TB3195

管理基于 Cortex®-M7 的 MCU 的高速缓存一致性

简介

本文档概述了不同场景下的高速缓存一致性问题，并就如何管理或避免高速缓存一致性问题提供了一些方法建议。

© 2018 Microchip Technology Inc.

目录

简介 1

1. 高速缓存策略概述 3

2. 支持的配置 4

3. 高速缓存一致性问题 5

4. 利用高速缓存维护 API 处理高速缓存一致性 7

5. 禁用 DMA 和 CPU 共享存储区的高速缓存 12

6. 相关资源 15

Microchip 网站 16

变更通知客户服务 16

客户支持 16

Microchip 器件代码保护功能 16

法律声明 17

商标 17

DNV 认证的质量管理体系 18

全球销售及服务网点 19

© 2018 Microchip Technology Inc.

1. 高速缓存策略概述

表 1-1. 高速缓存策略

读取策略（高速缓存未命中情况）：	
读取分配	基于 Cortex-M7 的 MCU 上的所有可高速缓存位置均为读取分配。这意味着，在发生高速缓存未命中时将分配数据高速缓存行，将 32 字节（见**注**）数据从主存储器放入高速缓存存储器。结果是，后续访问这些存储器位置将会产生高速缓存命中条件，因此会直接从高速缓存存储器读取数据。
写入策略（高速缓存命中情况）：	
回写	在高速缓存命中时，仅更新数据高速缓存而不会更新主存储器。高速缓存行被标记为脏，直至高速缓存行被驱逐或显式清除才会写入主存储器。
直写	在高速缓存命中时，数据高速缓存和主存储器都会更新。
写入策略（缓存未命中情况）：	
写入分配	在高速缓存未命中时，会分配高速缓存行并从主存储器加载数据。这意味着，在处理器上执行存储指令可能会导致发生突发读取，将来自主存储器的数据放入高速缓存。
无写入分配	在高速缓存未命中时，不会分配高速缓存行，数据将直接写入主存储器。在读取高速缓存未命中之后才会缓存行，然后使用“读取分配”策略加载高速缓存。

注： Cortex-M7 MCU 上的高速缓存行大小为 32 字节。

© 2018 Microchip Technology Inc.

2. 支持的配置

有读取和写入分配的回写：WB-RWA

- 提供最好的性能。高速缓存命中仅更新高速缓存存储器。如果写入时高速缓存未命中，将数据从主存储器复制到高速缓存。结果是，后续访问产生高速缓存命中。

有读取分配的回写（无写入分配）：WB-NWA

- 高速缓存命中仅更新高速缓存存储器。如果写入时高速缓存未命中，不会将数据放入高速缓存。这只在写入数据时有用，但不能立即读回。

有读取分配的直写（无写入分配）：WT-NWA

- 每次写入（可能高速缓存命中，也可能高速缓存未命中）都在主存储器上执行。这就失去了高速缓存的主要优势。
- 在部分程度上解决了高速缓存一致性问题。

不可高速缓存

- 每次读取和写入都在主存储器上执行。
- 不存在高速缓存一致性问题。

© 2018 Microchip Technology Inc.

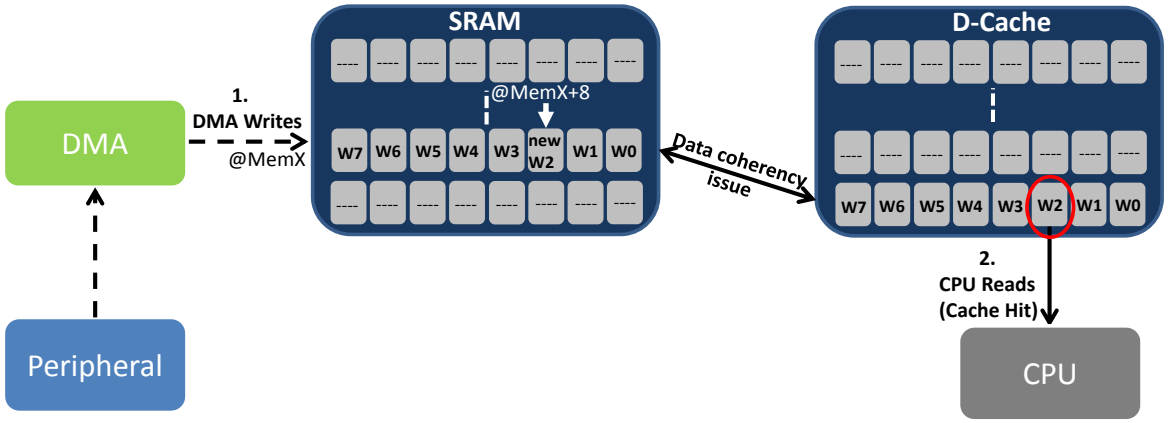
3. 高速缓存一致性问题

当多个总线主器件（如 CPU 和 DMA）在共享的存储器中看到的值相同时，就可以说该存储器区域具有一致性。

假设有一个 DMA 写入 SRAM 的应用。

条件：SRAM 上启用了高速缓存，并且可高速缓存属性被设为带有读取和写入分配（WB-RWA）的回写。CPU 此前已读取 DMA 缓冲区，因此，根据读取分配策略，高速缓存存储器中也存在相同的内容。

