Teknoloji Fakültesi Elektrik Elektronik Mühendisliği Bölümü

EE-302 Mikroişlemciler

Timer ve Uygulamaları (Zamanlayıcı ve Sayıcı Kavramı)

6. Hafta

Prof. Dr. Mehmet DEMİRTAŞ

Zamanlayıcı ve Sayıcı Kavramı

- Zamanlayıcı (timer) ve sayıcı (counter) birimleri hem mikroişlemci hem de mikro denetleyiciler için neredeyse bulunması zorunlu birimlerdir. Bu birimler için çok önceleri harici özel entegreler kullanılırdı. Mikro denetleyici yapılarının gelişim evresinde önemli bir yerde bu tip donanım birimlerinin mikro denetleyici yapısına eklenmesidir. Zamanlayıcı ve sayıcı donanım birimlerinin yaptığı işlemler yazılımsal olarak da yapılabilir. Fakat donanımsal olarak yapılan bu işler yazılımsal olarak yapıldığında, kesmeler bölümünde belirtildiği gibi yazılımın büyük çoğunluğu meşgul edilmiş olur. Zamanlayıcı/sayıcı birimleri ayrı ayrı ifade edilse dahi aynı birimlerdir. Aynı birim, kullanım şekline göre zamanlayıcı veya sayıcı olarak kullanılabilir.

Zamanlayıcı ve Sayıcı Kavramı

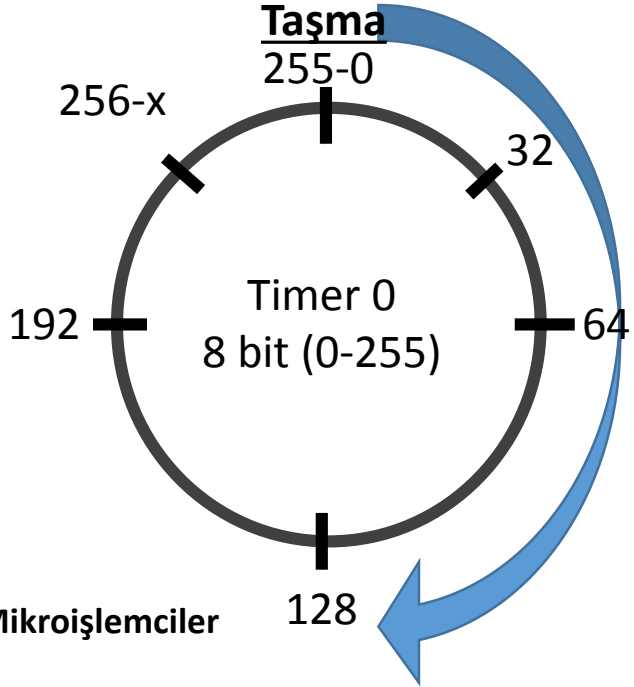
Bir birimin zamanlayıcı olarak kullanılması için değer artışının düzenli (periyodik) olması gerekmektedir. Sayıcı olarak kullanım da ise değer artışının düzenli olması gerekliliği yoktur. Örneğin bir fabrikada banttan geçen şişelerin sayımının sensor vasıtası ile yapıldığını farz edelim. Bu işlemde şişelerin banttan geçiş hızı önemli değildir. Bazen dakikada 100 şişe gelirken, bazen dakikada 150 şişe gelebilir. Önemli olan kaç adet şişe geçtiğinin tespitidir. Bu işlem için sayıcı birimi kullanılır. Aynı fabrikada şişelerin içine belli bir miktar sıvı konulduğunu farz edelim. Sıvının pompalandığı pompa borusunun çapı ve hızı belli olsun. Doldurulacak sıvı miktarının tespiti için sistemde pompa birimini belli bir süre aktif daha sonra pasif yapmak gereklidir. Bu işlem içinde mikro denetleyicinin zamanlayıcı birimi kullanılabilir.

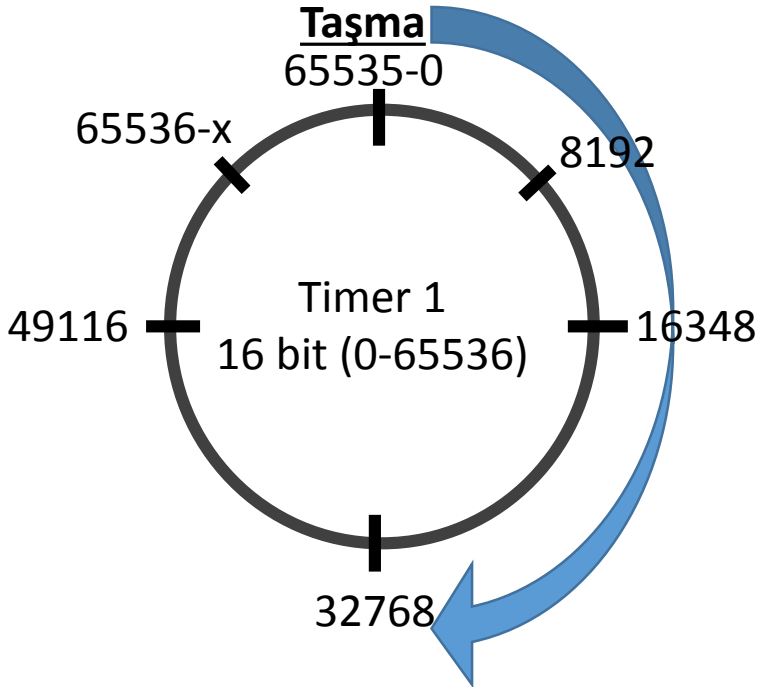
Zamanlayıcı ve Sayıcı Kavramı

Zamanlayıcı ve sayıcı birimlerinin ikisinin kendi içeriklerinin artışı için bir sinyal kaynağına (clock) ihtiyacı vardır. Zamanlayıcı ile sayıcı arasındaki fark olarak şunu ifade edebiliriz. Zamanlayıcı, artışını sinyal kaynağı olarak mikro denetleyicinin kristalinden sağlar. Yani denetleyicinin her komut çevriminde artacak şekilde çalışır. Sayıcı ise, sinyal kaynağı olarak harici girişlerden (dışarıdan) gelen sinyalleri kullanılır. Dolayısıyla zamanlayıcının artışı düzenli ve belli bir frekans değerinde olurken, sayıcının artışı düzenli olmak zorunda olmayan düzensiz frekans değerinde olmaktadır.

16f877'de kullanılan zamanlayıcı/sayıcılar

- Timer ya da sayıcılar pic'lerin içine yerleştirilmiş sayma görevine yarayan birimlerdir. 16f877a'nın içinde Timer0, Timer1 ve Timer2 olmak üzere 3 timer birimi bulunmaktadır. Timer0 ve Timer2 birimleri 8 bitlik, Timer2 ise 16 bitliktir. Bu şekilde düşünüldüğünde Timer0 ve Timer2 ile 255'e kadar, Timer1 ile 65535'e kadar sayma yada zamanlama işlemi yapılabilir.





TIMER_0 Birimi

- Timer0 birimi zamanlayıcı ve sayıcı olarak kullanılabilen bir donanım birimidir. Bu birimin özellikleri aşağıda verilmiştir.
- 8 bit Zamanlayıcı/Sayıcı kullanım,
- Okuma ve Yazma yapılabilir,
- 8 bit'lik yazılım ile programlanabilen bölme oranı (prescaler)
- Dahili ve harici saat (clock) sinyali kaynağı seçme,
- Timer0 değeri 0xFF'ten 0x00h'a (256-0) geçişinde taşma kesmesi,
- Harici saat kaynağı sinyal tetikleme kenarı seçimi (edge select)
- Timer0'in kontrolü OPTION kaydedicisi ile sağlanır.

