# STM32L0538-DISCO – HAL Fonksiyonları ve Parametre Açıklamaları

## HAL\_GPIO\_WritePin(GPIOx, GPIO\_PIN\_x, GPIO\_PIN\_SET/RESET);

Bu fonksiyon, belirtilen GPIO pinine HIGH (SET) ya da LOW (RESET) çıkışı verir. Örneğin bir LED’i açmak veya kapatmak için kullanılır.

### Parametreler:

GPIOx: Pinin ait olduğu port. Örneğin: GPIOA, GPIOB.

GPIO\_PIN\_x: Kontrol edilecek pin numarası. Örneğin: GPIO\_PIN\_5.

GPIO\_PIN\_SET/RESET: SET → pin HIGH (1), RESET → pin LOW (0) yapılır.

## HAL\_GPIO\_TogglePin(GPIOx, GPIO\_PIN\_x);

Belirtilen pinin çıkış seviyesini değiştirir. Eğer pin HIGH ise LOW yapılır, LOW ise HIGH yapılır. Genellikle LED yanıp sönme efektleri için kullanılır.

### Parametreler:

GPIOx: GPIO portu (örneğin GPIOB).

GPIO\_PIN\_x: Tersine çevrilecek pin (örneğin GPIO\_PIN\_4).

## HAL\_GPIO\_ReadPin(GPIOx, GPIO\_PIN\_x);

Bir giriş pininin lojik seviyesini okur. Genellikle buton gibi giriş birimlerinin durumunu kontrol etmek için kullanılır.

### Parametreler:

GPIOx: Hangi porttan okunacağı (örneğin GPIOA).

GPIO\_PIN\_x: Okunacak pin (örneğin GPIO\_PIN\_0).

## HAL\_TIM\_PWM\_Start(&htimx, TIM\_CHANNEL\_x);

Seçilen zamanlayıcı kanalında PWM çıkışı başlatır. Örneğin bir buzzer’a sinyal göndermek veya LED parlaklığını kontrol etmek için kullanılır.

### Parametreler:

&htimx: PWM için yapılandırılmış Timer örneği. Örn: &htim2.

TIM\_CHANNEL\_x: PWM çıkışı yapılacak kanal. Örn: TIM\_CHANNEL\_1.

## \_\_HAL\_TIM\_SET\_COMPARE(&htimx, channel, duty\_cycle);

PWM sinyalinin duty cycle oranını belirler. Duty cycle, sinyalin açık kalma süresini ifade eder. LED parlaklık ayarı bu şekilde yapılır.

### Parametreler:

&htimx: PWM üretimi için kullanılan Timer yapısı.

channel: Kullanılacak kanal (örneğin TIM\_CHANNEL\_1).

duty\_cycle: 0–timer period arası bir değer, sinyalin açık kalma süresi.

## HAL\_UART\_Transmit(&huartx, data, length, timeout);

UART üzerinden belirtilen veriyi gönderir. Bilgisayara mesaj göndermek gibi işlemler için kullanılır.

### Parametreler:

&huartx: Kullanılan UART portunun tanımlayıcısı. Örn: &huart1.

data: Gönderilecek veriyi içeren byte dizisi.

length: Kaç byte gönderileceği.

timeout: İşlemin tamamlanması için maksimum bekleme süresi (ms).

## HAL\_UART\_Receive(&huartx, data, length, timeout);

UART üzerinden gelen veriyi okur. Bilgisayardan veri alma işlemleri için kullanılır.

### Parametreler:

&huartx: Kullanılan UART portu.

data: Alınan verinin yazılacağı dizi.

length: Alınacak byte sayısı.

timeout: Maksimum bekleme süresi (ms).

## HAL\_ADC\_Start(&hadc1);

ADC modülünü başlatır ve analog girişten dijital örneklemeye hazırlanır.

### Parametreler:

&hadc1: Kullanılacak ADC yapısı (örneğin &hadc1).

## HAL\_ADC\_PollForConversion(&hadc1, timeout);

ADC dönüşümünün tamamlanmasını bekler. Bu süre boyunca işlemci bloklanabilir.

### Parametreler:

&hadc1: Kullanılan ADC örneği.

timeout: Maksimum bekleme süresi (ms).

