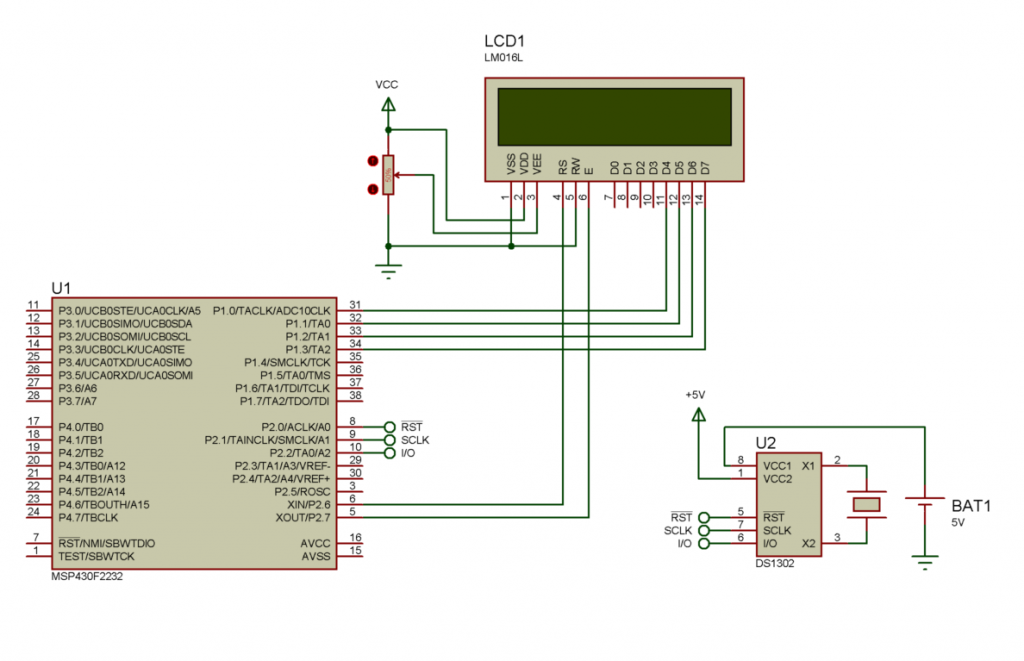
**MSP430 Uyg.32 – DS1302 RTC Uygulaması**

Ferudun GÖKCEGÖZ, 16 Ağustos 2011, Salı



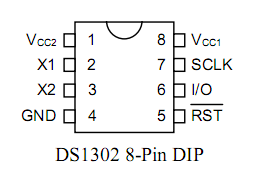
Merhaba arkadaşlar. 32.dersimizle yine sizlerle birlikteyiz. Umarım derslerimiz faydalı oluyordur. Bu aralar zamanım biraz bol olduğu için sık sık uygulamalar paylaşmaya çalışıyorum. Unutmayınki, bilgi paylaştıkça çoğalır. Hadi bakalım geçelim uygulamamıza. Bu dersimizdeki uygulamamızda MSP430 ile DS1302 RTC(gerçek zaman saati – real time clock) entegresiyle bir uygulama yapacağız. DS1302 entegresinden saniye, dakika,saat, gün,ay, yıl ve haftanın günü bilgisini alıp lcd ekranımızda yazdıracağız. Devre şemasını vererek başlayalım isterseniz…

**İşte devre şemamız…**



Devre şemasından da görüldüğü gibi MSP430 a bir adet lcd ve DS1302 entegresi bağlanmış durumda. DS1302 nin X1 ve X2 uçlarınada 32.768 kHz lik bir kristal bağlanmış durumda.

İlk olarak DS1302 den bahsedecek olursak, bu entegre Dallas Semiconductor (maxim) firmasının ürettiği bir rtc(real time clock) entegresidir. İçerisinde 2100 yılına kadar tüm gün ay yil verileri kayıtlıdır. Entegre enerjilendiğinde ve kristal stabil bir şekilde çalışırken saniye saniye içerisindeki veride güncellenip gerçek zaman sayılmaktadır. Bu entegre NV(non volatile) türden olmak üzere 31 byte lık bir ram e sahiptir. Bu kısım kullanıcı tarafından istenildiği gibi kullanılabilir. 2V tan 5.5V a kadar bir gerilim aralığına sahiptir. 2V ta çalışırken 300nA den daha az akım tüketmektedir. İçerisindeki verinin burst mod da okunma özelliği vardır. İllaki tek tek okumak mecburiyeti yoktur. 3 wire haberleşme yöntemini kullanmaktadır. TTL uyumlu olup, -40 tan +85 derecelere kadar çalışabilme özelliğine sahiptir. Entegremizin 8 pinlik DIP kılıf modelini aşağıda görebiliriz.



