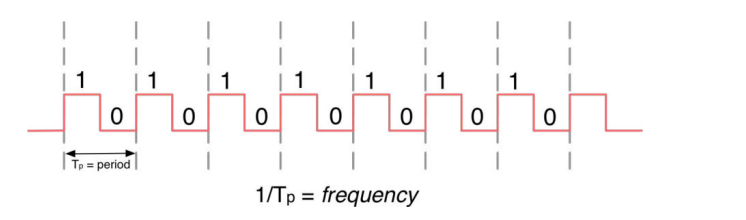
**STM32FXX TIMER**

**CLOCK(Saat):**

Saat, periyodik sinyaller üreten bir cihazdır. Bir bakıma mikrodenetleyicinin kalbidir. Bununla birlikte saat sinyali mikrodenetleyici de sadece tüm bileşenleri ve çevre birimlerini beslemek için kullanılamaz. Ayrıca, güç tüketimi, belirli bir çevre biriminin saat hızı ile doğrudan bağlantılı kritik bir husustur. Bazı MCU parçalarının saat hızını seçici olarak devre dışı bırakma veya azaltma yeteneğine sahip olmak, genel cihaz güç tüketimini optimize etmeyi sağlar.Bu, saatin hiyerarşik bir yapıda düzenlenmesini gerektirir ve geliştiriciye farklı hızlar ve saat kaynakları seçme olanağı verir.

Saat, genellikle % 50 görev döngüsüyle(duty cycle) kare dalga sinyali üreten bir cihazdır.



Bir tam dalganın oluşması için geçen süreye periyot denir. Periyot Tp harfi ile gösterilir. Birim zamanda oluşan dalga sayısına da frekans denir. Frekans f harfi ile gösterilir. Frekans Hertz olarak ifade edilir. Periyot ile frekans arasında T = 1/f veya f = 1/T şeklinde bir bağıntı vardır.

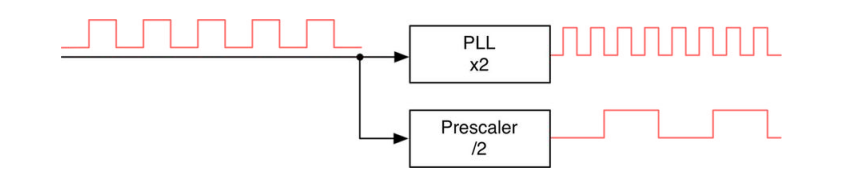
STM32 MCU'nın çoğunluğu alternatif olarak iki ayrı saat kaynağı bulunur: dahili bir RC osilatörü (Yüksek Hızlı Dahili (HSI) olarak adlandırılır) veya harici bir özel kristal osilatör (Yüksek Hızlı Harici (HSE) olarak adlandırılır). Genellikle harici kristal osilatör tercih edilir. Bunun sebebi:

• Harici bir kristal, özellikle PCB çalışma sıcaklıkları 25 ° C ortam sıcaklığından uzak olduğunda,% 1 doğrulukta rated derecelendirilen dahili RC şebekesine kıyasla daha yüksek bir hassasiyet sunar.

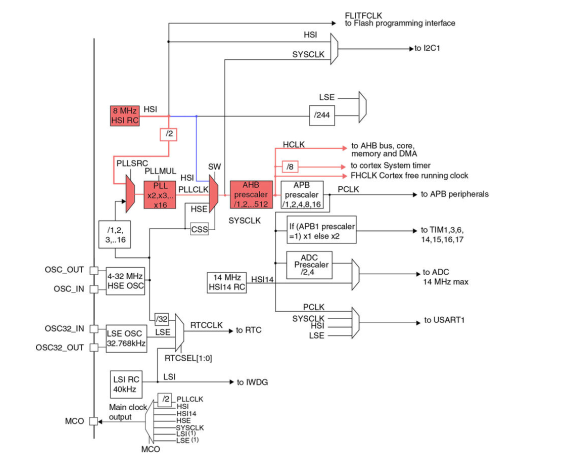
• Bazı çevre birimleri, özellikle yüksek hızlı olanlar, yalnızca belirli bir frekansta çalışan özel bir harici kristal tarafından saatlendirilebilir.

Yüksek hızlı osilatör ile birlikte, düşük hızlı osilatörde başka bir saat kaynağı olarak kullanılabilir, bu da harici bir kristal (Düşük Hız Harici (LSE) olarak adlandırılır) veya dahili özel RC osilatör (Düşük Dahili Hız (LSI)). Düşük hızlı osilatör, Gerçek Zamanlı Saat (RTC) ve Bağımsız Gözcü (IWDT) çevre birimini çalıştırmak için kullanılır.

Yüksek hızlı osilatörün frekansı, Cortex-M çekirdeğinin ne de diğer çevre birimlerinin gerçek frekansını oluşturmaz. Saat ağacı olarak da adlandırılan karmaşık bir dağıtım ağı, saat sinyalinin bir STM32 MCU içine yayılmasından sorumludur. Programlanabilir Faz Kiliti Döngüsü(PLL>>Phase-Locked Loops) ve ön ölçekleyiciler kullanarak, ulaşmak istediğimiz performanslara, belirli bir çevre birimi veya veri yolu için maksimum hıza ulaşılır. İhtiyaç duyulan kaynak frekansını arttırmak /azaltmak mümkündür.



