TB3195

Managing Cache Coherency on Cortex-M7 Based MCUs

Introduction

This document provides an overview of the cache coherency issue under different scenarios. It also suggests methods to manage or avoid the cache coherency issue.

Table of Contents

1. Cache Policies Overview

Table 1-1. Cache Policies

Read Policy (Cache miss case):	
Read Allocate	All cacheable locations on Cortex-M7 based MCUs are read allocate. This means that the data cache lines are allocated when a cache miss occurs, bringing 32 bytes (See **Note**) of data from the main memory into the cache memory. As a result, subsequent access to these memory locations will result in a cache hit condition, and the data is directly read from the cache memory.
Write Policy (Cache hit case):	
Write Back	On a cache hit, only the data cache is updated and not the main memory. The cache line is marked as dirty, and writes to the main memory are postponed until the cache line is evicted, or explicitly cleaned.
Write Through	On a cache hit, both the data cache and the main memory are updated.
Write Policy (Cache miss case):	
Write Allocate	On a cache miss, a cache line is allocated and loaded with the data from the main memory. This means that executing a store instruction on the processor might cause a burst read to occur to bring the data from the main memory to cache.
No Write Allocate	On a cache miss, a cache line is not allocated and the data is written directly into the main memory. Here, a line is not cached until a cache miss on a read occurs, which then loads the cache using the Read Allocate policy.

Note: The size of a cache line on Cortex-M7 MCUs is 32 bytes.

2. Supported Configurations

Write-back with read and write allocate: WB-RWA

- Provides the best performance. The cache hits only update the cache memory. Cache misses on a write, copy data from the main memory to the cache. As a result, subsequent access results in a cache hit.

Write-back with read allocate (no write allocate): WB-NWA

- The cache hits only update the cache memory. Cache misses on a write do not bring the data to the cache. This is advantageous only when the data is written, but not immediately read back.

Write-through with read allocate (no write allocate): WT-NWA

- Each write (either cache hit, or cache miss) is performed on the main memory. This negates the main advantage of having cache.
- Partially solves the cache coherency issue.

Non-cacheable

- Each read and write is performed on the main memory.
- No cache coherency related issues.

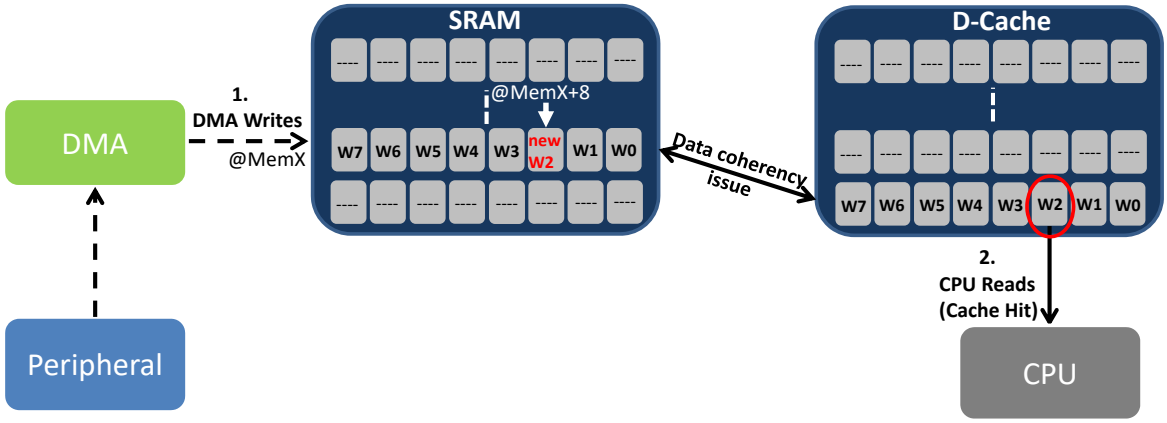
3. Cache Coherency Issues

A memory region is said to be coherent when multiple bus masters, for example, CPU and DMA have the same view of the memory shared between them.

Consider an application where the DMA writes to the SRAM.

Conditions: Cache is enabled on SRAM and the cacheability attribute is set to write-back with read and write-allocate (WB-RWA). The CPU has previously read the DMA buffer and therefore, the same is available in the cache memory due to the read allocate policy.

