1 RAYLI SİSTEM HAKKINDA BİLGİLENDİRME

Hafif raylı ulaşım sistemi ve ulaşım da kullanılan hafif raylı ulaşım araçları (Sirio) hakkında genel bilgiler içeren seminer.

1.1 Hafif Raylı Sistem

Hafif Raylı Sistem hattı 17 km uzunluğunda 28 duraktan oluşmaktadır.Hafif raylı sistem de 750Vdc voltaj mevcuttur.Bu gerilimi bir kısmını hafif raylı ulaşım araçları kullanmakta diğer kısmı da geri besleme olarak raylardan raylı sistem tesisinde toplanmaktadır.

1.2 Hafif Raylı Ulaşım Aracı(Sirio)

Sirio diğer hafif raylı sistem araçlarından ayıran en iyi özelliklerinden biri iki yönlü sürüş yeteneğinin olmasıdır

Diğer Birkaç Özelliği :

Max Hız	80 Km/h	**Koltuk Kapasitesi**	68 oturan+2 özürlü yeri(203 kişi(6kişi/m^2))
Gövdesi	Yüksek Dirençli Çelik	**Araç Uzunluğu**	32m
Güç	460-465Kw	**Platform Yüksekliği**	280mm

Resim 1: Hafif Raylı Ulaşım Aracı (Sirio)

Resimde görüldüğü üzere aracın rayla temas noktası 3 tür. Bu noktalardan iki uç kısım motor bölümleridir. Bu bölümlerin üst kısmında sürücü kontrol paneli mevcuttur ve bu sayede aracın sefer sonunda yönü değiştirmek zor olduğu için sürücü kabin değiştirerek araç geldiği istikametin tersi yönünde yol alabilmektedir.

Tarih	Konu	İMZA-Mühür-Kaşe
22.08.2011	Stajyer Eğitimi	

1.3 İş Yeri Güvenliği Talimatları

İş yeri iki kısımdan oluşmaktadır:

1-)**Atölye:** Raylı ulaşım araçlarının(Sirio) bakım ve onarımlarının yapıldığı kısımdır.

2-)Diğer birimler: Arge,Elektrik Sinyalizasyon,İnsan kaynakları,….vb

İş yerinde çalışanların can güvenlikleri açısından diğer birimlerden atölyeye geçiş yaparken uyulması gereken bir takım iş yeri talimatlarına uyulmasını gerekliliğinin anlatılması.

2 GÖREV ALACAĞIM BİRİMLERİN TANITIMI

Staj döneminde görev alacağım birimler

Staj dönemi boyunca Arge ve Elektrik sinyalizasyon bölümlerinde görev yapacağım.

Ağırlıklı olarak Staj konusuna uygun olarak Arge bölümünde Gömülü Sistem Programlaması üzerinde çalışmalara katılacağım.

a-)Arge : Hafif Raylı Ulaşım aracın (Sirio) elektrik ve elektronik sistemlerinin bakım,tamir ve geliştirme işlerinin yapıldığı birimdir.

Bu birimde :

1 Bilgisayar Müh.

1 Elektrik-Elektronik Müh.

1 Bilgisayar Teknolojileri ve Bilişim teknikleri Bölümü çalışmaktadır.

b-)Elektrik Sinyalizasyon: Hafif Raylı Sistemin genel koordinasyonu ve yeni projelerin fizibilitelerin çıkarıldığı daha çok sahada çalışan birimdir.

Bu birimde :

1 Elektronik Haberleşme Müh.

2 Elektrik-Elektronik Müh. çalışmaktadır.

ÖRNEKTİR KULLANILAMAZ

Tarih	Konu	İMZA-Mühür-Kaşe
22.08.2011	Stajyer Eğitimi	

GÖMÜLÜ SİSTEMLERİN TANINMASI

1-)STAJ BAŞLANGICI

Sorumlu mühendisin(Mehmet Asma-Bilgisayar Müh.) yönlendirmesi ile ilk olarak gömülü sistemler, mikroişlemciler temel yapı ve özellikleri hakkında internet üzerinden bilgi toplanması, sistemi ve bileşenlerini tanımak ilk amaçtır.Bu sebeple raporun bir kısmı teorik bilgi ağırlıklı olacaktır.

2-) İNTERNET ÜZERİNDEN GÖMÜLÜ SİSTEMLER HAKKINDA BİLGİ TOPLANMASI

Gömülü Sistem birçok sistemin bir arada olan bir yapı ihtiva eden halidir.Bu yapılar gömülü sistem üzerinde koordineli bir şekilde çalışması gerekir.Gömülü sistemlere verilebilecek en basit otomobildir.Örneğin otomobilde fren sistemi bir otomobilin içine gömülmüş bir sistemdir.Elektronik gömülü sistemler ise birçok elektronik bileşenlerin uyumlu bir şekilde çalışmasının bir ürünüdür.Örneğin : Trafik lambaları ve bunları kontrol eden elektronik kartlar bir gömülü sistem oluşturur.

Gömülü sistemleri hayatta kulanım alanları oldukça geniş ve elverişlidir.Gömülü sistemdeki bileşenlerin sistemle tam uyumda çalışabilmesi bu bileşenleri kontrol eden ve kendi kendine kara alabilen bir yapıya ihtiyaç vardır.Bu ihtiyacı gömülü sistemlerde mikrokontrol yapıları ve mikroişlemciler karşılamaktadır.

2.1-Gömülü Sistem Bileşenleri

Gömülü sistemleri bileşenleri temelde ikiye ayırırsak yanlış yapmış olmayız.Bu iki yapı şöyledir :

a-)Kontrol yapısı :Bu yapı mikrokontrol yapıları ve mikroişlemcilerden oluşmaktadır.

b-)Diğer tüm bileşenler:Bu yapı ise kontrol devresinin dışında kalan kısımdır yani kontrol yapıların kontrol ettiği ekipmanlardır.Ör:Transistör,7-Segment Display

2.2-Gömülü Sistemlerin Kullanım Alanları

Teknolojik Aletlerin yaygın kullanılması ile farkında olmadan aslında gömülü sistemleri hayatımızda sıkça kullanıyoruz.Gömülü sistemler günlük hayatta kullanılan aletler üzerinde bile mevcuttur.Basitten kompleks olana doğru gidecek olursak Ör:Elektronik saatler,televizyon kumandası,dot matrix dispaly görüntülü ekranlar,..vb.

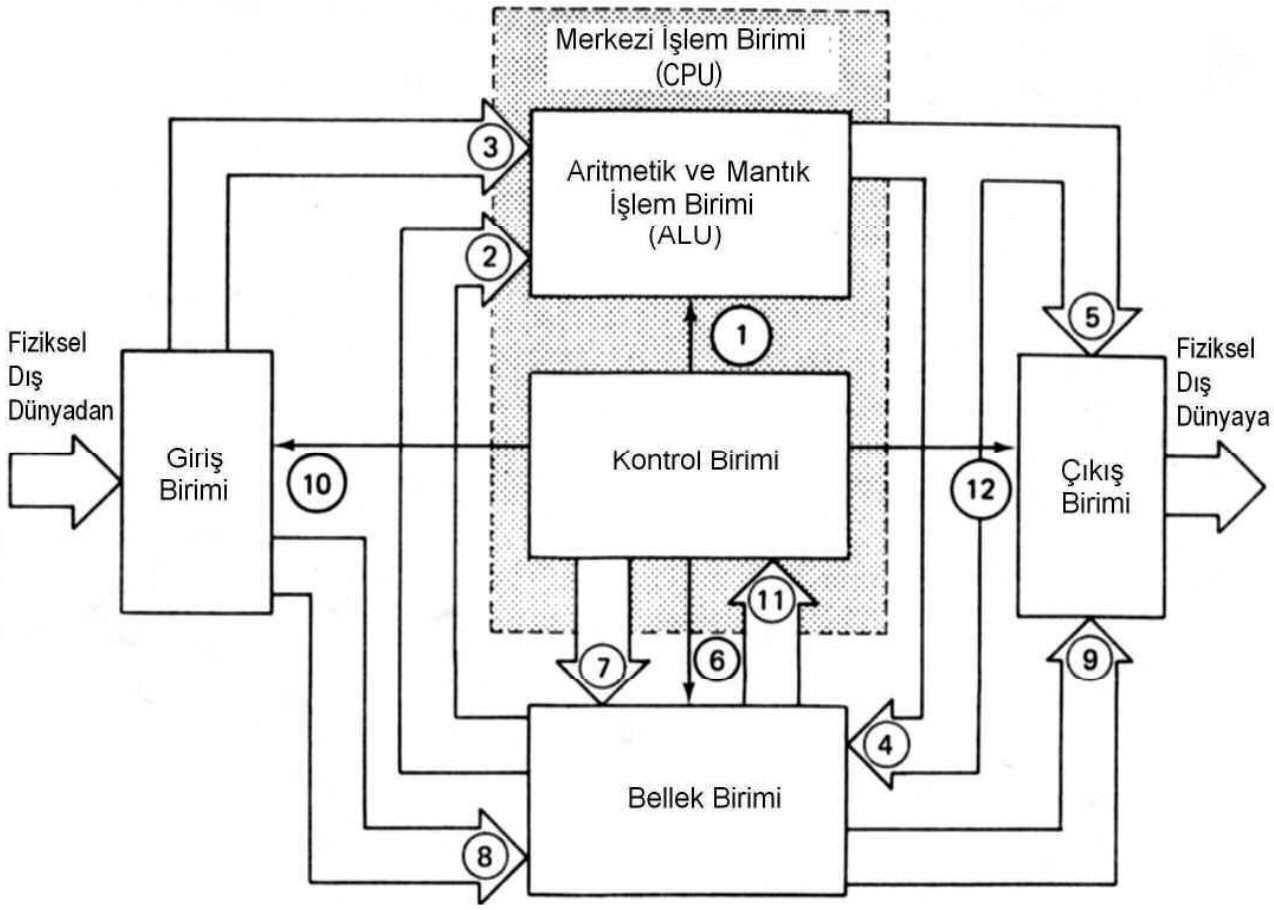
Bu kullanım alanlarından kavşak kontrol sistemlerinin ve hafif raylı ulaşım aracı(sirio) bünyesinde bulundurduğu gömülü sistemler ile ilgileneceğiz.Bu gömülü sistemlerin ilk olarak kontrol yapıları üzerinde çalışma yapılacak bu yüzden gelişen teknoloji ile birlikte artık programlanabilir mikrokontrol ve mikroişlemçileriler mevcuttur.İlk olarak mikrokontrol yapıların öğrenmesinden ve programlanmasından başlanacaktır.

Tarih	Konu	İMZA-Mühür-Kaşe
23.08.2011	Gömülü Sistemler	

MİKROİŞLEMÇİLERİN TEMEL YAPILARI

1-)MİKROİŞLEMÇİNİN TEMEL YAPISI

Mikroişlemciler gömülü sistemlerin kontrol ve yönetim işlemlerini yaparken çevresel devreler mevcuttur. Mikroişlemcilerde temel yapısı gömülü sistemlere benzerlik gösterir. Mikroişlemcilerde gömülü sistemler gibi birçok birimlerin bir araya gelmesi ile oluşmaktadır. Bu temel birimler şunlardır:



Şekil 1: Mikroişlemcinin Temel Sisteminin Blok Diyagramı

a-)Bellek Birimi: Mikroişlemcini çalışmasına yön verecek veya çalışması sırasında ihtiyaç duyacağı verilerin saklandığı birimdir ve Read Only Memory(ROM),Random Access Memory(RAM) olarak iki birime ayrılmaktadır.Bu iki birimde kendi içinde çeşitlenmektedir, şu şekilde ROM: 1-)ROM: Sadece okumanın yapılabildiği bellek türüdür.

2-)PROM: Sadece bir kez programlanabilir bellek türüdür.

3-)EPROM: Silinebilir ve programlanabilir belek türüdür.

4-)EEPROM: Elektriksel olarak programlanabilen ve silinebilen bellek türüdür ve hızlı programlanma özelliğine sahip olan EEPROM türüne Flash Eprom veya Flash bellek olarak da adlandırılmaktadır.

Tarih	Konu	İMZA-Mühür-Kaşe
24.08.2011	Mikroişlemciler	

RAM: 1-)SRAM: Statik okuma ve yazmanın yapılabildiği bellek türüdür.

2-)DRAM: Dinamik okuma ve yazmanın yapılabildiği bellek türüdür.

b-)Aritmetik ve Lojik işlem Birimi(ALU): Mikroişlemcinin üzerine düşen aritmetik ve lojik işlemleri yerine getiren birimdir.

c-)Kontrol Birimi: Mikroişlemcideki diğer birimleri koordinasyon, kontrol, çalışma düzenlerini belirleyen birimdir.

d-)Giriş/Çıkış Birimleri(I/O): Mikroişlemcinin veri giriş ve çıkışlarının yapıldığı ve PORT olarak adlandırılan birimlerdir.

KAYNAKLAR

1-)http://megep.meb.gov.tr/mte_program_modul/modul_pdf/523EO0014.pdf

2-)http://www.cizgi-tagem.org/resource/vfiles/tagem/dms_file/1298/PMS_05_MikroSisYap.pdf

3-)http://web.itu.edu.tr/~sgunduz/courses_2005/mikroisl/slides/d2.pdf

2-)MİCROİŞLEMCİLERİN PROGRMALANMASI

Sorumlu mühendise verilen tavsiyelerin yerine getirildiğini bilgilendirerek ,kısa bir sözlü anlatım yapılmıştır.Mikroişlemci programlama işinin teorik bilgilerle sadece az bir kısmın tecrübe edilebileceğini ve uygulamalar yapma gerektiğini tarafıma tavsiye edilerek,öncelik olarak PIC lerle başlamam söylenmiş ve PIC18F452 datasheet 'i verilmiştir.

2.1-)PIC18F452 DATASHEET

PIC18F452 datasheet incelenerek PIC18F452 bazı genel özelliklerini içeren aşağıdaki veri özeti ve tablo hazırlanmıştır.

MİKROİŞLEMCİ	FLASH BELLEK(Bytes)	RAM (Bytes)	EEPROM(Bytes)
PIC18F452	16K	768	256

Çalışma Frekansı: DC-40MHZ **Giriş/Çıkış Portları**: PORT A,B,C,D,E

Kesme Kaynağı: 17

Bu datasheet genel olarak incelenmiş ve sorumlu mühendise bu şekil datasheet tamamen ayrıntılı bir şekilde incelemenin mümkün olmadığı bildirerek .PIC18F452 internet üzerinden uygulamalı örneklerle ilgili birkaç doküman toplanarak sorumlu mühendise sunulmuş ve incelemesini ardından tavsiye ettiği dökümandaki uygulamalara geçilmiştir.

Tavsiye edilen dökümanlar: HI-TECH ile PIC Programlama C dilinde

Tarih	Konu	İMZA-Mühür-Kaşe
24.08.2011	PIC18F452	

2.2-) PIC18F452 PROGRAMLAMA

Tavsiye edilen dökümandaki uygulamalar incelenerek başlanmıştır ve bu uygulamaların Proteus Programında Simülasyon edilmesinin onayını sorumlu mühendis den alınarak uygulamalara başlanmıştır.

2.3-)PIC18F452 UYGULAMALARI

2.3.1-)I/O BİRİMLERİN AYARLANMISI VE LED UYGULAMALARI

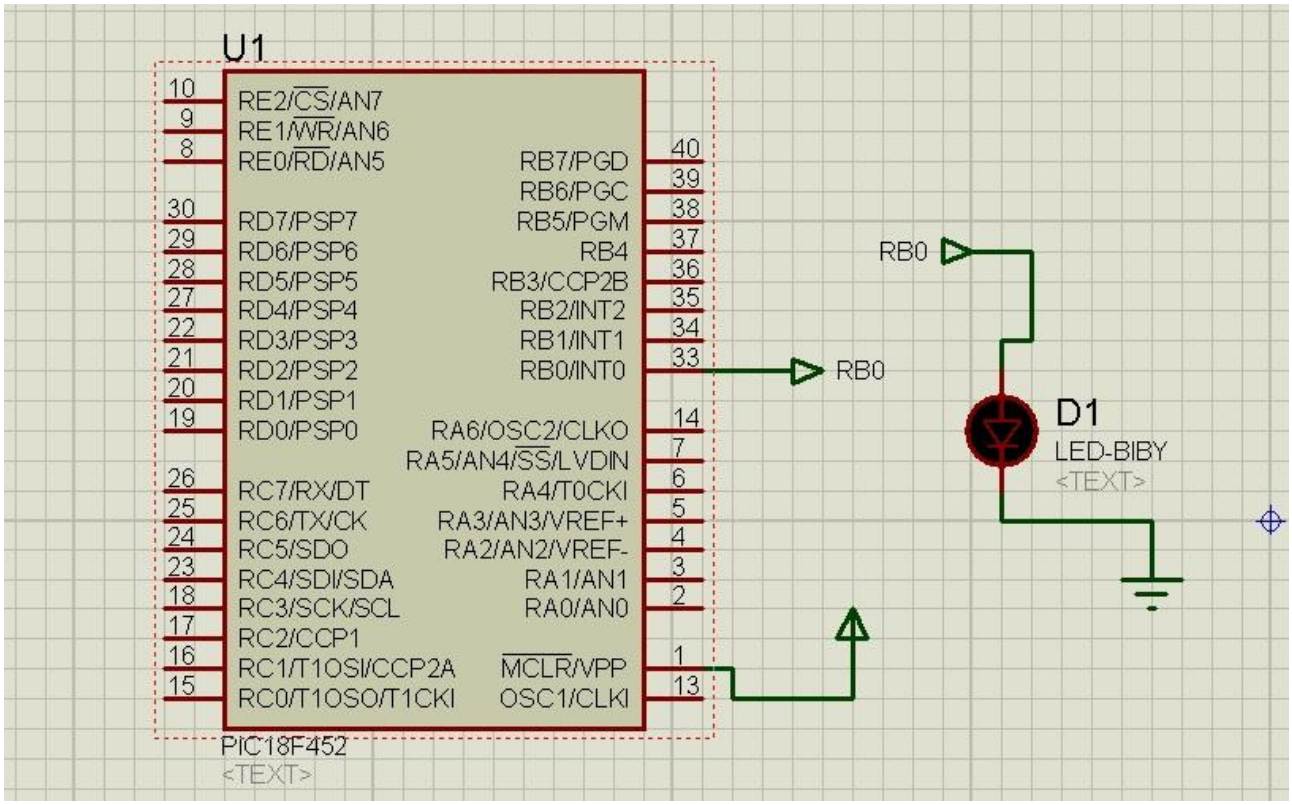
A-) Led yakıp söndüren devre

Bu uygulamada D1 ledini yakıp söndürme işlemini yapacağız ilk olarak PORTB'nin RBO pini çıkış olarak belirliyoruz.Delay.h ile main.c(kaynak dosyası) DelayMs(1000); ile devre işlemini geciktirerek led yakıp söndürüyoruz.

----------KOD(main.c)----------

```
#include <htc.h>

#include "delay.h" // Gecikme yaratacak kütüphane

void main(void) // Ana fonksiyon alanı

{

TRISB=0x00; // PORTB çıkıs olarak yönlendiriliyor

PORTB=0x00; // PORTB'nin tüm çıkısları sıfırlanıyor

while (1) // Sonsuz döngüye giriliyor

{

RB0=1; // Led yanacak

DelayMs(1000);DelayMs(250); // Yarım saniye beklenecek

RB0=0; // Led sönecek

DelayMs(1000);DelayMs(250); // Yarım saniye beklenecek
```

Tarih	Konu	İMZA-Mühür-Kaşe
25.08.2011	PIC18F452 Uygulamaları	



Devre 1:Proteus üzerindeki devre çizimi

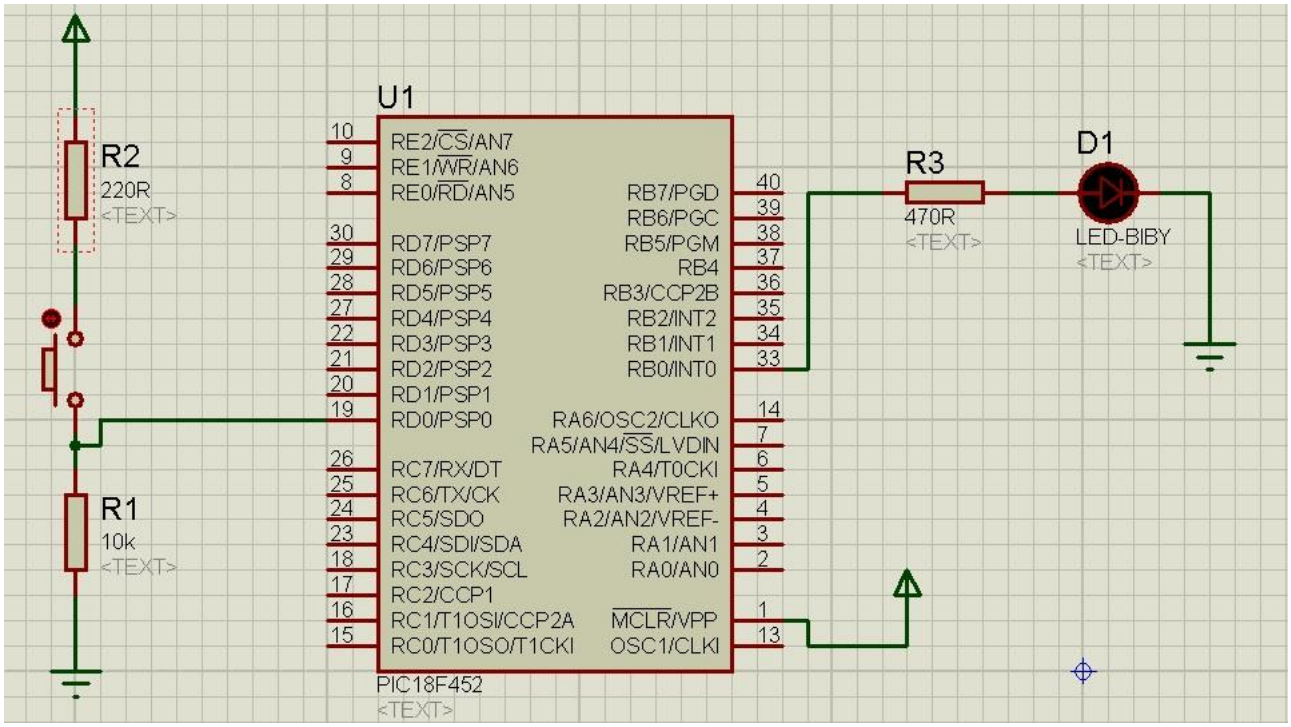
B-) Buton ile Led Kontrolü

Devredeki buton ile led'i kontrol ederek yanıp sönmesini sağlamak.Bu uygulamada PORTD'i çıkış olarak ayarlanarak sıfırlama yapılmış ve kontrol pini olara RD0 seçilmiştir.