图 3-1. 高速缓存一致性问题——DMA 写入 SRAM



其中，

1.DMA 从外设读取数据并更新 SRAM 中的接收缓冲区。

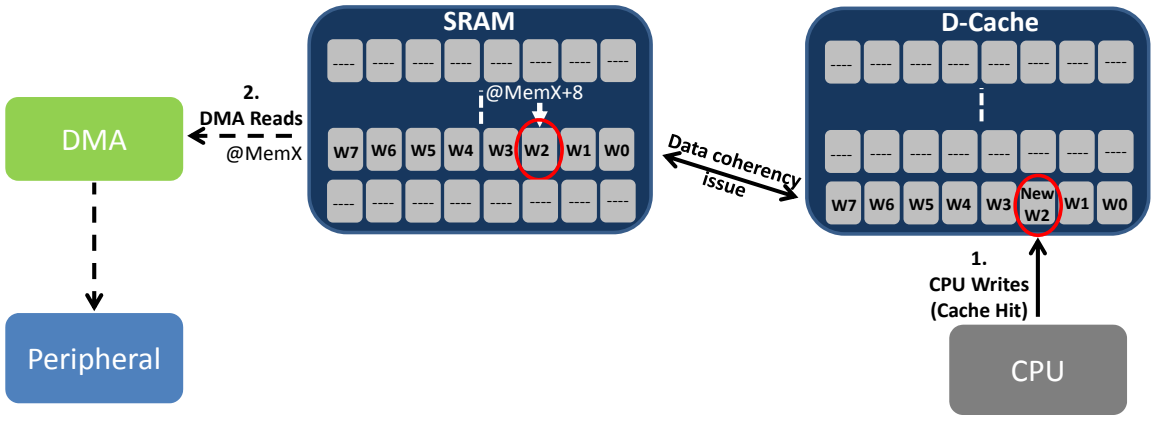
2.当 CPU 试图读取接收缓冲区时，它将读取高速缓存中存在的数据，而不是 SRAM 中可用的新数据。

假设有另一个 DMA 从 SRAM 读取的应用示例。

条件：SRAM 上启用了高速缓存，并且可高速缓存属性被设为 WB-RWA。

© 2018 Microchip Technology Inc.

图 3-2. 高速缓存一致性问题——DMA 从 SRAM 读取



其中，

1.当高速缓存策略被设置为 WB-RWA 时，CPU 会更新传输缓冲区中要传输的数据，将仅更新高速缓存而不会更新主存储器。

2.当 DMA 读取传输缓冲区时，它会读取主存储器中存在的旧值，而不是由 CPU 更新、仍在高速缓存中的新值。

4. 利用高速缓存维护 API 处理高速缓存一致性

此解决方案要求应用在运行时使用 Cortex-M7 高速缓存维护操作来管理高速缓存。高速缓存维护 API 使用户能够执行以下操作：

1.启用或禁用高速缓存——打开或关闭高速缓存。

2.使高速缓存无效——将高速缓存行标记为无效。由于读取分配和写入分配策略，后续访问会强制将数据从主存储器复制到高速缓存。

3.清理高速缓存——将标记为脏的高速缓存行回写到主存储器。

Cortex 单片机软件接口标准（Cortex Microcontroller Software Interface Standard，CMSIS）提供了以下数据高速缓存维护 API：

表 4-1. CMSIS 数据高速缓存维护 API

高速缓存维护 API	描述
SCB_EnableDCache (void)	启用数据高速缓存。启用之前，整个数据高速缓存无效。
SCB_DisableDCache (void)	禁用数据高速缓存。禁用高速缓存之前，对数据高速缓存进行清理，将脏数据刷新到主存储器。
SCB_InvalidateDCache(void)	使整个数据高速缓存无效。
SCB_InvalidateDCache_by_Addr (uint32_t * addr, int32_t dsize)	按地址使数据高速缓存行无效。
SCB_CleanDCache(void)	清理数据高速缓存。
SCB_CleanDCache_by_Addr (uint32_t *addr, int32_t dsize)	按地址清理数据高速缓存行。
SCB_CleanInvalidateDCache(void)	清理并使整个数据高速缓存无效。
SCB_CleanInvalidateDCache_by_Addr(uint32_t *addr, int32_t dsize)	按地址清理并使数据高速缓存行无效。

