21.03.2025

SİVAS CUMHURİYET ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
GÖMÜLÜ SİSTEMLER DERSİ
LABORATUVAR FÖYÜ

Bölüm 3

INTERRUPT (KESMELER)

Dersin Hocası: Doç. Dr. Ahmet Gürkan YÜKSEK

Dersin Asistanı: Arş. Gör. Ahmet Utku ELİK

İçindekiler

Öğrencinin;

Grup No :

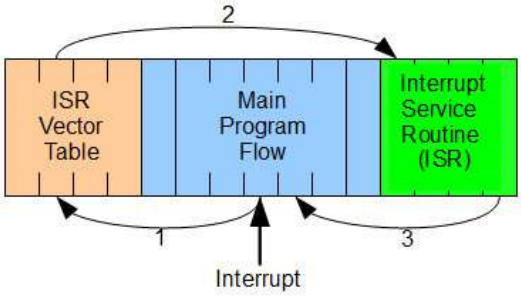
Ad Soyad :

Okul No :

Teslim Tarihi :

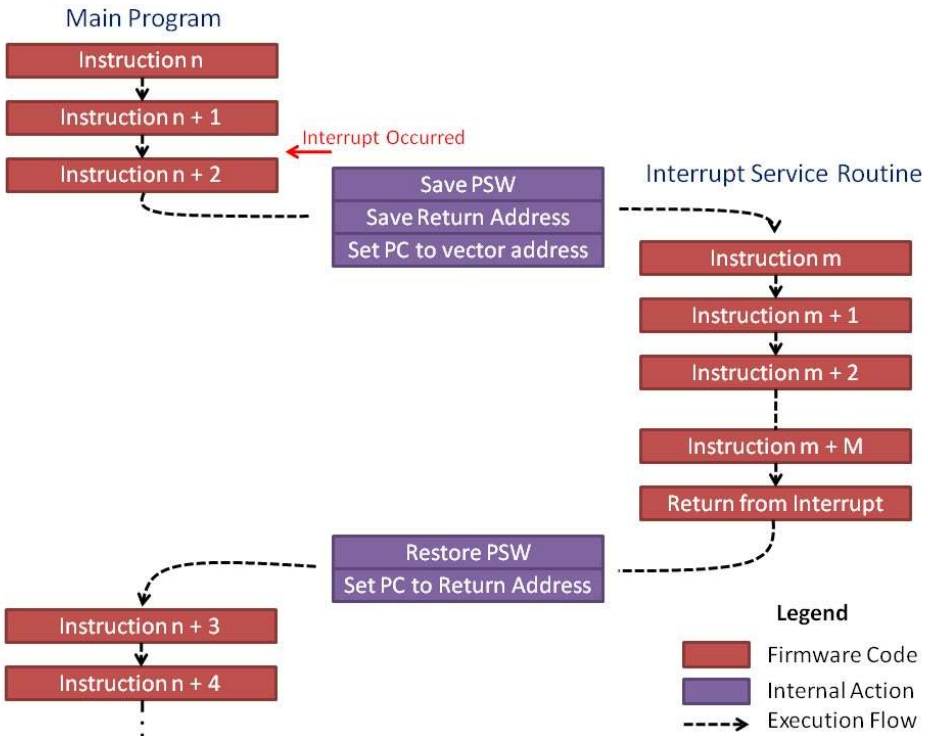
1. Kesme (Interrupt)

Mikrodenetleyiciler hafızalarında yazılan programı satır satır okuyarak ilgili komutun gerektirdiği işlemleri gerçekleştirirler. Programın akışına göre subroutine denilen alt programlara dallanabilirler. Fakat bazı durumlarda mikrodenetleyicinin program akışının dışına çıkması istenebilir. Bu durumlarda interrupt (kesme) devreye girer. Bir Interrupt subroutine den tamamen farklıdır. Programın alt programa dallanması gereken yerler program yazılırken belirlenir. Fakat kesme program dışında bir dış uyarıcı ile gerçekleşir. Bu dış uyarıcı seri haberleşme biriminden gelen bir veri, timer'ın dolmasıyla oluşan bir sinyal, ya da bir sensörün bağlı olduğu pine gelen 5V (lojik 1) sinyal olabilir. Kesme sinyali geldiğinde program kesme vektörüne dallanarak burada bulunan komutları işlemeye başlar.



Şekil 1. Kesme Servis Akışı (Interrupt Service Flow)

Kesme, işlemcinin ana program akışını durdurarak, önceden belirlenmiş özel bir kod bloğunu (Interrupt Service Routine - ISR) çalıştırmasını sağlayan bir mekanizmadır. Kesme meydana geldiğinde işlemci ana programı geçici olarak durdurur, kesme işleyicisini (ISR) çalıştırır ve ardından ana programa kaldığı yerden devam eder. Kesme kullanmanın temel amacı, işlemcinin gereksiz yere sürekli kontrol yapmasını (polling) önlemek ve kaynakları daha verimli kullanmaktır.



Şekil 2. Kesme Program Akışı (Interrupt Program Flow)

Kesme geldiğinde mikrodenetleyici “Program Counter(PC)” da tuttuğu hangi satırda kaldığı bilgisini “stack” denilen yığın belleğe geçici olarak atar ve kesmenin yönlendirdiği satıra gider kesme programı bittiğinde stack e dönüp hangi satırda kaldığı bilgisini alır ve PC’ye yazarak kaldığı yerden devam eder.

Genel olarak mikrodenetleyicilerde bir kesme isteği geldiği zaman işlemci aşağıdaki adımları izlemektedir;

1. Modüllerin bir tanesinden kesme isteği gelir.

2. İşlemci kesmenin yetkilendirilmiş olup olmadığını kontrol eder. Yetkilendirilmemiş ise kesmeyi görmezden gelir.

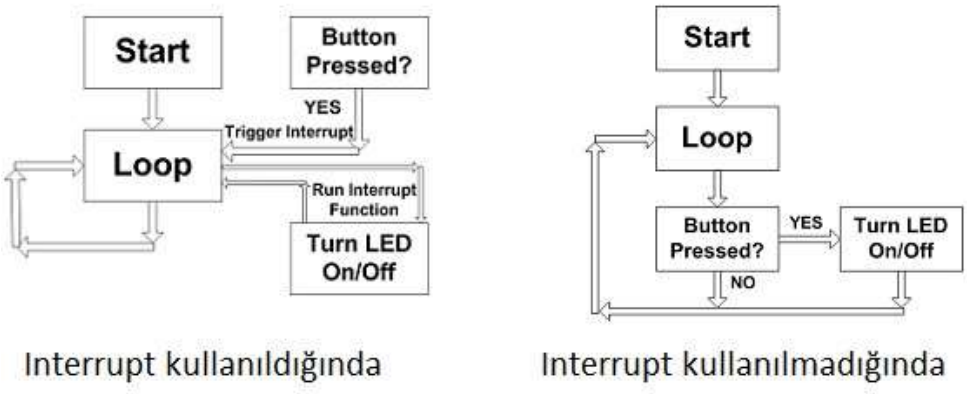
3. Yetkilendirilmiş kesmede ise işlemci, işlettiği buyruk setinin adresini (Program Counter, PC) ve durum bilgilerini (Status Register, SR) içeren kaydedicilerin verilerini yığın kaydedicisine (Stack Pointer, SP) saklar. Böylelikle kesme programı bittiğinde kaldığı yere geri dönebilir.