**PLL Method**



**Clock Tree**

Saat ağacı yapılandırması Sıfırla ve Saat Kontrolü (RCC) adlı özel bir çevre birimi aracılığıyla gerçekleştirilir ve esas olarak üç adımdan oluşan bir işlemdir:

1. HSE kullanılıyorsa, yüksek hızlı osilatör kaynağı seçilir (HSI veya HSE) ve uygun şekilde yapılandırılır.

2. SYSCLK'yi yüksek hızlı osilatör tarafından sağlanan frekanstan daha yüksek bir frekansla beslemek istiyorsak, ana PLL'yi (PLLCLK sinyali sağlayan) yapılandırmamız gerekir. Aksi takdirde bu adımı atlayabiliriz.

3. Ardından, yüksek hızlı saatin (HCLK - çekirdeği, DMA'ları ve AHB veriyolunu besleyen) istenen frekansa ve gelişmiş frekanslarına ulaşmak için doğru AHB, APB1 ve APB2 (varsa) ön ölçekleyici ayarlarını seçiyoruz. Belirli bir STM32 mikrodenetleyici için PLL'ler ve ön ölçeklerin sadece bazı kombinasyonlar geçerlidir ve bunların yanlış yapılandırılması potansiyel olarak MCU'ya zarar verebilir veya en azından arızalara neden olabilir (yanlış bir saat yapılandırması anormal davranışlara, garip ve öngörülemeyen sıfırlamalara vb. yol açabilir).

**Timer nedir ?**

Timer , basit anlamda bir sayıcıdır(sayaç). Belirli bir sayıdan aşağı birer birer düşer yada istediğimiz noktaya çıkartabiliriz. Buna ek olarak aralıklarıda seçebiliriz ve uygulayabiliriz. Bu sayma işlemini clock hızına göre yapabildikleri gibi dışarıdan verilen kesmelerle yönetilebilmektedir. Sayma işlemini işlemcimizin clock hızına göre yapmaktadır.

Genellikle, bir zamanlayıcı sıfırdan belirli bir değere kadar sayar, bu da çözünürlüğü için maksimum işaretsiz değerden daha yüksek olamaz (örneğin, sayaç 65535'e ulaştığında 16 bitlik bir zamanlayıcı taşar), ancak tam tersi de sayılabilir.

Bir STM32 mikrodenetleyicideki zamanlayıcıların çeşitli özellikleri vardır:

• Zamanlayıcı olarak kullanılabilirler (tüm STM32 zamanlayıcılarında ortak olan özelliktir).

• Harici bir olayın frekansını ölçmek için kullanılabilirler (giriş yakalama modu). (input capture mode)

• Bir çıkış dalga formunu kontrol etmek veya bir sürenin ne zaman geçtiğini belirtmek için (çıkış karşılaştırma modu) (output compare mode).

• Her kanalda bağımsız olarak kenara hizalı veya merkez hizalı modda PWM sinyalleri oluşturmak için (PWM modu).

**Temel zamanlayıcılar:** bu kategorideki zamanlayıcılar, STM32 MCU'lardaki en basit zamanlayıcılardır. Zaman tabanı üreteci olarak kullanılan 16 bitlik zamanlayıcılardır ve çıkış / giriş pinleri yoktur. Temel zamanlayıcılar, DAC çevre birimini beslemek için de kullanılır, çünkü güncelleme olayları DAC için DMA isteklerini tetikleyebilir (bu nedenle genellikle en az bir DAC sağlayan STM32 MCU'larında kullanılabilir). Temel zamanlayıcılar diğer zamanlayıcılar için “master” olarak da kullanılabilir.

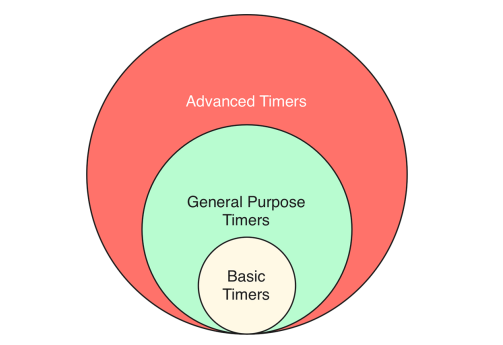
>>>(TIM6 and TIM7)

**Genel amaçlı zamanlayıcılar:** Modern bir gömülü mikrodenetleyicinin bir zamanlayıcısının uygulaması beklenen klasik özellikleri sağlayan 16/32-bit zamanlayıcılardır. Çıkış karşılaştırması (zamanlama ve gecikme üretimi), Tek Darbe Modu, giriş yakalama (harici sinyal frekansı ölçümü için), sensör arayüzü (encoder, hall sensor) vb. için herhangi bir uygulamada kullanılırlar. Açıkçası, genel amaçlı bir zamanlayıcı olabilir. Temel bir zamanlayıcı gibi zaman tabanı üreticisi olarak kullanılır. Bu kategorideki zamanlayıcılar dört programlanabilir giriş / çıkış kanalı sağlar.

