Kaan KANDEMİR
Elektronik ve Haberleşme Mühendisi

07 Aralık 2013

# Stm32F4 Discovery RTC modülü

**4 Yorum**

Merhaba arkadaşlar bu yazımda Stm32F4 Discovery kartımızın RTC birimini kullanıcaz.St application team hazırlamış olduğu kaynağa göre işlemlerimizi yapıcaz. Ve bir kaç önemli noktaya değinerek yazıma devam edeceğim. Öncelikle RTC nedir ? nasıl çalışır sorularının cevabını etepic.com da ki RTC konusu ile alakalı anlatımı sizlerle paylaşıcam. Hocamız gayet güzel anlaşılır bir şekilde anlatmış.

RTC Nedir ?

"Hepinizin bildiği gibi saat yapma merakı pek çok kimsede bulunmakta. Saat yapmanın yolu ise bir RTC kullanmaktan geçiyor.
RTC denilen şey öyle enteresan bir malzemeki özellikle saat bilgilerinin RTC hafızasında tutulma şekli nedeni ile pek çok kimsenin aklı karışıyor ve genelde kestimeden gidip kullanılmaya çalışılıyor. Bunun ne anlama geldiğini yazının ileriki bölümlerinde açıklayacağım.

RTC dediğimiz malzemeler üzerlerinde bulunan kristal osilatör ile kendi saat pulslarını üretip zamanı tutan entegrelerdir.
Genelde çalıştıkları kristaller 32769Hz dir. Bunu bölerek 1Hz lik saat pulslarını elde etmektedirler. En çok kullanılan modeller olarak DS1302 ve DS1307 malzemelerini sayabiliriz. Bu malzemeler genelde bir işlemciye bağlanarak kullanılırlar. Bağlantı şekline bakılırsa DS1302 3 pinden seri haberleşme protokolünü , DS1307 ise I2C protokolünü kullanmaktadır.

Her ikisinin ortak yönü ise zaman bilgilerini tutma yöntemidir. BCD yani Binary Coded Desimal formatında tutulur bu bilgiler.
Bu yüzden elinizdeki zaman bilgisini doğrudan entegreye ayar bilgisi olarak yazamaz ve aynı şekilde entegreden okuduğunuzuda direk zaman bilgisi olarak kullanamazsınız. Yukarıda bir laf eetim kestirmeden kullanma terimi burada ortaya çıkıyor. Her ne kadar doğrudan kullanamazsınız desemde bir yolu var. Zira entegreden okunan bilgileri HEX formatında ekrana verirseniz doğru saat bilgilerini görürsünüz. Ama bu yanıltıcıdır. Zira aslında desimal bir bilgiyi hex olarak ekrana veriyor ama onu desimal kabul ediyorsunuz. Her neyse biraz karışık gibi gözüksede format şeklini anlayınca dediklerimi daha kolay anlayacaksınız.
Her iki entegreninde register tablosuna bir bakarsanız hemen hemen aynı formatı görürsünüz. Örneğin Saniye hafızasına bir bakalım isterseniz.
Bit7 ……..Bit0
CH |….10Sec….. | …….SEC……..|
Bu şekil şunu ifade etmektedir. Bit CH clock Halt bitidir. Fabrikasyon olarak 1 (HIGH) oalrak ayarlanmıştır. Yada siz o biti ne zaman 1 yaparsanız saat çalışmasını durdurur. Sonraki gelen 3 bit (bit6-Bit5-Bit4) saniye değerinin onlar basamağını tutar.
Sonraki gelen 4 bit ise saniye değerinin birler basamağını tutar.

Şimdi DS1302 den okuduğumuz saniye değeri şöyle olsun SN=%0010 0110 Bu sayının direk desimal karşılığı 38 sayısıdır.
Ama saniye karşılığı şöyle hesaplanmalıdır. %0010 sayısı desimal karşılığı 2 sayısıdır. Onlar basmağı olduğuna göre bu 20 olacaktır.
Diğer parça ise %0110=6 sayısına eşittir. O halde okunan saniye değeri aslında 26 değerini göstermektedir.
Şimdi bunu çevirmeden nasıl kestirmeden gösterebildiğimize bir bakalım isterseniz. Sayı gerçekte 38 sayısına eşit idi.
38 sayısının hex karşılığı ise 26 dır. O halde ben okunan sayıyı ekrana yazrken $SN şeklinde yazdırırsam ekranda 26 sayısını göreceğim demektir. İşte kestirme budur. Ama unutmayın bu işi ancak LCD ekranda yapabilirsiniz. 7 segment displayde göstermek için ne yapıyoruz? Önce onlar basamağını alıp onun 7 segment karşılığını buluyoruz. Bunu yaparkende Onlar=SN DIG 1 şeklinde bir komut kullanıyoruz. Bu durumda Onlar değeri örneğimize göre 3 olur 2 olmaz. Şayet komutu şöyle kullanabilse idik olabilirdi Onlar=$SN DIG 1 . BU durumda Onlar=2 olacak idi. Ama ne yazıkki böyle bir komut şekli yok.

Dakika değeride benzer şekilde hafızada tutulmaktadır.
Bit7 | —10Min— | ––Min— — |
Yani ilk 4 bit dakika değerinin birler basamağını veriyor, sonra takip eden 3 bit onlar basamağını veriyor ve sonda yer alan 7. bit ise daima sıfır olarak okunuyor.
Aynı şekilde okuduğumuz dakika değeri diyelimki Dak=%01010011 şeklinde olsun. Şimdi bu sayının direk desimal karşılığı, 83 sayısıdır. Ama gerçek dakika karşılığı ise %0101=5 sayısına eşit ama onlar basmağı olduğu için 50 olacaktır.
%0011 sayısı ise 3 desimal değerine eşittir. O halde sayının dakika karşılığı 53 sayısı olacaktır. Benzer şekilde 83 sayısının desimal karşılığı ise 53 sayısıdır.

Sonuç olarak RTC ile doğru şekilde çalışabilmek için daima entegreye yazacağımız saat ayar değerini BCD formatına çevirip yazmalı, entegreden okunan zaman bilgilerinide BCD formatından desimal formata çevirerek kullanmalıyız.
Bunu yapmazsanız epeyce sorun yaşarsınız.
Şimdi çevirme işlemlerini nasıl yapacağız biraz onlara bakalım. Örneğimizi Saniye birimi üzerinden vereceğim ama diğerleri de aynı yöntemle çevrilebilecektir.
Şimdi bir saat ayarı yaptığımız düşünelim ve ayarladığımız saniye değeri de 45 olsun. Bu saniye değerini DS1302 (veya DS1307) ye yazacağımızı farz edelim. BCD formatına bakılırsa 4 sayısı saniyenin onlar basamağını teşkil edecek, 5 değeride birler basamağını teşkil edecektir. Bunu BCD formatında yazarsak %0100 0101 şeklinde bir sayı elde etmemiz gerekecektir.
İşte adım adım bunu nasıl yaptığımızı aşağıda görebilirsiniz.
İşlem 1. Temp=SN DIG 1 , buradan Temp=%00000100 olacaktır. Onlar basamağını ayırarak almış olduk.
İşlem 2. Temp=Temp<<4, buradan Temp=%01000000 şeklini alacaktır. Onlar basamağını olması gereken yere 4 bit kaydırdık.
İşlem 3. RTC_SN=SN DIG 0 buradan RTC_SN=%00000101 , bir ler basamağını aldık.
İşlem 4. RTC_SN=RTC_SN+Temp buradan RTC_SN=%01000101 şeklini alacaktır. Bu istediğimiz formattır. Dolayısıyla bu sayıyı direk entegreye yazabiliriz artık.