REGISTER 5-1: OPTION_REG REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**

bit 6 **INTEDG**

bit 5 **T0CS**: TMR0 Clock Source Select bit

1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit

1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

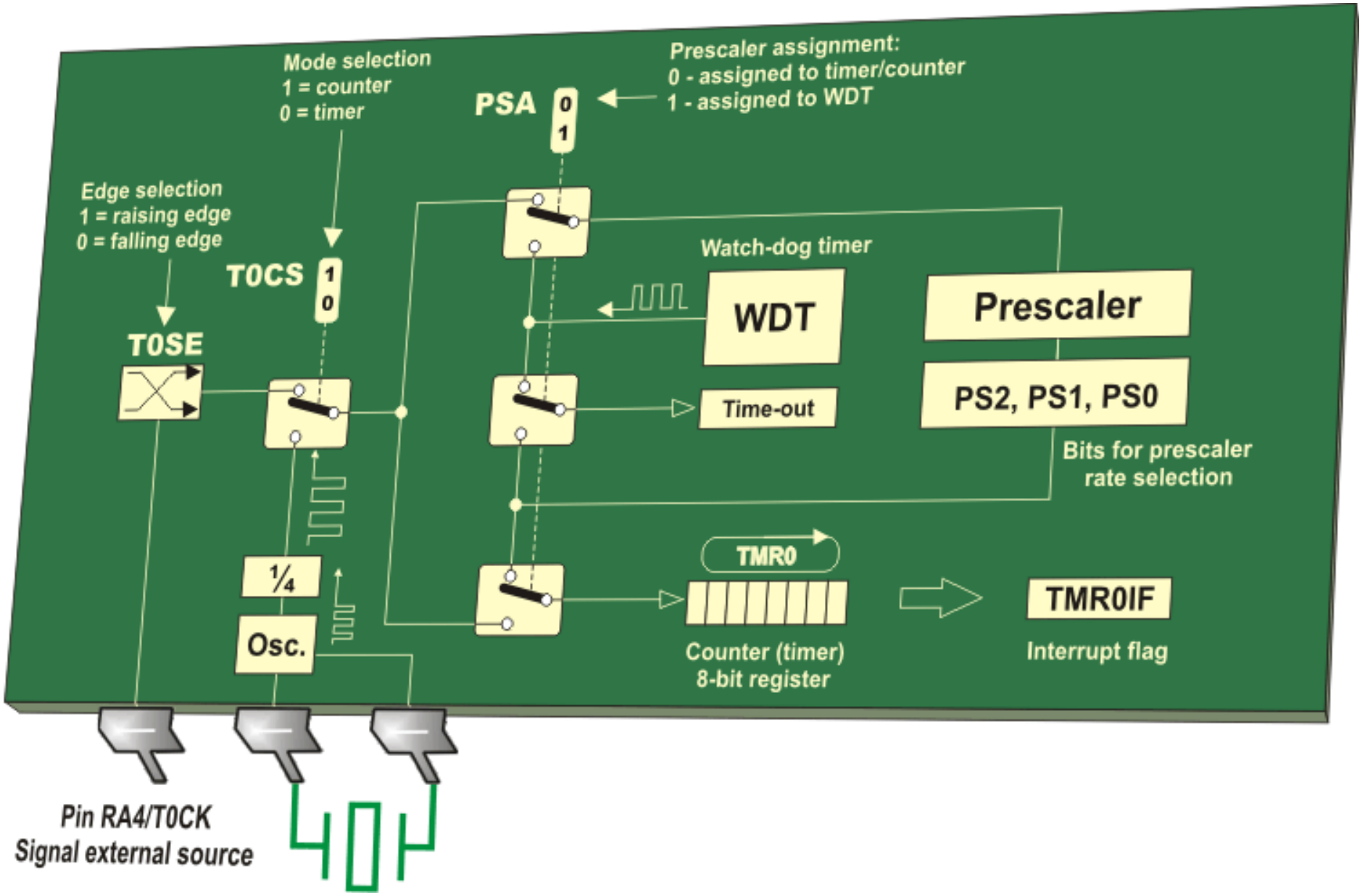
bit 3 **PSA**: Prescaler Assignment bit

1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

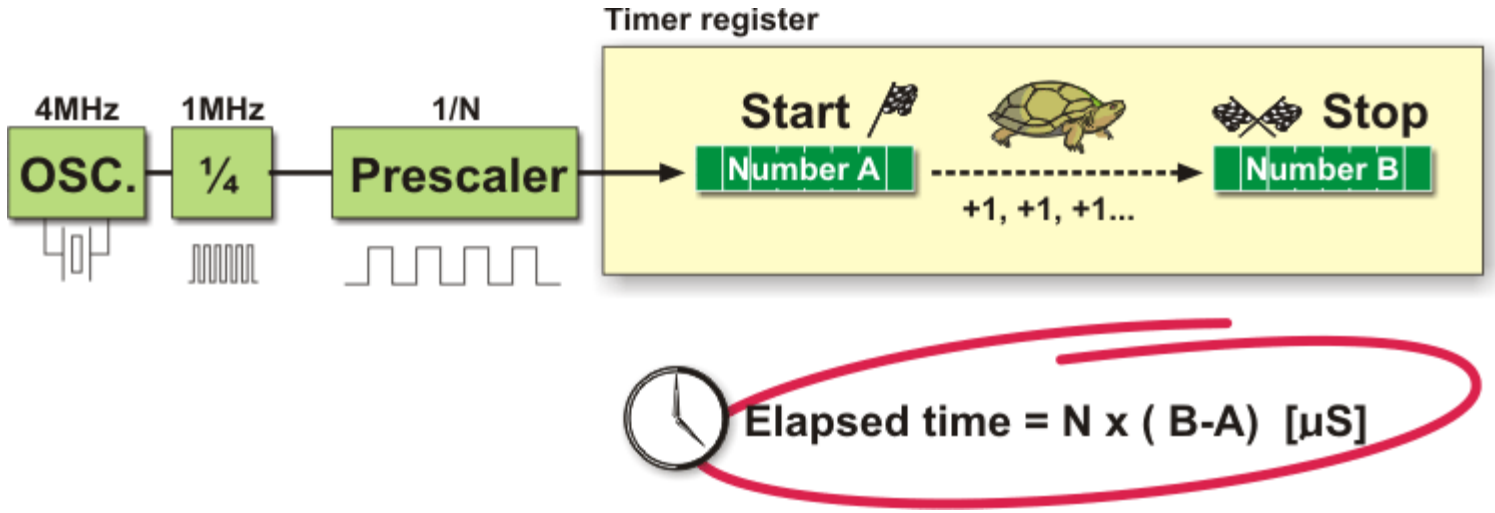
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown


Mode selection
1 = counter
0 = timer
Prescaler assignment:
0 - assigned to timer/counter
1 - assigned to WDT
PSA
0
1
Edge selection
1 = raising edge
0 = falling edge
T0CS
1
0
T0SE
Watch-dog timer
WDT
Prescaler
Time-out
PS2, PS1, PS0
Bits for prescaler
rate selection
TMR0
1/4
Osc.
TMR0IF
Counter (timer)
8-bit register
Interrupt flag
Pin RA4/T0CK
Signal external source

Prescaler (frekans bölücü)

