

Bu bölümde program Yazmaya yeni başlayanların kullanmakta çekindiği bir kavram olan **Kesme (Interrupt)** kullanımını ele alacağız.

Kesme (Interrupt) tanımlayacak olursak, “bir programının normal çalışması esnasında, işleyişin bırakılıp kesmenin gerekliliğin yapıp tekrar normal programa dönülmesidir”. PIC lerde çeşitli kaynaklardan kesme alabilir. RBO,Portb Değişim, Timer 0-1-2, PortD, Eprom, Seri İletişim ve ADC çevrimi bunlardan bazılarıdır. PIC16F877 de 15 adet kesme kaynağı vardır.

PIC 16F877 de bu kesmeleri aktif etmek ya da iptal etmek için kullanılan INTCON Registeri mevcuttur. Bu registerin açıklaması aşağıda ele alacağız.

	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (x)	Features
INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

GIE: Tüm kesme işlemlerini etkin/iptal etme bayrağı

PEIE: Arabirim kesme aktif yapma bayrağı.

Çevresel ünitelerden gelen kesmeleri aktif etmek için kullanılan bittir. Bu çevresel ünitelerden bazıları USART, EPROM Yazma/Okuma, Paralel İletişim sayılabilir.

TOIE: TMR0 sayıcı kesmesini aktif yapma bayrağı

Tmr0, **Programın genel Akışından bağımsız olarak çalışan bir sayıcıdır**. Bu kesme ise bu sayıcının FF den 00 a geçişi anında oluşur. Bu bit bu kesmeyi aktif eder.

INTE: Harici kesmeyi aktif yapma bayrağı

INTE bayrağı, RBO pininden gelen tetiklemeye göre çalışır. Bu bit bu Interrupt ı aktif eder.

RBIE: PORTB (4, 5, 6, 7) deki değişiklik kesmesini aktif yapma bayrağı

TOIF: TMR0 sayıcısı zaman aşımı bayrağı

Bu bit TMR0 Sayıcısından dolayı oluşmuş bir Interrupt varsa “1” olur

INTF: Harici kesme bayrağı

Bu bit RBO kesmesinden dolayı oluşmuş bir Interrupt varsa “1” olur

RBIF: PORTB değişiklik bayrağı

Bu bit PORTB değişiklik oluşmuş bir Interrupt varsa “1” olur

Ayrıca yukarıdaki INTCON registerinden başka Interrupt ile ilgili olan Option Register de vardır.

OPTION REGISTER:

Option Register da, eğer INTCON üzerinden TMR0 taşma (\$FF den \$00 e geçerken) Interrupt ı aktif (TOIE) edilmişse bu TMR0 sayıcısı için gerekli olan saat sinyalinin (clock) kaç bölünmesi gerektiğini belirten üç adet bit vardır. Bahse konu olan Option Registerin yapısı ve bit lerin neler olduğu aşağıda açıklanmıştır.

Option Register							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBP}}\text{U}$	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0

RBPU: bu bit portb de bulunan dâhili PullUp dirençlerini aktif eder.

INTEDG: Aktif edilen Interrupt ın (**INTE**, **RBIE**) oluşmasını sağlayacak sinyalin düşen kenar ya da yükselen kenar seçimi yapar.

TOCS: TMR0 için saat sinyali (Clk) girişinin seçimini yapar

1 = TOCKI pin

0 = PIC için kullanılan Osilatör Kullanılır

TOSE: Eğer TOCKI seçildiyse saat sinyalinin Hangi kenarda sayma işlemi gerçekleşeceğini seçimini yapar.

1= TOCKI pin düşen kenar

0 = TOCKI pin Yükselen kenar

PSA: Bölme işleminin hangi sayıcı içi olacağını belirtir.

1 = WDT

0 = Timer0

PS2	PS1	PS0	TMR0 için	WDT için
0	0	0	1 / 2	1 / 1
0	0	1	1 / 4	1 / 2
0	1	0	1 / 8	1 / 4
0	1	1	1 / 16	1 / 8
1	0	0	1 / 32	1 / 16
1	0	1	1 / 64	1 / 32
1	1	0	1 / 128	1 / 64
1	1	1	1 / 256	1 / 128

PS2-PS0 bitleri ise sayma işlemi için gerekli olan saat darbesinin oranı belirler. **Eğer 1 / 128 seçilirse 128 makine saykılında (Osilatör Frekansı /4) TMR0 sayıcısı 1 kez sayar.** Aşağıdaki Proje ile ilk **SOFTWARE** Interrupt kodumuzu yazalım

Proje6;

Amaç: TMRO kesmesi kullanarak zaman gecikmesi sağlamak

Kullanılan Yeni Komutlar;**Symbol:**

Komut istediğimiz bir PORT 'a, port pinine, Kaydediciye takma ad atamak için kullanılır. Atanan bu isim yazımda ve kodların okunabilirliğini artırır.

Toggle:

Bir Pinin durumunu değiştirmek için kullanılır. Eğer pin "1" se "0", "0" sa "1" olur.

Inc:

Bir değişkenin içeriğini artırır. **Dec** komutu inc komutunun tersini yapar azaltır.

Aşağıda Kodları ve şeması verilmiş projeyi gerçekleştirip TMRO kesmesi hakkındaki hesaplamalara geçebiliriz.

