BMS 302 MİKROKONTROLCÜLER

ZAMANLAYICI VE SAYICILAR

Prof.Dr. Mutlu AVCI

Çukurova Üniversitesi Biyomedikal Mühendisliği Bölümü

Bahar 2020

- En son dersimizde Kesmeler, gereklilikleri ve kullanımlarını tanımış harici donanım kesmelerini incelemiştik.

- Bu dersin içeriği dahili kesmelerden olan zamanlayıcı kesmelerini kullanabilmek için zamanlayıcı birimlerinin tanınması, sayıcı olarak kullanılmaları, zamanlama ayarlamaları ve kurulmaları kavramları öğrenilecektir.

- Dersin sonunda zamanlayıcı/sayıcı birimini kullanan CCS C dili ile gömülü yazılım geliştirilebilir hale gelinmesi hedeflenmektedir.

GİRİŞ

- Zamanlayıcı birimini evimizde kurduğumuz bir çalar saate benzetebiliriz.
- Çalar saat kurmadan herhangi bir vakitte uyanmamız veya o vakte kadar başka işlerle uğraşmamız gerekirse ya aklımızı meşgul ettiği için uyku konforumuzun düşmesi yada uyumuyor başka işlerle uğraşıyorsak ara sıra saati kontrol etmemiz gerekir.
- Her iki durumda da tam zamanlamayı kaçırma olasılığımız veya aklımızı sürekli zaman geldi mi sorusuyla meşgul edip sık sık saati kontrol etmek gibi yük oluşturmamız söz konusudur.

- Bir mikrokontrolcü mikrosaniyeler ölçeğinde komut çevrimi yapabilirken saniyeler mertebesinde gelen ürünleri sayması gerektiğini düşünelim.

- Mikrokontrolcüyü sürekli ürün bekler halde tutmak başka işlem yapmasına, haberleşmesi gereken birimlerle iletişim kurmasına engel olacaktır.

- Benzer şekilde sürekli ürün geldi mi kontrolünün yapılması da gereksiz ve verimsiz bir işlemci kullanımı olacaktır.

- İşlemciyi meşgul etmeden sayılacak nesnenin gelmesi durumunda kesme ile işlemciye haber verecek bir birimin olması zaruridir.

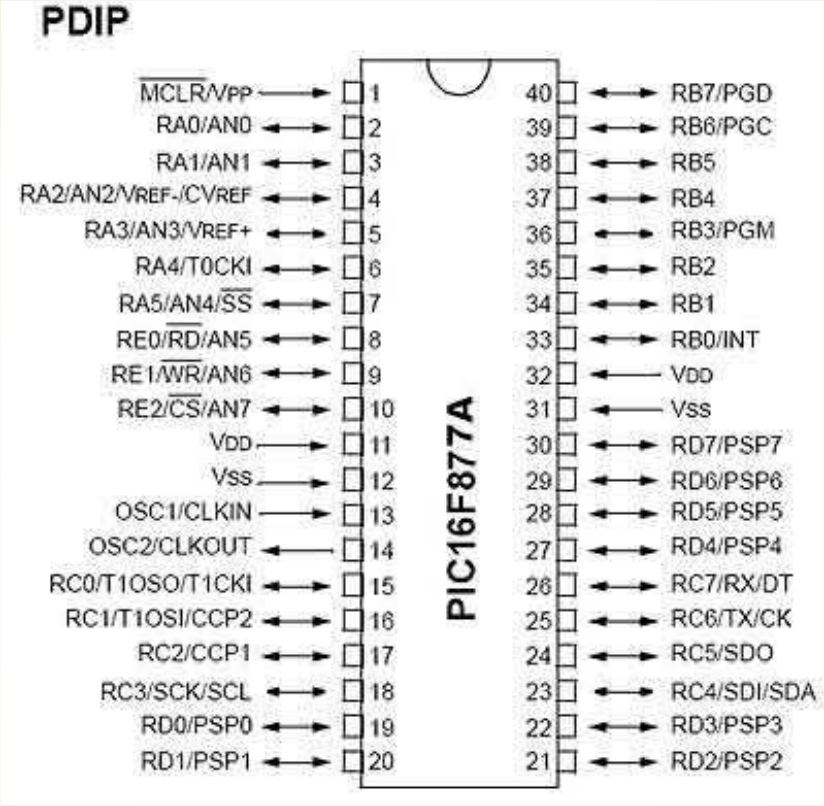
- Zamanlayıcı ve sayıcı birimlerinin yetenekleri bunlardır.
- Zamanlayıcı (timer) ve sayıcı (counter) birimleri hem mikroişlemci hem de mikro kontrolcüler için olması zorunlu standart birimlerdir.
- Zamanlayıcı ve sayıcı birimleri ayrı ifade edilseler dahi aynı birimleridir. Aynı birim kullanım şekline göre zamanlayıcı veya sayıcı olarak kullanılabilir.
- Zamanlayıcı ve sayıcı birimlerinin ikisinin kendi içeriklerinin artışı için bir sinyal kaynağına (clock) ihtiyaç vardır.

