Renesas RA Family

Guidelines for Using the S Cache on the System Bus

Introduction

Caches can effectively improve instruction or data access speed for microcontroller and microprocessor systems with mismatch between CPU and slower SRAM. Even though there are no internal caches in the Cortex-M23 and Cortex-M33 processors, for some Renesas RA Family Cortex-M33 MCUs, there are system level caches for both instruction cache and data cache present, which help to improve instruction and data fetch speed.

The cache enabling and configuration for the instruction cache are handled by the Renesas Flexible Software Package (FSP). The cache enabling, disabling, and flushing functionality for the data cache are demonstrated in this application project with reference software projects provided. In addition, this application project provides guidelines and example code for keeping the data cache coherent. Use this application project as a reference resource for S Cache operations.

The data cache is named S Cache in the Renesas RA Family Cortex-M33 MCU Hardware User's Manual. The S Cache is on the MCU's system bus. The instruction cache is named C Cache and is on the code bus. This application note is focused on the data cache usage of the RA MCUs. For consistency, this application note uses **S Cache** throughout the rest of the application note. At the time of the release of this application project, the RA Family MCU groups that support the S Cache are RA6M5, RA6M4, RA6E1, and RA4M3. User can review the MCU Hardware User's Manual Buses section and look for the Cache section to understand whether the any new MCUs include S Cache and its general operations.

For other RA6 Series MCUs which do not have S Cache, they are provided with SRAMHS. Access to the SRAMHS is always no wait state. Use the SRAMHS on these MCUs when improved SRAM access is needed.

The example project provided is based on EK-RA6M5. You can easily port the example project to other MCUs which support S Cache. The performance improvement of using S Cache on an MCU varies based on the MCUs memory access speed, memory size, the nature of the SRAM access pattern of the application code, You need to analyze all these aspects when evaluating the S Cache.

Required Resources

Development tools and software

- The e^2 studio ISDE v2021-10 or greater
- Renesas Flexible Software Package (FSP) v3.5.0 or later
- SEGGER J-link® USB driver

The above three software components: the FSP, J-Link USB drivers and e^2 studio are bundled in a downloadable platform installer available on the FSP webpage at renesas.com/ra/fsp.

Hardware

- EK-RA6M5 Evaluation Kit for RA6M5 MCU Group (http://www.renesas.com/ra/ek-ra6m5)
- Workstation running Windows® 10
- One USB device cables (type-A male to micro-B male)

Prerequisites and Intended Audience

This application note assumes you have some experience with the Renesas e^2 studio IDE development. You must be familiar with importing, building, and debugging a Renesas RA Family MCU project based on FSP packages. In addition, users are required to read the entire Hardware User's Manual Caches section prior to proceeding to the rest of this application note:

The intended audience is product developers who wish to use the S Cache feature to improve the system performance.

Contents

1. Overview of the S Cache on the System Bus 3
1.1 S Cache Architecture 3
1.2 S Cache Specifications 4
1.3 Using the Memory Protection Unit with S Cache 5
1.4 S Cache Operation 6

2. Using S Cache in An Application 7
2.1 Using S Cache to Improve MCU Performance 7
2.2 Configuring the S Cache Registers on RA6M5 8
2.3 Design Considerations when Using S Cache 8
2.3.1 Evaluating CPU Performance when S Cache is Enabled 8
2.3.2 Allocating Memory Access 8
2.3.3 Design for Data Structure Grouping and Alignment 9
2.3.4 Considerations on S Cache Update Strategy 9
2.3.5 Keeping S Cache Coherent 9

3. Example Project 10
3.1 Overview 10
3.2 Import and Run the Example Project 10
3.3 Demonstration of How to Keep S Cache Coherent 12
3.4 Demonstration of MCU Performance Improvement 15

4. References 16

5. Website and Support 16

Revision History 17

1. Overview of the S Cache on the System Bus

A cache is a smaller, faster memory, located closer to a processor core than main memory. It stores copies of the data from frequently used main memory locations. Some RA Family Cortex-M33 MCUs implement both C Cache on the Code Bus and S Cache on the System Bus to reduce the average cost (time or energy) to access data from the main memory.

1.1 S Cache Architecture

Read the Buses > Overview section in the Renesas RA Family Cortex-M33 MCU Hardware User's Manual to understand the S Cache architecture. The bus system architecture for RA Family Cortex-M33 MCUs which have S Cache is shown in the following graphic.