Proton Plus Kodları:

```

Device 16F877
XTAL 4
OPTION_REG = %00000011      'Tmr0 için 1/16 oranı var
Symbol LED = PORTD.0        'Led Yazılan Heryerde Portd.0 kabul et
Symbol GIE=INTCON.7         'Global int
Symbol TMR0F=INTCON.2       'Tmr0 Flag
Symbol TMR0E=INTCON.5       'Tmr0 enable BIT

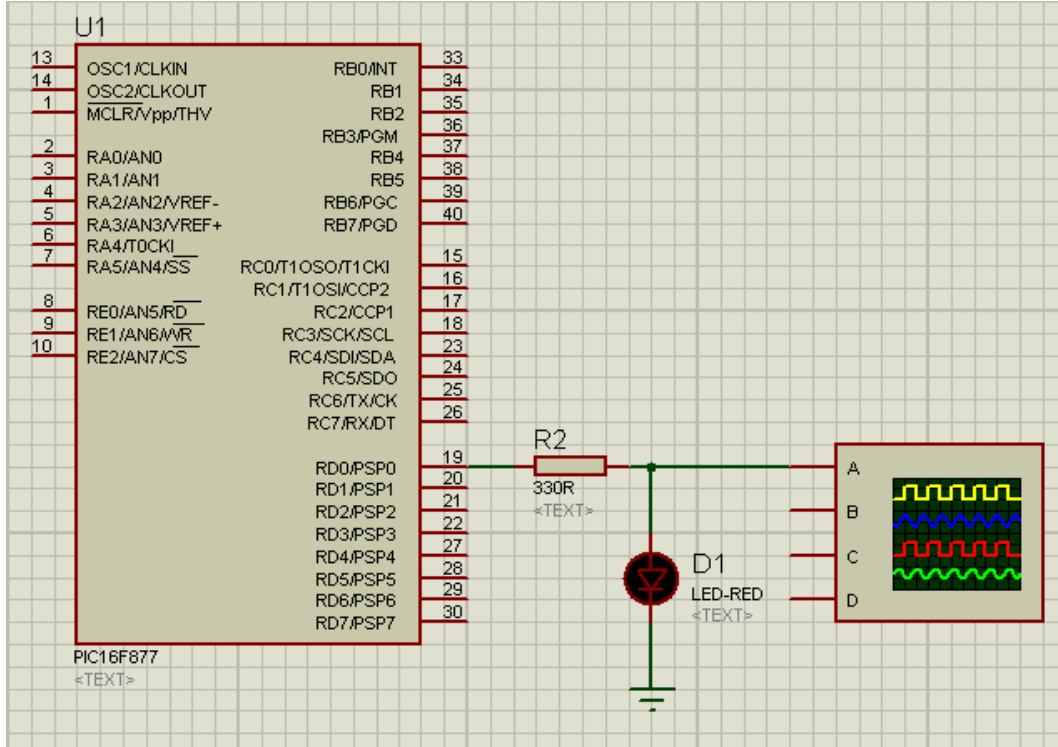
TMR0E=1                      'TMRO Aktif
GIE=1                        'Tüm Kesmeler Aktif
Output PORTB                'PortB Çıkış

Dim SAYI As Byte            'Sayı Adında 1 bytelık Değişken
Dim Dongu As Byte          'Dongu Adında 1 Bytelık Değişken
On Interrupt GoTo Int      'Software int. oluştuğunda INT etiketine git.
TMR0=6                      'Tmr0 6 değerini yükle
Basla:                      'Basla Etiket
  For Dongu=$00 To $FF      '00 dan FF e kadar
    PORTB=Dongu            'PortB ye At
    DelayMS 1              '1 Ms bekle
  Next
  GoTo Basla                'Basla Etiketine Git
'KESME ALT PROGRAM BAŞLANGICI-----
Disable                      'Tüm Software Interruptlar kapalı
Int:                          'INT Etiket
  TMR0=6                    'Tmr0 a 6 değerini yükle
  TMR0F=0                   'Kesme Tarafından aktif edilen bayrak temizlenir
  Inc SAYI                  'Sayı değişkenini 1 artır
  If SAYI=125 Then          'Eğer Sayı=125 ise
    Toggle LED              'led (portd.0) durumunu değiştir.
    SAYI=0                  'Sayı ya 0 ata
  EndIf                     'If den çık
  Resume                    'ana programa geri gön
  Enable                     'Tum Software Interruptlarını Aktif yap
'KESME ALT PROGRAM SONU-----
End                          'Program Sonu

```

ISIS Simulasyon Şeması:

Devrenin çalışmasını proje Klasörünü içerisindeki videodan da izleyebilirsiniz.



Yukarda Kullanılan TMR0 ile yapılan kesmenin zamanı 500 milisaniyedir. Gelin isterseniz bunu beraber hesaplayalım.

$$Osc = 4000000 \text{ olarak kabul ettiğimiz bir PIC de,}$$

$$Makina Saykılı = \frac{osc}{4} \text{ den } = 1000000 \text{ Hz dir}$$

Yukardaki Çözümde PIC in komut işleme süresinin (Makine Saykılı) 4Mhz lik Osilatör ile, 1 Mhz Olduğu Hesaplanıyor. Bu hespla 1 Makine Saykılının süresi $T = \frac{1}{F}$ den $T = \frac{1}{1\,000\,000}$ formüle edersek **1uS** olarak hesaplarız.

Yukardaki işlemden Sonra şimdi gelelim "Tmr0 in zaman taşmasının" ne zaman oluşacağına. Dikkat ettiyseniz Programın Başında "TMR0=6" ataması yapılmış bu atamayla Tmr0 sayıcısının "6" sayısından başlaması sağlanmıştır. Böylece Tmr0 in 250 adım sayması sağlanmıştır(256-6). Ayrıca `OPTION_REG = %00000011` ayarlaması ile 1/16 oranı belirlenmiştir. Buradan bizim çıkaracağımız anlam şudur; **her 16 makine saykılında Tmr0 sayıcısı 1 artsın**. Burada her 16uS de bir Tmr0 "1" Artacaktır. Ön ayar ile 250 ye Kadar saydığımız Tmr0 $Zaman = 250 * 16 = 4000uS$ lik zaman biriminde kesmeye girer. Sonuç Olarak Aşağıdaki Formül karşımıza çıkar.

$$KesmeZamanı \text{ (Saniye)} = \frac{1}{\frac{OscFrekansı \text{ (Hz)}}{4}} * (256 - Tmr0 \text{ Öndeğer})$$

```
If SAYI=125 Then
  Toggle LED
  SAYI=0
EndIf
```

Programda geçen yandaki komut satırı ile bu kesmenin her 125 defa oluşmasında LED ile tanımlanan PortD.0 numaralı çıkışı durum değiştirir. Aşağıdaki formülle bu durum değiştirme süresi 500mS olarak ayarlandığını görebilirsiniz.

$$Zaman = 125 * KesmeZamanı$$

$$Zaman = 125 * 4mS = 500mS$$

Proje 7;

Amaç: RBO ile dış etki ile **Software** interrupt çalışması ve 2x16 Alpha LCD Menü Çalışması.