在使用按地址清理并使数据高速缓存无效的 API 时：

addr——必须与高速缓存行大小边界对齐。这意味着 DMA 缓冲区地址必须与 32 字节边界对齐。

dsize——必须是高速缓存行大小的倍数。这意味着 DMA 缓冲区大小必须是 32 字节的倍数。

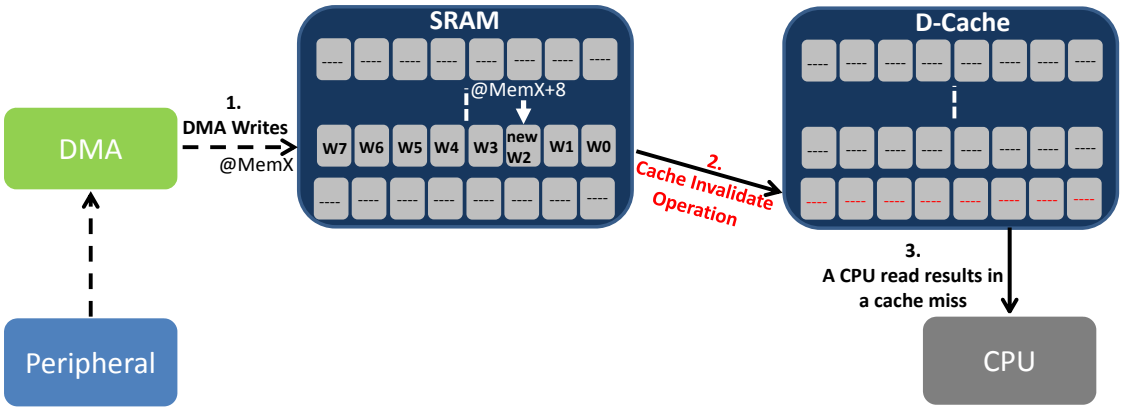
在 DMA 写入 SRAM 时使用高速缓存维护 API

条件：缓存策略为 WB-RWA。CPU 首先访问接收缓冲区（rx_buffer[]），将它高速缓存在数据高速缓存中。

1. DMA 将数据写入 rx_buffer[]。
2. 执行高速缓存无效操作以使高速缓存的 rx_buffer[]无效。
3. CPU 试图读取 rx_buffer[]时导致高速缓存未命中，因为在第 2 步中已使 rx_buffer[]无效。

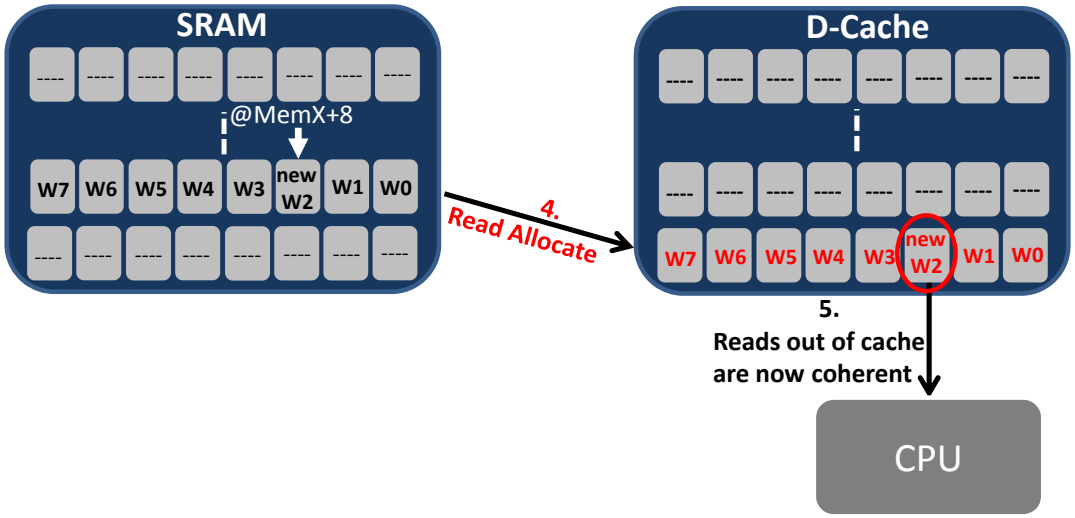
© 2018 Microchip Technology Inc.

