DMA原理介绍

不撑了不撑了

DMA的基本定义

DMA，全称Direct Memory Access，即直接存储器访问。

DMA传输将数据从一个地址空间复制到另一个地址空间，提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。当CPU初始化这个传输动作，传输动作本身是由DMA控制器来实现和完成的。DMA传输方式无需CPU直接控制传输，也没有中断处理方式那样保留现场和恢复现场过程，通过硬件为RAM和IO设备开辟一条直接传输数据的通道，使得CPU的效率大大提高。

DMA的主要特征

·每个通道都直接连接专用的硬件DMA请求，每个通道都同样支持软件触发，这些功能通过软件来配置。

·在同一个DMA模块上，多个请求间的优先权可以通过软件编程设置（共有四级：很高、高、中等和低），优先权设置相等时由硬件决定（请求0优先于请求1，依此类推）。

·独立数据源和目标数据区的传输宽度（字节、半字、全字），模拟打包和拆包的过程。源和目标地址必须按数据传输宽度对齐。

·支持循环的缓冲器管理。

·每个通道都有3个事件标志（DMA半传输、DMA传输完成和DMA传输出错），这3个事件标志逻辑或成为一个单独的中断请求。

·存储器和存储器间的传输、外设和存储器、存储器和外设之间的传输。

·闪存、SRAM、外设的SRAM、APB1、APB2和AHB外设均可作为访问的源和目标。

·可编程的数据传输数目：最大为65535（0xFFFF）。

STM32F411x系列芯片DMA控制器

DMA的工作框图如下图所示。DMA控制器和Cortex™-M4核心共享系统数据总线，执行直接存储器数据传输。当CPU和DMA同时访问相同的目标（RAM或外设）时，DMA请求会暂停CPU访问系统总线达若干个周期，总线仲裁器执行循环调度，以保证CPU至少可以得到一半的系统总线（存储器或外设）带宽。

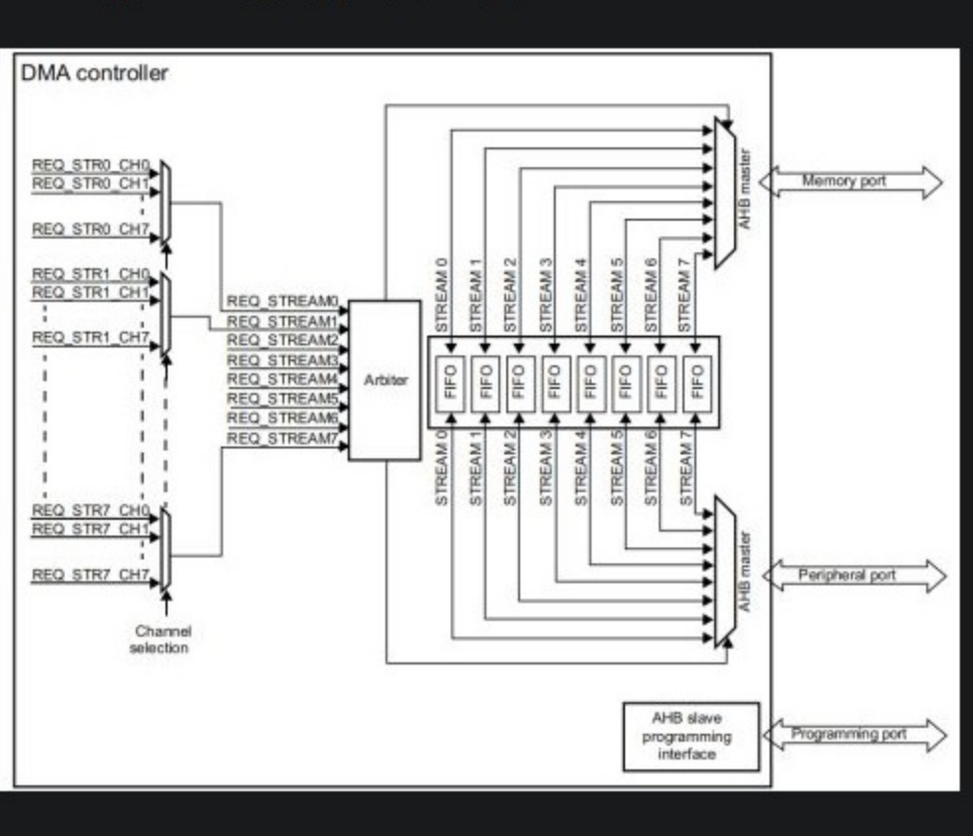
DMA控制器传输作为AHB主设备操作直接存储器，它可以控制AHB总线的控制矩阵以启动AHB传送。它可以执行以下信息交换：