Kullanılan Yeni Komutlar;

Select..EndSelect: Komut değerler arasındaki seçimlere göre işlem yapar. Konuyu örnek üzerinden anlatacak olursak

```
Select Menu_Durum
  Case 0
    Print At 1,6,"1"
  Case 1
    Print At 1,6,"2"
  Case Else
    Print At 1,6,"3"
EndSelect
```

Yukarıdaki kod bloğunda **Menu_Durum** adındaki değişkenin içeriği eğer "0" sa hemen altındaki kodları (**Print At 1,6,"1"**) çalıştıracaktır. Eğer Menu_Durum "1" se onun altındaki komut satırı işletilir, Eğer ne "0" nede "1" se "**Case Else**" ile devam eden satır işletilir. "**EndSelect**" ile sorgulamadan çıkılır.

Print: Komut ile LCD ye bilgi göndermek için kullanılır. Komutun doğru çalışması için LCD tanımlamalarının yapılması gerekir. Aşağıdaki kodlarla lcd nin bağlantısının nasıl yapıldığı ve bazı özellikleri tanımlanmıştır.

```
LCD_DTPIN = PORTD.4      'Lcd nin dataları Portd.4 en başlıyor
LCD_RSPIN = PORTE.0     'LCD RS pin Porte.0
LCD_ENPIN = PORTE.1     'Lcd EN pin Porte.1
LCD_INTERFACE = 4       '4-bit Interface
LCD_LINES = 2           'Lcd 2 satır
LCD_TYPE = 0           '
LCD_COMMANDS = 2000
LCD_DATAUS = 50
```

Bu tanımlamaları kendi kodlarımızın içerisine yazdığımız gibi "**INCLUDE**" bildirisi ile kullanılan dosyalarla da yapabiliriz. Burada sırası gelmişken **INCLUDE** bildirisinden bahsedeyim.

Include bildirisi ile daha önceden hazırlanmış tanımlama dosyalarını yazdığımız programa dahil etmiş oluruz. Proton Plus da daha önceden hazırlanmış dosyalar mevcuttur. Kullanımı aşağıdaki gibidir.

INCLUDE "DosyaAdı"

Aşağıdaki kodlar "proton_4.inc" dosyasından bir bölümü bulunmaktadır. Görüldüğü üzere birçok tanımlama yapılmıştır. Bunlar arasında lcd ile ilgili tanımlamalar, seri giriş çıkış için tanımlamalar vardır. Unutulmaması gereken husus "include" bildirisi ile tanımlanan dosyanın kendi yazdığımız programla derlenmesi yazıldığı satırda gerçekleşir. Bundan dolayı bu tanımlamaları programımızın ilk satırlarında yapmakta fayda vardır. Bu dosyaların Proton Plus tarafından kullanılabilen dosyalar olması gerekir.

Ayrıca dosyaların;

1. Doğrudan dosyanın bulunduğu klasörü yazarak ("C:\MY_INCLUDES\dosya. inc")
2. Proton Plus klasörü içerisinde
3. Derleyici klasörü içerisinde
4. INC klasörünün içerisinde

Olmalıdır.

"Proton_4.inc" dosyasından bir bölüm

```

Device = 16F877
XTAL = 4

LCD_DTPIN = PORTD.4
LCD_RSPIN = PORTE.0
LCD_ENPIN = PORTE.1
LCD_INTERFACE = 4 ' 4-bit Interface
LCD_LINES = 2
LCD_TYPE = 0
LCD_COMMANDUS = 2000
LCD_DATAUS = 50

SCL_PIN = PORTC.3
SDA_PIN = PORTC.4

SERIAL_BAUD = 9600
RSOUT_PIN = PORTC.6
RSOUT_MODE = TRUE
RSOUT_PACE = 1
RSIN_PIN = PORTC.7
RSIN_MODE = TRUE

H SERIAL_BAUD = 9600 ' Set baud rate to 9600
H SERIAL_RCSTA = %10010000 ' Enable serial port and continuous receive
H SERIAL_TXSTA = %00100100 ' Enable transmit and asynchronous mode
H SERIAL_CLEAR = 0n ' Enable Error clearing on received characters

```

Programımızda geçen aşağıdaki kod bloğu yerine

```

Device = 16F877
'--LCD BAGLANTILARI-----
    XTAL = 4
    LCD_DTPIN = PORTD.4 'Lcd nin dataları Portd.4 en başlıyor
    LCD_RSPIN = PORTE.0 'LCD RS pin Porte.0
    LCD_ENPIN = PORTE.1 'Lcd EN pin Porte.1
    LCD_INTERFACE = 4 '4-bit Interface
    LCD_LINES = 2 'Lcd 2 satır
    LCD_TYPE = 0
    LCD_COMMANDUS = 2000
    LCD_DATAUS = 50
'--LCD BAGLANTILARI-----
    ALL_DIGITAL TRUE '16f877 nin tüm uçları Dijital

```

Sadece include "proton 4.inc" yazmamız yeterli olacaktır.

Print Komutuna geri dönecek olursak. Print Komutunun alabileceği argümanlar dan bahsetmeden konuyu kapatmamak gerekir. Aşağıdaki listede Print Komutunun alabileceği çok kullanılan argümanlar ve işlevleri yer almaktadır.

AT ypos,xpos	Lcd nin belli segmentine bilgi gönderir	Print at 1,1,"Proton Plus"
CLS	Lcd yi temizler	Cls
BIN{1..32}	Binary bilgi gönderir	PRINT "VAR1= ", BIN VAR1
DEC{1..10}	Desimal sayı gönderir	DIM FLT AS FLOAT FLT = 3.145 PRINT DEC2 FLT '3,14 yazar
HEX{1..8}	Hex sayı yazdırır	PRINT "VAR1= ", HEX VAR1
REP c\n	Belli karakteri (c), N kadar yazdırır	
STR array\n	Array Tanımlı bir değişkenin içeriğindeki n kadar değişkeni yazdırır.	DIM MYARRAY [10] AS BYTE STR MYARRAY = "HELLO" PRINT STR MYARRAY \5
CSTR cdata	Send string data defined in a <u>CData</u> statement	