Şimdi Okuma işleminden sonra BCD formatından Desimal formata nasıl çevirme yapacağız onu görelim.
Aynı örneği kullanırsak daha iyi anlarsınız her halde. Gerçek saniye değerimiz 45 iken bunu RTC ye %01000101 yani desimal olarak 69 değeri ile yazmış idik. O halde bu sefer RTC den okunan 69 değerini gerçek saniye değeri olan 45'e nasıl çevireceğimizi adımlar halinde görelim.

İşlem 1. Temp=RTC_SN & %01110000 Temp=%01000000 oldu.
İşlem 2. Temp=Temp>>4 Temp=%00000100 oldu.

İŞlem 3. SN=RTC_SN & %00001111 SN=%00000101 oldu
İşlem 4. SN=(Temp*10)+SN SN=45 oldu

Görüldüğü gibi son derece kolay bir işlem. O halde yapmamız gerekenleri yeniden tekrarlayalım.
RTC de saat ayarı yapıldıktan sonra bu ayarları RTC ye yazmadan önce onları BCD formatına çevirmemiz gerekiyor.
RTC den zaman okuması yaptıktan sonra okunanları kullanmadan önce onları BCD formatından Desimal formata çevirmemiz gerekiyor.

Saat registerine bakacak olursak kayıt şeklinin aşağıdaki gibi olduğunu görürüz.
Bit7 | Bit6 | Bit5 | Bit4 | Bit3 – Bit0 |
12/24 | 0 |10/A-P| Hr | Hr |
Bit7 saatin 12 saat mi yoksa 24 saat formatına göremi tutulduğunu belirlemektedir. Bu bit 1 ise format 12 saat esasına göredir. Bu durumda 5 nolu bit saatin AM mi (sabah) yoksa PM mi (öğleden Sonra) olduğuna karar verir. BU bit 1 ise PM sıfır ise AM dir.
Şayet Bit7 sıfır ise saat 24 saat formatına göre tutuluyor demektir ve 5 nolu bit bu sefer saat değerinin onlar basamağını verir. Onlar basmağı yalnıza 1 veya 2 olabilir. O nedenle bit5 High ise saat 20 ve üzeri sıfır ise 20 nin altında demek olur.
Şimdi her iki formatta örnek yaparak konuyu inceleyelim.
Saat formatımız 12 saat şeklinde olsun. Saat öğleden sonra 3:45 iken okumamız gereken saat değerine bir bakalım isterseniz.
Okunacak değeri %1010 0011 şeklinde olacaktır. Ne anlama geliyor bakalım.
Bit7=1 olduğuna göre formatımız 12 saat üzerinden çalışıyor demektir.
Bit5=1 olduğundan zaman öğleden sonrasını (PM) gösteriyor demektir.
Bit3-0= %0011 = 3 gösterdiğine göre saat=3 olduğunu gösteriyor demektir.
Tabiiki okunan saniye değeride bize 45 değerini verecektir.

Şimdi aynı saati 24 saat formatında görelim.
Okunacak saat değeri %0001 0101 şeklinde olmalıdır. Ne anlama gelior bakalım.
Bit7=0 olduğundan saat formatı 24 saat esasına göredir.
Bit5=0 olduğuna göre onlar basamağı onlar hanesi=2 değildir.
Bit4=1 olduğundan saat onlar basamağı 1 dir.
Bit3-0= arası ise bize 5 değerini verdiği için saat değeri 15 olarak bulunmuş olacaktır.
Dikkat ederseniz 24 saat formatında Bit7-Bit4 arası bitlerin yalnızca ilk 2 tanesi yani %00xx x ile işaretlenmiş olanlar önem arzediyor. Saat 10 dan küçük ise Bu bitler %0000 şeklinde okunacak, saat 9 dan büyük ve 20 den küçük ise Bu bitler %0001 şeklinde yani bir olarak okunacak ve 19 dan büyük ise %0010 şeklinde yani 20 olarak okunacaktır.
Diğer lerini izah etmeye gerek görmüyorum artık. Onlarda da benzer şekilde kullanılmaktadır.

Son olarak bir kaç tavsiyede bulunmak isterim.
Birincisi her iki entegrede de harici pil bağlanarak saatin elektrik kesintilerden etkilenmemesi sağlanmıştır.
DS 1302 de şebeke voltajı giriş bacağı 1 nolu pindir. Pil voltajı ise 8 nolu pinden girilir. Pil olarak 3V luk Lityum pillerin kullanılmasını tavsiye ederim. RTC hangi bacakta daya yüksek voltaj var ise beslemeyi oradan almaya çalışır. Şebeke beslemesi se 5V dolayında olur ise sistem sorunsuz çalışır.
DS1307 de Şebeke voltaj girişi 8 nolu pindendir. Pil girişi ise 3 nolu pinden yapılır.

Mümkün olduğunca DS1307 kullanmaya çalışın. Önemli özellikleri var. Her şeyden önce bir SQW/OUT bacağı varki pek çok işe yarayabilir. Bazılarını açıklayayım.
1. Saatlerin genelde ileri gittiğinden yada geri kaldığından şikayet edilmektedir. Bu sorun genelde kristal bacaklarına bağlanacak 5-30 pf kondansatörle giderilebilmektedir. Ancak yaptığınız işlemin doğruluğunu ancak bu çıkış bacağını kullanarak anlayabilirsiniz. Bu bacak size osilatörden bölünerek elde edilen bazı frekans değerleri verebilir.

Mesela ,
– 1 Hz
– 4096 Hz
– 8192 Hz
– 32768 hz
Şöyle bir bakarsanız bu çıkışları ne gibi işlerde kullanabileceğiniz kolaylıkla görebilirsiniz.
Mesela çıkışı 32768 Hz olarak ayarlarsamki bu direk kristal frekansıdır, osilatörümün doğru çalışıp çalışmadığını bu sinyalin frekansını ölçerek anlayabilirim. Şayet değer 32768 den küçük ise krital bacakalrına 5-33 pf arası bir kondansatör bağlayarak doğru değeri yakalamaya çalışırım.