4. İşlemci ilgili kesmenin kesme vektör tablosundaki adresi kontrol eder ve orada bulunan adrese giderek kesme programının işlemlerini gerçekleştirir. Eğer kesme vektör tablosunda bir adres yoksa işlemci ana programa geri döner.

5. İşlemci kesme programını bitirdikten sonra Yığın Kaydedicisinden kaldığı yer ve durum bilgilerini alarak ana programa geri döner.

1.1. Kesme (Interrupt) - Sürekli Kontrol (Polling)

Mikrodenetleyicilerde bir olayın algılanması için başlıca Polling (Sürekli Kontrol / Yoklama) ve Interrupt (Kesme) yöntemleri kullanılır. Polling yönteminde işlemci, belirli aralıklarla ilgili donanımın durumunu kontrol eder; kesme yönteminde ise donanım, bir olay meydana geldiğinde işlemciyi keserek haber verir.



Şekil 3. Polling – Interrupt Arasındaki Fark

Bu iki yaklaşımın farklarını detaylıca inceleyelim:

1.1.1. Polling (Sürekli Kontrol / Yoklama)

Polling yönteminde işlemci, bir döngü içinde sürekli olarak çevre bileşen cihazlarını veya bayraklarını kontrol eder. Örneğin, bir butona basılıp basılmadığını anlamak için işlemci her döngüde buton girişinin durumuna bakabilir. Bu yöntem uygulanması kolay ve doğrudandır; özel bir donanım desteği gerekmez. Polling'in avantajı, mantığının basit olması ve kesmeler gibi ek mekanizmalara ihtiyaç duymamasıdır. Ancak polling ciddi dezavantajlar getirir: İşlemci sürekli meşgul durumda olduğu için verimsizdir ve güç tüketimini artırır. Bir olay gerçekleştiğinde, işlemci bir sonraki kontrol zamanına kadar bunu fark edemeyebilir, bu da latansı yükseltir. Örneğin 100 ms aralıklarla polling yapan bir sistemde olay, döngünün hemen başında meydana gelirse 100 ms'ye yakın bir gecikme ile algılanacaktır. İşlemci bu süre boyunca başka bir iş yapamaz, sürekli sorgulama ile zaman harcar. Sonuç olarak polling, basit uygulamalarda yeterli olsa da sık ve zaman kritik olaylarda yetersiz kalır.

1.1.2. Interrupt (Kesme)

Kesme yönteminde çevre bileşen, işlemciye asenkron bir sinyal göndererek "dikkat, bir olay oldu" der. İşlemci de anında o işi bırakıp ilgili kesme rutinine dallanır. Bu yöntemde işlemci, olay yoklaması yapmadığı için boşta kalabilir veya başka işler yürütebilir; sadece kesme geldiğinde tepki verir. Bu durum işlemcinin verimliliğini artırır ve güç tasarrufu sağlar. Ayrıca olay gerçekleştiği anda işlendiği için gecikme genellikle polling'e kıyasla çok düşüktür. Kesmelerin bir diğer faydası, farklı öncelik seviyeleri atayarak kritik olaylara diğerlerinden önce müdahale edebilmektir. Örneğin bir acil kesme (yüksek öncelikli) geldiğinde, işlemci düşük öncelikli bir işi yapıyor olsa bile onu yarıda kesip acil olana yönelebilir. Kesmelerin dezavantajı ise yazılım karmaşıklığını artırmasıdır. Her an kesilebilme durumu, program akışını öngörmeyi zorlaştırır ve dikkatli programlama gerektirir. Bir kesme rutininde yapılacak işlemler çok uzun veya yoğun ise, bu durum diğer kesmeleri geciktirebilir ve sistem performansını olumsuz etkileyebilir. Bu nedenle kesme servis rutinlerinin olabildiğince kısa ve verimli tutulması önerilmektedir.

Yukarıdaki karşılaştırmayı özetlemek gerekirse; polling, uygulanması kolay ancak işlemciyi sürekli meşgul eden ve olaylara tepki süresini artırabilen bir yöntemdir. Interrupt ise doğru kullanıldığında işlemciyi oldukça verimli kılar ve hızlı tepki sağlar. Çoğu gerçek zamanlı ve enerji duyarlı tasarımda kesme tabanlı yaklaşım tercih edilir. Ancak çok basit ve zaman kritikli olmayan uygulamalarda, özellikle kesme yönetiminin gereksiz karmaşıklık katacağı durumlarda polling de kullanılabilir. Tasarımcı, sistemin gereksinimlerine göre bu iki yöntemden uygun olanını veya hibrit bir yaklaşımı seçmelidir.

Tablo 1. Interrupt ve Polling Karşılaştırması

Özellik	**Polling (Sürekli Kontrol)**	**Interrupt (Kesme)**
Verimlilik	Düşük (CPU sürekli kontrol eder)	Yüksek (CPU sadece gerektiğinde tepki verir)
Tepki Süresi	Gecikmeli (Belirli bir döngü sonrası kontrol edilir)	Hızlı (Olay anında işlemci kesilir)
Enerji Tüketimi	Yüksek (CPU sürekli çalışır)	Düşük (CPU uyuyabilir ve sadece gerektiğinde çalışır)
Kullanım Alanları	Yavaş sensör okuma, düşük hız uygulamaları	Gerçek zamanlı sistemler, hızlı tepki gerektiren uygulamalar

1.2. Kesme (Interrupt) Çalışma Mekanizması

Kesme mekanizması ile işlemcinin gereksiz yere sürekli olayları kontrol etmesi (polling) yerine, sadece gerektiğinde tepki vermesi sağlanır. Bir kesme gerçekleştiğinde işlemci mevcut işlemi durdurur, ilgili kesme işleyicisini çalıştırır ve daha sonra ana program akışına kaldığı yerden devam eder.

Bir kesme meydana geldiğinde mikrodenetleyici içerisinde oldukça düzenli bir işlem akışı gerçekleşir. Kesmelerin çalışma prensibini adım adım inceleyelim:

1. **Kesme Talebi (Interrupt Request, IRQ) Oluşması:** Harici bir olay (butona basılması, seri haberleşme, sensör verisi) veya dahili bir olay (zamanlayıcı taşması, hata durumu) kesme talebi (IRQ) oluşturur.