•外设到内存

•内存到外设

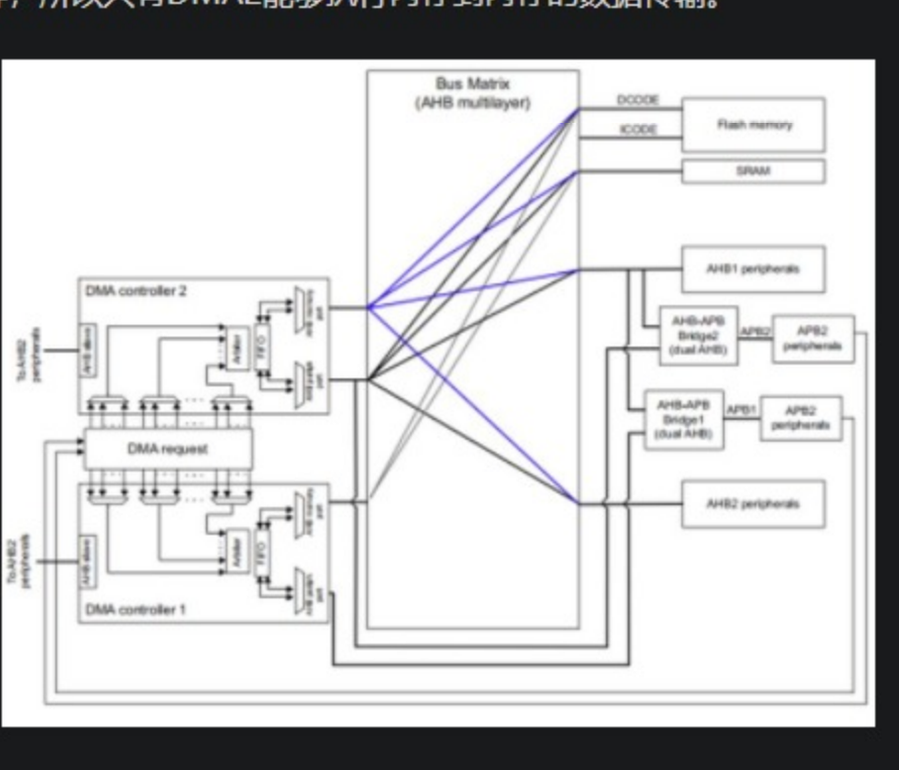
•内存到内存

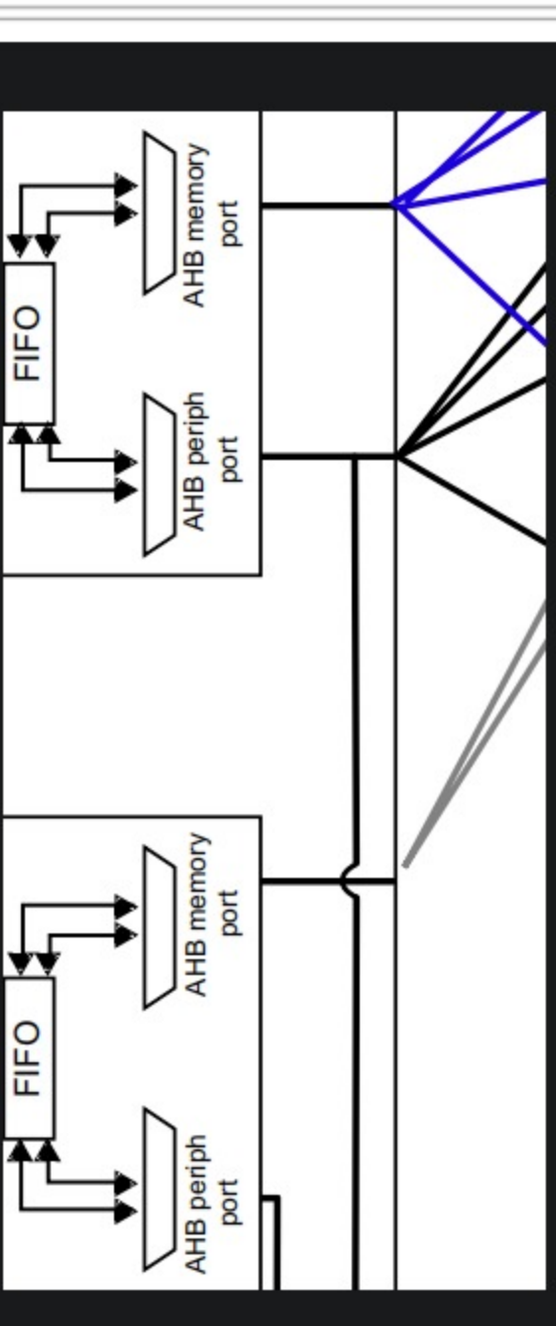
DMA控制器提供两个AHB主端口：AHB内存端口（用于连接到内存）和AHB外设端口（用于连接到外设）。但是，为了允许内存到内存的传输，AHB外设端口也必须可以访问内存。AHB从端口用于对DMA控制器的编程控制（仅支持32位访问）。



DMA处理

对于两个DMA控制器的系统实现如下图所示。DMA1控制器AHB外设端口没有像DMA2控制器那样连接到总线矩阵，所以只有DMA2能够执行内存到内存的数据传输。