图 4-1. DMA 写入 SRAM 之后执行高速缓存无效操作



4. 由于读取分配策略，将会分配高速缓存行，并将数据从 SRAM 中的 rx_buffer[]复制到分配的高速缓存行。
5. 然后，CPU 将从高速缓存中读取一致的数据。

图 4-2. 执行高速缓存无效操作之后，将从高速缓存中读取一致的数据



以下代码示例（使用 GCC 编译器）显示了如何定义与高速缓存行大小边界对齐的 DMA 缓冲区。BUFFER_SIZE 必须是高速缓存行大小（32 字节）的倍数。DMA_TRANSFER_SIZE 是 DMA 传输的字节数。完成 DMA 读取操作后，使用高速缓存无效 API 使高速缓存中的接收缓冲区无效。main 函数使用高速缓存维护 API 来启用数据高速缓存。

注： 本技术简介中提供的所有代码示例均参考 Microchip 的 Atmel 软件框架（ASF3）下可用的 API 函数。

代码示例——完成 DMA 传输后执行高速缓存无效操作

```
/* The rx_buffer is aligned to 32-byte boundary.The BUFFER_SIZE is a multiple of cache line
size (32-bytes)*/
#define BUFFER_SIZE         32

__attribute__ ((aligned (32))) uint8_t rx_buffer[BUFFER_SIZE];

volatile bool rx_xfer_done;
/**
 * \brief XDMAC interrupt handler.
 */
void XDMAC_Handler(void)
{
    uint32_t dma_status;

    dma_status = xdmac_channel_get_interrupt_status(XDMAC, XDMA_CH_RX);

    if (dma_status & XDMAC_CIS_BIS)
    {
        rx_xfer_done = true;
        SCB_InvalidateDCache_by_Addr((uint32_t*)rx_buffer, DMA_TRANSFER_SIZE);
    }
}
int main (void)
{
    ......
    /* Enabling the D-Cache */
    SCB_EnableDCache();
    /* Setup and trigger a DMA transfer */
    ......
    while (false == rx_xfer_done);
    /* Access to the rx_buffer[] is coherent now */
}
```

在 DMA 从 SRAM 读取时使用高速缓存维护 API

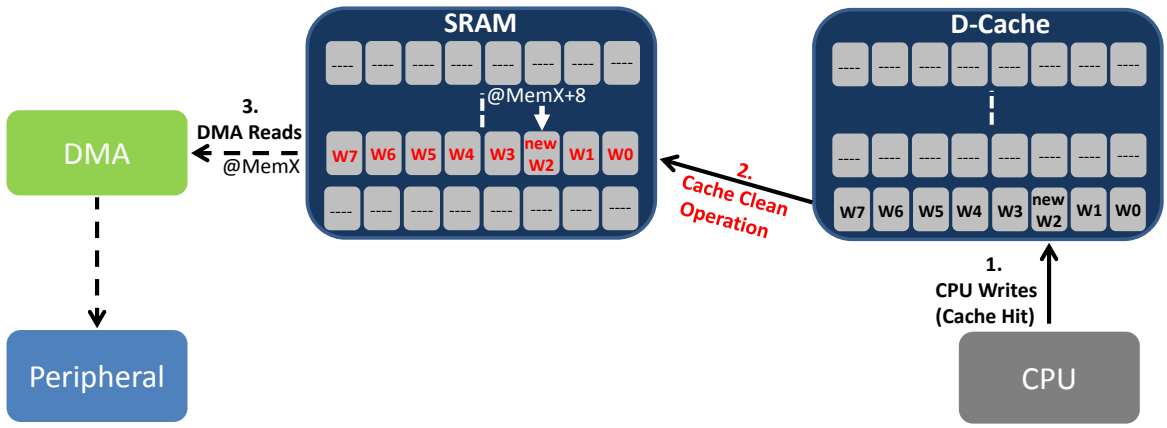
条件：缓存策略设为 WB-RWA。CPU 首先访问传输缓冲区（tx_buffer[]），将它高速缓存在 D-Cache 中。

1.CPU 将数据写入 tx_buffer[]，并由 DMA 传输。

2.在启用 DMA 传输之前，执行高速缓存清理操作，将高速缓存的 tx_buffer[]刷新到 SRAM 中。

3.DMA 将从 SRAM 中读取一致的数据。

图 4-3. 在 CPU 向数据高速缓存写入之后执行高速缓存清理操作



代码示例——在 CPU 向数据高速缓存写入之后执行高速缓存清理操作

```
int main (void)
{
     ......

    strcpy(tx_buffer, "DMA Transmit String");

    SCB_CleanDCache_by_Addr((uint32_t*)tx_buffer, DMA_TRANSFER_SIZE);

    xdmac_channel_enable(XDMAC, XDMA_CH_TX);
}
```

在上述代码示例中，在启用 DMA 传输之前，CPU 执行缓存清理操作，将传输缓冲区中的更新数据写入 SRAM。

注： 如果使用了 DMA 链接描述符，则每次更新描述符时，应用必须清理与链接描述符地址对应的高速缓存，以维持 DMA 和 CPU 之间的一致性。

重要： 所有高速缓存操作均在 32 字节的高速缓存行上执行。因此，如果上例中的传输和接收缓冲区的大小不是 32 字节的倍数，则高速缓存无效或高速缓存清理操作可能会产生意外行为，如以下代码示例所示。