>>>(TIM2, TIM3, TIM4, TIM5, TIM12, TIM13, TIM15, TIM16 and TIM17)

**Gelişmiş zamanlayıcılar:** bu zamanlayıcılar bir STM32 MCU'daki en eksiksiz zamanlayıcılardır. Genel amaçlı bir zamanlayıcıda bulunan özelliklere ek olarak, motor kontrolü ve dijital güç dönüştürme uygulamalarında kullanılır.

>>>(TIM1)



**Timer Önemli Terimler:**

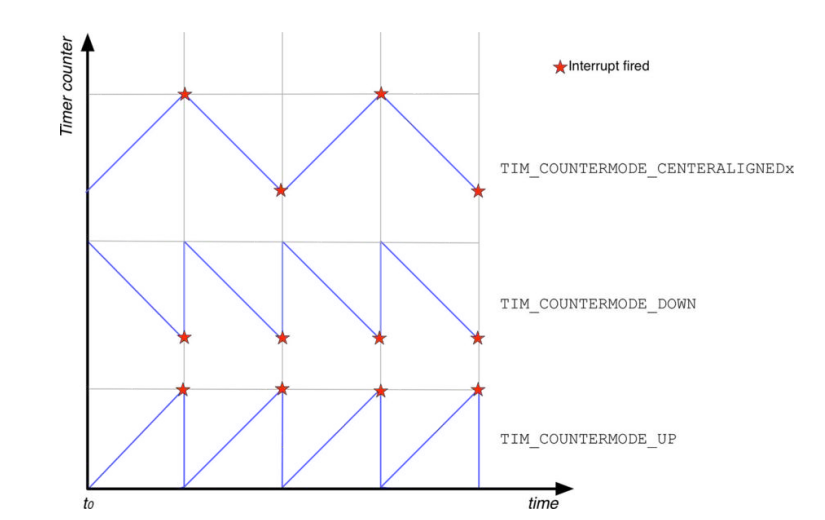
• Prescaler:(Ön ölçekleyici) Zamanlayıcı saatini 1 ile 65535 arasında değişen bir faktöre böler (bu, ön ölçekleyici kaydının 16 bit çözünürlüğe sahip olduğu anlamına gelir). Örneğin, zamanlayıcının bağlı olduğu veri yolu 24MHz'de çalışıyorsa, 24'e eşit bir ön ölçekleyici değeri sayım frekansını 1MHz'e düşürür.

• CounterMode: Zamanlayıcının sayım yönünü tanımlar ve aşağıdaki değerlerden birini alabilir. Bazı sayma modları yalnızca genel amaçlı ve gelişmiş zamanlayıcılarda kullanılabilir. Temel zamanlayıcılar için yalnızca TIM\_COUNTERMODE\_UP tanımlanır.

a) TIM\_COUNTERMODE\_UP

b) TIM\_COUNTERMODE\_DOWN

c) TIM\_COUNTERMODE\_CENTERALIGNED



• Period(Periyot): Tekrar sayacı yeniden başlatmadan önce zamanlayıcı sayacı için maksimum değeri ayarlar. Bu, 16 bitlik zamanlayıcılar için 0x1 ila 0xFFFF (65535) ve 32 bitlik zamanlayıcılar olarak uygulayan MCU'larda TIM2 ve TIM5 zamanlayıcılar için 0x1 ila 0xFFFF FFFF arasında bir değer olduğunu varsayabilir. Dönem 0x0 olarak ayarlanırsa, zamanlayıcı başlamaz. Auto-reload register (ARR) olarak da adlandırılır.

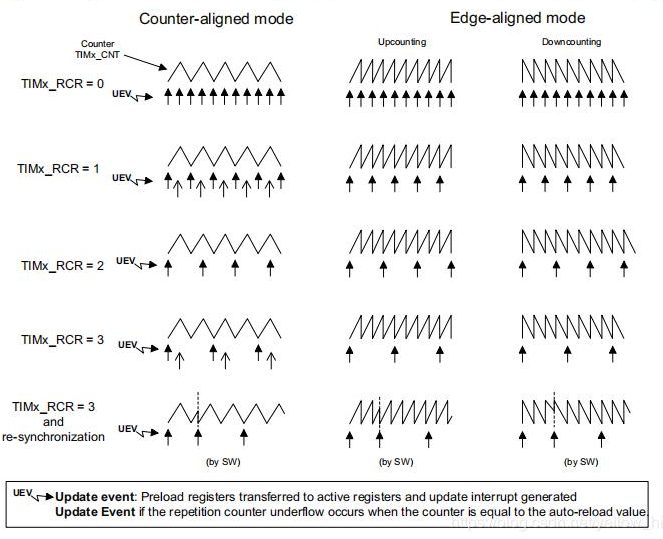
• ClockDivision: Bu bit alanı, dahili zamanlayıcı saat frekansı ile ETRx ve TIx pinlerindeki dijital filtreler tarafından kullanılan örnekleme saati arasındaki ayrım oranını gösterir. Aşağıdaki bir değeri alabilir ve yalnızca genel amaçlı ve gelişmiş zamanlayıcılarda kullanılabilir.

