AN4031
应用笔记

使用 STM32F2 和 STM32F4 DMA 控制器

前言

本应用笔记对如何使用 STM32F2xx 和 STM32F4xx 系列直接存储器访问控制器 （DMA）进行了说明。 STM32F2xx/F4xx DMA 控制器所具有的系统架构、多层总线矩阵和存储系统等，使其能够为应用提供很高的数据带宽，让用户可以开发出响应迅速的应用软件。

本应用笔记还提供了一些使用DMA的技巧，以便开发者能够充分利用STM32F2和STM32F4 DMA 的特性来为不同的外设和子系统的 DMA 请求保证足够快的响应时间。

本文档中，使用 "STM32F2/F4 设备 " 代替 STM32F2xx 和 STM32F4xx，使用 “DMA” 代替 DMA 控制器。

本应用笔记适用于*表 1* 中所列产品。

表 1. 适用产品

类型	产品料号
微控制器	STM32F2xx (STM32F205, STM32F207, STM32F215, STM32F217) STM32F4xx (STM32F401, STM32F405, STM32F407, STM32F415, STM32F417, STM32F427, STM32F429, STM32F437, STM32F439)

本应用笔记应与 STM32F2/F4 参考手册 （RM0031、 RM0090 和 RM0368）共同阅读使用。

目录

表格索引

图片索引

1 DMA 控制器说明

DMA 是一种 AMBA 先进高性能总线 （AHB）模块，它具有三个 AHB 端口：1 个用于 DMA 编程的从端口和 2 个允许 DMA 在从模块之间进行数据传输的主端口（外设和存储器端口）。

DMA 使得数据传输在后台进行，而无需 Cortex-Mx 处理器干预。在数据传输过程中，主处理器能够执行其他任务，仅当需要处理一个完整数据块时才会被中断。

可以在不显著影响系统性能的情况下进行大量数据的传输。DMA 主要用于实现不同外设模块的集中数据缓冲存储 （通常在 SRAM 中）。在分布式方案中，每个外设均需实现其各自的本地数据存储，该解决方案从硅片面积的使用和功耗方面来说是比较便宜的。

STM32F2/F4 DMA 控制器充分利用了 Cortex-Mx 哈佛结构和多层总线系统的优势，保证了 DMA 传输和 CPU 中断响应的及时性。

1.1 DMA 传输属性

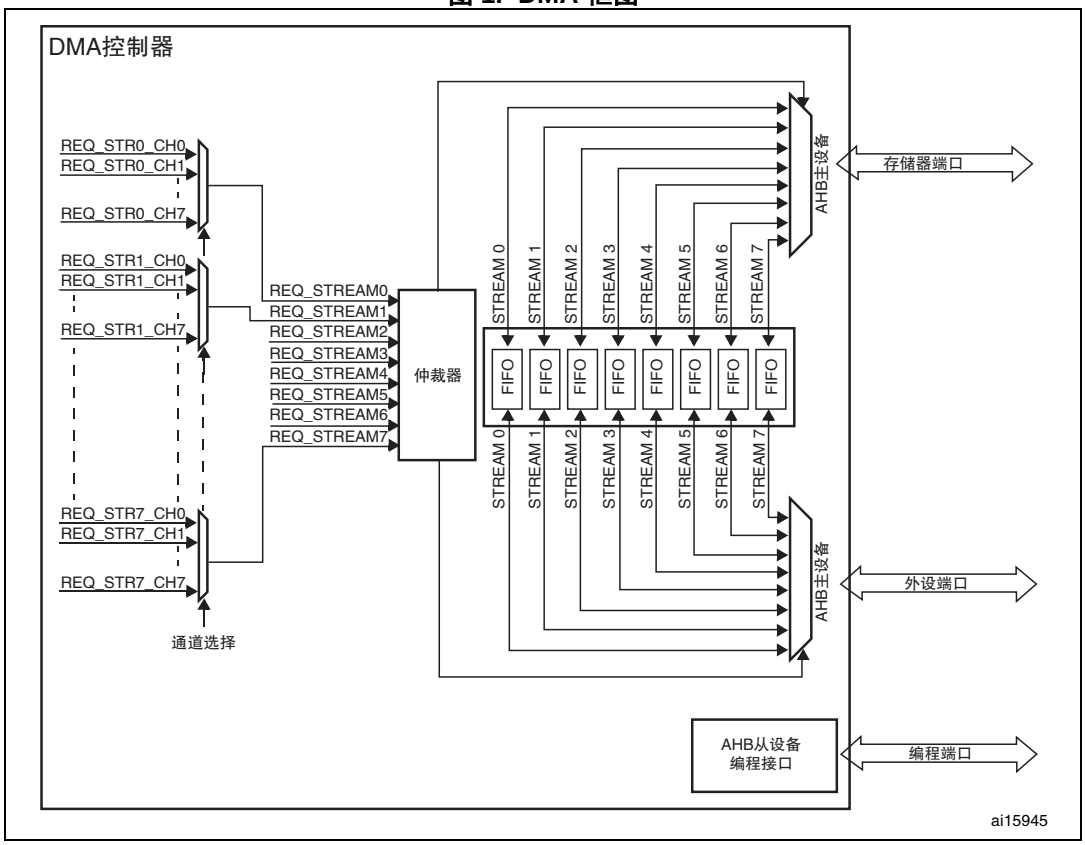
DMA 传输的属性如下：

- DMA 数据流 / 通道
- 数据流优先级
- 源和目标地址
- 传输模式
- 传输数据量大小 （仅当 DMA 为流量控制器时）
- 源 / 目标地址递增或非增
- 源和目标数据宽度
- 传输类型
- FIFO 模式
- 源 / 目标批量传输数据量大小
- 双缓冲区模式
- 流控

STM32F2/F4 器件集成了 2 个 DMA 控制器，每个 DMA 有两个端口，一个外设端口和一个存储器端口，它们可以同时工作。

图 1 显示了 DMA 框图。

图 1. DMA 框图



下面的章节对 DMA 的每个传输属性进行详细描述。

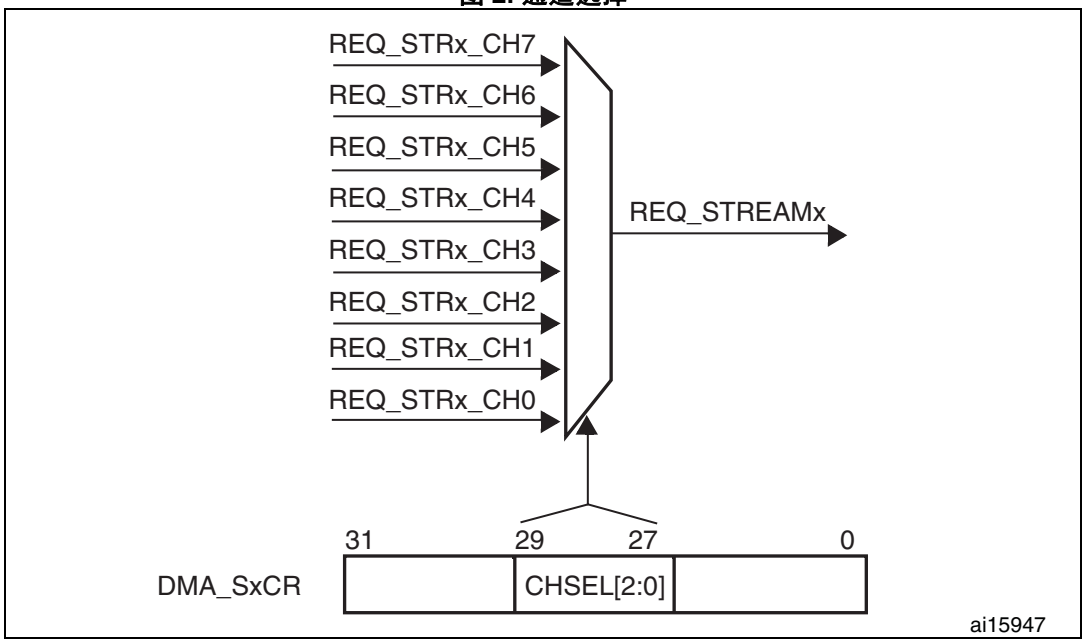
1.1.1 DMA 数据流 / 通道

STM32F2/F4 器件集成了 2 个 DMA 控制器，提供了共计多达 16 个数据流 （每个控制器 8 个），每个数据流都可用来处理来自一个或多个外设的存储器访问请求。

每个数据流共有多达 8 个可选通道 （请求），可由软件配置，允许多个外设启动 DMA 请求。

图 2 描述了某个特定数据流的通道选择。

图 2. 通道选择



注： *在一个数据流中，同一时刻仅有一个通道 / 请求是有效的。*

多个使能的 DMA 数据流不能为同一个外设 DMA 请求服务。

表 2 和*表 3* 描述了 DMA 数据流 / 通道对应的外设请求 （面向除 STM32F401 之外的所有产品）， STM32F401 产品的请求配置见*表 4* 和*表 5*。

表 2. DMA1 请求映射

外设请求	数据流 0	数据流 1	数据流 2	数据流 3	数据流 4	数据流 5	数据流 6	数据流 7
通道 0	SPI3_RX	-	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
通道 1	I2C1_RX	-	TIM7_UP		TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
通道 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
通道 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
通道 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
通道 5	UART8_TX(1)	UART7_TX(1)	TIM3_CH4 TIM3_UP	UART7_RX(1)	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX(1)	TIM3_CH3
通道 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	-	TIM5_UP	-
通道 7	-	TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