Entegremizin pin lerinden de bahsedelim.

**X1 ve X2 = 32.768 kHz lik kristalin bağlanacağı pinlerdir.**

**Vcc1 ve Vcc2 = Beslemelerin bağlanacağı pinlerdir.**

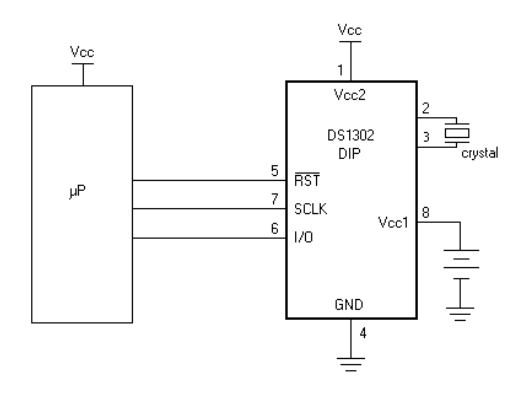
**Gnd = Ground**

**SCLK = Seri haberleşmede gereken clock sinyalinin uygulanacağı pin.**

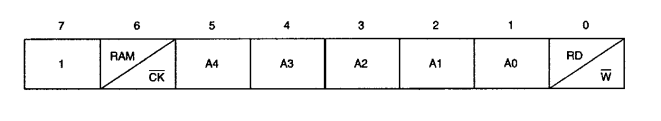
**I/O = Data alış verişinin olduğu pin.**

**$RST = Entegreyi slave olarak seçmek için kullanılan pin.**

Entegremize ait standart bağlantı şeması aşağıda verilmiştir. Bu şekilde bir bağlantı ile herhangi bir mikroişlemci ile haberleşebilir.

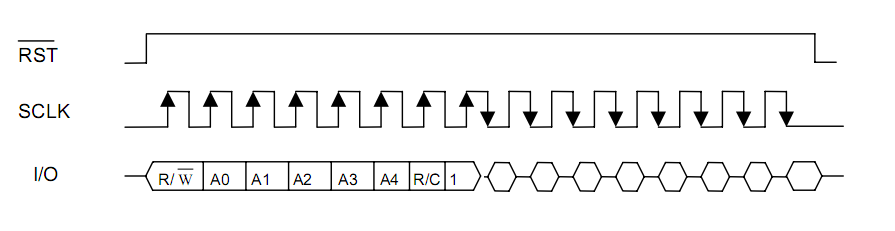


DS1302 ye gönderilen komutlar bir formatta gönderilmelidir. Bunun için öncelikle adres/komut byte ı gönderilmelidir. Aşağıda bu adres/komut byte ının bit bit ayrıntısını görmekteyiz.

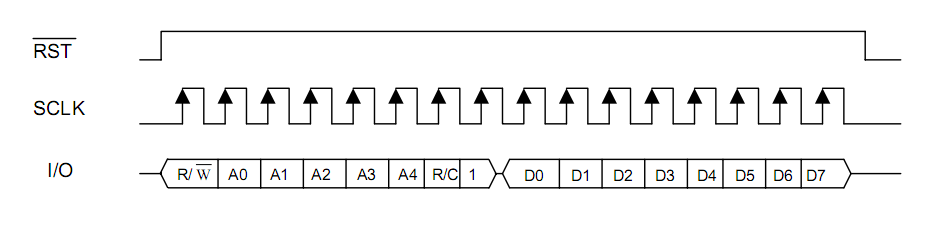


Gönderilen bu byte ın 7.biti her zaman lojik 1 olmalıdır. 6.biti ise okuma/yazma yapılan register ların (RAM) mi yoksa (Saat&Takvim) lere ait register lar mı olduğunu belirler. Bundan sonraki 5 bit adres bitleridir. 0.bit ise okuma veya yazma yapılacağını belirler. Okuma yapılacaksa bu bit Lojik1, yazma yapılacaksa Lojik0 olmalıdır. Gelelim entegremizden bir byte data okuma ve bir byte yazma işlemlerinin yapılacağına…

