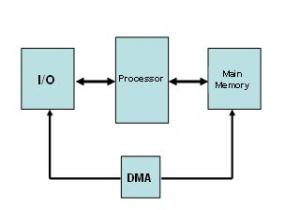
**DMA(Direct Memory Access)**

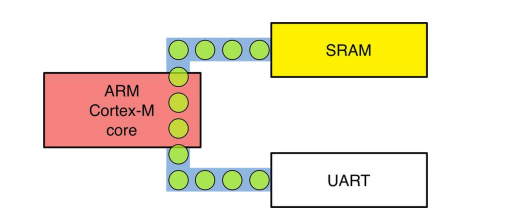
*Hazırlayan: Arif Mandal*

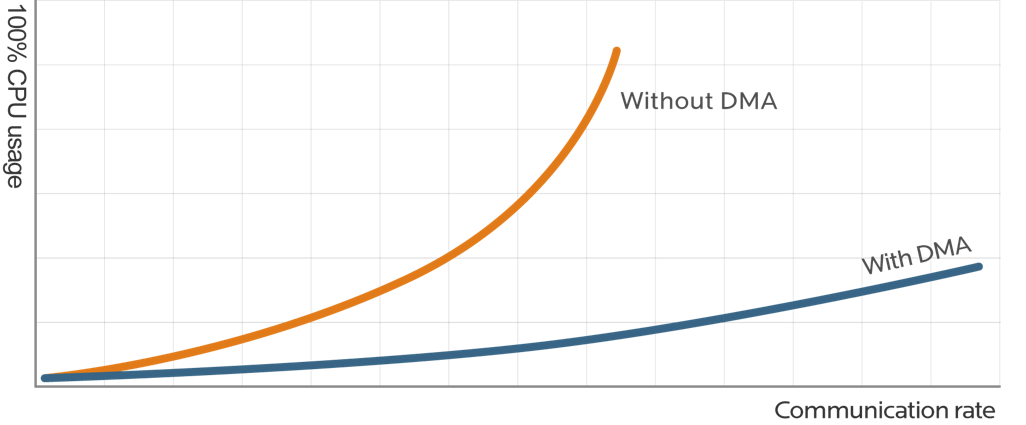
Her gömülü uygulamanın dış dünyayla veri alışverişi yapması veya çevre birimlerini çalıştırması gerekir. Örneğin, mikrodenetleyici, bir UART kullanarak sistemdeki diğer modüllerle mesaj alışverişi yapabilir veya mevcut SPI arabirimlerinden birini kullanarak verileri harici bir flaşta depolayabilir. Bu, dahili SRAM veya flash bellek ile çevresel kayıtlar arasında belirli bir miktarda verinin aktarılmasını içerir ve aktarımı gerçekleştirmek için belirli sayıda CPU döngüsü gerektirir. Bu, bilgi işlem gücü kaybına sebep olur. (CPU aktarım sürecinde işgal edilir).Genel performansların azalmasına ve nihayetinde önemli asenkron olayların kaybına yol açar.

Doğrudan Bellek Erişimi (DMA) denetleyicisi, MCU çevre birimlerinin Cortex-M çekirdeği müdahalesi olmadan dahili belleklere erişmesini sağlayan özel ve programlanabilir bir donanım birimidir.

