

PIC'le İlgili Temel Kavramlar:

Mikroişlemci ve mikrodenetleyicilerin farkı: Mikroişlemciler hafıza veya giriş çıkış entegresi gibi çevre elemanlar ile birlikte aritmetik işlemler yapabilen, karar verebilen entegrelerdir. Mikrodenetleyici ise mikroişlemcinin yanısıra kendi program hafızası ve giriş çıkış özelliklerini içinde barındıran entegrelerdir. Bu özellikleri sayesinde yüklü bir programı uygulamaya dönüştürerek bir kontrol sistemi olarak kullanılabilir.

Hafıza entegresi çeşitleri: Hafıza entegreleri bilgilerin ve program parçalarının saklanmasına yarayan entegrelerdir. RAM (random access memory) program çalıştığı sürece bilgilerin yazıldığı ve okunduğu, enerjinin kesilmesi durumunda içindeki tüm bilgiyi kaybeden bellek türüdür. ROM (read only memory) sadece bir kez yüklenip sonra içindeki bilgiler silinip değiştirilemeyen, tekrar programlanamayan bellek türüdür. İçindeki bilgiler enerji kesintisinden etkilenmez. EPROM (erasable programmable read only memory) ROM'un mor ötesi ışık altında silinebilen bir alt türüdür. Silindikten sonra içine tekrar program yazılabilir ama ROM'da olduğu gibi enerji kesildiğinde içindeki bilgiler kaybolmaz. Uzun süre herhangi bir ışık altında kaldığında silindiği için, içindeki bilgileri korumak için ışık aldığı pencerenin üstünün kapalı tutulması gerekir. EEPROM (electrical eprom) EPROM'un ışıkla değil, elektrikle silinebilen türüdür. EEPROM'lar silme gerilimi verilmediği sürece silinmedikleri için mikrodenetleyiciler için ek bellek alanı olarak sıkça kullanılır.

Program Memory: Program çalışırken kullanılacak kodun saklı tutulduğu hafıza alanı.

Data Memory: Çalışma esnasında saklanması gereken verilerin tutulduğu bellek alanı.

Akümülatör (W): Mikrodenetleyicide tüm aritmetik ve lojik işlemlerin gerçekleştirildiği bellek alanıdır.

Register: Mikroişlemcinin değişik fonksiyonlarını kullanmak için gerekli özel tanımlı hafıza alanlarıdır. Bazıları sabit olup bazıları kullanıcılar tarafından değişken olarak kullanılabilirler.

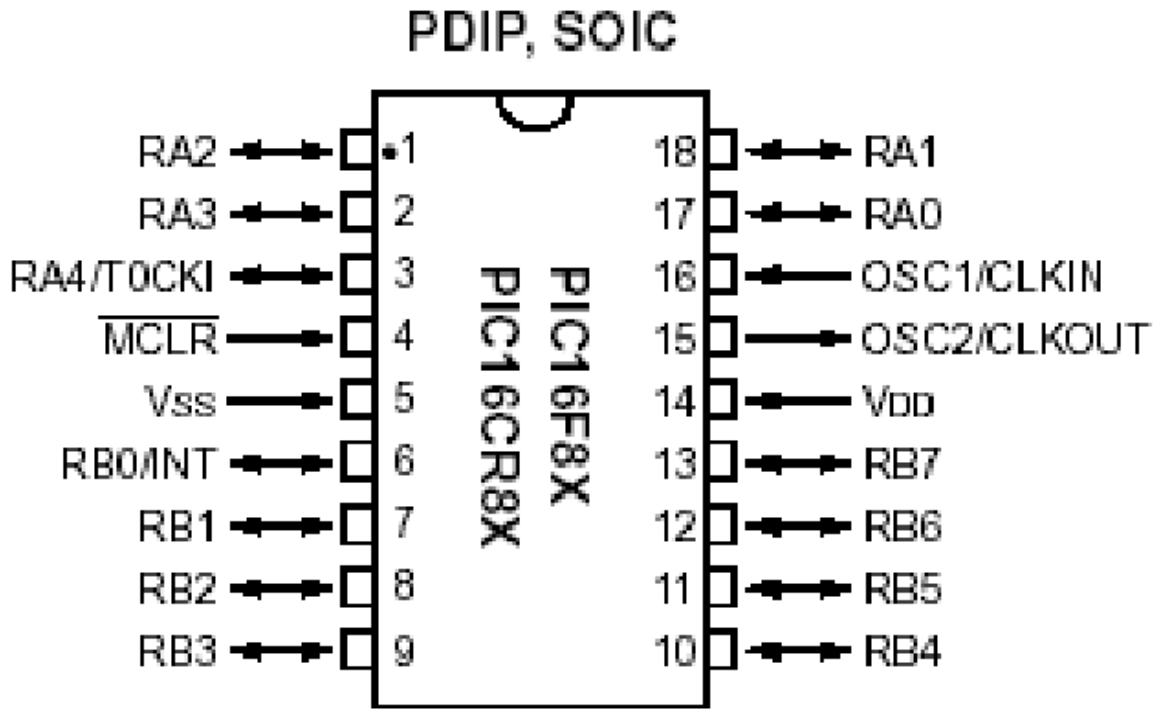
PC (Program counter): Mikroişlemci tarafından yürütülecek komutun adresini tutan özel tanımlı bir registerdır.

PIC (Peripheral Interface Controller): Çevresel üniteleri denetleyici arabirim.

PIC Ailesinin Mimarisi

Pic mikroişlemcileri belirli özelliklere ve eklemelere dayalı olarak serilere ayrılmıştır. Bunlardan temel olan P16F84'ün pin yapısı ve genel mimarisi aşağıda verilmiştir.

Pin Diyagramı:



Pic'in $\overline{\text{MCLR}}$ bağlantısı reset ve programlama anlarını normal çalışmadan ayırmaya yarar. Pic'in bu bacağına 5V verildiğinde Pic, içindeki programı çalıştırma moduna girer. 13V verildiğinde içine yeni program yüklenmeye hazır hale gelir, toprağa bağlanırsa Pic resetlenir. Vss bacağı toprak girişidir, Vdd bacağı 5V besleme girişidir.

OSC1 ve OSC2 bacakları Pic'in çalışma frekansını belirleyen kristali veya RC osilatörünü bağlamak içindir. Kalan 13 bacak bilgi girişi ve çıkışı için kullanılabilen haberleşme bacaklarıdır. Bunların 8'i PortB 5'i PortA olarak ayrılmış ve RB0, RB4, RB5, RB6, RB7'ye kesme girişi özelliği, RA4'e ise zamanlayıcı için harici osilatör girişi özelliği eklenmiştir.

[illegible]

P16F84'ün data belleği Bank0 ve Bank1 olarak ikiye bölünmüştür. Bu iki bölümün de başında 11'er byte'lık sabit veri bölgeleri vardır. Bu bölgeler Pic'in çalışma sırasındaki kendi durumuyla ilgili bilgilerini içerdikleri için Specific Function Register (Özel İşlev Belleği) olarak adlandırılırlar. Data belleğinin kalan kısmı işlem sırasında kullanılacak değişkenlere ayrılmıştır. Pic, enerji verildiği zaman standart olarak Bank0'ı işlem belleği olarak kullanır. Bank1'deki işlemleri ve değişken bölgesini kullanmak için öncelikle Bank1'e geçiş komutunu vermek gerekir. Pic'teki işlemci 8 bit veriyoluna sahiptir ve bu yüzden işlemci 255'ten büyük sayıları tek seferde işleyemez. Buna

Status byte'ı: Bu byte'ın her iki bankta da bulunmasının nedeni bank seçiminin iki yönlü olmasıdır.

Bit 0: Carry biti işlem sırasında sonucun 255'in (11111111b) üstüne çıkması durumlarında (taşma) otomatik olarak 1 olur.

Bit 1: Digit Carry biti işlem sırasında sonucun 15'in (00001111b) üstüne çıkması durumlarında otomatik olarak 1 olur.

Bit 2: Zero biti işlem sırasında sonucun 0 olması durumunda otomatik olarak 1 olur.

Bit 3: Power-down biti Pic'e besleme voltajı geldiğinde otomatik olarak 1 olur, sistem uyku moduna girince 0 olur.

Bit 4: Time-out biti sistemin kilitlenmesi nedeniyle resetleme olana kadar 1 kalır, olduğunda ise 0 olur.

Bit 5: Data belleğinde Bank0-Bank1 seçiminin yapılmasını sağlar. 1 olduğunda Bank1, 0 olduğunda Bank0 seçilir.

Bit 6: P16F877 gibi daha çok özelliğe sahip mikroişlemcilerde var olan Bank0, Bank1, Bank2 ve Bank3'ten hangisinin seçileceğini belirlemek için gereken Bit 5'ten sonraki ikinci binary sayıyı oluşturur. Sadece Bank0 ve Bank1'in bulunduğu mikrodenetleyicilerde sabit olarak 0'da kalır.

Bit 7: Dolaylı adresleme için kullanılan bank seçme biti.

PortA-PortB byte'ları:

Bu byte'lar A ve B bacaklarına gelen bilgiyi görmek ve bu bacaklara bilgi yüklemek için kullanılır. Toplam 5 tane A bacağı olduğu için PortA byte'ının kalan üç bitinin değerinin önemi yoktur. Bu byte'lara verilen değer binary karşılığı o bacaklardaki işareti belirler.

Ör. PortA'ya 13 decimal değerini verdiğimizde 13'ün karşılığı olan 00001101 sayısı RA0, RA2 ve RA3 bacaklarını 5V'a çıkarır, kalan A bacaklarını 0V'a çeker.

TrisA-TrisB byte'ları:

Uygulama bakımından PortA ve PortB byte'larına benzeyen bu byte'lar A ve B bacaklarının giriş mi yoksa çıkış mı olarak kullanılacağını belirler.

PortA byte'ında

olduğu gibi TrisA'da da üst üç bitin değerinin önemi yoktur.

Ör. PortA'ya 13 decimal değerini verdiğimizde 13'ün karşılığı olan 00001101 sayısı RA0, RA2 ve RA3 bacaklarını giriş bacağı, kalan A bacaklarını çıkış bacağı yapar.

Option_Reg Byte'ı:

Bit 0-2: Bu üç bit, timer (zamanlayıcı) kullanıldığında timer'ın kaç saat periyodunda bir içindeki sayıyı arttıracaklarını belirler. Aşağıdaki tabloda bu üç bite verilen değerler için zamanlayıcının (TMR0) ve sistemin kilitlenmesi nedeniyle resetleme için çalışan zamanlayıcının (WDT-Watchdog Timer) kaç saat periyodunda bir sayacağını gösteriliyor.

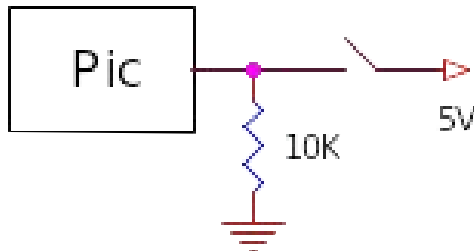
Bit Value	TMR0 Rate	WDT Rate
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Bit 3: (PSA - Prescaler Assignment bit) Çalışma sırasında hangi zamanlayıcının devrede olacağını belirler. 1 olduğunda WDT, 0 olduğunda TMR0 devreye girer.

Bit 4: (T0SE - TMR0 Source Edge Select bit) RA4 dış osilatör girişi kullanıldığında TMR0'ın saat darbesinin hangi kenarında sayacağını belirler. 1 ise düşen kenar, 0 ise yükselen kenarda sayım gerçekleşir.