1. 这些请求仅适用于 STM32F42xx 和 STM32F43xx。

表 3. DMA2 请求映射

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

1. 这些请求仅适用于 STM32F42xx 和 STM32F43xx。

表 4 和*表 5* 描述了 STM32F401 产品的 DMA 数据流 / 通道对应的外设请求。

表 4. STM32F401 产品的 DMA1 请求映射

外设请求	数据流 0	数据流 1	数据流 2	数据流 3	数据流 4	数据流 5	数据流 6	数据流 7
通道 0	SPI3_RX	-	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
通道 1	I2C1_RX	I2C3_RX	-	-	-	I2C1_RX	I2C1_TX	I2C1_TX
通道 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
通道 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
通道 4	-	-	-	-	-	USART2_RX	USART2_TX	-
通道 5	-	-	TIM3_CH4 TIM3_UP	-	TIM3_CH1 TIM3_TRIG	TIM3_CH2	-	TIM3_CH3
通道 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	I2C3_TX	TIM5_UP	-
通道 7	-	-	I2C2_RX	I2C2_RX	-	-	-	I2C2_TX

表 5. STM32F401 产品的 DMA2 请求映射

外设请求	数据流 0	数据流 1	数据流 2	数据流 3	数据流 4	数据流 5	数据流 6	数据流 7
通道 0	ADC1	-	-	-	ADC1	-	TIM1_CH1 TIM1_CH2 TIM1_CH3	-
通道 1	-	-	-	-	-	-	-	-
通道 2	-	-	-	-	-	-	-	-
通道 3	SPI1_RX	-	SPI1_RX	SPI1_TX	-	SPI1_TX	-	-
通道 4	SPI4_RX	SPI4_TX	USART1_RX	SDIO	-	USART1_RX	SDIO	USART1_TX
通道 5	-	USART6_RX	USART6_RX	SPI4_RX	SPI4_TX	-	USART6_TX	USART6_TX
通道 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
通道 7	-	-	-	-	-	-	-	-

STM32F2/F4 DMA 请求映射经过了精心设计，软件应用可以更加灵活地为关联的外设请求配置 DMA 传输通道，并且通过复用相应的 DMA 数据流和通道可覆盖大部分应用。

1.1.2 数据流优先级

每个 DMA 端口都有一个仲裁器来处理 DMA 数据流之间的优先级。数据流优先级可由软件配置（共有 4 个软件级别）。如果两个或更多的 DMA 数据流具有同样的软件优先级别，则使用硬件优先级 （数据流 0 比数据流 1 优先级高，以此类推）。

1.1.3 源和目标地址

DMA 传输由源地址和目标地址决定。源和目标 （地址）均应当在 AHB 或 APB 存储范围之内，并且与传输宽度保持一致。

1.1.4 传输模式

DMA 能够实现 3 种不同的传输模式：

- 外设到存储器，
- 存储器到外设，
- 存储器到存储器 （仅 DMA2 能够实现该传输，在这种模式下，禁用循环和直接模式。）

1.1.5 传输数据量大小

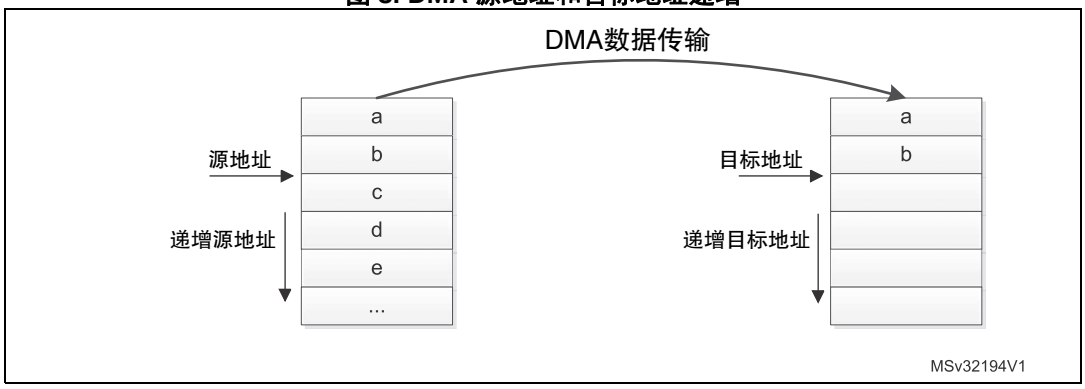
仅当 DMA 为流量控制器时，才必须定义其传输数据量大小。实际上，传输数据量值决定了从数据源到目标传输的数据总量。

传输数据量大小由 DMA_SxNDTR 寄存器值和外设端的数据传输宽度决定。随着已接收的请求 （批量或单个的），传输数据总量减少，减少量为已传输数据量。

1.1.6 递增源 / 目标地址

可以将 DMA 配置为每次数据传输之后自动递增源和 / 或目标地址。

图 3. DMA 源地址和目标地址递增



1.1.7 源和目标数据宽度

源和目标数据宽度定义如下：

- 字节 （8 位）
- 半字 （16 位）
- 字 （32 位）

1.1.8 传输类型

- 循环模式：循环模式能够管理循环缓冲区和连续数据流 （DMA_SxNDTR 寄存器自减到 0 后会自动重载预先设置的值）。
- 正常模式：一旦 DMA_SxNDTR 寄存器自减到零，传输即会停止 （DMA_SxCR 寄存器中的 EN 位等于 0）。

1.1.9 DMA FIFO 模式

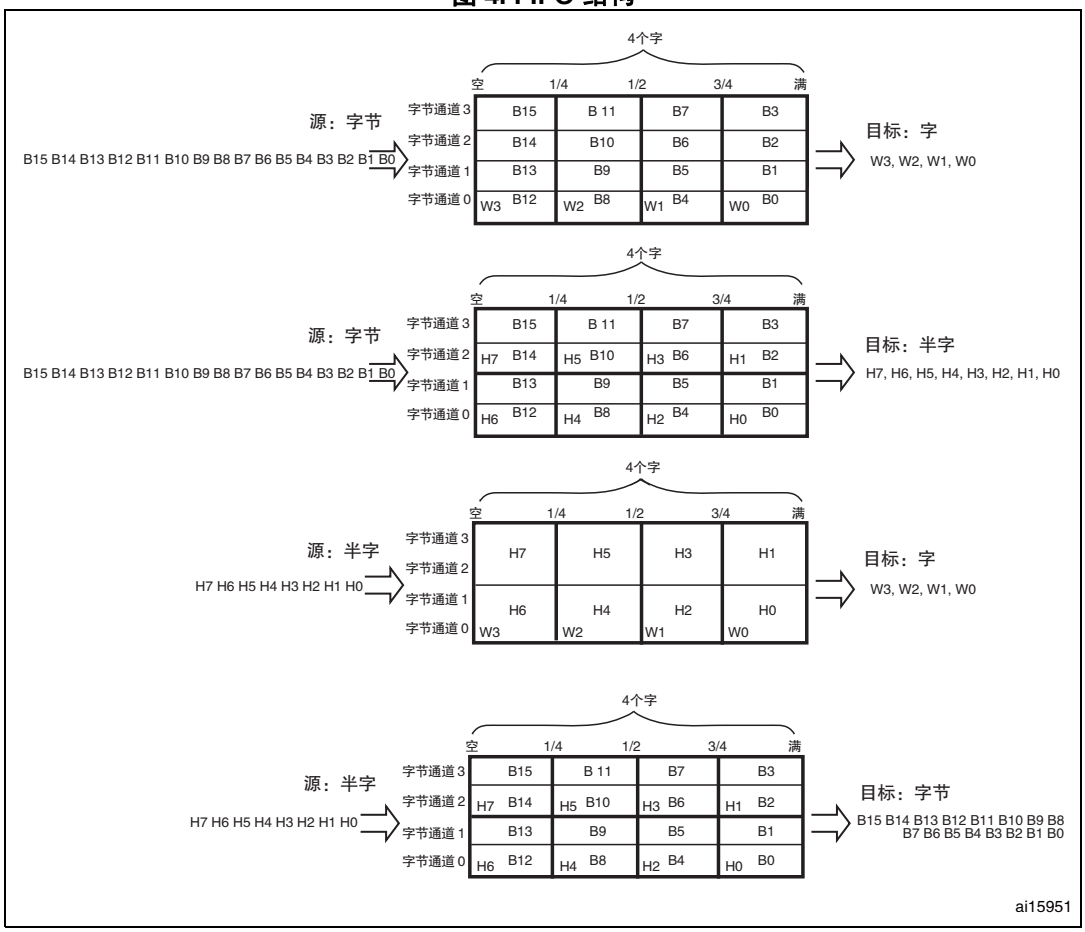
每个数据流都有一个独立的 4 字（4 * 32 位）FIFO，阈值级别可由软件配置为 1/4、1/2、3/4 或满。 FIFO 用于在源数据传输到目标之前临时存储这些数据。

DMA FIFO 可通过软件使能或禁用；禁用时，使用直接模式。如果使能了 DMA FIFO，则数据打包 / 拆包以及批量可用。 DMA FIFO 阈值决定了 DMA 存储器端口请求时间。

STM32F2/F4 器件上的 DMA FIFO 能够：

- 减少 SRAM 存取次数，因此可为其他主设备访问总线矩阵提供更多时间而不会有额外的并发访问请求，
- 允许软件处理批量传输，从而优化传输带宽，
- 允许对数据进行打包和拆包，以适应源和目标数据宽度的不同，而无需额外 DMA 存取。

