Renesas RA8 Series

Getting Started with RA8 Dual-Core MCU Memory Architecture, Configurations and Topologies

Introduction

This application note is designed to provide guidance on effectively utilizing the memory resources of the RA8 dual-core family of devices, with a focus on optimizing performance for dual-core applications.
As MCUs advance in complexity, on-chip memory may not always be sufficient to support demanding requirements, making the use of off-chip memory essential for fully realizing the potential of advanced dual-core applications. The RA8 dual-core MCU offers extensive on-chip memory along with support for off-chip memory interfaces, ensuring efficient memory management in dual-core environments.

This application note covers:

- Overview of internal MRAM, SRAM, and Cache architectures in the RA8 dual-core MCU, for dual-core operation.
- Overview of External memory, such as OSPI flash/RAM, SDRAM, and a chip-selectable external interface for additional memories, and guidelines on interfacing with OSPI and SDRAM/memory bus to efficiently manage memory in a dual-core system.
- Instructions on connecting external memory devices such as OSPI, SDRAM, and chip-select addressable memory space for seamless dual-core execution.
- Steps to enable and allocate TCM for optimized execution on each core.
- Explanation of ECC and parity memory on RA8 dual-core devices, including configuration for dual-core applications.
- Essential steps to maximize memory efficiency, ensuring smooth and high-performance execution in dual-core RA8-based designs.

Required Resources

- Flexible Software Package (FSP) v6.0.0 or later
  Note: FSP support for different boards is listed below.
  - For RA8P1(FSP 6.0.0 or later is required).
  - For RA8T2(FSP 6.1.0 or later is required).
  - For RA8D2 and RA8M2 (FSP 6.2.0 or later is required).

- Renesas e² studio - Version: 2025-04.1 or later is required.

Supported MCU Groups

At the time of the release, the supported MCU groups are:

- RA8P1 Group
- RA8T2 Group
- RA8D2 Group
- RA8M2 Group

Note: The App Note is developed for the RA8P1 dual-core MCU as a reference. However, since the RA8T2, RA8D2, and RA8M2 MCUs share the same architecture, these resources can also serve as valuable references for Memory Architecture, Configurations, and Topologies for other dual-core MCUs.

Contents

1. Dual-Core MCU Architecture

1.1 Overview of Dual-Core Architecture

Dual-core MCUs integrate two processing cores into a single chip to enhance performance, enable parallel processing, and improve power efficiency. These MCUs are commonly used in real-time applications, IoT devices, industrial automation, and automotive control systems. The architecture can be categorized into:

Homogeneous Dual-Core MCUs: Both cores are identical, often executing similar tasks in parallel or load balancing computational processes.

Heterogeneous Dual-Core MCUs: Each core has a different architecture or instruction set, enabling distinct tasks: one core handles real-time operations while the other handles general-purpose computations.

Renesas RA8 dual-core MCU architecture falls into the heterogeneous dual-core MCU category, with Cortex-M85 (CM85) designated as Core 0/CPU0, serving as the high-performance processing unit, while Cortex-M33 (CM33) is defined as Core 1/CPU1, handling additional real-time control and system management tasks.

1.2 RA Dual-Core Bus Architecture

The bus architecture in RA dual-core MCUs is engineered to support efficient communication and data exchange between the cores and peripherals. These devices primarily use a shared bus architecture, in which both cores share a common system bus that connects to memory and peripheral components. To maintain smooth operation, arbitration mechanisms are employed to prevent conflicts and ensure fair access to shared resources. In addition, a dedicated bus is provided for accessing ITCM and DTCM, further enhancing performance and reducing contention.

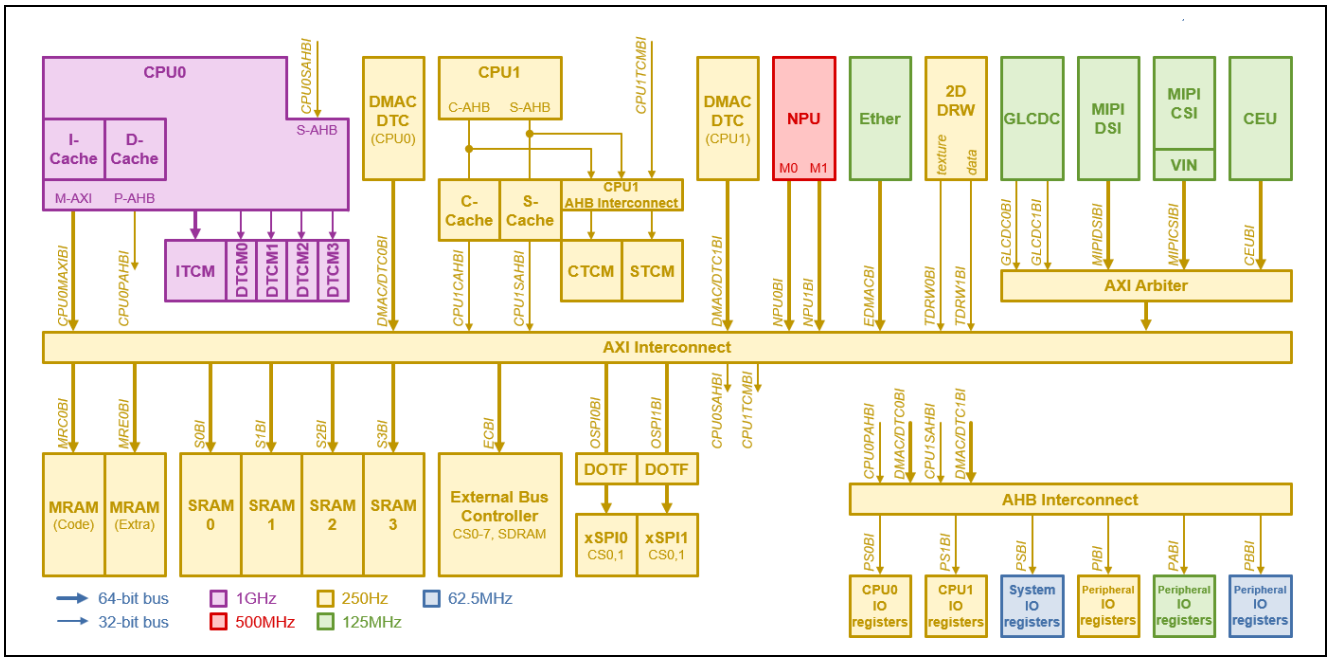


Figure 1. RA8 Dual-Core Bus Architecture Overview

Key components of the RA8 dual-core bus architecture include System Bus AXI, AHB, and APB, which facilitate communication between CM85, CM33, memory, and peripherals.