----------KOD(main.c)----------

```
#include <htc.h>

void main(void) // Ana fonksiyon alanı

{ADCON1=0x07; // PORTA dijital olarak yönlendiriliyor

TRISD=0x01; // RA0 giris olarak yönlendiriliyor

TRISB=0x00; // PORTB çıkıs olarak yönlendiriliyor

PORTB=0x00; // PORTB'nin tüm çıkısları sıfırlanıyor

PORTD=0x00; // PORTA'nın tüm çıkısları sıfırlanıyor

for(;;) {   RB0=RD0; // RB0 çıkısı RA0 girisine esitleniyor}  // Sonsuz döngüye giriliyor
```

Tarih	Konu	İMZA-Mühür-Kaşe
25.08.2011	PIC18F452 Uygulamaları	



Devre 2:Proteus üzerindeki devre çizimi

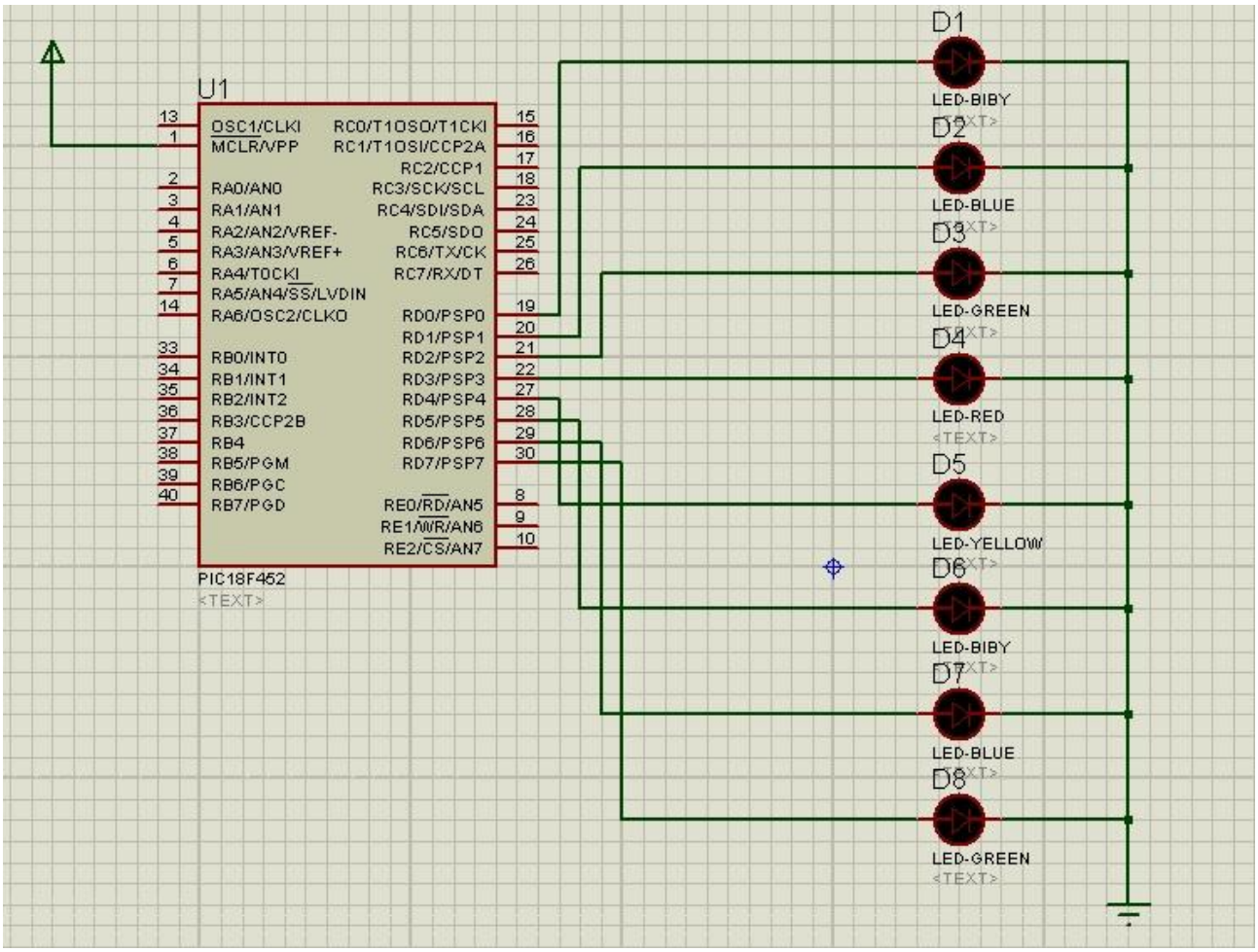
C-) Led ile Animasyon Uygulaması

Devre farklı renkteki Ledlerin sırsıyla baştan sona sırasıyla ,sona ulaştığında sondan başa doğru sırasıyla yanarak oluşturulan bir animasyon devresidir.

----------KOD(main.c)----------

```
#include <htc.h>
#include "delay.h" // Gecikme kütüphanesi
void main(void) // Ana fonksiyon alanı
{
char led=1; // led seklinde bir sabit tanımlanıyor
TRISD=0x00; // PORTD çıkıs olarak yönlendiriliyor
PORTD=0x00; // PORTD'nin tüm çıkısları sıfırlanıyor
for(;;) // Sonsuz döngüye giriliyor
{PORTD=led; // PORTD led değiskenine esitleniyor
led=led<<1; // led birimi bir sola kaydırılıyor
DelayMs(1000); // 100ms bekleniyor
```

```
if(led==0x80) // Eğer PORTD=0x80 olursa alt islemleregeçiliyor
{
for(;;) // Tekrar sonsuz döngüye giriliyor
{
PORTD=led; // PORTB led değiskenine esitleniyor
led=led>>1; // led birimi bir sağa kaydırılıyor
DelayMs(1000); // 100ms bekleniyor
if(led==0x01) // Eğer PORTB=0x01 olursa ikincisonsuz döngüden
break; // birinci sonsuz döngüye giriliyor
}}}}
```



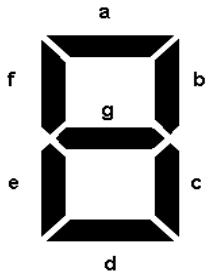
Devre 3:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
26.08.2011	PIC18F452 Uygulamaları	

2.3.2-)I/O BİRİMLERİN AYARLANMISI VE 7-SEGMENT VE DOT-MATRİX UYGULAMALARI

A-)7-Segment Uygulaması

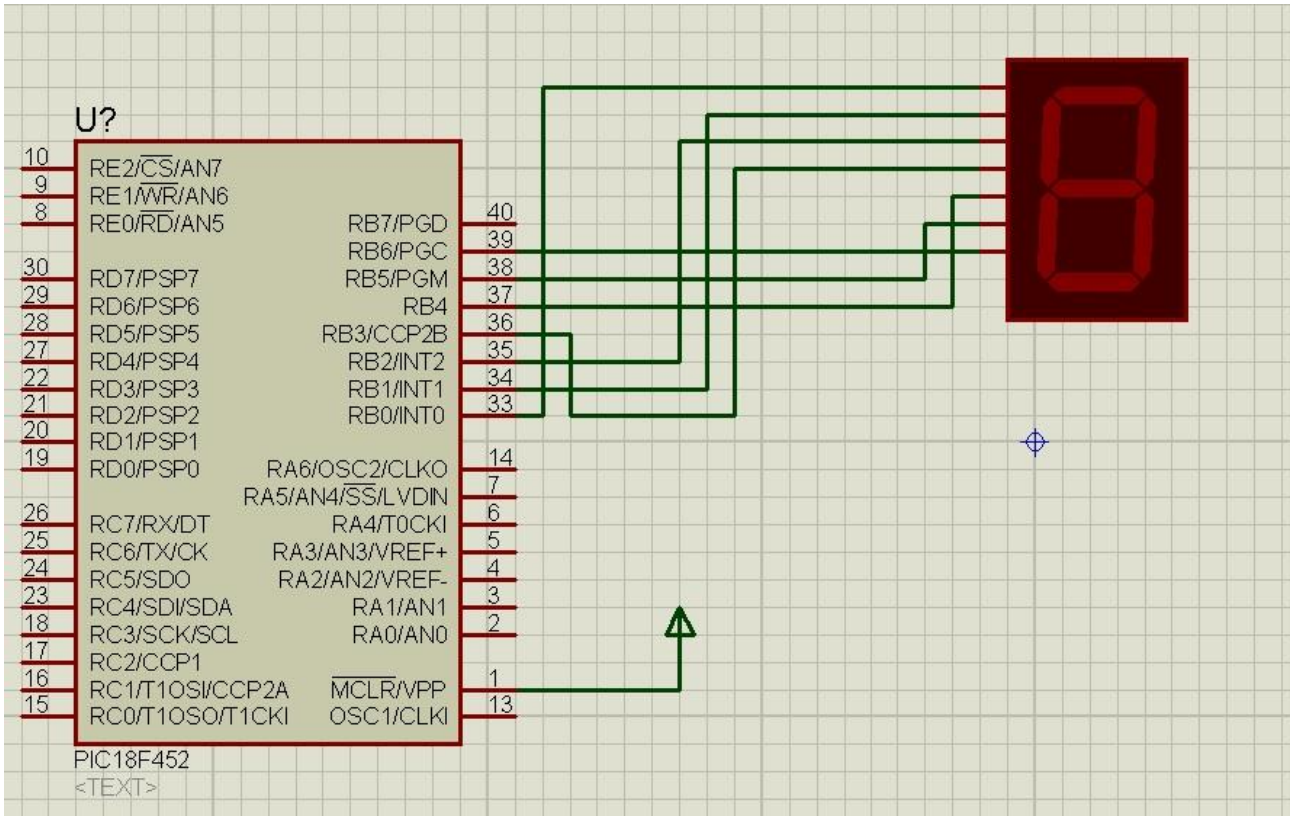
Devre sonsuz döngü içerisinde sürekli olarak PORTB'ye bilgi göndermektedir.PORTB ilk pinden itibaren 7-Segment Display'a resimde görüldüğü gibi alfabetik sıra ile bağlanmıştır.PIC sonsuz döngüde bırakılma sebebi uygulamanın amacı gereği 0-9 kadar saydırmak olduğu içindir.PORTB'e gönderilecek bilgiler sırası ile belli olduğu için hexdecimal şeklinde bir dizi oluşturarak çıkış olarak belirlediğimiz PORTB'e sırası ile göndereceğiz ve sayılan sayıların fark edilebilmesi içinde 1000 milisaniye bir geciktirme uyguluyoruz.Devrenin kod ve şeması aşağıda mevcuttur.



----------KOD(main.c)----------

```
#include <htc.h>

#include "Delay.h" // Gecikme kütüphanesi

const unsigned char segment[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

int j;

void main(void) // Ana fonksiyon alanı

{char i; // Herhangi bir değisken tanımlanıyor

TRISB=0x00; // PORTB çıkıs olarak yönlendiriliyor

PORTB=0x00; // PORTB'nin tüm çıkısları sıfırlanıyor

for(;;) // Sonsuz döngüye giriliyor

for(i=0;i<10;i++){

PORTB=segment[i]; // Seven segment değerleri alınıyor

DelayMs(1000); // 1000ms bekleniyor}}
```

Tarih	Konu	İMZA-Mühür-Kaşe
29.08.2011	PIC18F452 Uygulamaları	



Devre 4:Proteus üzerindeki devre çizimi

B-)İkili 7-Segmenti Buton ile Kontrol Uygulaması

Devre 7 Segment-Mpx2-cc,PIC18f452,2 Butondan oluşmaktadır.Bir önceki uygulamada PIC clock ile saydırma yapmıştık.Bu devrede ise saymayı PIC dışarıdan aldığı veri ile yapacaktır.PIC aldığı veri butonlar sayesinde sağlanarak devre çiziminde görüldüğü gibi üsteki buton ileri doğru alttaki buton geri doğru saymayı sağlamaktadır.Bu devrede de sabit hexdecimal dizimizden faydalanıyoruz.PORTA dijital girişe çeviriyoruz veri almak için PORTB'i ve PORTC'i kullanarak ikili 7-Segment Display kontrol ediyoruz.Burada PORTC'nin görevi harf göster fonksiyonun 7 Segment Display' 1 ve 2 bacağını sürekli olarak hexdecimal olarak önce 0x02 gönderilir ve PORTB'den sayı bilgisi gönderilir ardından aynı şekil PORTC 0x01 yapılarak PORTB'den sayı bilgisi gönderilir.Bu şekilde 7 Segment Display'e hangi sayını yazılacağı kontrol edilir.harf göster fonksiyonuna gidecek harfin belirlenmesi işlemi de sonsuz döngü içinde sürekli olarak RA0 ve RA1 kontrol edilmektedir.

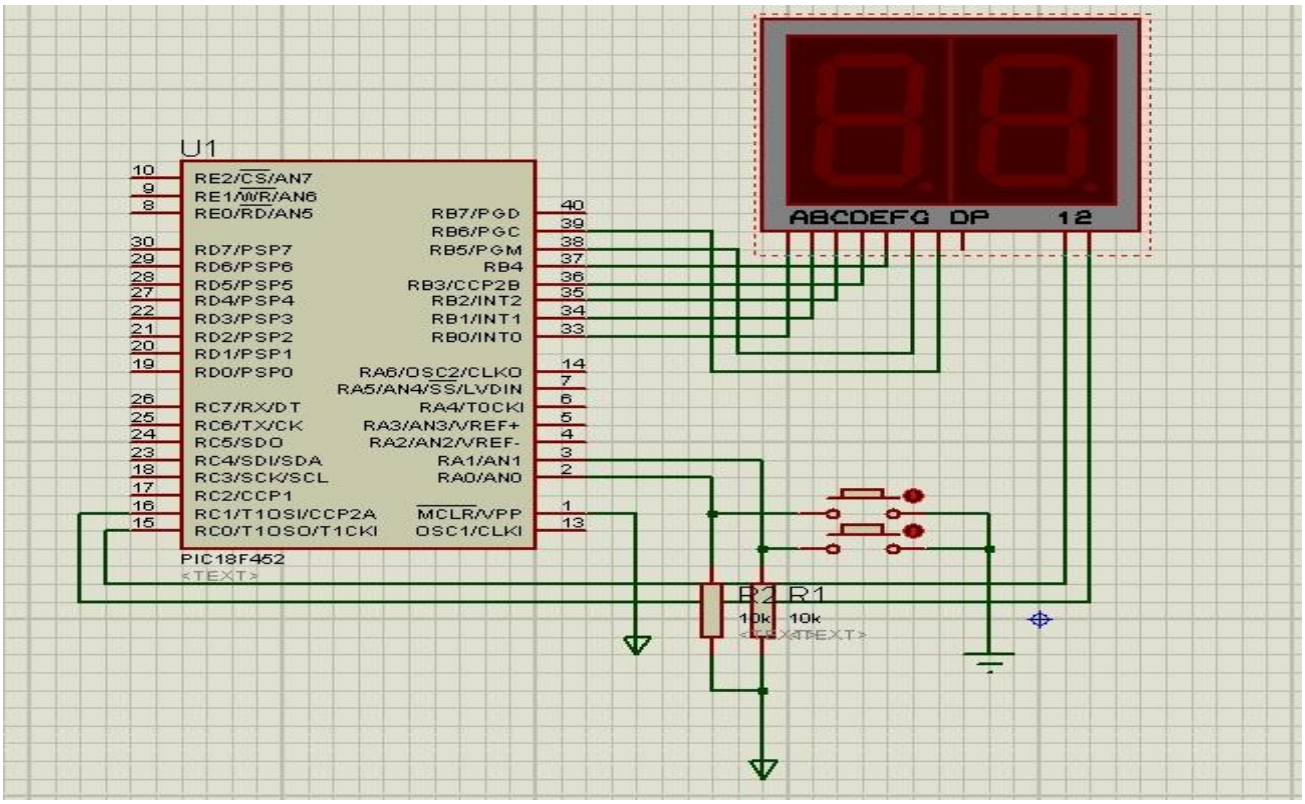
----------KOD(main.c)----------

```
#include <htc.h>

#include "Delay.h" // Gecikme kütüphanesi

const unsigned char segment[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
```

Tarih	Konu	İMZA-Mühür-Kaşe
29.08.2011	PIC18F452 Uygulamaları	

```
void sayi_goster(char i) // Sayı göstermeye yarayan fonk
{
PORTC=0x02; // PORTC'de 2 değeri gönderiliyor
PORTB=segment[i/10]; // i'nin 10'a bölümü gösteriliyor
DelayMs(5); // 5ms bekleniyor
PORTC=0x01; // PORTC'de 1 değeri gönderiliyor
PORTB=segment[i%10]; // i'nin 10'a bölümünden kalanı  gös
DelayMs(5); // 5ms bekleniyor}
void main(void) {   // Ana fonksiyon alanı
int i; // Herhangi bir değisken tanımlanıyor
ADCON1=0x07; // PORTA dijital yapılıyor
TRISA=0x03; // PORTA'nın ilk iki pini giris
TRISB=0x00; // PORTB çıkıs olarak yönlendiriliyor
TRISC=0x00; // PORTC çıkıs yapılıyor Devamı Sağ Tarafta
```

```
PORTA=0x00; // PORTA'nin tüm çıkısları sıfırlanıyor
PORTB=0x00; // PORTB'nin tüm çıkısları sıfırlanıyor
PORTC=0x00; // PORTC'nin tüm çıkısları sıfırlanıyor
for(;;) // Sonsuz döngüye giriliyor
{if(RA0==1) {    // RA0'pini 0 mı?
 while(RA0); // Buton bırakıldı mı diye bakılıyor
i++; // Değisken artırılıyor
if(i>99) // Eğer değisken 99'dan büyükse 0 oluyor
i=0;}
else if(RA1==1) {// RA1'pini 0 mı?
while(RA1); // Buton bırakıldı mı diye bakılıyor
i--; // Değisken azaltılıyor
if(i<0) // Eğer değisken 0'dan küçükse 99 oluyor
i=99;}sayi_goster(i); // O anki sayı gösteriliyor}}
```



Devre 5:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
30.08.2011	PIC18F452 Uygulamaları	

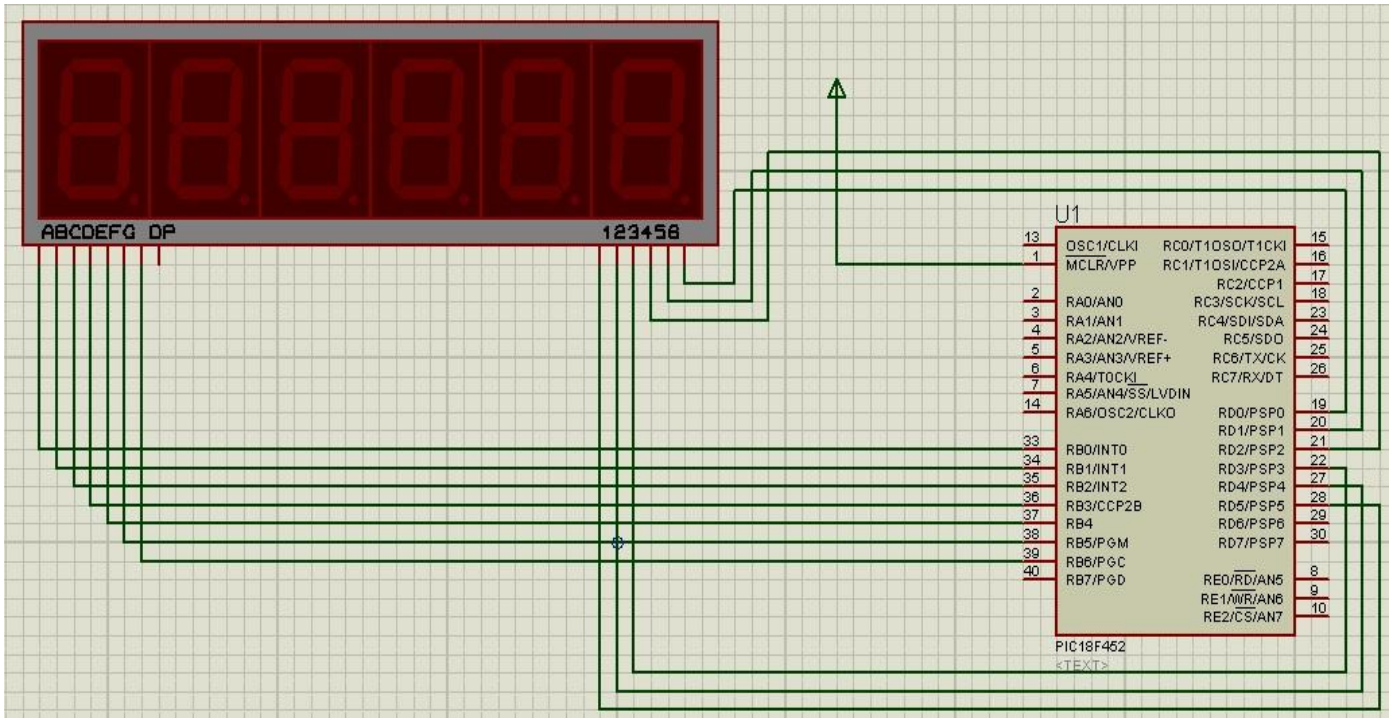
C-)Dijital Saat Uygulaması

Uygulamanın amacı PIC ile Dijital saat yapmaktır.Bir önceki uygulamada kullanılan ikili 7-SEG Display yerine burada altılı 7-SEG Display kullanacağız çünkü saniye dakika ve saat hücrelerini hesaba kattığımızda top 6 tane 7-SEG Display gereklidir.Bu uygulamada PIC'in iki portunu kullanacağız.PORTB den sayının hexdecimal karşılığını PORTD den display seçimini yapacağız. PIC sonsuz döngü içerisinde bir müddet PORTB ve PORTD den sürekli aynı veri gönderilir bizim değişimi algılayabilmemiz için sonsuz döngü içerisi de başka bir döngü ile sayma işlemi yapılır ve if ve else yapıları ile saatin çalıştığı rakam aralığı PIC'e yüklenmiş olur.