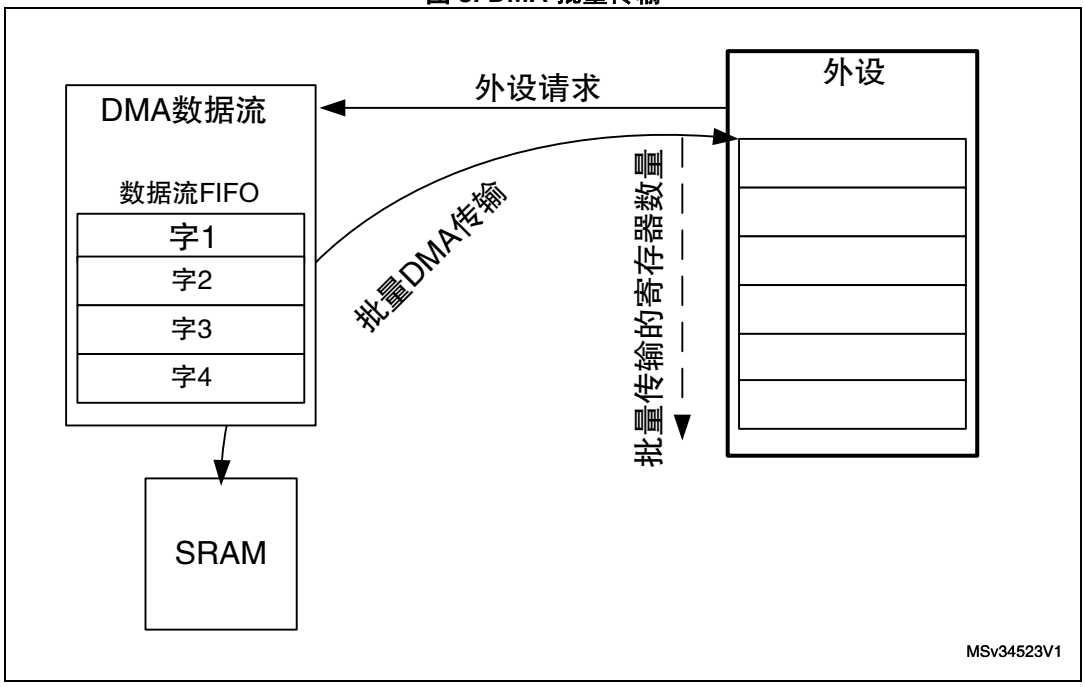
图 4. FIFO 结构



1.1.10 源和目标大小

DMA FIFO 的实现保证了批量传输 （正常进行）。

图 5. DMA 批量传输



作为对外设批量传输请求的响应，DMA 读 / 写批量传输数据量（4x、8x 或 16x 个数据单元，数据单元可以是字、半字或字节）对应的数据单元的数目。DMA 外设端口的批量传输数据量大小需根据外设需求 / 能力来设置。

存储器端口的 DMA 批量传输数据量大小和 FIFO 阈值配置必须匹配。在存储器端口上进行批量传输时， FIFO 中要有足够的数据。*表 6* 描述了存储器批量传输数据量、 FIFO 阈值配置和数据量大小的可能组合。

为了保持数据连贯性，批量传输中每次传输是不会把打断的 AHB 传输锁定，AHB 总线矩阵仲裁器在批量传输过程中不会剥夺 DMA 控制器这个主设备的访问权。

表 6. 可能的批量传输配置

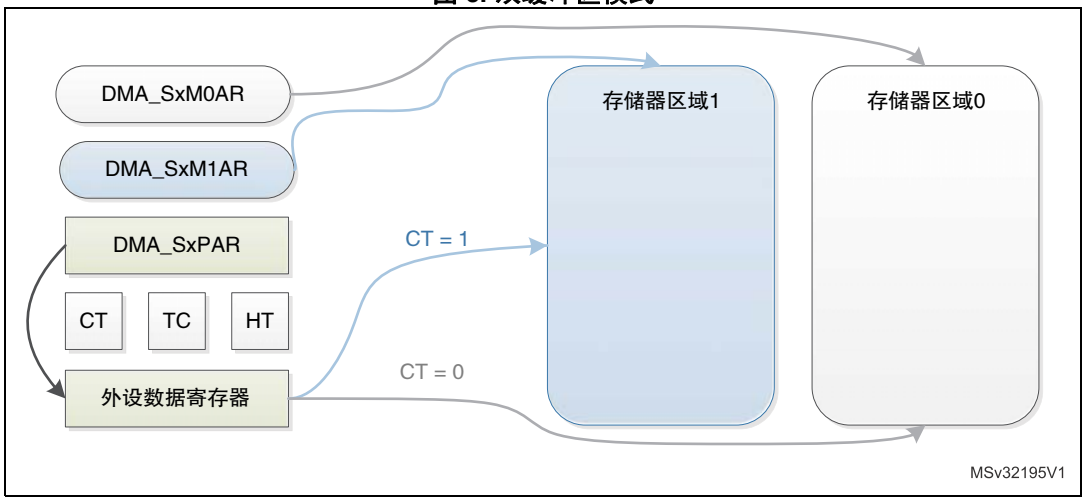
MSIZE	FIFO 级别	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16
字节	1/4	4 个字节的 1 次批量传输	禁止	禁止
	1/2	4 个字节的 2 次批量传输	8 个字节的 1 次批量传输	
	3/4	4 个字节的 3 次批量传输	禁止	
	满	4 个字节的 4 次批量传输	8 个字节的 2 次批量传输	16 个字节的 1 次批量传输
半字	1/4	禁止	禁止	禁止
	1/2	4 个半字的 1 次批量传输		
	3/4	禁止		
	满	4 个半字的 2 次批量传输	8 个半字的 1 次批量传输	
字	1/4	禁止	禁止	
	1/2			
	3/4			
	满	4 个字的 1 次批量传输		

1.1.11 双缓冲区模式

除了有两个存储器指针之外，双缓冲区数据流的工作方式与常规（单缓冲区）数据流的一样。启用双缓冲区模式时，循环模式将自动启用，存储器指针在传输事务结束 （DMA_SxNDTR 寄存器达到 0）时切换。

这样，软件在处理一个存储器区域的同时， DMA 传输还可以填充 / 使用第二个存储器区域。

图 6. 双缓冲区模式



在双缓冲区模式下，即使数据流使能时，也可更新 AHB 存储端口的基址（DMA_SxM0AR 或 DMA_SxM1AR）：

- 当 DMA_SxCR 寄存器的 CT（Current Target，当前目标）位等于 0 时，当前 DMA 传输的存储目标是存储区域 0，因而存储区域 1 （DMA_SxM1AR）的基址可被更新。
- 当DMA_SxCR寄存器的CT位等于1时，当前DMA传输的存储目标在存储区域1，因而存储区域 0 （DMA_SxM0AR）的基址可被更新。

1.1.12 流控

流量控制器是控制数据传输长度的单元，能够控制 DMA 传输停止。

流量控制器可以是 DMA 或外设。

- 使用 DMA 作为流量控制器：

这种情况下，使能 DMA 数据流之前，需要在 DMA_SxNDTR 寄存器中定义传输数据量。响应 DMA 请求时，传输数据量值减少，减少量为已传输的数据量 （取决于请求类型：突发或单个）。

当传输数据量值自减到 0 时， DMA 传输完成， DMA 数据流停止。

- 使用外设作为流量控制器：

这种情况下待传输的数据项的数目是未知的。当所传输的是最后的数据时，外设通过硬件向 DMA 控制器发出指示。只有 SD/MMC 外设支持这种模式。

1.2 设置 DMA 传输

可使用下面的过程配置 DMA 数据流 x （这里的 x 为数据流编号）：

1. 如果使能了数据流，通过复位 DMA_SxCR 寄存器中的 EN 位将其禁止，然后读取此位以确认没有正在进行的数据流操作。将此位写 0 不会立即生效，因为实际上只有所有当前传输都已完成时才会真正变为 0。当所读取 EN 位的值为 0 时，才表示可以配置数据流了。因此在开始任何数据流配置之前，需要等待 EN 位清 0。应将先前的数据块 DMA

传输中在状态寄存器（DMA_LISR 和 DMA_HISR）中置 1 的所有数据流专用的位清 0，然后才可重新使能数据流。

2. 在 DMA_SxPAR 寄存器中设置外设端口寄存器地址。外设事件发生后，数据会从此地址传输到外设端口或从外设端口传输到此地址。
3. 在 DMA_SxMA0R 寄存器 （在双缓冲区模式的情况下还有 DMA_SxMA1R 寄存器）中设置存储器地址。外设事件发生后，将从此存储器地址读取数据或将数据写入此存储器地址。
4. 在 DMA_SxNDTR 寄存器中配置要传输的数据项的总数。每出现一次外设事件或每出现一个节拍的批量传输，该值都会递减。
5. 使用 DMA_SxCR 寄存器中的 CHSEL[2:0] 选择 DMA 通道 （请求）。
6. 如果外设用作流控制器而且支持此功能，请将 DMA_SxCR 寄存器中的 PFCTRL 位置 1。
7. 使用 DMA_SxCR 寄存器中的 PL[1:0] 位配置数据流优先级。
8. 配置 FIFO 的使用情况 （使能或禁止，以及发送和接收阈值）。
9. 在 DMA_SxCR 寄存器中配置数据传输方向、外设和存储器增量 / 固定模式、单次或批量传输、外设和存储器数据宽度、循环模式、双缓冲区模式和传输完成一半和 / 或全部完成，和 / 或错误中断。
10. 通过将 DMA_SxCR 寄存器中的 EN 位置 1 使能数据流。

