STM32 ile Gömülü Yazılım

5 Mayıs 2021 Çarşamba 07:50

Giriş

5 Mayıs 2021 Çarşamba 08:02

Giriş

Kaynaklar

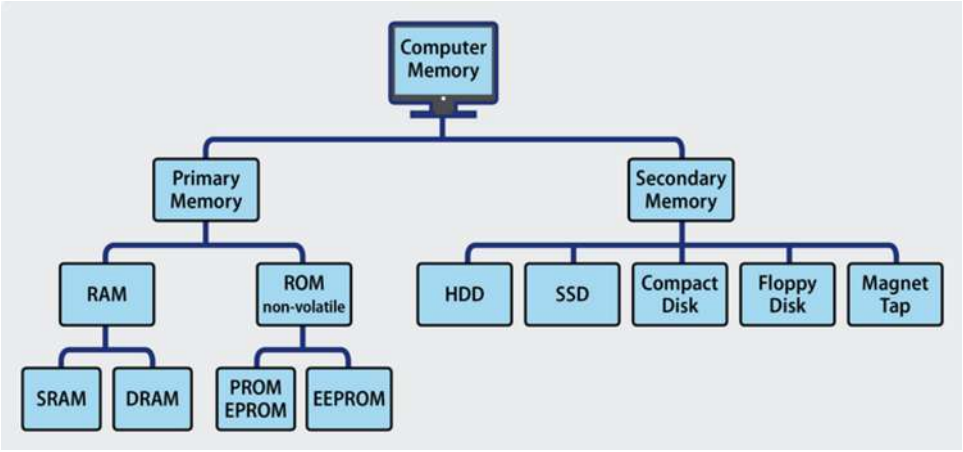
- Bu belge oluşturulurken https://www.udemy.com/course/stm32f4-discovery-kart-ile-arm-dersleri/ linkteki eğitim kursu izlenirken alınan notlardan oluşmaktadır.

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.

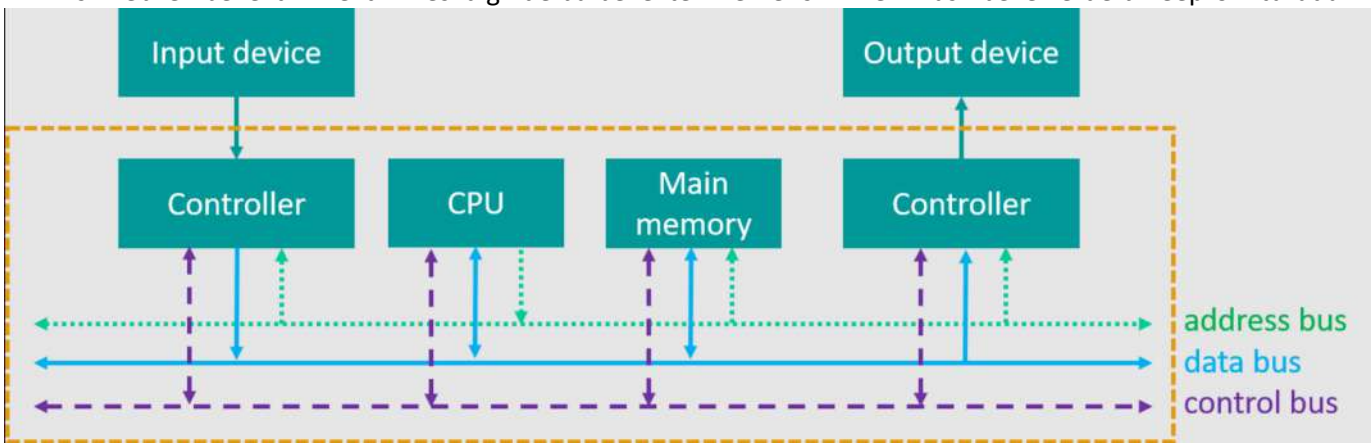
Mikroişlemci

- 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.
  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.
  REGISTER, hafızadaki veriler ALU tarafından işlenirken kullnılan geçici ve kalıcı saklayıcılardır. Registerler işlemcinin çekirdeğinde olduklarından verilere ulaşmak daha hızlı gerçekle 3- Control Unit: Kontrol birimi, işlemcinin çalışmasını yönlendiren birimdir. İşlemci içerisindeki ve dışarısındaki birimlerin senkron şekilde çalışmasını sağlar.



- RAM, ROM ve EEPROM hafızasının 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/çışı 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.

Mikrodenetleyici

- 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 (real time) 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.

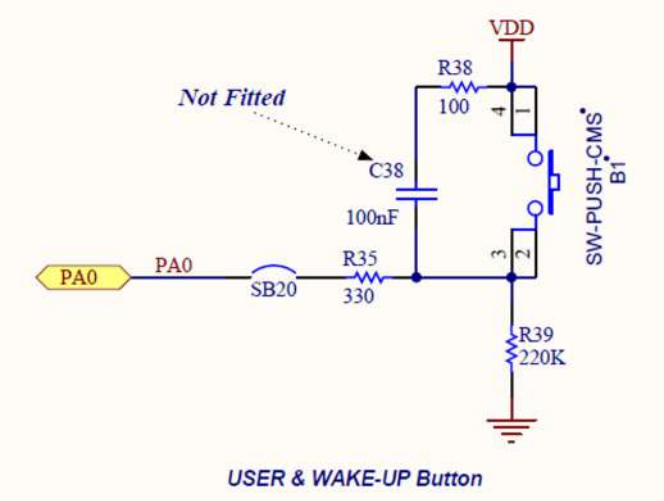
01 GPIO

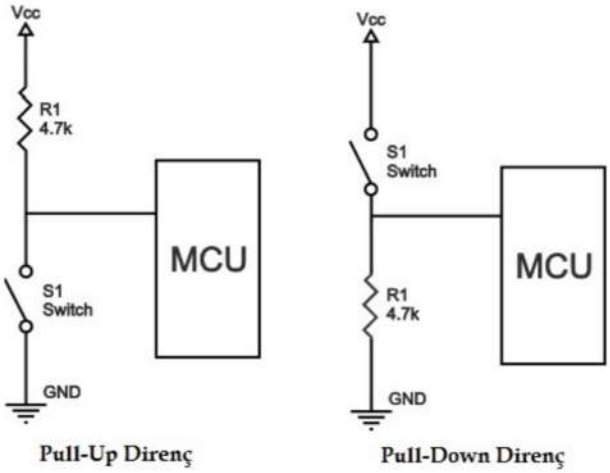
5 Mayıs 2021 Çarşamba 08:02

01 GPIO

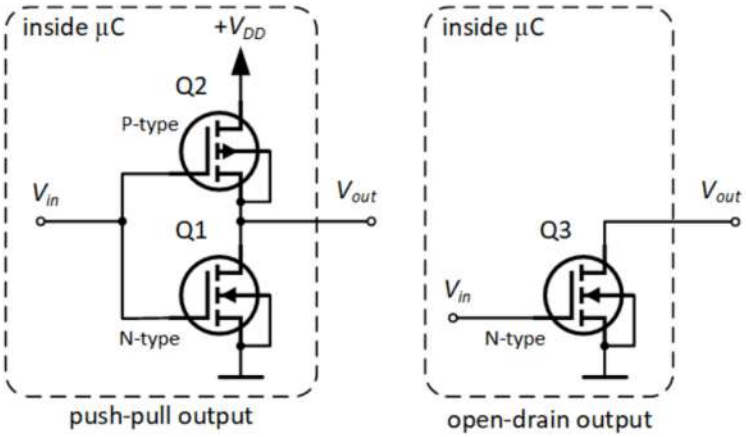
Giriş

- Butonlar ve anahtarlar mikrodenetleyiciye giriş pini üzerinden lojik 1 ve lojik 0 olarak bilgi girişini sağlayan mekanik elemanlardır.
- Resimde görüldüğü gibi kullanıcı butonu A portunun 0. pinine bağlı ve pull down durumundadır.

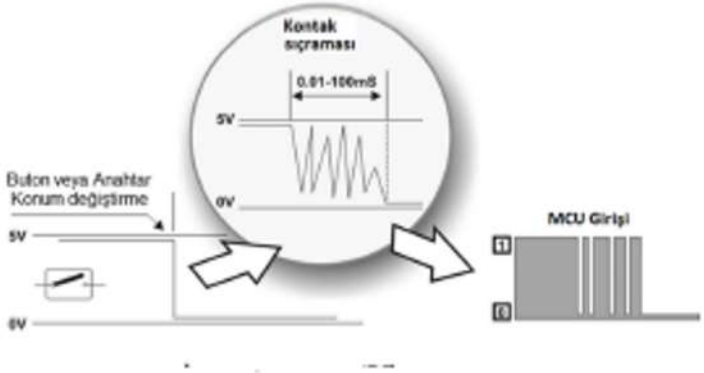


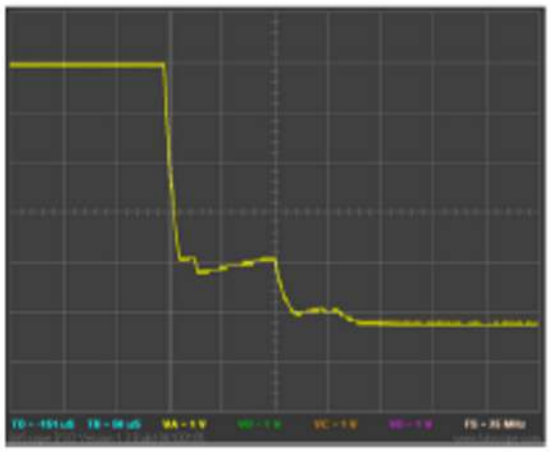


- PullUp bağlantıda GPIO girişi direnç üzerinden + beslemeye (VCC/VDD) bağlanır.
  Butona basılmadığı durumda GPIO girişinde lojik 1 vardır.
  Butona basıldığı durumda girişe 0V (lojik 0) uygulanmış olur.
- PullDown bağlantıda, GPIO girişi direnç üzerinden GND ye bağlanır.
  Butona basılmadığı durumda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