Bit 5: (T0CS - TMR0 Clock Source Select bit) Clock sinyalinin RA4 bacağından mı yoksa harici girişten mi alınacağına karar vermek için kullanılır. 1 yapıldığında RA4 bacağından 0 yapıldığında harici girişten sinyal alınacağını belirtir.

Bit 6: (INTDEG - Interrupt Edge Select bit) Interrupt'ın hangi kenarda işleme konulacağını belirtir. 1 yapılırsa yükselen kenarda, 0 yapılırsa düşen kenarda interrupt



Bit 7: (RBP0 - PORTB Pull-up Enable bit) Pic içindeki PortB pull-up dirençlerinin kullanılıp kullanılmayacağına karar vermek için kullanılır.

1 yapılırsa kullanılmayacağını, 0 yapılırsa

devreye girecektir.

Intcon Byte'ı:

Bit 0: (RBIF – RB Port Change Interrupt Flag bit) RB 4, RB 5, RB 6 ve RB 7 ‘den herhangi birine kesim gelip gelmediğini gösteren flag bitidir. Eğer bunlardan herhangi birine kesim geldiyse bu bit otomatik olarak 1 olur.

Bit 1: (INTF – RB0/INT Interrupt Flag bit) RB 0’a ait flag bitidir. RB 0’a kesim geldiğinde otomatik olarak 1 olur.

Bit 2: (TOIF – TMR0 Overflow Interrupt flag bit) Timer’a ait flag bitidir. Timer’a girilen değerden başlayarak sayı 255’i geçtiğinde kesim zaman taşması kesmesi gelir ve TOIF otomatik olarak 1 olur.

Bit 3: (RBIE – RB Port Change Interrupt Enable bit) RB 4, RB 5, RB 6, RB 7 kesme hattının kesim girişini açmak için kullanılır. 1 yapıldığında kesim girişine izin verildiğini, 0 yapıldığında ise izin verilmediğini gösterir.

Bit 4: (INTE – RB0/INT Interrupt Enable bit) RB 0’a kesim girişi sağlamak için kullanılır. 1 yapıldığında interrupt girişine izin verildiğini, 0 yapıldığında izin verilmediğini gösterir.

Bit 5: (TOIE – TMR0 Overflow Interrupt Enable bit) Timer kesim girişini sağlamak için kullanılır. 1 yapıldığında kesim girişine izin verildiğini, 0 yapıldığında ise izin verilmediğini gösterir.

Bit 7: (GIE – Global Interrupt Enable bit) Tüm kesim girişlerinin ilk kontrol bitidir. Herhangi bir interrupt’ı kullanmak için 1 yapılması şarttır, 0 yapıldığında hiçbir interrupt çalışmaz; ancak 1 olduğunda herhangi bir interrupt’ın devreye girmesi için RBIE, INTE veya TOIE’nin de 1 yapılması gerekir.

Assembler Komut Seti:

Tabloda assembler dili komut seti gösteriliyor. En solda komutun adı, sağında bu komut için nelerin girilmesi gerektiği (f data belleğinden gelecek bilgiyi, d sonucun yazılacağı byte’ı, b sonucun yazılacağı biti, k ise dönecek sabit sayıyı ifade eder) ve en sağda komutun anlamı yazılıdır.

ADDWF	f, d	W ile f’i toplar.
ANDWF	f, d	W ile f’i VE işlemine tabi tutar.
CLRF	f	f’i sıfırlar.
CLRW		W’yi sıfırlar.
COMF	f, d	f’in tersini alır.

DECF	f, d	f'teki sayıyı bir azaltır.
DECFSZ	f, d	f'teki sayıyı bir azaltır, sonuç 0'sa altındaki komutu atlar.
INCF	f, d	f'teki sayıyı bir arttırır.
INCFSZ	f, d	f'teki sayıyı bir arttırır, sonuç 0'sa altındaki komutu atlar.
IORWF	f, d	W ile f'i VEYA işlemine tabi tutar.
MOVF	f, d	f'teki sayıyı hedef gösterilen yere taşır.
MOVWF	f	W'deki sayıyı f'te hedef gösterilen yere taşır.
NOP		Bir periyod boyunca hiçbir şey yapmaz.
RLF	f, d	f'teki 8 biti başa Status,0 bitini ekleyerek sola doğru kaydırır, dokuzuncu bitteki değeri sıfırıncı bite atar.
RRF	f, d	f'teki 8 biti başa Status,0 bitini ekleyerek sağa doğru kaydırır, dokuzuncu bitteki değeri sıfırıncı bite atar.
SUBWF	f, d	W'deki sayıyı f'teki sayıdan çıkartır.
SWAPF	f, d	İlk dört bitle son dört bitin yerlerini değiştirir.
XORWF	f, d	W ile f'i XOR işlemine tabi tutar.
BCF	f, b	f'teki bir biti sıfırlar.
BSF	f, b	f'teki bir biti set eder.
BTFSC	f, b	f'teki sayı 0'sa altındaki komutu atlar.
BTFSS	f, b	f'teki sayı 1'se altındaki komutu atlar.
ADDLW	k	Verilen sabit sayıyı W'deki sayıya ekler.
ANDLW	k	Verilen sabit sayıyla W'deki sayıyı VE işlemine tabi tutar.
CALL	k	Alt programı programın içine enjekte eder.
CLRWD		WDT'deki değeri sıfırlar.
GOTO	k	Programda istenilen yere atlar.
IORLW	k	Verilen sabit sayıyla W'deki sayıyı VEYA işlemine tabi tutar.
MOVLW	k	Verilen sabit sayıyı W'nin içine atar.
RETFIE		Kesme sonunda yapılan işlemten normal programa dönüş.
RETLW	k	Tablola sırasında sayı değerini W'ye atar.
RETURN		Call'la çağırılan alt programın sonu.
SLEEP		Uyku moduna geçiş komutu.
SUBLW	k	W'deki sayıyı verilen sabit sayıdan çıkartır.
XORLW	k	Verilen sabit sayıyla W'deki sayıyı ÖZELVEYA işlemine tabi tutar

Burada verilen örneklerde görünmese de Assembly dilinde program içinde kullanılan açıklama metinleri önlerine ; işareti eklenerek ayrılırlar.

ÖRNEK PROGRAMLAR:

Portları giriş ve çıkış olarak tanımlama:

PROCESSOR 16F84 Her programın başında bu tanımlama yapılır.

INCLUDE "P16F84.INC" Assembly uzantısından hex uzantısına geçiş sırasında programın byte'ların ve bitlerin adres karşılıklarını görebilmesi için yazılmış dosya

MAIN:	Başlangıç ayarları, RA0'a buton, RB0'a led bağlı
BSF STATUS, RP0	Status byte'ının 5. biti (RP0) 1 yapıldı; bank1 seçildi
MOVLW 00000001b	00000001b sayısı W'ye atandı
MOVWF TRISA	W'deki sayı TrisA'ya taşındı, RA0 giriş, kalan A bacakları çıkış
CLRF TRISB	TrisB 00000000b oldu, bütün B bacakları çıkış
BCF STATUS, RP0	Bank0'a dönüş
START:	Start bloğunun başlangıcı
CLRF PORTB	B bacaklarının hepsi 0V
BTFSS PORTA,0	RA0 1'se alttaki satırı atla, değilse alt satıra geç.
GOTO START	START bloğunun başına git.
BSF PORTB,0	B' nin 0. portunu 5V yap.
GOTO START	START bloğunun başına git.
END	Program sonu.

Öncelikle led'in takılacağı RB0 çıkış, buton takılacağı RA0 giriş olarak seçildi ve program döngüsünün başında B bacakları sıfırlandı. Sonra program butonun basılı olup olmadığını kontrol etmeye başladı. Butona basılı olmadığı zaman RA0 0V görecektir ve program tekrar tekrar buton kontrolü yapacak, butona basıldığında RA0 5V görecektir ve program RB0'ı set edip tekrar kontrol döngüsüne girecektir.

Gecikme kontrolüyle kare dalga üretme:

PROCESSOR 16F84	
INCLUDE "P16F84.INC"	
DEL EQU 0CH	Değişkenler için ilk bellek adresi 0Ch'ta DEL değişkeni oluştu.
SEL EQU 0DH	0Dh'ta SEL değişkeni tanımlandı.
MAIN:	
BSF STATUS, RP0	Bank1 seçildi.
CLRF TRISB	B portlarının hepsi output(çıkış) olarak ayarlandı.

BCF STATUS, RP0	Bank0'a dönüş.
START:	
BSF PORTB,0	RB0 set edildi, RB0'a bağlı led yandı.
CALL DELAY	DELAY bloğu çağırıldı.
BCF PORTB,0	RB0 reset edildi, RB0'a bağlı led söndü.
CALL DELAY	
GOTO START	Start bloğunun başına gidildi.
DELAY:	DELAY program bloğu
MOVLW .250	Decimal 250'nin binary karşılığı (11111010b) W'ye atandı.
MOVWF DEL	W'deki sayı DEL değişkenine atandı.
DEL1:	DEL1 program bloğu
MOVLW .100	Decimal 100'ün binary karşılığı (01100100b) W'ye atandı.
MOVWF SEL	W'deki sayı SEL değişkenine atandı.
DEL2:	DEL2 program bloğu
NOP	Bir işlemci periyodu boyunca bekle
DECFSZ SEL,F	SEL'deki sayıyı bir azalt, sonuç 0'sa alt satırı atla
GOTO DEL2	DEL2 program bloğuna git.
DECFSZ DEL,F	DEL'deki sayıyı bir azalt, sonuç 0'sa alt satırı atla
GOTO DEL1	DEL1 program bloğuna git.
RETURN	DELAY'in çağırıldığı yere geri dönüş.
END	

Programın başında gecikme işleminde kullanılmak üzere DEL ve SEL değişkenleri oluşturuldu ve hazırlık olarak B portları çıkış yapıldı. Programdaki START döngüsünde önce RB0 set edildi, sonra DELAY işlemi gerçekleştirildi, RB0 reset edildi ve tekrar DELAY işlemi gerçekleştirildi. Sonra tekrar tekrar START döngüsü çalıştı. DELAY işleminin başında DEL'e 250 değeri verildi ve DEL1 işleminde de SEL'e 100 değeri verildi. DEL2 işleminde bir periyod beklendikten sonra SEL sayısı bir küçültüldü ve DEL2 işlemi tekrar başlatılarak SEL sayısı 0 olana kadar DEL2 işlemi tekrarlandı.

SEL sayısı 0 olunca DEL sayısı bir küçültüldü ve DEL2 döngüsü DEL değeri 0 olana kadar tekrarlandı. Pic'in işlemcisi kristalin frekansının 4'te biri hızında çalıştığı için 4 MHz'lik kristalle bir işlem periyodu 1 mikrosaniye'dir ve yazılmış olan DELAY döngüsü yaklaşık olarak $250 \times 100 = 25000$ mikrosaniye bekleme sağlayacağı için çıkışta 50 milisaniye periyodlu kare dalga görülür.

Makro kullanımı:

PROCESSOR 16F84

INCLUDE "P16F84.INC"

INCLUDE "MYMAC.MAC" Hazır makroların bulunduğu MAC uzantılı dosyadaki makroları kullanmak için

DEL EQU 0CH

SEL EQU 0DH

DELAY1 Macro DELAY1 makrosunun tanımlanması

CALL DELAY DELAY1 makrosunun içeriği

CALL DELAY

CALL DELAY

CALL DELAY

ENDM Makronun bitimi

MAIN:

Bank1 Bank1 makrosunun içeriği işlenir, Bank1'e geçilir.