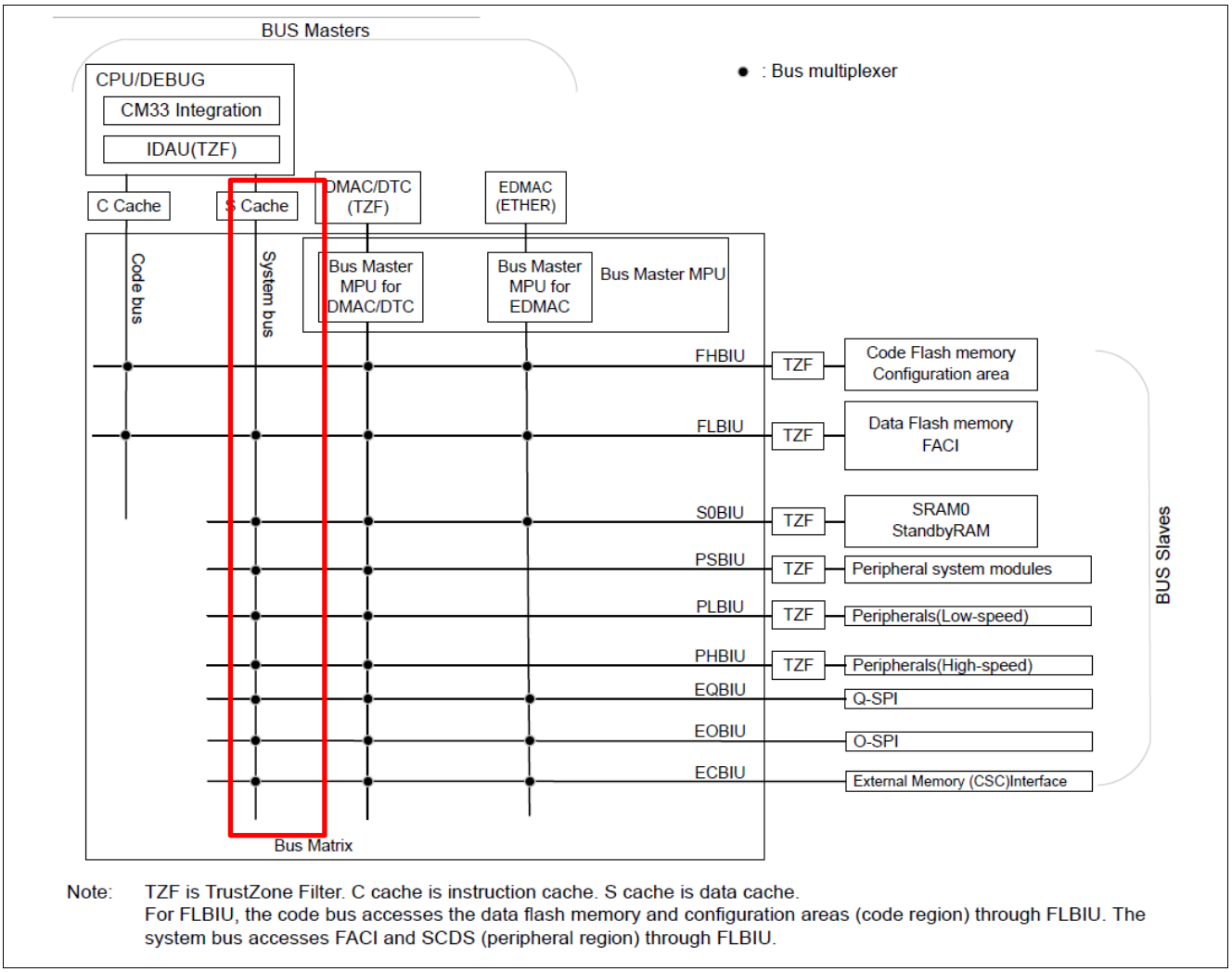


Figure 1. Bus Architecture for RA Cortex-M33 with S Cache

Table 1 is the bus master specification for RA6M4 and RA6M5. For arbitration between masters, the analysis in this application note is based on the following priority sequence:

EDMAC > DMAC/DTC > CPU

Table 1. Bus Specification for RA6M4 and RA6M5

Bus Master Name	Bus Interface Maximum Frequency	Synchronization	Specifications
Code bus	200 MHz	ICLK	Connected to the CPU Instruction Cache (C Cache) for instructions and operands
System bus	200 MHz	ICLK	Connected to the CPU Data Cache (S Cache) for data access operations
DMAC/DTC	200 MHz	ICLK	Connected to the DMAC/DTC
EMAC (Ether)	100 MHz	PCLKA	Connected to the EDMAC

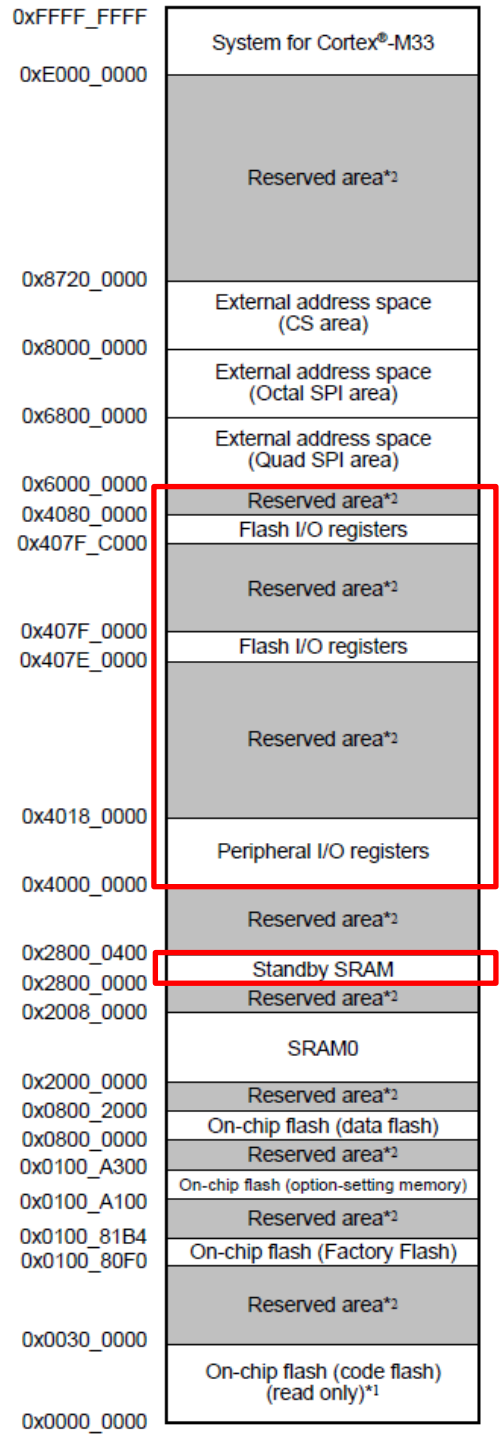
1.2 S Cache Specifications

Read the Buses > Caches > Overview section in the Renesas RA Family Cortex-M33 MCU Hardware User's Manual to understand the available configurations of the S Cache on RA6M4 and RA6M5 to understand the S Cache specifications. The following table has a summary of the key features:

Table 2. S Cache Specifications

Parameter	S Cache
Capacity	2 KB
Way	2-way set associative
Line size	32/64 bytes (defaults to 32 bytes)
Number of entries	32/16 entry/way (defaults to 32 entry per way)
Write way	Write through, non-write allocate
Replace way	2 way: LRU (Least recently used)
S Cache support area	0x20000000-0xDFFFFFFF except Standby SRAM area (0x2800_0000 to 0x2FFF_FFFF). Note: Peripheral area 0x4000_000 to 0x5FFF_FFFF must not have the cacheable attribution in the Arm® MPU.

Note that the Peripheral area 0x4000_000 to 0x5FFF_FFFF must not have the cacheable attribute in the Arm MPU as well as the Standby SRAM area. This is the default setting of the MCU. Use caution when updating the Memory Protection Unit (MPU) configurations to avoid accidentally making this section cacheable. Figure 2 shows these areas on the RA6M5 address space.



Note 1. See Table 4.1. The capacity of the flash differs depending on the product.
Note 2. Do not access reserved areas.

Figure 2. Memory Areas that Need to be Non-Cacheable

1.3 Using the Memory Protection Unit with S Cache

When S Cache is enabled, whether an SRAM region is cacheable depends on the MPU configuration. The MPU is programmable and the configuration of the MPU regions is managed by several memory mapped MPU registers. The MPU can be used to protect memory regions by defining access permissions.

Although the Cortex-M23 and Cortex-M33 processors do not have an internal level 1 cache, the cache attributes produced by the MPU settings are exported to the processor's top level. The RA Family MCU S Cache can utilize this feature to vary the cacheable setting for the SRAM regions. For example, for any algorithms where the variables need to be updated and flushed very frequently, using the MPU to configure these areas as non-cacheable may benefit the system.

For Cortex-M33 MCUs, by default, the MPU is disabled after reset. If the MPU is enabled, the default setting will enable all SRAM areas except the Standby SRAM area as cacheable. The default MPU setting will also set the Peripheral area 0x4000_000 to 0x5FFF_FFFF as non-cacheable. This setting is the default device memory attribute based on the Cortex-M33 architecture. The user should not change this configuration.

The user can set up the MPU to define additional memory regions as non-cacheable. When doing so, the user must consider using the memory barrier instructions before updating the MPU registers and before executing the memory regions affected by the new MPU configuration after the MPU configuration update. For an example of how to use the memory barrier instructions, refer to the application note from Arm on the use of memory barrier instructions for the Cortex®-M processor called *Arm Cortex-M Programming Guide to Memory Barrier instructions (ArmDAI0321A)*.