Figure 3-1. Cache Coherency Issue - DMA Writes to SRAM



Where,

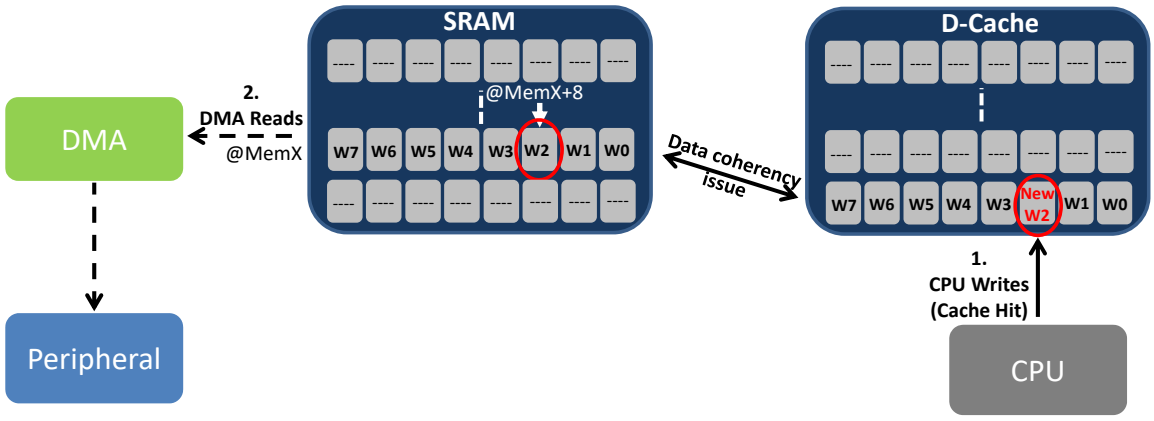
1. The DMA reads the data from the peripheral and updates the receive buffer in the SRAM.

2. When the CPU tries to read the receive buffer, it will read the data present in the cache and not the new data available in the SRAM.

Consider another example, where the DMA reads from the SRAM.

Conditions: The cache is enabled on the SRAM, and the cacheability attribute is set to WB-RWA.

Figure 3-2. Cache coherency Issue - DMA Reads from SRAM



Where,

1. The CPU updates the data to be transmitted in a transmit buffer as the cache policy is set to WB-RWA, only the cache is updated and not the main memory.

2. When the DMA reads the transmit buffer, it reads the old value present in the main memory and not the latest value updated by the CPU which is still in the cache.

4. Using Cache Maintenance APIs to Handle Cache Coherency

This solution requires the application to manage the cache at run-time using the Cortex-M7 cache maintenance operations. The cache maintenance APIs enable users to perform these actions:

1. Enable or disable cache – Cache on or off.

2. Invalidate cache – Marks the cache lines as invalid. Subsequent access forces the data to be copied from the main memory to the cache, due to the read-allocate and write-allocate policies.

3. Clean cache – Writes the cache lines, which are marked as dirty, back to the main memory.

The Cortex Microcontroller Software Interface Standard (CMSIS) provides the following D-Cache maintenance APIs:

Table 4-1. CMSIS Data Cache Maintenace APIs

Cache Maintenance API	Description
SCB_EnableDCache (void)	Enables data cache. Invalidates the entire data cache before enabling it.
SCB_DisableDCache (void)	Disables data cache. Cleans the data cache to flush dirty data to main memory before disabling the cache.
SCB_InvalidateDCache(void)	Invalidate the entire data cache.
SCB_InvalidateDCache_by_Addr (uint32_t * addr, int32_t dsize)	Invalidate the data cache line by address.
SCB_CleanDCache(void)	Cleans the data cache.
SCB_CleanDCache_by_Addr (uint32_t *addr, int32_t dsize)	Cleans the data cache line by address.
SCB_CleanInvalidateDCache(void)	Cleans and Invalidates the entire data cache.
SCB_CleanInvalidateDCache_by_Addr(uint32_t *addr, int32_t dsize)	Cleans and Invalidates the data cache line by address.

When using the cache clean and cache invalidate by address APIs:

addr – Must be aligned to the cache line size boundary. This means that the DMA buffer address must be aligned to the 32-byte boundary.