CLRF TRISB

Bank0 Bank0 makrosunun içeriği işlenir, Bank0'a dönülür.

START:

BSF PORTB,0

DELAY1 Delay1 makrosunun içeriği olan 4 kez

DELAY işlenir.

BCF PORTB,0

DELAY1

GOTO START

DELAY:

```

MOVLW .250
MOVWF DEL
DEL1:
MOVLW .100
MOVWF SEL
DEL2:
NOP
DECFSZ SEL,F
GOTO DEL2
DECFSZ DEL,F
GOTO DEL1
RETURN
END

```

MYMAC.MAC'in içeriği:

Bank0 MACRO	Bank0 makrosunun tanımlanması
BCF STATUS, RP0	Bank0 makrosunun içeriği
ENDM	Makronun bitimi
Bank1 MACRO	Bank1 makrosunun tanımlanması
BSF STATUS, RP0	Bank1 makrosunun içeriği
ENDM	Makronun bitimi

Programın başında mymac.mac dosyası programın içine yerleştirilir; böylece bu dosyadaki Bank0 ve Bank1 makroları kullanıma açılır. Makro tanımlamalarında önce makronun adı sonuna "Macro" sözcüğü eklenerek yazılır, sonra makro çağırıldığında yapılacak işlemler yazılır ve makronun bitimine "Endm" yazılarak makro tamamlanır. Program içinde makro kullanılacağı zaman sadece adı yazılır.

Tablo kullanımı:

```

PROCESSOR 16F84
INCLUDE "P16F84.INC"
INCLUDE "MYMAC.MAC"
VERI EQU 0CH

```

Veri adında bir değişken tanımlandı.

#DEFINE BUTON PORTA, 0
verdi.

MAIN:

Bank1

MOVLW .1

MOVWF TRISA

CLRF TRISB

Bank0

CLRF VERI

CLRF PORTB

START:

BTFSS BUTON

GOTO START

INCF VERI

CALL TABLO

MOVWF PORTB

BUTONCHECK:

BTFSC BUTON

göremeden

GOTO BUTONCHECK

GOTO START

TABLO:

MOVLW VERI

ADDWF PCL,1

satır

RETLW 01000000b

sonucu olan

RETLW 01111001b

Örneğin,

RETLW 00100100b

seçeneğine , 1 ise

RETLW 00110000b

Define komutuyla RA0'a "BUTON" adını

Bank1 makrosuyla Bank1'e geçildi.

RA0 giriş yapıldı.

B portları çıkış yapıldı.

Bank0 makrosuyla Bank0'a dönüldü.

Veri değişkenini sıfırladı.

B portlarını sıfırladı.

RA0'a 1 geliyorsa alt satırı atlayacak.

Veri'deki sayı bir arttı.

TABLO program bloğunu çağırıyor.

TABLO'nun sonucunda W'ye atanan
değeri PortB byte'ına yazar.

Butondan elimizi çekmemizi bekliyor. Bu
olmasaydı Pic çok hızlı olduğu için biz

0'dan 9'a kadar saymayı bitirirdi.

Veri'deki sayı W'ye atanıyor.

Tablo çağırma komutunun bulunduğu

numarasına W'deki sayıyı ekleyip

sayıya karşılık gelen satıra atlar.

Veri'deki sayı 0'sa ilk RETLW

ikinci, 2 ise üçüncü,... RETLW satırına

RETLW 00011001b	geçerek oradaki binary sayıyı W'ye aktarır.
RETLW 00010010b	Burada yazılı olan binary sayılar 0'dan 9'a kadar
RETLW 00000010b	sayıların ortak anot display'de görünmeleri için
RETLW 01111000b	gerekli olan değerlerdir. Bu sayılar belirlenirken
RETLW 00000000b	display'in a bacağından g bacağına kadar
RETLW 00010000b	sırayla RB0-RB6 portlarına bağlı olacağı
END	düşünülmüştür.

Programdaki TABLO bloğu aşağıdaki gibi de yazılabilir.

TABLO:

MOVLW VERI

DT 01000000b, 01111001b, 00100100b, 00110000b, 00011001b,
00010010b, 00000010b, 01111000b, 00000000b, 00010000b

Hexadecimal Decimal Çevirici:

PROCESSOR 16F84

INCLUDE "P16F84.INC"

SAYI EQU 0Ch

YUZ EQU 0Dh

ON EQU 0Eh

BIR EQU 0Fh

TEMP EQU 10h

MAIN:

BSF STATUS, RP0

CLRF TRISA

CLRF TRISB

BCF STATUS, RP0

START:

MOVLW 7Bh

7Bh (01111011) sayısını W'ye atar.

MOVWF SAYI

W'deki değeri Sayı'ya atar.

CALL CONVERT_DECIMAL

Convert_Decimal bloğunu çağırır.

MOVF YUZ,W	Convert_Decimal sonucunda Yuz'e atanan değeri W'ye atar
BCF STATUS,C	Carry bitini sıfırlıyor.
RLF YUZ,F	YUZ değişkeninin içindeki sayıları dört kez sola kaydırıyor.
RLF YUZ,F	
RLF YUZ,F	
RLF YUZ,F	
MOVF YUZ,W	YUZ değişkeninin son değerini W'ye atar.
ADDLW ON	ON değişkeninin son değerini W'deki sayıyla toplar.
MOVWF PORTB	Çıkan sonucu PortB'ye atar.
MOVF BIR,W	BIR değişkeninin son değerini W'ye atar.
MOVWF PORTA	W'deki değeri PortA'ya atar.
CONVERT_DECIMAL:	
CLRF YUZ	
CLRF ON	
CLRF BIR	
DEC_100:	
MOVLW .100	Decimal 100 değerini W'ye atar.
SUBWF SAYI,W	SAYI'daki değerden W'deki değeri çıkarır.
MOVWF TEMP	Sonucu TEMP değişkenine atar.
BTFSS STATUS,C	Status,C (Carry) biti 1'se alttaki satırı atla
GOTO DEC_10	DEC_10 bloğuna gider.
MOVF TEMP,W	TEMP'teki sayı değerini W'ye atar.
MOVWF SAYI	W'deki sayı değerini SAYI'ya atar.
INCF YUZ,F	YUZ'deki sayıyı bir arttırır.
GOTO DEC_100	DEC_100 bloğuna gider.
DEC_10:	
MOVLW .10	Decimal 10 değerini W'ye atar.
SUBWF SAYI,W	SAYI'daki değerden W'deki değeri çıkarır.
MOVWF TEMP	Sonucu TEMP değişkenine atar.

BTFSS STATUS,C	Status,C (Carry) biti 1'se alttaki satırı atla
GOTO DEC_1	DEC_10 bloğuna gider.
MOVF TEMP,W	TEMP'teki sayı değerini W'ye atar.
MOVWF SAYI	W'deki sayı değerini SAYI'ya atar.
INCF ON,F	ON'daki sayıyı bir arttırır.
GOTO DEC_10	DEC_10 bloğuna gider.
DEC_1:	
MOVLW .1	Decimal 10 değerini W'ye atar.
SUBWF SAYI,W	SAYI'daki değerden W'deki değeri çıkarır.
MOVWF TEMP	Sonucu TEMP değişkenine atar.
BTFSS STATUS,C	Status,C (Carry) biti 1'se alttaki satırı atla
RETURN	Convert_Decimal'in çağırıldığı yere geri döner.
MOVF TEMP,W	TEMP'teki sayı değerini W'ye atar.
MOVWF SAYI	W'deki sayı değerini SAYI'ya atar.
INCF BIR,F	BIR'deki sayıyı bir arttırır.
GOTO DEC_1	DEC_10 bloğuna gider.
END	

Bu programda Carry bitinin taşma anlarında 1'den 0'a dönmesi ve RLF (RRF'de de aynı şekilde geçerli) işleminde byte'ın içindeki bitlerin Carry bitiyle beraber kayması özellikleri kullanılıyor. Bu programa göre programın içinde girilmiş olan 7Bh sayısı decimal'e çevrilerek, biri RB4-RB7, biri RB0-RB3 ve biri RA0-RA3 arasındaki portlara bağlı üç tane 74LS247 entegresi yoluyla Pic'ten çıkan 4'er bit üzerinden üç ayrı display'e gönderilir.

Programın başındaki BIR, ON, YUZ değişkenleri display'e gidecek birler, onlar ve yüzler basamaklarındaki sayıları tutacak değişkenler, TEMP ve SAYI program çalışırken çıkan sonuçların kaydedildiği geçici değişkenlerdir. Başta 7Bh sayısı SAYI'ya yüklendi ve DEC_100'ün içinde decimal 100 sayısı SAYI değişkenindeki sayıdan çıkartıldı. Pic'e voltaj verildiğinde her zaman 1

olan Status,C (Carry) biti, işlem sırasında sonuç 0'ın altına düşerse veya 255'in üstüne çıkarsa 0 olur. SAYI değişkeninden 100 çıkartıldıkça Carry biti 0 olmadığı sürece program yeni SAYI olarak eski sayının 100 eksikliğini atar ve YUZ değişkenindeki değeri bir arttırır. Böylece SAYI'daki değerin 100'ün kaç katı olduğu YUZ değişkenine yazılır. Carry biti 0 olduğunda program SAYI'nın içindeki değerde başka 100 bulunmadığını anlar ve onlar basamağını kontrol etmek için DEC_10 bloğuna gider. DEC_10 ve DEC_1'te de benzer işlemler 10 ve 1 çıkartılarak yapılır. Tek fark DEC_1'de de Carry biti sıfırlandığında programın başka hane kalmadığı için Convert_Decimal'in çağırıldığı yere geri dönmesidir. Program yüzler ve onlar hanesini sırasıyla RB4-RB7 ve RB0-RB3 üzerinden yolladığı için YUZ değişkeninin ilk dört biti PortB'nin son dört biti olmak üzere dört kez sola kaydırılıyor ve elde edilen sayı ON değişkenindeki sayıyla toplanıp W'ye atılıyor. Bu işlemten hemen önce Carry biti sıfırlanıyor çünkü Carry biti taşmadan dolayı 0 olduktan bir işlem sonra tekrar kendini set ediyor. RLF sırasında Carry biti de kaydığandan YUZ değişkeninin ilk dört bitine 1 gelmemesi için Carry bitinin işlem öncesinde sıfırlanması gerekir. YUZ değişkeninde RLF işlemi yapılırken ilk dört bitin 0 olmasını istememizin nedeni toplama sırasında herhangi bir 1'in PortB'ye gidecek sayı değerinde değişikliğe neden olmaması gerektiğidir.

Ör. YUZ: 00000100 ON: 00000010

Carry sıfırlanmadan önce Carry ve W: **1** 00000100

Carry sıfırlanmazsa RLF sonrası Carry ve W: 0 0000100**1** **Eski**

Carry biti

Carry sıfırlandıktan sonra Carry ve W: **0** 00000100

İlk RLF sonrası Carry ve W: 0 0000100**0**

Programın kalanında W'de oluşturulan yüzler ve onlar basamağı sayısı PortB'ye, BIR değişkenindeki sayı da PortA'ya atanarak display'de görüntü elde ediliyor.