1.4 S Cache Operation

Read the Buses > Caches > Operation section in the Hardware User's Manual to understand the access flow from CPU to S cache. Once the S cache is enabled, access to the cacheable area follows the access flow as shown in Figure 3.

The S cache function works when it is enabled, and cacheable access is performed from the CPU. When an SRAM access to the cacheable area is initiated, the cache first checks the address of CPU access request and compares the address with the entries in the cache tag. Then based on this, the CPU determines whether the CPU access is a hit or a miss.

If the access is a read, the system behavior varies according to the following rules:

- For a read hit, the cache reads required data from the cache data and returns it to CPU. In a cache read hit, there is a 0 bus wait cycle.
- For a read miss, the cache reads one cache line data from memory and stores it into the cache data. The cache then returns the required data. In cache read miss, the number of bus cycles used is same as when cache is disabled.

If the access is a write, the system behavior varies according to the following rules:

- For a write hit, the cache processes a write cycle to cache data and a write cycle to memory.
- For a write miss, the cache processes a write cycle to memory. There is no impact on cache data.

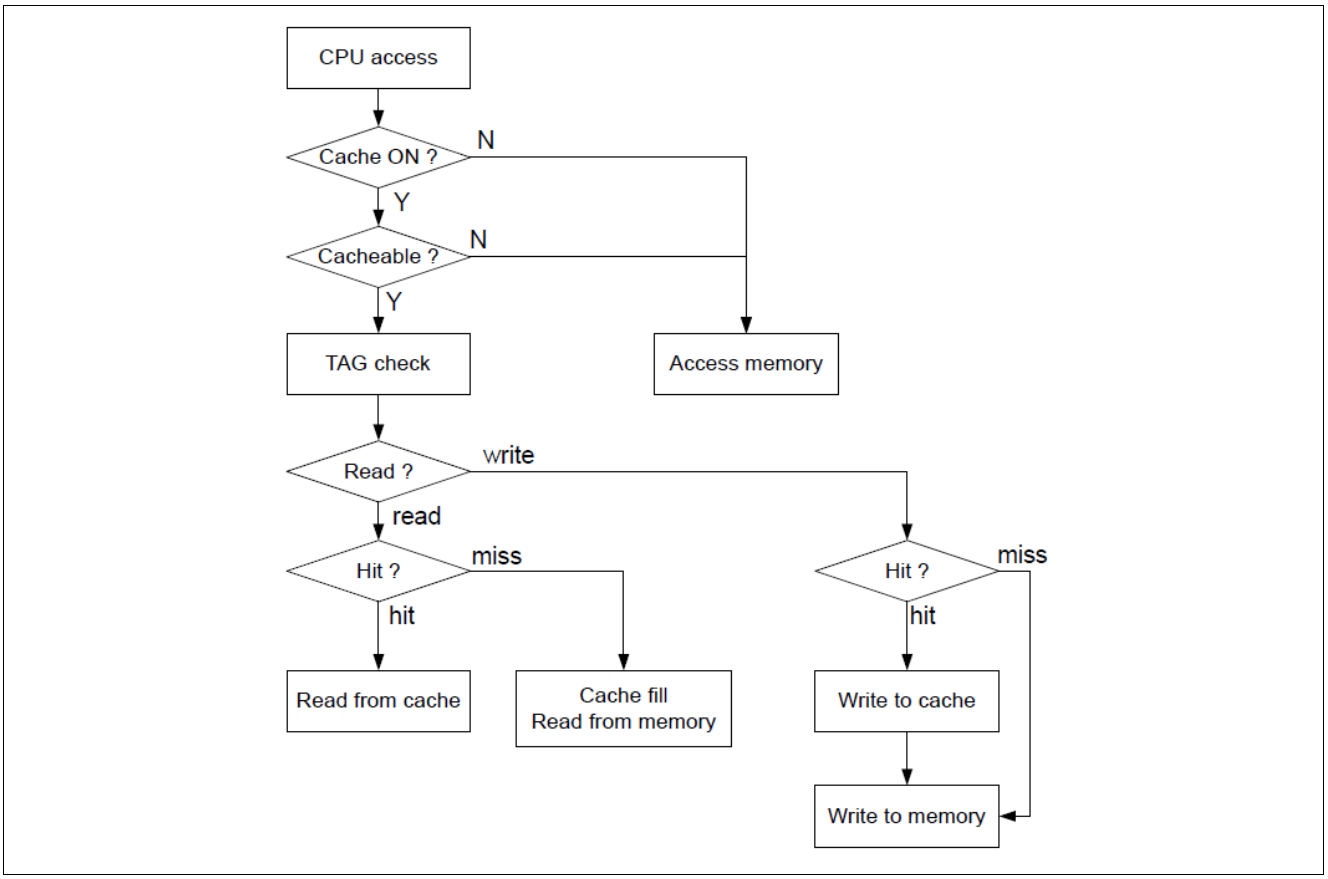


Figure 3. Access flow from CPU to S cache

2. Using S Cache in An Application

Consider using S cache for improved MCU performance based on the analysis in this section. Guidelines on when to use S cache in an application, usage notes for using S cache and how to keep S cache coherent are addressed in this section.

2.1 Using S Cache to Improve MCU Performance

RA6M5 and RA6M4 have a maximum system clock of 200 MHz. Access to SRAM is a slower process compared to the CPU speed. The analysis of using S cache on RA MCU assumes the Error Correction Code on the SRAM is disabled (which is the default setting from the MCU and FSP point of view). Under this condition, the read access to S cache is 1 cycle with cache hit and access to SRAM is 4 cycles (with 1 wait state) when the system bus is operating at over 100 MHz. This application project demonstrates the MCU performance improvement when the CPU is operating at 200 MHz. When operating at 100 MHz or less, read access to S cache is 1 cycle with cache hit and access to SRAM is 3 cycles (with 0 wait state).

For RA4M3, the maximum system clock is 100 MHz. Accessing SRAM is always 0 wait states. Read access to S cache is 1 cycle with cache hit and read access to SRAM is 3 cycles (with 0 wait state). For this reason, enable S cache if improved system performance is desired.

For RA MCUs with S cache support, consider enabling S cache to boost system performance when the data processed by the CPU exhibits a significant spatial locality, like in the case of a working buffer which does not need to be updated frequently.

Note that when S cache is enabled, and the above condition is met, the more frequently the data in S cache is used without needing an update, the larger the benefit of using S cache.

For a cache miss, the bus access cycle is same as when the cache is disabled for all MCUs which support S cache. And there is no performance improvement from cache write operations. For data not frequently used, filling the cache is an initial operation that will not be repeated or is repeated with very low frequency.

2.2 Configuring the S Cache Registers on RA6M5

The following table summarizes the S cache registers, their functionality and the application functions used in this application project to configure these registers. Refer to the included example projects to look at the detailed definitions for these functions.