----------KOD(main.c)----------

```
#include<p18f452.h>
#include<delays.h>
#pragma config OSC=HS
#pragma config WDT=OFF
 #pragma config LVP=OFF
const unsigned char SAYI []=
{0x3F,0x06,0x5B,0x4F,0x66,
0x6D,0x7D,0x07,0x7F,0x6F};
int s1=5,s2=5,d1=9,d2=5,sa1=3;
int  sayac=0,sayac2=0,a=0, sa2=2;
void main(void){
TRISD=0;TRISB=0;PORTD=0;PORTB=0;
while(1){
for(sayac=0;sayac<25;sayac++){
for(sayac2=1;sayac2<33;sayac2*=2){
PORTD=sayac2;
if(sayac2==1){
PORTB=~SAYI[s1];
if(s1==10)
{s1=0;s2++;}}}
else if(sayac2==2){PORTB=~SAYI[s2];
if(s2==6){s2=0;d1++;}}
else if(sayac2==4){PORTB=~SAYI[d1];
if(d1==10)
{d1=0;d2++;}}
else if(sayac2==8){PORTB=~SAYI[d2];
if(d2==6){d2=0;sa1++;}}
else if(sayac2==16){
PORTB=~SAYI[sa1];
if(sa1==9)
{sa1=0;sa2++;}}
else if(sayac2==32){
PORTB=~SAYI[sa2];}
Delay1KTCYx(2);}
if(sa1==3){
 if(sa2==2&&s1==9&&s2==5&&d1==9&&d2==5){
Delay10KTCYx(10);sa2=0;
sa1=0;d1=0;d2=0;s2=0;s1=0;}}
if(sayac==24)
s1++;}}}
```

Tarih	Konu	İMZA-Mühür-Kaşe
30.08.2011	PIC18F452 Uygulamaları	



Devre 6:Proteus üzerindeki devre çizimi

D-) Dot-Matrix Kayan Yazı Uygulaması

Devrede 4 tane(5x7) Dot-Matrix ,6 tane demultiplexer ,PIC18F452 kullandık.Dot-Matrix yapısını ve çalışma prensibini bilmeden PIC programlama işi tamamlanamaz.(5x7)Dot-matrix 5 üste 7 alta olmak üzere toplam 12 girişi vardır.Üsteki 5 giriş hangi sütun veya sütunların olacağını alttaki 7 giriş ise hangi satır veya satırların olacağını belirlememizi sağlar.1 tane Dot-Matrix bir her harfi oluşturmak için döngü kurulur ve hangi sütun ve satır olacağını da hexdecimal halindeki dizilerle döngünün içerisinde belirleriz.Bu devre ise biz 4 tane Dot-Matrix kullanacağımız için veri portlarını zaman içerisinde farklı Dot-Matrix'lere 6 tane demultiplexer kullandık.PIC'in PORTA, PORTB, PORTD portlarından PORTA demultiplexerları kontrol için,PORTB'i Dot-Matrixlerin sütunlarını kontrol için,PORTC'i Dot-Matrixlerin satırlarını kontrol için kullandık.Bu uygulamada aynı anda bir port da birden fazla verinin olması gerekiyor bu da programlama mantığına ters ve donanım üstünde elde etmek mümkün değildir.Bu yüz bir demulitplexerlar ve PIC ile portlarındaki bilgileri insanın algılamasının mümkün olmayacağı bir hızla değiştirerek bir süreklilik varmış gibi gösterebiliyoruz bu devrenin de çalışma prensibi bunun üstüne kurulmuştur.

----------KOD(main.c)----------

```
#include <htc.h>

#include "Delay.h"

unsigned char M[]={0x7F,0x02,0x04,0x02,0x7F}; // M

unsigned char A[]={0x7E,0x09,0x09,0x09,0x7E}; // A
```

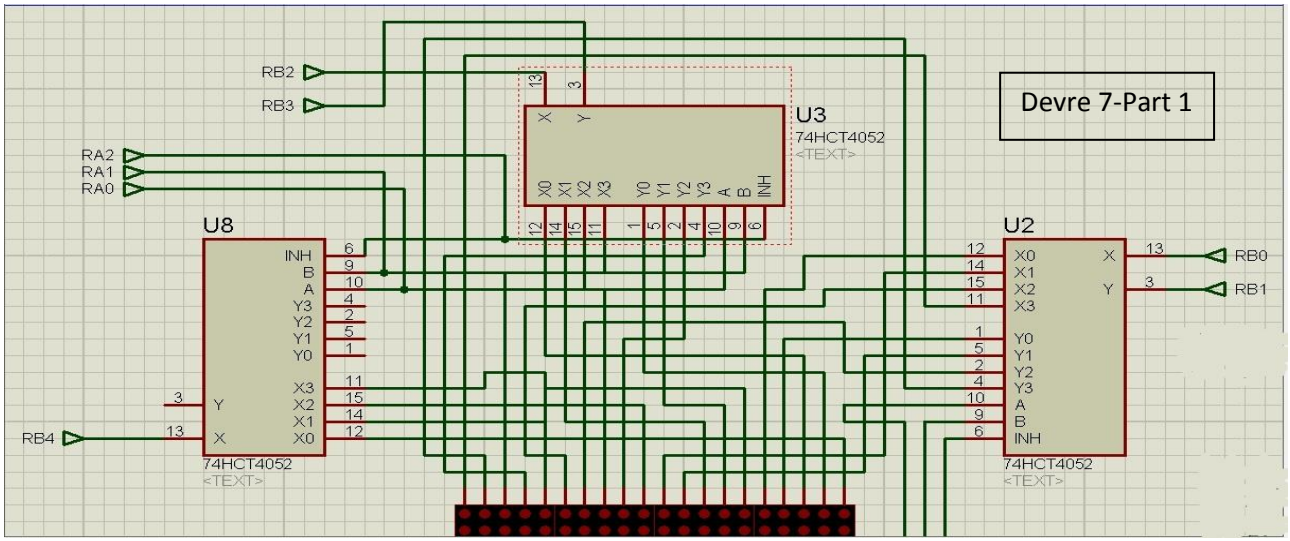
Tarih	Konu	İMZA-Mühür-Kaşe
31.08.2011	PIC18F452 Uygulamaları	

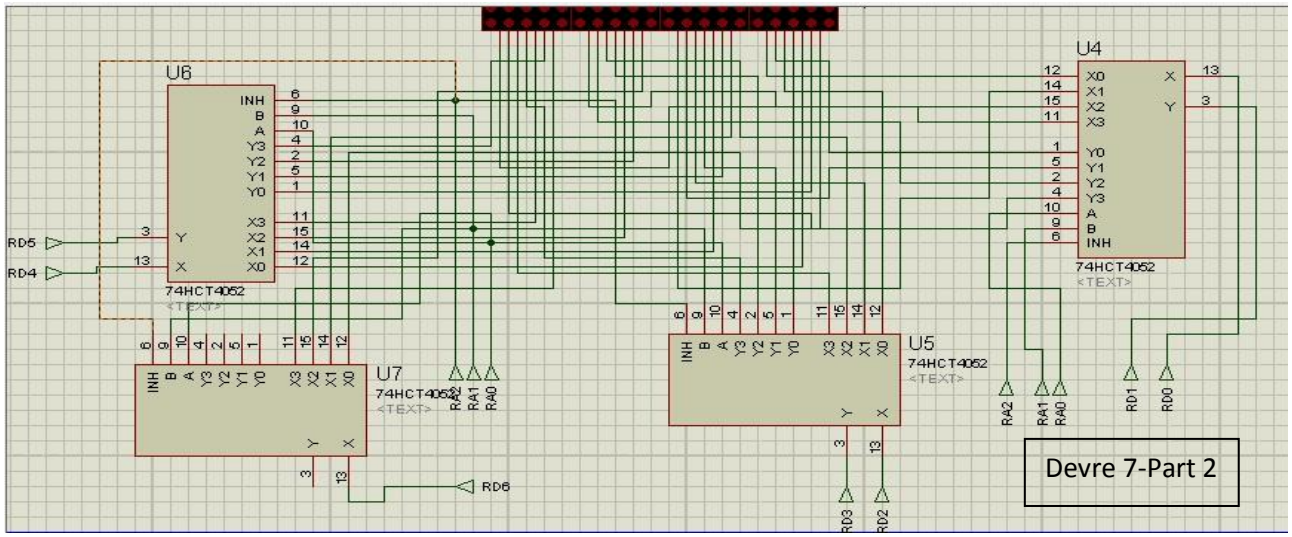
```
unsigned char L[]={0x7F,0x40,0x40,0x40,0x40}; // L
unsigned char I[]={0x00,0x42,0x7F,0x42,0x00}; // I
int Harf[4]={M,A,L,I};
unsigned int say[]={0x01,0x02,0x04,0x08,0x10};
void harf_goster(unsigned char H[]){
 int j;
for(j=0;j<5;j++){
PORTB=say[j];
PORTD=~H[j];
DelayMs(2);}}
void main(void){
TRISA=0x00;
TRISB=0x00;
TRISD=0x00;
PORTA=0x00;
PORTB=0x00;
PORTD=0x00;
ADCON1=0x07;
int i,k,a;
for(;;)
for(k=0;k<4;k++){
if(k==0){
for(a=0;a<150;a++){
PORTA=0;
harf_goster(Harf[0]);}}}//Devamı Sağ Tarafta
```

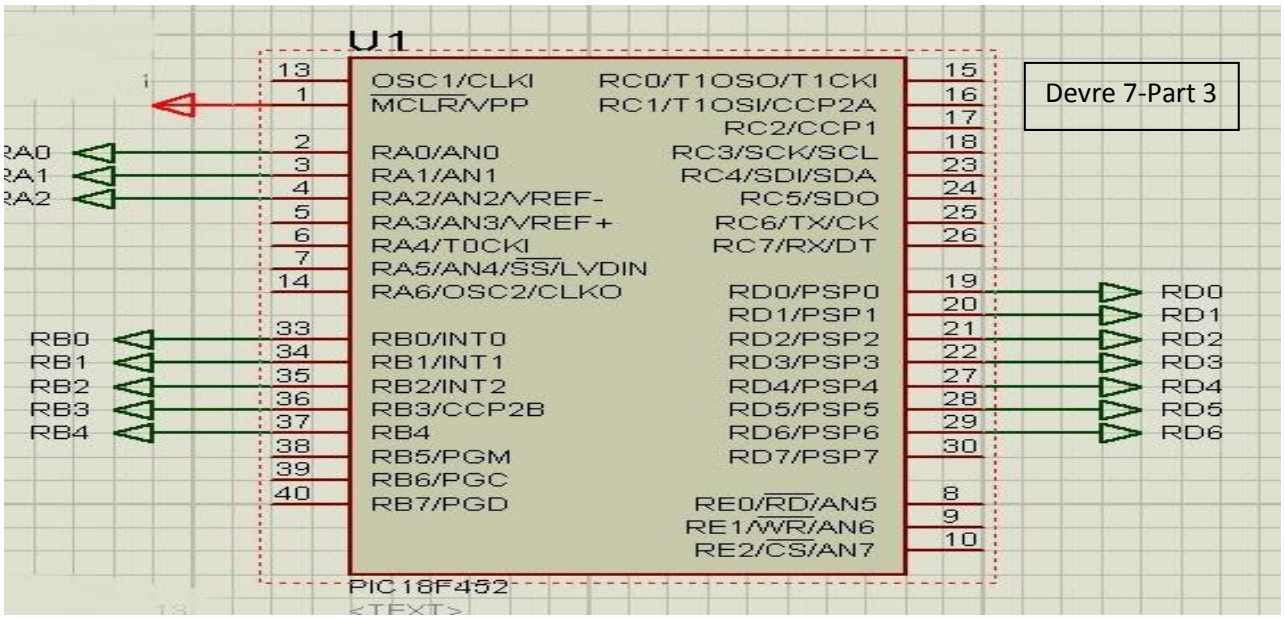
```
else if(k==1){
for(a=0;a<100;a++)
for(a=0;a<100;a++){
PORTA=1;
harf_goster(Harf[0]);
PORTA=0;
harf_goster(Harf[1]);}}
else if(k==2){
for(a=0;a<100;a++){
PORTA=2;
harf_goster(Harf[0]);
PORTA=1;
harf_goster(Harf[1]);
PORTA=0;
harf_goster(Harf[2]);}}
else
for(a=0;a<8000;a++){
PORTA=3;
harf_goster(Harf[0]);
PORTA=2;
harf_goster(Harf[1]);
PORTA=1;
harf_goster(Harf[2]);
PORTA=0;
harf_goster(Harf[3]);}}}
```

Tarih	Konu	İMZA-Mühür-Kaşe
01.08.2011	PIC18F452 Uygulamaları	

Devre 7:Proteus üzerindeki devre çizimi







Tarih	Konu	İMZA-Mühür-Kaşe
01.08.2011	PIC18F452 Uygulamaları	

2.3.3-)INTERRUPT UYGULAMALARI

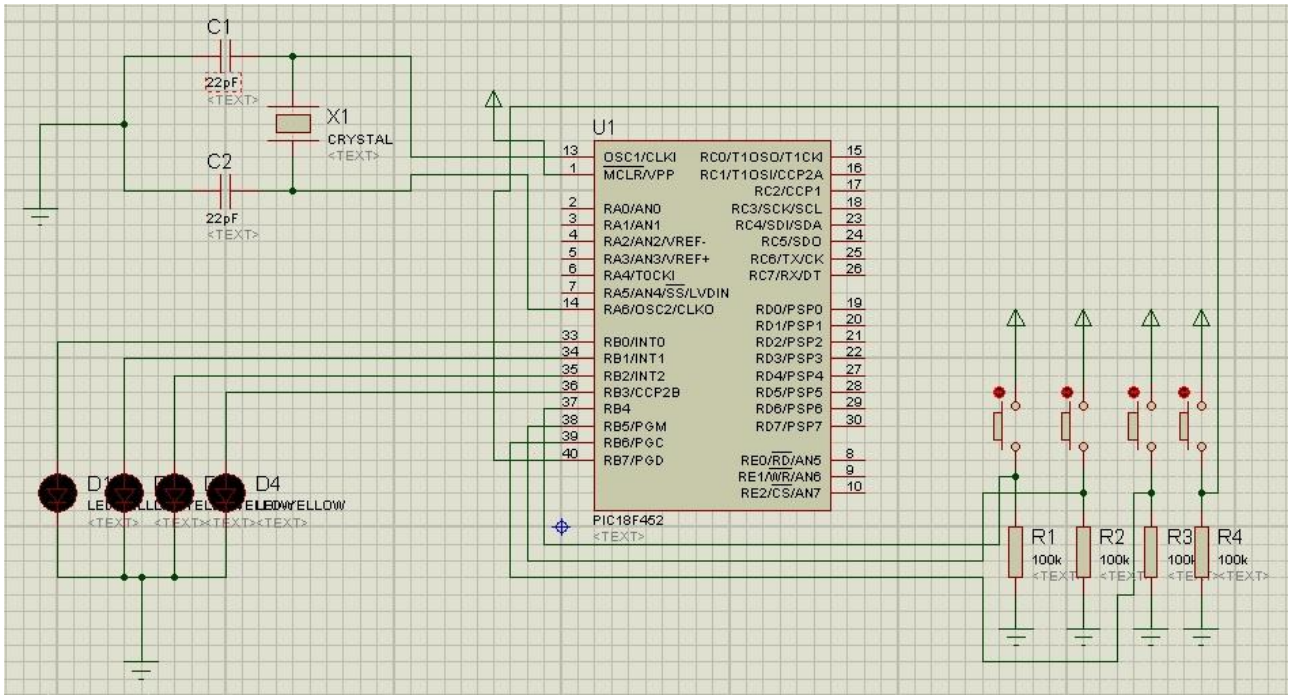
A-)Farklı Kesme Kaynakları ile Led Uygulaması

PIC18F452'nin 10 tane kesme register vardır.Bu kesme kaynakları çok bulunması yapılan uygulamalar ve projelerin esnek olmasını sağlamaktadır.Kesme kısaca PIC normal main kısmını işletirken kesme(Interrupt) olarak belirlenen, dışarıdan veya PIC clock'u ile yükselen veya alçalan kenarla tetiklenen bit'in koda göre 1 veya 0 olduğunda yani kesme gerçekleştiğinde PIC main kısmında yaptığı işe ara vererek kesme kaynağı bloğu içindeki kısmı işleme koymasına kesme(Interrupt) denir.Bizim uygulamamızda farklı kesme kaynaklarından oluşan kesmeler için farklı Ledleri yakıp söndürerek her farklı kesme için PIC'e farklı işlemler yaptırabileceğimizi göstermek uygulamanın amacıdır.

----------KOD(main.c)----------

```
#include<htc.h>

#include "Delay.h"

void main(void) { TRISB=0xF0;    // Ana fonksiyon alanı// RB4..RB7 giris, diğerleri çıkıs

PORTB=0x00; RBIF=0;    // PORTB sıfırlanıyor// RB4..RB7 kesme bayrağı temizleniyor

RBIE=1; GIE=1;  // RB4..RB7 kesme izni veriliyor// Genel kesme izni veriliyor

for(;;); }                              // Dslemci sonsuz döngüde bekletiliyor

static void interrupt  isim(void)              // Kesme fonksiyonu // Kesme fonksiyon ismi (önemsiz)

{if(RBIF) GIE=0; // RB4..RB7 kesme olusmus mu bakılıyor Kesme varsa, baska kesme gelmemesi için
          //genel kesme sıfırlanıyor

if(RB4)  PORTB=0x01; // ilk butona basıldıysa// ilk ledi yak

else if(RB5) PORTB=0x02; // ikinci butona basıldıysa// ikinci ledi yak

else if(RB6) PORTB=0x04;// Üçüncü butona basıldıysa// Üçüncü ledi yak

else if(RB7) PORTB=0x08; // Dördüncü butona basıldıysa// Dördüncü ledi yak

RBIF=0; // Tekrar dıs kesme alınabilmesi için kesme bayrağı temizleniyor

GIE=1;  // Tekrar kesme alınabilmesi için genel kesme bayrağı set ediliyor}}
```

Tarih	Konu	İMZA-Mühür-Kaşe
02.08.2011	PIC18F452 Uygulamaları	



Devre 8:Proteus üzerindeki devre çizimi

B-)Basit Kavşak Kontrolü

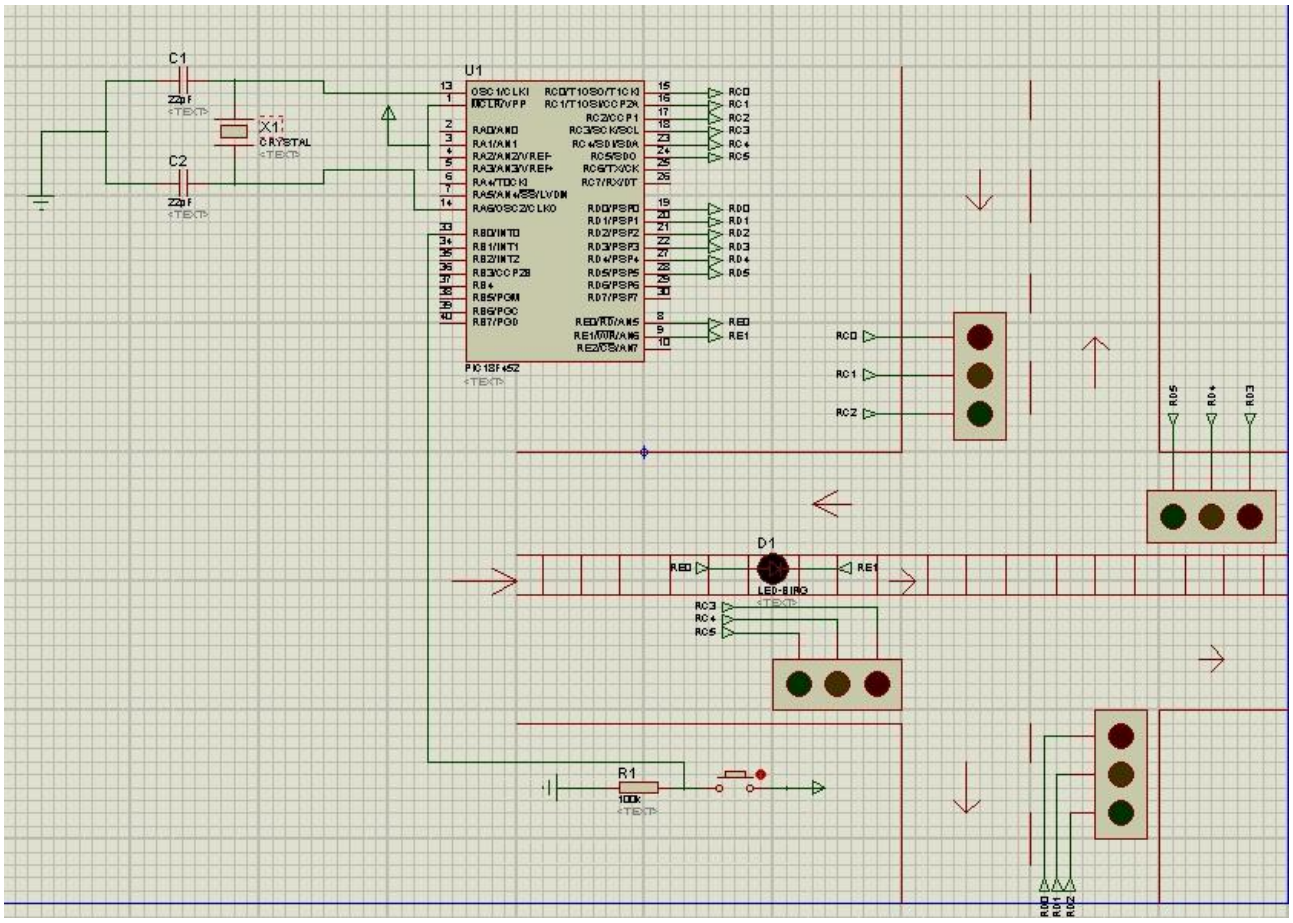
PIC bu uygulamada kavşağın normal akışı düzenlerken tramvay hattından gelen sinyalle tüm trafik ışıklarını kırmızı yaparak tramvay için Led'e ters yönde akım vererek Led ışığını değiştirerek geçişini sağlamaktadır.Devrede PIC'in PORTC ve PORTD ile trafik lambalarını,PORTE ile Led'i kontrol edilmektedir.Kesme bit'i olarak INT0 belirlenmiştir.RE0 bit'ini 0, RE1 bit'ini 1 yaparak Led'in kırmızı yanmasını INT0 bit'i 0 durumundayken butonla 1 yapılarak kesme sinyali verilmektedir.Kesme oluşması ile RE0 ile RE1 bit'leri değer değiştirilerek Led yeşil renge dönüşmesi sağlanarak tramvay geçişi sağlanmaktadır.