Aşağıdaki kodları Proton Plus ile yazarak derleyiniz ve ISIS simülasyonunu kurarak çalıştırınız.

```

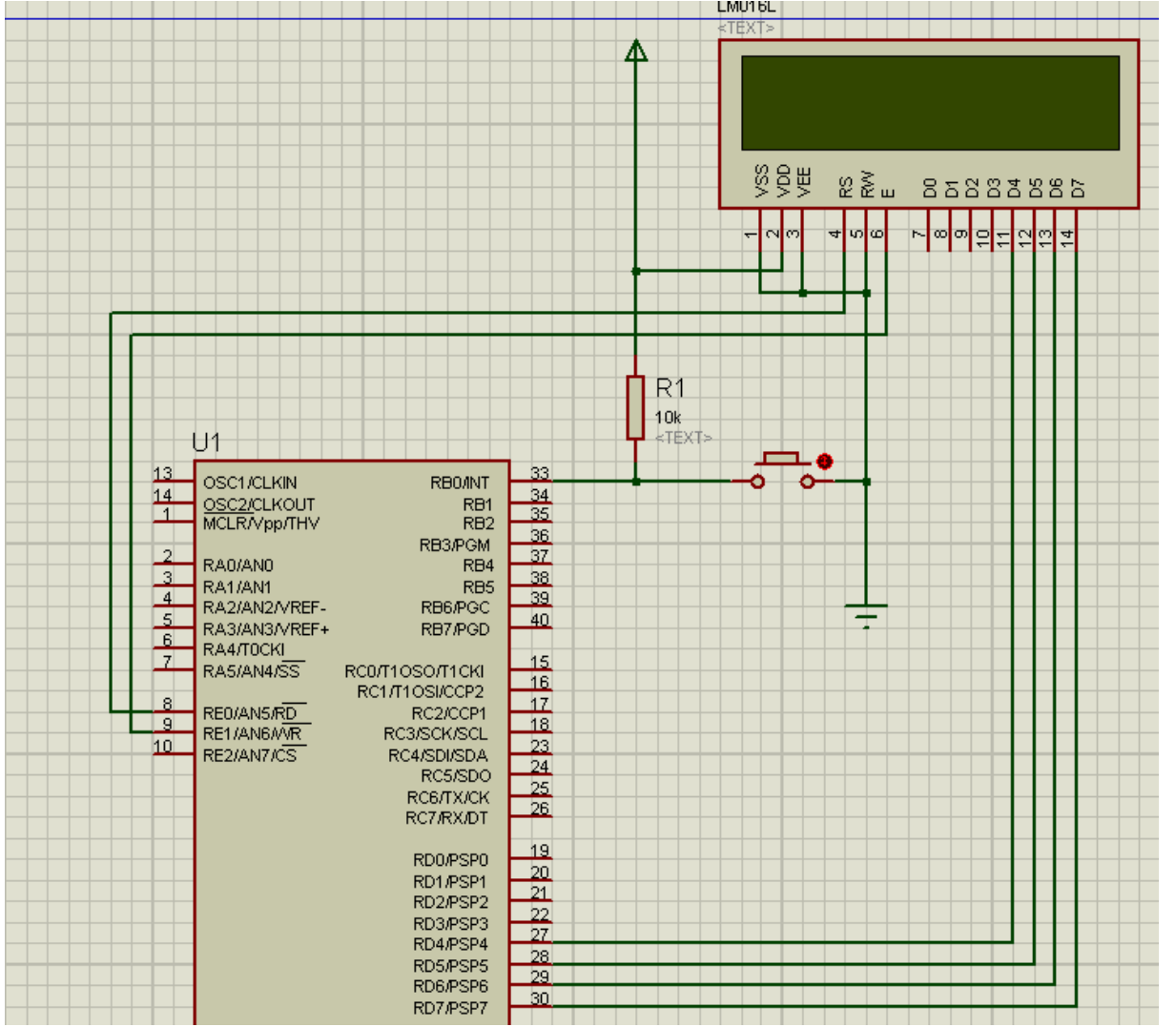
Device = 16F877
'--LCD BAGLANTILARI-----
XTAL = 4
LCD_DTPIN = PORTD.4           'Lcd nin dataları Portd.4 en başlıyor
LCD_RSPIN = PORTE.0          'LCD RS pin Porte.0
LCD_ENPIN = PORTE.1          'Lcd EN pin Porte.1
LCD_INTERFACE = 4            '4-bit Interface
LCD_LINES = 2                'Lcd 2 satır
LCD_TYPE = 0
LCD_COMMANDUS = 2000
LCD_DATAUS = 50
'--LCD BAGLANTILARI-----
ALL_DIGITAL TRUE           '16f877 nin tüm uçları Dijital
Symbol INTF = INTCON.1      'RBO External Interrupt Flag
Symbol INTE = INTCON.4      'RBO External Interrupt Enable
Symbol GIE = INTCON.7      'Global Interrupt Enable
INTE=1                       'RBO Kesmesi aktif
GIE=1                         'Aktif Olan Tüm kesmelere Yetki ver
Input PORTB.0                'portb.0 giriş
Dim Menu_Durum As Bit        'Menu_Durum adında Değişken (bit)
Print At 1,1,"Menu"          'LCD nin 1. satırının 1 Sütunundan
                              'Başlayarak MENU yaz
On Interrupt GoTo INT        'Interrupt Oluşunca INT ye git

basla:
Select Menu_Durum            'Menu_Durum u Kontrol et
Case 0                        'Eğer 0 sa
Print At 1,6,"1"             'LCD nin 1 satır 6 sütununa "1" yaz
Case 1                        'Eğer 1 se
Print At 1,6,"2"             'LCD nin 1 satır 6 sütununa "2" yaz
EndSelect                    'Select ten çık
DelayMS 10                   'Delayms 10
GoTo basla                   'Basla ya dön
Disable                       'Yüm int. Kapalı

INT:
If Menu_Durum=0 Then         'Menu_durum=0 sa
Menu_Durum=1                 'Menu_Durum=1
Else                           'Menu_durum=0 değilse
Menu_Durum=0                 'Menu_durum u 0 ya
EndIf                          'if sonu
INTF=0                        'oluşan int flag'ini temizle
Resume                         'ana programa dön
Enable                         'tüm int açık
End                            'program sonu

```

ISIS Simulasyon devresi



(Devrenin Çalışmasını Proje7 altındaki proje7.avi dosyasından izleyebilirsiniz.)