Table 3. S Cache Register Configuration Demonstrated in the Application Project

Registers	Functionality	API created in application code
SCACTL: S Cache Control Register	Enable and disable S cache	`void enable_s_cache(void);` `void disable_s_cache(void);`
SCAFCT: S Cache Flush Control Register	Flush or do not flush the S cache	`void flush_s_cache(void);`
SCALCF: S Cache Line Configuration Register	Configuration register that configures the S cache line size to 32 or 64 (default is 32)	`void select_s_cache_line_size(bool line_size_32);`

For other S Cache related registers, the application project uses the default setting after MCU reset. Table 4 is a summary of these registers and their default settings used in the application project.

Table 4. Registers Configured at Default MCU Reset State

Registers	Functionality	Default Settings used in the Application Project
CSAR: Cache Security Attribution Register	This register defines the security attributes of registers for Cache Control, Line Configuration, and Cache Error.	This register is write-protected by the PRCR register. The default setting is used in the application project. Both secure and non-secure projects can use these attributes.
CAPOAD: Cache Parity Error Operation After Detection Register	This register defines the action the MCU will take when a Cache Parity Error is detected. The options are Non-Maskable Interrupt or Reset.	The default setting is Non-Maskable Interrupt. This setting is used in the application project. Demonstrations on the handling of the NMI interrupt are out of the scope of this application project.

2.3 Design Considerations when Using S Cache

This section provides more background on looking at the CPU performance when using S cache. Some tips to maximize the MCU bus performance are discussed. And finally, guidelines on how to design the software to benefit from the S Cache update scheme are provided.

2.3.1 Evaluating CPU Performance when S Cache is Enabled

As explained in section 2.1, enabling S cache can improve system performance for some applications. The analysis in section 2.1 focuses on the time saving from the bus cycle access point of view. Aside from the bus access, instruction cycles are also a factor which influences the system performance. Therefore, the perceived system performance improvement will not be proportional to the bus cycle savings.

The analysis of the system performance improvement based on the example project provided in this application project is provided in later sections.

2.3.2 Allocating Memory Access

Several guidelines for memory allocations should be considered when designing the software for the purpose of improved performance, for example, when S Cache is enabled on RA MCUs.

- Variables often accessed together should be close to one-another in memory. This increases the likelihood that the other variable will already be in the cache after the processor has accessed the first variable, thus avoiding cache misses.
- When accessing data linearly, use vectors or arrays. Linked lists, hash maps, dictionaries and so forth are great data structures for many things, but they are not cache friendly. Iterating through such a data structure involves many cache misses. If performance is important, stick to arrays. In addition, use

arrays of values instead of arrays of pointers. Accessing the variable using a pointer invariably involves a cache miss. So, for fast array access, dispense with the pointers and go with values.

2.3.3 Design for Data Structure Grouping and Alignment

When looking at how a program accesses memory, design decisions can be made that will take the most advantage of cache. If a data set that a program is working on is smaller than the cache line size of the processor, it is important to make sure that the data is read into one cache line. This is done by grouping the data together in a structure and aligning that structure, so it stays in a cache line.

For example, suppose a function uses local variables i and j as subscripts into a 2-dimensional array, they might be declared as follows:

```
int i, j;
```

These variables are commonly used together, but they can fall in different cache lines, which could be detrimental to performance. If the variables are used in a part of the program that is performance-critical, we could instead declare them as follows:

```
struct { int i, j; } sub;
```

This relies on the compiler's default alignment for structures. This default alignment is typically enough to ensure that the structure would be aligned in cache such that both indexes would be in the same cache line. `i` and `j` must now be referred to as `sub.i` and `sub.j`.

The alignment of the structure can be specified if the compiler supports this feature. Here is an example using the attribute feature of GCC to align a structure on an 8-byte boundary:

```
struct { int i, j; } sub __attribute__ ((aligned (8)));
```

2.3.4 Considerations on S Cache Update Strategy

The RA MCUs use the Least Recently Used (LRU) policy as the cache replace method. With the Cache Write-through, no-write allocate policy, the cache is filled upon read miss as shown in Figure 3.

To benefit from the LRU policy, design the system with the following points in mind to avoid cache replace events whenever possible:

- Use the data while still in cache. Consider that data usage and if possible, load data from the memory to the cache just once, use them or do some modifications on them, and then return it back to the operating memory. If we need to store the same data from SRAM to cache, we are not using the cache optimally.
- Reduce the number of times data which is already saved to the cache is written to memory if these variables are updated. For example, in a sorting algorithm, we can reduce the instances of writing the original array by employing some intermediate variables.

2.3.5 Keeping S Cache Coherent

Cache coherency needs to be considered when the cacheable region is accessed by both the CPU and other bus masters (such as DTC, DMAC). For shared memory between MCU and other bus masters (DTC and DMAC), S cache needs to be flushed prior to CPU access. Otherwise it might use stale data from the S cache since other bus masters might have updated the SRAM. This can be achieved by one of two ways; the first method is the preferred method for most applications:

- Flush S Cache in the application code
  Since software developers know which regions are common for CPU and other masters and they know when the CPU or another master writes to the command regions, the software developer can decide on what regions are cacheable by setting up the MPU. Next, for the cacheable regions, the software developer can flush the S cache prior to the other bus master’s access to the common region. This method is demonstrated in this application project.
- Flush S cache at the end of bus master transfer
  This method may incur more overhead when frequent transfers are needed. This method is demonstrated in this application project.

3. Example Project

3.1 Overview

This example project demonstrates how to enable and disable S Cache, how to handle S Cache coherency and how to use the cycle counter on the debug unit Data Watchpoint and Trace Unit (DWT) to evaluate the CPU performance improvement when S Cache is enabled.

System setup:

- A sine and cosine data set are stored in code flash.
- The data set is then transferred to the SRAM via a DMA channel.
- Next, the standard deviation of $sine^2$+$consine^2$ are calculated by reading the sine cosine data from the buffer in the SRAM.

The FSP modules used in this example project include r_dma, r_agt, and Arm® CMSIS DSP library. Their functionalities are explained briefly as follows:

- r_dmac: transfer data to DAC register to generate the sine and cosine wave
- r_agt: time the DMA transfer of the DAC data
- Arm CMSIS DSP module: calculate the standard deviation of ($sine^2$ + $cosine^2$)

In addition, the cycle counter on the debug unit Data Watchpoint and Trace Unit (DWT) is used to track CPU cycles used in a fixed set of calculations when S Cache is disabled or enabled.

Analysis of S Cache Usage:

- The deviation of ($sine^2$ + $cosine^2$) will be larger if S Cache coherency is broken. See section 3.3 for this analysis.
- $Sine^2$ + $cosine^2$ calculation should be faster when S Cache is enabled. See section 3.4 for this analysis.
- This application project provides routines to update S Cache line size. But it does not demonstrate the line size configuration to CPU performance. For set associative cache, line size primarily influences the cache miss time penalty. Larger line size means larger penalty in time when a cache miss happens because it takes longer to bring the line in to the cache.
  To show the set associative cache line size influence on the CPU performance, frequent S Cache misses need to be simulated. This is not demonstrated in this example project because there is no frequent S Cache miss designed in the performance analysis routine. On the other hand, for a cache of constant size, using larger line size increases spatial locality which can be helpful for some applications. User should analyze the application at hand to select the line size that supports the best performance of the system. This is typically achieved through empirical investigation. Once the line size is determined for a system, it should not be randomly changed unless a new analysis is performed.