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



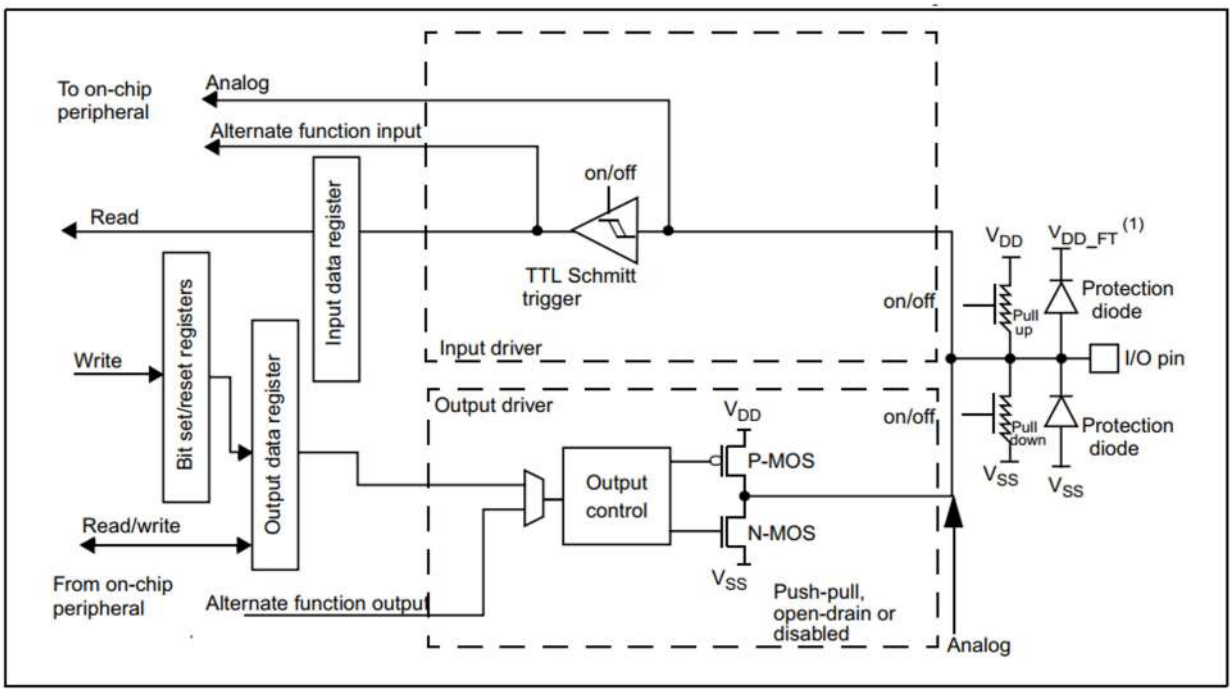


- Ark problemini https://akademi.robolinkmarket.com/buton-arki-nedir-nasil-cozulur/ linkten donanımsal ve yazılımsal olarak paylaşılan çözümleri inceleyip uygulayabiliriz.

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


To on-chip peripheral
Analog
Alternate function input
on/off
Read
Input data register
TTL Schmitt trigger
Write
Bit set/reset registers
Output data register
Input driver
Output driver
V_{DD}
V_{DD_FT} (1)
Protection diode
on/off
Pull up
I/O pin
P-MOS
Output control
N-MOS
Pull down
Protection diode
V_{SS}
V_{SS}
Read/write
From on-chip peripheral
Alternate function output
Push-pull, open-drain or disabled
Analog

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.

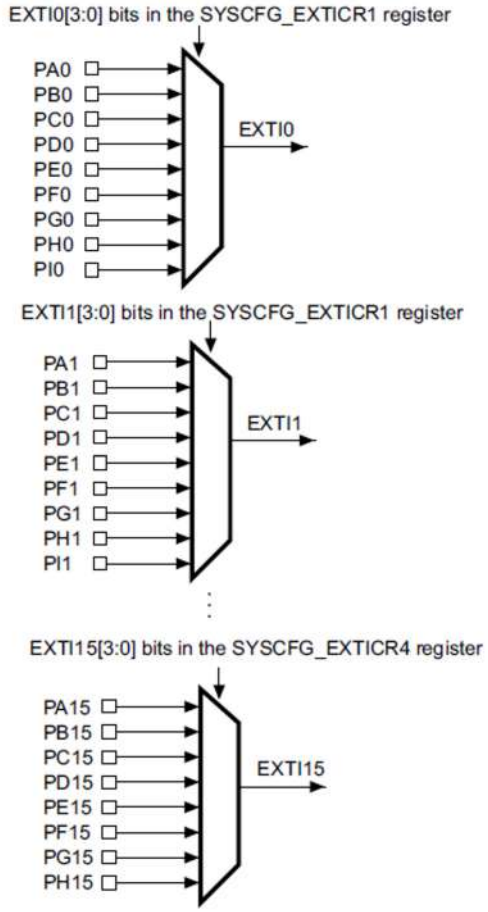
02 EXTI

5 Mayıs 2021 Çarşamba 08:02

02 EXTI

Giriş

- 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 kandi içindeki donanımlardan gelen kesmeleri anlayabiliriz.
- STM32F407 mikrodenetleyicisi için porttaki 0.pin EXTI0 kanalına bağlıdır.

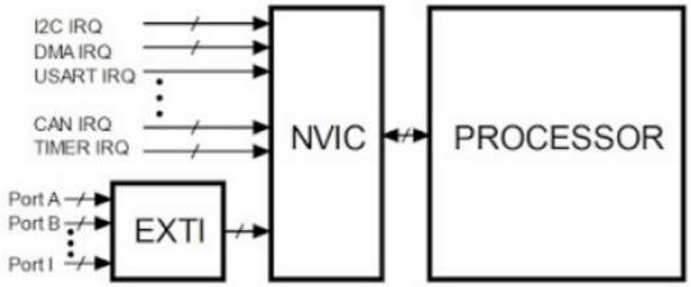


- Bunlar dışında 7 tane daha kanal vardır. Toplamda 23 kanal vardır.

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

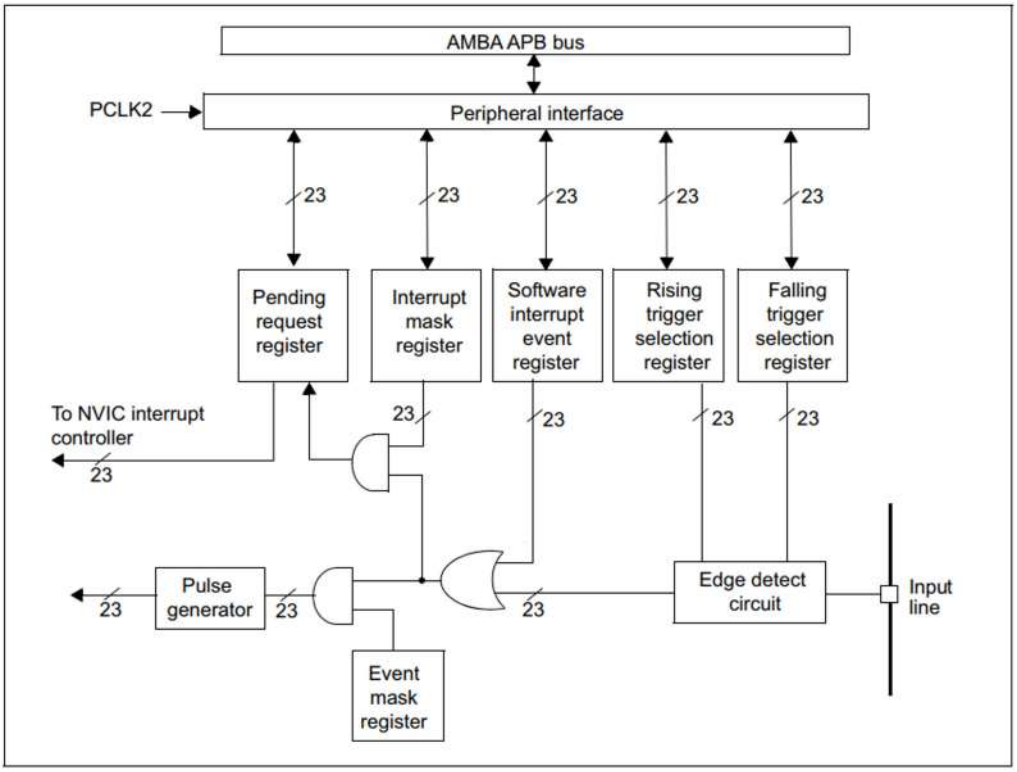
- Karmaşık kesme isteklerinin işlemciye sürekli yük getirmemesi için işlemci içerisinde özel bir donanım bloğu oluşturulmuştur. Bu donanıma interrupt controller adı verilir.
- Kesme kontrolörü haklı bir sebeple gelen kesme isteği neticesinde düzgün işleyen programı askıya alarak kesme fonksiyonu (interrupt 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.

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	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	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	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	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	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

- **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 kesme (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 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, 8 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.

ÇÖZÜNÜRLÜK	ADC ÇEVRİM HIZI
12 Bit	12 Cycle
10 Bit	10 Cycle
8 Bit	8 Cycle
6 Bit	6 Cycle

- 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.
- Clock değeri ADC'nin bağlı olduğu clock hattına bağlıdır.
- Tüm işlemcilerde aynı mantıktır fakat formül işlemciye göre farklılık gösterebilir bunun için kaynaklardan

bakılması gerekir.