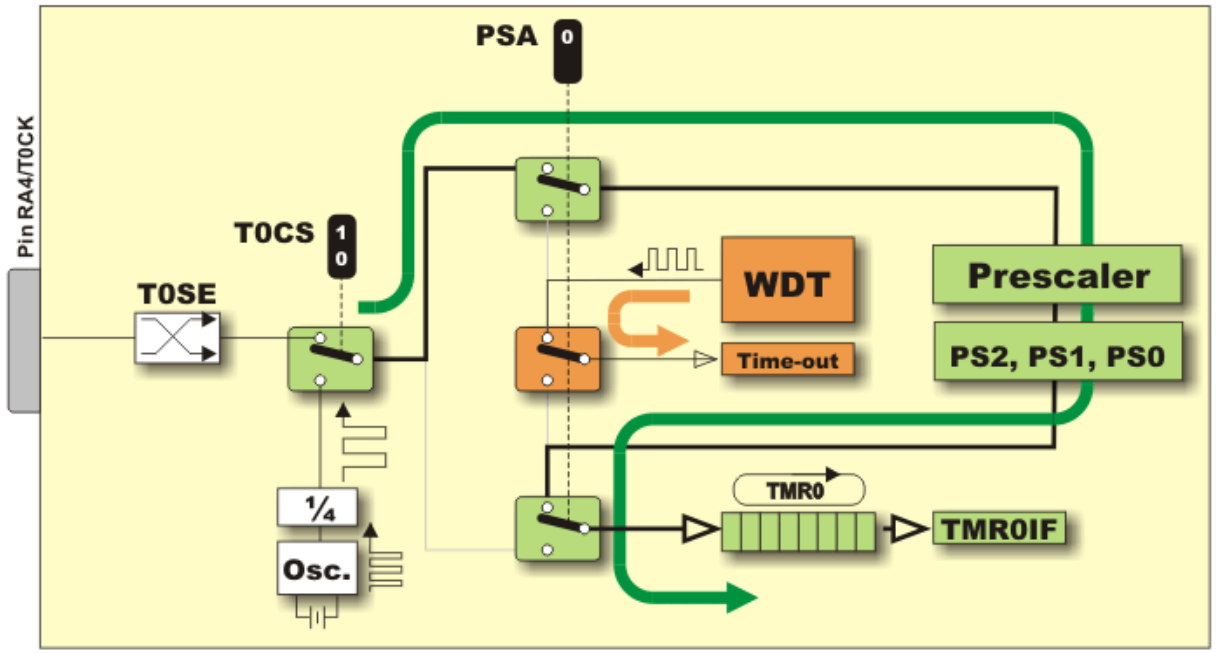
- Mikro denetleyicilerde çalıştırılacak Timer modülleri için PRESCALER (Frekans Bölücü) modülü kullanılmaktadır. Bu modül harici veya dahili saykıl ile çalışırken Timer'ın ilerleme hızını yavaşlatmak veya istenilen oranda ayarlamak için kullanılır. Bu ayar OPTION registerinin 0-2. bitleri ile yapılır. Geçen zamanın kısa hesabı aşağıda belirtilmiştir.

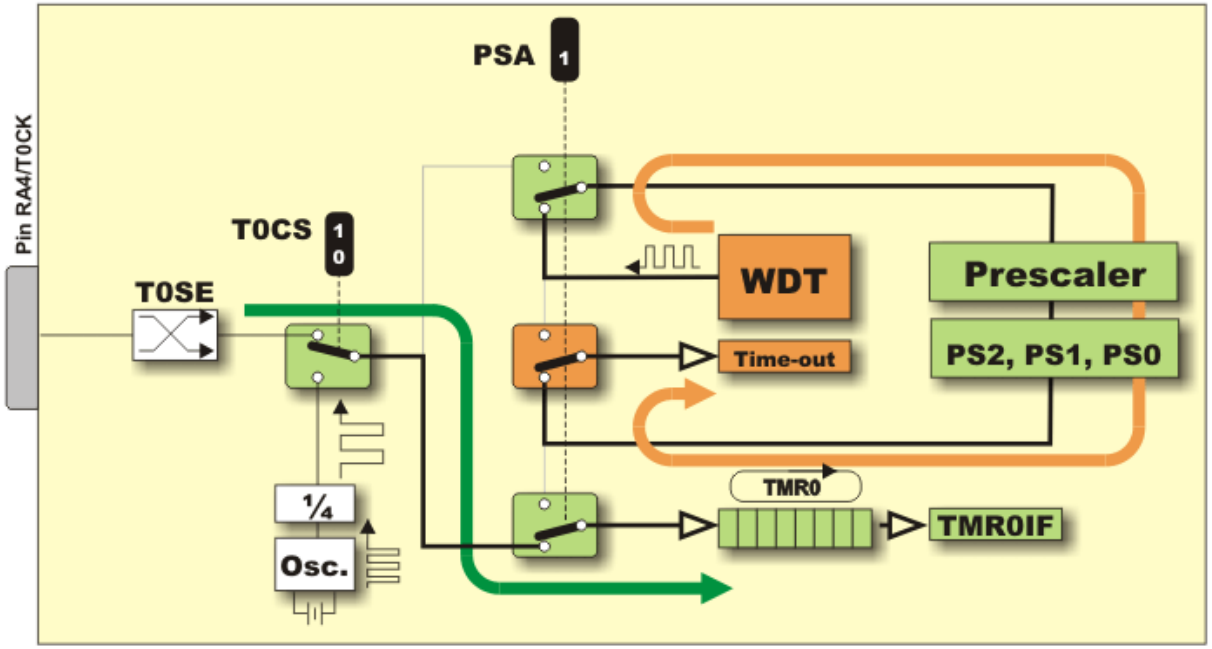
Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128



Prescaler (frekans bölücü)

- Timer0 için frekans bölücü oranının 1/1 kullanılmasını sağlamak için frekans bölücü WDT'ye yönlendirilir. Böylece TIMER0 frekans bölücüyü kullanmadan 1/1 oranla çalışabilir. Bunun için OPTION registerinin 3. biti (PSA)'nın «1» yapılması gerekmektedir.





CCS C'de Timer_0 (RTCC) Kullanım Komutları

- **SETUP_TIMER_0 ( ) Fonksiyonu «setup_timer_0(mod)»**
- Timer0 zamanlayıcım ayarlarını yapmaya yarayan bir fonksiyondur.
- Fonksiyondaki *"mod"* kısmına aşağıda belirtilen sabit tanımlamalardan biri veya birden fazlası yazılabilir. İki sabit arasında "I" operatörü konulur. Bu fonksiyonda kullanılacak sabitler denetleyicinin başlık dosyasında (Örneğin, 16F877.h gibi) yazılıdır.
- **Kesme Aktif Kenar Ayarı Sabitleri;**
- "RTCC_INTERNAL" = Timer0 clock kaynağının denetleyici çalışma frekansı olacağını bildirir.
- "RTCC_EXT_L_TO_H" =Timer0 clock kaynağının dış çevreden (RA4-TOCKI) alınacağını ve sinyalin her yükselen kenarında tetikleme olacağını bildirir.
- "RTCC_EXT_H_TO_L" = Timer0 kaynağının dış çevreden (RA4-TOCKI) alınacağını ve sinyalin her düşen kenarinda tetikleme olacağını bildirir.

CCS C'de Timer_0 (RTCC) Kullanım Komutları

- **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_DlV_256"

- Sabitin sonundaki rakamlar bölme oranını belirtir.
- **Sadece PIC18XXX ürünleri için** RTCC_OFF, RTCC_8_BIT mod komutları da mevcuttur.
- Örnek;

setup_timer_0 (RTCC_DIV_4 I RTCC_EXT_L_TO_H);

komutu ile Timer0'in harici kaynaktan besleneceği (sayıcı olarak kullanılacağı), harici kaynak sinyalinin her yükselen kenarında artış olacağı ve bölme oranının 4 olduğu belirtilmiş olur.