- Zamanlayıcı artışını sinyal kaynağı olarak mikro kontrolcünün kristalinden sağlar. Yani kontrolcünün her komut çeviriminde artacak şekilde çalışır.

- Sayıcı ise, sinyal kaynağı olarak harici girişlerden gelen sinyalleri kullanır.

- Dolayısıyla zamanlayıcının artışı düzenli bir şekilde olurken, sayıcının artışı düzenli olmak zorunda olmayan düzensiz frekans değerlerinde olabilir.

- PIC16F877A mikrokontrolcüsü içinde 3 farklı zamanlayıcı vardır. Bunlardan 2 tanesi aynı zamanda sayıcı birimidir, bir de Watch Dog Timer birimi vardır.
- Bu birimler Timer0, Timer1 ve Timer2 olarak adlandırılır. WDT bunlardan biraz daha özel ve farklı amaçlı bir birimdir.
- Timer0 -> 8 bit (Z/S), Timer1 -> 16 bit (Z/S) ve Timer2 -> 8 bit (Z) çözünürlüklüdür.
- Her bir Timer biriminin kullanımı için: Kurulum ayarlaması, başlama değeri ayarlaması ve kesme fonksiyonunun C kodunun yazılması gereklidir.



Şekil 1 PIC 16F877A blok diyagramı üzerinde Timer / Counter uçları

Timer0

- 8 bit Zamanlayıcı/Sayıcı
- Okuma ve Yazma
- 8 bitlik yazılım ile programlanabilen bölme oranı (prescaler)
- Dahili ve harici saat (clock) sinyali seçimi
- Timer0 değerinin 256'dan 0'a geçişinde taşma kesmesi
- Harici saat kaynağı sinyal tetikleme kenarı seçimi (edge select)
- Timer0 sayıcı olarak kullanıldığında RA4 ucundan gelen işaretleri sayar

SETUP_TIMER_0 ( ) Fonksiyonu:

- Bu fonksiyon Timer0 zamanlayıcı ayarlarını yapmaya yarar.

Kullanımı ;

setup_timer_0(mod);

şeklindedir.

- Fonksiyonun "mod" kısmına aşağıda belirtilen sabit tanımlamalardan biri veya birden fazlası yazılabilir.
- İstenilen özelliği sağlayacak sabitlerin arasına ' | ' operatörü konulur.

- “RTCC_INTERNAL” = Timer0 saat kaynağının denetleyici çalışma frekansı olacağını bildirir.
- “RTCC_EXT_L_TO_H” = Timer0 saat sinyalinin dışarıdan (RA4 ucundan) alınacağını ve sinyalin her yükselen kenarda tetikleme olacağını belirtir.
- “RTCC_EXT_H_TO_L” = Timer0 saat sinyalinin dışarıdan (RA4 ucundan) alınacağını ve sinyalin her düşen kenarda tetikleme olacağını belirtir.
- Bölme Oranı (Prescaler) Sabitleri : RTCC_DIV_2, RTCC_DIV_4, RTCC_DIV_8, RTCC_DIV_16, RTCC_DIV_32, RTCC_DIV_64, RTCC_DIV_128, RTCC_DIV_256. Sabit sonundaki rakamlar bölme oranını belirtir.

