STM32 ile Gömülü Yazılım

5 Mayıs 2021 Çarşamba 07:50

Giriş
01 GPIO
02 EXTI
03 DMA
04 ADC
05 DAC
06 TIMER
07 PWM
08 UART
09 SPI
10 I2C
11 USB
12 CAN

Giriş

5 Mayıs 2021 Çarşamba 08:02

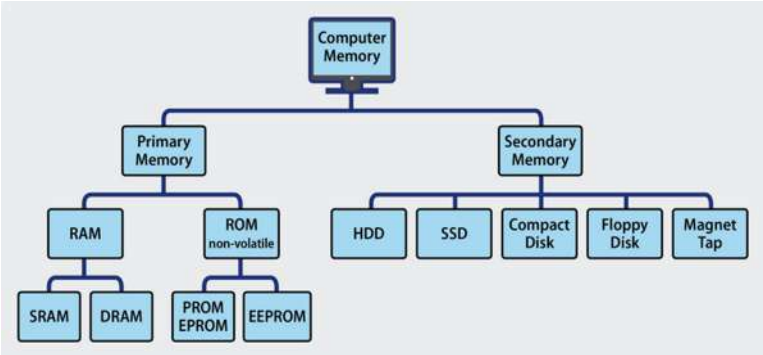
Giriş

Kaynaklar

- Bu belge oluşturulurken https://www.udemy.com/course/stm32f4-discovery-kart-ile-arm-dersleri/ linkteki eğitim kursu izlenirken alınan notlardan oluşmaktadır.

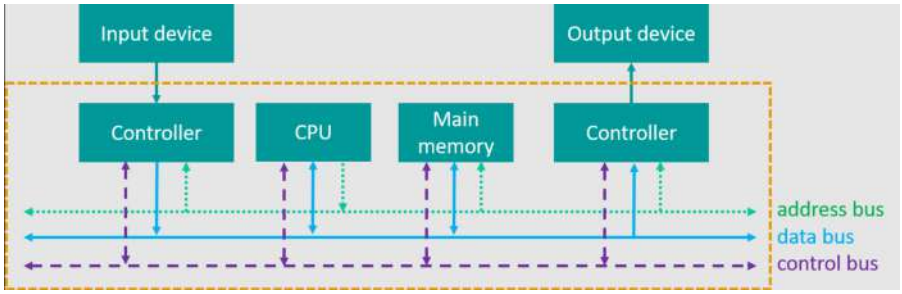
Giriş

- https://www.elektrikport.com/universite/gomulu-sistem-nedir/8658#ad-image-0 ile https://maker.robotistan.com/mikroislemci/ linkteki Gömülü Sistem, Mikroişlemci, Mikrodenetleyici nedir sorularına cevap veren yazıları okuyabilirsiniz.
- https://coskuntasdemir.com/gomulu-yazilimlar/stm32-hal-donanim-soyutlama-katmani-kutuphaneleri.html linkten donanım soyutlama katmanı hakkındaki yazıyı okuyabilirsiniz.
- Mikroişlemci yapısında bir CPU (Central Processing Unit), ön bellek ve Input/Output (Giriş/Çıkış) birimleri bulunan devrelere microprocessor denir.
- Bu üç temel unsur birbirlerine bus, iletişim yolları ile bağlıdır.
- Mikroişlemcinin beyni CPU'dur. Veri akışı ve veri işleme bu birim sayesinde gerçeleşir.
- Bu veri işleme genellikle CPU içerisinde yer alan ALU (Aritmetic Logic Unit)'da uygulanır. Bu birimde sayısal ve lojik işlemler yapılır. Tüm dijital elektronik işlemler CPU'ların en temel işlemleridir.
- CPU'ların içerisinde 8-16-32-64 bitlik registerler bulunmaktadır. Register, bilgilerin geçici sürede depolanmasını sağlarlar.
- CPU'lar, mikroişlemcinin hafızasındaki programları bulma, çağırma ve onları çalıştırma görevi görürler. Veri İşleme Adımları; Veriyi Getirmek (Fetch), Veriyi Çözmek (Dekode), Veriyi İşlemek (Execute), Veriyi Hafızata, Geri Depolamak (Store)
- Merkezi işlem birimi, **üç** birimden oluşur.
    - **Aritmetik Mantık Birimi (ALU)**, hafıza biriminden gelen verilerin işlenmesinde görev alır. Bu işlemlerise aritmetik olarak toplama, çıkarma, bölme ve çapmadır. İkili sayı tabanındaki (binary) mantık işlemleri VE (AND), OR (VEYA) ve bit kaydırma işlemleridir.
    - **Kayıtçılar (Registers)**, hafızadaki veriler ALU tarafından işlenirken kullnılan geçici ve kalıcı saklayıcılardır. Registerler işlemcinin çekirdeğinde olduklarından verilere ulaşmak daha hızlı gerçekle
    - **Kontrol Birimi (CU)**, işlemcinin çalışmasını yönlendiren birimdir. İşlemci içerisindeki ve dışarısındaki birimlerin senkron şekilde çalışmasını sağlar.



- RAM, ROM ve EEPROM, hafızanın temel birimleridir.
  Mikroişlemciye atılan veriler ilk olarak hafızaya gelir ve burada depolanır. CPU'ların doğrudan eriştiği birim bellektir. Bellekte iki tane birincil hafıza birimi vardır.
    - **RAM (Random Access Memory)**, mikroişlemcinin elektrik alması durumunda **geçici hafıza** olarak kullandığı birimdir. Elektrik kesildiği zaman bu veriler silinir ve bir daha kullanılmaz. RAM, diğer hafıza birimleri gibi verileri önceden verilen bir sırayla dizmez. Bu sebeple ismi rastgele erişim bellek olarak konulmuştur. RAM, **Dinamik** Rastgele Erişim Bellek ve **Statik** Rastgele Erişim Bellek olmak üzere **ikiye** ayrılır.
    - **ROM (Read Only Memory)**, **sadece okunabilir** bir bellektir. Elektrik kesildiğinde bu bellekteki veriler silinmez. ROM üzerindeki yazılmış fabrikasyon yazılımlar kullanıcılar tarafından değiştirilip, silinemez.

- **EEPROM (Electronically Erasable Programmable Read-Only Memory)**, elektrik ile defalarca **yazılıp silinebilen** bellektir. Elektrik kesildiğinde bu bellekteki veriler silinmez. Flash belleklerde bir eeprom türüdür.



- Giriş - Çıkış birimleri mikroişlemci ile dış dünyanın sinyaller aracılığı ile haberleştiği birimdir.
  Bu giriş ve çıkışlar; Giriş/ Çıkış portları, harici elektronik birimler, fiziksel cihazlar ve yazılımlar olabilir.
- CPU daki veri akışının aktarılması, bellek ve Giriş/Çıkşı birimlerinin bağlantılarını sağlayan **üç** çeşit bus vardır.

  Address Bus, verinin okunacağı veya verinin yazılacağı bölgeyi belirten adres bilgilerinin taşınmasını sağlar. Tek yönlü bir veri yoludur.
  Data Bus, CPU'dan bellek ve Giriş/Çıkış portlarına veya bu birimlerden CPU'ya çift yönlü bir hat vardır.
  Control Bus, mikroişlemcideki birimler arasında iletişimi sağlayan sinyalleri ileten, kontrol eden veri hattıdır. Her mikroişlemci farklı sayıda Control Bus'a sahiptir.
- Mikroişlemcili bir sistemin içerisinde bulunması gereken temel bileşenlerden RAM, ROM, ALU, kontrol ünitersi ve I/O ünitesini tek bir çip içerisinde barındıran entegre devreye microcontroller denir.
- Mikrodenetleyici, dışarıdan gelen bir veriyi hafızasına alan, derleyen ve sonucunda çıktı elde eden bir bilgisayardır. Mikrodenetleyicilerin yapısında; CPU, RAM, ROM, I/O Portları, Seri ve Paralel Portlar, Zamanlayıcılar, ADC ve DAC çevre birimleri
- Mikrodenetleyiciler gerçek zamanlı işlemlerde oldukça başarılılardır.
- Mikrodenetleyiciler herhangi bir işi çok küçük boyutlarda ve daha düşük enerjide yaparlar.
- Mikroişlemcili ile kontrol edilecek bir sistemi kurmak için gerekli olan minimum donanımda CPU, RAM, I/O bulunmalıdır. Bunlar arasında veri alışverişini sağlamak için ise veri yolu, adres yolu ve kontrol yolu gereklidir. Birimler arasındaki iletişimi sağlayan bu yolları yerleştirmek içinde bir anakart gereklidir.
- Mikrodenetleyici ile kontrol edilecek sistemde ise yukarıda saydığımız birimler tek zaten mikrodenetleyici içerisinde bulunmaktadır. Bu da maliyetin daha düşük olacağı anlamına gelir.
- Mikrodenetleyiciler çok az sayıda ve karmaşık olmayan komutlarla programlanabilen sistemlerdir.
- Mikronetleyiciler hız bakımıından mikroişlemcilerden daha hızlıdır. Güç tüketimi mikrodenetleyicilerde daha azdır. Fiyatları mikroişlemcilere göre daha uygundur.
- Mikroişlemcili sistemlerde harici donanım desteği gerekli iken, mikrodenetleyicilerde bu gereklilik çok azdır.

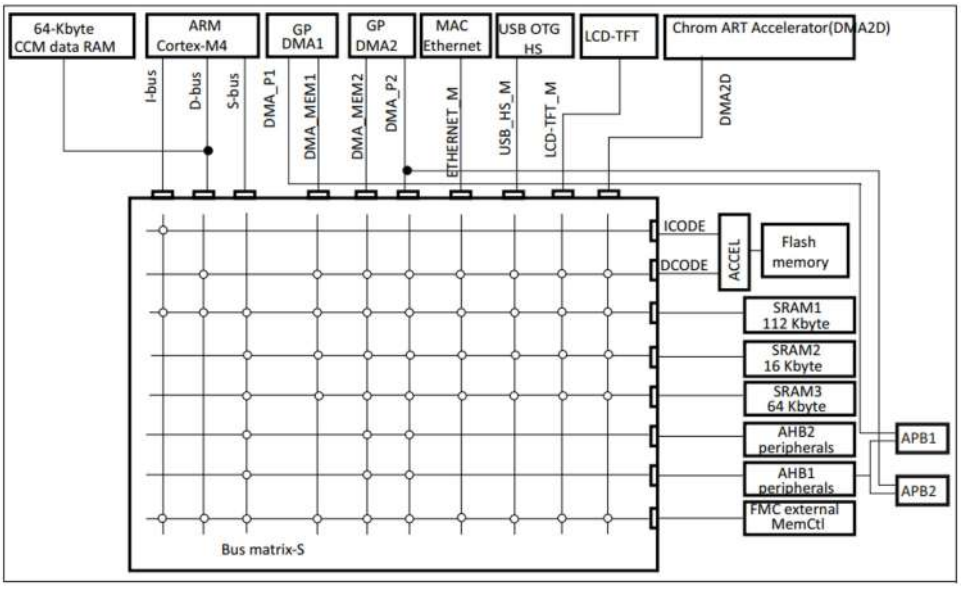
Bellek ve Veri Yolu Mimarisi

- Bellek mimarisi, sistemin bellek kaynaklarının nasıl yapılandırıldığını ve yönetildiğini belirler.
  - **ROM (Read-Only Memory)**
    - Kalıcı veriler ve program kodu saklanır.
    - Mikrodenetleyicinin yeniden başlatılmasında bile veriler silinmez.
  - **Flash Memory**
    - Program ve veri saklama için kullanılan yeniden programlanabilir ROM türüdür.
    - Genellikle firmware ve yazılım güncellemeleri burada saklanır.
  - **SRAM (Static RAM)**
    - Geçici veri depolama için kullanılır.
    - Hızlı erişim süresine sahiptir ve genellikle çalışma sırasında kullanılan veri ve değişkenler burada saklanır.
  - **EEPROM (Electrically Erasable Programmable ROM)**
    - Küçük veri parçalarını kalıcı olarak saklamak için kullanılır.
    - Elektriksel olarak silinebilir ve yeniden programlanabilir.
  - **CCM (Core Coupled Memory)**
    - ARM Cortex-M4 işlemcisine yakın, düşük gecikmeli bellek alanıdır.
    - Yüksek performans gerektiren işlemler için kullanılır.
- Veri yolu mimarisi, farklı bileşenlerin birbirleriyle nasıl iletişim kurduğunu ve veri transferi yaptığını

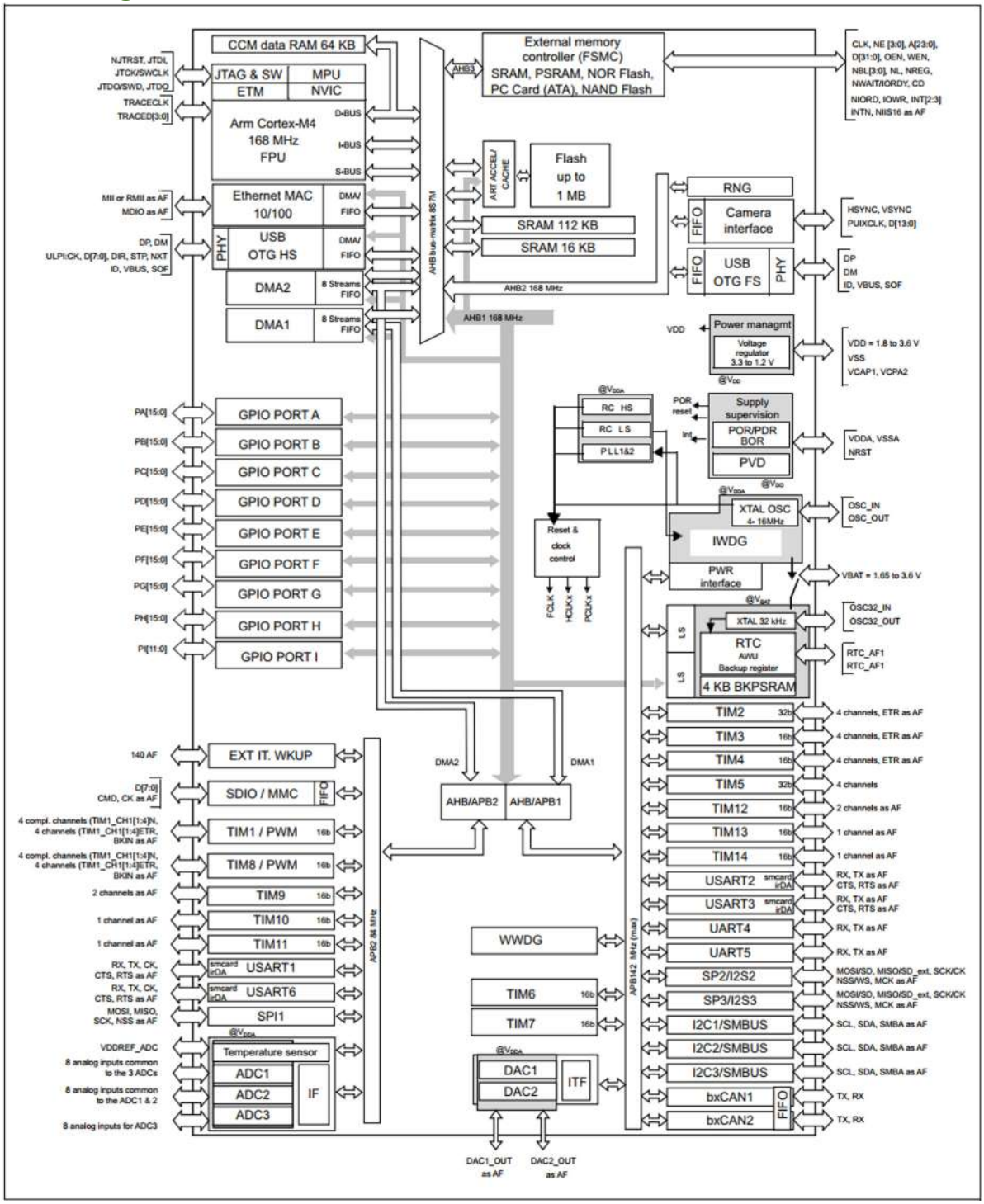
belirler.

- **System Bus (Sistem Veri Yolu)**
  - CPU, bellek ve çevresel birimler arasındaki ana iletişim hattıdır.
  - ARM Cortex-M4 gibi işlemciler genellikle AHB (Advanced High-performance Bus) veya APB (Advanced Peripheral Bus) kullanır.
- **AHB (Advanced High Performance Bus)**
  - Yüksek hızlı veri transferi sağlar.
  - Genellikle CPU, bellek ve yüksek hızlı çevresel birimler arasında kullanılır.
- **APB (Advanced Peripheral Bus)**
  - Daha düşük hız gerektiren çevresel birimler için kullanılır.
  - Daha az karmaşıktır ve daha az güç tüketir.
- **DMA (Direct Memory Access)**
  - CPU müdahalesi olmadan veri transferi yapar.
  - Bellekler ve çevresel birimler arasında hızlı veri transferi sağlar.
- **Bus Matrix:**
  - Birden fazla master ve slave bileşenin aynı anda veri transferi yapmasını sağlar.
  - STM32F407 şemasında görüldüğü gibi, bus matrix-S veri yollarını birbirine bağlar ve verimli veri transferi sağlar.
  - **I-bus**, **D-bus** ve **S-bus** veri yolları ile bellek ve çevresel birimlere bağlanır. I-bus **talimatlar** için, D-bus **veriler** için ve S-bus **sistem veri yolu** için kullanılır.

- STM32F407 mikrodenetleyicisinin sistem mimarisi, 32-bit çok katmanlı AHB veri yolu matrisi üzerine kuruludur. Bu matriste sekiz master ve yedi slave bileşen bulunur:
  - Master bileşenler, veri yolu üzerinde kontrol yetkisine sahiptir ve veri transferini başlatabilir. İşlemci veya DMA gibi master bileşenler, veri yolu üzerinden slave bileşenlere erişim talimatları göndererek veri okuyabilir veya yazabilir.
    - Cortex®-M4 with FPU core I-bus, D-bus and S-bus
    - DMA1 memory bus
    - DMA2 peripheral bus
    - Ethernet DMA bus
    - USB OTG HS DMA bus
  - Slave bileşenler, genellikle veri depolama veya belirli bir işlevi yerine getirme rolüne sahiptir. Bunlar, kendi başlarına aktif olarak veri yolu üzerinde veri transferi başlatamazlar. Bu yüzden veri alışverişi yapabilmek için bir master bileşenin aktivasyonuna ve yönlendirmesine ihtiyaç duyarlar.
    - Internal Flash memory ICode bus
    - Internal Flash memory DCode bus
    - Main internal SRAM1 (112 KB)
    - Auxiliary internal SRAM2 (16 KB)
    - AHB1 peripherals including AHB to APB bridges and APB peripherals
    - AHB2 peripherals
    - FSMC

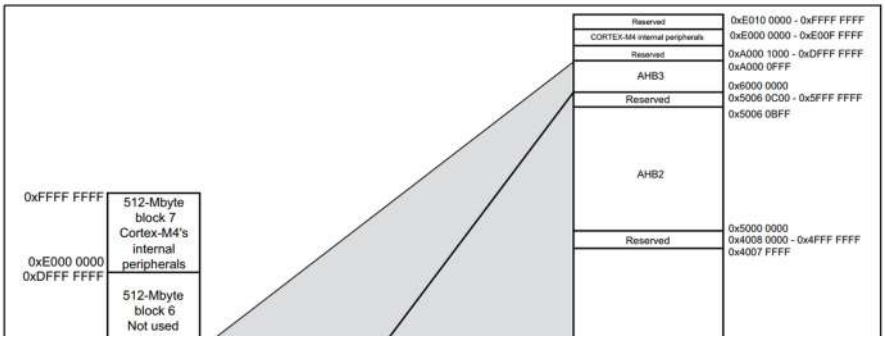


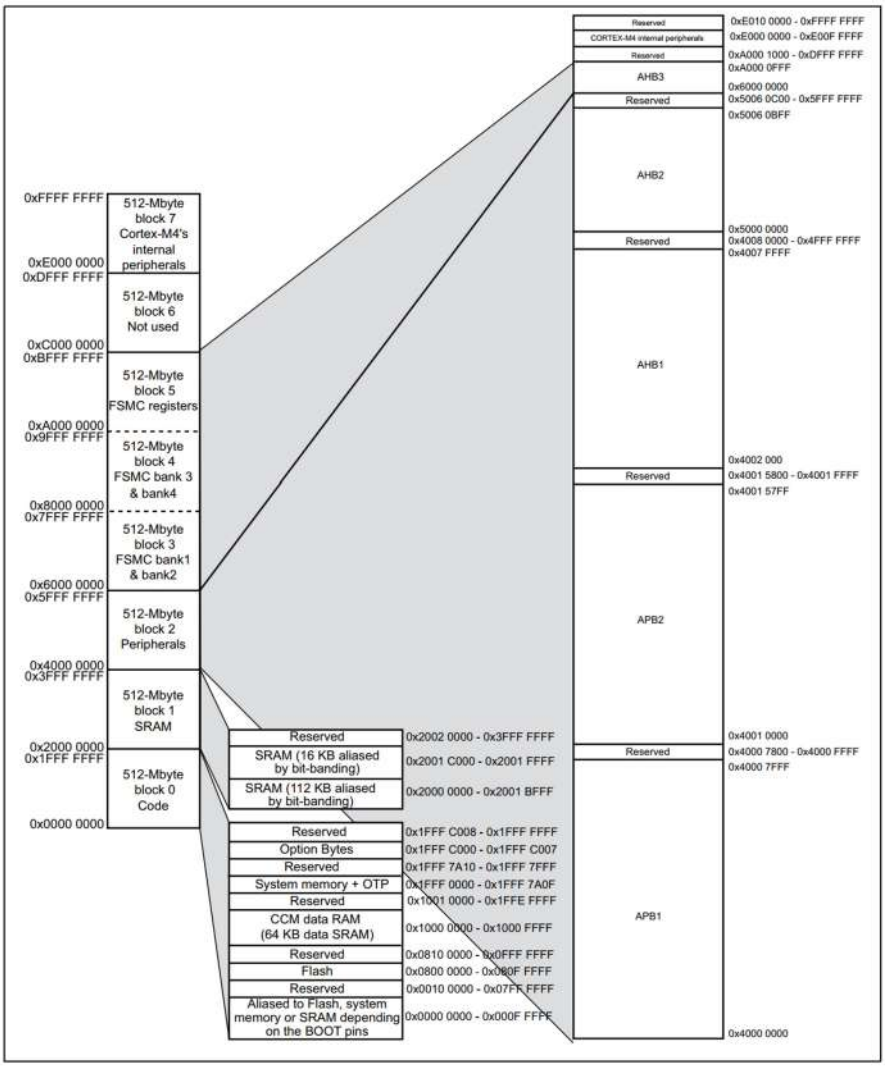
Block Diagram



Memory Map

- İşlemcimiz veri yolu 32 bittir.
- 2^32 işleminden 4.294.967.296 sonucu çıkar ve bu 4GB adresleyebilme kapasitesi olduğunu söyler. Bu sayıyı decimalden hep formatına çevirirsek 1 0000 0000 sayısını verir ve bundan 1 çıkarırsak FFFF FFFF sayısını verir. Adres uzayım 0x00000000 dan başlar 0xFFFFFFFF'a kadar devam eder.
- Hex formatta her bir rakam 4 bite denk gelir.




0xFFFF FFFF
512-Mbyte block 7 Cortex-M4's internal peripherals
0xE000 0000
0xDFFF FFFF
512-Mbyte block 6 Not used
0xC000 0000
0xBFFF FFFF
512-Mbyte block 5 FSMC registers
0xA000 0000
0x9FFF FFFF
512-Mbyte block 4 FSMC bank 3 & bank4
0x8000 0000
0x7FFF FFFF
512-Mbyte block 3 FSMC bank1 & bank2
0x6000 0000
0x5FFF FFFF
512-Mbyte block 2 Peripherals
0x4000 0000
0x3FFF FFFF
512-Mbyte block 1 SRAM
0x2000 0000
0x1FFF FFFF
512-Mbyte block 0 Code
0x0000 0000
Reserved
0x2002 0000 - 0x3FFF FFFF
SRAM (16 KB aliased by bit-banding)
0x2001 C000 - 0x2001 FFFF
SRAM (112 KB aliased by bit-banding)
0x2000 0000 - 0x2001 BFFF
Reserved
0x1FFF C008 - 0x1FFF FFFF
Option Bytes
0x1FFF C000 - 0x1FFF C007
Reserved
0x1FFF 7A10 - 0x1FFF 7FFF
System memory + OTP
0x1FFF 0000 - 0x1FFF 7A0F
Reserved
0x1001 0000 - 0x1FFE FFFF
CCM data RAM (64 KB data SRAM)
0x1000 0000 - 0x1000 FFFF
Reserved
0x0810 0000 - 0x0FFF FFFF
Flash
0x0800 0000 - 0x080F FFFF
Reserved
0x0010 0000 - 0x07FF FFFF
Aliased to Flash, system memory or SRAM depending on the BOOT pins
0x0000 0000 - 0x000F FFFF
Reserved
0xE010 0000 - 0xFFFF FFFF
CORTEX-M4 internal peripherals
0xE000 0000 - 0xE00F FFFF
Reserved
0xA000 1000 - 0xDFFF FFFF
AHB3
0xA000 0FFF
0x6000 0000
Reserved
0x5006 0C00 - 0x5FFF FFFF
AHB2
0x5006 0BFF
0x5000 0000
Reserved
0x4008 0000 - 0x4FFF FFFF
AHB1
0x4007 FFFF
0x4002 000
Reserved
0x4001 5800 - 0x4001 FFFF
APB2
0x4001 57FF
0x4001 0000
Reserved
0x4000 7800 - 0x4000 FFFF
APB1
0x4000 7FFF
0x4000 0000

01 GPIO

5 Mayıs 2021 Çarşamba 08:02

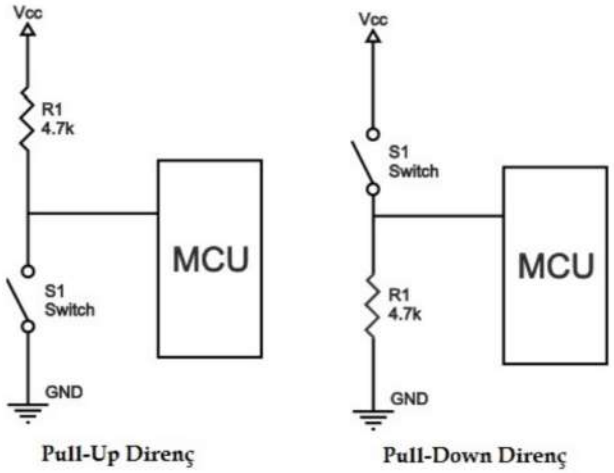
01 GPIO

Giriş

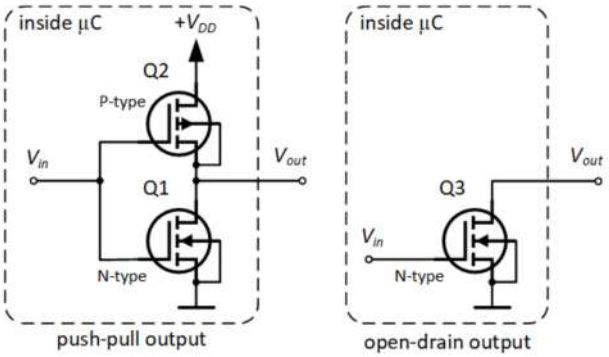
- Butonlar ve anahtarlar mikrodenetleyiciye giriş pini üzerinden lojik 1 ve lojik 0 olarak bilgi girişini sağlayan mekanik elemanlardır.

Bağlantılar

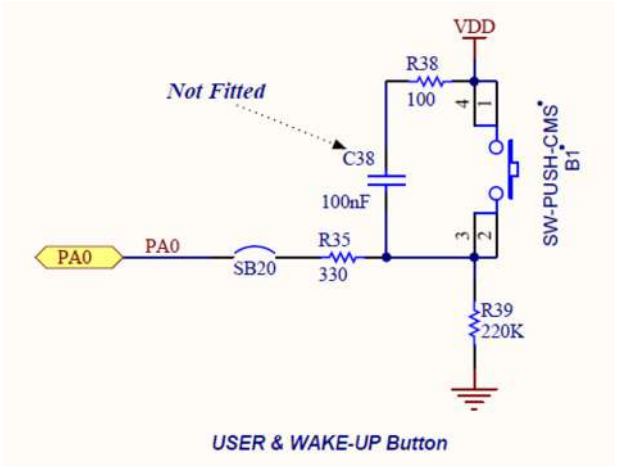
- **Pull-Up** bağlantıda GPIO girişi direnç üzerinden + beslemeye (VCC/VDD) bağlanır.
  - Butona basılmadığı durumda GPIO girişinde lojik 1 vardır.
  - Butona basıldığı durumda girişe 0V (lojik 0) uygulanmış olur.
- **Pull-Down** bağlantıda, GPIO girişi direnç üzerinden GND ye bağlanır.
  - Butona basılmadığı durumda girişte lojik 0 bulunur.
  - Butona basıldığı durumda buton üzerinden lojik 1 uygulanmış olur.



- **Push-pull** çıkışlar, bir pini yüksek veya düşük seviyeye aktif bir şekilde çekebilir.
  Bu yapılandırmada, çıkış aşamasında genellikle iki transistör bulunur: biri çıkışı VCC'ye (yüksek voltaj) çekerken, diğeri çıkışı toprak seviyesine (düşük voltaj) çeker. Bu iki transistör birbirinin tamamlayıcısı olarak çalışır; **biri açıkken diğeri kapalıdır**. Bu sayede çıkış hızlı bir şekilde yüksekten düşüğe veya düşükten yükseğe geçebilir. Push-pull çıkışlar genellikle LED'ler veya iç direnci olan diğer yüklerle kullanılır.
  - Daha hızlı geçiş hızları sağlar çünkü çıkış direk olarak hem yüksek hem de düşük voltaj seviyelerine çekilebilir.
  - Çıkışta bulunan iki transistör sayesinde daha yüksek akım taşıyabilir.
  - Kısa devre riski daha yüksektir. Eğer her iki transistör de yanlışlıkla aynı anda aktif olursa, Vcc ve toprak arasında direkt bir bağlantı oluşur.
- **Open drain** (veya open collector) çıkışında, çıkış noktasında yalnızca bir transistör bulunur ve bu transistör çıkışı yalnızca toprağa (düşük seviyeye) çekebilir. Yüksek seviyeye çıkması için harici bir çekme direncine (pull-up resistor) ihtiyaç duyar. Bu direnç, çıkışı VCC'ye çekerken transistör kapalıdır.
  - Birden fazla open drain çıkışı aynı hat üzerinde bağlanabilir (örneğin I^2C gibi veri yollarında kullanılır). Bu, birbiri ile iletişim halinde olan cihazların çakışmadan veri alışverişinde bulunmasını sağlar.
  - Transistör yalnızca tek bir yönde çalıştığı için tasarım daha basit olabilir.
  - Harici bir çekme direncine ihtiyaç duyar.
  - Genellikle push-pull çıkışlara göre daha yavaş geçiş hızlarına sahiptir.

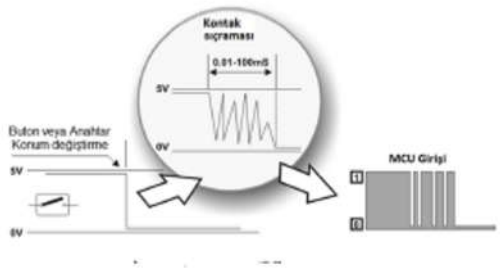


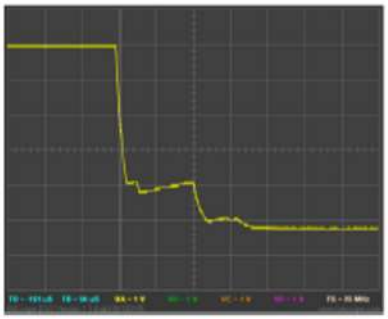
- Resimde görüldüğü gibi STM32F446RE Nucleo bordunda kullanıcı butonu A portunun 0. pinine bağlı ve pull down durumundadır.



Ark

Buton ve anahtarda konum değiştiğinde arktan dolayı mikrodenetleyici girişinde çok sayıda istenmeyen lojik değer oluşur. Bu duruma ark deniyor.





- Ark problemini https://akademi.robolinkmarket.com/buton-arki-nedir-nasil-cozulur/ linkten donanımsal ve yazılımsal olarak paylaşılan çözümleri inceleyip uygulayabiliriz.

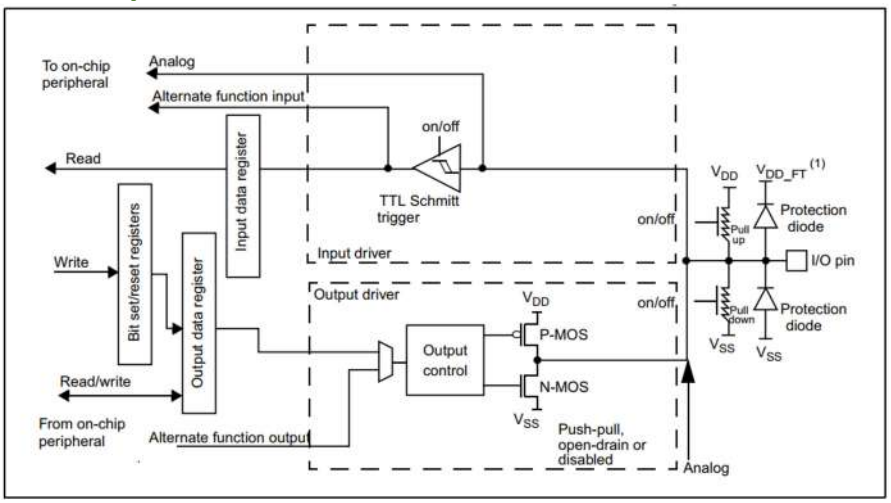
Kontrol Yöntemleri

- GPIO pinlerini kontrol etmek için iki temel yöntem vardır. Bunlar interrupt ve polling.
  İşlemcinin ve uygulamanın gereksinimlerine bağlı olarak her iki yöntem de tercih edilebilir.
- **Polling yöntemi**, mikrodenetleyici tarafından belirli bir durumun sürekli olarak kontrol edilmesine dayanır. Örneğin, bir GPIO pininin durumu sürekli bir döngü içinde kontrol edilebilir.
  Avantajları basit ve doğrudan bir yaklaşım ile donanım ve yazılım karmaşıklığı düşüktür.
  Dezavantajları sürekli olarak işlem yaparak sistem kaynaklarını tüketir. Anında tepki verme yeteneği sınırlıdır.
  Basit uygulamalarda veya sürekli düşük güç tüketimi gerektiren durumlarda tercih edilebilir. Kesmelerin işlemi engelleyeceği veya karmaşık hale getireceği durumlarda kullanışlıdır. Zamanlama veya hızlı tepki gerekli olmadığında kullanılabilir.
- **Interrupt yöntemi**, bir olay (örneğin, GPIO pininin durum değiştirmesi) gerçekleştiğinde normal programın çalışmasını kesip belirli bir kesme servis rutinini çalıştırarak olaya tepki verir.
  Avantajları düşük enerji tüketimi, çünkü işlemci, beklenmeyen olaylar olana kadar bekler. Anında tepki verme yeteneği yüksektir.
  Dezavantajları, Kod karmaşıklığı ve debug işlemleri artabilir. Zamanlaması hassas olabilir ve bazı durumlarda kesmeler birbirini engelleyebilir.