$$T_{conv} = \frac{\text{Sampling time} + \text{Cycles}}{\text{ADC CLOCK}}$$

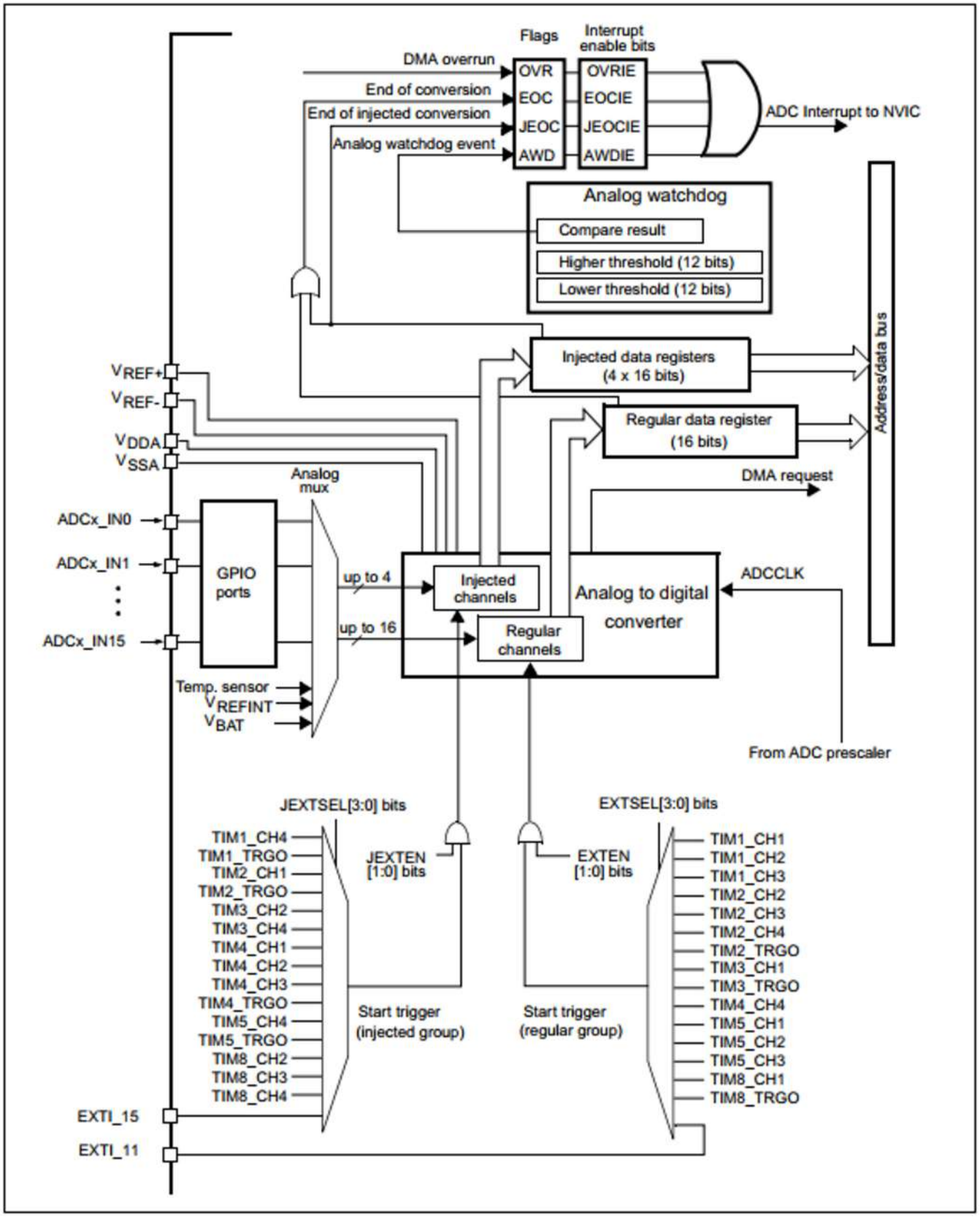
Çalışma Modları

- **Single Conversion Mode** (Tek Dönüşüm Modu): Bu mod, 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** (Sürekli Dönüşüm Modu): Bu modda ADC, başlatıldığı andan itibaren sürekli olarak dönüşümler gerçekleştirir. Otomatik durma olmadığı için dönüşümler devam eder, kullanıcı tarafından durdurulana kadar devam eder.
- **Scan Mode** (Tarama Modu): Bu modda ADC, 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** (Kesikli/Süreksiz Mod), 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


Flags
Interrupt enable bits
DMA overrun
OVR
OVRIE
End of conversion
EOC
EOCIE
End of injected conversion
JEOC
JEOCIE
Analog watchdog event
AWD
AWDIE
ADC Interrupt to NVIC
Analog watchdog
Compare result
Higher threshold (12 bits)
Lower threshold (12 bits)
Injected data registers (4 x 16 bits)
Regular data register (16 bits)
Address/data bus
DMA request
VREF+
VREF-
VDDA
VSSA
Analog mux
ADCx_IN0
ADCx_IN1
ADCx_IN15
GPIO ports
up to 4
up to 16
Injected channels
Regular channels
Analog to digital converter
ADCCLK
Temp. sensor
VREFINT
VBAT
From ADC prescaler
JEXTSEL[3:0] bits
EXTSEL[3:0] bits
TIM1_CH4
TIM1_TRGO
TIM2_CH1
TIM2_TRGO
TIM3_CH2
TIM3_CH4
TIM4_CH1
TIM4_CH2
TIM4_CH3
TIM4_TRGO
TIM5_CH4
TIM5_TRGO
TIM8_CH2
TIM8_CH3
TIM8_CH4
JEXTEN [1:0] bits
EXTEN [1:0] bits
Start trigger (injected group)
Start trigger (regular group)
TIM1_CH1
TIM1_CH2
TIM1_CH3
TIM2_CH2
TIM2_CH3
TIM2_CH4
TIM2_TRGO
TIM3_CH1
TIM3_TRGO
TIM4_CH4
TIM5_CH1
TIM5_CH2
TIM5_CH3
TIM8_CH1
TIM8_TRGO
EXTI_15
EXTI_11

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)**, Bu register, 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	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)**: ADC modülünün genel durumunu gösteren bu register, özellikle birden fazla ADC'nin kullanıldığı durumlarda ortak durumu izlemek için kullanılır.
- **ADC_CCR (Common Control Register)**: Bu 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.

04 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.

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4096}$$

- 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.

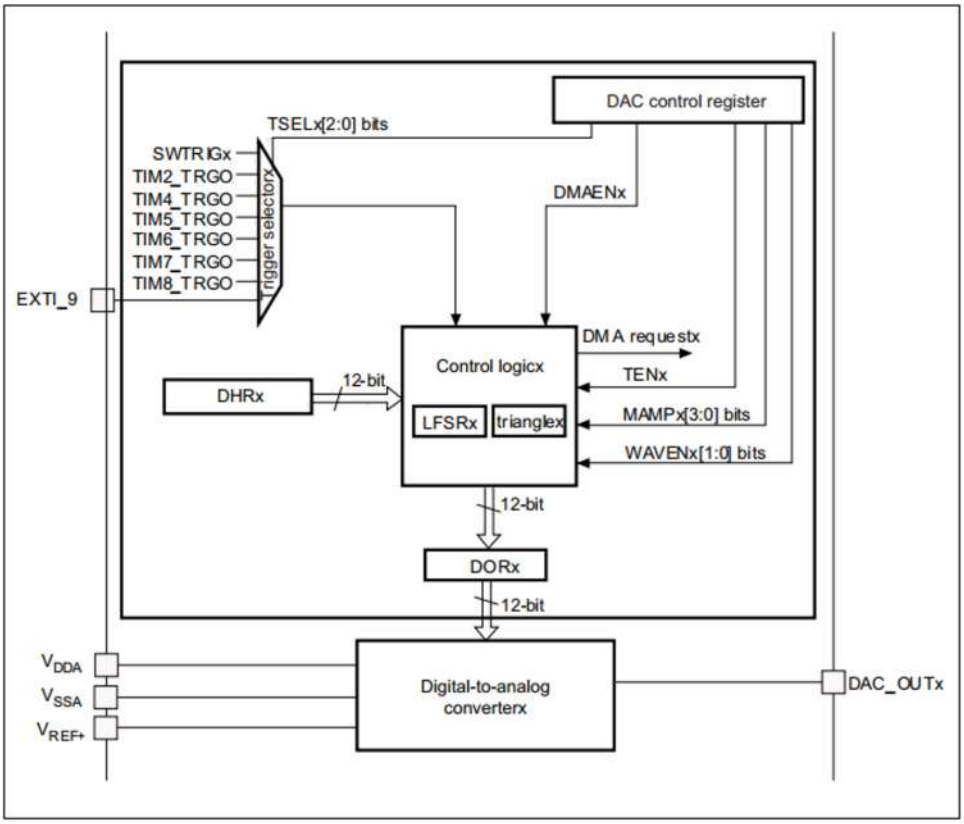
Tetikleme İşlemleri

- 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.

05 DMA

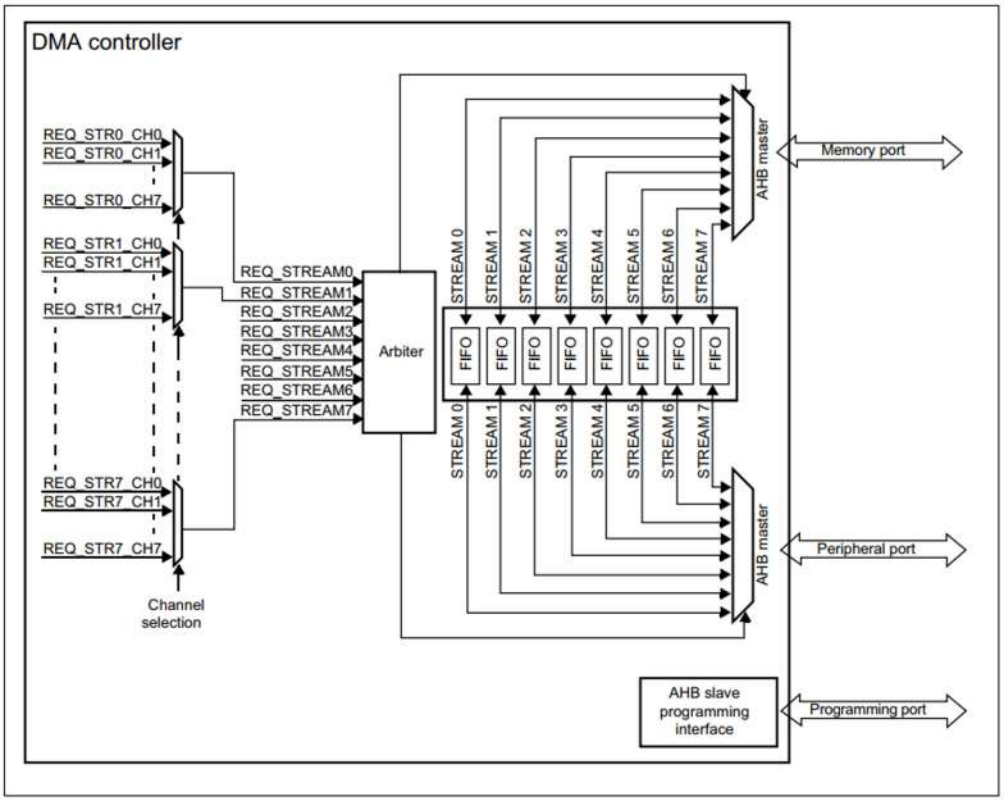
5 Mayıs 2021 Çarşamba 08:03

05 DMA

Giriş

- https://mikrodunya.wordpress.com/2016/06/23/dma-direct-memory-access-dogrudan-bellek-erisimi/
- DMA(Direct Memory Adres) gelişmiş mikrodenetleyicilerde periphal-memory veya memory-memory arasındaki veri transferlerini hiç bir CPU işlemini kullanmadan sağlaması amacıyla oluşturulmuştur.
- Çok veri alışverişi yapıldığı durumlarda kullanılması gerekir.
- 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 peripheral'dan hafızaya, hafızadan peripheral'a ve hafızadan hafızaya olmak üzere hızlı bir şekilde kaynak adresten hedef adrese aktarır.
  peripheral -> memory
  mermory -> peripheral
  memory -> memory
- 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.
- 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



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		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.

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.
- **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.
- 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.