CCS C'de Timer_0 (RTCC) Kullanım Komutları

- **SET_TIMER0 ( ) ve SET_RTCC ( ) Fonksiyonu**
- SET_TIMER0 ve SET_RTCC ( ) fonksiyonları aynı anlamdadır. İki fonksiyonda Timer0 biriminin saymaya başlama değerini belirlemek için kullanılırlar. Tüm sayıcılar yukarı doğru sayar ve maksimum değere ulaşınca sıfıra döner ve tekrar yukarı saymaya başlar.
- **set_timer0 (*deger*);** veya **set_rtcc (*deger*);**
- Fonksiyonda *"değer"* kısmına Timer0 biriminin saymaya başlama değeri yazılır. Timer0 birimi 8 bit'lik olduğu için buraya yazılacak değer 0 ile 255 arasında olmalıdır.
- Örnek;

set_timer0 (45);

komutu ile Timer0 birimi saymaya 45'den itibaren başlar.

CCS C'de Timer_0 (RTCC) Kullanım Komutları

- **Timer0 Kesmesi (#INT_TIMER0)**

Timer0 biriminin gerekli kurulumu yapıldıktan sonra istenirse taşma kesmesi de oluşturulabilir. Kesmeler bölümünde anlatılan INTCON kaydedicisinin TOIE bit'i ile timer0 taşma kesmesi oluşturulup oluşturulmayacağı belirlenir. Bu iş için CCS C'de kesmeler bölümünde anlatılan aynı mantıkla #INT_TIMER0 komutu ile kesme fonksiyonu oluşturulur.

Burada dikkat edilmesi gereken nokta kesme fonksiyonu içinde mutlaka set_timer0 ( ) komutu ile programda set edilen değerin tekrar kesme fonksiyonu içinde de belirtilmesi gerekliliğidir. Çünkü program kesme fonksiyonuna gittiğinde, TMR0 kaydedicisi içeriği sıfırlanmış demektir. Timer0 biriminin tekrar başlangıçta istenen değer ile saymaya başlanması isteniyorsa kesme fonksiyonunda mutlaka tekrardan set_timer0 ( ) komutu ile set değeri belirtilmelidir.

CCS C'de Timer_0 (RTCC) Kullanım Komutları

- **Timer0 Kesmesi (#INT_TIMER0)**
- Timer0 kaydedicisi için oluşturulacak kesmenin oluşma süresi aşağıdaki formül ile hesaplanabilir.

- **Kesme Süresi =Tosc x (Bölme Oranı) x (256 – TMR0'a yazılan değer)**

- Tosc periyodu; *Tosc= 1/fosc* formülünden bulunur.
- fosc değeri bilindiği gibi PIC denetleyiciye bağlanan osilatör frekansının 4'e bölümü ile elde edilir.
- fosc = Denetleyici Osilatör Frekansı/4

CCS C'de Timer_0 (RTCC) Kullanım Komutları

- **GET_TIMERx ( ) Fonksiyonu**
- Bu fonksiyon komutu istenen zamanlayıcı ve sayıcının kaydedici (TMR0, TMR1 vb. kaydedicilerin) değerini bize verir. GET_TIMER0 komutu ile GET_RTCC0 ayni anlamdadır. get_timerx( ) komutunda x yerine istenen zamanlayıcı / sayıcı biriminin numarası yazılır. Bu komutta dikkat edilmesi gereken nokta Timer0 ve Timer2 birimleri 8 bit'lik olduğundan bu komut kullanıldığında geri dönüş değerinin kaydedileceği değişken 8 bit'lik olmalıdır. Örneğin Timer1 birimi 16 bit olduğundan bu birimin kaydedicisi içeriği bu komutla okunmak istendiğinde geri dönüş değerinin kaydedileceği değişken 16 bit'lik olmalıdır.
- *değer*= get_timer0( ), *değer* = get_rtcc( ), *değer*= get_timer1( ),
- *değer* = get_timer2( ), *değer* = get_timer3( )
- Örnek;
- x = get_timer2( ); *//* Timer2 biriminin TMR2 kaydedici değeri x değişkenine atanır.

TIMER_1 Birimi

- Timer1 birimi zamanlayıcı ve sayıcı olarak kullanılabilen bir donanım birimidir. Bu birimin özellikleri aşağıda verilmiştir.
- 16 bit Zamanlayıcı/Sayıcı kullanım,
- Okuma ve Yazma yapılabilir,
- Dahili ve harici saat (clock) sinyali kaynağı seçme,
- Timer1 değeri 0xFFFF'ten 0x0000'a (65536-0) geçişinde taşma kesmesi,
- Harici saat kaynağı sinyal tetikleme kenarı seçimi (edge select)
- Timer1'in kontrolü T1CON kaydedicisi ile sağlanır.

T1CON KAYDEDİCİSİ

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

Şekil-10.6. *T1CON kaydedicisi.*

Bit 7-6 = Kullanılmayan bit'ler. Okunurken bu bit'ler "0" bilgisi olarak okunur.

Bit 5-4 **T1CKPS1-T1CKPS0** = Timer1 bölme (prescaler) oranı
- 11 = 1:8
- 10 = 1:4
- 01 = 1:2
- 00 = 1:1

Bit 3 **T1OSCEN** = Timer1 osilatör kontrol bit'i
- 1:Osilatör açık
- 0:Osilatör kapalı

Bit 2 **$\overline{\text{T1SYNC}}$** = Timer1 dış clock sinyali senkronizasyon kontrol bit'i
- 1:Harici clock sinyali girişi ile senkronize yok
- 0:Harici clock sinyali girişi ile senkronize

Bit 1 **TMR1CS** = Timer1 clock kaynağı seçme bit'i
- 1:RC0/T1OSO/T1CKI pin'inden dış clock sinyali
- 0:Dâhili clock sinyali (F_{osc} /4)

Bit 0 **TMR1ON** = Timer1 açma-kapama bit'i
- 1:Timer1 açık
- 0:Timer1 kapalı

'ER 6-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
- 11 = 1:8 prescale value
- 10 = 1:4 prescale value
- 01 = 1:2 prescale value
- 00 = 1:1 prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit
- 1 = Oscillator is enabled
- 0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)

bit 2 **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit

When TMR1CS = 1:
- 1 = Do not synchronize external clock input
- 0 = Synchronize external clock input

When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit
- 1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
- 0 = Internal clock (Fosc/4)