DMA事务由给定数量的数据传输序列组成，可以传输的数据项的数量及其宽度（8位，16位或32位）可以通过软件编程实现。每个DMA传输都包含三个操作步骤：

·从外设数据寄存器或者从当前外设/存储器地址寄存器指示的存储器地址取数据，第一次传输时的开始地址是DMA_CPARx或DMA_CMARx寄存器指定的外设基地址或存储器单元；

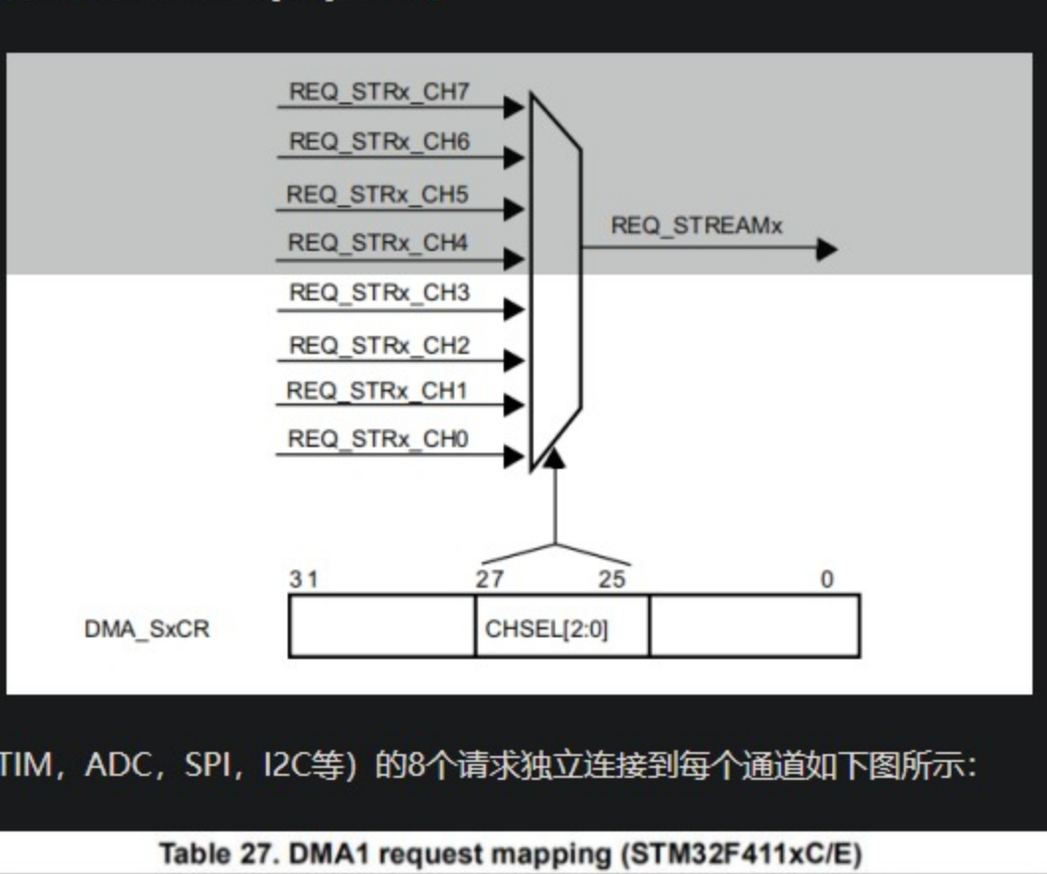
·存数据到外设数据寄存器或者当前外设/存储器地址寄存器指示的存储器地址，第一次传输时的开始地址是DMA_CPARx或DMA_CMARx寄存器指定的外设基地址或存储器单元；

