.inc dosyasının içeriği verilmiştir.

LIST

; P16F84.INC Standard Header File, Version 2.00 ;Microchip Technology, Inc.

NOLIST

; This header file defines configurations, ;registers, and other useful bits of
; information for the PIC16F84 microcontroller. ;These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before ;this file is
; included. The processor may be selected the ;following ways:

; 1. Command line switch:

; C:\ MPASM MYFILE.ASM /PIC16F84

; 2. LIST directive in the source file

; LIST P=PIC16F84

; 3. Processor Type entry in the MPASM full-screen interface

=====

; Revision History

=====

;Rev: Date: Reason:

;2.00 07/24/96 Renamed to reflect the name change ;to PIC16F84.

;1.01 05/17/96 Corrected BADRAM map

;1.00 10/31/95 Initial Release

=====

; Verify Processor

=====

IFNDEF __16F84

MESSG "Processor-header file mismatch. ;Verify selected processor."

ENDIF

```

=====
;
;   Register Definitions
;
=====
W          EQU   H'0000'
F          EQU   H'0001'

;----- Register Files-----
INDF       EQU   H'0000'
TMR0       EQU   H'0001'
PCL        EQU   H'0002'
STATUS     EQU   H'0003'
FSR        EQU   H'0004'
PORTA      EQU   H'0005'
PORTB      EQU   H'0006'
EEDATA     EQU   H'0008'
EEADR      EQU   H'0009'
PCLATH     EQU   H'000A'
INTCON     EQU   H'000B'
OPTION_REG EQU   H'0081'
TRISA      EQU   H'0085'
TRISB      EQU   H'0086'
EECON1     EQU   H'0088'
EECON2     EQU   H'0089'

;----- STATUS Bits -----
IRP        EQU   H'0007'
RP1        EQU   H'0006'
RP0        EQU   H'0005'
NOT_TO     EQU   H'0004'
NOT_PD     EQU   H'0003'
Z          EQU   H'0002'
DC         EQU   H'0001'
C          EQU   H'0000'

```

```

;----- INTCON Bits -----
GIE            EQU    H'0007'
EEIE           EQU    H'0006'
T0IE           EQU    H'0005'
INTE           EQU    H'0004'
RBIE           EQU    H'0003'
T0IF           EQU    H'0002'
INTF           EQU    H'0001'
RBIF           EQU    H'0000'

;----- OPTION Bits -----
NOT_RBPU       EQU    H'0007'
INTEDG         EQU    H'0006'
T0CS           EQU    H'0005'
T0SE           EQU    H'0004'
PSA            EQU    H'0003'
PS2            EQU    H'0002'
PS1            EQU    H'0001'
PS0            EQU    H'0000'

;----- EECON1 Bits -----
EEIF           EQU    H'0004'
WRERR          EQU    H'0003'
WREN           EQU    H'0002'
WR             EQU    H'0001'
RD             EQU    H'0000'

;=====
;    RAM Definition
;=====
__MAXRAM H'CF'
__BADRAM H'07', H'50'-H'7F', H'87'

```

```

;=====
;   Configuration Bits
;=====

_CP_ON      EQU    H'000F'
_CP_OFF     EQU    H'3FFF'
_PWRTE_ON   EQU    H'3FF7'
_PWRTE_OFF  EQU    H'3FFF'
_WDT_ON     EQU    H'3FFF'
_WDT_OFF    EQU    H'3FFB'
_LP_OSC     EQU    H'3FFC'
_XT_OSC     EQU    H'3FFD'
_HS_OSC     EQU    H'3FFE'
_RC_OSC     EQU    H'3FFF'
LIST

```


YARARLANILAN KAYNAKLAR

1. Pic16F84 Data Sheets, Microchip
(www.microchip.com)

