Modül 9: MSP430 ile DS1302 Gerçek Zaman Saat Kullanımı ve Uygulamaları

Giriş

RTC(Real Time Clock/Gerçek Zaman Saat) entegreleri adından anlaşılacağı gibi elektronik devre ve sistemlerde gerçek saat bilgisine ihtiyaç duyulduğunda kullanılır. Burada gerçek zaman saatinden kasıt günlük hayatta kullandığımız saattir. Gerçek zaman saati harici entegre olarak yada mikrodenetleyicilerde özel birim olarak bulunabilir.

Mikrodenetleyicilerde bir çok zamanlayıcı birim bulunmasına rağmen gerçek zaman saatinin önemi ve amacı farklıdır. Gerçek zaman saati entegreleri yada birimleri kendine ait hassas kristal ile çalışıp yüksek doğrulukla saat bilgisi elde ederler. Mikrodenetleyicilerde bulunan zamanlayıcıların hassasiyeti genel olarak daha düşüktür. Ayrıca gerçek zaman saatleri çok düşük güç tüketimine sahip olup harici pil ile beslenip saat ve zaman bilgilerini uzun süre saklayabilirler. Bu sayede kullanıldığı sistemin enerjisinden bağımsız olarak yıllarca saat bilgilerini saklayabilirler.

Gerçek zamanlı saatlerin kullanımı yaygındır. Gerçek zaman saat bilgisi gereken tüm uygulamalarda kullanılabilir. En bilinen örneği ise bilgisayarlarımızda kullandığımız sistem saatimizdir. Bilgisayarlar sistem saatini bilgisayar anakartı üzerinde bulunan RTC birimi sayesinde edinmektedirler. Aynı şekilde bilgisayar anakartı üzerinde ki pili sökünce sistem saati ve bios ayarları sıfırlanır. Saatin ve sistem ayarlarının sıfırlanmasının nedeni saat ve ayarların tutulduğu hafızanın bu pil ile beslenmesidir. Normal şartlarda bilgisayarımız çalışmasa bile saat bilgisi ve bios ayarlarımız anakart üzerinde bulunan pil sayesinde silinmez.

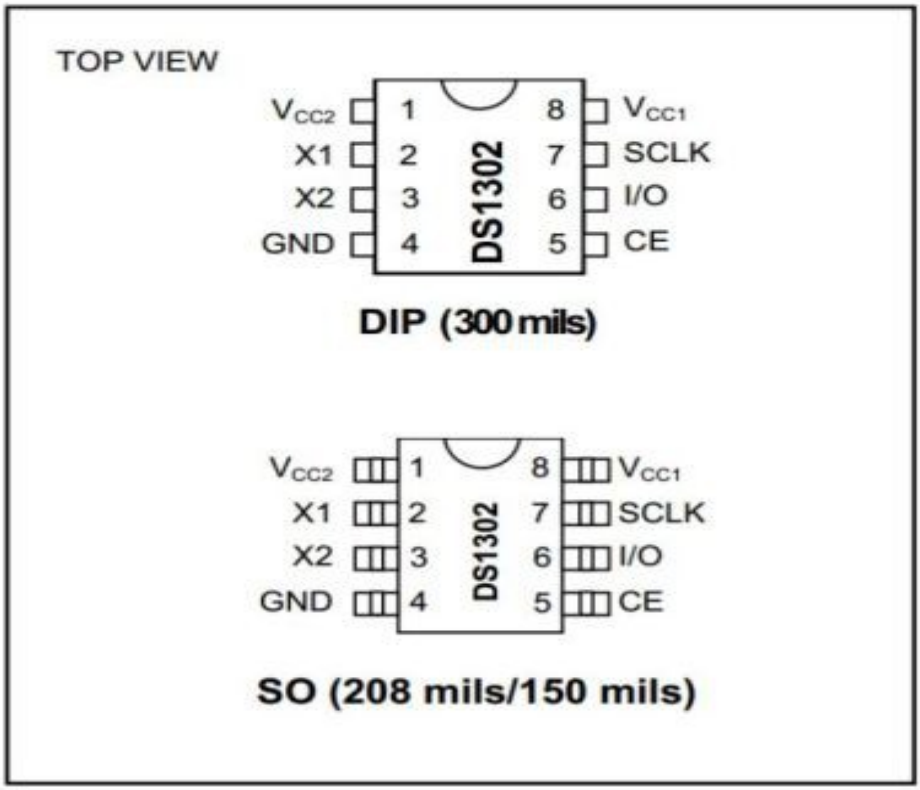
MSP430 geliştirme kartımız üzerinde harici entegre olarak DS1302 gerçek zamanlı saat entegresi ve gerekli devresi bulunmaktadır. DS1302 entegresi maxim integrated firması tarafından üretilen yaygın olarak kullanılan gerçek zaman saat entegresidir. Bu modül kapsamında DS1302 entegresi incelenip, MSP430 denetleyiciler ile kullanımına değinilecektir.

DS1302 Entegresi

DS1302, maxim integrated firmasının ürettiği gerçek zaman saat ve batarya destekli RAM entegresidir. Özellikleri aşağıdaki gibidir.

- Saniye, Dakika, Saat, Ayın günleri, Ay, Haftanın günleri ve Yıl(artık yıl dahil) sayabilme
- 2100 yılına kadar geçerli
- 31x8 bit pil batarya destekli RAM
- Düşük pin gerekesinimi için seri bağlantı
- 2.0V - 5.5V çalışma gerilimi
- 2V'ta 300nA'den düşük akım tüketimi
- Saat bilgisi yada RAM okuma/yazma için tekli veya çoklu byte(Burst Mode) veri transferi
- 8-Pin DIP yada 8-Pin SO SMD paket seçeneği

- Basit 3-hat seri arayüz
- TTL uyumlul(Vcc=5V)
- -40°C - +85°C Endüstriyel çalışma sıcaklığı(Opsiyonel)
- DS1202 Uyumlu
- Underwriters Laboratories (UL®) onaylı