Anında tepki gerektiren durumlarda (örneğin, düğme basıldığında). Enerji tüketiminin daha fazla toleranslı olduğu durumlarda. Sık sık kontrol etmenin pratik olmadığı durumlar için uygun bir seçenektir.

- Genel olarak, interrupt yöntemi, enerji tüketimi veya anında tepki gereksinimleri gibi durumlarda daha uygun olabilirken, sadece belirli durumlarda kontrol yapılması gereken basit uygulamalarda polling sorgulama kullanılabilir.

Birim Yapısı



Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOx_MODER** (where x = C..I/J/K)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**GPIOx_ OTYPER** (where x = A..I/J/K)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_ OSPEEDR** (where x = A..I/J/K except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOx_PUPDR** (where x = C..I/J/K)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = A..I/J/K)	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = A..I/J/K)	Reserved																ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = A..I/J/K)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR** (where x = A..I/J/K)	Reserved															LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL** (where x = A..I/J/K)	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH** (where x = A..I/J)	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **GPIOx_MODER** (Mode Register), her pin için iki bit kullanılır. Giriş, çıkış, alternatif fonksiyon veya analog

modunu seçmek için kullanılır.

- **GPIOx_OTYPER** (Output Type Register), her pin için bir bit kullanılır. Push-pull veya Open-drain çıkış tipini seçmek için kullanılır.
- **GPIOx_OSPEEDR** (Output Speed Register), her pin için iki bit kullanılır. Çıkış hızını kontrol etmek için kullanılır.
- **GPIOx_PUPDR** (Pull-up/Pull-down Register), her pin için iki bit kullanılır. Dahili pull-up veya pull-down direncini etkinleştirmek için kullanılır.
- **GPIOx_IDR** (Input Data Register), her pin için bir bit kullanılır. Pinin mevcut durumunu okumak için kullanılır.
- **GPIOx_ODR** (Output Data Register), her pin için bir bit kullanılır. Çıkış durumunu ayarlamak veya temizlemek için kullanılır.
- **GPIOx_BSRR** (Bit Set/Reset Register), her pin için iki bit içerir. Bir GPIO pininin durumunu set etmek veya resetlemek için kullanılır.
- **GPIOx_LCKR** (Lock Register), her pin için bir bit içerir. GPIO pin konfigürasyonunun kilitlenmesini sağlar.
- **GPIOx_AFRL** ve **GPIOx_AFRH** (Alternate Function Low/High Register), her biri 32-bit uzunluğunda iki registerdır ve her pin için dört bit içerir. GPIO pinlerinin alternatif fonksiyonlarını belirlemek için kullanılır.

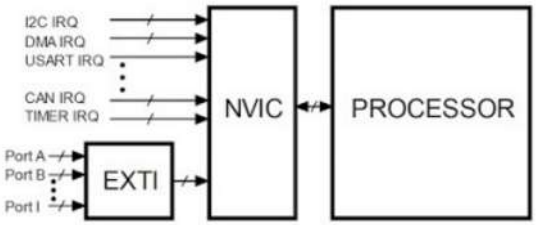
02 EXTI

5 Mayıs 2021 Çarşamba 08:02

02 EXTI

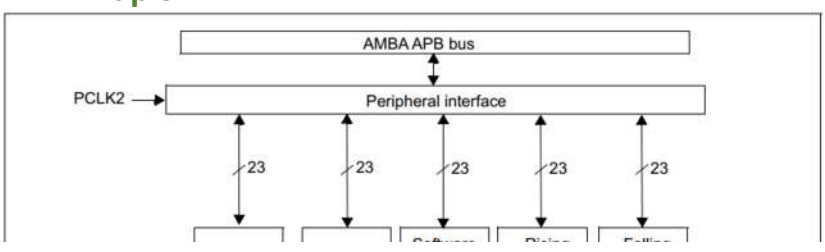
Giriş

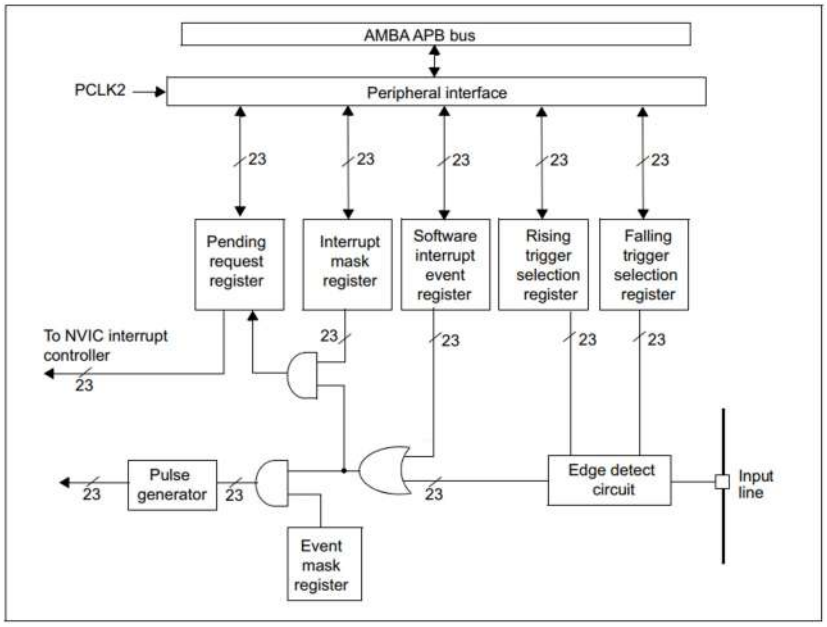
- Polling method sürek işleciyi meşgul ettiği işlemlerdir. Bu işlemler için **while**, **for** döngüleri kullanılıyor.
- Önceliği yüksek işlerin mikrodenetleyici tarafından ana program akışını keserek yapılmasına interrupt denir.
- Eğer bir kesme kaynağından mikrodenetleyiciye uyarı gelirse mikrodenetleyici yapmakta olduğu işi bekletir, kesme alt programına gider, o programı icra eder, daha sonra ana programda kaldığı yerden devam eder.
- Kesmeleri genellikle **çok hızlı** yapılması gereken işlemlerde, **anlık tepki** verilmesi gereken yerlerde kullanırız.
- Harici bir kaynaktan oluşan olaylardan dolayı meydana gelen kesmelere, **harici kesmeler** denir. Harici kaynak olarak, dış ortamdan pinler vasıtasıyla gelecek kesme ve kendi içindeki donanımlardan gelen kesmeleri anlayabiliriz.
- Karmaşık kesme isteklerinin işlemciye sürekli yük getirmemesi için işlemci içerisinde özel bir donanım bloğu oluşturulmuştur. Bu donanıma interrupt controller adı verilir.
- Kesme kontrolörü haklı bir sebeple gelen kesme isteği neticesinde düzgün işleyen programı askıya alarak kesme fonksiyonu (interrupt function) olarak adlandırılan özel kod parçasını işlemeye başlar.
- Kesme fonksiyonunun işletilmesinin bitiminde program kaldığı yerden çalışmaya devam eder.
- NVIC kontrolör mikroişlemci içerisindeki önemli donanım kesmelerini (DMA, USART, CAN, I2C ve Timer gibi) ve ayrıca External Interrupt (EXTI) adı verilen donanım vasıtasıyla portlardan gelen kesmeleri kontrol eder.



- İnterrupt kullanmak için üç farklı yapıyı ayarlamak gerekiyor. SYSCFG, EXTI ve NVIC yapılarını ayarlanarak interrupt kullanabilirim. İlk ikisi MCU ile alakalı iken üçüncüsü MPU ile alakalıdır.
- Hatlarımız dışardan multiplexer sayesinde içeriğiye bağlanıyor. Bu bağlanan hatlar aslında içeride EXTI Line olarak tanımlanıyor. Bu bağlantının birinden interrupt bekleniyor ve öncesinden SYSCFG ile söylemem gerekiyor.
- Aynı hatta bağlı yapıda birden fazla interrupt olamaz.
- Daha sonra hatlardan gelen interrupt görmemesi için maskelenmiş durumda olan hattı, EXTI ile önce kaldırmamız gerekiyor sonra gelen sinyalde yükselen kenarda mı yoksa düşen kenarda mı interrupt girmesini istediğimi belirtmem gerekiyor.
- En son bu gelen Interruptlar NVIC yapısında toplanıyorlar. Bu yapı ile birleşen hatlara IRQ olarak adlandırıyoruz.
- NVIC ayarlamasında MPU kısmında Interruptın geleceğini söylemem gerekiyor bunun için işlemcinin kendi datashetini kullanarak ulaşabiliriz.

Birim Yapısı





- STM32 mikrodenetleyicilerindeki interrupt kaynaklarının ve öncelik yapılandırmalarının bir özetini aşağıdaki tabloda göstermektedir. Her bir satır, belirli bir kesme kaynağına dair detayları içerir.
    - **Position,** kesmenin öncelik tablosundaki sıralamasını gösterir. Kesme kaynakları genelde pozisyon numarasıyla tanımlanır. Bu sıralama, mikrodenetleyicinin NVIC (Nested Vector Interrupt Controller) yapılandırmasıyla ilgilidir.
    - **Priority,** kesmenin varsayılan önceliğini belirtir.
    - **Type of Priority**, kesmenin öncelik türünü ifade eder.
        - Bazı kesmeler (örneğin Reset, NMI, HardFault) sabit öncelikli **fixed** olup değiştirilemezdir.
            - Reset: Sistem resetlendiğinde çalışır.
            - HardFault: Ciddi hatalar (örneğin, yanlış bellek erişimi) durumunda çalışır.
        - Çoğu çevresel birim kesmeleri (örneğin, EXTI, CAN, USART v.b) ise yazılım tarafından ayarlanabilir **settable** yapıdadır. Kullanıcı, uygulamanın gereksinimlerine göre bu kesmelerin **Preemption Priority** ve **SubPriority** değerlerini ayarlayabilir.
    - **Acronym,** kesmenin kısa adı verilmiştir. Bu, kesmenin hangi kaynağa ait olduğunu anlamamızı sağlar. Örneğin; NMI: Non-Maskable Interrupt (maskelenemez kesme), EXTI0: EXTI hattı üzerinden tetiklenen dış kesme, CAN1_RX0: CAN1 modülünden gelen RX kesmesi
    - **Address, k**esmenin vektör tablosundaki vektör adresini gösterir.

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	fixed	HardFault	All class of fault	0x0000 000C
-	0	settable	MemManage	Memory management	0x0000 0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C - 0x0000 002B
-	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
-	4	settable	Debug Monitor	Debug Monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	5	settable	PendSV	Pendable request for system service	0x0000 0038
-	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	PVD through EXTI line detection interrupt	0x0000 0044
2	9	settable	TAMP_STAMP	Tamper and TimeStamp interrupts through the EXTI line	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C

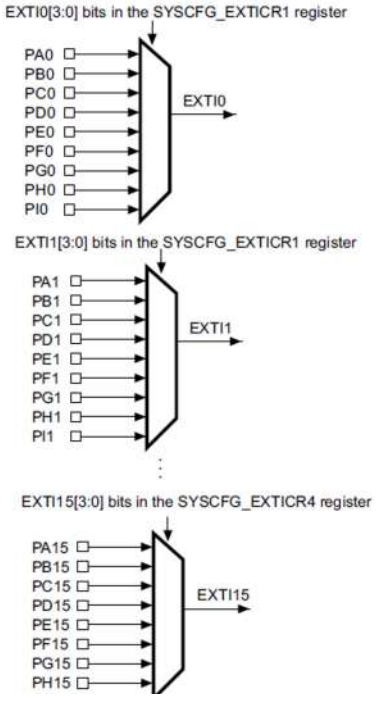
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000 0068
11	18	settable	DMA1_Stream0	DMA1 Stream0 global interrupt	0x0000 006C
12	19	settable	DMA1_Stream1	DMA1 Stream1 global interrupt	0x0000 0070
13	20	settable	DMA1_Stream2	DMA1 Stream2 global interrupt	0x0000 0074
14	21	settable	DMA1_Stream3	DMA1 Stream3 global interrupt	0x0000 0078
15	22	settable	DMA1_Stream4	DMA1 Stream4 global interrupt	0x0000 007C
16	23	settable	DMA1_Stream5	DMA1 Stream5 global interrupt	0x0000 0080
17	24	settable	DMA1_Stream6	DMA1 Stream6 global interrupt	0x0000 0084
18	25	settable	ADC	ADC1, ADC2 and ADC3 global interrupts	0x0000 0088
19	26	settable	CAN1_TX	CAN1 TX interrupts	0x0000 008C
20	27	settable	CAN1_RX0	CAN1 RX0 interrupts	0x0000 0090
21	28	settable	CAN1_RX1	CAN1 RX1 interrupt	0x0000 0094
22	29	settable	CAN1_SCE	CAN1 SCE interrupt	0x0000 0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK_TIM9	TIM1 Break interrupt and TIM9 global interrupt	0x0000 00A0
25	32	settable	TIM1_UP_TIM10	TIM1 Update interrupt and TIM10 global interrupt	0x0000 00A4
26	33	settable	TIM1_TRG_COM_TIM11	TIM1 Trigger and Commutation interrupts and TIM11 global interrupt	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000 00B8
31	38	settable	I2C1_EV	I^2C1 event interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I^2C1 error interrupt	0x0000 00C0
33	40	settable	I2C2_EV	I^2C2 event interrupt	0x0000 00C4
34	41	settable	I2C2_ER	I^2C2 error interrupt	0x0000 00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000 00D0
37	44	settable	USART1	USART1 global interrupt	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_Alarm	RTC Alarms (A and B) through EXTI line interrupt	0x0000 00E4
42	49	settable	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000 00E8
43	50	settable	TIM8_BRK_TIM12	TIM8 Break interrupt and TIM12 global interrupt	0x0000 00EC
44	51	settable	TIM8_UP_TIM13	TIM8 Update interrupt and TIM13 global interrupt	0x0000 00F0
45	52	settable	TIM8_TRG_COM_TIM14	TIM8 Trigger and Commutation interrupts and TIM14 global interrupt	0x0000 00F4
46	53	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000 00F8
47	54	settable	DMA1_Stream7	DMA1 Stream7 global interrupt	0x0000 00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000 0100
49	56	settable	SDIO	SDIO global interrupt	0x0000 0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000 0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000 010C
52	59	settable	UART4	UART4 global interrupt	0x0000 0110
53	60	settable	UART5	UART5 global interrupt	0x0000 0114
54	61	settable	TIM6_DAC	TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	0x0000 0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000 011C
56	63	settable	DMA2_Stream0	DMA2 Stream0 global interrupt	0x0000 0120
57	64	settable	DMA2_Stream1	DMA2 Stream1 global interrupt	0x0000 0124
58	65	settable	DMA2_Stream2	DMA2 Stream2 global interrupt	0x0000 0128
59	66	settable	DMA2_Stream3	DMA2 Stream3 global interrupt	0x0000 012C
60	67	settable	DMA2_Stream4	DMA2 Stream4 global interrupt	0x0000 0130
61	68	settable	ETH	Ethernet global interrupt	0x0000 0134
62	69	settable	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000 0138
63	70	settable	CAN2_TX	CAN2 TX interrupts	0x0000 013C
64	71	settable	CAN2_RX0	CAN2 RX0 interrupts	0x0000 0140

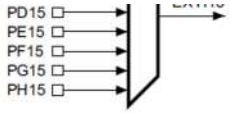
- Kesme öncelikleri iki farklı kavramla ifade edilir.
    - **Preemption Priority**, daha düşük bir Preemption Priority değerine sahip bir kesme, daha yüksek bir Preemption Priority değerine sahip bir kesmenin işlenmesini kesebilir. Kesme sırasındaki en üst seviye

- Kesme öncelikleri iki farklı kavramla ifade edilir.
    - **Preemption Priority**, daha düşük bir Preemption Priority değerine sahip bir kesme, daha yüksek bir Preemption Priority değerine sahip bir kesmenin işlenmesini kesebilir. Kesme sırasındaki en üst seviye kontrolü sağlar. Bu, daha kritik işlemlerin daha az kritik olanları kesintiye uğratmasına izin verir.
    - **SubPriority**, Aynı Preemption Priority seviyesine sahip kesmeler arasında sıralama yapılmasını sağlar. Daha düşük bir SubPriority değerine sahip olan kesme, aynı Preemption Priority seviyesinde daha yüksek önceliğe sahiptir. Yalnızca bir kesme zaten işlenirken birden fazla aynı öncelikteki kesme bekliyorsa kullanılır.
    - Preemption Priority **kritik öncelik sıralamasını** yönetirken, SubPriority **aynı seviyedeki** kesmeler arasında işlenme sırasını belirler.
- Bununla ile ilgili bir örnek senaryo yapalım
    - Diyelim ki bir sistemde üç kesme olsun.
        - Timer (TMR), kritik zamanlama gerektiriyor.
        - USART, veri iletişimi için kullanılıyor.
        - GPIO ise butona basıldığında bir işlem başlatıyor.
    - Şimdi de bunlar için önceliklerin ayarlanması işlemini yapalım.
        - Timer, en kritik olduğu için Preemption Priority: 0, SubPriority: 0
        - USART, orta kritiklikte, Preemption Priority: 1, SubPriority: 0
        - GPIO, en düşük öncelik, Preemption Priority: 2, SubPriority: 1
    - Daha sonrasında çalışma şekli aşağıdaki gibi olur.
        - Timer kesmesi çalışırken başka bir kesme geldiğinde, Timer kesmesi yalnızca daha düşük Preemption Priority değerine sahip kesmeler tarafından kesilebilir (örneğin, hiçbiri bu durumda)
        - USART kesmesi çalışırken Timer kesmesi gelirse, Timer kesmesi (daha yüksek önceliğe sahip olduğu için) USART kesmesini kesebilir.
        - GPIO kesmesi çalışırken USART veya Timer kesmeleri gelirse, GPIO kesmesi kesilir, daha yüksek öncelikli kesmeler işlenir.
- NVIC'nin Preemption Priority ve SubPriority için kaç bit ayıracağı, **Priority Grouping** ile belirlenir.

Priority Grouping	Preemption Priority Bits	SubPriority Bits
NVIC_PRIORITYGROUP_0	**0**	**4**
NVIC_PRIORITYGROUP_2	**2**	**2**
NVIC_PRIORITYGROUP_4	**4**	**0**

    - **Örnek olarak** NVIC_PRIORITYGROUP_2 seçildiğinde,
        - 2 bit **Preemption Priority**, 2 bit **SubPriority** kullanılabilmektedir.
        - Bu durumda, **Preemption Priority** değeri [0, 3] arasında olabilir, **SubPriority** ise [0, 3] arasında olabilir.
- STM32F407 mikrodenetleyicisi için porttaki 0.pin EXTI0 kanalına bağlıdır.





- Bunlar dışında 7 tane daha kanal vardır. Toplamda 23 kanal vardır.

EXTI line 16 is connected to the PVD output
EXTI line 17 is connected to the RTC Alarm event
EXTI line 18 is connected to the USB OTG FS Wakeup event
EXTI line 19 is connected to the Ethernet Wakeup event
EXTI line 20 is connected to the USB OTG HS (configured in FS) Wakeup event
EXTI line 21 is connected to the RTC Tamper and TimeStamp events
EXTI line 22 is connected to the RTC Wakeup event

Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_ MEMRMP	Reserved																														MEM_MODE	
	Reset value																															x	x
0x04	SYSCFG_PMC	Reserved								MII_RMII_SEL	Reserved							Reserved															
	Reset value									0																							
0x08	SYSCFG_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SYSCFG_CMPCR	Reserved																								READY	Reserved						CMP_PD
	Reset value																									0							0

- **SYSCFG_MEMRMP** (Memory Remap Register), mikrodenetleyicinin bellek haritalamasını yapılandırmak için kullanılır. Bellek haritalaması, sistemdeki farklı bellek alanları arasındaki bağlantıları yönetir. Örneğin, boot sektörünü değiştirmek veya haritalamayı farklı bir bellek bölgesine taşımak için kullanılabilir.
- **SYSCFG_PMC** (Peripheral Mode Configuration Register), çeşitli periferiklerin davranışlarını yapılandırmak için kullanılır. Özellikle çeşitli periferiklerin hangi güç modunda çalışacaklarını belirlemek için kullanılır.
- **SYSCFG_EXTICR** (External Interrupt Configuration Registers), harici kesmelerin hangi pinlere bağlı olduğunu yapılandırmak için kullanılır. Genellikle harici donanım kesmelerini bir GPIO pinine atanabilir ve bu registerlar aracılığıyla bu atamalar yapılır.
- **SYSCFG_CMPCR** (Compensation Cell Control Register), gerilim takibi ve düzeltme için kullanılır. Gerilim takibi, mikrodenetleyicinin çalışma gerilimini izleyerek enerji verimliliğini artırabilir.

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_IMR	Reserved									MR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EMR	Reserved									MR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	EXTI_RTSR	Reserved									TR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	EXTI_FTSR	Reserved									TR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	EXTI_SWIER	Reserved									SWIER[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	EXTI_PR	Reserved									PR[22:0]																						

	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**EXTI_PR**	Reserved	PR[22:0]																													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **EXTI_IMR** (Interrupt Mask Register), harici kesmelerin genel olarak etkinleştirilip etkinleştirilmeyeceğini kontrol eder. Her bit, belirli bir harici kesme hattını temsil eder ve bu bitlerin set olması, ilgili kesmenin etkinleştirildiği anlamına gelir.
- **EXTI_EMR** (Event Mask Register**)**, EXTI modülü, hem interrupt hem de event modlarında çalışabilir. Belirli bir harici kesme hattının olay modunda çalışıp çalışmayacağını kontrol eder. Yine, her bit belirli bir kesme hattını temsil eder.
- **EXTI_RTSR** (Rising Trigger Selection Register), bir harici kesmenin hangi kenardan rising edge tetikleneceğini belirler. Her bit, bir kesme hattını temsil eder ve bu bitlerin set olması, ilgili kesmenin yükselen kenardan tetikleneceği anlamına gelir.
- **EXTI_FTSR** (Falling Trigger Selection Register), bir harici kesmenin hangi kenardan falling edge tetikleneceğini belirler. Yine, her bit bir kesme hattını temsil eder ve bu bitlerin set olması, ilgili kesmenin düşen kenardan tetikleneceği anlamına gelir.
- **EXTI_SWIER** (Software Interrupt Event Register), yazılımsal olarak bir harici kesme talebi oluşturmak için kullanılır. Her bit, belirli bir harici kesmeyi temsil eder ve bu bitin set olması, ilgili kesme hattına bir yazılımsal talep gönderileceği anlamına gelir.
- **EXTI_PR** (Pending Register), hangi harici kesmelerin beklediğini gösterir. Her bit, belirli bir kesme hattını temsil eder ve bu bitlerin set olması, ilgili kesmenin beklediği anlamına gelir. Yazılım tarafından temizlenmelidir.

03 DMA

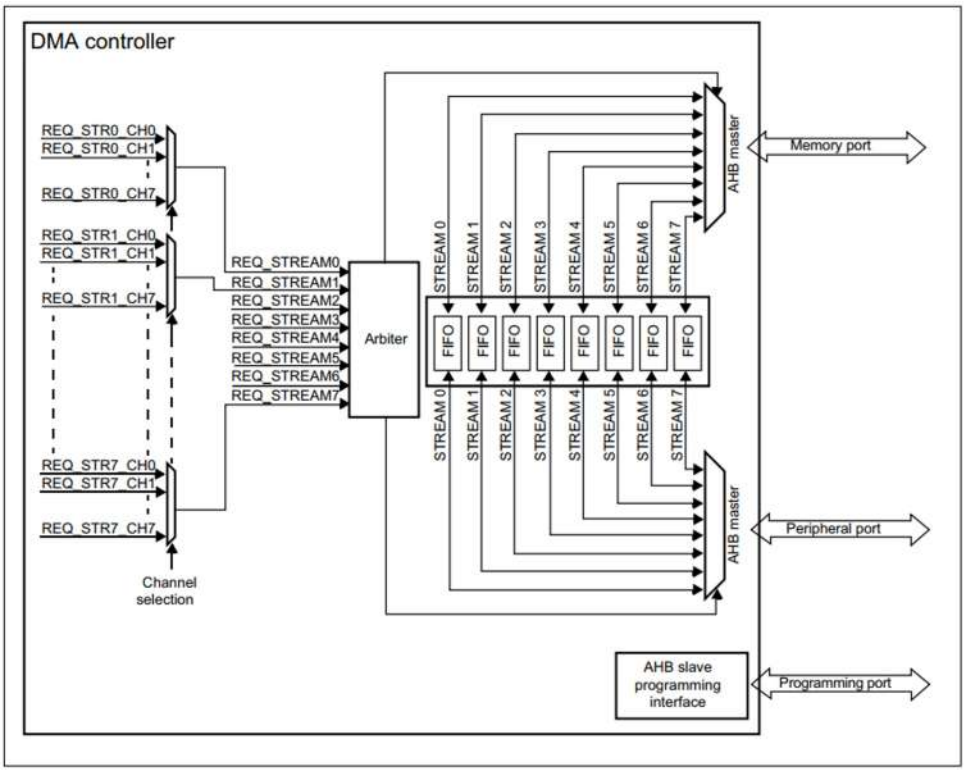
5 Mayıs 2021 Çarşamba 08:03

03 DMA

Giriş

- https://mikrodunya.wordpress.com/2016/06/23/dma-direct-memory-access-dogrudan-bellek-erisimi/
- **DMA** (Direct Memory Adres), veri transferlerini hiç bir CPU işlemini kullanmadan sağlaması amacıyla oluşturulmuştur.
- Çeşitli çevre birimlerinden okuduğumuz verileri bir değişkene atarız. Bu değişkenler RAM'de depolanır. Bu işlem normalde çevre birimlerinde okunan verinin CPU'ya alınıp ardından RAM'e yazılır. Ancak CPU kullanımı hem işlemciyi yorar hemde kayıplara yol açar. DMA sayesinde verileri direk olarak **RAM'e** yazma imkanı buluruz.
- DMA donanımı CPU'dan bağımsız olarak verilerimizi hızlı bir şekilde kaynak adresten hedef adrese aktarır.
    - **peripheral -> memory,** bir çevre biriminden (peripheral) gelen veri doğrudan belleğe (memory) aktarılır.
    - **mermory -> peripheral,** bellekten (memory) alınan veri doğrudan bir çevre birimine (peripheral) gönderilir.
    - **memory -> memory,** bir bellek bölgesindeki (memory) veri başka bir bellek bölgesine aktarılır.
- Bu sayede CPU'nun yükünü hafifletmiş oluruz. Sistem sanki 2 CPU ile çalışıyormuş gibi düşünebiliriz. Örneğin bilgisayarlarımızda bulunan 4 gerçek 4 sanal çekirdekteki sanal, aslında DMA diyebiliriz. DMA isteği için çevresel birim tarafından (ADC, DAC, I2C vs) DMA kontrolcüsüne istek gönderilir, kontrolcüde bu isteğin sırası gelince ilgili çevresel birime geri bildirimde bulunur ve işlem kaynak adresten hedef adrese doğru gerçekleşir.
- DMA işlemi, özellikle **çok fazla veri** alışverişi yapıldığı durumlarda kullanılmalıdır. CPU Yükünü Azaltma, Verimlilik, Hız, Kesintisizlik gibi nedenlerden dolayı sistem performansını önemli ölçüde artırır.
- STM32F4'te iki adet DMA vardır. DMA1'in DMA 2'den kanal 1'in kanal 2'den yüksek olduğu bilinmektedir. Öncelik sırası belirtmek için dört seviye vardır. Low, Medium, High, Very High.
  Aynı anda birçok kanal kullanıldığında hangi kanalın öncelik değeri fazla ise ilk o kanal alınır.
  DMA'lar paralel olarak çalışmazlar, seri olarak çalışırlar. Bu nedenle hangisinin sırası geldi ise o anda o çalışır.

Birim Yapısı



- **AHB Master Portları:** Şemanın sağ tarafında görülen AHB Master portları, DMA'nın hem bellek portuna

hem de peripheral portlara veri aktarımı yapabilmesini sağlar. Yüksek performanslı bir veri yolu olan AHB'yi kullanarak, verilerin hızlı bir şekilde taşınmasını sağlar.
    - **Programming Port**, DMA'nın programlanmasını ve yapılandırılmasını sağlar. Mikrodenetleyicinin çekirdeği (CPU), DMA'yı programlamak için bu portu kullanır. Bu port üzerinden DMA akışları, kanallar, öncelikler ve diğer parametreler ayarlanır.
    - **Memory Port**, DMA'nın belleğe erişmesini sağlar. Bellek portu üzerinden DMA, veriyi bellekten alabilir veya belleğe yazabilir.
    - **Peripheral Port**, DMA'nın peripheral cihazlarla veri alışverişini gerçekleştirir. Örneğin, bir UART cihazından veri almak veya bir ADC'den ölçülen veriyi bellek adresine aktarmak için bu port kullanılır.
- **Channel:** Şemanın sol tarafında, her bir kanal için (CH0-CH7) bir seçim süreci olduğunu görebiliriz. Bu kanallar, belirli bir donanım isteğine (örneğin, ADC, SPI, UART gibi) karşılık gelir. Her bir kanal belirli bir DMA akışına (Stream) atanabilir.
- **Streams:** Her bir DMA denetleyicisi 8 ayrı akışa sahiptir (Stream 0 - Stream 7). Bu akışlar, aynı anda birden fazla DMA işleminin yürütülmesine izin verir. Bir akış, belirli bir kaynaktan (örneğin, bir peripheral) veriyi alıp belirli bir hedefe (örneğin, bellek) iletebilir. Her akış, belirli bir kanal tarafından tetiklenir ve bu kanal üzerinden veri taşır.
  Her bir stream, 8 olası kanal isteği arasından seçilebilecek bir DMA isteği ile ilişkilendirilmiştir. Çevresel birimlerden (TIM, ADC, SPI, I2C, vb.) gelen 8 request, her bir kanala bağımsız olarak bağlanır ve bu bağlantı ürünün uygulanmasına bağlıdır. DMA istek eşlemeleri için aşağıdaki tabloları inceleyebilirsiniz.

DMA1 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX	-	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
Channel 1	I2C1_RX	-	TIM7_UP	-	TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Channel 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
Channel 5	UART8_TX(1)	UART7_TX(1)	TIM3_CH4 TIM3_UP	UART7_RX(1)	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX(1)	TIM3_CH3
Channel 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	-	TIM5_UP	-
Channel 7	-	TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

DMA2 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	ADC1	SAI1_A(1)	TIM8_CH1 TIM8_CH2 TIM8_CH3	SAI1_A(1)	ADC1	SAI1_B(1)	TIM1_CH1 TIM1_CH2 TIM1_CH3	-
Channel 1	-	DCMI	ADC2	ADC2	SAI1_B(1)	SPI6_TX(1)	SPI6_RX(1)	DCMI
Channel 2	ADC3	ADC3	-	SPI5_RX(1)	SPI5_TX(1)	CRYP_OUT	CRYP_IN	HASH_IN
Channel 3	SPI1_RX	-	SPI1_RX	SPI1_TX	-	SPI1_TX	-	-
Channel 4	SPI4_RX(1)	SPI4_TX(1)	USART1_RX	SDIO	-	USART1_RX	SDIO	USART1_TX
Channel 5	-	USART6_RX	USART6_RX	SPI4_RX(1)	SPI4_TX(1)	-	USART6_TX	USART6_TX
Channel 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
Channel 7	-	TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX(1)	SPI5_TX(1)	TIM8_CH4 TIM8_TRIG TIM8_COM

- **Arbiter:** DMA denetleyicisinde, aynı anda birden fazla akışın (Stream) çalışabileceğini söyledik. Ancak aynı anda birden fazla akışın aynı hedefe (örneğin, belleğe) erişmeye çalışması durumunda, bir çakışma olabilir. Bu çakışmayı yönetmek için bir arbiter bulunur. Arbiter, hangi akışın öncelikli olacağına ve hangi akışın belleğe veya peripheral porta erişim sağlayacağına karar verir.
- **FIFO:** Her akışın (Stream) bir FIFO tampon belleği (buffer) vardır. Bu FIFO tamponları, veri akışını geçici olarak depolamak için kullanılır. FIFO kullanımı, DMA'nın veriyi daha etkin bir şekilde yönetmesine yardımcı olur, özellikle de veri hızları arasında bir uyuşmazlık olduğunda (örneğin, veri kaynağı veriyi çok hızlı gönderiyorsa veya hedef veriyi daha yavaş alıyorsa)

Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**DMA_LISR**	Reserved				TCIF3	HTIF3	TEIF3	DMEIF3	Reserved	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Reserved	FEIF2	Reserved				TCIF1	HTIF1	TEIF1	DMEIF1	Reserved	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Reserved	FEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0004	**DMA_HISR**	Reserved				TCIF7	HTIF7	TEIF7	DMEIF7	Reserved	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserved	FEIF6	Reserved				TCIF5	HTIF5	TEIF5	DMEIF5	Reserved	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserved	FEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0008	**DMA_LIFCR**	Reserved				CTCIF3	CHTIF3	TEIF3	CDMEIF3	Reserved	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Reserved	CFEIF2	Reserved				CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Reserved	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Reserved	CFEIF0
	Reset value					0	0	0	0		0	0	0	0	0	Reserved	0					0	0	0	0	Reserved	0	0	0	0	0	Reserved	0
0x000C	**DMA_HIFCR**	Reserved				CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Reserved	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	-	CFEIF6	Reserved				CTCIF5	CHTIF5	CTEIF5	CDMEIF5	-	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	-	CFEIF4
	Reset value					0	0	0	0		0	0	0	0	0	-	0					0	0	0	0	-	0	0	0	0	0	-	0
0x0010	**DMA_S0CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	**DMA_S0NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	**DMA_S0PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	**DMA_S0M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	**DMA_S0M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	**DMA_S0FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1

- **DMA_LISR** ve **DMA_HISR** (Low/High Interrupt Status Register), DMA'nın düşük ve yüksek öncelikli kesmelerin durumunu izleyen register'lardır. Her bir bit, ilgili DMA kanalındaki bir kesmeyi temsil eder.
- **DMA_LIFCR** ve **DMA_HIFCR** (Low/High Interrupt Flag Clear Register), DMA'nın düşük ve yüksek öncelikli kesme bayraklarını temizlemek için kullanılır. Her bir bit, ilgili DMA kanalındaki bir kesme bayrağını temsil eder.
- **DMA_SxCR** (Stream x Configuration Register), DMA'nın belirli bir kanalının yapılandırma register'ıdır. Kanalın çalışma modu, transfer yönü, veri genişliği, bellek ve perifer adresi inkrement modu gibi özellikleri içerir.
- **DMA_SxNDTR** (Stream x Number of Data Register), İlgili DMA kanalında aktarılacak veri miktarını belirten register'dır.
- **DMA_SxPAR** (Stream x Peripheral Address Register), DMA'nın belirli bir kanalındaki perifer başlangıç adresini belirten register'dır.
- **DMA_SxMxAR ve DMA_SxMxAR** (Stream x Memory 0/1 Address Register), DMA'nın belirli bir kanalındaki bellek başlangıç adreslerini belirten register'lardır. Bazı STM32 modellerinde birden fazla bellek adresi kullanılabilir.
- **DMA_SxFCR** (Stream x FIFO Control Register), DMA FIFO (First In, First Out) kontrolünü sağlayan register'dır. FIFO'nun kullanılması, DMA transfer performansını artırabilir.