·执行一次DMA_CNDTRx寄存器的递减操作，该寄存器包含未完成的操作数目。

通道选择

事件发生后，外设向DMA控制器发送请求信号，DMA控制器根据通道优先级来处理请求。当DMA控制器访问外围设备时，确认信号即由DMA控制器发送到外围设备，一旦外围设备从DMA控制器收到确认信号，它就会释放其请求，当外设取消了该请求后，DMA控制器将释放确认信号。如果外设还有更多请求，它可以启动下一轮请求操作。

每个数据流流都与一个DMA请求相关，该DMA请求可以从8个可能的通道请求中选择，由DMA_SxCR寄存器中的CHSEL [2:0]位控制。



来自外设（TIM，ADC，SPI，I2C等）的8个请求独立连接到每个通道如下图所示：

Table 27. DMA1 request mapping (STM32F411xC/E)

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX	I2C1_TX	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
Channel 1	I2C1_RX	I2C3_RX	-	-	-	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Channel 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	-	-	-	-	-	USART2_RX	USART2_TX	-
Channel 5	-	-	TIM3_CH4 TIM3_UP	-	TIM3_CH1 TIM3_TRIG	TIM3_CH2	-	TIM3_CH3
Channel 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	I2C3_TX	TIM5_UP	USART2_RX
Channel 7	-	-	I2C2_RX	I2C2_RX	-	-	-	I2C2_TX

Table 28. DMA2 request mapping (STM32F411xC/E)

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	ADC1	-	-	-	ADC1	-	TIM1_CH1 TIM1_CH2 TIM1_CH3	-
Channel 1	-	-	-	-	-	-	-	-
Channel 2	-	-	SPI1_TX	SPI5_RX	SPI5_TX	-	-	-
Channel 3	SPI1_RX	-	SPI1_RX	SPI1_TX	-	SPI1_TX	-	-
Channel 4	SPI4_RX	SPI4_TX	USART1_RX	SDIO	SPI4_RX	USART1_RX	SDIO	USART1_TX
Channel 5	-	USART6_RX	USART6_RX	SPI4_RX	SPI4_TX	SPI5_TX	USART6_TX	USART6_TX
Channel 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
Channel 7	-	-	-	-	-	SPI5_RX	SPI5_TX	-

