AN4667
应用笔记

STM32F7系列系统架构和性能

前言

STM32F7系列器件是首款基于ARM®Cortex®-M7的32位微控制器。利用ST的ART加速器™和L1缓存的优势，STM32F7系列器件实现了Cortex®-M7的最大理论性能。

基准测试分数稳步达到了1082 CoreMark和462 DMIPS，无论代码是通过嵌入式Flash存储器执行，还是通过内部RAM或者外部存储器（SRAM、SDRAM或者Quad SPI Flash存储器）执行。

STM32F7系列器件的高性能源自：

- 强力的超标量流水线和DSP性能提供了一个具有低中断时延的快速实时响应
- 对大容量外部存储的高效访问
- 适合复杂计算的高性能浮点运算能力

本应用笔记呈现了STM32F7的全面架构以及存储接口和特性，它们提供了更高的灵活度以实现最佳的性能以及额外的代码和数据大小。它还提供了有助于提高系统性能和卸载CPU的多主机架构。

本应用笔记同样提供了STM32F7系列器件在多种存储分区配置下（不同代码和数据位置)的架构性能以及使能DMA的架构性能的软件演示。

本应用笔记随附X-CUBE-32F7PERF嵌入式软件包，其中包括两个项目：

- Stm32f7_performances项目的目的是演示STM32F7架构在不同配置下的性能，即使用ART accelerator™和高速缓存时代码的执行和数据在不同存储空间中的存储。
- Stm32f7_performances_DMAs目的是演示多主机配置下的架构性能。

每个项目都是采用以下可用板子完成：STM32756G-EVAL，STM32F769I-EVAL和STM32F723E-DISCO板。

目录

表格索引

图片索引

1 STM32F7系列的系统架构概览

1.1 Cortex®-M7内核

STM32F7系列器件基于高性能的ARM®Cortex®-M732位RISC内核，工作频率高达216MHz。Cortex®-M7内核具有高性能浮点单元（FPU）。该内核可实现单精度浮点单元或双精度浮点单元（取决于STM32F7系列器件），支持所有ARM®单精度和双精度数据处理指令和数据类型。它还具有一组DSP指令和提高应用安全性的一个存储器保护单元（MPU）。Cortex®-M4到Cortex®-M7的向前兼容性允许为Cortex®-M4编译的二进制数直接在®-M7上运行。

Cortex®-M7的特性是具有分支预测和双指令执行的6/7-级超标量流水线。分支预测允许分支解析以预测下一个分支，因此将环路消耗的周期数从每个环路4和3个周期减少为1个周期。双指令的特征是允许内核同时执行两条指令，并且与指令的顺序无关，由此来增加指令吞吐率。

1.2 Cortex®-M7系统缓存

这些器件嵌入了Cortex®-M7，具有1级缓存（L1-cache），该缓存可分为两个独立的高速缓存：数据缓存（D-cache）和指令缓存
（I-cache），支持Harvard架构，可实现最佳性能。这些缓存使得即使在高频率下也可以达到零等待状态。

默认情况下，指令和数据缓存是禁用的。

ARM CMSIS库提供了两个使能数据和指令缓存的函数：

- SCB_EnableICache()用于使能指令缓存
- SCB_EnableDCache()用于使能数据缓存

更多关于怎样使能和停用缓存的信息，请参考“ARMv7-M架构参考手册”。

有关STM32F7系列上L1-cache使用的更多详细信息，还请参考*STM32F7系列产品上的1级缓存*应用笔记（AN4839）。

*表 1*总结了STM32F7系列中每个器件的缓存空间大小。

表1. STM32F7系列器件缓存空间大小

器件	指令缓存空间大小	数据缓存空间大小
STM32F74xxx和STM32F75xxx	4 KB	4 KB
STM32F72xxx和STM32F73xxx	8 KB	8 KB
STM32F76xxx和STM32F77xxx	16 KB	16 KB

1.3 Cortex®-M7总线接口

Cortex®-M7具有五个接口：AXIM、ITCM、DTCM、AHBS和AHBP。这部分将对它们进行逐一阐述。

除了AHBS接口以外，所有这些接口都是主接口，AHBS接口是一个从接口，可连接其他主接口到Cortex®-M7。

1.3.1 AXI总线接口

AXI作为高级可扩展接口。Cortex®-M7实现了AXIM AMBA4，它是一个64位宽的接口，用以获得更大取指和数据加载带宽。

如果缓存使能，任何不是对TCM或者AHBP接口的访问由适当的缓存控制器处理。用户需要考虑到并非所有的存储区域都可以缓存，这取决于它们的类型。具有共享存储、器件或者强秩序类型的存储区域无法缓存。只有典型的非共享存储器才可以缓存。

更多关于存储器属性和行为的信息，请参考“ARMv7-M架构参考手册”。

为了修改存储区域的类型和属性，可以使用MPU使其可以缓存。这可以通过配置TEX字段和MPU_RASR寄存器中的S、C和B位来完成。

*表 2*总结了在cortex®-M7复位之后存储区域的属性。

表2. Cortex®-M7复位之后默认的存储属性

地址范围	区域名称	类型	属性	Execute Never?
0x00000000-0x1FFFFFFF	代码	Normal	可缓存、透写、读缺失分配	无
0x20000000-0x3FFFFFFF	SRAM	Normal	可缓存、回写、读缺失和写缺失分配	无
0x40000000-0x5FFFFFFF	外设	设备	非共享	有
0x60000000-0x7FFFFFFF	RAM	Normal	可缓存、回写、读缺失和写缺失分配	无
0x80000000-0x9FFFFFFF	RAM	Normal	可缓存、透写、读缺失分配	无
0xA0000000-0xBFFFFFFF	外部器件	设备	可共享	有
0xC0000000-0xDFFFFFFF	外部器件	设备	非共享	有
0xE0000000-0xE000FFFF	私有外设总线	强序	-	有
0xE0010000-0xFFFFFFFF	供应商系统	设备	非共享	有

有关MPU使用的更多详细信息，请参考*内存保护单元*章节中的*STM32F7系列Cortex®-M7处理器编程手册*（PM0253）。

在STM32F7系列器件中，64位AXI主控总线通过一个高性能的AXI到multi-AHB桥接器件来连接内核到总线矩阵，该桥接器件具有四个主接口：

- 到内部Flash存储的1x 64位AHB
- 到总线矩阵的3x 32位AHB

1.3.2 TCM总线接口

作为紧密耦合的TCM存储器用来提供内核到内部RAM存储器的连接。TCM接口具有哈佛架构，因此这里有一个ITCM（指令TCM）和DTCM（数据TCM）接口。ITCM具有一个64位的存储接口，而DTCM分为两个32位端口：D0TCM和D1TCM。

1.3.3 AHBS总线接口

Cortex®-M7 AHBS（AHB从设备）是一个32位宽接口，可以提供系统到ITCM、D1TCM和D0TCM的访问。然而在STM32F7架构中，AHBS只允许与DTCM-RAM的数据相互传输（见*图 1*）。AHBS上无法访问ITCM总线，因此不支持DMA与ITCM RAM之间的数据传输。对于DMA与ITCM接口上的-Flash存储之间的数据传输，所有的传输被强制通过AHB总线。AHBS接口可以在内核处于睡眠状态时使用，因此DMA传输可以在低功耗模式下进行。

1.3.4 AHBP总线接口

AHBP接口（AHB外设）是一个单独的32位宽的接口，专门用于CPU和外设的连接。它只用于数据访问。取指令从不在该接口上进行。在STM32F7架构中，这条总线连接Cortex-M7内核的AHBP外设总线到AHB总线矩阵。该总线连接到AHB1、AHB2、APB1和APB2外设。

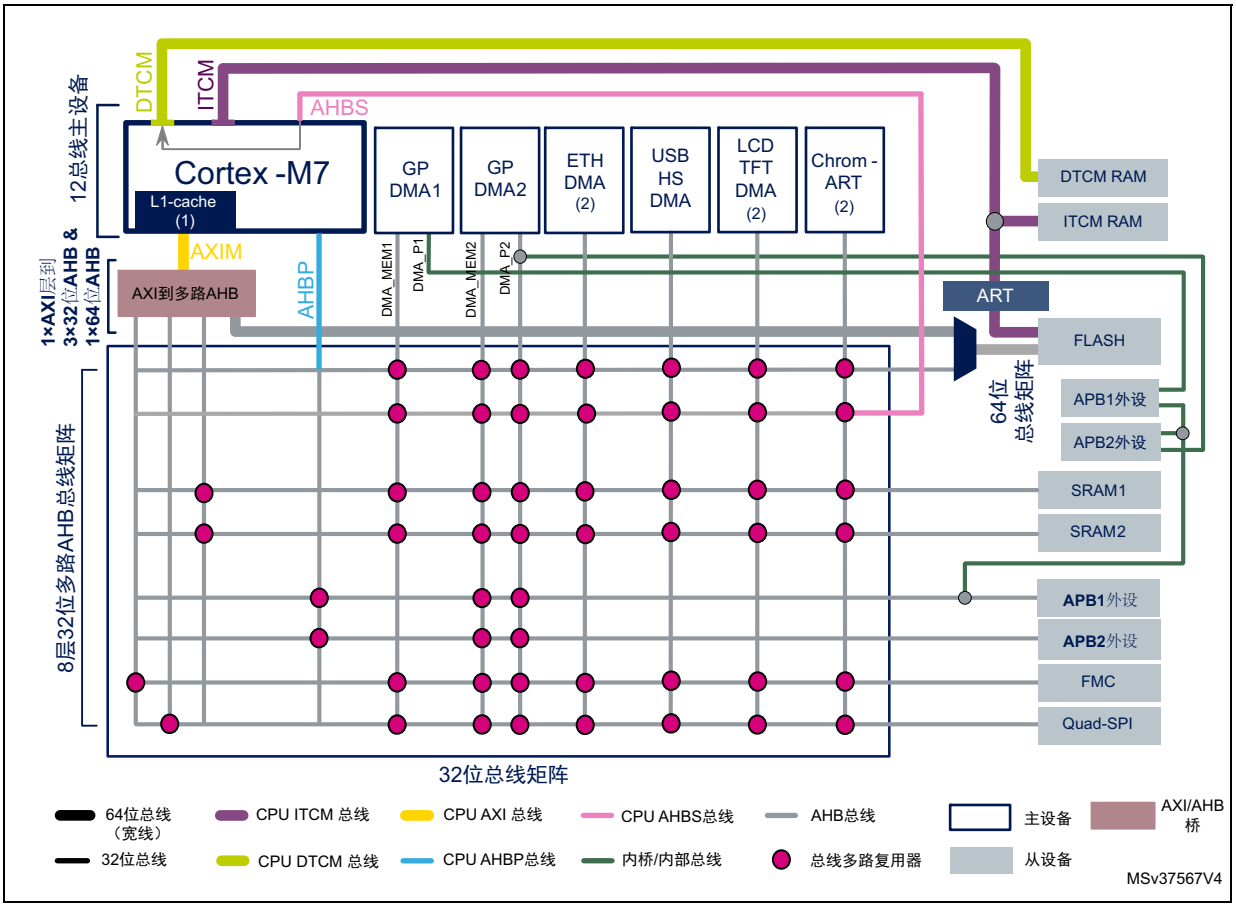
1.4 STM32F7总线矩阵

STM32F7系列器件的特征是具有一个216 MHz的总线矩阵，该总线矩阵实现内核、主设备和从设备的互连。该总线矩阵允许内核、主设备和从设备之间的多个并行访问路径存在，即使当几个高速外设同时工作时，也可以实现并发访问和高效运行。CPU和它的总线矩阵可以工作在同样的频率，即216 MHz。

一个内部的仲裁器解决了总线矩阵上的冲突和主设备的总线并发访问。仲裁器采用轮询调度算法。

*图 1*显示了STM32F7系列器件的整体系统架构和总线矩阵连接。

图1. STM32F7系列系统架构



1. I/D缓存空间大小：
   - 对于STM32F74xxx和STM32F75xxx器件：4 KB。
   - 对于STM32F72xxx和STM32F73xxx器件：8 KB。
   - 对于STM32F76xxx和STM32F77xxx器件：16 KB。
2. 主设备不可用于STM32F72xxx和STM32F73xxx器件。

STM32F7总线矩阵互连：

- 十二个总线主设备或发起者：
  - 三条从AXI转AHB桥出来的32位AHB总线
  - -一条从AXI转AHB桥出来的64位AHB总线，连接到嵌入式FLASH存储器
  - Cortex®-M7 AHB外设总线
  - DMA1存储器总线
  - DMA2存储器总线
  - DMA2外设总线
  - 以太网DMA总线(a)
  - USB OTG HS DMA总线
  - LCD-TFT控制器DMA总线(a)
  - Chrom-Art加速器®（DMA2D）存储器总线(a)
- 八个总线从设备：
  - AHB总线上的内嵌Flash存储器（用于Flash读/写访问、代码执行和数据访问）
  - Cortex®-M7 AHBS从接口（仅用于DTCM-RAM的DMA数据传输）。
  - 内部主SRAM1
  - 内部辅助SRAM2
  - AHB1外设（包括AHB至APB总线桥、APB1和APB2外设）
  - AHB2外设
  - FMC存储器接口
  - Quad SPI存储器接口

a. 不支持STM32F72xxx和STM32F73xxx器件。

1.5 STM32F7存储器

STM32F7器件根据 STM32F72xxx/STM32F73xxx，STM32F74xxx/STM32F75xxx或STM32F76xxx/STM32F77xxx器件（参考*表 3*，*表 4*和*表 5*）嵌入了不同空间大小的闪存、不同空间大小的SRAM、分散架构和外部存储器接口（如FMC和Quad-SPI）。这样的配置为用户分配应用存储资源提供了灵活性，用户可以根据需要配置，并且获得良好的性能与应用代码长度之间的折衷。

1.5.1 嵌入式 Flash

每个STM32F7系列器件都有各自的闪存空间大小（参考*表 3*、*表 4*和*表 5*）。在STM32F72xxx和STM32F73xxx器件中，闪存可以128位宽进行数据读取访问，而在STM32F74xxx和STM32F75xxx器件中，闪存以256位宽进行数据读取访问。在STM32F76xxx和STM32F77xxx器件中，存储器模式使能（单存储区模式中为256位访问，双存储区模式中为128位访问）后，闪存可以128位宽或256位宽进行数据读取访问。

所有器件中，闪存均可通过三个主接口进行读或/和写访问。

- 64位ITCM接口：

  它通过ITCM总线（*图 2*中的路径1）连接嵌入式Flash存储器与Cortex-M7，同时用于程序执行和常量的数据读取。到Flash存储器写访问不允许通过这条总线完成。

  Flash存储器可以由CPU通过ITCM从开始地址0x00200000访问。

  由于嵌入式Flash存储器相对内核速度较慢，自适应实时加速器（ART）用来释放Cortex-M7内核的性能，同时在高达216 MHz的CPU频率时，允许来自Flash存储器的零等待执行。STM32F7ART仅适用于ITCM接口上的闪存访问。选择了存储区模式后，它可实现统一的指令缓存，并实现STM32F72xxx和STM32F73xxx器件中128位x64行的分支缓存，

  STM32F74xxx和STM32F75xxx器件中256位 x 64行的分支缓存，STM32F76xxx和STM32F77xxx器件中128/256位 x 64行的分支缓存。ART可用于指令和数据访问，这提高了顺序代码和循环的执行速度。ART同样实现了预取（ART预取）。

- 64位AHB接口：

  它通过AXI/AHB桥（*图 2*中的路径2）连接嵌入式Flash存储器和Cortex-M7。它用于代码执行、读访问和写访问。Flash存储器可以由CPU通过AXI从开始地址0x08000000访问。