一旦使能了数据流，即可响应连接到该数据流的外设发出的任何 DMA 请求。

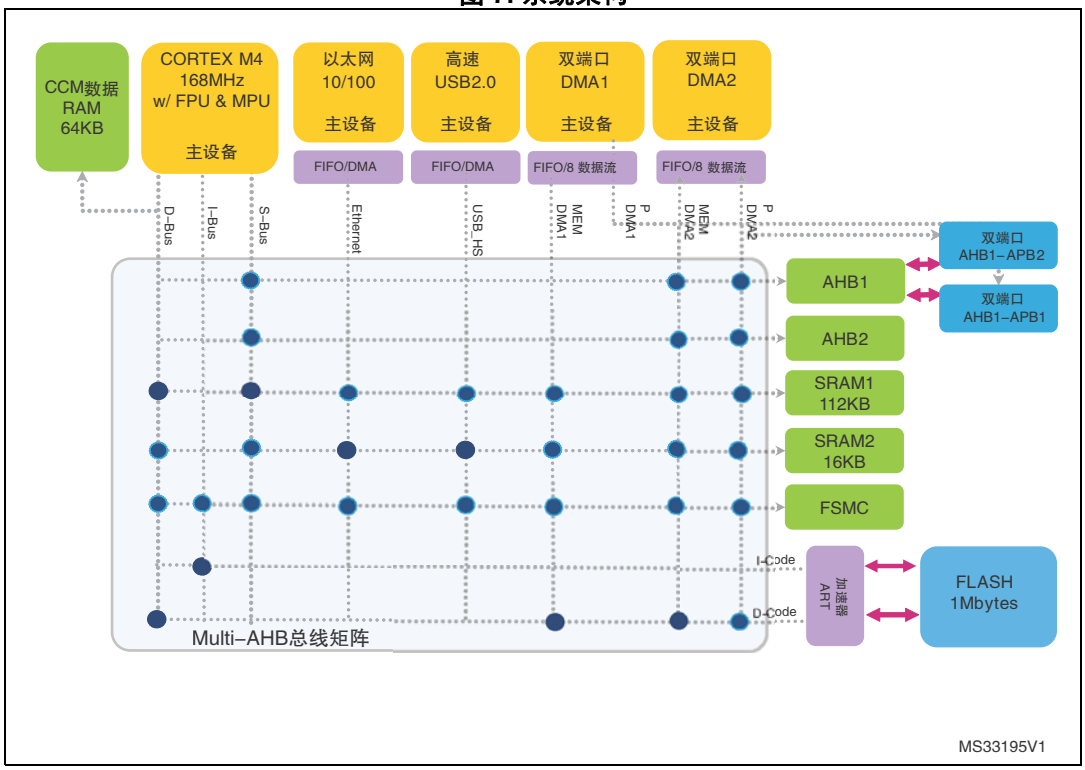
2 系统性能考虑

STM32F2/F4 器件集成了多主 / 多从架构：

- 八个主设备：
  - Cortex™-Mx 内核 I-bus
  - Cortex™-Mx 内核 D-bus
  - Cortex™-Mx 内核 S-bus
  - DMA1 存储器端口总线
  - DMA2 存储器端口总线
  - DMA2 外设端口总线
  - 以太网专用 DMA 总线
  - 高速 USB 专用 DMA 总线
- 八个从设备：
  - 内部 Flash ICode 总线
  - 内部 Flash DCode 总线
  - 内部主 SRAM1 （112KB， STM32F401x 上是 64KB）
  - 内部辅 SRAM2 （16 KB）（STM32F401 上无）
  - 内部辅 SRAM3 (64 KB) （仅适用于 STM32F42x/F43x 器件）
  - AHB1 外设 （包括 AHB-to-APB 总线桥和 APB 外设）
  - AHB2 外设
  - AHB3 外设 （FMC）（STM32F401 上无）

主设备和从设备通过多层总线矩阵相连接，即使在多个高速外设同时工作时也能确保并行存取和高效操作。该结构如下图 （适用于 STM32F40x/F41x）。

图 7. 系统架构



2.1 多层总线矩阵

多层总线矩阵使主设备能够执行并行数据传输，只要其寻址不同的从模块。除了 Cortex-Mx 哈佛架构和双 AHB 端口 DMA 以外，该结构还增强了数据传输并行能力，因此降低了执行时间，优化了 DMA 效率和功耗。

2.1.1 定义

- AHB 主设备：能够启动读写操作的总线主设备。在一个定义的时间段内，仅有一个主设备能够获取总线所有权。
- AHB 从设备：响应主设备读或写操作的总线从设备。总线从设备向主设备返回成功、失败或等待状态信号。
- AHB 仲裁器：总线仲裁器保证在同一时刻仅有一个主设备可以进行读或写操作。
- AHB 总线矩阵：多层 AHB 总线矩阵使每一层的 AHB 主设备和 AHB 从设备与其专用 AHB 仲裁器互连。仲裁采用循环调度算法。

2.1.2 循环调度优先级方案

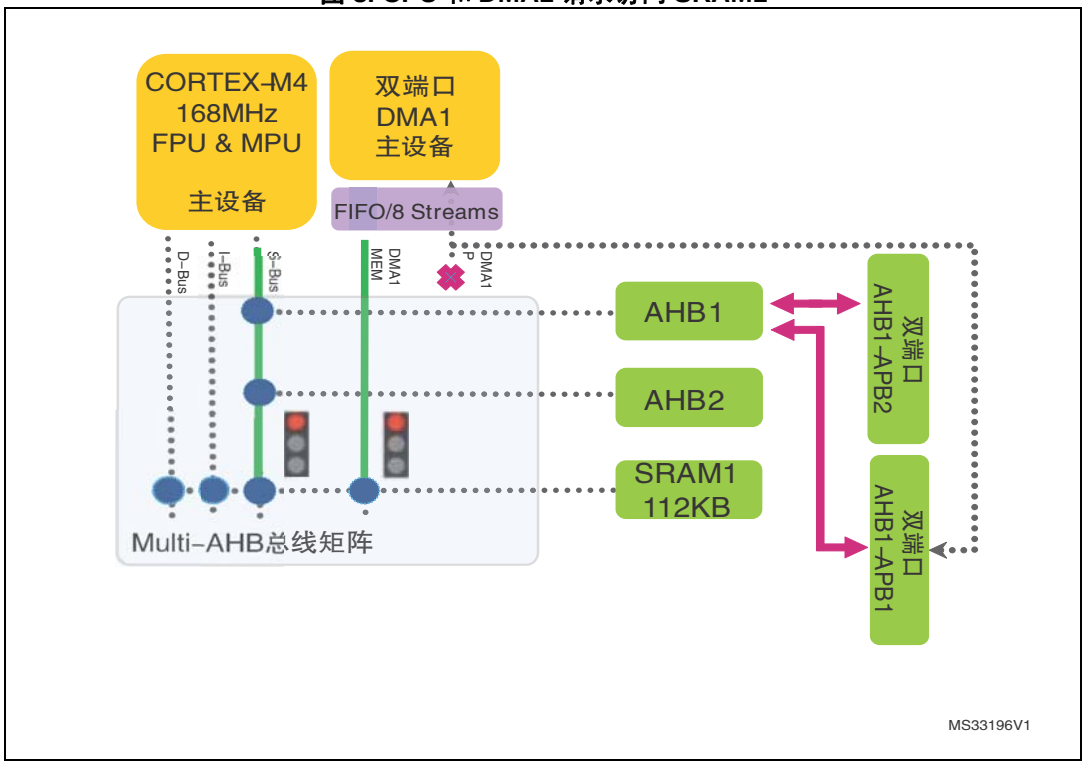
为确保每个主设备访问从设备时延迟尽量短，在总线矩阵层执行循环调度优先级方案：

- 循环调度仲裁策略使总线带宽合理分配。
- 限定最大延时。
- 循环调度以 1x （1 次）传输为单位。

当多个 AHB 主设备试图同时访问同一个 AHB 从设备时，总线矩阵仲裁器介入以解决访问冲突 。

在下面的例子中 （*图 8*）， CPU 和 DMA1 均试图访问 SRAM1 以读取数据。

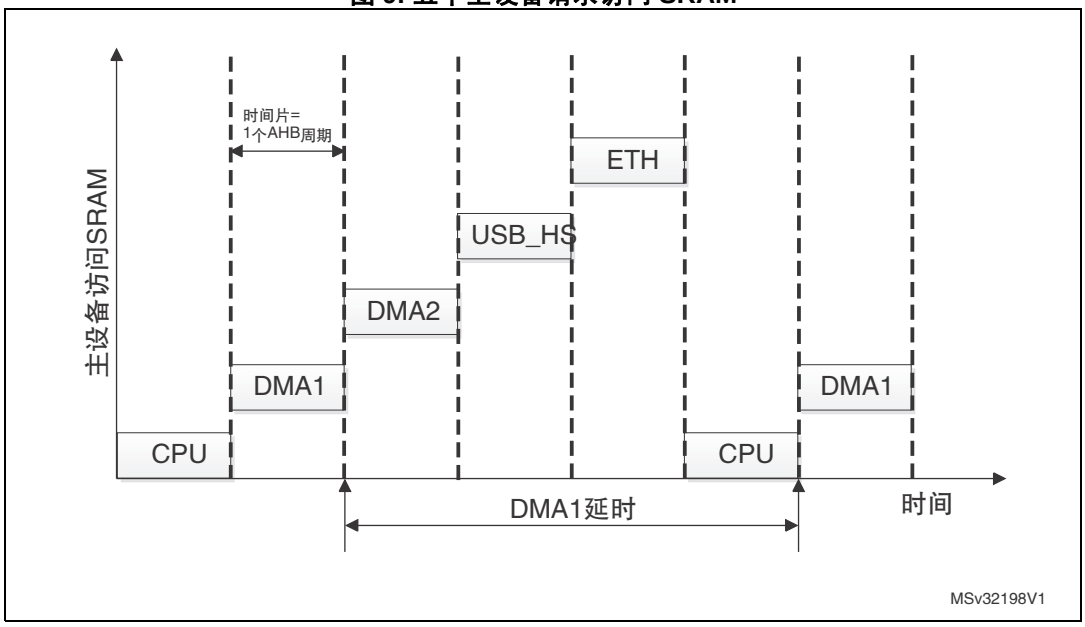
图 8. CPU 和 DMA1 请求访问 SRAM1



如上述示例总线访问请求同时发生的情况下，就需要总线矩阵仲裁。为了解决这种问题，需要应用循环调度策略：如果本次最后赢得总线控制权的主设备是 CPU，则在下一次访问中 DMA1 将赢得总线控制权并首先访问 SRAM1。 CPU 随后方可有权访问 SRAM1。