These buses collectively form an interconnected matrix that enables efficient routing of data and control signals across the RA8 dual-core system. Both the CM85 and CM33 cores, which can act as independent bus masters, communicate seamlessly with various on-chip memory blocks such as ITCM, DTCM, SRAM, and MRAM. In addition, the architecture provides smooth peripheral access via the APB and connects to external interfaces via AXI or AHB expansions. By intelligently distributing traffic across buses with different performance characteristics and access priorities, the RA8 ensures that real-time tasks, background processing, and peripheral operations can run concurrently without contention, essential for high-

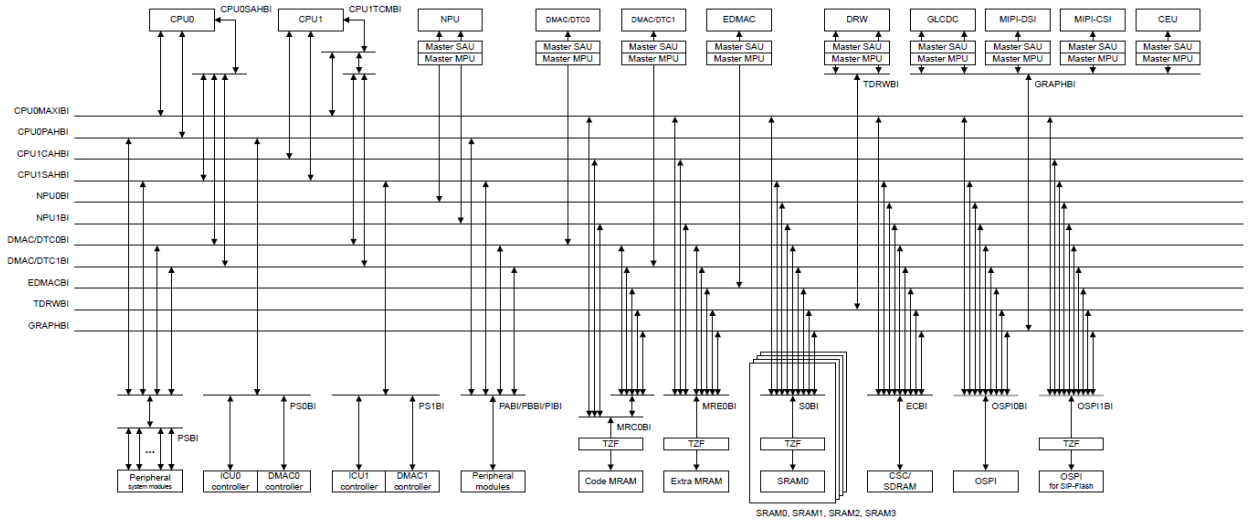
performance dual-core applications requiring parallelism and synchronized inter-core interaction.

1.3 System Bus and Data Path

The system consists of multiple bus interfaces that connect processing units and peripherals. The CPU0, based on the Arm® Cortex®-M85, connects to the Master-AXI (M-AXI) and Peripheral AHB (P-AHB) bus interfaces, while the CPU1, featuring the Arm® Cortex®-M33, connects to the C-AHB and S-AHB interfaces. There are Neural Processing Units (NPUs) connected to the interfaces NPU0 and NPU1 bus interfaces. The Direct Memory Access bus (DMAC/DTC1BI/EDMACBI) includes interfaces for DMAC,DTC0, DTC1, and EDMAC. The system also includes a dedicated bus (GRAPHBI/TDRWBI) for image and display processing via the Camera Engine Unit (CEU), Graphics LCD Controller (GLCDC0 and GLCDC1), and Drawing Engine (DRW0 and DRW1). For multimedia applications, the MIPI-DSI interface is used for display, while the MIPI-CSI interface supports video input via VIN.

It includes connections to various memory units, such as Code MRAM and Extra MRAM (read-only), as well as multiple SRAM modules (SRAM0, SRAM1, SRAM2, and SRAM3) for high-speed data storage. Each CPU0 and CPU1 has a dedicated system AHB connection (CPU0 S-AHB and CPU1 S-AHB) to ensure smooth data flow. External memory interfaces include OSPI connections for SPI-Flash, SDRAM, and CSC for general external device support.

Peripheral synchronization is achieved through modules linked to different clock domains, including PCLKA, PCLKB, and ICLK, ensuring coordinated operation. The ICU controllers (ICU0 and ICU1) are integrated with CPU0 and CPU1 to manage efficient interrupt handling. Additionally, the peripheral system module manages multiple controllers, including cache, memory protection, debug, security, and bus control, ensuring robust system operation. Figure 2 shows the RA8 Dual-Core system Bus connection.



Note: "TZF" is Slave TrustZone Filter.

Figure 2. RA8 Dual-Core System Bus Connection

Figure 3 shows the RA8 dual-core System Bus Access mapping table for Bus Master and Slave.

Table 15.2 System bus access path

		Master													
	Name	CPU0MAXIBI	CPU0PAHBI	CPU1CAHBI	CPU1SAHBI	NPU0BI	NPU1BI (R-only)	DMAC/DTC0BI	DMAC/DTC1BI	EDMACBI	GLCDC0/1BI(R-only)	TDRW0/1BI	MIPIDSIBI	MIPICSIBI (W-only)	CEUBI (W-only)
Slave	MRC0BI (Read)	T	F	T	F	F	T	T	T	T	T	T	T	F	F
	MRC0BI (Write)	T	F	T	F	F	F	T	T	F	F	F	F	F	F
	MRE0BI (Read)	T	F	T	F	F	T	T	T	T	T	T	T	F	F
	CPU0SAHBI	F	F	F	T	F	F	T	T	F	F	F	F	F	F
	CPU1TCMBI	T	F	F	F	F	F	T	T	F	F	F	F	F	F
	S0BI	T	F	F	T	T	T	T	T	T	T	T	T	T	T
	S1BI	T	F	F	T	T	T	T	T	T	T	T	T	T	T
	S2BI	T	F	F	T	T	T	T	T	T	T	T	T	T	T
	S3BI	T	F	F	T	T	T	T	T	T	T	T	T	T	T
	ECBI	T	F	F	T	T	T	T	T	T	T	T	T	T	T
	OSPI0BI	T	F	F	T	T	T	T	T	T	T	T	T	T	T
	OSPI1BI	T	F	T	T	T	T	T	T	T	T	T	T	T	T
	PBBI	F	T	F	T	F	F	T	T	F	F	F	F	F	F
	PABI	F	T	F	T	F	F	T	T	F	F	F	F	F	F
	PIBI	F	T	F	T	F	F	T	T	F	F	F	F	F	F
	PS0BI	F	T	F	F	F	F	F	F	F	F	F	F	F	F
	PS1BI	F	T	F	T	F	F	F	F	F	F	F	F	F	F
	PSBI	F	T	F	T	F	F	T	T		F	F	F	F	F

Figure 3. RA8 Dual-Core System Bus Access Path

1.4 Core Communication and Synchronization

Core communication and synchronization are essential for efficient interaction between the CM85 and CM33 cores within the CPUs. Inter-Processor Communication (IPC) enables hardware sharing, data exchange, and coordination through interrupt events, ensuring seamless operation between the two cores.

To maintain proper synchronization and prevent resource conflicts, IPC employs multiple locks and semaphore mechanisms. CM85 and CM33 share memory regions, with mutual exclusion mechanisms to prevent race conditions.

Developers can choose between separate or shared hardware semaphores based on application requirements, ensuring mutually exclusive access to shared resources between CM85 and CM33.

Also, for efficient data transfer, IPC integrates FIFO buffers:

- A write-only FIFO for CM85 and a read-only FIFO for CM33, allowing CM85 to send data to CM33.
- A write-only FIFO for CM33 and a read-only FIFO for CM85, enabling bidirectional communication.

These mechanisms ensure reliable, structured data exchange and shared-memory access between the two cores, enabling seamless synchronization and coordination.

2. Memory Architecture of RA Dual-Core MCU

The RA8 dual-core MCU has an address space of 2^32 bits, providing 4 GB of addressable space ranging from 0x0000_0000 to 0XFFFF_FFFF. It also supports a variety of memories, both internal and external to the MCU.

The different types of internal and external memory are shown in Table 1 below.

