AN2548

应用笔记

使用STM32F101xx和STM32F103xx DMA 控制器

介绍

这篇应用笔记描述了怎么使用STM32F101xx 和 STM32F103xx的直接内存访问（DMA）控制器。STM32F101xx和STM32F103xx DMA 控制器、Cortex-M3™ 内核、先进的微控制器总线架构和存储系统，是为了提供一个高的数据带宽，并开发响应时间非常快的软件。

这篇文档也描述了怎样充分利用这些特性，以及对于不同的外设和子系统怎样保证正确的响应时间。

在下文中STM32F101xx 和STM32F103xx都记作STM32F10xxx，DMA控制器都记作DMA。

1 DMA控制器

DMA是一个AMBA先进的高性能总线（AHB），它的特点是它具有两个AHB端口：一个从端口，用于DMA编程；一个是主端口，主端口允许DMA在不同的从模块间发起数据传输。

DMA允许数据传输在后台传输，而不需要Cortex-M3处理器的干预。在这个操作中，主处理器能够执行其他的任务。cpu仅仅在要处理的数据块传输完毕后才被中断。大量的数据传输的同时不会对系统性能产生大的影响。

DMA主要用来为不同的外设模块实现数据缓冲存储（一般存储在系统的主存储器忠）。这种解决方案和分布式解决方案（每个外设需要实现自己当地数据存储）相比，无论在硅的使用和功耗上都要更胜一筹。

STM32F10XXXDMA控制器充分利用了Cortex-M3 Harvard架构和多层的总线系统来保障非常低的DMA数据传输延时和CPU执行/中断事件检测/服务。

1.1 DMA的主要特性

DMA具有以下的特性：

1 7个DMA通道（channel 1到7）支持单向的从源端到目的端的数据传输

2 DMA通道优先级可通过硬件和软件编程

3 支持存储器到存储器，存储器到外设，外设到存储器，外设到外设的数据传输

4 能够对硬件/软件传输进行控制

5 传输时内存和外设指针自动增加

6 传输数据的大小可编程

7 总线错误自动管理

8 循环模式/非循环模式

9 可传输高达65536个数据

DMA目标是为所有外设提供相关的大的数据缓冲区，这些缓冲区一般位于系统的SRAM中

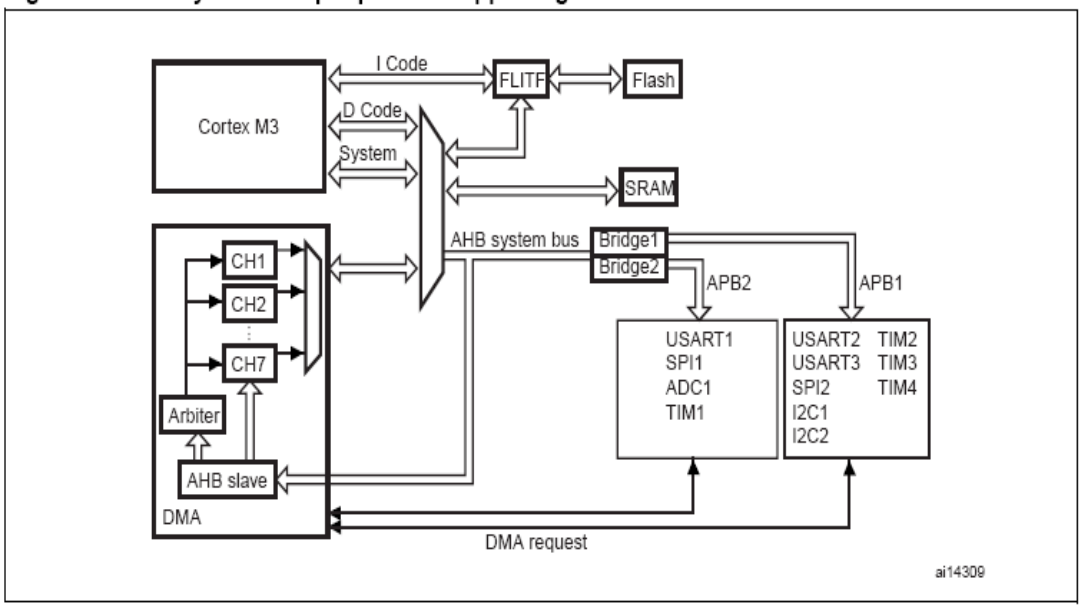
每一个通道在特定的时间里分配给唯一的外设，连接到同一个DMA通道的外设不能够同时使用（在DMA活跃时），所有的连接到给定的通道的外设中只有一个能够拥有DMA功能