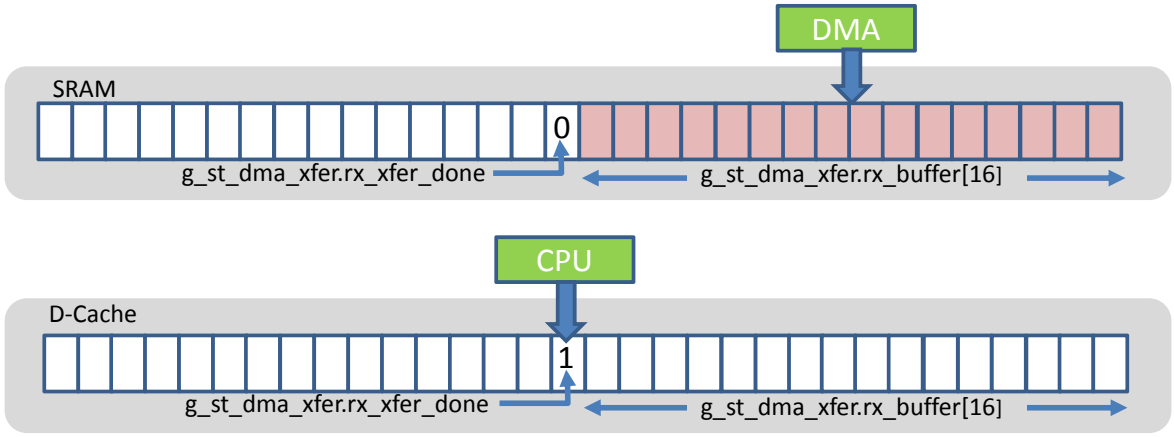
代码示例——不是 32 字节倍数的 DMA 缓冲区产生的影响

```
#define BUFFER_SIZE        16

typedef struct
{
    /* The rx_buffer is aligned to 32-byte boundary.
       The BUFFER_SIZE is 16-bytes which is not a multiple of the cache line size.
    */
    __attribute__ ((aligned (32))) uint8_t rx_buffer[BUFFER_SIZE];

    bool rx_xfer_done;

}st_dma_xfer;

static st_dma_xfer g_st_dma_xfer;

/**
 * \brief XDMAC interrupt handler.
 */
void XDMAC_Handler(void)
{
    uint32_t dma_status;

    dma_status = xdmac_channel_get_interrupt_status(XDMAC, XDMA_CH_RX);

    if (dma_status & XDMAC_CIS_BIS)
    {
        g_st_dma_xfer.rx_xfer_done = true;
        SCB_InvalidateDCache_by_Addr((uint32_t*)g_st_dma_xfer.rx_buffer,
              DMA_TRANSFER_SIZE);
    }
}
```

在上述代码示例中，接收缓冲区为 16 字节。DMA 从外设读取 16 字节到 SRAM 中的 g_st_dma_xfer.rx_buffer[]并生成 DMA 中断。在 DMA ISR 中，CPU 将数据高速缓存中的 g_st_dma_xfer.rx_xfer_done 标志置 1。此存储器位置先前由 CPU 访问，因而在数据高速缓存中可用。然后执行高速缓存无效操作，以使高速缓存行无效。

© 2018 Microchip Technology Inc.

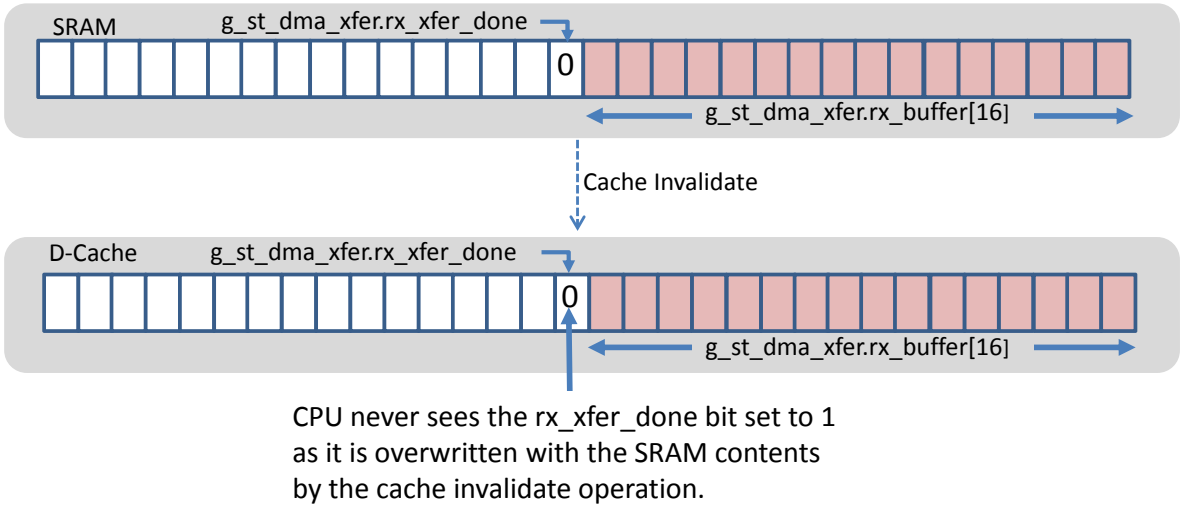
图 4-4. DMA 更新 SRAM 中的接收缓冲区，CPU 更新数据高速缓存中的标志