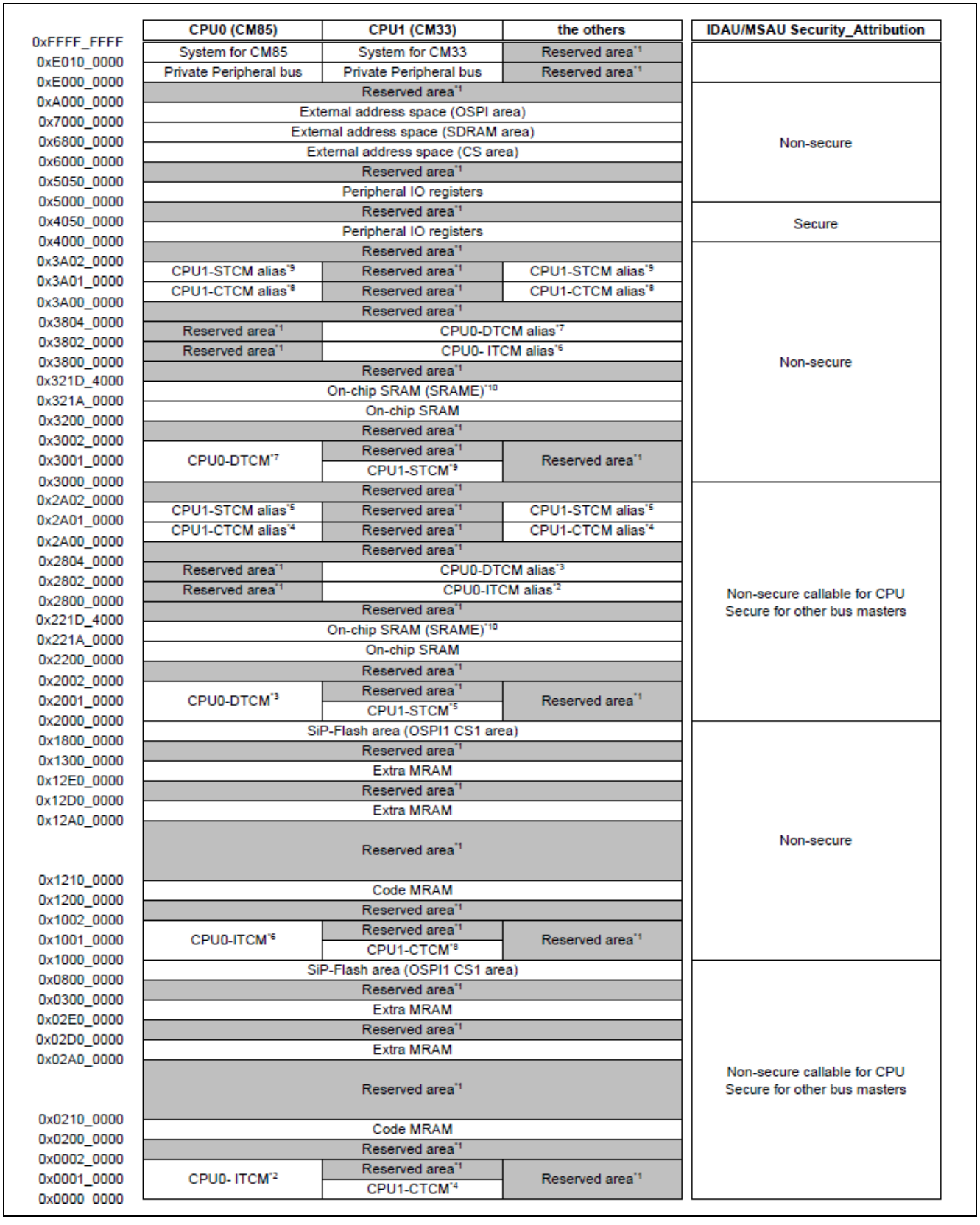
Table 1. RA8 Dual-Core Memory Example

Internal Memory	
Code memory	Up to a Maximum of 1 MB of MRAM memory(Non-Volatile).
SRAM	2 MB (256 KB CM85 TCM RAM, 128 KB CM33 TCM RAM, and 1664 KB of users SRAM) of on-chip high-speed SRAM with Error Correction Code (ECC). Note: In some of the RA8 dual-core Part numbers, the SRAM is limited to 1792 KB
ITCM and DTCM	128 KB (16 blocks × 8KB) with ECC for Tightly coupled Instruction Memory, 128 KB (16 blocks × 8KB) with ECC for Tightly coupled Data Memory for CM85. 64 KB with ECC for Tightly Coupled Instruction memory and 64 KB with ECC for Tightly Coupled Data Memory for CM33. **Note:** These are part of the 2MB SRAM Memory.
ROM	On-chip immutable ROM contains First Stage Bootloader (FSBL)
I/D Cache	16 KB of I-Cache with ECC and 16 KB of D-Cache with ECC for CM85. Code-bus Cache (C-Cache): 16 KB with ECC and System-bus Cache (S-Cache): 16 KB with ECC for CM33 Note: In some of the RA8 dual-core Part numbers, the CPU1 doesn't contain C/S cache
Option-setting memory	The option-setting memory determines the MCU's state after a reset.
Dedicated external Memory Support using OSPI or QSPI (2 slots of 256 MB)	
OSPI Flash	256 MB *2 external memory address space shared for CM85 and CM33
OSPI RAM	256 MB *2 external memory address space shared for CM85 and CM33
QSPI Flash	256 MB *2 external memory address space shared for CM85 and CM33
Dedicated external SDRAM Memory Support 128 MB	
SDRAM	128 MB (SDRAM) external memory address space shared for CM85 and CM33
Dedicated Chip Select memory-mapped addresses are for additional SRAM and peripherals.	
CS Area (Chip Select addressable Area)	8 X 16 MB addressable memory regions for additional SRAM, and other memory-mapped connections shared for CM85 and CM33.

All these memories are not in a contiguous area. There are reserved areas for internal/future use. The memory layout can be mainly divided into 4 different categories. Internal code memory (MRAM), Internal data memory (SRAM), External memory address space interface, and Peripheral Registers. Another important feature of the RA8 dual-core MCU is that the memory regions are aliased to Secure and Non-secure areas.

For instance, the regions from 0x0000_0000 to 0x0FFF_FFFF are aliased to 0x1000_0000 to 0x1FFF_FFFF for Secure and Non-secure access.

Details of the individual memory will be explained in the following sections. Figure 4 shows the RA8 dual-core Memory Map in detail.