Süreç

- DMA kullanarak herhangi bir işlemci üzerinde veri transferi gerçekleştirmek için genel adım sıralaması aşağıdaki gibidir.
    - **Kullanılacak DMA Denetleyicisinin Belirlenmesi**
        - Uygulamanız için uygun olan DMAx denetleyicisini belirleyin. İşlemcinizin hangi DMA kanallarının kullanılabilir olduğunu öğrenin.

- **Yapılandırma Ayarların Yapılması**
  - Verilerin **nereden nereye** transfer edileceğini belirleyin.
  - Transfer edilecek **veri miktarını** (byte, half-word, word) belirleyin
  - Veri transferinin **yönünü** (kaynak->hedef) ve **modunu** (normal, döngüsel, vb.) ayarlayın.
  - Transfer tamamlandığında veya hata durumunda kesme kullanmak istiyorsanız, ilgili kesme yapılandırmalarını yapın ve kesme hizmet rutini (ISR) tanımlayın.
- **Transferin Tetiklenmesi**
  - Tüm yapılandırmalar tamamlandıktan sonra, DMA kanalını başlatın. Bu işlem, yazılım veya donanım tetiklemesi ile gerçekleştirilebilir.
- **Transferin İzlenmesi ve Tamamlanmasının Beklenmesi**
  - Transfer sırasında DMA durumunu izleyin ve transferin tamamlanmasını bekleyin. Eğer kesme kullanıyorsanız, kesme servis rutininde gerekli işlemleri yapın.
  - Transfer tamamlandıktan sonra DMA kanalını devre dışı bırakın ve gerekli temizleme işlemlerini yapın.

04 ADC

5 Mayıs 2021 Çarşamba 08:02

03 ADC

Giriş

- Doğada var olan bütün fiziksel büyüklükler (ısı, ışık, ses, zaman vs.) analog büyüklük kavramına girer.
- Dünyadaki herhangi bir şeyi dijital sistemlerimiz ile ölçmek, değerlendirmek, işlemek ve bu değerlere göre işlem yapabilmek için ADC (Analog Digital Converter) ihtiyaç vardır.
- ADC modülleri gerek harici, gerek dahili olsun hepsi bir referans voltaja ihtiyaç duyarlar. Genellikle mikroişlemcilerde referans voltajı işlemcinin besleme gerilimidir. Bu değer aynı zamanda ayarlar yapılarak harici olarak verilebilir.
- STM32'de 12-bit ADC, ardışık yaklaşım prensibine dayanan bir analog-dijital çeviricidir. Bu çevirici, 16 harici kaynaktan, iki dahili kaynaktan ve VBAT kanalından gelen sinyalleri ölçebilmek için en fazla 19 multiplexli kanala sahiptir. Kanalların A/D dönüşümü single, continuous, scan veya discontinuous modda gerçekleştirilebilir. ADC'nin sonucu, sola ya da sağa hizalanmış 16-bit veri kaydına depolanır.
- Analog watchdog özelliği, uygulamanın giriş voltajının kullanıcı tanımlı üst veya alt sınırları aşmasını algılamasına olanak tanır.

Çözünürlük

- ADC'ler 10, 12, 16, 24 vb. bit çözünürlükte bulunurlar.
- STM32F407'de ADC'ler 6, 8, 10 ve 12 bit çözünürlükte çalışabilirler ve referans voltajı default 3.3V'dur.
- ADC modülün 10 bit olduğunu düşünelim. 2^10 = 1024 değeri okunacak maksimum değerdir yani 0V = 0, 3.3V = 1023 değeri bize döner. Buradan her bit değerin alacağı voltaj değerini 3,3 / 1024 = 0,0032 olarak buluruz. Buradan da biz ADC modülünden okuduğumuz değeri bu ifade ile çarparsak voltaj değerini buluruz. 640 değeri için 640 * 0,0032 = 2,048 V olarak buluruz.
- STM32F407'de 0 - 3.6V aralığında ölçümler yapılabilmektedir. Buradaki voltaj aralığında ADC birimin beslemesi (VDDA-VSSA) ile ilgili bir durumdur.
- ADC birimin besleme voltajı (VDD) ve referans gerilimi (VREF), ADC birimin ölçebileceği gerilim aralığını belirler.
- Her ne olursa olsun ADC birimi 3.6V'dan fazlasını ölçemez.
- Analog bir değerden dijital bir değer dönüşüm yapılırken dikkat edilmesi gereken hususlar vardır. Bunlardan en önemlisi, ölçülecek analog gerilim değerinin dönüşümü yapacak çipin **ölçüm aralığında** olması gerekir. Diğer en önemli nokta, ölçüm yapılacak **hassasiyetin belirlenmesi** ve buna uygun bir **genişliğinde** bir dönüştürücü seçilmelidir.
- Ölçüm hassasiyetinde önemli olan dönüşüm yapacak sistemin bir **çözünürlüğüdür**.
- Resolution = VREF/(2^n-1)
- Örneğin 0 - 3.3V aralığı arası ölçüm yapabilen bir ADC ölçüm ünitesinin ölçebileceği minimum değer yaklaşık olarak formülden 8 bit çözünürlük için 12mV, 10 bit çözünürlük için 3,2mV, 12 bit çözünürlük için 805uV'tur.
- Çözünürlük arttıkça (bit sayısı arttıkça), ADC'nin ölçebileceği minimum voltaj değeri küçülür ve bu da daha hassas ölçümler yapabilmenizi sağlar.

Çevrim Süresi

- https://controllerstech.com/adc-conversion-time-frequency-calculation-in-stm32/ linkten ADC için çevrim süresinin nasıl hesaplandığı ile ilgili yazıyı okuyabiliriz.
- STM32F407'de ADC birimin ulaşabileceği maximum hız 36 MHz'dir. Bu hız aynı zamanda ADC çözünürlüğü ile ters orantılıdır. Çözünürlük **arttıkça** ADC birimin ölçüm hızı **düşmektedir**.
- Çevrim süresi hesabı için üç değere ihtiyaç var. Bunlar **Cycles**, **Sampling Time** ve **Clock**'tur.
- Cycles değeri seçilen Resolution değerine bağlıdır.
- Sampling Time ve Clock değerleri ise istediğimiz çevrim süresine göre değiştirebiliriz.
- ADC'de örnekleme süresi genellikle 3, 15, 28, 56, 84, 112, 144, 480 ADC saat çevrimi (cycles) olarak seçilebilir. Bu, sinyali yeterince doğru bir şekilde yakalamak için önemlidir.
- Clock değeri ADC'nin bağlı olduğu clock hattına bağlıdır.
- Tüm işlemcilerde aynı mantıktır fakat formül işlemciye göre farklılık gösterebilir bunun için kaynaklardan bakılması gerekir.

- Tconv = Sampling Time + Cycles / ADC Clock
  - **Tconv**: Dönüşüm süresi (conversion time)
  - **Sampling Time**: ADC'nin sinyali örnekleme süresi (örnekleme çevrimleri ile belirtilir)
  - **Cycles**: ADC çözünürlüğüne bağlı olarak kullanılan ek çevrimler. 12-bit ADC için bu değer genellikle 12 çevrimdir.
  - **ADC Clock**: ADC'nin saat frekansı
- Eğer hızlı bir sinyali ölçmeniz gerekiyorsa, yüksek bir dönüşüm frekansı seçmelisiniz. Örneğin, ses sinyalleri veya hızlı değişen analog sinyaller için dönüşüm frekansı yüksek olmalıdır.
  Ancak çok yüksek frekanslar, özellikle gürültülü ortamlarda sinyal doğruluğunu azaltabilir, bu nedenle filtreleme ve örnekleme süresi dikkatle seçilmelidir.
- Daha yavaş sinyaller için (örneğin sıcaklık ölçümü gibi) düşük bir dönüşüm frekansı yeterli olabilir. Daha uzun örnekleme süresi kullanarak sinyali daha doğru bir şekilde ölçebilirsiniz.
  Ayrıca düşük dönüşüm frekansı güç tüketimini azaltabilir.
  Yüksek Çözünürlük, 12-bit çözünürlük gibi daha hassas ölçümler sağlar, ancak dönüşüm süresini artırır.
- Örnekleme süresi ne kadar düşükse, dönüşüm o kadar hızlı olur, ancak ölçüm hassasiyeti ve doğruluğu azalabilir.
- Genel olarak belirli bir sinyal frekansını doğru şekilde örneklemek için **Nyquist kriterine** uymak gerekir. Bu kritere göre, örnekleme frekansı, ölçülen sinyalin en yüksek frekans bileşeninin en az iki katı olmalıdır.
  - **Yavaş değişen sinyaller** (örneğin sıcaklık sensörleri, basınç sensörleri, potansiyometreler), dönüşüm frekansı birkaç kHz olabilir (1-10 kHz aralığı genellikle yeterlidir).
  - **Orta hızlı sinyaller** (örneğin ses sinyalleri, motor hız sensörleri), dönüşüm frekansı genellikle 10 kHz - 100 kHz aralığında olmalıdır.
  - **Hızlı sinyaller** (örneğin RF sinyalleri, yüksek frekanslı analog sinyaller), dönüşüm frekansı 100 kHz ile MHz mertebesinde olabilir.

  Ancak çoğu uygulama için 10 kHz - 1 MHz aralığı yeterlidir.

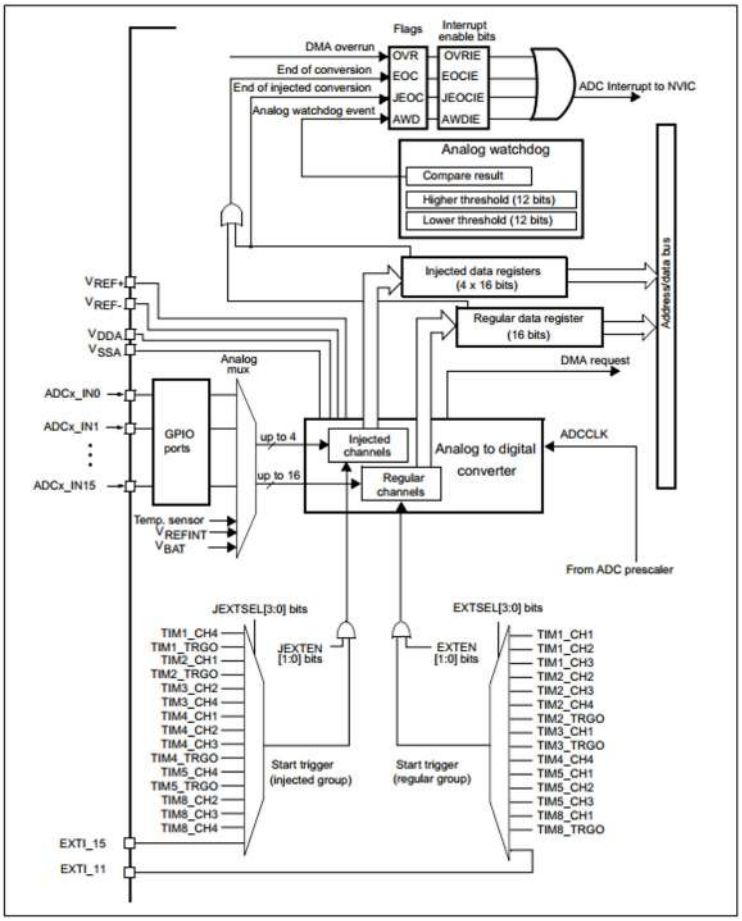
Çalışma Modları

- **Single Conversion Mode**, bir **tek** dönüşüm gerçekleştirildikten sonra ADC'nin otomatik olarak durmasını sağlar. Her dönüşüm, başlatma komutu ile başlatılır ve tamamlandığında ADC otomatik olarak durur.
- **Continuous Conversion Mode**, başlatıldığı andan itibaren **sürekli** olarak dönüşümler gerçekleştirir. Otomatik durma olmadığı için dönüşümler devam eder, kullanıcı tarafından durdurulana kadar devam eder.
- **Scan Mode,** belirli bir kanal listesini otomatik olarak **tarama** yeteneğine sahiptir. Tarama modu, birden fazla kanalı tek bir dönüşüm başlatma komutu ile sırayla ölçmeyi sağlar.
- **Discontinuous Mode**, kullanıcı belirli bir kanal listesinin **ardışık** olarak ölçülmesini sağlayabilir. Ancak, kanal arasında belirli bir gecikme bulunabilir.

Ölçüm Yöntemleri

- ADC ölçümlerini almak için kullanılan farklı yöntemler şunlardır: Polling, Interrupt ve DMA
- http://www.elektrobot.net/stm32-adc-kullanimi-polling-interrupt-ve-dma/ ile https://controllerstech.com/stm32-adc-single-channel/ linkten Polling, Interrup ve DMA metodu kullanarak yapılan örnekleri inceleyebiliriz.
- **Polling** yöntemi, mikrodenetleyici ADC'nin çevrim süresince farklı bir işlem yapmaz ve çevrimin bitmesini bekler. Yapılacak ölçümün çok hızlı olmasının gerekmediği yada uzun zaman aralıklarında tek ölçüm yapılmasının yeterli olduğu durumlarda sıklıkla kullanılır.
- **Interrupt** yöntemi, ADC dönüşümü tamamlandığında bir kesme çağrısı gerçekleşir. Böylece mikrodenetleyicinin başka işlerle meşgulken dahi ADC verilerini işlemesine izin verir.
  Daha karmaşık uygulamalarda, dönüşüm tamamlandığında hemen yanıt verilmesi gereken durumlar için uygundur. Verimli kullanım, mikrodenetleyicinin diğer görevlere odaklanmasını sağlar.
- **DMA** yöntemi, ADC sonuçları doğrudan belleğe kopyalanır, bu da CPU'nun dahil olmadan çalışmasına olanak tanır. Büyük veri setlerini hızlı bir şekilde işlemek ve mikrodenetleyicinin CPU'sunu diğer görevlere odaklamak için uygundur. Bellek yönetimi konusunda dikkatlice ele alınması gerekebilir.
- DMA'nın Interrupt ile kullanımından en büyük farkı, ADC'nin çevrimi tamamladıktan sonra elde ettiği değeri hafıza bölgesine DMA tarafından yazılmasıdır. Böylece mikrodenetleyici hiç bir şekilde ADC işlemleri ile meşgul olmaz. Özellikle çok sayıda ölçümün ard arda ve hızlı yapılmasının istendiği durumlarda DMA kullanılır.

Birim Yapısı



Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_SR**	Reserved																										OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value																											0	0	0	0	0	0
0x04	**ADC_CR1**	Reserved					OVRIE	RES[1:0]		AWDEN	JAWDEN	Reserved						DISC NUM [2:0]			JDISCEN	DISCEN	JAUTO	AWD SGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]					
	Reset value						0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**ADC_CR2**	Reserved	SWSTART	EXTEN[1:0]			EXTSEL [3:0]			Reserved	JSWSTART	JEXTEN[1:0]		JEXTSEL [3:0]				Reserved				ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON
	Reset value		0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0		0							0	0
0x0C	**ADC_SMPR1**	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**ADC_SMPR2**	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**ADC_JOFR1**	Reserved																				JOFFSET1[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**ADC_JOFR2**	Reserved																				JOFFSET2[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**ADC_JOFR3**	Reserved																				JOFFSET3[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x20	**ADC_JOFR4**	Reserved																				JOFFSET4[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_HTR**	Reserved																				HT[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x28	**ADC_LTR**	Reserved																				LT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**ADC_SQR1**	Reserved								L[3:0]				Regular channel sequence SQx_x bits																			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**ADC_SQR2**	Reserved	Reserved	Regular channel sequence SQx_x bits																													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**ADC_SQR3**	Reserved	Regular channel sequence SQx_x bits																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**ADC_JSQR**	Reserved										JL[1:0]		Injected channel sequence JSQx_x bits																			
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**ADC_JDR1**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**ADC_JDR2**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**ADC_JDR3**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**ADC_JDR4**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**ADC_DR**	Reserved																Regular DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **ADC_SR** (Status Register), ADC durumunu izleyen bu register, dönüşüm tamamlandığında, taşma veya analog bekçi olaylarının gerçekleştiğini belirten bayrakları içerir.
- **ADC_CR1** (Control Register 1), dönüşüm kesmelerini etkinleştirme, scan modunu kontrol etme, discontinuous modu ve enjekte dönüşümleri yönetme gibi temel ADC kontrol ayarlarını içerir.
- **ADC_CR2** (Control Register 2), ADC'nin genel kontrolünü sağlayan bu register, ADC'nin etkinleştirilmesi, continuous conversion modu, DMA modu, kalibrasyon ve harici tetikleme seçenekleri gibi ayarları içerir.
- **ADC_SMPR1** ve **ADC_SMPR2** (Sampling Time Register 1 ve 2), örnekleme süresini belirleyen bu registerlar, her bir kanalın örnekleme süresini ayarlamanızı sağlar.
- **ADC_DR** (Data Register), Dönüşüm sonuçlarını depolar; yani ADC tarafından ölçülen analog sinyalin dijital karşılığını içerir.

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_CSR	Reserved										OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value											0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
												ADC3								ADC2								ADC1					
0x04	ADC_CCR	Reserved								TSVREFE	VBATE	Reserved				ADCPRE[1:0]		DMA[1:0]		DDS	Reserved	DELAY [3:0]				Reserved		MULTI [4:0]					
	Reset value									0	0					0	0	0	0	0		0	0	0	0			0	0	0	0	0	
0x08	ADC_CDR	Regular DATA2[15:0]																Regular DATA1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- ADC'deki **common registerlar**, birden fazla ADC modülünün ortak kullanıldığı durumlar için genel ayarları ve durumu izlemek için tasarlanmış registerlardır. Bu registerlar, birden fazla ADC'nin ortak özelliklerini kontrol etmek ve izlemek için kullanılır.
- **ADC_CSR** (Common Status Register), modülünün genel durumunu gösterir. Özellikle birden fazla ADC'nin kullanıldığı durumlarda ortak durumu izlemek için kullanılır.
- **ADC_CCR** (Common Control Register), ortak ayarları içerir. Örneğin, referans voltajlarını (VREF+ ve VREF-) belirlemek gibi genel ADC kontrol parametrelerini içerir.
- **ADC_CDR** (Common Data Register), birden fazla ADC kullanıldığında, çeşitli ADC'lerden gelen verileri depolar.

05 DAC

5 Mayıs 2021 Çarşamba 08:02

04 DAC

Giriş

- DAC, **"Digital-to-Analog Converter"** dijital sinyalleri analog sinyallere dönüştürmek için kullanılır. Genellikle mikrodenetleyiciler, bilgisayarlar, ses sistemleri ve diğer dijital cihazlar gibi dijital veri kaynaklarından gelen dijital verileri, analog çıkış cihazlarına (örneğin hoparlörler veya ses sistemleri) uygun bir şekilde aktarmak için kullanılırlar.
- STM32F407, 0-3.3 V arasında tüm gerilimleri çıkış olarak vermemizi sağlar.
- STM32F407 mikrodenetleyicisi içerisinde dahili olarak 12 bit tampona sahip, iki adet DAC birimi bulunur. Bu birimler sayesinde dijital bir veriyi analog bir veriye dönüştürerek çıkış üretilebilir.
- STM32F407'ye ait DAC birimleri 8 bit veya 12 bit değerinde çıkış üretilebilirler.
- 12 bit değerinde kullanılırken, veri 16 bitlik kaydedici içerisinde sola veya sağa dayalı şekilde kullanılabilir.
- DAC biriminin önemli özelliklerinden bir tanesi, gürültü veya sinyali üretebilme özelliğidir.
- Üçgen dalga üretebilme özelliğine sahiptir.
- DAC birimleri APB1 veri yoluna bağlıdır, kullanmak için aktif etmek gereklidir.
- DAC için hangi pin/pinler kullanılacaksa ilgili pin/pinler GPIOA->CRL registerından analog moda alınmalıdır.

Çözünürlük

- STM32'de DAC çözünürlüğünü arttırmak için Vref+ girişi bulunmaktadır fakat bu pin yüksek işlemcilerde bulunmaktadır. Vref+ ve Vref- pini bulunmayan işlemcilerde bu pinler dahili olarak **VDDA** ve **VSSA**' ya bağlıdır. VDDA ve VSSA ise VDD ile VSS'ye bağlanması zorunludur. Buradanda Vref+ geriliminin besleme gerilimini **geçemeyeceğini** anlıyoruz.

  $$DACoutput = Vref \times DOR / 4096$$

- Yukarıdaki ifade ile DAC çıkış voltajı hesaplanır. Biz DAC değerlerimizi DHR registerına yazarız ve tetikleme sonucunda DHR'deki veri DOR registerına aktarılır, DOR registerını sadece okuyabiliriz.
- 12 bitlik çözünürlüğe sahip bir DAC biriminin referans gerilimleri Vssa = 0 V, Vdda = +3 V ele alınır ise, adım başına üreteceği voltaj şu şekilde hesaplanır; DACoutput = Vref/4095
  Buradan adım başına düşen voltaj, DACoutput=3V/4095 = 732,600732
  Örneğin 1V elde etmek isteniyorsa:, 1/0,000732600 = 1365 değeri elde edilir.

Çalışma Modları

- STM32 mikrodenetleyicilerinde DAC modülü genellikle tek kanal, çift kanal, üçgen dalga ve gürültü oluşturma modları gibi farklı çalışma modlarına sahiptir.
- **Tek Kanal Modu**, Tek bir DAC kanalı üzerinden analog çıkış sağlar. Örneğin, STM32 mikrodenetleyicilerinde "DAC_Channel_1" kullanarak tek kanal modunda DAC'ı kullanabilirsiniz.
- **Çift Kanal Modu**, iki DAC kanalı üzerinden bağımsız olarak analog çıkış sağlar.
  Örneğin, STM32 mikrodenetleyicilerinde "DAC_Channel_1" ve "DAC_Channel_2" kullanarak çift kanal modunda DAC'ı kullanabilirsiniz.
- **Üçgen Dalga Modu**, DAC, üçgen dalga formunu üretebilir. Bu modda, DAC çıkışı belirli bir frekansta bir üçgen dalga formunu takip eder.
- **Gürültü Oluşturma Modu**, DAC, belirli bir frekansta gürültü sinyali üretebilir. Bu modda, DAC çıkışı belirli bir frekansta rasgele değerler üreterek bir gürültü sinyali oluşturur.

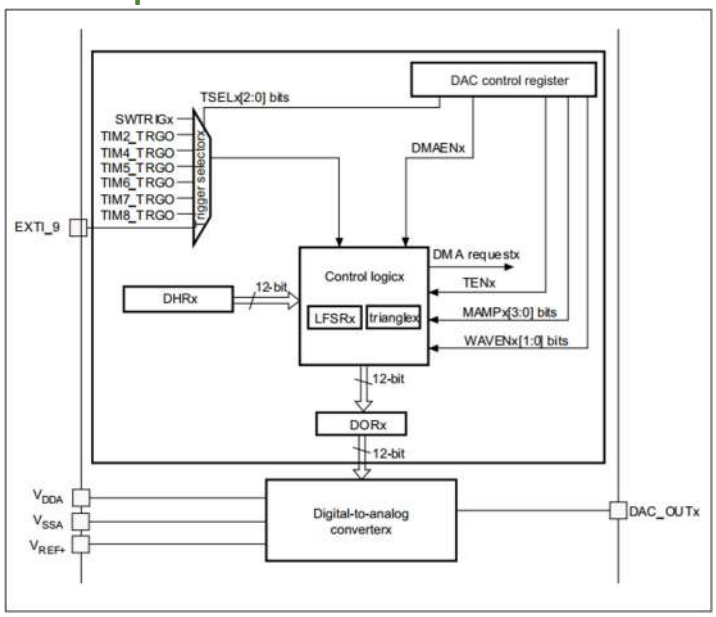
Tetikleme İşlemleri

- Genellikle yazılımsal ve harici tetikleme (triggering) yöntemleri ile kullanılabilir. Bu yöntemler, DAC'nin çıkışını kontrol etmek ve çıkış verisini belirli bir zamanlama veya olaya bağlamak için kullanılır.
- **Software Triggering**, Yazılımsal tetikleme, mikrodenetleyici yazılımı tarafından kontrol edilen bir tetikleme yöntemidir. Yazılım, DAC çıkışını başlatmak veya durdurmak için özel bir komut kullanır. Bu yöntem, zamanlama ile ilgili hassas kontrol gerektiren durumlarda kullanışlıdır. Örneğin, bir zamanlayıcı kesmesi veya belirli bir durum gerçekleştiğinde DAC çıkışını güncellemek için yazılımsal tetikleme kullanılabilir.
- **External Triggering**, DAC modülünü dış bir olaya (örneğin, bir zamanlayıcı kesmesi, bir GPIO değişikliği veya başka bir harici sinyal) bağlamak anlamına gelir. Harici bir sinyal algılandığında veya belirli bir durum gerçekleştiğinde, DAC çıkışını güncellemek için harici bir sinyal kullanılabilir.

Farklılıkları

- DAC ve PWM, her ikisi de dijital sinyalleri analog sinyallere dönüştürmek için kullanılan yöntemlerdir, ancak farklı çalışma prensiplerine sahiptirler.
- DAC, doğrudan dijital değerleri analog voltaj veya akıma dönüştürürken, PWM, darbe genişliği modülasyonu yoluyla bir analog etki oluşturur.
- DAC, genellikle doğrudan analog çıkış sağlar ve daha hassas bir çözünürlük sunabilir. PWM ise daha çok göreceli ve yaklaşık bir çözünürlük sağlar.
- DAC, genellikle özel bir entegre devre içerirken, PWM, genellikle bir mikrodenetleyici tarafından kontrol edilir.
- DAC, yüksek hassasiyet gerektiren ses uygulamalarında daha tercih edilebilirken, PWM, motor hız kontrolü gibi uygulamalarda daha uygun olabilir.

Birim Yapısı



Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE 2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE 1[2:0]		TSEL1[2 :0]			TEN1	BOFF1	EN1
0x04	DAC_ SWTRIGR	Reserved																														SWTRIG2	SWTRIG1
0x08	DAC_ DHR12R1	Reserved																				DACC1DHR[11:0]											
0x0C	DAC_ DHR12L1	Reserved																DACC1DHR[11:0]												Reserved			
0x10	DAC_ DHR8R1	Reserved																								DACC1DHR[7:0]							
0x14	DAC_ DHR12R2	Reserved																				DACC2DHR[11:0]											
0x18	DAC_ DHR12L2	Reserved																DACC2DHR[11:0]												Reserved			
0x1C	DAC_ DHR8R2	Reserved																								DACC2DHR[7:0]							
0x20	DAC_ DHR12RD	Reserved				DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]											
0x24	DAC_ DHR12LD	DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]												Reserved			
0x28	DAC_ DHR8RD	Reserved																				DACC2DHR[7:0]				DACC1DHR[7:0]							
0x2C	DAC_ DOR1	Reserved																				DACC1DOR[11:0]											
0x30	DAC_ DOR2	Reserved																				DACC2DOR[11:0]											
0x34	DAC_SR	Reserved		DMAUDR2		Reserved														DMAUDR1		Reserved											

- **DAC_CR** (Control Register), DAC'nin genel kontrolünü sağlayan bu register, örneğin çıkış voltaj seviyesi, çıkış güçlendirme ve trigger seçeneklerini içerir.
- **DAC_SWTRIGR** (Software Trigger Register), yazılım tetikleme işlemlerini kontrol etmek için kullanılır.
- **DAC_DHR** (Data Holding Register), bu register'lar, DAC'ye gönderilecek dijital veriyi içerir.
- **DAC_SR** (Status Register), DAC durumunu izlemek için kullanılır.
- **DAC_DOR** (Data Output Register), DAC'nin çıkışından okunan gerçek zamanlı dijital çıkış verisini temsil eder. Dönüştürülen analog sinyalin temsil ettiği dijital değeri içerir.

06 TIMER

5 Mayıs 2021 Çarşamba 08:02

06 TIMER

Giriş

- Timer modülünün temel görevi zamanlama yapmaktır. İşlemci frekanasına bağlı olarak çalışırlar. Dışarıdan gelen pulse darbelerini sayarlar. İşlemciye tanıtılan bir süre ile, geçen süreyi karşılaştırma ve belli bir süre sonunda kesme üretme gibi işlemlerde kullanılırlar.
- Sayıcı birimi sabit bir frekans kaynağı ile besleniyorsa Timer olarak çalışır. Zamanlayıcının bir adımı 1/f süresine denk gelir. Örneğin 1 kHz ile beslenen bir zamanlayıcının her adımı 1 ms demektir.
- 1kHz ile beslenen zamanlayıcıyı t1 anında okuduğumuzda değeri 100, t2 anında okuduğumuzda değeri 250 ise, t2-t1 arasında geçen süre 150ms demektir. Zamanlayıcılar ile bu şekilde zaman ölçümü ya da periyodik işlemlerin gerçekleştirilmesini sağlarlar.
- Timer, belirli bir süre veya sayım gerçekleştirdikten sonra, sayaç değeri belirli bir sınırı aşarsa veya taşarsa, overflow durumu ortaya çıkar. Bu zamanlayıcı bir belirli sayıya kadar sayıyorsa sayaç bu sayıya ulaştığında, taşma **overflow** gerçekleşir ve sayaç sıfırlanarak yeniden başlar.

Yapılandırma

- STM32 mikrodenetleyicilerinde Timer ayarlarken, **prescaler** ve **period** değerleri zamanlayıcının üreteceği frekansı veya zamanı belirler. Bu değerler genellikle belirli bir zaman aralığı, frekans veya PWM sinyali elde etmek için seçilir.
- Prescaler, zamanlayıcıya giren **saat sinyalini bölmek** için kullanılır. Örneğin, eğer bir zamanlayıcı 72 MHz'lik bir saat sinyaliyle çalışıyorsa ve prescaler değeri 71 olarak ayarlandıysa, zamanlayıcıya gelen sinyalin frekansı 1 MHz olur (72 MHz / (71 + 1))
  - Prescaler değeri şu formülle hesaplanır:

    $f_{timer} = f_{clock} / (Prescaler + 1)$
    - **f_clock,** Timer'ın besleme frekansı (genellikle sistem saat frekansı)
    - **Prescaler,** Timer'a uygulanan saat sinyalini bölen değer (0'dan başlar, dolayısıyla +1 yapılır)
- Period değeri, timer'ın kaç clock döngüsünde bir sıfırlanacağını belirler
  - Period değeri ile timer'ın toplam zaman aralığı şu şekilde hesaplanır:

    $T_{period} = (Period + 1) \times 1 / f_{timer}$
    - **T_period:** Timer'ın toplam zaman aralığı (örneğin bir PWM sinyali için bir periyod)
    - **Period:** Auto-Reload Register (ARR) değeri (0'dan başlar, bu yüzden +1 yapılır)
    - **f_timer:** Prescaler ile ayarlanmış timer frekansı
- Eğer 1 kHz'lik bir sinyal üretmek istiyorsanız ve sistem saat frekansınız 72 MHz ise:
  - Önce uygun bir prescaler değeri seçin:

    Eğer prescaler = 71999 seçilirse, f_{timer}= 72MHz / 72000 = 1kHz
  - Period değeri olarak ise:

    Period = 999 seçilirse, toplam zaman T_{period} = 1000 × 1 / 1kHz = 1s
  - 1 ms zaman tabanına sahip bir zamanlayıcı elde ediyorsunuz ve böylece 1000 kere 1 ms aralıklarla sayma yapacak sonucunda 1s sonra taşma olacak.
- Eğer timer ile 100ms'lik bir gecikme yaratmak istiyorsanız:
  - Eğer prescaler = 7199 seçilirse, f_{timer} = 72MHz / 7200 = 10kHz
  - Period = 999 olarak seçilirse, T_{period} = 1000 × 1 / 10kHz = 0,1s = 100ms
  - 0,1 ms zaman tabanına sahip bir zamanlayıcı elde ediyorsunuz ve böylece 1000 kere 0,1 ms aralıklarla sayma yapacak sonucunda 100ms sonra taşma olacak.

$$UpdateEvent = \frac{Timer_{clock}}{(Prescaler + 1)(Period + 1)}$$

- Yukarıdaki görseldeki formül, prescaler ve period değerlerini bir araya getirerek timer'ın ne sıklıkla bir güncelleme olayı üreteceğini doğrudan hesaplayan bir yöntem sunar. Daha önce bahsedilen iki aşamalı süreç ise, önce **timer clock frekansını**, ardından **çıkış sinyal frekansını** belirleyen bir yaklaşımı takip eder. Her iki yöntem de timer yapılandırmasında kullanılır; ancak görseldeki formül, daha doğrudan ve kompakt

bir yaklaşım sunar.

İşlevler

- **Capture**, zamanlayıcının mevcut değerini özel bir kaydediciye kopyalama işlemidir. Bu, bir dış olayın gerçekleştiği belirli bir zamanı yakalamak için kullanılabilir. Örneğin, dışardan gelen sinyalin belirli bir durumu algılandığında, zamanlayıcı değeri bu anda "yakalanır" ve kaydedilir. Bu, belirli olayların zaman damgalarını elde etmek için sıklıkla kullanılır.
- **Compare**, zamanlayıcı değerini bir belirli değerle karşılaştırma işlemini ifade eder. Zamanlayıcı, belirli bir değere ulaştığında veya onu geçtiğinde, bu bir olayın tetiklenmesine neden olabilir. Örneğin, belirli bir zaman geçtikten sonra bir işlemi başlatmak için compare özelliği kullanılabilir. Bu, periyodik işlemleri kontrol etmek veya belirli bir süreyi takip etmek için yaygın olarak kullanılır.
- **Pulse Width Modulation (PWM)**, genellikle bir dijital sinyalin darbe genişliğini modüle etme tekniğini ifade eder. PWM, bir sinyalin belirli bir süre boyunca HIGH ve belirli bir süre boyunca LOW olduğu bir sinyal üretir. Bu modülasyon tekniği, analog sinyal davranışını taklit etmek veya kontrol etmek için yaygın olarak kullanılır. Çoğu mikrodenetleyicide PWM birimleri de Timer ünitelerine bağlı olarak çalışırlar.