***Bir Byte Okuma İşlemi***

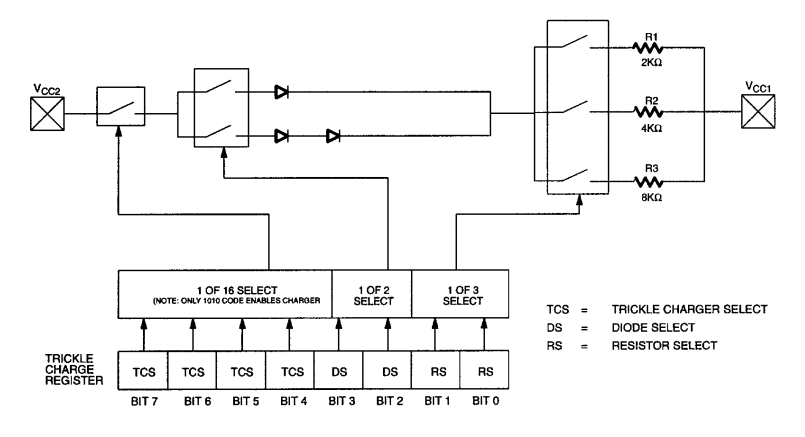
Timing diyagramından da görüldüğü gibi data okuma işlemine başlanmadan önce RST pini high a çekilir. Daha sonra adres/komut byte ı her yükselen kenarda bir bit olmak üzere gönderilir. Bu byte gönderildikten sonra her düşen kenarda ds1302 nin gönderdiği byte ın bitleri okunur.

***Bir Byte Yazma İşlemi***

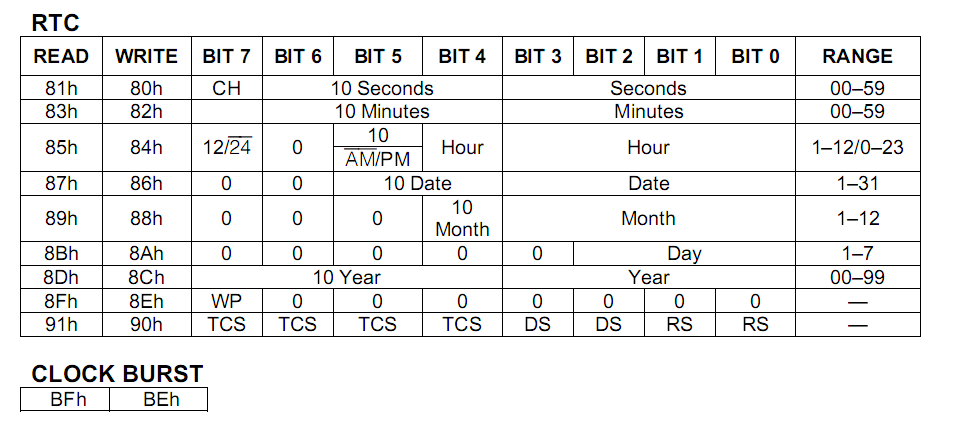


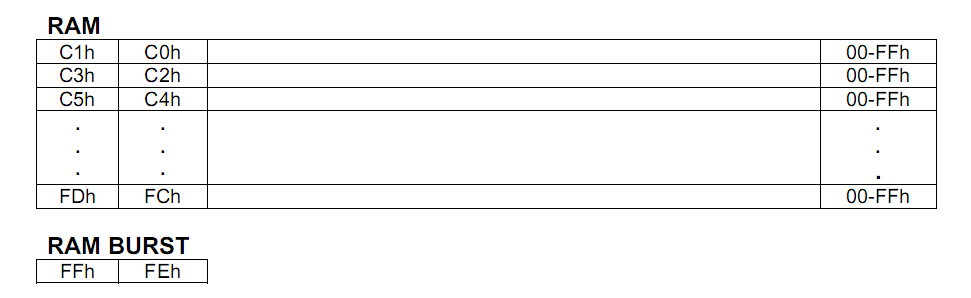
Yine timing diyagramından da görüldüğü gibi data yazma işlemine başlanmadan önce RST pini high a çekilir. Daha sonra adres/komut byte ı her yükselen kenarda bir bit olmak üzere gönderilir. Bu byte gönderildikten sonra yine her yükselen kenarda ds1302 ye gönderilecek byte ın bitleri gönderilir.