支持DMA的外设如下表, DMA服务的外设和总线系统结构也在下图中

Table 1. Peripherals served by DMA and channel allocation

Peripherals		CH1	CH2	CH3	CH4	CH5	CH6	CH7
ADC	ADC1	ADC1						
SPI	SPI1		SPI1_RX	SPI1_TX				
	SPI2				SPI2_RX	SPI2_TX		
USART	USART1				USART1_TX	USART1_RX		
	USART2						USART2_RX	USART2_TX
	USART3		USART3_TX	USART3_RX				
I^2C	I^2C1						I2C1_TX	I2C1_RX
	I^2C2				I2C2_TX	I2C2_RX		
TIM	TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
	TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
	TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
	TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

Figure 1. Bus system and peripherals supporting DMA



2 性能考虑

STM32F10xxx有两个主模块Cortex-M3处理器和DMA。他们通过总线矩阵连接到从总线，Flash Memory总线，SRAM总线和AHB系统总线。从总线轮流连接到两个APB总线以服务所有的嵌入的外设（参见上图）。

总线矩阵有两个主要的特性，实现系统性能的最大化和减少延时：

1 循环优先级调度

2 多层结构和总线挪用

2.1 循环优先级调度

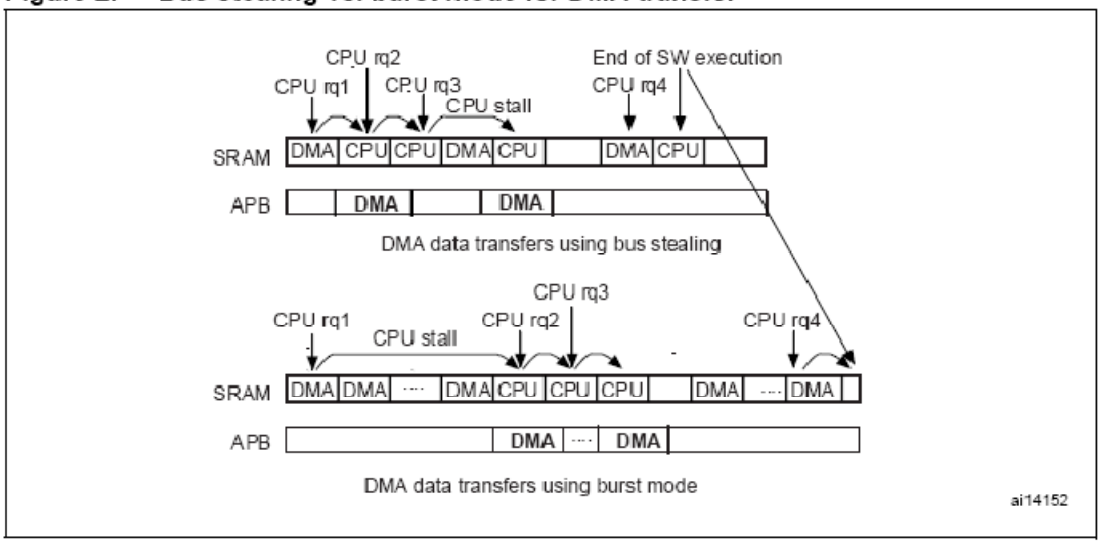
NVIC和Cortex-M3处理器实现了高性能低延时中断调度。所有的Cortex-M3指令既可以在单周期内执行，也可以在周期级上被中断。为了在系统水平上保证这个特性，DMA和总线矩阵必须确保DMA不能够长时间占用总线。循环优先级调度能够确保CPU能够在任何一个周期访问所有的从总线。因此，第一个数据最大总线系统延时在cpu看来，就是一个总线周期（两个APB时钟周期的最大值）

2.2 多层结构和总线挪用

多层结构允许两个两个主设备并发执行数据传输，只要他们寻址到不同的设备。在Cortex-M3 哈佛架构，这种结构提高了数据的并行性，因此减少了执行时间并且优化了DMA效率。从Flash存储器取指是通过完全独立的总线，所以DMA和CPU只是在需要通过一个给定的从总线进行数据访问时才会产生竞争。