Birimler

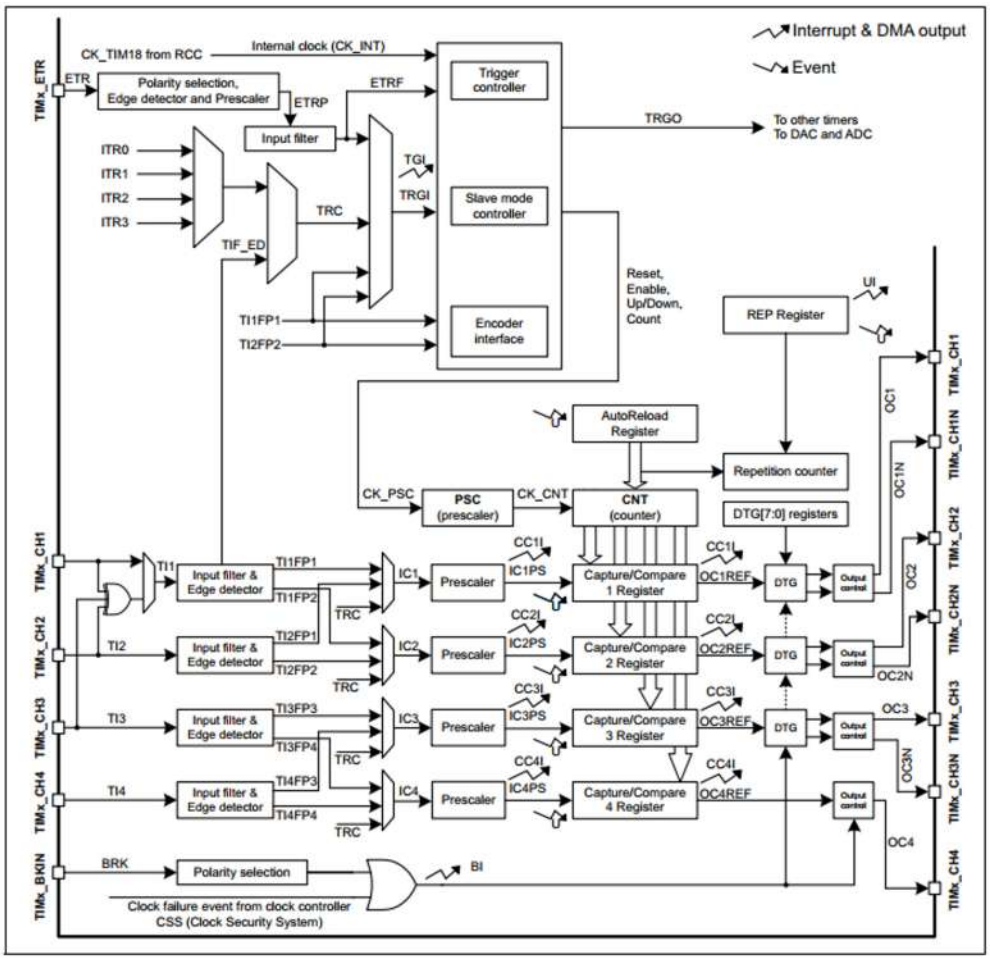
- STM32F407VG işlemcisinde toplam 17 adet timer birimi bulunur.
  10 adet **General Purpose**, 2 adet **Advanced Control**, 2 adet **Basic**, 1 adet **Independent Watchdog (IWDG)**, 1 adet **Window Watchdog (WWDG)** timer, 1 adet **Systemtick** timer var.

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complemen- tary output	Max interface clock (MHz)	Max timer clock (MHz)
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	84	168
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	84	168
	TIM10 , TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	84	168
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	42	84
	TIM13 , TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	42	84
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	42	84

Advanced Control

TIM1, TIM8

- TIM1 ve TIM8, yüksek hızlı APB2 veri yolu (84 MHz) üzerinde bulunurlar. Eğer APB2 prescaler değişkeni 1 değerinden farklı ise bu timer birimlerinin saat frekansı, APB2'nin frekans değerinin iki katı olur. Yani, bu timer birimlerinin maksimum çalışma frekansları 168 MHz olabilir.
- TIM1 ve TIM 8 birimleri 16 bitlik sayıcıya sahiptirler.
- Bu sayıcılar; yukarı, aşağı ve merkezlenmiş modlarda sayma yapabilirler.
- Bu sayıcıların otomatik geri yükleme özellikleri bulunmaktadır.
- Bu timer birimlerinde 4x16 adet yüksek çözünürlüklü capture/compare kanalı da bulunur.
  Bu kanallar giriş çıkış olarak ayarlanabilir, çıkış karşılaştırabilir, PWM sinyali üretebilir, sinyal yakalayabilir ve harici bir PWM sinyalini algılayabilirler.


Interrupt & DMA output
Event
TIMx_ETR
ETR
CK_TIM18 from RCC
Internal clock (CK_INT)
Polarity selection, Edge detector and Prescaler
ETRP
ETRF
Input filter
Trigger controller
TRGO
To other timers
To DAC and ADC
ITR0
ITR1
ITR2
ITR3
TRC
TIF_ED
TGI
TRGI
Slave mode controller
Reset, Enable, Up/Down, Count
Encoder interface
TI1FP1
TI2FP2
UI
REP Register
AutoReload Register
Repetition counter
CK_PSC
PSC (prescaler)
CK_CNT
CNT (counter)
DTG[7:0] registers
TIMx_CH1
TI1
Input filter & Edge detector
TI1FP1
TI1FP2
IC1
Prescaler
CC1I
IC1PS
Capture/Compare 1 Register
OC1REF
DTG
Output control
OC1
OC1N
TIMx_CH1N
TIMx_CH2
TI2
TI2FP1
TI2FP2
IC2
CC2I
IC2PS
Capture/Compare 2 Register
OC2REF
OC2
OC2N
TIMx_CH2N
TIMx_CH3
TI3
TI3FP3
TI3FP4
IC3
CC3I
IC3PS
Capture/Compare 3 Register
OC3REF
OC3
OC3N
TIMx_CH3N
TIMx_CH4
TI4
TI4FP3
TI4FP4
IC4
CC4I
IC4PS
Capture/Compare 4 Register
OC4REF
OC4
TIMx_CH4
TIMx_BKIN
BRK
Polarity selection
BI
Clock failure event from clock controller
CSS (Clock Security System)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Reserved																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIMx_DIER	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0		0	0	0	0	0	0	0	0
0x14	TIMx_EGR	Reserved																								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																									0	0	0	0	0	0	0	0
0x18	TIMx_CCMR1 Output compare mode	Reserved																	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Reserved																	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2 Output compare mode	Reserved																	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input capture mode	Reserved																	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Reserved																	CC4NP	Reserved	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	TIMx_CNT	Reserved																	CNT[15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																	PSC[15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																	ARR[15:0]															
	Reset value																		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Reserved																								REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Reserved																	CCR1[15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Reserved																	CCR2[15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	Reserved																	CCR3[15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	Reserved																	CCR4[15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIMx_BDTR	Reserved																	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Reserved																				DBL[4:0]					Reserved		DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **TIMx_CR1 (Control Register 1)**, Timer'ın genel kontrol ayarlarını içerir. Timer'ı etkinleştirme, zamanlama

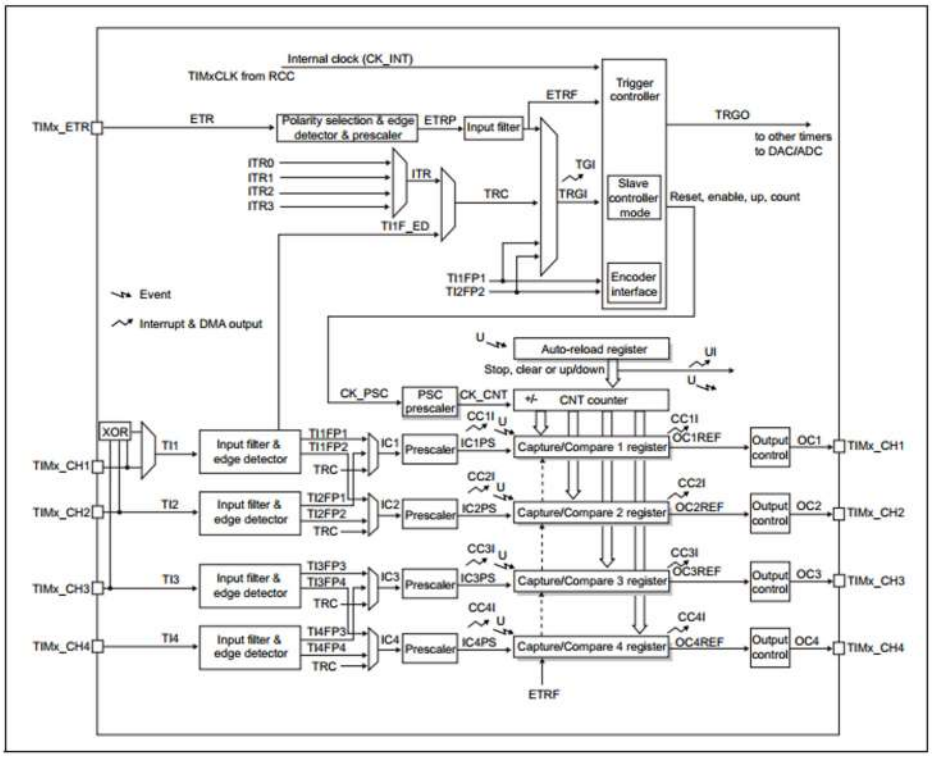
modu seçimi, otomatik yeniden başlatma etkinleştirme gibi ayarları içerir.

- **TIMx_CR2 (Control Register 2)**, Timer'ın özel kontrol ayarlarını içerir. Bu register, master mode seçimi gibi özellikleri kontrol eder.
- **TIMx_SMCR (Slave Mode Control Register)**, Timer'ın slave modunu kontrol eder. Dış bir kaynaktan senkronize olma veya bir başka timer'ı takip etme gibi işlevleri içerir.
- **TIMx_DIER (DMA/Interrupt Enable Register)**, DMA ve kesme interrupt izinlerini kontrol eder. Belirli olayların tetiklenmesi durumunda bir kesme talebi veya DMA transferi başlatma gibi işlevleri etkinleştirir veya devre dışı bırakır.
- **TIMx_SR (Status Register)**, Timer'ın durumuyla ilgili bilgileri içerir. Taşma, karşılaştırma olayları gibi çeşitli olayları takip eder.
- **TIMx_EGR (Event Generation Register)**, Olayların elle tetiklenmesini sağlar. Bu register üzerinden bir olayı (event) hemen tetikleyebilirsiniz.
- **TIMx_CCMR1 ve TIMx_CCMR2 (Capture/Compare Mode Register 1 ve 2)**, Capture/compare modu için ayarları içerir. Timer'ın çeşitli modlarını, giriş ve çıkış ayarlarını belirler.
- **TIMx_CCER (Capture/Compare Enable Register)**, Capture/compare kanallarını etkinleştirme veya devre dışı bırakma işlemlerini kontrol eder.
- **TIMx_CNT (Counter Register)**, Timer'ın ana sayaç değerini içerir. Bu register, zamanlayıcının sayma işlemini temsil eder.
- **TIMx_PSC (Prescaler Register)**, Timer'ın ön bölücü prescaler değerini içerir. Bu değer, timer'ın sayma hızını kontrol eder.
- **TIMx_ARR (Auto-Reload Register)**, Timer'ın otomatik yeniden başlatma değerini içerir. Bu değer, sayacın bir döngü tamamladığında otomatik olarak tekrar başlamasını sağlar.
- **TIMx_RCR (Repetition Counter Register)**, İleri dönüş (overflow) olayının tekrar sayısını kontrol eder.
- **TIMx_CCR1, TIMx_CCR2, TIMx_CCR3, TIMx_CCR4 (Capture/Compare Register 1, 2, 3, ve 4)**, Capture/compare modunda kullanılan karşılaştırma değerlerini içerir. Bu değerler, belirli bir zaman noktasında veya karşılaştırma olayında kullanılır.
- **TIMx_BDTR (Break and Dead-Time Register)**, Timer'ın kesme ve ölü zaman ayarlarını içerir.
- **TIMx_DCR (DMA Control Register)**, DMA transferlerini kontrol eder.
- **TIMx_DMAR (DMA Address Register)**, DMA transferleri için adres bilgisini içerir.

General Purpose

TIM2, TIM3, TIM4, ve TIM5

- TIM2, TIM3, TIM4, ve TIM5 birimleri, düşük hızlı APB1 (42 MHz) veri yolu üzerinde bulunmaktadır. Eğer APB1 prescaler değeri 1 den farklı ise bu timerların clock frekansları beslendikleri frekansların 2 katına çıkar. Yani 84 MHz clock frekansına sahip olur.
- TIM3 ve TIM4 16-bit'lik sayıcıya, TIM2 ve TIM5 32-bit'lik sayıcıya sahiptirler.
- Bu sayıcılar up, down ve auto-reload modlarda sayma yapabilirler.
- Ayrıca bu sayıcıların otomatik yükleme özellikleri de vardır.
- 16-bit genişliğinde kontrol edilebilir prescaler değeri vardır.
- Bu timer biriminde 4x16 adet yüksek çözünürlüktü capture/compare kanalı bulunur. Bu kanallar; Input Capture, Output Compare, PWM, One-Pulse'dır.
- Dahili diğer Timer birimleri ile senkronizasyon
- Interrupt ve DMA üretimi mevcuttur.
- Clock kaynağı seçimi


Internal clock (CK_INT)
TIMxCLK from RCC
TIMx_ETR
ETR
Polarity selection & edge detector & prescaler
ETRP
Input filter
ETRF
Trigger controller
TRGO
to other timers
to DAC/ADC
ITR0
ITR1
ITR2
ITR3
ITR
TRC
TGI
TRGI
Slave controller mode
Reset, enable, up, count
TI1F_ED
TI1FP1
TI2FP2
Encoder interface
Event
Interrupt & DMA output
U
Auto-reload register
UI
Stop, clear or up/down
CK_PSC
PSC prescaler
CK_CNT
+/-
CNT counter
XOR
TI1
Input filter & edge detector
TI1FP1
TI1FP2
TRC
IC1
Prescaler
IC1PS
CC1I
Capture/Compare 1 register
OC1REF
Output control
OC1
TIMx_CH1
TIMx_CH2
TI2
TI2FP1
TI2FP2
IC2
IC2PS
CC2I
Capture/Compare 2 register
OC2REF
OC2
TIMx_CH3
TI3
TI3FP3
TI3FP4
IC3
IC3PS
CC3I
Capture/Compare 3 register
OC3REF
OC3
TIMx_CH4
TI4
TI4FP3
TI4FP4
IC4
IC4PS
CC4I
Capture/Compare 4 register
OC4REF
OC4
ETRF

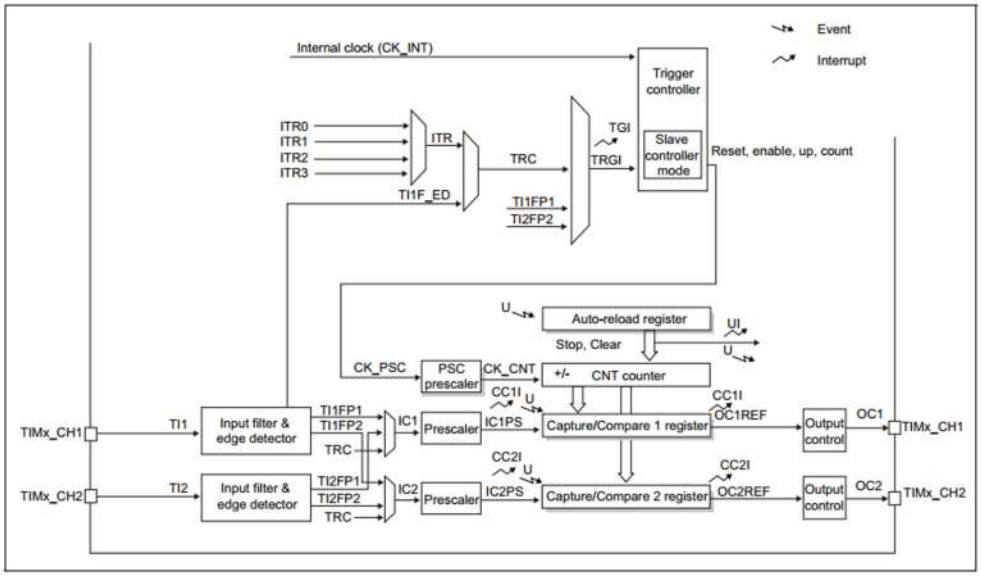
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Reserved																								TI1S	MMS[2:0]			CCDS	Reserved		
	Reset value																									0	0	0	0	0			
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIMx_DIER	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	TIMx_EGR	Reserved																									TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	TIMx_CCMR1 Output Compare mode	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2 Output Compare mode	Reserved																OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Reserved																IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Reserved																CC4NP	Reserved	CC4P	CC4E	CC3NP	Reserved	CC3P	CC3E	CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x24	TIMx_CNT	CNT[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)																ARR[15:0]															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x34	TIMx_CCR1	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Reserved																			DBL[4:0]					Reserved	DBA[4:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0		
0x4C	TIMx_DMAR	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TIM2_OR	Reserved																Reserved				ITR1_RMP		Reserved									
	Reset value																					0	0										
0x50	TIM5_OR	Reserved																Reserved								IT4_RMP	Reserved						
	Reset value																									0	0						

- **TIMx_CR1 (Control Register 1)**, Timer'ın genel kontrol ayarlarını içerir. Timer'ın etkinleştirme, zamanlama modu seçimi, otomatik yeniden başlatma etkinleştirme gibi ayarları içerir.
- **TIMx_CR2 (Control Register 2)**, Timer'ın özel kontrol ayarlarını içerir. Bu register, master mode seçimi gibi özellikleri kontrol eder.

- **TIMx_SMCR (Slave Mode Control Register)**, Timer'ın slave (köle) modunu kontrol eder. Dış bir kaynaktan senkronize olma veya bir başka Timer'ın takip etme gibi işlevleri içerir.
- **TIMx_DIER (DMA/Interrupt Enable Register)**, DMA (Direct Memory Access) ve kesme (interrupt) izinlerini kontrol eder. Belirli olayların tetiklenmesi durumunda bir kesme talebi veya DMA transferi başlatma gibi işlevleri etkinleştirir veya devre dışı bırakır.
- **TIMx_SR (Status Register)**, Timer'ın durumuyla ilgili bilgileri içerir. Taşma, karşılaştırma olayları gibi çeşitli olayları takip eder.
- **TIMx_EGR (Event Generation Register)**, Olayların elle tetiklenmesini sağlar. Bu register üzerinden bir olayı (event) hemen tetikleyebilirsiniz.
- **TIMx_CCMR1** ve **TIMx_CCMR2 (Capture/Compare Mode Register 1 ve 2)**, Capture/compare modu için ayarları içerir. Timer'ın çeşitli modlarını, giriş ve çıkış ayarlarını belirler.
- **TIMx_CCER (Capture/Compare Enable Register)**, Capture/compare kanallarını etkinleştirme veya devre dışı bırakma işlemlerini kontrol eder.
- **TIMx_CNT (Counter Register)**, Timer'ın ana sayaç değerini içerir. Bu register, zamanlayıcının sayma işlemini temsil eder.
- **TIMx_PSC (Prescaler Register)**, Timer'ın ön bölücü (prescaler) değerini içerir. Bu değer, timer'ın sayma hızını kontrol eder.
- **TIMx_ARR (Auto-Reload Register)**, Timer'ın otomatik yeniden başlatma değerini içerir. Bu değer, sayacın bir döngü tamamladığında otomatik olarak tekrar başlamasını sağlar.
- **TIMx_CCR1, TIMx_CCR2, TIMx_CCR3, TIMx_CCR4 (Capture/Compare Register 1, 2, 3, ve 4)**, Capture/compare modunda kullanılan karşılaştırma değerlerini içerir. Bu değerler, belirli bir zaman noktasında veya karşılaştırma olayında kullanılır.
- **TIMx_DCR (DMA Control Register)**, DMA transferlerini kontrol eder.
- **TIMx_DMAR (DMA Address Register)**, DMA transferleri için adres bilgisini içerir.
- **TIMx_OR (Option Register)**, Timer'ın özel seçeneklerini kontrol eder. Bu register, özel özelliklerin etkinleştirilmesi veya devre dışı bırakılması için kullanılır.

TIM9, TIM10, TIM11, TIM12, TIM13, TIM14

- TIM9 yüksek hızlı APB2 (84 MHz) ve TIM12 düşük hızlı APB1 (42 MHz) üzerinde bulunmaktadır.
- Bu birimlerin frekansları diğerlerinde olduğu gibi veriyolu hızlarının iki katında çalışabilirler.
- TIM9 ve TIM12 birimleri 16 bitlik sayıcıya sahiptirler. Bu sayıcılar sadece yukarı sayma yapabilirler. Ayrıca bu sayıcıların otomatik geri yükleme özellikleri de bulunmaktadır.
- Bu timer birimlerinde 2x16 adet yüksek çözünürlüklü capture/compare kanalı da bulunur.
  Bu kanallar giriş öıkış olarak ayarlanabilir, çıkış karşılaştırabilir, PWM sinyali üretebilir, sinyal yakalayabilir ve harici bir PWM sinyalini algılayabilirler.
- TIM10 ve TIM11 yüksek hızlı APB2 (84 MHz) ve TIM13 ve TIM14 düşük hızlı APB1 (42 MHz) üzerinde bulunmaktadır. Bu birimlerin frekansları diğerlerinde olduğu gibi veriyolu hızlarının iki katında çalışabilirler.
- Bu birimler 16 bitlik sayıcıya sahiptirler. Bu sayıcılar sadece yukarı sayma yapabilirler. Ayrıca bu sayıcıların otomatik geri yükleme özellikleri de bulunmaktadır.
- Bu timer birimlerinde 2x16 adet yüksek çözünürlüklü capture/compare kanalı da bulunur.
  Bu kanallar giriş öıkış olarak ayarlanabilir, çıkış karşılaştırabilir, PWM sinyali üretebilir, sinyal yakalayabilir ve harici bir PWM sinyalini algılayabilirler.



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																							0	0	0				0	0	0	0
0x0C	TIMx_DIER	Reserved																														CC1IE	UIE
	Reset value																															0	0
0x10	TIMx_SR	Reserved																						CC1OF		Reserved						CC1IF	UIF
	Reset value																							0								0	0
0x14	TIMx_EGR	Reserved																														CC1G	UG
	Reset value																															0	0
0x18	TIMx_CCMR1 Output compare mode	Reserved																									OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Reserved																								IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Reserved																												CC1NP	Reserved	CC1P	CC1E
	Reset value																													0		0	0
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TIMx_OR	Reserved																														TI1_RMP	
	Reset value																															0	0

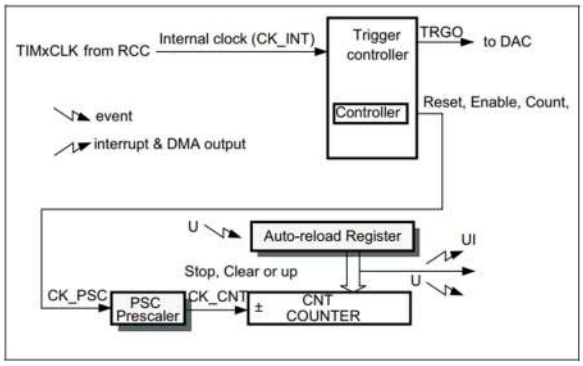
- **TIMx_CR1 (Control Register 1)**, Timer'ın genel kontrol ayarlarını içerir. Etkinleştirme, zamanlama modu seçimi, otomatik yeniden başlatma etkinleştirme ve diğer bazı genel ayarları içerir.
- **TIMx_DIER (DMA/Interrupt Enable Register)**, DMA (Direct Memory Access) ve kesme (interrupt) izinlerini kontrol eder. Belirli olayların tetiklenmesi durumunda bir kesme talebi veya DMA transferi başlatma gibi işlevleri etkinleştirir veya devre dışı bırakır.
- **TIMx_SR (Status Register)**, Timer'ın durumuyla ilgili bilgileri içerir. Taşma, karşılaştırma olayları gibi çeşitli olayları takip eder.
- **TIMx_EGR (Event Generation Register)**, Olayların elle tetiklenmesini sağlar. Bu register üzerinden bir olayı event hemen tetikleyebilirsiniz.
- **TIMx_CCMR1 (Capture/Compare Mode Register 1)**, Yakalama/karşılaştırma modu için ayarları içerir. Timer'ın çeşitli modlarını, giriş ve çıkış ayarlarını belirler.
- **TIMx_CCER (Capture/Compare Enable Register)**, Capture/compare kanallarını etkinleştirme veya devre dışı bırakma işlemlerini kontrol eder.
- **TIMx_CNT (Counter Register)**, Timer'ın ana sayaç değerini içerir. Bu register, zamanlayıcının sayma işlemini temsil eder.
- **TIMx_PSC (Prescaler Register)**, Timer'ın ön bölücü prescaler değerini içerir. Bu değer, timer'ın sayma hızını kontrol eder.
- **TIMx_ARR (Auto-Reload Register)**, Timer'ın otomatik yeniden başlatma değerini içerir. Bu değer, sayacın bir döngü tamamladığında otomatik olarak tekrar başlamasını sağlar.
- **TIMx_CCR1 (Capture/Compare Register 1)**, Capture/compare modunda kullanılan karşılaştırma değerini içerir. Bu değer, belirli bir zaman noktasında veya karşılaştırma olayında kullanılır.
- **TIMx_OR (Option Register)**, Timer'ın özel seçeneklerini kontrol eder. Bu register, özel özelliklerin etkinleştirilmesi veya devre dışı bırakılması için kullanılır.

Basic Timer

TIM6, TIM7

- TIM6 ve TIM7 Basic Timer birimleri genel sayaç olarak kullanılabilecekleri gibi, spesifik olarak DAC biriminin tetikleyicisi olarak da kullanılabilmektedir.
- 16-bit genişliğinde auto-reload upcounter yani otomatik geri yüklenen artan sayaca sahiptir.
- 16-bit genişliğinde kontrol edilebilir prescaler değere sahiptir.
- DAC birimi için tetikleme çıkışlarına sahiptir.

- Interrupt ve DMA üretimi mevcuttur.
- Çalışma prensibi genel amaçlı timer'ların çalışma prensibi ile aynıdır.

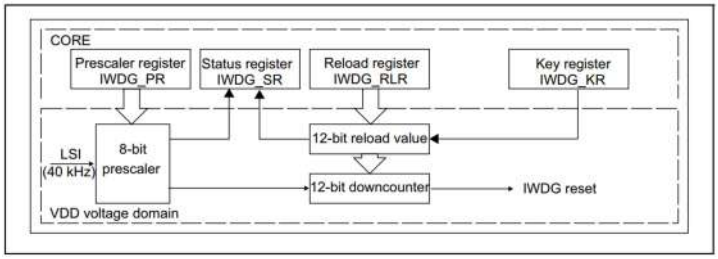


Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																								ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																									0				0	0	0	0
0x04	TIMx_CR2	Reserved																									MMS[2:0]			Reserved				
	Reset value																										0	0	0				
0x0C	TIMx_DIER	Reserved																							UDE	Reserved							UIE
	Reset value																								0								0
0x10	TIMx_SR	Reserved																															UIF
	Reset value																																0
0x14	TIMx_EGR	Reserved																															UG
	Reset value																																0
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

- **TIMx_CR1 (Control Register 1)**, Timer'ın genel kontrol ayarlarını içerir. Örneğin, Timer'ın etkinleştirilmesi, zamanlama modu seçimi, otomatik yeniden başlatma etkinleştirme gibi ayarlar bu register üzerinden yapılmaktadır.
- **TIMx_CR2 (Control Register 2)**, dış tetikleyici konfigürasyonları gibi timer'ın belirli özelliklerini ayarlamanızı sağlar.
- **TIMx_DIER (DMA/Interrupt Enable Register)**, DMA ve interrupt izinlerini kontrol eder. Belirli olayların tetiklenmesi durumunda bir kesme talebi veya DMA transferi başlatma gibi işlevleri etkinleştirir veya devre dışı bırakır.
- **TIMx_SR (Status Register)**, bir taşma durumu overflow olup olmadığını veya bir karşılaştırma olayının gerçekleşip gerçekleşmediğini belirtir.
- **TIMx_EGR (Event Generation Register)**, Olayların elle tetiklenmesini sağlar. Bu register üzerinden bir olayı event hemen tetikleyebilirsiniz.
- **TIMx_CNT (Counter Register)**, Timer'ın ana sayaç değerini içerir. Bu register, zamanlayıcının sayma işlemini temsil eder.
- **TIMx_PSC (Prescaler Register)**, Timer'ın prescaler değerini içerir. Bu değer, timer'ın sayma hızını kontrol eder.
- **TIMx_ARR (Auto-Reload Register)**, Timer'ın otomatik yeniden başlatma değerini içerir. Bu değer, sayacın bir döngü tamamladığında otomatik olarak tekrar başlamasını sağlar.

Independent Watchdog (IWDG)

- IWDG, işlemci saatinden bağımsız, kendine ait dahili RC osilatörden (LSI 32 KHz) beslenen bir watchdog timerdır.
- Mikrodenetleyici içerisindeki amacı da bekçilik yapmaktır. Mikrodenetleyici, harici sebeplerden veya kodlardaki bir hata sebebiyle kilitlenebilir. Mikrodenetleyici kilitlendiğinde, yürüttüğü işlemler durur. Bu tür durumlarda mikrodenetleyicinin tekrar başlatılması gereklidir. İşte watchdog timerlar burada devreye girerler. Watchdog timerlarda belirlenen bir süre sonunda sıfırlanırlar ve işlemciyi resetlerler.

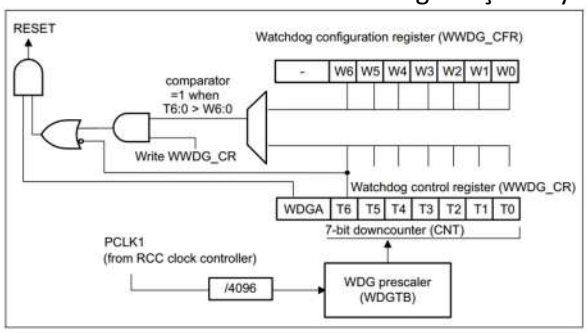


Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Reserved																													PR[2:0]		
	Reset value																														0	0	0
0x08	IWDG_RLR	Reserved																				RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Reserved																														RVU	PVU
	Reset value																															0	0

- **IWDG_KR (Key Register)**, IWDG'yi kontrol etmek için kullanılan anahtar değerleri içerir. İlgili anahtar değerleri yazılarak IWDG'nin başlatılması, yeniden başlatılması veya durdurulması gibi işlemler gerçekleştirilir.
- **IWDG_PR (Prescaler Register)**, IWDG'nin zamanlayıcı değerini belirlemek için kullanılır. Zamanlayıcı değeri, bu ön bölücü ile çarparak IWDG'nin zamanlamasını elde eder.
- **IWDG_RLR (Reload Register)**, IWDG'nin zamanlayıcı değerini reload value içerir. IWDG'nin çalışması sırasında bu değer zaman içinde azalır, eğer bu değer sıfıra ulaşırsa, IWDG bir reset sinyali üretir.
- **IWDG_SR (Status Register)**, IWDG'nin durumunu gösteren bilgiler içerir. Örneğin, zaman aşımı durumu gibi bilgiler burada bulunabilir.

Window Watchdog (WWDG)

- WWDG birimi belirli bir pencere içerisinde counter kaydedicisine tekrar değer yüklenebildiği için bu isimle anılmaktadır.
- Ayarlanabilir süre penceresine sahiptir.
- Anormal erken ve anormal geç uygulama davranışını algılayabilir.
- Önceden belirlenen duruma göre işlemciyi resetler.



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	WWDG_CR	Reserved																								WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	WWDG_CFR	Reserved																						EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	WWDG_SR	Reserved																															EWIF
	Reset value																																0

- **WWDG_CR (Control Register)**, WWDG'nin temel kontrol ayarlarını içerir. Özellikle, WWDG'nin etkinleştirilmesi, zamanlayıcı değeri (down-counter) ayarlanması ve bir reset talep biti bulunmaktadır.
- **WWDG_CFR (Configuration Register)**, WWDG'nin daha fazla konfigürasyon ayarlarını içerir. Örneğin, window modunu etkinleştirme, zaman aşımı değeri ve window değeri gibi ayarları içerir.
- **WWDG_SR (Status Register)**, WWDG'nin durumunu gösteren bilgiler içerir. Örneğin, zaman aşımı durumu ve window durumu gibi bilgiler burada bulunabilir.

07 PWM

25 Haziran 2021 Cuma 23:48

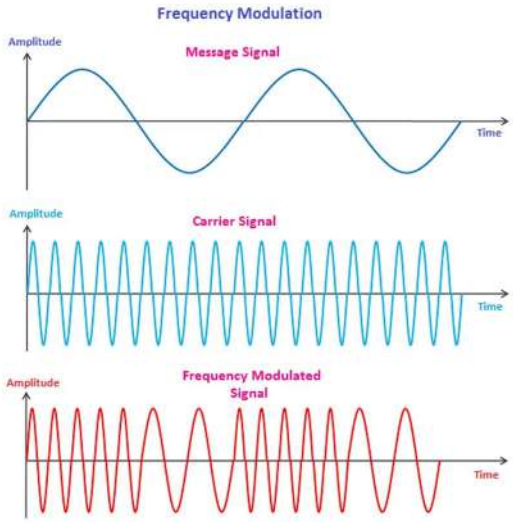
07 PWM

Giriş

- https://www.aydinlatma.org/pwm.html, https://berkannaydin.medium.com/pwm-nedir-5d20287970b5 , https://www.elektrikport.com/teknik-kutuphane/pwm-(sinyal-genislik-modulasyonu)-teknigi-nedir/11717#ad-image-0 , https://www.otomasyonavm.com/tr/pwm-nedir linklerdeki makaleleri okuyabiliriz.
- PWM, **Pulse Width Modulation** (Darbe Genişlik Modülasyonu) bir kare dalga sinyalin, yüksek seviyede kalma süresine müdahale ederek, bu sinyalin gerilimin ortalama değerinin değiştirilmesi olarak tanımlanabilir.
  PWM sinyal ise analogdan farklı olarak sinyalin voltajını değil frekansını referans alır.

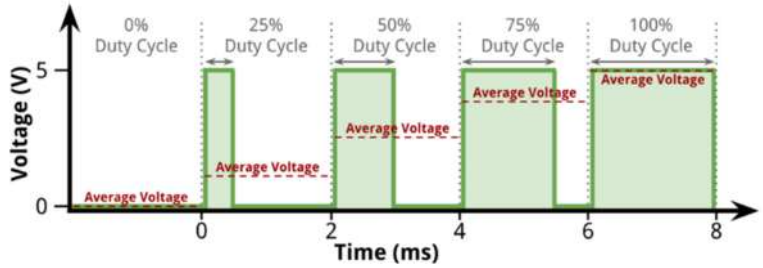
Modülasyon

- Modülasyon haberleşme sistemlerinde sıklıkla kullanılan bir yöntemdir. Bu yöntemde bir girdi sinyali (input) bir taşıyıcı (carrier) sinyal ile taşınır.



- Doğadaki sinyallerin çoğu analog sinyaldir. Elektronik cihazlar ise dijital sinyaller ile işlemleri yapmaktadır.
- Çıktı olarakta tekrar analog çıktılar üretilmektedir.

Görev Döngüsü



- PWM tekniğinde **açık** ve **kapalı** süresi görev döngüsü yani **duty cycle** ile tanımlanır.
  Ton açık süreyi, **Toff** kapalı süreyi temsil eder.
- **Pulse Width**, Ton süresi kadardır. **Period**, Ton ile Toff sürelerin toplamıdır.
- Dutcy Cycle aşağıdaki formül ile hesaplanır.