2. **NVIC ile Kesme Denetimi:** Mikrodenetleyicinin NVIC (Nested Vector Interrupt Controller) birimi, kesme talebini alır ve değerlendirir. NVIC, tüm kesme isteklerini alır ve her birine ait etkinlik (enable) bitine ve öncelik seviyesine göre hangi kesmenin CPU'ya iletileceğine karar verir. Sadece etkinleştirilmiş ve önceliği uygun olan kesmeler NVIC tarafından işleme konur. Bir kesme oluştuğunda NVIC, onun önceliğini mevcut durumda çalışan (veya bekleyen) diğer kesmelerle karşılaştırır; en yüksek öncelikli kesme işlem için seçilir. NVIC aynı zamanda nested (iç içe) kesme desteği sağlar, yani bir kesme işlenirken daha yüksek öncelikli bir kesme gelirse, NVIC CPU'yu yeni kesmeye geçirebilir.

3. **Kesmenin CPU Tarafından Tanınması:** NVIC tarafından CPU'ya iletilen kesme talebi, CPU çekirdeği içinde kontrol edilir. CPU, o an yürütmekte olduğu talimatı tamamlar ve bir kesme geldiğini fark eder. Ardından kesmeye yanıt vermek için donanım düzeyinde bir dizi işlem gerçekleşir: Program sayacı (PC) ve bazı işlemci durum kayıtları stack'e otomatik olarak kaydedilir. Bu sayede, kesme bittiğinde programa kaldığı yerden devam edebilmek mümkün olur.

4. **Kesme Vektör Tablosundan İlgili Kesme Servis Rutini (ISR)'nin Çağrılması:** Her bir kesme türü için bellekte bir vektör tanımlıdır. Kesme vektör tablosu, genellikle bellek haritasının başlangıcında yer alan ve her kesmenin ISR başlangıç adresini tutan bir tablodur. CPU, kesme geldiğinde NVIC'ten gelen kesme numarasını kullanarak bu tabloda ilgili ISR adresini bulur. Ardından PC, bu adrese yüklenir ve böylece program akışı kesme servis rutinine dallanmış olur.

5. **Kesme Servis Rutininin (ISR) Yürütülmesi:** ISR içinde genellikle şunlar yapılır: öncelikle kesmeye neden olan bayrak durumu kontrol edilir ve gerekiyorsa temizlenir. Sonra uygulanmak istenen işlem gerçekleştirilir (örneğin LED yakma, veri okuma, sayacı sıfırlama vb.). ISR mümkün olduğunca hızlı bitirilmeli ve uzun süren işlemler içeriyorsa bu işlemlerin ana döngüye bırakılması önerilmektedir.

6. **Ana Programa Geri Dönüş:** Dönüş işlemiyle birlikte CPU, stack'e kaydettiği PC ve durum kayıtlarını geri yükler. Böylece kesme öncesinde kaldığı yerden program yürütülmesine devam edilir. Eğer bu arada başka kesme talepleri biriktiyse (bekliyorsa), NVIC dönüşte bunları değerlendirir ve gerekiyorsa hemen bir sonraki kesmeye geçer. ISR tamamlandıktan sonra, işlemci ana programın kesildiği noktadan devam eder.

Yukarıdaki adımlar kesme mekanizmasının genel işleyişini özetlemektedir. Önemli noktalar: Her bir kesme kaynağının bir vektör adresi vardır (vektör tablosu), NVIC birimi kesme önceliklerini ve çoklama işlemlerini yönetir, kesme sırasında CPU otomatik olarak bağlam kaydetme/yükleme yapar ve ISR tamamlandığında program normal akışına döner. Bu süreç, donanım tarafından büyük ölçüde desteklendiği için oldukça hızlıdır.

1.3. Kesme İşleyicisi (Interrupt Service Routine - ISR)

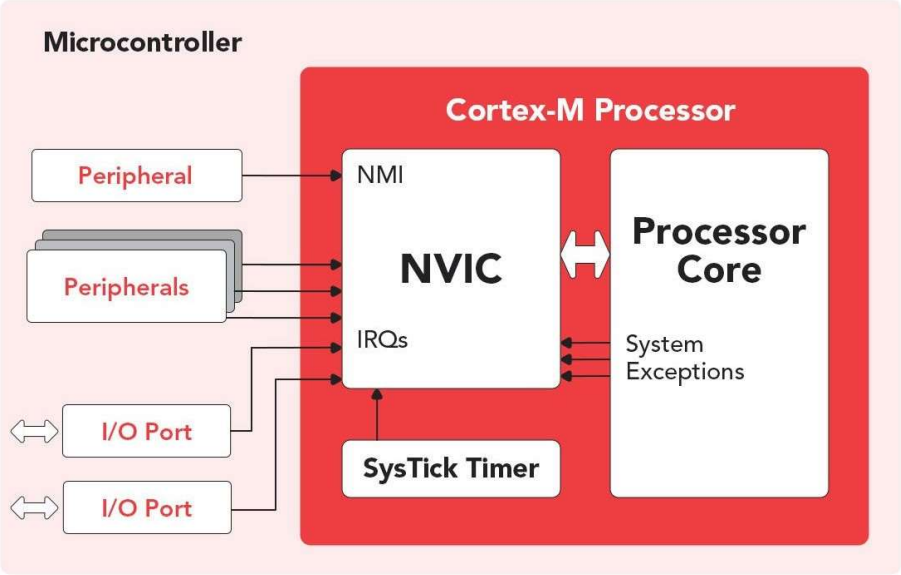
Kesme işleyicisi (ISR), kesme tetiklendiğinde çalıştırılan özel bir fonksiyondur. ISR metodu içinde dikkat edilmesi gereken noktalar;

- ISR içinde uzun süreli işlemler yapılmamalıdır.
- ISR içinde bekleme fonksiyonları (delay) kullanılmamalıdır.
- ISR içinde global değişkenlere erişirken dikkat edilmelidir (volatile tanımlaması!).

1.4. NVIC (Nested Vector Interrupt Controller)

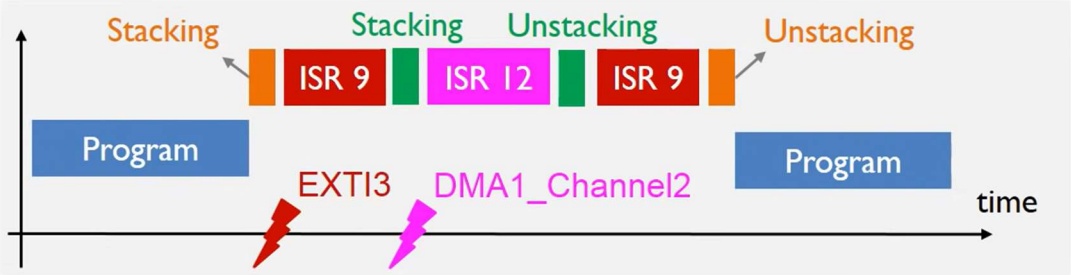
NVIC (Nested Vector Interrupt Controller), STM32 ve ARM tabanlı mikrodenetleyicilerde kesmeleri yöneten özel donanım birimidir. NVIC, kesmelere öncelik atama ve belirli kesmelere izin verme gibi görevleri yerine getirir.