1Hz çıkışı ise saat projelerinde orta ledlerin yakılıp söndürülmesinde kullanabilirim. Bu çıkış ile direk ledleri sürerseniz her sn de bir ledleriniz yanıp sönecektir.
Bu çıkışın nasıl ayarlanacağı konusuna da biraz değinelim isterseniz.
DS1307 de toplam 8 adet register vardır. Bunlar sırası ile SN – Dak – Saat – Haftnın Günü – Ayın günü – Ay – Yıl – CTRL
En sonda yer alan Kontrol (CTRL) registeri nin bitlerine bakacak olursak şu bilgileri tuttuğunu görürüz.
Bit7=Out biti bu bit 1 ise SQW/OUT pini 1 dir. Bu bit=0 ise SQW/Out pini sıfırdır. Bir yerde frekans çıkış pininin lojşk seviyesini ayarlamaktadır.
Bit6-Bit5=0 olup bir fonksiyonları yoktur.
Bit4=SQWE bitidir. Bu bit=0 ise Osilatör çıkışı iptal , bu bit 1 ise osilatör çıkış aktiftir.
Bit3-2= 0 olup bir fonksiyonları yoktur.
Bit1-Bit2= RS1-RS0 bitleridir ve Çıkış bacağından alınacak frekansı bu bitler belirler.
%00 ise 1 Hz
%01 ise 4096 Hz
%10 ise 8192 Hz
%11 ise 32768 Hz dir.

Diyelimki çıkış bacağından 32768 hzlik bir çıkış almak istiyorsunuz. O halde CTRL=%10010011 şeklinde bir değeri kontrol registerine yazmak gerekir.
Diyelimki 1 Hz lik çıkışalacaksınız CTRL=%10010000 şeklinde bir değer vermeniz yeterlidir.

Hepsi bu kadar.
Ete “

**Alıntıdır.

AN3371
Application note göre basit bir kaç temeli görelim.

STM32 serisinde ki RTC birimi BCD timere bağlı olarak çalışmakta ve bu RTC modülü bizlere calendar, alarm,periodic wakeup unit,digital calibration,synchronization, time stamp,and advanced tamper detection özellikleri sağlıyor.

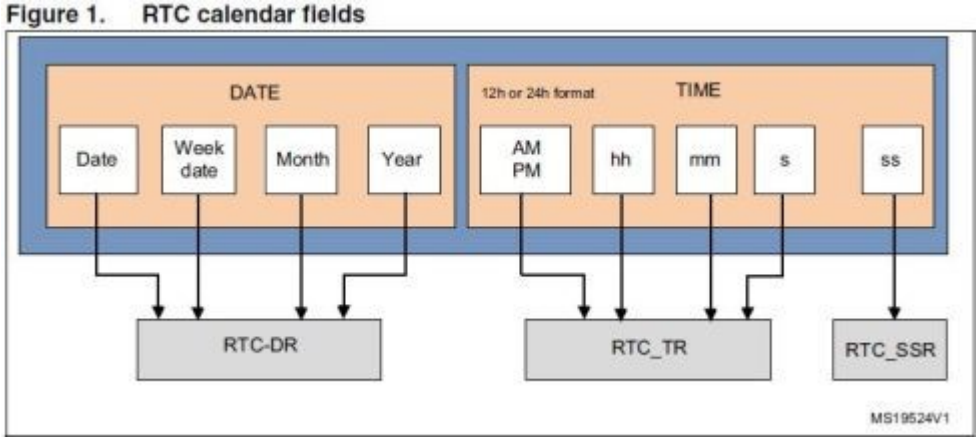
RTC Calender kısımı ile yapabileceklerimiz ;

● Calendar with:
– sub-seconds (not programmable)
– seconds
– minutes
– hours in 12-hour or 24-hour format
– day of the week (day)
– day of the month (date)

– month
– year
● Calendar in binary-coded decimal (BCD) format
● Automatic management of 28-, 29- (leap year), 30-, and 31-day months
● Daylight saving time adjustment programmable by software

Diyagrama dökülmüş hali

**Figure 1. RTC calendar fields**




(https://kaankandemir.files.wordpress.com/2013/12/capture.jpg)

32 bitlik BCD yazlımsal sayıcıyı ayarlayabilirsiniz ve bunu yazılımsal olarak otomatikmen gün ay saat olarak kendisi çevirecektir.BCD formattaki bu değişkeninizi LCD ye basabilir görüntüleyebilirsiniz.. Stm32 de ki rtc modüllerini yazılımsal convert yaparken ayrıca müdahale etmeniz gerekmemekte çünkü bu işlemler hardware tarafından otomatik olarak yapılmakta.

RTC Calender İnit ayarları

**Initializing the calendar**

*Table 2* describes the steps required to correctly configure the calendar time and date.

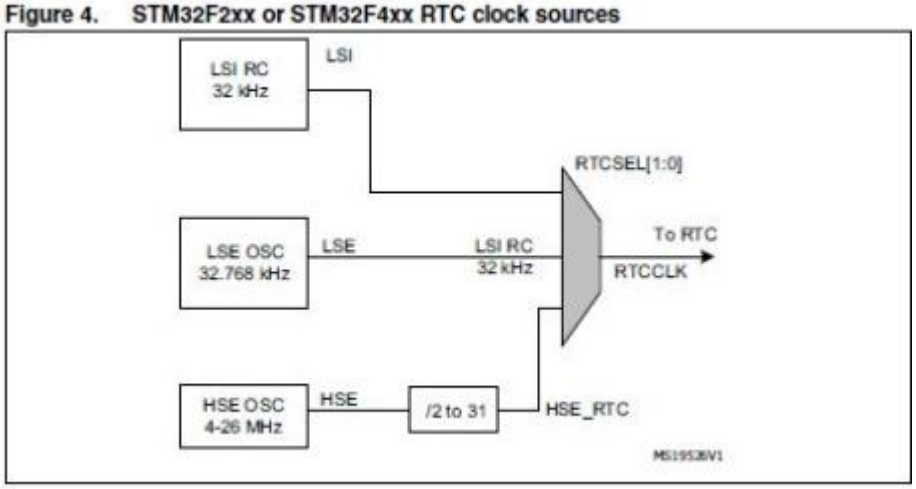
**Table 2. Steps to initialize the calendar**

| Step | What to do | How to do it | Comments |
|---|---|---|---|
| 1 | Disable the RTC registers Write protection | Write "0xCA" and then "0x53" into the RTC_WPR register | RTC registers can be modified |
| 2 | Enter Initialization mode | Set INIT bit to '1' in RTC_ISR register | The calendar counter is stopped to allow update |
| 3 | Wait for the confirmation of Initialization mode (clock synchronization) | Poll INITF bit of in RTC_ISR until it is set | It takes approximately 2 RTCCLK clock cycles for medium density devices |
| 4 | Program the prescalers register if needed | RTC_PRER register: Write first the synchronous value and then write the asynchronous | By default, the RTC_PRER prescalers register is initialized to provide 1Hz to the Calendar unit when RTCCLK = 32768Hz |
| 5 | Load time and date values in the shadow registers | Set RTC_TR and RTC_DR registers | |
| 6 | Configure the time format (12h or 24h) | Set FMT bit in RTC_CR register | FMT = 0: 24 hour/day format<br>FMT = 1: AM/PM hour format |
| 7 | Exit Initialization mode | Clear the INIT bit in RTC_ISR register | The current calendar counter is automatically loaded and the counting restarts after 4 RTCCLK clock cycles |
| 8 | Enable the RTC Registers Write Protection | Write "0xFF" into the RTC_WPR register | RTC Registers can no longer be modified |