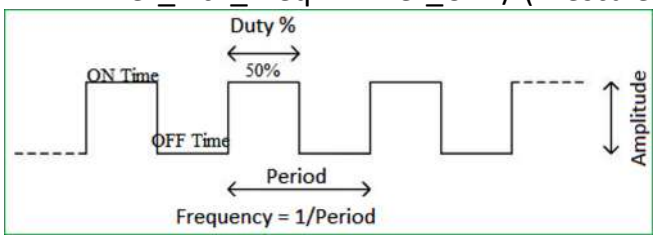
  Duty Cycle= (Ton / Ton + Toff) × 100

  Ton: Açık süresi
  Toff: Kapalı süresi
- Giriş voltaj değeri ile Duty cycle değerini çarparak Ortalama Çıkış Gerilimini hesaplıyoruz.
- Frekans ise aşağıdaki formül ile hesaplanır. Frekans birimi Hz, Periyot birimi s'dir.

  f = 1 / T

  f: Frekans (Hz)
  T: Periyot (s)

- PWM frekansını hesaplamak için, aşağıdaki formüllerden yararlanmamız lazım;
  Period = (Timer_Tick_Freq / PWM_Freq) - 1
  PWM_Freq = Timer_Tick_Freq / (Period + 1)
  Timer_Tick_Freq = Timer_CLK / (Prescaler + 1)



- Timer frekansı kullanıcı tarafından belirlenir. Aynı zamanda PWM'de istenilen frekansta çalışılacağı düşünülecek olursa, bizim belirleyeceğimiz iki değer var. Bunlardan biri prescaler, diğeri ise period. Aslında temel olarak PWM'in istenilen frekansta çalışması için prescaler değeri küçük bir değer seçilir ve period bu değere göre ayarlanır.

Mod Durumu

- **Mod 1**, Yukarı doğru sayarken CNT < CCRX (Capture Compare Register) dan düşükse kanal aktif, diğer durumda pasif olur. Aşağı doğru sayarken CNT > CCRX ise kanal pasif, değilse aktif olur.
- **Mod 2**, Yukarı doğru sayarken CNT < CCRX (Capture Compare Register) dan düşükse kanal pasif, diğer durumda aktif olur. Aşağı doğru sayarken CNT > CCRX is kanal aktif, değilse pasif olur.

Uygulamaları

- Endüstride iletişim, motor kontrol, ısıtma, aydınlatma gibi önemli bir çok alanda kullanılmaktadır.
  - Servo motor kontrolünde, servo motorların konumunu kontrol etmek için
  - Ses kontrolünde, hoparlörlerin sesini kontrol etmek için
  - Oyun konsollarında, oyun kontrollerini sağlamak için
- PWM, ışık kaynağını hızlı bir şekilde açık kapatarak parlaklığı ayarlamayı sağlayan bir modülasyon çeşididir.
- Anahtarlama işleminde açık kalma süresi ne kadar yüksek olursa yüke sağlanan güç yani ışık parlaklığı o kadar fazla olur.

08 UART

5 Mayıs 2021 Çarşamba 08:03

08 UART

Giriş

- https://cennttceylnn.medium.com/elektronik-haberleşme-protokolleri-nedir-d11a6d3a5957 link üzerinden haberleşme protokolleri hakkında bilgi alabiliriz.
- https://www.ercankoclar.com/2018/04/uart-iletisim-protokolu-ve-mikroc-kutuphanesi/ https://arduinodestek.com/uart-haberlesme-nedir-ve-nasil-gerceklesir/
- https://youtu.be/uktFwZX2TTE, https://youtu.be/K0JuUAQYsaQ ve https://youtu.be/UCXVFJSrIbE protokol hakkında videolardan bilgi edinebiliriz.
- https://youtu.be/GRmYKJgAtQ4 , https://youtu.be/NDwpWbXJ0sc , https://youtu.be/vrSzdoKv558 , https://youtu.be/vzRuLn_Gzx8 , https://youtu.be/ic8NUSytU-g , https://youtu.be/y7ZETFlohp0 , https://youtu.be/1IGm99He7g4 linklerinden STM32 ile yapılmış örnek uygulamaları izleyebiliriz.
- UART (Universal Asynchronous Receiver Transmitter), 1 ve 0'lardan oluşan verileri iki dijital sistem arasında alıp verme işlemlerinde kullanılan bir iletişim protokolüdür.

Avantajları ve Dezavantajları

Avantajlar;

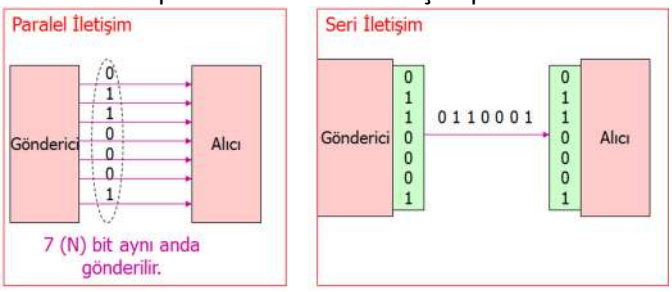
- Sadece iki kablo kullanır.
- Saat sinyali gerekli değildir.
- Hata denetimine izin vermek için bir eşlik biti vardır.
- Veri paketinin yapısı, her iki taraf da buna göre ayarlandığı sürece değiştirilebilir.
- İyi belgelenmiş ve yaygın olarak kullanılan yöntem.

Dezavantajlar;

- Veri çerçevesinin boyutu maksimum 9 bit ile sınırlıdır.
- Birden çok bağımlı veya birden çok ana sistemi desteklemez.
- Her UART'ın baud hızı, birbirinin %10'u dahilinde olmalıdır.

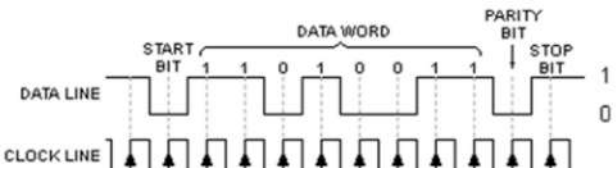
İletişim Yöntemi

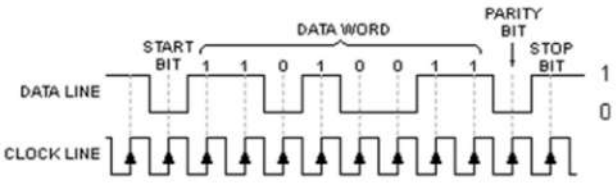
- Dijital sistemlerde iletişim paralel ve seri olmak üzere iki türlü yapılır.
- **Paralel** iletişimde bilgi vericiden alıcıya aynı anda birden fazla bit gidecek şekilde gönderilir. Böylece tek hamlede birden fazla veri karşı tarafa gönderildiği için iletişim hızı yüksektir. Fakat bu iletişim türünde kullanılan hat sayısı fazladır ve uzun mesafeler için uygun değildir.
- **Seri** iletişimde ise vericiden alıcıya gönderilecek bilgi, tek hat üzerinden sırayla gönderilir. Bu şekilde giden bilginin, tek hamlede tek biti gönderebileceği için iletişim hızı yavaşlatır. Fakat seri iletişimde hat sayısı azdır ve uzun mesafeli iletişim için daha uygundur.
- USART protokollü bir seri iletişim protokolüdür.



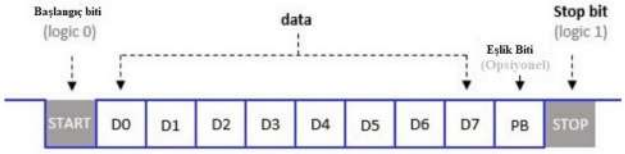
İletişim Modu

- USART protokolünde veriler senkron veya asenkron olarak alınabilirler.
- **Senkron** veri alışverişinde bir data hattı ve bir clock hattı bulunmalıdır.
  Daha hattından gidecek veriler clock hattından gönderilen sinyalin her düşen veya yükselen kenarında alıcıya iletilir.





- **Asenkron** modda ise verilerin iletilmesinde bir clock hattına ihtiyaç duyulmaz.



Parametreler

- Verilerin gönderilmeye başlayacağı, alıcıya bir başlangıç için **Start** sinyali ile bildirir ve hemen arkasından veriler akmaya başlar daha sonrasında **Stop** biti ile sonlandırılır.
- Start biti "O" stop biti "1" verilerinden oluşmaktadır. Veri bitleri ise 7 veya 8 bit olabilir.
- Verilerin doğru olarak gönderilip gönderilmediğini anlamak için kullanılan **Parity** biti bulunmaktadır. Parity bitinin gönderilmesi şart değildir.
- Parity biti genellikle üç farklı şekilde kullanılır. Even, Odd ve None olarak seçilebilir.
  Parite, **Even** olarak kullanıldığında, iletilen verinin toplamda çift sayıda yüksek seviyeye sahip olması gerekmektedir. Eğer iletilen verinin yüksek seviyeye sahip bit sayısı tek ise, parite biti 1 olacak şekilde ayarlanır ve toplamda çift sayıda yüksek seviye olmasını sağlar. Eğer iletilen verinin yüksek seviyeye sahip bit sayısı zaten çift ise, parite biti 0 olarak ayarlanır. **Odd** ise bunun tam tersidir. Parite bitinin kullanılmadığı durumlarda **None** seçeneğini kullanırız ve böylece iletilen verinin doğruluğu kontrol edilmez. Parite biti için boş bir bit alanı bırakılır ve sadece veri bitleri iletimi gerçekleştirilir.
- Parite biti, basit bir hata kontrol mekanizmasıdır ve veri bütünlüğünü sağlamak için kullanılır. Ancak, parite biti tek başına tüm hataları tespit etmek veya düzeltmek için yeterli değildir. Daha güvenli ve sağlam hata kontrol yöntemleri için farklı yöntemler ve algoritmalar kullanılabilir, örneğin **CRC** (Cycle Redundancy Check)
- **Baudrate** saniyede gönderilen bit sayısıdır ve bps (bits per second - saniyede gönderilen bit sayısı) birimi ile ölçülür. Standart veri gönderme hızları 110, 150, 300, 600, 1200, 2400, 4800, 9600, 56000, 115200 gibi hızdadır. Baudrate hızı **arttıkça** veri iletim mesafesi **azalır**.
- İki cihaz arasında UART haberleşmesi yapılıyorsa, her iki cihazın da **aynı** baud oranı, veri bitleri, parite biti ve stop bitleri yapılandırmasına sahip olması gerekmektedir. Bu parametrelerin doğru bir şekilde yapılandırılması, güvenilir ve hatasız veri iletimini sağlar.

Veri İletimi Süresi

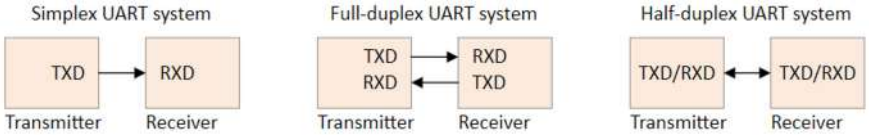
- Belirli bir baud hızında (baud rate) veri iletiminin ne kadar sürdüğünü hesaplamak için aşağıdaki hesaplama yöntemi kullanılır.
  - **Baud Hızı (Baud Rate):** Bu, saniyede kaç karakter gönderildiğini belirtir. Eğer her karekter 1 bit'e eşitse, bu durumda saniyede kaç bit gönderildiğini de gösterir.
    Örneğin, 9600 baud, saniyede 9600 bit'in iletildiği anlamına gelir.
  - **Bir Bit'in İletim Süresi:** Bir bit'in iletilmesi için geçen süre, baud hızı ile ters orantılıdır. Yani bir bit'in iletim süresi, Bit Süresi = 1 / Baud Rate hesap edilir. Saniye cinsinden sonuç verir.
  - **Bir Byte'ın İletim Süresi:** Bir byte 8 bit'ten oluşur, fakat genellikle veri iletiminde başlangıç biti, durdurma biti ve bazen hata kontrol biti gibi ek bitler de olabilir. Standart bir asenkron iletişimde bu, genellikle 10 bit olarak kabul edilir. Byte Süresi = Bit Sayısı × Bit Süresi şeklinde hesap edilir.
- Baud hızına göre bir byte'ın iletim süresini hesaplamak için kullanılan formül,

  Byte Süresi = 10 / Baud Hızı
  - Bu formülü kullanarak farklı baud hızları için byte süreleri;
    115200 baud hızı için Byte Süresi ≈ 86.8 µs
    57600 baud hızı için Byte Süresi ≈ 173.6 µs
    38400 baud hızı için Byte Süresi ≈ 260.4 µs
    19200 baud hızı için Byte Süresi ≈ 520.8 µs
    9600 baud hızı için Byte Süresi ≈ 1.04 ms
    4800 baud hızı için Byte Süresi ≈ 2.08 ms

Çalışma Modları

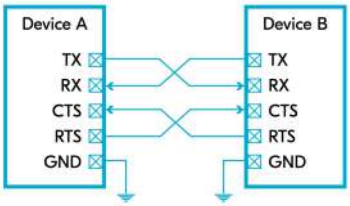
- UART'ın üç çalışma modu vardır. Bunlar Full Duplex, Half duplex ve Simplex'dir.

- **Full Duplex** iletişimde, veri gönderme ve veri alma işlemleri aynı anda ve bağımsız olarak gerçekleştirilir.
- **Half Duplex** iletişimde, veri gönderme ve veri alma işlemleri aynı hattı paylaşan cihazlar arasında sırayla gerçekleştirilir.
- **Simplex** iletişimde, veri iletimi sadece bir yönde gerçekleşir.



Pin Yapısı

- **TX** (Transmit) ve **RX** (Receive), UART haberleşmesinde kullanılan veri gönderme ve veri alma hatlarını temsil eder.
- **CTS** (Clear To Send) ve **RTS** (Request To Send) ise UART haberleşmesinde kullanılan kontrol sinyalleridir. RTS ve CTS sinyalleri, veri akışını kontrol etmek için kullanılır ve genellikle aşırı yüklenmeyi önlemek veya veri kaybını engellemek için kullanılır.
- **TX** hattı, veri gönderici tarafından kullanılan seri veri gönderme hattını temsil eder. Veri gönderici, veri paketini seri olarak TX hattına gönderir ve bu hattı kullanarak veriyi alıcıya iletir.
- **RX** hattı, veri alıcı tarafından kullanılan seri veri alma hattını temsil eder. Veri alıcı, veriyi seri olarak RX hattından alır ve bu hattı kullanarak veriyi işler.
- Veri gönderici tarafından veri göndermeye hazır olduğunu bildirmek için **RTS** sinyalini HIGH seviyeye çeker. Bu, veri alıcının veriyi almasını ve işlemeye başlamasını sağlar.
- Veri alıcı tarafından veri almayı ve işlemeyi kabul etmek için hazır olduğunu belirtmek için **CTS** sinyalini HIGH seviyede tutar. Bu, veri göndericinin veriyi göndermeye başlamasını sağlar.
- RX giriş pini ile TX çıkış pini ve CTS giriş pini ile RTS çıkış pini birbirlerine ters bağlanır.

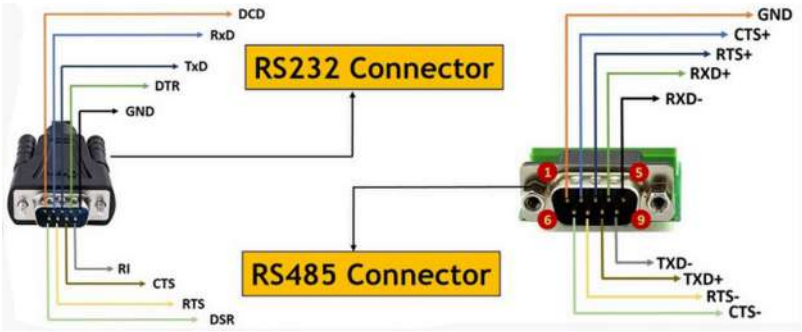


Fiziksel Standartlar

- USART, seri iletişimde geniş bir kullanım alanına sahiptir ve çeşitli protokollerin uygulanmasına olanak tanır. Bu protokoller arasında RS232, RS422, RS485 fiziksel standartlar yer alır.
- https://blog.direnc.net/rs232-ve-rs485-nedir-kullanim-alani-avantaj/, https://www.elektrikde.com/rs-232-rs-485-ve-rs-422-seri-iletisim/ ve https://youtu.be/ChCRlU2kEE0 link üzerinden fiziksel standartlar hakkında bilgi edinebiliriz.
- **RS232**, tek bir veri iletim hattı üzerinden iletişim sağlar ve asenkron veri iletimine uygun bir protokol kullanır. Genellikle 15 volt ile -15 volt arasında bir gerilim seviyesi kullanır. Seri iletişim yaklaşık 15 metre kadardır.
- **RS-422** ve **RS-485**, her ikisi de seri haberleşme standardı olan ve diferansiyel sinyal iletimini kullanan protokollerdir. RS485, RS422'nin bir üst kümesidir, bu nedenle tüm RS422 cihazları RS485 tarafından kontrol edilebilir.
    - RS422 hattında yalnızca bir verici bulunabilirken, bu vericiden gelen sinyalleri alabilen birden fazla alıcı olabilir.
    - RS485, birden fazla cihaz arasında noktadan noktaya veya çok noktaya bağlantılar sağlayabilir.
- RS232 ile RS485 standartlarının özellikleri şu şekildedir:

	RS232	RS485
Voltaj Sistemi	Gerilim seviyesine dayalı	Diferansiyel
Tek Hatta Toplam Sürücü ve Alıcı	1 Sürücü, 1 Alıcı	32 Sürücü, 32 Alıcı (Aynı anda bir Sürücü etkin)
Hat Yapılandırması	Noktadan Noktaya	Çok Aktarmalı
Maksimum Operasyonel Mesafe	15M/50FT	1200M / 3000FT
Maksimum Veri İletim Hızı	1 MBit/sn	10 MBit/sn
Maksimum Sürücü Çıkış Voltajı	±25V	-7V ila +12V
Alıcı Giriş Direnci	3 ila 7 kΩ	12 kΩ
Alıcı Giriş Voltaj Aralığı	±15V	-7V ila +12V
Alıcı Duyarlılığı	±3V	±200mV

- RS232 ve RS485 konnektörlerinin pin bağlantı bilgileri şu şekildedir:

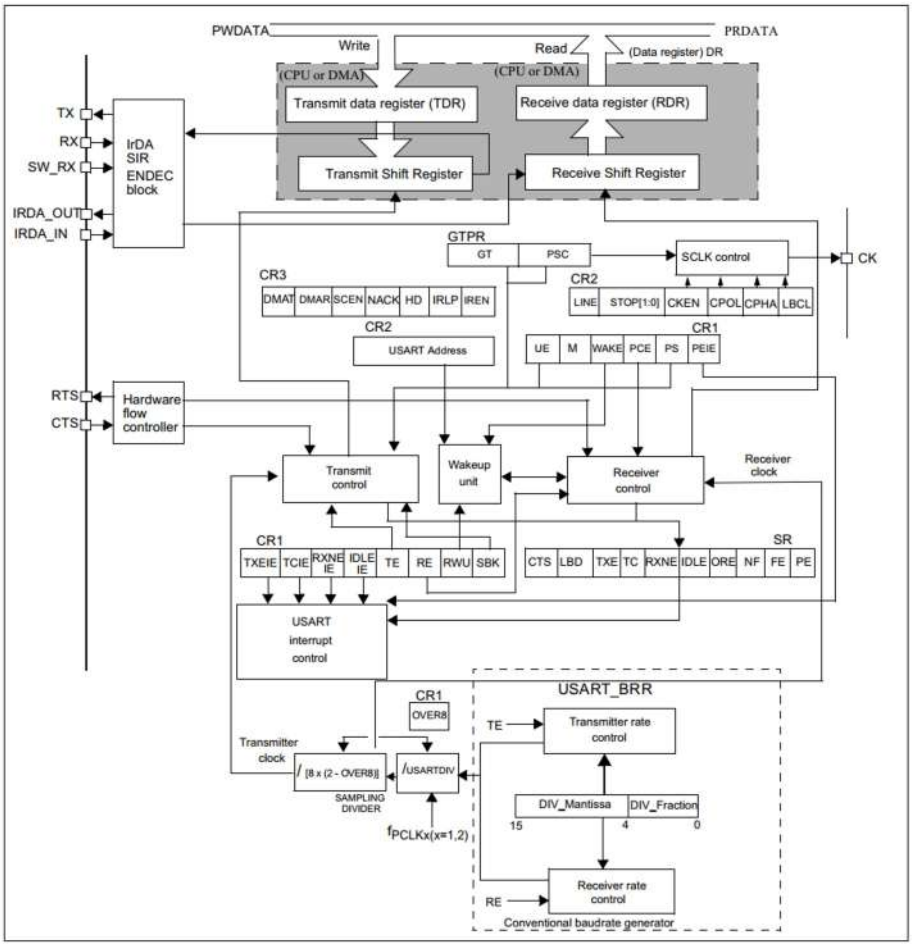


- RS-232, eski ve yaygın olarak kullanılan bir seri haberleşme standardıdır. Ancak daha yüksek veri hızları, uzun mesafe iletimi ve çok noktalı haberleşme gerektiren uygulamalarda RS-485 gibi daha modern haberleşme standartları tercih edilmektedir.
- RS422 ve RS485 protokollerinde **2-wire** ve **4-wire** bağlantı tipleri, veri iletişim şekline göre farklılık gösterir.
    - 2-Wire bağlantı, **Half-duplex** iletişim sağlar. Bu yüzden iletim ve alım aynı anda yapılamaz. Sinyal hattı, **A** ve **B** hatları üzerinden veri iletilir. Bu iki tel, hem verici hem de alıcı tarafında kullanılır, bu nedenle sadece bir verici aktif olduğunda iletişim gerçekleşir.
    - 4-Wire bağlantı, **Full-duplex** iletişim sağlar. Bu sayede veri aynı anda her iki yönde de iletilebilir. İki ayrı çift sinyal hattı kullanılır. Bir çift **Tx+, Tx-** veriyi gönderirken, diğer çift **Rx+, Rx-** veriyi alır. Bu sayede, veri iletimi ve alımı için bağımsız hatlar oluşturulur.

Haberleşme Metotları

- UART üzerinden Polling, Interrupt ve DMA olmak üzere üç farklı haberleşme yapılabilir.
- https://deepbluembedded.com/how-to-receive-uart-serial-data-with-stm32-dma-interrupt-polling/, https://controllerstech.com/uart-receive-in-stm32/ ve https://controllerstech.com/uart-transmit-in-stm32/ link ile bu metotlar ile yapılan örnekleri inceleyebiliriz.
- **Polling** yani Yoklama yöntemi, mikrodenetleyici tarafından sürekli olarak UART veri alımını kontrol etmek için kullanılır. Mikrodenetleyici, UART veri alımını düzenli aralıklarla sorgular ve yeni veri varsa onu işler. Veri gelene kadar mikrodenetleyici diğer işlemleri yapamaz ve sürekli olarak UART'ı kontrol etmek zorunda kalır. Bu yöntem, basit uygulamalarda ve düşük hızlı veri iletiminde tercih edilebilir.
  Yalnızca UART kullanıyorsak ve başka bir şey kullanmıyorsak bu kullandığımız polling yöntemi kullanmak iyidir, aksi takdirde diğer tüm işlemler etkilenecektir.
- **Interrupt** yöntemi, UART'dan veri alındığında veya veri gönderildiğinde mikrodenetleyiciye kesme sinyali göndererek mikrodenetleyicinin normal işlemesini kesmesini sağlar. Bu yöntem, mikrodenetleyicinin sürekli olarak UART'yı sorgulamaktan kurtulmasını sağlar ve daha etkili bir şekilde diğer görevlerini gerçekleştirmesine olanak tanır.
  Interrupt yöntemi, yüksek hızlı veri iletimi veya **zamanlama hassasiyeti** gerektiren uygulamalarda tercih edilir.
- **DMA** yöntemi, veri transferini mikrodenetleyicinin müdahalesi olmadan doğrudan bellekten yapılmasını sağlar. DMA denetleyici, UART'dan gelen veya UART'a gönderilecek verileri doğrudan bellekten okur veya belleğe yazar. Bu yöntem, mikrodenetleyicinin UART veri transferiyle uğraşmadan diğer işlemleri gerçekleştirmesine olanak sağlar ve veri transferinde yüksek hızlı ve verimli bir çözümdür.
  DMA yöntemi, yüksek hızlı veri transferlerinde veya **sürekli veri akışı** gerektiren uygulamalarda kullanılır.
- Özellikle USART ile gönderilecek yüklü miktarda verimiz var ise, bu veriyi döngü içerisinde göndermek, işlemci zamanının önemli bir bölümünü harcayacaktır. Baud hızımız ne kadar az ise, gönderme hızımız o kadar düşecek, dolayısıyla bekleme hızımız da o kadar artacaktır.
- Gönderme işleminin bitmesini beklemeden işimize devam edebilmek için ya DMA ya da EXTI kullanırız.
- Interrupt kullanımında ise CPU tarafından saniyede çok sayıda kesinti yapılması gerekecektir. Bu yüzden çok etkili yöntem değildir. Verileri doğrudan belleğe yönlendirmek için DMA biriminin kullanılması en etkili yöntemdir.

Birim Yapısı



Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USART_SR**	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	**USART_DR**	Reserved																						DR[8:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	**USART_BRR**	Reserved																DIV_Mantissa[15:4]												DIV_Fraction [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USART_CR1**	Reserved																	OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USART_CR2**	Reserved																		LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			
	Reset value																			0	0	0	0	0	0	0	0	0	0		0	0	0	0
0x14	**USART_CR3**	Reserved																				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USART_GTPR**	Reserved																GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **USART_SR** (Status Register), Bu kayıt, iletişim durumu hakkında bilgi sağlar. Örneğin, veri alımı veya iletimi tamamlandığında veya hata durumlarında flag içerir.
- **USART_DR** (Data Register), iletişimde iletilen veya alınan veriyi tutar. Veriyi bu kayıta yazarak iletimi başlatabilir veya bu kayıttan okuyarak alınan veriyi elde edebilirsiniz.
- **USART_BRR** (Baud Rate Register), iletişim hızını ayarlamak için kullanılır.
- **USART_CR1** ve **USART_CR2** (Control Register 1 ve 2), UART modunu, iletim ve alım parametrelerini ve diğer iletişim ayarlarını kontrol eder.
- **USART_CR3** (Control Register 3), üçüncü kontrol kaydıdır ve DMA ayarları gibi daha gelişmiş ayarları kontrol eder.
- **USART_GTPR** (Guard Time and Prescaler Register), zamanlama ayarları için kullanılır.

09 SPI

5 Mayıs 2021 Çarşamba 08:03

09 SPI

Giriş

- https://ozdenercin.com/2019/02/01/spi-seri-haberlesme-protokolu/ , https://arduinodestek.com/spi-haberlesme-protokolu-nedir-ve-nasil-gerceklesir/ https://devreyakan.com/spi-nedir/ linkinden ayrıntılı bilgilere ulaşabiliriz.
- SPI (Serial Peripheral Interface), mikrodenetleyiciler, sensörler, dijital IC'ler ve diğer entegre devreler arasında seri veri iletişimi için kullanılan bir seri senkron iletişim protokolüdür
- Özellik ve kullanım olarak I2C'ye benzer. I2C'de olduğu gibi bir adet Master cihaz bulunur. Bu cihaz hatta bağlı çevresel cihazları kontrol eder.
- Çevresel cihazlarla veya diğer mikrodenetleyicilerle veri transferi sağlayan yazılım/donanım tabanlı seri iletişim protokolüdür. Bu haberleşme şekli karşılıklı iki tarafın **clocklarının senkronize çalışmasıyla** data iletişimi sağlamaktadır.
- SPI, genellikle düşük maliyetli, düşük güç tüketimi gerektiren uygulamalarda kullanılır. Özellikle mikrodenetleyiciler, sensörler, veri dönüştürücüler, hafıza kartları ve diğer entegre devreler arasında veri iletişimi için tercih edilir.
- SPI'nin hızlı ve basit bir iletişim protokolü olması, çeşitli uygulama alanlarında yaygın olarak kullanılmasını sağlar.

Avantajları ve Dezavantajları

Avantajlar;

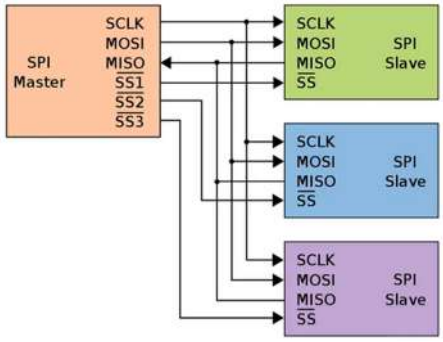
- Başlatma ve durdurma biti yok, böylece veriler kesintisiz olarak aktarılabilir.
- I2C gibi karmaşık bağımlı adresleme sistemi yok.
- I2C'den daha yüksek veri aktarım hızı (neredeyse iki kat daha hızlı).
- Ayrı MISO ve MOSI hatları, böylece veri aynı anda gönderilebilir ve alınabilir.

Dezavantajlar;

- Dört kablo kullanır (I2C ve UART'lar iki kablo kullanır).
- Verilerin başarıyla alındığına dair bir onay yok (I2C de vardır).
- UART'taki eşlik biti gibi hata denetimi biçimi yok.
- Yalnızca tek bir master'a izin verir.

Bağlantılar

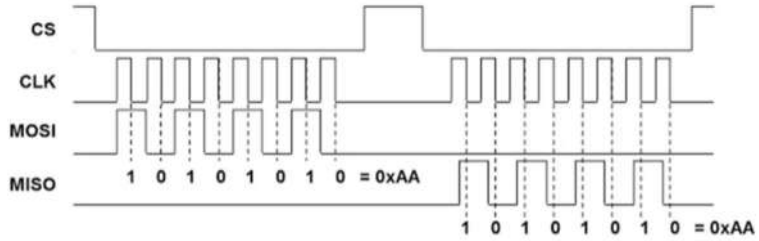
- SPI, genellikle dört telli bir bağlantıyla gerçekleştirilir:
  MOSI (Master Out Slave In), Master cihazdan (genellikle mikrodenetleyici) slave cihaza veri gönderir.
  MISO (Master In Slave Out), Slave cihazdan master cihaza veriyi gönderir.
  SCLK (Serial Clock): Saat sinyali, veri iletim hızını senkronize eder.
  Bu sinyal sadece master cihaz tarafından üretilir.
  SS/CS (Slave Select/Chip Select), İletişim kurulacak slave cihazı seçer.
- Slave cihaz donanımsal olarak seçildiği için I2C iletişimindeki gibi adres gönderilmez. Fakat birden fazla slave cihazın SPI veri yoluna bağlanması için birden fazla SS/CS pini kullanır.
  Tüm pinlerin kullanılmasına ihtiyaç yoktur.



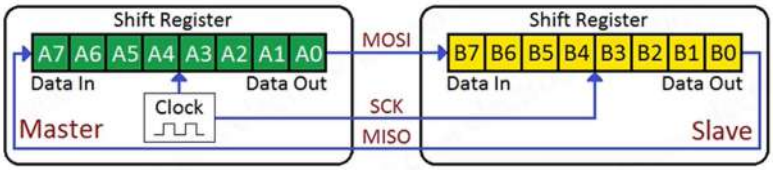
Veri İletimi

- Master, saat sinyalini verir.

- Master, SS/CS pinini, slave etkinleştiren bir **LOW** voltaj durumuna geçirir.
- Master, verileri MOSI hattı boyunca her seferinde bir bit olarak slave'e gönderir. Slave, bitleri aldıkça okur.
- Bir yanıt gerekiyorsa, bağımlı, MISO hattı boyunca her seferinde bir bit veriyi master'a döndürür. Master, bitleri aldıkça okur.



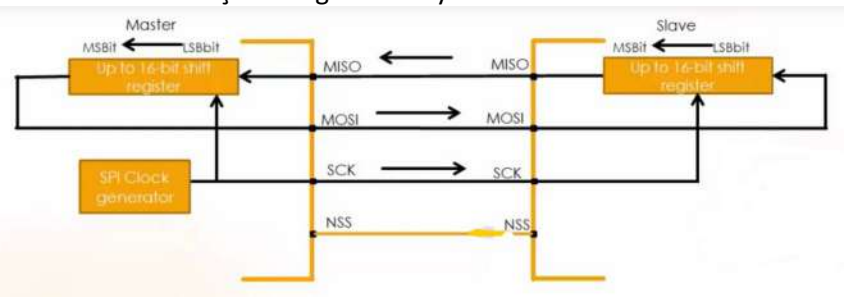
- SPI'da veri iletim sırası genellikle **8 bitlik** veri paketleri halinde olur, ancak bu uzunluğu değiştirilebilir. Veri iletim sırası, verilerin en yüksek veya en düşük anlamlı bit ile başlayıp bitişebileceği şekilde yapılandırılabilir.



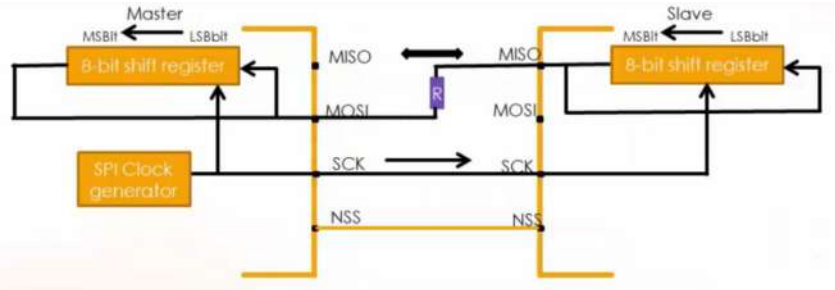
- Ana cihaz, iletişimi **başlatır** ve **sonlandırır**. Slave cihazlar ise ana cihazın taleplerine **yanıt** verir.

Veri Yolu

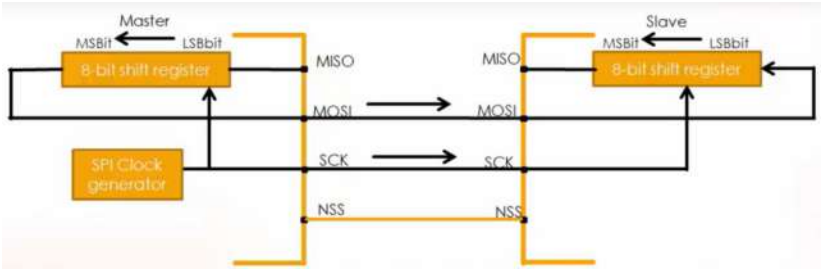
- SPI haberleşmesi için üç farklı mod vardır. Bunlar Full Duplex, Half duplex ve Simplex'dir.
- İkisi çift yönlü iken diğeri tek yönlü haberleşmedir. Bu modlar hakkında detaylı bilgi almak için https://fastbitlab.com/spi-bus-configuration-discussion-full-duplex-half-duplex-simplex/ linkteki yazıyı okuyabiliriz.
- Tek yönlü olan Simplex iletişiminde SS/CS pini kullanılmayabilir, ancak çift yönlü olan Full Duplex ve Half duplex iletişimde ve birden fazla slave cihazı varsa SS/CS pini kullanışlı olabilir.
- **Full Duplex** , hem veri **gönderme** hem de veri **alma** işlemlerinin **aynı anda** gerçekleştirir.
  Veri iletimi için iki ayrı iletişim hattı kullanılır. Her iki tarafta bağımsız olarak veri gönderebilir ve alabilir, bu nedenle iletişim hızı genellikle yüksektir.



- **Half Duplex,** sadece bir veri gönderme ya da bir veri alma işleminin aynı anda gerçekleştirildiği bir çalışma modudur.
  Bu modda, genellikle **tek bir çift yönlü** iletişim hattı kullanılır. Her iki taraf da **aynı iletişim hattı**nı kullanarak veri gönderebilir veya alabilir, ancak aynı anda her iki yönde veri iletimi yapılamaz.