a) TIM\_CLOCKDIVISION\_DIV1

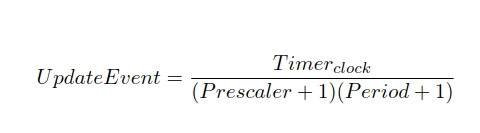
b) TIM\_CLOCKDIVISION\_DIV2

c) TIM\_CLOCKDIVISION\_DIV4

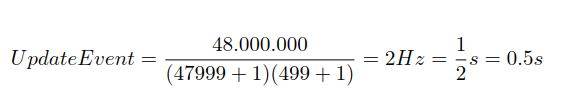
• RepetitionCounter(Tekrarlama Sayacı): Her zamanlayıcının taşma durumunu izleyen özel bir güncelleme kaydı vardır. Bu, daha sonra göreceğimiz gibi belirli bir IRQ da oluşturabilir. RepetitionCounter, güncelleme kaydı ayarlanmadan ve karşılık gelen olayın yükseltilmesinden önce (etkinse) zamanlayıcının kaç kez taştığını söyler. RepetitionCounter yalnızca gelişmiş zamanlayıcılarda kullanılabilir.(advanced timer)



**Timer Update Event Hesaplanması:**

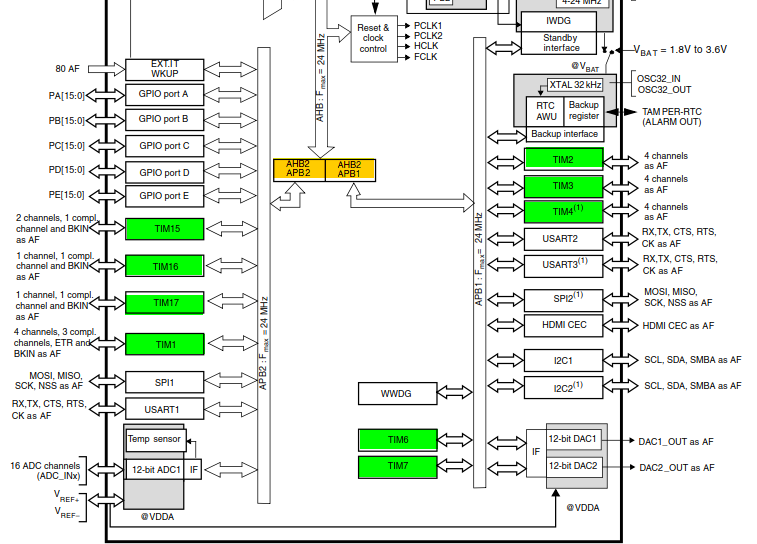


**FORMÜL**



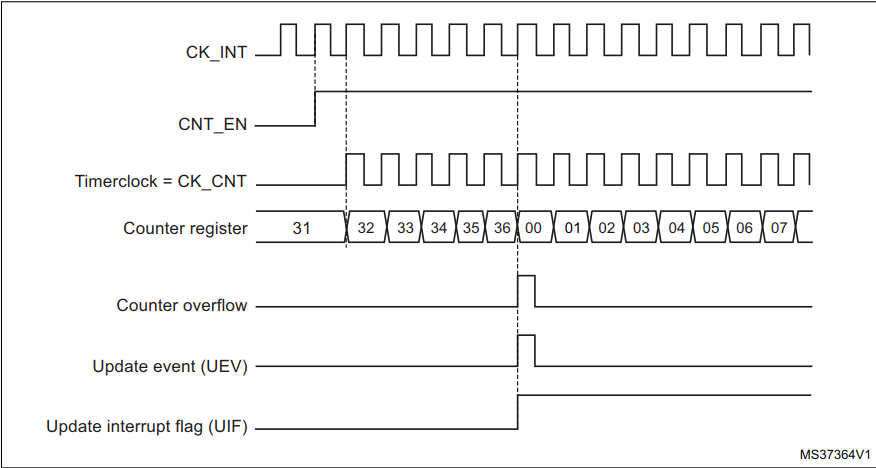
Hert to Second Calculator Link: <https://www.unitjuggler.com/convert-frequency-from-Hz-to-s(p).html>

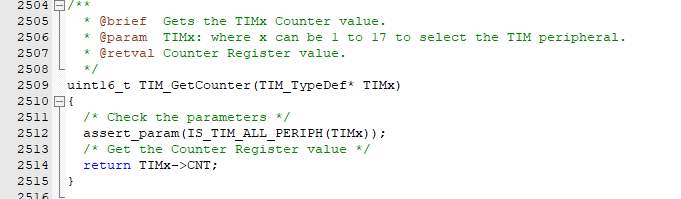
Update Event Calculator: <https://www.mikroe.com/timer-calculator>



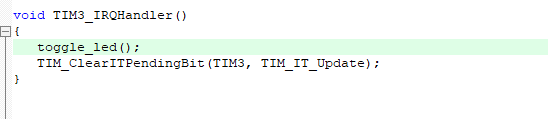
**1)COUNTER MODE:**

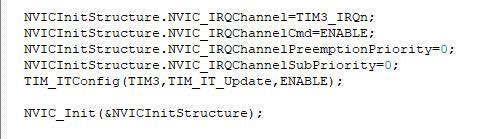
Sayaç 0'dan otomatik yeniden yükleme değerine (TIMx\_ARR kaydının içeriği) kadar sayar, ardından 0'dan yeniden başlar ve bir sayaç taşması olayı oluşturur.