	CPU0 (CM85)	CPU1 (CM33)	the others	IDAU/MSAU Security_Attribution
0xFFFF_FFFF	System for CM85	System for CM33	Reserved area*1	
0xE010_0000	Private Peripheral bus	Private Peripheral bus	Reserved area*1	
0xE000_0000	Reserved area*1			Non-secure
0xA000_0000	External address space (OSPI area)			
0x7000_0000	External address space (SDRAM area)			
0x6800_0000	External address space (CS area)			
0x6000_0000	Reserved area*1			
0x5050_0000	Peripheral IO registers			
0x5000_0000	Reserved area*1			Secure
0x4050_0000	Peripheral IO registers			
0x4000_0000	Reserved area*1			
0x3A02_0000	CPU1-STCM alias*9	Reserved area*1	CPU1-STCM alias*9	Non-secure
0x3A01_0000	CPU1-CTCM alias*8	Reserved area*1	CPU1-CTCM alias*8	
0x3A00_0000	Reserved area*1			
0x3804_0000	Reserved area*1	CPU0-DTCM alias*7		
0x3802_0000	Reserved area*1	CPU0- ITCM alias*6		
0x3800_0000	Reserved area*1			
0x321D_4000	On-chip SRAM (SRAME)*10			
0x321A_0000	On-chip SRAM			
0x3200_0000	Reserved area*1			
0x3002_0000	CPU0-DTCM*7	Reserved area*1	Reserved area*1	
0x3001_0000		CPU1-STCM*9		
0x3000_0000	Reserved area*1			
0x2A02_0000	CPU1-STCM alias*5	Reserved area*1	CPU1-STCM alias*5	Non-secure callable for CPU Secure for other bus masters
0x2A01_0000	CPU1-CTCM alias*4	Reserved area*1	CPU1-CTCM alias*4	
0x2A00_0000	Reserved area*1			
0x2804_0000	Reserved area*1	CPU0-DTCM alias*3		
0x2802_0000	Reserved area*1	CPU0-ITCM alias*2		
0x2800_0000	Reserved area*1			
0x221D_4000	On-chip SRAM (SRAME)*10			
0x221A_0000	On-chip SRAM			
0x2200_0000	Reserved area*1			
0x2002_0000	CPU0-DTCM*3	Reserved area*1	Reserved area*1	
0x2001_0000		CPU1-STCM*5		
0x2000_0000	SiP-Flash area (OSPI1 CS1 area)			
0x1800_0000	Reserved area*1			
0x1300_0000	Extra MRAM			
0x12E0_0000	Reserved area*1			
0x12D0_0000	Extra MRAM			
0x12A0_0000	Reserved area*1			Non-secure
0x1210_0000	Code MRAM			
0x1200_0000	Reserved area*1			
0x1002_0000	CPU0-ITCM*6	Reserved area*1	Reserved area*1	
0x1001_0000		CPU1-CTCM*8		
0x1000_0000	SiP-Flash area (OSPI1 CS1 area)			
0x0800_0000	Reserved area*1			
0x0300_0000	Extra MRAM			
0x02E0_0000	Reserved area*1			
0x02D0_0000	Extra MRAM			
0x02A0_0000	Reserved area*1			Non-secure callable for CPU Secure for other bus masters
0x0210_0000	Code MRAM			
0x0200_0000	Reserved area*1			
0x0002_0000	CPU0- ITCM*2	Reserved area*1	Reserved area*1	
0x0001_0000		CPU1-CTCM*4		
0x0000_0000				

Figure 4. RA8 Dual-core Memory Map

2.1 Internal Memory

In RA8 dual-core, On-chip memory plays a critical role in system performance by enabling fast data access, reducing latency, and ensuring data integrity. The memory architecture incorporates various performance

optimizations using cache, tightly coupled memory (TCM), prefetch buffers to enhance execution efficiency, along with Error Correction Code (ECC), and zero wait states.

Internal memory can be mainly divided into Code MRAM, SRAM, and TCM for Instruction and Data access. From Figure 4, you can see the MRAM placed at an address of (0x0200_0000 - 0x0210_0000) for secure access and (0x1200_0000- 0x1210_0000) for non-secure access. Similarly, the SRAM is placed at addresses (0x2200_0000 - 0x221A_0000) for secure access and (0x3200_0000 - 0x321A_0000) for non-secure access. In addition, TCM for Instructions is placed at addresses (0x0000_0000 - 0x0001_1FFF) for secure access and (0x1000_0000 - 0x1001_1FFF) for non-secure access. TCM for data is located at (0x2000_0000 - 0x2001_1FFF) for secure access and (0x3000_0000 - 0x3001_1FFF) for non-secure access.

SiP-Flash area (OSPI1 CS1 area), which is (0x0800_0000 – 0x1000_0000) for secure, and SiP-Flash area (OSPI1 CS1 area), which is (0x1800_0000 – 0x2000_0000) for non-secure, are selectively available on MCU devices where the OSPI flash is inside the MCU instead of having an external OSPI as part of the board.

On-chip SRAM (SRAME), which is (0x221A_0000 - 0x221D_4000) for secure access and (0x221A_0000 - 0x321D_4000) for non-secure access, is available in selective MCU devices where additional SRAM for large data storage is placed inside the MCU.

Internal memory also has Extra MRAM sections. This is not used for storing the program but for internal use. On-chip flash (option-setting memory) is placed at an address of (0x12E0_0000 - 0x12D0_0000, 0x12A0_0000 - 0x1300_0000) for secure access and (0x02E0_0000 - 0x02D0_0000, 0x02A0_0000 - 0x0300_0000) for non-secure access as part of the Extra MRAM.

2.1.1 MRAM Code Memory

Magneto-resistive Random Access Memory (MRAM) is a cutting-edge non-volatile memory technology that combines the speed of SRAM with the long-term data retention of flash memory. This hybrid advantage makes MRAM particularly suitable for embedded systems, where fast read/write access, low power consumption, and high endurance are critical. One of MRAM's standout features is its ability to retain data without power, which improves energy efficiency, system reliability, and rapid recovery in the event of a power failure. These characteristics are especially beneficial in real-time and embedded applications, where consistent performance and resilience are essential. In systems with dual-core architectures, MRAM further enhances efficiency by minimizing memory requirements, enabling more compact, power-conscious designs.

A key differentiator of MRAM lies in its operational advantages over traditional memory types. Unlike flash memory, which necessitates time-consuming erase cycles before data can be rewritten, MRAM supports direct bit-level writing without prior erasure. This enables significantly faster and more efficient read/write operations. Furthermore, while flash memory degrades after a limited number of write/erase cycles, MRAM offers much higher endurance, capable of millions of cycles with minimal wear. This exceptional durability makes MRAM ideal for applications demanding frequent data updates.

MRAM also excels in power efficiency. It requires less energy to operate and does not depend on high-voltage programming, which aligns well with the needs of modern low-power embedded systems. Additionally, MRAM demonstrates robust resistance to environmental factors such as radiation and electromagnetic interference, ensuring stable operation in demanding sectors such as automotive, aerospace, and industrial automation.

In conclusion, MRAM offers a compelling blend of speed, endurance, and non-volatility, integrating the strengths of SRAM, flash, and DRAM. This makes it an ideal memory solution for microcontrollers and embedded systems that require a balance of performance, data integrity, and energy efficiency.

In the Renesas RA8 dual-core MCU, the code MRAM stores instructions and constant data, functioning similarly to flash memory. It offers up to 1 MB of user-accessible memory, segmented into code MRAM and option-setting memory. MRAM programming is performed via a page buffer and option-setting memory in an additional MRAM using MACI commands, and both code and option-setting memory support simultaneous read/write operations during background tasks. Programming granularity includes 1, 2, 4, or 8 bytes for general data, and 16 or 32 bytes for code MRAM. While it does not support dual-bank or block swap functionality, it provides a wide range of security features, including tamper protection, startup area

protection, permanent block protection, TrustZone support, and an anti-rollback counter. Additional safety features include software-based protection, ECC support for correcting double-bit errors and detecting triple-bit errors, and robust error detection for invalid operations. The boot area can be securely configured to 8 KB, 16 KB, or 32 KB, ensuring safe firmware updates.