- 32位AHB接口：

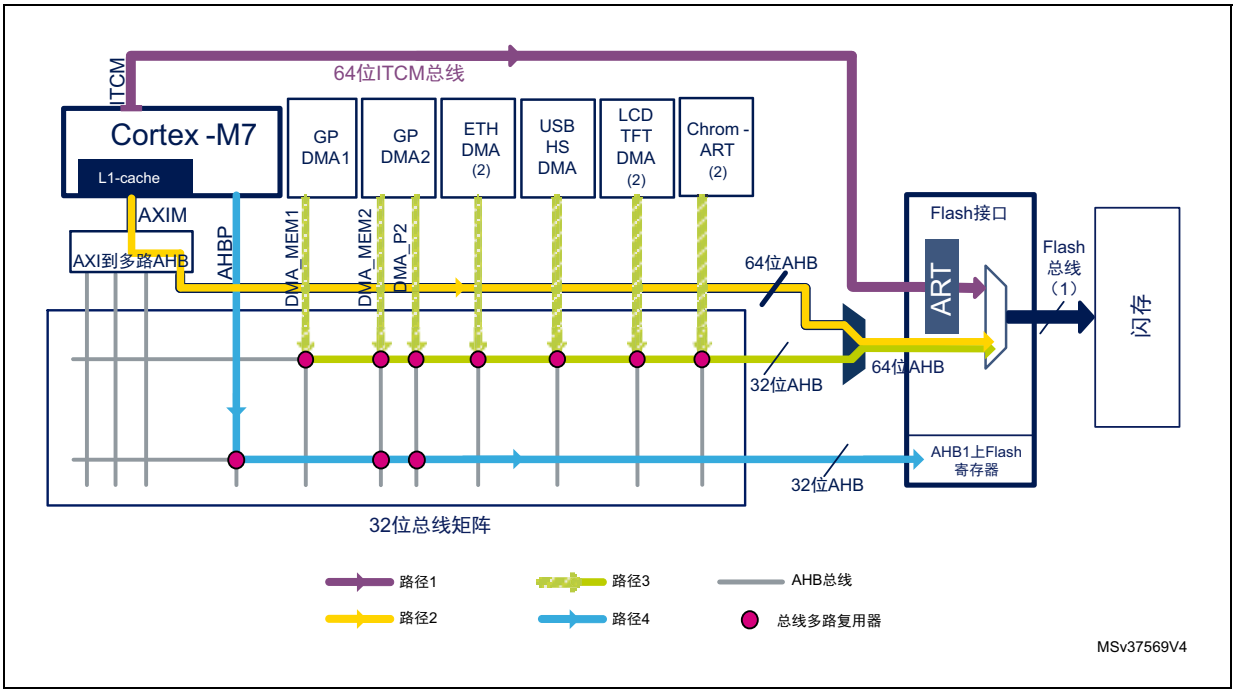
  它用于DMA和Flash存储器的传输（*图 2*中的路径3）。DMA闪存访问从地址0x08000000开始执行。

对于控制、配置、寄存器访问，Flash存储器接口可以通过AHBP/AHB1外设路径访问，这个路径是一条32位AHB总线（*图 2*中的路径4）。

如果到Flash存储器的访问在从0x0800 0000开始的地址完成，这就会自动通过AXI/AHB执行。指令或/和数据缓存应该在这种配置下使能以获得对Flash存储器的类似零等待访问。

如果到Flash存储器的访问在从0x0200 0000开始的地址完成，这就会自动通过ITCM总线执行。应使能ART加速器™以获得通过ITCM总线到Flash存储器的等效零等待状态访问。通过设置FLASH_ACR寄存器中的位9来使能ART，通过设置相同寄存器中的位8来使能ART预取。

图2. Flash存储器接口（路径：1、2、3、4）



1. 闪存宽度为：
   - 对于STM32F74xxx和STM32F75xxx器件：256比特。
   - 对于STM32F72xxx和STM32F73xxx器件：128比特。
   - 对于STM32F76xxx和STM32F77xxx器件：单存储区模式中为256比特，双存储区模式中为128比特。
2. 主设备不可用于STM32F72xxx和STM32F73xxx器件。
3. 阴影的路径意思是有多种路径可能。

1.5.2 嵌入式SRAM

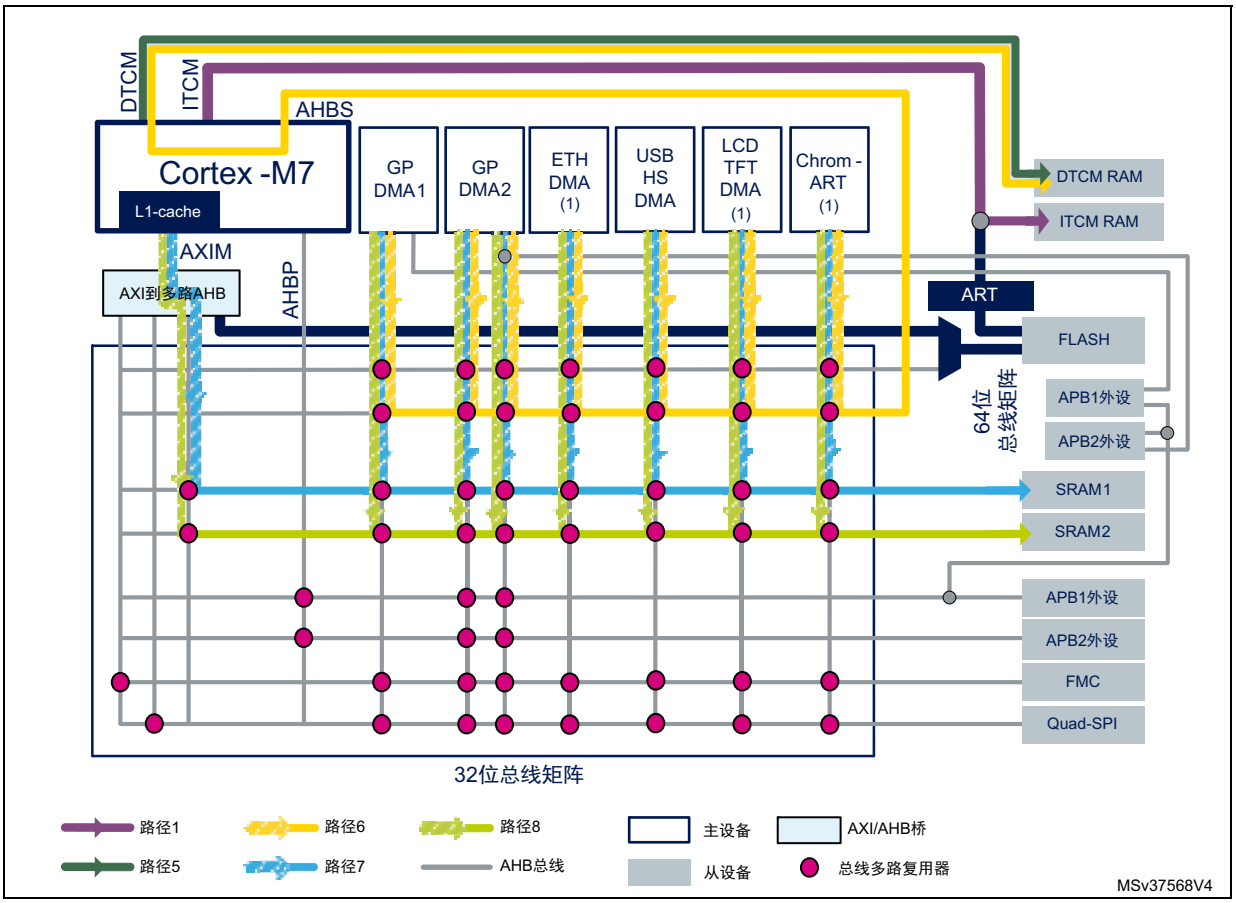
STM32F7系列器件具有一个带分散架构的大SRAM。它一共分为四个块：

- 映射到地址0x0000 0000的指令RAM（ITCM-RAM），仅可供内核访问，即通过*图 3*中的路径1。它可按字节、半字（16位）、全字（32位）或双字（64位）访问。ITCM-RAM可以在最大的CPU时钟速度下访问，没有任何延迟。ITCM-RAM由总线竞争保护，因为只有CPU可以访问这片RAM区域。
- DTCM-RAM在TCM接口上映射到地址0x2000 0000，可以被来自AHB总线矩阵上的所有AHB主设备访问：由CPU通过DTCM总线（*图 3*中的路径5）和由DMA通过内核的专用AHBS总线，即*图 3*中的路径6。它可按字节、半字（16位）、全字（32位）或双字（64位）访问。DTCM-RAM可以在最大的CPU时钟速度下访问，没有任何延迟。通过主设备（DMA）到DTCM-RAM的并发访问和它们的优先级可以被Cortex-M7中的从控制寄存器（CM7_AHBSCR寄存器）处理。相对于其他主设备（DMA），CUP具有访问DTCM-

RAM的更高优先级。更多关于这个寄存器的信息，请参考“*ARM® Cortex®-M7处理器技术参考手册*”。

- SRAM1可以被所有来自AHB总线矩阵的AHB主设备访问，也就是所有通用DMA和专用DMA。SRAM1可按字节、半字（16位）或全字（32位）访问。对于可能的SRAM1访问，请参考*图 3*（路径7）。它可以用于数据读取和代码执行。
- SRAM2可以被所有来自AHB总线矩阵的AHB主设备访问。所有通用DMA和专用DMA均可访问此存储区域。SRAM2可按字节、半字（16位）或全字（32位）访问。对于可能的SRAM2访问增加“路径”，请参考*图 3*（路径8）。它可以用于数据读取和代码执行。

图3. DMA对一些内部存储器的不同访问（路径：5，6，7，8）



1. 主设备不可用于STM32F72xxx和STM32F73xxx器件。
2. 阴影的路径意思是有多种路径可能。

表 3，*表 4*和*表 5*总结了STM32F7系列器件的内部存储器映射和存储器空间大小：

表3. STM32F74xxx/STM32F75xxx器件 的内部存储器总结

存储器类型	存储区域	起始地址	结束地址	大小	访问接口
FLASH	FLASH-ITCM	0x0020 0000	0x002F FFFF	1 MB	ITCM（64位）
	FLASH-AXIM	0x0800 0000	0x080F FFFF		AHB（64位） AHB（32位）
RAM	DTCM-RAM	0x2000 0000	0x2000 FFFF	64 KB	DTCM（64位）
	ITCM-RAM	0x0000 0000	0x0000 3FFF	16 KB	ITCM（64位）
	SRAM1	0x2001 0000	0x2004 BFFF	240 KB	AHB（32位）
	SRAM2	0x2004 C000	0x2004 FFFF	16 KB	AHB（32位）

表4. STM32F72xxx/STM32F73xxx器件 的内部存储器总结

存储器类型	存储区域	起始地址	结束地址	大小	访问接口
FLASH	FLASH-ITCM	0x0020 0000	0x0027 FFFF	512 KB	ITCM（64位）
	FLASH-AXIM	0x0800 0000	0x0807 FFFF		AHB（64位） AHB（32位）
RAM	DTCM-RAM	0x2000 0000	0x2000 FFFF	64 KB	DTCM（64位）
	ITCM-RAM	0x0000 0000	0x0000 3FFF	16 KB	ITCM（64位）
	SRAM1	0x2001 0000	0x2003 BFFF	176 KB	AHB（32位）
	SRAM2	0x2003 C000	0x2003 FFFF	16 KB	AHB（32位）

表5. STM32F76xxx/STM32F77xxx器件的内部存储器总结

存储器类型	存储区域	起始地址	结束地址	大小	访问接口
FLASH	FLASH-ITCM	0x0020 0000	0x003F FFFF	2 MB(1)	ITCM（64位）
	FLASH-AXIM	0x0800 0000	0x081F FFFF		AHB（64位） AHB（32位）
RAM	DTCM-RAM	0x2000 0000	0x2001 FFFF	128 KB	DTCM（64位）
	ITCM-RAM	0x0000 0000	0x0000 3FFF	16 KB	ITCM（64位）
	SRAM1	0x2002 0000	0x2007 BFFF	368 KB	AHB（32位）
	SRAM2	0x2007 C000	0x2007 FFFF	16 KB	AHB（32位）

1. 单存储区中为2 MB，双存储区中为2 x 1 MB。

1.5.3 外部存储器

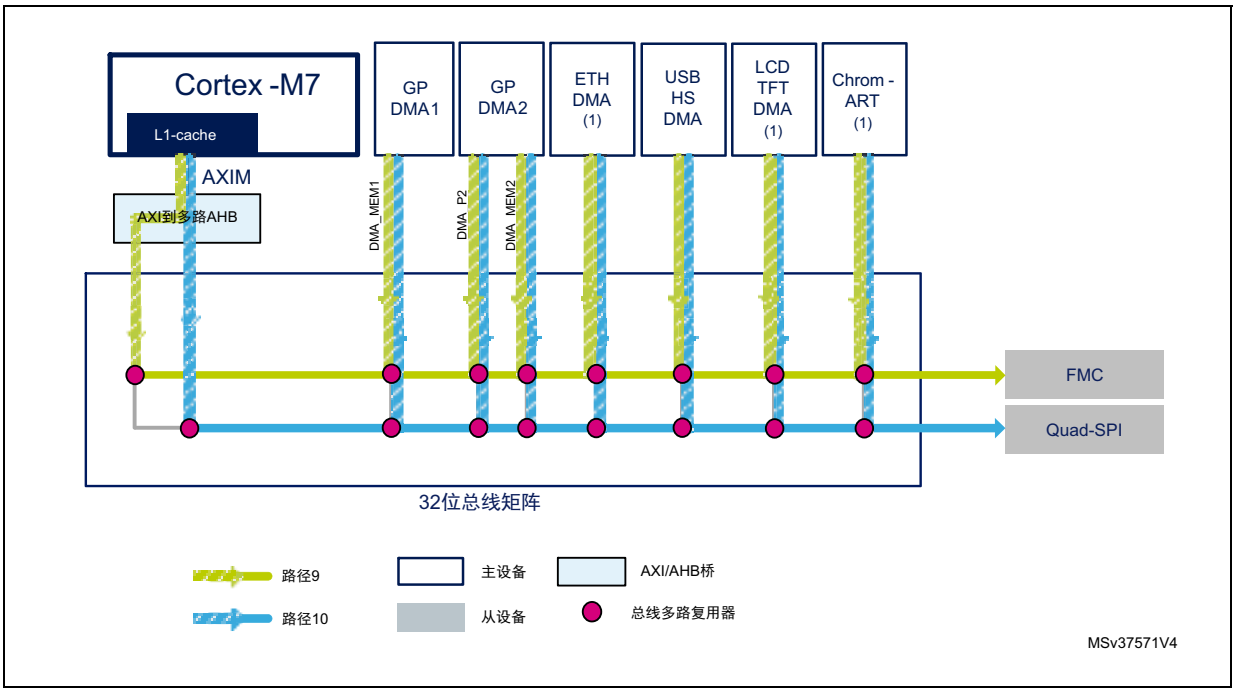
除了内部存储器和存储控制器，比如USB和SDMMC之外，用户可以用可变存储控制器（FMC）和Quad SPI控制器扩展STM32F7的存储。

*图 4*显示了通过AXI/AHB总线互连CPU、不同的DMA和外部存储的可能路径。如*图 4*所示，外部存储可以利用Cortex®-M7缓存的优点，因此无论是数据载入/存储或者代码执行位置，都可以获得最高性能。这样性能与和大存储容量就可以相容。