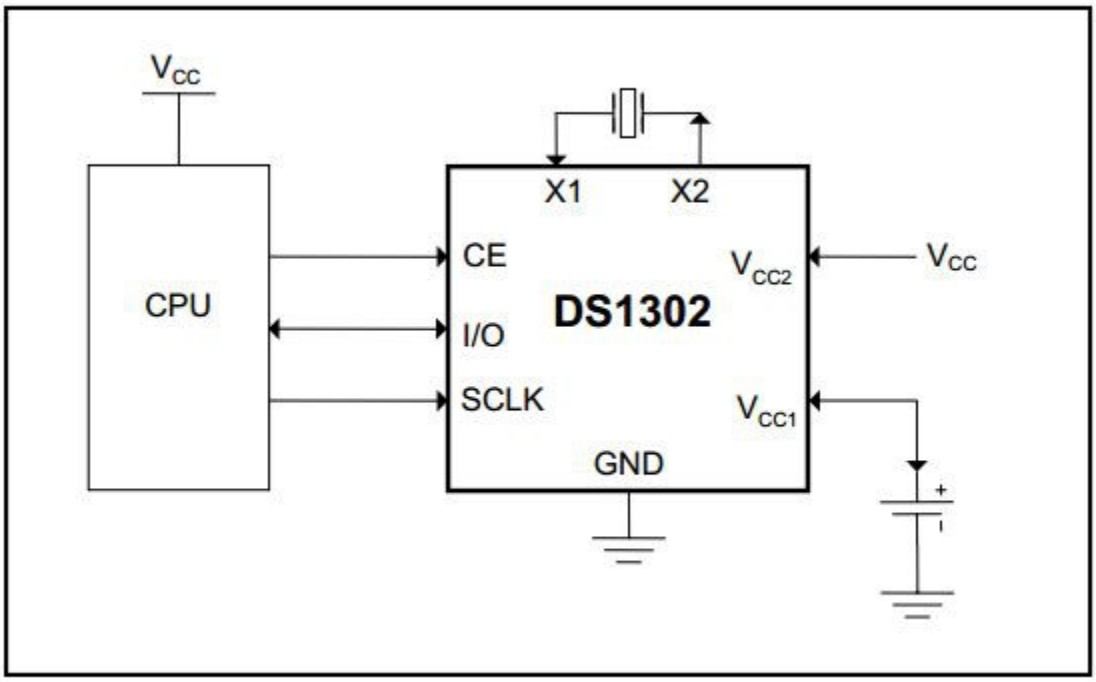


Şekil 1: DS1302 Paket Tipleri ve Pin Bağlantıları

Şekil 1'de DS1302 entegresine ait pin bağlantıları ve paket tipleri görülmektedir. MSP430 geliştirme kartımız üzerinde U5 referans numaralı 8-DIP kılıf yapısında DS1302 entegresi ve gerekli uygulama devresi bulunmaktadır. DS1302 entegresi aynı zamanda 31x8bit batarya destekli RAM hafıza ve kendi kullandığı pili şarj edebilmek için dahili şarj devresi barıdırmaktadır. Biz bu modül kapsamında DS1302 entegresinin RTC özelliğini kullanıp uygulamalarını yapacağız. Detaylı bilgi için DS1302 entegresinin kullanma kılavuzu incelenebilir.

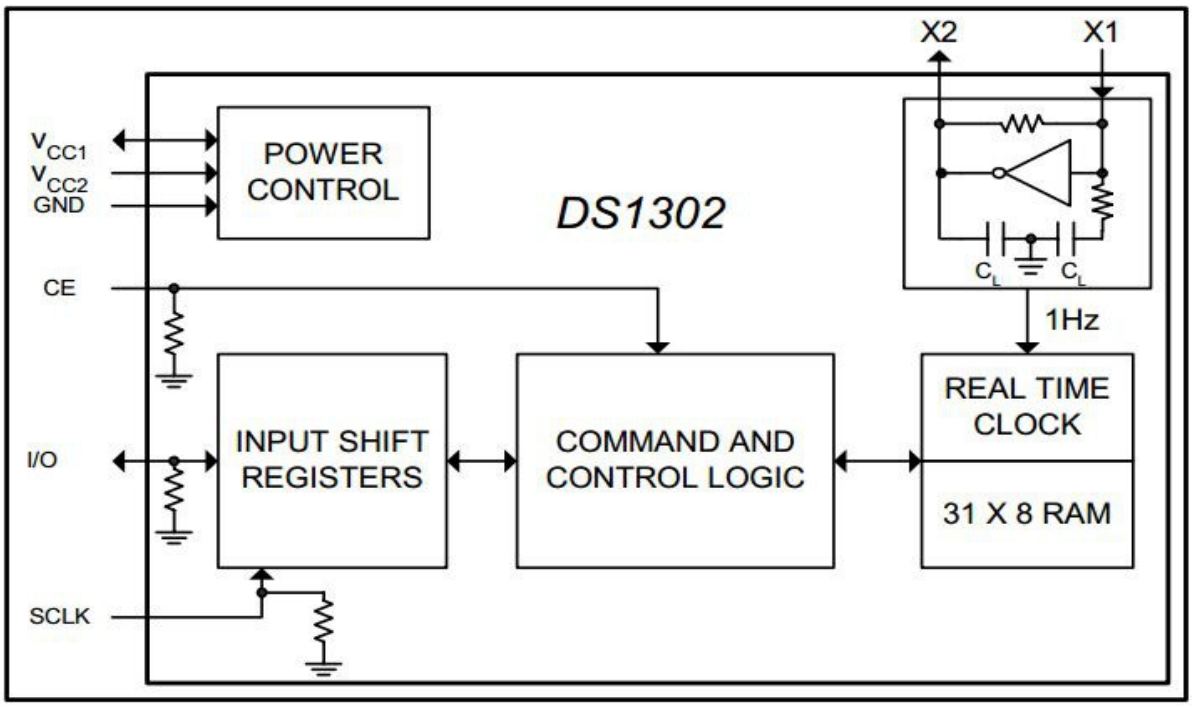
DS1302 entegresi içerisinde gerçek zamanlı saat/takvim ve 31 Byte statik RAM bulundurur. Gerçek zaman saati saniye, dakika, saat, ayın günü, ay, haftanın günü ve yıl bilgilerini içerir. DS1302 entegresi RTC gerekli zamanlamalarını harici olarak bağlanan 32.768 kHz kristal ile hassas bir şekilde sağlar. Aynı zamanda DS1302 harici olarak pil ile beslendiği için kullanıldığı sistemin enerjisi kesilse dahil kendi içinde çalışmaya devam edip saat/tarih ve RAM bilgilerini saklar

DS1302 entegresi haberleşmek için basit senkron haberleşme kullanır. Haberleşmek için sadece 3 pin gereklidir. Bu pinler CE(Chip Enable/Çip Yetki), I/O(Input/Output-Giriş/Çıkış) ve SCLK(Serial Clock/Seri Saat İşareti) pinleridir. Bu pinler sayesinde denetleyicimizi DS1302 entegresi ile haberleştirebilir saat, tarih bilgilerini okuyup yazabiliriz.



Şekil 2: DS1302 Temel Uygulama Devresi

Şekil 2'de görüldüğü gibi DS1302 basit bir uygulama devresine sahiptir. Görüldüğü gibi X1, ve X2 pinlerine 32.768 kHz kristal bağlanması, Vcc1 pinine pil, Vcc2 pinine besleme gerilimi, GND pininin sistemin GDN'sine bağlanması ve haberleşme pinlerinin kullanılan denetleyiciye bağlanması yeterlidir. VCC1 pinine piyasada blunan tipik 3V CR2032 gibi bir pil bağlanması yeterlidir. Pil bağlanmaz ise sistemin sistemin enerjisi kesilmesi durmunda saat/tarih bilgileri ve RAM bellek içerisinde bulunan bilgiler silinir.



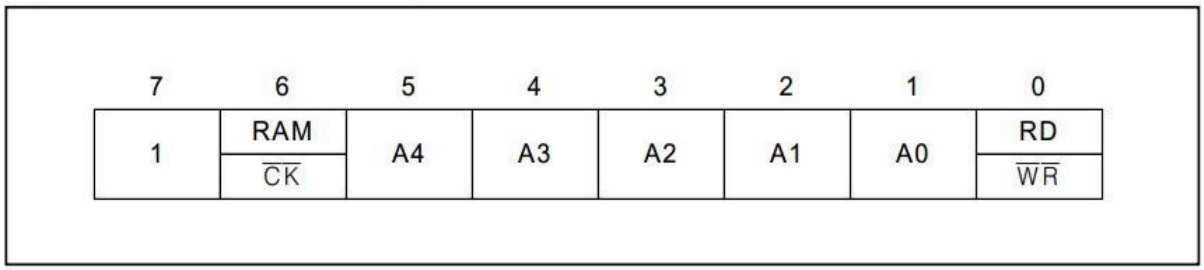
Şekil 3: DS1302 İç Yapısı Blok Diyagramı

Şekil 3'te DS1302 entegresinin iç yapısı görülmektedir. Görüldüğü gibi osilatör devresi sayesinde RTC birimine 1Hz saat işareti gönderilmektedir. Bu 1Hz saat işareti RTC biriminin temel işaretidir. Tüm sayma/zamanlama işlemleri bu işarete göre yapılır. Dolayısıyla osilatör devresinin doğruluğu direk olarak RTC birimini etkiler. Bu nedenle hassas bir zamanlama için kaliteli, yüksek doğruluklu kristal kullanmakta fayda var. Aynı zamanda PCB çiziminde kristal bağlantılarına ayrıca dikkat edilmeli ve kullanım kılavuzunda bulunan öneriler dikkate alınmalıdır.