Mikrodenetleyici yapısı içerisinde bulunan herbir çevre birim (peripheral) üzerinden kesme (interrupt) işlemi gerçekleştirilebilir. Pratikte kullanılan mikrodenetleyicinin özelliğine göre kesme işleminin desteklendiği çevre birim sayısı değişiklik gösterebilmektedir.



Şekil 4. ARM Mimarisine Sahip Mikrodenetleyilerdeki NVIC Tabanlı Interrupt Blok Diyagramı

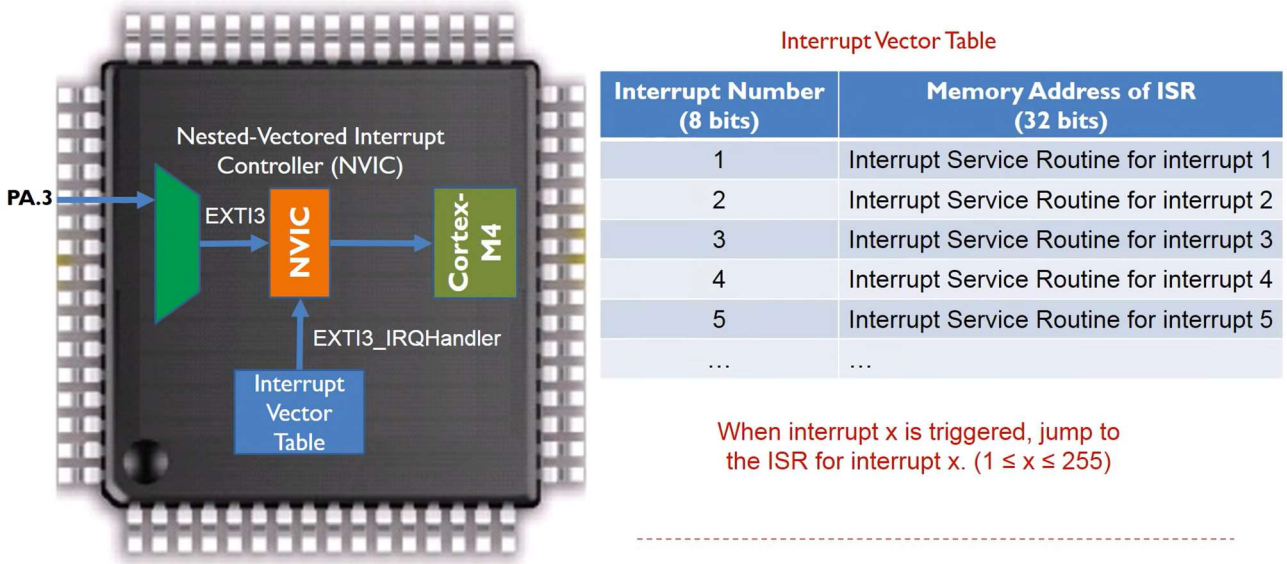
ARM tabanlı mikrodenetleyicilerde NVIC, kesme önceliği düşük olan bir ISR çalışırken, daha yüksek öncelikli bir kesme geldiğinde işlemci onu öncelikli olarak çalıştırır. Yüksek öncelikli kesme bittikten sonra sırasıyla düşük öncelikli kesme ve devamında main program yürütülür.



Şekil 5. ARM Tabanlı Mikrodenetleyicilerde NVIC ile Çoklu Kesme Yönetimi

1.5. Kesme Vektör Tablosu (Interrupt Vector Table, IVT)

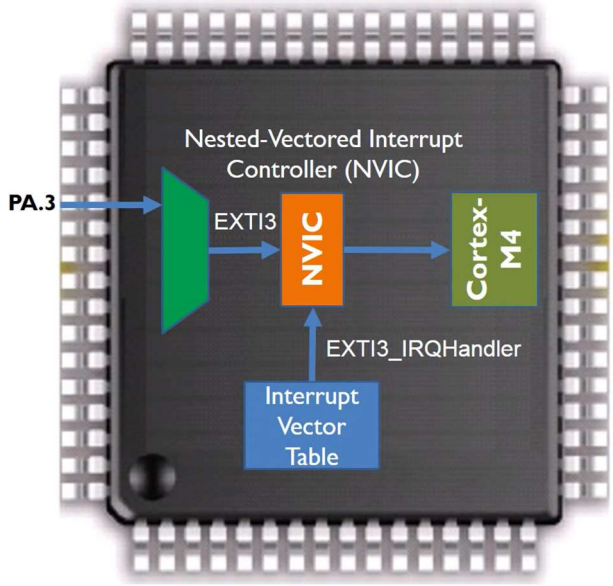
Kesme Vektör Tablosu (IVT), mikrodenetleyicide oluşan kesmelerin hangi fonksiyonlar tarafından işleneceğini belirleyen bir bellek yapısıdır. Her kesme vektör tablosunda bir başlangıç adresi bulunur ve kesme türüne bağlı olarak belirli bir ISR (Interrupt Service Routine) fonksiyonuna yönlendirme yapılır.



Şekil 6. Kesme Vektör Tablo (IVT) Yapısı

IVT, kesmelere karşılık gelen fonksiyon adreslerinden oluşan bir tablodur. Her kesmenin belirli bir vektör adresi vardır ve bu adres, işlemcinin ilgili ISR'yi (Interrupt Service Routine) çağırmasını sağlar. Bir kesme meydana geldiğinde, mikrodenetleyici IVT içindeki ilgili adresi okuyarak, uygun ISR fonksiyonunu çalıştırır.

Örneğin harici kesme (EXTI) olarak yapılandırılan PA3 pini için kesme isteği geldiğinde ilk olarak NVIC bu isteği değerlendirir ve sonrasında bu isteğin Kesme Vektör Tablosunda (IVT) karşılık gelen ISR metodu olan EXT3_IRQHandler metodunu çağırır.