*图 4*中的路径9显示了使用CPU或DMA可能的FMC访问。

*图 4*中的路径10显示了使用CPU或DMA可能的Quad-SPI访问。

图4. 外部存储器接口（路径：9、10）

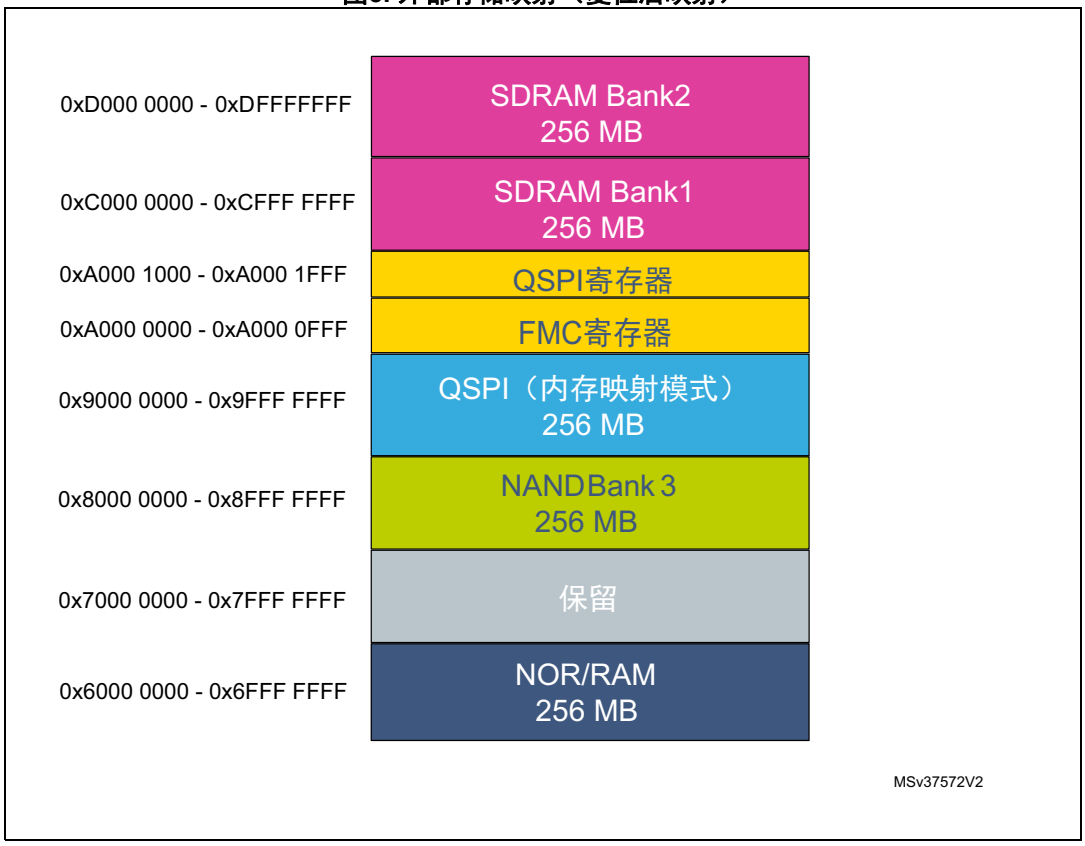


1. 主设备不可用于STM32F72xxx和STM32F73xxx器件。
2. 阴影的路径意思是有多种路径可能。

所有的外部存储器可以被所有的主设备访问（CPU和DMA）。因此允许存储器/存储器DMA传输或者外设/存储DMA传输。

*图 5*汇总了复位之后的外部存储的存储映射和它们的地址范围（SYSCFG_MEMRMP寄存器中的SWP_FMC[1:0]字节设置为0）。

图5. 外部存储映射（复位后映射）



可变存储控制器（FMC）接口

STM32F7 FMC控制器连接存储映射器件，包括SRAM、ROM、NOR/NAND Flash存储和SDRAM器件。它用于程序执行（除了NAND Flash之外）或者数据载入/存储操作。

STM32F7 FMC特征：

- 4个同时支持不同存储器的存储区域。
- 每个存储区域有独立的片选控制
- 每个存储区域可独立配置
- 可对时序进行编程，以支持各种器件
- 8/16/32位数据总线
- 外部异步等待控制
- 有两个bank提供与同步DRAM（SRAM）存储器的连接

所有FMC外部存储器与控制器共享地址、数据和控制信号。

每个外部器件可以通过唯一的片选信号访问。FMC一次只能访问一个外部器件。

两个默认的SDRAM存储区域不可缓存。因此即使缓存使能，数据和指令也不会通过缓存。为了利用缓存加速的优点，SDRAM存储区域可以由0xC000 0000和0xD000 0000分别重新映射到0x6000 0000和0x7000 0000，这是默认的可缓存区域。通过设置SYSCFG_MEMRMP寄存器中的字段SWP_FMC[1:0]=01来完成。如果重映射在应用中不适用，Cortex®-M7 MPU可以用来修改默认的SDRAM存储器的性质，使其可以缓存。

所有可能链接到FMC的外部存储器将会受益于数据和L1缓存（*图 4*中的路径9），使它们有更大的数据或/和代码长度，达到最高性能。

Quad-SPI接口

STM32F7系列器件集成了一个Quad SPI存储接口，它是一个专用的通信接口，连接单、双或者Quad SPI Flash存储器。这种多宽度接口支持传统的单线SPI串行输入输出，也支持双线和四线的串行命令。除此之外，接口支持双倍数据速率（DDR）读出命令，这意味着地址传输和地址读出在通信时钟的两个边沿执行。它允许两倍的数据/指令吞吐率，因此提高了对外部Quad SPI Flash存储器的访问效率。

它可以在以下三种模式下工作：

- 直接模式：使用Quad SPI寄存器执行全部操作
- 状态轮询模式：周期性读取外部FLASH状态寄存器，而且标志位置1时会产生中断
- 存储映射模式：外部Flash进行存储映射，系统将其视作内部存储器

在存储映射模式下，STM32F7 Quad SPI接口可以管理高达256 MB的Flash存储器，存储地址从0x9000 0000到0X9FFF FFFF。它映射到一个可执行区域，因此不需要重映射。

相比FMC，Quad-SPI允许连接一个外部Flash存储器，这种连接成本较低（小封装使得PCB面积减小）并且使用GPIO较少。对于任意闪存大小，在单－四模式（4位）下使用6个GPIO，在双－四模式（８位）下使用10个GPIO。

如*图 4*（路径10）所示，Quad SPI映射到一个AHB上的专用层，并且可以受益于L1-缓存。这允许以良好的性能执行代码并且从Quad-SPI载入数据。

Quad-SPI同样可以通过所有的在AHB总线矩阵上的主设备访问，特别是Chrom-ART加速器®和LCD-TFT，使高效的数据传输能够完成，特别是图像，因为图像应用中需要高帧率的显示。

请参考*STM32微控制器上的Quad-SPI（QSPI）接口*应用笔记（AN4760），了解如何使用STM32微控制器来配置、编程以及读取外部Quad-SPI存储器。

1.6 DMA

STM32F7系列器件集成了两个通用的直接存储器访问（GP DMAx），这可以提供存储器之间和存储与外设之间的高速数据传输。DMA用于对CPU减荷，并且允许一些传输在内核处于低功耗模式时进行。每个DMA具有8个数据流，每个数据流具有8个通道。

STM32F7系列器件还嵌入了用于以太网、USB OTG、LCD-TFT和Chrom-ART accelerator® 的其他专用DMA，STM32F72xxx和STM32F73xxx器件除外，它们仅嵌入了一个专用DMA（USB OTG）。所有的DMA可以访问下列内部存储器：嵌入式Flash存储器、SRAM1、SRAM2和DTCM，通过Cortex®-M7的AHBS总线访问。所有DMA同样可以通过FMC和Quad SPI控制器通过总线矩阵访问外部存储器。

AHBS上无法访问ITCM总线，因此不支持ITCM RAM上的DMA数据传输。对于ITCM接口上从/到闪存的DMA传输，所有传输都必须通过AHB总线进行，也就是所有传输地址都将自动从ITCM地址转换为AXI地址。

可能的GP DMA传输列举如下：

- GP DMA1传输：
  - DMA1不能寻址AHB1/AHB2外设
  - DMA1只能完成APB1与存储器之间的相互传输
- GP DMA2传输：
  - DMA2可以完成所有可能的传输

由于采用了AXI到多路AHB桥和总线矩阵架构， 因此可以有效管理CPU和DMA对不同从设备的并发访问，降低了延迟。当CPU和其他系统主设备试图同时访问DTCM-RAM时，Cortex-M7中也可以在TCU层对LSU（加载/存储单元）和AHB从设备总线访问之间的争用进行管理。只有一个主设备可以访问AHBS，这由总线矩阵仲裁进行管理。

CPU和DMA对同一存储器的访问可能会影响系统性能。

如果CPU和DMA不是同时访问同一存储器，则不存在争用。因此，如果CPU没有激活其他主设备，则性能保持不变（参见*图 6*）。

如果CPU和一个或多个主设备之间存在对存储器的并发访问（见*图 7*），则根据DMA访问存储器的速度（相比于外部存储器，对内部存储器的访问相对较快）和读/写并发是否完成，性能会降低。

由并发主设备访问产生的延迟取决于多种因素，其中一些可在*第 3.2 节：使用DMA的CPU存储器范围性能*中进行处理。

图6. DMA与CPU之间无存储器并发访问

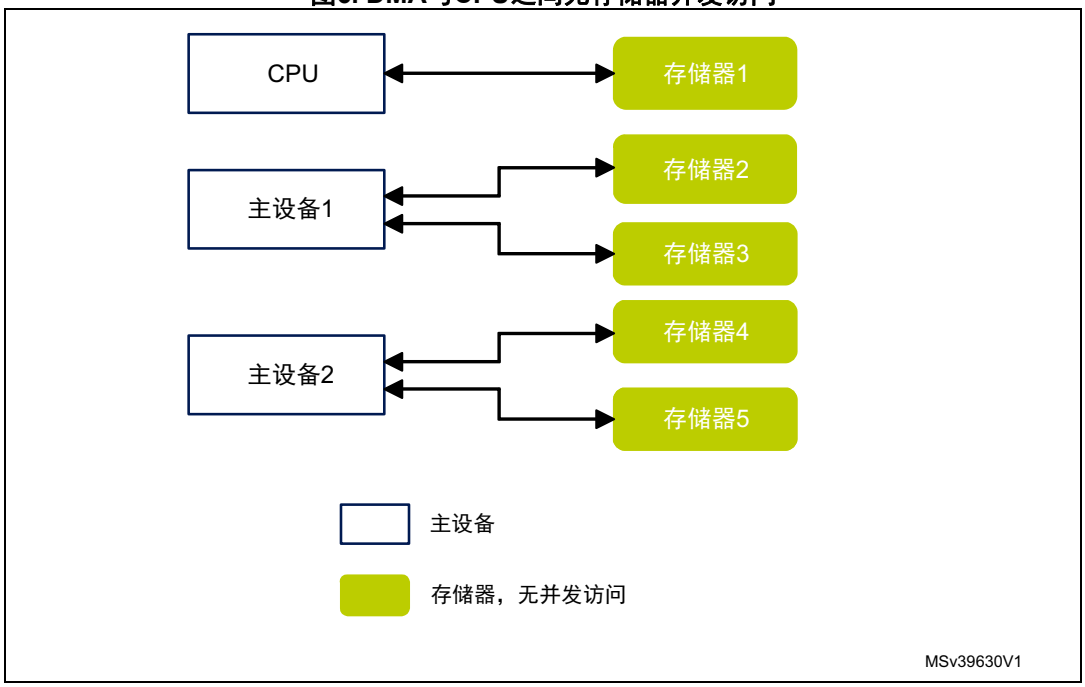
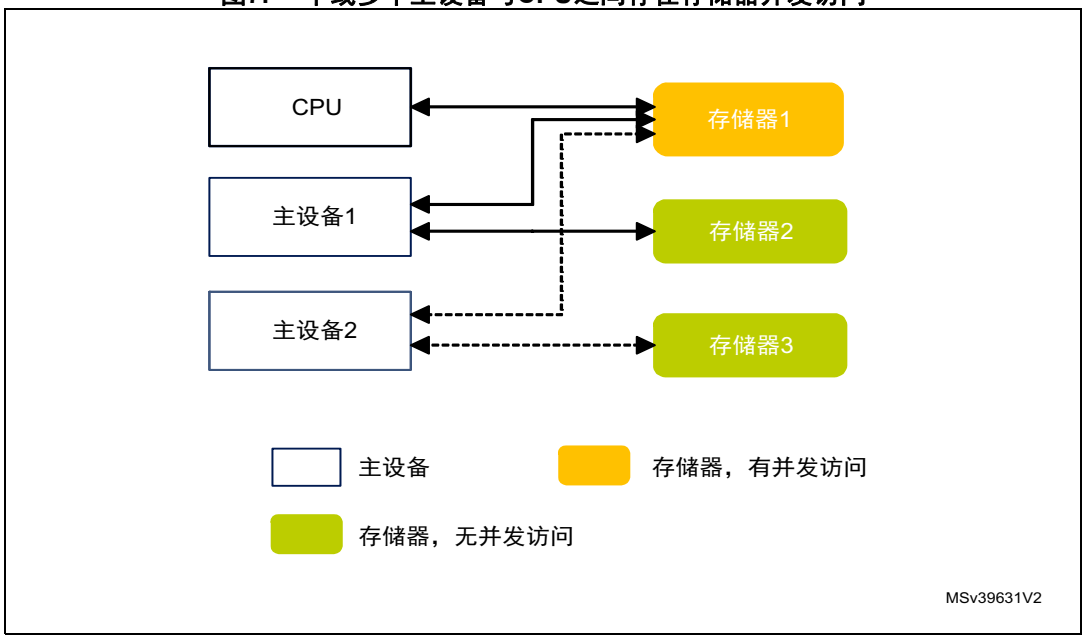


图7. 一个或多个主设备与CPU之间存在存储器并发访问



*第 2.3 节*展示了不同场景下使能多个主设备（DMA）时的系统性能演示。

*第 3.2 节*提供了基于图6和7的典型场景所得到的不同结果和分析。

1.7 STM32F7系列器件之间的主要差异（从架构角度看）

*表 6*仅总结了STM32F7系列器件之间架构相关的差异（不包括外设差异）

表6. STM32F7系列器件之间的主要差异

	STM32F74xxx/ STM32F75xxx	STM32F72xxx/ STM32F73xxx	STM32F76xxx/ STM32F77xxx
Cortex-M7版本	r0P1	r1P0	r1P0
指令和数据缓存空间大小	4 KB	8 KB	16 KB
FPU	单精度浮点单元	单精度浮点单元	单精度和双精度浮点单元
Flash大小	1 MB	512 KB	2 MB
存储区模式/Flash宽度	单存储区，采用256位访问	单存储区，采用128位访问	– 单存储区中：256位访问 – 双存储区中：128位访问
DTCM-RAM	空间大小：64 KB @ 0x20000000	空间大小：64 KB @ 0x20000000	空间大小：128 KB @ 0x20000000
SRAM1	空间大小：240 KB @ 0x20010000	空间大小：176 KB @ 0x20010000	空间大小：368 KB @ 0x20020000
SRAM2	空间大小：16 KB @ 0x2004C000	空间大小：16 KB @ 0x2003C000	空间大小：16 KB @ 0x2007C000
ETH-DMA，LCD-TFT-DMA和Chrom-ART	可提供	不可用	可提供

2 典型应用

本应用笔记提供了两个软件示例，首先显示了对于数据存储或代码执行以及对于内部或外部存储器进行的CPU访问的STM32F7性能。其次，这些示例显示了典型场景中CPU加载/存储数据到一个存储器时对DMA使用的影响。

使用的是FFT示例，由CMSIS库提供。stm32f7_performances项目可以用来作为一个框架，用户可以在项目中集成自己的应用。对于stm32f7_performances_DMAs项目可以使用同样的示例，不过幅度计算除外，为了分配更多RAM用于DMA传输，幅度计算已被删除。

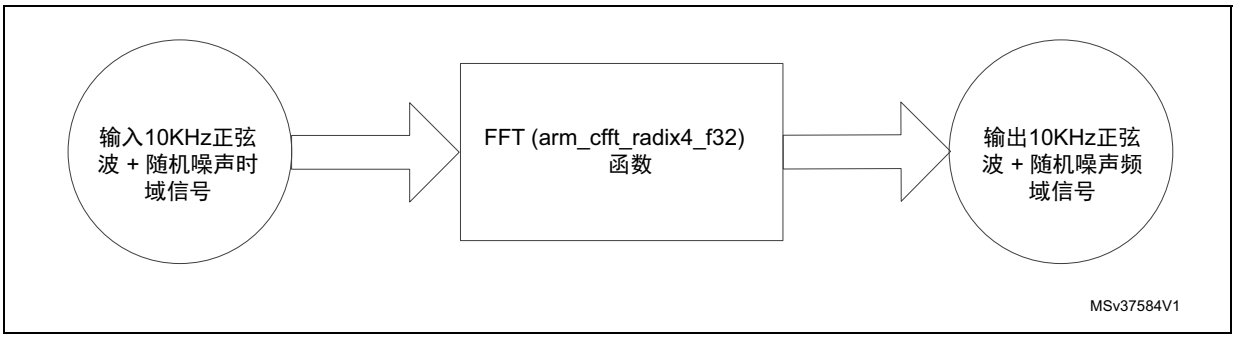
2.1 FFT演示

使用FFT示例是因为它受益于Cortex-M7的浮点单元，并且包括多个环路、数据载入/存储操作，可以在不同的路径/存储完成。代码可以从内部或者外部存储执行。示例包括使用复杂FFT、复数幅度和最大函数计算输入信号频域的最大能量仓。它使用FFT 1024点，计算基于单精度浮点。输入信号是一个10 KHz的混有白噪声的正弦波。*图 8*显示了转换的框图。