3.2 Import and Run the Example Project

Import project `using_s_cache_ra6m5` into an e^2 studio workspace. Click **Generate Project Content** and compile the example project. Next, connect **J10 USB Debug** port on EK-RA6M5 to the development PC. Right click on the project `using_s_cache_ra6m5` and select **Debug As > Renesas GDB Hardware Debugging**.

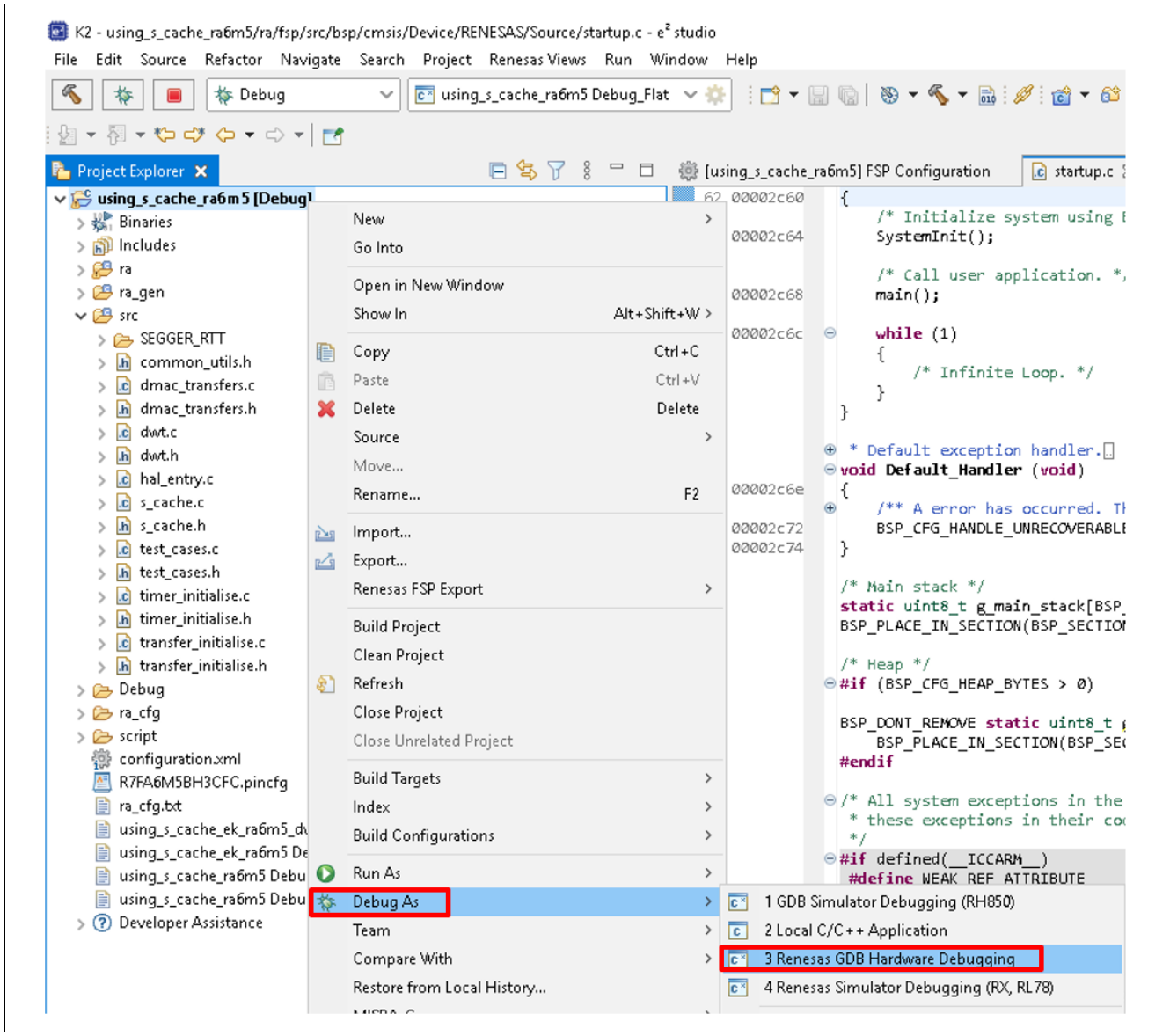


Figure 4. Using S Cache Example Project

Connect to RTT viewer.