Timer Interrupt'ı Kullanımı:

PROCESSOR 16F84

INCLUDE "P16F84.INC"

ORG 00h	Besleme verildiğinde Pic'in programa
başladığı yer	
GOTO MAIN	
ORG 04h	Program nerede olursa olsun interrupt
geldiğinde	
GOTO WAKE_UP	işleme girecek olan program bloğu
MAIN:	
BSF STATUS,RP0	
MOVLW H'F0'	11110000 sayısı W'ye yüklendi.
MOVWF OPTION_REG	Bu sayı Option_Reg'e yüklendi. Böylece iç
pullup'lar	
BCF TRISA,0	kapatıldı, RBO interrupt'ı yükselen kenar,
timer	
CLRF TRISB	clock sinyal kaynağı olarak RA4 ve düşen
BCF STATUS,RP0	kenarlarda sayması, timer olarak TMR0'ın
	iki periyotta bir sayması seçildi.
MOVLW 0xFE	Decimal 254 (11111110b) sayısı W'ye
yüklendi.	
MOVWF TMR0	W'deki sayı TMR0 değeri oldu, böylece
timer	
CLRF PORTB	254'ten saymaya başlayacak, 256 (0)
	saydığı anda interrupt gelecek.
CLRF PORTA	
BCF INTCON,T0IF	Timer interrupt flag'i sıfırlandı.
BSF INTCON,T0IE	Timer interrupt hakkı açıldı.
BSF INTCON,GIE	Genel interrupt hakkı açıldı.
LOOP:	
MOVF TMR0,W	
MOVWF PORTB	
GOTO LOOP	
WAKE_UP:	Interrupt gelince ORG 04h programı
	Wake_Up'ı çalıştırır.

BCF INTCON,T0IF

Interrupt gelince bir olan timer interrupt
flag'i sıfırlanıyor.

BSF PORTA,0

RETFIE

Programın interrupt'la kesildiği yere

dönüş.

END

Pic'te kullanılabilen toplam 5 çeşit interrupt vardır. Bunların en çok kullanılan üçü TMR0, RB0 ve RB4-RB7 interrupt'larıdır. Üstteki programda TMR0 interrupt'ı kullanılmıştır, ama RB0 ve RB4-RB7 interrupt'larının kullanımı TMR0 interrupt'ının kullanımına çok benzer. RB0 ve RB4-RB7 interrupt'larında clock sinyali olmadığı için Option_Reg'deki zamanlama ve timer'la ilgili ayarların bir önemi yoktur. Ancak, pull-up ayarlamaları ve sadece RB0 interrupt'ı için RB0 interrupt gelme kenarı (yükselen kenar veya düşen kenar) ayarlamaları için Option_Reg ayarlanır. RB0 ve RB4-RB7 interrupt'larının TMR0 interrupt'ından diğer farkı RBO'nın INTE ve INTF bitlerini RB4-RB7'ninse RBIE ve RBIF bitlerini interrupt izni (enable) ve interrupt flag bitleri olarak kullanmasıdır. Pic'te aynı anda birden çok interrupt kaynağı kullanılabilir, bu durumda interrupt'ın hangi kaynaktan geldiğini anlamanın en kolay yolu flag bitlerini kontrol etmektir. RB4-RB7 interruptından kesme geldiğinde interrupt'ın hangi bacadan geldiğini anlamanın yolu ise bacakları BTF ile tek tek kontrol etmektir.

Telefon Tuşu Numara Algılama:

PROCESSOR 16F84

INCLUDE "P16F84.INC"

DEL EQU 0CH

TEMP EQU 0DH

START:

CALL PORT_INIT

CLRF PORTA

LOOP:

CALL KEY_SCAN

CALL WRITE_DATA

CALL DELAY

PB4	PB5	PB6	PB7	
1	2	3	A	--1 PB0
4	5	6	B	--2 PB1
7	8	9	C	--3 PB2
*	0	#	D	--5 PB3

GOTO LOOP	
PORT_INIT:	
BSF STATUS, RP0	
CLRF TRISA	A bacakları çıkış
MOVLW 0FH	
MOVWF TRISB	RB0-RB3 giriş, RB4-RB7 çıkış atandı
BCF STATUS, RP0	
RETURN	
KEY_SCAN:	Tuş tarama komutu
BSF PORTB,4	Yalnız RB4 set edilerek 1. sütun kontrol
edilecek.	
BCF PORTB,5	
BCF PORTB,6	
BCF PORTB,7	
BTFSS PORTB,0	1. satıra 5V iletiliyorsa alt satırı atla.
GOTO ET1	
MOVLW 01H	01h sayısını W'e ata.
MOVWF TEMP	W'deki sayıyı TEMP'e at.
RETURN	
ET1:	
BTFSS PORTB,1	2. satıra 5V iletiliyorsa alt satırı atla.
GOTO ET2	
MOVLW 04H	01h sayısını W'e ata.
MOVWF TEMP	W'deki sayıyı TEMP'e at.
RETURN	

Herhangi bir tuş üstünden RB4-RB7 grubundan gelen 5V RB0-RB3 grubuna iletilmediği sürece program üstte ilk sütunun ilk iki satırının kontrolünde görüldüğü gibi sırayla bütün sütunları satır satır kontrol eder.

WRITE_DATA:	
MOVF TEMP,W	KEY_SCAN'de TEMP'e yüklenen değer
veya	
MOVWF PORTA	değer yüklenmediyse de 0 PortA'ya
atandı.	

RETURN

DELAY: 256 periyod bekler.

CLRF DEL
sıfırlamak 256

DLY: yapmakla aynı şeydir.

DECFSZ DEL,F

GOTO DLY

RETURN

Telefon tuş setinde bir tuşa basıldığı zaman o tuşun ait olduğu satır ve sütundaki hatlar birbirine kısa devre olur. Bu programda RB0-RB3 sırayla 1-4. satırlara, RB4-RB7 sırayla 1-4. sütunlara bağlıdır. Program basılı tuşu bulana kadar, sütunlara sırayla 5V veriyor ve her sütun için satır satır 5V iletilmiş mi diye kontrol ediyor. Basılı tuşu bulamadığı sürece TEMP'in içindeki son değer PortA'ya atanarak 74LS247 aracılığıyla display'e yazılıyor. Tuş bulunduğu zaman o tuşa ait değer TEMP'e yazılarak PortA'ya atanıyor.

Pic Basic:

LCD, EEPROM, analog dijital çevirme, seri haberleşme, shift register, çeşitli şekillerde sinyal çıkışı almak ve birçok sık görülen uygulamanın Pic'le kolay yapılabilmesi için bu işleri yapan Assembly komutları makrolar halinde toplanmış ve Pic Basic dili oluşturulmuştur. Bu program dili bu tür işleri sadece parametreleri olan birer komuta indirgemıştır. Bunlara örnek olarak 24Cxx serisi EEPROM'larla bilgi alışverişini sağlayan I2CREAD, I2CWRITE, LCD ekrana yazı yazan LCDOUT, koşul kontrolünü sağlayan IF...THEN...ELSE...ENDIF ve WHILE...WEND, Pic'in içindeki EEPROM'la bilgi alışverişini sağlayan READ, WRITE ve senkron ve asenkron seri haberleşmeyi sağlayan SHIFTIN, SHIFTOUT, SERIN, SEROUT komutları gösterilebilir.

Pic Basic'te Matematik Operatörleri:

Operatör	Açıklaması
+	Toplama
-	Çıkarma

*	Çarpma
**	16 bit çarpma $W0 = 250$ $W1 = W0 ** 1000$ 250000 = %111101000010010000 Sonucun üst 16 bitini W1'e atar.
/	Bölme
//	Bölümden kalan $B = 12$ $C=5$ $A = B // C$ B'nin C'ye bölümünden kalan 2'yi A'ya atar.
<<	Sola kaydırma $B0 = B0 << 3$ B0'ın içindeki bitler üç kez sola kaydı.
>>	Sağa kaydırma
ABS	Mutlak değer $A=5$ $B=2$ $C=(B - A)$ $C=253$ $C = ABS (B - A)$ $C = 3$
COS	Kosinüs 0-360 derece 0-255 binary karşılıklarıyla ifade ediliyor. $B0=63$ $B1=COS B0$ $B1=0$ B0'daki 63 sayısı 90 derecenin karşılığı.
SIN	Sinüs
DIG	Sayının içinden hane seçme $B0=123$ $B1=B0 \rightarrow DIG1$ $B1=2$
MAX	Verilen iki sayı arasından en büyük sayıyı seçer
MIN	Verilen iki sayı arasından en küçük sayıyı seçer
NCD	Sayının içinde en soldaki 1 bitinin kaçınıcı bit olduğunu verir $B0=NCD \%01001000$ $B0=7$
DCD	Söylenen biti 1 yapar, gerisini sıfırlar
REV	Değişken içinde yanındaki sayı kadar alt bitin sırasını ters çevirir $A=01000011$ $B=A REV3$ $B=01000110$
SQR	Karekök alır $B=4$ $A=SQR B$ $A=2$ $B=5$ $A=SQR B$ $A=2$
&	İki sayının binary karşılığını VE işlemine sokar
	İki sayının binary karşılığını VEYA işlemine sokar
^	İki sayının binary karşılığını ÖZELVEYA işlemine sokar
~	Bir sayının binary karşılığını DEĞİL işlemine sokar
&/	İki sayının binary karşılığını VEDEĞİL işlemine sokar

/	İki sayının binary karşılığını VEYADEĞİL işlemine sokar
^/	İki sayının binary karşılığını ÖZELVEYADEĞİL işlemine sokar

Pic Basic'te Karşılaştırma Operatörleri:

=, ==	Eşit
<>, !=	Eşit değil
<	Küçüktür
>	Büyüktür
<=	Küçük eşit
>=	Büyük eşit

Pic Basic'te Mantıksal Operatörler:

AND, &&	Mantıksal VE
OR,	Mantıksal VEYA
XOR, ^^	Mantıksal ÖZELVEYA
NOT AND	Mantıksal VEDEĞİL
NOT OR	Mantıksal VEYADEĞİL
NOT XOR	Mantıksal ÖZELVEYADEĞİL

Pic Basic'te Komutlar:

@	Tek satırlık assembly komutu kullanmak için.
ADCIN Channel, Var	Channel'daki analog değerın karşılığı olan dijital değeri Var'a atar.
ASM..ENDASM	Assembly komut grubu kullanmak için.
BRANCH Index, [Label...]	Label dizisinde Index sayısını karşılayan seçeneğe yönlendirir.
CALL Label	Assembly'deki CALL komutunun aynısı
CLEAR	Bütün değişkenleri sıfırlar
CLEARWDT	WDT'nin içindeki değeri sıfırlayarak WDT'yi kapatır.
COUNT Pin, Period, Var	Belirli bir periyod içinde gelen puls sayısını Var'a atar
DATA Location, Constant	Pic'e program yüklenirken dahili EEPROM'a Location adresinden başlayarak Constant'ları yükler

DEBUG Var {,Var}	Define komutuyla önceden belirlenmiş bir pin üzerinden önceden belirlenmiş bir hızda (baud rate) seri haberleşme yoluyla Var'daki veriyi yollar.
DEBUGIN {Timeout, Label} [Var]	Define komutuyla önceden belirlenmiş bir pin üzerinden önceden belirlenmiş bir hızda (baud rate) seri haberleşme yoluyla gelen veriyi Var'a atar.
DISABLE	Interrupt girişi ve Debug işlemlerini engeller.
DISABLE DEBUG	Debug işlemini engeller.
DISABLE INTERRUPT	INTCON.GIE'yi sıfırlayarak interrupt girişini kapatır.
DTMFOUT Pin, {Onms, Offms,} [Tone]	Tone tuşunun telefondaki sesini Onms süresi boyunca tuşlar arasında Offms süresi kadar bekleterek çıkarır. Onms ve Offms belirtilmemişse 200ms Onms, 50 ms Offms süresi kullanır.
EEPROM Location, Constant	DATA'yla aynı işi yapar
ENABLE	DISABLE komutundan sonra interrupt girişine ve Debug komutuna izin verir.
ENABLE DEBUG	DISABLE komutundan sonra Debug komutuna izin verir.
ENABLE INTERRUPT	DISABLE komutundan sonra interrupt girişine izin verir.
END	Assembly'deki END'in aynısıdır.
FOR i=Start TO End Task NEXT i	Start değerinden End değerine kadar i'yi birer birer arttırarak Task'i uygular.
FREQOUT Pin, Onms, Frequency	Pin bacağında Onms süresi boyunca Frequency frekansında sinüs dalgası üretir.
GOSUB Label	Call komutuyla aynı işi yapar.
GOTO Label	Assembly'deki GOTO işleminin aynısı.
HIGH Pin	Pin bacağını bir yapar.
HPWM Channel, Dutycycle, Frequency	16F877 gibi PWM modülü olan mikrodnetleyicilerde bu modülün Channel numaralı kanalından % $(Duty+1)/2.56$ duty cycle'da Frequency frekansında PWM çıkışı sağlar
HSERIN	16F77 gibi USART özelliği olan mikrodnetleyicilerde