RB0 kesmesinin çalışmasından bahsedecek olursak, Deveredende anlaşılacağı üzere Portb nin 0 numaralı pinine girilen bir tetikleme ile çalışır. Interrupt alt programına gitmek için tetikleme sinyalinin düşen kenar yada yükselen kenar seçimi için OPTION_REG in 6 biti kullanılır.(Default olarak düşen kenardır). Interrupt alt programında menu_durumu kontrol edilir ve durum değişmesi sağlanır.(Burada Toggle komutunun kullanılmamasının sebebi Toggle komutunun sadece port pininin durumunu değiştirmesidir.)

Buraya kadar anlatılan kesme türleri Software kesme olarak adlandırılan kesme türüdür. Bu kesme türü ile yapılan çalışmalarda bazı sakıncalar oluşabilir. Bunlardan en önemlisi komut işleme esnasında oluşan kesmeyi algılayamamaktır. Burada hardware kesme önem kazanmaktadır. Software kesme derleyici tarafından her komutun önüne "kesme oluşumu" komutu atamak gibidir. Bundan dolayı uzun bekleme komutlarında ("*DelayMs 1000*" gibi) bekleme komutu icra edilmeden kesme oluşmaz. Hardware kesmesinde ise böyle bir problem yaşanmaz.

Hardware kesmesinde ise, kesme altprogramına girer girmez sistemi etkileyen kayıtçı durumlarının yedeklenmesi ve kesmeden çıkarken bunların tekrardan yüklenmesi gerekir. Yedekleme işlemini **Context Save**, geri alma işlemini **Context Restore** komutları yapar.

Hardware Kesmesi için intcon kayıtçısının ayarlanması için yapılması gereken önemli hususlardan biri de Ayar yapmaya başlamadan önce Global Int bayrağının "0" yapılması gerekliliğidir. **Ayrıca kesme altprogramı içerisinde mümkün olan en kısa zamanda çıkılması gerekir. Diğer uzun beklemelerde ana program çalışmayabilir.**

Proje 8;

Amaç: RBO ile dış etki ile **Hardware** interrupt çalışması ve 2x16 Alpha LCD Menü Çalışması.

Kullanılan Yeni Komutlar;

Repeat..Until: Komut ile until ile bildirilen şart gerçekleşene kadar repeat ile until arasındaki komutların işlenmesi sağlanır. Komutun **for..next** ile kullanım açısından pek bir farkı olmamakla beraber repeat..until komutu for..next komutundan daha hızlı çalışır ve daha az yer kaplar.

```
Repeat
  Print At 2, 1, @Durum, " "
  DelayMS 100
  Inc Durum
Until Durum > 255
```

Yukarıdaki komut satırı ile sonsuz bir döngüye girilmiştir. Çünkü durum değişkeni byte olarak tanımlandığından hiçbir zaman desimal 255 den daha büyük değer alamaz.

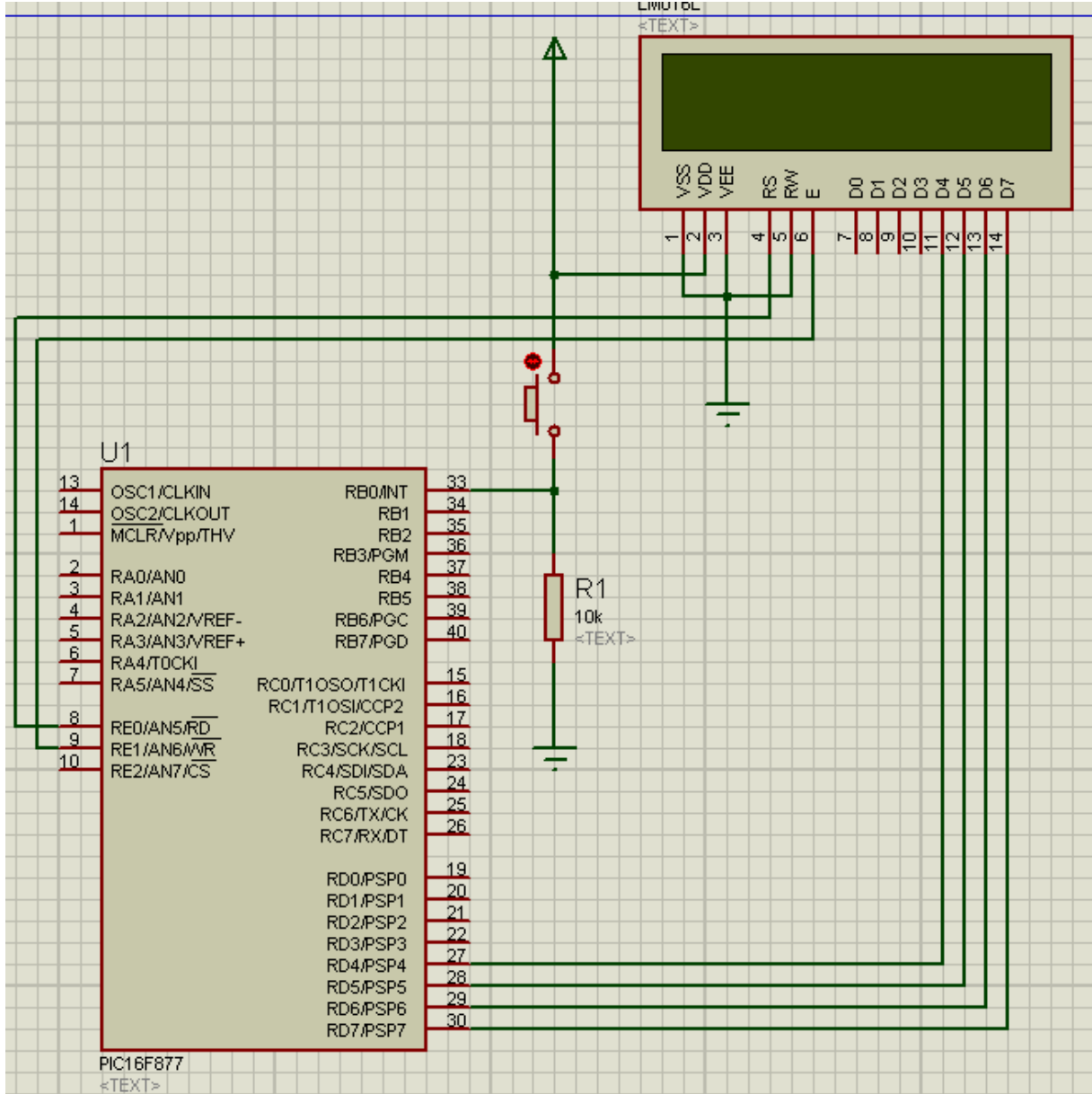
```
Include "proton_4.inc" 'Tanımlamalar
On Hardware Interrupt GoTo Kesme 'Hardware Interrupt Oluşunca INT ye git
Symbol INTF = INTCON.1 'RBO KESME BARAĞI
Symbol INTE = INTCON.4 'RBO KESME YETKİ
Symbol GIE = INTCON.7 'GLOBAL INT. YETKİ
GIE=0 'tüm kesmeler kapalı
INTE=1 'RBO Kesmesi aktif
GIE=1 'Aktif 0 an Tüm kesmelere Yetki ver
Input PORTB 'portb giriş
Dim Sayı As Byte 'Sayı adında byte değişken
Dim Durum As Byte 'Durum adında byte değişken
Print At 1,1,"KESME " 'LCD nin 1. satırının 1 Sütunundan Başlayarak "KESME " yaz

basla: 'Başla Etiket
Repeat '\
  Print At 2,1,@Durum," " '2 satırı 1 sütununa Durum un içeriğini yaz
  DelayMS 100 '100 ms bekle
  Inc Durum
Until Durum > 255 '/durum 255 den büyükse alt satıra geç değilse repeat dön
GoTo basla 'Basla ya dön

Kesme: 'kesme altprogramı
Context SAVE 'kayıcılarının içeriğini sakla
INTF=0 'oluşan kesme bayrağını 0 la
Print At 1,1,"KESME ",@Sayı 'lcd ye "kesme" ve sayı değişkeninin içeriğini yaz
Inc Sayı 'sayı değişkenini bir arttır
DelayUS 10 '10us bekle
Context Restore 'tüm kayıtçıların içeriğini tekrar yükle ve kesmeden çık
Stop 'program sonu
```