- Örneğin:

```
setup_timer_0(RTCC_EXT_L_TO_H | RTCC_DIV_4) ;
```

- Yukarıdaki örnek ile Timer0'ın harici kaynaktan (RA4 ucundan) besleneceği (sayıcı olarak kullanılacağı) harici kaynak sinyalinin her yükselen kenarında artış olacağı ve bölme oranının 4 olacağı yani her dört saat darbesinde sayımın bir artacağı belirtilmiştir.

SET_TIMER0 ( ) ve SET_RTCC( ) Fonksiyonları

- SET_TIMER0 ve SET_RTCC fonksiyonları aynı anlamdadır.
- İki fonksiyonda Timer0 biriminin saymaya başlama değerini belirlemek için kullanılır.
- Tüm sayıcılar yukarı doğru sayar ve maksimum değere ulaşınca sıfıra döner.
- Timer0 8 bitlik bir sayıcı olduğu için, saymaya başlama değeri 0 ile 255 (0 ve 255 dahil) arasındadır.

Bu fonksiyonların kullanımı ;

- set_timer0(250);

veya

- set_RTCC (250);

şeklindedir. İki komutta saymaya 250 den başlanacağını belirtir.

TIMER0 Kesmesi (#INT_TIMER0)

- Timer0 biriminin gerekli kurulumu yapıldıktan sonra istenirse taşma kesmesi de oluşturulabilir.
- Bu işlem için kesmelerin kullanımına benzer bir mantıkla *#INT_TIMER0* komutunun ardından kesme fonksiyonu yazılabilir.
- Burada dikkat edilmesi gereken nokta kesme fonksiyonu içine mutlaka *set_timer0 ( )* komutu ile programda belirtilen değerin tekrar kesme fonksiyonu içinde belirtilmesi gerekliliğidir.
- Timer0 kaydedici için oluşturulacak kesmenin oluşma süresi aşağıda gösterilen şekilde hesaplanır.

Kesme Oluşma Süresi = T_{komut} x (Bölme Oranı)x(256-Timer0 başlangıç değeri)

T_{komut} = 1/f_{komut}

f_{komut} = Kontrolcü Osilatör Frekansı / 4 = f_{osc}/4

Timer0 kesmesinin harici kaynak kullanılarak oluşturulmak istenmesi durumunda ise, kaç sinyal sonunda kesmenin oluşturulacağı

Kesme Oluşma Sayısı = (256 – Timer0 başlangıç değeri) x (Bölme Oranı)

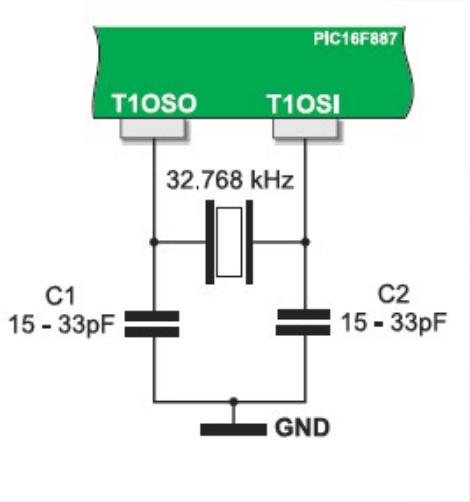
Şeklinde hesaplanır.

Timer1 Birimi

- 16 bit zamanlayıcı/Sayıcı birimidir.
- Okuma ve Yazma yapılabilir.
- Dahili ve harici saat (clock) sinyali seçimi mevcuttur.
- Timer1 değerinin 65536'den 0'a geçişinde taşma kesmesi oluşur.
- Timer1 *Senkron Sayıcı*, *Senkron Zamanlayıcı* ve *Asenkron Sayıcı* olmak üzere 3 farklı moda kullanılabilir.
- Timer1 harici saat sinyali için RC0 ve RC1 uçlarını kullanabilir.
- Sayıcı modunda RC0 ucundan gelen sinyali sayar.

- Harici saat darbesi için kristal osilatörlü devre bağlantısı.