bit 0 **TMR1ON**: Timer1 On bit
- 1 = Enables Timer1
- 0 = Stops Timer1

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

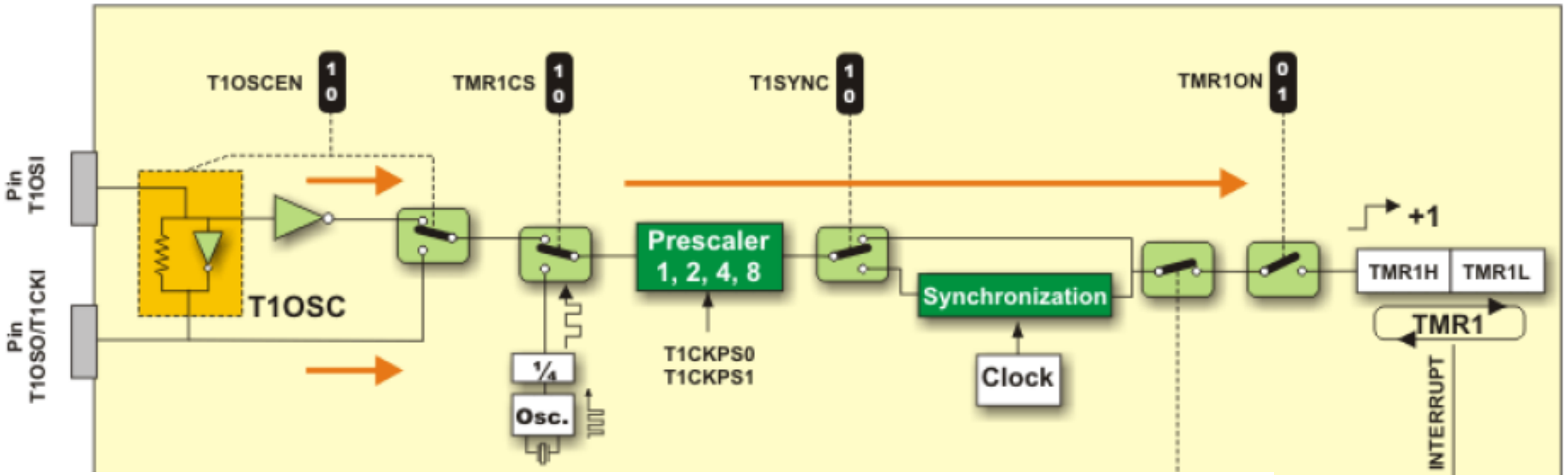
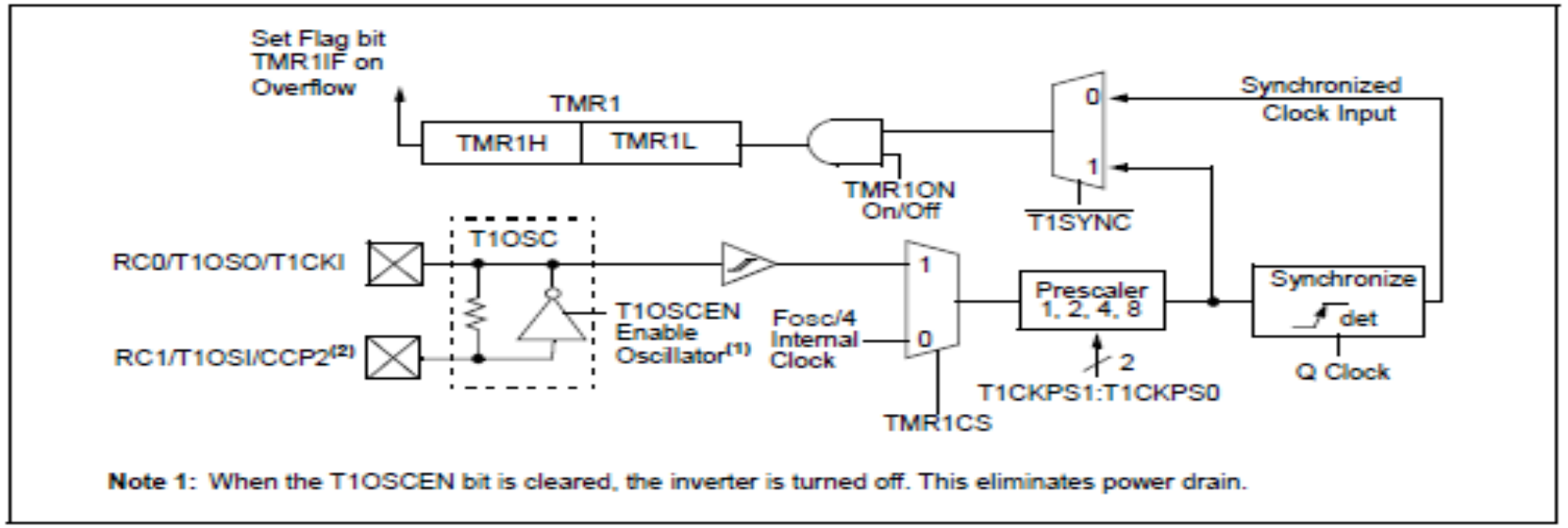


FIGURE 6-2: TIMER1 BLOCK DIAGRAM



Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

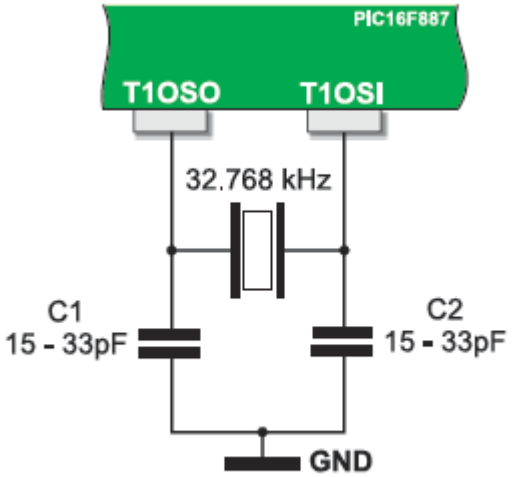
TIMER_1 Birimi

- Timer1 birimi 3 değişik modda kullanılabilir.
- Senkron Zamanlayıcı
- Senkron Sayıcı
- Asenkron Sayıcı

Zamanlayıcı modunda Timer1 her komut çevriminde bir artar. Sayıcı modunda ise Timer1, RCO/T1OSO/T1CKI pin'inden gelen saat sinyalinin **her yükselen kenarında bir artar.** Timer1 aynı zamanda dahili bir sıfırlama (reset) kaynağına sahiptir. Bu sıfırlama sinyali CCP modülünden gelen sıfırlama sinyalidir. CCP modülünden gelen sinyal ile Timer1 içeriği sıfırlanabilir. Diğer sıfırlama metotları TMR1H ve TMR2H kaydedici içeriklerini sıfırlamaz.

TIMER 1 Harici Osilatör Bağlantısı

- Timer1 biriminin ayrıca T1OSI (giriş-input) ve TlOSO (çıkış-output) pin'leri arasına harici osilatör bağlanabilir. Bu harici osilatör 32 Khz ile 200 Khz arasında olabilen düşük güçlü kristal ve kondansatörlü osilatör olabilir. Bu sayede uyku modunda bile Timer1 birimi çalışmaya devam eder. Bu girişler aslında çok daha ideal bir çalışma frekansı olan 32 Khz için tasarlanmıştır. Osilatör kristaline göre kullanılması gereken kondansatörler Tabloda verilmiştir. Osilatörün ilk çalışmada kararlı bir sinyal üretmeye başlaması için gerekli süre program komutları ile verilmelidir.