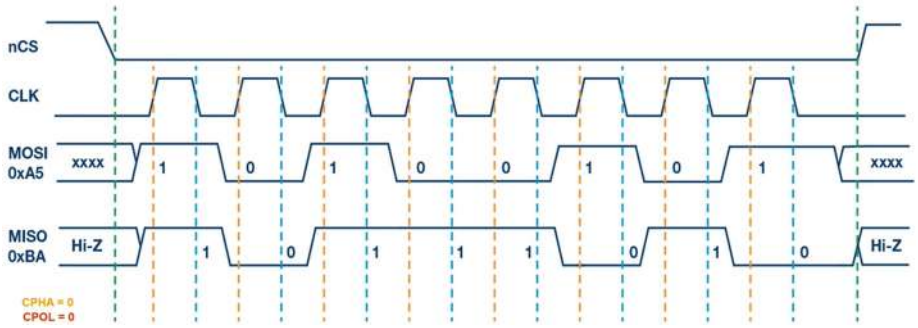


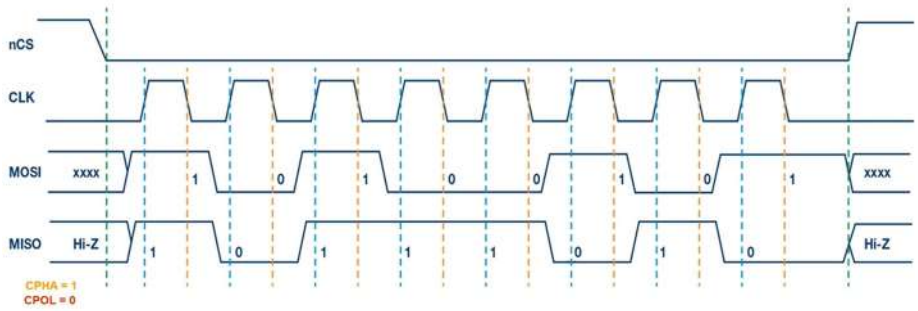
- **Simplex,** sadece **bir yönde** veri iletimine izin veren bir çalışma modudur.
  Genellikle tek bir iletişim hattı kullanılır ve veri gönderme veya veri alma işlemi yapılır. Her iki tarafta aynı anda veri gönderip alamaz.
- Eğer master sadece veri gönderip slave sadece veri alacaksa, bu durumda **MOSI** hattı kullanılır. Diğer bir durumda, master sadece veri alıp slave sadece veri gönderecekse, bu durumda **MISO** hattı tercih edilir.
- Bu iletişimde SS/CS pini kullanılmayabilir. Çünkü Simplex modda genellikle tek yönlü iletişim olduğu için sadece master cihazın veri gönderme veya alım işlemlerini kontrol etmesi yeterlidir.

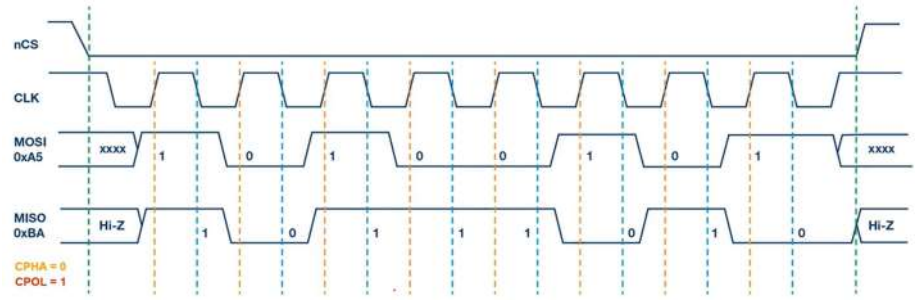


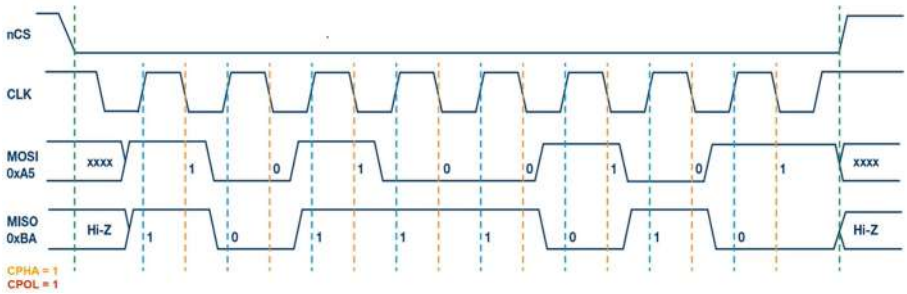
Çalışma Modları

- SPI iletişiminde saat sinyalinin nasıl üretileceğini belirler. Clock polarity (CPOL), saat sinyalinin **yüksek** veya **düşük** seviyede başlayacağını belirtir, clock phase (CPHA) ise veri okuma/yazma işleminin saat sinyalinin hangi **kenarında** gerçekleşeceğini belirler.
- **Saat sinyali**, CPOL değeri 0 olduğunda düşük seviyeden, 1 olduğunda ise yüksek seviyeden başlar.
  Veri, CPHA değeri 0 iken saat sinyalinin yükselen kenarında yazılır sonraki düşen kenarında okunur, CPHA değeri 1 iken ise düşen kenarında yazılır, sonraki yükselen kenarında yazılır.
- **SPI Modu 0 (CPOL=0, CPHA=0)**
  CPOL = 0: Saat sinyali, düşük seviyede başlar.
  CPHA = 0: Veri değişikliği saat sinyalinin yükselen kenarında olur.
- **SPI Modu 1 (CPOL=0, CPHA=1)**
  CPOL = 0: Saat sinyali, düşük seviyede başlar.
  CPHA = 1: Veri değişikliği saat sinyalinin düşen kenarında olur.
- **SPI Modu 2 (CPOL=1, CPHA=0)**
  CPOL = 1: Saat sinyali, yüksek seviyede başlar.
  CPHA = 0: Veri değişikliği saat sinyalinin yükselen kenarında olur.
- **SPI Modu 3 (CPOL=1, CPHA=1)**
  CPOL = 1: Saat sinyali, yüksek seviyede başlar.
  CPHA = 1: Veri değişikliği saat sinyalinin düşen kenarında olur.

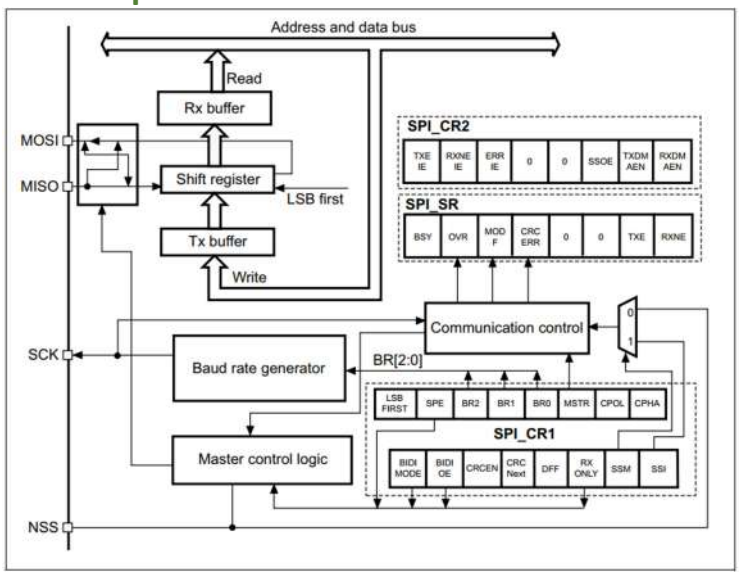








Birim Yapısı



Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	*LSBFIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2	Reserved																								TXEIE	RXNEIE	ERRIE	FRF	Reserved	SSOE	TXDMAEN	RXDMAEN
	Reset value																									0	0	0	0		0	0	0
0x08	SPI_SR	Reserved																							FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																								0	0	0	0	0	0	0	1	0
0x0C	SPI_DR	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CRCPR	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPI_RXCRCR	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TXCRCR	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **SPI_CR1** ve **SPI_CR2** (Control Register 1 ve 2), iletişimin modunu (Master veya Slave) ve saat hızını, Data frame format, Half duplex iletişim ve diğer özellikleri yapılandırmak için kullanılır.
- **SPI_SR** (Status Register), SPI haberleşme durumunu izlemek için kullanılır. İletimin tamamlanıp tamamlanmadığı, veri alımı durumu gibi bilgileri içerir.
- **SPI_DR** (Data Register), Veri gönderip almak için kullanılır. Gönderilen veya alınan veriyi bu kayıt aracılığıyla işleyebilirsiniz.
- **SPI_CRCPR** (CRC Polynomial Register) ve **SPI_RXCRCR/SPI_TXCRCR** (CRC Receive/Transmit Register), SPI verilerinin döngüsel hata denetimi (CRC) için kullanılır.

Haberleşme Metotları

- SPI üzerinden Polling, Interrupt ve DMA olmak üzere üç farklı haberleşme yapılabilir.
- https://deepbluembedded.com/stm32-spi-tutorial/ linkinden konu hakkındaki bilgileri inceleyebiliriz.

10 I2C

5 Mayıs 2021 Çarşamba 08:03

10 I2C

Giriş

- https://www.ercankoclar.com/2018/01/i2c-iletisim-protokolu-ve-mikroc-kutuphanesi/ ve https://ozdenercin.com/2019/01/25/i2c-seri-haberlesme-protokolu/ linklerinden ayrıntılı bilgilere ulaşabiliriz.
- https://www.ti.com/lit/an/slva704/slva704.pdf linkten Texas Instruments'in I2C protokü hakkında yazdığı makaleyi okuyabiliriz.
- I2C prokolünün geliştirlme amacı, düşük hızlı çevre birimlerinin ana kartları, cep telefonları, gömülü sistemler gibi elektronik cihazlara daha az kablo ihtiyacı ile bağlanabilmesini sağlamaktadır.

Avantajları ve Dezavantajları

Avantajlar;

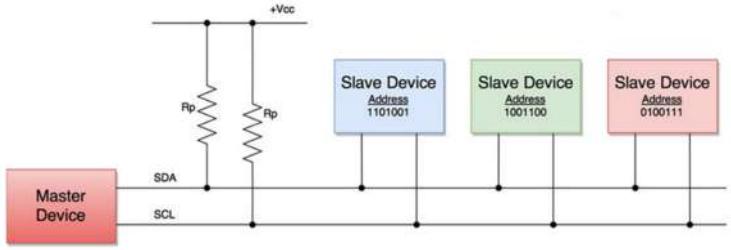
- Sadece iki telli bir yapıya sahiptir (SCL ve SDA), bu nedenle **az pin** kullanımı ile birçok cihazın bağlanmasını sağlar.
- Aynı hat üzerinden **çift yönlü iletişim** sağlar. Hem master hem de slave cihazlar veri gönderebilir ve alabilir.
- Bir dizi cihazın (EEPROM, sensörler, ekranlar vb.) bağlanmasını destekler, bu da çok **çeşitli uygulamalara** olanak tanır.
- Master ve slave cihazlar arasındaki bağlantıları daha **esnek** hale getirir. Çoklu master bağlantılarına izin verir.
- **Open-drain çıkışı**, güç tüketimini azaltır ve daha güvenilir bir iletişim sağlar.
- Y**üksek veri iletim hızlarına** izin verecek şekilde tasarlanmıştır.

Dezavantajlar;

- I2C'nin **uzun hat mesafelerinde** performansı düşük olabilir. Bu durum, iletişim hızını düşürmek veya ek güçlendirme önlemleri almak gerektirebilir.
- Birden çok masterın bulunduğu sistemlerde **çatallanma** sorunları ortaya çıkabilir. Bu durum, çakışmaları önlemek için dikkatlice senkronize edilmiş bir sistem gerektirir.
- Başlangıçta **karmaşık** olabilir ve doğru yapılandırma ve senkronizasyon gerektirebilir.
- Önceden belirlenmiş bir adres yapısı kullanır, bu nedenle **güvenlik** açısından zayıf olabilir.
- Uzun hat mesafelerinde veya yüksek hızlarda iletişimde, **elektromanyetik girişim** sorunları ortaya çıkabilir.

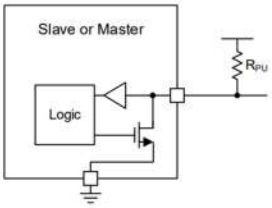
Bağlantılar

- I2C iletişiminde sadece iki hat vardır. Bunlar SDA (Serial Data Line) ve SCL (Serial Clock Line) hatlarıdır.
- Genellikle +5V ve +3.3V voltajlarda çalışmakla beraber, I2C protokolü daha pratik voltaj seviyelerine de izin vermektedir.
- SDA hattı harberleşmeyi başlatıp, sonlandırır. SCL ise veri hattı kondigurasyonunu sağlar.



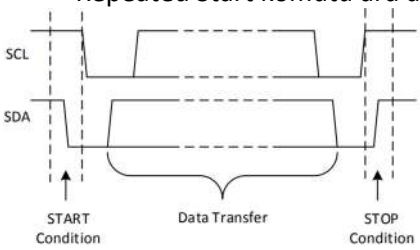
Çift Yönlü Haberleşme

- I2C, aynı hat üzerinde open-drain/open-collector ile bir giriş buffer kullanır, bu da tek bir veri hattının çift yönlü veri akışı için kullanılmasına olanak tanır.
- Bu hatlar ayrıca pull-up direncine ihtiyaç duyarlar.
- Yalnızca bir cihaz veri yolu hattını toprağa çekebilir veya veri yolu hattını serbest bırakır ve böylece pull-up direncinin voltajı yükseltmesine izin verir.
- Hattı LOW seviyeye çekmek için FET transistör **tetiklenir** böylece hat toprak ile kısa devre olur.
- Hattı HIGH seviyeye çekmek için FET transistör **kapatılır** böylece hat pull-up direnci vasıtasıyla voltaj seviyesine çekilir.

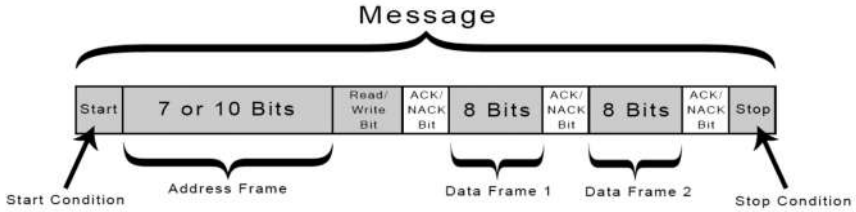


Veri İletimi

- I2C hattı SDA hattının lojik high seviyesinden lojik low seviyeye düşmesi ile başlar. Aynı şekilde lojik low seviyesinden lojik high seviyeye çıkması ile sonlanır. SDA hattının haberleşmeyi başlatabilmesi için SCL hattı da high olmalıdır.
- SCL hattı lojk high seviyesinde iken SDA hattı high seviyesine çekilirse haberleşme sonlanır.
- I2C veri gönderiminde start biti "0" stop biti "1" verilerinden oluşmaktadır.
- I2C veriyolu multimaster bir yapıdadır. Bu sayede letişim hattında birden fazla cihaz olabilir. Master cihazlarda bir saat sinyali ve data gönderildiği anda diğer cihazların tamamı slave moduna geçerler.
- Multimaster I2C haberleşmesinde Repeated Start komutu vardır ve sıklıkla kullanılır. I2C haberleşmesinde 2 adet cihaz olduğunu varsayalım.
  Birinci master cihaz start komutu gönderdi ve start komutundan sonra gerekli adres bilgilerini gönderdi. Tüm bu işlemler sürecinde I2C hattı birinci master cihaz tarafından kullanıldığından dolayı I2C hattı idle durumda olmayacaktır. Birinci cihaz stop durumu göndermeden önce haberleşmede bir değişiklik yapmak isterse Repeated Start komutunu gönderir ve böylece 1.Master cihazın slave cihaz ile I2C haberleşmesi kopmamış olur. Multimaster olmayan durumlarda Repeated Start komutunu kullanmaya gerek yoktur. Repeated Start komutu ard arda gelen start stop komutlarından oluşur.

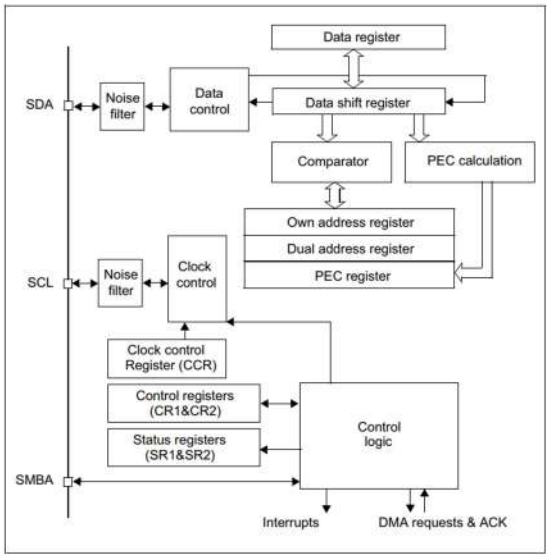


- İlk olarak SDA ve SCL hatları HIGH konumdadırlar. Daha sonra SDA hattı master tarafından **LOW** seviyeye çekilerek **iletişimin başlayacağı**, slave cihazlara bildirilir.
- Bu bildirimi alan slave cihazlar, adres bilgisini beklemeye başlarlar.
  Adres bilgisi slave cihazların yapısına göre 7 bit, 10 bit veya 16 bit olabilirler.
- Master cihaz hangi slave cihaz ile haberleşmek istiyorsa onun adres bilgisini gönderdikten sonra, **okuma** mi yoksa **yazma** mı yapacağını belirtir. Adres hangi slave cihazın ise o cihaz master ile iletişim kurmaya başlar.
- Adres kendisine ait olan slave cihaz, master cihaza verinin gönderildiği veya verinin alındığını doğrulamak için bir **ACK** (Acknowledge) kabul biti gönderir.
- Veri transferi işlemi gerçekleşir. Bu transfer iki yönlü de olabilir. (Slave'den Master'a veya Master'dan Slave'e)



- I2C haberleşmesinde 1 master cihaz ve birden fazla slave cihaz olduğunu varsayalım. Master cihaz herhangi bir slave cihaza erişmek için start komutundan sonra ilgili slave cihazın adresini gönderir. Aynı hatta bağlı olan slave cihazların tamamı bu mesajları alır ancak sadece bu mesaja sahip olan slave cihaz Ack mesajını göndererek iletişimin kurulduğu master cihaza bildirir ve Ack mesajını alan master cihaz adres bilgisinden hemen sonra veri göndermeye başlar.

Birim Yapısı



Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	I2C_CR2	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	I2C_OAR1	Reserved																ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	I2C_DR	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	I2C_SR1	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAUL	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	I2C_CCR	Reserved																F/S	DUTY	Reserved	CCR[11:0]												
	Reset value																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	I2C_TRISE	Reserved																											TRISE[5:0]			
	Reset value																												0	0	0	1	0
0x24	I2C_FLTR	Reserved																											ANOFF	DNF[3:0]			
	Reset value																												0	0	0	0	0

- **I2C_CR1** (Control Register 1), Ana kontrol registerıdır. I2C Peripherals'ı etkinleştirir veya devre dışı bırakır. Gönderme tamamlandığında kesme (interrupt) etkinleştirir. Acknowledge kontrol biti ile Yazılım sıfırlama biti bulunmaktadır.
- **I2C_CR2** (Control Register 2), ikinci kontrol registerıdır. Saat frekansını belirler.
- **I2C_OAR1** (Own Address Register 1), I2C'nin kendi adresini ayarlamak için kullanılır.
- **I2C_DR** (Data Register), veri göndermek veya almak için kullanılır.
- **I2C_SR1** ve **I2C_SR2** (Status Register 1 ve 2**)**, I2C'nin durumu hakkında bilgi sağlar. Birçok farklı durumu içerir, örneğin, START biti durumu, adres gönderme durumu, veri alım durumu vb.
- **I2C_CCR** (Clock Control Register), I2C saat frekansını kontrol eder.
- **I2C_TRISE** (Rise Time Register), yükselme süresini ayarlamak için kullanılır.
- **I2C_FLTR** (Filter Register), I2C hatlarında gürültüyü azaltmaya yönelik bir filtreleme mekanizması sağlar.

Haberleşme Metotları

- SPI üzerinden Polling, Interrupt ve DMA olmak üzere üç farklı haberleşme yapılabilir.
- https://deepbluembedded.com/stm32-i2c-tutorial-hal-examples-slave-dma/ linkinden konu hakkındaki bilgileri inceleyebiliriz.

11 USB

15 Nisan 2023 Cumartesi 14:12

11 USB

Giriş

- https://www.st.com/resource/en/user_manual/dm00108129-stm32cube-usb-device-library-stmicroelectronics.pdf link üzerinden ST'nin STM32Cube USB Device Library hakkında yayınladığı içeriği okuyabiliriz. USB Device Library tüm STM32 mikrodenetleyicileri için geneldir, yalnızca HAL katmanı her STM32 aygıtına uyarlanmıştır.
- USB (Universal Serial Bus), bilgisayarlar ve diğer elektronik cihazlar arasında veri aktarımı, güç sağlama ve diğer işlevleri gerçekleştirmek için kullanılan bir seri haberleşme ve bağlantı standardıdır.
- USB protokolü, donanım ve yazılımın bir araya gelerek veri alışverişini, cihaz tanıma ve yönetimini, güç dağıtımını ve diğer USB özelliklerini sağlar.
- USB protokolü, bir USB bağlantısı üzerinden çalışan cihazlar arasındaki iletişimi düzenler. Bu protokol, USB ana cihazı (**host**) ve USB aygıtları (**devices**) arasındaki etkileşimi sağlar. USB ana cihazı, genellikle bir bilgisayar veya USB bağlantı noktasına sahip başka bir cihazdır. USB aygıtları ise klavyeler, fareler, yazıcılar, depolama cihazları, kamera ve diğer birçok elektronik cihazı içerir.
- USB protokolü, veri iletimini, veri paketleme ve paketlerin gönderilmesi, hata kontrolü, cihaz tanıma ve iletişim sırasında diğer işlevleri düzenler.

Veri Hızı

- USB protokolü, veri aktarım hızına göre sınıflandırılmış çeşitli USB 1.1, USB 2.0, USB 3.0, USB 3.1, USB 3.2 versiyonları vardır.
    - USB 1.1,
        - Low-Speed: 1.5 Mbps
        - Full-Speed: 12 Mbps
        - İlk yaygın olarak kullanılan USB standardıdır. Klavye, fare gibi cihazlar için yeterlidir.
    - USB 2.0
        - Hi-Speed: 480 Mbps
        - Önceki versiyona göre önemli bir hız artışı sağlar. Çoğu USB cihazı (flash bellek, harici sabit diskler) bu standardı kullanır.
    - USB 3.0
        - SuperSpeed: 5 Gbps
        - Önceki nesillere göre çok daha hızlıdır ve veri iletimini büyük ölçüde hızlandırır. Ayrıca, daha fazla güç sağlayabilir.
    - USB 3.1
        - SuperSpeed+: 10 Gbps
        - Veri hızını iki katına çıkararak daha hızlı veri transferi sağlar. USB Type-C konektörlerinin tanıtımı da bu dönemde yapılmıştır.
    - USB 3.2
        - SuperSpeed+: 10 Gbps, 20 Gbps (çift hat)
        - USB 3.1 ile aynı hızı sunar ancak bazı konfigürasyonlarda 20 Gbps'ye kadar veri transferine olanak tanır. USB Type-C, bu versiyonla daha yaygın hale gelmiştir.

      USB4
        - 40 Gbps'ye kadar veri transfer hızı
        - Thunderbolt 3 ile uyumludur ve çok daha yüksek veri hızları sunar. USB4, USB Type-C konektörünü kullanır ve önceki versiyonlarla geriye dönük uyumludur.

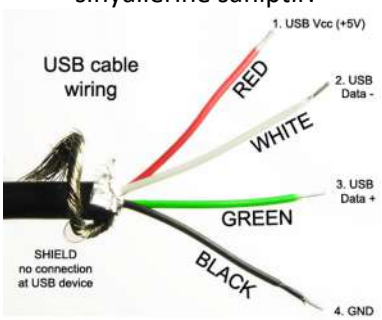
Bağlantı Türü

- USB protokolü, farklı bağlantı türlerine (USB-A, USB-B, USB-C) uygun şekilde uyarlanmıştır.
    - USB-A
        - En yaygın kullanılan USB konektör tipidir ve genellikle bilgisayarlar, şarj cihazları, klavyeler ve

fareler gibi cihazlarda bulunur.
    - USB 1.1, USB 2.0, USB 3.0 ve USB 3.1 versiyonlarıyla uyumludur.
    - USB 3.0 ve üstü için mavi renkli bir iç kısım, USB 2.0 için ise genellikle beyaz veya siyah renkli iç kısım ile ayırt edilir.
- USB-B
    - Daha çok yazıcılar, tarayıcılar ve bazı harici sabit diskler gibi büyük çevre birimlerinde kullanılan bir konektör tipidir.
    - Standart USB-B, USB 1.1 ve USB 2.0 için kullanılır.
      USB 3.0/3.1 Micro-B, Daha hızlı veri transferi için ekstra pimlere sahip genişletilmiş bir versiyondur.
    - USB 3.0/3.1 sürümlerinde, genellikle mavi renkli iç kısımlar ve ek pinler bulunur.
- Mini-USB
    - Daha küçük cihazlar için geliştirilmiş bir bağlantı türüdür. Eski dijital kameralar, bazı taşınabilir sabit diskler ve eski mobil cihazlarda bulunur.
    - Genellikle USB 2.0 ile uyumludur.
    - Artık yaygın olarak kullanılmıyor ve yerini Micro-USB ve USB-C'ye bıraktı.
- Micro-USB
    - Özellikle mobil cihazlar, taşınabilir sabit diskler ve bazı küçük elektronik cihazlar için yaygın olarak kullanılır.
    - USB 2.0 Micro-B, yaygın olarak Android telefonlar ve tabletler gibi cihazlarda kullanılır.
      USB 3.0/3.1 Micro-B, ekstra veri hatlarına sahip, daha hızlı veri transferi için genişletilmiş bir versiyon.
    - İnce ve küçük yapısıyla mobil cihazlarda yaygın olarak tercih edilir. USB 3.0 sürümü ek pimlere sahiptir.
- USB-C
    - Yeni nesil USB konektör tipidir ve simetrik yapısıyla hem yukarı hem de aşağı doğru takılabilir. Çok yönlülüğü sayesinde, hem veri aktarımı hem de şarj işlemleri için kullanılır.
    - USB 2.0'dan USB4'e kadar tüm versiyonlarla uyumludur.
    - Yüksek veri aktarım hızları sunar (USB 3.1, USB 3.2 ve USB4 ile 40 Gbps'ye kadar).
      Güç aktarımında 100W'a kadar destek sağlar (USB Power Delivery standardı ile).
      Çift yönlü veri ve güç transferi mümkündür.
      Aynı kablo üzerinden video sinyali aktarımı yapılabilir (Alt Mode özelliği ile)

Sinyaller

- USB protokolü, verileri D + ve D- veri sinyal hatları aracılığıyla ana bilgisayar ve harici çevresel cihazlar arasında seri olarak gönderir ve alır. USB, iki veri hattının yanı sıra , cihazı güçlendirmek için VCC ve GND sinyallerine sahiptir.



- USB protokolü aynı zamanda enerji yönetimi, cihaz tanıma ve konfigürasyon, veri transfer modları (örneğin, kontrol, veri, kesme, yüksek hızlı, vb.), bağlantı kesme ve diğer USB özelliklerini içeren çeşitli protokol

katmanlarından oluşur.

- USB protokolünün önemli özelliklerinden bazıları şunlardır: Yüksek hızlı veri transferi sağlar. Çoklu cihaz bağlantısına izin verir. Tak-çalıştır özelliği ile cihazların anında tanınmasını sağlar. Otomatik güç yönetimi ve şarj desteği sunar. Evrensel olarak desteklenir ve birçok cihaz ve işletim sistemi tarafından uyumlu bir şekilde kullanılabilir.
- USB protokolü, milyarlarca cihazda kullanılan ve yaygın olarak benimsenen bir standarttır. USB, bilgisayarlar, mobil cihazlar, oyun konsolları, yazıcılar, depolama aygıtları, ağ cihazları ve daha pek çok elektronik cihazda yaygın olarak kullanılan bir bağlantı standardıdır.

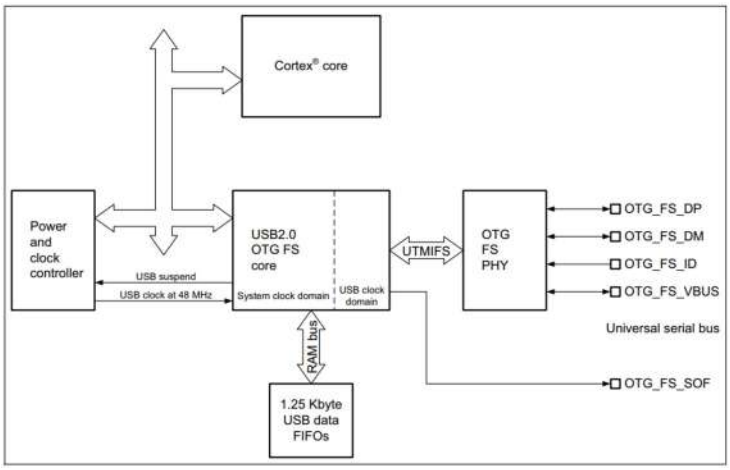
Modüller

- STM32Cube USB Middleware, STM32 mikrodenetleyicilerinde USB iletişimi için genel bir ara katmandır. Bu kütüphane, farklı USB profillerini destekler ve USB iletişimi için gerekli protokol katmanlarını ve işlevleri sağlar.
- STM32Cube USB Middleware; CDC (Communication Device Class), HID (Human Interface Device), Custom HID, MSC (Mass Storage Class), DFU (Device Firmware Upgrade), USB Audio ve diğer birçok USB profiline sahip olabilir.
  Bu kütüphane, USB iletişimi için genel bir çerçeve sağlar ve projenize özelleştirmeler yapmanıza olanak tanır.
  - **USB CDC (Communication Device Class)**, seri veri iletişimi için bir USB profili olarak kullanılır. STM32 mikrodenetleyicileri, USB CDC kütüphanesi aracılığıyla bilgisayara seri bir bağlantı noktası olarak görünebilir ve UART gibi geleneksel seri iletişim verilerini USB üzerinden iletebilir. Bilgisayara sanal bir seri port gibi görünür. Seri veri gönderme ve alma fonksiyonlarını sağlar ve USB üzerinden seri haberleşme protokollerini destekler.
  - **USB HID (Human Interface Device)**, klavye, fare, joystick gibi insan arayüz cihazlarını temsil eden bir USB profilidir.
    STM32 mikrodenetleyicilerinde USB HID kullanarak klavye ve fare verilerini alıp gönderebilirsiniz. HID raporlarını oluşturmanızı, USB HID sınıfını uygulamanızı ve kullanıcı arayüzü cihazlarını etkileşimli olarak çalıştırmanızı sağlar.
    Custom HID, USB HID sınıfının özelleştirilmiş bir versiyonudur. Özel ihtiyaçlara göre farklı cihazların veri alışverişini sağlar.
    STM32 mikrodenetleyicileri, Custom HID sınıfını kullanarak kullanıcı tanımlı HID raporları oluşturabilir ve bu raporlar üzerinden veri iletebilir.
  - **USB MSC (Mass Storage Class)**, bir USB profili olarak kullanılan bir USB kütüphanesidir ve USB üzerinden dosya sistemi verilerinin depolanmasını ve iletilmesini sağlar.
    STM32 mikrodenetleyicileri, USB MSC kütüphanesi aracılığıyla harici bir bellek cihazı gibi davranabilir ve dosyaları bilgisayara aktarabilir. USB üzerinden veri depolama ve aktarma işlevlerini sağlar.
  - **USB DFU (Device Firmware Upgrade)**, aygıt yazılımının güncellenmesi için kullanılan bir sınıftır. DFU aygıt sınıfını destekleyen bir mikrodenetleyici, özel bir DFU protokolü aracılığıyla yazılım güncellemelerini alabilir. Bu, yeni bir firmware sürümünü USB üzerinden yüklemek için kullanışlıdır ve mikrodenetleyicinin firmware güncellemelerini kolaylaştırır.
  - **USB Audio,** ses verilerini işlemek ve iletmek için kullanılan bir USB sınıfıdır.
    STM32 mikrodenetleyicileri, bu sınıfı kullanarak ses akışı sağlayabilir ve ses cihazlarıyla USB üzerinden etkileşimde bulunabilir.
- Bu kütüphaneler ve ilgili fonksiyonlar, STM32 mikrodenetleyicileri ile USB kullanarak farklı veri iletişim senaryolarını gerçekleştirmenizi sağlar. Projenizin gereksinimlerine ve kullanmak istediğiniz USB profiline bağlı olarak, ilgili kütüphaneleri projenize dahil ederek ve ilgili fonksiyonları kullanarak veri alışverişini yapabilirsiniz.

Birimler

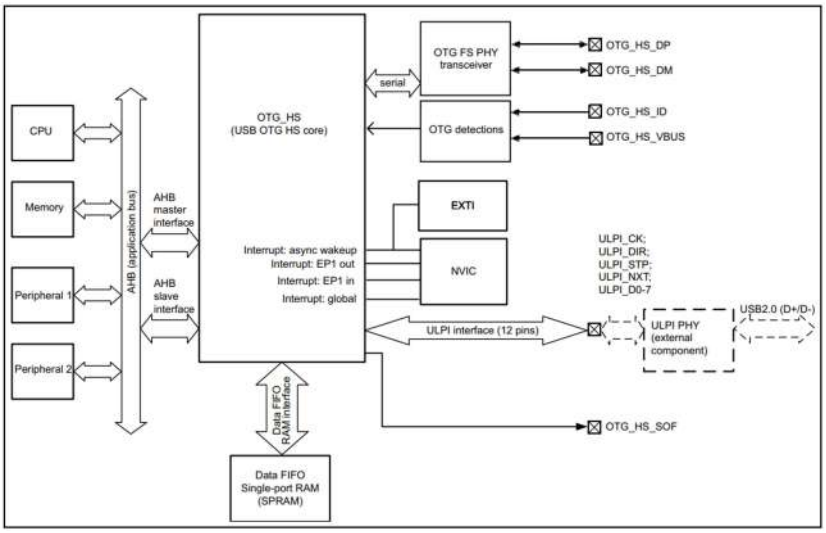
USB Full-Speed Device (USB-FS)

- STM32 mikrodenetleyicilerinde, özellikle düşük ve orta seviye modellerde bulunan bir USB cihaz birimidir.
- Hem USB cihazı (Device) hem de ana bilgisayar (Host) modlarında çalışabilir.
  - Full-Speed USB 2.0 desteği (12 Mbps).
  - USB cihazı olarak çalışmak üzere tasarlanmıştır.
  - Bu birim, genellikle USB cihazı modunda çalışan uygulamalar (örneğin, klavye, fare, USB bellek) için kullanılır.



USB High-Speed Device (USB-HS)

- STM32'nin üst düzey mikrodenetleyicilerinde bulunan bir USB birimidir.
  - High-Speed USB 2.0 desteği (480 Mbps).
  - USB cihazı olarak çalışır ve yüksek hızlı veri aktarımı gerektiren uygulamalarda kullanılır.



12 CAN

14 Ocak 2023 Cumartesi 17:55

12 CAN

Giriş

- Konu hakkında detaylı yazılmış yazılara http://www.barissamanci.net/Makale/15/can-bus-nedir-can-protokolu-incelemesi/, https://en.wikipedia.org/wiki/CAN_bus ve https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial linklerden ulaşabilirsiniz. Ayrıca https://youtu.be/SQgUDyu3dZw?si=pMJeZfnZgAG9leOS linkteki videoyu da izleyebilirsiniz.