  RC0 (T1OSO), RC1 (T1OSI)



- ***SETUP_TIMER_1 ( ) Fonksiyonu***

- Bu komut Timer1 zamanlayıcı ayarlarını yapmaya yarayan bir fonksiyondur. Kullanımı ;

- setup_timer_1(mod);

şeklindedir.

- Fonksiyonun “mod” kısmına aşağıda belirtilen sabit tanımlamalardan biri veya birden fazlası yazılabilir. İki sabit arasına ‘ | ’ operatörü konulur.

“T1_DISABLED” = Timer1’i kapat.

“T1_INTERNAL” = Timer1 saat kaynağının denetleyici çalışma frekansı olacağını bildirir.

“T1_EXTERNAL” = Timer1 saat sinyali kaynağının harici olacağını belirtir.

“T1_EXTERNAL_SYNC” = Timer1’in harici sinyal kaynağı ile senkronize olacağını belirtir.

“T1_CLK_OUT” = Timer1 osilatörünün kullanılacağını bildirir.

- Bölme Oranı (Prescaler) Sabitleri :

T1_DIV_BY_1, T1_DIV_BY_2, T1_DIV_BY_4, T1_DIV_BY_8. Sabit sonundaki rakamlar bölme oranını belirtir.

Fonksiyonun kullanımı ;

setup_timer_1(T1_EXTERNAL | T1_DIV_BY_4)

şeklindedir. Yukarıdaki örnek ile Timer1'ın harici kaynaktan besleneceği (sayıcı olarak kullanılacağı) ve bölme oranının 4 olacağı belirtilmiştir.

SET_TIMER1 ( ) Fonksiyonu

- Bu komut Timer1 biriminin saymaya başlama değerini belirlemek için kullanılır. Timer1 16 bitlik bir sayıcı olduğu için, saymaya başlama değeri 0 ile 65535 (0 ve 65535 dahil) arasında olmalıdır. Kullanımı;

```
set_timer1(65530);
```

- şeklindedir.

- Timer1 kesmesi, kesmenin oluşacağı süre ve harici kaynakla kesmenin gerçekleştirilmesi işlemleri Timer0 kesmesinde anlatılan yöntemle gerçekleştirilip hesaplanabilir.

TIMER1 Kesmesi (#INT_TIMER1)

Kesme Oluşma Süresi = T_{komut} x (Bölme Oranı)x(65536-Timer1 başlangıç değeri)

T_{komut} = 1/f_{komut}

f_{komut} = Kontrolcü Osilatör Frekansı / 4 = f_{osc}/4

Timer1 kesmesinin harici kaynak kullanılarak oluşturulmak istenmesi durumunda ise, kaç sinyal sonunda kesmenin oluşturulacağı

Kesme Oluşma Sayısı = (65536 – Timer1 başlangıç değeri) x (Bölme Oranı)

Şeklinde hesaplanır.

Timer2 Birimi

- Timer2 birimi Timer0 ve Timer1 birimlerinden farklı olarak sadece zamanlayıcı olarak kullanılan 8 bitlik bir birimdir.
- Timer2 biriminde Timer2 zamanlayıcısı için denetleyicinin dahili saat sinyali (fosc/4) kullanılır.
- Bu sinyal *bölme oranı (prescaler)* ile çarpılarak kaynak saat sinyalini oluşturur.
- 8 bitlik Timer2 zamanlayıcı kaydedicisi içindeki değer denetleyici frekansı ile bölme oranı ile hesaplanan frekansta artar.

- PR2 8 bitlik yazılabilir ve okunabilir bir periyot kaydedicisidir, 0 ile 255 arasında bir değer alabilir.
- Timer2 kaydedicisi içindeki değer, 0'dan başlayarak sayar ve PR2 kaydedicisi içine yazılan değere eşit olduğunda bir eşitlik sinyali üretir ve Timer2 kaydedicisi değerini sıfırlar.
- Eşitlik sinyali sonunda PR2 kaydedicisi de 255 değerini alır.
- *Postscaler* değeri meydana gelen bu eşitlik sinyallerinden kaç defa meydana geldikten sonra Timer2 kesmesi sinyalinin oluşacağını belirler.

Örneğin Postscaler değeri 2 ise, 2 defa Timer2 kaydedicisi değeri PR2 kaydedicisi değerine eşit olunca Timer2 kesmesi oluşur.

