28.03.2025

SİVAS CUMHURİYET ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
GÖMÜLÜ SİSTEMLER DERSİ
LABORATUVAR FÖYÜ

Bölüm 4

TIMERS and COUNTERS - 1

(Zamanlayıcı ve Sayıcılar - 1)

Dersin Hocası: Doç. Dr. Ahmet Gürkan YÜKSEK

Dersin Asistanı: Arş. Gör. Ahmet Utku ELİK

İçindekiler

Öğrencinin;

Grup No :

Ad Soyad :

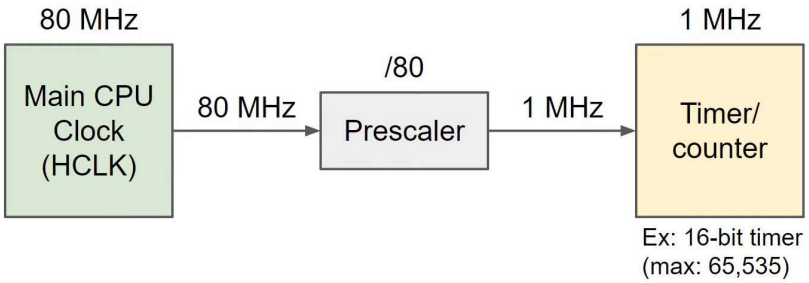
Okul No :

Teslim Tarihi :

1. ZAMANLAYICI ve SAYICILAR

Zamanlayıcı (timer), belirli zaman aralıklarını ölçmek veya zaman bazlı işlemler gerçekleştirmek için kullanılan özelleşmiş bir elektronik saattir. Mikrodenetleyici içinde bir timer genellikle donanım sayacı şeklinde uygulanır ve her clock (saat) darbesinde değeri artan/azalan bir sayaca sahiptir. Örneğin 16 bitlik bir timer 0'dan 65.535'e kadar sayıp taşar (overflow yapar) ve tekrar başa döner. Bir timer sıfırdan yukarı doğru sayarak geçen süreyi ölçebilir (stoptimer gibi) ya da belirli bir süre sonunda kesme üretmek için kullanılabilir.

Sayıcı (counter) ise, bir olay sayacıdır; mikrodenetleyici dışından gelen sinyalleri saymak için kullanılır. Özetle, timer dahili clock frekansını kullanır ve zamanı ölçmek/delay (gecikme) oluşturmak için, counter ise harici bir sinyalin palslerini ve olayları saymak için kullanılır.



Şekil 1. Basit Zamanlayıcı/Sayıcı Yapısı

Timer ile counter kavramları genellikle aynı donanım biriminin farklı çalışma şekilleridir. Mikrodenetleyicilerdeki timer modülleri hem zamanlayıcı hem de sayıcı olarak yapılandırılabildikleri için literatürde genellikle Timer/Counter şeklinde birlikte anılırlar.

1.2. Zamanlayıcı/Sayıcı Temel Bileşenleri

Timer/Counter çevre bileşeni temel olarak Saat Kaynağı (Clock Source), Ön Bölücü (Prescaler, PSC), Sayaç (Counter, CNT) ve Otomatik Yükleme Kaydedicisi (Auto Reload Register, ARR)'nden oluşur.

1.2.1. Saat Kaynağı (Clock Source)

Timer/Counter biriminin sayım yapabilmesi için bir saat (clock) sinyaline ihtiyaç duyar. İşte bu sinyali sağlayan kaynağa clock source denir. Saat kaynağı dahili/harici olarak seçilebilir. Dahili Saat (Internal Clock), mikrodenetleyicilerin sistem saatinden (APB1/APB2 bus clock gibi) türetilir. Harici Saat (External Clock) ise Timer giriş pinine (örneğin ETR) bağlanan bir sinyal kullanılır. Sayıcı (counter) modlarında, dış olayların (örneğin buton basımı, sinyal darbesi) sayılması için harici saat sinyali kullanılır.

1.2.2. Ön Bölücü (Prescaler, PSC)

Prescaler, saat kaynağının frekansını bölerek zamanlayıcının çalışma frekansını düşürür. Bu sayede sayacın daha geniş zaman aralıklarında saymasını sağlar. Timer/Counter çevre birimine gelen saat frekansını, Prescaler kaydedicisine yüklenen değerin bir fazlası kadar saat frekansını düşürmektedir.

$$f_{CL\_CNT} = \frac{f_{CL\_PSC}}{PSC + 1}$$

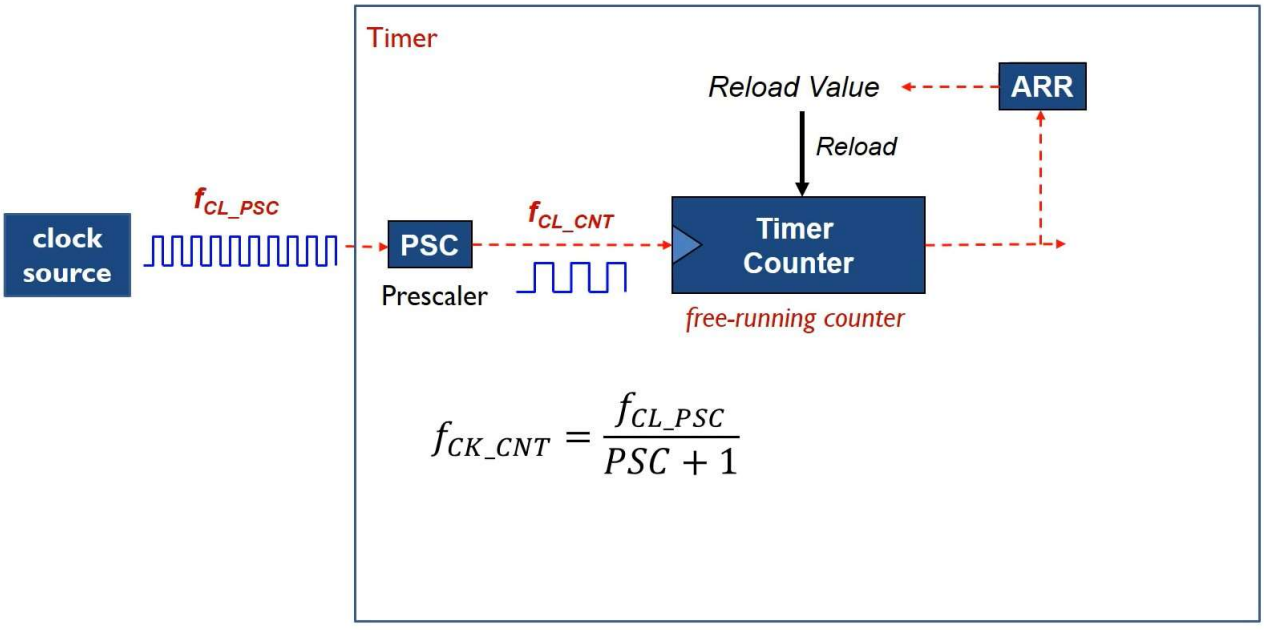
1.2.3. Sayıcı (Counter, CNT)

