KONTROL VE OTOMASYON KULÜBÜ

C DİLİ İLE MİKROKONTROLÖR PROGRAMLAMA EĞİTİMİ

Serhat Büyükçolak
Ahmet Sakallı

2009-2010 Güz Dönemi Eğitimleri

Mikrokontrolör

Gömülü sistemlerin bir alt dalı olan mikrokontrolör tabanlı sistemler öncelikle çok geniş kullanım alanına sahiptir. Doğru elektronik donanımlarla kullanıldığında mikrokontrolör uygulamalarının başta kontrol mühendisliği olmak üzere, elektronik mühendisliği, bilgisayar mühendisliği, telekomünikasyon mühendisliği hatta elektrik mühendisliği gibi bir çok meslek dalında çok önemli uygulama alanlarının olduğu görülür.

Mikrokontrolör sistemler bu gün dijital kontrol uygulamalarının yoğun gerçekleştirildiği düzeneklerdir. Yine bir çok kontrol uygulamalarında kontrolör olarak seçilen elektronik donanım mikrokontrolörler olmuştur. Bu bakımdan mikrokontrolör bilmenin kontrol mühendisine getireceği avantajlar şüphesiz ki oldukça büyüktür.

Yine Elektrik – Elektronik Fakültesi içindeki diğer bölümler de mikrokontrolör sistemlerini çok farklı alanlarda kullanmaktalar. Bu özellikleri, mikrokontrolörleri çok önemli ve hakim olunması gereken donanımlar haline getiriyor.

Neden C dili?

Uygulamalarda mikrodenetleyiciler C dili ile, assembly dili ile basic dili ile ve daha bir çok dil ile programlanmaktadır. Bu dillerin her birinin muhakkak ki birtakım avantajları ve dezavantajları bulunmaktadır. Bu diller içerisinden C dilini seçmemizin nedeni, bu dilin en yaygın kullanılan programlama dili olması, yapısal programlamaya uygun olması, akış denetimlerini daha kolay ve etkin sağlayabilmesi ve gerek kütüklere gerek bayraklara erişmede zorluk çıkarmaması gibi bir çok nedene bağlanabilir. Yine okulumuzda mikrokontrolörler ile hazırlanan kontrol sistemlerinde ağırlıklı olarak C dilinin kullanılması, kulübümüzün öz görevlerini takip etmekteki istekliliğimiz de bu eğitimde ağırlıklı olarak C dili kullanmak istememizin nedenleri arasındadır.

Giriş ve Önemli Notlar

Bu eğitimde basit anlamda mikrokontrolör programlama ve microchip firmasının ürettiği pic16f84,pic168f77 ve pic16f628 kodlu mikrokontrolörler üzerinden anlatım yapılacaktır.C dili ile nasıl programlanacağına, hangi programların kullanılacağına ilişkin bilgi verilecek ve HI-TECH PIC C anlatılacaktır.