SETUP_TIMER_2 ( ) Fonksiyonu

- Bu fonksiyon Timer2'nin zamanlayıcı ayarlarını yapmaya yarayan fonksiyondur. Kullanımı ;

setup_timer_2(mod, periyot, postscale) ;

şeklindedir.

- Fonksiyonun "mod" kısmına aşağıda belirtilen sabit tanımlamalardan biri veya birden fazlası yazılabilir.

"T2_DISABLED" = Timer2'yi kapat

Bölme Oranı (Prescaler) Sabitleri : T2_DIV_1, T2_DIV_4, T2_DIV_16. Sabit sonundaki rakamlar bölme oranını belirtir.

Periyot : Periyot kısmına 0 ile 255 arasında bir tamsayı girilir. Zamanlayıcı periyot değerine ulaşınca kendini sıfırlar. PR2 değerini temsil eder.

Postscale : Postscale kısmına 1 ile 16 arasında bir tamsayı girilir. Postscale değeri, Timer2 kesmesi oluşması için Timer2'nin kendini kaç kez sıfırlaması gerektiğini belirtir.

Örneğin;

setup_timer_2(T2_DIV_BY_16, 195, 5) ;

komutu ile Timer2 biriminin bölme oranı 16, PR2 kaydedicisinin değeri 195 ve postscale değeri 5 olarak ayarlanmıştır.

SET_TIMER2 ( ) Fonksiyonu

- Bu komut Timer2 biriminin saymaya başlama değerini belirlemek için kullanılır. Timer2 8 bitlik bir sayıcı olduğu için, saymaya başlama değeri 0 ile 255 (0 ve 255 dahil) arasında olmalıdır.

Kullanımı;

set_timer2(100);

şeklindedir.

- Saymaya 100 den başlar PR2 nin değerine kadar sayar

TIMER2 Kesmesi (#INT_TIMER1)

Kesme Oluşma Süresi = T_{komut} x (Bölme Oranı)x (PR2+1) x (Postscale)

T_{komut} = 1/f_{komut}

f_{komut} = Kontrolcü Osilatör Frekansı / 4 = f_{osc}/4

Harici kullanımı yoktur.

- **Watch Dog Timer**
- WDT birimi denetleyici içinde gömülü bulunan ve ek olarak dışarıdan hiçbir elemanın bağlantısına ihtiyaç duymayan bir R/C osilatörüdür.
- Kontrolcü osilatör girişlerine bağlanan osilatörden bağımsızdır.
- Kontrolcü olsilatöre bağlanmamış olsa ya da kontrolcünün uyku durumunda olmasında bile WDT çalışmaya devam eder.
- WDT birimi 8 bitlik bir zamanlayıcıdır. Bu nedenle WDT 0'dan 255'e kadar sayar ve 255'den sonra tekrar 0'a döndüğünde kontrolcüyü sıfırlar (resetler).
- Eğer kontrolcü uyku modunda iken WDT taşması meydana gelirse kontrolcü uyku modundan çıkar ve programı kaldığı yerden icra etmeye devam eder.

WDT birimini açma ya da kapama işlemi programın başında belirtilen sigorta komutlarıyla gerçekleştirilir.

FUSE WDT // WDT Açık

FUSE NOWDT // WDT Kapalı

SETUP_WDT ( ) Fonksiyonu

- WatchDog Timer ayarları için kullanılan fonksiyondur.

Kullanımı ;

setup_wdt (mod) ;

şeklindedir.

Fonksiyonda “mod” kısmına taşma süresini belirleyen bölme sabitleri yazılır. Bu bölme sabitleri şunlardır :

- WDT_18MS = WDT taşma süresi 18 ms
- WDT_36MS = WDT taşma süresi 36 ms
- WDT_72MS = WDT taşma süresi 72 ms
- WDT_144MS = WDT taşma süresi 144 ms
- WDT_288MS = WDT taşma süresi 288 ms
- WDT_576MS = WDT taşma süresi 576 ms
- WDT_1152MS = WDT taşma süresi 1152 ms //1.152 saniye
- WDT_2304MS = WDT taşma süresi 2304ms // 2.304 saniye