----------KOD(main.c)----------

```
#include <htc.h>

#include "Delay.h"

unsigned char isik[]={0x09,0x0A,0x0C,0x12,0x21,0x11};

int j[]={0,1,2,3,4,5,0,0,0};int k[]={4,5,0,0,0,1,2,3,4};

int i,x=0,y=0,q;

void main(void) {

TRISB=0x01;TRISC=0x00;TRISD=0x00;TRISE=0x00;PORTB=0x00;PORTC=0x00;
```

Tarih	Konu	İMZA-Mühür-Kaşe
02.08.2011	PIC18F452 Uygulamaları	

```
PORTD=0x00;PORTE=0x00;

INT0IF=0;INTEDG0=1;

INT0IE=1;GIE=1;PORTE=2;

for(;;){

for(i=0;i<10000;i++){

PORTC=isik[j[x++]];

PORTD=isik[k[y++]];

DelayMs(85);}}}

static void interrupt isim(void) {

if(INT0IF) {for(i=0;i<30000;i++){

PORTC=isik[0];PORTD=isik[0];PORTE=1;}}}
```



Devre 9:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
02.08.2011	PIC18F452 Uygulamaları	

2.3.4-)TİMER,CAPTURE,COMPARE,PWM,TUŞ TAKIMI,LCD VE ADC UYGULAMALARI

A-)TİMER UYGULAMALARI

A.1-)Timer0 ile Sayma Uygulaması

Bu uygulamada Timer0 birimini harici clock kullanarak Led 500ms bir yanıp sönmesini sağlamaktır.Bu işlemi yapabilmek için aşağıdaki Timer0 kontrol eden register ile gerekli ayarlamaları yaptık dan sonra dahili clock üretimi ile her 0,5 saniyede Led yanıp sönecektir.

T0CON: TIMER0 Kontrol Eden Register

TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

TMR0ON:Timer0 açma kapatma biti(0:Kapatma,1:Açma)
T08BIT:Timer0 8 bit veya 16 bit Timer/Counter seçme biti(0: 8 bit timer 1: 16 bit timer)
T0CS : Timer0 sinyal seçim biti (0: Dahili, 1: Harici)
T0SE : Timer0 kenar seçim biti (0: Yükselen kenar, 1: Düşen kenar)
PSA : Frekans bölücü seçim biti (0: Prescaler Timer0 için, 1: Prescaler WDT için)
T0PS2,T0PS1,T0PS0 : Bölüm oranını belirleyen bitler

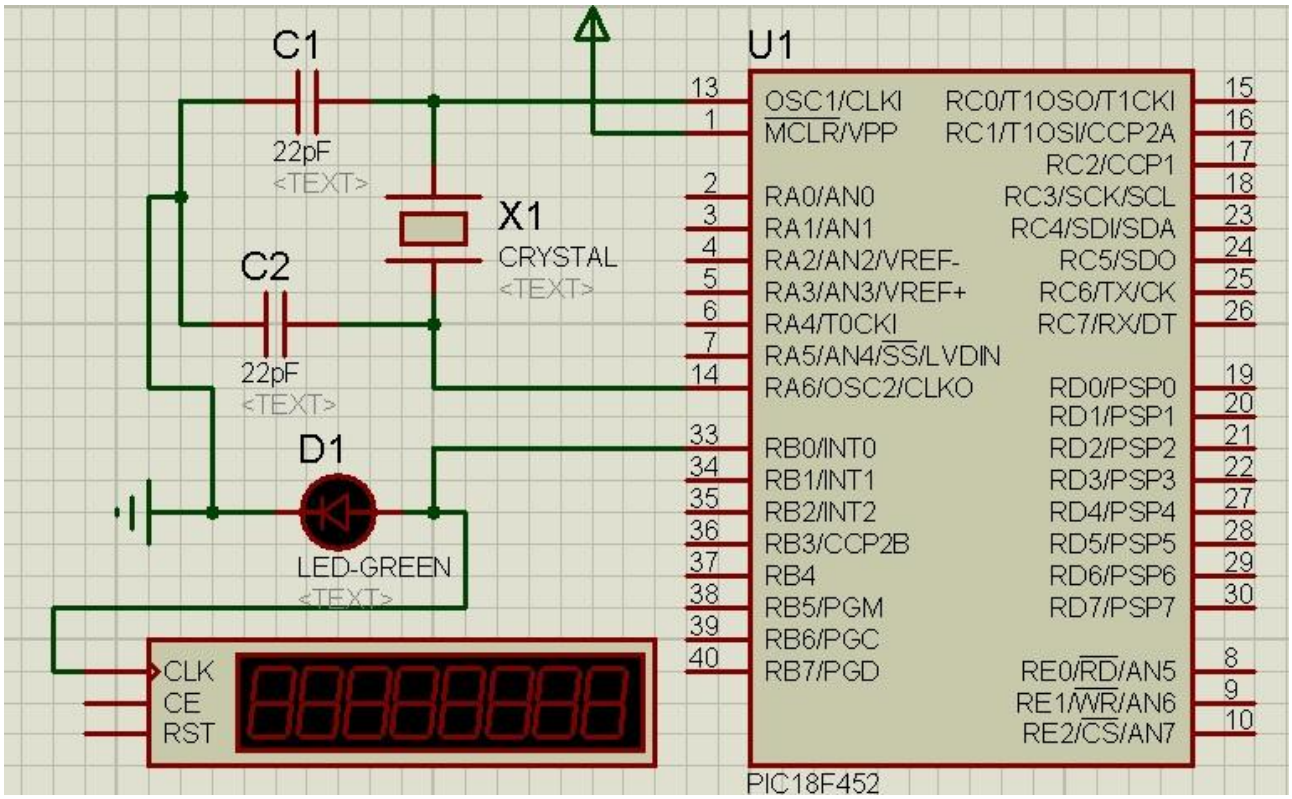
T0PS2	T0PS1	T0PS0	TMR0
1	1	1	1:256
1	1	0	1:128
1	0	1	1:64
1	0	0	1:32
0	1	1	1:16
0	1	0	1:8
0	0	1	1:4
0	0	0	1:2

TMR0IE : Timer0 kesme izin biti
TMR0IF : Timer0 kesme bayrak biti

----------KOD(main.c)----------

```
#include <htc.h>

#include "Delay.h"

void main(void) { // Ana fonksiyon alanı

TRISB=0x00;// PORTB çıkıs olarak ayarlanıyor

PORTB=0x00;

T0CON=0xD3;// Dahili osilatör

TMR0=55; // TMR0=55 oluyor

TMR0IF=0;        // TMR0 kesme bayrağı temizleniyor
```

```
TMR0IE=1; GIE=1;// TMR0 kesmesine izin veriliyor// Genel kesme izni veriliyor

for(;;)CLRWDT();}

static void interrupt  Kesme (void) { // Kesme fonksiyonu

        char i=0; // 125'e kadar sayacak değisken

        if(TMR0IF) {// TMR0 kesmesi olusmus mu

                GIE=0; i++;// Kesme varsa, baska kesme gelmemesi için genel kesme sıfırlanıyor
                if(i>0 & i<125) RB0=1;// 0 ile 125 arası 1 ol

                else if(125<i && i<250) RB0=0; //125 ile 250 arası 0 ol

                else if(i>250) i=0;// 250'yi asarsa değisken 0 olsun

                TMR0=55; // 255-55=200 birim sayacak

                TMR0IF=0; // Tekrar dıs kesme alınabilmesi için kesme bayrağı temizleniyor

                GIE=1; // Tekrar kesme alınabilmesi için genel kesme bayrağı set ediliyor}}
```



Devre 10:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
05.09.2011	PIC18F452 Uygulamaları	

A.2-) Timer2 Uygulaması

Bu uygulamada Timer2 kullanarak Timer2 ile artırılan değişkenin değerini Led üzerinden takip edeceğiz.Bu takip etme işlemini de dışarıdan alınan bir sinyalle RA0 girişinden sağlayacağız Timer2 geniş bir Postscale tablosuna sahiptir.Bu Postscale bize iyi bir esneklik kazandırır.T2CON Timer2 kontrol eden register gerekli ayarları aşağıda verilen bilgiler doğrultusunda yapılmalıdır.

T2CON: TIMER2 Kontrol Eden Register

-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS0	T2CKPS0

bit 7 bit 0

TMR2=PR2 : Timer2 sayıcısının yüksek değerlikli bitini tutan kaydedici
TOUTPS3, TOUTPS2, TOUTPS1, TOUTPS0 : Postscale değerleri

TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	Postscale
0	0	0	0	1:1
0	0	0	1	1:2
0	0	1	0	1:3
0	0	1	1	1:4
0	1	0	0	1:5
0	1	0	1	1:6
0	1	1	0	1:7
0	1	1	1	1:8
1	0	0	0	1:9
1	0	0	1	1:10
1	0	1	0	1:11
1	0	1	1	1:12
1	1	0	0	1:13
1	1	0	1	1:14
1	1	1	0	1:15
1	1	1	1	1:16

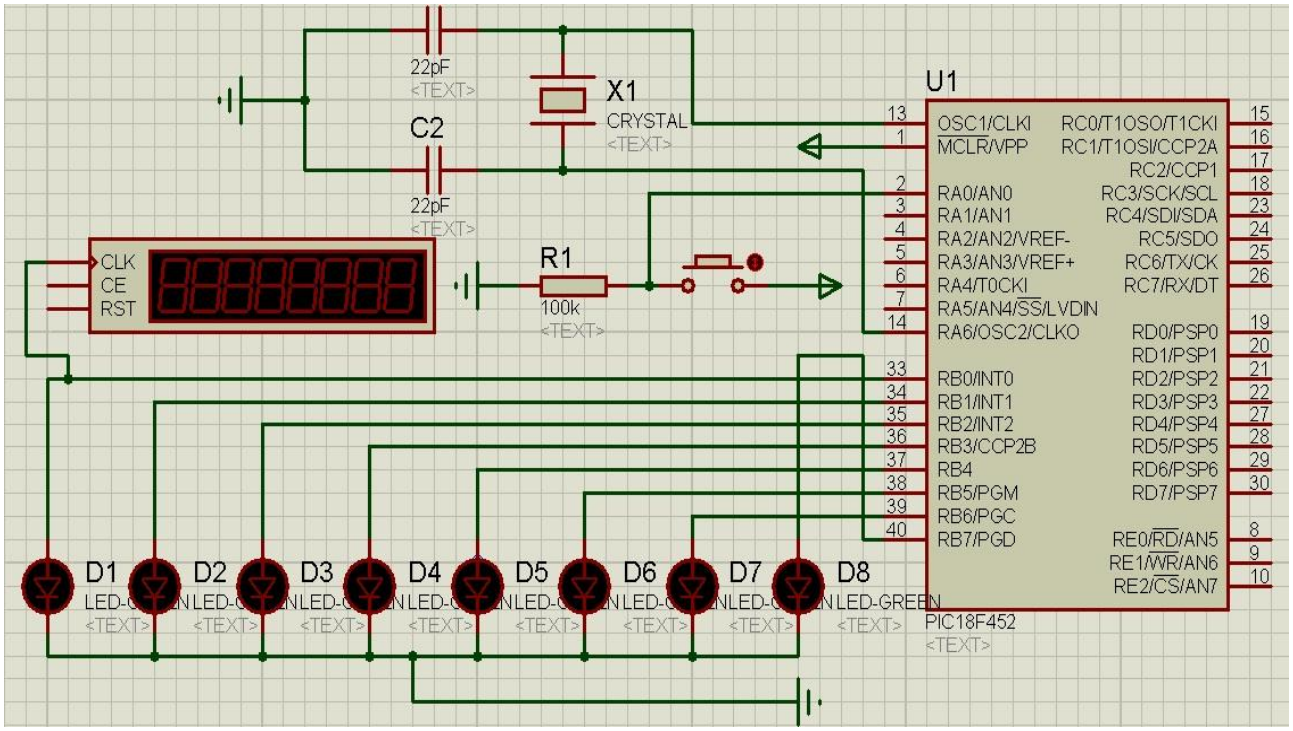
TMR2ON : Timer2 sayıcısını açma biti
T2CKPS1, T2CKPS0 : Prescaler değerler (1:1, 1:4, 1:16)
TMR2IE : Timer2 kesme izin biti
TMR2IF : Timer2 kesme bayrak biti

----------KOD(main.c)----------

```
#include <htc.h>

char i; // Genel değisken tanımlanıyor

void main(void) // Ana fonksiyon alanı

{ADCON1=0x07; // PORTA dijital yapılıyor
```

```
TRISA=0x01; // RA0 giris

TRISB=0x00; // PORTB çıkıs olarak ayarlanıyor

PORTA=0x00; // PORTA sıfırlanıyor

PORTB=0x00; // PORTB sıfırlanıyor

PR2=250; // PR2 değerine 250 yükleniyor

T2CKPS1=1; // Prescaler 1:16 oluyor

T2CKPS0=1; // Prescale 1:16 oluyor

TOUTPS3=1;

TOUTPS2=1;

TOUTPS1=1;

TOUTPS0=1;

TMR2IF=0; // TMR1 kesme bayrağı temizleniyor

TMR2IE=1; // TMR1 kesmesine izin veriliyor

TMR2ON=1; // TMR1 çalıstırılıyor Devamı Sağ //Tarafta
```

```
PEIE=1; // Yardımcı kesme izni veriliyor

GIE=1; // Genel kesme izni veriliyor

for(;;)

{if(RA0) // Butona basıldı mı

PORTB=i; // Değiskenin değeri PORTB'ye //yansıtılıyor}}

static void interrupt  Kesme (void)// Kesme fonksiyonu

{if(TMR2IF) // TMR2 kesmesi olusmus mu

{GIE=0; // Kesme varsa, baska kesme gelmemesi //için genelkesme sıfırlanıyor

i++; // Değisken 1 artırılıyor

TMR2IF=0; // Tekrar dıs kesme alınabilmesi için //kesmebayrağı temizleniyor

GIE=1; // Tekrar kesme alınabilmesi için genel //kesmebayrağı set ediliyor}}
```



Devre 11:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
06.09.2011	PIC18F452 Uygulamaları	

B-)CAPTURE,COMPARE VE PWM(Pulse Width Modulation)

B.1-) Capture Uygulaması

Capture birimin amacı yakalama yapaktır ve Timer1 veya Timer3 ile beraber çalışmaktadır.Capture,Compare ve PWM modülleri CCP1CON/CCP2CON register lar ile kontrol edilirler.Bu iki register la ilgili bilgiler aşağıda verilmiştir.Bu uygulamada CCP1'in her düşen kenar yakalamasında bir değişkeni artırması,CCP2'nin ise her 4. yükselen kenar yakalamasında yine aynı değişkeni bir azaltması ve bu değişkenin PORTB'den görülmesi amaçlanmıştır.

CCP1CON/CCP2CON REGISTER

----	----	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

CCPxX, CCPxY : PWM düşük değerlikli bitleri. Yüksek değerlikli 8 bit ise CCPRxL kaydedicisinde bulunur.

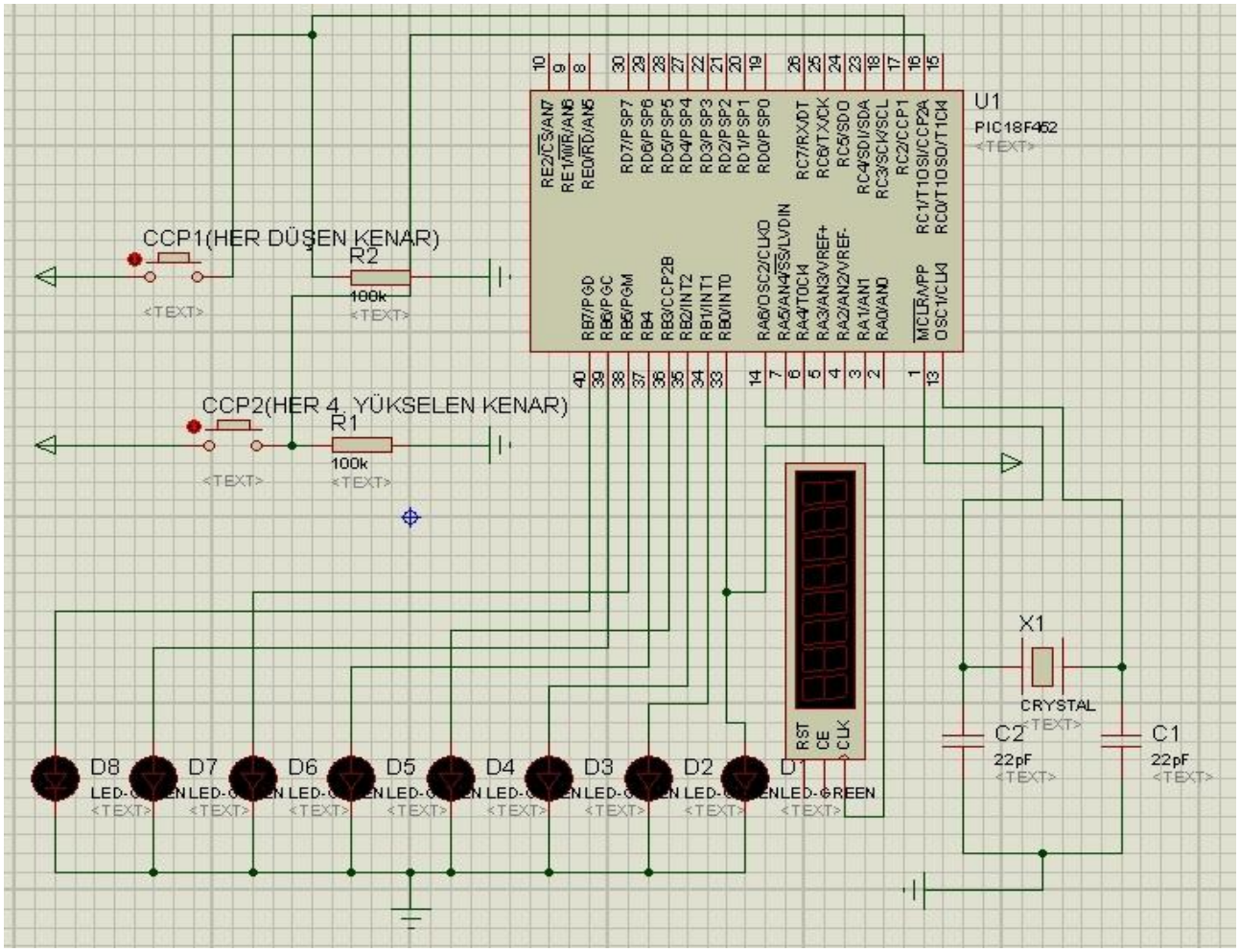
CCPxM3, CCPxM2, CCPxM1, CCPxM0 : CCP mod seçme bitleri. Modların ne olacakları aşağıdaki gösterilmiştir.

0000 : CCP etkin değil
0100 : Capture modu, her düşen kenarda
0101 : Capture modu, her yükselen kenarda
0110 : Capture modu, her 4. yükselen kenarda
0111 : Capture modu, her 16. yükselen kenarda
1000 : Compare modu, denklik durumunda CCPx pini 1 olsun, CCPxIF bayrağı çekilsin
1001 : Compare modu, denklik durumunda CCPx pini 0 olsun, CCPxIF bayrağı çekilsin
1010 : Compare modu, denklik durumunda CCPx pini değişmesin, CCPxIF bayrağı çekilsin
1011 : Compare modu, denklik durumunda CCPx pini değişmesin, CCPxIF bayrağı çekilsin
11xx : PWM modu

----------KOD(main.c)----------

```
#include <htc.h>

char i; // Genel değisken tanımlanıyor

void main(void) // Ana fonksiyon alanı

{

TRISB=0x00; // PORTB çıkıs olarak ayarlanıyor

TRISC=0x06; // CCP1 ve CCP2 giris

PORTB=0x00; // PORTB sıfırlanıyor

PORTC=0x00; // PORTC sıfırlanıyor
```

```
CCP1CON=0x04; // CCP1 her düşen kenar modunda
CCP2CON=0x06; // CCP1 her 4. yükselen kenar modunda
CCP1IF=0; // CCP1 ve CCP2 kesme bayrakları temizleniyor
CCP2IF=0;
CCP1IE=1; // CCP1 ve CCP2 kesme izinleri veriliyor
CCP2IE=1;
PEIE=1; // Yardımcı kesme izni veriliyor
GIE=1; // Genel kesme izni veriliyor
for(;;)CLRWDT();}
static void interrupt Kesme(void) // Kesme fonksiyonu
{if(CCP1IF) // CCP1 kesmesi varsa Devamı Sağ Tarafta
{GIE=0; // Baska kesme gelmesi engelleniyor
i++; // Değisken bir artırılıyor
PORTB=i; // Değisken değeri PORTB'ye yansıtılıyor
CCP1IF=0; // Yeni CCP1 kesmesi için bayrak temizleniyor
GIE=1; // Genel kesme izni veriliyor}
if(CCP2IF) // CCP2 kesmesi varsa
{GIE=0; // Baska kesme gelmesi engelleniyor
i--; // Değisken bir azaltılıyor
PORTB=i; // Değisken değeri PORTB'ye yansıtılıyor
CCP2IF=0; // Yeni CCP2 kesmesi için bayrak temizleniyor
GIE=1; // Genel kesme izni veriliyor}}
```



Devre 12:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
06.09.2011	PIC18F452 Uygulamaları	

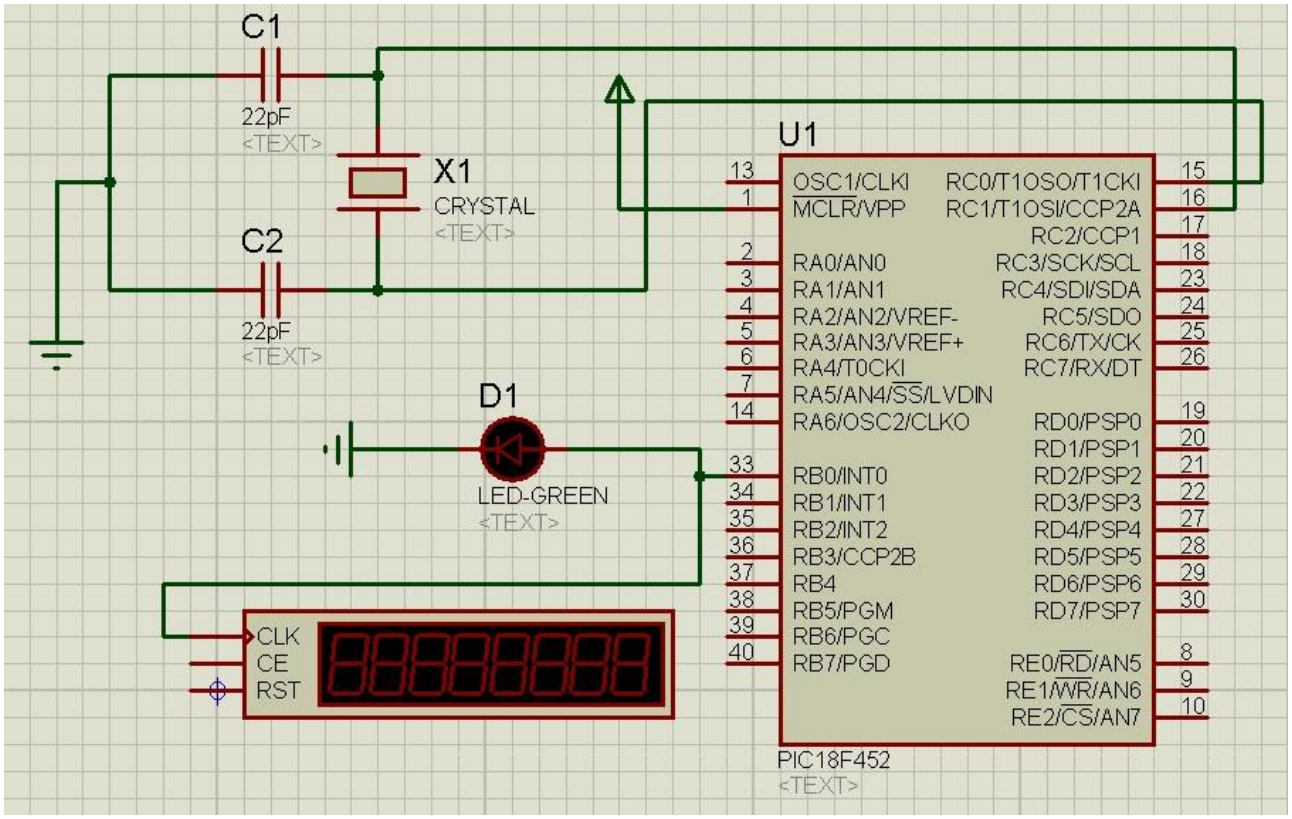
B.2-) Compare Uygulaması

Compare modül Timer1 ve Timer3 ile beraber çalıştırılmalıdır.Capture ve Compare modüllerinde iki tane register kullanılır.Capture modülü yakalama için Compare modülü karşılaştırma için kullanılır ve CCPR1H,CCPR1L registerlar 8'er bitlik registerlardır. CCPR1L register ilk biti bit0 ve CCPR1H register son biti bizim kullanacağımız 16 bitlik alanın bit15 oluşturur.Bu uygulamada Timer1 harici clock kullanılarak karşılaştırma yapılan sayıya rastlandığında kesme gerçekleşerek RB0 bağlı Led yanıp sönecektir.