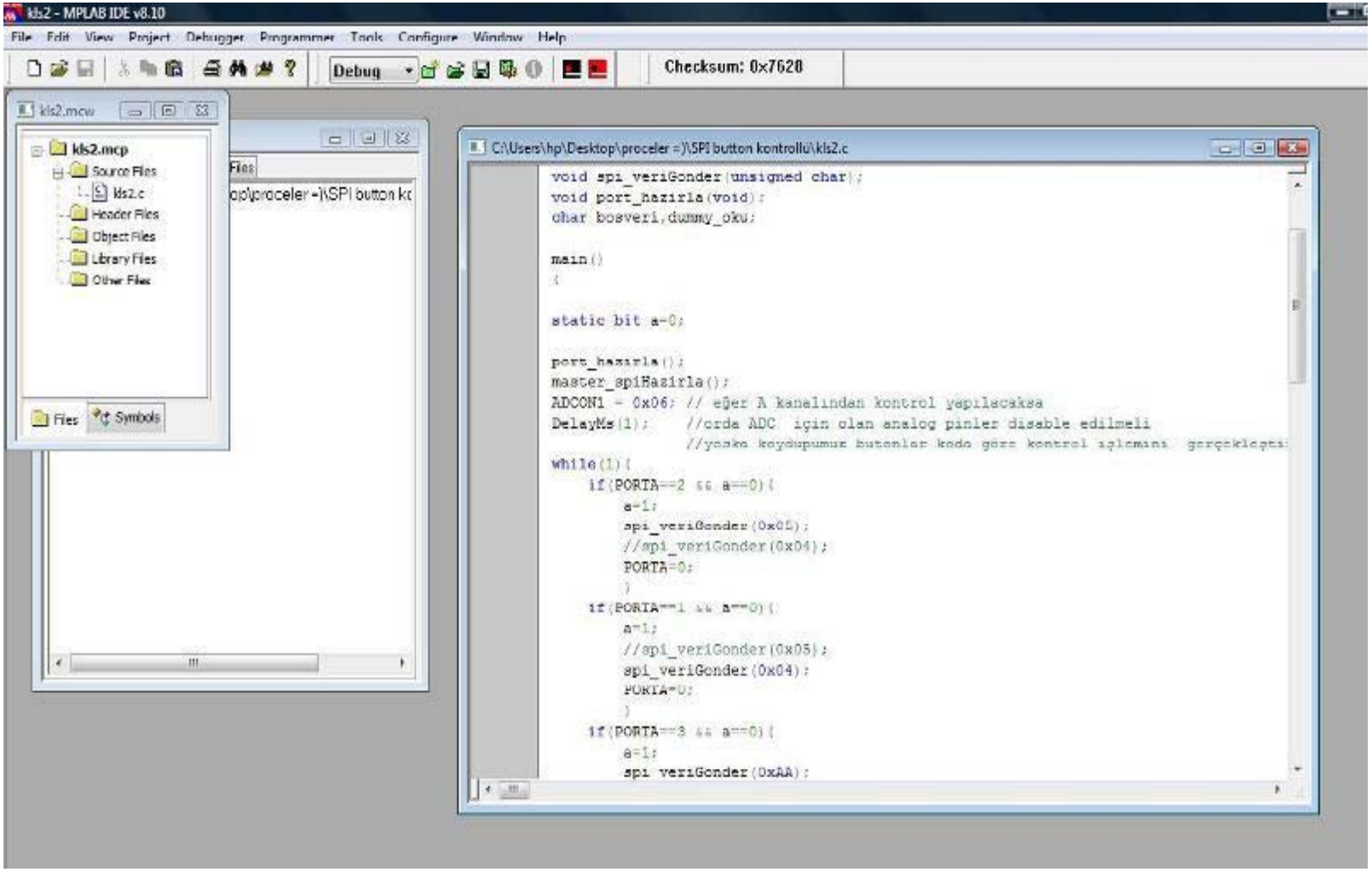
Bu eğitime katılan arkadaşların "**C programlama dili"** konusunda bilgi sahibi oldukları varsayılacaktır.

Eğitimde kullanılacak temel programlar;