(https://kaankandemir.files.wordpress.com/2013/12/capture2.jpg)

RTC clock configuration

**Figure 4. STM32F2xx or STM32F4xx RTC clock sources**

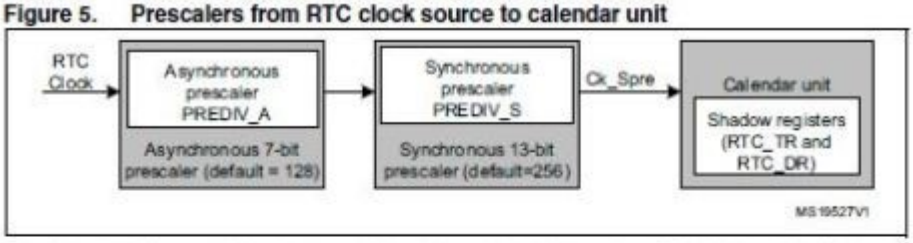



(https://kaankandemir.files.wordpress.com/2013/12/capture3.jpg)

How to adjust the RTC calendar clock
The RTC features several prescalers that allow delivering a 1 Hz clock to calendar unit, regardless of the clock source.

RTC Calender saat ayarlarını yapalım

**Figure 5. Prescalers from RTC clock source to calendar unit**




*The length of the synchronous prescaler depends on the product. For this section, it is represented on 13 bits.*

The formula to calculate ck_spre is:

$$ck\_spre = \frac{RTCCLK}{(PREDIV\_A + 1) \times (PREDIV\_S + 1)}$$

where:

- RTCCLK can be any clock source: HSE_RTC, LSE or LSI
- PREDIV_A can be 1,2,3,..., or 127
- PREDIV_S can be 0,1,2,..., or 8191

*Table 3* shows several ways to obtain the calendar clock (ck_spre) = 1 Hz.

**Table 3. Calendar clock equal to 1 Hz with different clock sources**

| RTCCLK Clock source | Prescalers | | ck_spre |
|---|---|---|---|
| | PREDIV_A[6:0] | PREDIV_S[12:0] | |
| **HSE_RTC = 1MHz** | 124 (div125) | 7999 (div8000) | 1 Hz |
| **LSE = 32.768 kHz** | 127 (div128) | 255 (div256) | 1 Hz |
| **LSI = 32 kHz(1)** | 127 (div128) | 249 (div250) | 1 Hz |
| **LSI = 37 kHz(2)** | 124 (div125) | 295 (div296) | 1 Hz |

1. For STM32L1xx, LSI = 37 KHz, but LSI accuracy is not suitable for calendar application.
2. For STM32F2xx and STM32F4xx, LSI = 32 KHz, but LSI accuracy is not suitable for calendar application.

(https://kaankandemir.files.wordpress.com/2013/12/capture4.jpg)

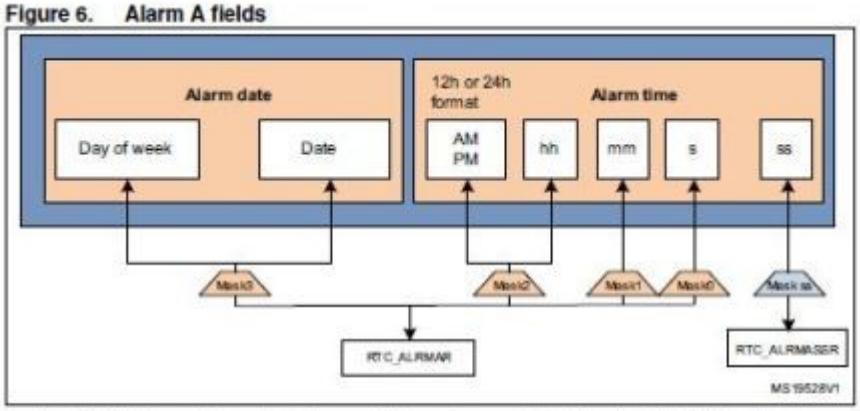
Stm32 de 2 adet alarm bulunmakta ve ikiside aynı yapıya sahip.Alarm değerleri user tarafından girilir ve kurulur. Her alarm kuruldugunda bu işlemler yapılır;

-Alarmların kombinasyonu saniye, dakika,saat ve tarih alanları bağımsız olarak seçilen ya da zengin bir sağlamak için kamufle edilebilir.

-Alarm kurdugumuzda düşük güç modunda çalışılır.

-Alarm kurulduğunda bir alarm kesmesi için bi flag atanmalı ve işimiz bittiğinde bu bayragı temizlemeliyiz.

**Figure 6. Alarm A fields**




1. RTC_ALRMAR is an RTC register. The same fields are also available for the RTC_ALRMBR register.
2. RT_ARMASSR is an RTC register. The same field is also available for the RTC_ALRMBR register.
3. Maskx are bits in the RTC_ALRMAR register that enable/disable the RTC_ALARM fields used for alarm A and calendar comparison. For more details, refer to *Table 5*.
4. Mask ss are bits in the RTC_ALRMASSR register.

An alarm consists of a register with the same length as the RTC time counter. When the RTC time counter reaches the value programmed in the alarm register, a flag is set to indicate that an alarm event occurred.

The STM32 RTC alarm can be configured by hardware to generate different types of alarms. For more details, refer to *Table 5*.