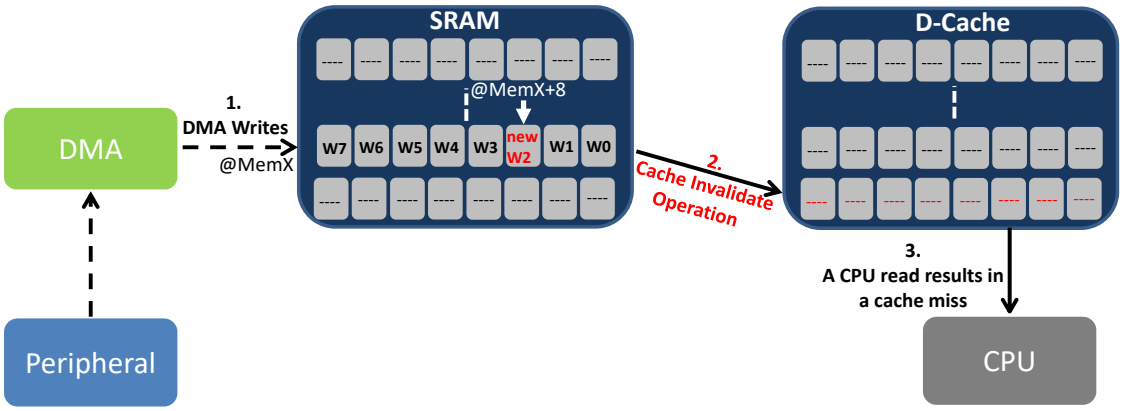
dsize – Must be a multiple of the cache line size. This means that the DMA buffer size must be a multiple of 32-bytes.

Using cache maintenance API when DMA writes to SRAM

Conditions: The cache policy is WB-RWA. The CPU initially accessed the receive buffer (rx_buffer[]), and cached it in the D-Cache.

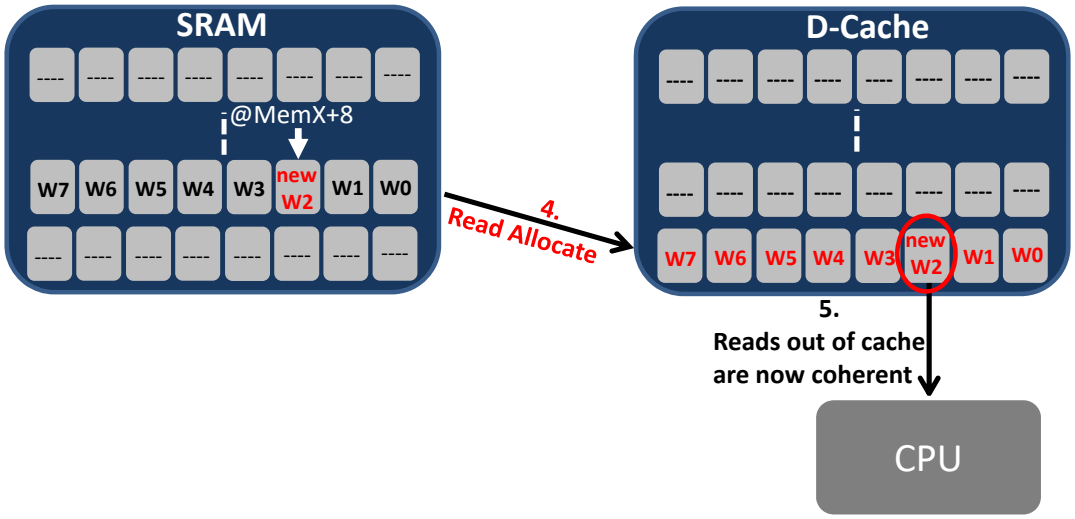
1. DMA writes data to the rx_buffer[].
2. A cache invalidate operation is performed to invalidate the cached rx_buffer[].
3. CPU tries to read the rx_buffer[] and results in a cache miss as rx_buffer[] was invalidated in step 2.

Figure 4-1. Cache Invalidate Operation After DMA Writes to SRAM



4. Due to the read-allocate policy, a cache line is allocated and copies data from the rx_buffer[] in the SRAM to the allocated cache line.
5. The CPU reads from the cache will then be coherent.

Figure 4-2. After a Cache Invalidate Operation, Reads Out of D-Cache by CPU are Coherent



The following code sample shows (using the GCC compiler) how to define the DMA buffers aligned to the cache line size boundary. The BUFFER_SIZE must be a multiple of the cache line size (32-bytes). The DMA_TRANSFER_SIZE is the number of bytes transferred by the DMA. Once the DMA read operation is complete, the receive buffer in cache is invalidated using the cache invalidate API. The main function enables the data cache, using the cache maintenance APIs.