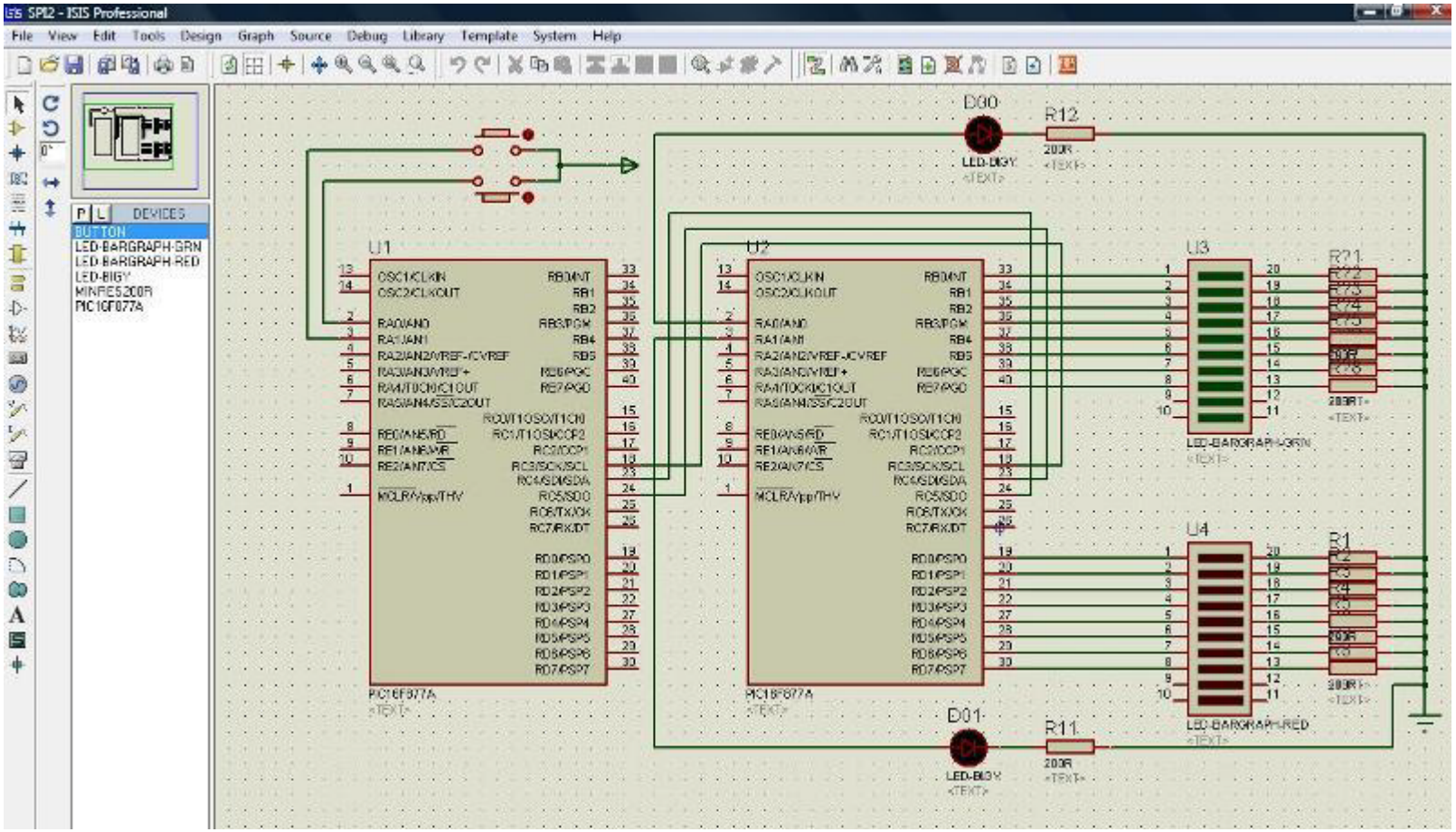
MPLAB IDE →Mikrodenetleyiciye C kodu yazarken kullanılacak program

Proteus ISIS → Kodumuzu yazdık denemeden olmaz, Simülasyonuda buradan yapacağız

MPLAB :


kls2 - MPLAB IDE v8.10
File Edit View Project Debugger Programmer Tools Configure Window Help
Debug
Checksum: 0x7620
kls2.mcw
kls2.mcp
Source Files
kls2.c
Header Files
Object Files
Library Files
Other Files
Files
Symbols
C:\Users\hp\Desktop\proceler =)\SPI button kontrollu\kls2.c
void spi_veriGonder(unsigned char);
void port_hazirla(void);
char bosveri,dummy_oku;
main()
{
static bit a=0;
port_hazirla();
master_spiHazirla();
ADCON1 = 0x06; // eğer A kanalından kontrol yapılacaksa
DelayMs(1);
while(1){
if(PORTA==2 && a==0){
a=1;
spi_veriGonder(0x05);
//spi_veriGonder(0x04);
PORTA=0;
}
if(PORTA==1 && a==0){
a=1;
//spi_veriGonder(0x05);
spi_veriGonder(0x04);
PORTA=0;
}
if(PORTA==3 && a==0){
a=1;
spi_veriGonder(0xAA);

PROTEUS :


ISIS SPI2 - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
BUTTON
LED-BARGRAPH-GRN
LED-BARGRAPH-RED
LED-BIGY
MINRES200R
PIC16F877A
U1
U2
U3
U4
D00
D01
R12
R11
LED-BARGRAPH-GRN
LED-BARGRAPH-RED

C Temelleri

C dilinde değişkenler: (HITECH Compiler için)

Veri Çeşidi	Açıklama
bit	Bir bitlik veri taşır. 0 ya da 1 olabilir.
char	8 bitlik veri taşır. -127 ve +127 arasında değerler alabilir.
unsigned char	8 bitlik veri taşır. 0 ve +255 arasında değerler alabilir.
int	16 bitlik veri taşır. -32767 ve +32767 arasında değerler alabilir.
unsigned int	16 bitlik veri taşır. 0 ve 65535 arasında değerler alabilir.
long	32 bitlik veri taşır. -2147483647 ve +2147483647 arasında değerler alabilir
unsigned long	32 bitlik veri taşır. 0 ve +4294967295 arasında değerler alabilir.
float	24 veya 32 bitlik veri taşır.
double	24 veya 32 bitlik veri taşır.