{ParityLabel,} {Timeout,Label}, [Var]	donanım asenkron seri haberleşme hattından veri bekler, gelen veriyi Var'a atar, Timeout ms süre boyunca veri gelmezse Label bloğuna gider.
HSEROUT [Var]	16F77 gibi USART özelliği olan mikrodenetleyicilerde Var'daki veriyi donanım asenkron seri haberleşme hattına yollar.
I2CREAD DataPin, ClockPin, Control, Address, Var	Control ve Address verilerini yollar, harici EEPROM'un Address bölgesinden gelen veriyi Var'a atar.
I2CWRITE DataPin, ClockPin, Control, Address, Var	Control ve Address bilgilerini yollayıp, harici EEPROM'un Address bölgesine Var'daki veriyi atar.
IF..THEN..ELSE..ENDIF	If'ten sonra gelen şart doğruysa Then'den sonraki işlemi, değilse Else'ten sonraki işlemi yapar.
INPUT Pin	Belirtilen Pin'i giriş yapar.
LCDIN Address, Var	LCD'deki RAM'in Address bölgesindeki veriyi Var'a atar.
LCDOUT Item	Item'deki değeri LCD'ye gönderir
{LET} Var = Value	Var'a Value'daki işlem veya değeri atar. Gerekli değil
LOOKDOWN Search, [List], Var	Search'teki veriyi List'in içinde arar, o verinin List'teki 0'dan başlayarak kaçınıcı veri olduğunu Var'a atar.
LOOKDOWN2 Search,Test, [List], Var	LOOKDOWN'dan farkı List'in içinde 16 bit gerektiren (256'dan büyük) değerler kullanılabilmesi ve Test için karşılaştırma operatörü kullanılırsa şartı sağlayan ilk sayıyı seçmesidir.
LOOKUP Index, [List], Var	List'in içinde Index'teki sayıya karşılık gelen veriyi Var'a atar.
LOOKUP2 Index, [List], Var	LOOKUP'tan farkı List'in içinde 256'dan büyük sayı kullanılabilmesidir.
LOW Pin	Belirtilen Pin'i sıfırlar.
NAP Period	$18 \times 2^{\text{Period}}$ ms süre için uyku moduna girer.
ON DEBUG GOTO Label	Debug geldiğinde Label'daki işlemleri yapar.

ON INTERRUPT GOTO Label	Interrupt geldiğinde Label'daki işlemleri yapar.
OUTPUT Pin	Pin'i çıkış bacağı yapar.
PAUSE Period	Period'daki sayı kadar milisaniye bekler.
PAUSEUS Period	Period'daki sayı kadar mikrosaniye bekler.
POT Pin, Scale, Var	Pin'deki potansiyometrenin ayarına göre Scale içindeki uygun değeri Var'a atarak
PULSIN Pin, State, Var	Pin'e gelen State (0 veya 1) pulsun süresinin kaç 10us (4MHz; 20MHz'de 2us) olduğunu Var'a atar.
PULSOUT Pin, Period	Pin üzerinden Period'daki sayı kadar 10us(4MHz; 20MHz'de 2us) süresince olan işaretin(0 veya 1) tersi puls verir.
PWM Pin, Duty, Cycle	Pin'den % (Duty+1)/2.56 duty cycle'da Cycle'daki değer kadar periyod boyunca PWM çıkışı sağlar.
RANDOM Var	0-255 arasında rastgele bir sayı seçip Var'a atar.
RCTIME Pin, State, Var	Pin'in State(0 veya 1) durumunda kalma süresinin kaç 10us (4MHz; 20MHz'de 2us) olduğunu Var'a atar.
READ Address, Var	Dahili EEPROM'un Address bölgesindeki bilgiyi Var'a atar.
READCODE Address, Var	READ komutundan farkı 16 bit verileri kullanması.
RESUME	Interrupt'la devreye giren program bloğunu bitirir, ana programın kaldığı yere geri dönüş sağlar.
RETURN	Call'la çağrılan program bloğunu bitirir, ana programın kaldığı yere geri dönüş sağlar.
REVERSE Pin	Pin girişse çıkış yapar, çıkışsa giriş yapar.
SELECT CASE Var CASE Expr1 Statement CASE Expr2 Statement CASE ELSE Statement END SELECT	Var'ın içindeki değeri Expr'lerin içinde arar, bulduğu seçeneğin altındaki Statement komutunu işler, bulamazsa Case Else'in altındaki Statement komutunu işler.
SERIN Pin, Mode,	Pin'e bağlı asenkron seri haberleşme hattından Mode'la

{Timeout, Label}, [Qual], Var	belirtilen hızda (baud rate) veri bekler, gelen veriyi Var'a atar, Timeout ms süre boyunca veri gelmezse Label bloğuna gider. [Qual] varsa verinin içinde Qual'dan sonraki ilk veriyi Var'a atar.
SERIN2 Pin, Mode, {Timeout, Label}, [Qual], Var	SERIN'den farkı standart bağlantı hızlarından farklı hızlar (baud rate) kullanılmasına izin vermesidir. Mode'daki sayı (1000000/hız)-20 olarak hesaplanır.
SEROUT Pin,Mode, [Var]	Var'ın içindeki verileri Pin üzerinden Mode hızında (baud rate) asenkron seri haberleşme hattına gönderir.
SEROUT2 Pin, Mode, [Var]	SEROUT'tan farkı standart bağlantı hızlarından farklı hızlar (baud rate) kullanılmasına izin vermesidir. Mode'daki sayı (1000000/hız)-20 olarak hesaplanır.
SHIFTIN Datapin, Clockpin, Mode, Var	Datapin'e bağlı senkron seri haberleşme hattından Clockpin'e bağlı ortak osilasyonu kullanarak Mode'la belirtilen transfer protokolünü kullanarak veri alır ve Var'a atar.
SHIFTOUT Datapin, Clockpin, Mode, Var	Datapin'e bağlı senkron seri haberleşme hattına Clockpin'e bağlı ortak osilasyonu kullanarak Mode'la belirtilen transfer protokolünü kullanarak Var'daki verileri gönderir.
SLEEP Period	Period s süresi için uyku modunda kalır.
SOUND Pin, [Note, Duration,{Note, Duration}]	Pin üstünden Note (1-127) notasını Duration*12ms süreyle üretir.
STOP	Sonsuz nop döngüsü.
SWAP Var1, Var2	Var1 ve Var2'nin içeriklerini takas eder.
TOGGLE Pin	Pin'deki değeri (0 veya 1) tersine çevirir.
WHILE Condition Statement WEND	Condition doğrulandığı sürece Statement komutunu uygular.
WRITE Address, Var	Var'daki veriyi dahili EEPROM'un Address bölgesine atar.
WRITECODE Address,Var	WRITE komutundan farkı 16 bit verileri kullanması.

XIN Datapin, Zeropin, {Timeout, Label}, [Var]	Datapin ve Zeropin'e bağı X-10 interface entegresinden veri bekler, gelen veriyi Var'a atar, Timeout ms süre boyunca veri gelmezse Label bloğuna gider.
XOUT Datapin, Zeropin, [Housecode\Keycode	Datapin ve Zeropin'e bağı X-10 interface entegresine Housecode ve Keycode verilerini yollar.

Açıklamalar:

Burada verilen örneklerde görünmese de Pic Basic'te program içinde kullanılan açıklama metinleri önlerine ' işaretini eklenerek ayrılırlar.

Zamanlama kullanan bütün fonksiyonlar standart olarak 4 Mhz'lik kristal osilatör kullanıldığı öngörülerek ayarlanmıştır. Örneğin 20 MHz'lik bir osilatör kullanılacaksa programın başında "Define osc 20" komutuyla yeni osilatör programa tanıtılır. Aşağıda görülen tanımlamalar o komutların kullanılacağı programın başında yazılmalıdır.

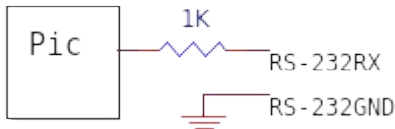
Debug Komutu İçin Gerekli Tanımlamalar:

DEFINE DEBUG_REG PORTB Debug verisinin yollanacağı bacağı RB0

DEFINE DEBUG_BIT 0 olarak seçti.

DEFINE DEBUG_BAUD 2400 Debug verisinin yollanacağı hızı 2400 olarak belirledi.

DEFINE DEBUG_MODE 1 Debug verisini invert ederek gönderecek, 0 olsaydı olduğu gibi gönderecekti.



Debugin Komutu İçin Gerekli Tanımlamalar:

DEFINE DEBUGIN_REG PORTB Debug verisinin alacağı bacağı seçti.

DEFINE DEBUGIN_BIT 0

DEFINE DEBUGIN_BAUD 2400 Debug verisinin alınacağı hızı 2400 olarak belirledi.

```
DEFINE DEBUGIN_MODE 1
```

Debug verisinin invert edilmiş olarak
geleceği anlatıldı, 0 olsaydı olduğu gibi
geleceği anlaşılacaktı.