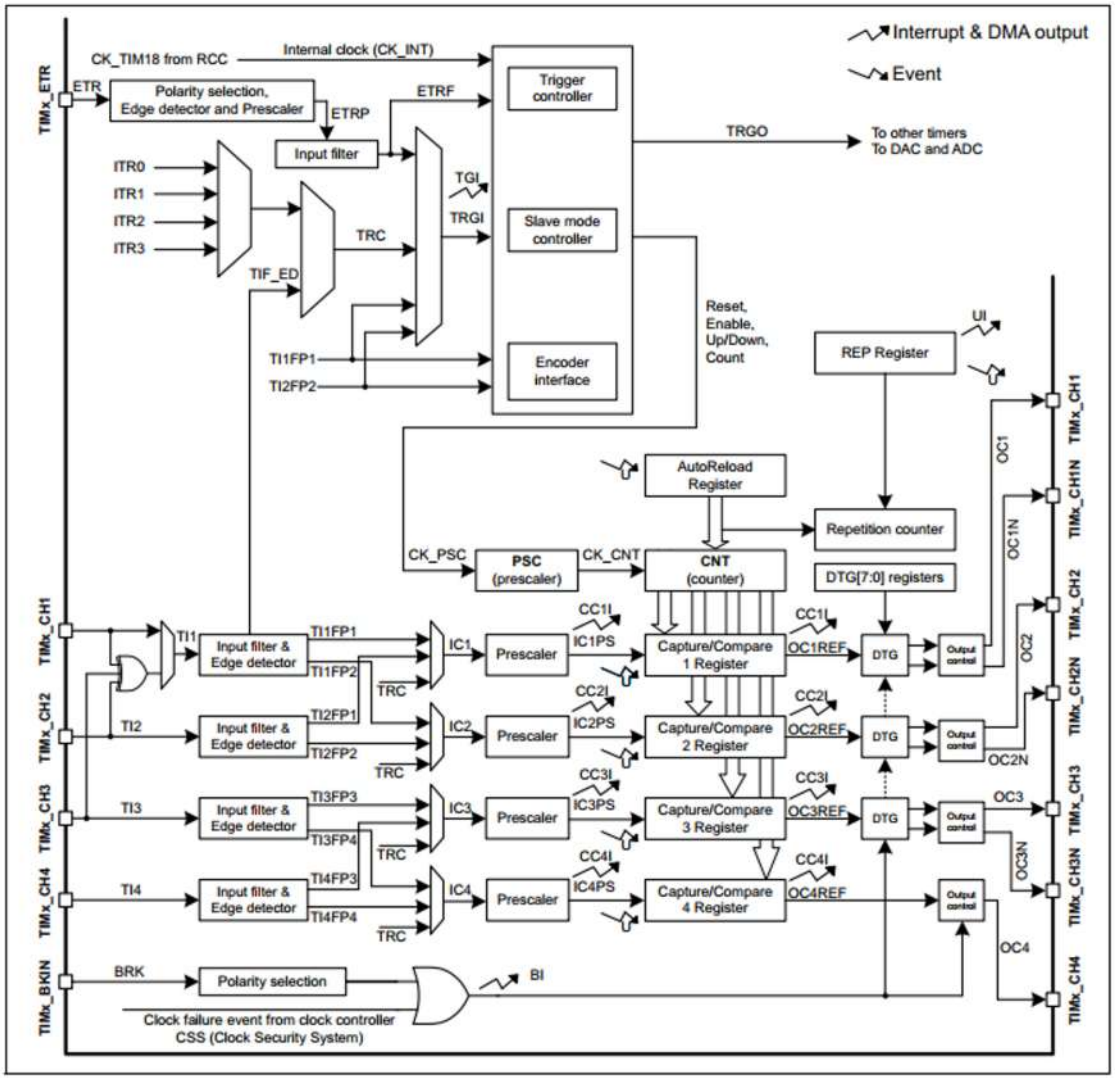
Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complemen-tary output	Max interface clock (MHz)	Max timer clock (MHz)
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	84	168
General purpose	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,	16-bit	Up	Any integer between 1	Yes	0	No	42	84

	TIM14	16-bit	Up	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.



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
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETP S [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

0x08	TIMx_SMCR	Reserved								ETP	ECE	SMS [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]				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	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

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

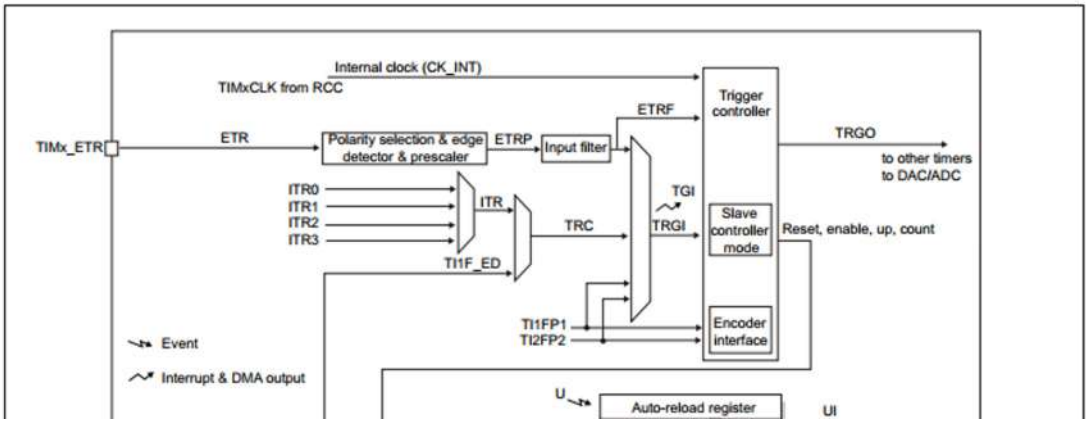
Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

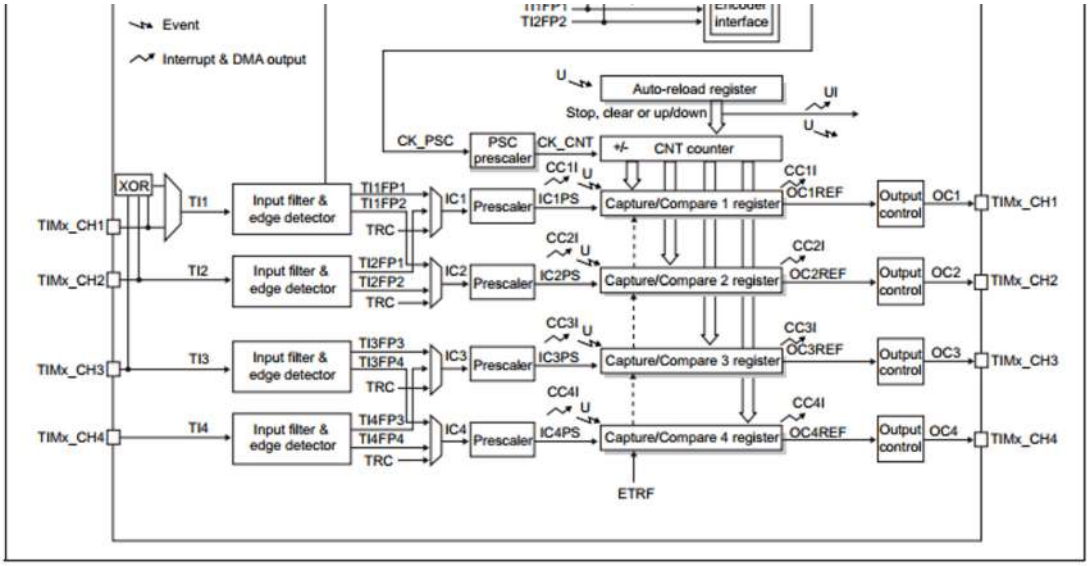
- **TIMx_CR1 (Control Register 1)**, Timer'ın genel kontrol ayarlarını içerir. Timer'ı etkinleştirme, zamanlama modu seçimi, otomatik yeniden başlatma etkinleştirme gibi ayarları içerir.
- **TIMx_CR2 (Control Register 2)**, Timer'ın özel kontrol ayarlarını içerir. Bu register, master mode seçimi gibi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





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	OC1G	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
		ARR[31:16]																															

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
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
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
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
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
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.

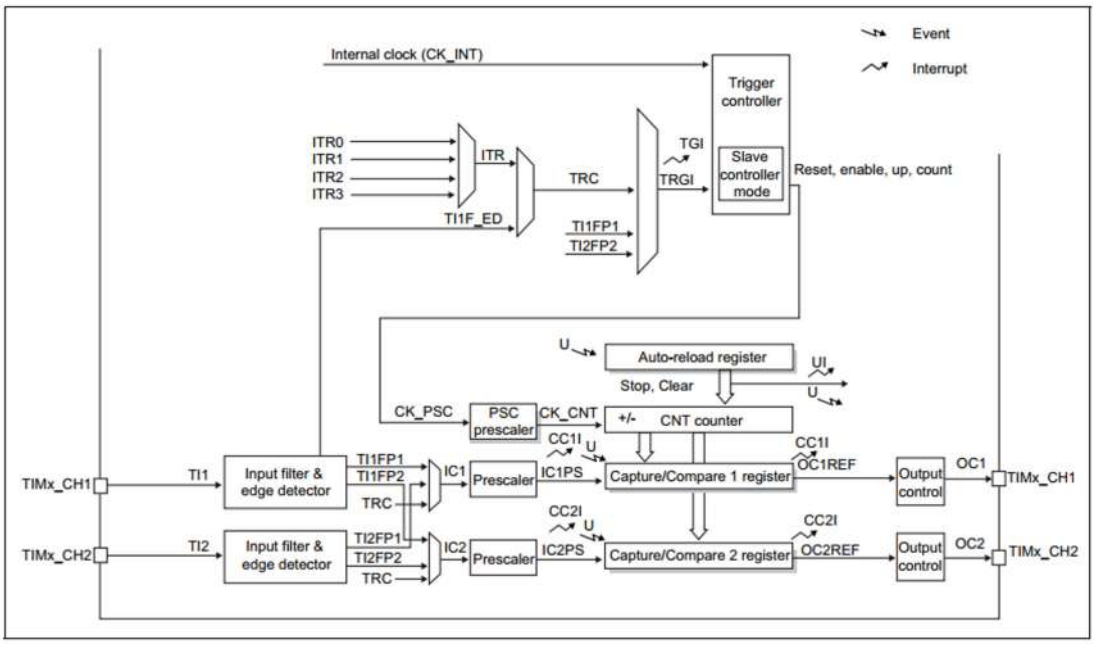
TIM9, TIM10, TIM11, TIM12, TIM13, TIM14

- TIM9 yüksek hızlı APB2 (84 MHz) ve TIM12 düşük hızlı APB1 (42 MHz) üzerinde bulunmaktadır

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

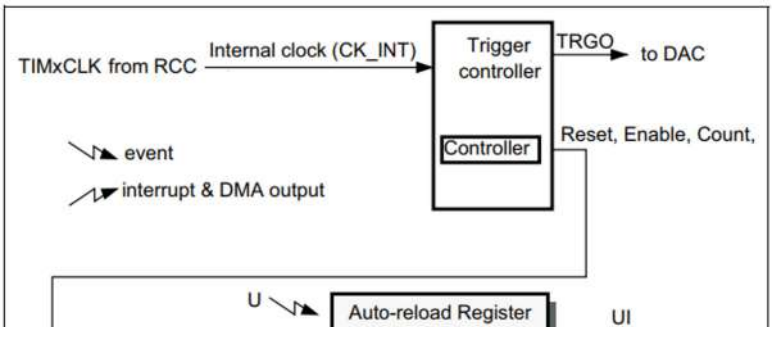
0x20	TIMx_CCER	Reserved																												CC1NP	Reserved	CC1P	CC1E
	Reset value																													0		0	0
0x24	TIMx_CNT	Reserved														CNT[15:0]																
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved														PSC[15:0]																
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	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	0	0
0x50	TIMx_OR	Reserved																														TI1_RMP	
	Reset value																															0	0

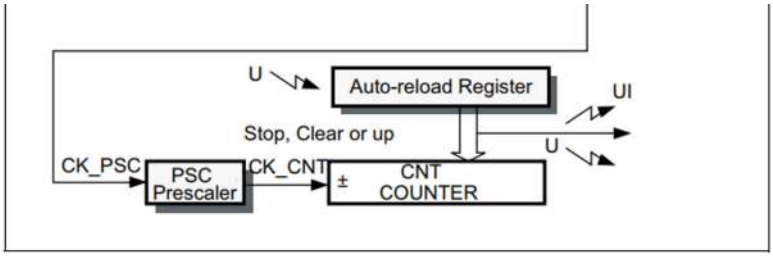
- **TIMx_CR1 (Control Register 1)**, Timer'ın genel kontrol ayarlarını içerir. Etkinleştirme, zamanlama modu seçimi, otomatik yeniden başlatma etkinleştirme ve diğer bazı genel ayarları içerir.
- **TIMx_DIER (DMA/Interrupt Enable Register)**, DMA (Direct Memory Access) ve kesme (interrupt) izinlerini 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.