Timer biriminin sayaç değerini saklayan kayıt alanıdır. Her gelen clock darbesinde mevcut sayaç değeri bir birim artar veya azalır. 16-Bitlik bir sayıcı 0 - 65.535 aralığında sayma işlemi yapar. Örneğin yukarı sayan bir sayaç sınır değeri olan 65.535'i geçme durumunda veya önceden belirlenen Otomatik Yükleme Kaydedicisi (ARR) değerine ulaşma durumunda sıfırlanır ve gelen clock sinyalleri ile sayma işlemine devam eder.

1.2.4. Otomatik Yükleme Kaydedicisi (Auto Reload Register, ARR)

ARR kaydedicisi Sayıcının ulaşacağı maksimum veya minimum değeri belirlemektedir. Counter değeri bu değere ulaştığında taşma (overflow) olarak adlandırılan bir kesme isteği gerçekleşir ve sayaç değeri sıfırlanır. ARR değeri ile zamanlayıcı periyodu ve PWM sinyallerinin görev süresi (duty cycle) değerleri belirlenmektedir.

Bütün bu temel bileşenler bir araya gelerek Timer/Counter çevre biriminin temel yapısını oluşturmaktadır. Bu yapı aşağıdaki şekilde verilmektedir.



Şekil 2. Timer/Counter Çevre Biriminin Temel Yapısı

1.3. Timer/Counter Çalışma Modları

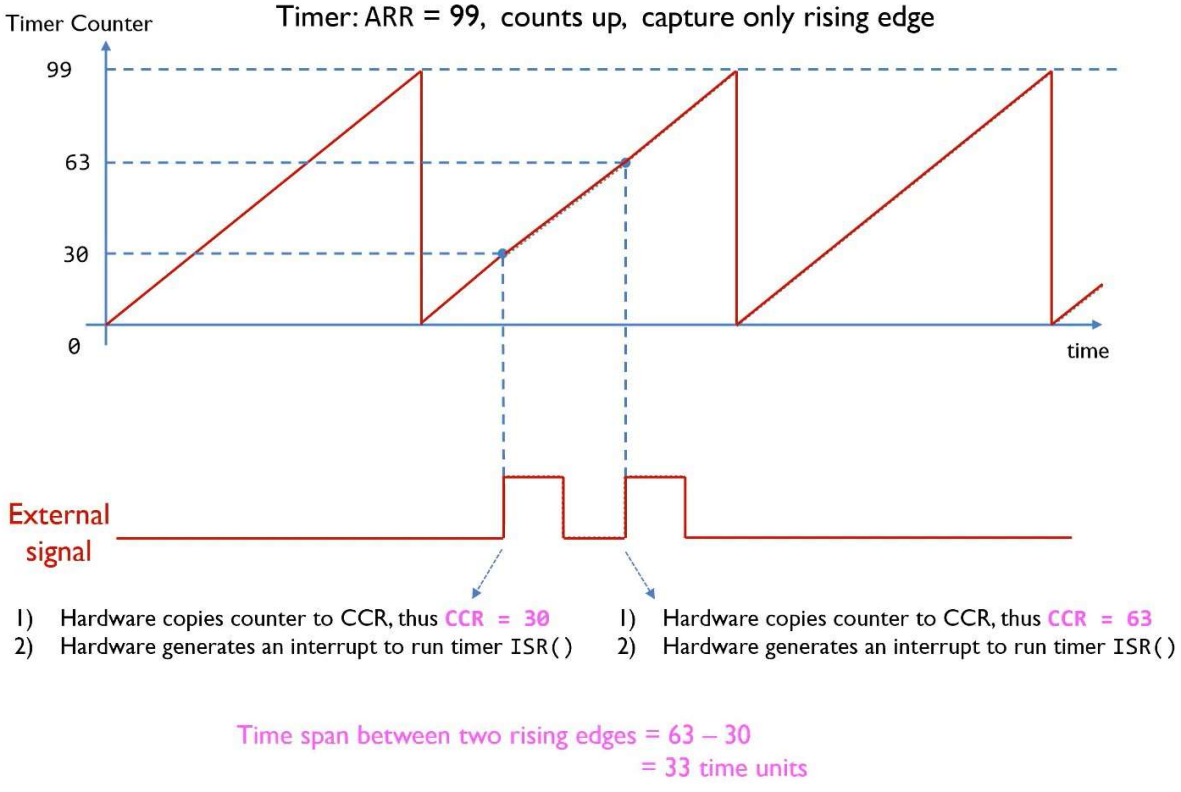
Timer/Counter çevre birimi sabit periyotlu kesme istekleri üretmenin yanı sıra gelişmiş birçok amaç için kullanılabilir. ARM mimarisine sahip STM32 mikrodenetleyicileri Input Capture, Output Compare, PWM ve One Pulse çalışma modlarına sahiptir. Bu çalışma modlarını inceleyecek olursak;

1.3.1. Input Compare Mode (Giriş Yakalama Modu)

Input Capture modu, bir dış sinyalin zaman bilgilerini (örneğin sinyalin ne zaman geldiği, ne sıklıkta geldiği) donanım destekli olarak yakalayarak, bu anlara ait zaman sayacını (CNT) bir kaydediciye (capture register) aktaran bir zamanlayıcı (timer) modudur. Her tetikleme geldiğinde timer CNT değeri CCR'ye kopyalanır ve istenirse kesme oluşur. İki ardışık yakalama (capture) değeri farkı alınarak arada geçen süre hesaplanır.