由于高速缓存行无效，因此 CPU 在 main 函数中访问 g_st_dma_xfer.rx_xfer_done 标志会使得整个 32 字节高速缓存行从 SRAM 复制到数据高速缓存（由于读取分配策略）。这会将 g_st_dma_xfer.rx_xfer_done 标志重写为 0。

因此，CPU 永远不能将 g_st_dma_xfer.rx_xfer_done 标志置 1。

图 4-5. 高速缓存无效操作无意中破坏数据高速缓存中存在的数据



造成此问题的原因是 DMA 缓冲区不是 32 字节的倍数。请注意，即使 DMA 配置为向外设传输/从外设接收 32 字节的非整数倍数据，DMA 缓冲区也必须是 32 字节的整数倍，以免破坏同一高速缓存行中定义的变量。例如，如果 DMA 配置为从外设读取/向外设写入 50 字节，则 DMA 缓冲区的大小必须是 64。

© 2018 Microchip Technology Inc.

5. 禁用 DMA 和 CPU 共享存储区的高速缓存

在此方法中，CPU 和 DMA 共享的存储区通过存储器保护单元（Memory Protection Unit，MPU）被定义为不可高速缓存，同时将只能由 CPU 访问的存储区保留为可高速缓存。

用例：共享存储器可由 CPU 和 DMA 同时更新。例如，CPU 和 DMA 可以同时更新 *GMAC 接收缓冲区描述符条目*中的所有权位。

优点：对应用透明。无需高速缓存维护。将驱动程序从没有高速缓存的 MCU 移植到有高速缓存的 MCU 变得很容易。

缺点：需使用 MPU 来创建专用的不可高速缓存存储区。这需要复杂的链接器脚本文件。

配置 MPU 以创建不可高速缓存的存储区：

通过 SAM Cortex-M7 MCU，用户可以创建多达 16 个 MPU 区域。下表所示为用于配置和启用存储区的 MPU 寄存器。有关详细信息，请参考 http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0646b/BIHJJABA.html

表 5-1. MPU 寄存器

MPU 寄存器	描述
MPU_RNR	选择 MPU_RBAR 和 MPU_RASR 寄存器引用的存储区。有效值范围为 0-15，对应 16 个 MPU 区域。
MPU_RBAR	定义 MPU 区域的基址。区域起始地址必须与区域的大小对齐。（即，64 KB 区域必须在 0x00010000 或 0x00020000 地址上以 64KB 的倍数对齐）。
MPU_RASR	定义 MPU 区域的区域大小和存储属性，然后启用该区域。允许的区域大小下限为 32 个字节，且必须是 2 的幂次方。
MPU_CTRL	启用/禁用 MPU

MPU_RASR 寄存器的 TEX、C 和 B 位定义存储区的可高速缓存性。下表所示为标准存储区类型的编码。

表 5-2. MPU 访问权限属性

TEX	C	B	可共享	存储器类型	描述
000	1	0	是	标准	直写，无写入分配
000	1	1	是	标准	回写，无写入分配
001	0	0	是	标准	不可高速缓存
001	1	1	是	标准	回写、写入和读取分配（启用高速缓存时，SAM Cortex-M7 MCU 上的默认高速缓存策略）

有关详细信息，请参见 Arm®信息中心提供的“4.6.6 MPU 访问权限属性”一节。有关配置 MPU 的更多信息，请参见 TB3179——如何配置存储区保护单元（MPU）。

以下代码示例将禁用 MPU，然后配置并启用从 0x2045F0000 开始的一部分 SRAM 存储区，大小为 4096 字节，不可高速缓存。使用默认（WB-RWA）高速缓存策略将剩余 SRAM 存储区保留为可高速缓存。将访问权限（Access Permission，AP）设为完全访问以允许特权和非特权软件拥有 RW 访问权限，稍后它将启用 MPU。

© 2018 Microchip Technology Inc.