Table 2. RA8 Dual-Core MRAM Memory Specification

Item	Code MRAM	Option-setting memory in extra MRAM
Memory capacity	User area: 1 MB max	See section 7, Option-Setting Memory.
Read cycle	See section 60.5.2. MRCFREQ : Code MRAM Frequency Notifications Register	See section 60.5.3. MREFREQ : Extra MRAM Frequency Notifications Register
Programming method	Write page buffer	MACI command
Protection	Protects against erroneous rewriting of the MRAM	
Dual bank function	Not available	Not available
Block swap function	Not available	Not available
Background operations (BGOs)	• The code MRAM can be read while option-setting memory is being programmed • Option-setting memory can be read while the code MRAM is being programmed.	
Units of programming	Program data buffer: 1/2/4/8 bytes Code MRAM: 32 bytes	16 bytes
MACI command	Not available	Program: 16 bytes Forced stop Status clear Configuration set: 16 bytes Increment counter: 1 bit Read counter: 8 bytes
Security function	• Protect against illicit tampering with or reading out of data in MRAM • OFS area protection • Startup area select setting protection – BTFLG, BTSIZE and MSUACR registers are protected by the FSPR bit. • Permanent block protect setting protection – User area is permanently protected from programming operation by permanent block protect function. • MRAM protection for TrustZone – Protection for MRAM area (program) – Protection for MRAM area (read) – Protection for register • Extra MRAM program mode entry protection • Anti-rollback counter • HUK protection	
Safety function	Software protection • MACI command protection by MENTRYR register • User area protection by MRCP0 and MRCP1 registers • User area protection by the block protection setting. Error protection • Error is detected when unintended commands or prohibited settings occur. The MACI command is not accepted after an error detection. Boot area protection • The startup area select function allows customer to safely update the boot firmware. The size of the startup area can be selected from 8 KB, 16 KB or 32 KB. ECC support for double-bit error correction and triple-bit error detection.	
Interrupt request	• MRAM_MRCRD (Code MRAM read error) Enabled by INTENBTC and INTENBDC bits. • MRAM_MRERD (Extra MRAM read error) Enabled by INTENBTE and INTENBDE bits. • MRAM_MRCPR (Code MRAM sequencer error) Enabled by MRCAEIE bits. • MRAM_MREPR (Extra MRAM sequencer error) Enabled by CMDLKIEE and MREAEIE bits. • MRAM_ENDOFPE (Extra MRAM sequencer ready (processing end)) Enabled by MRDYIE bit.	

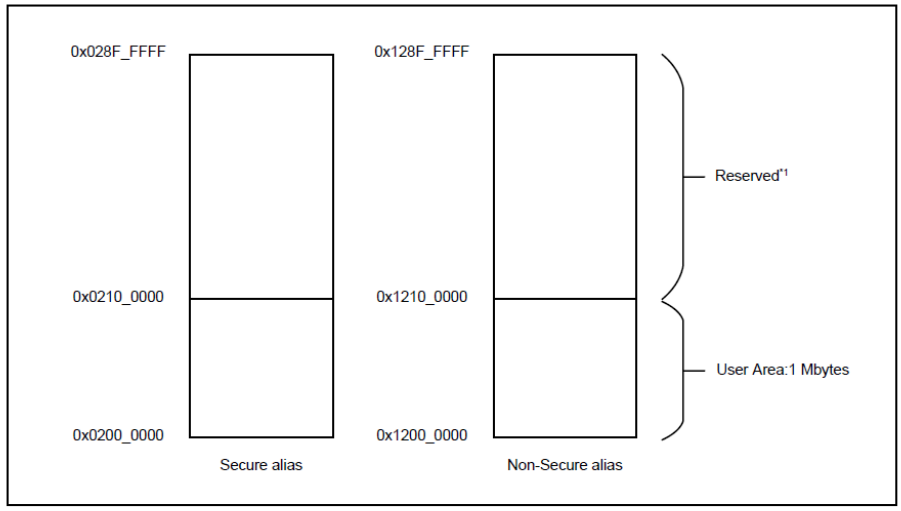
Figure 5. RA8 Dual-Core Internal MRAM Memory specifications

Additionally, the code MRAM is designed to store user application code and constant data, with support for self-programming. This allows applications to update code memory without requiring external programming

tools, enabling seamless, efficient in-field firmware updates. To assist with this, the Renesas Flexible Software Package (FSP) provides HAL drivers for modifying both code and data flash memory.

In the RA8 dual-core MCU series, the MRAM code memory serves as non-volatile storage for the firmware executed by the MCU upon power-up. This memory holds the core instructions that drive the system's functionality. Programming or writing to the MRAM code memory is typically done using serial-based tools, such as the Renesas Flash Programmer or the J-Link Flash Programmer.

The memory layout can be viewed in Figure 6, which illustrates the linear memory map of the MRAM code memory.



Note 1. Bus error is generated.

Figure 6. RA8 Dual-Core Map of the Code MRAM Memory in Linear Mode

2.1.1.1 Prefetch Buffer

The MRAM prefetch buffer feature of RA8 dual-core accelerates CPU instruction fetches from the code MRAM by reducing latency between the bus master and memory access. Specifically implemented for the target memory region 0x0000_0000 to 0x00FF_FFFF (1 MB), it supports instruction fetches from both CPU0 and CPU1 in a dual-core configuration. Each CPU is equipped with a 64-byte prefetch buffer that helps anticipate and preload instructions before the processor requests them.

The buffer is designed as a fully associative cache structure, which allows any incoming instruction address to map to any entry in the buffer, maximizing hit rates and minimizing fetch delays. Each entry holds 256 bits of aligned instruction data, and the prefetch buffer contains 2 such entries, enabling efficient lookahead and retrieval. This structure ensures that instruction streams are delivered quickly to the CPU cores, significantly improving execution speed when running code from MRAM, and maintaining system responsiveness even when accessing non-volatile memory.

Function	Implementation
Target Region	0x000_0000 ~ 0x00F_FFFF (1 MB)
Target Bus Master	CPU0 and CPU1 instruction fetch
Capacity	64 Bytes/CPU
Associativity	Full associative
	256 bits/entry (256 bits aligned data), 2 entries
Access Cycle	Hit: 0 wait Miss: See section 60.5.2. MRCFREQ : Code MRAM Frequency Notifications Register.

Figure 7. RA8 Dual-Core MRAM Prefetch Buffer Overview

2.1.1.2 Programming Procedure

Code MRAM programming is performed in 32-byte units via a program data buffer accessed through the system bus. To avoid bus stalls during programming, only one bus master should perform write access at a time, and data should not cross 32-byte boundaries. Before writing, the system must check that no other programming is in progress using the PRGBSYC and ABUFFULL flags in the MRCPS register. Data is written to the buffer and programmed into MRAM when one of the following occurs: 1) The buffer is full, the write crosses a 32-byte boundary, or 2) the MRCFL bit is set. Errors during programming can be detected via PRGERRC and ECCERRC bits. For extra MRAM programming procedures, refer to MACI commands in section 60.7 in the RA8 Dual-Core's UM.

2.1.1.3 Operating Mode - MRAM Read/Program Mode

Code MRAM and extra MRAM operate in separate read and program modes, with automatic transitions between them based on specific conditions. For code MRAM, programming mode is triggered when the program data buffer is complete, a write crosses a 32-byte boundary, or the MRCFL bit is set. Once programming is completed, the system automatically returns to reading mode. For extra MRAM, mode transitions are manually controlled by modifying the MENTRYR register or through W-HUK zeroization. When MENTRYR = 0x0000 and no zeroization is active, the MRAM remains in read mode; changing these conditions switches it to program mode.

2.1.1.4 MACI Commands - Programmable Area with MACI commands

Extra MRAM includes programmable areas for security, safety, and system configuration, which can be accessed using the MACI command. These areas include One-Time Programmable (OTP) regions, with and without Error Correction Code (ECC). In OTP areas with ECC, each 16-byte unit can be programmed only once, and bits can change only from 1 to 0; reprogramming is not allowed due to ECC constraints. In OTP areas without ECC, 16-byte units can be programmed multiple times, provided no bits are changed from 0 to 1. Reliability in non-ECC areas is ensured through redundant data configurations.