İsis şeması aşağıdaki gibidir. Çalışmasına gelince **On_Hardware_Interrupt GoTo** Kesme komutuna dikkat edecek olursak kırmızı olarak renginin değişmesi bu kesme türünün hardware olduğu anlamına gelir. Lcd nin ilk satırında kesmenin oluşma (butona basma) sayısını (Sayı değişkeni), ikinci satırda ise normal programın çalışması (Repeat Until) esnasında oluşan durum değişkeninin aldığı değerler yer alır. Butona her bastığımızda kesme sayısı bir artacak ve ana programa tekrar dönecektir.

Programın çalışmasını proje8 klasörünün içerisinde bulabilirsiniz.



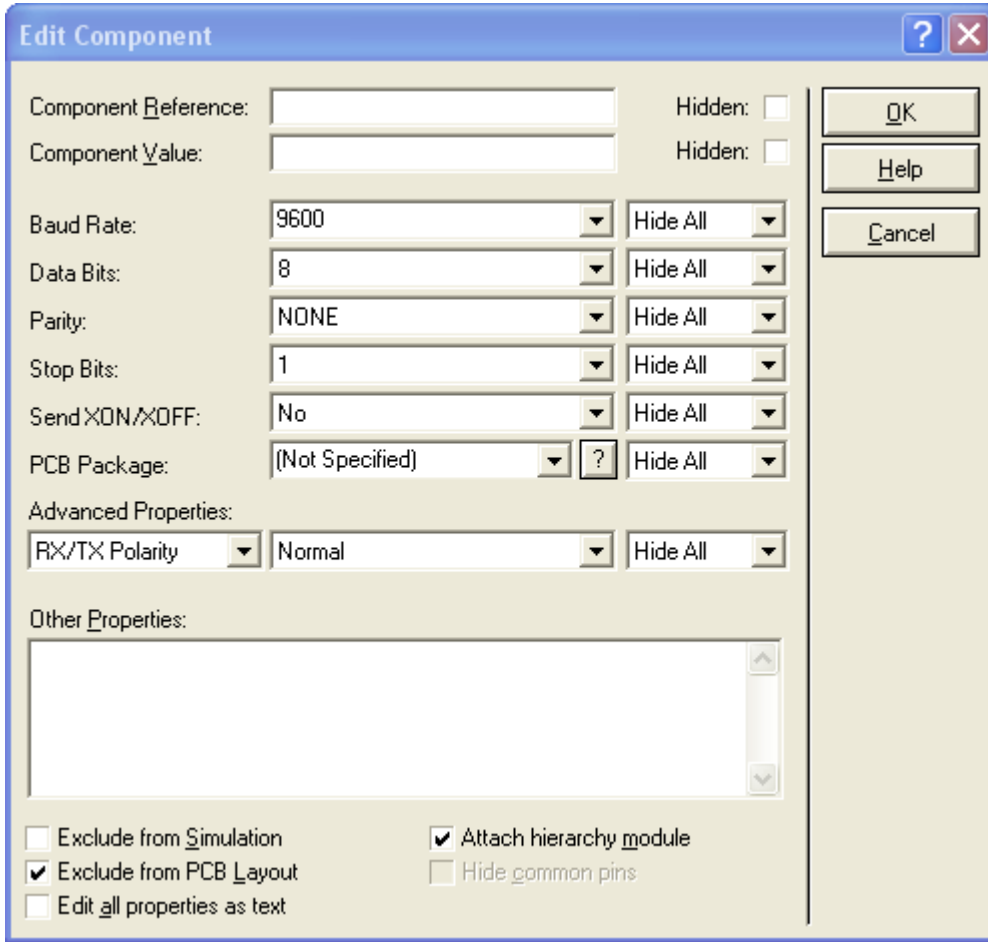
Proje 9;

Amaç: Seri iletişim kesme çalışması (Hardware kesme kullanarak Çevresel birimlerle ilgili kesme çalışması)

Seri İletişim: Seri iletişim iki cihaz arasında bilgi alışverişi yapmak için kullanılan yöntemlerden bir tanesidir. Bu yöntemin avantajları arasında daha az iletişim hattı kullanılması, dezavantajları arasında ise diğer yöntemlere göre daha yavaş olması, söylenebilir. Birçok bilgisayarda olan ve bir standart haline gelen RS232 yöntemi seri iletişime bir örnektir.