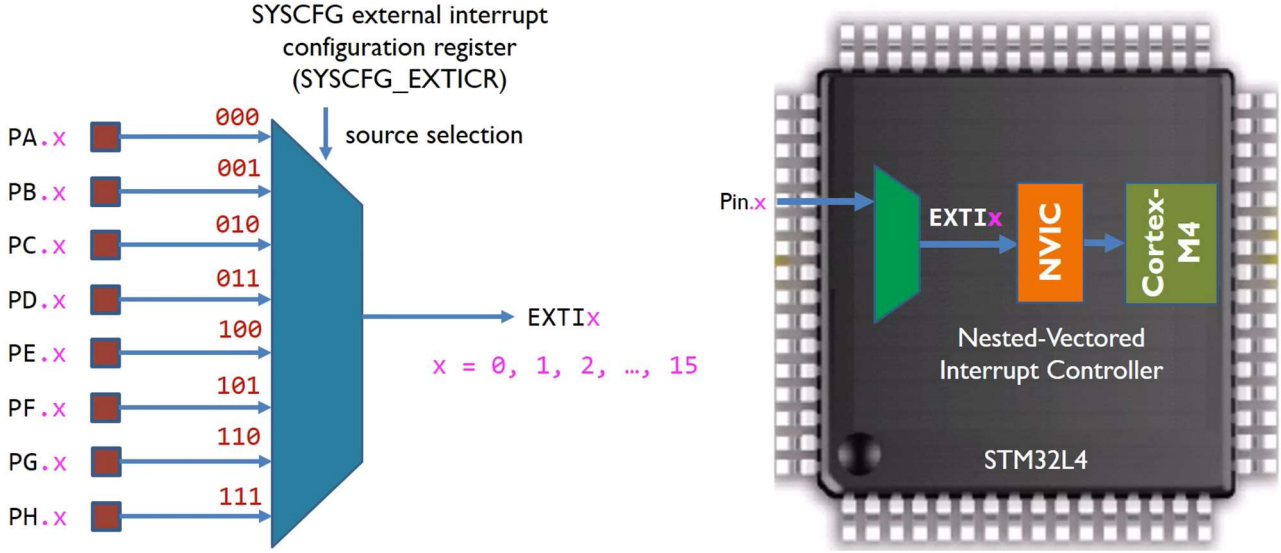
Şekil 7. EXTI3_IRQHandler Kesme Değerlendirme Süreci

1.6. Kesme (Interrupt) Türleri

Mikrodenetleyicilerde kesme türleri aşağıdaki gibi sınıflandırılır:

1.6.1. Harici Kesmeler (External Interrupts)

Mikrodenetleyiciye dış dünyadan gelen kesmelerdir. Örneğin bir butona basıldığında veya hareket sensöründen sinyal geldiğinde kesmenin oluşması gibi. Harici kesmeler genellikle belirli GPIO pinlerine atanır ve EXTI (External Interrupt Controller) tarafından yönetilir.



Şekil 8. Harici Kesme (EXTI) Kaynakları

1.6.2. Dahili Kesmeler (Internal Interrupts)

Mikrodenetleyicinin iç modülleri tarafından üretilen kesmelerdir. Örneğin;

- **Zamanlayıcı (Timer) Kesmesi:** Belirli bir süre geçtiğinde tetiklenen kesme
- **Analog-Dijital Çevirici (ADC) Kesmesi:** ADC dönüşümü tamamlandığında tetiklenen kesme
- **Seri Haberleşme (UART, SPI, I2C) Kesmesi:** Veri gönderildiğinde veya alındığında tetiklenen kesme
- **DMA Kesmesi:** Bellek erişimi tamamlandığında tetiklenen kesme
- **Hata Kesmesi:** Aşırı ısınma veya bellek erişim hatası gibi durumlarda tetiklenen kesme

1.6. Kesme (Interrupt) Kullanım Alanları

Kesme mekanizması, birçok gömülü sistem uygulamasında kritik öneme sahiptir. Bunlardan bazıları;

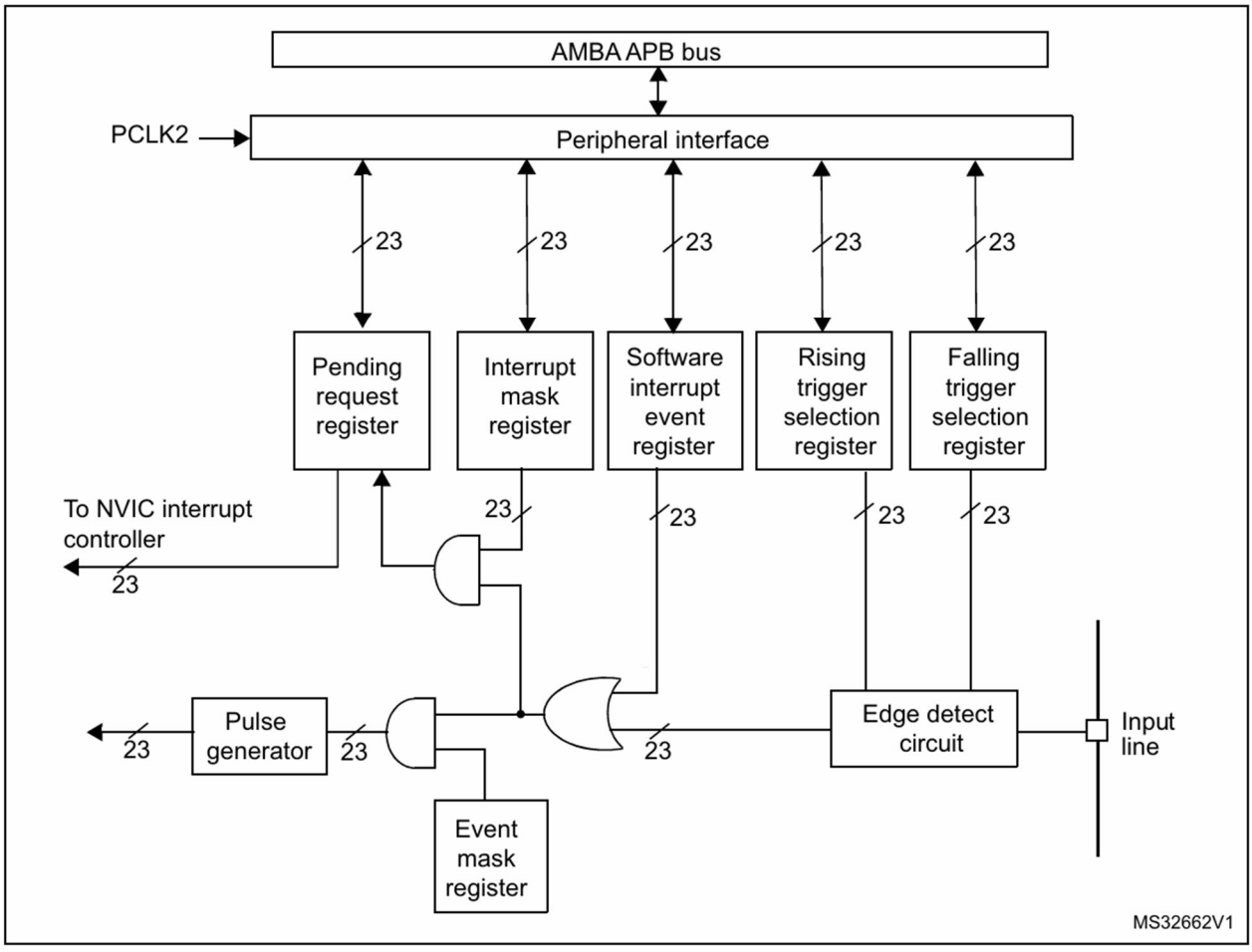
Tablo 2. Kesme Kullanım Alanları

Uygulama	**Kesme Kullanımı**
Gerçek Zamanlı Sistemler	Anlık veri işleme, anında tepki verme
Sensör Tabanlı Uygulamalar	Buton basma, hareket sensörleri
Seri Haberleşme	UART, SPI, I2C üzerinden veri alındığında işlem yapma
Güç Yönetimi	Mikrodenetleyiciyi uyku modundan uyandırma