Address (Offset address: BASE_MC)	Area name	Memory type	ECC
0x00C9_F020 ~ 0x00C9_F0AF	Configuration	MRAM	Use
0x00C9_F0C0 ~ 0x00C9_F7FF	Configuration	MRAM	Use
0x00E0_7400 ~ 0x00E0_74BF	Hash of OEM_ROOT_PKn (n = 0 to 3)	OTP	Use
0x00E0_7600 ~ 0x00E0_760F	FSBL setting	OTP	Use
0x00E0_7610 ~ 0x00E0_769F	Start address of measurement report Start address of Code Certificate	OTP	Use
0x00E0_76A0 ~ 0x00E0_76FF	General purpose OTP	OTP	Use
0x00E1_7700 ~ 0x00E1_777F	Permanent Block Protect Setting for Secure	OTP	Don't use
0x00E1_7780 ~ 0x00E1_77FF	Permanent Block Protect Setting	OTP	Don't use
0x00E1_7900 ~ 0x00E1_790F	OFS Permanent Lock Setting	OTP	Don't use
0x00E1_7910 ~ 0x00E1_791F	REVOKE	OTP	Don't use
0x00E1_7920 ~ 0x00E1_792F	Zeroization HUK in OTP Enable	OTP	Don't use
0x00E1_7930 ~ 0x00E1_793F	Anti-Rollback Counter Setting	OTP	Don't use
0x00E1_7950 ~ 0x00E1_79FF	Reserved	OTP	Don't use
—	Anti-Rollback Counter	OTP	Don't use

Figure 8. RA8 Dual-Core Programmable area with MACI commands

2.1.1.5 Security function

The extra MRAM sequencer in security-enabled products provides key security features, including permanent protection of Option Function Select (OFS) settings, startup area and size security flags, and permanent block protection. It also supports TrustZone-based MRAM protection, safeguards for the extra MRAM configuration area, an anti-rollback counter, and protection for critical keys such as the Hardware Unique Key (HUK) and OEM_ROOT_KEY.

2.1.1.6 Protection function

The software protection function restricts MRAM programming operations based on specific register and configuration settings. It includes multiple layers of protection, such as code MRAM programming protection, block-level protection, error detection, and startup program protection, enhancing system security and preventing unauthorized memory modifications.

2.1.1.7 Boot Mode

There are two serial programming modes for boot operations: one using the SCI9 interface and the other using the USBFS interface. Each mode utilizes specific I/O pins and supports different communication interfaces, as detailed inFigure 9 and Figure 10.

Pin Name	I/O	Mode to be Used	Use
MD	Input	Boot mode (for the SCI interface) Boot mode (for the USB interface)	Selection of operating mode
P208/RXD9	Input	Boot mode (for the SCI interface)	For host communication (to receive data through SCI)
P209/TXD9	Output		For host communication (to transmit data through SCI)
USB_DP, USB_DM	I/O	Boot mode (for the USB interface)	Data input/output of USB
USB_VBUS	Input		Detection of connection and disconnection of USB cables

Figure 9. RA8 Dual-Core I/O Pins Used in Boot Mode

Main clock oscillator or external clock is connected	Yes	No	No
Sub clock oscillator is connected*1	Yes or No	Yes	No
Available interface	SCI or USB	SCI or USB	SCI
Tool connection time*2	Up to 1 second	Up to 2 seconds	Up to 3 seconds

Figure 10. Available Communication Interface Used in Boot Mode

Note: For the detailed description of MRAM Programming, Operating Modes, Protection function, security function and Boot Mode please refer to the MRAM section of the RA8 dual-core User's Manual.

2.1.2 Option Setting Memory

The option-setting memory determines the state of the MCU after a reset. It is allocated to the configuration setting area and the program flash area of the MRAM memory. The available methods of setting are different for the two areas. Option setting memory may differ in size and layout for Cortex-M85 and Cortex-M33 based devices.

The registers are detailed in the "*Option Setting Memory" chapter in the RA8 Dual-Core User's Hardware Manual*.

Option-setting memory registers have a discontinuous address space layout in the code flash memory. Sometimes the registers may be located in a portion of the MRAM memory that is near reserved areas in the internal MRAM. It is possible that some customers may inadvertently store data in the registers of option-setting memory or write into the reserved area in internal MRAM. This may result in improper functionality. It is advised that the user check to ensure that no unwanted data is written to these locations prior to programming the internal MRAM by reviewing the map file or s-record files generated by the compiler during binary creation. For instance, settings in the flash option registers can enable the Independent Watchdog Timer (IWDT) immediately after reset. If data stored in the program ROM inadvertently overlaps the option setting memory register, it is possible to turn on the IWDT without realizing it. This may cause the debugger to have communication problems with the board. Additionally, security attribution of code flash option-setting memory can impact which value is applied at runtime, so the user must confirm that the required values are programmed into the option-setting memory.

The image below shows the option setting memory, which consists of the option function select registers on the RA8 dual-core MCUs.