FFT处理所花费的周期数也是基于系统节拍定时器进行计算的。该示例已运行于STM32F732E-DISCO，
STM32756G-EVAL和STM32769I-EVAL板。结果通过UART显示在Hyperterminal上，对于所有电路板可显示于IDE printf指示器上，对于STM32F7xxxx-EVAL板，也可显示在LCD-TFT上。

FFT演示显示了当前项目配置、系统频率、缓存的不同配置、ART、ART预取（开/关）和有外部存储（SDRAM或Quad-SPI）情况下的存储配置。

图8. FFT示例框图



2.2 CPU存储器访问演示的项目配置

Keil MDK-ARM、IAR Embedded Workbench和STM32工具链的System Workbench提供了CPU存储器访问演示。此项目的8/9配置可支持不同的数据选择和代码位置。

配置采用以下的规则命名：

N-ExecutionRegionPath_rwDataRegion 其中：

N：配置的序号。

ExecutionRegionPath：存储地址和代码执行路径。用户必须区分执行区域和载入区域。执行区域是应用程序执行的存储器位置。加载区域是最初由Flash加载器所加载并稍后复制（通过某些工具链自动生成的子程序）的应用程序所在的存储器位置，如果这两个地址位置不同，则在执行区域。

rwDataRegion：RW/Zero初始化数据、堆栈的存储位置。

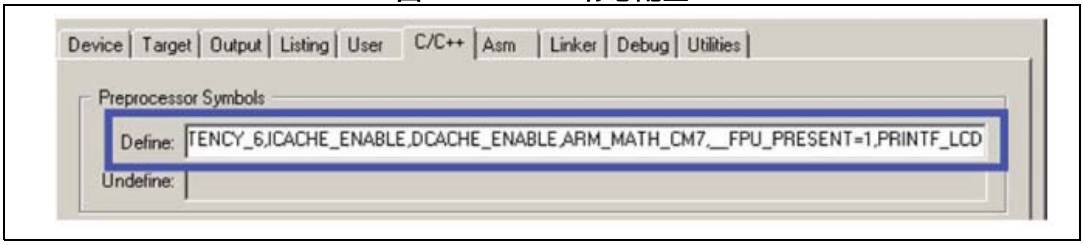
提出以下配置：

- ***1-FlashITCM_rwRAM-DTCM***：在7个等待状态时，程序从Flash-ITCM开始执行，ART和ART预取使能并且数据载入/存储在DTCM-RAM。
- ***2-FlashITCM_rwSRAM1***：在7个等待状态时，程序从Flash-ITCM开始执行，ART和ART预取使能并且数据载入/存储在SRAM1且D-缓存使能。
- ***3-FlashAXI_rwRAM-DTCM***：在7个等待状态时，程序从Flash-AXI开始执行，I-缓存使能并且数据载入/存储在DTCM-RAM，D-缓存使能。因为一些常量从Flash-AXI载入来计算FFT，所以数据缓存也被使能。
- ***4-FlashITCM_rwRAM-DTCM***：在7个等待状态时，程序从Flash-AXI开始执行，I-缓存使能并且数据载入/存储在SRAM1，D-缓存使能。
- ***5-RamITCM_rwRAM-DTCM***：程序从ITCM-RAM开始执行，数据载入/存储在DTCM-RAM。在这个配置中什么都不应使能。
- ***6-Quad SPI_rwRAM-DTCM***：该配置包括两种情况，Quad-SPI Flash存储器运行在54 MHz的频率，DDR模式使能：
  - 情况1（***6_1-Quad SPI_rwRAM-DTCM***）：代码从Quad-SPI Flash存储器开始执行，I-缓存使能并且数据载入/存储在DTCM-RAM。因为FFT处理使用大量常数，在这种情况下，只读数据放置在Quad-SPI Flash存储器中。因此数据缓存也被使能。
  - 情况2（***6_2-Quad SPI_rwRAM-DTCM***）：只有代码在Quad-SPI Flash存储器中存储和执行。只读数据存储在Flash-ITCM中。读/写数据存储在DTCM-RAM中。I-缓存、ART和ART预取在这种情况下使能。
- ***7-ExtSDRAM-Swapped_rwDTCM***：此配置不可用于STM32F723E-Discovery工作空间。这里，程序从FMC-SDRAM开始执行，数据载入/存储在DTCM-RAM。I-缓存和D-缓存使能。注意在这种配置下，常量数据存储在SDRAM中，这就是为什么D-缓存使能的原因。在这种配置下，CPU的时钟是200 MHz，而SDRAM运行频率是100 MHz。

- **7-ExtPSRAM_rwDTCM**：此配置仅可用于STM32F723E-Discovery工作空间。这里，程序从FMC-PSRAM开始执行，数据载入/存储在DTCM-RAM。I-缓存和D-缓存使能。在这种配置下，常量数据存储在PSRAM中，这就是为什么D-缓存也使能的原因。
- **8-FlashAXI-ROFTCM_rwSRAM1**：此配置仅可用于STM32F723E-Discovery工作空间。这里，只有代码是存放在Flash-AXI中并从Flash-AXI执行的。只读数据从Flash-ITCM获取。读/写数据位于SRAM1中。I-Cache，D-Cache，ART和ART-prefetch使能。

每个配置都有自己的标志设置。这些标志在配置项目中是可以设置的。*图 9* 显示了这些为MDK-ARM工具链定义的标志。

图9. MDK-ARM标志配置



对于所有的配置，代码优化配置为3级优化，时间优化优先。

项目标志描述：

- **DCACHE_ENABLE**：如果在配置项目中定义，则数据缓存使能。
- **ICACHE_ENABLE**：如果在配置项目中定义，则指令缓存使能。
- **ART_ENABLE**：如果在配置项目中定义，则ART加速器使能。
- **PF_ART_ENABLE**：如果在配置项目中定义，则ART加速器中的预取使能。
- **FLASH_WS**：配置内部Flash等待周期的数量：
  FLASH_WS=FLASH_LATENCY_X，其中X= 0（0等待周期）到15（15等待周期）。
- **DATA_IN_ExtPSRAM**：如果在配置项目中定义，外部PSRAM被配置成用于数据存储或者代码执行。
- **PSRAM_MEM_BUS_WIDTH**：配置外部PSRAM的总线带宽，配置如下所示：
  PSRAM_MEM_BUS_WIDTH= FMC_NORSRAM_MEM_BUS_WIDTH_X，其中X= 8、16或者32。STM32F723E-DISCO板上提供的PSRAM是16位PSRAM。
- **DATA_IN_ExtSDRAM**：如果在配置项目中定义，SDRAM被配置成用于数据存储或者代码执行。
- **SDRAM_MEM_BUS_WIDTH**：配置外部SDRAM的总线带宽，配置如下所示：
  SDRAM_MEM_BUS_WIDTH= FMC_SDRAM_MEM_BUS_WIDTH_X，
  其中X= 8、16或者32。

- **SDRAM_ADDRESS_SWAPPED**：如果在配置项目中定义，SDRAM地址从0xC000 0000到0x6000 0000重新映射。重映射SDRAM允许将其放在可缓存的区域。
- **DATA_IN_QSPI**：如果在配置项目中定义，Quad-SPI Flash存储器被配置成用于只读数据存储或代码执行。
- **QSPI_CLK_PRESCALER**：定义了Quad-SPI时钟预分频器，配置如下：
  ***QSPI_CLK_PRESCALER*=X**，其中X = 0到255。
- **QSPI_DDRMODE**：如果在配置项目中定义，Quad-SPI配置在DDR模式。
- **QSPI_INSRUCTION_1_LINE，QSPI_INSRUCTION_4_LINES**：第一个标志用于配置Quad SPI Flash指令发送为一线模式，而第二个标志配置指令发送为四线模式。如果不存在标志，Quad-SPI将会被配置为一行指令。这些标志不可用于STM32F723E-DISCO板。
- **QSPI_XIP_MODE**：如果在配置项目中定义，Quad-SPI配置在XIP模式，也就是直接在芯片内执行。注意当缓存使能时，XIP模式对性能没有影响。

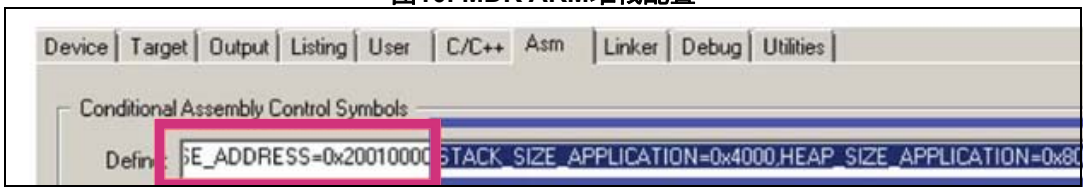
  PRINTF_LCD，PRINTF_UART，PRINTF_VIEWER：这些标志用于显示示例的结果，分别在LCD、hyperterminal （115200、7位、奇校验、一个停止位、没有HW流程控制）或者在IDE printf显示器上显示。
  对于STM32F723E-DISCO板，只应使用PRINTF_UART和PRINTF_VIEWER。更多关于如何使用不同显示模式的详细信息，请参考X-CUBE-32F7PERF嵌入式软件包中相应项目所属的自述文件。

基于这些模板，用户可以创造新的代码执行/数据存储位置配置，通过合并足够的设置，修改分散文件或链接器和设置足够的Flash加载器。

还要注意，对于MDK-ARM工具链，为了修改分散文件中的RAM区域，即堆栈区域，用户必须相应地修改ASM菜单中的堆栈空间大小。分散文件中的区域大小不是主应用程序的实际堆栈大小。用户必须对STACK_SIZE_APPLICATION和HEAP_SIZE_APPLICATION标志值进行修改，以强制使它们的值与分散文件中配置的堆栈大小区域一致。

*图 10*显示了修改这些标志的地方（蓝色框标出）。还有一个在外部存储用来数据存储时使用的初始堆栈指针。它的大小是1 KB，可以被startup_stm32f723xx.s、startup_stm32f756xx.s和startup_stm32f769xx.s启动文件中的Stack_Size_Init变量修改。初始堆栈指针基本地址可在ASM菜单中配置，如*图 10*粉色框所示。

图10. MDK ARM堆栈配置



不同配置的分散文件放置在演示项目的*MDK-ARM\scatter_files*路径下。

对于IAR（EWARM）工具链，链接器位于EWARM \ icf_files下。

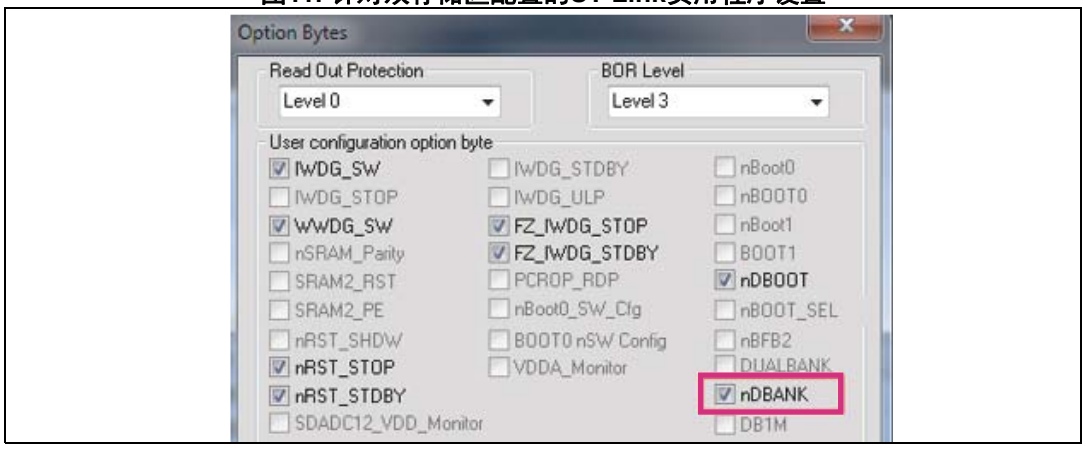
对于System Workbench工具链，链接器位于SW4STM32\<project folder configuration>下。

同样注意，在模板项目中，中断可从内部Flash存储器执行：只使用系统节拍定时器中断。

在STM32F769I EVAL板的情况下，该项目不包含双存储区配置环境。但是，用户可以在单存储区中使用相同的项目，不过必须按照以下两个步骤：

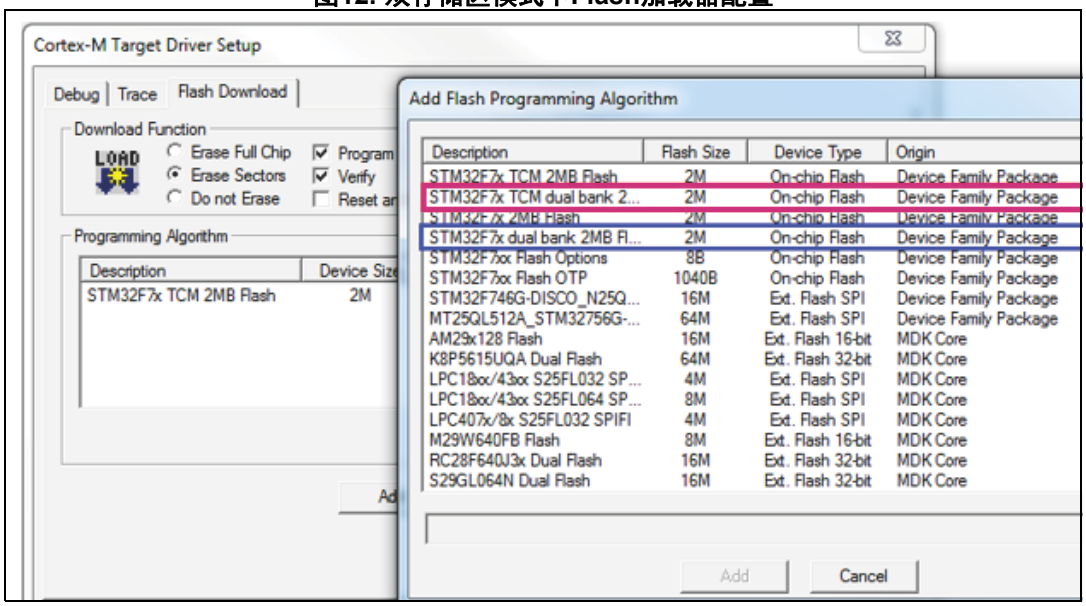
1. 使用ST-Link实用程序（v3.8.0或更高版本）配置双存储区中的闪存。请参见*图 11*。
   取消选中nDBANK框，将闪存设置为双存储区模式。

图11. 针对双存储区配置的ST-Link实用程序设置



2. 在工具链中设置Flash加载器双存储区。请参见*图 12*。
   如果用户使用Flash-TCM，则必须选择Flash加载器（粉色框）。如果使用Flash-AXI，则必须选择Flash加载器（蓝色框），如*图 12*中所示。

图12. 双存储区模式下Flash加载器配置



注： *用户可通过恢复ST-Link/Flash加载器原有配置返回到单存储区配置。*

2.3 DMA激活的CPU存储器访问演示的项目配置

Keil MDK-ARM、IAR Embedded Workbench和STM32工具链的System Workbench提供了DMA激活演示。项目配置基于与先前*第 2.2 节*中所述相同的模板。项目具有三种配置：

1. FlashITCM_rwRAM-DTCM
2. FlashITCM_rwSRAM1
3. FlashAXI_rwRAM-DTCM

除了以下差别之外，DMA激活演示使用与CPU存储器访问演示相同的配置，差别如下：

- SDRAM访问用于DMA传输的所有配置。
- 所用CPU频率为200 MHz，而非SDRAM访问所用的216 MHz。
- 修改分散文件，将FFT过程和演示其他模块所用的执行/数据存储器分开来，以防止链接器引入的不确定，避免影响结果。
- 对于所有项目，数据缓存均使能。