Yalnız, iletişime başlamadan önce her iki cihazında ayarlanması gereken bazı parametreler vardır. İletişim hızı (BaudRate), veri sayısı (Databits), eşlik bit çeşidi (Parity), başlangıç ve bitiş bitlerinin olup olmaması bunlardan bazılarıdır.

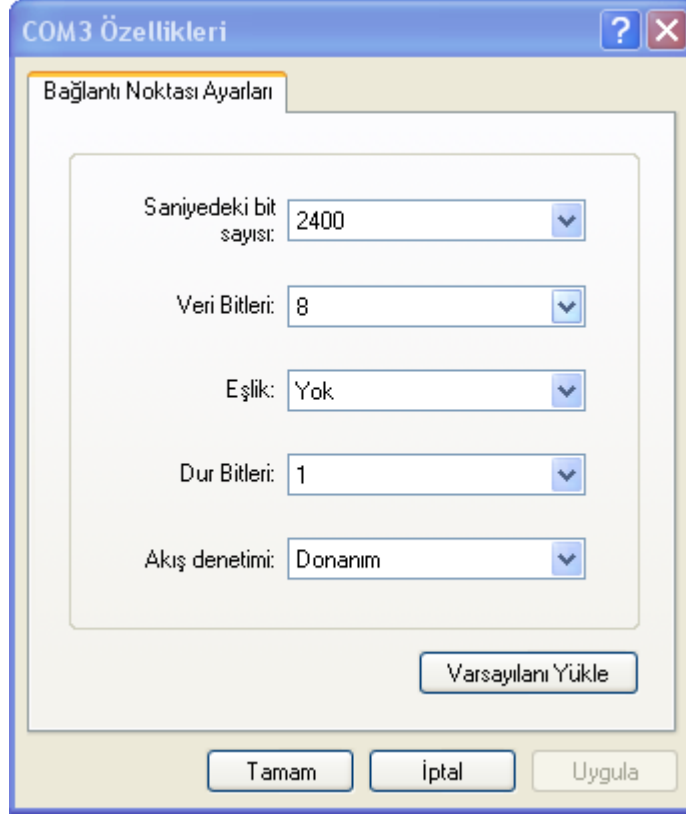
Aşağıda ISIS de yer alan virtual terminal in ayar penceresi görülmektedir.



The screenshot shows the 'Edit Component' dialog box in ISIS. The dialog has a title bar with a question mark and a close button. The main area contains several fields and dropdown menus for configuring serial communication parameters. The 'Baud Rate' is set to 9600, 'Data Bits' to 8, 'Parity' to NONE, 'Stop Bits' to 1, and 'Send XON/XOFF' to No. The 'PCB Package' is set to '(Not Specified)'. Under 'Advanced Properties', 'RX/TX Polarity' is set to 'Normal'. There is an empty text area for 'Other Properties'. At the bottom, there are four checkboxes: 'Exclude from Simulation' (unchecked), 'Attach hierarchy module' (checked), 'Exclude from PCB Layout' (checked), and 'Hide common pins' (unchecked). On the right side, there are three buttons: 'OK', 'Help', and 'Cancel'.

Yukarıdaki ayarlama da görüldüğü üzere 9600bps, 8bitdata, NoneParity, 1 Stop bit olarak ayarlanmıştır.(Kısaca 9600N81Olarak geçer)

Projeyi devre üzerinde hazırladığımızda ise, PC ile iletişim kurması için Windows da bulunan HyperTerminal programı kullanılabilir. Bazı PC lerde birden daha çok seri iletişim portu olduğundan dolayı ilk yapacağımız, hazırladığımız MCU ile iletişim yapacak PC portunu seçmek olacaktır. Aşağıdaki şekilde HyperTerminal programının ayar penceresi görülmektedir.



Kullanılan Yeni Komutlar;

HSerIn: Komut ile USART (Universal Synchronous Asynchronous Receiver Transmitter) arabirimi kullanarak (*bu arabirim donanımsal olarak PIC içerisinde bulunmalı*) seri bilgi alışı sağlanır. Komutun kullanımı aşağıdaki gibidir.

HSERIN, 1000, Timeout, [DEĞİSKEN]

Komutta kullanılan ilk değer (1000) zaman aşımı değeridir. Bu değer milisaniye birimindedir. Eğer seri bilgi alımında bir değer aşımı olursa "TimeOut" etiketine daller. Alınan bilgi "Değişken" in içerisine aktarılır.

Aşağıdaki tabloda Proton Plus ile kullanabileceğimiz Seri iletişim protokolü ile ilgili komutları bulabilirsiniz.

Usart Gereklimi ?	Komut	Açıklama
Evet	Hserin	USART birimi olan MCU lar için seri bilgi alır HSERIN Timeout , Timeout Label , Parity Error Label , [Modifiers , Variable { , Variable... }]
Evet	Hserout	USART birimi olan MCU lar için seri bilgi gönderir HSEROUT [Item { , Item... }]
Evet	Hserin2	İki Adet Usart modülü olan MCU larda ikincil modül için Hserin ile aynı HSERIN2 Timeout , Timeout Label , Parity Error Label , [Modifiers , Variable { , Variable... }]
Evet	Hserout2	İki Adet Usart modülü olan MCU larda ikincil modül için Hserout ile aynı HSEROUT2 [Item { , Item... }]
Evet	Hrsin	USART birimi olan MCU lar için seri bilgi alır Variable = HRSIN , { Timeout , Timeout Label }
Evet	HrsOut	USART birimi olan MCU lar için seri bilgi gönderir HRSOUT Item { , Item... }
Evet	Hrsin2	İki Adet Usart modülü olan MCU larda ikincil modül için Hrsin ile aynı Variable = HRSIN2 , { Timeout , Timeout Label }
Evet	HrsOut2	İki Adet Usart modülü olan MCU larda ikincil modül için Hrsout ile aynı HRSOUT2 Item { , Item... }
Hayır	Rsln	Sabit Gönderim hızı ve Pin İle Asenkron Bilgi Gönderir Variable = RSIN , { Timeout Label }
Hayır	RsOut	Sabit Gönderim hızı ve Pin İle Asenkron Bilgi alır RSOUT Item { , Item... }
Hayır	Serin	Asenkron Seri Bilgi Alır SERIN Rpin { \ Fpin } , Baudmode , { Plabel, } { Timeout , Tlabel, } [InputData]
Hayır	SerOut	Asenkron Seri Bilgi Gönderir SEROUT Tpin { \ Fpin } , Baudmode , { Pace, } { Timeout , Tlabel, } [OutputData]
Hayır	ShIn	Senkron Seri Bilgi Alır Var = SHIN dpin , cpin , mode , shifts
Hayır	ShOut	Senkron Seri Bilgi Gönderir SHOUT Dpin,Cpin,Mode, [OutputData{\Bits}{,OutputData{\Bits}..}]

Komutları verdikten sonra, 16F877 de kullanılan ve çevresel birimlerden gelen kesmeleri koordine eden kayıtlardan biri olan PIE1 (Pripheral Interrupt Enable) kayıtcısını biraz inceleyelim.