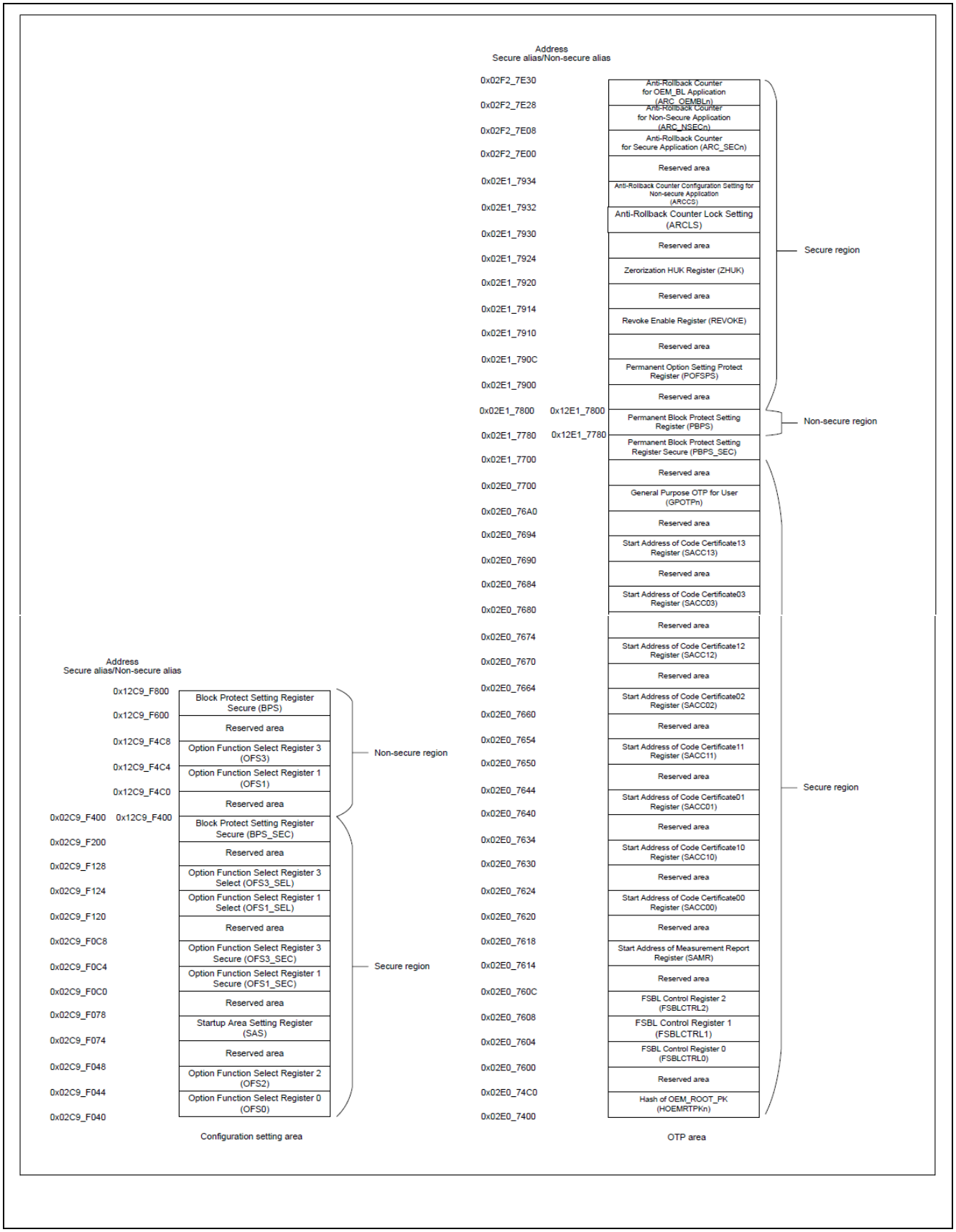


Figure 11. RA8 Dual-Core MCU Option Settings Memory Map

2.1.2.1 Option Setting Memory Registers

The following is a summary of the option setting memory registers. Make sure that they are correctly configured before first programming the MCU for startup. To check the configuration, review the map file and the output file (in hex or s-record format) to confirm the values programmed into the option setting memory registers.

Table 3. RA8 Dual-Core Memory

MCU Sub-system Control Settings	
OFS0 register	Independent Watchdog Timer (IWDT) auto start
	IWDT timeout, frequency, windowing, interrupt type, and low power mode behavior
	Watchdog Timer (WDT) auto start
	WDT timeout, frequency, windowing, interrupt type, and low power mode behavior
OFS1 register	PVD0 startup settings after reset
	HOCO startup settings after reset
	Software Debug Control
	Set ECC function of TCM and CACHE
OFS2 register	DCDC enabled at reset
Code Memory Setting	
SAS register	Startup area selection
DUALSEL register	Code Flash memory mode upon reset
BANKSEL registers	Controls swapping banks at the next reset
BPS, PBPS registers	Turns off the erasing and programming capability of selected blocks of code flash memory. Note that when PBPS are set to disable the flash erase and programming, it can never be reverted.
Firmware Verification and Update Control Registers	
FSBLCTRLx registers	Controls the operation of the First Stage Boot Loader, which verifies the integrity and authenticity of an OEM boot loader (called OEM_BL) starting at the beginning of application executable memory.
SACCx register	Identifies Locations in code flash memory where x.509 certificates are stored for validating the application when FSBL is operating in Secure Boot mode.
	SACC0/1 is selected by the FSBL based on the MCU Start Area Selection and the Bank mode. The user can reference the “Secure Boot” section in the Hardware User’s Manual to understand this selection process.
SAMR register	FSBL stores the measurement report to the SRAM address specified by the SAMR register
HOEMRTPK register	This register can be programmed only by the MCU boot firmware. It is programmed after a code image is validated. This register contains a processed Hash value.
CFGDxLOCK register	Specify write protection for the corresponding Lockable Configuration Data Areas in Data Flash. Note that when the protection is enabled, it can never be reverted.
ARCLS register	Controls Anti Rollback Counter Lock functionality. Note that if this lock functionality is enabled, it can never be reverted. This register can be set using the Renesas Flash Programming (RFP) tool.
ARCCS register	Configures the Anti Rollback Counter operation for the non-secure application. Note that when this configuration is disabled, the “Increment Counter” or Read Counter” command cannot be issued anymore. This register can be set using the Renesas Flash Programming (RFP) tool.
ARC_SECn registers	Anti-Rollback Counter for Secure Application
ARC_NSECn registers	Anti-Rollback Counter for Non-Secure Application
ARC_OEMBLn registers	Anti-Rollback Counter for OEMBL

Renesas FSP Configurator supports setting the option memory in BSP settings, as shown in the following Figure 12 for the RA8 dual-core MCU.

EK-RA8P1

Settings

Property	Value
R7KA8P1KFLCAC	
RA8P1	
RA8P1 Device Options	
OFS Registers	
OFS0 (Option Function Select Register 0) Settings	Enabled
Independent WDT	
WDT0	
OFS2 (Option Function Select Register 2) Settings	Enabled
DCDC	Enabled
CVM Reset	Enabled
NPU Security Attribution Initial Value	Non Secure
NPU Privilege Attribution Initial Value	Unprivilege
SAS (Startup Area Setting Register) Settings	Disabled
Size of Start-Up Area select	8KB
OFS1 (Option Function Select Register 1) Settings	Disabled
Voltage Detection 0 Circuit Start	Voltage monitor 0 reset is disabled after reset
Voltage Detection 0 Level	1.60 V
HOCO Oscillation Enable	HOCO oscillation is disabled after reset
Voltage Detection 0 Low Power Consumption	Voltage monitor 0 Low Power Consumption Disabled
WDT/IWDT Software Debug Control	Disabled (WDT and IWDT continue operating while the CPU is in the debug state)
Tightly Coupled Memory (TCM)/Cache ECC	Disable ECC function for TCM and Cache
OFS1_SEC (Option Function Select Register 1 Secure) Settings	Enabled
Voltage Detection 0 Circuit Start	Voltage monitor 0 reset is disabled after reset
Voltage Detection 0 Level	1.60 V
HOCO Oscillation Enable	HOCO oscillation is disabled after reset
Voltage Detection 0 Low Power Consumption	Voltage monitor 0 Low Power Consumption Disabled
WDT/IWDT Software Debug Control	Disabled (WDT and IWDT continue operating while the CPU is in the debug state)
Tightly Coupled Memory (TCM)/Cache ECC	Disable ECC function for TCM and Cache
OFS1_SEL (OFS1 Register Select) Settings	Enabled
Voltage Detection 0 Level Security Attribution	VDSEL setting loads from OFS1_SEC
Voltage Detection 0 Circuit Start Security Attribution	PVDAS setting loads from OFS1_SEC
Voltage Detection 0 Low Power Consumption Security Attribution	PVDLPSEL setting loads from OFS1_SEC
WDT/IWDT Software Debug Control Security Attribution	SWDBG setting loads from OFS1_SEC
Tightly Coupled Memory (TCM)/Cache ECC Security Attribution	INITECCEN setting loads from OFS1_SEC
OFS3 (Option Function Select Register 3) Settings	Disabled
OFS3_SEC (Option Function Select Register 3 Secure) Settings	Enabled
OFS3_SEL (OFS3 Register Select) Settings	Enabled
BPS (Block Protect Setting Register) Settings	Disabled
BPS_SEC (Block Protect Setting Register Secure) Settings	Disabled
PBPS_SEC (Permanent Block Protect Setting Register Secure) Settings	Disabled
PBPS (Permanent Block Protect Setting Register) Settings	Disabled
RA8P1 Family	

Figure 12. Option Memory Settings in FSP Configuration for RA8 Dual-Core MCU

You can also use the `objdump` command to check the settings of the option setting memory. Following is an example for dumping the option setting registers:

```
arm-none-eabi-objdump.exe -s --start-address=0x02C9F040 --stop-
address=0x02C9F4C4 elf_from_gcc.elf
```