(https://kaankandemir.files.wordpress.com/2013/12/capture5.jpg)

Kurulum ayaları

**Table 4. Steps to configure the alarm**

| Step | What to do | How to do it | Comments |
|---|---|---|---|
| 1 | Disable the RTC registers Write protection | Write "0xCA" and then "0x53" into the RTC_WPR register | RTC registers can be modified |
| 2 | Disable alarm A | Clear ALRAE(1) bit in RTC_CR register. | |
| 3 | Check that the RTC_ALRMAR register can be accessed | Poll ALRAWF(2) bit until it is set in RTC_ISR. | It takes approximately two RTCCLK clock cycles (clock synchronization). |
| 4 | Configure the alarm | Configure RTC_ALRMAR(3) register. | The alarm hour format must be the same(4) as the RTC Calendar in RTC_ALRMAR. |
| 5 | Re-enable alarm A | Set ALRAE(5) bit in RTC_CR register. | |
| 6 | Enable the RTC registers Write protection | Write "0xFF" into the RTC_WPR register | RTC registers can no longer be modified |

1. Respectively ALRBE bit for alarm B.
2. Respectively ALRBWF bit for alarm B.
3. Respectively RTC_ALRMBR register for alarm B.
4. As an example, if the alarm is configured to occur at 3:00:00 PM, the alarm will not occur even if the calendar time is 15:00:00, because the RTC calendar is 24-hour format and the alarm is 12-hour format.
5. Respectively ALRBE bit for alarm B.
6. RTC alarm registers can only be written when the corresponding RTC alarm is disabled or during RTC Initialization mode.

(https://kaankandemir.files.wordpress.com/2013/12/capture6.jpg)

RTC periodic wakeup

## RTC periodic wakeup unit

Like many STMicroelectronics microcontrollers, the STM32 provides several low power modes to reduce the power consumption.

The STM32 features a periodic timebase and wakeup unit that can wake up the system when the STM32 operates in low power modes. This unit is a programmable downcounting auto-reload timer. When this counter reaches zero, a flag and an interrupt (if enabled) are generated.

The wakeup unit has the following features:

- Programmable downcounting auto-reload timer.
- Specific flag and interrupt capable of waking up the device from low power modes.
- Wakeup alternate function output which can be routed to RTC_ALARM output (unique pad for alarm A, alarm B or Wakeup events) with configurable polarity.
- A full set of prescalers to select the desired waiting period.

(https://kaankandemir.files.wordpress.com/2013/12/capture7.jpg)

### Periodic timebase/wakeup configuration for clock configuration 1

*Figure 8* shows the prescaler connection to the timebase/wakeup unit and *Table 8* gives the timebase/wakeup clock resolutions corresponding to configuration 1.

The prescaler depends on the Wakeup clock selection:

- WUCKSEL[2:0] =000: RTCCLK/**16** clock is selected
- WUCKSEL[2:0] =001: RTCCLK/**8** clock is selected
- WUCKSEL[2:0] =010: RTCCLK/**4** clock is selected
- WUCKSEL[2:0] =011: RTCCLK/**2** clock is selected

(https://kaankandemir.files.wordpress.com/2013/12/capture8.jpg)

RTC reference clock detection
The reference clock (at 50 Hz or 60 Hz) should have a higher precision than the 32.768 kHz LSE clock. This is why the RTC provides a reference clock input (RTC_50Hz pin) that can be used to compensate the imprecision of the calendar frequency (1 Hz).
The RTC_50Hz pin should be configured in input floating mode.
This mechanism enables the calendar to be as precise as the reference clock.
The reference clock detection is enabled by setting REFCKON bit of the RTC_CR register.
When the reference clock detection is enabled, PREDIV_A and PREDIV_S must be set to their default values: PREDIV_A = 0x007F and PREVID_S = 0x00FF.
When the reference clock detection is enabled, each 1 Hz clock edge is compared to the nearest reference clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. The update window is 3 ck_calib periods (ck_calib is the output of the coarse calibration block).
If the reference clock halts, the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a detection window centered on the Synchronous Prescaler output clock (ck_spre) edge. The detection window is 7 ck_calib periods.
The reference clock can have a large local deviation (for instance in the range of 500 ppm), but in the long term it must be much more precise than 32 kHz quartz.
The detection system is used only when the reference clock needs to be detected back after a loss. As the detection window is a bit larger than the reference clock period, this detection system brings an uncertainty of 1 ck_ref period (20 ms for a 50 Hz reference clock) because we can have 2 ck_ref edges in the detection window. Then the update window is used, which brings no error as it is smaller than the reference clock period.
We assume that ck_ref is not lost more than once a day. So the total uncertainty per month would be 20 ms *1* 30 = 0.6 s, which is much less than the uncertainty of a typical quartz (1.53 minute per month for 35 ppm quartz).

Bu kısım biraz karışık geldi gözüme o yüzden orjinal halini koydum.

Yazılım içerisinde kullanacağımız fonksiyonları burada açıklamışlar Firmware API

## RTC firmware driver API

This driver provides a set of firmware functions to manage the following functionalities of the RTC peripheral:

- Initialization
- Calendar (Time and Date) configuration
- Alarm (alarm A and alarm B) configuration
- Wakeup timer configuration
- Daylight saving configuration
- Output pin configuration
- Digital calibration configuration
- Synchronization configuration
- Time-stamp configuration
- Tamper configuration
- Backup data register configuration
- RTC Tamper and Time-stamp pin selection and Output type configuration
- Interrupts and flag management

For the STM32F2xx family, the RTC driver stm32f2xx_rtc.c/.h can be found in the directory: STM32F2xx_StdPeriph_Lib_vX.Y.Z\Libraries\STM32F2xx_StdPeriph_Driver.

For the STM32L1xx family, the RTC driver stm32l1xx_rtc.c/.h can be found in the directory: STM32L1xx_StdPeriph_Lib_vX.Y.Z\Libraries\STM32L1xx_StdPeriph_Driver.

For the STM32F4xx family, the RTC driver stm32f4xx_rtc.c/.h can be found in the directory: STM32F4xx_StdPeriph_Lib_vX.Y.Z\Libraries\STM32F4xx_StdPeriph_Driver.

For the STM32F0xx family, the RTC driver stm32f0xx_rtc.c/.h can be found in the directory: STM32F0xx_StdPeriph_Lib_vX.Y.Z\Libraries\STM32F0xx_StdPeriph_Driver.

These four drivers provide a fully compatible API making it easy to move from one product to another.

(https://kaankandemir.files.wordpress.com/2013/12/capture9.jpg)

Sırayla kullanacağımız fonksiyonlarımız ne işe yaradıkları ve kullanım şekilleri açıklanmakta

**Start with the RTC driver**
Before using the RTC features:
● Enable the RTC domain access (see following note)
● Configure the RTC prescaler (Asynchronous and Synchronous) and RTC hour format using the RTC_Init() function.

Note: After a reset, the backup domain (RTC registers, RTC backup data registers and backup SRAM) is protected against any possible unwanted write access. To enable access to the RTC domain and RTC registers:
– Enable the Power Controller (PWR) APB1 interface clock using the RCC_APB1PeriphClockCmd() function.
– Enable the access to the RTC domain using the PWR_BackupAccessCmd() function on STM32F2xx and STM32F4xx devices, or the PWR_RTCAccessCmd() function on STM32L1xx and STM32F0xx devices.
– Select the RTC clock source using the RCC_RTCCLKConfig() function.
– Enable RTC Clock using the RCC_RTCCLKCmd() function.