C Temelleri

Değişkenler mikrokontrolörde RAM'de tutulur. Ancak buradaki kaynakları etkin kullanmak gerektiğinden eğer değişkenimiz program boyunca değişmeyecekse tanımlamanın başına "**const"** ekleyerek bu değişkeni EEPROM ya da Flash bellekte tutmak yerinde olacaktır.

Fonksiyon her çağırıldığında değerinin değişmesini istemediğimiz değişkenler için yine tanımın başına **"static"** ekliyoruz.

Eğer resetleme esnasında bir değişkenimizin değerinin değişmesini istemiyorsak tanımın başına **"persistent"** yazmamız yeterli olacaktır.

Bir değişken sadece belli bir kod tarafından değil de bir kesme alt programı ya da donanım tarafından değiştirilebiliyorsa değişkenimizin başına **"volatile"** anahtar kelimesini ekleriz. Özellikle kütük adresleri tanımlanırken bu anahtar kelimenin kullanılması önemlidir.

Operatörler:

Operatör	Açıklama
() []	Parantez
!	Mantıksal değil
~	Bit tersi
*	İşaretçi operatörü
+ - * /	Aritmetik operatörler
%	Modül
++	1 artır
--	1 eksilt
&	Adres (önde kullanılırsa)
&	Ve operatörü (iki değerin “ve”lenmiş halini döndürür)
|	Veya operatörü (iki değerin “veya”lanmış halini döndürür)
^	Xor operatörü (iki değerin “xor”lanmış halini döndürür)
<< >>	Kaydırma operatörleri
>= <= > <	Karşılaştırma operatörleri
sizeof	Boyut bulma operatörü
==	Mantıksal eşittir
!=	Mantıksal eşit değil
||	Mantıksal veya
&&	Mantıksal ve
=	Eşitleme operatörü
+= -= *= /=	Eşitlik operatörleri
&= |= ^=	Eşitlik operatörleri
%= >>= <<=	Eşitlik operatörleri

C Temelleri

Program Akış Kontrolü:

```
1.  if (yargı) {
2.  işlemler;
3.  }
4.  else if (yargı) {
5.  işlemler;
6.  }
7.  else {
8.  işlemler;
9.  }
```

```
    switch (yargı) {
1.  case değer1:
2.  işlemler;
3.  break;
4.  case değer2:
5.  işlemler;
6.  break;
7.  default:
8.  işlemler;
9.  }
10
 .
11
 .
12
 .
```

C Temelleri

Döngüler:

While döngüsü:

```
1. while (yargı) {
2. işlemler;
3. }
```

For döngüsü:

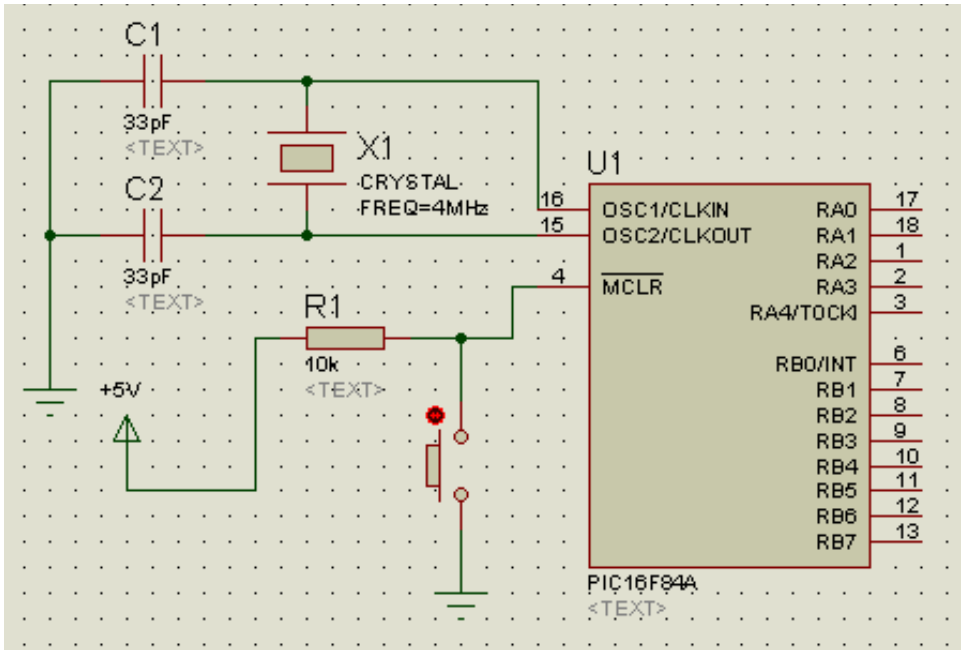
```
1. for (başlangıç işlemleri; yargı; döngü işlemleri) {
2. işlemler;
3. }
```