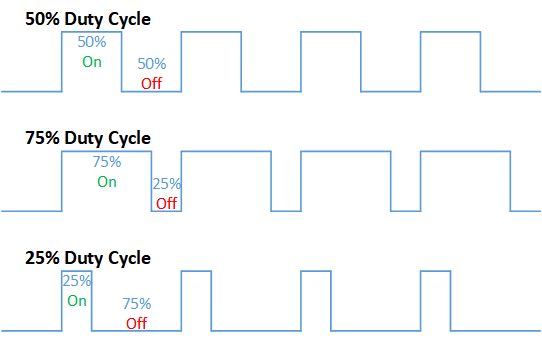


**2)INTERRUPT MODE:**



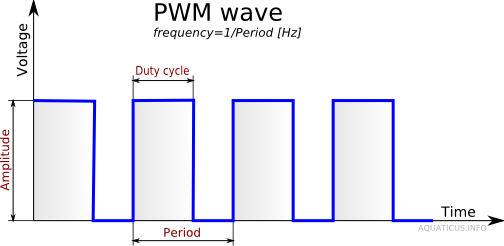


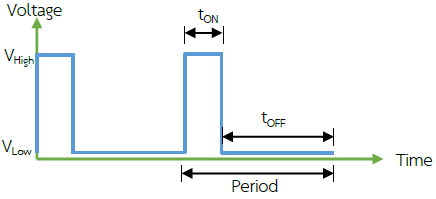
**3)PWM Mode(Output Compare):**

**Pulse Width Modulation (Darbe-Pals Genişlik Modülasyonu ):**

PWM sinyalin genişliklerini kontrol ederek, çıkışta üretilmek istenen sinyalin elde edilmesi tekniğidir.

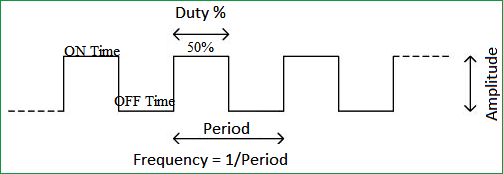
Bu kontrolde tamamen anahtarlama ile sağlanır. Anahtarlama(transistor) ne kadar hızlı yapılırsa, PWM ile aktarılan sinyalin gücü o kadar da artar.