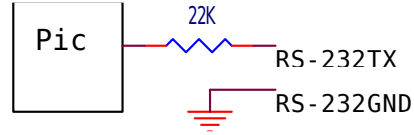
Hserin, Hserout Komutu İçin Gerekli Tanımlamalar:

```
DEFINE HSER_RCSTA 90h
```

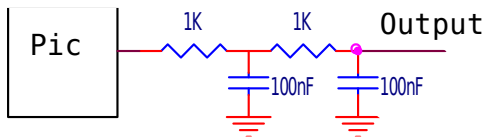
```
DEFINE HSER_TXSTA 20h
```

```
DEFINE HSER_BAUD 2400
```

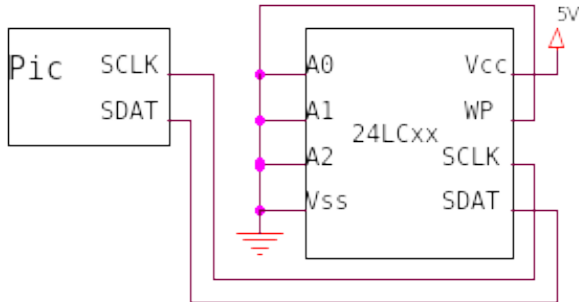
```
DEFINE HSER_SPBRG 25
```



DTMF ve Freqout Komutlarının Kullanımı İçin Gerekli Devre:



I2C READ, I2C WRITE Komutlarının Kullanımı İçin Gerekli Devre:



LCD Komutları İçin Gerekli Tanımlamalar:

```
DEFINE LCD_DREG PORTB
```

```
DEFINE LCD_DBIT 4
```

```
DEFINE LCD_RSREG PORTB
```

```
DEFINE LCD_REBIT 0
```

```
DEFINE LCD_RWREG PORTB
```

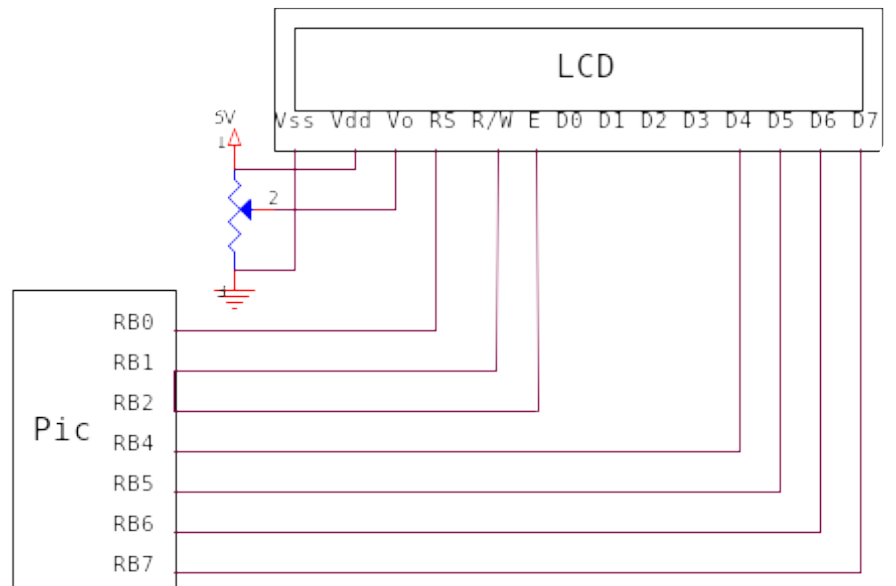
```
DEFINE LCD_RWBIT 1
```

```
DEFINE LCD_EREG PORTB
```

```
DEFINE LCD_EBIT 2
```

```
DEFINE LCD_BITS 4
```

```
DEFINE LCD_LINES 2
```



LCD'de Karakter Yazma Dışındaki İşlemler:

LCDOUT \$FE, 1	Ekranı temizleme
LCDOUT \$FE, 2	Üst satırın ilk harfine gitme
LCDOUT \$FE, \$0C	İmleçi gizler
LCDOUT \$FE, \$0E	Alt çizgiyi açar
LCDOUT \$FE, \$0F	Yanıp sönen imleçi açar
LCDOUT \$FE, \$10	İmleçi bir adım sola alma
LCDOUT \$FE, \$14	İmleçi bir adım sağa alma
LCDOUT \$FE, \$C0	İmleçi ikinci satırın ilk harfine götürme
LCDOUT \$FE, \$94	İmleçi üçüncü satırın ilk harfine götürme
LCDOUT \$FE, \$D4	İmleçi dördüncü satırın ilk harfine götürme

Pauseus Komutuyla Kullanılabilecek Minimum Bekleme

Süreleri:

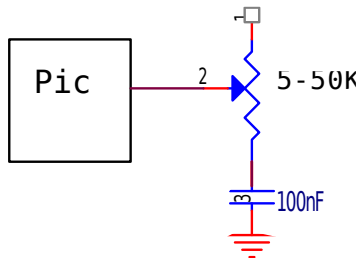
4MHz osilatör 24us

8MHz osilatör 12us

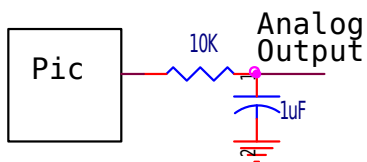
10MHz osilatör 8us

20MHz osilatör 3us

Pot Komutunun Kullanımı İçin Gerekli Devre:



PWM Komutunun Kullanımı İçin Gerekli Devre:

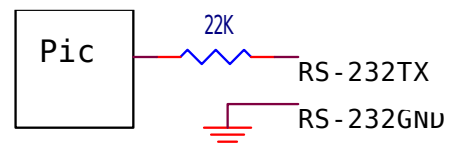


Serin Komutunda Mode'a Belirtilebilen Hız Tanımları:

T sinyalin aynı şekilde, N ise invert edilerek aktarıldığını belirtir. Yandaki sayı saniyede gönderilen veri paketi sayısıdır.

T2400 N2400

T1200 N1200



T9600 N9600

T300 N300

Serin2 Komutunda Kullanılabilen Hızlar ve Mode'a Yazılan

Karşılıkları:

300 3313

600 1646

1200 813

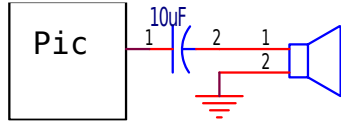
2400 396

4800 188

9600 84

19200 32

Sound Komutunun Kullanımı İçin Gerekli Devre:



Shiftin-Shiftout Komutlarında Kullanılabilen Mode'lar:

Shiftin modu:	No:	Açıklama:
MSBPRES	0	Veri en yüksek bitten başlayarak okunuyor, clock gönderilmeden önce veri okunuyor, clock boşken sıfırda.
LSBPRES	1	Veri en düşük bitten başlayarak okunuyor, clock gönderilmeden önce veri okunuyor, clock boşken sıfırda
MSBPOST	2	Veri en yüksek bitten başlayarak okunuyor, clock gönderildikten sonra veri okunuyor, clock boşken sıfırda.
LSBPOST	3	Veri en düşük bitten başlayarak okunuyor, clock gönderildikten sonra veri okunuyor, clock boşken sıfırda.
MSBPRES	4	Veri en yüksek bitten başlayarak okunuyor, clock gönderilmeden önce veri okunuyor, clock boşken set durumunda.
LSBPRES	5	Veri en düşük bitten başlayarak okunuyor, clock gönderilmeden önce veri okunuyor, clock boşken set durumunda.
MSBPOST	6	Veri en yüksek bitten başlayarak okunuyor, clock gönderildikten sonra veri okunuyor, clock boşken set

		durumunda.
LSBPOST	7	Veri en düşük bitten başlayarak okunuyor, clock gönderildikten sonra veri okunuyor, clock boşken set durumunda.

Shiftout modu:	No:	Açıklama:
LSBFIRST	0	Veri en düşük bitten başlayarak gönderiliyor, clock boşken sıfırda.
MSBFIRST	1	Veri en yüksek bitten başlayarak gönderiliyor, clock boşken sıfırda.
	4	Veri en düşük bitten başlayarak gönderiliyor, clock boşken set durumunda.
	5	Veri en yüksek bitten başlayarak gönderiliyor, clock boşken set durumunda.

16F877 ile İlgili Açıklamalar:

16F877 33 I/O bacağı olan bir mikrodenetleyicidir. Genel özellikleri

16F84'e benzer, ancak 8 adet 10 bit analog dijital çevirici, USART desteği ve iki adet PWM modülü içermesi onu üstün kılar.

HPWM komutuyla çalışan PWM modülleri çıkış bacağı olarak RC1ve RC2'yi kullanır.

ADC özelliğinin kullanılması için ADCON0 ve ADCON1 byte'larının istenilen özelliğe göre ayarlanması gerekir.

ADCON1 byte'ı:

Bit 3-0: PCFG3-PCFG0 analog/dijital port seçimi kontrol bitleridir.

Tabloda görülen V_{REF-} ve V_{REF+} 10 bit analog dijital çevriminde 0 ve 1024 sayılarının karşılığı olan voltaj değerlerini, A'lar ADC olarak kullanılacak bacakları, D'ler ise dijital I/O bacaklarını ifade ediyor.

PCFG3 PCFG0	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	V_{REF+}	V_{REF-}
0000	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}
0001	A	A	A	A	V_{REF+}	A	A	A	RA3	V_{SS}

0010	D	D	D	A	A	A	A	A	V _{DD}	V _{SS}
0011	D	D	D	A	V _{REF+}	A	A	A	RA3	V _{SS}
0100	D	D	D	D	A	D	A	A	V _{DD}	V _{SS}
0101	D	D	D	D	V _{REF+}	D	A	A	RA3	V _{SS}
011x	D	D	D	D	D	D	D	D	V _{DD}	V _{SS}
1000	A	A	A	A	V _{REF+}	V _{REF-}	A	A	RA3	RA2
1001	D	D	A	A	A	A	A	A	V _{DD}	V _{SS}
1010	D	D	A	A	V _{REF+}	A	A	A	RA3	V _{SS}
1011	D	D	A	A	V _{REF+}	V _{REF-}	A	A	RA3	RA2
1100	D	D	D	A	V _{REF+}	V _{REF-}	A	A	RA3	RA2
1101	D	D	D	D	V _{REF+}	V _{REF-}	A	A	RA3	RA2
1110	D	D	D	D	D	D	D	A	V _{DD}	V _{SS}
1111	D	D	D	D	V _{REF+}	V _{REF-}	D	A	RA3	RA2

Bit 6-4: Kullanım dışı bitler.

Bit 7: 0 olduğunda gelen 10 bitlik değer veri için açılmış olan 2x8 bitlik bölgeye üst 10 biti dolduracak şekilde, 1 olduğunda ise alt 10 biti dolduracak şekilde yüklenir. Analog dijital çevriminin 10 bit olması analog bilginin üst ve alt limitler arasındaki bölgenin 1024'e (0-1023) bölünmesi halinde karşılık geldiği değer dijital bilgi olarak gönderilmesi demektir. Örneğin 0V V_{REF-}, 5V V_{REF+} yapıldığında analog girişteki 2.5V 511 sayısı olarak işlenir.

ADCON0 byte'ı:

Bit 0: 1 ise ADC kullanılacağı, 0 ise kullanılmayacağı seçilmiş olur.

Bit 1: Kullanım dışı bit.

Bit 2: 1 yapıldığında analog dijital çevrimi başlar, bir çevrim bitince otomatik olarak sıfırlanır.

Bit 5-3: Pic aynı anda sadece bir analog dijital çevrimi yapabilir. Birden çok çevrim kullanıldığında bu çevrimler sırayla yapılır. Bu üç bitin yanyana beraber olarak değeri kaçınıcı (0-7) kanalın çevrime gireceğini belirler.

Bit 7-6: Analog dijital çevrimi için osilasyon kaynağını seçmeye yarar.

Güvenilir ölçüm için en yavaş olan ve sadece ADC hattı için kullanılan Pic içi RC osilatörü tercih edilmeli.

00 =FOSC/2 İşlemci osilatörünün yarısı hızında
01 =FOSC/8 İşlemci osilatörünün 8’de 1’i hızında
10 =FOSC/32 İşlemci osilatörünün 32’de 1’i hızında
11=FRC RC osilatörü seçeneği

Pic Basic’le uygulama örnekleri:

24Cxx Yazma, Okuma ve LCD Kullanımı:

Include “MODEDEFS.BAS” EEPROM’la seri haberleşme yapıldığı için gerekli

DEFINE LCD_DREG PORTB LCD’ye gidecek data hattının RB4’ten
DEFINE LCD_DBIT 4 başlayacağını belirtiyor.