----------KOD(main.c)----------

```
#include <htc.h>
void main(void) // Ana fonksiyon alanı
{
TRISB=0x00; // PORTB çıkıs olarak ayarlanıyor
PORTB=0x00; // PORTB sıfırlanıyor
TMR1CS=0; // Timer1 Harici kaynaktan besleniyor
T1SYNC=0; // Senkronizasyon yok
TMR1ON=1; // Timer1 açılıyor
CCP1M0=1; // CCP1 compare modunda, timer1 resetlenecek
CCP1M1=1;
CCP1M2=0;
CCP1M3=1;
CCPR1H=500/256; // 500'e esitlenecek
CCPR1L=500%256;
CCP1IF=0; // CCP1 ve CCP2 kesme bayrakları temizleniyor
CCP1IE=1; // CCP1 ve CCP2 kesme izinleri veriliyor
PEIE=1; // Yardımcı kesme izni veriliyor
GIE=1; // Genel kesme izni veriliyor
for(;;)CLRWDT();}
static void interrupt Kesme(void) // Kesme fonksiyonu
// Kesme fonksiyon ismi (önemsiz)
{
char i; // Değiskenler tanımlanıyor
if(CCP1IF) // CCP1 kesmesi varsa
{
GIE=0; i++;  // Baska kesme gelmesi engelleniyor
// Değisken bir artırılıyor
if(i==1) // Değisken 1 ise RB0=1 olur
RB0=1;
else if(i==2) // Değisken 2 ise RB0=0 olur
{
RB0=0;
i=0;
}
CCP1IF=0; // Yeni CCP1 kesmesi için bayrak temizleniyor
GIE=1; // Genel kesme izni veriliyor}}
```

Tarih	Konu	İMZA-Mühür-Kaşe
07.09.2011	PIC18F452 Uygulamaları	



Devre 13:Proteus üzerindeki devre çizimi

B.3-)PWM Uygulaması

PWM modülü sinyal üretmek için kullanılır.Sinyalin PIC ile beraber kullanım alanları mevcuttur PIC ler arası haberleşmede veri göndermede kullanılmaktadırlar.PWM sinyalinde önemli olan noktalardan biride doluluk oranıdır.Bu oran:

PWM period = [(PR2) + 1] • 4 • TOSC •(TMR2 prescale value)

PWM duty cycle = (CCPR1L:CCP1CON<5:4>) •TOSC • (TMR2 prescale value)

PWM sinyali çözünürlüğü ise **PWM Resolution (max) = Log(F_{OSC}/F_{PWM})/Log(2)** şeklinde hesaplanır.Bu uygulamada duty cycle değiştirmeden PWM periyodunu değiştirerek buzzer ve osilatör üzerinden üretilen sinyalin değiştiğini gözlemleyeceğiz.Timer2, PWM modülü ile birlikte çalışmaktadır.

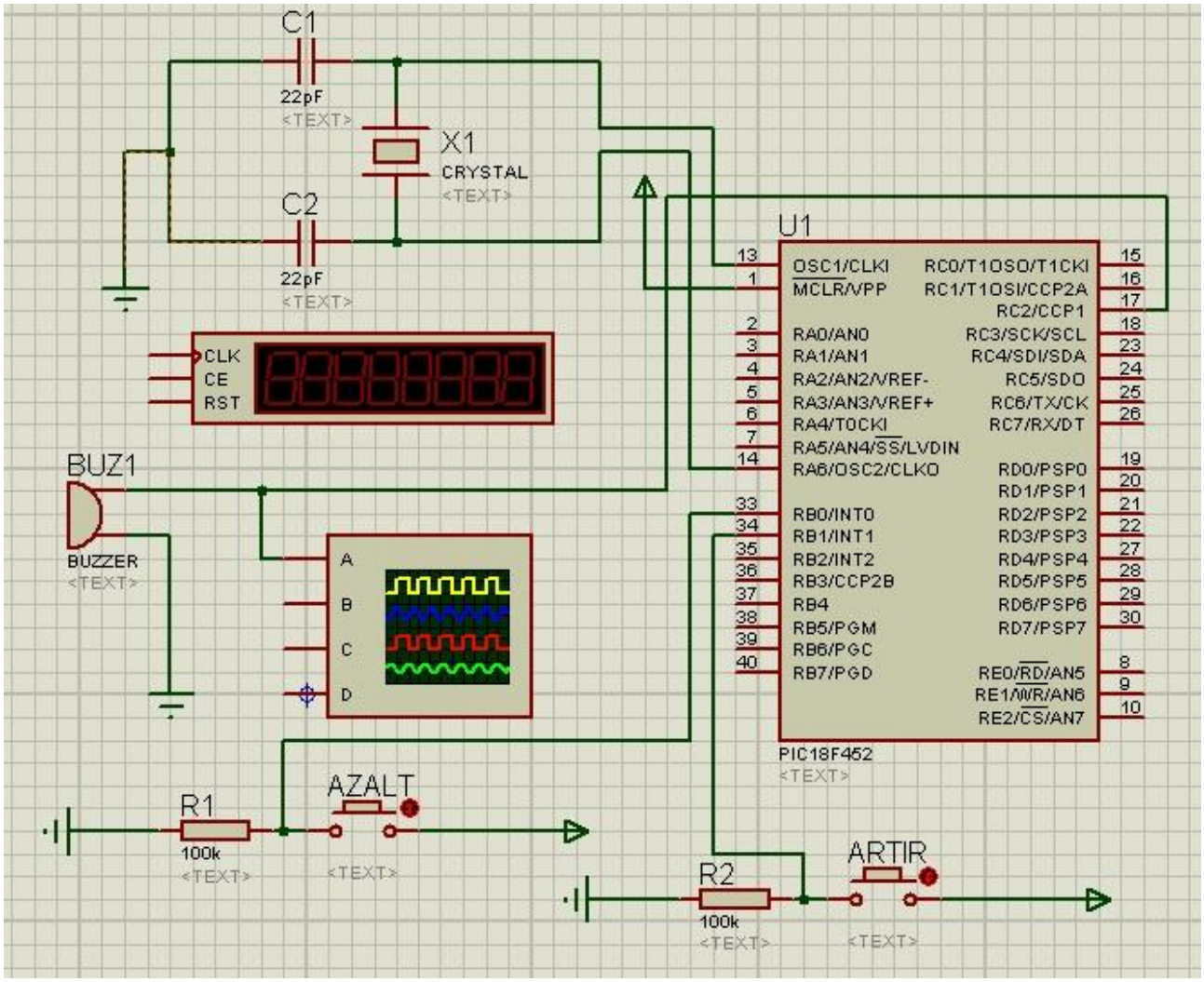
----------KOD(main.c)----------

```
#include <htc.h>
void main(void) // Ana fonksiyon alanı
{
char i=100;
ADCON1=0x07; // PORTA dijital oluyor
TRISB=0x03; // RA0 ve RA1 giris
TRISC=0x00; // PORTC çıkıs olarak ayarlanıyor
PORTB=0x00; // PORTA sıfırlanıyor
PORTC=0x00; // PORTC sıfırlanıyor
CCPR1L=0x3E; // Duty registere 250 yükleniyor
```

Tarih	Konu	İMZA-Mühür-Kaşe
07.09.2011	PIC18F452 Uygulamaları	

```
CCP1X=1; // Duty cycle 1ms periyodunda
CCP1Y=0;
T2CKPS1=1; // Prescaler 1:16 oluyor
T2CKPS0=1;
TOUTPS3=0; // Postscale 1:1 oluyor
TOUTPS2=0;
TOUTPS1=0;
TOUTPS0=0;
CCP1M0=1; // CCP1 PWM modunda
CCP1M1=1;
CCP1M2=1;
CCP1M3=1;
TMR2ON=1; // Timer 1 çalıstırılıyor
for(;;)
{CLRWDT();//Devamı Sağ Tarafta
```

```
if(RB0) // Azalt butonuna basılmıs mı
{
while(RB0); // Butondan elin çekilmesi bekleniyor
i-=5; // PR2 birimi 5 azaltılıyor
if(i<65) // Değisken 65'ten küçükse
i=65; // Tekrar 65'e esit olsun
}
if(RB1) // Artır butonuna basılmıs mı
{
while(RB1); // Butondan elin çekilmesi bekleniyor
i+=5;
if(i>250) // Değisken 250'den bütükse
i=250; // Tekrar 250'ye esit olsun
}
PR2=i; // Değisken PR2'ye esitleniyor}}
```



Devre 14:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
07.09.2011	PIC18F452 Uygulamaları	

C-)TUŞ TAKIMI ,LCD VE ADC UYGULAMALARI

C.1-)Tuş Takımı ve LCD Uygulaması

Tuş takımını çalışma prensibi satır ve sütunlardan oluşmaktadır.Satırlara sürekli olarak sinyal verilerek sütunlar taranır yada sütunlara sinyal verilerek satırlar taranır.Satırlara PORTE üzerinden sürekli sinyal vererek sütunlar PORTD üzerinden taranmaktadır.LCD çalışması kullanılan modele göre değişmektedir.Bu uygulamada LM016L modelini kullanmaktayız.Bu modelde 8bit data girişi vardır.Biz uygulamamızda yaygın bir kullanım olan 4 bit üzerinden veri gönderme şeklini tercih ettik.LCD'nin data haricinde RS,RW,E girişleri bulunmaktadır.

RS : Gelen bilginin komut mu data mı olduğu bu uçla belirlenir(0: Komut, 1: Data)
RW : LCD'ye veri yazma ya da okuma yetkilendirme ucudur (0: Yazma, 1: Okuma)
E : Enable ucudur, LCD'ye bilgi giriş çıkışını kontrol eden uçtur, düşen kenar tetiklemelidir.
D0,**D1..D7**: Data uçlarıdır, bilgi giriş çıkışları bu bacaklar sayesinde olur.

LCD üzerinde yazı yazılma işlemlerinin yapılabilmesi için Datasheet çalışma bilgileri kullanılır.Bu bilgiler şöyledir:

Display sıfırlanmalı : Bunun için LCD'ye 0x02 veya 0x03 gönderilip 2ms beklenir
Display silinmeli : Bunun için LDD'ye 0x01 gönderilip 1ms beklenilmeli
Fonksiyon ayarı yapılmalı : 4 bitlik iletişim 5x7 karakter kullanacağımızdan 0x28 gönderilip 40us beklenilmeli
Giriş modu ayarlanmalı : İmleç her karakterden sonra sağa kayacağından 0x06gönderilip 40us beklenilmeli
İmleç ayarı : Display kapalı uygulama yapacağımızdan 0x0B gönderilir

Bu bilgiler doğrultusunda LCD için lcd.h ve lcd.c dosyaları oluşturulmalıdır.
lcd.h:Fonksiyonların tanımlanması, lcd.c:Tanımlanan fonksiyonların işlevlerinin belirlenmesi

----------KOD(lcd.h)----------

```
#define rs RC0 //Pin tanımlamaları
#define rw RC1
#define e RC2
#define lcd_port PORTB
/* LCD'de kullanilan komutlarin tanimlamasi*/
#define Sil 1 // Ekrani temizler
#define BasaDon 2 // Imleci sol üst köseye getirir
#define SolaYaz 4 // Imlecin belirttigi adres azalarak gider
#define SagaYaz 6 // Imlecin belirttigi adres artarak gider
#define ImlecGizle 12 // Göstergeyi ac, kursor görünmesin
#define ImlecAltta 14 // Yanip sönen blok kursor
#define ImlecYanSon 15 // Yanip sönen blok kursor
#define ImlecGeri 16 // Kursoru bir karakter geri kaydır
#define KaydirSaga 24 // Göstergeyi bir karakter sağa kaydır
#define KaydirSola 28 // Göstergeyi bir karakter sola kaydır
#define EkraniKapat 8 // Göstergeyi kapat (veriler silinmez)
#define BirinciSatir 128 // LCD'nin ilk satir baslangıç adresi//(DDRAM adres)
#define IkinciSatir 192 // Đkinci satirin baslangıç adresi
#define KarakUretAdres 64 // Karakter üreteci adresini belirle//(CGRAM adres)
/* LCD'de Kullanilan Fonksiyon Seçimi */
#define CiftSatir8Bit 56 // 8 bit ara birim, 2 satir, 5*7 piksel
#define TekSatir8Bit 48 // 8 bit ara birim, 1 satir, 5*7 piksel
#define CiftSatir4Bit 40 // 4 bit ara birim, 2 satir, 5*7 piksel
#define TekSatir4Bit 32 // 4 bit ara birim, 1 satir, 5*7 piksel
extern void veri_yolla(unsigned char);
extern void lcd_clear(void);
extern void lcd_yaz(const char *s);
extern void lcd_gotoxy(unsigned char x, unsigned char y);
extern void lcd_init(void);
extern void lcd_komut(unsigned char c);
```

----------KOD(lcd.c)----------

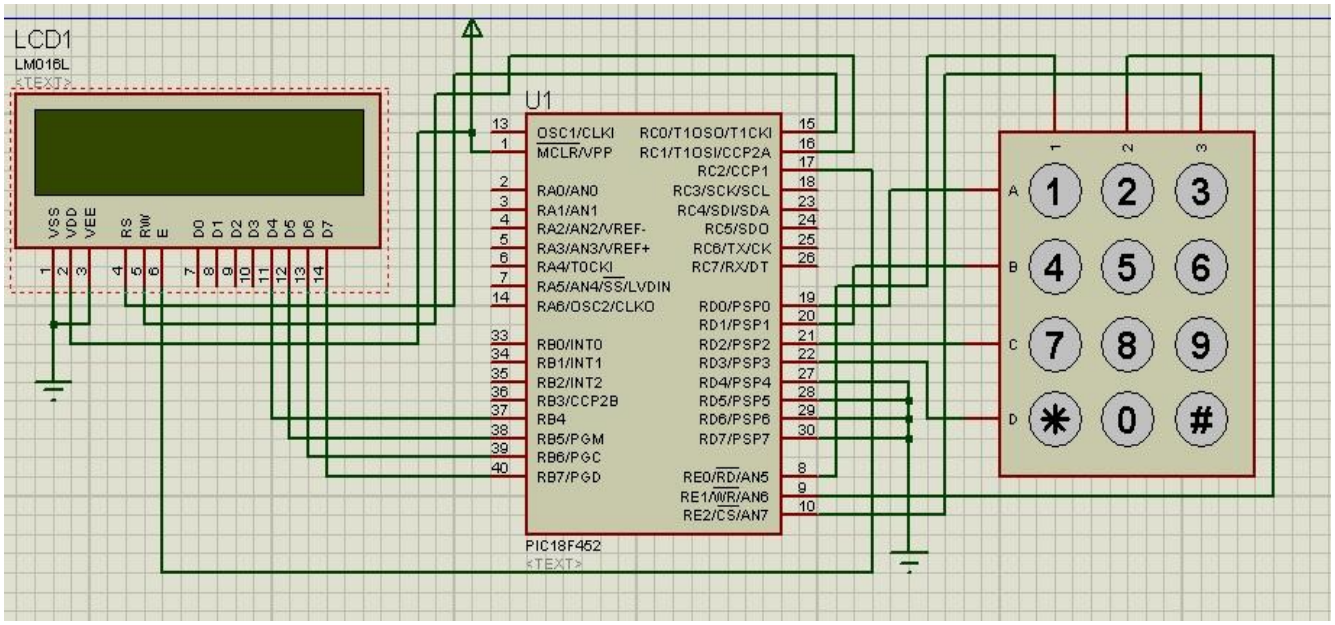
```
#include <pic18.h>
#include "lcd.h" // lcd.h dosyası tanımlanıp, değerler alınıyor
#include "delay.h" // Gecikme fonksiyonu tanımlanıyor
void lcd_busy(void) // 500us bekleme
{
DelayUs(250);
DelayUs(250);
}
void lcd_komut(unsigned char c) // Komut gönderme fonksiyonu
{
rw=0; // LCD'ye yazma yapılacak
rs=0; // LCD'ye komut gönderilecek
e=1; // Düsen kenar tetikleme olduğu için E önce 1
lcd_port = ( c & 0xF0 ); // Yüksek değerlikli bitler gönderiliyor
e=0; // E, 0 yapılıyor
lcd_busy(); //Belirli süre bekleniyor
e=1; // E, 1 yapılıyor
lcd_port = ( (c & 0x0F)<<4 ); // Düsük değerlikli bitler
gönderiliyor
e=0; // E, 0 yapılıyor
lcd_busy(); // Belirli bir süre bekleniyor
}
void veri_yolla(unsigned char c)
{
rw=0; rs=1; // Komut yolladan tek farkı, RS'nin 1 olması
e=1;
lcd_port = ( c & 0xF0 );
e=0;
lcd_busy();
e=1;
lcd_port = ( (c & 0x0F)<<4 );
e=0;
lcd_busy();
}
void lcd_clear(void) // LCD siliniyor
{
lcd_komut(0x1);
DelayMs(2);
}
void lcd_yaz(const char * s) // LCD'ye string ifade gönderiliyor
{
lcd_busy();
while(*s)
veri_yolla(*s++);
}
void lcd_gotoxy(unsigned char x,unsigned char y) //LCD'nin belli
//bölgesine gidiliyor
{
if(x==1)
lcd_komut(0x80+((y-1)%16));
else
lcd_komut(0xC0+((y-1)%16));}
void lcd_init() // LCD ilk yükleme ayarları yapılıyor
{
rs = 0;e = 0;rw = 0;DelayMs(15);e=1;lcd_komut(0x02);DelayMs(2);
lcd_komut(CiftSatir4Bit);lcd_komut(SagaYaz);lcd_komut(ImlecGizle);
lcd_clear();lcd_komut(BirinciSatir);}
```

Tarih	Konu	İMZA-Mühür-Kaşe
08.09.2011	PIC18F452 Uygulamaları	

----------KOD(main.c)----------

```
#include<htc.h>

#include "Delay.h"

#include "lcd.h"

unsigned char satir[]={1,2,4};

unsigned char sutun[]={1,2,4,8};

unsigned char sayilar[4][3] =

{1,2,3,4,5,6,7,8,9,42,0,35};

unsigned char keypad_oku(void)

{

PORTD=0x00;

int a,b;char t=12;

for(a=0;a<3;a++)

for(b=0;b<4;b++)

{

PORTE=satir[a];

if(PORTD==sutun[b])

t=sayilar[b][a];

}

return t;

}

void main(void)

{

unsigned char say;

TRISB=0x00;

TRISE=0x00;

TRISC=0x00;

TRISD=0xFF;

PORTB=0x00;

PORTE=0x00;

PORTD=0x00;

PORTC=0x00;

lcd_init();// LCD ilk ayarları
yapılıyor

lcd_yaz("Basilan Tus=");

for(;;)

{

CLRWDT();

say=keypad_oku();

lcd_gotoxy(1,13); // LCD 1x13'e
//gidiliyor

if(say==35)

veri_yolla(say);

else if(say==42)

veri_yolla(say);

else if(say<10&&say>=0)

veri_yolla(say+48);

else;

}

}
```

Tarih	Konu	İMZA-Mühür-Kaşe
08.09.2011	PIC18F452 Uygulamaları	



Devre 15:Proteus üzerindeki devre çizimi

C.2-)ADC Uygulaması(Voltmetre)

ADC analog bir veriyi(volt,sıcaklık..vb) dijital veriye çeviren modüldür.Bu uygulamada ilk iki analog kanalına bağlı potlardan okunan 0-5V arasındaki voltajların LCD'de görünmesi ve kesmeden faydalanarak her çevrim kesimine gidildiğinde RC5'e bağlı Led'in yanıp sönmesi amaçlanmaktadır.ADC modül kontrolü ADCON0 ve ADCON1 ile sağlanmaktadır.

ADCON0 REGİSTER

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	---	ADON

bit 7 bit 0

ADCON1 REGİSTER

ADFM	ADCS2	---	---	PCFG3	PCFG2	PCFG1	PCFG0

bit 7 bit 0

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC

GO/DONE: ADC çevrimini başlatan bittir,çevrim bitince 0 olur.
ADON:ADC birimini açan bit(1:Açık 0:Kapalı)
ADFM: ADRESH ve ADRESL register'ların sağa mı sola mı yaslanacağını seçme bit(1:Sağa 0:Sola)
ADCS2: Yardımcı frekans seçme biti.

PCFG3, PCFG2, PCFG1, PCFG0: Portların Analog/Dijital Giriş Çıkış olarak ayarlanması için kullanılan bitlerdir.