已用于CPU存储器访问性能（stm32f7_performances）的同样的FFT演示也用于DMA激活演示性能（stm32f7_performances_DMAs）。

仅保留FFT计算部分，已经删去了幅度计算部分，以便为DMA传输分配更多RAM。

演示中使用两个主设备执行传输：两个通用DMA：DMA1和DMA2。DMA2配置为执行存储器到存储器的传输，而DMA1用来执行存储器到SPI3以及SPI3到存储器的传输。每个DMA可以分别配置存储器源和目标。该配置在main.h文件中完成，此文件包含了执行DMA1/2传输所需的所有定义。

要激活DMA1/SPI3传输，SPI3_MOSI（PC12）必须连接到SPI3_MISO（PC11）。使用STM32F7xxxx-EVAL板的情况下，microSD卡必须从其插槽中取出。

通过三个步骤可以得到给定场景的结果：

- 步骤1：在main.h中配置不同的DMA参数
- 步骤2：检查模式：检查配置和DMA传输
- 步骤3：结果模式：得到周期数形式的结果

2.3.1 步骤1：配置不同的DMA参数

每个主设备基本上有四个参数要配置：

- DMA传输的使能/禁用
- DMA源地址配置
- DMA目标地址配置
- DMA传输大小配置

DMA传输的使能/禁用：

要使能或禁用DMA，定义“#define USE_DMA**X**”必须设置为1或0来分别使能或禁用DMA**X**。这里**X** = 1 或 2。

通过将定义USE_DMA1和USE_DMA2设置为1，用户可以分别使能每个DMA或同时使能它们。

DMA源和目标地址配置

存储器源和目标地址应通过以下定义进行配置：

对于DMA1：

#define DMA**X**_SRC_ADDRESS：存储器源的起始地址

#define DMA**X**_DST_ADDRESS：存储器目标的起始地址

DMA1通过SPI3将数据从存储器源传输到存储器目标（存储器源 -> DMA1_Stream5 -> SPI3_TX -> SPI3_RX -> DMA1_Stream0 -> 存储器目标）。

DMA2直接将数据从存储器A传输到存储器B。

DMA**X**_SRC_ADDRESS和DMA**X**_DST_ADDRESS的可能值为：

- FLASHAXI_DMA_START_ADDRESS
- FLASHTCM_DMA_START_ADDRESS
- DTCMRAM_DMA_START_ADDRESS
- SRAM1_DMA_START_ADDRESS
- SRAM2_DMA_START_ADDRESS
- SDRAM_DMA_START_ADDRESS[b]

不要修改dma_utilities.h文件中提供的这些定义，它们与分散文件中配置的映射是一致的。

DMA传输大小配置：

DMA**X**传输大小必须通过以下定义进行配置：

#define DMA**X**_TRANSFER_SIZE：传输大小，以字节为单位

传输大小必须是DMA存储器源可用大小和DMA存储器目标可用大小之间的最小值。为此，提供了宏MIN（a，b）。这样可以避免DMA访问无效的存储器区域。

DMA**X**_TRANSFER_SIZE可能值为：

- DTCMRAM_DMA_AVAILABLE_SIZE
- SRAM1_DMA_AVAILABLE_SIZE
- SRAM2_DMA_AVAILABLE_SIZE
- SDRAM_DMA_AVAILABLE_SIZE[b]
- FLASHAXI_DMA_AVAILABLE_SIZE
- FLASHTCM_DMA_AVAILABLE_SIZE

DMA2传输大小配置示例，其中SRAM1是源，DTCMRAM是目标：

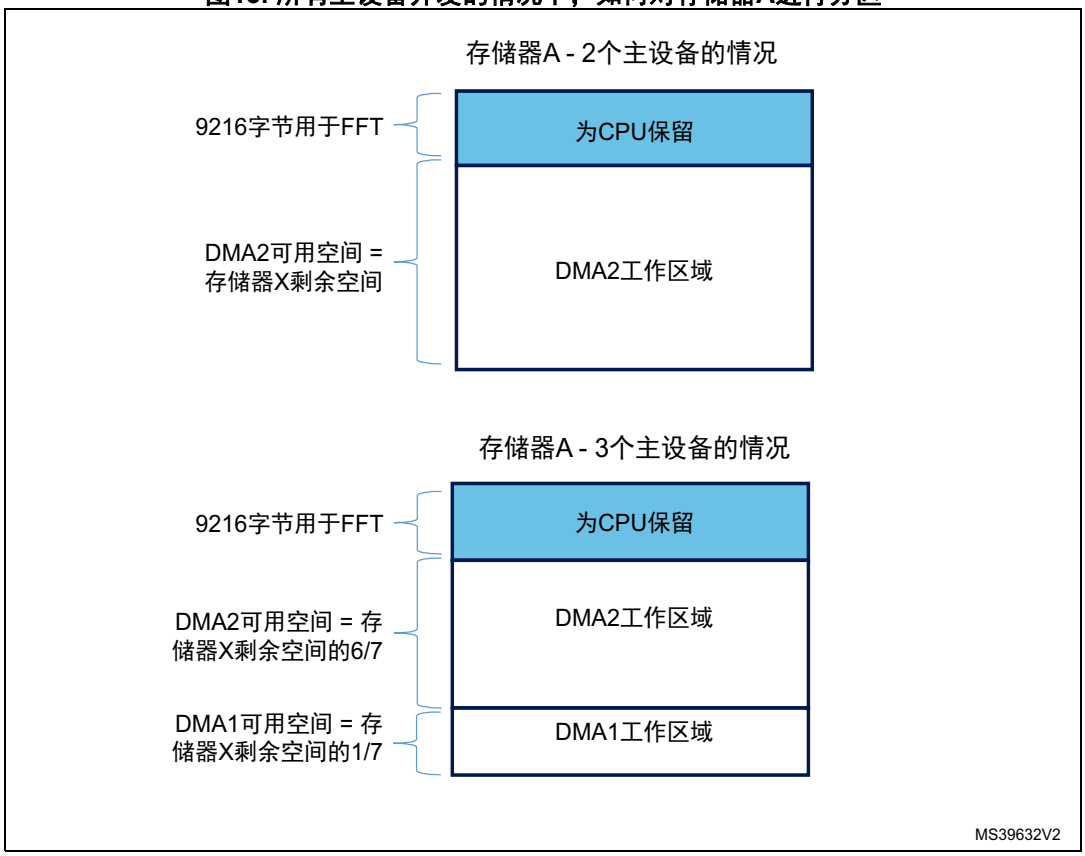
```
#define DMA2_TRANSFER_SIZE  MIN(SRAM1_DMA_AVAILABLE_SIZE,
DTCMRAM_DMA_AVAILABLE_SIZE)
```

如果最小空间大小大于64 KB，则传输大小强制为64 KB。

当用户运行DMA**X**且并发运行存储器**A**上的CPU时，必须如*图 13*中所示分割存储器。该图显示了软件中两个和三个主设备之间并发的情况下，如何对存储器**A**（例如SRAM1）分区。图中顶部的分区显示CPU和DMA2共享的存储器**A**。图中底部的分区显示三个主设备存储器**A**：CPU，DMA1和DMA2。这种分区允许每个DMA在FFT过程之后完成其各自的传输，以确保在所有FFT过程中均可保证并发性。分散文件中，为FFT保留的RAM大小为9216字节（用于CPU访问）。

b. 不可用于STM32F723E-DICO板。

图13. 所有主设备并发的情况下，如何对存储器A进行分区



下面两个示例说明了在两种场景下如何配置DMA1和DMA2（见*第 3.2 节：使用DMA的CPU存储器范围性能第 38 页*）：

配置2 /场景3：

```
/* DMA2 */
#define DMA2_SRC_ADDRESS          DTCMRAM_DMA_START_ADDRESS
#define DMA2_DST_ADDRESS          SRAM1_DMA_START_ADDRESS
#defineDMA2_TRANSFER_SIZE         MIN (DTCMRAM _REMAINING_SIZE,
SRAM1_REMAINING_SIZE)
```

配置1 /场景4：

```
/* DMA2 */
#define DMA2_SRC_ADDRESS          SRAM1_DMA_START_ADDRESS
#define DMA2_DST_ADDRESS          DTCMRAM_DMA_START_ADDRESS
#defineDMA2_TRANSFER_SIZE         MIN (SRAM1_REMAINING_SIZE,
(DTCMRAM_REMAINING_SIZE*6/7))
/* DMA1 */
#define DMA1SPI_SRC_ADDRESS       SRAM2_DMA_START_ADDRESS
#define DMA1SPI_DST_ADDRESS       (DTCMRAM_DMA_START_ADDRESS +
(DTCMRAM_REMAINING_SIZE*6/7))
#define DMA1SPI_TRANSFER_SIZE     MIN (SRAM2_REMAINING_SIZE,
(DTCMRAM_REMAINING_SIZE/7))
```

2.3.2 步骤2：检查配置和DMA传输

配置DMA的不同参数后，用户在通过运行“检查模式”中示例得到结果之前，必须检查此配置和DMA传输。

执行检查有两个目的：

- 数据的DMA传输完整性是正确的。
- DMA传输在FFT过程之后完成，确保在所有FFT过程中，DMA仍可用于总线矩阵。
- 无硬故障发生（LCD（或Hyperterminal）显示所有配置，且至少有一个LED被点亮）。

如果上述任一条件不匹配，则结果会不一致，用户必须修改DMA配置。

要以“检查模式”运行任意场景，应将以下定义设置为1：

```
#define CHECK_TRANSFER              1
```

然后编译并运行示例。

如果满足了三个检查条件，则LCD（或Hyperterminal）会显示当前配置，且绿色LED亮。这种情况下，所选场景已准备好进行步骤3，可获取结果。

如果红色LED亮，则至少一个DMA传输未正常执行。错误信息显示在LCD上（或Hyperterminal）上，表明哪个DMA发生错误以及错误数据偏移量。在此阶段，用户必须验证DMA所用的存储器地址范围，并重新编译该示例，使得在进入步骤3之前只有绿色LED亮。SPI1传输的情况下，检查SPI3_MOSI/SPI3_MISO连接。

如果红色LED闪烁，则数据传输完整性是正确的，但是在FFT传输完成之前至少有一个DMA传输已完成。显示一条错误信息，说明哪个DMA有问题。在此阶段，用户必须增加此DMA的传输大小。

如果发生了硬故障或应用程序崩溃，则表示DMA覆盖了应用程序所用的数据，包括堆栈在内。用户必须注意DMA访问的地址选择，主要是当CPU、DMA1和DMA2访问同一个存储器而再现*图 7*中所示的并发时。

2.3.3 步骤3：得到结果

如果只有绿色LED亮，并且配置显示在LCD上，那么结果将保持一致，并准备好进行当前场景。

在此阶段，用户可以“结果模式”重新编译示例（CHECK_TRANSFER=0），加载并运行它。LCD（或者STM32F723E-DISCO板情况下的hyperterminal）显示了当前配置，并在显示器底部显示出周期数。

3 结果和分析

3.1 CPU存储器访问性能

本节解释每个特性激活，随后是使用的配置，同时将会给出获得的FFT处理消耗的周期数结果。

对于STM32F74xxx和STM32F75xxx器件，结果由KEIL MDK-ARM V5.14.0工具链、STM32F7xx套件（版本1.1.0）和Cube固件（版本1.0.0）获得。

对于STM32F76xxx和STM32F77xxx器件，结果由KEIL MDK-ARM V5.17.0工具链、STM32F7xx套件（版本2.6.0）和Cube固件（版本1.4.0）获得。

对于STM32F72xxx和STM32F73xxx器件，结果由KEIL MDK-ARM V5.21.1工具链、STM32F7xx套件（版本2.9.0）和Cube固件（版本1.6.0）获得。

MDK-ARM代码优化配置是3级：为时间优化。

对于所有从Flash-ITCM开始执行代码的配置，ART加速器和ART预取均使能，因为指令和/或只读数据（常量）流经ITCM（*图 2*中的路径1）。下列配置属于这种情况：

- “1-FlashITCM_rwRAM-DTCM”
- “2-FlashITCM_rwSRAM1”
- "6_2-QuadSPI_rwRAM-DTCM"(c)
- "8-FlashAXI-ROFTCM_rwSRAM1"(c)(d)

对于所有从Flash-AXI（AXI/AHB）开始执行代码的配置，指令缓存使能，因为指令流经了*图 2*中的路径2。在这种配置下，只读数据（常量）同样流经了路径2（同样的图片），这就是为什么数据缓存也使能的原因。下列配置属于这种情况：

- “3-FlashAXI_rwRAM-DTCM”
- “4-FlashAXI_rwSRAM1”

在“5-RamITCM_rwRAM-DTCM”配置情况下，都不使能，因为代码和数据分别存储在ITCM-RAM和DTCM-RAM。

在“6-Quad SPI_rwRAM-DTCM”配置情况下，两种情况显示了性能之间的不同：

- 存储只读数据在Quad-SPI Flash存储器中，即与指令相同的位置（实例1）。
- 存储只读数据在除了Quad-SPI Flash存储器的其它存储器中，将会存储在Flash-ITCM（实例2）。

c. 只有通过ITCM总线的只读数据流。

d. 仅可用于STM32F723E-DISCO板。

对于“6_1-QuadSPI_rwRAM-DTCM”配置，因为实例1中指令和数据存储在Quad-SPI Flash存储器中，只有I-缓存和D-缓存使能。

对于“6_2-QuadSPI_rwRAM-DTCM”配置，因为只读数据存储在Flash-ITCM中，ART和ART预取使能并且D-缓存关闭。对于两种Quad-SPI情况，读/写数据存储在DTCM-RAM中。

对于所有可以访问DTCM-RAM的作为数据存储存储器的配置（*图 3*中的路径5），它们不具有任何特定特征激活，因为内核直接以零等待状态访问与RAM耦合。

对于所有可以访问SRAM1的作为数据存储存储器的配置（*图 3*中的路径7），数据缓存使能。这是下列配置的情况：

- “2-FlashITCM_rwSRAM1”
- “4-FlashAXI_rwSRAM1”
- "8-FlashAXI-ROFTCM_rwSRAM1"[d]

如果应用可以通过FMC访问外部存储器，以用于对数据存储数据存储器和/或代码执行（*图 4*中的路径9），那么数据和/或指令缓存使能。这是下列配置的情况：

- “7-ExtSDRAM-Swapped_rwDTCM”[e]
- "7-ExtPSRAM_rwDTCM"[f]

注： *对于“7-ExtSDRAM-Swapped_rwDTCM”配置，SDRAM发生交换（从0xC000 0000到0x6000 0000重映射以允许缓存使用），因为默认的从0xA000 0000到0xDFFF FFFF的MPU属性区域是一个器件存储类型区域，无法缓存。*

结果

结果由STM32F723E-DISCO、STM32756G-EVAL和STM32769I-EVAL板获得。CPU运行于216 MHz，V_{DD}=3.3 V，有7个到内部Flash存储器的等待状态访问。

e. 仅可用于STM32F7xxx-EVAL板。

f. 仅可用于STM32F723E-DISCO板。

表 7、*表 8*和*表 9*显示了FFT演示所得到的MDK-ARM结果，分别处于STM32F74xxx/STM32F75xxx、STM32F72xxx/STM32F73xxx和STM32F76xxx/STM32F77xxx器件的各自配置下：

表7. STM32F74xxx和STM32F75xxx器件的结果MDK-ARM

特征配置	存储位置配置	CPU周期数(1)
ART + ART-PF开启	1-FlashITCM_rwRAM-DTCM	118577
ART + ART-PF + D-缓存开启	2-FlashITCM_rwSRAM1	135296
I-缓存 + D-缓存开启	3-FlashAXI_rwRAM-DTCM	118653
I-缓存 + D-缓存开启	4-FlashAXI_rwSRAM1	145917
-	5-RAMITCM_rwRAM-DTCM	112428
I-缓存 + D-缓存开启 （常量在Quad-SPI中）	6_1-QuadSPI_rwRAM-DTCM	176441
I-缓存 + ART + ART-PF开启 （在Flash TCM中为常数）	6_2-QuadSPI_rwRAM-DTCM	126900
I-缓存 + D-缓存开启 （在SDRAM中为常数）	7-ExtSDRAM-Swapped_rwDTCM(2)	128398