代码示例——创建不可高速缓存的存储区的 MPU 配置

```
#define SRAM_NOCACHE_START_ADDRESS          (0x2045F000UL)
#define NOCACHE_SRAM_REGION_SIZE            0x1000
#define MPU_NOCACHE_SRAM_REGION             (11)
#define INNER_OUTER_NORMAL_NOCACHE_TYPE(x)  ((0x01 << MPU_RASR_TEX_Pos ) | ( DISABLE <<
MPU_RASR_C_Pos ) | ( DISABLE << MPU_RASR_B_Pos ) | ( x << MPU_RASR_S_Pos))

    /* Disable the MPU region */
    MPU->CTRL = MPU_DISABLE;
    dw_region_base_addr =
        SRAM_NOCACHE_START_ADDRESS |
        MPU_REGION_VALID |
        MPU_NOCACHE_SRAM_REGION;

    dw_region_attr =
        MPU_AP_FULL_ACCESS    |
        INNER_OUTER_NORMAL_NOCACHE_TYPE( SHAREABLE ) |
        mpu_cal_mpu_region_size(NOCACHE_SRAM_REGION_SIZE) |
        MPU_REGION_ENABLE;

    MPU->RBAR = dw_region_base_addr;
    MPU->RASR = dw_region_attr;

    /* Enable the MPU region */
    MPU->CTRL = (MPU_ENABLE | MPU_PRIVDEFENA);
    __DSB();
    __ISB();
```

可以修改链接器脚本文件，以定义不可高速缓存的存储空间，并将 DMA 缓冲区链接到不可高速缓存的存储区，如以下 GNU 链接器脚本代码示例所示。

修改链接器脚本以创建不可高速缓存数据的存储区域

```
/* Memory Spaces Definitions */
MEMORY
{
  rom (rx)  : ORIGIN = 0x00400000, LENGTH = 0x00200000
  ram (rwx) : ORIGIN = 0x20400000, LENGTH = 0x0005F000
  ram_nocache (rwx) : ORIGIN = 0x2045F000, LENGTH = 0x00001000
}
/* Section Definitions */
SECTIONS
{
    ……
        .ram_nocache (NOLOAD):
        {
        . = ALIGN(4);
        _s_ram_nocache = .;
        *(.ram_nocache)
        . = ALIGN(4);
        _e_ram_nocache = .;
           } > ram_nocache

    .ram_nocache_data : AT (_etext + SIZEOF(.relocate))
        {
        . = ALIGN(4);
        _s_ram_nocache_vma = .;
        _s_ram_nocache_lma = LOADADDR(.ram_nocache_data);
        *(.ram_nocache_data)
        . = ALIGN(4);
        _e_ram_nocache_vma = .;
        } > ram_nocache
    ……….
}
```

在上述链接器脚本示例中，将.ram_nocache_data 部分的加载存储器地址指定到.text 和.relocate 部分末尾。

在复位处理程序中使用以下代码将.ram_nocache 部分下定义的未初始化变量归零，并复制.ram_nocache_data 中已初始化变量的初始值（从闪存复制到 SRAM）。

修改 C 启动模式以初始化不可高速缓存数据的存储区域

```
extern uint32_t _s_ram_nocache;
extern uint32_t _e_ram_nocache;
extern uint32_t _s_ram_nocache_vma;
extern uint32_t _e_ram_nocache_vma;
extern uint32_t _s_ram_nocache_lma;

void Reset_Handler(void)
{
    uint32_t *pSrc, *pDest;
    .........

    /* Initialize the no cache data segment */
    pSrc = &_s_ram_nocache_lma;
    pDest = &_s_ram_nocache_vma;

    if (pSrc != pDest) {
        for (; pDest < &_e_ram_nocache_vma;) {
            *pDest++ = *pSrc++;
        }
    }

    /* Clear the no cache zero segment */
    for (pDest = &_s_ram_nocache; pDest < &_e_ram_nocache;) {
        *pDest++ = 0;
    }
    .........
}
```

在此应用中，可将 DMA 缓冲区分配给.ram_nocache 存储区，如以下代码示例所示。 如果应用已在无高速缓存存储区初始化变量，则必须将其定义在.ram_nocache_data 部分之下。

在不可高速缓存存储区中定义缓冲区的应用代码

```
__attribute__ ((section (".ram_nocache"), aligned (32))) uint8_t rx_buf[BUFFER_SIZE];
__attribute__ ((section (".ram_nocache"), aligned (32))) uint8_t tx_buf[BUFFER_SIZE];
```

避免高速缓存一致性问题的另一方法是使用紧耦合存储器（Tightly Coupled Memory，TCM），因为 TCM 的内容不会被高速缓存，且 CPU 和 DMA 都能访问它。 访问 TCM 的速度与访问高速缓存的速度差不多，而且不会受到高速缓存未命中和高速缓存一致性问题的困扰。