这就表明，一个主设备的传输延时取决于请求访问 AHB 从设备的其他等待主设备的数量。下面的例子中 （*图 9*），五个主设备同时试图访问 SRAM1。

图 9. 五个主设备请求访问 SRAM



DMA1 再次赢得总线矩阵 （访问权）并访问 SRAM1 的延时 （举例）等于其他主设备所有等待请求的执行时间。

2.1.3 进行总线矩阵仲裁，造成的 DMA 传输延时最差情况

DMA 主设备端口进行一次数据传输会遭遇的延时取决于其他主设备的传输类型和长度。

例如，如果我们考虑先前 DMA1 & CPU 的例子 （*图 8*），它们并行访问 SRAM， DMA 传输延时将随着 CPU 数据传输事务长度而变化。

如果总线访问首先给予 CPU 而 CPU 不是执行单次数据加载 / 存储，DMA 访问 SRAM 的等待时间可能从一个 AHB 周期 （单次数据加载 / 存储时间）延长为 N 个 AHB 周期，这里 N 为 CPU 数据传输事务中数据的数量。

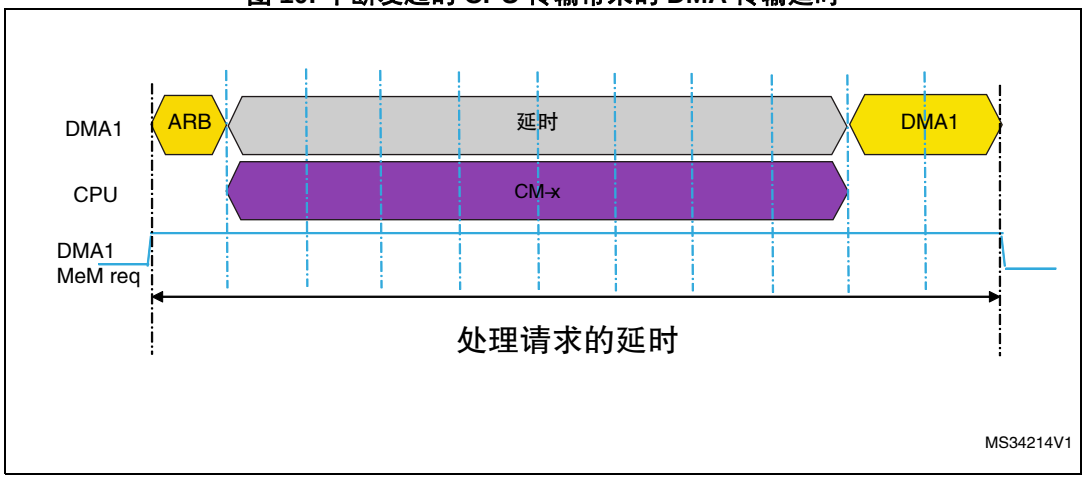
CPU 锁定 AHB 总线以保持其访问总线的所有权，减少了多次加载 / 存储操作过程中的延时以及进入中断的延时。这提高了固件的响应能力，但是可能导致 DMA 数据传输事务的延迟。

DMA1 与 CPU 并行访问 SRAM 的延时取决于传输类型：

- 中断 （上下文保护）发起的 CPU 传输：8 个 AHB 周期
- LDM/STM 指令发起的 CPU 传输：14 个 AHB 周期[(a)]
  - 在多达 14 个寄存器与存储器之间进行传输

a. 通过配置编译器，将加载 / 存储多重指令分解为单个加载 / 存储指令，可以降低由 LDM/STM 发起的传输的延时。

图 10. 中断发起的 CPU 传输带来的 DMA 传输延时



上图详细描述了一个 DMA 传输被因中断进入而发起的 CPU 多周期传输延迟的情形。DMA 存储器端口被触发，发出存储器访问请求。经过仲裁， AHB 总线未授权 DMA1 存储器端口访问，而由 CPU 系统总线来访问。可以看到在服务 DMA 请求之前有一段额外的延时。对于中断发起的 CPU 传输来说，延时为 8 个 AHB 周期。

当同时对一个从设备进行寻址且数据传输事务长度不是一个数据单元时，其他主设备 （如 DMA2, USB_HS, Ethernet…）也会碰到同样的情形。

为了提高 DMA 对总线矩阵的访问性能，建议避免总线竞争。

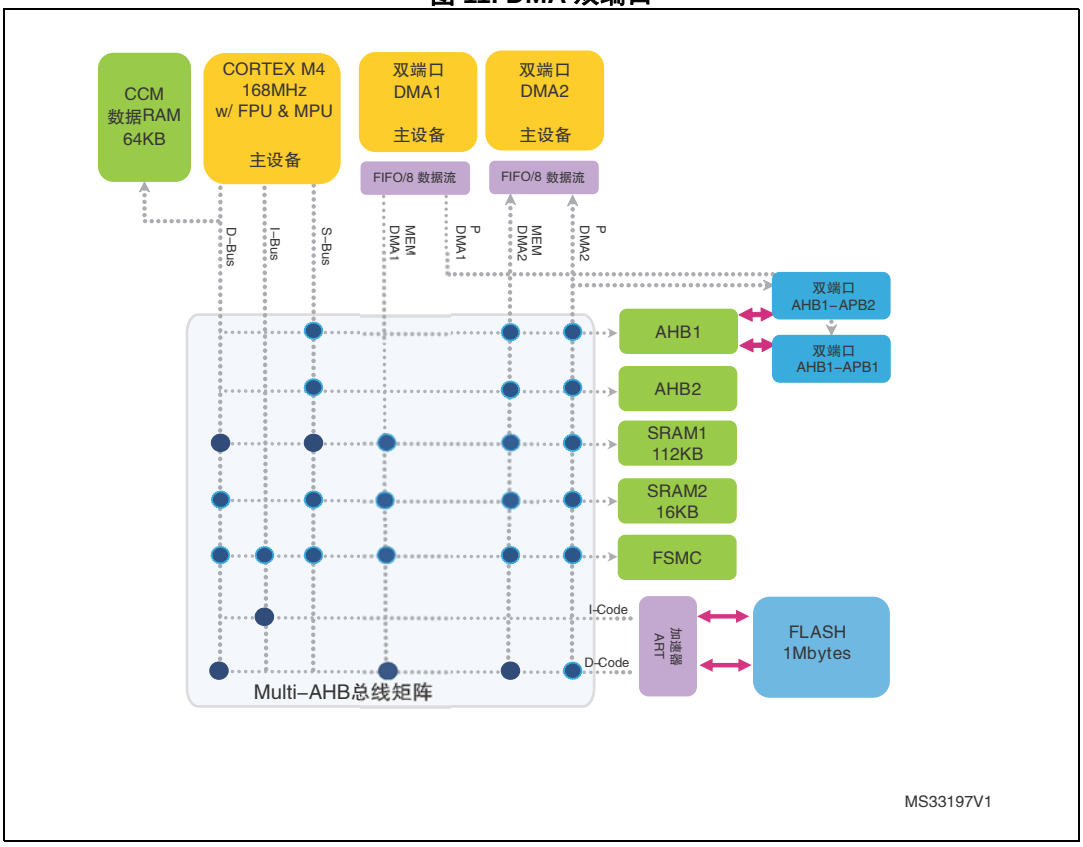
2.2 DMA 传输路径

2.2.1 双 DMA 端口

STM32F2/F4 器件集成了 2 个 DMA。每个 DMA 有 2 个端口，一个存储器端口和一个外设端口。利用外部总线矩阵和专用 DMA 路径，不仅在 DMA 级两个端口可以同时工作，还可以 DMA 和系统其它主设备同时工作。

同时操作优化了 DMA 效率，降低了响应时间 （请求和数据传输之间的等待时间）。

图 11. DMA 双端口



对于 DMA2：

- 通过总线矩阵， MEM （存储器端口）能够访问 AHB1、 AHB2、 SRAM1、 SRAM2、 FSMC 和 Flash 存储器 D-code。
- Periph （外设端口）能够访问：
  - 通过总线矩阵 （访问） AHB1、 AHB2、 SRAM1、 SRAM2、 FSMC 和 Flash 存储器 D-code，
  - 通过直接路径 （不经过总线矩阵）访问 AHB-to-APB2 桥。

对于 DMA1：

- 通过总线矩阵，MEM（存储器端口）能够访问 SRAM1、SRAM2、FSMC、Flash 存储器 D-code。
- Periph （外设端口）仅能通过直接路径 （不经过总线矩阵）访问 AHB-to-APB1 桥。