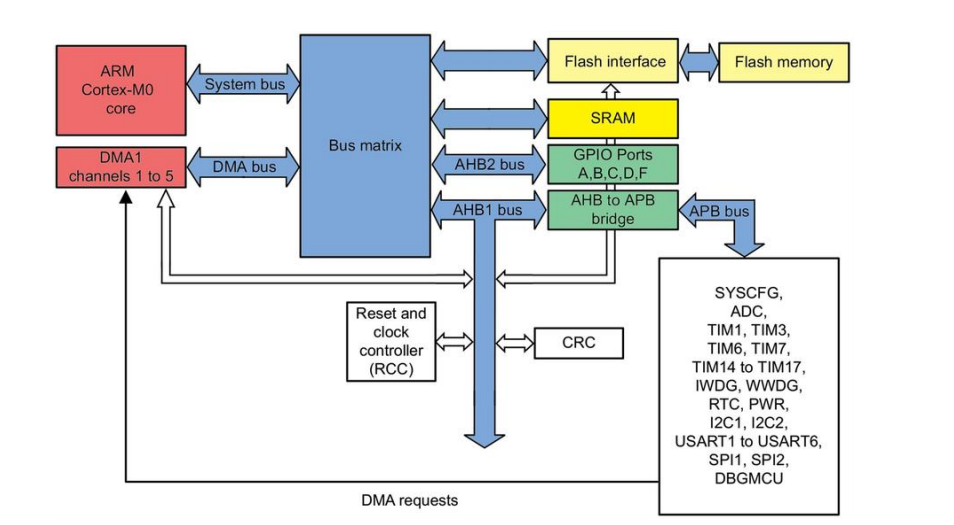


CPU, veri aktarımı sırasında oluşan işlem gücü kaybından tamamen kurtarılır (DMA yapılandırması hariç) ve paralel olarak başka etkinlikler de gerçekleştirebilir. DMA her iki şekilde de çalışacak şekilde tasarlanmıştır (yani bellekten çevresel birime veya çevresel birimden belleğe veri aktarımına izin verir) ve tüm STM32 mikro denetleyicileri en az bir DMA denetleyicisi sağlar. DMA, modern MCU'ların “gelişmiş” bir özelliğidir.



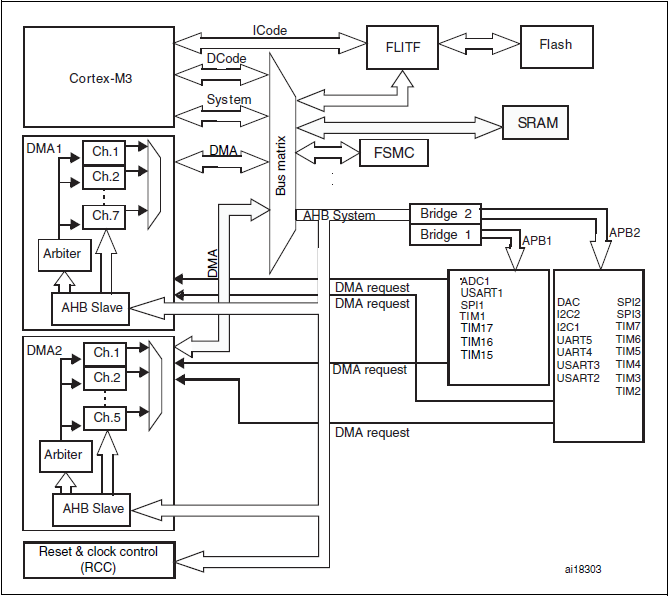


**Özellikleri:**

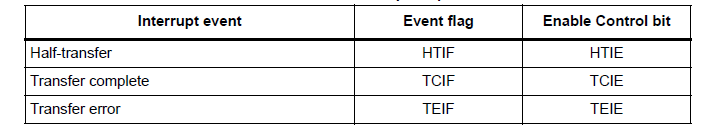
* 12 bağımsız ayarlanabilir kanal.
* Her kanal donanımın DMA talebine yönlendirilmiştir ve yazılımsal tetikleme yapılabilir. Ayarlama yazılım tarafından yapılır.
* DMA birimine kanallardan gelen talepler öncelik sırasına göre sıraya koyulabilir. Bu ayar da yazılım tarafından yapılır.
* Bağımsız kaynak ve transfer boyutu mevcuttur. Bayt, yarım word ve word biçiminde veri transfer edilebilir.
* Dairesel döngü tipi desteği vardır.
* 3 adet olay bayrağı bulunmaktadır. Yarım transfer(DMA Half Transfer), Transfer Tamamlanma (DMA Transfer Complete) ve Transfer Hatası(DMA Transfer Error) bayraklarıdır.
* Hafızadan hafızaya transfer desteği vardır.
* Çevre biriminden hafızaya, hafızadan çevre birimine ve çevre biriminden çevre birimine transfer desteği vardır.
* Flash, SRAM, APB ve AHB çevre birimlerine kaynak ya da hedef olarak erişim desteği vardır.
* 65535’e kadar veri gönderilebilir.