Ön işlemci tanımlamaları:

```
1. #define değişkenadı değer
```

PIC

Temel PIC bağlantı şekli:



* Simülasyonda OSC bağlantılarını ihtiyaç duymadığımız için yapmayacağız. Bu bağlantıya gerçekleme işlemini yaparken ihtiyaç duyacağız.

PIC

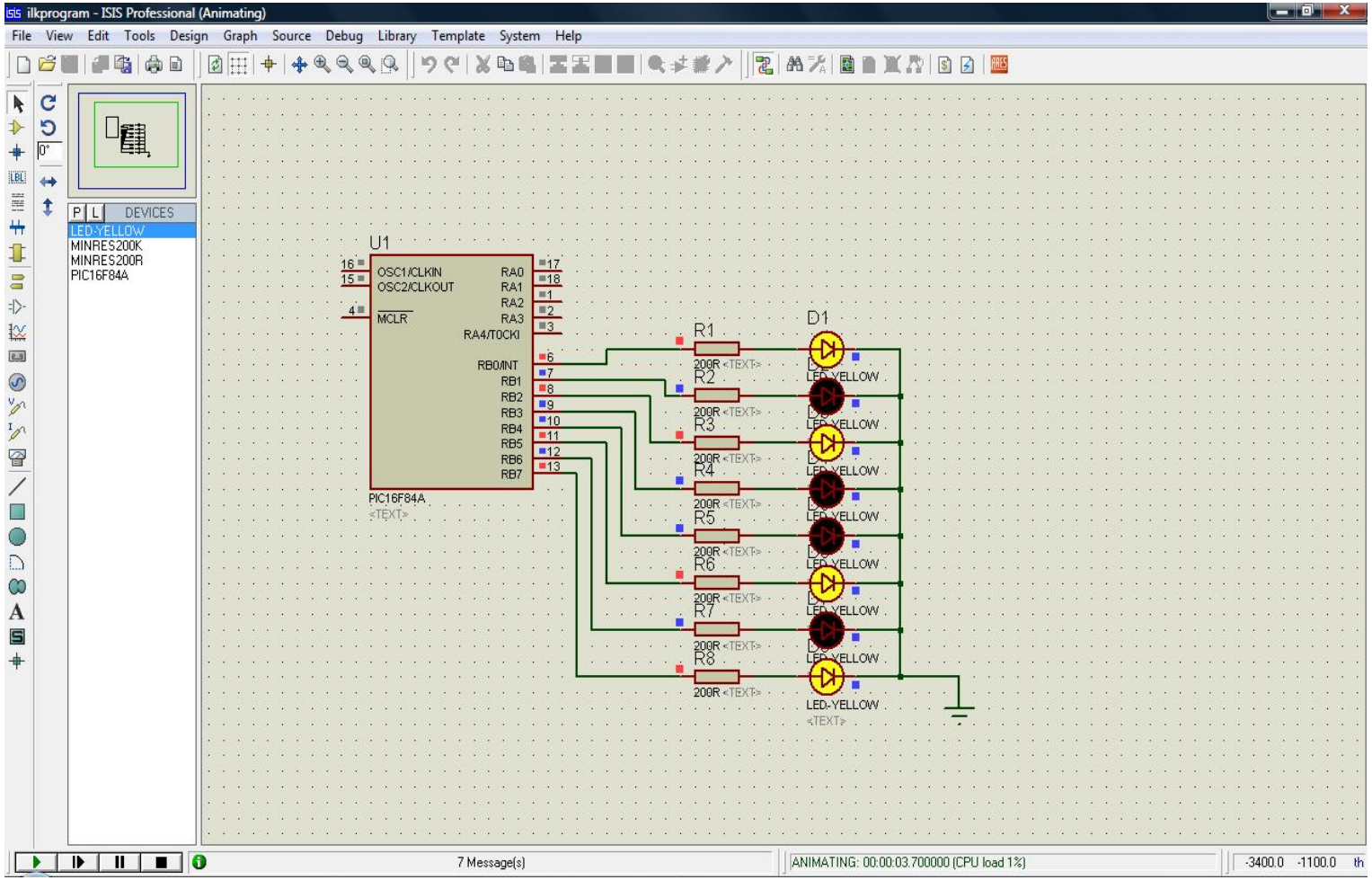
İlk PIC programı

İşin geyik kısmını sonlandırıp ilk programımıza giriş yapalım

```
1   #include<pic.h>
2
3   main()
4   {
5
6   TRISB=0; //B Portlarımızı çıkış yaptık
7
8   PORTB=0xA5;     //Hexadecimal tabanda bazı portlardan çıkış  verdirdik
9                   //PORTB = 0b10100101 aynı şeyi ifade ederdi.
10
11  while(1);       //Kodumuzu sonsuz döngüye soktuk.
12  }
```

PIC

Buda simülasyonu:



PIC

Şimdi gelelim kafaların karıştığı noktalara.TRIS, PORT neymiş neyden çıktı bunlar??!!
Cevabımız iki kelime: **“DATA SHEET”** yani **“VERİ KAĞIDI”**