2.2.2 DMA 传输状态

本节介绍 DMA 在外设端口级和存储器端口级的传输步骤：

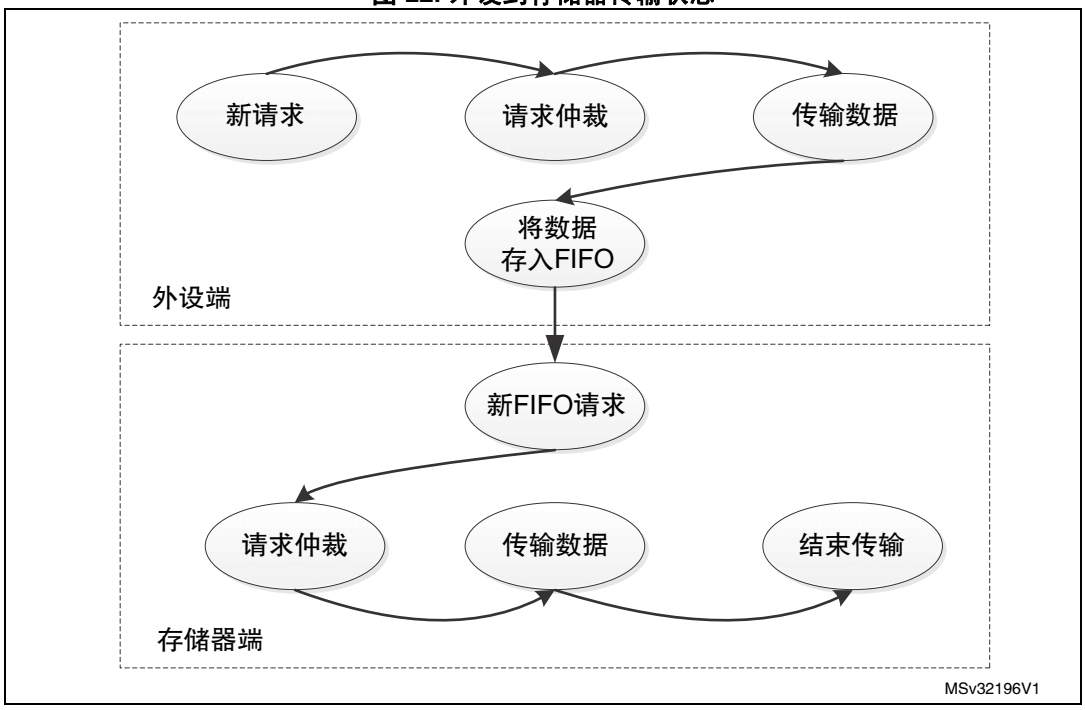
- 对于从外设到存储器的传输：

在这种传输模式下， DMA 需要两个总线访问来实现该传输：

– 外设端口的访问由外设请求触发，

– 存储器端口的访问可由 FIFO 阈值触发 （使用 FIFO 模式时）或外设读操作之后触发 （使用直接模式时）。

图 12. 外设到存储器传输状态



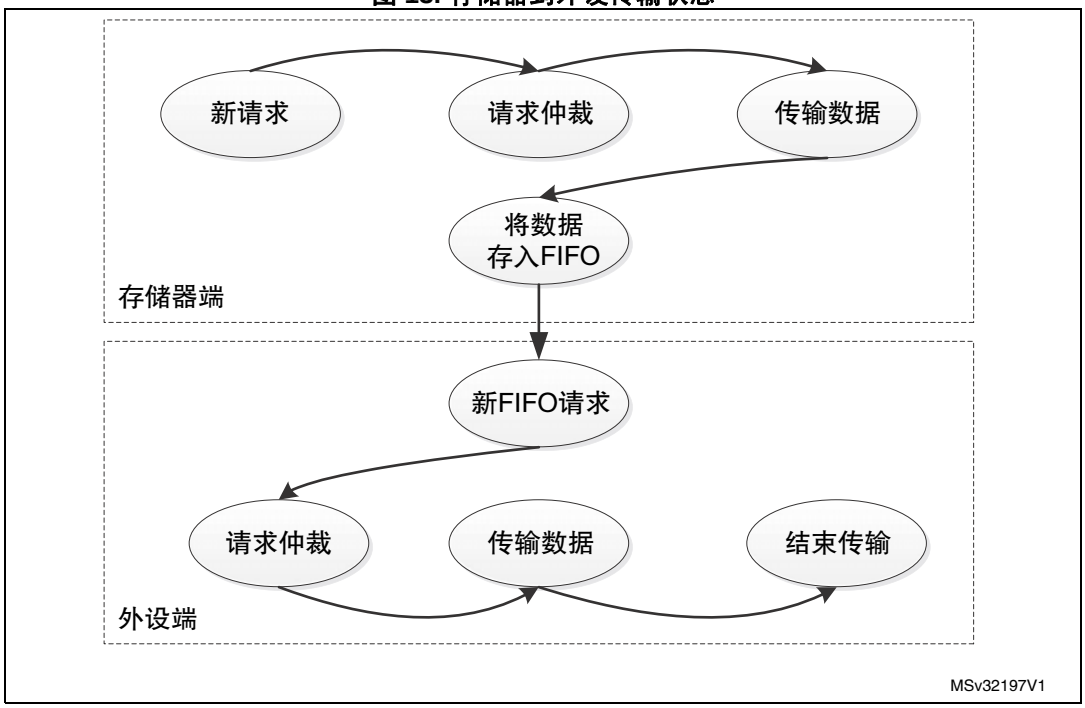
- 对于从存储器到外设的传输：

在这种传输模式下， DMA 需要两个总线访问来实现该传输：

– DMA 预计外设访问，从存储器中读取数据并将其存储在 FIFO 中，以保证在外设 DMA 请求被触发时能够立即进行直接数据传输。

– 当外设请求被触发，在 DMA 外设端口就产生了数据传输。

图 13. 存储器到外设传输状态



2.2.3 DMA 请求仲裁

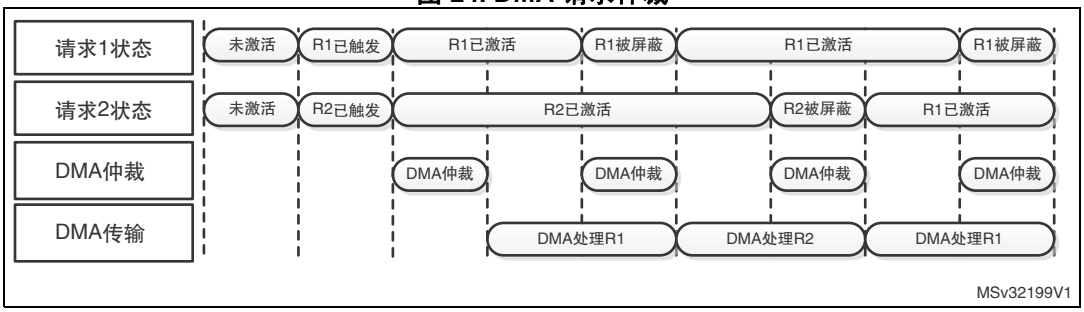
如*第 1.1.2 章节*所述，STM32F2/F4 DMA 集成了一个仲裁器，它能够根据优先级对每个（共两个）AHB 主设备端口（存储器和外设端口）8 个 DMA 数据流请求进行管理，并启动外设 / 存储器访问序列。

当超过一个 DMA 请求被激活时， DMA 控制器需要在激活的请求之间进行内部仲裁，并决定首先响应哪个请求。

下图描述了由 DMA 数据流 “request 1” 和 DMA 数据流 “request 2” 同时触发的两个循环 DMA 请求（请求 1 和 2 可以是任一 DMA 外设请求）。在触发 DMA 请求的下一个 AHB 时钟周期，DMA 仲裁器检查激活的等待请求，将访问权限交给拥有最高优先级的 “request 1” 数据流。

“request 1” 数据流的最后一个数据传输周期内进行下一个仲裁周期。此时， “request 1” 被隐藏，仲裁器仅能发现 “request 2” 是被激活的，因此这次访问权限交给 “request 2”，以此类推。

图 14. DMA 请求仲裁



一般建议：

- 高速 / 大带宽外设需要分配最高的 DMA 优先级。这能确保这些外设的数据传输延时尽量小，从而避免过载 / 欠载状态。
- 在带宽需求相同的情况下，相比工作在主模式 （可以控制数据流量）下的外设，建议为工作在从模式 （不可控制数据传输速度）的外设 DMA 请求分配更高的优先级。
- 基于总线矩阵多层结构，两个 DMA 能够并行工作，必要时多个高速外设请求可以在两个 DMA 之间平均分配。