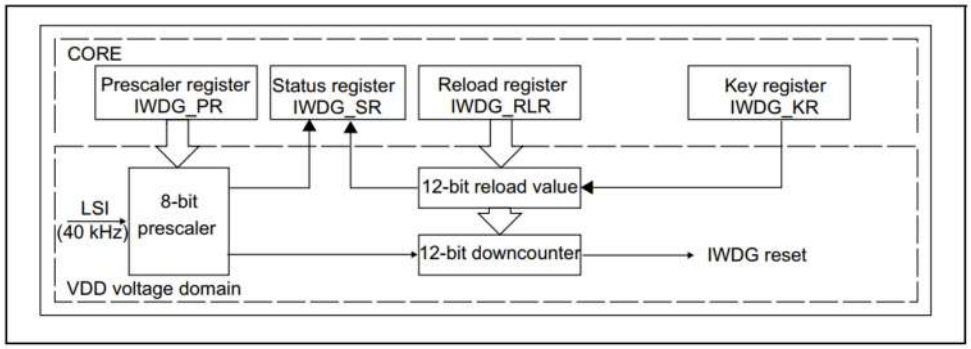
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																								ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																									0				0	0	0	0
0x04	TIMx_CR2	Reserved																									MMS[2:0]			Reserved				
	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.

kodlardaki bir hata sebebiyle kilitlenebilir. Mikrodenetleyici kilitlendiğinde, yarattığı 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.

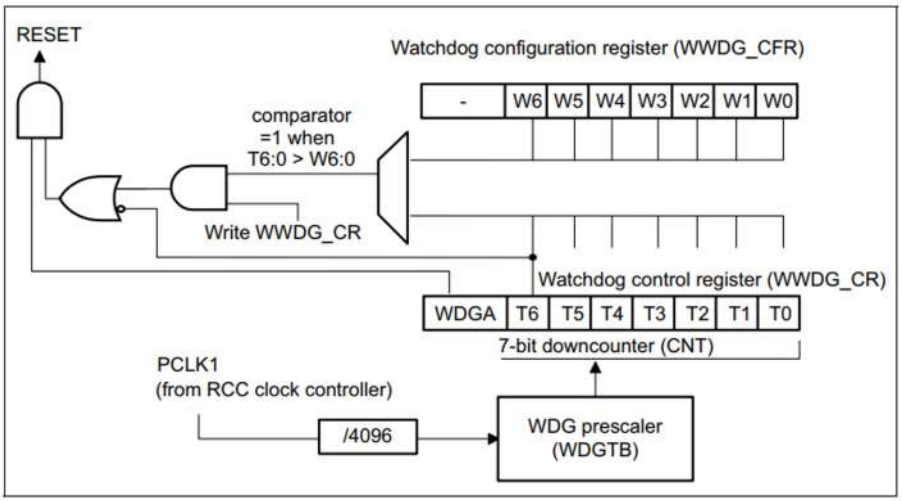


Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Reserved																													PR[2:0]		
	Reset value			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

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.

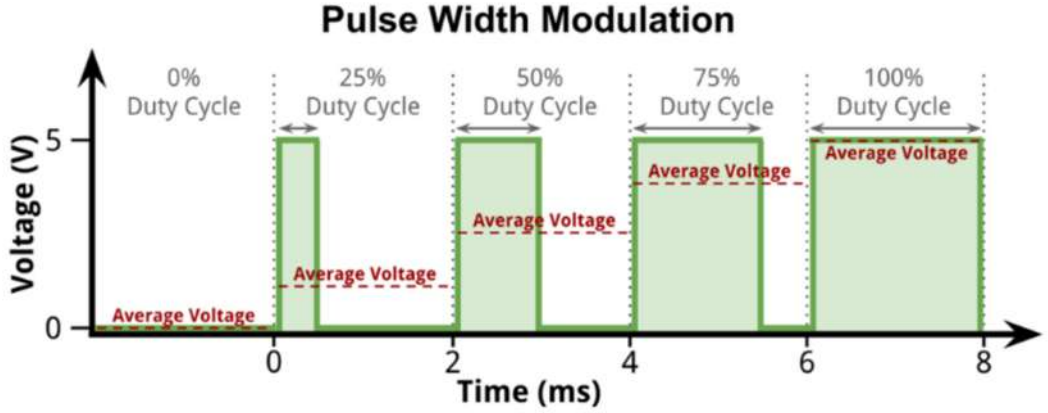
07 PWM

25 Haziran 2021 Cuma 23:48

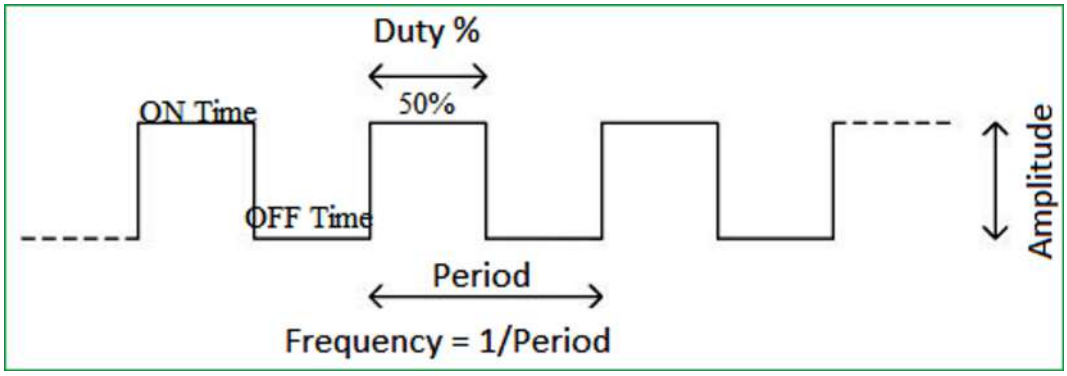
07 PWM

Giriş

- https://www.aydinlatma.org/pwm.html, https://berkannaydin.medium.com/pwm-nedir-5d20287970b5 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 endüstride iletişim, motor kontrol, ısıtma, aydınlatma gibi önemli bir çok alanda kullanılmaktadır.



- 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.



- 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.

$$Duty\ Cycle = \frac{T_{on}}{T_{on} + T_{off}} * 100$$

- 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.

$$f = \frac{1}{T}$$

- 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)
- Buradan şunu düşünmeliyiz. 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 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.

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;

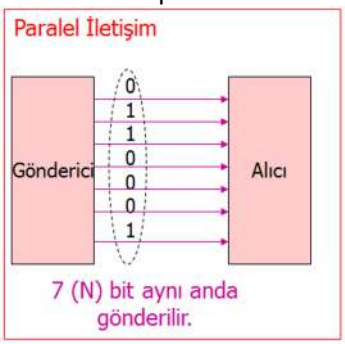
- Sadece iki kablo kullanır.
- Saat sinyali gerekli değildir.
- Hata denetimine izin vermek için bir eşlik biti vardır.
- Veri paketinin yapısı, her iki taraf da buna göre ayarlandığı sürece değiştirilebilir.
- İyi belgelenmiş ve yaygın olarak kullanılan yöntem.

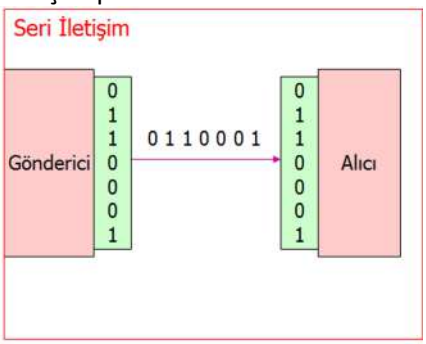
Dezavantajlar;

- 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

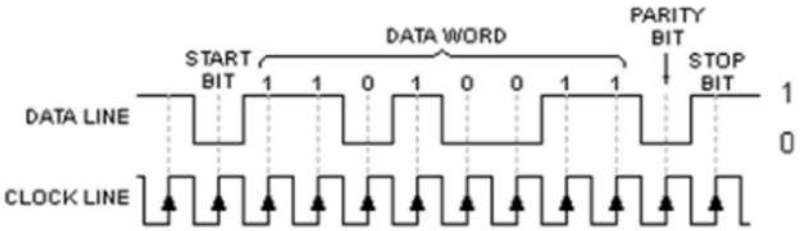
- Dijital sistemlerde iletişim paralel ve seri olmak üzere iki türlü yapılır.
- **Paralel** iletişimde bilgi vericiden alıcıya aynı anda birden fazla bit gidecek şekilde gönderilir. Böylece tek hamlede birden fazla veri karşı tarafa gönderildiği için iletişim hızı yüksektir. Fakat bu iletişim türünde 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.



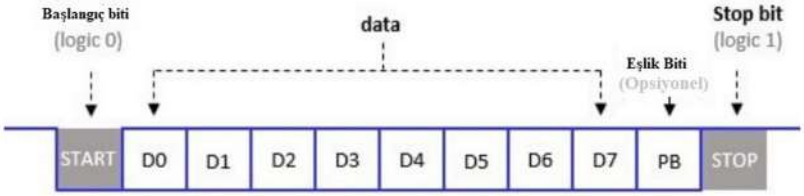


İletişim Modu

- 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.

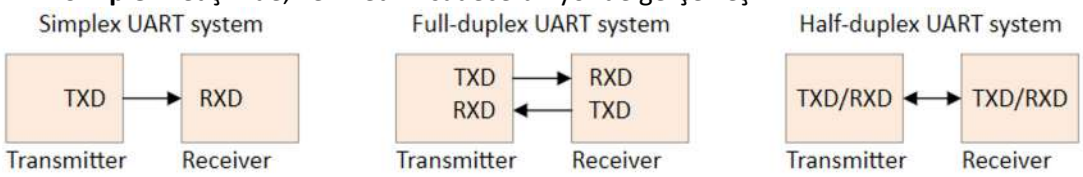


Parametreler

- 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.
- Bir bitin iletimi için gereken süre, 1 / Baudrate olarak hesap edilir. Saniye cinsinden sonuç verir.
- Örneğin 9600 baud rate ile haberleşme yapıyorsak saniyede 9600 bit yani 1200 byte veri gönderebiliyoruzdur. Bir bitin iletimi için geçen süreyi 1/9600'den çıkan sonucu daha sonra us çevirmek için 1000000 ile çarparız. Sonuç olarak 104,16us buluruz.
  115200 baudrate kullanımında 8,68us bulunur.
- 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.

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.