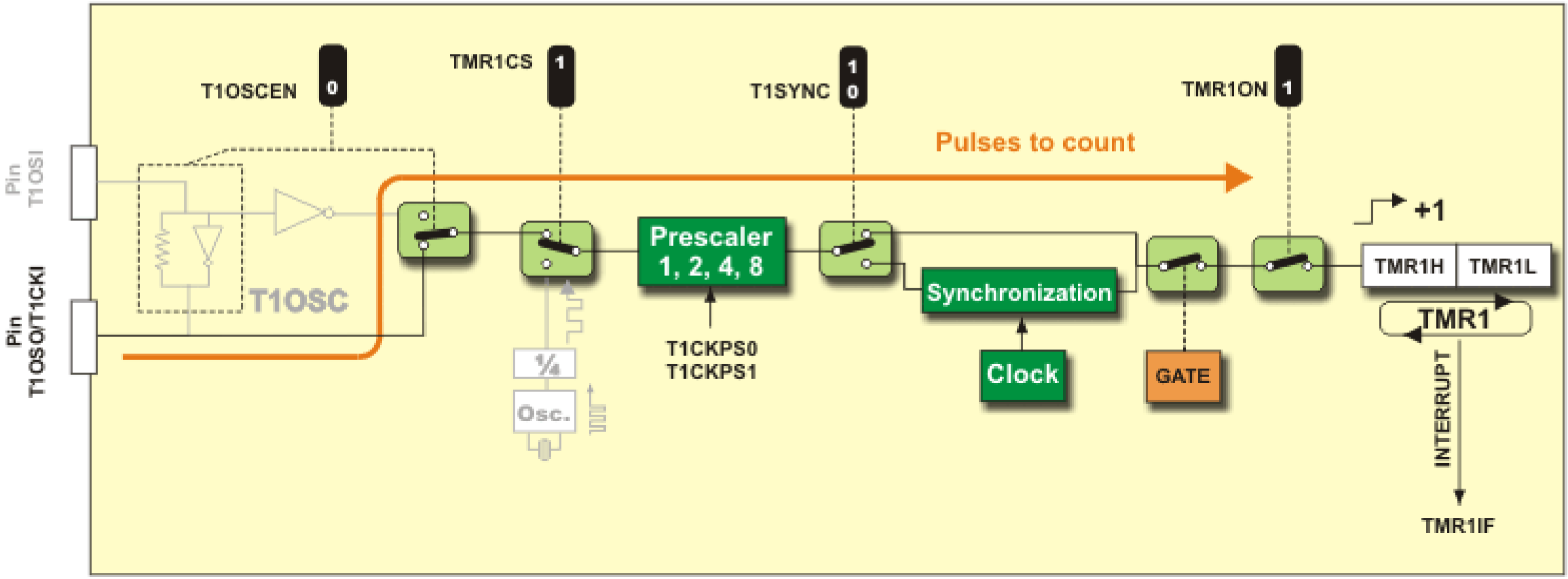


Kristal Frekansı	C1	C2
32 Khz	33 pF	33 pF
100 Khz	33 pF	33 pF
200 Khz	15 pF	15 pF

TIMER1’in Sayaç Modunda Kullanımı

- TIMER1, TMR1CS bitini ayarlayarak sayaç olarak çalışmaya başlar.
- RC0 / T1CKI pinine verilen darbeleri sayar ve harici saat girişi T1CKI'deki darbenin her yükselen kenarında artar. T1CON registerinin kontrol biti T1SYNC temizlenirse, harici saat girişleri TMR1 kayıt çiftine ulaşmadan önce senkronize edilir.
- Başka bir deyişle, Timer1 mikrodenetleyici saati ile senkronize edilir, böylece senkron bir sayaç haline gelir. Timer1 bir sayaç olarak çalışırken mikrodenetleyici uyku moduna ayarlanırsa, giriş pinleri palslarla beslenmiş olsa bile TMR1H ve TMR1L zamanlayıcı kayıtları değiştirilmez. Mikrodenetleyici saati uyku modunda çalışmadığından, senkronizasyon için kullanılacak saat girişi yoktur. Ancak prescaler, sadece basit bir frekans bölücü olduğu için pinlerde saat darbeleri olduğu sürece çalışmaya devam edecektir.

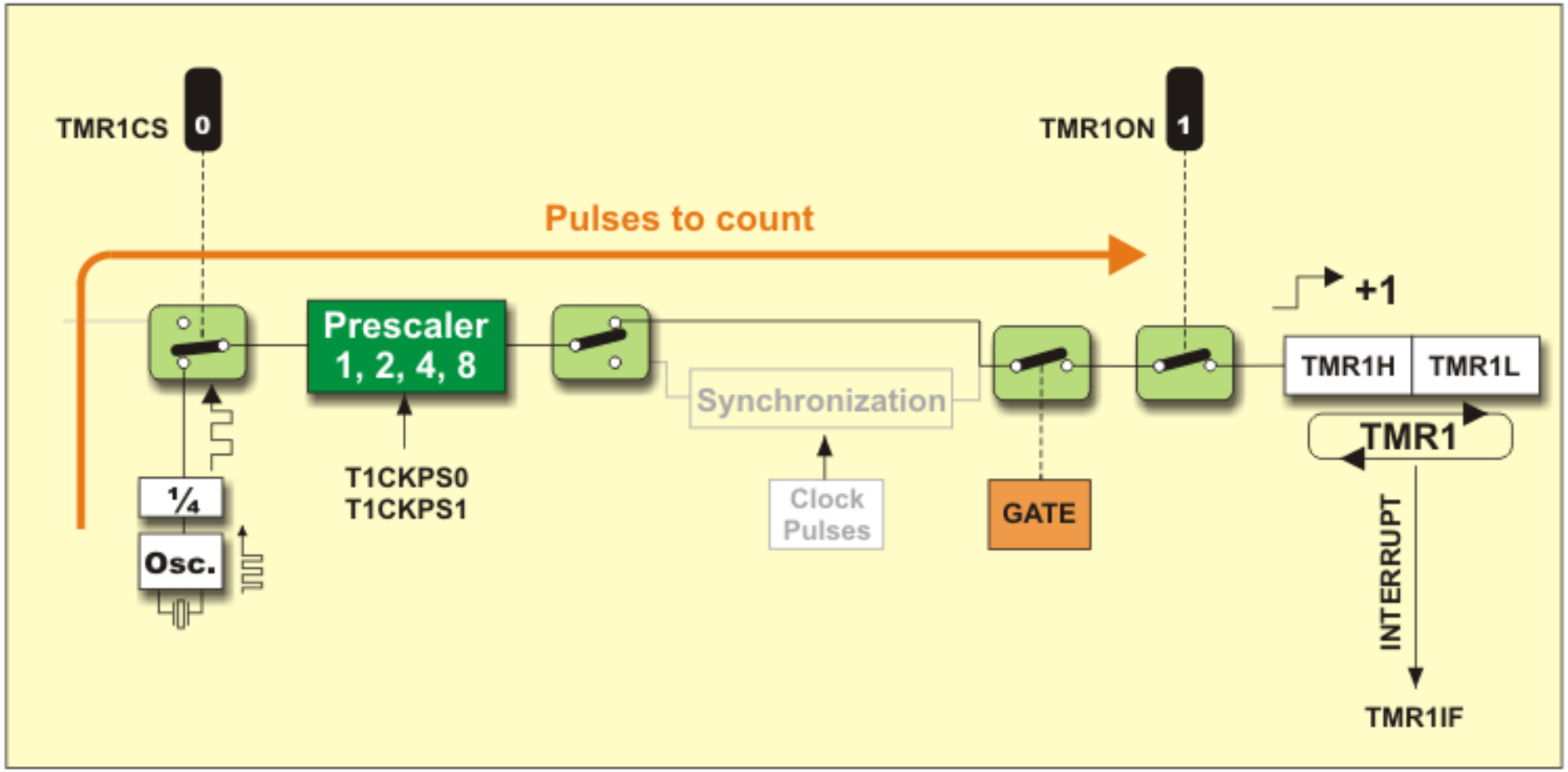
TIMER1'in Sayaç Modunda Kullanımı



TIMER1’in Zamanlayıcı Modunda Kullanımı

- Timer1'i zamanlayıcı moduna ayarlamak için TMR1CS bitini temizlemek gerekir. Bundan sonra, dahili osilatör tarafından üretilen her darbe, 16 bit zamanlayıcı kaydının artmasına neden olur. 4MHz osilatör kullanıldığı kabul edilirse, bu kayıt her mikrosaniyede bir artırılır. Bu modda, T1SYNC biti zamanlayıcıyı etkilemez, çünkü mikrodenetleyici ve diğer dahili modüller tarafından kullanılan dahili saat darbelerini sayar. Dolayısıyla senkronizasyona gerek yoktur.