1. 周期数的值可能会随着工具链的版本变化而变化。
2. HCLK以200 MHz运行。

表8. STM32F72xxx和STM32F73xxx器件的MDK-ARM结果

特征配置	存储位置配置	CPU周期数(1)
ART + ART-PF开启	1-FlashITCM_rwRAM-DTCM	123192
ART + ART-PF + D-缓存开启	2-FlashITCM_rwSRAM1	120615
I-缓存 + D-缓存开启	3-FlashAXI_rwRAM-DTCM	109667
I-缓存 + D-缓存开启	4-FlashAXI_rwSRAM1	122785
-	5-RAMITCM_rwRAM-DTCM	106030
I-缓存 + D-缓存开启 （常量在Quad-SPI中）	6_1-QuadSPI_rwRAM-DTCM	149442
I-缓存 + ART + ART-PF开启 （在Flash TCM中为常数）	6_2-QuadSPI_rwRAM-DTCM	121630
I-缓存 + D-缓存开启 （常量在PSRAM中）	7-ExtPSRAM_rwDTCM	188340
I-缓存 + D-缓存开启 + ART + ART-PF开启 （在Flash TCM中为常数）	8-FlashAXI-ROFTCM_rwSRAM1	115437

1. 周期数的值可能会随着工具链的版本变化而变化。

表9. STM32F76xxx和STM32F77xxx器件的MDK-ARM结果（单存储区/单精度FPU）

特征配置	存储位置配置	CPU周期数(1)
ART + ART-PF开启	1-FlashITCM_rwRAM-DTCM	117059
ART + ART-PF + D-缓存开启	2-FlashITCM_rwSRAM1	115004
I-缓存 + D-缓存开启	3-FlashAXI_rwRAM-DTCM	108491
I-缓存 + D-缓存开启	4-FlashAXI_rwSRAM1	112372
-	5-RAMITCM_rwRAM-DTCM	106275
I-缓存 + D-缓存开启（常量在Quad-SPI中）	6_1-QuadSPI_rwRAM-DTCM	142177
I-缓存 + ART + ART-PF开启（在Flash TCM中为常数）	6_2-QuadSPI_rwRAM-DTCM	120315
I-缓存 + D-缓存开启（在SDRAM中为常数）	7-ExtSDRAM-Swapped_rwDTCM(2)	117744

1. 周期数的值可能会随着工具链的版本变化而变化。
2. HCLK以200 MHz运行。

表10. STM32F76xxx和STM32F77xxx器件的MDK-ARM结果（双存储区/单精度FPU）

特征配置	存储位置配置	CPU周期数(1)
ART + ART-PF开启	1-FlashITCM_rwRAM-DTCM	123237
ART + ART-PF + D-缓存开启	2-FlashITCM_rwSRAM1	119645
I-缓存 + D-缓存开启	3-FlashAXI_rwRAM-DTCM	109209
I-缓存 + D-缓存开启	4-FlashAXI_rwSRAM1	112844

1. 周期数的值可能会随着工具链的版本变化而变化。

*表 10*中，删除了配置5、6_1、6_2和7，因为它们不依赖于内部闪存。它们具有与单存储区配置相同的性能。

图 14、*图 15*和*图 16*中表格分别显示了STM32F74xxx/STM32F75xxx、STM32F72xxx/STM32F73xxx和STM32F76xxx/STM32F77xxx器件的各自配置相对于配置5的相对比例，配置5具有最佳结果。

相对比例 = cycles_number_config_X / cycles_number_config_5

相对比例计算可以允许各个性能与最好性能做比较（5-RamITCM_rwRAM-DTCM），还可以在配置之间相互比较。

图14. STM32F74xxx和STM32F75xxx FFT相对比例周期数，采用MDK-ARM

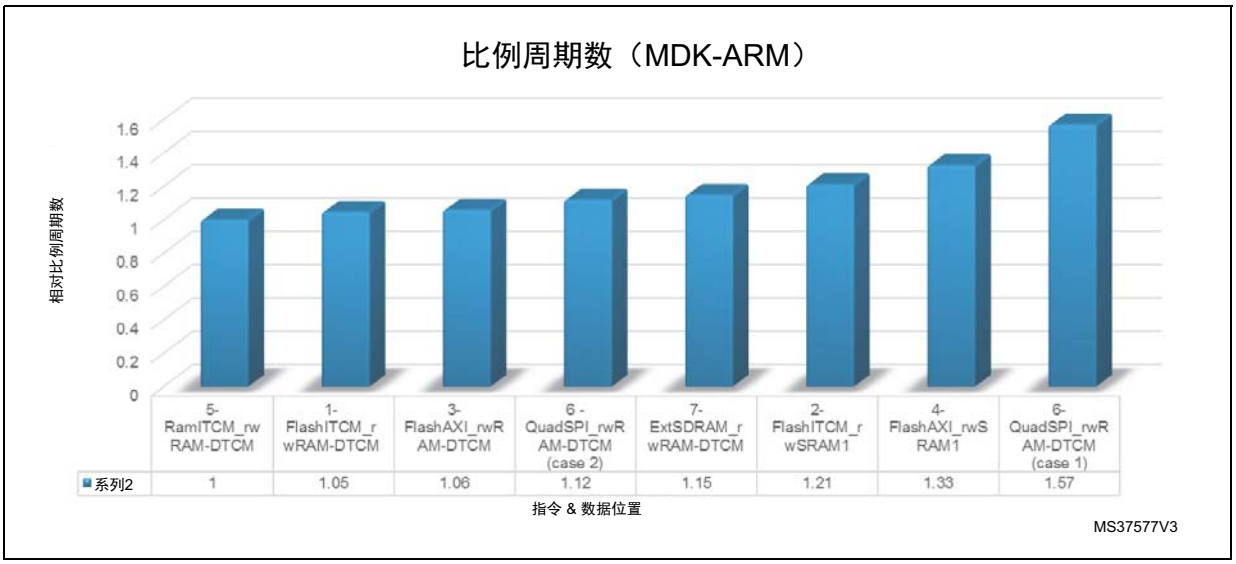


图15. STM32F72xxx和STM32F73xxx FFT相对比例周期数，采用MDK-ARM

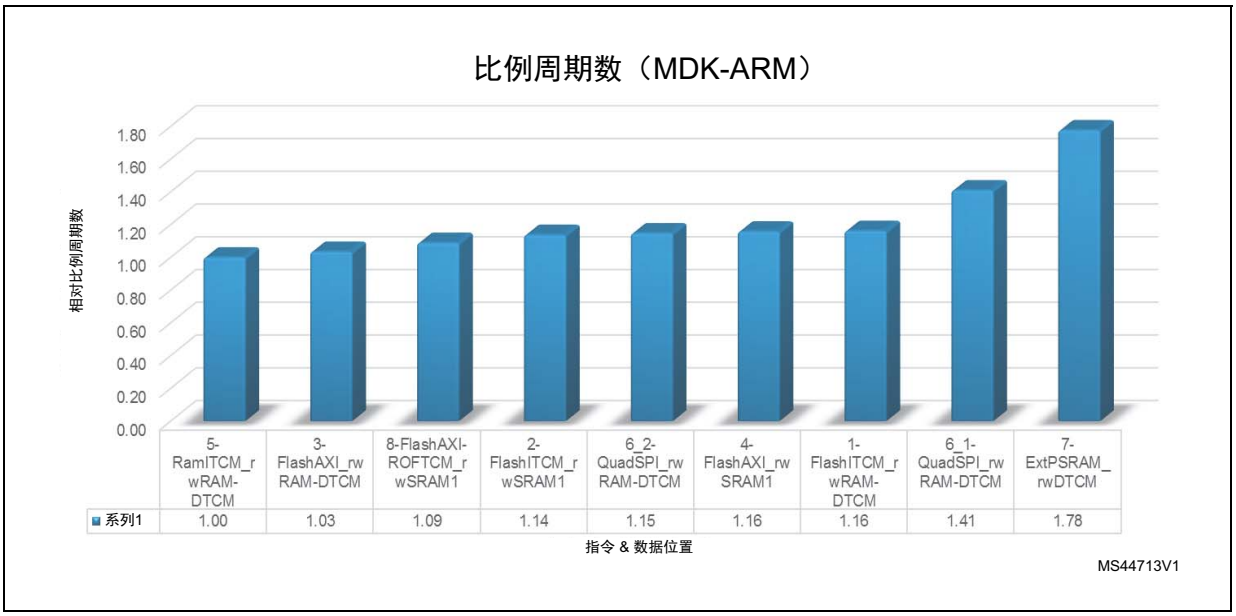
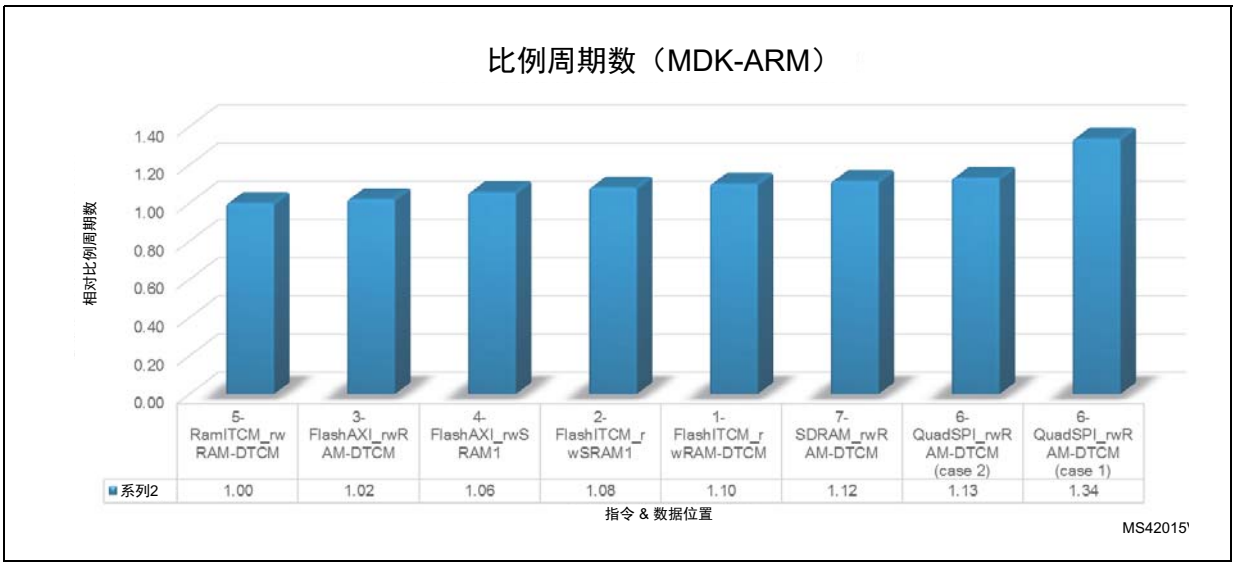


图16. STM32F76xxx和STM32F77xxx FFT相对比例周期数，采用MDK-ARM



分析

STM32F74xxx和STM32F75xxx器件：

如果用户将数据存储位置固定在DTCM-RAM中，而代码执行存储位置是变化的，则相对比例表明代码是否从Flash-ITCM或Flash-AXI执行，其性能与ART或指令缓存分别使能时几乎相同。

这是配置的情况：

- “1-FlashITCM_rwRAM-DTCM”
- “3-FlashAXI_rwRAM-DTCM”

如果用户将执行存储位置固定在Flash-ITCM或Flash-AXI中，而R/W数据存储位置是变化的，则相对比例表明，无论使用了哪个内部RAM进行数据存储，结果都几乎相同，因为对连接到AXI/AHB的RAM使能了R/W数据缓存。

这是几种配置的情况：

- “1-FlashITCM_rwRAM-DTCM”和“2-FlashITCM_rwSRAM1”
- “3-FlashAXI_rwRAM-DTCM”和“4-FlashAXI_rwSRAM1”

如果“7-ExtSDRAM_Swapped_rwDTCM”配置相对比例与“1-FlashITCM_rwRAM-DTCM”或者“3-FlashAXI_rwRAM-DTCM”配置的相比较，结果是接近的，因为在这些配置中都使能了缓存。因此代码执行或者来自内部存储的数据保存不影响性能。

注意，在“5-RamITCM_rwRAM-DTCM”配置情况下，它具有最佳的性能（最低的周期数）。这是因为ITCM和DTCM总线上没有竞争，并且没有执行缓存维护（如果与缓存使用相比，没有使用缓存）。同样在给出的FFT示例中，没有CPU和其它主设备到DTCM-RAM的并发访问。

如果将“6-Quad-SPI_rwRAM-DTCM”配置（实例X）的实例1和实例2的两个比例进行比较，注意在性能上存在明显的不同，这是因为演示使用了大量常量数据。

对于实例1（6_1-Quad-SPI_rwRAM-DTCM）来说，只读数据和指令存储在Quad-SPI Flash存储器，因为指令获取和读取数据在总线矩阵上产生了并发访问，由此产生了一个延迟。

对于实例２（6_2-Quad SPI_rwRAM-DTCM）来说，只读数据和代码是分开的。只读数据存储在Flash-TCM，因此避免了只读数据和指令获取产生的并发访问，CPU从AXI获取指令和数据从TCM载入可以同时发生。这就是为什么第二例的性能明显比第一例好的原因。

STM32F76xxx和STM32F77xxx器件：

对于这些器件，除了使用高速缓存的情形，这些分析与对STM32F74xxx和STM32F75xxx器件的分析几乎相同。

如果比较*表 7*的配置5和*表 9*的配置5，可以注意到，对于STM32F76xxx和STM32F77xxx器件，其性能比STM32F74xxx/ STM32F75xxx器件提高了约5%。这可能是由于Cortex-M7版本的FPU中实现了一些增强功能，它们嵌入到了STM32F76xxx和STM32F77xxx器件中。

得益于STM32F76xxx和STM32F77xxx器件的高速缓存大小，在所有使用高速缓存、SRAMx和AXI路径的情形中，性能都得到了提高。

例如，由于高速缓存大小从4 KB增加到了16 KB，相比于STM32F74xxx和STM32F75xxx器件，配置3和4排名首位。

在Quad-SPI的情况下，并且采用更精确的配置6_2时，相对于STM32F74xxx和STM32F75xxx器件，性能提高了约19.5%，包括由于FPU增强功能所带来的性能提升。

在使用外部存储器如SDRAM情况下，相对于STM32F74xxx和STM32F75xxx器件，由于高速缓存大小增加，性能也提高了约10%，包括由于FPU增强功能所带来的性能提升。

使用双存储区的情况下，尽管闪存接口宽度从256位降至128位，但是其性能在某些情况下与STM32F74xxx和STM32F75xxx器件的性能相似或更好，比如当通过AXI/AHB访问闪存，且使能了高速缓存时。

表11. 单存储区与双存储区的性能比较

项目配置	单存储区模式（CPU周期数）	双存储区模式（CPU周期数）	双存储区 vs 单存储区降低量（%）
1-FlashITCM_rwRAM-DTCM	117059	123237	5.28
2-FlashITCM_rwSRAM1	115004	119645	4.04
3-FlashAXI_rwRAM-DTCM	108491	109209	0.66
4-FlashAXI_rwSRAM1	112372	112844	0.42

*表 11*表明，通过AXI/AHB进行闪存访问的情况下，单或双存储区模式中性能是相同的。如果通过TCM进行闪存访问，采用当前算法，性能会降低约5%。

STM32F72xxx和STM32F73xxx器件：

基于*表 8*，除了Flash-TCM访问，STM32F72xxx和STM32F73xxx的结果与STM32F7系列的其他器件几乎是相同的。STM32F72xxx和STM32F73xxx器件的闪存宽度为128位，比STM32F7系列其他器件的闪存宽度要小，但是具有相同的64行ART加速器。

相比于其他STM32F7系列器件，当闪存通过TCM接口进行读取访问时，STM32F72xxx和STM32F73xxx的性能会降低。双存储区模式下，它们与STM32F76xxx/STM32F77xxx器件具有相同的性能。