Aşağıdaki şekilde Input Capture moduyla bir sinyalin periyodunun ölçülmesini gösteren zaman diyagramı sunulmuştur. Bu örnekte timer pinine uygulanan harici sinyalin ilk ve ikinci yükselen kenar anlarındaki Sayaç (CNT) içerisinde bulunan değerler Yakalama/Karşılaştırma Kaydedicisine (CCR) aktarıldıktan sonra iki arşışık yakalama (captue) değerlerinin farkı alınarak 33 zaman birimi (time unit) kadar bir fark elde edilmiştir. Bu zaman birimi timer/counter çevre biriminin çalışma frekansı ile ilgilidir. Örneğin timer/counter biriminin bağlı bulunduğu saat frekansı f_{HCLK} = 84MHz, PSC = 8399 değeri için sayıcının çalışma frekansı yani sayma hızı ve devamında 33 zaman biriminin karşılık geldiği süre şu şekilde hesaplanır;

$$f_{CNT} = \frac{f_{HCLK}}{PSC + 1} = \frac{84.000.000}{8400} = 10kHz = 0.1\,ms \rightarrow 33 * 0.1\,ms = \mathbf{3.3\,ms}$$



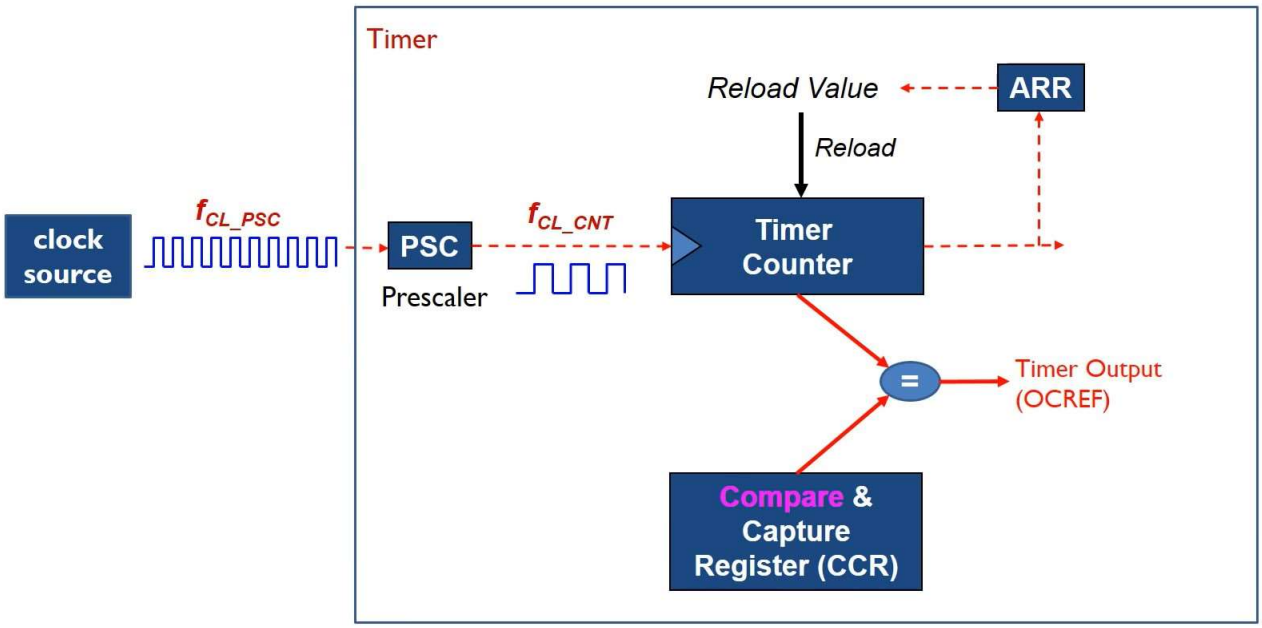
Şekil 3. Timer/Counter Input Capture Mode Örneği

Bu mod genellikle aşağıda yer alan uygulamalarda kullanılmaktadır.

- Dijital sinyallerin periyodu veya frekansını ölçmek
- PWM sinyalin frekans ve duty cycle değerini ölçmek
- Dış tetikleyici olaylar arasındaki süreyi ölçmek (Örn. encoder ile devir hesabı)

1.3.2. Output Compare Mode (Çıkış Karşılaştırma Modu)

Output Compare modu, zamanlayıcının sayaç değeri (CNT) belirli bir karşılaştırma değerine (CCRx, Capture/Compare Register) ulaştığında, bir olay oluşturulmasını sağlayan bir moddur. Bu mod, belirli zaman aralıklarında dışarıya sinyal üretmek, olay tetiklemek veya zamanlayıcıya bağlı çıkış pinlerini kontrol etmek için kullanılır.



Şekil 4. Timer/Counter Output Compare Yapısı

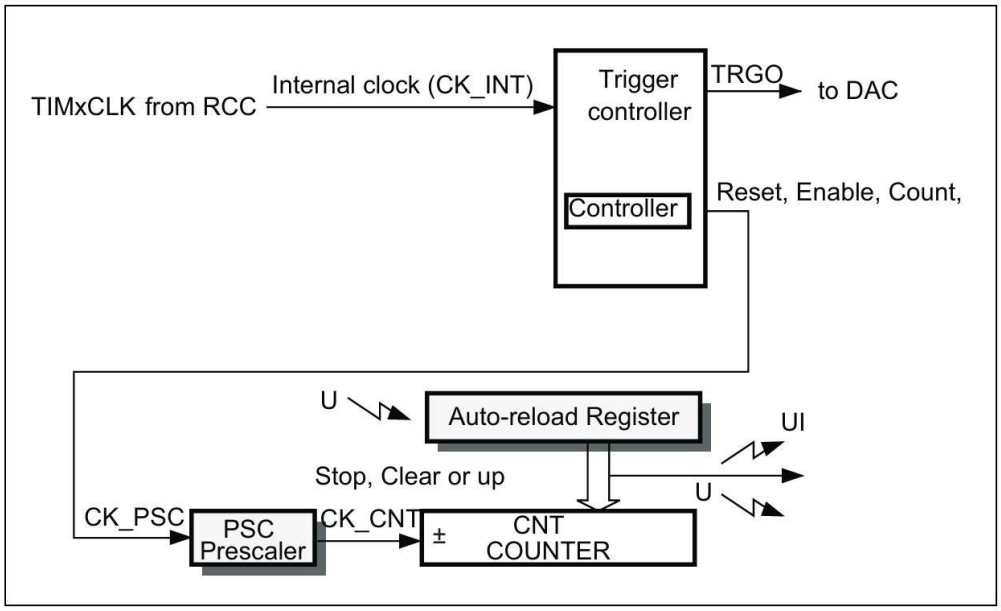
Output Compare modu, Timer/Counter temel yapısında bulunan ARR değeri ile gerçekleştirilen overflow kesmesine benzer yapıda çalışmaktadır. Farklı olarak timer birimi ilgili pin üzerinden çıkış sinyalleri üretebilmekte ve bir timer birimi içerisinde 4 adet birbirinden bağımsız periyodik zaman kesme isteklerinin üretilebiliyor olmasıdır.