当其他DMA控制器工作在突发模式下，STM32F10xxx DMA的数据传输（总线挪用）只需要单个总线周期。当使用总线挪用存取机制时，CPU等待数据的最大时间是很短的（一个总线周期）。CPU访问SRAM和DMA存取就是自然地交叉存取。CPU访问和DMA通过APB总线存取外设就可以并行进行。即使更多的数据访问使用突发模式DMA可能会更快些，CPU停止的那段长时间很少被复原。下图列出了总线挪用和突发机制的区别

Figure 2. Bus stealing vs. burst mode for DMA transfer



在极端的情况下（CPU从内存的一个地方复制一块数据到内存中的另一个地方），软件的执行必须等到整个DMA传输完毕。当然，CPU大部分时间完成的是数据处理，CPU和DMA交叉存取数据的频率很低。

STM32F10XXX总线结构固有的并行性，加上DMA总线挪用机制保证了CPU不会长时间地等待从SRAM中读取数据。总线挪用机制的DMA是总线使用效率更高，明显地减少了软件执行时间。

2.3 DMA延迟

DMA完成从外设到SRAM 存储器的数据传输有三个步骤

1 DMA请求仲裁

2 从外设中读取数据（DMA源）

3 将读取的数据写入到SRAM中（DMA目的地）

当DMA把数据从内存中传输到外设（例如SPI传送），操作必须按照以下顺序

1 DMA请求仲裁

2 从SRAM中读取数据

3 将读取到的数据通过APB总线写入到外设中。

每个通道总的服务时间是总裁时间（1 AHB时钟周期），外设访问时间（2个AHB时钟周期）和SRAM读/写访问时间（1个AHB时钟周期用于单独的读/写操作，2个周期用于写操作后SRAM读）的和。

当DMA通道空闲或者是第三个操作完成后，DMA比较所有挂起的DMA请求的优先级，高优先级的通道将会被服务，DMA开始执行第二个操作。当一个通道正在服务时（第二，三个操作正在进行），没有其他的通道能够被服务，而不管它的优先级如何。

当至少同时有两个DMA通道使能时，最高优先级通道的DMA延迟时间为正在传输的时间加上下个将被服务的DMA通道（挂起优先级最高）。

2.4 数据总线带宽限制

数据总线带宽限制主要是因为APB总线比系统SRAM和AHB总线速度慢。对于最高优先级通道，以下两种情况是需要考虑的：

1 当多余一个DMA通道使能时，最高优先级通道在APB总线上的请求数据带宽必须比APB传输率低25%

2 尽管高速/高优先级DMA传输通常发生在APB2上（更快的APB总线），但是CPU和其他DMA通道可以在APB1上访问外设。大约3/4的APB传输是在APB1上完成的，最小的APB2频率依赖于最快的DMA通道数据带宽，最大的APB时钟分频因子由下列的等式给出：

f_{AHB}〉（$2*N_2+6*N_1$）$*B_{MAX}$

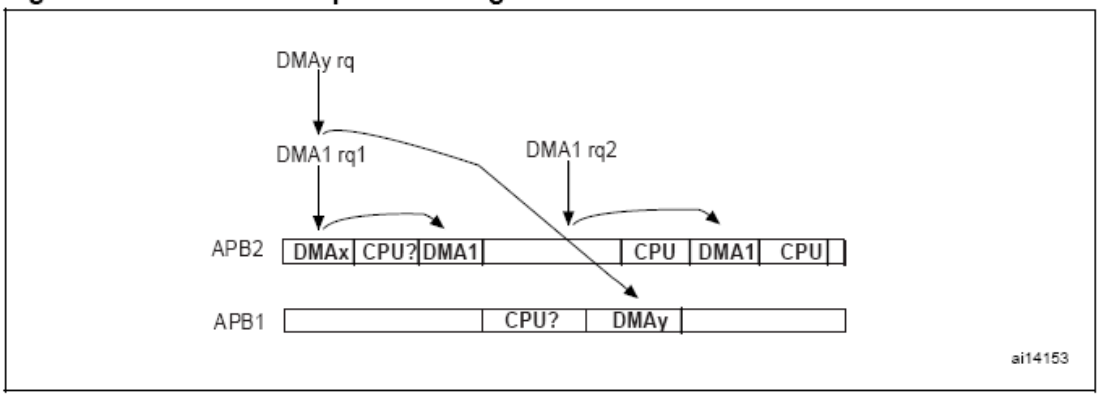
如果$N_2 < N_1$ 则$N_1 <$（ f_{AHB}/ B_{MAX}）/8