RTC entegrelerinde olduğu gibi entegre enerjisiz kaldığında gerçek zamanın doğru bir şekilde saymaya devam etmesi için bu entegrelerde bir şarj sistemi vardır. Bu nedenle iki adet Vcc pini vardır. Bunlardan birine normal besleme gerilimi, diğerine ise yüksek kapasiteli bir kondansatör veya az akım harcandığından değiştirilebilmeye uygun lityum piller bağlanabilir. Bu şarj sistemi Vcc2 den Vcc1 doğrudur. Ve Trickle Charger register ın dan ayarlanmaktadır. Entegreyi yazılımda ilk ayarlarını başlatırken, bu kaydediciyede uygun bir değer yüklenir. Aşağıdaki bu şarj sistemine ait diyagramı görmekteyiz.



Entegrenin içerisindeki dataları okurken (saniye,dakika, ram deki data gibi) veya yazarken aynı adresten okuyamayız. Bu nedenle her verinin bir okuma adresi ve bir de yazma adresi bulunmaktadır. Bunu aşağıdaki tablo ile daha net görebiliriz.





DS1302 ile alakalı bu kadar bilgi verdikten sonra gelelim yazılımımıza. Daha detaylı bilgi öğrenmek isteyenler için entegreye datasheeti [**eskisini**](http://fgokcegoz.files.wordpress.com/2011/08/ds130_eski.pdf) ve [**yenisini**](http://fgokcegoz.files.wordpress.com/2011/08/ds130_yeni.pdf) indirip inceleyebilirler. Eski datasheet ile yenisi arasında görünüm açısından biraz farklılıklar olmakla beraber, entegrenin RST ismindeki pininin ismini CE(Chip Enable) olarak değiştirmişler. İyide yapmışlar..

**İşte uygulamaya ait yazılımımız…**

|  |  |
| --- | --- |
| **[DS1302.h](http://www.mcu-turkey.com/msp430-uyg-32/" \l "codesyntax_1" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-32/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-32/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

#ifndef \_DS1302\_H\_

#define \_DS1302\_H\_

#define DS1302\_SCLK P2OUT\_bit.P1

#define DS1302\_IO P2OUT\_bit.P2

#define DS1302\_I0\_IN P2IN\_bit.P2

#define DS1302\_RST P2OUT\_bit.P0

#define SCLK\_DIR P2DIR\_bit.P1

#define IO\_DIR P2DIR\_bit.P2

#define RST\_DIR P2DIR\_bit.P0

#define ADR\_WR\_CREG 0x8E

#define ADR\_RD\_CREG 0x8F

#define ADR\_WR\_TCR 0x90

#define ADR\_RD\_TCR 0x91

#define ADR\_WR\_BURST 0xBE

#define ADR\_RD\_BURST 0xBF

**extern** void Init\_DS1302(void);

**extern** void Read\_DS1302\_Burst\_Mode(void);

#endif //\_DS1302\_H\_

|  |  |
| --- | --- |
| **[DS1302.c](http://www.mcu-turkey.com/msp430-uyg-32/" \l "codesyntax_2" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-32/#codesyntax_2)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-32/#codesyntax_2)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

#include "io430.h"

#include "in430.h"

#include "DS1302.h"

struct CONVERT {

unsigned char NIB\_L:4;

unsigned char NIB\_H:4;

}CONV;

**extern** unsigned char DS1302[7];

void Init\_Line (void)

{

RST\_DIR = 1;

SCLK\_DIR = 1;

IO\_DIR = 1;

DS1302\_RST = 0;

DS1302\_SCLK = 0;

DS1302\_IO = 0;

}

void Reset\_Line(void)

{

DS1302\_SCLK = 0;

DS1302\_RST = 0;

DS1302\_RST = 1;

}

void Write\_Single\_Byte(unsigned char byte)

{

unsigned char i;

IO\_DIR = 1;

for(i=0;i<8;++i)

{

if(byte&0x01)

DS1302\_IO = 1;

else

DS1302\_IO = 0;

DS1302\_SCLK = 0;

DS1302\_SCLK = 1;

byte >>=1;

}

}

unsigned char Read\_Single\_Byte(void)

{

unsigned char j,TEMP=0x00;

IO\_DIR = 0;

for(j=0;j<8;++j)

{

DS1302\_SCLK = 1;

DS1302\_SCLK = 0;

TEMP >>=1;

if(DS1302\_I0\_IN)

TEMP |= 0x80;

else

TEMP &= 0x7F;

}

return TEMP;

}

void Init\_DS1302(void)

{

unsigned char i;

Init\_Line();

Reset\_Line();