DATASHEET leri kullanma kılavuzu olarak da adlandırabiliriz.Eğitimimizin bir amacıda insanlara datasheet kullanmayı öğretebilmektir.Çünkü temel bilgileri kazandıktan sonra datasheeti açıp yazacaksınız.İhtiyaç duyacağınız her şey adım adım orada yazıyor olacak.

İkinci problemde genelde şudur;”İyi hoş ama biz nerden bulucaz bu data sheetleri…”
Onunda cevabı kolay buyurun size bazı datasheet siteleri;

http://www.datasheetcatalog.com/

http://www.alldatasheet.com/

Burada arama çubuğuna istediğimiz elemanı yazarak ilgili datasheeti bulabiliriz.(Örneğin Pic16f84a vs.)

Olmadı mı? Buyrun 2. Çözüm: ***“GOOGLE”***

PIC

Data Sheet:

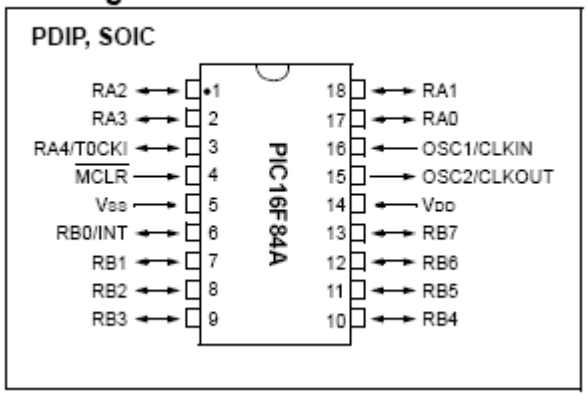
MICROCHIP

PIC16F84A
Data Sheet

18-pin Enhanced FLASH/EEPROM
8-bit Microcontroller

Pin Diagrams

PDIP, SOIC



TER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INT

bit 7

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in softwa
0 = None of the RB7:RB4 pins have changed state