仲裁器

仲裁器根据通道请求的优先级来启动外设/存储器的访问。优先权管理分2个阶段：

·软件：每个通道的优先权可以在DMA_CCRx寄存器中设置，有4个等级：最高优先级、高优先级、中等优先级、低优先级；

·硬件：如果2个请求有相同的软件优先级，则较低编号的通道比较高编号的通道有较高的优先权。比如：如果软件优先级相同，通道2优先于通道4。

注意：在大容量产品和互联型产品中，DMA1控制器拥有高于DMA2控制器的优先级。

DMA通道数据量

每个通道都可以在有固定地址的外设寄存器和存储器地址之间执行DMA传输。DMA传输的数据量是可编程的，最大达到65535（0xFFFF）。包含要传输的数据项数量的寄存器，在每次传输后递减。外设和存储器的传输数据量可以通过DMA_CCRx寄存器中的PSIZE和MSIZE位编程得到。

DMA传输模式

数据传输源和目的地都可以寻址整个4 GB区域中的外围设备和存储器，其地址介于0x0000 0000和0xFFFF FFFF之间。传输方向使用DMA_SxCR寄存器中的DIR [1:0]位进行配置，并提供三种可能性：存储器到外围设备，外围设备到存储器或存储器到存储器的传输。

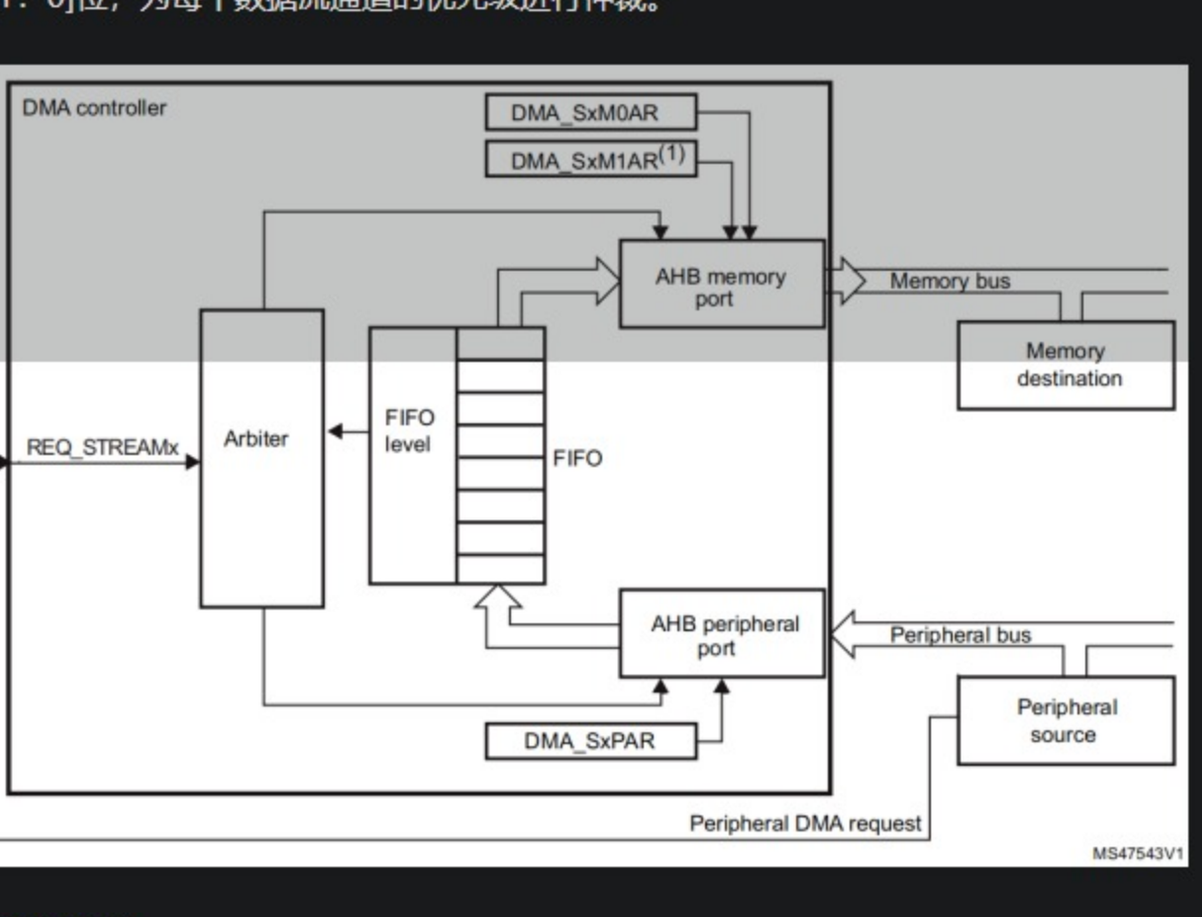
Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR
01	Memory-to-peripheral	DMA_SxM0AR	DMA_SxPAR
10	Memory-to-memory	DMA_SxPAR	DMA_SxM0AR
11	reserved	-	-