TIMER1'in Zamanlayıcı Modunda Kullanımı



CCS C'de Timer_1 Kullanım Komutları

- **SETUP_TIMER1 ( ) Fonksiyonu :** Bu komut Timer1 zamanlayıcı ayarlarını yapmaya yarayan bir fonksiyondur.

  setup_timer1 (*mod);*

- Fonksiyondaki *"mod"* kısmına aşağıda belirtilen sabit tanımlamalardan biri veya birden fazlası yazılabilir. İki sabit arasında "|" operatörü konulur. Bu fonksiyonda kullanılacak sabitler denetleyicinin başlık dosyasında (Örneğin, l6F876.h) yazılıdır.
- **Timer1 Kontrol Sabitleri;**
- "T1_DISABLED" =Timer1'i kapat anlamında kullanılır.
- "T1_INTERNAL" = Timer1'in clock sinyalinin kaynağının denetleyici frekansı olacağını belirtir.
- "T1_EXTERNAL"=Timer1'ln clock sinyalinin kaynağının harici olacağını belirtir.
- "T1_EXTERNAL_SYNC" =Timer1 harici sinyal kaynağıyla senkronize olacağını bildirir.
- "T1_CLK_OUT" =Timer1 osilatörünün kullanılacağını bildirir.

CCS C'de Timer_1 Kullanım Komutları

- **Timer1 Zamanlayıcı Bölme Oranları (Prescaler) Sabitleri;**

"T1_DIV _BY_1, T1_DIV _BY_2, T1_DIV _BY_4, T1_DIV _BY_8"

- Sabitin sonundaki rakamlar bölme oranını belirtir.

Örnek;

Setup_timer1 (T1_EXTERNAL | T1_DIV _BY_4)

Örnekte, Timer1 birimi, harici kaynaktan saat sinyali alacak ve bölme oranı 4 olacak şekilde kurulmuştur.

CCS C’de Timer_1 Kullanım Komutları

- **SET_TIMER1 ( ) Fonksiyonu**
- Bu komut Timer1 biriminin saymaya başlama değerini belirlemek için kullanılırlar. Tüm sayıcılar yukarı doğru sayar ve maksimum değere ulaşınca sıfıra döner ve tekrar yukarı saymaya baslar.
- **set_timer1 (*değer);***
- Fonksiyonda *"değer"* kısmına Timer1 biriminin saymaya başlama değeri yazılır. Timer1 birimi 16 bit'lik olduğu için buraya yazılacak değer O ile 65536 arasında olmalıdır.
- Örnek;

set_timer1(764);

komutu ile Timer1 birimi saymaya 764'den itibaren baslar.

CCS C'de Timer_1 Kullanım Komutları

- **Timer1 Kesmesi (#INT_TIMER1)**
- Timer1 istenen değerden saymaya başlar ve 65536 sayısına ulaşınca Timer1 kesme taşması meydana gelir. Timer1 kesmesine izin verme bit'i PIE1 kaydedicisinin 0.bit'i olan TMR1IE bit'idir. **Timer1 kesmesi #int_timer1 komutu ile ifade edilir. Dikkat edilmesi gereken bir nokta vardır. Kesme fonksiyonu içinde mutlaka set_timer1 ( ) komutu ile programda set edilen değerin tekrar kesme fonksiyonu içinde de belirtilmesi gerekliliğidir.**
- Çünkü program kesme fonksiyonuna gittiğinde, Timer1 taşmış yani 0x0000 değerine gelmiş demektir. Timer1 biriminin tekrar başlangıçta istenen değer ile saymaya başlanması isteniyorsa kesme fonksiyonunda mutlaka tekrardan set_timer1( ) komutu ile set değeri belirtilmelidir.

CCS C’de Timer_1 Kullanım Komutları

- **Timer1 Kesmesi (#INT_TIMER1)**
- Timer1 kesme süresi aşağıdaki formül ile hesaplanabilir.

- **Kesme Süresi =Tosc x (Bölme Oranı) x (65536 – TMR1'a yazılan değer)**

- Tosc periyodu; *Tosc= 1/fosc* formülünden bulunur.
- fosc değeri bilindiği gibi PIC denetleyiciye bağlanan osilatör frekansının 4'e bölümü ile elde edilir.
- fosc = Denetleyici Osilatör Frekansı/4

Örnek Programlar (tmr0_zamanlayıcı)

```
#include <16f877.h>
#fuses XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD
#use delay (clock=4000000)
#use fast_io(b)
int i=0, z;
const int digit[16]={0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7C, 0x07, 0x7F, 0x6F
#int_timer0
void  timer0_kesme ()
{    set_timer0(60);
     i++;
     if (i==10)
        output_high(pin_b0);
     if (i==20)
        {
           output_low(pin_b0);
           i=0;
        }
}
```

Örnek Programlar (tmr0_zamanlayıcı)

```
/********** ANA PROGRAM FONKSİYONU********/
void main ( )
{
   setup_psp(PSP_DISABLED);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
   setup_CCP1(CCP_OFF);
   setup_CCP2(CCP_OFF);

   set_tris_b(0x00);
   set_tris_d(0x00);
   output_b(0x00);
```

Örnek Programlar (tmr0_zamanlayıcı)

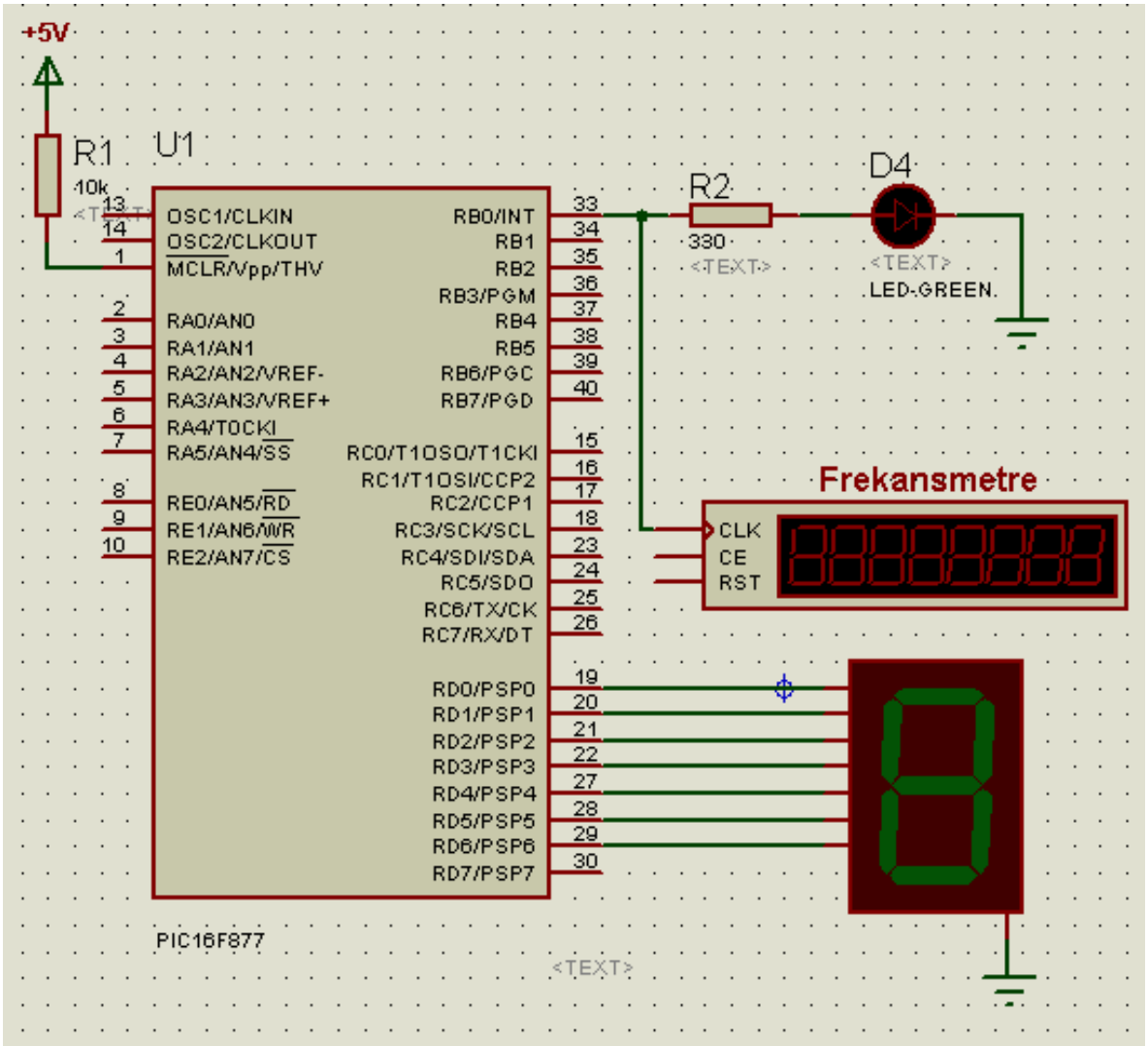
```
    output_b(0x00);

    setup_timer_0(RTCC_INTERNAL | RTCC_DIV_256);
    set_timer0(60);

    enable_interrupts(INT_timer0);
    enable_interrupts(GLOBAL);

    while(1)
    {
       for(z=0;z<=15;z++)
       {
          output_d(digit[z]);
          delay_ms(100);
       }
    }
}
```