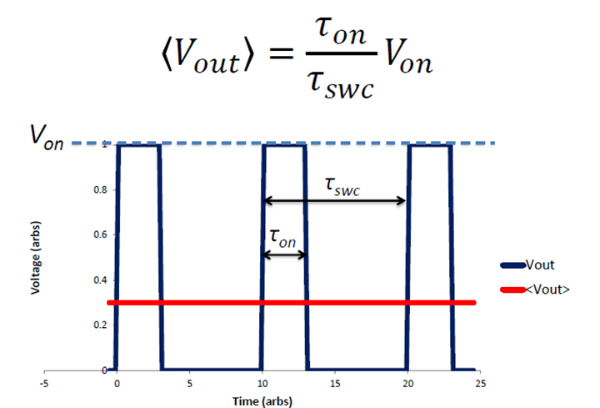


****

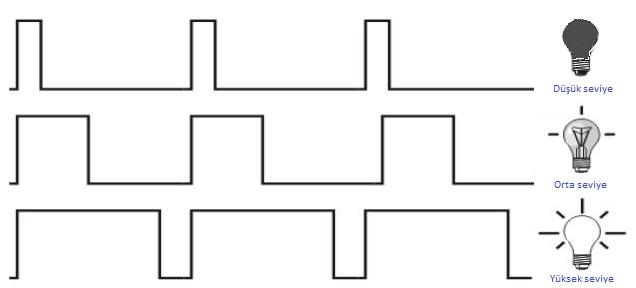
**Duty Cycle(Görev Döngüsü) Formula:**

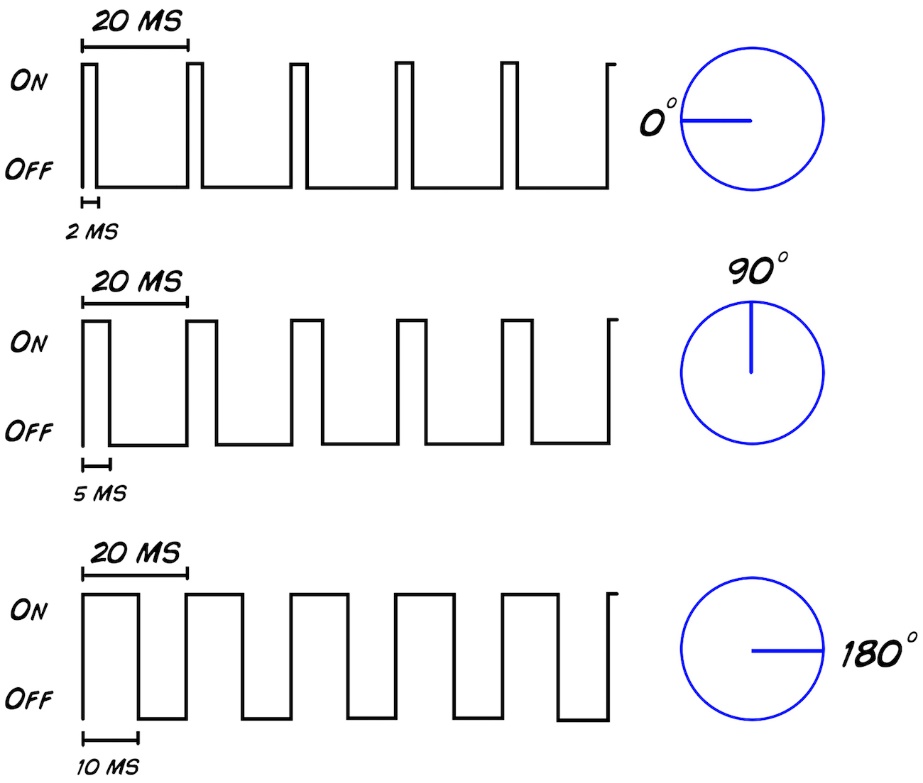
Görev döngüsü yapılan işlemin periyodunu belirtir.

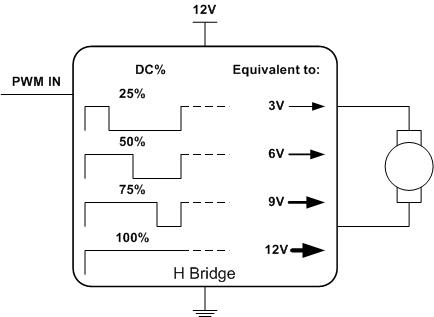


**Output Voltage(Vout) Formula(Çıkış Voltajı):**

**PWM Uygulama Alanları:**

1)Ledin Parlaklığını Kontrol Edebiliriz.

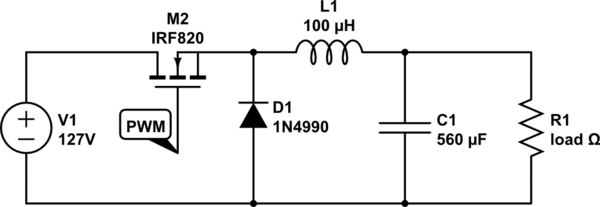
2)Servo Motor Kontrolü Yapabiliriz.

3) Motor Hız Kontrolü Yapabiliriz.

4) Haberleşme Sistemleri:

Telekomünikasyon sistemlerinde sinyallerin bozulmadan taşınmasında PWM tekniği kullanılmaktadır.

5) Voltaj Regülatör Devreleri:(Buck ve Boost Converter)



6) Ses Efektleri ve Yükselticiler:

Osilatörler ile birlikte bazı ses efektlerini elde etmek için PWM tekniği kullanılabilir.

**PWM Advantages:**

1)Güç tüketimi az.