Note: All the code samples provided in this tech brief refer to the API functions available under Microchip's Atmel Software Framework (ASF3).

Code Showing Cache Invalidate Operation After DMA Transfer is Complete

```
/* 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 */
}
```

Using cache maintenance API when DMA reads from SRAM

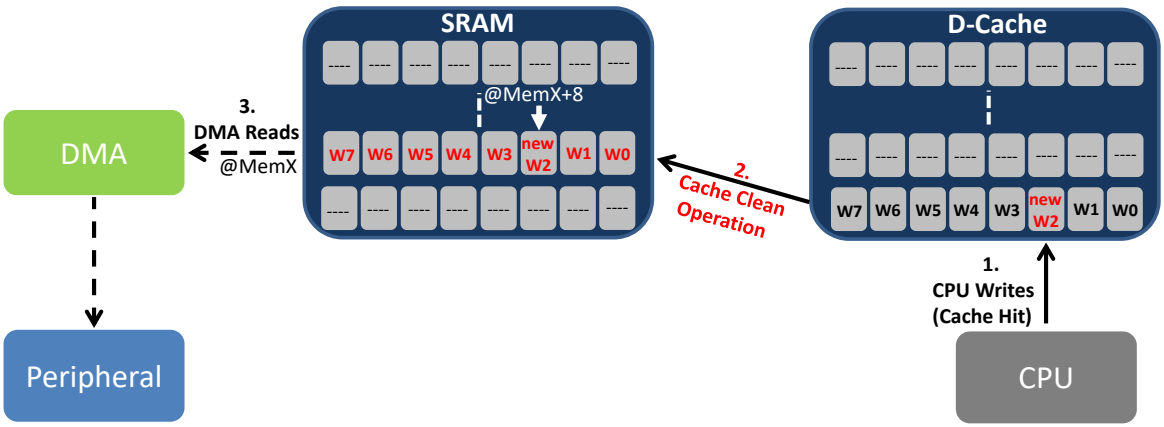
Conditions: The cache policy is set to WB-RWA. The CPU initially accessed the transmit buffer (tx_buffer[]), and cached it in the D-Cache.

1. The CPU writes data to the tx_buffer[] which will be transmitted by the DMA.

2. A cache clean operation is performed to flush the cached tx_buffer[] into the SRAM before enabling the DMA transfer.

3. The DMA reads from the SRAM will now be coherent.

Figure 4-3. Cache Clean Operation After CPU Writes to D-Cache



Code Showing A Cache Clean Operation after the CPU writes to D-Cache

```
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);
}
```

In the previous code sample, before enabling the DMA transfer, a cache clean operation by CPU writes the updated data in the transmit buffer to the SRAM.

Note: If the DMA link descriptors are used, then every time the descriptors are updated, the application must clean the cache corresponding to the link descriptor addresses to maintain coherency between the DMA and the CPU.

Important: All the cache operations are performed on a cache line of 32-bytes. As a result, if the size of the transmit and receive buffers in the above example are not a multiple of 32-bytes, a cache invalidate or cache clean operation could lead to unexpected behavior as shown in the following code sample.

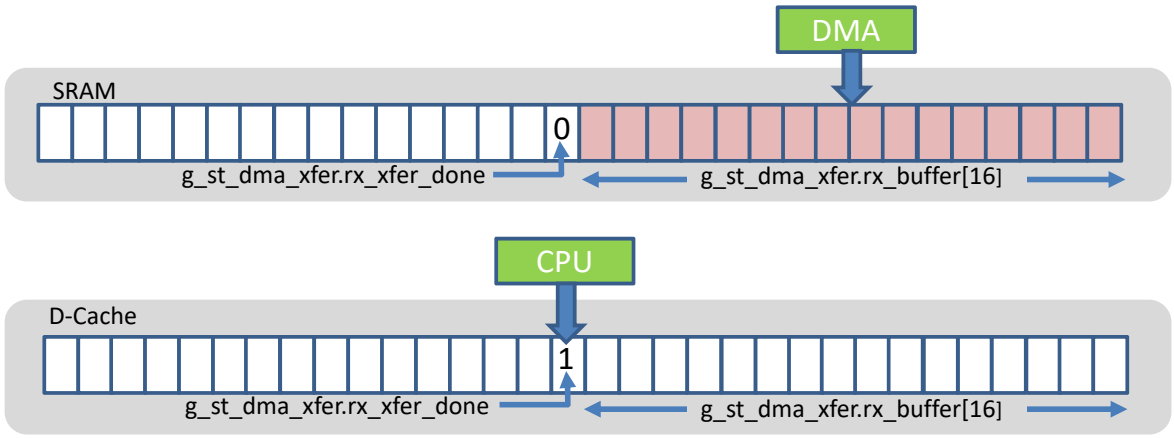
Code Illustrating the Effect of DMA Buffers That are Not a Multiple of 32-Bytes

```
#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);
    }
}
```