2.3 AHB-to-APB 桥

STM32F2/F4 产品集成了两个 AHB-to-APB 桥， APB1 和 APB2，它们与外设相连。

2.3.1 双 AHB-to-APB 端口

STM32F2/F4 AHB-to-APB 桥是双端口结构，允许通过两种不同的路径进行访问：

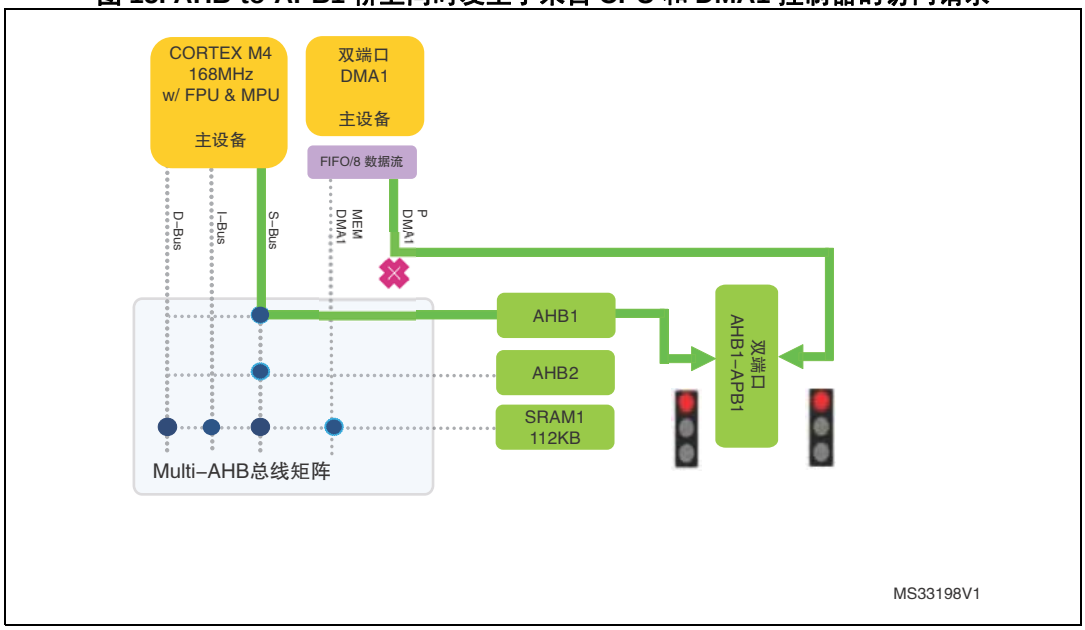
- 从 DMA1 到 APB1 或从 DMA2 到 APB2 可以形成直接路径（不经过总线矩阵）；这种情况下，访问不会被总线矩阵的仲裁带来延迟。
- 从 CPU 或 DMA2 的共同路径（通过总线矩阵），需要通过总线矩阵仲裁来赢得总线访问权。

2.3.2 AHB-to-APB 桥仲裁

由于这些产品上实现了DMA直接路径，在AHB-to-APB桥级使用仲裁器来解决并行访问请求。

下图描述了 AHB-APB1 桥上由 CPU （通过总线矩阵访问）和 DMA1 （通过直接路径访问）产生的并发访问请求。

图 15. AHB-to-APB1 桥上同时发生了来自 CPU 和 DMA1 控制器的访问请求



为了分配总线访问权限， AHB-APB 桥采用循环调度策略：

- 循环调度以 1x APB 传输为调度单位。
- 限定 DMA 外设端口的最大延时 （1 次 APB 传输）。

只有 CPU 和 DMA 控制器能够产生到 APB1 和 APB2 总线的并发访问：

- 对于 APB1，如果 CPU、 DMA1 和 / 或 DMA2 同时请求访问，就形成了并发访问。
- 对于 APB2，如果 CPU 和 DMA2 同时请求访问，就形成并发访问。

3 如何预估 DMA 延时

基于微控制器设计一个固件程序时，用户必须确保其不会发生欠载 / 过载，所以必须知道每次 DMA 数据传输的确切 DMA 延时，核验内部系统是否能够维持应用所需的数据带宽。

3.1 DMA 传输时间

3.1.1 默认 DMA 传输时间

如*第 2.2.2 章节*所述，要实现从外设到存储器的 DMA 传输，需要两次总线访问：

- 一次访问是在外设端口，由外设请求触发，它需要进行：
  - DMA 外设端口请求仲裁
  - 外设地址计算
  - 从外设中读取数据到 DMA FIFO （DMA 源）
- 另一次访问是在存储器端口，可由 FIFO 阈值触发（使用 FIFO 模式时）或外设读操作之后触发 （使用直接模式时），它需要进行：
  - DMA 存储器端口请求仲裁
  - 存储器地址计算
  - 在 SRAM （DMA 目标）中写加载数据

从存储器传输数据到外设时，也需要两次访问，如*第 2.2.2 章节*所述：

- 第一次访问：DMA 预计外设访问，从存储器中读取数据并将其存储在 FIFO 中，以保证在 DMA 外设请求被触发时立即进行直接数据传输。这个操作需要：
  - DMA 存储器端口请求仲裁
  - 存储器地址计算
  - 从存储器中读取数据到 DMA FIFO （DMA 源）
- 第二次访问：当外设请求被触发，在 DMA 外设端口就产生了数据传输。这个操作需要：
  - DMA 外设端口请求仲裁
  - 外设地址计算
  - 在外设地址 （DMA 目标）中写加载数据

一般来说， DMA 数据流的总传输时间 T_S 等于：

$$T_S = T_{SP} \text{ (外设访问 / 传输时间) } + T_{SM} \text{ (存储器访问 / 传输时间)}$$

其中：

T_{SP} 是 DMA 外设端口访问和传输的总时间，它等于：
$T_{SP} = t_{PA} + t_{PAC} + t_{BMA} + t_{EDT} + t_{BS}$

其中：

表 7. 使用不同的 DMA 路径，对应的外设端口访问 / 传输时间

说明	使用总线矩阵		DMA 直接路径
	到 AHB 外设	到 APB 外设	
t_{PA}：DMA 外设端口仲裁	1 个 AHB 周期	1 个 AHB 周期	1 个 AHB 周期
t_{PAC}：外设地址计算	1 个 AHB 周期	1 个 AHB 周期	1 个 AHB 周期
t_{BMA}：总线矩阵仲裁 （无并发访问请求时）(1)	1 个 AHB 周期	1 个 AHB 周期	N/A
t_{EDT}：有效数据传输	1 个 AHB 周期(2)(3)	2 个 APB 周期	2 个 APB 周期
t_{BS}：总线同步	N/A	1 个 AHB 周期	1 个 AHB 周期

1. 对于 STM32F401 产品，t_{BMA} 等于 0。
2. 对于 FMC，由于使用了外部存储器，需要额外加上一个周期的时间。额外的 AHB 周期根据外部存储器时序而增加。
3. 批量传输的情况下，有效数据传输时间取决于批量传输长度 （INC4 t_{EDT}= 4 个 AHB 周期）。

- T_{SM} 是 DMA 存储器端口访问和传输的总时间，它等于：
  $T_{SM} = t_{MA} + t_{MAC} + t_{BMA} + t_{SRAM}$

其中：

表 8. 存储器端口访问 / 传输时间

说明	延迟
t_{MA}：DMA 存储器端口仲裁	1 个 AHB 周期
t_{MAC}：存储器地址计算	1 个 AHB 周期
t_{BMA}：总线矩阵仲裁 （无并发访问请求时）(1)	1 个 AHB 周期(2)
t_{SRAM}：SRAM 读或写访问	1 个 AHB 周期

1. 对于 STM32F401 产品，t_{BMA} 等于 0。
2. 对于连续 SRAM 访问 （期间无其他主设备访问同一个 SRAM），t_{BMA} = 0 个周期。

3.1.2 并发存取时对应的 DMA 传输时间

当多个主设备试图同时访问同一个从设备时，*第 3.1.1 章节*中所述的 DMA 请求响应时间可能会增加额外延时。

对于外设和存储器的最长访问 / 传输时间， DMA 数据流服务的总延时受下列因素影响：

- 当多个主设备同时访问同一个 AHB 目标时， DMA 延时将受到影响；当总线矩阵仲裁器将访问权限交给 DMA 时，才开始进行 DMA 传输，如*第 2.1.2 章节*所述。
- 当多个主设备 （DMA 和 CPU）访问同一个 AHB-to-APB 桥时，由于 AHB-to-APB 桥仲裁， DMA 传输时间会有延迟，如*第 2.3.2 章节*所述。

3.2 示例

3.2.1 从 ADC 到 SRAM 的 DMA 传输

本示例适用于 STM32F2xx、STM32F405、STM32F407、STM32F415、STM32F417、STM32F42x 和 STM32F43x 产品。

ADC 配置为连续三重交替模式。这种模式下，它以最大 ADC 速率（36 MHz）对一个模拟输入通道进行连续转换。 ADC 预分频器设为 2，采样时间设为 1.5 周期，交替模式的两次连续 ADC 采样之间的延时设为 5 个周期。

DMA2 stream0 将 ADC 转换值传输到 SRAM 缓冲区。DMA2 通过直接路径完成对 ADC 的访问；而通过总线矩阵访问 SRAM。

表 9. DMA 外设 （ADC）端口传输延时

AHB/APB2 频率 传输时间	F_{AHB} = 72 MHz/ F_{APB2} = 72 MHz AHB/APB 比 = 1	FAHB = 144 MHz/ FAPB2 = 72 MHz AHB/APB 比 = 2
t_{PA}：DMA 外设端口仲裁	1 个 AHB 周期	1 个 AHB 周期
t_{PAC}：外设地址计算	1 个 AHB 周期	1 个 AHB 周期
t_{BMA}：总线矩阵仲裁	N/A(1)	N/A(1)
t_{EDT}：有效数据传输	2 个 AHB 周期	4 个 AHB 周期
t_{BS}：总线同步	1 个 AHB 周期	1 个 AHB 周期
T_{SP}：对外设端口的 DMA 传输总时间	5 个 AHB 周期	7 个 AHB 周期

1. DMA2 通过直接路径访问 ADC：无需总线矩阵仲裁。

表 10. DMA 存储器 （SRAM）端口传输延时

CPU/APB2 频率 传输时间	FAHB = 72MHz/ FAPB2=72MHz AHB/APB 比 = 1	FAHB = 144MHz/ FAPB2=72MHz AHB/APB 比 = 2
t_{MA}：DMA 存储器端口仲裁	1 个 AHB 周期	1 个 AHB 周期
t_{MAC}：存储器地址计算	1 个 AHB 周期	1 个 AHB 周期
t_{BMA}：总线矩阵仲裁	1 个 AHB 周期(1)	1 个 AHB 周期 (1)
t_{SRAM}：SRAM 写访问	1 个 AHB 周期	1 个 AHB 周期
T_{SM}：对存储器端口的 DMA 传输总时间	4 个 AHB 周期	4 个 AHB 周期