Write\_Single\_Byte(ADR\_WR\_CREG);

Write\_Single\_Byte(0x00);

Reset\_Line();

Write\_Single\_Byte(ADR\_WR\_TCR);

Write\_Single\_Byte(0xAB);

Reset\_Line();

Write\_Single\_Byte(ADR\_WR\_BURST);

for(i=0; i<7; i++)

Write\_Single\_Byte(DS1302[i]);

Write\_Single\_Byte(0x00);

Reset\_Line();

}

unsigned char Convert\_Dec(unsigned char val)

{

CONV.NIB\_H = val/16;

CONV.NIB\_L = val%16;

return(CONV.NIB\_H\*10+CONV.NIB\_L);

}

void Read\_DS1302\_Burst\_Mode(void)

{

unsigned char i;

Reset\_Line();

Write\_Single\_Byte(ADR\_RD\_BURST);

for(i=0; i<7; i++)

DS1302[i] = Read\_Single\_Byte();

Reset\_Line();

for(i=0; i<7; i++)

DS1302[i] = Convert\_Dec(DS1302[i]);

}

|  |  |
| --- | --- |
| **[main.c](http://www.mcu-turkey.com/msp430-uyg-32/" \l "codesyntax_3" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-32/#codesyntax_3)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-32/#codesyntax_3)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

#include "io430.h"

#include "in430.h"

#include "main.h"

#include "lcd\_4bit.h"

#include "DS1302.h"

unsigned char DS1302[7]={0x20,0x13,0x14,0x27,0x07,0x03,0x11};

void Lcd\_Write\_Int(unsigned char val)

{

lcd\_putch(val/10+48);

lcd\_putch(val%10+48);

}

void main( void )

{

WDTCTL = WDTPW + WDTHOLD;

DCOCTL=CALDCO\_1MHZ;

BCSCTL1=CALBC1\_1MHZ;

P1SEL = 0x00;

P1OUT = 0x00;

P1DIR = 0xFF;

P2SEL = 0x00;

P2OUT = 0x00;

P2DIR = 0xFF;

lcd\_init();

Init\_DS1302();

for(;;)

{

Read\_DS1302\_Burst\_Mode();

lcd\_goto(1,5);

Lcd\_Write\_Int(DS1302[2]);

lcd\_putch(':');

Lcd\_Write\_Int(DS1302[1]);

lcd\_putch(':');

Lcd\_Write\_Int(DS1302[0]);

lcd\_goto(2,1);

Lcd\_Write\_Int(DS1302[3]);

lcd\_putch('/');

Lcd\_Write\_Int(DS1302[4]);

lcd\_putch('/');

Lcd\_Write\_Int(DS1302[6]);

lcd\_goto(2,14);

switch(DS1302[5])

{

case 1: lcd\_puts("MON");

case 2: lcd\_puts("TUE");

case 3: lcd\_puts("WED");

case 4: lcd\_puts("THU");

case 5: lcd\_puts("FRI");

case 6: lcd\_puts("SAT");

case 7: lcd\_puts("SUN");

}

\_\_delay\_cycles(100000);

}

}

Yazılımı deneyip proteusta simulasyonunu yapmak isteyen arkadaşlar DS1302.c ve DS1302.h dosyalarıyla beraber daha önce kullandığımız lcd\_4bit.c ve lcd\_4bit.h dosyalarınıda eklemeyi unutmasınlar…

Bu arada şundan da bahsetmeliyimki, entegre içerisindeki datalar hex olarak saklanıp decimal olarak yorumlanırlar. Örnek olarak şöyle söyleyelim. Mesela saniye değeri o an hexadecimal olarak 0×34 olsun. Decimal olarak bu değer 52 ye karşılık gelmektedir. Ancak bu böyle yorumlanmaması gerekir. Saniye değeri o an 0×34 değeri olarak okunuyorsa saniye değeri o an gerçek olarak 34 tür. Bu nedenle DS1302 ye data gönderilirken ve alınırken bu durum gözardı edilmemelidir. Bu durum nedeniylede yazılımda Convert\_Dec() isimli bir altprogram kullanılmaktadır.

DS1302 kütüphanesindeki diğer altprogramlardan bahsedecek olursak,

void Init\_Line (void) = Haberleşme hattını kullanıma açar.

void Reset\_Line(void) = Haberleşme hattını resetlemeye yarar.

void Write\_Single\_Byte(unsigned char byte) = DS1302 ye bir byte data göndermeye yarar.

unsigned char Read\_Single\_Byte(void) = DS1302 den bir byte data almaya yarar.