In the previous code sample, the receive buffer is16 bytes. The DMA reads 16 bytes from the peripheral into the g_st_dma_xfer.rx_buffer[] in SRAM and generates a DMA interrupt. In the DMA ISR, the CPU sets the g_st_dma_xfer.rx_xfer_done flag to 1 in D-cache. This memory location was previously

accessed by the CPU and therefore it is available in the D-cache. A cache invalidate operation is then performed, thereby invalidating the cached line.

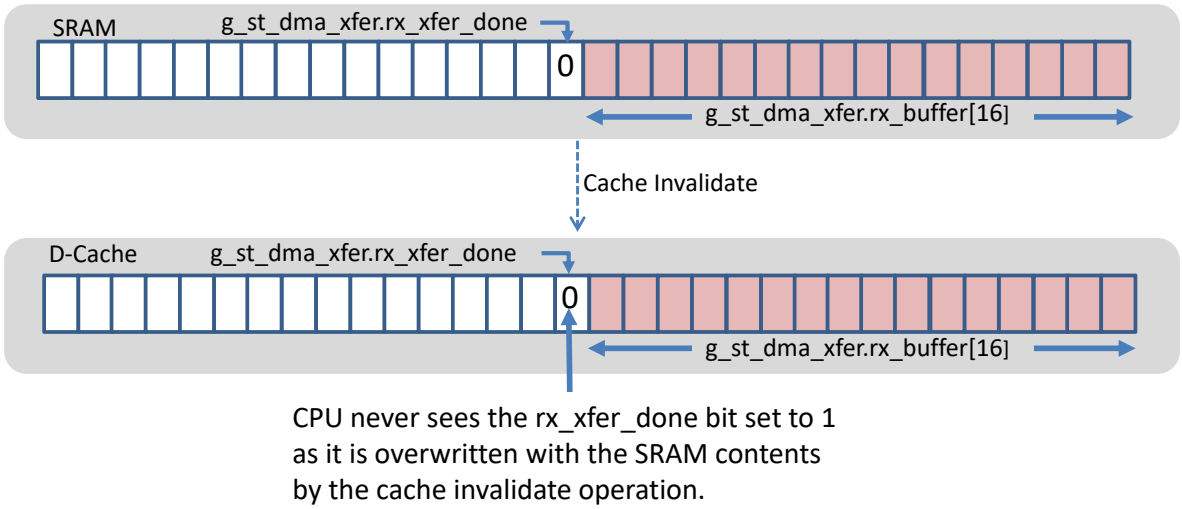
Figure 4-4. DMA Updates the Receive Buffer in SRAM and CPU Updates a Flag in D-Cache



Since the cache line is invalidated, access to the g_st_dma_xfer.rx_xfer_done flag by the CPU in the main function, result in the entire cache line of 32 bytes copied from the SRAM to the D-cache (due to the read allocate policy). This overwrites the g_st_dma_xfer.rx_xfer_done flag back to 0.

As a result, the CPU never sees the g_st_dma_xfer.rx_xfer_done flag set to 1.

Figure 4-5. A Cache Invalidate Operation Inadvertantly Corrupts Data Present in the Data Cache



This issue is caused as the DMA buffer is not a multiple of 32 bytes. Note that even if the DMA is configured to transfer a non-integer multiple of 32 bytes of data to/from a peripheral, the DMA buffer must be an integer multiple of 32 bytes to avoid corruption of variables defined in the same cache line. For example, if the DMA is configured to read/write 50 bytes to/from a peripheral then the DMA buffers must be of size 64.

5. Disabling Cache on Memory Regions Shared by the DMA and CPU

In this approach, the memory regions shared by the CPU and DMA are defined as non-cacheable using the Memory Protection Unit (MPU), while leaving the memory regions that are only accessed by the CPU as cacheable.

Use case: Shared memory can be updated by the CPU and DMA simultaneously. For example, the Ownership bit in the *GMAC receive buffer descriptor entry* can be updated simultaneously by the CPU and DMA.