2. STM32 – EXTI

STM32F407VGT6 mikrodenetleyicisi, ARM Cortex-M4 tabanlı bir yapıya sahiptir ve 84 adede kadar IRQ (Interrupt Request) kanalını destekleyen NVIC (Nested Vector Interrupt Controller) modülü içerir. STM32F407VGT6 mikrodenetleyicisinde yer alan EXTI (External Interrupt/Event Controller), harici kaynaklardan gelen olaylara tepki verilmesini sağlayan güçlü ve esnek bir kesme (interrupt) kontrol birimidir. EXTI modülü, mikrodenetleyicinin GPIO pinleri üzerinden gelen sinyallerle tetiklenen kesmelerin ya da olayların işlenmesini sağlar. EXTI birimi, özellikle gerçek zamanlı sistemlerde, kullanıcı girişlerinde, sensör tetiklemelerinde ve uyku modundan uyanma işlemlerinde yaygın olarak kullanılır. STM32F407VGT6 mikrodenetleyicisinde toplam 16 adet EXTI hattı bulunur. Bunlar;

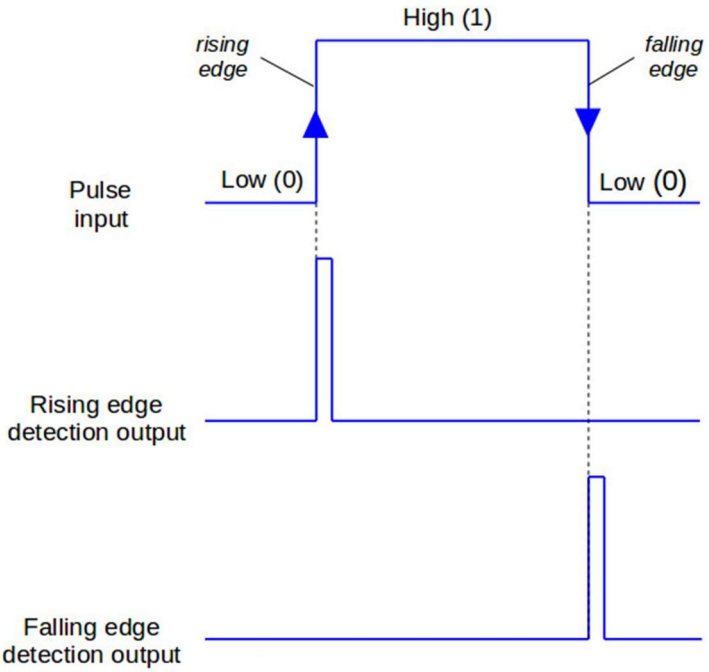
- EXTI0 – EXTI15: 16 adet hattın her biri bir GPIO pinine atanabilir.
- Bu hatlar, PAx, PBx, PCx, PDx, PEx... gibi portların x numaralı pinleri ile eşlenebilir.
- Her bir EXTI hattı aynı anda yalnızca bir GPIO pinine bağlanabilir.



Şekil 9. STM32F407VGT6 Harici Kesme (EXTI) Kontrol Blok Diyagramı

EXTI birimi, farklı kenar tetikleme seçenekleri sunar. Bu, gelen sinyalin değişim yönüne göre kesme oluşturulmasını sağlar. Tetikleme türleri şu şekildedir;

- **Rising Edge (Yükselen Kenar):** Giriş sinyali 0 → 1 olduğunda kesme oluşturur
- **Falling Edge (Düşen Kenar):** Giriş sinyali 1 → 0 olduğunda kesme oluşturur
- **Both Edges (Çift Kenar):** Hem yükselen hem düşen kenarda kesme oluşturur



Şekil 10. Yükselen/Alçalan Kenar Tespiti (Rising/Falling Edge Detection)

EXTI hatları NVIC (Nested Vector Interrupt Controller) aracılığıyla kesme olarak işlemciye bildirilir. NVIC, gelen kesmeyi öncelik sırasına göre işler ve uygun ISR'yi çağırır.

Tablo 3. EXTI – NVIC İlişkisi

EXTI Hattı	**NVIC IRQ**	**Fonksiyon**
EXTI0	EXTI0_IRQn	void EXTI0_IRQHandler(void)
EXTI1	EXTI1_IRQn	void EXTI1_IRQHandler(void)
EXTI2	EXTI2_IRQn	void EXTI2_IRQHandler(void)

3. STM32 – EXTI HAL

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

HAL EXTI Kütüphanesi, Harici bir kesme meydana geldiğinde, EXTI hattının kesme bayrağını temizlemek ve kullanıcıya ait callback fonksiyonunu (ISR) tetiklemek için kullanılır.

HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin)

Metot Açıklaması : Harici bir kesme meydana geldiğinde, EXTI hattının kesme bayrağını temizlemek ve kullanıcıya ait callback fonksiyonunu (ISR) tetiklemek için kullanılır. Kesme vektör fonksiyonu (IRQ Handler) içinde çağrılır.

Parametreler : Kesme oluşan pin değişkeni.

Dönüş Değeri : Dönüş değeri yok (void).

HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)

Metot Açıklaması : Bir EXTI kesmesi oluştuğunda, kullanıcının kendi işlemci kodlarını yazdığı kesme işleyicisidir (ISR - Interrupt Service Routine). Bu fonksiyon kullanıcı tarafından override edilir. HAL kütüphanesi içinde zayıf (__weak) olarak tanımlıdır.