**Time and date configuration**
To configure the RTC Calendar (Time and Date), use the RTC_SetTime() and RTC_SetDate() functions.
To read the RTC Calendar, use the RTC_GetTime(), RTC_GetDate() and

RTC_GetSubSecond() functions.
To add or subtract one hour to/from the RTC Calendar, use the RTC_DayLightSavingConfig() function.
3.1.2 Alarm configuration
RTC Alarm
To configure the RTC alarm, use the RTC_SetAlarm() function.
To enable the selected RTC alarm, use the RTC_AlarmCmd() function.
To read the RTC alarm, use the RTC_GetAlarm() function.
RTC Alarm Sub-second
To configure the RTC alarm sub-second, use the RTC_AlarmSubSecondConfig() function.
To read the RTC alarm sub-second, use the RTC_GetAlarmSubSecond() function.
3.1.3 RTC wakeup configuration
To configure the RTC Wakeup Clock source, use the RTC_WakeUpClockConfig() function.
To configure the RTC WakeUp Counter, use the RTC_SetWakeUpCounter() function.
To enable the RTC WakeUp, use the RTC_WakeUpCmd() function.
To read the RTC WakeUp Counter register, use the RTC_GetWakeUpCounter() function.

**Outputs configuration**
The RTC has two different outputs:
● AFO_ALARM, used to manage the RTC alarm A, alarm B and WaKeUp signals. To output the selected RTC signal on RTC_AF1 pin, use the RTC_OutputConfig() function.
● AFO_CALIB, used to manage the RTC Clock divided by a 64 (512 Hz) signal and the calendar clock (1 Hz). To output the RTC Clock on the RTC_AF1 pin, use the RTC_CalibOutputCmd() function.
**Digital calibration configuration**
To configure the RTC Coarse calibration value and the corresponding sign, use the RTC_CoarseCalibConfig() function.
To enable the RTC Coarse calibration, use the RTC_CoarseCalibCmd() function.
To configure the RTC smooth calibration value and the calibration period, use the RTC_SmoothCalibConfig() function.
**TimeStamp configuration**
To configure the RTC_AF1 trigger and enable the RTC TimeStamp, use the RTC_TimeStampCmd() function.
To read the RTC TimeStamp Time and Date register, use the RTC_GetTimeStamp() function.
To read the RTC TimeStamp sub-second register, use the RTC_GetTimeStampSubSecond() function.
The TAMPER1 alternate function can be mapped either to RTC_AF1(PC13) or RTC_AF2 (PI8) depending on the value of TAMP1INSEL bit in RTC_TAFCR register. You can use the RTC_TimeStampPinSelection() function to select the corresponding pin.
**Tamper configuration**
To configure the RTC Tamper trigger, use the RTC_TamperConfig() function.
To configure the RTC Tamper filter, use the RTC_TamperFilterConfig() function.
To configure the RTC Tamper sampling frequency, use the RTC_TamperSamplingFreqConfig() function.
To configure the RTC Tamper pins input precharge duration, use the RTC_TamperPinsPrechargeDuration() function.
To enable the precharge of the Tamper pin, use the RTC_TamperPullUpCmd() function.
To enable the TimeStamp on Tamper detection event, use the RTC_TimeStampOnTamperDetectionCmd() function.
To enable the RTC Tamper, use the RTC_TamperCmd() function.

The TIMESTAMP alternate function can be mapped to either RTC_AF1 or RTC_AF2 depending on the value of the TSINSEL bit in the RTC_TAFCR register. You can use the RTC_TamperPinSelection() function to select the corresponding pin.

**Backup data registers configuration**
To write to the RTC backup data registers, use the RTC_WriteBackupRegister() function.
To read the RTC backup data registers, use the RTC_ReadBackupRegister() function.
**Function groups and description**
The STM32 RTC driver can be divided into 14 function groups related to the functions embedded in the RTC peripheral.
- RTC configuration to the default reset state
- RTC initialization and configuration functions
- RTC time and date configuration functions
- RTC alarm configuration functions
- RTC wakeup timer configuration functions
- RTC daylight saving configuration functions
- RTC output pin configuration functions
- RTC digital calibration (coarse and smooth) configuration functions
- RTC time-stamp configuration functions
- RTC Tamper configuration functions
- RTC backup registers configuration functions
- RTC tamper, time-stamp pin selection
- RTC shift control synchronization function
- RTC flags and IT management functions

Application notlarından sonra Stm32f4 için kodumuzu paylaşalım;

```
/* Includes ————————————————————————————*/
#include “stm32f4xx.h”
#include “stm32f4xx_rtc.h”
#include “stm32f4xx_pwr.h”
#include “tft_lcd.h”
#include <stdio.h>
#include “touch.h”

/*Touch extern */
extern unsigned int xxx,yyy;
extern unsigned char flag;

RTC_TimeTypeDef RTC_TimeStructure;
RTC_DateTypeDef RTC_DateStructure;
RTC_AlarmTypeDef RTC_AlarmStructure;

RTC_InitTypeDef RTC_InitStructure;
NVIC_InitTypeDef NVIC_InitStructur;
EXTI_InitTypeDef EXTI_InitStructure;

uint8_t RTC_HandlerFlag;
static int8_t sec, min, hour;
static int8_t asec, amin, ahour;
static int8_t day, month,weekday;
static int16_t year;
```

```
uint8_t strMonth[][12] = {“Jan.”, “Feb.”, “Mar.”, “Apr.”, “May “,
“Jun.”, “Jul.”, “Aug.”, “Sep.”, “Oct.”,
“Nov.”, “Dec.”};

uint8_t strWeekday[][7] = {“Mon.”, “Tues.”, “Wedn.”, “Thurs.”, “Fri. “,
“Satur.”, “Sun.”};

uint32_t AsynchPrediv = 0, SynchPrediv = 0;// void Delay (uint32_t nCount)
// {
// for(; nCount != 0; nCount–);
// }

void RTC_kur(void)
{

/* Enable the PWR clock */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);

/* Allow access to RTC */
PWR_BackupAccessCmd(ENABLE);

// #if defined (RTC_CLOCK_SOURCE_LSI) /* LSI used as RTC source clock*/
/* The RTC Clock may varies due to LSI frequency dispersion */
/* Enable the LSI OSC */
RCC_LSICmd(ENABLE);

/* Wait till LSI is ready */
while(RCC_GetFlagStatus(RCC_FLAG_LSIRDY) == RESET)
{
}

/* Select the RTC Clock Source */
RCC_RTCCLKConfig(RCC_RTCCLKSource_LSI);

/* Enable the RTC Clock */
RCC_RTCCLKCmd(ENABLE);

/* Wait for RTC APB registers synchronisation */
RTC_WaitForSynchro();

/* Configure the RTC data register and RTC prescaler */
RTC_InitStructure.RTC_AsynchPrediv = 0x7F;
RTC_InitStructure.RTC_SynchPrediv = 0xFF;
RTC_InitStructure.RTC_HourFormat = RTC_HourFormat_24;
RTC_Init(&RTC_InitStructure);

/* Set the date: */
RTC_DateStructure.RTC_Year = 13; //Year Set
RTC_DateStructure.RTC_Month = RTC_Month_November ; //Month date
RTC_DateStructure.RTC_Date = 20; // Day count set
RTC_DateStructure.RTC_WeekDay = RTC_Weekday_Wednesday; //Day set
RTC_SetDate(RTC_Format_BCD, &RTC_DateStructure);

/* Set the time to */
```