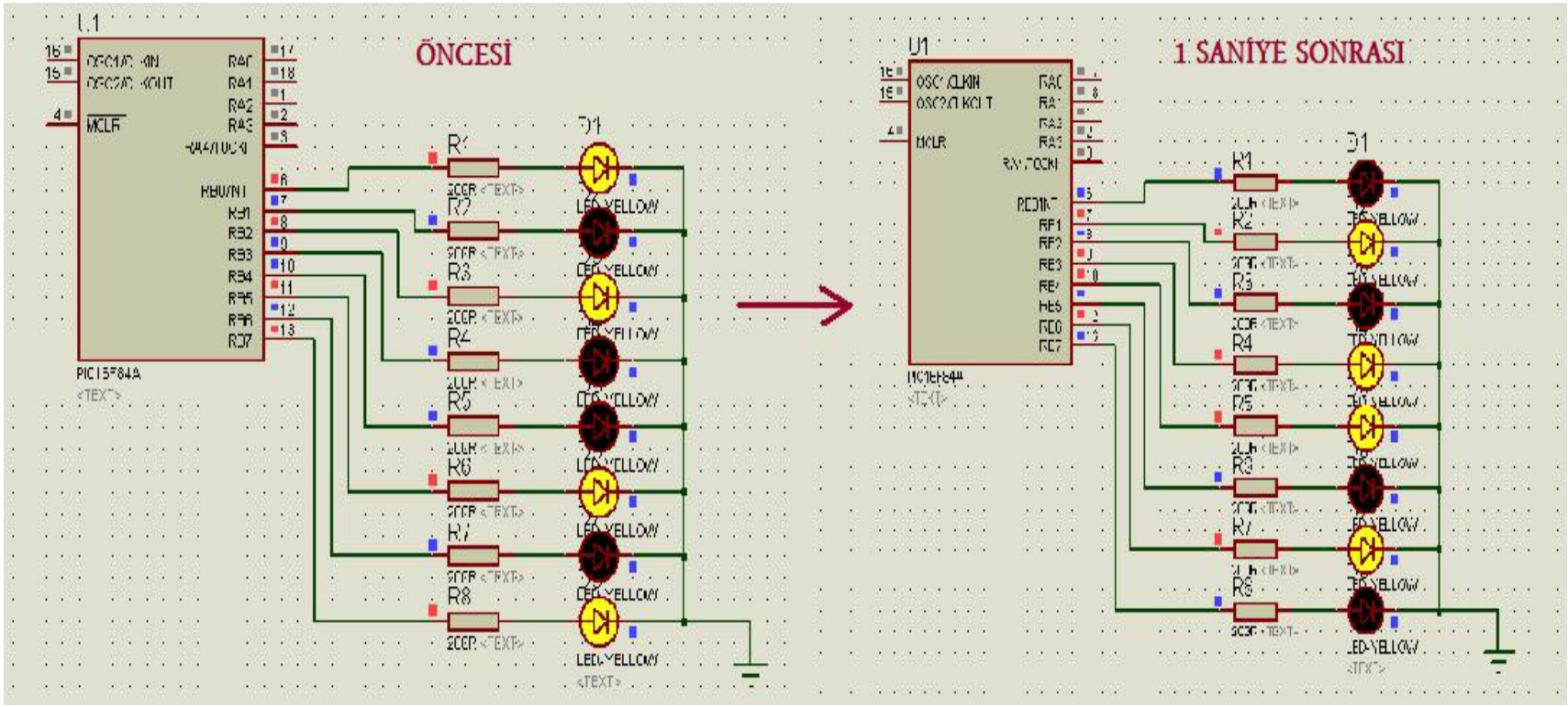
PIC

Örnek 1 : XOR

```
1   #include <pic.h>
2   #include "delay.h"
3
4   main()
5   {
6       int i;
7
8       TRISB=0x00;
9
10      PORTB=0xA5;
11
12      while(1)
13      {
14          PORTB^=0xFF;     //XOR işlemi
15
16          for(i=0;i<4;i++)     //------|
17          {                    //       |--> 1 saniye bekle
18              DelayMs(250);    //       |
19          }                    //------|
20      }
21  }
```

PIC

Buda simülasyonu:



PIC

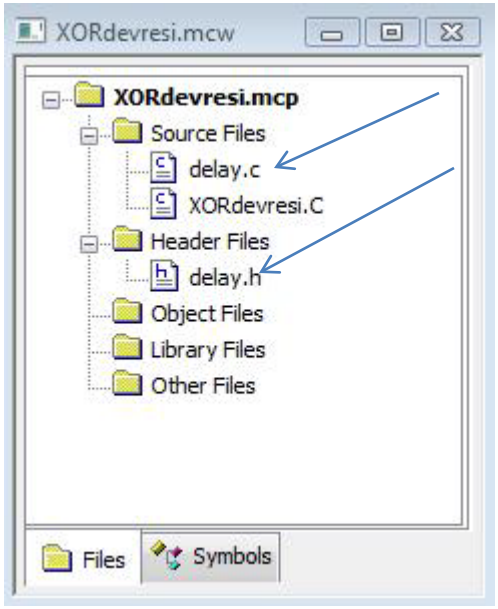
DelayMs() nedir?

Kodumuza baktığımızda “#include “delay.h” ” gibi ifadeler görüyoruz.

```
1   #include <pic.h>
2   #include "delay.h"
3
```

Burada kodumuza delay kütüphanesini dahil ediyoruz.Delay kütüphanesi bekleme fonksiyonlarını barındıran hazır bir kütüphanedir. DelayMs( ) ve DelayUs( ) fonksiyonları bu kütükte tanımlıdır.

Unutulmamalıdır ki; koda dahil ettiğimiz kütüphaneleri “Source Files” ve “Header Files” a ➡ eklememiz gerekmektedir.