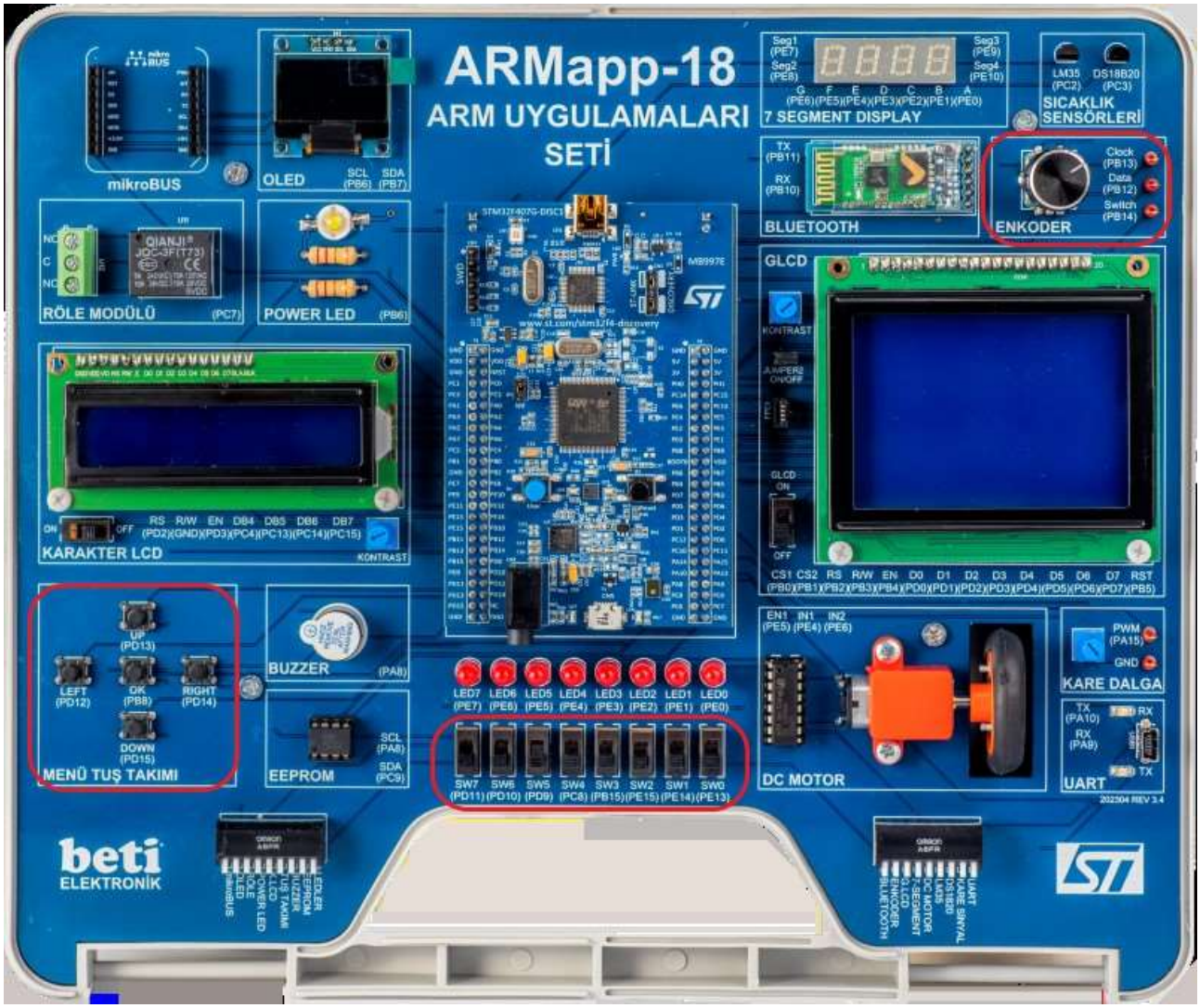
Parametreler : Kesme oluşan pin değişkeni.

Dönüş Değeri : Dönüş değeri yok (void).

4. ARM APP DENEY KİTİ – EXTI

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

- 8-Bit LED Devresi (PE0 – PE7)
- Menü Buton Tuş Takımı (PB8, PD12 – PD15)
- ENKODER (PB12 – PB14)



Şekil 11. ARM APP Deney Kiti

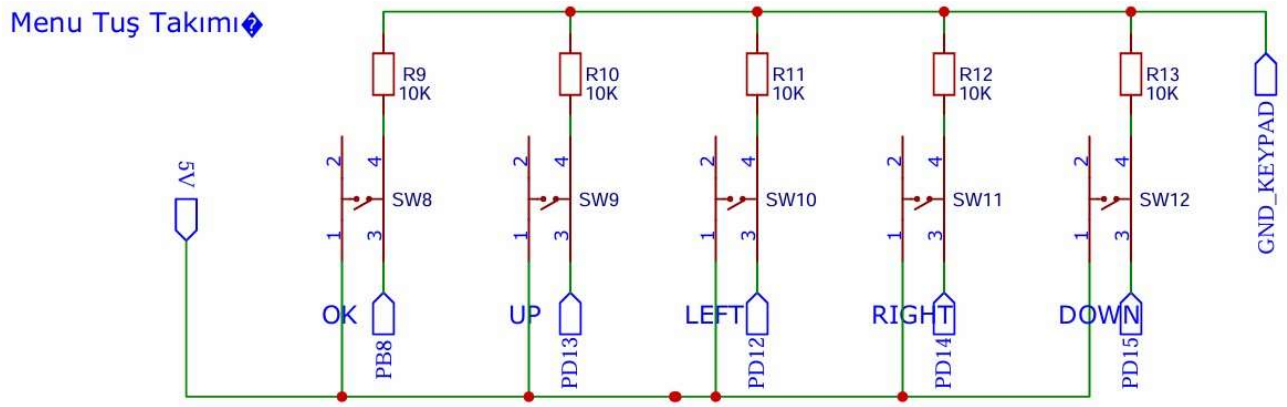
Interrupt (kesme) ve EXTI (harici kesme) çevre birimi, mikrodenetleyici uygulamalarında genellikle dış dünyadan gelen olaylara hızlı ve verimli bir şekilde tepki vermek amacıyla kullanılır. ARM APP Deney Kiti içerisinde bulunan STM32F407VGT6 mikrodenetleyicisi, GPIO pinlerinden birine bağlı olan buton, sensör ya da başka bir donanım biriminden gelen sinyal değişimini EXTI hattı üzerinden algılayarak işlemciye kesme isteği (interrupt request) gönderir. Bu kesme isteği sonucunda işlemci, önceden tanımlanmış olan kesme servis rutinini (ISR) çalıştırarak, gelen olaya bağlı olarak örneğin bir LED'i yakma, bir sayaç başlatma veya sistemi uyandırma gibi işlemleri gerçekleştirir. Bu yapı sayesinde, işlemcinin sürekli olayları kontrol etmesi gerekmeden dış dünyaya anlık ve etkili tepkiler verilebilir.

5. PRATİK UYGULAMA

Laboratuvar uygulamasına başlamadan önce EXTI 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 Menü Tuş Takımı Devresindeki butonları ve 8-Bit LED devresini kullanarak harici kesme EXTI uygulamasını geliştirelim.