RESTART_WDT ( ) Fonksiyonu

WDT'yi sıfırlamak (reetlemek) için kullanılan fonksiyondur. Kullanımı ;

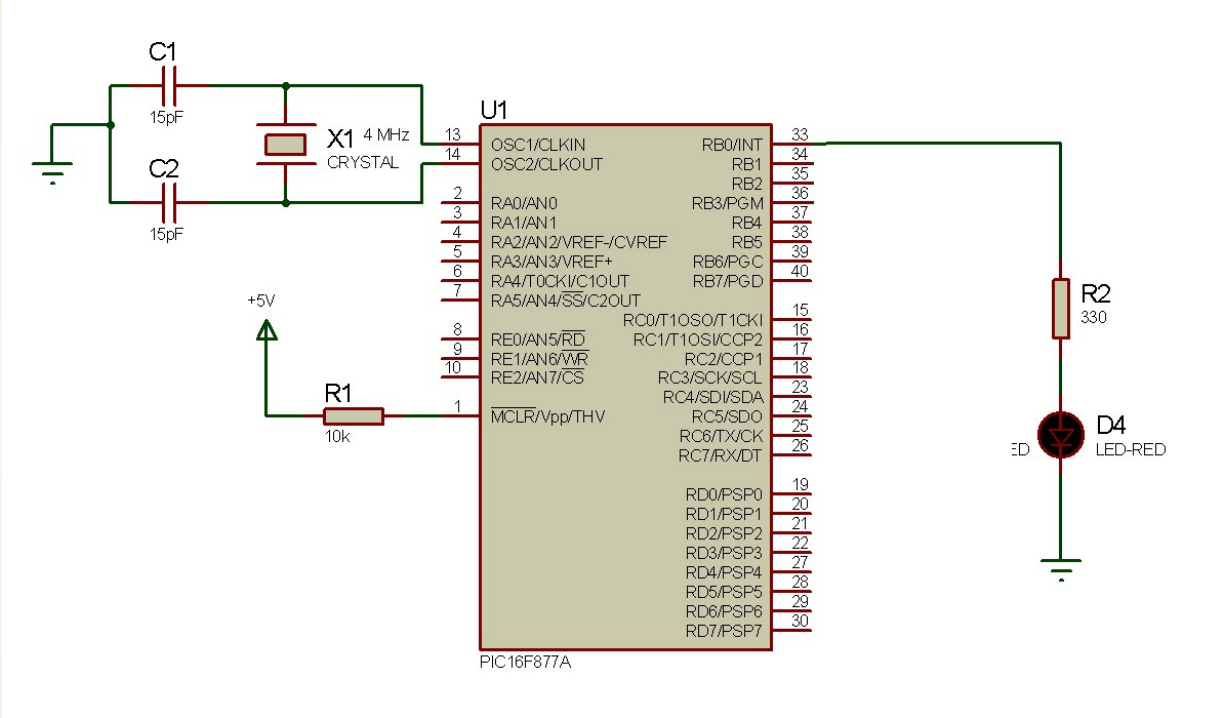
restart_wdt ( ) ;

şekilndedir.

Sleep ( ) Komutu

- Bu komut ile mikro kontrolcü uyku moduna geçer. Kullanımı ;
- sleep () ;
- şeklindedir.
- Sleep modundan kontrolcüyü çıkarmak için aşağıdaki yollardan biri uygulanır.
- MCLR reseti
- WDT reseti
- INT ucundan gelen kesmeler

Örnek:



Yukarıdaki devre için Timer0 zamanlayıcısını kullanarak RB0 ucuna bağlı LED lambayı yaklaşık 1 Hz frekansla yakıp söndüren CCS C yazılımını gerçekleştirin.

```
#include <ornek.h>
#use delay(clock=4000000)
int i=0
#int_timer0
void kesme( ){
set_timer0(60);
i++;
if (i==10)
output_toggle(Pin_B0);
i=0;}
```

```
void main(){
setup_timer_0(RTCC_INTERNAL | RTCC_DIV_256);
Set_timer_0(60);
enable_interrupts(INT_TIMER0);
enable_interrupts(GLOBAL);
while(1);
}
```