7	6	5	4	3	2	1	0
PSP1E	AD1E	RC1E	TX1E	SS1E	CCP1E	TMR2E	TMR1E

Yukarıdaki şekilde görüldüğü üzere bu kayıtcı kesmelere yetki vermek için kullanılır

Bit no	Adı	Açıklama
0	TMR1E	TMR1 sayıcısının Kesme Yetki Biti
1	TMR2E	TMR2 kesme yetki biti
2	CCP1E	Capture/Compare/PWM kesme yetki biti
3	SS1E	Senkron Seri kesme yetki biti
4	TX1E	USART Gönderme (TX) kesme yetki biti
5	RC1E	USART Alma (RX) kesme yetki biti
6	AD1E	A/D çevirici kesme yetki biti
7	PSP1E	Paralel port (PSP-PordD)Yazma Okuma Kesme Yetki Biti

Kesme oluştuğunda ise **PIR1** kayıtcısının ilgili biti set olarak kesmenin oluştuğunu belirtir. Kesmeden çıkmadan önce set olan bu değerin temizlemesi gerekir($RC1IF = 0$).

Aşağıda PIR1 kayıtcısının yapısı görülmektedir.

7	6	5	4	3	2	1	0
PSP1F	AD1F	RC1F	TX1F	SS1F	CCP1F	TMR2F	TMR1F

Bit no	Adı	Açıklama
0	TMR1F	TMR1 sayıcısının "Kesme Oluştur" bayrağı
1	TMR2F	TMR2 "Kesme Oluştur" bayrağı
2	CCP1F	Capture/Compare/PWM "Kesme Oluştur" bayrağı
3	SS1F	Senkron Seri "Kesme Oluştur" bayrağı
4	TX1F	USART Gönderme (TX) "Kesme Oluştur" bayrağı
5	RC1F	USART Alma (RX) "Kesme Oluştur" bayrağı
6	AD1F	A/D çevirici "Kesme Oluştur" bayrağı
7	PSP1F	Paralel port (PSP-PordD)Yazma Okuma "Kesme Oluştur" bayrağı

```

Device = 16F877
XTAL = 4
on_interrupt GoTo kesme      'hardware kesmesi
ADCON1=7                    'TÜM GİRİŞLER DİJİTAL
HSERIAL_BAUD = 9600         ' Seri iletişim hızı 9600 bPs
HSERIAL_RCSTA = %10010000  ' seriport açık ve alıma her zaman açık
HSERIAL_CLEAR = On         ' bilgi alındığında Tampon belleği temizle
LCD_DTPIN = PORTD.4
LCD_RSPIN = PORTE.0
LCD_ENPIN = PORTE.1
LCD_INTERFACE = 4          ' 4-bit Interface
LCD_LINES = 2
LCD_TYPE = 0
LCD_COMMANDUS = 2000
LCD_DATAUS = 50
Symbol PEIE = INTCON.6     ' Peripheral Interrupt Enable
Symbol GIE = INTCON.7     ' Global Interrupt Enable
Symbol RCIE = PIE1.5      ' USART Receive Interrupt yetki
Symbol RCIF = PIR1.5     ' USART Receive Interrupt bayrağı
GIE=0                      'tüm kesmeler kapalı
PEIE=1                    'Çevre birim kesmesi aktif
RCIE=1                    'Usart alım kesmesi aktif
GIE=1                    'tüm kesmeler aktif
Dim satir As Byte        'satir adında byte değişken
Dim Dongu As Byte       'donu adında byte değişken
Dim GelenBilgi As Byte
satir=1

basla:                    'ana program başlangıç
For Dongu=1 To 16
Print At 2,Dongu,"#"     '2 satırın "Dongu" sütünuna "#" yaz
DelayMS 100             '100ms bekle
Next
For Dongu=1 To 16
Print At 2,Dongu," "    '2 satırın "Dongu" sütünuna " " yaz
DelayMS 100
Next
GoTo basla

hata:
Print At 1,satir,"Hata"
Return

kesme:                   'kesme alt programı
Context SAVE            'kayıcılarının içeriğini sakla
HSerIn 1000,hata,[GelenBilgi] 'Hardware Usart Kullanarak bilgiyi al
'"gelenbilgi" 'değişkeninin içeriğine at eğer timeout olursa (1000 ms) "Hata" ya dallan

Print At 1,1,"Gelen Deger ",Dec GelenBilgi 'gelen bilginin değerini yazdır
RCIF = 0                'USART Receive interrupt Flag Temizleniyor
Context Restore        'tüm kiyıtlı reg. tekrar yüklenip keseden çıkılıyor
End                    'program sonu

```

Projenin çalışması için söylenecek olan, ISIS de var olan ve seri iletişim için kullanılan Virtual terminal aracı ile, kullandığımız 16f877 işlemcisine seri bilgi gönderiyoruz. Gönderilen bilginin değerini birinci satırda, ana programın normal olarak çalıştığını sembolize eden, ve soldan sağa doğru hareket eden “#” sembolünü ikinci satırda görürüz.(Projenin Çalışması ile ilgili video proje9 klasörünün içerisinde.)

Her bilgi gönderdiğimizde “kesme:” alt programına atlayarak daha önce ayarladığımız hızda (9600bps) aldığı bilgiyi [GelenBilgi] Değişkenine atar. Daha sonra

```
"Print At 1,1,"Gelen Deger ",Dec GelenBilgi"
```

ile gelenbilgi değerinin Decimal karşılığını ekranda yazdırır. (Devrenin ISIS şeması aşağıdaki gibidir. ISIS Proje dosyasını proje9 klasörünün altında bulabilirsiniz.)