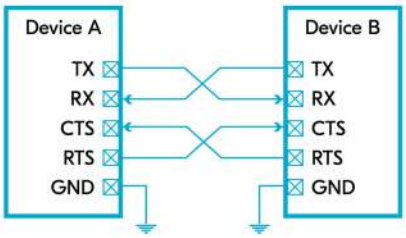


Pin Yapısı

- 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.

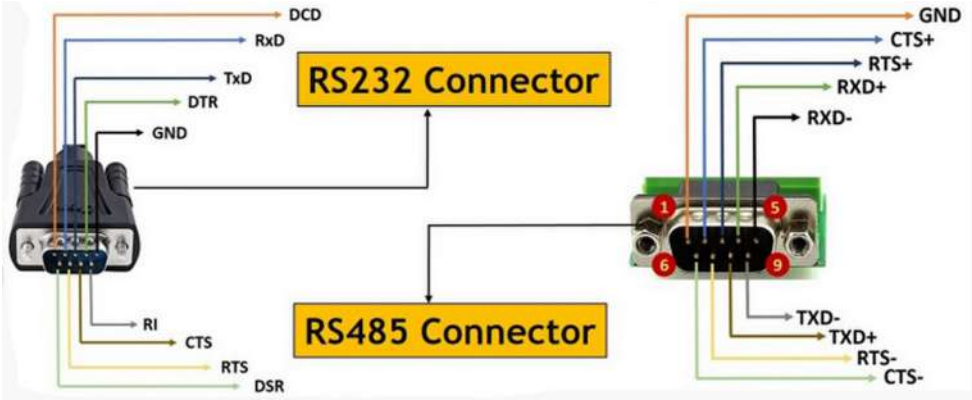


Fiziksel Standartlar

- 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-485** ve **RS-422**, 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.
- RS485, birden fazla cihaz arasında noktadan noktaya veya çok noktaya bağlantılar sağlayabilir. Düşük hızlardan (baud hızı) yüksek hızlara kadar geniş bir veri iletim hızı aralığına sahiptir. Genellikle 100 kbps ila 10 Mbps arasında değişen hızlarda iletişim sağlar. Uçtan uca maksimum mesafe 1200 metreye kadar olabilir.
- 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:

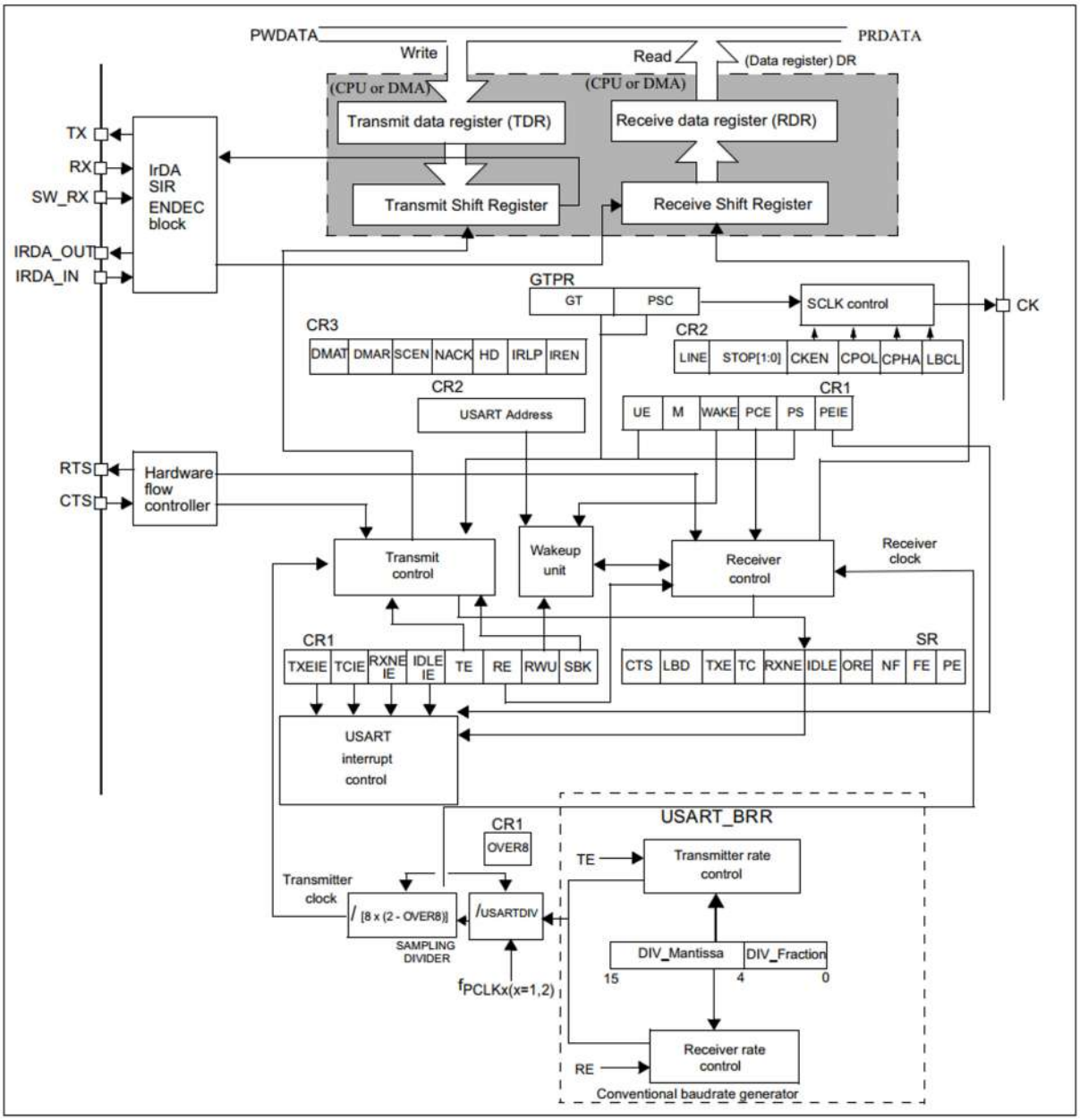


- RS-232, eski ve yaygın olarak kullanılan bir seri haberleşme standardıdır. Ancak daha yüksek veri hızları, uzun mesafe iletimi ve çok noktalı haberleşme gerektiren uygulamalarda RS-485 gibi daha modern haberleşme standartları tercih edilmektedir.

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


PWDATA
PRDATA
Write
Read
(Data register) DR
(CPU or DMA)
(CPU or DMA)
Transmit data register (TDR)
Receive data register (RDR)
Transmit Shift Register
Receive Shift Register
TX
RX
SW_RX
IrDA SIR ENDEC block
IRDA_OUT
IRDA_IN
GTPR
GT
PSC
SCLK control
CK
CR3
DMAT DMAR SCEN NACK HD IRLP IREN
CR2
LINE STOP[1:0] CKEN CPOL CPHA LBCL
CR2
USART Address
CR1
UE M WAKE PCE PS PEIE
RTS
CTS
Hardware flow controller
Transmit control
Wakeup unit
Receiver control
Receiver clock
CR1
TXEIE TCIE RXNE IE IDLE IE TE RE RWU SBK
SR
CTS LBD TXE TC RXNE IDLE ORE NF FE PE
USART interrupt control
USART_BRR
CR1
OVER8
TE
Transmitter rate control
Transmitter clock
/ [8 x (2 - OVER8)]
/USARTDIV
SAMPLING DIVIDER
DIV_Mantissa
DIV_Fraction
15
4
0
fPCLKx(x=1,2)
Receiver rate control
RE
Conventional baudrate generator

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	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 bayraklar (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 (Control Register 1)** ve **USART_CR2 (Control Register 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'da veri transfer hızı I2C veri yolundan daha hızlıdır.
- 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;

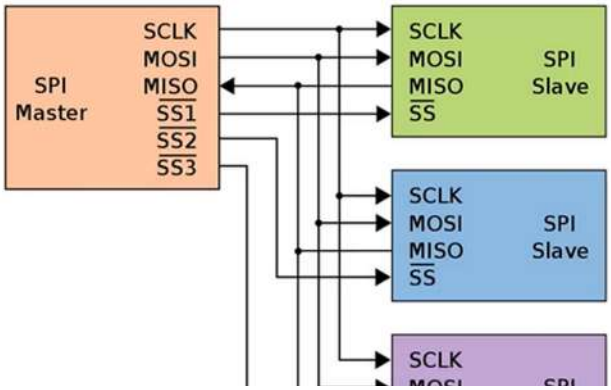
- Başlatma ve durdurma biti yok, böylece veriler kesintisiz olarak aktarılabilir.
- I2C gibi karmaşık bağımlı adresleme sistemi yok.
- I2C'den daha yüksek veri aktarım hızı (neredeyse iki kat daha hızlı).
- Ayrı MISO ve MOSI hatları, böylece veri aynı anda gönderilebilir ve alınabilir.

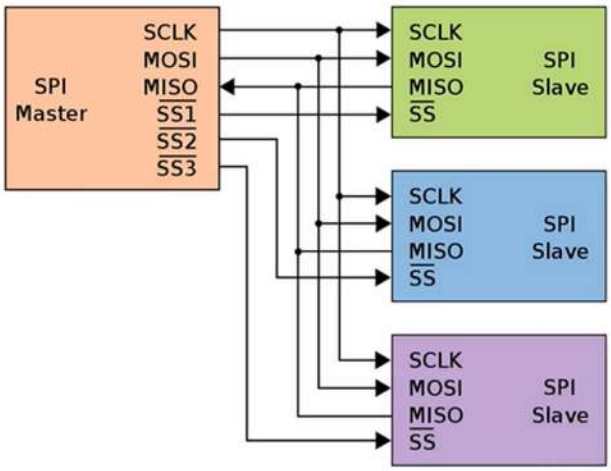
Dezavantajlar;

- 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

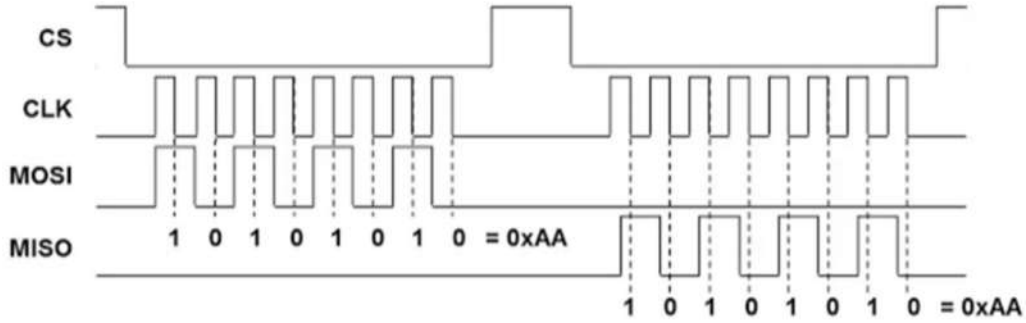
- SPI, genellikle dört telli bir bağlantıyla gerçekleştirilir:
  MOSI (Master Out Slave In), Master cihazdan (genellikle mikrodenetleyici) slave cihaza veri gönderir.
  MISO (Master In Slave Out), Slave cihazdan master cihaza veriyi gönderir.
  SCLK (Serial Clock): Saat sinyali, veri iletim hızını senkronize 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.