DEFINE LCD_RSREG PORTB LCD’ye karakter yazımıyla LCD’nin iç işlemlerini

DEFINE LCD_RSBIT 0 birbirinden ayıran RS bitine ayrılan port

DEFINE LCD_EREG POR LCD’ye karakter gönderildikten sonra karakterin

DEFINE LCD_EBIT 1 görünmesi için 1 olacak enable bitine ayrılan port

DEFINE LCD_LINES 2 LCD ekranın satır sayısı

DEFINE LCD_BITS 4 LCD’ye kaç bit üzerinden data aktarılacağı

sclk var PORTA.0 EEPROM’a clock sinyalinin verileceği port’a sclk denildi

sdatt var PORTA.1 EEPROM’a data’nın verileceği port’a sdatt denildi

addr var byte 24Cxx’e gönderilecek bilgi için birer byte uzunluğunda

veri var byte adres ve veri değişkenleri tanımlandı.

cont con %10100000 Seri haberleşmede kullanılan kontrol byte’ı

Main:

addr=\$01 24Cxx’e Veri’yi yüklemek için 1 adresi seçildi.

veri=\$23 Veri değişkenine 23h yüklendi.

LCDOUT \$fe,1	LCD ekran temizledi.
LCDOUT hex2 veri , “EEPROM’a gidiyor”	Veri’deki değerin hexadecimal karşılığı ve tırnak içindeki ifade LCD’de göründü.
PAUSE 1000	1000 milisaniye bekledi.
I2CWRITE sdat, sclk, cont, addr, [veri]	24Cxx’in 1 adresine Veri’deki değer yüklendi
veri=0	Veri’deki değer sıfırlandı.
LCDOUT \$fe,1	LCD ekran temizledi.
LCDOUT hex2 veri	Veri’deki değerin hexadecimal karşılığı LCD’de göründü
PAUSE 1000	1000 milisaniye bekledi.
I2CREAD sdat, sclk, cont, addr, [veri]	24Cxx’in 1 adresindeki değer Veri’ye yüklendi.
LCDOUT \$fe,1	LCD ekran temizlendi.
LCDOUT hex2 veri	Veri’ye geri yüklenmiş olan 23h sayısı LCD’ye gönderildi.
Done:	
@ sleep	Assembly dilindeki sleep
komutu çağırıldı.	

Bu programın başında, EEPROM’la bilgi aktarımı seri haberleşme yoluyla yapıldığı için MODEDEFS.BAS dosyası eklendi. LCD kullanılacağı için LCD’nin bacaklarının Pic’e bağlanacağı portlar “Define” komutuyla tanıtıldı. Bu LCD tanıtma şekli bir kalıptır ve portların seçimi ve LCD’de kullanılacak satır sayısı haricinde standarttır. I2CREAD VE I2CWRITE komutları için gerekli olan parametreler sclk, sdat, addr ve veri birer byte’lık değişkenler (var) olarak, cont ise sabit sayı (con) olarak tanımlandı. Pic Basic’te değişkenler bit, byte ve 16 bitlik word olarak tanımlanabilir. Cont, seri haberleşmede kontrol amacıyla yollanan bir bytelık bir veri olan %10100000’ı ifade ediyor. Sclk, EEPROM’a bilgi gönderilirken verilen clock sinyalinin gönderildiği bacağı, sdat ise bilginin gönderildiği bacağı temsil eder. Bu programda da görüldüğü gibi Pic Basic’te bacakların adlandırılmasında da değişken olarak tanımlama

geçerli. Değişkenlere değer atanırken, değer hexadecimale olarak yazılması için başına \$, binary olarak yazılması için başına % işareti koyulur. Decimal olarak yazmak için herhangi bir işaret kullanılmaz. Programda EEPROM'la data alışverişi olduğunu göstermek için Veri'deki değer EEPROM'a yollanmadan önce LCD'ye aktarıldı, Veri sıfırlandıktan sonra Veri'deki değer tekrar LCD'ye aktarıldı ve son olarak I2C_READ komutuyla Veri'ye alınan değer LCD'ye aktarıldı. LCD'ye değerler aktarılırken o değer hexadecimale karşılığının yazılması için LCDOUT komutunun ardından Hex; decimal karşılığının yazılması için Dec veya #; binary karşılığının yazılması içinse Bin yazılır ve sonuna kaç basamak kullanılacağı yazılır. Her üç işlemin de arasında PAUSE 1000 komutuyla birer saniye beklendi, böylece sonuçları görebilmemiz için bize zaman tanınmış oldu. Son olarak Assembly dilinden @ ile çağırılan sleep komutuyla Pic uyku moduna alındı.

Bu programda Pic'in dahili EEPROM'u kullanılmak istenseydi I2C_WRITE ve I2C_READ komutları yerine sadece data ve adresin belirtilmesi gereken WRITE ve READ komutları kullanılacaktı. Bu iki komutun parametre düzenleri:

WRITE addr, veri ve READ addr, veri

16F877 ile ADC Kullanımı:

DEFINE LCD_DREG PORTB	LCD'ye gidecek data hattının RB4'ten
DEFINE LCD_DBIT 4	başlayacağı belirtiliyor.
DEFINE LCD_RSREG PORTB	LCD'ye karakter yazımıyla LCD'nin iç
	işlemlerini
DEFINE LCD_RSBIT 0	birbirinden ayıran RS bitine ayrılan port
DEFINE LCD_EREG PORTB	LCD'ye karakter gönderildikten sonra
	karakterin
DEFINE LCD_EBIT 1	görünmesi için 1 olması gereken enable
	bitine ayrılan port
DEFINE LCD_LINES 2	LCD ekranın satır sayısı
DEFINE LCD_BITS 4	LCD'ye kaç bit üzerinden data
	aktarılacağı
DEFINE ADC_BITS 8	8 bit ADC çözünürlüğü kullanımı seçildi.

DEFINE ADC_CLOCK 3	ADC için RC (11) osilatörü
seçildi.	
DEFINE ADC_SAMPLEUS 50	Doğru veriler almak için 50ms ADC'yi
bekletti.	
adcddata var byte	adcddata değişkeni
tanımlandı.	
A var byte	
B var byte	
C var byte	
Start:	
ADCON1=0	8 ADC portunun hepsi analog giriş
olarak ayarlandı.	
ADC_Okuma:	
ADCIN 0, adcddata	AN 0 girişindeki (RA0) analog verinin
karşılığı adcddata'ya atıldı.	
adcddata= (adcddata+1)*500/256	adcddata'daki değer AN0'daki analog
değere uyduruldu.	
Ekрана_Yaz:	
A=adcddata DIG 2	
B=adcddata DIG 1	
C= adcddata DIG 0	
LCDOUT \$fe, 1	Ekran temizlendi.
LCDOUT #A, ".", #B, #C, " V"	adcddata'daki değerin decimal
karşılığı ve V yazıldı.	
PAUSE 100	
Done:	
GOTO ADC_Okuma	

Bu programda RA0'a bağlı olan 0-5V arası bir voltaj kaynağının çıkışı 16F877'deki ADC özelliği kullanılarak ölçülüyor ve LCD'de gösteriliyor. Bu programda kullanılan ADC tanımlamaları yanlarındaki değerler dışında bir kalıptır. Burada 10 bit yerine 8 bit ADC kullanılmış, ancak Pic ADC verilerini hep 10 bit olarak kullanırken, Pic Basic'teki ADCIN komutu 10 biti 8 bite düşürüyor. 0-5V aralığı 256 eşit parçaya bölünmüş olduğu için asıl değeri

tekrar elde etmek için $(adcdata+1)*500/256$ işlemi kullanıldı. Pic ondalıklı sayılarda noktadan sonraki bölümü göremediği için sayı 5 yerine 500'le çarpıldı ve LCD'ye veri aktarılırken son sayının önce ilk rakamı, sonra bir nokta ve daha sonra kalan iki rakam aktarıldı.

DS1302 RTC Kullanımı:

INCLUDE "MODEDEFS.BAS"

DEFINE LCD_DREG PORTB	LCD'ye gidecek data hattının RB4'ten
DEFINE LCD_DBIT 4	başlayacağı belirtiliyor.

DEFINE LCD_RSREG PORTB LCD'ye karakter yazımıyla LCD'nin iç işlemlerini

DEFINE LCD_RSBIT 0 birbirinden ayıran RS bitine ayrılan port

DEFINE LCD_EREG PORTB LCD'ye karakter gönderildikten sonra karakterin

```

#define LCD_EBIT 1          görünmesi için 1 olması gereken enable
                             bitine
                             ayrılan port

```

```

#define LCD_LINES 2           LCD ekranın satır sayısı

```

DEFINE LCD_BITS 4 LCD'ye kaç bit üzerinden data
aktarılabileceği

RST var PORTA.0 RA0'a RST adını verdi.

IO var PORTA.1 RA1'e IO adını verdi.

SCLK var PORTA.2 RA2'ye SCLK adını verdi.

TRISB=0 LCD için B bacakları çıkış yapıldı.

```
rtcyear    var byte
```

```
rtcdlay    var byte
```

```
rtcmonth var byte
```

```
rtcdate    var byte
```

```
rtchr      var byte
```

```
rtcmin    var byte
```

```
rtcsec    var byte
```

```
rtccontrol var byte
```

Main:

LOW RST RST bacağını sıfırladı.

LOW SCLK	SCLK bacağını sıfırladı.
rtcyear=\$02	rtcyear değişkenine 02h
sayısını atadı.	
rtcday=\$06	
rtcmonth=\$04	
rtcdate=\$13	
rtchr=\$13	
rtcmin=\$42	
rtcsec=\$00	
GOSUB SetTime	SetTime program bloğunu
çağırdı.	
GOTO Main_Loop	Main_Loop program bloğuna
gitti.	
Main_Loop:	
GOSUB GetTime	GetTime program bloğunu
çağırdı.	
LCDOUT \$fe,1	Ekranı temizledi.
LCDOUT hex2 rtcmonth, "/", hex2 rtcdate, "/", hex2 rtcyear	ay/gün/yıl şeklinde tarihi yazdı.
LCDOUT \$fe, \$c0	İkinci satıra geçti
LCDOUT hex2 rtchr, ":", hex2 rtcmin, ":", hex2 rtcsec	saat:dakika:saniye şeklinde zamanı
yazdı.	
Pause 300	
GOTO Main_Loop	
SetTime:	
RST=1	Reseti kaldırarak haberleşme
hattını açtı.	
SHIFTOUT IO, SCLK, LSBFIRST, [\$8e, 0]	DS1302'yi ayarlayama kodunu
yolladı.	
RST=0	Resetleme
RST=1	Reseti kaldırarak haberleşme
hattını açtı.	

SHIFTOUT IO, SCLK, LSBFIRST,

[\$be,rtcsec,rtcmin,rtchr,rtcdate,rtcmonth,rtcday,rtcyear, 0]

DS1302'ye ayarlama yapılacağı bildirilmişti, şimdi de yeni zaman bilgileri yollandı.

RST=0

Resetleme

Return

GetTime:

RST=1

Reseti kaldırarak haberleşme

hattını açtı.

SHIFTOUT IO, SCLK; LSBFIRST, [\$bf]

DS1302'den zaman bilgisi alma

kodunu yolladı.

SHIFTIN IO,SCLK,LSBPRES, [rtcsec, rtcmin, rtchr, rtcdate, rtcmonth, rtcday, rtcyear, rtccontrol]

DS1302'den zaman bilgisi alınacağı bildirilmişti, şimdi de zaman bilgileri alındı.