2. STM32 – TIMER

STM32F407VGT6 mikrodenetleyicisi, farklı işlevleri destekleyen çeşitli timer (zamanlayıcı) türlerine sahiptir. Bu timerlar, donanımsal özelliklerine göre üç ana gruba ayrılır: Basic Timer (Temel Zamanlayıcılar), General-Purpose Timer (Genel Amaçlı Zamanlayıcılar) ve Advanced-Control Timer (Gelişmiş Zamanlayıcılar). Her bir grup farklı uygulamalar için optimize edilmiştir ve sahip oldukları özellikler de buna göre değişir. Bu timer'lar, sinyal üretimi, ölçümü, motor kontrolü, PWM üretimi, zaman gecikmesi, kesme tetikleme ve daha pek çok görev için kullanılabilir.

2.1. Basic Timer (Basit Zamanlayıcı)

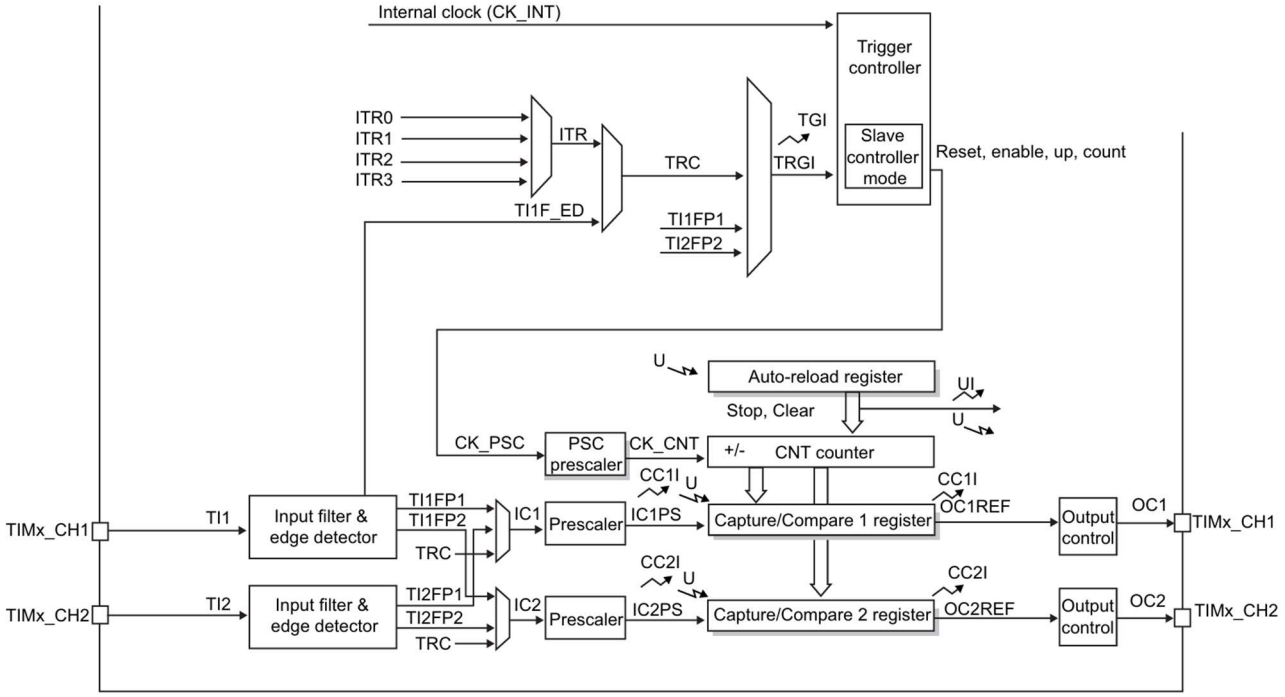
Basic Timer grubu yalnızca zaman tabanlı görevler için kullanılır. STM32F407VGT6'da TIM6 ve TIM7 olmak üzere iki adet basic timer bulunur. Bu timerlar 16-bit genişliğe sahiptir ve yalnızca zamanlayıcı olarak görev yaparlar. Input capture, output compare, PWM üretimi veya herhangi bir pin bağlantısı gibi gelişmiş işlevleri desteklemezler. Genellikle kesme üretmek veya DAC tetiklemek amacıyla kullanılırlar. Örneğin belirli aralıklarla kesme üretip bu kesmede bir sayacı artırarak basit bir zaman tabanı oluşturmak mümkündür. Ayrıca bu timerlar mikrodenetleyicinin harici bir olayla etkileşim kurmadan kendi içinde zaman ölçümü yapmasını sağlar.