DS1302 3-Wire Haberleşme Protokolü

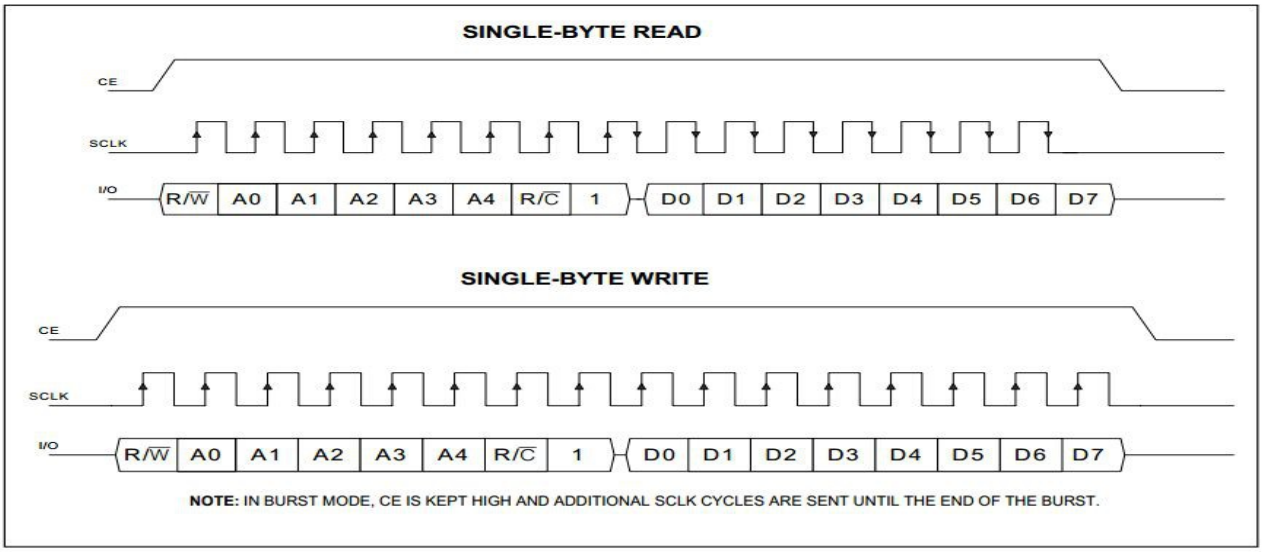
DS1302 entegresi haberleşmek için 3 hat gerektiren 3-wire senkron seri haberleşme protokolünü kullanır. Bu sayede karşı taraftaki kontrolcü tarafından bilgiler okunup yazılabilir. DS1302 slave olarak çalışır yani karşı taraftaki master denetleyici tarafından kontrol edilir.

Yazılımsal olarak DS1302 entegresi ile haberleşmeye başlamadan önce master cihazın kontrol komutunu göndermesi gerekir. Kontrol komutu veri transferini başlatır. Kontrol komutu içerisinde haberleşme ile ilgili bilgiler bulunmaktadır.



Şekil 4: DS1302 Kontrol Komutu Yapısı

Şekil 4'te kontrol komutunun detaylı yapısı görülmektedir. Kontrol komutunda bit 7 daima birdir. Bu bit eğer sıfır olursa DS1302 ile haberleşme yapılamaz. Bit 6 ise RAM bellekten mi yada RTC biriminden mi okuma/yazma yapılacağını belirtir. Bu bit eğer sıfır ise okuma/yazma işlemi RTC birimi için yapılır bir ise RAM bellek için yapılır. Bit 0 ise okuma/yazma işlemi seçmek için yapılır. Bu bit bir ise DS1302'den okuma, sıfır ise DS1302'ye yazma yapılır. Diğer kalan 5-1 bitleri ise okuma/yazma yapılacak adresi belirtir. Kontrol komutunu isteğimiz doğrultusunda gönderdikten sonra DS1302'den okuma/yazma yapabiliriz.



Şekil 5: DS1302 Veri Transfer Yapısı

Şekil 5'te DS1302'ye ait haberleşme yapısı görülebilir. DS1302 ile haberleşmede her zaman veri okuma/yazmaya LSB yani en düşük değerli bit ile başlanır. Yani LSB ilk(LSB First) haberleşme yapılır. Haberleşmeye başlamadan önce CE yani yetki pininin yüksek seviyeye çekilmesi gerekir. Haberleşme olmadığı durumlarda CE ve SCLK pini düşük seviyede tutulur. Şekil 5'te üst kısımda görüldüğü gibi tek byte okuma işlemi öncelikle CE pini yüksek seviyeye çekilerek kontrol komutunun uygun şekilde gönderilmesi gerekir. Haberleşme işleminde SCLK pininden saat işareti master tarafından sağlanır. İlgili adresi okumak için kontrol komutu I/O pini üzerinden gönderildikten sonra master tarfından 8 düşen kenar saat darbesi daha gönderilerek istenen veri I/O pininden seri olarak alınır. Okuma işlemi yapıldıktan sonra CE ve SCLK pinleri düşük seviyeye çekilir.

Benzer şekilde yazma işlemi için yine CE pini yüksek seviyeye çekilerek gerekli kontrol komutu I/O pini üzerinden gönderilir. Kontrol komutu gönderildikten sonra ilgili adrese yazılmak istenen veri 8 yükselen kenar saat darbesi ile I/O pini üzerinden DS1302'ye gönderilir. Sonrasında CE ve SCLK pinleri düşük seviyeye çekilerek yazma işlemi tamamlanır. Bu şekilde zamanlamalara sadık kalarak DS1302'nin istenen adresine veri yazabilir yada okuyabiliriz.

RTC

READ	WRITE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	RANGE
81h	80h	CH	10 Seconds			Seconds				00–59
83h	82h		10 Minutes			Minutes				00–59
85h	84h	12/$\overline{24}$	0	10 $\overline{\text{AM/PM}}$	Hour	Hour				1–12/0–23
87h	86h	0	0	10 Date		Date				1–31
89h	88h	0	0	0	10 Month	Month				1–12
8Bh	8Ah	0	0	0	0	0	Day			1–7
8Dh	8Ch	10 Year				Year				00–99
8Fh	8Eh	WP	0	0	0	0	0	0	0	—
91h	90h	TCS	TCS	TCS	TCS	DS	DS	RS	RS	—

CLOCK BURST

BFh	BEh

Şekil 6: DS1302 RTC Birimi Hafıza Haritası

Şekil 6'da DS1302 RTC birimine ait hafıza haritasını görebilirsiniz. Görüldüğü gibi haritada yukarıdan aşağı sıraısyla saniye, dakika, saat, gün, ay, haftanın günü ve yıl bilgileri tutulmaktadır. Saat/Tarih bilgileri BCD olarak tutulmaktadır. Aynı zamanda DS1302 am/pm yada 24 saat zaman uygulamasınada desteklemektedir. Soldaki iki sütunda ise ilgili hafıza bölgelerine, okuma/yazma yapmak için gerekli kontrol komutları bulunmaktadır. Örneğin DS1302'den dakika bilgisi okumak için 83h kontrol komutunu göndermemiz gerekmektedir. Benzer şekilde saat bilgisi yazmak için ise 84h kontrol komutunu göndermemiz gerekmektedir. Bu sayede kontrol komutlarını kullanarak istediğiniz saat/takvim bilgisini okuyup yazabilirsiniz. Ayrıca tabloda bulunan CH biti saati durdurup uyku moduna almak için kullanılır. Bu bit bir ise saat birimi durur sıfır ise nornal çalışmasına devam eder. Tabloda bulunan WP biti ise yazma koruma bitidir. Yazma işlemine başlamadan önce bu bitin sıfır yapılması gerekir. Aksi takdirde yazma işlemine izin verilmez. Haritanın sonunda bulunan byte ise pil şarj devresinin kontrol bytedır. Pil şarj devresi ile ilgili gerekli ayarlamalar bu byte ile yapılır. Detaylı bilgi için DS1302 kullanma kılavuzuna bakılabilir.