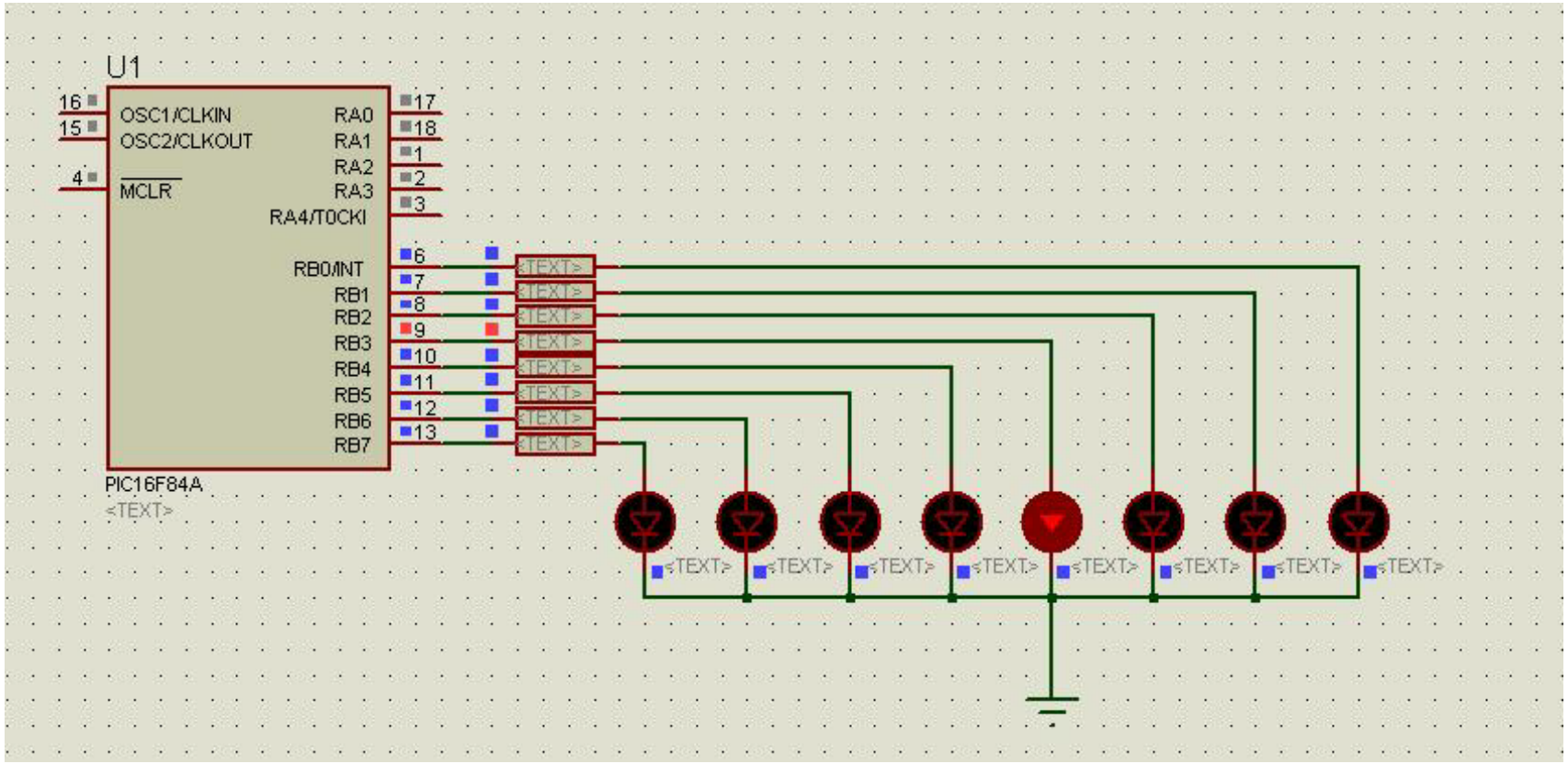
PIC

Örnek 2 : Karaşimşek devresi

```
1   /*meşhur karaşimşek... =)*/
2
3   #include <pic.h>
4   #include "delay.h"
5
6   main()
7   {
8       int i,j;
9
10      TRISB=0;
11
12      PORTB=1;
13
14      while(1)
15      {
16          for(i=0;i<7;i++)
17          {
18              PORTB=PORTB<<1;
19              DelayMs(100);
20          }
21          for(j=0;j<7;j++)
22          {
23              PORTB=PORTB>>1;
24              DelayMs(100);
25          }
26      }
27  }
28
```

PIC

Buda simülasyonu:



TEŞEKKÜRLER