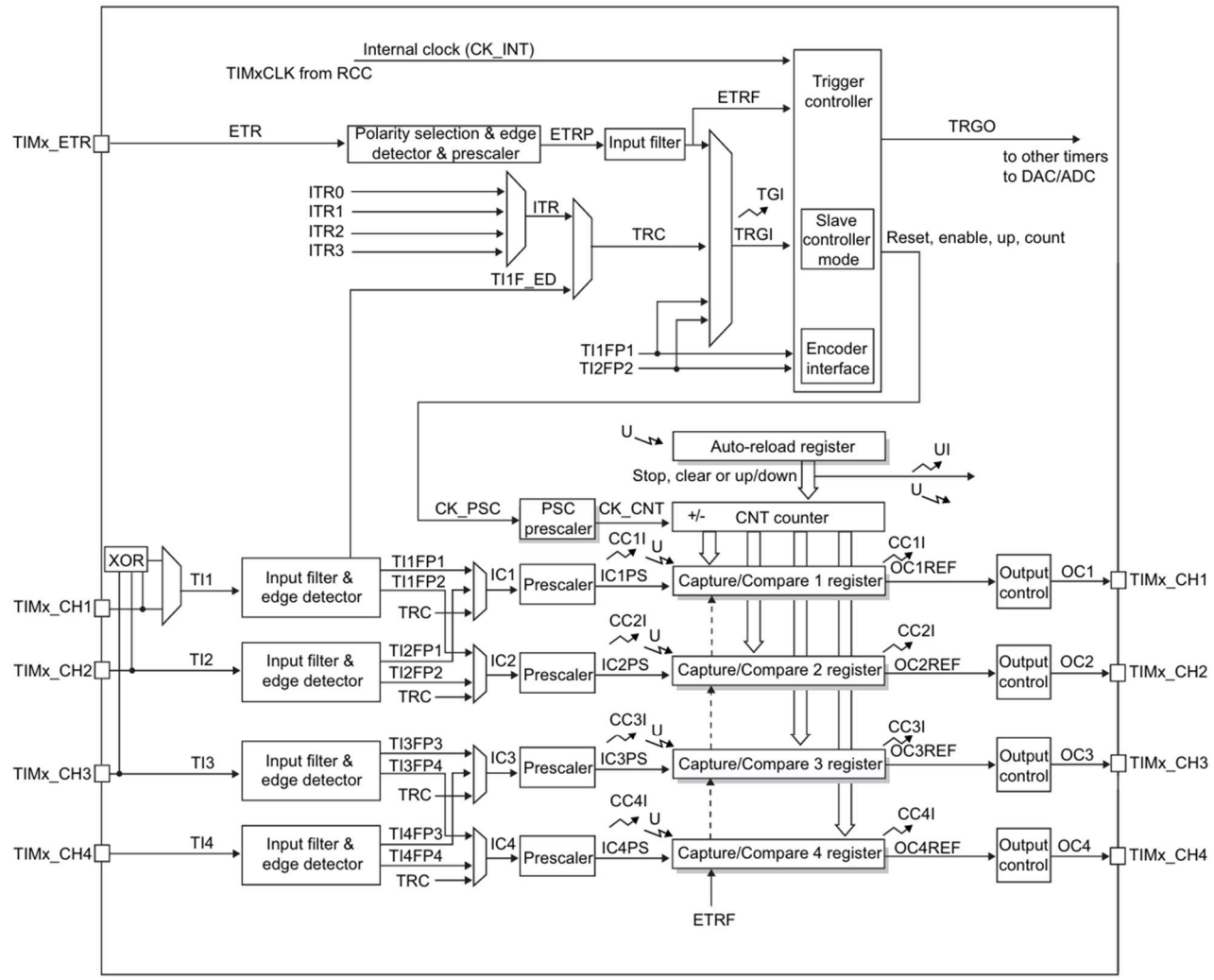
Şekil 5. Basic Timer Blok Diyagram

2.2. General-Purpose Timer (Genel Amaçlı Zamanlayıcı)

General-Purpose Timer grubu daha gelişmiş özelliklere sahiptir ve çok yönlü kullanım imkânı sunar. Bu grupta STM32F407VGT6'da TIM2, TIM3, TIM4, TIM5 gibi büyük timerlar (bazıları 32-bit) ile birlikte TIM9, TIM10, TIM11, TIM12, TIM13 ve TIM14 gibi küçük 16-bit timerlar yer alır. General-purpose timer'lar input capture, output compare, PWM üretimi, one-pulse mode ve encoder interface gibi birçok özelliği destekler. Ayrıca DMA ve interrupt ile çalışabilirler.



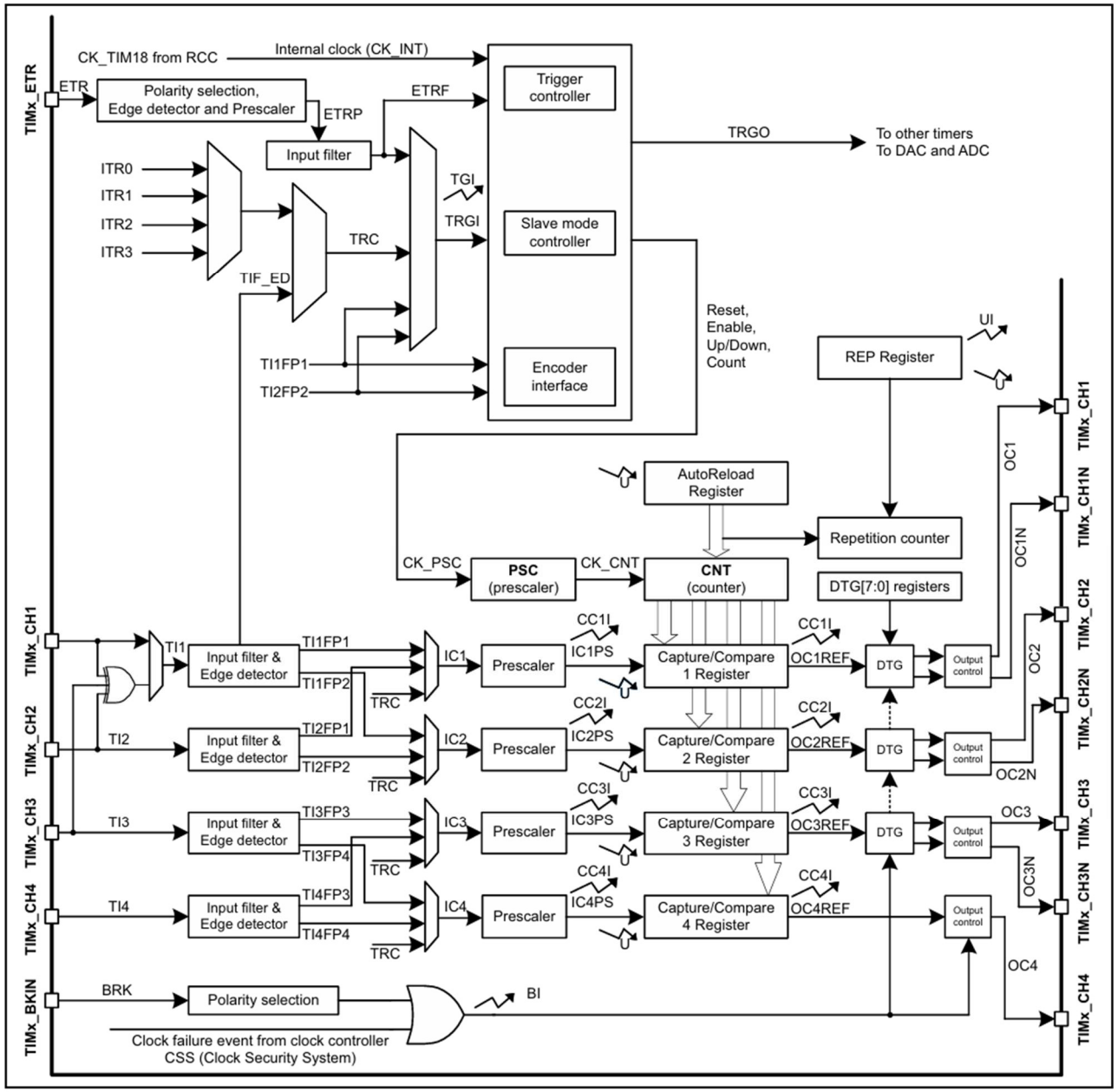
Şekil 6. General Purpose Timer (TIM9 – TIM12)