外设到存储器模式

使能此模式时（通过将DMA_SxCR寄存器中的EN位置1），每次发生外设请求时，数据流都会启动传输从数据源来填充到FIFO。当达到FIFO的阈值水平时，FIFO的内容被清空并存储到目标地址。

当外设请求结束传输时（对于外设流控制器），或当DMA_SxNDTR寄存器达到零时或将DMA_SxNDTR寄存器中的EN位清零，则传输停止。

当赢得了相应流的仲裁时，该数据流通道才可以访问AHB源或目标端口，使用DMA_SxCR寄存器中的PL [1：0]位，为每个数据流通道的优先级进行仲裁。

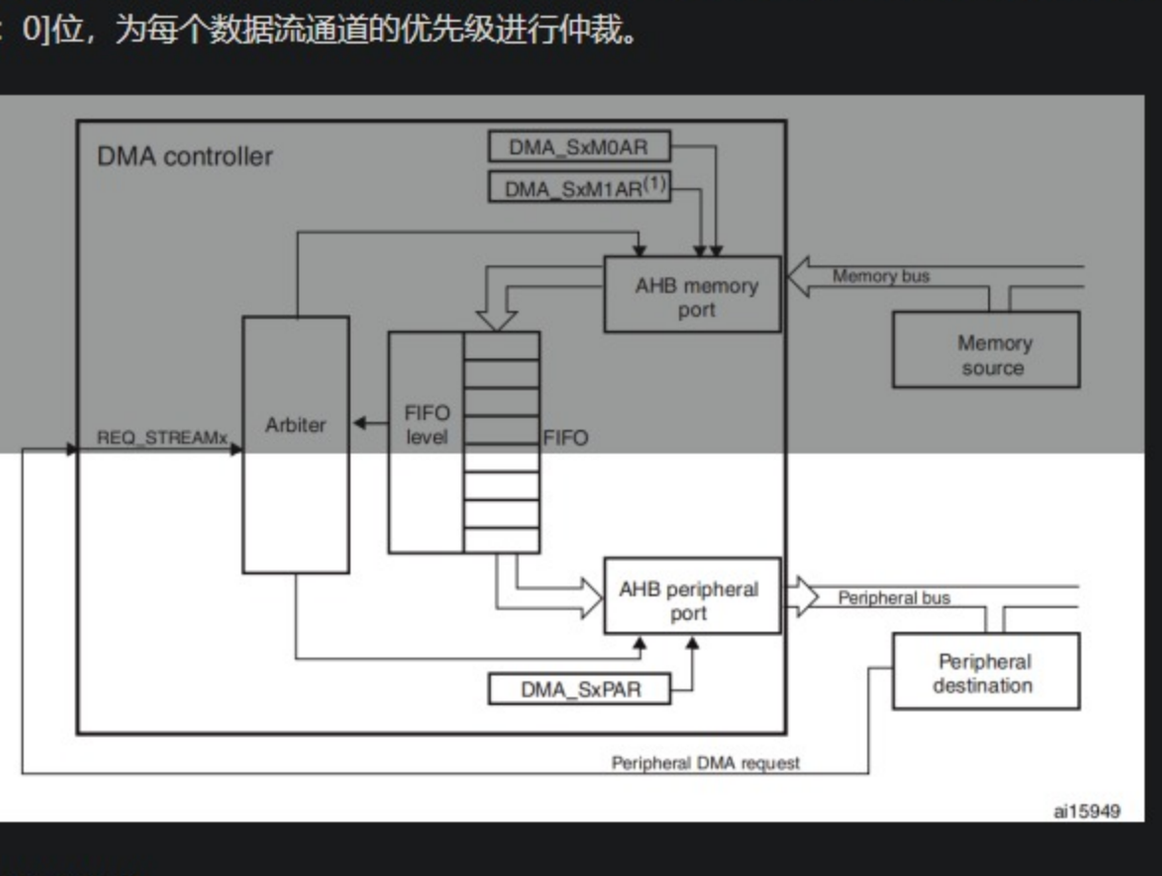


存储器到外设模式

使能该模式时（通过将DMA_SxCR寄存器中的EN位置1），该数据流通道立即启动传输，以完全填充FIFO。每次发生外设请求时，FIFO的内容都会被清空并存储到目的地。当FIFO未满时，会从内存中重新加载数据。

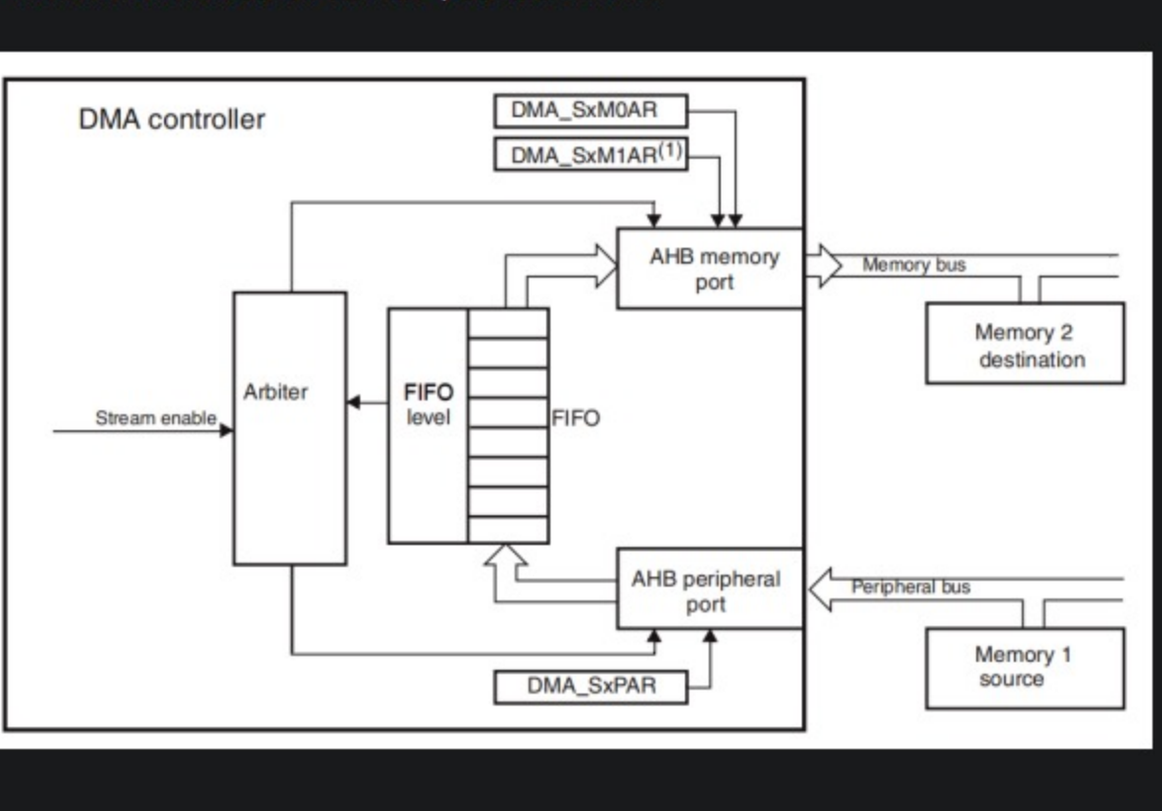
当外设请求结束传输时（对于外设流控制器），或者当DMA_SxNDTR寄存器达到零时或将DMA_SxNDTR寄存器中的EN位清零，则传输停止。