Haritanın sol alt köşesinde ise Clock Burst adı altında iki tane kontrol komutu(BFh, BEh) bulunur. Burst komutları ile RTC hafıza haritasına çoklu byte okuma/yazma yapılır. Örneğin BFh komutu gönderildikten sonra haritanın başından itibaren sırasıyla 8 byte veri okunur. Benzer şekil de BFh komutu ile RTC haritasına baştan itibaren 8 byte veri yazılabilir. Çoklu okuma/yazma işleminde benzer olarak CE pini haberleşme işlemi bitene kadar yüksek seviyede tutulmalıdır.

RAM

C1h	C0h		00-FFh
C3h	C2h		00-FFh
C5h	C4h		00-FFh
. . .	. . .		. . .
FDh	FCh		00-FFh

RAM BURST

FFh	FEh

Şekil 7: DS1302 RAM Bellek Haritası

Şekil 7'de DS1302 RAM belleğine ait hafıza haritası görülmektedir. RTC biriminin hafıza haritası ile anlatılanlar RAM hafıza haritası içinde geçerlidir. RAM hafızada 31 byte olduğu için burst mod okuma/yazma işlemlerinde 31 byte okuma/yazma yapılır.

DS1302 3-Wire Haberleşme Kütüphanesi

DS1302 ile MSP430 denetleyicimizi haberleştirip saat bilgilerini okuyup yazabillmek için gerekli yazılımsal kütüphanenin hazırlanması gerekir. Bu kütüphane içerisinde başlangıç işlemleri ve haberleşme işlemleri ile ilgili gerekli fonksiyonların modüler bir şekilde hazırlanması gerekir. Kütüphanemizi hazırladıktan sonra MSP430 denetleyicimiz ile DS1302 entegresini haberleştirebilir saat ve takvim bilgilerini okuyup yazabiliriz.

MSP430 denetleyiciler için hazırlanmış olan DS1302 kütüphanesi aşağıdaki gibidir.

DS1302.h başlık dosyası

```
#ifndef __DS1302_H
#define __DS1302_H

#ifdef __cplusplus
 extern "C" {
#endif

// DS1302 port ve bit tanımlamaları
#define DS1302_SCLK_BIT          BIT0
#define DS1302_SCLK_PORT         P2OUT
#define DS1302_SCLK_PORT_DIR     P2DIR
#define DS1302_SCLK_PORT_SEL     P2SEL
#define DS1302_SCLK_PORT_SEL2    P2SEL2
#define DS1302_IO_BIT            BIT1
#define DS1302_IO_PORT           P2OUT
#define DS1302_IO_PORT_IN        P2IN
#define DS1302_IO_PORT_DIR       P2DIR
#define DS1302_IO_PORT_SEL       P2SEL
#define DS1302_IO_PORT_SEL2      P2SEL2
#define DS1302_CE_BIT            BIT2
#define DS1302_CE_PORT           P2OUT
#define DS1302_CE_PORT_DIR       P2DIR
#define DS1302_CE_PORT_SEL       P2SEL
#define DS1302_CE_PORT_SEL2      P2SEL2
```

```
// Haberleşme işlemleri için gerekli makro tanımlamaları
#define     HIGH_SCLK DS1302_SCLK_PORT     |=    DS1302_SCLK_BIT
#define     LOW_SCLK  DS1302_SCLK_PORT     &=  ~DS1302_SCLK_BIT
#define     HIGH_IO           DS1302_IO_PORT        |=     DS1302_IO_BIT
#define     LOW_IO            DS1302_IO_PORT        &=   ~DS1302_IO_BIT
#define     HIGH_IO_DIR       DS1302_IO_PORT_DIR    |=     DS1302_IO_BIT
#define     LOW_IO_DIR        DS1302_IO_PORT_DIR    &=   ~DS1302_IO_BIT
#define     READ_IO           DS1302_IO_PORT_IN     &      DS1302_IO_BIT
#define     HIGH_CE           DS1302_CE_PORT        |=     DS1302_CE_BIT
#define     LOW_CE            DS1302_CE_PORT        &=   ~DS1302_CE_BIT

// Saat ve tarih bilgisini tutan yapı
struct _DS1302 {
unsigned char Saniye;
unsigned char Dakika;
unsigned char Saat;
unsigned char AyinGunu;
unsigned char Ay;
unsigned char HaftaninGunu;
unsigned char Sene;
};

// DS1302 kütüphanesinde kullanılan fonksiyon prototipleri
void DS1302_Ayarla(void);
void DS1302_Byte_Yaz(unsigned char veri);
unsigned char DS1302_Byte_Oku(void);
void DS1302_Veri_Yaz(unsigned char adres, unsigned char veri);
unsigned char DS1302_Veri_Oku(unsigned char adres);
unsigned char Sayi_Cevir(unsigned char);
unsigned char BCD_Cevir(unsigned char);
void DS1302_Saat_Tarih_Yaz(struct _DS1302 *ptr);
void DS1302_Saat_Tarih_Oku(struct _DS1302 *ptr);

#ifdef __cplusplus
}
#endif

#endif /*__DS1302_H*/
```

DS1302 başlık dosyasında DS1302 ile ilgili port tanımlamaları, port işlemleri ile ilgili makro kodlar ve kullanılan fonksiyonların prototipleri bulunmaktadır.
Ayrıca DS1302 başlık dosyasında _DS1302 isimli yapı değişkenimiz bulunmaktadır. Yapı yani struct tipinde ki değişkenler C programlama dilinden bilindiği gibi içerisinde çoklu sayıda ve çeşitli tipde değişkenleri tutabilirler. Uygulamalarınıza göre kendi yapılarınızı oluşturup kodunuzu daha anlaşılır hale getirebilir, daha verimli kodlar yazabilirsiniz. DS1302 uygulamasında saat ve takvim bilgilerini tutabilmek için bir çok değişken tanımlamamız gerekir. Bu değişkenlerin derli toplu olması ve kullanımı kolay olması açısından _DS1302 adında bir yapı tanımlanmıştır. Saat ve tarih bilgileri bu yapıdan türetilen değişkenlerde saklanmaktadır. Bu sayede tek bir yapı değişkeni ile saat/takvim bilgilerini saklayabiliriz. Uygulamada kullanımını görüldüğünde yapı değişkeninin mantığı daha iyi anlaşılabilir.