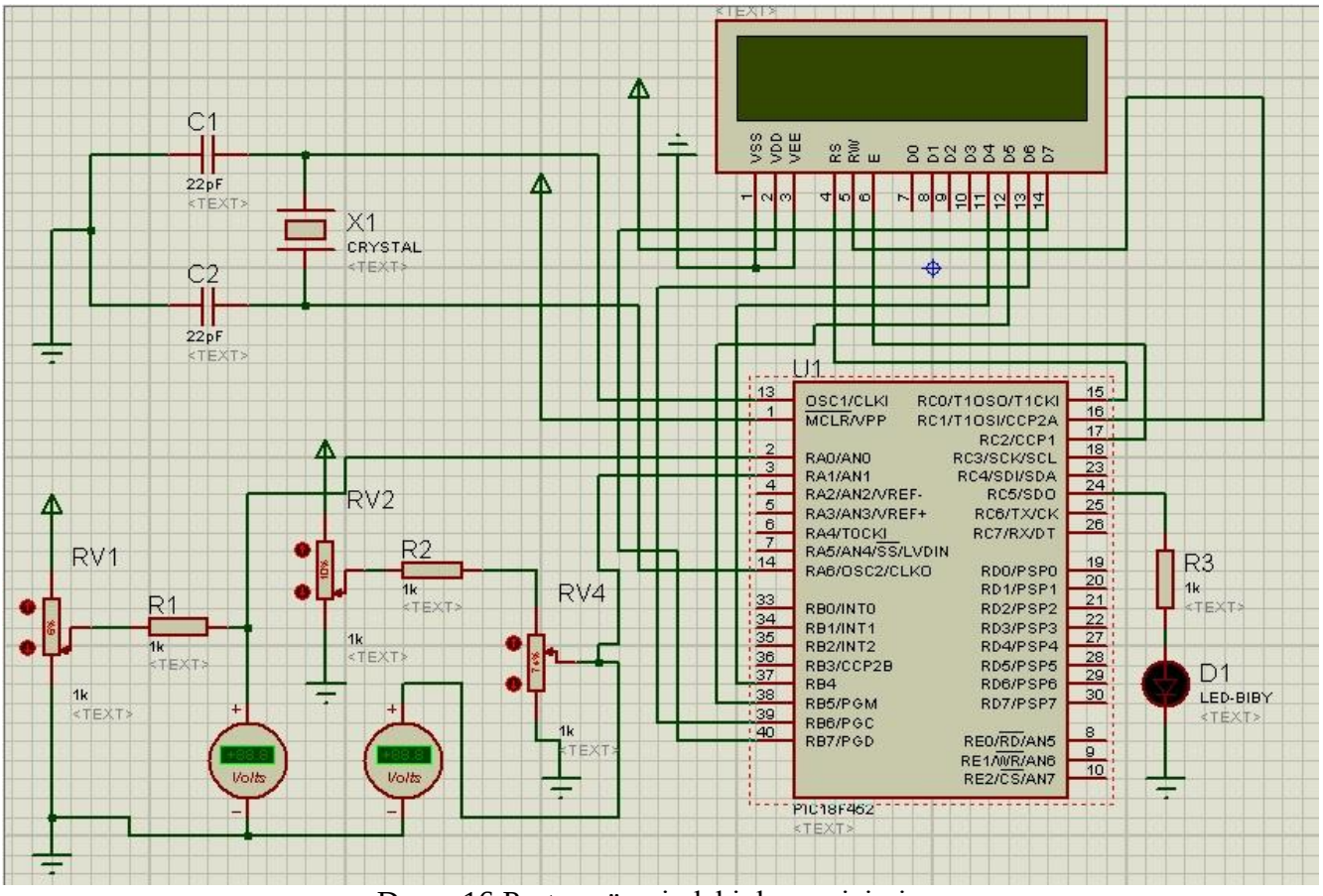
PCFG <3;0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	$V_{REF}+$	$V_{REF}-$	C/R
0000	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}	8/0
0001	A	A	A	A	$V_{REF}+$	A	A	A	AN3	V_{SS}	7/1
0010	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}	5/0
0011	D	D	D	A	$V_{REF}+$	A	A	A	AN3	V_{SS}	4/1
0100	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}	3/0
0101	D	D	D	D	$V_{REF}+$	D	A	A	AN3	V_{SS}	2/1
011x	D	D	D	D	D	D	D	D	----	----	0/0
1000	A	A	A	A	$V_{REF}+$	$V_{REF}-$	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}	6/0
1010	D	D	A	A	$V_{REF}+$	A	A	A	AN3	V_{SS}	5/1
1011	D	D	A	A	$V_{REF}+$	$V_{REF}-$	A	A	AN3	AN2	4/2
1100	D	D	D	A	$V_{REF}+$	$V_{REF}-$	A	A	AN3	AN2	3/2
1101	D	D	D	D	$V_{REF}+$	$V_{REF}-$	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	V_{DD}	AN2	1/0
1111	D	D	D	D	$V_{REF}+$	$V_{REF}-$	D	A	AN3	V_{SS}	1/2

----------KOD(main.c)----------

```
#include <htc.h>
#include "delay.h" // Gecikme kütüphanesi tanımlanıyor
#include "lcd.h" // LCD kütüphanesi tanımlanıyor
int voltaj_1;
int voltaj_2;
char i=1;
void main(void) // Ana fonksiyon alanı
{
TRISA=0x03; // Analog giris için
TRISB=0x00; // LCD için çıkıs
TRISC=0x00;
PORTB=0x00;
PORTC=0x00;
ADCON1=0xC0; // AN0 ve AN1 analog// Sağa dayalı yazılıyor
ADON=1; // ADC açılıyor
ADIF=0; // ADC bayrağı temizleniyor
ADIE=1; // ADC kesmesi izni veriliyor
PEIE=1; // Genel ve yardımcı kesme izinleri veriliyor
GIE=1;
lcd_init(); // LCD ilk ayarları yapılıyor
lcd_yaz("1.Voltaj=");
lcd_gotoxy(2,1);
lcd_yaz("2.Voltaj=");
 for(;;)
{
CLRWDT();
ADCON0=0x05; // AN0 seçiliyor
DelayUs(25);// Çevrim baslatılıyor
```

Tarih	Konu	İMZA-Mühür-Kaşe
08.09.2011	PIC18F452 Uygulamaları	

```
while(!ADON);
voltaj_1=(int)((ADRESH*256+ADRESL)/2); // Hesaplama yapılıyor
lcd_gotoxy(1,10); // Okunan değer LCD'ye yazılıyor
veri_yolla(voltaj_1/1000+48);
veri_yolla((voltaj_1%1000)/100+48);
veri_yolla('.');
veri_yolla((voltaj_1%100)/10+48);
veri_yolla(voltaj_1%10+48);
veri_yolla('V');
ADCON1=0x0D; // AN1 seçiliyor
DelayUs(25);// Çevrim baslatılıyor
while(!ADON);
voltaj_2=(int)((ADRESH*256+ADRESL)/2); // Hesaplama yapılıyor
lcd_gotoxy(2,10); // Okunan değer LCD'ye yazılıyor
veri_yolla(voltaj_2/1000+48);
veri_yolla((voltaj_2%1000)/100+48);
veri_yolla('.');
veri_yolla((voltaj_2%100)/10+48);
veri_yolla(voltaj_2%10+48);
veri_yolla('V');}}
static void interrupt led_yaz_son(void)
{
if(ADIF) // Çevrim bitis kesmesi bekleniyor
{GIE=0; // Baska kesme gelmesi engelleniyor
i=!i; // Her kesme de değili alınıyor
RC5=i; // Değer RC5'e aktarılıyor
ADIF=0; // Kesme bayrağı sıfırlanıyor
GIE=1; // Genel kesme alımı açılıyor}}
```



Devre 16:Proteus üzerindeki devre çizimi

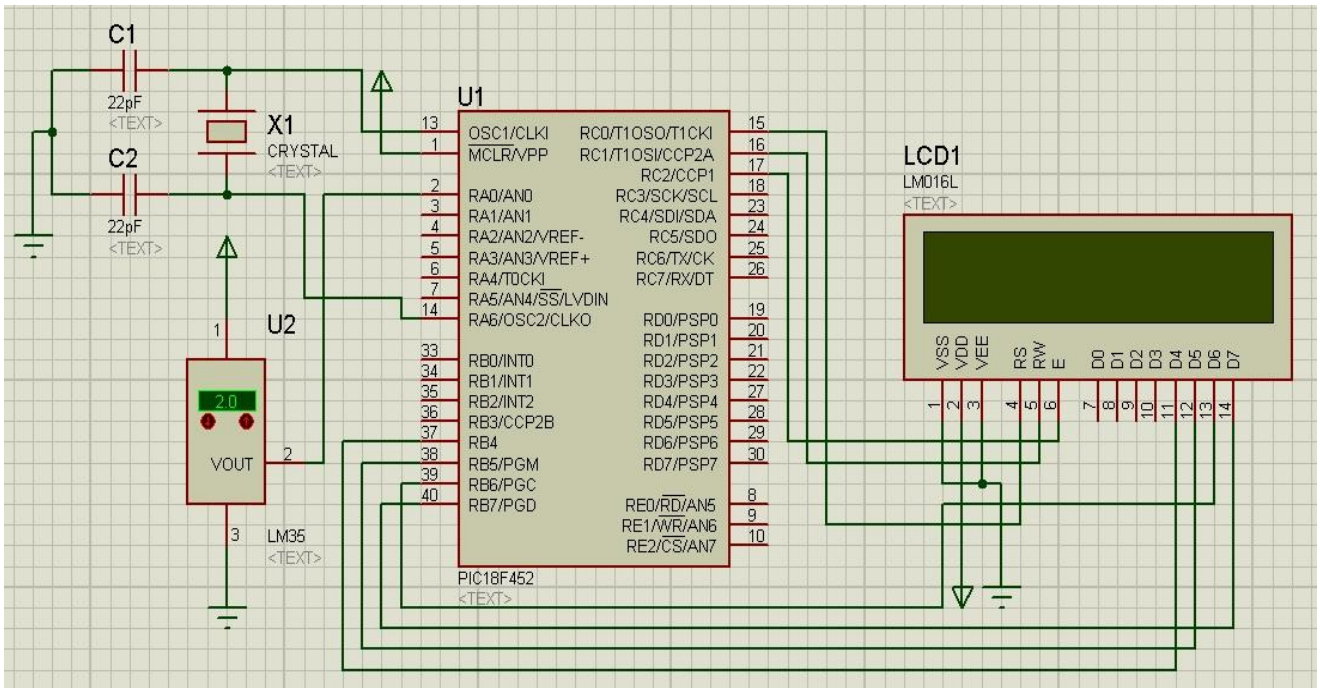
Tarih	Konu	İMZA-Mühür-Kaşe
09.09.2011	PIC18F452 Uygulamaları	

C.3-)ADC Uygulaması(Termometre)

Bu uygulamada aynı register ayarları kullanarak LM35 ile ortam sıcaklığını ölçmeyi amaçlıyoruz.LM35 1^0C değişimde 10 mV bir ile ölçüm yaparız. LM35, 3-4 derece hata payına sahip, kullanım kolaylığı itibariyle hassasiyet gerekmeyen sıcaklık uygulamalarında sıkça tercih edilen bir sıcaklık sensörüdür.

----------KOD(main.c)----------

```
#include <htc.h>
#include "delay.h" // Gecikme kütüphanesi tanımlanıyor
#include "lcd.h" // LCD kütüphanesi tanımlanıyor
void main(void) // Ana fonksiyon alanı
{int sicaklik;
TRISA=0x03; // Analog giris için
TRISB=0x00; // LCD için çıkıs
TRISC=0x00;PORTB=0x00;PORTC=0x00;PORTA=0x00;
ADCON1=0xC0;// AN0 analog // Sağa dayalı yazılıyor
ADON=1; // ADC açılıyor
lcd_init(); // LCD ilk ayarları yapılıyor
lcd_yaz("Sicaklik Degeri");
for(;;)
{CLRWDT();
ADCON0=0x05; // AN0 seçiliyor
DelayUs(25);
// Çevrim baslatılıyor
while(!ADON);
sicaklik=(int)((ADRESH*256+ADRESL)*48); // Hesaplama yapılıyor
lcd_gotoxy(2,5); // Okunan değer LCD'ye yazılıyor
veri_yolla(sicaklik/1000+48);
veri_yolla((sicaklik%1000)/100+48);
veri_yolla('.');
veri_yolla((sicaklik%100)/10+48);
veri_yolla(sicaklik%10+48);
veri_yolla(0xDF); // Derece isareti olusturuyor
veri_yolla('C');}}
```



Devre 17:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
09.09.2011	PIC18F452 Uygulamaları	

D-)EEPROM UYGULAMALARI

D.1)EEPROM veri yazma ve okuma

Elektriksel silinip,yazılabilen hafıza türüdür.Bilgiler EEPROM üzerinde bilgiler elektrik kesintisi ile silinip gitmezler.PIC18F452 256 bytes EEPROM hafızaya sahiptir.Bu uygulamada PIC hafızasına bilgi yazıp okuyacağız.EEPROM yazıp silmek içinde kütüphane oluşturacağız.Kütüphane Eeprom.h , Eeprom.c iki dosyadan oluşur.

----------KOD(Eeprom.h)----------

```
extern unsigned char Eeprom_oku(unsigned char adres);
extern unsigned char Eeprom_yaz(unsigned char adres,unsigned char data);
```

----------KOD(Eeprom.c)----------

```
#include <pic18.h>

#include "eeprom.h"

#include "Delay.h"

unsigned char Eeprom_oku

(unsigned char adres)

{

EEADR=adres;

EECON1=0x01;

return EEDATA;

} // Devamı Sağ Tarafta
```

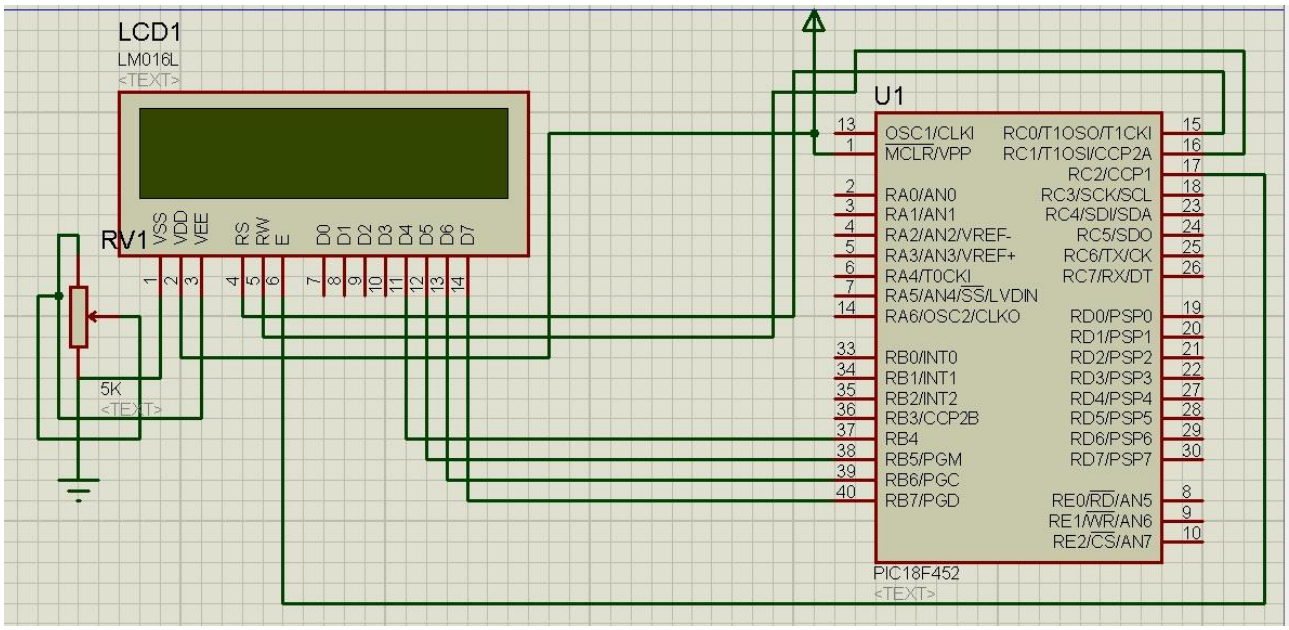
```
unsigned char Eeprom_yaz

(unsigned char adres,unsigned char data)

{EEADR=adres;

EEDATA=data;

EECON1=0x04;EECON2=0x55;

EECON2=0xAA;EECON1=0x06;

while(WR);

WREN=0;

if(WR==1)

WRERR=0;    return WRERR;}
```

----------KOD(main.c)----------

```
#include<htc.h>

#include "Delay.h"

#include "lcd.h"// LCD kütüphanesi tanımlanıyor

#include "eeprom.h"// Eeprom kütüphanesi tanımlanıyor

void main(void) {// Ana fonksiyon alanı

TRISB=0x00; // LCD için çıkıs

TRISC=0x00;

PORTB=0x00;
```

Tarih	Konu	İMZA-Mühür-Kaşe
09.09.2011	PIC18F452 Uygulamaları	

```
PORTC=0x00;
lcd_init();// LCD ilk ayarları yapılıyor
if(Eeprom_yaz(0x00,'M'));  if(Eeprom_yaz(0x01,'A')); // Data eeproma yazılıyor
if(Eeprom_yaz(0x02,'L'));if(Eeprom_yaz(0x03,'I'));
if(Eeprom_yaz(0x04,' '));if(Eeprom_yaz(0x05,'G'));
if(Eeprom_yaz(0x06,'U'));if(Eeprom_yaz(0x07,'R'));
if(Eeprom_yaz(0x08,'B'));if(Eeprom_yaz(0x09,'U'));
if(Eeprom_yaz(0x0A,'Z'));lcd_yaz("Eeprom'daki Veri");
lcd_gotoxy(2,1);veri_yolla(Eeprom_oku(0x00));  // Eepromdaki bilgi LCD'ye yazdırılıyor
veri_yolla(Eeprom_oku(0x01));veri_yolla(Eeprom_oku(0x02));
veri_yolla(Eeprom_oku(0x03));veri_yolla(Eeprom_oku(0x04));
veri_yolla(Eeprom_oku(0x05));veri_yolla(Eeprom_oku(0x06));
veri_yolla(Eeprom_oku(0x07));veri_yolla(Eeprom_oku(0x08));
veri_yolla(Eeprom_oku(0x09));veri_yolla(Eeprom_oku(0x0A));for(;;)CLRWDT();}
```



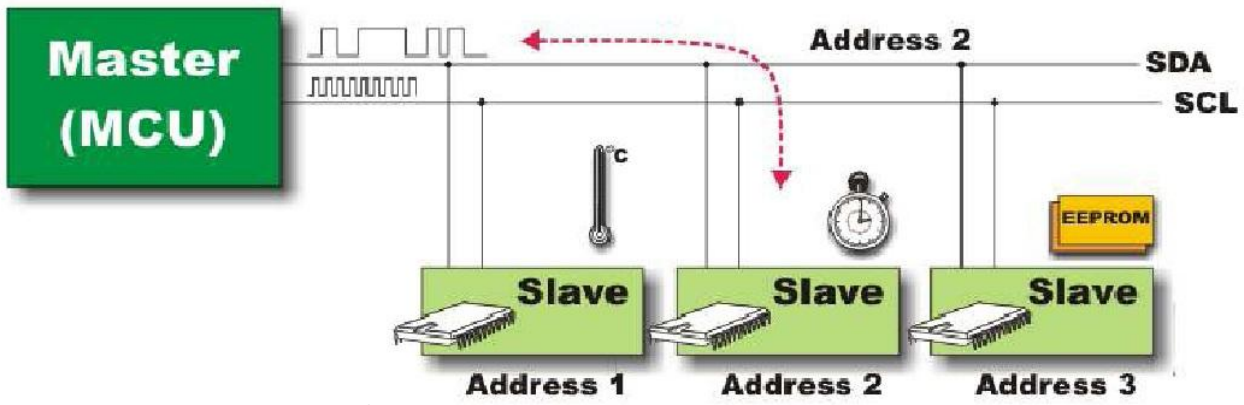
Devre 18:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
09.09.2011	PIC18F452 Uygulamaları	

2.3.5-) I²C (Inter-Integrated Circuit) İLE VERİ İLETİŞİM UYGULAMALARI

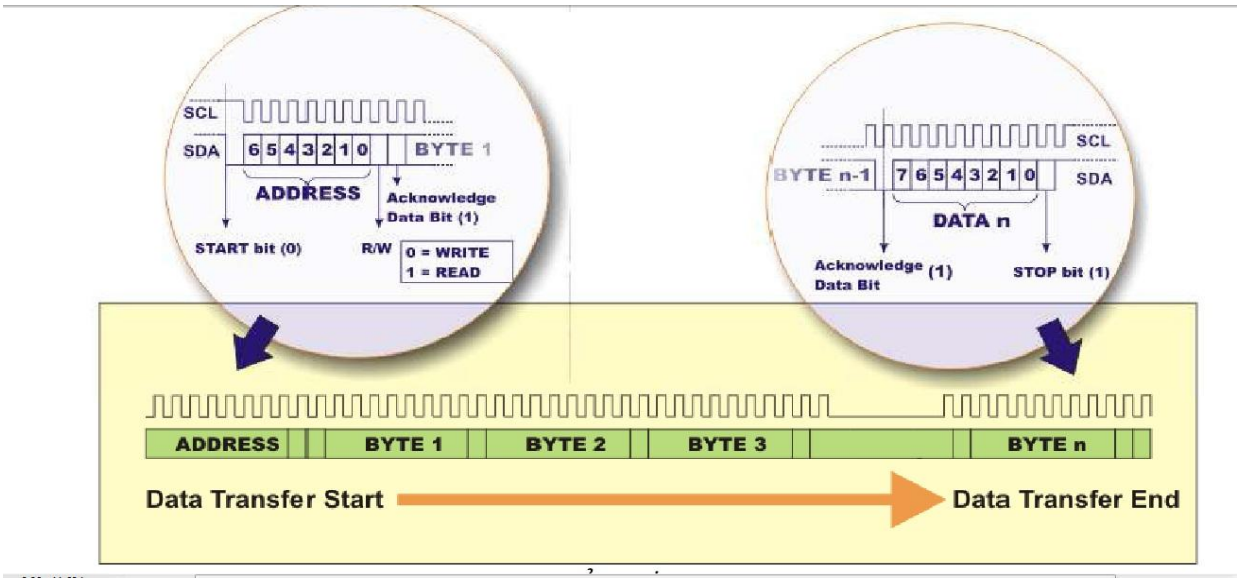
A.1-)I²C TANITIMI

PIC içerisinde bulunan MSSP(Master Synchronous Serial Port) modülünde bulunan bir veri haberleşme birimidir. I²C ile kısa mesafeli,hızlı ve pin sayısının önemli olduğu veri iletişimleri için kullanılmaktadır. I²C seri iletişimi Philips firması tarafından geliştirilen bir protokoldür. I²C protokolü SDA ve SCL pinleri üzerinde gerçekleştirilir. I²C birkaç elektronik bileşenin 2 hat üzerinden veri iletişimini sağlarken bir tane Master bileşen olur ve bu bileşen clock sinyalini kendi üretir ve diğer bileşenler sadece veri giriş ve çıkışları olur. I²C protokolünde Master-Slave ilişkisi aşağıdaki gibidir.