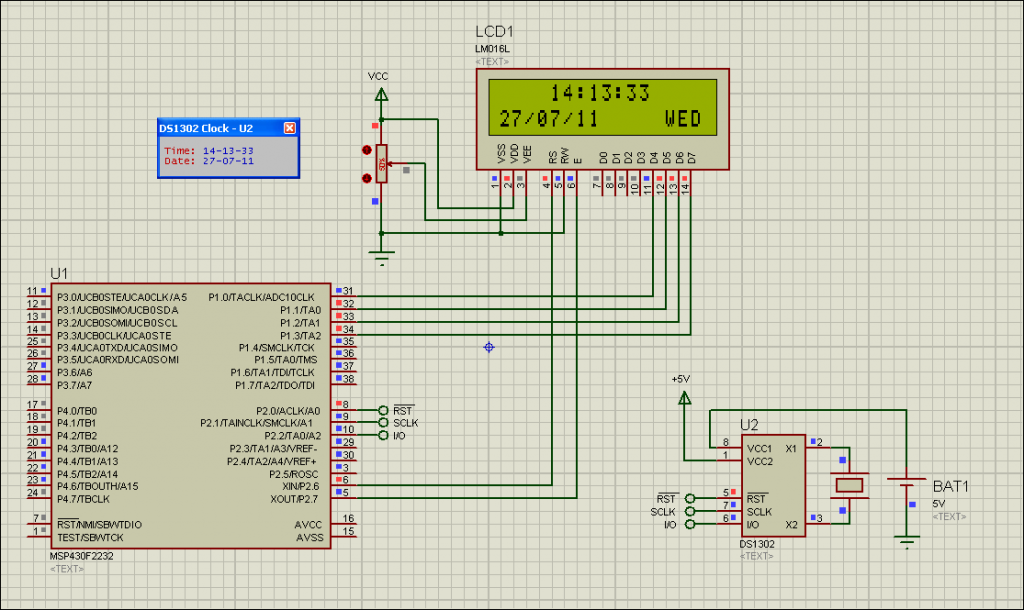
void Init\_DS1302(void) = DS1302 yi kullanıma hazırlar.

unsigned char Convert\_Dec(unsigned char val) = Hex2Dec dönüşümünü yapar.

void Read\_DS1302\_Burst\_Mode(void)  = Burst Mod da tüm tarih saat bilgilerini okumaya yarar.

Main fonksiyonunda yapılan işlemlerden de kısaca bahsedecek olursak sonsuz döngü içerisinde ilk olarak burst mod da tüm tarih saat bilgileri okunup DS1302[] adlı diziye atılır. Daha sonra bu dizideki elemanlar sırasıyla lcd ekrana yazdırılır. Ardından da yaklaşık olarak 100 msn lik bir gecikme verilip aynı işlemler sürekli tekrarlanır. Proteus üzerindeki simulasyon sonuçlarınıda inceleyecek olursak, DS1302 nin saati doğru bir şekilde okunup diğer verilerle birlikte ekrana yazdırılmaktadır.

Ekran görüntüsünüde verelim.



DS1302.c dosyasının içindeki tek byte yazma ve okuma fonksiyonları kullanılarak ds1302 nin ram inede istenilen veri yazdırılabilir. Sizde uygulama üzerinde çalışırken bu gibi şeylerle uğraşırsanız, konuya daha hakim olabilirsiniz. Yine uygulamaya ait komple proje dosyasını paylaşmıyorum:)

Bu arada bir sonraki dersimizde; t domenindeki bir transfer fonksiyonunu Z domenine çevirip nasıl mikroişlemci ile algoritması hazırlanıp koşturulur, o konu hakkında bir uygulamamız olacak. Uygulamayı launchpad üzerinde yapacağız. Bu nedenle oldukça faydalı ve konu içeriği itibariyle oldukça ilgi çekici olacağını düşünüyorum. Çünkü biz bu uygulamayı assembly kodlarıyla 8051 tabanlı bir mikrodenetleyicide sakarya üniversitesindeyken kontrol laboratuvarında yapmıştık. Oldukça ilginç geleceğine eminim. Neyse, bir sonraki dersimizde görüşmek üzere.. Şimdilik Hoşçakalın…

**Ferudun GÖKCEGÖZ**

**fgokcegoz@yahoo.com**