Örnek Programlar (tmr0_zamanlayıcı)



Örnek Programlar (tmr0_sayıcı)

```
#include <16f877.h>
#fuses XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD
#use delay (clock=4000000)
#use fast_io(a)
#use fast_io(b)
#use fast_io(d)
int sayi=0;
int i,led;
//******************** Timer0 Kesmesi ********************************
#int_timer0
void  timer0_kesme ()
{       set_timer0(252);
        sayi++;
        output_b(sayi);
    if (sayi==15)
       sayi=0;
}
/********** ANA PROGRAM FONKSİYONU********/
```

Örnek Programlar (tmr0_sayıcı)

```
/********* ANA PROGRAM FONKSİYONU********/
void main ( )
{  setup_psp(PSP_DISABLED);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
   setup_CCP1(CCP_OFF);
   setup_CCP2(CCP_OFF);

   set_tris_a(0x10);
   set_tris_b(0x00);
   set_tris_b(0x00);

   output_b(0x00);

   setup_timer_0(RTCC_EXT_H_TO_L | RTCC_DIV_2);
   set_timer0(252);
```

Örnek Programlar (tmr0_sayıcı)

```
setup_timer_0(RTCC_EXT_H_TO_L | RTCC_DIV_2);
set_timer0(252);

enable_interrupts(INT_timer0);
enable_interrupts(GLOBAL);

while(1)
{
 led=1;

   for(i=0;i<=7;i++)
   {
      output_d(led);
      led=led<<1;
      delay_ms(50);
   }

   led=0x80;
```

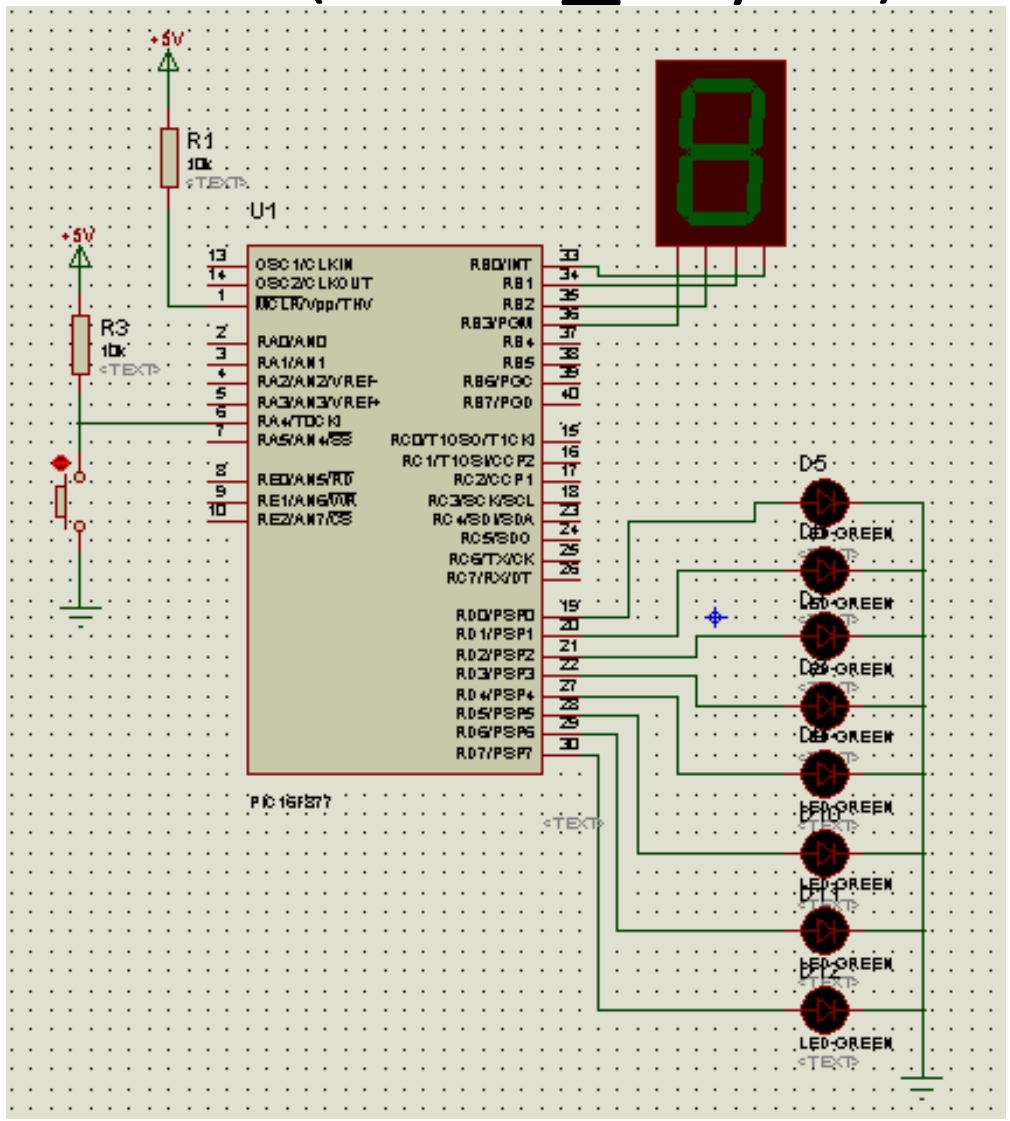
Örnek Programlar (tmr0_sayıcı)

```
        {
            output_d(led);
            led=led<<1;
            delay_ms(50);
        }

        led=0x80;
        output_d(led);
        delay_ms(50);

         for(i=0;i<=6;i++)
        {
            led=led>>1;
            output_d(led);
            delay_ms(50);
        }
    }
}
```

Örnek Programlar (tmr0_sayıcı)



Kaynaklar

- CCS C Programlama Kitabı, Serdar Çiçek, Altaş Yayıncılık
- Mikroelektronika C programlama e-kitabı «https://www.mikroe.com/ebooks/pic-microcontrollers-programming-in-c»