Şekil 2: I²C protokolünde Master-Slave Diyagramı

I²C protokolünde farklı hızlar veri iletişimlerini Master bileşen belirlemektedir.Genel olarak Master bileşen mikro kontroller kullanılmaktadır.Mikrokontrol birimi olarak da biz PIC kullanacağız. I²C protokolünde yavaş (100kbit/s), hızlı (400kbit/s) ve yüksek hızlı (3,4 Mbit/s) olmak üzere çeşitli hızlarda iletişim sağlanabilir. I²C protokolü veri gönderim alımı byte byte yapılmaktadır.Bu protokolün genel yapısı aşağıdaki şekilde gözükmektedir.Fakat bu yapı kullanılan elektronik bileşene göre bazen farklılıklar göstermektedir.



Şekil 3: I²C veri iletişim protokolü

Tarih	Konu	İMZA-Mühür-Kaşe
12.09.2011	I²C TANITIMI	

A.2-)I^{2}C İLE HARİCİ EEPROM UYGULAMASI

Bu uygulamada harici 2 tane EEPROM üzerine veri yazıp veri okuyacağız.EEPROM olarak Atmel 24C512 EEPROM'larını ve mikrokontrol elemanı olarak da 2 tane I^{2}C çıkışı bulunan PIC18F8722 kullanacağız.İlk önce PIC üzerindeki I^{2}C veri iletişimini sağlayan ayarları yapacağız. I^{2}C veri iletişim kontrolünü 3 tane register yapmaktadır.Bu registerlar SSPxSTAT, SSPxCON1, SSPxCON2 dır.Şimdi bu registerların her bir bitinin ne işlev gördüklerine bakalım.

SSPxSTAT:

SMP	CKE	D/A	P$^{(1)}$	S$^{(1)}$	R/W$^{(2,3)}$	UA	BF

bit7 bit0

SMP : 1: I^{2}C modunda düşük hızlarda (100KHz, 1MHz) slew rate kullanım dışıdır,
0: I^{2}C modunda yüksek (400KHz) hızlarda slew rate kullanımı aktiftir

CKE : 1: Transfer aktiften IDLE durumuna geçerken olacak
0: Transfer IDLE durumundan aktife geçerken olacak

D/A : I^{2}C modunda veri ve adres bilgilerini gösteren bittir.
(1: Data gönderilmiş ya da alınmış, 0: Adres gönderilmiş ya da alınmış)

P$^{(1)}$: I^{2}C modunda sonlandırma bitidir. (1: Durdurma biti son bulur, 0: Son bulmaz)

S $^{(1)}$: I^{2}C modunda başlangıç bitidir. (1: Başlangıç biti son bulur, 0: Son bulmaz)

R/W$^{(2,3)}$: SPI ya da I2C modunda okuma mı yoksa yazma mı yaptığını belirten bittir.
(1: Okuma, 0: Yazma)

UA : 10 bitlik I^{2}C modunda adres güncelleme bitidir. UA=1 ise SSPADD kaydedicisine gönderilen adres güncellenir.

BF : Alım modunda (1: Alım tamamlandı (SSPBUF dolu), 0: Alım tamamlanmadı)

SSPxCON1:

WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

bit7 bit0

WCOL : Gönderim yapılırken yazıldı mı (1: Yazıldı, hata var, 0: Yazılmadı)
Başlangıçta WCOL=0 yapılmalı.

SSPOV : Alım yapılırken yeni alım yapıldı mı (1: Yapıldı, hata var, 0: Yapılmadı)
Başlangıçta SSPOV=0 yapılmalı.

SSPEN : 1: SCL, SDA için ayarlandı, 0: Normal pin

CKP : IDLE seviyesi belirleme biti (1: IDLE yüksek seviye, 0: IDLE düşük seviye)

SSPM3,SSPM2,SSPM1,SSPM0: Mod seçim bitleri, aşağıda I$_2$C için gerekli olan tanımlamalar aşağıdaki tabloda mevcuttur.

SSPM3	SSPM2	SSPM1	SSPM0	MOD
1	1	1	1	I^{2}C Slave Mod,10-bit adresleme ile Start ve Stop biti kesmeleri açar
1	1	1	0	I^{2}C Slave Mod,7-bit adresleme ile Start ve Stop biti kesmeleri açar
1	0	1	1	I^{2}C Ürün yazılımında Master Mod
1	0	0	0	I^{2}C Master Mod,clock = F_{OSC}/(4*(SSPxADD + 1))
0	1	1	1	I^{2}C Slave Mod,10-bit adresleme
0	1	1	0	I^{2}C Slave Mod,7-bit adresleme

Tarih	Konu	İMZA-Mühür-Kaşe
12.09.2011	I^{2}C ile PIC18F8722 Uygulaması	

SSPxCON2:

WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

bit7 bit0

GCEN : Slave modda genel çağırma yetkilendirme bitidir. GCEN=1 olduğunda 0000h adresi çağrıldığında kesmeye gidilir.

ACKSTAT : I^2C modunda iletim durumunda statü kabul bitidir.
(1: Alınan veri kabul edilmemiştir, 0: Alınan veri kabul edilmiştir)

ACKDT : I^2C modunda master alım durumunda veri kabul bitidir.
(1: İletilen veri kabul edilmemiştir, 0: Veri kabul edilmiştir)

ACKEN : I^2C modunda eş zamanlı art arda veri alımı yetkilendirme bitidir.
(1: Yetkilendirme açılır ve ACKDT veri biti iletilir, bit donanım tarafından otomatik temizlenir)

RCEN : I^2C modunda alım yetkilendirme bitidir.
(1: I2C modunda alım vaziyetine geçilir, 0: Alım modu devre dışı)

PEN : I^2C modunda stop durum ayarının yapıldığı bittir.
(1: SDA ve SCL pinleri stop durumu alır, 0: Stop durumu oluşturulmaz)

RSEN : I^2C modunda start durumu yenileme bitidir.
(1: SDA ve SCL pinlerinin start durumu tekrarlanır, 0: Yenileme oluşmaz)

SEN : I^2C modunda start durum ayarının yapıldığı bittir.
(1: SDA ve SCL pinleri start durumu alır, 0: Start durumu oluşturulmaz)

Bu registerlara göre PIC Master bileşen olarak ,ardından SDA ve SCL çıkışların bulunduğu pinler I^2C veri iletişimi için ayarlanır ve IDLE seviyesi belirlenir.IDLE belli bir süre bekleme modudur.SSPxCON2 register sonraki kontroller için sıfırlanır.SSPAD veri iletişim hızı seçilir.Slaw rate ayarları yapıldıktan sonra CKE biti ile düşen kenar tetiklemeli yoksa yükselen kenar tetiklemeli olacağı seçilir.Kullanılacaksa kesme bayrakları temizlenir.SSPAD hız ayarları aşağıdaki tabloda mevcuttur.

BAUD RATE

F_{OSC}	**F_{CY}**	**F_{CY}*2**	**BRG Value**	**FSCL (2 Rollovers of BRG)**
40 MHz	10 MHz	20 MHz	18h	400 kHz
40 MHz	10 MHz	20 MHz	1Fh	312.5 kHz
40 MHz	10 MHz	20 MHz	63h	100 kHz
16 MHz	4 MHz	8 MHz	09h	400 kHz
16 MHz	4 MHz	8 MHz	0Ch	308 kHz
16 MHz	4 MHz	8 MHz	27h	100 kHz
4 MHz	1 MHz	2 MHz	02h	333 kHz
4 MHz	1 MHz	2 MHz	09h	100 kHz
4 MHz	1 MHz	2 MHz	00h	1 MHz

Tarih	**Konu**	**İMZA-Mühür-Kaşe**
13.09.2011	I^2C ile PIC18F8722 Uygulaması	

Bu bilgiler doğrultusunda ve Atmel 24C512 EEPROM datasheet'in den yararlanarak ilk önce I2C.h ve I2C.c kütüphanemizi oluşturmalıyız ki her veri iletişiminde (main.c) ana fonksiyonda daha rahat kullanımı olur.

----------KOD(I2C.h)----------

```
/* i2c_init(); ile i2c'nin ilk ayarlarını yap* i2c_wait(); ile IDLE için beklenir, yazım yapılmaz* i2c_start(); ile
SCK ve SDA start durumu alır* i2c_restart(); ile SCK ve SDA start durumu tekrarlanır* i2c_stop(); ile SCK
ve SDA stop durumu alır* veri=i2c_read(ack); ile okuma yapılır* ack=i2c_write(veri); ile yazma yapılır*
SSPADD hız ayarı i2c.c dosyası içerisinde yapılmalıdır.*/
#define TRIS_SCK TRISC3 // SCK ve SDA uçları belirlenir
#define TRIS_SDA TRISC4
extern void i2c_init(void);
extern void i2c_wait(void);
extern void i2c_start(void);
extern void i2c_restart(void);
extern void i2c_stop(void);
extern unsigned char i2c_read(unsigned char ack);
extern unsigned char i2c_write(unsigned char i2c_data);
```

----------KOD(I2C.c)----------

```
#include <pic18.h>
#include "I2C.h"
#include "Delay.h"
void i2c_init(void) {
TRIS_SCK=1; // SCK ve SDA giris olarak ayarlanıyor
TRIS_SDA=1;
SSP1CON1 = 0x38;
SSP1CON2 = 0x00; // SSPCON2 sonraki ayarlamlar için temizlniyor
/* 4Mhz'de
* SSPADD=0x09: 400KHz
* SSPADD=0x0C: 333KHz
* SSPADD=0x27: 100Khz*/
SSP1ADD = 0x09;
CKE=0; // Slew rate kullanılacak
SMP=1; // Kenar: Aktiften IDLE'a
SSP1IF=0; // Kesme bayrakları temizleniyor
BCL1IF=0;}
void i2c_wait(void)
{
while (( SSP1CON2 & 0x1F ) | RW );// IDLE için beklenir, yazım yapılmaz
}
void i2c_start(void)
{
i2c_wait(); // IDLE beklenir
SEN=1; // SCK ve SDA start durumu alır
}
void i2c_restart(void){
i2c_wait(); // IDLE beklenir
RSEN=1; // SCK ve SDA start durumu tekrarlanır}
```

```
void i2c_stop(void)
{
i2c_wait(); // IDLE beklenir
PEN=1; // SCK ve SDA stop durumu alır
}
unsigned char i2c_read(unsigned char ack)
{
unsigned char i2c_data;
i2c_wait(); // IDLE beklenir
RCEN=1; // Alım vaziyetine geçilir
i2c_wait(); // IDLE beklenir
i2c_data=SSP1BUF; // Veri alınır
i2c_wait(); // IDLE beklenir
if(ack) // ACK biti durumu
{
ACKDT=1; // Alım kabul edilmedi
}
else
{
ACKDT=0; // Alım kabul edildi
}
ACKEN=1; // ACKDT veri biti iletilir
return(i2c_data);
}
unsigned char i2c_write(unsigned char i2c_data)
{
i2c_wait(); // IDLE beklenir
SSP1BUF=i2c_data; // Veri gönderiliyor
return(ACKSTAT); // 1: Alınan veri kabul edilmemistir
// 0: Alınan veri kabul edilmistir  }
```

----------KOD(main.c)----------

```
#include <htc.h>
#include "Delay.h" // Gecikme kütüphanesi tanımlanıyor
#include "lcd.h" // LCD kütüphanesi tanımlanıyor
#include "I2C.h"
                                        // I2C kütüphanesi tanımlanıyor
void write_ext_eeprom(int sec,unsigned char address1,unsigned char address2, unsigned char data)
{
i2c_start();
if(sec==1)
i2c_write(0xA6);
else
i2c_write(0xA2);
i2c_write(address1);
i2c_write(address2);
i2c_write(data);
i2c_stop();
DelayMs(10);   }
```

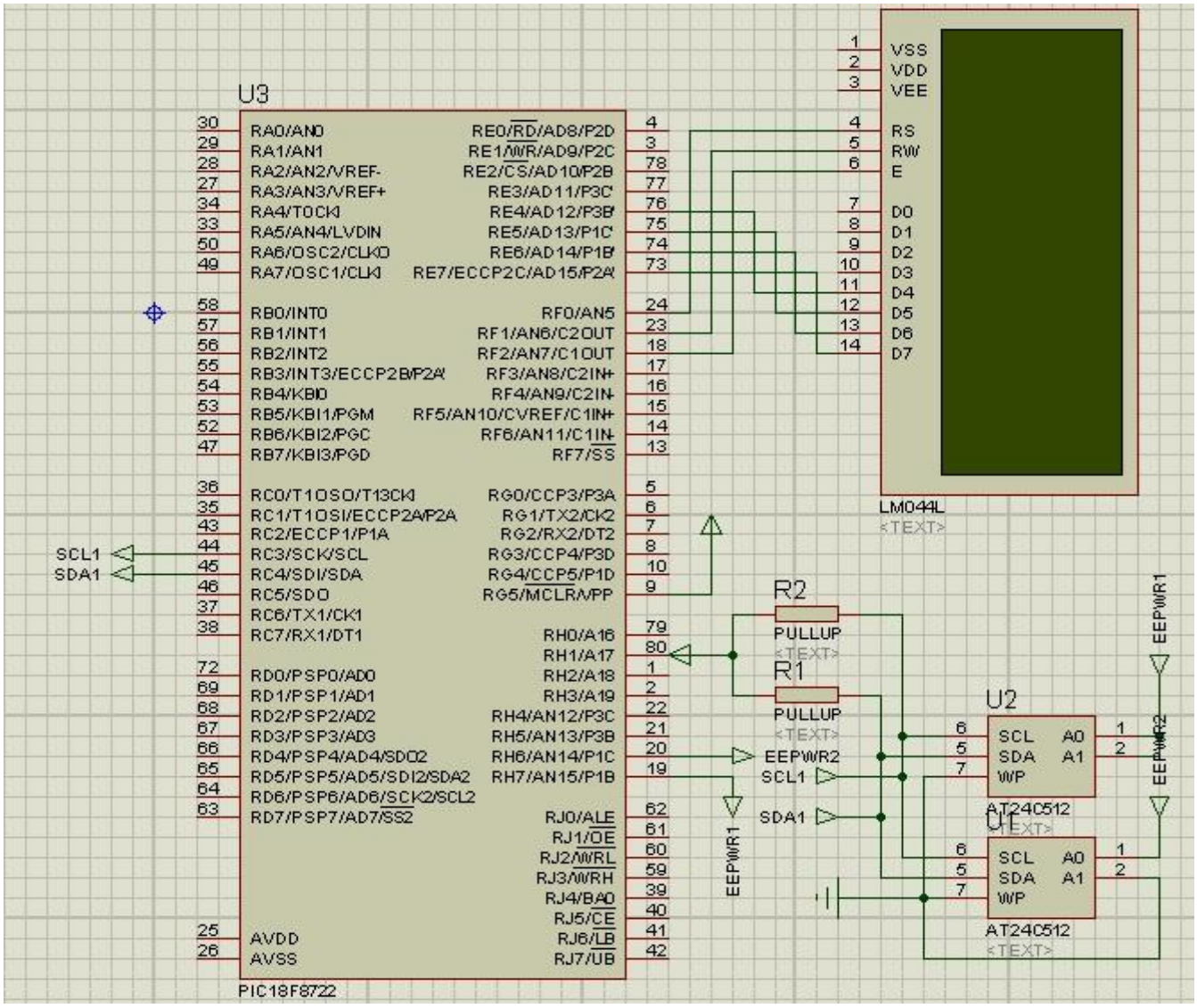
Tarih	Konu	İMZA-Mühür-Kaşe
14.09.2011	I^2C ile PIC18F8722 Uygulaması	

```
unsigned char read_ext_eeprom(int sec,unsigned char address1,unsigned char address2)
{
unsigned char data;
i2c_start();
if(sec==1)
i2c_write(0xA6);//Eeprom1 Device adress
else
i2c_write(0xA2); //Eeprom2 Device adress
i2c_write(address1);
i2c_write(address2);
i2c_restart();
if(sec==1)
i2c_write(0xA7);
else
i2c_write(0xA3);
data=i2c_read(0);
i2c_stop();
return(data); }
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config LVP=OFF
void main(void) {
unsigned char i=0;
TRISE=0x00; // Çıkıslar ayarlanıyor
TRISF=0x00;TRISC=0x00;TRISH=0x00;
PORTE=0x00; // Portlar sıfırlanıyor
PORTF=0x00;PORTC=0x00;
PORTH=0x80;ADCON1=0X0F;
lcd_init(); // LCD ve I2C ilk ayarları yapılıyor
i2c_init();
write_ext_eeprom(1,0x00,0x00,'M'); write_ext_eeprom(1,0x00,0x08,'A');
write_ext_eeprom(1,0x00,0x10,'L');write_ext_eeprom(1,0x00,0x18,'I');
// Eepromdaki bilgi LCD'ye yazdırılıyor
lcd_yaz("EP1:");lcd_gotoxy(1,5);
veri_yolla(read_ext_eeprom(1,0x00,0x00));
veri_yolla(read_ext_eeprom(1,0x00,0x08));
veri_yolla(read_ext_eeprom(1,0x00,0x10));
veri_yolla(read_ext_eeprom(1,0x00,0x18));
lcd_gotoxy(2,1);
PORTH=0x40;
write_ext_eeprom(2,0x00,0x00,'G'); write_ext_eeprom(2,0x00,0x08,'U');
write_ext_eeprom(2,0x00,0x10,'R');write_ext_eeprom(2,0x00,0x18,'B');
write_ext_eeprom(2,0x00,0x20,'U');write_ext_eeprom(2,0x00,0x28,'Z');
// Eepromdaki bilgi LCD'ye yazdırılıyor
lcd_yaz("EP2:");lcd_gotoxy(2,5);
veri_yolla(read_ext_eeprom(2,0x00,0x00));veri_yolla(read_ext_eeprom(2,0x00,0x08));
veri_yolla(read_ext_eeprom(2,0x00,0x10));veri_yolla(read_ext_eeprom(2,0x00,0x18));
veri_yolla(read_ext_eeprom(2,0x00,0x20));veri_yolla(read_ext_eeprom(2,0x00,0x28));
for(;;);
}
```

Bu uygulamada lch.h ve lcd.c kütüphaneleri 2x16 öncelik verdiğimiz lcd.c üzerinde sadece şu kodların değiştirilmesi ile 4x20 lcd kütüphanesine dönüştürmüş oluruz.

----------KOD(lcd.c(2x16)değişecek kısım)----------

```
void lcd_gotoxy(unsigned char x,unsigned char y) //LCD'nin belli //bölgesine gidiliyor
{
if(x==1)
lcd_komut(0x80+((y-1)%20));
else if(x==2)
lcd_komut(0xC0+((y-1)%20));
else if(x==3)
lcd_komut(0x94+((y-1)%20));
else
lcd_komut(0xD4+((y-1)%20));
}
```

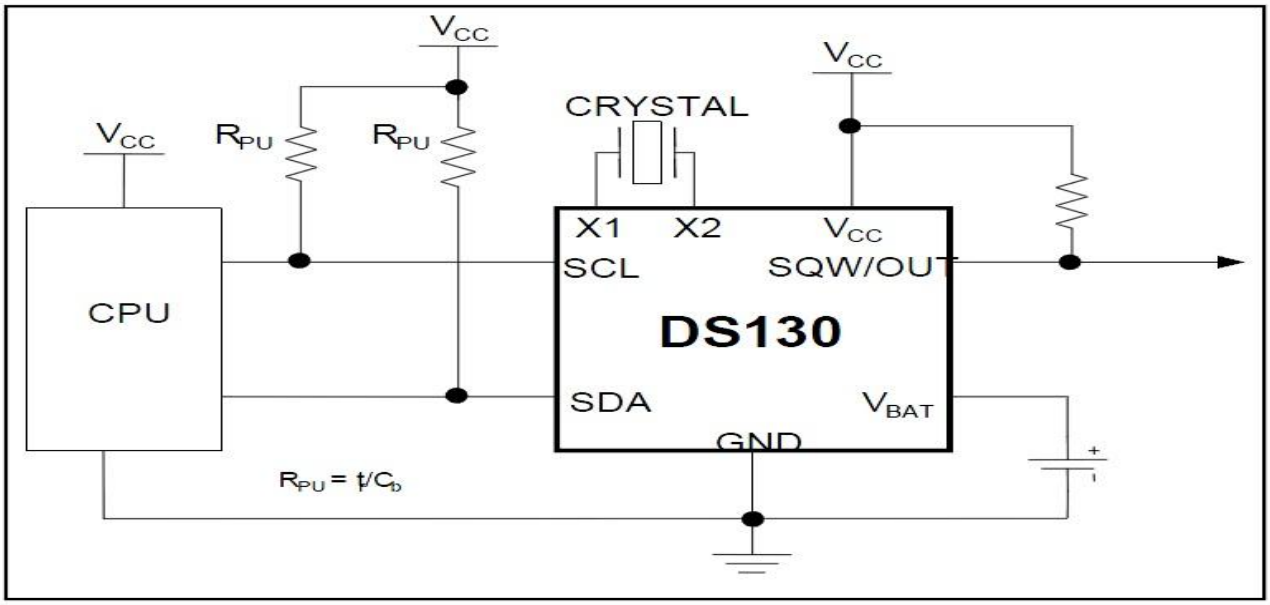


Devre 19:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
15.09.2011	I^2C ile PIC18F8722 Uygulaması	

A.3-)I^{2}C İLE RTC(Real Time Clock) UYGULAMASI

RTC entegreleri ile birkaç farklı yolla haberleşmek mümkündür.Biz bu uygulamada Bir önceki harici EEPROM uygulamasında kullandığımız I^{2}C protokolünü kullanacağız.I2C.c ve I2C.h kütüphanelerimizi aynen kullanacağız fakat (main.c) kısmındaki oluşturduğumuz fonksiyonlar üzerinde birkaç değişiklik yapılarak RTC entegresi ile haberleşmiş olacağız.I2C kütüphanemiz deki değişiklik DS1307 RTC entegresinden dolayıdır.Bu yüzden bu entegreyi biraz tanımamız gerekmektedir.DS1307 ile ilgili bazı bilgiler aşağıda mevcuttur.