Nedir

- Açılımı **Control Area Network Bus** yani Kontrol Alan Ağı Veri Yolu'dur.
- 1980'li yıllarda Robert Bosch tarafından **otomotiv** de kablo yumağı yerine bir kablodan yazılım kontrollü veri transferini sağlamak amacıyla geliştirilmiştir.
- 1987 ortaları itibari ile de Intel ve Philips gibi yarıiletken üreticileri CAN çiplerini piyasaya sunmaya başladılar. Böylelikle başta otomobiller olmak üzere her türlü taşıtlardan sanayi ürünlerine kadar birçok alanda CAN en yaygın kullanılan **veri yolu protokolü** haline geldi.
- **Güvenliğin** çok önemli olduğu **gerçek zamanlı** uygulamalarda kullanılır. Öyle ki istatistiksel olasılık hesapları sonucunda bir asırda bir tane tespit edilemeyen mesaj hatası yapabileceği hesaplanmıştır.
- Uygulama alanı yüksek hızlı ağlardan düşük maliyetli çoklu kablolamalı sistemlere kadar geniştir.
- CAN BUS otomobil elektroniği, akıllı motor kontrolü, robot kontrolü, akıllı sensörler, asansörler, makine kontrol birimleri, kaymayı engelleyici sistemler, trafik sinyalizasyon sistemleri, akıllı binalar ve laboratuvar otomasyonu gibi uygulama alanlarında maksimum 1Mbit/sn'lik bir haber veri iletişimi sağlar.
- İletişim hızı **40 m'de 1Mbit/sn** iken 1 km uzaklıklarda 40Kbit/sn'ye düşmektedir.
- CAN diğer protokollerden farklı olarak **adres temelli değil mesaj temelli** çalışmaktadır.
  Her mesaja özgü bir **ID numarası** vardır. Mesajlar **çerçeveler** ile iletilirler.

Katmanlar

- CAN BUS sistemleri genellikle **Nesne Katmanı** (Object Layer) , **İletim Katmanı** (Transfer Layer), **Fiziksel Katman** (Physical Layer) olmak üzere üç ana katmana ayrılır
- Nesne Katmanının görevi,
  - Hangi mesajın transfer edileceğini tespit etmek,
  - İletim katmanında hangi mesajın alınacağına karar vermek,
  - Donanımla ilgili uygulamaya arayüz sağlamaktır.
- İletim Katmanının görevi,
  - İletim katmanının başlıca görevi transfer protokolüdür.
    Örneğin; frame kontrolü, mesaj önceliği belirleme, hata kontrolü, hata sinyalleşmesi ve hata kapatma.
  - İletim katmanı yeni bir mesajı yollamadan önce iletim hattının boş olmasına dikkat eder. Aynı zamanda iletim hattından veri alınmasından da sorumludur.
  - Ayrıca senkron iletişim için veri transferi sırasında bit zamanlamasının bazı parametrelerini göz önünde bulundurur.
  - CAN BUS üzerinden haberleşen tüm sistem bileşenlerine ünite (nod) denir.
- Fiziksel Katmanın görevi,
  - Fiziksel katman, üniteler arasında veri haberleşmesi sırasındaki tüm elektriksel kısımdır.

Genel Özellikler

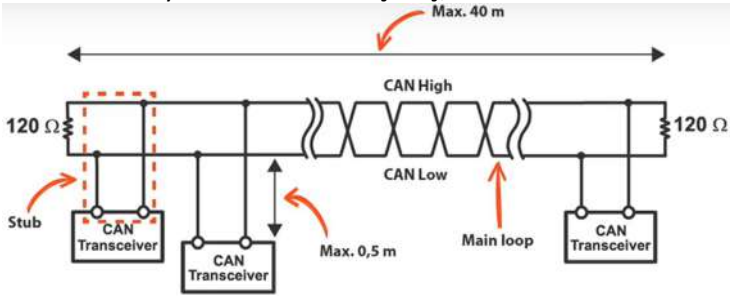
- Mesaj Önceliği
- Kayıp Zaman Güvenliği
- Yapılandırma Esnekliği
- Senkronizasyonlu çoklu kabul: Aynı veri birçok ünite tarafından alınabilir.
- Sistemdeki veri yoğunluğunu kaldırabilme
- Çok efendili (Multi master) çalışma
- Hata tespiti ve hataya ilişkin sinyalleri üretme
- Mesaj yollanmasında hata oluşması halinde mesajın iletim hattının boş olduğu bir anda mesajın otomatik olarak tekrar yollanması
- Ünitelerde oluşan geçici ve kalıcı hataları ayırt edebilme ve özerk olarak kalıcı hatalı üniteleri kapatabilme

Çalışma Mantığı

- **Haberleşme Yapısı,**
  - Standart olarak **half-duplex** haberleşme yapısını kullanır.
  - Aynı anda yalnızca bir cihaz veri gönderebilir; diğer cihazlar bu sırada gönderim yapamaz, yalnızca dinleyebilir.
- **Çakışma Önleme Mekanizması,**
  - Çakışmaları önlemek için **CSMA/CR (Carrier Sense Multiple Access with Collision Resolution)** mekanizmasını kullanır:
    - Bir cihaz veri göndermek istediğinde, önce iletişim hattının boş (IDLE) olup olmadığını kontrol eder.
    - Hattın boş olduğunu algılarsa veri göndermeye başlar.
    - Eğer başka bir cihaz aynı anda veri göndermeye başlarsa, çakışma çözümü için **çerçeve önceliği** devreye girer.
      - CAN protokolünde daha düşük ID'ye sahip olan mesajın önceliği daha yüksektir ve bu mesaj gönderilmeye devam eder.
- **Çatışma Durumu ve Çözüm,**
  - Birden fazla cihaz aynı anda veri göndermeye çalışırsa çakışma oluşabilir.
  - Çakışma durumunda:
    - Çakışmaya dahil olan cihazlar, iletim ortamını dinleyerek çerçeve önceliğine göre bir çözüm uygular.
    - Düşük öncelikli mesaj gönderen cihazlar hattı serbest bırakır ve hattın boşalmasını bekler.
    - Yüksek öncelikli mesaj gönderilmeye devam eder.
- **Mesaj Önceliklendirme,**
  - Mesaj öncelikli bir sistemdir.
    - İnternet protokollerinde cihazlara numara atanırken, CAN protokolünde mesajlara numara atanır.
    - Bu numaralar, mesajların öncelik sırasını belirler.
  - Daha düşük ID'li mesajlar, daha yüksek önceliğe sahiptir ve iletim hattına önce gönderilir.
- **Multi-Master Tasarımı,**
  - CAN-BUS, **multi-master** yapıya sahiptir.
    - Tüm cihazlar, eşit öncelikli olarak veri gönderme hakkına sahiptir.
    - Ancak önceliklendirme mekanizması sayesinde çakışmalar çözülür.
- **Veri Alımı ve Filtreleme,**
  - CAN-BUS hattına gönderilen tüm veriler, tüm cihazlar tarafından alınır.
    - Her cihaz, içerisindeki filtreleme mekanizması ile yalnızca kendisini ilgilendiren mesajları işler.
    - İlgisiz mesajlar, filtreleme sistemi tarafından göz ardı edilir.
- **Hız ve Mesafe Sınırları,**
  - Veri iletim hızı ve hattın uzunluğu arasında bir ilişki vardır:
    - **1 Mbit/sn** hızında iletişim için maksimum hat uzunluğu **40 metre** ile sınırlıdır.
    - **40 Kbit/sn** hızında iletişim için maksimum hat uzunluğu **1 kilometreye** kadar çıkabilir.

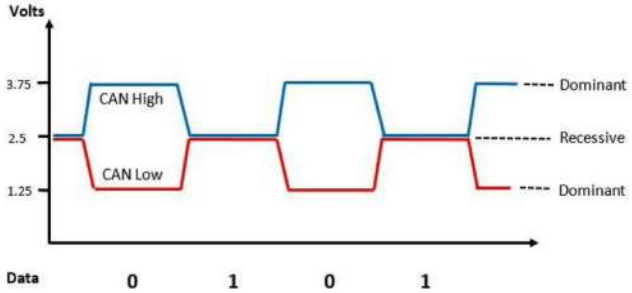
Veri Yolu ve Bağlantı Noktaları

- Genel olarak bir CAN bağlantı noktası aşağıda gösterilmiştir. Bağlantı noktaları **düğüm** olarak isimlendirilirler. Mikrodenetleyici ve CAN kontrolcüsünden oluşmaktadırlar.
- Bizim sistemlerimizde bu CAN kontrolcüsü çipe dahil edilmiştir fakat harici entegre olarakta kullanılabilir.
- CAN kontrolcüsü CAN veri yoluna direk bağlanır.
- Bu veri yolu **iki telden** oluşmuş, iki tarafı **120 Ohm** dirençlerle sonlandırılmıştır.



- CAN yapısında alıcı ve verici birbirinden fiziksel olarak **bağımsızdır**. Fakat düğümlerin yapısı gereği gönderilen mesaj alıcıdan dinlenebilmektedir. Bu sayede veriyi gönderen işlemci gönderdiği veri ile okuduğu

veriyi karşılaştırarak **hata** ve mesajların **öncelik** seviyelerine göre iletilmelerine olanak sağlanır. Multi master yapıda çalışabilen bu sistem için bu özellik önemlidir.

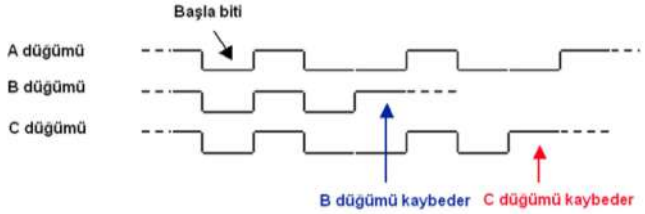


- CAN IDLE modda ise yani veri iletişimin aktif olmadığı durumda CANH ve CANL hatları 2.5V seviyesindedir. Data transfer edilmeye başladığı anda CANH hattı 3.75V seviyesine yükselirken, CANL ise 1.25V seviyelerine kadar düşmektedir. Bu durumda her iki sinyal seviyesi arasında 2.5V görülmektedir. Buradaki iletilişim iki veri yolu hattındaki voltaj farkına dayanır.

	dominant	**recessive**
dominant	dominant	dominant
recessive	dominant	recessive

	0	**1**
0	0	0
1	0	1

- Yukarıdaki şekilde görüldüğü gibi hattın lojik seviyesi **iki farklı değer** alabilmektedir ve bu değerlerin seviyesi gözükmektedir.
- Lojik 1, recessive (çekingen); lojik 0 dominant (baskın) olarak adlandırılmaktadır.
  Bunun sebebi hatta farklı düğümlerden aynı anda 0 ve 1 yazılması durumunda **0'ın 1 karşı baskın** gelmesidir.
- Lojik 0'ın lojik 1'e baskın gelmesi sonucu **küçük mesaj ID** sine sahip mesajlar **öncelik** kazanırlar.
- Bir düğüm tarafından mesaj gönderilmesi kararlaştırıldığında mesaj yol boşalana kadar bekletilir.
- Her düğüm yolu devamlı izlemektedir.
- Yol boşaldıktan sonra düğüm yola başla işaretini vererek mesajı yollamaya başlar.
- Mesaj her düğüme ulaşmaktadır ve ilişkisi olan düğümler mesajı okuyup işlemektedirler.
- Eğer yol boşaldığında birden fazla düğüm yola mesaj yazmaya başlarsa **düşük ID'li** mesajı yazan düğüm yolu ele geçirir ve diğer düğümler aradan çekilerek tekrar göndermek üzere yolun boşalmasını beklerler.
- Bu mekanizma şu şekilde çalışır. Yazılan her bitin aynı anda okunduğundan bahsetmiştik.
  Bir düğüm veri yoluna mesaj yazarken 1 yazdığında 0 okuyorsa eğer, başka bir düğümünde yola mesaj yazdığını anlar ve onun önceliği yüksek olduğundan veri yolunu ona bırakır. Yol boşaldığında tekrar göndermeye çalışır.
  - Örneğin yola aynı anda veri yazmaya çalışan A, B ve C adında 3 düğümümüz olsun. A düğümü yola 36 (100100), B düğümü 47(101111) ve C düğümü 37(100101) yazsın. Aşağıdaki şekilde bu durum gösterilmiştir.

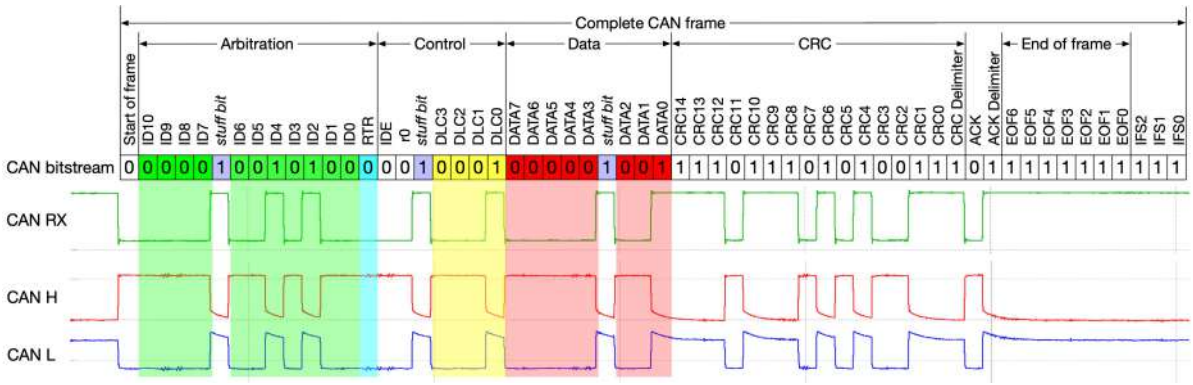


  - A düğümü mesajını gönderdikten sonra daha önemli bir mesaj yoksa C düğümü sonrada B düğümü mesajını gönderir.

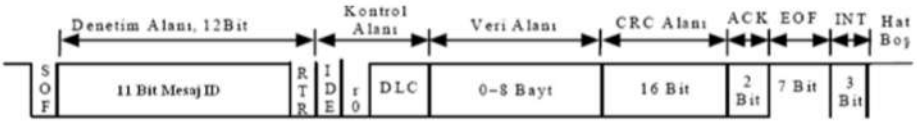
Mesajlar

- CAN sistemlerinde veriler paketlerin halinde iletilir. Ancak burada iki tip paketleme yapılır ve bu paketlemelerin özel adları vardır.
  11 bit tanımlayıcıya sahip olanlar **CAN2.0A** diğer adıyla **Standart** CAN, **29 bit** tanımlayıcıya sahip olanlara ise **CAN2.0B** diğer adıyla **Extended** CAN denir. Aralarındaki temel fark ise tanımlanacak mesaj sayısıdır.
  - Standart CAN'de 2^11 = 2048 mesaj tanımlanır.
  - Extended CAN'de 2^29 = 536 870 912 mesaj tanımlanır.
- Bu bilgilerin tutulduğu alana **Mesaj ID** alanı denir. **Mesaj önceliğini tanımlamada** buradaki sayı dikkate alınır.
- Veri yolunda mesajlar çerçevelere bölünerek iletilmektedir. İki farklı mesaj türü vardır.
  - Bunlar **Veri Çerçevesi (Message Frame)** ve **İstek Çerçevesi (Remote Transmit Request Frame)**'dir.
  - Farkları ise veri çerçevelerinin en fazla 8 byte uzunluğa kadar veri taşıyabilmesi, istek çerçevesinde ise belli bir mesaja ait verinin istenmesidir.

Frame



Base Frame



Denetim Alanı

- Her çerçeve **SOF (Start Of Frame)** sinyali ile başlar. Bu sinyal 1 bitlik dominanttır.
- Ardından 12 bitlik **Denetim Alanı** gelmektedir.
    - 11 bit **Mesaj ID** alanıdır ve bu alandaki ID değeri ile mesajlar etiketlenir. ID alanındaki ilk 7 bit ardışıl olamaz.
    - Denetim alanındaki son bit **RTR** diye adlandırılır ve özel anlamı vardır. Bu bit dominant ise gönderilen çerçeve **veri çerçevesi**dir ve veri alanında, ID alanında tanımlanan mesaja ait veri vardır demektir. Eğer bu bit 1 ise çerçeve, **İstek Çerçevesi'**dir ve veri alanı yoktur.
- Bu çerçevenin ID alanındaki değer ile belirlenen mesaja ait veri ilgili bilgi, düğümlerden istenmektedir.
- Bu çerçeveyi alan, ID alanındaki değeri okuyarak hangi veriyi göndermesi gerektiğini anlar ve yol boşa çıktığında gönderir. Bu sayede CAN protokolü master ve slave olarak çalışabilmektedir.

Kontrol Alanı

- Denetim alanından sonra **Kontrol Alanı** gelmektedir.
- Bu alanın ilk biti **IDE** diye isimlendirilir ve bu çerçevenin 2.0A çerçevesi olduğunu belirten dominant bir bittir.
- Bu bittin ardından bir bitlik kullanılmayan **rezerve alan** gelmektedir.
- Daha sonra 4 bitlik **DLC** diye isimlendirilen bir alan gelir. DLC alanı gönderilen verinin **kaç byte** olduğunu söyler.

Veri Alanı

- Kontrol alanını veri alanı takip etmektedir.
- Veri alanı en fazla 8 byte olabilmektedir.

CRC Alanı

- Veri alanını CRC alanı takip eder.
- Bu alan 16 bitliktir ve 15 bitlik **CRC** (Cyclic Redundancy Check) bilgisi ile resesif CRC Delimiter bitinden oluşmaktadır.
- CRC alanı gönderilen SOF alanından CRC alanına kadar gönderilen verinin doğru olup olmadığının anlaşılması için bir değerdir.
- Veriyi gönderen düğüm veri üzerinde bir takip işlemler yaparak 15 bitlik CRC değerini hesaplar ve çerçeveye ekler.
- Alan düğüm veriyi aldığında göndericinin yaptığı işlemleri ile aynı işlemleri yapar ve CRC'yi tekrar hesaplar.
- Alınan ve gönderilen CRC tutarlı ise veri doğru gönderilmiştir.
- Alıcı düğümlerden en az 1 tanesi bile veriyi yanlış aldıysa veri tekrar gönderilmelidir.

ACK Alanı

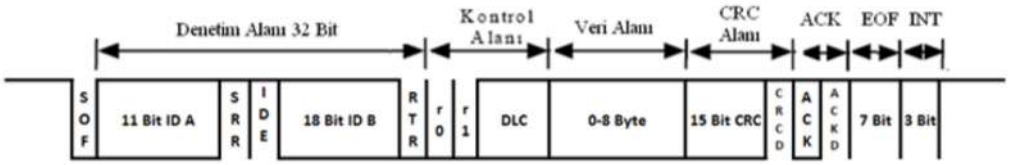
- CRC alanını ACK alanı takip eder. Bu alan 2 bitliktir.
- İlk bitini gönderici resesif olarak gönderir. Eğer veri en az bir alıcı tarafından doğru alınmışsa alıcı yola dominant bit yazar. Böylece gönderici mesajın en az bir alıcı tarafından alındığını anlar. Eğer gönderici dominant biti okuyamasa ACK işaretinden kaynaklı bir hata olduğuna kanaat getirir ve veriyi tekrar yollar.
- Bu alanın ikinci biti ise ACK delimeter olarak adlandırılır ve resesiftir.
  Bunun bir mesaj iletimiyle açıklamalı. Gönderici başla biti ile iletim hattında şuan gönderici benim der. Ardından Mesaj ID Alanı, Veri Alanı, CRC Alanı gönderilir. Alındı Bilgisi Alanında ise iletim ortamı

resesif(çekinik) tutulur. Eğer tüm ünitelerden biri, mesaj onu ilgilendirse ya da ilgilendirmese dahi, mesajı alabiliyorsa iletim ortamını dominant(baskın) yapar ve böylece gönderici en az bir ünite veriyi alabildiği için bitir bitini yollayıp iletim ortamını diğer ünitelerin kullanımına bırakır. Yani Alındı Bilgisi Alanında "Aldınız mı?" sorusuna yanıt beklenir. Eğer Alındı Bilgisi sürecinde herhangi bir üniteden alındığına dair bilgi alınmazsa ACK hatasından kaynaklı hata oluştuğunu belirten Hata Çerçevesi üretilir ve gönderici tekrar yollanmaya çalışır.
Eğer gönderen İstek Çerçevesi yollamışsa, alıcı da iletim hattının boş bir anında cevabını göndericiye yollar.

EOF Alanı

- ACK Alanının arkasından çerçevenin sonlandırıldığını belirten 7 bitlik **EOF** (End of Frame) alanı gelir.
  Bu alandan bitler resesiftir.
- Daha sonra ise çerçeveler arasında boşluk bırakmak amacıyla 3 bitlik INT alanı gelmektedir ve bitleri resesiftir. Böylece temel çerçevede bir mesaj gönderilmiş ve alınmış olur.

Extended Frame



- CAN2.0B'de mesaj ID si 29 bittir.
- Geriye uyumluluk açısından CAN2.0B geliştirilirken 2.0A göz önünde bulundurularak geliştirilmiştir.
- İki protokolde aynı yolda çalışabilmektedir fakat 2.0A düğümlerine 29 bitlik ID'li mesajlar gönderilmemelidir.

Denetim Alanı

- Extended Frame'de dominant **SOF** ile başlar.
- Ardından 2.0A'da olduğu gibi 11 bitlik **IDA** alanı gelir.
- Ardından 2.0A'daki dominant RTR biti yerine resesif **SRR** biti gelmektedir.
- Ardından 2.0A'daki ofsete denk gelecek şekilde **IDE** biti gelir.
  Tek farkı 2.0B'de bu bit resesiftir çünkü 29 bitlik ID ye sahip mesajlar iletilmektedir.
- IDE bitinden sonra 18 bitlik ikinci **ID B** alanı gelir.
- Ardından dominant **RTR** biti gelerek bu çerçevenin veri çerçevesi olduğunu belirtir.

Kontrol Alanı

- Denetim alanından sonra kontrol alanı gelmektedir.
- Kontrol alanının ilk iki biti **rezerv alandır** ve kullanılmaz.
- Son dört bit **DLC** alanını oluşturur ve gönderilen verinin kaç byte olduğunu söyler.

Request Frame

- Çoğu zaman veri yolu okunan, oluşan bilgilerin gönderilmesi ile çalışmaktadır.
- Bazen düğümler istekte bulunurlar.
- Bunu istek çerçevesi ile yaparlar.
- İstek çerçevesi ile veri çerçevesinin 2 farkı vardır.
- Bunlar istek çerçevelerinde RTR biti resesiftir ve istek çerçevelerinin veri alanı yoktur.
- Kalan kısımlar veri çerçevesi ile aynıdır.

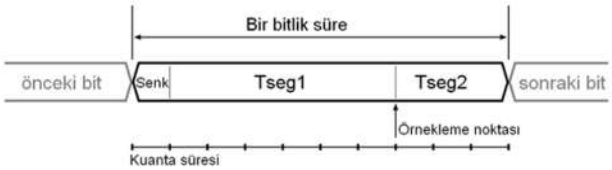
Error Frame

- Veri yolunda hata oluştuğunda hata çerçevesi gönderilmektedir.
- Hata çerçevesi iki alandan oluşmaktadır.
- Bu alanlar hata bayrakları (Error Flags) ve hata ayıracı (Error Delimiter) alanıdır.
- Aktif ve pasif olmak üzere iki adet hata bayrağı vardır.
- Yolun durumuna göre aktif veya pasif hata oluşturulmaktadır.
- CAN haberleşmesinde harici clock yoktur.
- Senkronizasyon lojik değişimlere göre yapılmaktadır.
- Altı veya daha fazla adet aynı lojik seviyeden bitin ardışıl okunması, senkronizasyonun kaybolduğu anlamına gelip alıcıda hata oluşturmaktadır.
- Bu yüzden aynı lojik seviyede beş ardışıl bit gönderildikten sonra araya karşı seviyeden bir bit sıkıştırılır.
- Bunu CAN donanımı yapmaktadır ve bu işlemin adına bit stuffing denilmektedir.

Bit Time

- Çoğu diğer seri protokolün aksine, CAN protokolünde bit hızı direkt olarak baud rate önbölücüsünü kurarak ayarlanmaz.

- CAN donanımlarında baud rate önbölücüsü vardır fakat **kuanta** denilen küçük bir zaman dilimini üretmek için kullanılır.
- Bir bitlik süre **3 kısma** bölünmüştür.
    - Birinci kısım **senkronizasyon** kısmıdır ve **sabit olarak 1 kuanta** uzunluğundadır.
    - Takip eden kısımlar ise **Tseg1** ve **Tseg2** olarak isimlendirilir ve kullanıcı tarafından uzunlukları kuanta cinsinden ayarlanabilir.
- Bir bitlik periyot minimum **8** maksimum **25** kuanta uzunluğunda olmalıdır.



- Gönderilen bitin alıcıda alındığı nokta **örnekleme noktası** diye isimlendirilir ve Tseg1 sonundadır.
- Tseg1 ve Tseg2 oranı ayarlanarak örnekleme noktası bir bitlik zaman içerisinde kaydırılabilir.
  Bunu yapmamızdaki amaç iletişim hattının uzunluğuna göre sistemin kararlı çalışabilmesini sağlamaktır.
    - Uzun iletim hatları kullanıyorsak örnekleme noktası geri çekilmelidir.
        - Örnekleme noktası neden ileri alınır?
            - Osilatör hassasiyeti düşük ise örnekleme noktası ileri kaydırılır.
            - Bus üzerindeki bit hızındaki ufak sapmaları telafi eder.
            - Ek olarak alıcı bit zamanlamalarını ayarlayarak vericiye kilitlenebilirler.
- Her bit, kullanıcı tarafından ayarlanabilen **synchronous jump width** denilen 1 ile 4 kuanta süresi arasında değer alan bir değişken tarafından ayarlanabilir.
- Bit hızı aşağıdaki bağıntı ile hesaplanır.

  $$\text{Baud Rate} = P_{CLK} / (\text{Prescaler} * (1 + Tseg1 + Tseg2))$$
    - Örneğin, Baud Rate 125kHz (125000 bit/s), PCLK'yi 60MHz ve örnekleme noktasını %70 olarak kullandığımızı varsayalım. Bir bitlik periyot toplam kuanta sayısı ile hesaplanır ve bu değer (1 + Tseg1 + Tseg2)'dir. Bu değere **KUANTA** diyelim ve yukarıdaki bağıntıyı tekrar düzenleyelim.

      $$\text{Prescaler} = PCLK / (\text{Bit Rate} * KUANTA)$$
    - Bilinen değerlerimizi denkleme yerine koyalım.

      $$\text{Prescaler} = 60M / (125k * KUANTA)$$
    - Bir bitlik periyodun 8 ila 25 kuanta arasında olduğunu biliyoruz.
      Bu bilgiyi kullanarak Prescaler tam sayı olacak şekilde KUANTA yerine 8 ile 25 arasında uygun bir sayı seçelim.
      KUANTA = 16, Prescaler = 30 olacak şekilde denklemi çözeriz.
    - Şimdi Tseg1 ile Tseg2 arasındaki oranı ayarlayalım.

      $$16 = (1 + Tseg1 + Tseg2)$$
    - Örnekleme noktası oranı periyodun %70'ine denk gelmesi için,

      $$\text{Örnekleme Noktası} = (KUANTA * 70) / 100$$

      Dolayısıyla 16 * 0.7 = 11.2 olur. Buradan Tseg1 = 10 için ve Tseg2 = 5 olarak bulunur. Bu durumda örnekleme noktası %68.8'e denk gelir.
    - Synchronous jump width değeri de aşağıdaki şekilde hesaplanır.
      Tseg2 >= 5 TKUANTA ise SJW = 4 tür.
      Tseg2 < 5 TKUANTA ise SJW = (Tseg2 -1) TKUANTA'dır. Bizim örneğimizde SJW = 4'tür.

Hata Önleme

- CAN protokolünün **beş adet** hata önleme yöntemi vardır.
- Herhangi bir hata oluştuğunda gönderici veriyi tekrar gönderir böylece işlemcinin olaya müdahale etmesine gerek kalmaz.
- Hata önleme yöntemlerinden 3 tanesi **çerçeve** düzeyinde, 2 tanesi ise **bit** düzeyindedir.
- Çerçeve düzeyinde olanlar **çerçeve formatı**, **CRC** ve **ACK** kontrolüdür.
- Bit düzeyinde olanlar ise **bit** ve **bit stuffing** kontrolüdür.

Frame Kontrolü

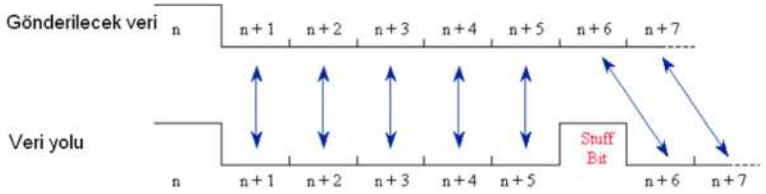
- Çerçeve formatı kontrolünde, alıcı veriyi aldıktan sonra verinin formatını kontrol eder ve çerçeve yapısı ile uyumlu olup olmadığını karşılaştırır. Alınan veride eksik alan varsa veri reddedilir ve veri yoluna hata çerçevesi bırakılır. Bu sistem doğru formatta veri alımını sağlar.
- CRC kontrolünde, SOF bitinden CRC bitlerinin başına kadar olan bitler bir takım işlemlerden geçirilerek CRC

kodu üretilir. Alıcıda bu CRC kodu alınan veri ile karşılaştırılır ve alınan bitlerin doğru olup olmadığını sınar. Format kontrolünden sonra alıcının bitleri kontrol etmesi ile formata uyan fakat hatalı mesajların önüne geçilmiş olur.

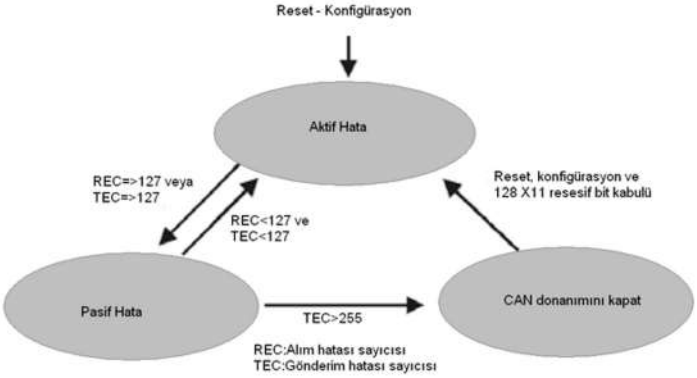
- ACK kontrolünde ACK mesajının göndericiye ulaşmamasıdır. ACK bitinin anlamı alınan mesajı alıcının onaylamasıdır. Gönderici CRC bitlerini gönderdikten sonra ACK bitini resesif olarak gönderir. Alıcılardan en az bir tanesinin hattaki resesif olan ACK bitini dominant bitle ezmesi beklenir. Eğer zaman aşımı sonucu ACK biti göndericiye ulaşmamışsa ACK bitinde hata olduğu şeklinde yorumlanır. Bunun sonucu göndericide hata oluşur ve ACK onayını alana kadar aynı mesajı tekrar gönderir.

Bit Kontrolü

- Bit stuffing hatasında, gönderici ile alıcı arasında saat darbeleri gönderilmez. Bunun yerine veri yolundaki CANL ve CANH hatlarındaki lojik değişimler ile senkronizasyon sağlanır. Bunun sonucu olarak aynı lojik seviyeden 5'ten fazla bitin art arda gelmesi senkronizasyonun bozulduğu anlamına gelir ve alıcıda hataya sebebiyet verir. Bunu engellemek için gönderici aynı 5 seviyeden sonra karşı seviyeden bir bit göndererek iletişime devam eder.
- Bunun sonucu olarak herhangi bir düğüm herhangi bir anda hata mesajı oluşturmak istediğinde veri yoluna 6 adet dominant bit yazar ve hataya sebebiyet verir.



- Bit düzeyindeki diğer hata ise bit kontrolüdür. Veri yolu boşaldığında düğümlerin mesaj göndermek için veri yolunu mesajların ID'lerine göre ele geçirdiğini söylemiştik. Her düğüm veri yoluna yazdığı biti tekrar geri okuyarak kendisinin gönderdiğinden daha önemli bir mesaj var mı diye bakar. Eğer daha önemli mesaj varsa geri çekilerek veri yolunun boşalmasını bekler. Böylece veri yolunu bir düğüm kazanmış olur ve bitleri göndermeye devam eder. Aynı zamanda da gönderdiği bitleri geri okur. Eğer veri yolunu kazanan düğüm gönderdiği seviyeden farklı bir seviye okursa hata oluşturur.
- CAN yapısının temelinde her mesajın farklı bir ID ile etiketlendiğini söylemiştik. Mesajları etiketlerken bu kurala dikkatle uymamız gerekmektedir. Yoksa farklı düğümlerin aynı ID ye ait farklı veriler göndermesi alıcıda CRC hatasına sebep olacaktır.



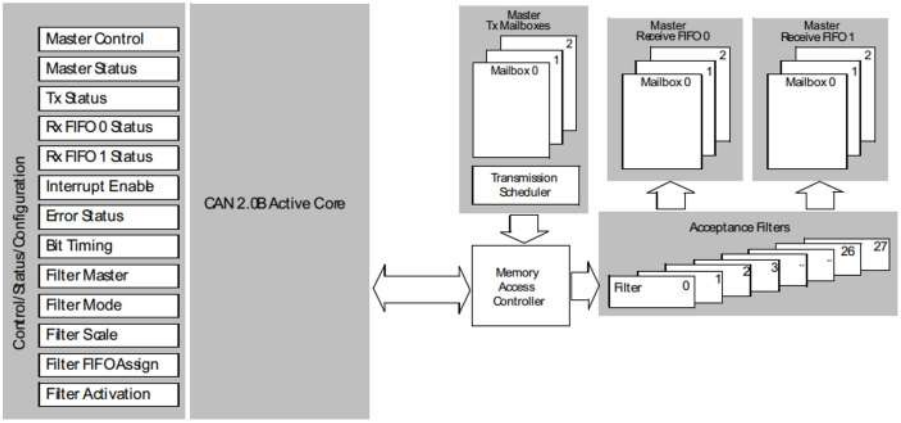
- CAN donanımı oluşan hatalara göre hata durumları arasında geçiş yapmaktadır. **İki adet hata sayıcısı** vardır.
- Bunlar göndericide oluşturulan hataları ve alınan hataları sayarlar.
- Herhangi bir sayıcı 127 ve büyük bir değere ulaşırsa donanım pasif hata moduna girer.
- Bu modda gelen hata çerçevelerini cevaplamaya devam eder fakat hata oluşturduğunda dominant bitler yerine resesif bitler gönderir.
- Eğer gönderim hata sayısı 255'i geçerse donanım kapatılır ve hattaki iletişime karışmaz ve etkilenmez.
- Haberleşmeyi tekrar başlatmak için işlemcinin olaya müdehale etmesi gerekmektedir.
- CAN donanımını resetler ve tekrar konfigüre eder. Bu mekanizmalar düğüm hatalı duruma düştüğünde art arda hata mesajları göndererek veri yolunu meşgul etmesini önlemek içindir.