用例：缓冲区大小超过高速缓存大小（16 KB）。

优点：对性能没有影响。对应用透明（无需高速缓存维护）。

缺点：需修改链接器脚本文件。

有关使用 TCM 的更多信息，请参见参考部分中提供的链接。

6. 相关资源

有关更多信息，请参见以下可下载文档：

1. Arm® Cortex-M7 处理器技术参考手册——L1 缓存
2. Arm® Cortex-M7 处理器技术参考手册——存储器保护单元
3. http://ww1.microchip.com/downloads/en/AppNotes/Atmel-44047-Cortex-M7-Microcontroller-Optimize-Usage-SAM-V71-V70-E70-S70-Architecture_Application-note.pdf
4. 如何配置存储器保护单元（MPU）
5. Atmel SMART SAM V7x TCM 存储器
6. Atmel SMART SAM E70 TCM 存储器

© 2018 Microchip Technology Inc.

Microchip 网站

Microchip 网站 http://www.microchip.com/为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的互联网浏览器即可访问，网站提供以下信息：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题（FAQ）、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

变更通知客户服务

Microchip 的变更通知客户服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 http://www.microchip.com/。在“支持”（Support）下，点击“变更通知客户”（Customer Change Notification）服务后按照注册说明完成注册。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（FAE）
- 技术支持

客户应联系其代理商、代表或应用工程师（FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过以下网站获得技术支持： http://www.microchip.com/support

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿意与关心代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如

© 2018 Microchip Technology Inc.

果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，否则在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、Microchip 徽标、AnyRate、AVR、AVR 徽标、AVR Freaks、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KeeLoq、Kleer、LANCheck、LINK MD、maXStylus、maXTouch、MediaLB、megaAVR、MOST、MOST 徽标、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32 徽标、Prochip Designer、QTouch、SAM-BA、SpyNIC、SST、SST 徽标、SuperFlash、tinyAVR、UNI/O 和 XMEGA 是 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

ClockWorks、The Embedded Control Solutions Company、EtherSynch、Hyper Speed Control、HyperLight Load、IntelliMOS、mTouch、Precision Edge 和 Quiet-Wire 为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、memBrain、Mindi、MiWi、motorBench、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Inc.在美国的服务标记。

Silicon Storage Technology 为 Microchip Technology Inc.在除美国外的国家或地区的注册商标。

GestIC 是 Microchip Technology Inc.的子公司 Microchip Technology Germany II GmbH & Co. KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2018, Microchip Technology Incorporated 版权所有。

ISBN： 978-1-5224-3744-4

AMBA、Arm、Arm7、Arm7TDMI、Arm9、Arm11、Artisan、big.LITTLE、Cordio、CoreLink、CoreSight、Cortex、DesignStart、DynamIQ、Jazelle、Keil、Mali、Mbed、Mbed Enabled、NEON、

POP、RealView、SecurCore、Socrates、Thumb、TrustZone、ULINK、ULINK2、ULINK-ME、ULINK-PLUS、ULINKpro、µVision 和 Versatile 是 Arm Limited（或其子公司）在美国和/或其他国家/地区的商标或注册商标。

DNV 认证的质量管理体系

ISO/TS 16949

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。Microchip 的 PIC® MCU 和 dsPIC® DSC、KEELOQ®跳码器件、串行 EEPROM、单片机外设、非易失性存储器及模拟产品严格遵守公司的质量体系流程。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

© 2018 Microchip Technology Inc.

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：1-480-792-7200 传真：1-480-792-7277 技术支持： http://www.microchip.com/support 网址： www.microchip.com **亚特兰大** 德卢斯，乔治亚州 电话：1-678-957-9614 传真：1-678-957-1455 **奥斯汀，德克萨斯州** 电话：1-512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：1-774-760-0087 传真：1-774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：1-630-285-0071 传真：1-630-285-0075 **达拉斯** 艾迪生，德克萨斯州 电话：1-972-818-7423 传真：1-972-818-2924 **底特律** 诺维，密歇根州 电话：1-248-848-4000 **休斯敦，德克萨斯州** 电话：1-281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：1-317-773-8323 传真：1-317-773-5453 电话：1-317-536-2380 **洛杉矶** 米申维耶霍，加利福尼亚州 电话：1-949-462-9523 传真：1-949-462-9608 电话：1-951-273-7800 **罗利，北卡罗来纳州** 电话：1-919-844-7510 **纽约，纽约州** 电话：1-631-435-6000 **圣何塞，加利福尼亚州** 电话：1-408-735-9110 电话：1-408-436-4270 **加拿大 - 多伦多** 电话：1-905-695-1980 传真：1-905-695-2078	**中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 珠海** 电话：86-756-3210040 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **台湾地区 - 新竹** 电话：886-3-577-8366	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880- 3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4450-2828 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-67-3636 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 赖阿南纳** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-7288-4388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820

© 2018 Microchip Technology Inc.