Şekil 7. General Purpose Timer (TIM2 – TIM5)

2.3. Advanced-Control Timer (Gelişmiş Kontrol Zamanlayıcı)

Advanced-Control Timer grubu ise en gelişmiş timerlardır ve motor sürme gibi hassas zaman kontrolü gerektiren uygulamalar için tasarlanmıştır. STM32F407VGT6'da TIM1 ve TIM8 olmak üzere iki adet advanced timer bulunur. Bu timerlar 16-bit genişliğe sahiptir ancak donanımsal olarak en zengin zamanlayıcılardır. PWM üretimi için center-aligned mode, edge-aligned mode gibi seçenekler sunarlar. Dead-time insertion, complementary output (örneğin CH1 ve CH1N), break input gibi özellikler barındırırlar. Bu sayede motor sürücü devrelerinde güvenli ve kararlı kontrol sağlarlar. Ayrıca repetition counter, master-slave senkronizasyon desteği, hall sensör arayüzü gibi motor kontrol uygulamaları için özel destek sunarlar. TIM1 ve TIM8, aynı zamanda encoder modunu da destekler ve gelişmiş PWM üretimi için tercih edilirler.



Şekil 8. Advanced-Control Timer (TIM1, TIM8)

3. STM32 – TIMER KESME PERİYOT HESABI

STM32 mikrodenetleyicilerde timer (zamanlayıcı) birimi kullanarak periyodik zaman kesmesi (interrupt) oluşturmak için, doğru frekans ve süre ayarlarının yapılması gerekir. Bu ayarları yapabilmek için bazı temel matematiksel denklem hesaplamalarını bilmek gerekir. Bu hesaplamalar timer'ın çalışma frekansına, prescaler (bölücü), auto-reload register (ARR) ve bazen clock division gibi parametrelere bağlıdır.

Timer birimi, mikrodenetleyiciye bağlı olan saat kaynağından aldığı clock sinyali ile saymaya başlar. Timer'ın sayaç değeri (CNT), belirli bir değere kadar artar ve bu değere ulaştığında tekrar sıfırlanır. Eğer interrupt (kesme) etkinleştirilmişse, bu sıfırlama anında bir kesme oluşur. Böylece belirli aralıklarla çalışan bir zaman tabanı (time base) oluşturulabilir.

Bu süreci doğru hesaplamak için temel bir matematiksel denklem kullanılır. Timer periyodu yani kesmelerin oluşma aralığı aşağıdaki formül ile hesaplanır:

$$\boldsymbol{T_{kesme}} = \frac{(PSC + 1) * (ARR + 1)}{f_{HCLK}}$$

Burada PSC, timer'ın prescaler (bölücü) değeridir. ARR ise sayaç değeri için ulaşılacak eşik değeri (Auto-Reload Register) olarak kullanılır. f_{HCLK} ise timer'ın beslendiği saat frekansıdır. Bu değer, mikrodenetleyicide APB1 veya APB2 bus frekansına bağlıdır.

Pratik bir örnek üzerinden istediğimiz aralıklarda kesme isteklerinin oluşması için timer birimine ait PSC ve ARR değerlerinin nasıl belirleneceğini inceleyelim. Örneğin 500ms aralıklar ile kesme isteği üretmek istediğimiz bir senaryo için hesaplamaları yapalım. İlk olarak kullanıacağımız timer biriminin bağlı olduğu clock hattının çalışma frekansını ve PSC/ARR değer aralıklarını öğrenmemiz gerekiyor. f_{HCLK} = 84MHz, 16-bit PSC (0 – 65.535), 16-bit ARR (0 – 65.535) ve T_{kesme} = 0.5 değerleri için;

$$0.5 = \frac{(PSC + 1) * (ARR + 1)}{84.000.000} \rightarrow (PSC + 1) * (ARR + 1) = 42.000.000$$

Bu durumda yukarıdaki formülde PSC ve ARR değerlerinden birini sabit alarak diğerini çözebiliriz. Genelde kolaylık olması açısından PSC = 8.399 seçilmektedir. Son olarak ARR değerini şu şekilde hesaplayabiliriz;

$$(ARR + 1) = \frac{42.000.000}{8400} \rightarrow \boldsymbol{ARR = 4.999}$$

Bu şekilde her 1 ms'de bir kesme oluşturulabilir. Timer bu yapılandırma ile başlatılır ve ilgili kesme fonksiyonu (interrupt service routine) içerisine yapılması istenen işlemler yazılır.

Sonuç olarak, periyodik kesme üretimi için şu adımlar izlenir: Timer clock frekansı belirlenir, istenen periyot için uygun PSC ve ARR değerleri hesaplanır, timer yapılandırılır ve kesme fonksiyonu etkinleştirilir. Bu işlemlerle çok hassas ve düzenli zaman tabanlı görevler gerçekleştirilebilir.