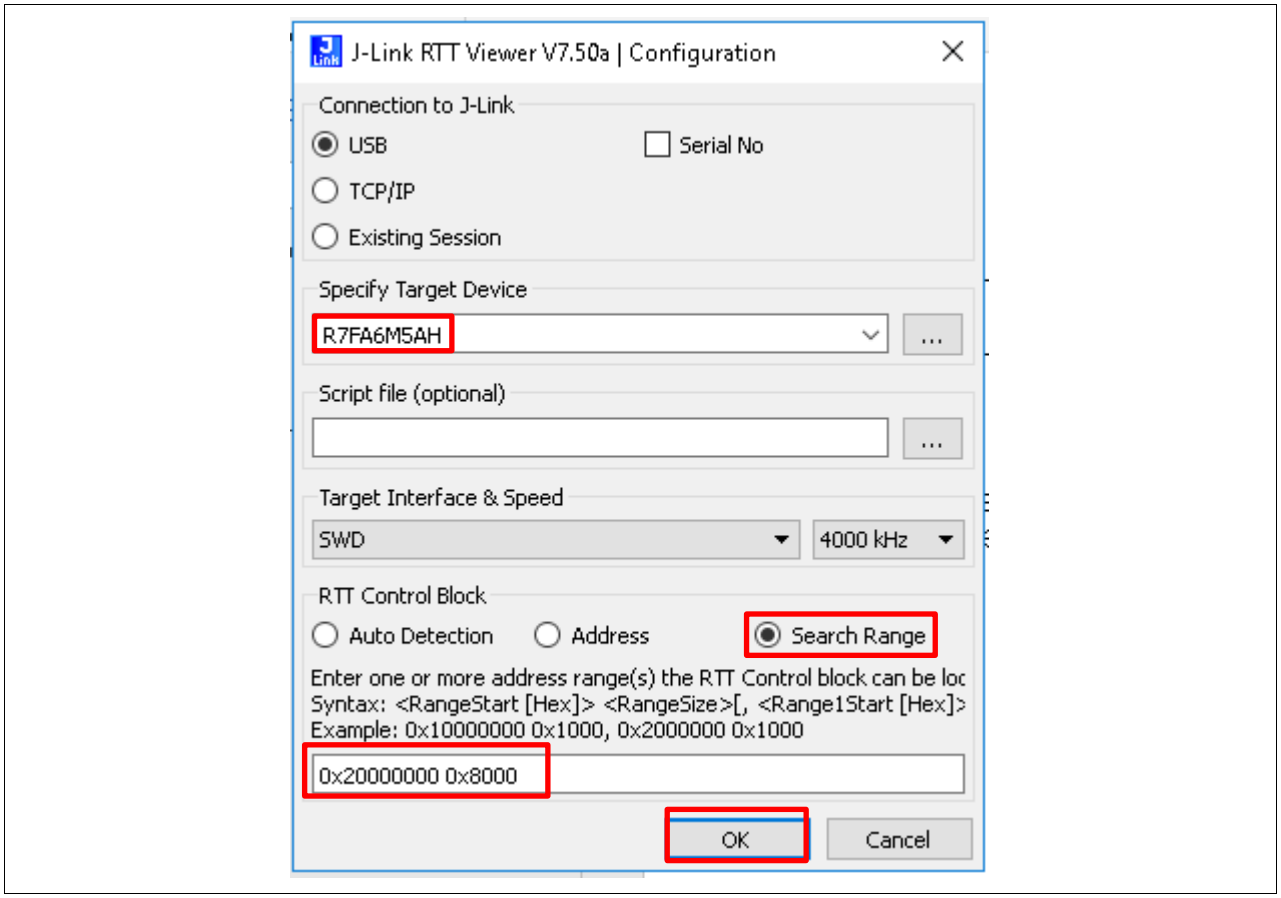


Figure 5. Connect to SEGGER RTT Viewer

The actions a user can take through the RTT user interface are: S Cache configuration, whether to flush S Cache, where to Flush S Cache, as well as the S Cache line configuration.

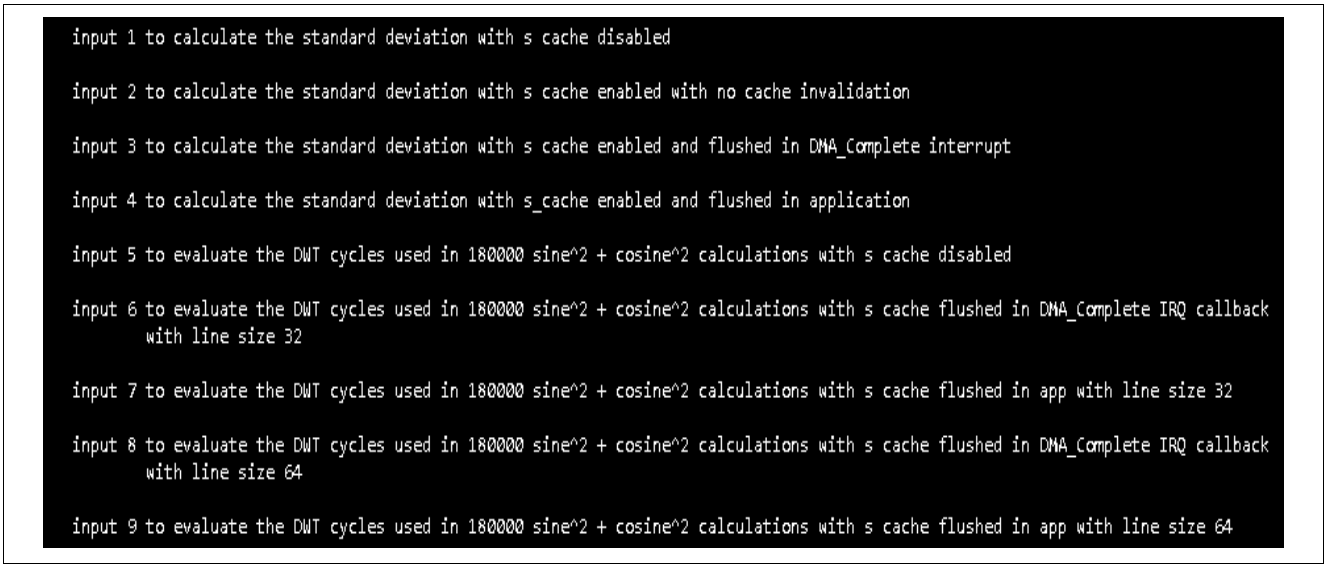


Figure 6. Actions Users Can Perform via RTT User Menu

3.3 Demonstration of How to Keep S Cache Coherent

When the S Cache is enabled and filled, the calculation uses the data from S Cache, which can be different from the data transferred to the SRAM via the DMA transfer. This example project demonstrated that when S Cache is disabled, the standard deviation of ($sine^2 + cosine^2$) is 0 as expected.

When S Cache is enabled, the S Cache is corrupted after DMA transfers data to SRAM. When (sine^2 - cosine^2) is calculated, the corrupted S Cache is used and hence generates larger standard deviation.

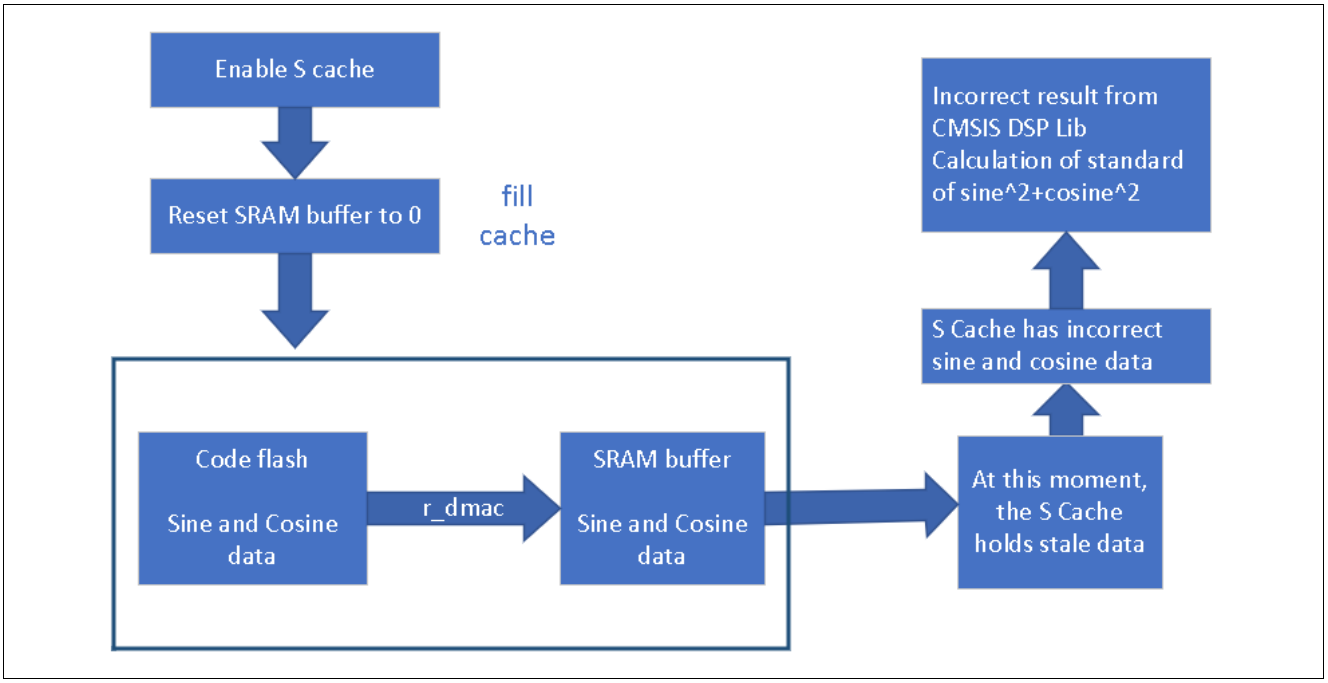


Figure 7. S Cache Coherency is Broken due to DMA Transfer to Common Area

When the S Cache is flushed in a DMA transfer complete interrupt callback and in the user application prior to the calculation of (sine^2 + cosine^2), S Cache coherency is restored.