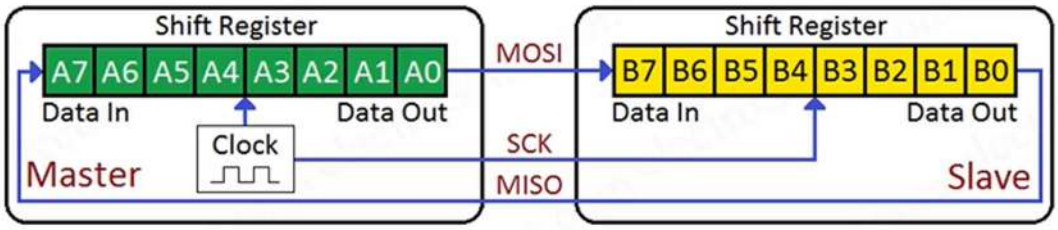


Veri İletimi

- 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.



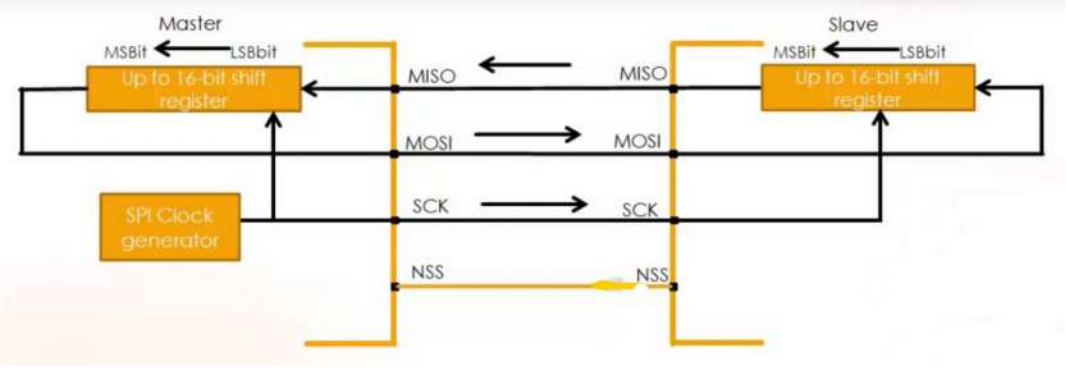
- SPI'da veri iletim sırası genellikle 8 bitlik veri paketleri halinde olur, ancak bu uzunluğu değiştirilebilir. Veri iletim sırası, verilerin en yüksek veya en düşük anlamlı bit ile başlayıp bitişebileceği şekilde yapılandırılabilir.



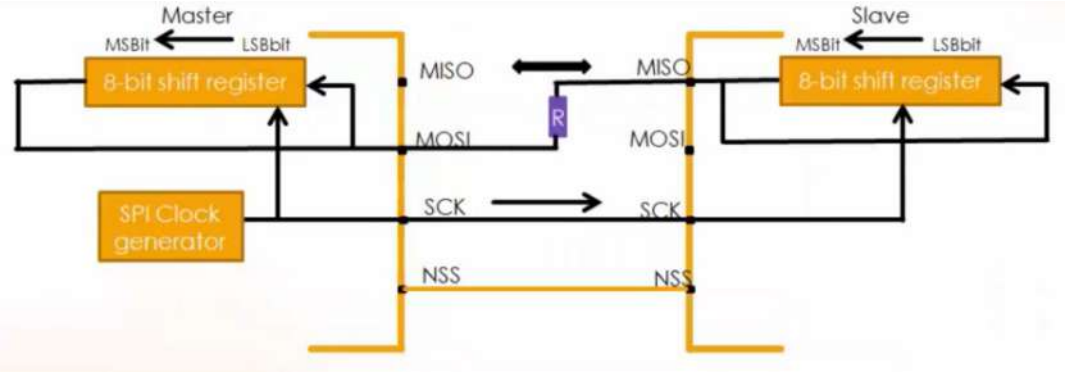
- Ana cihaz, iletişimi başlatır ve sonlandırır. Slave cihazlar ise ana cihazın taleplerine yanıt verir.

Veri Yolu

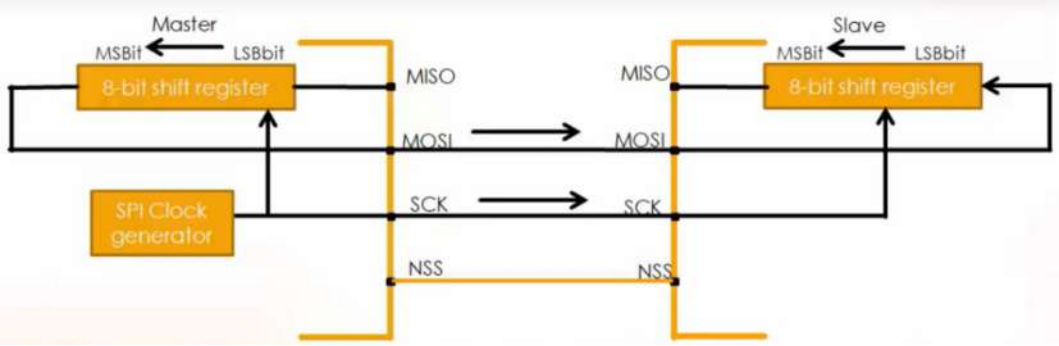
- SPI haberleşmesi için üç farklı mod vardır. Bunlar Full Duplex, Half duplex ve Simplex'dir.
- İkisi çift yönlü iken diğeri tek yönlü haberleşmedir. Bu modlar hakkında detaylı bilgi almak için https://fastbitlab.com/spi-bus-configuration-discussion-full-duplex-half-duplex-simplex/ linkteki yazıyı okuyabiliriz.
- 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 hattını kullanarak veri gönderebilir veya alabilir, ancak aynı anda her iki yönde veri iletimi yapılamaz.



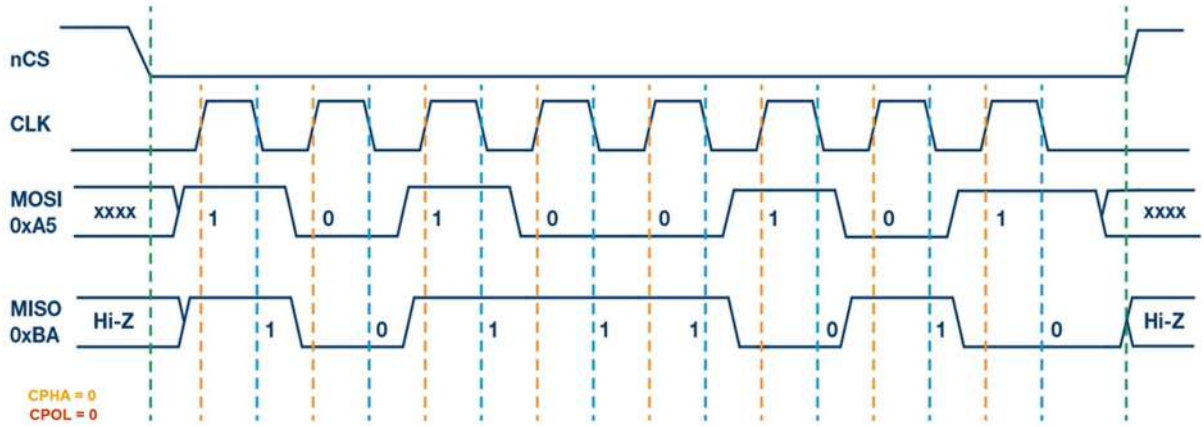
- **Simplex,** sadece **bir yönde** veri iletimine izin veren bir çalışma modudur.
  Genellikle tek bir iletişim hattı kullanılır ve veri gönderme veya veri alma işlemi yapılır. Her iki tarafta aynı anda veri gönderip alamaz.
- Eğer master sadece veri gönderip slave sadece veri alacaksa, bu durumda MOSI 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.