1. DMA 多次访问 SRAM 的情况下，如果没有其他主设备在此期间访问该 SRAM，则总线矩阵仲裁 （时间）为 0 个周期。

示例中，从 ADC DMA 触发 （ADC EOC）到将 ADC 值写入 SRAM 的 DMA 总延时，当 AHB/APB 预分频器等于 1 时该总延时为 9 个 AHB 周期， AHB/APB 预分频器等于 2 时为 11 个 AHB 周期。

注： *使用 FIFO 时，达到用户配置的 FIFO 门限时开始 DMA 存储器端口访问。*

3.2.2 SPI 全双工 DMA 传输

本示例适用于 STM32F2xx、STM32F405、STM32F407、STM32F415、STM32F417、STM32F42x 和 STM32F43x 产品，且使用 SPI1 外设的情况下。

配置两个 DMA 请求：

- DMA2_Stream2 用于处理 SPI1_RX 请求：此数据流被配置为最高优先级，以便及时服务接收到的 SPI1 数据，将这些数据从 SPI1_DR 寄存器传输到 SRAM 缓冲区。
- DMA2_Stream3 用于处理 SPI1_TX 请求：此数据流将数据从 SRAM 缓冲区传输到 SPI1_DR 寄存器。

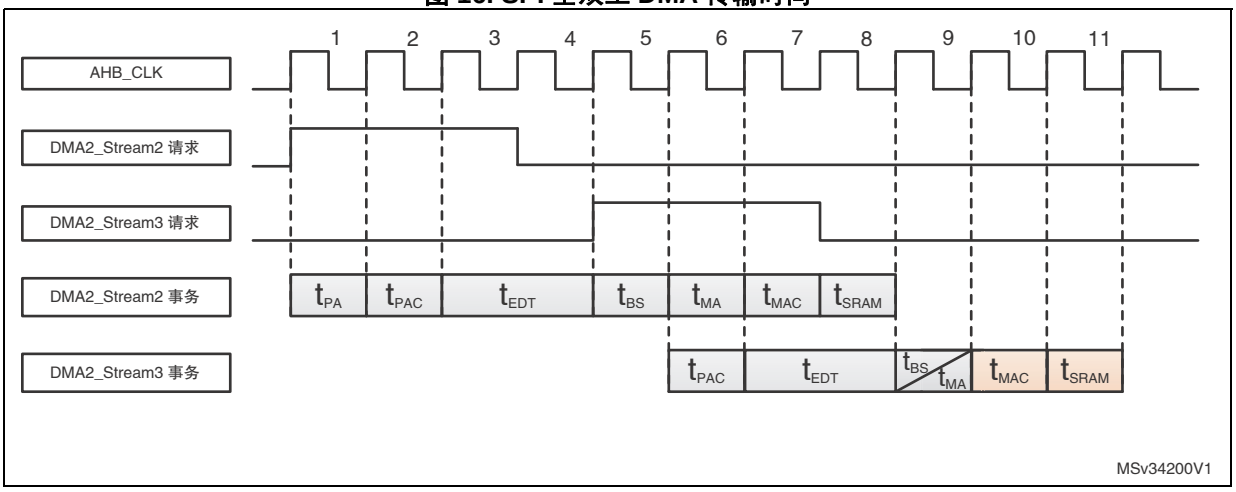
AHB 频率等于 APB2 频率（84 MHz），并设置 SPI1 以最大速率（42 MHz）运行。DMA2_Stream2（SPI1_RX）先于 DMA2_Stream3（SPI1_TX）2 个 AHB 周期触发。

在此配置下，CPU 对 I2C1_DR 寄存器无限轮询。考虑到 I2C1 外设挂在 APB1 总线上、SPI1 外设挂在 APB2 总线上，则系统路径如下：

- DMA2 通过直接路径访问 APB2（不经过总线矩阵），
- CPU 通过总线矩阵访问 APB1。

目的是证明 DMA 时序不受 CPU 在 APB1 上的轮询影响。下图简要说明了发送和接收模式下的 DMA 时序，以及每个操作的时序安排：

图 16. SPI 全双工 DMA 传输时间



本图说明了以下结论：

- CPU 在 APB1 上的轮询不影响 APB2 上的 DMA 传输延时。
- 对于 DMA2_Stream2（SPI1_RX）事务，在第 8 个 AHB 时钟周期无总线仲裁时间，因为最后一个访问 SRAM 的主设备为 DMA2 （因此无需重新仲裁）。
- 对于 DMA2_Stream3（SPI1_TX）事务，该数据流预先从 SRAM 中读取数据并将其写入 FIFO，然后一经触发， DMA 外设端口 （目标为 SPI1）便开始传输。
- 对于DMA2_Stream3，在DMA2_Stream2总线同步周期内执行DMA外设仲裁（1个AHB周期）。
  在当前 DMA 请求事务结束之前就有另外一个 DMA 请求时，总是执行这种优化操作。

4 对 DMA 控制器进行编程时的一些技巧和忠告

1. 停止 DMA 的软件序列

要断开连接到 DMA 数据流请求的外设，必须：

- 断开外设连接的 DMA 数据流，
- 待 DMA_SxCR 寄存器的 EN 位复位 （“0”）。

只有这样才能安全地禁止外设。

注： *在这两种情况下，传输完成中断标志 （DMA_LISR 或 DMA_HISR 中的 TCIF）置 1 将指示因数据流禁止而结束传输。*

2. 使能新的传输之前对 DMA 标志进行管理

使能新的传输之前，用户必须确定 DMA_LISRDMA_HISR 中的传输完成中断标志 （TCIF）已清 0。

一般建议，在开始新的传输之前，将 DMA_LIFCR 和 DMA_HIFCR 寄存器的所有标志位均清零。

3. 使能 DMA 的软件序列

使能 DMA 时，使用下面的软件序列：

- 配置适当的 DMA 数据流。
- 使能所用的 DMA 数据流 （设置 DMA_SxCR 寄存器的 EN 位）。
- 使能所用的外设。

注： *如果用户在使能相应的 DMA 数据流之前就使能了所用的外设，则由于 DMA 尚未准备好向外设发送其所需要的数据 （从存储器到外设进行传输的情况下），将会出现 “FEIF” (FIFO 错误中断标志）。*

4. NDTR=0 时，存储器到存储器传输

对 DMA 数据流进行配置使其实现正常模式下从存储器到存储器的传输，当 NDTR 达到 0 时，传输完成标志将置 1。此时，如果用户重新置位该数据流的使能位 （DMA_SxCR 中的 EN 位），存储器到存储器的传输将自动使用最后的 NDTR 值再次重新触发。

5. PINC/MINC=0， DMA 外设批量传输

禁止外设地址或存储器地址递增配置下的 DMA 批量数据传输，允许对
支持批量传输 （集成 FIFO）的内部或外部 （FSMC）外设寻址。这种模式保证了该 DMA 数据流在其数据传输过程中不被其他 DMA 数据流中断。

6. 两次映射 DMA 请求

当用户配置了两个（或更多）DMA 数据流服务于同一个外设请求时，软件应当保证在使能新的 DMA 数据流之前，当前 DMA 数据流完全被禁止（通过轮询 DMA_SxCR 寄存器的 EN 位）。

7. 最佳 DMA 吞吐量配置

若 STM32F4xx 的 AHB 频率不太高，而 DMA 为一个高速外设服务时，建议将堆栈置于 CCM（CPU 可通过 D-bus 对其直接寻址）中而不是 SRAM 上，否则将会在 CPU 和 DMA 访问 SRAM 存储器时产生额外的并发访问请求。

8. DMA 传输暂停

可以随时暂停 DMA 传输稍后重新开始；也可以在 DMA 传输结束前明确暂停其传输功能。

分为两种情况：

- 传输被中止，且后续不再从其停止点继续传输：无其他操作，仅将 DMA_SxCR 寄存器的 EN 位清零来禁止该数据流，并在 EN 位置位前保持等待。因此：
  - DMA_SxNDTR 寄存器中含有数据流停止时剩余数据项的数目，这样软件便可以确定数据流中断前已传输了多少数据项。
- 暂停传输，以便稍后通过重新使能该数据流来重新开始 （传输）：要从停止点重新开始该传输，软件必须在禁止数据流 （EN 位置 “0”）后读取 DMA_SxNDTR 寄存器，以便获取已接收的数据项数目。然后：
  - 必须更新外设和 / 或存储器地址以调整地址指针。
  - 必须使用要传输的剩余数据项的数目 （禁止数据流时读取的值）更新 SxNDTR 寄存器。
  - 然后可以重新使能数据流，从停止点重新开始传输。

注： *请注意，传输完成中断标志 （DMA_LISR 或 DMA_HISR 中的 TCIF）置 1 将指示因数据流中断而结束传输。*

5 结论

STM32F2/F4 DMA 控制器经过了精心设计，能够覆盖大部分嵌入式应用：

- 固件在选择合适的 16 数据流 X 16 通道（每个 DMA 控制器上 8 个）组合时更具灵活性，
- 减少了 DMA 传输的总延时，由于双 AHB 端口结构和到 APB 桥的直接路径，避免了 DMA 服务低速 APB 外设时 CPU 在 AHB1 访问上的暂停，
- 在 DMA 控制器上实现了 FIFO，使固件在源和目标之间配置不同的数据宽度时更具灵活性，并且使用递增批量传输模式时可以提高传输速度。

6 修订历史

表 11. 文档修订历史

日期	修订	变更
2014 年 2 月 4 日	1	初始版本。