4. STM32 – TIMER HAL

HAL (Hardware Abstraction Layer), STMicroelectronics tarafından sağlanan bir üst seviye API'dir ve donanım seviyesindeki işlemleri kolaylaştırır. STM32CubeIDE ortamında Timer yapılandırması ve kullanımı için HAL TIM sürücüsü kullanılır.

HAL TIM Kütüphanesi, konfigürasyon, başlatma, okuma ve kesme/DMA yönetimi için çeşitli fonksiyonlar içerir.

HAL_TIM_Base_Start_IT (TIM_HandleTypeDef * htim)

Metot Açıklaması : Bu fonksiyonla timer aktif hâle gelir ve her taşmada kesme oluşturur.

Parametreler : Belirtilen TIM için yapılandırma bilgilerini içeren handle (tanıtıcı) pointer değişken.

Dönüş Değeri : HAL_StatusTypeDef veri tipinde durum değişkeni.

HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)

Metot Açıklaması : Timer overflow (update event) kesmesi geldiğinde çağrılan callback fonksiyonudur. Bu fonksiyonu kullanıcı tanımlar. Timer'da kesme oluştuğunda otomatik olarak bu fonksiyon çağrılır. Hangi timer'dan kesme geldiği htim->Instance üzerinden kontrol edilebilir.

Parametreler : Belirtilen TIM için yapılandırma bilgilerini içeren handle (tanıtıcı) pointer değişken.

Dönüş Değeri : Dönüş değeri yok.

__HAL_TIM_SET_COUNTER (TIM_HandleTypeDef * htim, uint32_t value)

Metot Açıklaması : Timer sayacını (CNT) sıfırlar veya belirli bir değere ayarlar.

Parametreler : Belirtilen TIM için yapılandırma bilgilerini içeren handle (tanıtıcı) pointer değişken ve yeni Sayaç (CNT) değeri.

Dönüş Değeri : Dönüş değeri yok.

__HAL_TIM_GET_COUNTER (TIM_HandleTypeDef * htim)

Metot Açıklaması : Timer'ın anlık CNT (sayım) değerini okur. Genellikle süre ölçüm uygulamalarında kullanılır.

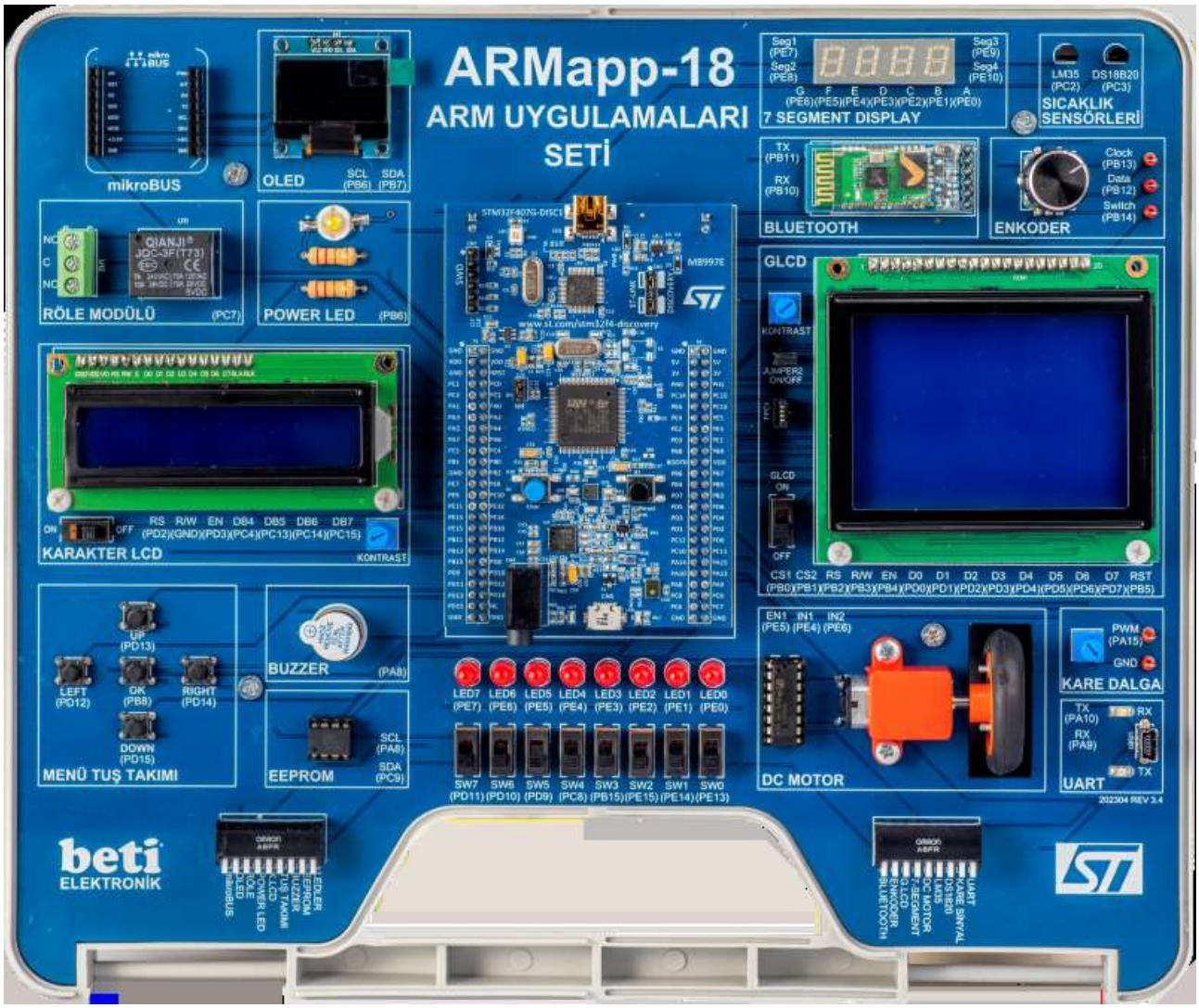
Parametreler : Belirtilen TIM için yapılandırma bilgilerini içeren handle (tanıtıcı) pointer değişken.

Dönüş Değeri : Güncel Sayaç (CNT) değerini uint32_t tipinde döndürür.