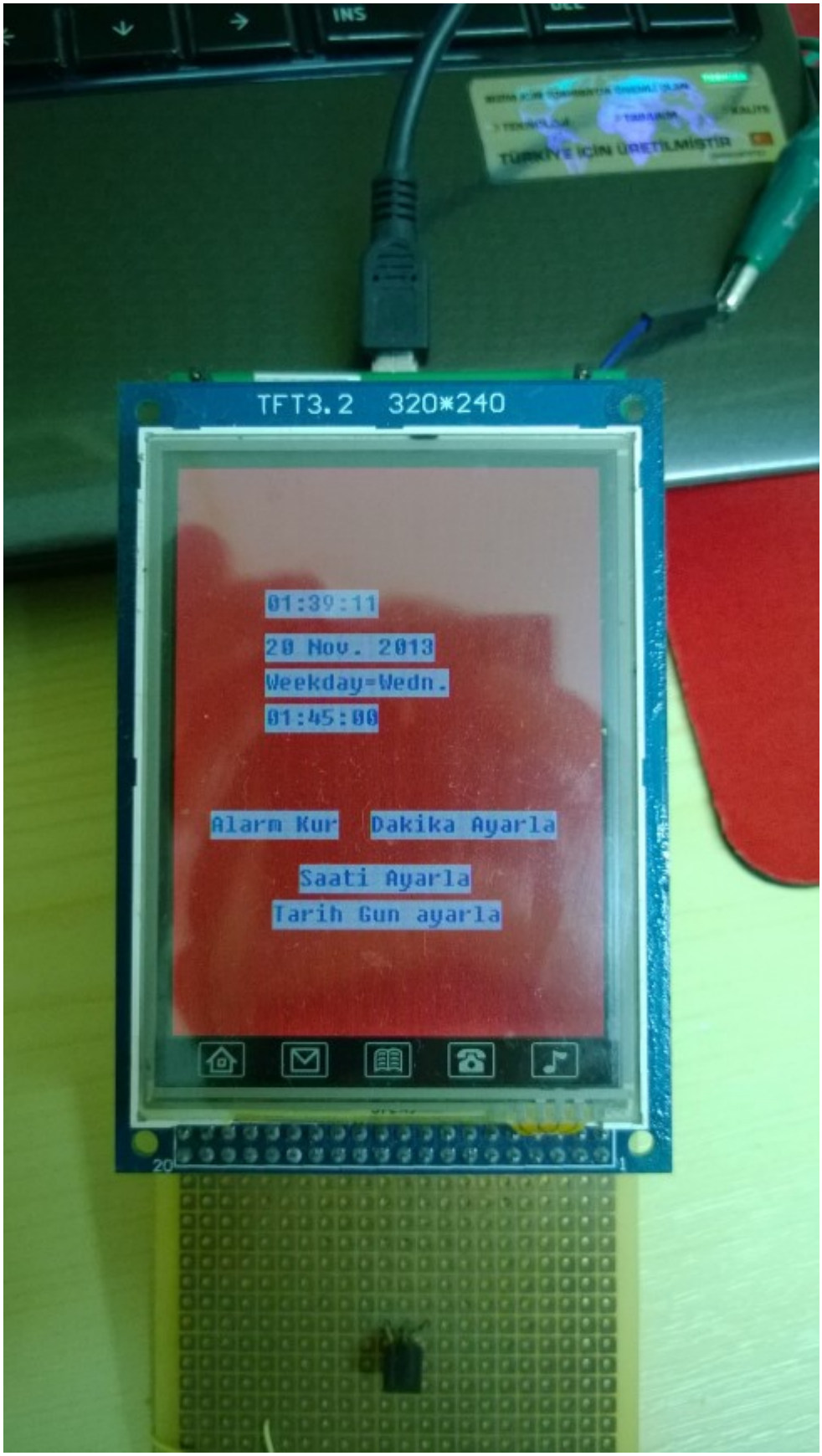
```
//RTC_TimeStructure.RTC_H12 = RTC_H12_PM;
RTC_TimeStructure.RTC_Hours = 01;
RTC_TimeStructure.RTC_Minutes = 37;
RTC_TimeStructure.RTC_Seconds = 30;

RTC_SetTime(RTC_Format_BIN, &RTC_TimeStructure);
RTC_SetDate(RTC_Format_BIN, &RTC_DateStructure);

/* Indicator for the RTC configuration */
RTC_WriteBackupRegister(RTC_BKP_DR0, 0x32F2);
}

void RTC_Alarm_IRQHandler(void)
{
/* Clear the EXTIL line 17 */
EXTI_ClearITPendingBit(EXTI_Line17);

/* Check on the AlarmA falg and on the number of interrupts per Second (60*8) */
if (RTC_GetITStatus(RTC_IT_ALRA) != RESET)
{
/* Clear RTC AlarmA Flags */
RTC_ClearITPendingBit(RTC_IT_ALRA);
RTC_HandlerFlag = ENABLE;
}
}

void RTC_Alarmkur(void)
{
/* EXTI configuration */
EXTI_ClearITPendingBit(EXTI_Line17);
EXTI_InitStructure.EXTI_Line = EXTI_Line17;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);

/* Enable the RTC Alarm Interrupt */
NVIC_InitStructur.NVIC_IRQChannel = RTC_Alarm_IRQn;
NVIC_InitStructur.NVIC_IRQChannelPreemptionPriority = 15;
NVIC_InitStructur.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructur.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructur);

/* Enable the alarm */
RTC_AlarmStructure.RTC_AlarmMask = RTC_AlarmMask_DateWeekDay;
RTC_AlarmStructure.RTC_AlarmTime.RTC_Seconds = 0;
RTC_AlarmStructure.RTC_AlarmTime.RTC_Minutes = 45;
RTC_AlarmStructure.RTC_AlarmTime.RTC_Hours = 01;

RTC_AlarmCmd(RTC_Alarm_A, DISABLE);
RTC_SetAlarm(RTC_Format_BIN, RTC_Alarm_A, &RTC_AlarmStructure);

RTC_ITConfig(RTC_IT_ALRA, ENABLE);
RTC_AlarmCmd(RTC_Alarm_A, ENABLE);
```

```
}

int main(void)
{
uint8_t TempStr[25];

float x1,y1;
float x2,y2;

GPIO_InitTypeDef GPIO_InitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_Init(GPIOD, &GPIO_InitStructure);
GPIO_ResetBits(GPIOD , GPIO_Pin_7); //CS=0;

/*Initalization*/
touch_init();
LCD_Init();
Delay(0x3FFFFF);
LCD_Clear(RED);
LCD_SetTextColor(GREEN);
LCD_BackLight(100);

LCD_SetDisplayWindow(0, 0, 239, 319);

RTC_kur();
RTC_Alarmkur();

while (1)
{
/*————————————————-Touch-Panel————————————————————-*/
BASA :
Read_Ads7846();

if(flag==1)
{
flag=0;

x1=xxx;
x2=x1/7.666666667;
// x2=x1/5.609375;
x2=x2-13;
x1=x2;

y1=yyy;
y2=y1/5.609375;
// y2=y1/7.666666667 ;
y2=y2-20;
y1=y2 ;
```

```
}

if ((x1>200) && (x1<215) && (y1>304) && (y1<320))
{
//LCD_WriteBMP(239,319 , 240,320, gImage_r3);
goto BASA ;
}

// Ekran silme fonksiyonu .
if ((x1>215) && (x1<240) && (y1>304) && (y1<320))
{
LCD_SetDisplayWindow(0, 0, 239, 319);
LCD_Clear(BLACK);
LCD_BackLight(100);
//LCD_WriteBMP(239,319 , 15, 16, gImage_logo);
goto BASA ;
}

// Ekran Isigi ayarlama fonksiyonu
if ((x1>0) && (y1<10) && (x1<101))
{
LCD_BackLight(x1);
goto BASA ;
}

// Ekrana piksel basma fonksiyonu .
if ((x1>0) && (y1>0))
{
Pixel(x1,y1,BLUE);
Pixel(x1+1,y1,BLUE);
Pixel(x1,y1+1,BLUE);
Pixel(x1+1,y1+1,BLUE);
}

/*————————————————————————————————————————————————————————————*/

/* Get info from RTC here */
RTC_GetTime(RTC_Format_BIN, &RTC_TimeStructure);

sec = RTC_TimeStructure.RTC_Seconds;
min = RTC_TimeStructure.RTC_Minutes;
hour = RTC_TimeStructure.RTC_Hours;

LCD_SetTextColor(LCD_COLOR_BLUE2);
sprintf((char *)TempStr, "%02d:%02d:%02d", hour , min, sec);
//GUI_Text(0,0,(uint8_t*)TempStr,White,Red);
LCD_StringLine(50,70,(u8*)TempStr);

RTC_GetDate(RTC_Format_BIN, &RTC_DateStructure);

year = RTC_DateStructure.RTC_Year + 2000;
month = RTC_DateStructure.RTC_Month;
day = RTC_DateStructure.RTC_Date;
weekday= RTC_DateStructure.RTC_WeekDay;
```

```
sprintf((char *)TempStr, "%02d %s %04d", day , strMonth[month-1], year);
//GUI_Text(0,20,(uint8_t*)TempStr,White,Red);
LCD_StringLine(50,95,(u8*)TempStr);

sprintf((char *)TempStr, "Weekday=%s", strWeekday[weekday-1]);
//GUI_Text(0,40,(uint8_t*)TempStr,White,Red);
LCD_StringLine(50,115,(u8*)TempStr);

RTC_GetAlarm(RTC_Format_BIN, RTC_Alarm_A, &RTC_AlarmStructure);

asec = RTC_AlarmStructure.RTC_AlarmTime.RTC_Seconds;
amin = RTC_AlarmStructure.RTC_AlarmTime.RTC_Minutes;
ahour = RTC_AlarmStructure.RTC_AlarmTime.RTC_Hours;

sprintf((char *)TempStr, "%02d:%02d:%02d", ahour , amin, asec);
//GUI_Text(0,60,(uint8_t*)TempStr,White,Red);
LCD_StringLine(50,135,(u8*)TempStr);

/*——————————Button-Settings——————————-*/
LCD_StringLine(20,195,"Alarm Kur");
LCD_StringLine(110,195,"Dakika Ayarla");
LCD_StringLine(70,225,"Saati Ayarla");
LCD_StringLine(55,245,"Tarih Gun ayarla");
}
}
```