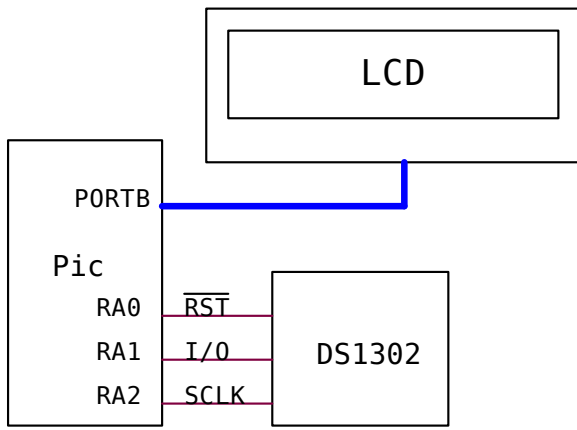
RST=0

Resetleme

RETURN

END

Programın başında seri haberleşme için gerekli Modedefs.bas eklendi ve LCD kullanımı için gerekli tanımlamalar yapıldı. Sonra da DS1302'nin belleğindeki bilgiler için uygun değişkenler tanımlandı. DS1302 bir saat entegresi ve kristal osilatörüyle beraber tam bir saat. Kendi içinde sırasıyla saniye, dakika, saat, gün, ay, haftanın kaçınıcı günü ve yıl bilgilerini içeren bir bellek sistemine sahip ve takılı osilatöre uyarak bu bellek bölümlerindeki sayıları otomatik olarak arttırıyor. Ancak başta bir saat ayarı olmadığı için programın içinde Pic buna başlangıç değerlerini aktarıyor, sonra saati göstermek için saat entegresinden tekrar tekrar son zaman bilgilerini alıp LCD ekrana aktarıyor. SHIFTOUT ve SHIFTIN komutları senkron seri haberleşmede kullanılan veri gönderme ve alma komutlarıdır. Bu yöntemde veri aktarımı için öncelikle RST 1 yapılarak data aktarımına izin veriliyor, sonra CLK sinyali veri'lerle eşzamanlı olarak gönderilerek veri gönderenle veri alanın aynı anda gerekli işlemleri



yapması sağlanıyor. Sırayla gelen iki SHIFTOUT,veya SHIFTIN komutu arasında bile RST'in sıfırlanmasının nedeni seri haberleşmede RST'in yükselen kenarı görüldüğü zaman Pic'e bağlı devrelerin haberleşme

yapılacağını anlamasıdır. Senkron seri haberleşmenin önemli olduğu 74HC595 shiftregister (kayan yazı entegresi) sisteminde ilk shiftregister devresine bir veri aktarılırken sırayla bütün shiftregister'lardaki verilerin bir sonrakine aktarılması veri aktarımı sırasında CLK sinyalinin bütün gruba gönderilmesi ve veri aktarımlarının aynı anda olmasıyla veri kaybı olmadan gerçekleşiyor.

INCLUDE "MODEDEFS.BAS"

oe var PORTA.0 DS1302'deki RST'in aynısı. OE output enable (çıkış izni) demek

sdat var PORTA.1

sclk var PORTA.2

Start:

PORTA=0

Main:

veri=\$03

SHIFTOUT sdat, sclk, LSBFIRST, [veri] Veri'deki değeri yollamak için komutu verdi.

oe=0

oe=1

veri'deki değer

oe=0

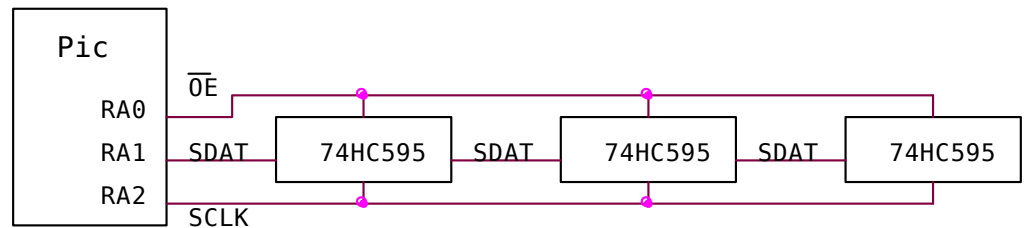
PAUSE 1000

GOTO Main

END

Oe'nin yükselen kenarında

74HC595'e gider.



Bu programda Pic kendisine en yakın 74HC595'ye 03h verisini yolluyor, 1 saniye bekliyor ve ikinci 03h verisini yolluyor. Pic ikinci veriyi yollarken ilk 74HC595'teki veri ikinci 74HC595'teki belleğe yükleniyor. Her bir saniyelik beklemeden sonra bu işlem tekrarlanıyor ve sırayla bütün 74HC595'lerin başta boş olan 8 bitlik belleklerine veri yazılmış oluyor. Bu entegrelere birer LM247'yle displayler takılsaydı sadece 3 sayıları içeren bir kayan yazı elde edilirdi. Şekilde üç tane 74HC595 birbirine bağlanmış ama bu sayıda bir sınırlama yok.

Asenkron Seri Haberleşme:

```
INCLUDE "MODEDEFS.BAS"
```

```
si var portb.0
```

Seri giriş bacağı tanımlandı.

```
so var portb.1
```

Seri çıkış bacağı tanımlandı.

```
veri var byte
```

```
baud con 84
```

Aktarma hızı (baud rate) 9600 veri

```
paketi/saniye oldu.
```

```
Start:
```

```
TRISB=1
```

si'yi giriş yaptı.

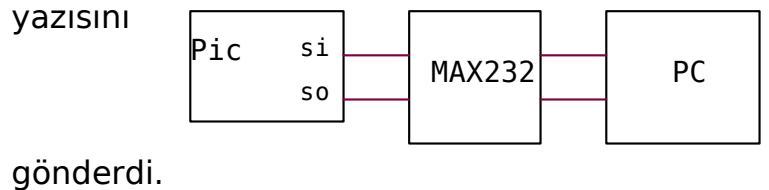
```
PORTB=0
```

```
Main:
```

```
PAUSE 1000
```

```
SEROUT2, so, baud, ["Merhaba"]
```

so çıkışından 9600 hızında "Merhaba" yazısını



gönderdi.

```
Loop:
```

```
SERIN2 si, baud, 100, Loop, [veri]
```

si'den 100 milisaniye boyunca veri gelmezse Loop'a gider, gelirse veri'ye atar.

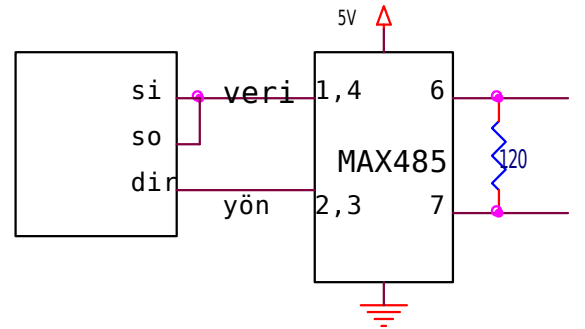
```
IF veri="x" then  
kodu)
```

veri'deki bilgi x harfiyse (x'in ASCII

```
SEROUT2 so, baud, ["X Alindi"]    "X Alindi" mesajını so
                                  üzerinden 9600    hızında
                                  gönderdi.

ELSE                               veri'deki bilgi x değilse "X Alinmadi"
mesajını so
SEROUT2 so, baud, ["X Alinmadi"]  üzerinden 9600 hızında
gönderdi.
ENDIF
```

Bu programda RS232 asenkron seri haberleşmesi kullanılarak önce "Merhaba" yazısı gönderildi, sonra hattan "x" verisinin gelip gelmediği kontrol edilip sonuca göre "x Alindi" veya "x Alinmadi" yazıldı. RS232 kısa mesafelerde (2-3 m), RS485 ise uzun mesafelerde (2km) ve elektromanyetik gürültü altında veri aktarımı sağlayabilen bir seri haberleşme şeklidir. Her ikisi için de kablodaki gürültüyle veriyi ayıran interface entegreleri (MAX232 ve MAX485) kullanılır. Yukarıdaki program Pic'e bağlı MAX232'nin seri çıkışının bilgisayarın seri portuna bağlanması yoluyla Hyperterminal programıyla haberleşmesini sağlamaktadır. Bu programda RS232 yerine RS485 haberleşmesi kullanılmak istenseydi MAX485 entegresinin aynı anda tek sinyal geçişine izin vermesi nedeniyle gereken veri akış yönü seçme işlemi için "dir" adında bir bacak tanımlanması ve serout komutundan önce dir'in 1, serin komutundan önce de dir'in 0 yapılması gerekir.



Interrupt Kullanımı:

Main:

Led var PORTB.1

TRISB=1

girişi)

OPTION_REG.7=1

dirençlerini kapattı.

RB0'ı giriş yaptı. (Interrupt

B bacaklarındaki pull-up

INTCON=%10010000

interrupt girişini açtı.

Start:

ON INTERRUPT GOTO Wake_Up
gidecek.

@ Sleep

Wake_Up:

DISABLE

gelirse

GOSUB Blink

INTCON.1=0

ENABLE

aktifleştirdi.

GOTO Start

girecek.

Blink:

Led=1

yaktı.

PAUSE 5000

Led=0

RETURN

döndü.

END

Bu programda öncelikle RB1 bacağı Led olarak adlandırılıyor, RB0 interrupt'ı kullanılacağı için RB0 giriş olarak ayarlanıyor ve RB0 interrupt'ı için GIE ve INTE set ediliyor. Start bloğunda interrupt geldiğinde Wake_Up bloğuna gitme komutu verildikten sonra Pic uyku moduna alınıyor. Interrupt geldiğinde Pic önce

RB0 interruptını ve genel

Interrupt gelirse Wake_Up bloğuna

Uyku moduna girdi.

Wake_Up sırasında başka interrupt

Wake_Up'ın başına dönmek için
interrupt girişini kapattı.

Interrupt flag'ini sıfırladı.

Interrupt girişlerini tekrar

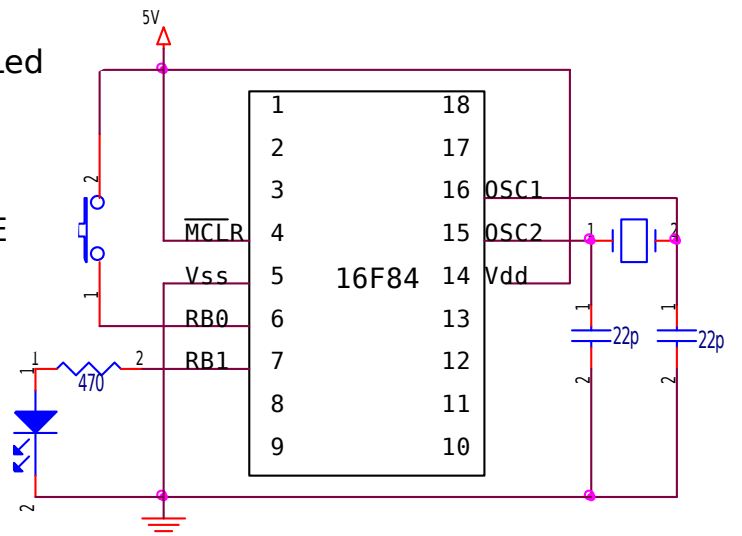
Tekrar interrupt döngüsüne

RB1'e bağlı Led'i

Bir saniye bekledi.

Led'i söndürdü.

Ana programa geri



interrupt girişlerini kapatıyor, Led'i 5 saniye süreyle yakıyor, interrupt flag'ini sıfırlıyor ve interrupt girişlerini tekrar açıp interrupt döngüsünü tekrarlıyor.

Böylece RB0'a bağlı tuşa basıldığı zaman Led 5 saniye yanar ve sonra tekrar basmamıza kadar uyku modunda kalır. Bu programdaki işlemi RB 4-7 interrupt hattıyla kullanmak için INTE ve INTF yerine RBIE ve RBIF kullanmak ve RB 4-7 girişlerini giriş olarak tanımlamak yeterli. TMR0 interrupt'ı kullanmak için ise Option_Reg'in içindeki timer ayarlarını yapmak ve INTE, INTF çiftini kullanmak gerekli.