DS1302.c Kütüphanesi kaynak dosyasının içeriği aşaıda ki gibidir.

```
#include "msp430.h"                 // MSP430 başlık dosyası
#include "ds1302.h"                 // DS1302 başlık dosyası

// DS1302 başlangıç ayarlarını yapan fonksiyon
void DS1302_Ayarla(void){
        unsigned char temp;
        DS1302_SCLK_PORT_DIR  |=  DS1302_SCLK_BIT;
        DS1302_SCLK_PORT_SEL  &= ~DS1302_SCLK_BIT;
        DS1302_SCLK_PORT_SEL2 &= ~DS1302_SCLK_BIT;
        DS1302_IO_PORT_DIR    |=  DS1302_IO_BIT;
        DS1302_IO_PORT_SEL    &= ~DS1302_IO_BIT;
        DS1302_IO_PORT_SEL2   &= ~DS1302_IO_BIT;
        DS1302_CE_PORT_DIR    |=  DS1302_CE_BIT;
        DS1302_CE_PORT_SEL    &= ~DS1302_CE_BIT;
        DS1302_CE_PORT_SEL2   &= DS1302_CE_BIT;
        LOW_CE;
        LOW_SCLK;
        DS1302_Veri_Yaz(0x80,0);
        DS1302_Veri_Yaz(0x90,0xa4);
        temp=DS1302_Veri_Oku(0x81);
        if((temp & 0x80) != 0)
        DS1302_Veri_Yaz(0x80,0);
}
// DS1302'ye bir byte veri yazan fonksiyon
void DS1302_Byte_Yaz(unsigned char veri){
        unsigned char temp;
        HIGH_IO_DIR;
        for(temp=0;temp<8;temp++){
        LOW_IO;
        if(veri & 0x01) HIGH_IO;
        veri >>= 1;
        LOW_SCLK;
        HIGH_SCLK;
        }
}
// DS1302'den bir byte veri okuyan fonksiyon
unsigned char DS1302_Byte_Oku(void){
        unsigned char temp,veri=0;
        LOW_IO_DIR;
        for(temp=0;temp<8;temp++){
        HIGH_SCLK;
        LOW_SCLK;
        veri >>= 1;
        if(READ_IO) veri |= 0x80;
        }
        return veri;
}
```

```
// DS1302'nin belirtilen adresine bir byte veri yazan fonksiyon
void DS1302_Veri_Yaz(unsigned char adres, unsigned char veri){
        LOW_SCLK;
        HIGH_CE;
        DS1302_Byte_Yaz(adres);
        DS1302_Byte_Yaz(veri);
        LOW_CE;
}
// DS1302'nin belirtilen adresinden bir byte veri okuyan fonksiyon
unsigned char DS1302_Veri_Oku(unsigned char adres){
        unsigned char temp;
        HIGH_CE;
        DS1302_Byte_Yaz(adres);
        temp = DS1302_Byte_Oku();
        LOW_CE;
        return temp;
}
// Girilen desimal sayıyı BCD sayıya çeviren fonksiyon
unsigned char BCD_Cevir(unsigned char sayi){
  unsigned char niblel, nibleh;
  niblel = sayi%10;
  nibleh = sayi/10;
  return((nibleh<<4)+niblel);
}
// Girilen BCD sayıyı desimal sayıya çeviren fonksiyon
unsigned char Sayi_Cevir(unsigned char BCD){
        unsigned char temp1,temp2;
        temp1=((BCD>>4)&0x0f)*10;
        temp2=BCD&0x0f;
        BCD=temp1+temp2;
        return BCD;
}
// DS1302'ye saat ve tarih ayarlarını kayıt eden fonksiyon
void DS1302_Saat_Tarih_Yaz(struct _DS1302 *DS1302){
        DS1302_Veri_Yaz(0x86,BCD_Cevir(DS1302->AyinGunu));
        DS1302_Veri_Yaz(0x88,BCD_Cevir(DS1302->Ay));
        DS1302_Veri_Yaz(0x8c,BCD_Cevir(DS1302->Sene));
        DS1302_Veri_Yaz(0x8a,BCD_Cevir(DS1302->HaftaninGunu));
        DS1302_Veri_Yaz(0x84,BCD_Cevir(DS1302->Saat));
        DS1302_Veri_Yaz(0x82,BCD_Cevir(DS1302->Dakika));
        DS1302_Veri_Yaz(0x80,0x00);
}
// DS1302'den saat ve tarih bilgilerini okuyan fonksiyon
void DS1302_Saat_Tarih_Oku(struct _DS1302 *DS1302){
        DS1302->Saniye = Sayi_Cevir(DS1302_Veri_Oku(0x81)&0x7f);
        DS1302->Dakika = Sayi_Cevir(DS1302_Veri_Oku(0x83));
        DS1302->Saat = Sayi_Cevir(DS1302_Veri_Oku(0x85));
        DS1302->AyinGunu = Sayi_Cevir(DS1302_Veri_Oku(0x87));
        DS1302->Ay = Sayi_Cevir(DS1302_Veri_Oku(0x89));
        DS1302->HaftaninGunu = Sayi_Cevir(DS1302_Veri_Oku(0x8b));
        DS1302->Sene = Sayi_Cevir(DS1302_Veri_Oku(0x8d));
}
```

DS1302.c Kütüphane kaynak dosyamızda kullanılan fonksiyonları kısaca açıklayalım.

DS1302_Ayarla: DS1302 başlangıç ayarlarını yapan fonksiyondur. Değişken almaz yada döndürmez. Başlangıçta bir kez çalıştırılmalıdır.
DS1302_Byte_Yaz: DS1302 entegresine bir byte veri yazan fonksiyondur. Girilen değişken değerini DS1302'ye gönderir. Değişken döndürmez. Kütüphane içindeki diğer fonksiyonlar tarafından kullanılır. Alt seviye yardımcı fonksiyondur.
DS1302_Byte_Oku: DS1302 entegresinden bir byte veri okuyan fonksiyondur. Değişken almaz. Okunan değeri değişken olarak geri döndürür. Kütüphane içindeki diğer fonksiyonlar tarafından kullanılır. Alt seviye yardımcı fonksiyondur.
DS1302_Veri_Yaz: DS1302'nin belirtilen adresine bir byte veri yazan fonksiyondur. Adres ve yazılacak veriyi değişken olarak alır. Değişken döndürmez.
DS1302_Veri_Oku: DS1302'nin belirtilen adresinden bir byte veri okuyan fonksiyondur. Adresi değişken olarak alır. Okunan değeri değişken olarak döndürür.
BCD_Cevir: Girilen değişken değerini BCD olarak çevirip döndüren fonksiyondur. Kütüphane içinde diğer fonksiyonlar tarafından kullanılır. Alt seviye yardımcı fonksiyondur.
Sayi_Cevir: Girilen BCD değerini sayıya çevirip döndüren fonksiyondur. Kütüphane içinde diğer fonksiyonlar tarafından kullanılır. Alt seviye yardımcı fonksiyondur.
DS1302 entegresinde RTC bilgileri BCD olarak saklandığı için saat/tarih okuma ve yazma işlemlerinde BCD_Cevir ve Sayi_Cevir fonksiyonlarından faydalanılmaktadır.
DS1302_Saat_Tarih_Yaz: Değişken olarak girilen _DS1302 yapı tipindeki değerleri DS1302'ye yazan fonksiyondur. Saat ve tarih ayarlama amacıyla kullanılır. Saniye bilgisi daima sıfır olarak yazılır.
DS1302_Saat_Tarih_Oku: DS1302 entegresinden saat ve tarih bilgilerini okur _DS1302 yapı tipindeki değişkene kayıt eder. Değişken olarak bilgilerin yazılacağı yapı değişkeninin adresini alır.

DS1302 kütüphanemiz yukarıda bahsedilen fonksiyonlardan oluşmaktadır. Bu kütüphaneyi kullanarak MSP430 kullandığınız uygulamalarda DS1302 entegesi ile RTC bilgisi elde edebilirsiniz. DS1302 ile ilgili bu kadar anlatımdan sonra uygulamalarına geçebiliriz.