Birim Yapısı

- STM32 serisi mikrodenetleyiciler gelen mesajları donanım seviyesinde işler ve yalnızca filtreyi geçen mesajlar posta kutusuna iletilir. Bu, işlemcideki yükü azaltmak ve ekstra kesme işlemine dikkat etmememizi sağlar.
- **bxCAN** adlı **Basic Extended CAN**, bir CAN ağ arabirimidir. CAN 2.0A ve 2.0B sürümlerini destekler. Çok sayıda gelen mesajı maksimum verimlilikle ve en az işlemci kullanımıyla yönetmek için tasarlanmıştır. Ayrıca,

mesajların transferi için öncelikli şartlardan da sorumludur.

- bxCAN modülü, CAN mesajlarının iletimini ve alımını tamamen otonom bir şekilde gerçekleştirir. Standart tanımlayıcılar (11 bit) ve genişletilmiş tanımlayıcılar (29 bit) donanım tarafından tamamen desteklenir.
- bxCAN'ın çalışma yöntemi aşağıda verilmiştir.

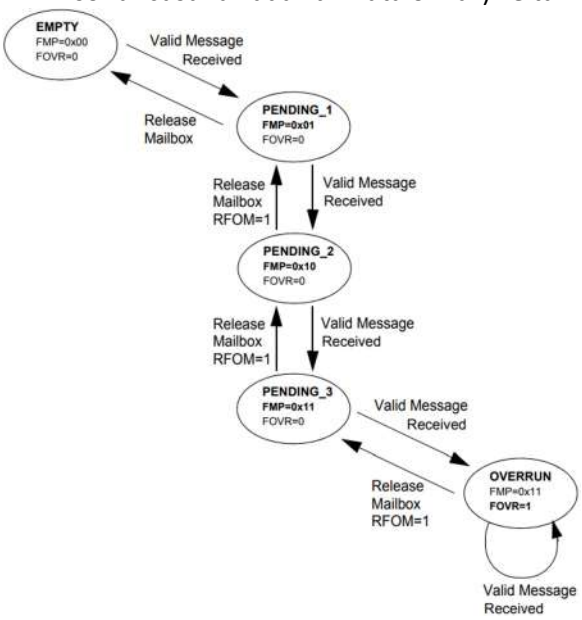


Mesaj Alımı

- CAN mesajlarının alımı için FIFO olarak düzenlenmiş üç posta kutusu sağlanmıştır.
- CPU yükünden tasarruf etmek, yazılımı basitleştirmek ve veri tutarlılığını garanti etmek için FIFO tamamen donanım tarafından yönetilir.
- Uygulama, FIFO'da depolanan mesajlara FIFO çıkış posta kutusu aracılığıyla erişilir.

Geçerli Mesaj

- Alınan bir mesaj, CAN protokolüne göre doğru bir şekilde alındığında geçerli sayılır (EOF alanının sonuncusuna kadar bir hata olmaz) ve tanımlayıcı filtrelemeden başarıyla geçer.



FIFO Yönetimi

- Boş durumdan başlayarak, alınan ilk geçerli mesaj FIFO'da PENDING_1 tarafından saklanır.
- Donanım, CAN_RFR kaydındaki FMP [1:0] bitlerinin 01b değerine ayarlanması olayını bildirir.
- Mesaj FIFO çıkış posta kutusunda bulunur. Yazılım, posta kutusu içeriğini okur ve CAN_RFR kayıt defterinde RFOM bitini ayarlayarak serbest bırakır.
- FIFO yeniden boşalır. Bu süre zarfında geçerli bir yeni mesaj alındıysa, FIFO PENDING_1 durumda kalır ve yeni mesaj, çıkış posta kutusunda bulunur.
- Uygulama posta kutusunu serbest bırakmazsa, bir sonraki geçerli mesaj PENDING_2 durumuna giren FIFO'da saklanır (FMP [1:0]= 10b)
- FIFO'yu PENDING_3 durumuna getiren bir sonraki geçerli mesaj için saklama işlemi tekrarlanır (FMP [1:0]= 11b).
- Bu noktada, yazılım RFOM bitini ayarlayarak çıkış posta kutusunu serbest bırakmalıdır, böylece bir posta kutusu bir sonraki geçerli mesajı saklamak için serbest kalır. Aksi halde, bir sonraki geçerli mesaj mesaj kaybına neden olacaktır.

Overrun

- FIFO, PENDING_3 durumuna geçtiğinde (yani üç posta kutusu doluysa), bir sonraki geçerli mesaj alımı aşılmaya neden olacak ve bir mesaj kaybolacaktır.
- Donanım, CAN_RFR kaydında FOVR bitin iayarlayarak aşırı çalışma durumunu bildirir.
- Hangi mesajın kaybolduğu FIFO'nun yapılandırmasına bağlıdır.
    - FIFO kilit işlevi devre dışı bırakılmışsa (CAN_MCR kaydındaki RFLM biti silindi), FIFO'da depolanan son mesaj yeni gelen mesajın üzerine yazılır. Bu durumda en son mesajlar her zaman uygulamaya açık olacaktır.
    - FIFO kilit işlevi etkinse (CAN_MCR kaydında RFLM biti ayarlanmışsa), en son mesaj atılır ve yazılım FIFO'daki en eski üç mesaja sahip olur.

Alımla İlgili Kesmeler

- Program yeni bir mesajın geldiğini nasıl biliyor?
- Mesaj FIFO'ya kaydedildikten sonra, FMP [1:0] bitleri güncellenir ve bir kesme isteği oluşturulur (CAN_IER kaydındaki FFIE biti ayarlanmışsa)
- Üç posta kutusunun tamamı doldurulduğunda, CAN_IER kaydındaki TAM bit ayarlanır.
- Bir aşırı yükleme sırasında FOVR bitinin ayarlandığına ve bir kesme işleminin yalnızca CAN_IER kaydının FOVIE biti ayarlanmışsa kesme üreteceğine dikkat edilmelidir. Aksi takdirde, bir aşırı yükleme sırasındaki tüm yeni mesajlar tamamen göz ardı edilir ve en az bir posta kutusu serbest bırakılıncaya kadar alımlarından dolayı kesinti olmaz.

Filtreleme

- Mesajların CAN veri yolundan nasıl işlendiğini öğrendik. Mesaj alma ve posta kutusuna gönderme koşullarından biri mesaj tanımlayıcısını bir filtreden geçirmektir.
- CAN protokolünde bir mesajın tanımlayıcısı bir düğümün adresi ile ilişkili değil, mesajın içeriği ile ilişkilidir.
- Sonuç olarak bir verici mesajını tüm alıcılara yayınlar.
- Mesaj alımında bir alıcı düğüm tanımlayıcı değerine bağlı olarak yazılımın mesaja ihtiyaç duyup duymadığına karar verir. Mesaj gerekiyorsa, SRAM'ye kopyalanır. Aksi takdirde, mesaj yazılıma müdahale edilmeden atılmalıdır.
- Bu gerekliliği yerine getirmek için, bxCAN Denetleyicisi, uygulamaya 28 yapılandırılabilir ve ölçeklenebilir filtre bankası sağlar.
- Bu donanım filtreleme, yazılım tarafından filtreleme yapmak için gerekli olacak CPU kaynaklarını korur.
- Her filtre bankası, iki adet 32-bit CAN_FxR0 ve CAN_FxR1 yazmaçlarından oluşur.
- Filtreleri uygulamaların ihtiyaçlarına göre optimize etmek ve uyarlamak için, her filtre bankası birbirinden bağımsız olarak ölçeklendirilebilir.

Ölçeklenebilir Genişlik

- Ölçek filtresine bağlı olarak, filtre bankası şunları sağlar:
    - STID[10:0], EXID[17:0], IDE ve RTR bitleri için bir 32 bit filtre
    - STID[10:0], RTR, IDE ve EXID[17:15] bitleri için iki adet 16 bit filtre.
- IDE biti (Tanıtıcı Genişletme Bit), filtrenin uzatılmış mesaj çerçevesi için tasarlandığını ve Uzaktan İletim İsteği (RTR) bitinin "Uzak" ileti türünü gösterdiğini anlamına gelir.
- Ek olarak, filtre **maske** modunda veya **listeleme** modunda yapılandırılabilir.

Maske Modu

- Maske modunda, tanımlayıcı kayıtları, tanımlayıcının hangi bitlerinin "eşleşmesi gerektiği" veya "umurumda değil" olarak ele alındığını belirten maske kayıtları ile ilişkilendirilir.
- Gelen mesajın kimliği (ID) ve maske değerleri (Mask ID) karşılaştırılır. Maske, hangi bitlerin kontrol edileceğini belirler.
    - Maske: 0xFFFF → Tüm bitler kontrol edilir.

Identifier Liste Modu

- Tanımlayıcı listesi modunda, maske kayıtları tanımlayıcı kayıtları olarak kullanılır.
- Böylece bir tanımlayıcı ve maske tanımlamak yerine, tek tanımlayıcı sayısını iki katına çıkaran iki tanımlayıcı belirtilir.
- Gelen tanıtıcının tüm bitleri, filtre kayıt defterinde belirtilen bitlerle eşleşmelidir.
- Gelen mesaj, filtre bankasında tanımlanan sabit bir listeyle karşılaştırılır. Sadece eşleşen mesajlar kabul edilir.
    - Liste: 0x123, 0x456 → Sadece bu iki mesaj kabul edilir.

Örnek Çalışma Modları

- CAN birimi, istenmeyen mesajları filtrelemek için kabul fitresi ve maske değerini kullanarak bu görevi yerine

getirmek için ürün yazılımı içerir.

- Filtre maskesi, alınan çerçevenin tanımlayıcısındaki hangi bitleri karşılaştıracağını belirlemek için kullanılır.
    - Bir maske biti sıfıra ayarlanmışsa (0x0000), gelen ID bit, filtre bitinden bağımsız olarak otomatik olarak alınır.
    - Bir maske biti bire ayarlanmışsa (0xFFFF), karşılık gelen ID biti, filtre biti ile karşılaştırılır. Eşleşme olursa kabul edilir, aksi taktirde çerçeve reddedilir.
- Örneğin yalnızca 00001234 kimliği içeren çerçeveleri almak istiyorsak maskeyi 1FFFFFFF olarak ayarlamalıyız.

Filtre	0x00001234
Maske	0x1FFFFFFF

    - Bir çerçeve geldiğinde ID'si filtre ile karşılaştırılır ve tüm bitlerin eşleşmesi gerekir (yani tüm bitler sırası ile tek tek karşılaştırılır). 00001234 kimliği ile eşleşmeyen herhangi bir çerçeve reddedilir.
- Örneğin 00001230 - 0000123F arasındaki ID 'li çerçeveleri almak istiyorsak maskeyi 1FFFFFF0 olarak ayarlamalıyız.

Filtre	0x00001230
Maske	0x1FFFFFFO

- Yalnızca 00001230 - 00001237 arasındaki çerçeveleri kabul etmek istiyorsak maskeyi 1FFFFFF8 olarak ayarlamalıyız.

Filtre	0x00001230
Maske	0x1FFFFFF8

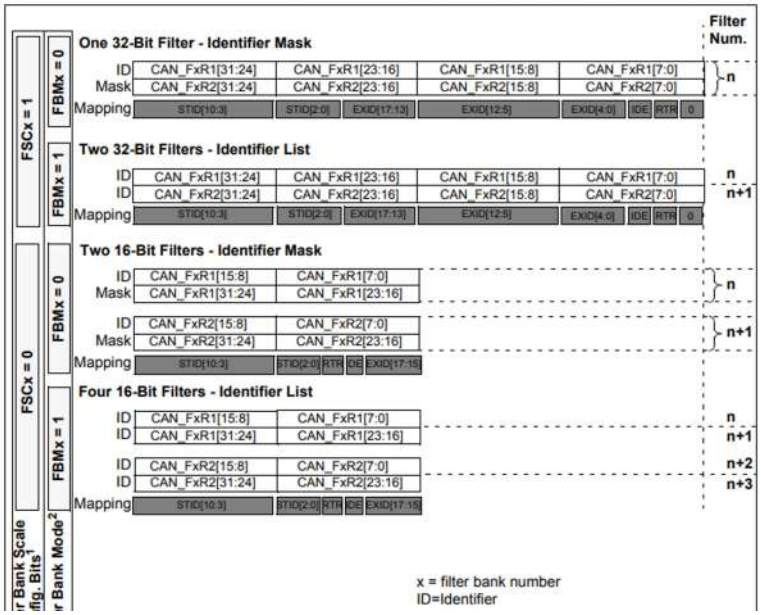
    - Bir çerçeve geldiğinde ID'si filtre ile karşılaştırılır ve 0.bit haricindeki tüm bitlerin eşleşmesi gerekir ve ayrıca 0.bit de 0 ile 8 arasında olmalıdır.
- Örneğin herhangi bir gelen çerçevenin kabul edilmesi isteniyorsa filtreyi ve maskeyi sıfıra ayarlamalıyız.

Filtre	0x0000/0
Maske	0x0000/0

    - İşlemin sonucunda tüm çerçeveler kabul edilir.

Filtre Bankası Ölçeği ve Mod Yapılandırması

- Filtre sıraları, karşılık gelen CAN_FMR kaydı vasıtasıyla konfigüre edilir.
- Bir filtre bankasını konfigüre etmek için, CAN_FAR kaydındaki FACT bitini temizleyerek etkisizleştirilmelidir.
- Filtre ölçeği, CAN_FS1R kaydındaki karşılık gelen FSCX biti vasıtasıyla yapılandırılmıştır.
- İlgili Maske/Tanımlayıcı kayıtları için tanımlayıcı listesi veya tanımlayıcı maske modu, CAN_FMR kaydındaki FBMx bitler vasıtasıyla yapılandırılmaktadır.
- Bir tanımlayıcı grubunu filtrelemek için, Maske/Tanımlayıcı kayıtlarını maske kipinde yapılandırın.
- Tek tanımlayıcıları seçmek için, Maske / Tanımlayıcı kayıtlarını tanımlayıcı liste modunda yapılandırın.
- Uygulama tarafından kullanılmayan filtreler devre dışı bırakılmalıdır.
- Bir filtre bankası içindeki her bir filtre, moda ve filtre sıralarının her birinin ölçeğine bağlı olarak O'dan maksimuma kadar numaralandırılır (Filtre Numarası olarak adlandırılır)



Filter Bank Scale Config. Bits[1]	Filter Bank Mode[1]

x = filter bank number
ID=Identifier
[1] These bits are located in the CAN_FS1R register
[2] These bits are located in the CAN_FM1R register

Filtre Index Eşleştirme

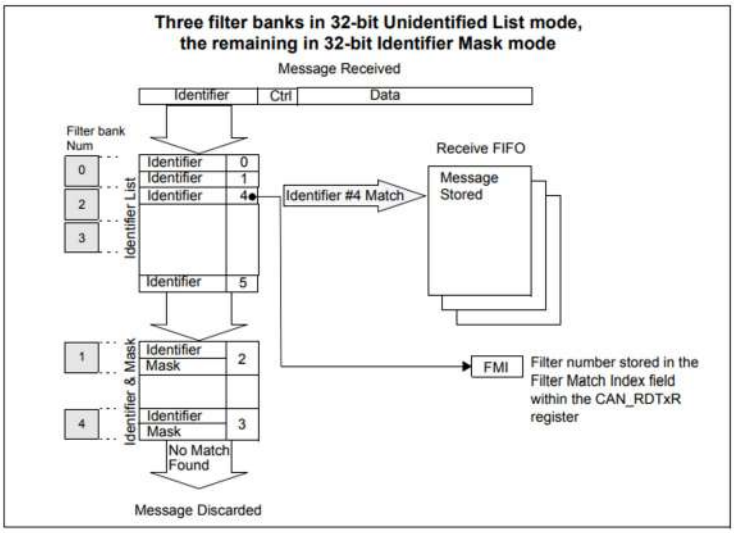
- FIFO'ya bir mesaj alındıktan sonra uygulamaya geçilebilir. Genellikle, uygulama verileri SRAM konumlarına kopyalanır.
- Verileri doğru yere kopyalamak için, uygulamanın verileri tanımlayıcı aracılığıyla tanımlaması gerekir. Bundan kaçınmak ve SRAM konumlarına erişimi kolaylaştırmak için CAN denetleyicisi bir Filtre Eşleme Dizini sağlar.
  • Bu indeks, filtre öncelik kurallarına göre mesajla birlikte posta kutusunda saklanır. Böylece, alınan her mesajın, ilişkili filtre eşleşme endeksi vardır.
- Filtre Eşleştirme dizini iki şekilde kullanılabilir:
  - Filtre Eşleme dizinini beklenen değerlerin bir listesiyle karşılaştırın.
  - Veri hedefi konumuna erişmek için bir dizideki dizin olarak Filtre Eşleştirme Dizinini kullanın.
- Maskelenmemiş filtreler için, yazılımın artık tanımlayıcıyı karşılaştırması gerekmez. Filtre maskelenmişse, yazılım yalnızca maskeli bitlerle karşılaştırmayı azaltır.
- Filtre numarasının endeks değeri, filtre sıralarının aktivasyon durumunu dikkate almaz.
- Ek olarak, her FIFO için bir tane olmak üzere iki bağımsız numaralandırma şeması kullanılır.

Filter Bank	FIFO0	Filter Num.
0	ID List (32-bit)	0, 1
1	ID Mask (32-bit)	2
3	ID List (16-bit)	3, 4, 5, 6
5	Deactivated ID List (32-bit)	7, 8
6	ID Mask (16-bit)	9, 10
9	ID List (32-bit)	11, 12
13	ID Mask (32-bit)	13

Filter Bank	FIFO1	Filter Num.
2	ID Mask (16-bit)	0, 1
4	ID List (32-bit)	2, 3
7	Deactivated ID Mask (16-bit)	4, 5
8	ID Mask (16-bit)	6, 7
10	Deactivated ID List (16-bit)	8, 9, 10, 11
11	ID List (32-bit)	12, 13
12	ID Mask (32-bit)	14

Filtre Öncelik Kuralları

- Filtre kombinasyonuna bağlı olarak bir tanımlayıcının birkaç filtreden başarılı bir şekilde geçtiği ortaya çıkabilir. Bu durumda, alıcı posta kutusunda depolanan filtre eşleşme değeri aşağıdaki öncelik kurallarına göre seçilir:
  - 32 bit filtre, 16 bit filtreye göre önceliklidir.
  - Eşit ölçekte filtreler için, Tanımlayıcı Maskesi moduna göre Tanımlayıcı Listesi moduna öncelik verilir.
  - Eşit ölçek ve moddaki filtreler için öncelik, filtre numarasından verilir (sayı ne kadar küçükse, öncelik o kadar yüksek olur).



- Bir mesajın alınmasında, tanımlayıcı ilk önce tanımlayıcı liste modunda konfigüre edilen filtrelerle karşılaştırılır.
- Bir eşleşme varsa, mesaj ilişkili FIFO'da saklanır ve eşleşen filtrenin indeksi Filtre Eşleştirme Dizininde

saklanır.

- Örnekte gösterildiği gibi, tanımlayıcı Tanımlayıcı # 2 ile eşleşir, böylece mesaj içeriği ve FMI 2 FIFO'da depolanır.
- Eşleşme yoksa, gelen tanımlayıcı daha sonra maske modunda yapılandırılan filtrelerle karşılaştırılır.
- Tanımlayıcı, filtrelerde yapılandırılan tanımlayıcılardan hiçbiriyle eşleşmezse, mesaj, yazılımı rahatsız etmeden donanım tarafından atılır.

Mesaj Saklama

- Yazılım ve CAN mesajlarının donanımı arasındaki arayüz posta kutuları vasıtasıyla gerçekleştirilir. Bir posta kutusu, bir mesajla ilgili tüm bilgileri içerir; tanımlayıcı, veri, kontrol, durum ve zaman damgası bilgisi.

Transmit Mailbox

- Yazılım, boş bir posta kutusundan iletilecek mesajı ayarlar. İletimin durumu, CAN_TSR kaydındaki donanım ile gösterilir.

Offset to transmit mailbox base address (bytes)	Register name
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

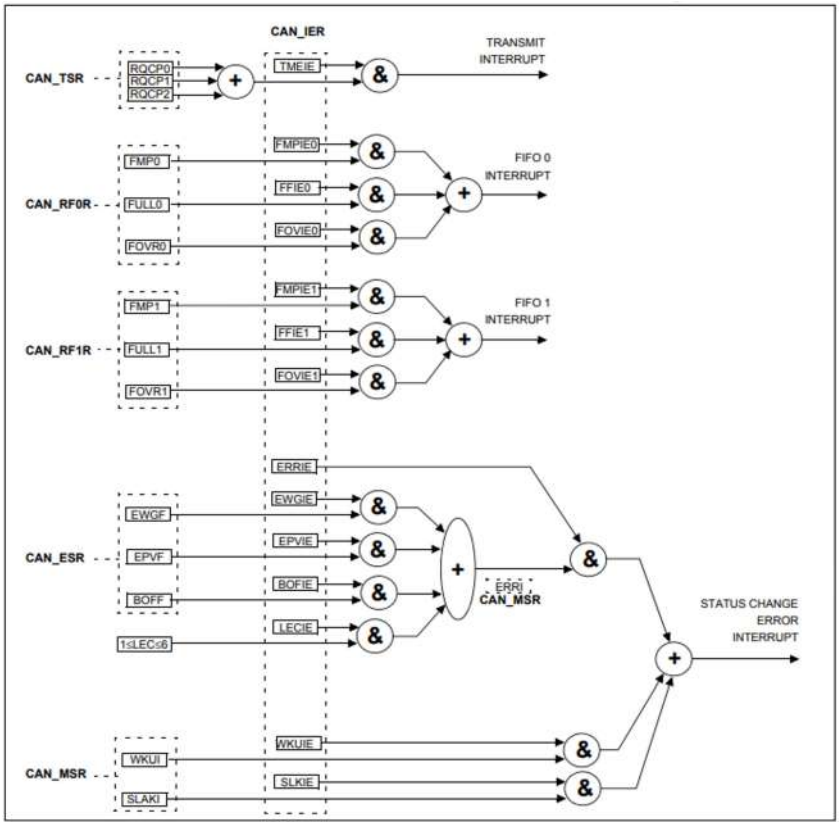
Receive Mailbox

- Bir mesaj alındığında, FIFO çıkış posta kutusundaki yazılım kullanılabilir.
- Yazılım mesajı ilettikten sonra (örneğin okuma) bir sonraki gelen mesajı mümkün kılmak için CAN_RFR kaydındaki RFOM biti aracılığıyla FIFO çıkış posta kutusunu serbest bırakmalıdır.
- Filtre eşleşme endeksi CAN_RDTXR yazmacının MFMI alanında saklanır.
- 16-bit zaman damgası değeri, CAN_RDTXR'nin TIME [15: 0] alanına kaydedilir.

Offset to receive mailbox base address (bytes)	Register name
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

CAN Kesmeleri

- Dört kesme vektörü bxCAN'a tahsis edilmiştir.
- Her kesme kaynağı, CAN Kesme Kesintisi Kaydı (CAN_IER) aracılığıyla bağımsız olarak etkinleştirilebilir veya devre dışı bırakılabilir.



- Aktarım kesintisi aşağıdaki olaylar tarafından oluşturulabilir:
  - Gönderme posta kutusu 0 boş, CAN_TSR kayıt setinde RQCPO bit.

- Gönderilen posta kutusu 1 boşalır, CAN_TSR kayıt kümesinde RQCP1 biti.
- Gönderme posta kutusu 2 boşalır, CAN_TSR kayıt kümesinde RQCP2 biti.

- FIFO 0 kesmesi aşağıdaki olaylar tarafından oluşturulabilir:
  - Yeni bir mesaj alımı, CAN_RFOR yazmacındaki FMPO bitleri '00' değildir.
  - FIFOO tam koşulu, CAN_RFOR kayıt kümesinde FULLO biti.
  - FIFOO aşırı çalışma koşulu, CAN_RFOR kayıt setinde FOVRO biti.
- FIFO 1 kesmesi aşağıdaki olaylar tarafından oluşturulabilir:
  - Yeni bir mesaj alımı, CAN_RF1R yazmacındaki FMP1 bitleri '00' değildir.
  - FIFO1 tam koşulu, CAN_RF1R kayıt kümesinde FULL1 biti.
  - FIFO1 aşırı çalışma koşulu, CAN_RF1R kayıt setinde FOVR1 biti.
- Hata ve durum değişikliği kesintisi aşağıdaki olaylar tarafından oluşturulabilir:
  - Hata durumu, hata koşulları hakkında daha fazla bilgi için CAN Error Status register'a (CAN_ESR) bakın.
  - Uyandırma durumu, SOF, CAN Rx sinyalinde izlenir.
  - Uyku moduna giriş.

Register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CAN_MCR	Reserved															DBF	RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	Reset value																1	0								0	0	0	0	0	0	0	0
0x004	CAN_MSR	Reserved																				RX	SAMP	RXM	TXM	Res.			SLAKI	WKUI	ERRI	SLAK	INAK
	Reset value																					1	1	0	0				0	0	0	1	0
0x008	CAN_TSR	LOW[2:0]			TME[2:0]			CODE[1:0]		ABRQ2	Res.			TERR2	ALST2	TXOK2	RQCP2	ABRQ1	Res.			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res..			TERR0	ALST0	TXOK0	RQCP0
	Reset value	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
0x00C	CAN_RF0R	Reserved																									RFOM0	FOVR0	FULL0	Reserved	FMP0[1:0]		
	Reset value																										0	0	0		0	0	
0x010	CAN_RF1R	Reserved																									RFOM1	FOVR1	FULL1	Reserved	FMP1[1:0]		
	Reset value																										0	0	0		0	0	
0x014	CAN_IER	Reserved															SLKIE	WKUIE	ERRIE	Res.		LECIE	BOFIE	EPVIE	EWGIE	Reserved	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	Reset value																0	0	0			0	0	0	0		0	0	0	0	0	0	0
0x018	CAN_ESR	REC[7:0]								TEC[7:0]									Reserved								LEC[2:0]			Reserved	BOFF	EPVF	EWGF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0	0		0	0	0
0x01C	CAN_BTR	SILM	LBKM	Reserved				SJW[1:0]		Reserved	TS2[2:0]			TS1[3:0]					Reserved							BRP[9:0]							
	Reset value	0	0					0	0		0	1	0	0	0	1	1									0	0	0	0	0	0	0	0
0x180	CAN_TI0R	STID[10:0]/EXID[28:18]												EXID[17:0]																	IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x184	CAN_TDT0R	TIME[15:0]																	Reserved						TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x188	CAN_TDL0R	DATA3[7:0]								DATA2[7:0]									DATA1[7:0]							DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x18C	CAN_TDH0R	DATA7[7:0]								DATA6[7:0]									DATA5[7:0]							DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1B0	CAN_RI0R	STID[10:0]/EXID[28:18]												EXID[17:0]																	IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1B4	CAN_RDT0R	TIME[15:0]																	FMI[7:0]							Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1B8	CAN_RDL0R	DATA3[7:0]								DATA2[7:0]									DATA1[7:0]							DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1BC	CAN_RDH0R	DATA7[7:0]								DATA6[7:0]									DATA5[7:0]							DATA4[7:0]							

0x1B8																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1BC	CAN_RDH0R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]					
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x200	CAN_FMR	Reserved																CAN2SB[5:0]						Reserved							FINIT
	Reset value																	0	0	1	1	1	0								1
0x204	CAN_FM1R	Reserved								FBM[27:0]																					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20C	CAN_FS1R	Reserved								FSC[27:0]																					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x214	CAN_FFA1R	Reserved								FFA[27:0]																					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x21C	CAN_FA1R	Reserved								FACT[27:0]																					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x240	CAN_F0R1	FB[31:0]																													
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . .	. . .	. . .																													
0x31C	CAN_F27R2	FB[31:0]																													
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

- **CAN Control and Status Registers**
  - **CAN_MCR** (Master Control Register), modülünün genel kontrolünü sağlar. Başlatma, uyku modu ve bus-off durumlarını yönetir.
    - **INRQ** (Initialization Request), modülünü başlatma moduna almak için kullanılır.
    - **SLEEP** (Sleep Mode Request), modülünü uyku moduna geçirmek için.
    - **TXFP** (Transmit FIFO Priority), Gönderim önceliğini ayarlar.
    - **ABOM** (Automatic Bus-Off Management), Otomatik bus-off yönetimini etkinleştirir.
  - **CAN_MSR** (Master Status Register), CAN modülünün durum bilgilerini verir. Modülün başlatma veya uyku modunda olup olmadığını ve hata durumlarını izler.
    - **INAK** (Initialization Acknowledge),Modülün başlatma modunda olduğunu belirtir.
    - **SLAK** (Sleep Acknowledge), Modülün uyku modunda olduğunu belirtir.
    - **ERRI** (Error Interrupt), Bir hata kesmesi meydana geldiğini gösterir.
    - **WKUI** (Wakeup Interrupt), Modülün uyandırıldığını belirtir.
  - **CAN_TSR** (Transmit Status Register), Mesaj gönderimiyle ilgili durumu raporlar.
    - **RQCPx** (Request Completed x), Her bir iletim kutusundaki (mailbox) iletimin tamamlandığını gösterir.
    - **TME** (Transmit Mailbox Empty), İletim kutularının boş olduğunu belirtir.
    - **ABRQx** (Abort Request x), İlgili iletim kutusundaki mesajın iptal edilmesini sağlar.
  - **CAN_RFxR** (Receive FIFO Register), Alım FIFO'larının durum bilgilerini ve kontrolünü sağlar. FIFO'daki mesaj sayısı, FIFO taşması ve FIFO mesaj boşaltma işlemlerini izler.
    - **FMPx** (FIFO Message Pending x), FIFO'da bekleyen mesaj sayısını gösterir.
    - **FULLx** (FIFO Full x), FIFO'nun dolu olup olmadığını belirtir.
    - **FOVRx** (FIFO Overrun x), FIFO taşması meydana geldiğinde set edilir.
    - **RFOMx** (Release FIFO Output Mailbox x), FIFO'dan bir mesajın serbest bırakılmasını sağlar.
  - **CAN_IER** (Interrupt Enable Register), CAN modülündeki kesme kaynaklarını etkinleştirir.
    - **TMEIE** (Transmit Mailbox Empty Interrupt Enable), İletim kutusu boş olduğunda kesmeyi etkinleştirir.
    - **FMPIEx** (FIFO Message Pending Interrupt Enable), FIFO'ya mesaj geldiğinde kesmeyi etkinleştirir.
    - **ERRIE** (Error Interrupt Enable), Hata kesmesini etkinleştirir.
  - **CAN_ESR** (Error Status Register), CAN modülündeki hata durumlarını izler. Hata uyarı bayrakları, bus-off durumu ve son hata kodu gibi bilgileri içerir.
    - **EWGF** (Error Warning Flag), Hata durumu uyarı bayrağı.
    - **BOFF** (Bus-Off Flag), Bus-off durumunu gösterir.
    - **LEC** (Last Error Code**)**, Son hata kodunu belirtir (bit hata, CRC hata vb.).
  - **CAN_BTR** (Bit Timing Register), CAN bit zamanlaması ve baud rate ayarlarını yapar. Yeniden senkronizasyon genişliği (SJW), zamanlama segmentleri (TS1 ve TS2) ve prescaler (BRP) bu registerda yer alır.

    - **CAN_BTR** (Bit Timing Register), CAN bit zamanlaması ve baud rate ayarlarını yapar. Yeniden senkronizasyon genişliği (SJW), zamanlama segmentleri (TS1 ve TS2) ve prescaler (BRP) bu registerda yer alır.
        - **BRP** (Baud Rate Prescaler), CAN hızını belirlemek için prescaler değeri.
        - **TS1** (Time Segment 1), Zamanlama segmenti 1.
        - **TS2** (Time Segment 2), Zamanlama segmenti 2.
        - **SJW** (Synchronization Jump Width), Yeniden senkronizasyon genişliği.
- **CAN Mailbox Register**: Posta kutuları, mesaj gönderimi ve alımı için kullanılan yapılar olup aşağıdaki registerlardan oluşur.
    - **CAN_TIxR ve CAN_RIxR** (Transmit/Receive Identifier Register)**,** gönderilecek/alınan mesajın kimlik bilgisini içerir.
        - **STID** (Standard Identifier), 11 bit Standard mesaj kimliğini içerir
        - **EXID** (Extended Identifier), 29 bit Extended mesaj kimliği için kullanılır
        - **IDE** (Identifier Extension), mesaj kimliği tipini seçer. Standard ya da Extended seçimi yapılır.
        - **RTR** (Remote Transmission Request), Mesaj tipini belirtir. Data frame ya da Remote frame seçimi yapılır.
        - **TXRQ** (Transmit mailbox request), mesajının gönderilmek üzere hazır olduğunu gösteren bir kontroldür. ilgili posta kutusu için iletimi talep etmek üzere yazılım tarafından ayarlanır. Posta kutusu boşaldığında donanım tarafından temizlenir
    - **CAN_TDTxR ve CAN_RDTxR** (Transmit/Receive Data Length Control and Time Stamp Register)**,** gönderilecek/alınan mesajın veri uzunluğunu ve isteğe bağlı zaman damgasını içerir
        - **DLC** (Data Length Code), Gönderilecek veri uzunluğunu belirtir (0-8 byte)
        - **TGT** (Transmit Global Time), zaman damgası kullanımı için ayar
        - **TIME** (Time Stamp), opsiyonel olarak zaman damgası değeri içerir.
    - **CAN_TDLxR, CAN_TDHxR, CAN_RDLxR ve CAN_RDHxR (**Transmit/Receive Data Low/High Register), gönderilecek/alınan mesajın düşük (ilk 4 byte) ve yüksek (son 4 byte) veri kısmını içerir.
        - **DATA**, Alınan mesajın veri içeriğini tutar
- **CAN Filter Registers:** Filtreler, CAN modülüne ulaşan mesajları filtrelemek için kullanılır. Aşağıdaki registerlar filtrelerin kontrolünü sağlar.
    - **CAN_FMR,** (Filter Master Register), filtrelerin genel kontrolünü sağlar.
        - **FINIT (Filter Initialization Mode)**: Filtrelerin yapılandırılmasını sağlamak için kullanılır
    - **CAN_FM1R** (Filter Mode Register), filtrelerin modunu ayarlar. Her bir filtre bankası için bir bit ayrılmıştır ve her bit için **Mask mode** veya **List mode** seçimi yapılır.
    - **CAN_FS1R** (Filter Scale Register), filtrelerin ölçeğini ayarlar. Her bir filtre bankası için bir bit ayrılmıştır ve her bit için **16 bit** veya **32 bit** seçimi yapılır.
    - **CAN_FFA1R** (Filter FIFO Assignment Register), filtrelerin hangi FIFO'ya yönlendirilmesi gerektiğini belirler. Her bir filtre bankası için bir bit ayrılmıştır ve her bit için **FIFO0** veya **FIFO1** seçimi yapılır.
    - **CAN_FA1R** (Filter Activation Register), filtrelerin etkin olup olmadığını kontrol eder. Her bir filtre bankası için bir bit ayrılmıştır ve her bit için **devre dışı** veya **etkin** seçimi yapılır.
    - **CAN_FiRx** (Filter Bank Registers), her filtre bankasının (CAN_F0R1, CAN_F0R2, …, CAN_F27R1, CAN_F27R2) **kimlik** ve **maske** bilgilerini içerir. İlgili bitlere bağlı olarak gelen mesajlar filtrelenir.