STM32CubeIDE'de proje oluşturduktan sonra gelen CubeMX modülü ile pin yapılandırmalarını gerçekleştirebilmek için ilk olarak Menü Buton Takımındaki "OK" butonunun mikrodenetleyiciye bağlı bulunduğu pinin Port ve Numalaralarını ARM APP Deney Kiti Şematik Tasarım dosyasına bakarak belirlenmelidir. Şekil 12'de ilgili devre şeması gösterilmiştir.



Şekil 12. ARM APP Deney Kiti Menü Buton Takımı Devre Şeması

Verilen devre şemasına göre GPIOB portunun 8 numaralı pinini CubeMX modülü üzerinden **GPIO_EXTI8** olarak yapılandırılması gerekmektedir. EXTI parametreleri ise aşağıdaki şekilde belirlenmelidir;

- **GPIO mode:** External Interrupt Mode with Rising edge trigger detection
- **GPIO Pull-up/Pull-down:** No pull-up and no pull-down

Kesme metodu tetiklendiği zaman bir LED'in yanıp sönmesini kontrol etmek için 8-Bit LED devresinde bulunan LED0'ı yani PE7 pinini GPIO_Output olarak yapılandırıp "Generate Code" ikonuna tıklayarak kod geliştirme arayüzüne geçiyoruz.

Öncelike stm32f4xx_it.c dosyası içerisinde yer alan EXTI9_5_IRQHandler metodunu inceleyelim;

```
void EXTI9_5_IRQHandler (void){
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_8);
}
```

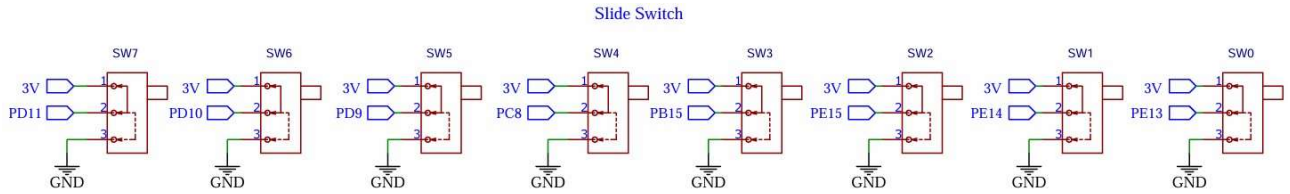
PB8 pininin GPIO_EXTI8 kesme vektör fonksiyonu olan EXTI9_5_IRQHandler metodu içerisinde harici bir kesme meydana geldiğinde, EXTI hattının kesme bayrağını temizlemek ve kullanıcıya ait callback fonksiyonunu (ISR) tetiklemek için HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_8) çağrılmaktadır. Bu metoda sağ tıklayıp "Open Declaration" seçeneği seçildiği zaman IDE stm32f4xx_hal_gpio.c dosyasındaki tanımlama kısmına yönlenicektir. Bu metodu inceleyecek olursak;

```
void HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin){
  /* EXTI line interrupt detected */
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
}
```

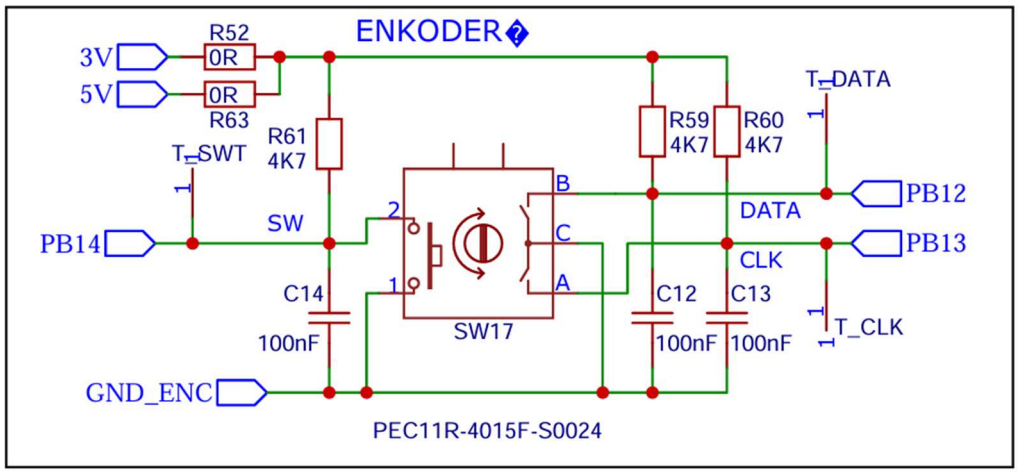
Metot parametre olarak aldığı harici kesmenin oluştuğu pin numarasını kontrol ettikten sonra kesme bayrağını sıfırlayıp kullanıcı tarafından override edilip düzenlenecek olan HAL_GPIO_EXTI_Callback() ISR metodunu çağırmaktadır. Bu noktadan sonra ilgili kesme işlemi gerçekleştiği zaman yapılması istenen komutları bu metot içerisinde tanımlayıp main.c dosyasını içerisinde override ederek kullanabiliriz. Örneğin her kesme isteği geldiğinde PE0 pinine bağlı bulunan LED0 Toggle işlemi yapmak için aşağıdaki kod yapısını kullanabiliriz;

```
#include "main.h"
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
      if(GPIO_Pin == GPIO_PIN_8){
            HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_0);
      }
}

int main(void){
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  while (1){
      //Loop içerisinde herhangi bir komut yok!
  }
}
```

EXTI çevre birimi üzerinden daha farklı harici kesme uygulaması geliştirmek için ARM App Deney Kiti üzerinde yer alan ve Şekil 13/14'de verilen Slide Switch ve Enkoder devre şemaları kullanılabilir.



Şekil 13. ARM APP Deney Kiti Slide Switch Devre Şeması



Şekil 14. ARM APP Deney Kiti Enkoder Devre Şeması

21.03.2025

6. LABORATUVAR DENEY UYGULAMASI

Uygulama isterleri ders esnasında paylaşılacaktır.

7. KAYNAKLAR

[1] https://www.youtube.com/watch?v=uFBNf7F3l60&list=PLrJ4K0hkaFI8xLgXP6MroMl8E5DtBH8-C&index=18

[2] https://www.youtube.com/watch?v=K0vmH2YGbOY&list=PLrJ4K0hkaFI8xLgXP6MroMl8E5DtBH8-C&index=25

[3] https://www.youtube.com/watch?v=uKwD3JuRWeA&list=PLrJ4K0hkaFI8xLgXP6MroMl8E5DtBH8-C&index=26

[4] https://www.youtube.com/watch?v=Fc7yXjt1bC8&list=PLrJ4K0hkaFI8xLgXP6MroMl8E5DtBH8-C&index=30