Uygulama 9.1 DS1302 Kullanımı

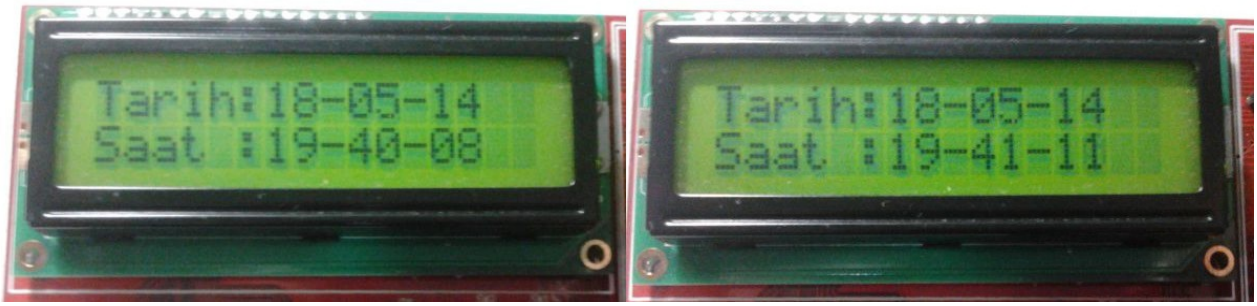
Bu uygulamamızda DS1302'nin temel çalışmasını inceleyeceğiz. DS1302'den saat ve tarih bilgilerini okuyup LCD ekranda göstereceğiz. Uygulamanın kodları aşağıda ki gibidir.

```
#include <msp430.h>                     // MSP430 başlık dosyası
#include "LCD.h"                        // LCD başlık dosyası
#include "DS1302.h"                     // DS1302 başlık dosyası

struct _DS1302 DS1302;                  // Saat/tarih bilgilerini tutan yapı
unsigned char bGecikmeSayac=0;          // Gecikme sayacı
void main(void) {
   WDTCTL  = WDTPW | WDTHOLD;           // Watchdog timeri durdur.
   BCSCTL1 = CALBC1_1MHZ;               // Dahili osilatörü 1MHz'e ayarla.
   DCOCTL = CALDCO_1MHZ;                // Dahili osilatörü 1MHz'e ayarla.
   DS1302_Ayarla();                     // DS1302 ayarları
   LCD_Ayarla();                        // LCD ayarları
   TA0CCTL0 = CCIE;                     // Timer0 CCR0 ayarları
   TA0CCR0  = 50000;                    // Timer0 kesme periyodu ~50ms
   TA0CTL   = TASSEL_2 + MC_2;          // Timer0 ayarları
   DS1302.Saniye=0;                     // Saniyeyi ayarla
   DS1302.Dakika=40;                    // Dakikayı ayarla
```

```
    DS1302.Saat=19;                                 // Saati ayarla
    DS1302.AyinGunu=18;                             // Ayın gününü ayarla
    DS1302.Ay=5;                                    // Ayı ayarla
    DS1302.HaftaninGunu=7;                          // Haftanın gününü ayarla
    DS1302.Sene=14;                                 // Seneyi ayarla
    DS1302_Saat_Tarih_Yaz(&DS1302);                 // Ayarlanan değerleri DS1302'ye yaz
    LCD_Temizle();                                  // LCD ekranı temizle
    _BIS_SR(GIE);                                   // Kesmeleri aç
    while(1){                                       // Sonsuz döngü
    DS1302_Saat_Tarih_Oku(&DS1302);                 // DS1302'den saat/tarih bilgisini oku
    LCD_Git_XY(1,1);                                // Kursörü 1.satır 1.sütuna götür
    LCD_Yazi_Yaz("Tarih:");                         // LCD ekrana Tarih: yazdır.
    LCD_Karakter_Yaz(DS1302.AyinGunu/10+0x30);      // Ay günü onlar basamağını yazdır
    LCD_Karakter_Yaz(DS1302.AyinGunu%10+0x30);      // Ay günü birler basamağını yazdır.
    LCD_Karakter_Yaz('-');                          // - yazdır.
    LCD_Karakter_Yaz(DS1302.Ay/10+0x30);            // Ayın onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Ay%10+0x30);            // Ayın birler basamağını yazdır.
    LCD_Karakter_Yaz('-');                          // - yazdır.
    LCD_Karakter_Yaz(DS1302.Sene/10+0x30);          // Senenin onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Sene%10+0x30);          // Senenin birler basamağını yazdır.
    LCD_Git_XY(2,1);                                // Kursörü 2.satır 1.sütuna götür.
    LCD_Yazi_Yaz("Saat :");                         // LCD ekrana Saat : yazdır.
    LCD_Karakter_Yaz(DS1302.Saat/10+0x30);          // Saatin onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Saat%10+0x30);          // Saatin birler basamağını yazdır.
    LCD_Karakter_Yaz('-');                          // - yazdır.
    LCD_Karakter_Yaz(DS1302.Dakika/10+0x30);        // Dakikanın onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Dakika%10+0x30);        // Dakikanın birler basamağını yazdır.
    LCD_Karakter_Yaz('-');                          // - yazdır.
    LCD_Karakter_Yaz(DS1302.Saniye/10+0x30);        // Saniyenin onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Saniye%10+0x30);        // Saniyenin birler basamağını yazdır.
    _BIS_SR(LPM0_bits);                             // Uykuya gir
    }                                               // Sonsuz döngü
}

// Timer_A0 Zamanlayıcısı CCR0 kesme vektörü
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
        if(++bGecikmeSayac>=20){                    // ~1sn oldu mu?
        bGecikmeSayac=0;                            // Sayacı sıfırla
        __bic_SR_register_on_exit(CPUOFF);          // İşlemciyi uykudan uyandır.
        }
        TA0CCR0 += 50000;                           // Zamanlayıcıyı yeniden kur.
}
```

Kodları incelersek işlemci çalışmaya başlar başlamaz ilk ayarlamaları ve osilatör ayarlarını yapar. Sonrasında DS1302 ve LCD ilk ayarlarını yapar. Devamında zamanlayıcı 50ms de bir kesme üretilecek şekilde kurularak DS1302'ye varsayılan saat tarih bilgisi yüklenir. Sonrasında LCD ekrana temizlenerek işlemci sonsuz döngüye girer. Sonsuz döngüde işlemci DS1302'den saat/tarih bilgisini okuyarak LCD ekranada görüntüler. Değerlerin görüntülenmesi için birler ve onlar şeklinde basamaklarına ayrılır ve karaktere dönüştürmek için 0x30 sayısı eklenir. Bu sayede okunan değerler LCD ekranda düzgün bir

şekilde görüntülenir. Sonrasında işlemci uyku moduna girer. 20 zamanlayıcı kesmesi yani yaklaşık 1 saniye sonrasında işlemci uykudan uyandırılır ve tekrar saat/tarih bilgileri okunup LCD ekranda görüntülenir. İşlemci sürekli olarak bu şekilde çalışmaya devam eder.



Şekil 8: Uygulama 9.1 Çalışma Görüntüleri

Kodlarımızı MSP430G2553 denetleyicimize yüklemeden önce geliştirme kartı ayarlarının yapılması gerekir. Uygulamayı çalıştırabilmek için sw24 anahtarları ve sw2 anahtarları on konumuna getirilerek DS1302 ve LCD kullanıma hazır hale getirilmelidir. Ayrıca sorun çıkmaması açısından diğer kullanılmayan birimlerin anahtar ve atlamaları off konumuna getirilmelidir. Geliştirme kartı ayarlarını yaptıktan sonra kodlarımızı derleyip kartımıza yükleyebiliriz.

Kodlarımızı sorunsuz bir şekilde yükledikten sonra herhangi bir hata yok ise uygulamamız çalışmaya başlyacaktır. Görüldüğü gibi uygulamamız kodların başında varsayılan olarak yüklenen saat ve tarih değerlerini yükleyerek çalışmaya başlar. Sonrasında uygulama normal olarak sayar, saat ve tarih değerlerini ekranda gösterir.