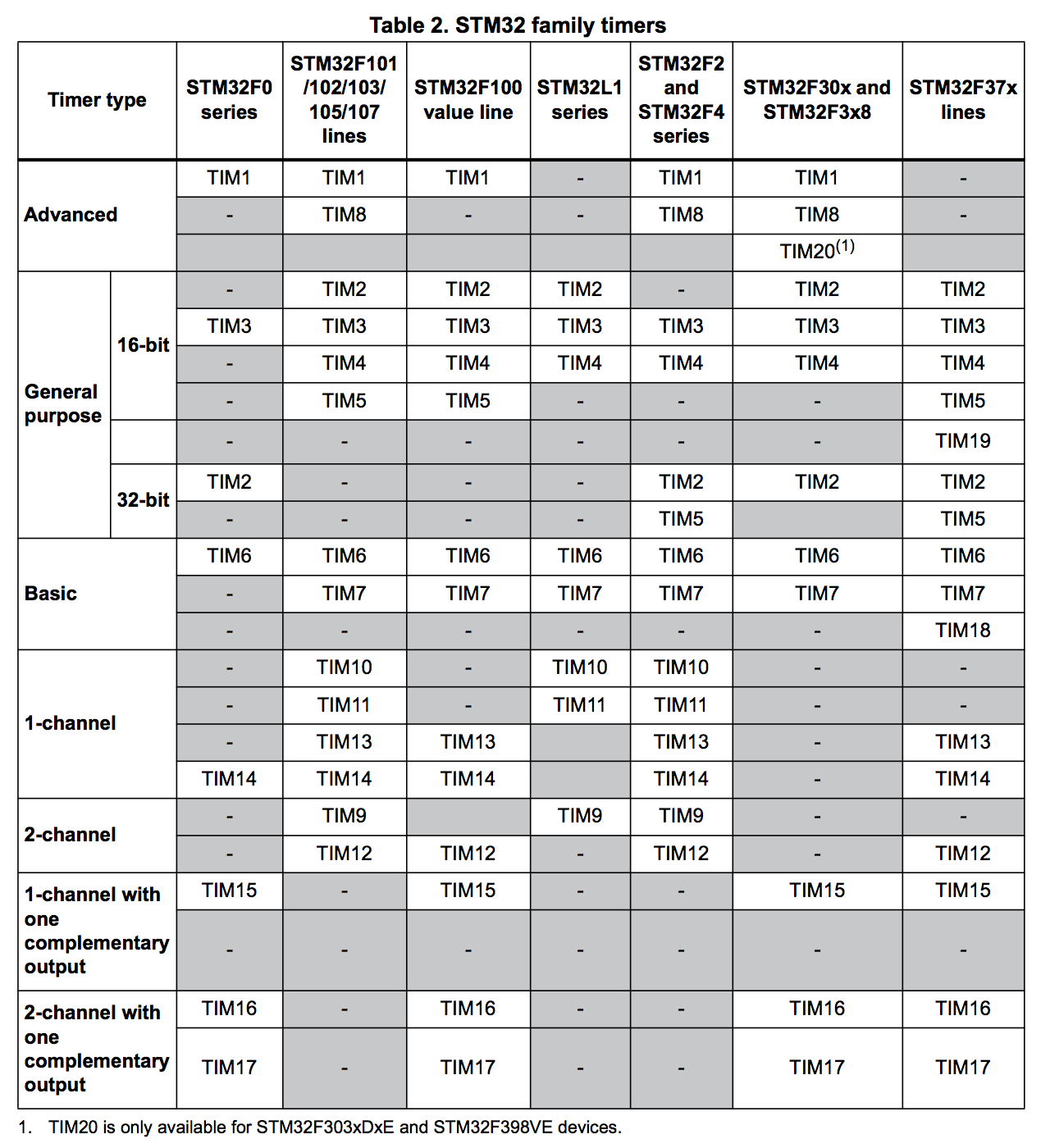
2)Sinyalleri uzun mesafelere bozulmadan taşımak için yüksek verimli.

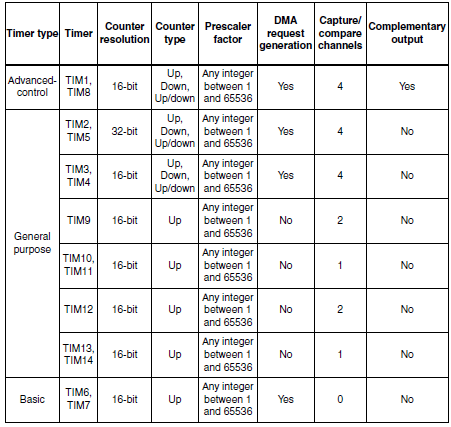
3)Uygulaması kolay.

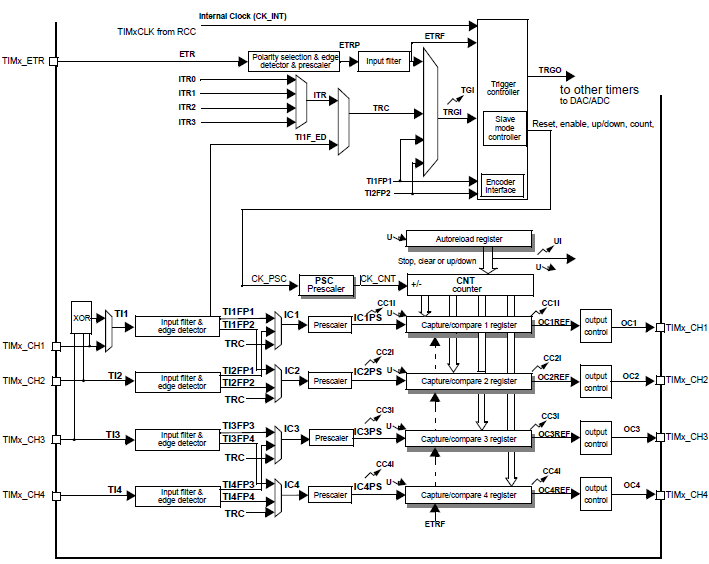
**STM32F Kartımız için PWM Frekans ve Duty Cycle Hesaplama:**

**Frekans Hesaplama:**

**Tim Pulse Hesaplama:**

****





• **OCMode:** Çıktı karşılaştırma modunu belirler .

• **Pulse :** Bu alanın içeriği CCRx kaydının içinde saklanır ve çıktı tetiklemek için kullanılır.

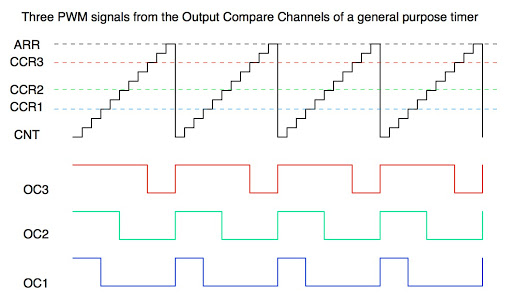
• **OCPolarity:** CCRx kaydı CNT ile eşleştiğinde çıkış kanalı polaritesini tanımlar.