通过AXI接口访问闪存时，相比于Flash-ITCM访问，性能会提高。这种情况下，缓存的效果补偿了闪存宽度的减小。请参考*表 8*中的“1-FlashITCM_rwRAM-DTCM”和“3-FlashAXI_rwRAM-DTCM”配置结果。

对TCM-RAM与对其他器件的访问性能是相同的，因为访问这些RAM时架构方面是没有差别的。

对于PSRAM结果，相比于利用其他STM32F7系列器件所得到的SDRAM结果，其性能是下降的，因为STM32F723-DISCO板上可用的PSRAM时序较慢，且为16位数据访问。

通过分离只读数据和指令，“8-FlashAXI-ROFTCM_rwSRAM1”配置表现出比“4-FlashAXI_rwSRAM1”配置更高的性能。配置8下，通过Flash-ITCM访问只读数据，通过Flash-AXI取指令，而配置4下，只读数据和指令在同一个总线接口（AXI）上取出。

3.2 使用DMA的CPU存储器范围性能

本章介绍一个或多个主设备被激活时的性能。

对于STM32F74xxx和STM32F75xxx器件，结果由KEIL MDK-ARM v5.16.0工具链、STM32F7xx套件（版本2.2.0）和Cube固件（版本1.0.0）获得。

对于STM32F76xxx和STM32F77xxx器件，结果由KEIL MDK-ARM v5.17.0工具链、STM32F7xx套件（版本2.6.0）和Cube固件（版本1.4.0）获得。

对于STM32F72xxx和STM32F73xxx器件，结果由KEIL MDK-ARM v5.21.1工具链、STM32F7xx套件（版本2.9.0）和Cube固件（版本1.6.0）获得。

MDK-ARM代码优化配置是3级：为时间优化。

结果

（*表 12*，*表 13*，*表 14*），（*表 15*，*表 16*，*表 17*）和（*表 18*，*表 19*，*表 20*）总结了不同场景下，STM32F74xxx/STM32F75xxx、STM32F72xxx/STM32F73xxx和STM32F76xxx/STM32F77xxx器件进行FFT过程分别所花费的周期数，以及在一个或多个主设备与CPU访问同一个存储器的情况下，性能降低的百分比。

结果通过STM32F723E-DISCO、STM32756G-EVAL和STM32769I-EVAL板得到，CPU运行于200 MHz（可实现@100 MHz的SDRAM访问），VDD=3.3 V，6等待状态访问内部闪存。由于闪存和高速缓存中数据和指令的某些对齐，估计的测量误差为0.4%。

此结果的条件为：

– DMA1和DMA2访问优先级已配置为高优先级，数据大小为1个字节。
– SPI3波特率为25 Mbit/s。
– CM7_AHBSCR寄存器保持其复位值。

STM32F74xxx和STM32F75xxx器件：

表12. 配置1：从Flash-ITCM执行 / FFT CPU数据存储于DTCM-RAM

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	69247	-
	DMA1_SPI3	-	-		
2	DMA2	SDRAM	SRAM1	69247	0
	DMA1_SPI3	-	-		
3	DMA2	SRAM1	**DTCM-RAM(1)**	70929	2.43
	DMA1_SPI3	-	-		
4	DMA2	SRAM1	**DTCM-RAM(1)**	71275	2.93
	DMA1_SPI3	SRAM2			
5	DMA2	SDRAM	**DTCM-RAM(1)**	70830	2.29
	DMA1_SPI3	SRAM2			
6	DMA2	**DTCM-RAM(2)**	SRAM1	70140	1.29
	DMA1_SPI3	-	-		
7	DMA2	**Flash-AXI(2)**	SRAM1	69596	0.50
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发写访问的存储器。
2. CPU和DMA之间并发读访问的存储器。

表13. 配置2：从Flash-ITCM执行 / FFT CPU数据存储于SRAM1

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	79925	-
	DMA1_SPI3	-	-		
2	DMA2	SDRAM	SRAM2	79925	0
	DMA1_SPI3	-	-		
3	DMA2	DTCM-RAM	**SRAM1**(1)	81720	2.25
	DMA1_SPI3	-	-		
4	DMA2	DTCM-RAM	**SRAM1**(1)	81828	2.38
	DMA1_SPI3	SRAM2			
5	DMA2	SDRAM	**SRAM1**(1)	81674	2.19
	DMA1_SPI3	SRAM2			
6	DMA2	**SRAM1**(2)	DTCM-RAM	81410	1.86
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发写访问的存储器。
2. CPU和DMA之间并发读访问的存储器。

表14. 配置3：从Flash-AXI执行 / FFT CPU数据存储于DTCM-RAM

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	68206	-
	DMA1_SPI3	-	-		
2	DMA2	**Flash-AXI**(1)	SRAM1	68865	0.97
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发读访问的存储器。

STM32F72xxx和STM32F73xxx器件

表15. 配置1：从Flash-ITCM执行 / FFT CPU数据存储于DTCM-RAM

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	73406	-
	DMA1_SPI3	-	-		
2	DMA2	SRAM1	**DTCM-RAM**(1)	74568	1.58
	DMA1_SPI3	-	-		
3	DMA2	SRAM1	**DTCM-RAM**(1)	74788	1.88
	DMA1_SPI3	SRAM2			

表15. 配置1：从Flash-ITCM执行 / FFT CPU数据存储于DTCM-RAM（续）

场景	主	源	目标	FFT周期数	降低（%）
4	DMA2	**DTCM-RAM(2)**	SRAM1	74321	1.25
	DMA1_SPI3	-	-		
5	DMA2	**Flash-AXI(2)**	SRAM1	74214	1.10
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发写访问的存储器。
2. CPU和DMA之间并发读访问的存储器。

表16. 配置2：从Flash-ITCM执行 / FFT CPU数据存储于SRAM1

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	78416	-
	DMA1_SPI3	-	-		
2	DMA2	DTCM-RAM	**SRAM1(1)**	79129	0.91
	DMA1_SPI3	-	-		
3	DMA2	DTCM-RAM	**SRAM1(1)**	79145	0.93
	DMA1_SPI3	SRAM2			
4	DMA2	**SRAM1(2)**	DTCM-RAM	79115	0.89
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发写访问的存储器。
2. CPU和DMA之间并发读访问的存储器。

表17. 配置3：从Flash-AXI执行 / FFT CPU数据存储于DTCM-RAM

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	66374	-
	DMA1_SPI3	-	-		
2	DMA2	**Flash-AXI(1)**	SRAM1	66982	0.92
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发读访问的存储器。

STM32F76xxx和STM32F77xxx器件：

表18. 配置1：从Flash-ITCM执行 / FFT CPU数据存储于DTCM-RAM（单存储区/单精度FPU）

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	69250	-
	DMA1_SPI3	-	-		
2	DMA2	SDRAM	SRAM1	69250	0
	DMA1_SPI3	-	-		
3	DMA2	SRAM1	**DTCM-RAM(1)**	70873	2.34
	DMA1_SPI3	-	-		
4	DMA2	SRAM1	**DTCM-RAM(1)**	71191	2.80
	DMA1_SPI3	SRAM2			
5	DMA2	SDRAM	**DTCM-RAM(1)**	70825	2.27
	DMA1_SPI3	SRAM2			
6	DMA2	**DTCM-RAM(2)**	SRAM1	70150	1.30
	DMA1_SPI3	-	-		
7	DMA2	**Flash-AXI(2)**	SRAM1	69587	0.49
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发写访问的存储器。
2. CPU和DMA之间并发读访问的存储器。

表19. 配置2：从Flash-ITCM执行 / FFT CPU数据存储于SRAM1（单存储区/单精度FPU）

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	72648	-
	DMA1_SPI3	-	-		
2	DMA2	SDRAM	SRAM2	72648	0
	DMA1_SPI3	-	-		
3	DMA2	DTCM-RAM	**SRAM1(1)**	73071	0.58
	DMA1_SPI3	-	-		
4	DMA2	DTCM-RAM	**SRAM1(1)**	73086	0.60
	DMA1_SPI3	SRAM2			
5	DMA2	SDRAM	**SRAM1(1)**	72964	0.43
	DMA1_SPI3	SRAM2			
6	DMA2	**SRAM1(2)**	DTCM-RAM	73082	0.60
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发写访问的存储器。
2. CPU和DMA之间并发读访问的存储器。

表20. 配置3：从Flash-AXI执行 / FFT CPU数据存储于DTCM-RAM（单存储区/单精度FPU）

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	65897	-
	DMA1_SPI3	-	-		
2	DMA2	**Flash-AXI**(1)	SRAM1	66155	0.39
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发读访问的存储器。

表21. 配置3：从Flash-AXI执行 / FFT CPU数据存储于DTCM-RAM（双存储区/单精度FPU）

场景	主	源	目标	FFT周期数	降低（%）
1	DMA2	-	-	66370	-
	DMA1_SPI3	-	-		
2	DMA2	**Flash-AXI**(1)	SRAM1	66960	0.89
	DMA1_SPI3	-	-		

1. CPU和DMA之间并发读访问的存储器。

分析

STM32F74xxx和STM32F75xxx器件：

1. 如果CPU和一个或多个主设备不访问同一个存储器，则不存在争用。对于配置1和2，从场景1和2得到的结果相同。请参考以下结果：
   - 配置1/场景2 vs 配置1/场景1。
   - 配置2/场景2 vs 配置2/场景1。
2. 当CPU和DMA访问同一个存储器时，会发生争用，这是因为对于连接到总线矩阵的存储器包括外部存储器（SRAM1，SRAM2，SDRAM，等等）或连接到CPU侧（紧耦合单元接口：TCU单元）的存储器（这里并发发生在DTCM-RAM级），在总线矩阵上会存在仲裁。请参考以下结果：
   - 配置1/场景1 vs 配置1/场景3。
   - 配置2/场景1 vs 配置2/场景3。

如结果所示，CPU和DMA2在DTCM-RAM上的并发写访问导致性能降低了约2.5%。

SRAM1/SRAM2的情况下，CPU和DMA2并发写访问导致性能降低约2.5%。

3. 当有另外的主设备与CPU对同一存储器进行写访问时，延迟会增加约0.5 %。请参考以下结果：
   - 配置1/场景3 vs 配置1/场景4。
   - 配置2/场景3 vs 配置1/场景4。
4. CPU和DMA之间对于给定存储器的读访问争用影响小于写访问争用影响。请参考以下结果：
   - 配置1/场景3 vs 配置1/场景6。
   - 配置2/场景3 vs 配置2/场景6。

如结果所示，CPU和DMA2在DTCM-RAM级上的并发读访问可导致性能降低1%至2%。

SRAM1/SRAM2的情况下，CPU和DMA2并发读访问导致性能降低1.5%至2.5%。

5. 如果存储器源相对较慢，如SDRAM，那么并发的影响会降低，因为在此存储器上存在一些延迟，这为CPU在总线矩阵上访问此存储器提供了一定带宽。请参考以下结果：
   - 配置1/场景4 vs 配置1/场景5。
   - 配置2/场景4 vs 配置2/场景5。

配置1中，场景4的DTCM-RAM被替代为场景5的SDRAM。此情形同样适用于配置2。

6. 从Flash-AXI或Flash-ITCM执行数据，和同时从闪存到另一存储器进行传输的时候，没有明显的争用。这是因为相比于使用高速缓存进行CPU访问，DMA对闪存的访问比较慢。DMA对闪存的访问由等待状态数（在闪存接口中进行配置）来决定速度（我们示例中为6个等待状态），而在CPU侧或ART中（Flash-ITCM访问的情况下）取指令/数据到高速缓存。此配置可消除延迟影响。请参考以下结果：
   - 配置1/场景7 vs 配置3/场景2。

由于DMA1可执行存储器与外设之间的传输，因此并发性对CPU和DMA1之间存储器的影响取决于外设速度。外设越快，并发影响越明显。例如，如果最大速度为25 MHz的SPI3被最大速度为50 MHz的SPI4所替代，则由于数据传输带宽较高，延迟会更加重要。不过，延迟仍在合理范围内。

总之，所有场景下，由于CPU和DMA之间对于不同存储器的读或写访问（DTCM-RAM，SRAMx，外部存储器）并发性引起的性能降低很低，约为3%。由于智能系统架构和总线矩阵，整体性能不受多个主设备并发的影响。

STM32F76xxx和STM32F77xxx器件：

对于STM32F76xxx和STM32F77xxx器件，除使用AXI/AHB进行CPU访问之外，其分析与对STM32F74xxx和STM32F75xxx的分析几乎相同。

*表 18*中，由于这些器件都处于同样的架构配置，因此其性能与STM32F74xxx和STM32F75xxx器件相同（见*表 12*）。

但是在*表 19*中，由于CPU和一个或多个主设备对存储器的访问并发性不影响性能，因此性能降低不超过0.6%。由于STM32F76xxx和STM32F77xxx器件的高速缓存大小增加的影响，这是可能实现的。

双存储区Flash配置的情况下（见*表 21*），如果对闪存进行并发访问，则性能下降与STM32F74xxx和STM32F75xxx器件相同。但是，单存储区Flash配置的情况下（见*表 20*），性能降低并不重要，因为不存在并发访问。

STM32F72xxx和STM32F73xxx器件：

对于STM32F72xxx和STM32F73xxx器件，整体来说，其分析仍然与其他器件相同。当从/对SRAM1或SRAM2进行读或写并发访问时，缓存大小对性能有直接影响。另外，如果在Flash-AXI上存在并发读访问（见*表 17*），那么即便Flash宽度小于其他STM32F7系列器件的宽度，由于其缓存大小相比STM32F74xxx和STM32F75xxx器件有所增加，因此并发影响仍然很低。

因此，缓存空间越大，对内存的争用越少。

4 STM32F7系列器件的性能排名和比较

本章提供了STM32F7系列器件中STM32F7xxxx器件的总体定位概述，以及如何选择STM32F7xxxx器件以满足性能和成本方面的应用需求。

4.1 STM32F7系列的性能排名

高速缓存的大小非常重要，对性能有直接影响：高速缓存空间越大，应用执行速度越快。

当代码从Flash-AXI执行且数据位于SRAM1中时，不同器件之间的性能取决于其高速缓存大小。例如，使用“3-FlashAXI_rwRAM-DTCM”和“4-FlashAXI_rwSRAM1”配置的情况下：

- STM32F76xxx和STM32F77xxx器件性能最好，因为它们嵌入了最大的缓存（16 KB），
- STM32F74xxx和STM32F75xxx器件性能最差，因为它们嵌入了最小的缓存（4 KB）。
- STM32F72xxx和STM32F73xxx器件性能排名中等，因为其缓存大小为8 KB。使用这些器件通过AXI总线访问Flash，高速缓存大小可以补偿其Flash宽度（128位）相比于其他器件（对Flash进行256位访问）的降低。

对于“6_1-QuadSPI_rwRAM-DTCM”配置（其中代码从Quad-SPI Flash执行），可以得到相同的排名。

闪存宽度对性能也有影响，特别是从TCM总线（Flash-ITCM）取数据/指令时。在“1-FlashITCM_rwRAM-DTCM”配置的情况下，相比于采用256位访问的其他STM32F7系列器件，STM32F72xxx和STM32F73xxx器件性能最低，因为它们采用128位访问闪存。

*表 22*总结了所有STM32F7系列器件的结果。

表22. STM32F7系列器件之间的性能比较

存储位置配置	STM32F74xxx/ STM32F75xxx (1)	STM32F72xxx/ STM32F73xxx (1)	STM32F76xxx/ STM32F77xxx 单存储区 (1)	STM32F76xxx/ STM32F77xxx 双存储区 (1)
1-FlashITCM_rwRAM-DTCM	118577	123192	117059	123237
2-FlashITCM_rwSRAM1	135296	120615	115004	119645
3-FlashAXI_rwRAM-DTCM	118653	109667	108491	109209
4-FlashAXI_rwSRAM1	145917	122785	112372	112844
5-RAMITCM_rwRAM-DTCM	112428	106030	106275	-
6_1-QuadSPI_rwRAM-DTCM	176441	149442	142177	-