Uygulama 9.2 DS1302'li Saat ve Takvim

Bu uygulamamızda DS1302 ve LCD ekranımızı kullanarak dijital saat ve takvim uygulaması yapacağız. Port 1'e bağlı butonlar ile saat ve tarih ayarlarını yapacağız. Aynı zamanda uygulamamız düşük güç tüketmesi için ayar tuşlarına basılmadığı yada saniye bilgisi değişmediği sürece uyku modunda kalacaktır. Uygulamamızın kodları diğer uygulamalara göre daha karışık ve yaklaşık 250 satırdan oluşmaktadır.
Uygulamanın kodları aşağıdaki gibidir.

```
#include <msp430.h>                     // MSP430 başlık dosyası
#include "LCD.h"                        // LCD başlık dosyası
#include "DS1302.h"                     // DS1302 başlık dosyası
// Butonlara ait pin tanımları
#define BUTTON_PORT_IN          P1IN
#define BUTTON_PORT_OUT         P1OUT
#define BUTTON_PORT_DIR         P1DIR
#define BUTTON_PORT_SEL         P1SEL
#define BUTTON_PORT_SEL2        P1SEL2
#define BUTTON_PORT_REN         P1REN
#define BUTTON_PORT_IE          P1IE
#define BUTTON_PORT_IES         P1IES
#define BUTTON_PORT_IFG         P1IFG
#define MENU_ARTTIR   BIT0
#define MENU_AZALT    BIT1
#define ARTTIR        BIT2
#define AZALT         BIT3
```

```
// LCD ile ilgili makrolar
#define Kursor_Ac LCD_Komut_Yaz(0x0e)
#define Kursor_Kapa LCD_Komut_Yaz(0x0c)
// Fonksiyon prototipleri
void Gun_Yazdır(unsigned char);
void Sayi_Yazdir(unsigned char);
void Saat_Tarih_Goster(void);
struct _DS1302 DS1302;                          // Saat/tarih bilgilerini tutan yapı
unsigned char bGecikmeSayac=0,bMenu=0; // Kullanılan değişkenler
void main(void) {
   WDTCTL  = WDTPW | WDTHOLD;           // Watchdog timeri durdur.
   BCSCTL1 = CALBC1_1MHZ;               // Dahili osilatörü 1MHz'e ayarla.
   DCOCTL = CALDCO_1MHZ;                // Dahili osilatörü 1MHz'e ayarla.
   DS1302_Ayarla();                     // DS1302 ayarları
   LCD_Ayarla();                        // LCD ayarları
   // Buton ayarları
   BUTTON_PORT_DIR  &= ~(MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
   BUTTON_PORT_REN  |=  (MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
   BUTTON_PORT_OUT  |=  (MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
   BUTTON_PORT_SEL    &= ~(MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
   BUTTON_PORT_SEL2 &= ~(MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
   BUTTON_PORT_IE   |=  (MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
   BUTTON_PORT_IE   |=  (MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
   BUTTON_PORT_IFG  &= ~(MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
   ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
   TA0CCTL0 = CCIE;                     // Timer0 CCR0 ayarları
   TA0CCR0  = 50000;                    // Timer0 kesme periyodu ~50ms
   TA0CTL   = TASSEL_2 + MC_2;          // Timer0 ayarları
   LCD_Temizle();                       // LCD ekranı temizle
   Saat_Tarih_Goster();                 // Saat ve tarih bilgisini göster
   while(1){                            // Sonsuz döngü
   _BIS_SR(GIE + LPM0_bits);            // Kesmeleri aç uykuya gir
   // Menü arttırma butonu kontrol ediliyor
   if(!(BUTTON_PORT_IN & MENU_ARTTIR)){
   _delay_cycles(1000);
   while(!(BUTTON_PORT_IN & MENU_ARTTIR));
   if(++bMenu>7) bMenu=0;}
   // Manü azaltma butonu kontrol ediliyor
   if(!(BUTTON_PORT_IN & MENU_AZALT)){
   _delay_cycles(1000);
   while(!(BUTTON_PORT_IN & MENU_AZALT));
   if(bMenu>1) bMenu--;}
   // Seçilen menüye göre LCD ekran güncellenir
   switch(bMenu){
   // Ay günü ayarının yapıldığı kısım
   case 1: LCD_Git_XY(1,9); Kursor_Ac;
   if(!(BUTTON_PORT_IN & ARTTIR)){
   _delay_cycles(1000);
   while(!(BUTTON_PORT_IN & ARTTIR));
   if(++DS1302.AyinGunu>31) DS1302.AyinGunu=1;
   Sayi_Yazdir(DS1302.AyinGunu);}
   if(!(BUTTON_PORT_IN & AZALT)){
```

```
_delay_cycles(1000);
while(!(BUTTON_PORT_IN & AZALT));
if(--DS1302.AyinGunu==0) DS1302.AyinGunu=31;
Sayi_Yazdir(DS1302.AyinGunu);}
LCD_Git_XY(1,9); break;
// Ay ayarının yapıldığı kısım
case 2: LCD_Git_XY(1,12); Kursor_Ac;
if(!(BUTTON_PORT_IN & ARTTIR)){
_delay_cycles(1000);
while(!(BUTTON_PORT_IN & ARTTIR));
if(++DS1302.Ay>12) DS1302.Ay=1;
Sayi_Yazdir(DS1302.Ay);}
if(!(BUTTON_PORT_IN & AZALT)){
_delay_cycles(1000);
while(!(BUTTON_PORT_IN & AZALT));
if(--DS1302.Ay==0) DS1302.Ay=12;
Sayi_Yazdir(DS1302.Ay);}
LCD_Git_XY(1,12); break;
// Sene ayarının yapıldığı kısım
case 3: LCD_Git_XY(1,15); Kursor_Ac;
if(!(BUTTON_PORT_IN & ARTTIR)){
_delay_cycles(1000);
while(!(BUTTON_PORT_IN & ARTTIR));
if(++DS1302.Sene>99) DS1302.Sene=0;
Sayi_Yazdir(DS1302.Sene);}
if(!(BUTTON_PORT_IN & AZALT)){
_delay_cycles(1000);
while(!(BUTTON_PORT_IN & AZALT));
if(--DS1302.Sene>99) DS1302.Sene=99;
Sayi_Yazdir(DS1302.Sene);}
LCD_Git_XY(1,15); break;
// Haftanın günü ayarının yapıldığı kısım
case 4: LCD_Git_XY(2,1); Kursor_Ac;
if(!(BUTTON_PORT_IN & ARTTIR)){
_delay_cycles(1000);
while(!(BUTTON_PORT_IN & ARTTIR));
if(++DS1302.HaftaninGunu>7) DS1302.HaftaninGunu=1;
Gun_Yazdır(DS1302.HaftaninGunu);}
if(!(BUTTON_PORT_IN & AZALT)){
_delay_cycles(1000);
while(!(BUTTON_PORT_IN & AZALT));
if(--DS1302.HaftaninGunu==0) DS1302.HaftaninGunu=7;
Gun_Yazdır(DS1302.HaftaninGunu);}
LCD_Git_XY(2,1); break;
// Saat ayarının yapıldığı kısım
case 5: LCD_Git_XY(2,9); Kursor_Ac;
if(!(BUTTON_PORT_IN & ARTTIR)){
_delay_cycles(1000);
while(!(BUTTON_PORT_IN & ARTTIR));
if(++DS1302.Saat>23)DS1302.Saat=0;
Sayi_Yazdir(DS1302.Saat);}
if(!(BUTTON_PORT_IN & AZALT)){
```

```
    _delay_cycles(1000);
    while(!(BUTTON_PORT_IN & AZALT));
    if(--DS1302.Saat>23)DS1302.Saat=23;
    Sayi_Yazdir(DS1302.Saat);}
    LCD_Git_XY(2,9); break;
    // Dakika ayarının yapıldığı kısım
    case 6: LCD_Git_XY(2,12); Kursor_Ac;
    if(!(BUTTON_PORT_IN & ARTTIR)){
    _delay_cycles(1000);
    while(!(BUTTON_PORT_IN & ARTTIR));
    if(++DS1302.Dakika>59)DS1302.Dakika=0;
    Sayi_Yazdir(DS1302.Dakika);}
    if(!(BUTTON_PORT_IN & AZALT)){
    _delay_cycles(1000);
    while(!(BUTTON_PORT_IN & AZALT));
    if(--DS1302.Dakika>59)DS1302.Dakika=59;
    Sayi_Yazdir(DS1302.Dakika);}
    LCD_Git_XY(2,12); break;
    // Ayarların tamamlandığı DS1302'nin güncellendiği kısım
    case 7:
    DS1302_Saat_Tarih_Yaz(&DS1302);
    Kursor_Kapa; LCD_Temizle();
    LCD_Git_XY(1,2); LCD_Yazi_Yaz("SAAT ve TARiH");
    LCD_Git_XY(2,4); LCD_Yazi_Yaz("AYARLANDI");
    bGecikmeSayac=0;
    _BIS_SR(GIE + LPM0_bits);
    Saat_Tarih_Goster(); bMenu=0;
    break;
    // Normal çalışmada çalışan kısım
    default: Kursor_Kapa;
    Saat_Tarih_Goster(); break;
    }
    }     // Sonsuz döngü
}
// Girilen 0-99 arası sayıyı LCD ekranda yazdıran fonksiyon
void Sayi_Yazdir(unsigned char Sayi){
        if(Sayi<100){
        LCD_Karakter_Yaz(Sayi/10+0x30);
        LCD_Karakter_Yaz(Sayi%10+0x30);
}}
// Saat ve tarih bilgisini DS1302'den okuyup LCD ekrana yazdıran fonksiyon
void Saat_Tarih_Goster(void){
    DS1302_Saat_Tarih_Oku(&DS1302);              // DS1302'den saat/tarih bilgisini oku
    LCD_Git_XY(1,1);                             // Kursörü 1.satır 1.sütuna götür
    LCD_Yazi_Yaz("Tarih:  ");                    // LCD ekrana Tarih: yazdır.
    LCD_Karakter_Yaz(DS1302.AyinGunu/10+0x30);  // Ay günü onlar basamağını yazdır
    LCD_Karakter_Yaz(DS1302.AyinGunu%10+0x30);// Ay günü birler basamağını yazdır.
    LCD_Karakter_Yaz('-');                       // - yazdır.
    LCD_Karakter_Yaz(DS1302.Ay/10+0x30);         // Ayın onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Ay%10+0x30);         // Ayın birler basamağını yazdır.
    LCD_Karakter_Yaz('-');                       // - yazdır.
    LCD_Karakter_Yaz(DS1302.Sene/10+0x30);       // Senenin onlar basamağını yazdır.
```

```
    LCD_Karakter_Yaz(DS1302.Sene%10+0x30);      // Senenin birler basamağını yazdır.
    LCD_Git_XY(2,1);                            // Kursörü 2.satır 1.sütuna götür.
    Gun_Yazdır(DS1302.HaftaninGunu);            // Haftanın gününü yazdır.
    LCD_Karakter_Yaz(' ');                      // Boşluk karakteri yazdır.
    LCD_Karakter_Yaz(DS1302.Saat/10+0x30);      // Saatin onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Saat%10+0x30);      // Saatin birler basamağını yazdır.
    LCD_Karakter_Yaz('-');                      // - yazdır.
    LCD_Karakter_Yaz(DS1302.Dakika/10+0x30);    // Dakikanın onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Dakika%10+0x30);    // Dakikanın birler basamağını yazdır.
    LCD_Karakter_Yaz('-');                      // - yazdır.
    LCD_Karakter_Yaz(DS1302.Saniye/10+0x30);    // Saniyenin onlar basamağını yazdır.
    LCD_Karakter_Yaz(DS1302.Saniye%10+0x30);    // Saniyenin birler basamağını yazdır.
}
// Girilen 1-7 arası değere göre haftanın günlerini yazdıran fonksiyon
void Gun_Yazdır(unsigned char Gun){
        switch(Gun){
        case 1:LCD_Yazi_Yaz("PZRTESi");break;
        case 2:LCD_Yazi_Yaz("SALI   ");break;
        case 3:LCD_Yazi_Yaz("CRSAMBA");break;
        case 4:LCD_Yazi_Yaz("PRSEMBE");break;
        case 5:LCD_Yazi_Yaz("CUMA   ");break;
        case 6:LCD_Yazi_Yaz("CMRTESi");break;
        case 7:LCD_Yazi_Yaz("PAZAR  ");break;
        default:break;
        }}
// Port 1 kesme vektörü, butonlara basılırsa işlemci uykudan uyandırılır.
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
  BUTTON_PORT_IFG &= ~(MENU_ARTTIR + MENU_AZALT + ARTTIR + AZALT);
  __bic_SR_register(GIE);
  __bic_SR_register_on_exit(CPUOFF);
}
// Timer_A0 Zamanlayıcısı CCR0 kesme vektörü
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
        if(++bGecikmeSayac>=20){                // ~1sn oldu mu?
        bGecikmeSayac=0;                        // Sayacı sıfırla
        __bic_SR_register_on_exit(CPUOFF);      // İşlemciyi uykudan uyandır.
        }
        TA0CCR0 += 50000;                       // Zamanlayıcıyı yeniden kur.
}
```