当赢得了相应流的仲裁时，该数据流通道才可以访问AHB源或目标端口，使用DMA_SxCR寄存器中的PL [1：0]位，为每个数据流通道的优先级进行仲裁。



存储器到存储器模式

配置同上。存储器到存储器模式不能与循环模式同时使用。



循环模式

循环模式可用于处理循环缓冲区和连续数据流（例如ADC扫描模式）。可以使用DMA_SxCR寄存器中的CIRC位来启用此功能。使能循环模式后，将在数据流通道配置阶段使用初始值自动装载要传输的数据，并且DMA请求将继续。

中断

每个DMA通道都可以在DMA传输过半、传输完成和传输错误时产生中断。为应用的灵活性考虑，通过设置寄存器的不同位来打开这些中断。

对于每个DMA数据流通道，可以在以下事件上产生中断：

•达到半转移

•转移完成

•传输错误

•FIFO错误（溢出，欠载或FIFO内存错误）

•直接模式错误

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE
FIFO overrun/underrun	FEIF	FEIE
Direct mode error	DMEIF	DMEIE

使用STM32CubeMX配置UART的DMA使能，得到以下初始化代码：

```
/* USART2 DMA Init */
/* USART2_RX Init */
hdma_usart2_rx.Instance = DMA1_Stream5;
hdma_usart2_rx.Init.Channel = DMA_CHANNEL_4;
hdma_usart2_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
hdma_usart2_rx.Init.PeriphInc = DMA_PINC_DISABLE;
hdma_usart2_rx.Init.MemInc = DMA_MINC_ENABLE;
hdma_usart2_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
hdma_usart2_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
hdma_usart2_rx.Init.Mode = DMA_CIRCULAR;
hdma_usart2_rx.Init.Priority = DMA_PRIORITY_LOW;
hdma_usart2_rx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
if (HAL_DMA_Init(&hdma_usart2_rx) != HAL_OK)
{
 Error_Handler();
}
__HAL_LINKDMA(uartHandle,hdmarx,hdma_usart2_rx);

/* USART2_TX Init */
hdma_usart2_tx.Instance = DMA1_Stream6;
hdma_usart2_tx.Init.Channel = DMA_CHANNEL_4;
hdma_usart2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
hdma_usart2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
hdma_usart2_tx.Init.MemInc = DMA_MINC_ENABLE;
hdma_usart2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
hdma_usart2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
hdma_usart2_tx.Init.Mode = DMA_NORMAL;
hdma_usart2_tx.Init.Priority = DMA_PRIORITY_LOW;
hdma_usart2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
if (HAL_DMA_Init(&hdma_usart2_tx) != HAL_OK)
{
  Error_Handler();
}
__HAL_LINKDMA(uartHandle,hdmatx,hdma_usart2_tx);
```

下面是配置DMA通道x的普遍过程（x代表通道号）：

·在DMA_CPARx寄存器中设置外设寄存器的地址。发生外设数据传输请求时，这个地址将是数据传输的源或目标；

·在DMA_CMARx寄存器中设置数据存储器的地址。发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址；

·在DMA_CNDTRx寄存器中设置要传输的数据量。在每个数据传输后，这个数值递减；

·在DMA_CCRx寄存器的PL[1:0]位中设置通道的优先级；

·在DMA_CCRx寄存器中设置数据传输的方向、循环模式、外设和存储器的增量模式、外设和存储器的数据宽度、传输一半产生中断或传输完成产生中断；

·设置DMA_CCRx寄存器的ENABLE位，启动该通道。

一旦启动了DMA通道，它既可响应连到该通道上的外设的DMA请求。当传输一半的数据后，半传输标志(HTIF)被置1，当设置了允许半传输中断位(HTIE)时，将产生一个中断请求。在数据传输结束后，传输完成标志(TCIF)被置1，当设置了允许传输完成中断位(TCIE)时，将产生一个中断请求。

编辑于 2020-05-06