**DMA Çalışma Prensibi:**

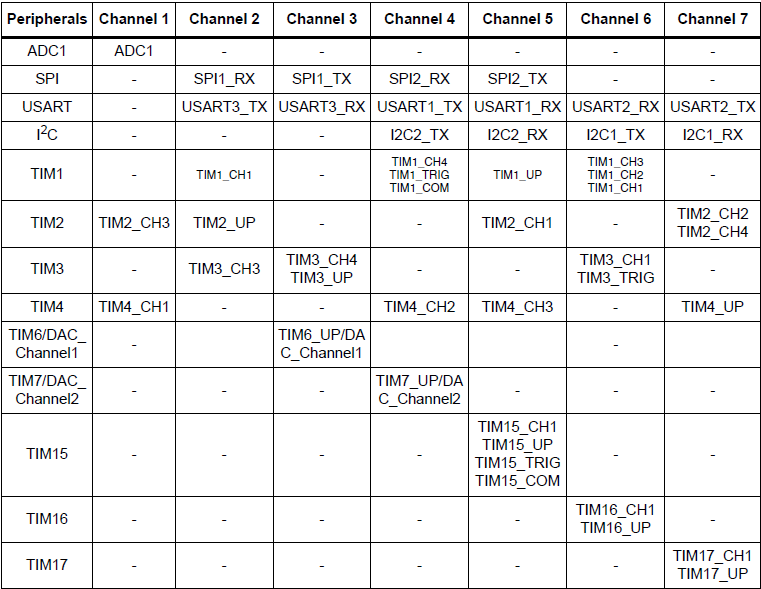
Bir DMA olayından sonra, çevre birimi DMA Kontrolörüne bir istek sinyali gönderir. DMA denetleyicisi, kanal önceliklerine bağlı olarak isteği sunar. DMA denetleyici çevre birimine erişir erişmez, çevre birimine DMA Denetleyici tarafından bir onay gönderilir. Çevre birimi, DMA Denetleyicisinden onay alır almaz isteğini serbest bırakır. Talep çevre birimi tarafından dağıtıldıktan sonra, DMA denetleyicisi onay'ı serbest bırakır. Daha fazla istek varsa, çevre birimi bir sonraki işlemi başlatabilir.



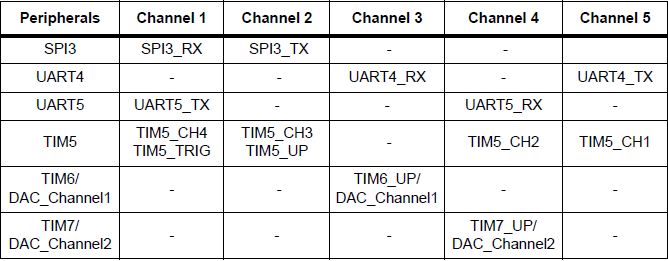
**DMA Kesme Bayrakları:**



**DMA1 İstek Kanalları:**

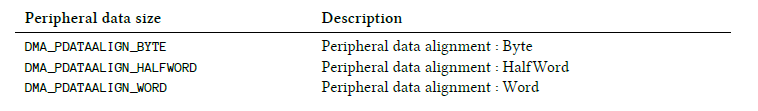


**DMA2 İstek Kanalları:**

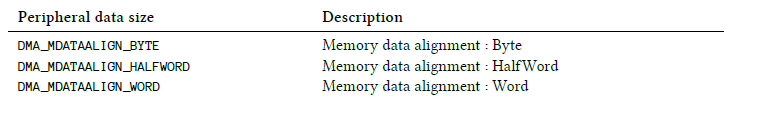


**DMA Terimler:**

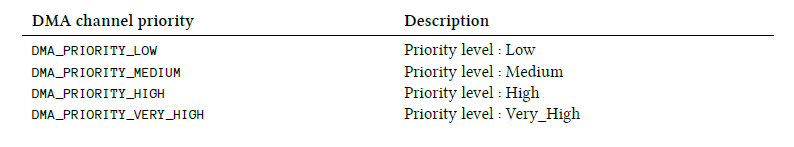
* **PeripheralBaseAddr:** Çevre biriminde verinin adresini tutar. Farklı çevre birimleri farklı adreslere sahiptir.
* **MemoryBaseAddr:** Verinin hafıza da yazıldığı kısımdır. Hedef adrestir.
* **DIR (Direction):** DMA transfer yönünü tanımlar.
* **PeripheralInc:** Çevre biriminde verinin yazılacağı adresi arttırmak için kullanılır. Uygulamalarda genellikle DISABLE olarak kalacaktır.
* **MemoryInc:** Çevre birimindenalınanher bir değer için hafızadaki adresi arttırmak için kullanılır.
* **PeripheralDataSize:** Kaynaktan alınacak verinin boyutunu tutar.



* **MemoryDataSize:** Hafızada gelen verinin saklanacağı boyutu tutar.



* **Mode:** Veri alındıktan sonra işlemin sonlanacağını veya tekrar edeceğini belirler.
* **Priority:** Kanal önceliğini belirler.



* **M2M:** Hafızadan hafızaya transferin yapılıp yapılmayacağını belirtir. Genellikle DISABLE olarak seçilir.
* **BufferSize:** Alacağımız veri sayısını belirtir. Örnek, 3 adet adc kanalı okuyacaksak 3 girilmesi gerekir.

**Not:**

DMA\_Init(DMA1\_Channel1, &DMAInitStructure);

DMA\_Cmd(DMA1\_Channel1,ENABLE);