其中f_{AHB}是AHB时钟频率

N_2和 N_1是APB1和APB2的时钟分频因子

B_{max}是APB2上的最大并行数据带宽

Figure 3. APB bus occupation during DMA transfers



1. DMA1 is the highest priority channel.

2.5 通道优先级选择

为了实现外设数据的连续传输，相关的DMA通道必须能够维持外设数据传输率和确保服务的延迟时间比两个连续的数据之间的时间短。

高速/高带宽外设必须拥有最高的DMA优先级，这确保了最大的数据延迟对于这些外设都是可以忍受的，而且可以避免over/under-run的情况

在相等的带宽需求的情况下，推荐给工作在从模式下（没有数据传输速度控制）的外设分配的优先级比工作在主模式（能够控制数据流）下的外设分配的优先级高。

缺省情况下 ,通道分配和硬件优先级(从1到7)是按照最快的外设分配最高优先级的顺序来分配。当然，在某些运用场合下也许不是按照这样来分配的，在这种情况下，用户能够为每一个通道配置软件优先级（分4种，从非常高到低），软件优先级优先于硬件优先级。

当几个外设（不管有没有使用DMA）并行使用时，用户必须确保内部系统能够维持运用所要求的总的数据带宽，必须从下列因素中选择一个折中方案：

1 每个外设的运用需求

2 内部数据带宽

2.5.1 运用需求

以SPI接口为例，SPI接口数据带宽是通过波特率除以SPI的数据字长度而得到的（由于一个满数据需要在每次发送之前/之后传输到/传输自SPI），假设SPI的波特率是18M，数据是以8位传输的，操作配置在单一模式下，内部数据带宽需求是2.25M/s，如果SPI在16位模式下，则数据带宽将是1.125M/s

2.5.2 内部数据带宽

内部数据带宽依赖于以下两个条件：

1 总线频率 可获得的数据带宽与总线时钟频率是成正比的

2 总线类型 AHB数据传输需要一个时钟周期（除了SRAM写后读访问需要2个周期），数据通过APB总线传输给外设需要花费2个APB时钟周期

推荐DMA总线使用保持在2/3以下，这样系统和CPU的性能就可在合理的水平

3 DMA编程示例

所有的示例都使用STM32F10xxx固件库 ,可以参考AN2564 STM32F10XXX DMA运用的例子。相关的固件库可以在ST网站上下载，网址为www.st.com

3.1 使用SPI传输实现ADC连续数据的获取

ADC配置为连续转换模式，该模式下，它能够将一个输入通道以最大速度进行连续转换。为了获得最大的连续转换速度，AHB总线频率设置为56MHz，ADC预分频为4，采样速度为13.5周期。这些设置通过DMA通道1传输到位于系统RAM中的缓冲区，通道1的数据带宽设置为0.54M/s

当ADC转换后的数据填充了缓冲区的一半后，软件计算出最大值并且是这些数字化的数据规格化（最大值设置为0xFF），ADC转换的结果通过SPI接口传输到外部。

ADC转换结果是通过SPI1接口来传输的，使用DMA通道3把数据从SRAM缓冲区中传输到SPI1的数据寄存器。要达到最大的DMA传输的速度0.875M/s，SPI1接口必须配置为16位 主传输模式 14M波特率的传输速度。

可是，当SPI1工作在主模式，SPI1有效的数据传输速度受数据的可用速率1M/S的限制时，优先级的配置如下：通道1（ADC）：VeryHigh，通道3(SPI1_TX):High

3.2 SPI直接传输实现ADC连续数据的获取

这个示例完成的功能和前面一个几乎相同，没有数据的标准化。CPU内部并没有使用这些数据，直接把ADC转换结果传输到SRAM缓冲区可以较少一半的总线占用。

所以，仅有DMA 通道1被使用，该通道的目的存储器地址设置为SPI数据寄存器，而不需要SRAM缓冲。

3.3 使用DMA实现GPIO快速数据传输

这个示例展示了如何将不同的外设用于DMA请求和数据传输。这个机制允许在没有使用CPU的情况下实现简单的快速并行同步接口

Timer1和DMA通道4连接到TIM1_TRIG，用来实现获取数据的接口，在GPIO的端口上可以获取8位并行数据，一个外部时钟信号作用在Timer1的外部触发器输入端。在外部触发器上升沿，定时器产生一个DMA请求，只要GPIO数据寄存器地址已设置到DMA通道4的外设地址，DMA控制器在每一次DMA请求时从GPIO端口读取数据，并把它存储到SRAM的缓冲器中。

4 修订记录

表1 修订记录

日期	修订	改变
2007-6-29	1	初次发布

5 版权声明：