## HAL\_ADC\_GetValue(&hadc1);

ADC sonucu olarak dijital değeri döndürür (0–4095 arası tipik).

### Parametreler:

&hadc1: ADC yapısı. Daha önce başlatılmış ve dönüşüm yapılmış olmalıdır.

## HAL\_LCD\_Init(&hlcd);

STM32L0538-DISCO üzerindeki segment LCD ekranı başlatır. COM ve SEG pin yapılandırmalarını, çalışma frekansını ve ekran sürme parametrelerini ayarlar.

### Parametreler:

&hlcd: LCD yapılandırma yapısı (LCD\_HandleTypeDef). CubeMX tarafından tanımlanır.

COM: Common pinler (COM0–COM3). Her biri ekranın bir satırını kontrol eder.

SEG: Segment pinleri. Hangi segmentlerin (çubukların) yanacağını belirler. SEG0–SEG27 arası kullanılır.

## HAL\_I2C\_Master\_Transmit(&hi2c, DevAddress, pData, Size, Timeout);

Master cihaz (STM32), belirli bir I2C slave cihazına veri gönderir. OLED ekran, sensör, EEPROM gibi çevre birimlerine veri aktarımı için kullanılır.

### Parametreler:

&hi2c: I2C yapılandırması tanımı. Örneğin &hi2c1.

DevAddress: Slave cihaz adresi. 7-bit adres << 1 yapılmalıdır.

pData: Gönderilecek veriyi içeren dizi.

Size: Gönderilecek veri boyutu (byte cinsinden).

Timeout: Maksimum bekleme süresi (ms).

**NVIC = Nested Vectored Interrupt Controller**  
Türkçesi: **İç İçe Vektörlenmiş Kesme Denetleyicisi**

NVIC, mikrodenetleyicide:

* **Kesmeleri (interrupts) etkinleştirir veya devre dışı bırakır.**
* **Kesme önceliklerini belirler.**
* Aynı anda gelen birden fazla kesmeyi **öncelik sırasına göre işler.**
* Düşük öncelikli kesme çalışırken, **daha yüksek öncelikli başka bir kesmenin** onu kesmesine izin verir (nested interrupt – iç içe kesme).

| **Görev** | **Açıklama** |
| --- | --- |
| **Kesmeleri etkinleştirme/devre dışı bırakma** | Belirli çevresel birimlerin (UART, SPI, TIM, EXTI...) kesmeleri açılır/kapatılır. |
| **Öncelik atama** | Her kesmeye bir "öncelik değeri" atanır; daha küçük değer, daha yüksek öncelik demektir. |
| **İç içe kesmelerin kontrolü** | Bir kesme devam ederken daha öncelikli başka bir kesme girerse, işlem ona geçer. |
| **Vektör tablosu üzerinden yönlendirme** | Her kesmenin karşılık geldiği fonksiyon adresi NVIC ile eşleştirilir. |

// Kesmeyi etkinleştir

HAL\_NVIC\_EnableIRQ(USART2\_IRQn);

// Öncelik ayarla: (IRQ numarası, ana öncelik, alt öncelik)

HAL\_NVIC\_SetPriority(USART2\_IRQn, 1, 0);

**Düşük sayılar daha yüksek önceliktir.** (0 en yüksek önceliktir.)

NVIC ile ilişkili kesme fonksiyonları stm32xx\_it.c içinde otomatik oluşur.

Kesme fonksiyonları void IRQHandler(void) şeklinde olur (örn. USART2\_IRQHandler).

**PPP = Yer tutucu (placeholder)**  
STM32 HAL dokümantasyonunda kullanılan genel bir ifade olup, yerine kullandığınız çevresel birimin adı (UART, SPI, I2C, vb.) gelir.

**EXTI = External Interrupt/Event Controller**  
Türkçesi: **Harici Kesme / Olay Denetleyicisi**

EXTI, bir **GPIO pini üzerinden gelen sinyale** göre mikrodenetleyicide bir **kesme (interrupt)** oluşturur. Bu sayede mikrodenetleyici, o pini sürekli kontrol etmek yerine sadece bir olay olduğunda işlem yapar → **enerji tasarrufu ve daha verimli programlama.**