1. 以周期数表示。

使用同样的算法，根据STM32F7器件和存储器访问，对性能的排名

排名从高到低给出：

Flash-ITCM访问（读） + ART 使能：

1. 单存储区模式下的STM32F76xxx和STM32F77xxx器件。
2. STM32F74xxx和STM32F75xxx器件。
3. 双存储区模式下，STM32F72xxx和STM32F73xxx器件/ STM32F76xxx和STM32F77xxx器件。

Flash-AXI访问（读） + 高速缓存使能：

1. 单存储区模式下的STM32F76xxx和STM32F77xxx器件。
2. 双存储区模式下的STM32F76xxx和STM32F77xxx器件。
3. STM32F72xxx和STM32F73xxx器件 / STM32F74xxx和STM32F75xxx器件(g).
4. STM32F74xxx和STM32F75xxx器件(g)。

SRAM1和SRAM2访问 + 高速缓存使能

1. STM32F76xxx和STM32F77xxx器件。
2. STM32F72xxx和STM32F73xxx器件。
3. STM32F74xxx和STM32F75xxx器件。

TCM-RAM访问：

所有器件具有相同的性能。

如前所述，相比于STM32F74xxx/STM32F75xxx器件，STM32F76xxx/STM32F77xxx器件和STM32F72xxx/STM32F73xxx器件中嵌入的cortex-M7的FPU经过了改进增强，这解释了在“5-RamITCM_rwRAM-DTCM”配置（见*表 22*）下与FFT演示相比的周期数差异。

外部存储器访问 + 高速缓存使能：

1. STM32F76xxx和STM32F77xxx器件。
2. STM32F72xxx和STM32F73xxx器件。
3. STM32F74xxx和STM32F75xxx器件。

g. 根据所用算法，STM32F74xxx/STM32F75xxx器件和STM32F72xxx/STM32F73xxx器件可能有相同的性能。

4.2 STM32F7器件选择指南

如果用户寻求最佳性能以及最大内部存储器空间，那么STM32F76xxx和STM32F77xxx器件将非常适合您的应用。

如果用户寻求最低成本，而性能可以稍低、内部存储器空间可以稍小，那么可以选择STM32F72xxx、STM32F73xxx、STM32F74xxx或STM32F75xxx器件。

但是，如果用户目标是图形应用，并且性能可以低于STM32F76xxx和STM32F77xxx器件，那么STM32F74xxx和STM32F75xxx器件比不具备DMA2D和LCD-TFT的 DMASTM32F72xxx和STM32F73xxx器件更适合。

对于高端图形应用，由于STM32F76xxx和STM32F77xxx器件具有DSI接口，可实现高速图形数据吞吐， 因而比STM32F74xxx和STM32F75xxx器件更适合。

如果应用需要连接以太网，那么STM32F74xxx、STM32F75xxx、STM32F76xxx和STM32F77xxx器件适合于用户应用，因为STM32F72xxx和STM32F73xxx器件不具备以太网外设。

STM32F7x3xx器件嵌入了一个高速USB OTG PHY，而所有其他STM32F7系列器件都不支持。

5 软件存储分区和建议

本节给出了如何在STM32F7存储器中分割代码和数据的建议，以获得取决于代码和数据大小的最佳折衷性能。

5.1 软件存储分区

由于CPU可以64位宽和零等待状态的方式直接访问TCM存储器，DTCM-RAM和ITCM-RAM的位置对于读/数据和指令获取分别是最佳位置。

因此ITCM-RAM（16KB）保留用于具有确定执行性的关键代码，比如不能等待缓存不命中的中断处理以及一些针对电机控制应用的关键控制环。

DTCM-RAM保留用于数据存储操作、常规访问，以及堆栈等关键实时数据。在使用RTOS的实时应用中，一般来说，堆会被大量使用。例如，如果DTCM-RAM为64 KB且对应用而言已足够，则DTCM-RAM分散方式（对于堆是32KB、对于栈为8KB，以及对于全局变量为24KB）相对合理。在前后台应用中，几乎不使用堆，DTCM-RAM将分给栈和全局变量。

当用户应用的代码长度不超过内部Flash存储器时，内部Flash将是最佳执行区域，可以采用下面这两种访问方式中的任何一种：

- 通过TCM接口访问内部Flash，并且使能ART加速器，或者
- 通过AXI接口，并使能缓存，以在216MHz时实现零等待周期

注意从DTCM-RAM中的Flash-ITCM/数据开始执行和从DTCM-RAM中的Flash-AXI/数据开始执行具有相同的CoreMark值，即5 CoreMark/MHz。

SRAM1可以在图像应用中保留作为一个图像帧缓冲器，这种应用使用16位模式的QVGA TFT，这需要相当大的图像数据，显示性能由LCD-TFT和DMA2D DMA实现。当DTCM-RAM没有更多空间可用时，这个存储器也可以用于数据存储操作，所以在那样的情况下，一个来自SRAM1的区域（使能数据缓存）可以保留用于全局变量，从而为关键数据留有更多的空间。

SRAM2可以保留用于诸如以太网和USB等外设，以存储缓冲区和描述符等数据。该存储器也可用于全局变量。

当应用需要更多的存储空间，它的代码和/或数据超过内部存储空间时，可以使用外部存储器扩展存储容量，并且不会造成性能损失。

例如一个通过FMC连接的高达250 MB大小的外部NOR Flash存储器可能包含应用指令，同时指令缓存使能，而常量数据存储在内部Flash存储器（通过使能ART或者缓存）。这就是应用中使用大量常量时，为了避免指令和数据的在总线矩阵上同一路径（*图 4*中的路径9）发生并发访问的情况。

在缺少内部RAM的情况中，数据存储可以通过使能数据缓存，经由FMC接口在外部SRAM或SDRAM上实现。这些存储器可能包含用于图形应用的帧缓冲器或用于非关键数据的帧缓冲器。同时，优先考虑的是在DTCM-RAM中存储非常重要的数据。

Quad-SPI Flash存储器可以用来存储只读数据，也就是相对大的图像文件或波，或者通过使能数据缓存，同时保持与内部Flash存储器访问相同水平的性能。

Quad-SPI Flash存储器也可以用于包含高达256 MB的存储映射模式中的应用，而且相比于必须连接到FMC接口的并行Flash存储器，以更小的STM32F7封装节省了若干GPIO。在CPU需要定期访问QuadSPI中的只读数据的情况下，应该将其映射到内部Flash的地址范围内。如果应用需要更大的空间和更好的执行性能，用户可以在Quad-SPI（载入区域）中载入应用，同时使用一个外部的SDRAM用于应用复制和执行（执行区域）。

5.2 建议

在DMA和CPU同时访问DTCM-RAM情况下，如果CPU不具有最高的访问优先级，那么应用速度可能会减慢。优先级可以由软件管理，使用关键代码部分中的CM7_AHBSCR寄存器（载入/存储数据在DTCM-RAM）。

要使用STM32F76xxx和STM32F77xxx器件获得最佳性能，在使用双存储区的情况下，建议通过AXI/AHB接口而不是TCM来访问闪存，以充分利用高速缓存空间所带来的性能提升。

要使用STM32F72xxx和STM32F73xxx器件获得最佳性能，建议通过AXI/AHB接口而不是TCM来访问闪存，以充分利用高速缓存空间所带来的性能提升。

当一个外部存储器用于执行区域时，如果存储器映射的区域具有默认属性Execute-Never（XN），则需要注意。在这种情况下，用户必须使用MPU将存储器修改成为一个可执行区域，不然会出现硬性故障。这就是复位后SDRAM存储区域就属于不可执行区域。参考*表 2*查看关于ARM的默认可执行区域。

当一个外部存储器用于数据存储，且它没有映射到一个可缓存区域时，如果重新分配一个可缓存区域则可以使用重映射，这是SDRAM存储区（SYSCFG_MEMRMP寄存器中，SWP_FMC [1:0] = 01）的情况，或者使用MPU将存储类型修改为正常类型存储。

访问外部存储器的性能取决于存储器的时序及其数据总线宽度。访问外部存储器的时间越短，性能越好。数据总线宽度越大，性能越好。

当大量的常量在应用中通过CPU载入时（这就是FFT应用的情况），用户应该验证它们是否存储在正确的存储位置，它们预期存储在非内部Flash存储器的其它位置。在某些情况下，如果变量意外存储在内部Flash存储器（TCM或AXI）中，且ART加速器或/和D-缓存没有使能，则应用会冲击所有的Flash等待状态，这时应用就会明显变慢。

如果可能，可尝试分离开数据和代码位置，特别是当其存储器连接到AHB/AXI时，要避免对总线矩阵进行并发访问，即通过AHB/AXI取数据，而通过TCM取指令。

对于Flash访问，要提高性能，可尝试将指令和只读数据访问分开。例如，只读数据通过Flash-TCM进行访问，而指令通过Flash-AXI进行访问，反之亦然。

如果DTCM-RAM用作数据位置，且所用变量为字节或/和半字类型， 由于STM32F7系列的该RAM中没有ECC管理，建议在DTCM接口（在DTCMCR寄存器中）中禁用DTCM-RAM的读-修改-写，以提高性能。

为此，可在system_stm32f7xx.c文件的SystemInit()中或主函数开始处添加以下C代码：

```
__IO uint32_t * CM7_DTCMCR = (uint32_t*)(0xE000EF94);
* CM7_DTCMCR &= 0xFFFFFFFD; /* 禁用读-修改-写 */
```

或者可以在启动文件的汇编代码中完成：

```
CM7_DTCMCR EQU 0xE000EF94
    LDR    R2, =CM7_DTCMCR
    LDR    R0, [R2]
    BFC    R0, #1, #1 ;Reset CM7_DTCMCR.RMW field
    STR    R0, [R2]
    DSB
    ISB
```

不建议在调用main函数之前使能缓存，即在分散载入阶段之前，因为可能会出现硬性故障。

6 结论

如今应用变得更加复杂，需要更多的微控制器提高效率和性能。得益于STM32F7智能架构，STM32F7系列器件是适合物联网（Internet of Things，IoT）应用的平台。它使需要更快速微控制器的开发人员能够轻松实现代码优化。

由于STM32F7具有两个独立的实现零等待执行性能的机制：用于内部Flash存储器的ST ART加速器™和用于内部Flash存储器和其它存储器（内部和外部）的L1-缓存（指令和数据缓存），可以得到更好的应用响应。用户可以借助缓存花更少的时间优化代码和数据大小，只需要添加外部存储资源即可，而且不会造成性能损失。即便DMA与CPU同时完成传输，得益于STM32F7系统架构，其性能也不会受影响。

要实现更高性能，用户可以选择STM32F76xxx和STM32F77xxx器件，充分利用高速缓存空间所带来的性能提升。

要得到低成本器件并且性能可接受，根据是否面向图形应用，用户可以选择STM32F72xxx/STM32F73xxx或STM32F74xxx/STM32F75xxx器件。

7 版本历史

表23. 文档版本历史

日期	版本	变更
2015年6月19日	1	初始版本。
2015 年 11 月 18 日	2	更新了封面。 更新了*图 2：Flash存储器接口（路径：1、2、3、4）*和*图 4：外部存储器接口（路径：9、10）*。 更新了*第 1.6 节：DMA*说明，增加了*图 6：DMA与CPU之间无存储器并发访问*和*图 8：FFT示例框图*。 更新了*第 2 节：典型应用*，增加了*第 2.3 节：DMA激活的CPU存储器访问演示的项目配置*。 更新了*第 3 节：结果和分析*，增加了*第 3.2 节：使用DMA的CPU存储器范围性能*。
2016年5月27日	3	更新了*第 1.2 节：Cortex®-M7系统缓存*，增加了STM32F76xxx和STM32F77xxx缓存大小。 更新了*第 1.4 节：STM32F7总线矩阵*说明。 更新了*图 1：STM32F7系列系统架构* L1-cache、存储空间，增加了注释1。 更新了*第 1.5 节：STM32F7存储器*： – 更新了*图 1.5.1：嵌入式Flash*，增加了STM32F76xxx和STM32F77xxx Flash存储器说明。 – 更新了*图 2：Flash存储器接口（路径：1、2、3、4）* L1-cache空间，增加了注释1。 更新了*第 1.5.2 节：嵌入式SRAM*： – 增加了*表 5：STM32F76xxx/STM32F77xxx器件的内部存储器总结*。 – 更新了*图 3：DMA对一些内部存储器的不同访问（路径：5，6，7，8）*和*图 4：外部存储器接口（路径：9、10）*，删去了L1-Cache空间。 增加了*第 1.7 节：STM32F7系列器件之间的主要差异 （从架构角度看）*。 更新了*第 2.2 节：CPU存储器访问演示的项目配置*： – 增加了*图 11：针对双存储区配置的ST-Link实用程序设置*。 – 增加了*图 12：双存储区模式下Flash加载器配置*。

表23. 文档版本历史（续）

日期	版本	变更
2016年5月27日	3（续）	更新了*第 3.1节：CPU存储器访问性能*。 – 增加了*表 9：STM32F76xxx和STM32F77xxx器件的MDK-ARM结果（单 存储区/单精度FPU）*。 – 增加了*表 10：STM32F76xxx和STM32F77xxx器件的MDK-ARM结果（双 存储区/单精度FPU）*。 – 更新了*图 14：STM32F74xxx和STM32F75xxx/xxT相对比例周期数，采用 MDK-ARM*。 – 增加了*图 16：STM32F76xxx和STM32F77xxx/xxT相对比例周期数，采用 MDK-ARM*。 – 增加了STM32F76xxx和STM32F77xxx分析。 – 增加了*表 11：单存储区与双存储区的性能比较*。 更新了*第 3.2节：使用DMA的CPU存储器范围性能*。 – 增加了*表 18：配置1：从Flash-ITCM执行/FFTCPU数据存储于DTCM-RAM（单存储区/单精度FPU）*。 – 增加了*表 19：配置2：从Flash-ITCM执行/FFTCPU数据存储于SRAM1（单存储区/单精度FPU）*。 – 增加了*表 20：配置3：从Flash-AXI执行/ FFT CPU数据存储于DTCM-RAM（单存储区/单精度FPU）*。 – 增加了*表 21：配置3：从Flash-AXI执行/ FFT CPU数据存储于DTCM-RAM（双存储区/单精度FPU）*。 – 增加了STM32F76xxx和STM32F77xxx分析。 – 更新了*第 5节：软件存储分区和建议*。 – 更新了*第 6节：结论*。

表23. 文档版本历史（续）

日期	版本	变更
2017 年 2 月 16 日	4	更新了整个文档，添加了对STM32F72xxx和STM32F73xxx器件的参考。 更新了*图 1：STM32F7系列系统架构*。 更新了*图 2：Flash存储器接口（路径：1、2、3、4）*。 更新了*表 1：STM32F7系列器件缓存空间大小*。 增加了*表 4： STM32F72xxx/STM32F73xxx器件 的内部存储器总结*。 更新了*图 3：DMA对一些内部存储器的不同访问（路径：5，6，7，8）*。 更新了*图 4：外部存储器接口（路径：9、10）*。 更新了*图 5：外部存储映射（复位后映射）*。 更新了*图 6：DMA与CPU之间无存储器并发访问*。 更新了*图 7：一个或多个主设备与CPU之间存在存储器并发访问*。 更新了*表 6：STM32F7系列器件之间的主要差异*。 更新了*图 13：所有主设备并发的情况下，如何对存储器A进行分区*。 更新了*第 3.1节：CPU存储器访问性能*说明： – 增加了*表 8：STM32F72xxx和STM32F73xxx器件的MDK-ARM结果*。 – 增加了*图 15：STM32F72xxx和STM32F73xxx FFT相对比例周期数，采用 MDK-ARM*。 更新了*第 3.2节：使用DMA的CPU存储器范围性能*说明： – 增加了*表 15：配置1：从Flash-ITCM执行/FFTCPU数据存储于DTCM-RAM*。 – 增加了*表 16：配置2：从Flash-ITCM执行/ FFT CPU数据存储于SRAM1*。 – 增加了*表 17：配置3：从Flash-AXI执行/ FFT CPU数据存储于DTCM-RAM*。 增加了*第 4节：STM32F7系列器件的性能排名和比较*。 更新了*第 5.2节：建议*。 更新了*第 6节：结论*。

表24. 中文文档版本历史

日期	版本	变更
2017年8月 28日	1	中文初始版本。