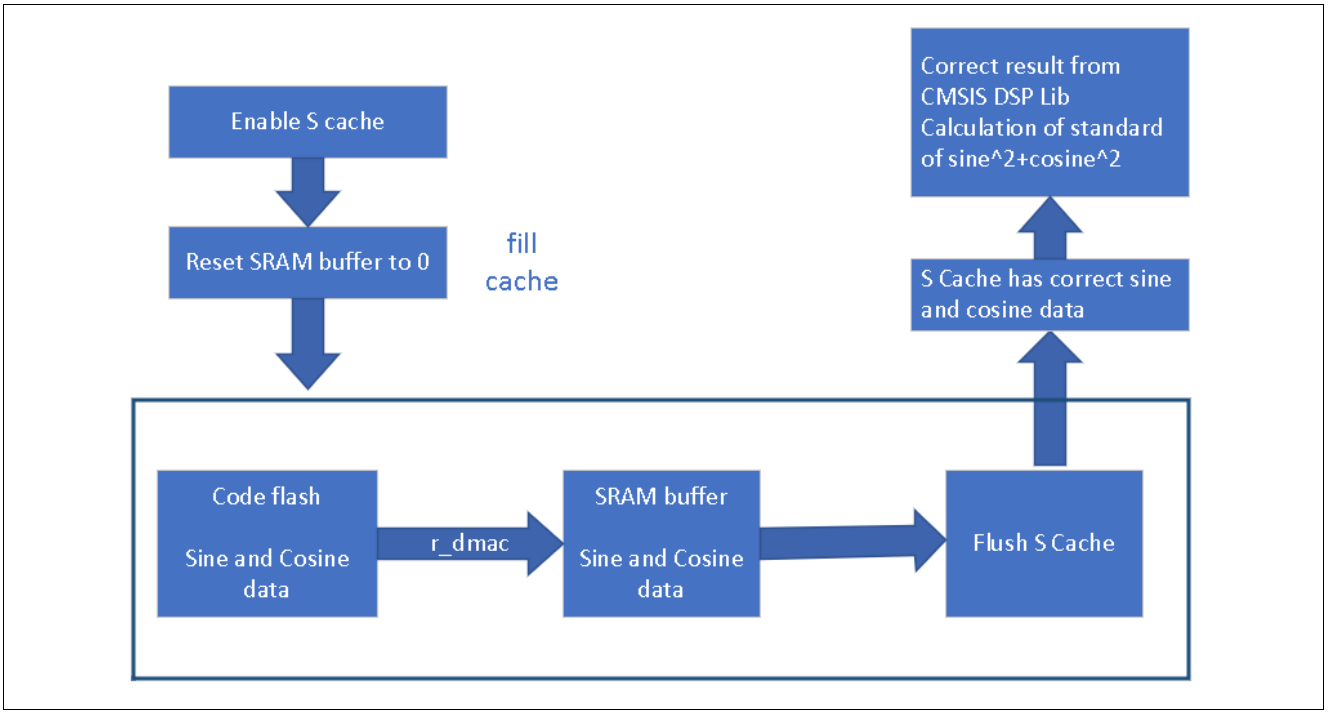


Figure 8. S Cache Coherency is Restored – Flush S Cache in Application Code

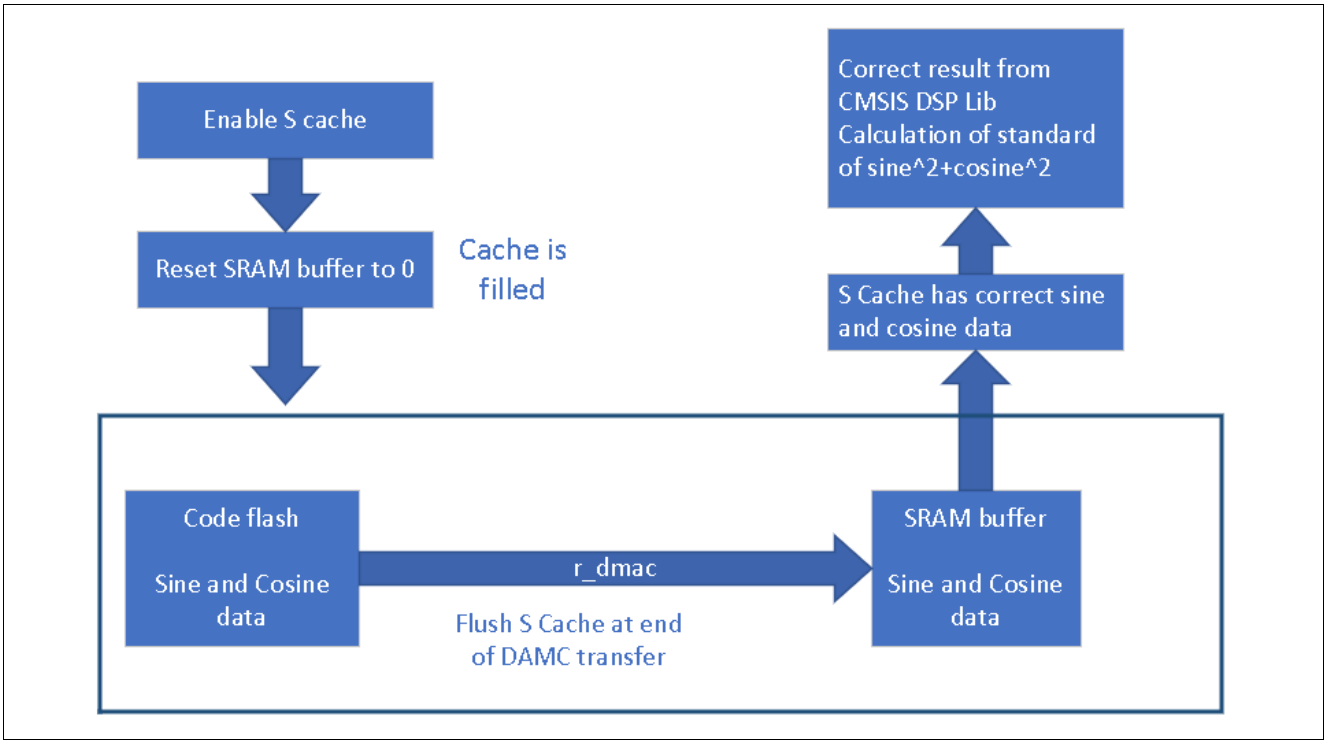


Figure 9. S Cache Coherency is Restored – Flush S Cache in DMA Transfer Complete Callback