Advantage: Transparent to the application. No cache maintenance is required. Porting a driver from a MCU without cache to a MCU with cache becomes easy.

Drawbacks: Requires the use of an MPU to create a dedicated non-cacheable memory region. This requires a complex linker script file.

Configuring the MPU to create a non-cacheable memory region:

Using the SAM Cortex-M7 MCUs users can create up to 16 MPU regions. The following table shows the MPU registers used to configure and enable a memory region. For detailed information, refer to the http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0646b/BIHJJABA.html

Table 5-1. MPU Registers

MPU Register	Description
MPU_RNR	Selects which memory region is referenced by the MPU_RBAR and MPU_RASR registers. Valid values range from 0-15 corresponding to the 16 MPU regions.
MPU_RBAR	Defines the base address of the MPU region. The region start address must align to the size of the region. (i.e., A 64KB region must be aligned on a multiple of 64KB, at 0x00010000 or 0x00020000).
MPU_RASR	Defines the region size and memory attributes of the MPU region and then enables that region. The smallest permitted region size is 32 bytes and must be a power of 2.
MPU_CTRL	Enables/Disables MPU

The TEX, C and B bits of the MPU_RASR register define the cacheability of the memory regions. The following table shows the encoding for the normal memory type.

Table 5-2. MPU Access Permission Attributes

TEX	C	B	Shareable	Memory Type	Description
000	1	0	Yes	Normal	Write-through, no write allocate
000	1	1	Yes	Normal	Write-back, no write allocate
001	0	0	Yes	Normal	Non-cacheable
001	1	1	Yes	Normal	Write-back, write and read allocate (Default cache policy on SAM Cortex-M7 MCUs when cache is enabled)

For detailed information, refer to the "section 4.6.6 MPU access permissions attributes", available on the ARM Information Center. For more details on configuring the MPU, refer to the TB3179 - How to Configure the Memory Protection Unit (MPU).

The following code example disables the MPU and then configures and enables a section of the SRAM memory region starting at 0x2045F0000 and of size 4096 bytes, as non-cacheable. Leaving the remaining SRAM memory region as cacheable with the default (WB-RWA) cache policy. The Access Permission (AP) is set to Full access which allows both privileged and non-privileged software to have RW access, and later it enables the MPU.

Code Showing MPU Configuration to Create a Non-Cacheable Memory Region

```
#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();
```

The linker script file may be modified to define a non-cacheable memory space, and place the DMA buffers to be linked to the non-cacheable memory area as shown in the following code sample for GNU linker script.

Linker Script Modifications to Create Memory Sections for Non-Cacheable Data

```
/* 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
```

```
    …….
}
```

The previous linker script example specifies the load memory address of the `.ram_nocache_data` section to be at the end of the `.text` and the `.relocate` sections.

Use the following code in the Reset Handler to zero the uninitialized variables defined under the `.ram_nocache` section, and to copy (from Flash to the SRAM) the initial values of initialized variables in `.ram_nocache_data`.

C Startup Code Modifications to Initialize the Memory Sections for Non-Cacheable Data

```
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;
    }
    ……
}
```

In the application, the DMA buffers can be allocated to the `.ram_nocache` memory region as shown in the following code sample. If the application has initialized variables in the no-cache memory region, then they must be defined to go under the `.ram_nocache_data` section.

Application Code to Define Buffers in Non-Cacheable Memory Section

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

Another way to avoid cache coherency is to use Tightly Coupled Memory (TCM) as the contents of TCM are not cached and can be accessed by both the CPU and the DMA. It can be accessed at similar speeds as accessing cache, without the penalty of a cache-miss and cache coherence issues.

Use case: Buffers with a size larger than the cache size (16 KB).

Advantages: No impact on performance. Transparent to the application (no cache maintenance required).

Drawbacks: Requires the linker script to be modified.

For more information on using TCM, refer to the links provided in the references section.

6. Relevant Resources

For additional information, refer to the following documents which are available for download from the following location:

1. ARM Cortex-M7 Processor Technical Reference Manual – L1 caches
2. ARM Cortex-M7 Processor Technical Reference Manual – Memory Protection Unit
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. How to Configure the Memory Protection Unit (MPU)
5. Atmel SMART SAM V7x TCM Memory
6. Atmel SMART SAM E70 TCM Memory

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-2966-1

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-7289-7561 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820