Şekil 4:DS1307 entegrenin devreye bağlanma şekli

Aşağıdaki Tablo DS1307'nin saat ve tarih bilgisini tutuğu RAM ait adres bilgilerini içermektedir.Bu DS1307 en önemli özelliklerinden biridir.Devrede elektrik olmasa bile V_{BAT} sayesinde saat bilgilerini güncel tutabilmektedir.DS1307 ile haberleşme EEPROM daki ile aynı olu sadece Device Address bilgisini değiştirerek sağlanmaktadır.(DS1307 Device Address:0xD0)

RTC_RAM

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

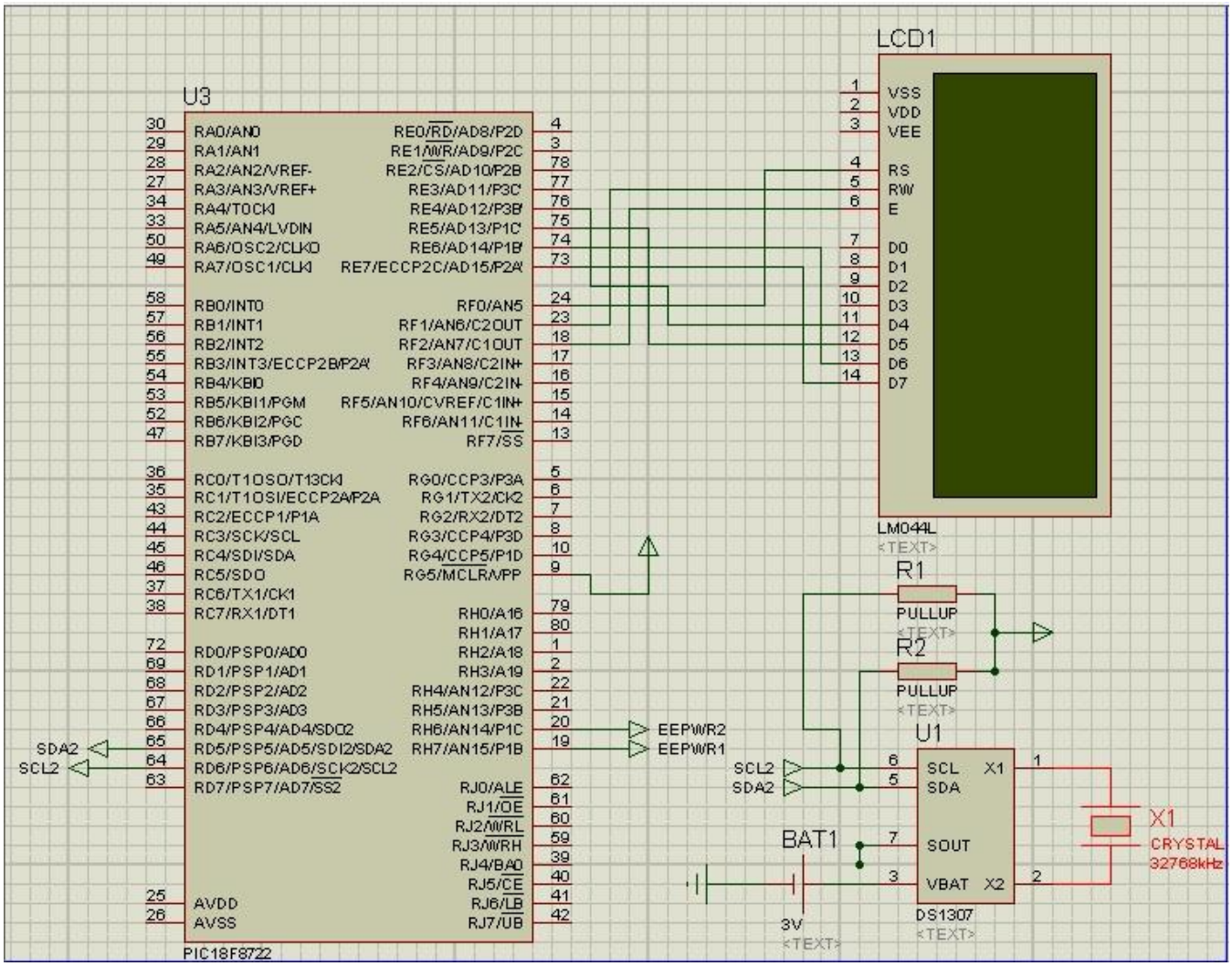
Tarih	Konu	İMZA-Mühür-Kaşe
15.09.2011	I^{2}C ile PIC18F8722 Uygulaması	

Bu bilgileri ve önceki uygulamadaki lcd.h , lcd.c ve I2C.h , I2C.c kullanarak RTC entegresinden saat ve tarih bilgisini okumak ve yazma yapılarak düzeltmek mümkündür.EEPROM uygulamasından farklı olarak bu uygulamada i2c_read fonksiyonuna 1 gönderilerek ACKDT bitini set ederek(ACKDT=1) İlk alınan byte sonra alım modunu kapatmamızdır.Bu şekilde kullanmak tercihendir, istenildiği vakit i2c_read fonksiyonundan gelen datayı bir karakter dizisine yönlendirerek yapılabilir.Sonsuz döngünün her başlangıcında karakter dizisi sıfırlanarak da yapılabilir.

----------KOD(main.c)----------

```
#include <htc.h>
#include "Delay.h" // Gecikme kütüphanesi tanımlanıyor
#include "lcd.h" // LCD kütüphanesi tanımlanıyor
#include "I2C.h"
void write_ext_eeprom2(unsigned char address, unsigned char data)
{
i2c_start();
i2c_write(0xD0);
i2c_write(address);
i2c_write(data);
i2c_stop();
DelayMs(100);
}
unsigned char read_ext_eeprom2(unsigned char address)
{
unsigned char data;
i2c_start();
i2c_write(0xD0);
i2c_write(address);
i2c_restart();
i2c_write(0xD1);
data=i2c_read(1);
i2c_stop();
return(data);
}
#pragma config OSC=HS
#pragma config WDT=OFF
#pragma config LVP=OFF
void main(void)
{
TRISE=0x00; // Çıkıslar ayarlanıyor
TRISF=0x00;
TRISC=0x00;
TRISJ=0x00;
PORTE=0x00; // Portlar sıfırlanıyor
PORTF=0x00;
PORTC=0x00;
PORTJ=0xFF;
ADCON1=0x0F;
lcd_init(); // LCD ve I2C ilk ayarları yapılıyor
i2c_init();
```

```
unsigned char sn,dk,sa;
unsigned char ay,gun,yil;
for(;;){
sn=read_ext_eeprom2(0x00);dk=read_ext_eeprom2(0x01);
sa=read_ext_eeprom2(0x02);gun=read_ext_eeprom2(0x04);
ay=read_ext_eeprom2(0x05);yil=read_ext_eeprom2(0x06);
lcd_gotoxy(4,2);
veri_yolla(((sa&0xF0)>>4)+48);veri_yolla((sa&0x0F)+48);
lcd_yaz(":");
veri_yolla(((dk&0xF0)>>4)+48);veri_yolla((dk&0x0F)+48);
lcd_yaz(":");
veri_yolla(((sn&0xF0)>>4)+48);veri_yolla((sn&0x0F)+48);
lcd_yaz("/");lcd_gotoxy(4,11);
veri_yolla(((gun&0xF0)>>4)+48);veri_yolla((gun&0x0F)+48);
lcd_yaz(".");
veri_yolla(((ay&0xF0)>>4)+48);veri_yolla((ay&0x0F)+48);
lcd_yaz(".");
veri_yolla(((yil&0xF0)>>4)+48);veri_yolla((yil&0x0F)+48);   } }
```



Devre 20:Proteus üzerindeki devre çizimi

Tarih	Konu	İMZA-Mühür-Kaşe
16.09.2011	I^2C ile PIC18F8722 Uygulaması	

2.3.6-)EUSART(Enhanced Universal Synchronous Receiver Transmitter) İLE HABERLEŞME UYGULAMALARI

A.1-)EUSART TANITIMI

EUSART ile haberleşme I^2C'nin aksine uzun mesafeli veri iletişimi için kullanılmaktadır.EUSART ile haberleşme iki farklı şekilde yapılmaktadır.Bu iki farklı iletişim senkron ve an-senkron olarak adlandırılırlar.Senkron haberleşmede veri bir clock eşliğinde yapılan haberleşmedir.An-senkron haberleşmede ise clock eşliğinde değil başlama ve bitiş bitleri kullanılır.Biz uygulamamızda an-senkron olanı kullanacağımız için onun üzerinde duracağız.An-senkron haberleşme yapılırken bilinmesi gereken birkaç yapı vardır.Bu yapılar veri iletişim hızı,başlangıç biti,veri uzunluğu,eşlik biti,bitiş bitidir.Veri iletişim hızı alıcı ve vericinin hangi hızlarda iletişim yapacağının belirlenmesidir.Başlangıç biti veri gönderilmesinin başlayacağını bildirir.Veri uzunluğu genel olarak 8-bit olarak yapılmaktadır.Eşlik biti ile veri bozukluklarının kontrolü yapılır.Bitiş biti veri iletişiminin bittiğini bildirir.Biz haberleşmemizde RS232 kullanacağız.RS232'de alıcı ve verici olmak üzere iki hat üzerinden gerçekleştireceğiz.RS232 ile haberleşmede lojik-0 için -3V ile -12V arasında,lojik-1 için +3V ile +12V arasında dönüşüm yapan Max232 entegresini kullanacağız.Bu entegreyi kullanma sebebimiz alıcı ve vericinin faklı voltaj değerlerinde çalışmasıdır.

A.2-)RS232 İLE HABERLEŞME

Bu uygulamamızda PIC18F8722,RS232 Port ve Max232 entegresi kullanacağız.PIC'in EUSART için iki tane özel pini bulunmaktadır.Bu pinler **RC6/TX1**(verici) ve **RC7/RX1**(alıcı) dir.Bu pinler PIC içerisinde TXSTAx ve RCSTAx registerları ile kontrol edilmektedir.Ayrıca veri iletişim hızını da BAUDCONx register ile kontrol edilmektedir.Bu registerlarla ilgili ayrıntılar aşağıdadır.

TXSTAx:

CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D

bit 7 bit 0

CSRC: An-Senkron moda dikkate alınmaz
TX9: İletişim modu seçme biti (1: 9 Bitlik İletişim, 0: 8 Bitlik İletişim)
TXEN: Transfere izin verme biti (1: Transfere izin ver, 0: Transfere izin verme)
SYNC: 1: Senkron mod, 0: Asenkron mod
SENDB: 1:Parca bir sonraki iletim üzerinde 0:iletim tamamlandı
BRGH: Hızlı/yavaş boudrate seçme biti (1: Hızlı mod, 0: Yavaş mod)
TRMT: İletim kaydedicisi boş/dolu biti (1: Dolu, 0: Boş)
TX9D: 9 bitlik modda parity veya 9. Bit

RCSTAx:

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

bit 7 bit 0

SPEN : 1: Seri port açık, 0: Seri port kapalı
RX9 : Alım modu seçme biti (1: 9 bitlik iletişim, 0: 8 bitlik iletişim)
SREN : Senkron modda (1: Tek alım açık, 0: Tek alım kapalı)
CREN : Asenkron modda (1: Devamlı almaya izin ver, 0: Devamlı almaya izin verme)
Senkron modda (1: CREN sıfırlanıncaya kadar alıma izin ver, 0: Almayı kes)
ADDEN: Asenkron modda RX9=1 ise (1: 8bit yüklemede kesmeye git, 9. bit parity olarak kullanılabilir)
FERR : Alımda hata algılaması biti (1: Hata var, 0: Hata yok)
OERR : Üst üste yazım hatası kontrolü (1: Hata var, 0: Hata yok)
RX9D : 9 bitlik modda parity veya 9. bit

Tarih	Konu	İMZA-Mühür-Kaşe
19.09.2011 20.09.2011	EUSART ile PIC18F8722 Uygulaması	

Bu registerlarla gerekli ayarlamalar yapıldıktan sonra Baud Rate Generator için gerekli ayarları yukarıdaki bitlerin durumuna göre belirlenmektedir.Baud Rate ile ilgi ayarlar aşağıdaki tabloda bulunmaktadır.

Baud Rate Formül Tablosu

SYNC	BRG16	BRGH	BRG/EUSART Mode	Baud Rate Formula
0	0	0	8-bit/Async	$F_{osc}[64(n+1)]$
0	0	1	8-bit/Async	$F_{osc}[16(n+1)]$
0	1	0	16-bit/Async	$F_{osc}[16(n+1)]$
0	1	1	16-bit/Async	$F_{osc}[4(n+1)]$
1	0	x	8-bit/sync	$F_{osc}[4(n+1)]$
1	1	x	16-bit/sync	$F_{osc}[4(n+1)]$

Bu tablodaki formüllerden bize gerekli olanı seçtik den sonra SPBRGx register atanması gereken sayısı aşağıdaki formüllü hesaplanır ve register atama yapılır.

SPBRGx = ((F_{OSC}/ Baud Rate)/16) – 1
= ((10000000/19200)/16) – 1
= [31.552] = 31

Bu bilgileri kullanarak şimdi usart.c ve usart.h oluşturarak kütüphanemizi oluşturalım.Kütüphane ile ilgili kodlar aşağıdadır.

----------KOD(usart.h)----------

```
#define BAUD 19200 //Boudrate
#define FOSC 1000000L //Kristal hızı
#define NINE 0 //9 bit iletisim kullanılacaksa 1 yapılmalı
#define HIGH_SPEED 1 //Hızlı iletisim yapılacaksa 1 yapılmalı
#define RX_PIN TRISC7 //Seri port pinleri
#define TX_PIN TRISC6
#define DIVIDER ((int)(FOSC/(16UL * BAUD) -1))
extern void usart_init(void);
extern void putch(unsigned char byte);
extern unsigned char getch(void);
```

----------KOD(usart.c)----------

```
#include <pic.h>
#include "stdio.h"
#include "usart.h"
void usart_init(void){
unsigned char speed,nine_bits;
unsigned int divider;
RX_PIN = 0; TX_PIN = 1; // Seri iletisim pinleri giris olarak tanımlanıyor
if(HIGH_SPEED==1) // Hızlı iletisim yapılacaksa
speed=0x04;
else // Yaval iletisim yapılacaksa
speed=0x00;
if(NINE==1) // 9 bitlik iletisim yapılacaksa
nine_bits=0x04;
else // 8 bitlik iletisim yapılacaksa
nine_bits=0x00;
SPBRG = DIVIDER; // Hız değeri SPBRG'ye yükleniyor
RCSTA = (nine_bits|0x90); // Alım kaydedicisi ayarlanıyor
TXSTA = (speed|nine_bits|0x20); // Gönderim kaydedicisi ayarlanıyor}
```

```
void putch(unsigned char byte){
// Tek byte gönder
while(!TXIF); // Transfer tamponu bos mu
TXREG = byte; // Tek byte gönderiliyor}
unsigned char getch(void){
// Tek byte al
while(!RCIF); // Alım tamponu bos mu
return RCREG; // Tek byte alınıyor}
```

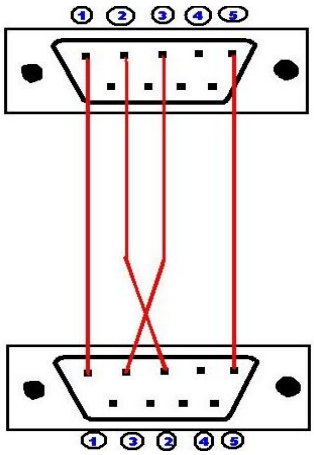
Kütüphanemizi oluşturduk dan sonra önceki uygulamalarımızda kullandığımız lcd kütüphanemizi de bu uygulamada kullanarak Virtual Termial den girilen karakteri LCD ekranına yazdıracağız.Kütüphaneleri kullanacağımız main.c aşağıdadır.Karakter

----------KOD(main.c)----------

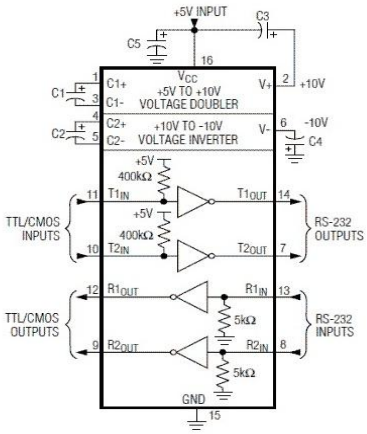
```
#include <htc.h>
#include <stdio.h> // printf için gerekli C standart giris çıkıs kütüphanesi
#include "delay.h" // Gecikme kütüphanesi tanımlanıyor
#include "lcd.h" // LCD kütüphanesi tanımlanıyor
#include "usart.h" // USART kütüphanesi tanımlanıyor
void main(void){ // Ana fonksiyon alanı
char i,j,x;
TRISB=0x00; // LCD için çıkıs
TRISC=0xF0; // USART pinleri giris olarak kabul ediliyor
PORTB=0x00;
lcd_init(); // LCD ilk ayarları yapılıyor
usart_init(); // USART ilk ayarları yapılıyor
lcd_yaz("*KAYSERI ULASIM AS.*");lcd_gotoxy(2,1);
lcd_yaz(" LCD istenilen");lcd_gotoxy(3,1);
lcd_yaz(" LCD karakteri yazar");lcd_gotoxy(4,1);
lcd_yaz("Yukleniyor");lcd_gotoxy(4,11);
for (x=0;x<7;x++){
lcd_yaz(".");DelayMs(150);}
lcd_clear();lcd_komut(ImlecYanSon); // Imlec yanıp sönecek
printf("KAYSERI ULASIM AS."); // Veri gönderiliyor.
for(;;){
if(j==0){ printf("\n1.Satir:");lcd_gotoxy(1,1);}
if(j==20){ printf("\n1.Satir:");lcd_gotoxy(2,1);} // 2.satıra geçiliyor
if(j==40){ printf("\n1.Satir:");lcd_gotoxy(3,1);} // 3.satıra geçiliyor
if(j==60){ printf("\n1.Satir:");lcd_gotoxy(4,1);} // 4.satıra geçiliyor
i=getch();// Karakter alınıyor.
if (i==8){
lcd_komut(ImlecGeri);veri_yolla(0);lcd_komut(ImlecGeri);//Silme işlemi
j--;}
else{
veri_yolla(i); // LCD yazılıyor.
j++;
if (j==80)
j=0;
}}}
```

Uygulamada kullanılacak olan Max232 entegresini nasıl çalıştığına dair geniş bilgiler datasheet bulunmaktadır.Bu bilgilere göre Max232 entegrsinin ,$R1_{in}$,$R1_{out}$,$R2_{in}$,$R2_{out}$,$T1_{in}$,$T1_{out}$,$T2_{in}$,$T2_{out}$ olmak üzere 4 giriş 4 çıkışı bulunmaktadır. $RX_{in/out}$: Alıcı giriş/çıkışları , $TX_{in/out}$: Verici giriş/çıkışları dır.Burada Max232 entegresi RS232 konektörüne normal bağlandığı için bilgisayarla PIC devresi arasındaki Serial Port kablosu çaprazlanmış olması gerekmektedir.Aşağıdaki resimde gözükmektedir.

Tarih	Konu	İMZA-Mühür-Kaşe
22.09.2011	EUSART ile PIC18F8722 Uygulaması	



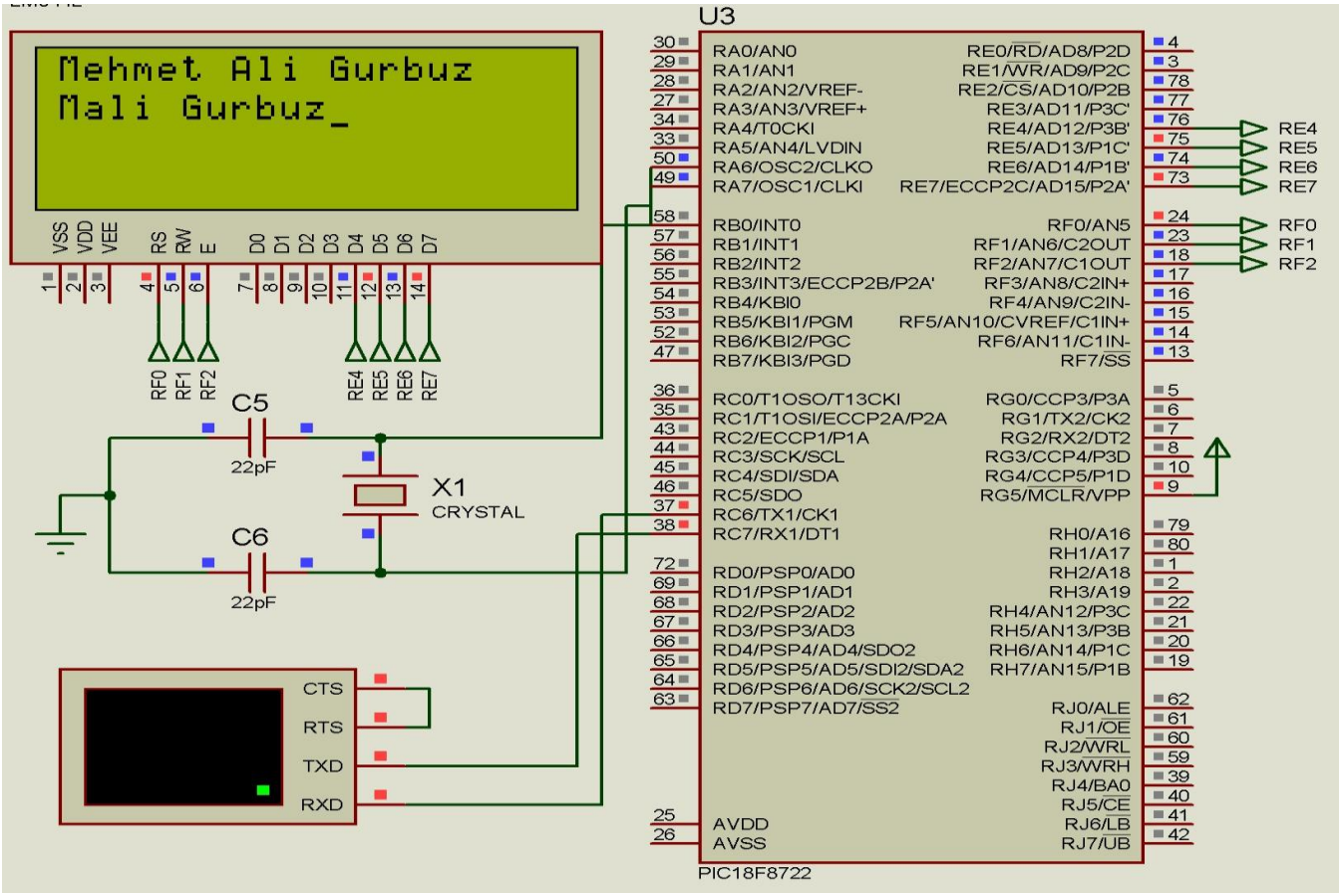
Şekil 5:RS232 Bağlantısı



Şekil6:Max232 Entegresi

Şekil7:Virtual Terminal Görütüsü

Devre Proteus çiziminde Max232 bulunmamaktadır.Çünkü Simülasyon üzerinde Virtual Terminal çıkış olarak 0V-5V aralığında çalışmaktadır.Kullanılan devrede Max232 mevcuttur ve uygulama devre üzerinde çalıştığı test edilip görülmüştür.



Devre 21:Proteus üzerindeki devre çizimi

KAYNAKLAR

1-)http://320volt.com/wp-content/uploads/2009/09/hi-tech-pic-programlama-dersleri-mplab-hi-tide.pdf
2-) http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf
3-) http://ww1.microchip.com/downloads/en/DeviceDoc/39646c.pdf

Tarih	Konu	İMZA-Mühür-Kaşe
23.09.2011	EUSART ile PIC18F8722 Uygulaması	