• **OCNPolarity:** Tamamlayıcı çıkış polaritesini tanımlar. Yalnızca TIM1 ve TIM8 gelişmiş zamanlayıcılarında bulunan ve ek özel kanallarda tamamlayıcı sinyaller oluşturmaya izin veren bir moddur. Bu özelliği özellikle motor kontrol uygulamaları için tasarlanmıştır.

• **OCFastMode:** Hızlı mod durumunu belirtir. Bu parametre yalnızca PWM1 ve PWM2 modunda geçerlidir ve TIM\_OCFAST\_DISABLE ve TIM\_OCFAST\_ENABLE değerlerini alabilir.

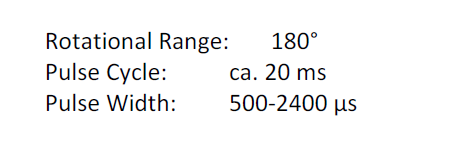
• **OCIdleState:** Zamanlayıcı boşta durumu sırasında kanal çıkış karşılaştırması pin durumunu belirtir. TIM\_OCIDLESTATE\_SET ve TIM\_OCIDLESTATE\_RESET değerlerini varsayabilir. Bu parametre yalnızca TIM1 ve TIM8 gelişmiş zamanlayıcılarında kullanılabilir.

**• OCNIdleState:** Zamanlayıcı boşta durumu sırasında tamamlayıcı kanal çıkışının pin durumunu karşılaştırmasını belirtir. TIM\_OCNIDLESTATE\_SET ve TIM\_OCNIDLESTATE\_RESET değerlerini alabilir. Bu parametre yalnızca TIM1 ve TIM8 gelişmiş zamanlayıcılarında kullanılabilir.



**Servo Motor:**

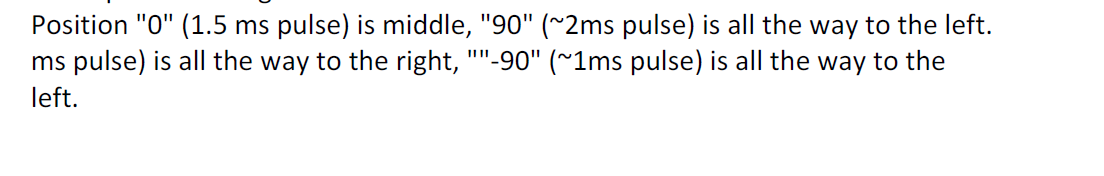
****



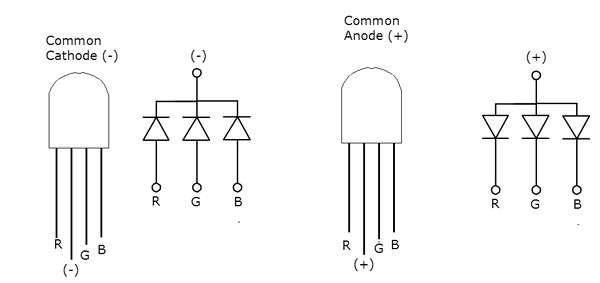
**Frekans:**

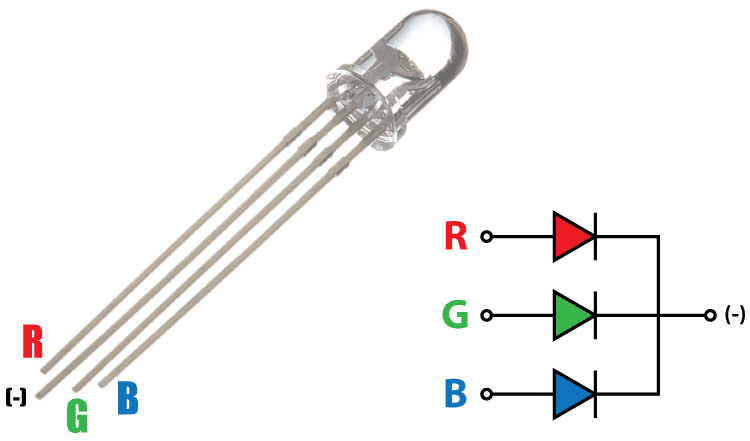
**Period:**

**Tim Pulse:**



**RedGreenBlue LED:**





**SysTick Timer:**

SysTick, tüm STM32 mikrodenetleyicileri tarafından sağlanan Cortex-M çekirdeğinin içinde yer alan özel bir zamanlayıcıdır. Diğer zamanlayıcıların input capture / output compare gibi özellikleri olduğu için, bu zamanlayıcıları sıradan zamanlama için kullanmak, işlemci kaynaklarını boşa harcamak anlamına gelmektedir.

Delay foksiyonlarında Systick timer kullanmak işlemcimizi yormaz, güç tüketimini azaltır.