Figure 10 is an example run of the S Cache coherency handling routines provided in this application project.

```
  < 1
00>
00> DMAC dma_transfer_sine_cosine_operation in progress.
00>
00> DMAC dma_transfer_sine_cosine_operation transfer completed.
00>
00> Standard deviation when s cache is disabled is 0
00>
  < 2
00>
00> DMAC dma_transfer_sine_cosine_operation in progress.
00>
00> DMAC dma_transfer_sine_cosine_operation transfer completed.
00>
00> Standard deviation when s cache is enabled but not flushed is 2879112
00>
  < 3
00>
00> DMAC dma_transfer_sine_cosine_operation in progress.
00>
00> DMAC dma_transfer_sine_cosine_operation transfer completed.
00>
00> Standard deviation when s cache is enabled and flushed in dma transfer complete callback is 0
00>
  < 4
00>
00> DMAC dma_transfer_sine_cosine_operation in progress.
00>
00> DMAC dma_transfer_sine_cosine_operation transfer completed.
00>
00> Standard deviation when s cache is enabled and flushed in app is 0
00>
```

Figure 10. Demonstration of How to Keep S Cache Coherent

Table 5. Standard Deviation of Sine² + Cosine²

S Cache Configuration	Standard Deviation
Disabled	0
Enabled but S Cache not Flushed after DMA Transfer	Around 2879112
Enabled and S Cache Flushed in DMA Complete Transfer	0
Enabled and S Cache Flush in Application Code	0

3.4 Demonstration of MCU Performance Improvement

In this example project, 1000 cycles of (180 sine² + cosine²) calculations are performed. The number of DWT cycles used for this calculation is captured and displayed on the RTT Viewer.

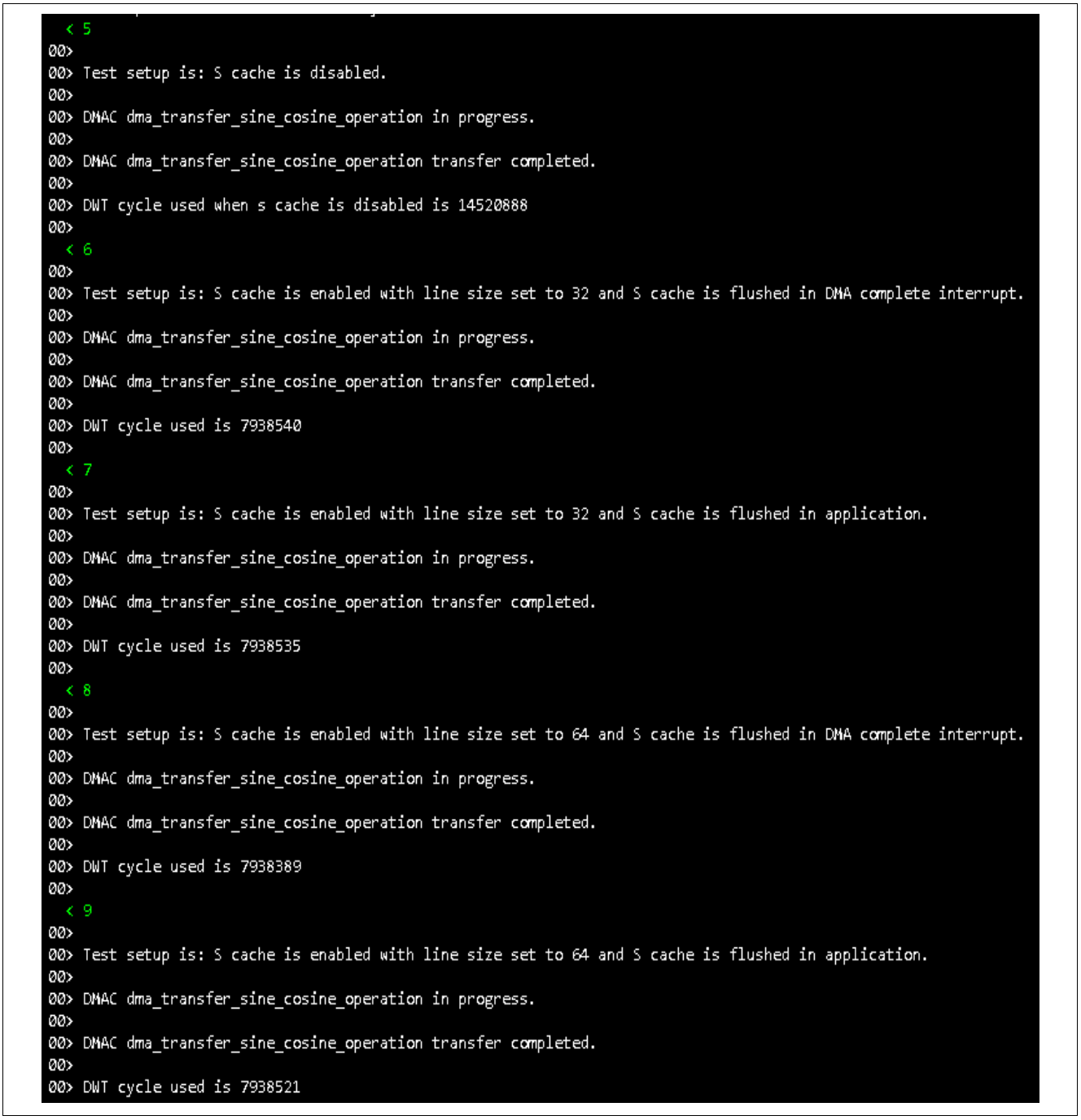


Figure 11. Demonstration of CPU Performance Improvement when S Cache is Enabled

From the output presented in the above example, the CPU performance improvement is about 40%. This presented CPU performance increase depends on savings from bus access as well as instruction cycle access.

As explained in the overview section 3.1, this example project does not demonstrate the line size influence on the CPU performance. The number of DWT cycle counter stays about the same for 32-byte or 64-byte line size configuration.

Also, notice that the CPU performance stays about the same when using the two different flushing methods, whether flushing at the end of the DMA transfer or in the application. This is because the number of flushing operations is only once for both methods. To reduce S Cache flushing influence on CPU performance, when to flush the S Cache needs to be carefully considered. If frequent S Cache flushing is inserted synchronously to the flow of the application, the performance of the system might be negatively influenced under extreme conditions.

4. References

RA6M5 Group User's Manual: Hardware: https://www.renesas.com/document/man/ra6m5-group-users-manual-hardware?language=en&r=1493931

5. Website and Support

Visit the following URLs to learn about the RA family of microcontrollers, download tools and documentation, and get support.

EK-RA2E1 Resources	renesas.com/ra/ek-ra6m5
RA Product Information	renesas.com/ra
Flexible Software Package (FSP)	renesas.com/ra/fsp
RA Product Support Forum	renesas.com/ra/forum
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.0.0	Jan.06. 22	-	First release document

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.