Uygulama da dokunmatik ekran desteğiyle Alarm kurma, saat dakika gün ayarları özellikleri eklenecektir umarım en kısa zamanda bitirip sizlerle paylaşırım. Projenin şuan ki halini aşağıdaki resimde görebilirsiniz.




(https://kaankandemir.files.wordpress.com/2013/12/boxfiles_13120713190410.jpg)

Umarım yardımcı olabilmişimdir eğer sizlerinde arayüz hakkında fikirleri olursa benimle paylaşırsanız cok memnum olurum. Kolay gelsin , iyi çalışmalar.

Posted by kaankandemir in STM32
Tagged: Stm32f4 RTC

# 4 thoughts on "Stm32F4 Discovery RTC modülü"

1. **Gökhan Beken** *dedi ki:*
   04 Haziran 2014, 11:20
   Teşekkür ederim, işime yaradı.

   **Cevapla**
2. **BATUHAN** *dedi ki:*
   02 Ocak 2015, 00:36
   TEŞEKKÜR EDERİZ KAAN ABİ BENİM BÖYLE BİR UYGULAMA YAPMAM LAZIMDI IAR ÜZERİNDEN STM32F4 İLE BU YAZINIZ SANIRIM İŞİME YARAYACAK.SİZDEN RİCAM BUNUN BAĞLANTI PIN ŞEMASINI GÖNDEREBİLİRMİSİNİZ.

   **Cevapla**
3. **kaankandemir** *dedi ki:*
   02 Ocak 2015, 20:28
   Merhaba ,

Bu uygulamada herhangi harici bir entegre kullanılmamıstır. RTC modul stm32f4 içinde dahili olarak var zaten direk kodu kullanabilirsiniz.

Kolay gelsin…

**Cevapla**

4. **Mustafa Tekdemir** *dedi ki:*
   26 Eylül 2016, 10:52
   Hocam sorunumu çözdüm, işlemci enerjisi kesildikten sonra enerji verdiğimde Flash'a yazdığım Flag kontrolü ile tekrar RTC biriminin tekrar init edilmesini engelliyorum sadece RTC_DR ve RTC_TR registerlarını okuyarak saati gözlemlemeye devam ediyorum. Kolay gelsin iyi çalışmalar.

   **Cevapla**