Uygulamada Port1'in 0,1,2,3 bitlerine bağlı sırasıyla Menü Arttır, Menü Azalat, Arttır, Azalt olmak üzere 4 adet ayar butonu vardır. Butonlar basıldıktan sonra bırakılana kadar işlem yapmazlar.
Başlangıçta uygulama normal olarak saat ve takvim bilgilerini ekranda gösterir. Normal çalışmada menü arttır butonuna basılırsa ayar moduna geçilir ve imleç ayın gün değeri altında görülür. Ayar menüsünde sıraısyla menü arttır ve menü azalt butonları ile değerler arasında gidip gelebilirsiniz. Ayarlamak istediğiniz değeri, arttır ve azalt butonları ile ayrlayabilirsiniz. İmleç dakika değerinin altına geldiğinde tüm ayarlar yapıldıktan sonra

tekrar menü arttır tuşuna basılırsa ayarlama işlemi tamamlanır ve LCD ekranda saat ve tarih ayarlandı yazısı görülür. Sonrasında uygulamamız normal çalışmasına döner. Bu şekilde saat ve takvim uygulamasını istediğiniz gibi ayarlayıp kullanabilirsiniz.
Uygulama kodları düşük güç tüketimine göre yazılmıştır. Saat ve tarih bilgileri yaklaşık olarak saniyede bir güncellenir. Güncelleme olmadığı yada ayar modunda olunmadığı zamanlarda işlemci uyku moduna sokularak güç tasarrufu sağlanır.

Uygulama kodlarını MSP430G2553 denetleyicimize yüklemeden önce geliştirme kartı ayarlarının yapılması gerekir. Uygulamayı çalıştırabilmek için sw24, sw2 ve sw4 anahtarlarından 1,2,3,4 numaralı anahtarlar on konumuna getirilmeli ve jp1 atlaması gnd konumuna getirilmelidir. Ayrıca sorun çıkmaması açısından diğer kullanılmayan birimlerin anahtar ve atlamaları off konumuna getirilmeli yada boşta bırakılmalıdır. Geliştirme kartı ayarlarını yaptıktan sonra kodlarımızı derleyip kartımıza yükleyebiliriz.

Şekil 9: Uygulama 9.2 Çalışma Görüntüleri

Şekil 9'da uygulamaya ait çalışma görüntüleri görülmektedir. Başlangıçta uygulama normal olarak çalışıp saat ve takvim bilgilerini LCD ekranda görüntüler. Menü arttır tuşuna basılmasıyla ayar moduna geçilir. Ayar modunda gerekli işlemler yapıldıktan sonra menü arttır butonuna basılarak imleç dakika değerinin altına getirilir. Sonrasında son olarak menü arttır butonuna basılarak ayarlama işlemi tamamlanır. Ayarlama işleminin tamamlamasından sonra ekranda saat ve tarih ayarlandı yazısı görülür. Sonrasında uygulama normal çalışmasına devam eder.

Geliştirme kartımız üzerinde DS1302 entegresi için pil bulunduğundan kartımızın enerjisi kesilse dahi saat ve tarih bilgileri kaybolmaz.

İstenirse uygulamanın devre şeması çıkarılarak harici olarak kurulabilir. Böylece küçük ve pratik bir masa saati yada takvim yapabilirsiniz.

Bu modül kapsamında DS1302 entegresi incelenip MSP430 dentleyiciler ile kullanımına ilişkin örnek uygulamalar yapılmıştır. Bu uyugulamalardan faydalanılıp çeşitli uygulamalar geliştirilebilir.