4. ARM APP DENEY KİTİ – TIMER

Laboratuvar uygulamalarında kullanılan ARM APP Deney Kiti içerisinde TIMER çevre biriminin kullanımı için birçok komponent bulunmaktadır. Bunlar;

- 7 Segment Display
- Enkoder
- DC Motor



Şekil 9. ARM APP Deney Kiti

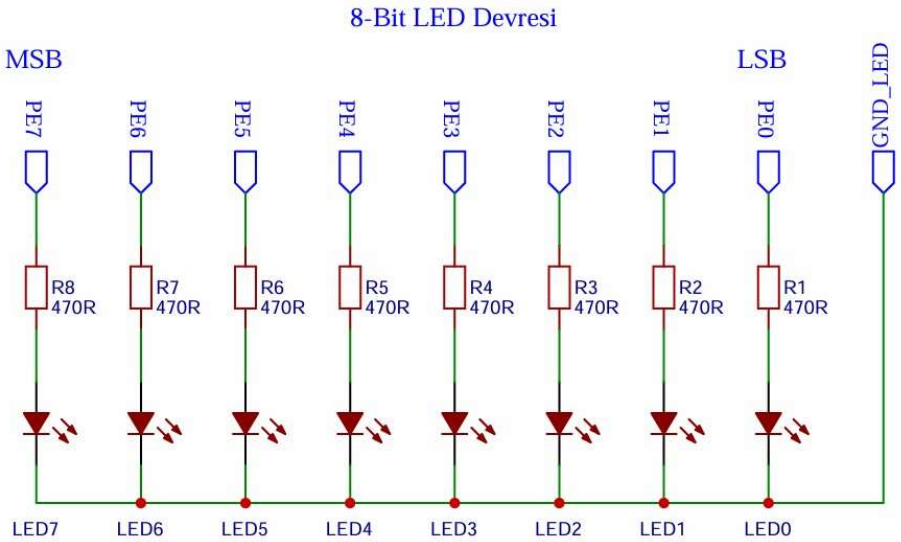
Timer/Counter çevre birimi mikrodenetleyici uygulamalarında genellikle dijital olayları saymak, zaman ölçmek, geciktirme oluşturmak, olaylar arasındaki süreyi ölçmek, belirli periyotlarda sinyalleri otomatik üretebilmek ve PWM sinyali üretebilmek için kullanılır.

5. PRATİK UYGULAMA

Laboratuvar uygulamasına başlamadan önce TIMER çevre birimi ile ilgili bir pratik uygulama yapalım. Öncelikle bir Gömülü Sistem uygulamasına başlamadan önce yapılması gereken mikrodenetleyici ile uygulamada kullanılacak olan bileşenin donanımsal bağlantılarının yapılması ve kontrol edilmesi gerekmektedir. Ders kapsamında kullanılan deney kitinde mikrodenetleyici ve bileşenler PCB kart üzerinde donanımsal olarak bağlı bulunduğu için kullanıcıyı kablolama ve bağlantı işlemlerinden soyutlayarak doğrudan Gömülü Sistem Yazılımı geliştirmesine imkan tanımaktadır.

Gömülü Yazılım geliştirmeye başlarken uygulamada kullanılacak olan bileşenlerin mikrodenetleyiciye bağlı bulunduğu pinlerin yapılandırılması gerekmektedir. Yapılandırma işlemi laboratuvar uygulamalarında kod geliştirme için kullanılacak olan STM32CubeIDE içerisinde yer alan CubeMX modülü ile yapılmaktadır. Örneğin ARM APP Deney Kitinde bulunan 8-Bit LED devresinde bulunan LED'leri zaman kesmesi kullanarak belirli periyotlarda Toggle etme üzerinde bir uygulama geliştirelim.

STM32CubeIDE'de proje oluşturduktan sonra gelen CubeMX modülü ile pin yapılandırmalarını gerçekleştirebilmek için 8-Bit LED Devresinin mikrodenetleyicideki bağlı bulunduğu pinlerin Port ve Numalaralarını ARM APP Deney Kiti Şematik Tasarım dosyasına bakarak belirlenmelidir. Şekil 10'da ilgili devre şeması gösterilmiştir. Bu pinlerin GPIO_Output olarak yapılandırılması yeterli olacaktır.



Şekil 10. ARM APP Deney Kiti Potansiyometre Devre Şeması

TIM yapılandırmasını CubeMX modülü üzerinden TIM6 olarak yapılandırılması gerekmektedir. TIM parametreleri ise aşağıdaki şekilde belirlenmelidir;

- **Prescaler (PSC)** **:** 8400 - 1
- **Counter Mode** **:** Up
- **Counter Period (ARR)** **:** 10000 - 1
- **Auto-Reload Preload** **:** Enable
- **NVIC TIM6 Global Interrupt** **:** Enable
- $\mathbf{F_{hclk}}$ **:** 84MHz

CubeMX üzerinde pin yapılandırması bittikten sonra “Generate Code” ikonuna tıklayarak kod geliştirme arayüzünü açıp aşağıda verilen kodu yazıp “Debug/Run” ikonuna tıklayıp yazılımı geliştirme kartına yüklüyoruz.

```
#include "main.h"

TIM_HandleTypeDef htim6;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM6_Init (void);

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
	if(htim == &htim6){
		HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_0);
	}
}
int main(void){
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM6_Init();

  while (1){

  }
}
```

6. LABORATUVAR DENEY UYGULAMASI

Uygulama isterleri ders esnasında paylaşılacaktır.

7. KAYNAKLAR

[1] https://www.youtube.com/watch?v=VfbW6nfG4kw&list=PLrJ4K0hkaFI-kCnsu34ri9rkw7o-Nh0xr

[2] https://www.youtube.com/watch?v=aLCUDv_fgoU&list=PLrJ4K0hkaFI-kCnsu34ri9rkw7o-Nh0xr&index=3

[3] https://www.youtube.com/watch?v=zkrVHIcLGww&list=PLrJ4K0hkaFI-kCnsu34ri9rkw7o-Nh0xr&index=4

[4] https://www.youtube.com/watch?v=2FoZ7kHOdT0&list=PLrJ4K0hkaFI-kCnsu34ri9rkw7o-Nh0xr&index=5

[5] https://www.youtube.com/watch?v=VopqMBT_yXI&list=PLrJ4K0hkaFI-kCnsu34ri9rkw7o-Nh0xr&index=8