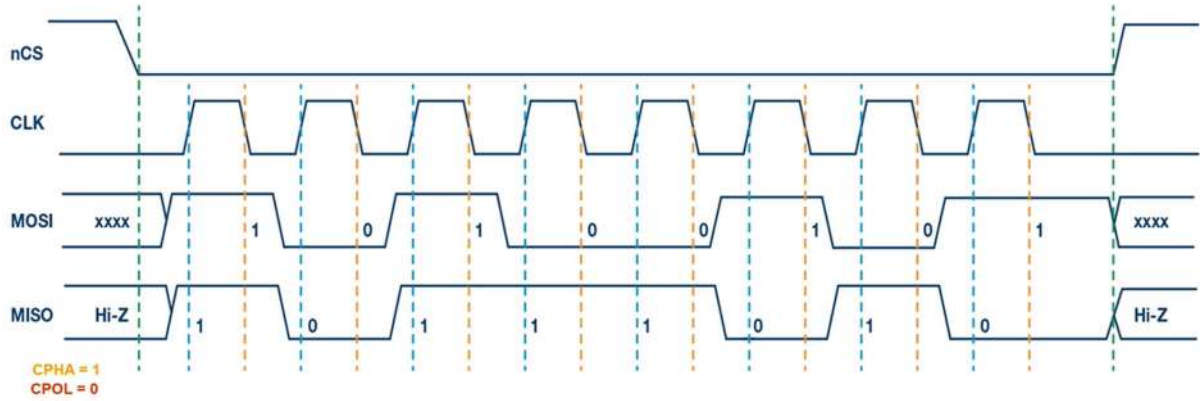


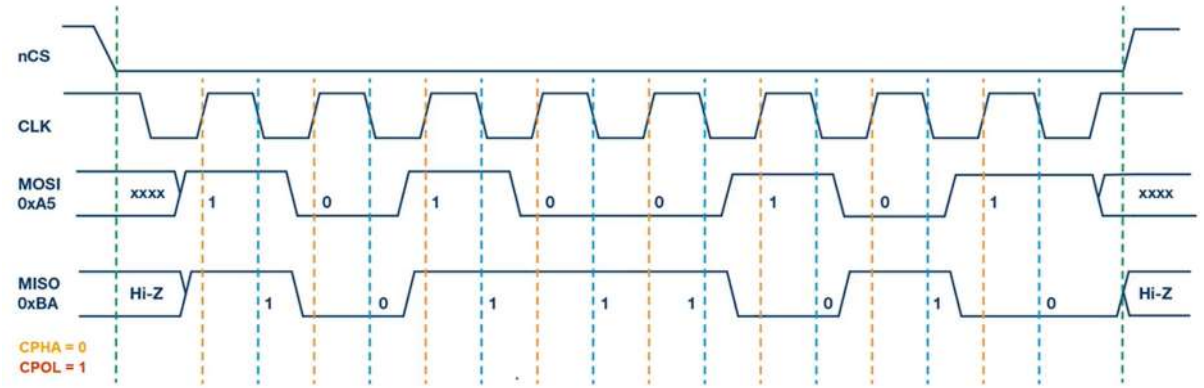
Çalışma Modları

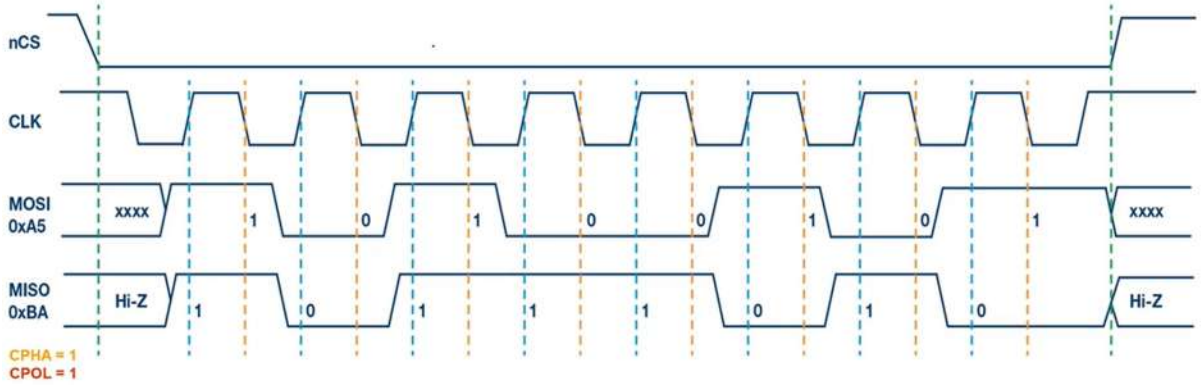
- SPI iletişiminde saat sinyalinin nasıl üretileceğini belirler. Clock polarity (CPOL), saat sinyalinin **yüksek** veya **düşük** seviyede başlayacağını belirtir, clock phase (CPHA) ise veri okuma/yazma işleminin saat sinyalinin hangi **kenarında** gerçekleşeceğini belirler.
- **SPI Modu 0 (CPOL=0, CPHA=0)**
  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.

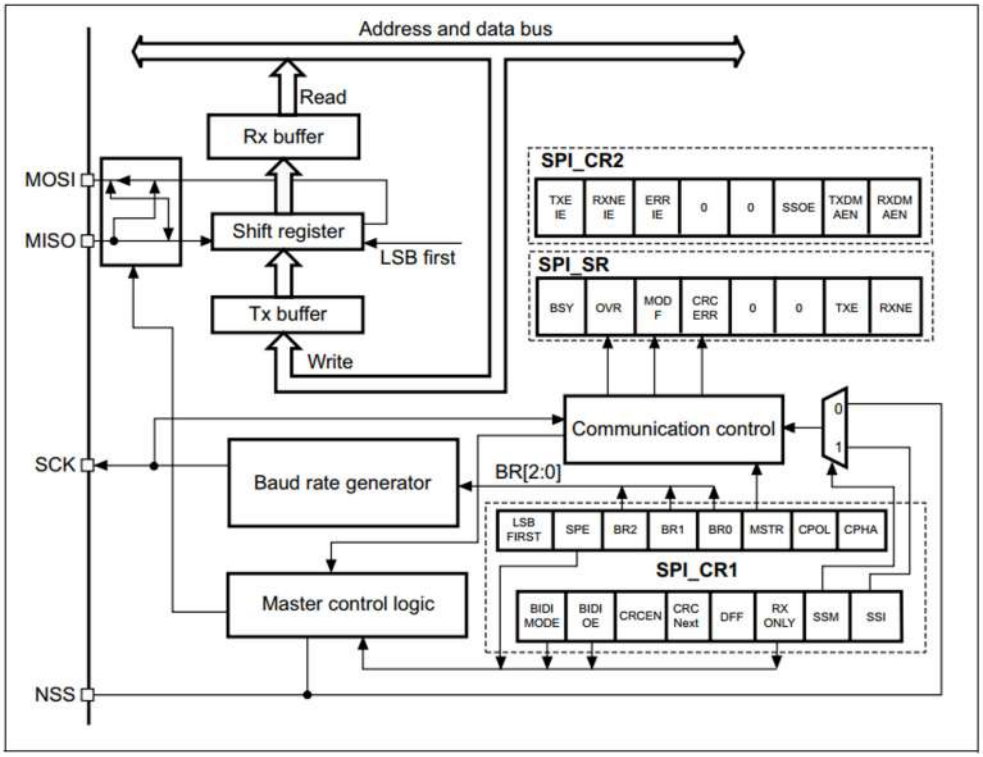








Birim Yapısı



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 (Control Register 1)** ve **SPI_CR2 (Control Register 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?ts=1702118996423&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fen-us%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US%2526searchTerm%253DUnderstanding%2Bthe%2BI%2B2C%2BBus%2526nr%253D1136 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;

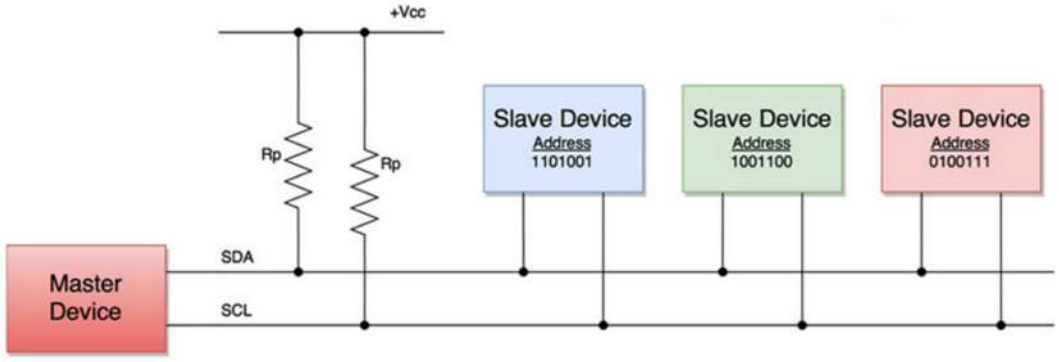
- Sadece iki telli bir yapıya sahiptir (SCL ve SDA), bu nedenle **az pin** kullanımı ile birçok cihazın bağlanmasını sağlar.
- Aynı hat üzerinden **çift yönlü iletişim** sağlar. Hem master hem de slave cihazlar veri gönderebilir ve alabilir.
- Bir dizi cihazın (EEPROM, sensörler, ekranlar vb.) bağlanmasını 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.

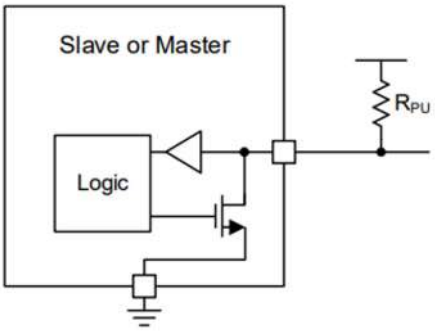


Çift Yönlü Haberleşme

- 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

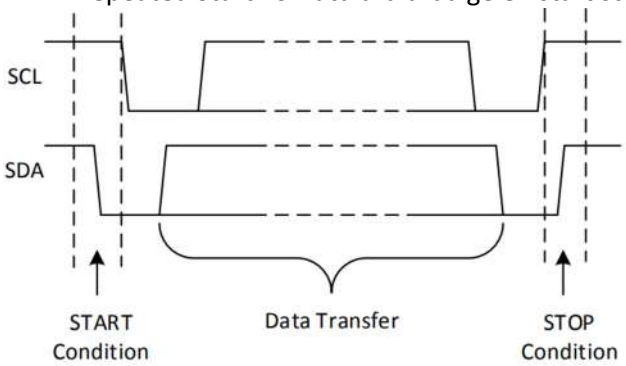
direncinin voltajı yükseltmesine izin verir.

- 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.

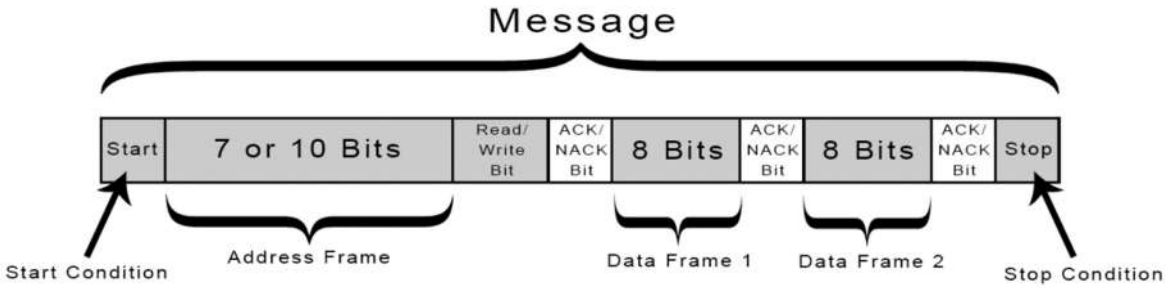


Veri İletimi

- 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.

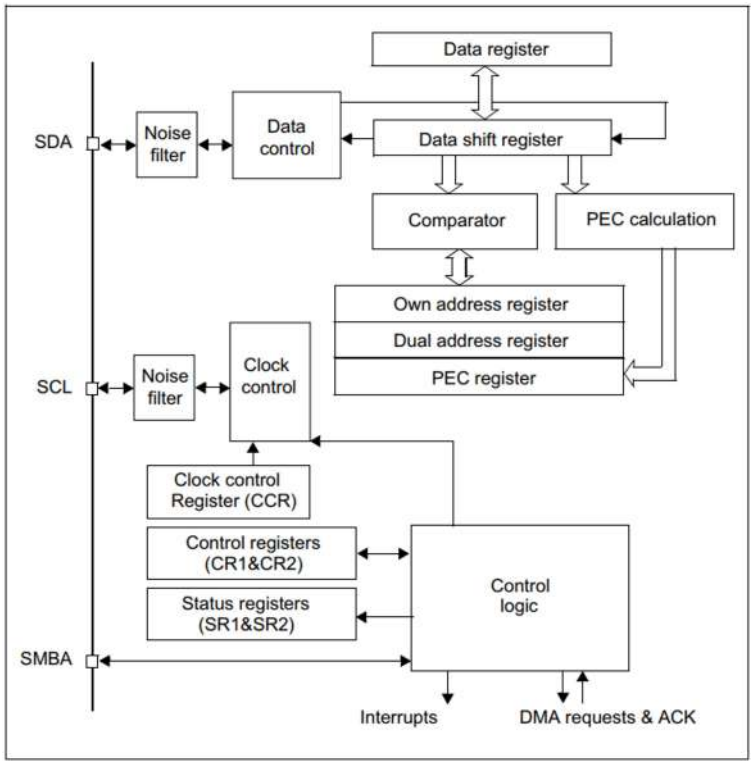


- İlk olarak SDA ve SCL hatları HIGH konumdadırlar. Daha sonra SDA hattı master tarafından **LOW** seviyeye çekilerek **iletişimin başlayacağı**, slave cihazlara bildirilir.
- Bu bildirimi alan slave cihazlar, adres bilgisini beklemeye başlarlar.
  Adres bilgisi slave cihazların yapısına göre 7 bit, 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	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.