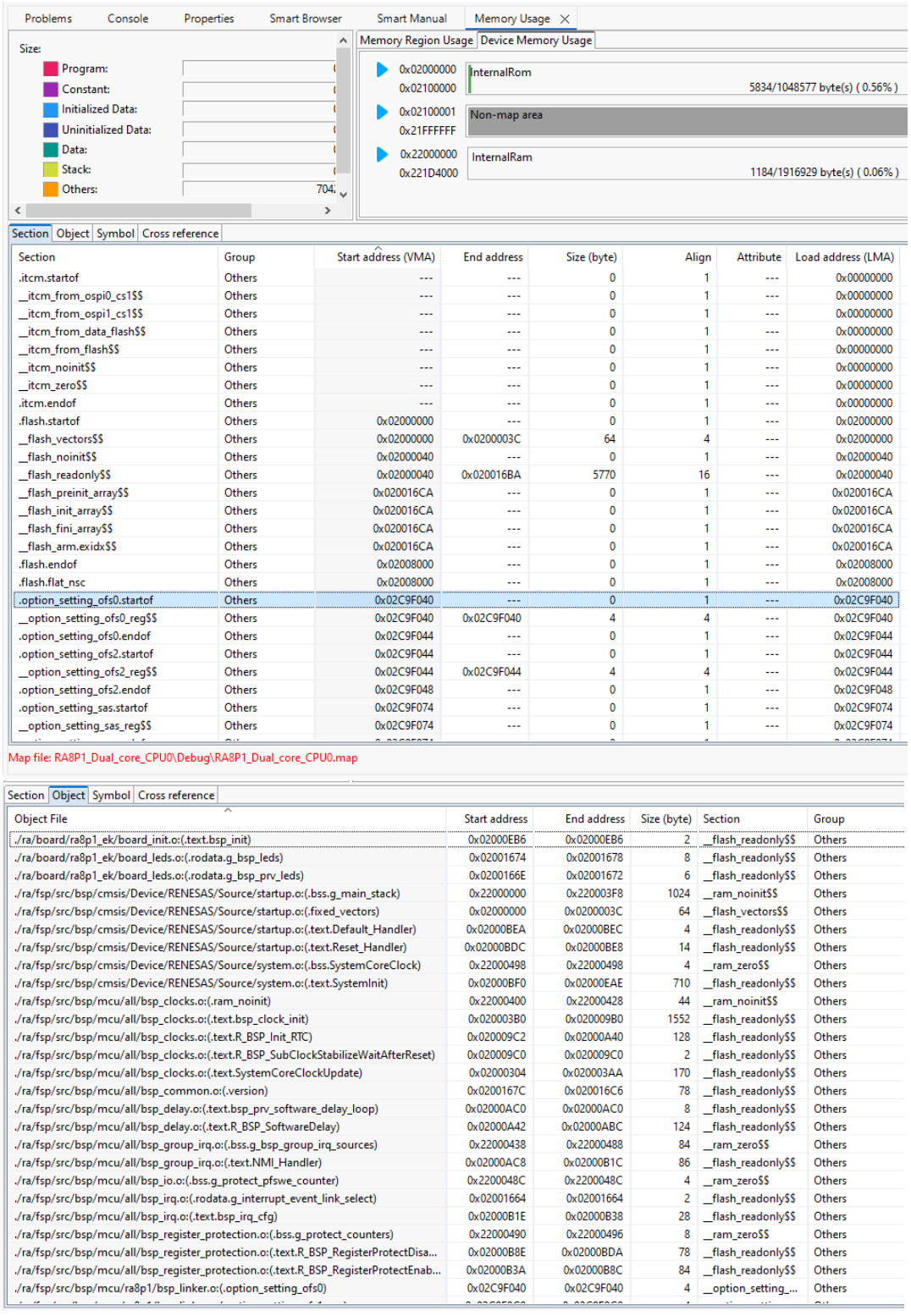


Figure 13. Using the Memory Usage View to Identify Option Setting Address Locations Programmed

2.1.3 On-Chip ROM

RA8 dual-core's On-chip immutable read-only memory (ROM) contains the First Stage Bootloader (FSBL), which is a bootloader that is permanently embedded in read-only memory, and it cannot be altered or modified. The primary purpose of such a bootloader is to provide the initial bootstrapping process for the

MCU while ensuring its security and the integrity of the first application (typically, an OEM-developed bootloader) executed. FSBL is programmed into the ROM during the manufacturing process. This cannot be altered without physically altering the hardware, making it immune to external tampering or unauthorized modifications. FSBL enhances device security by preventing unauthorized or malicious code from loading during the boot process. This ensures that only trusted and verified code can execute.

The primary function of the FSBL is to initialize the hardware and load the second-stage bootloader or main firmware from a secure and trusted source, such as a secure storage medium or a secure boot mechanism. In the RA8 dual-core MCU, the FSBL plays a critical role in implementing secure boot procedures. It verifies the authenticity and integrity of the second-stage bootloader or firmware before allowing it to execute. This helps protect against firmware-level attacks and unauthorized code execution. The FBSL can also facilitate firmware recovery in case of a system failure or corruption of the main firmware. It can be designed with a minimal, stable codebase to restore the system to a known-good state.

The FSBL's operation is managed by the Option-Setting Registers. Registers used during the development phase can be configured via the BSP tab property setting. The FSBL, when enabled, can verify the integrity and authenticity of an OEM bootloader (OEM_BL) or a normal application (without bootloader capability) starting from the application executable memory. The OEM_BL or a normal application is verified at initial programming and prior to execution. For RA8 dual-core MCUs, user can use the FSP MCUboot module to establish the OEM_BL or use their own custom bootloader.

For specifications on secure boot operations, the user can reference the *RA8 Dual-Core User's Manual: Hardware "section 52.7 Secure Boot"*. For examples on how to use the Secure Boot feature, users can refer to the application project *(OEM_BL and Application Design Using the RA8 First Stage Bootloader)*.

2.1.4 SRAM

RA8 dual-core contains on-chip 2MB (256 KB CM85 TCM RAM, 128 KB CM33 TCM RAM and 1664 KB of users SRAM) of on-chip high-speed SRAM with Error Correction Code (ECC).

SRAM Specification

Parameter		SRAM0	SRAM1	SRAM2	SRAM3
SRAM capacity		512 KB	512 KB	512 KB	128 KB
SRAM address	Secure alias	0x2200_0000 to 0x2207_FFFF	0x2208_0000 to 0x220F_FFFF	0x2210_0000 to 0x2217_FFFF	0x2218_0000 to 0x2219_FFFF
	Non-secure alias	0x3200_0000 to 0x3207_FFFF	0x3208_0000 to 0x320F_FFFF	0x3210_0000 to 0x3217_FFFF	0x3218_0000 to 0x3219_FFFF
ECC region*1	Secure alias	0x221A_0000 to 0x221A_FFFF	0x221B_0000 to 0x221B_FFFF	0x221C_0000 to 0x221C_FFFF	0x221D_0000 to 0x221D_3FFF
	Non-secure alias	0x321A_0000 to 0x321A_FFFF	0x321B_0000 to 0x321B_FFFF	0x321C_0000 to 0x321C_FFFF	0x321D_0000 to 0x321D_3FFF
Access		Wait states are not inserted into the access cycle by default. If the ICLK frequency is greater than 125 MHz, a wait state is required. If the ICLK frequency is less than or equal 125 MHz, a wait state is not required.			
Data retention function		Not available in Deep Software Standby mode			
Module-stop function		Module-stop state can be set to reduce power consumption			
Error checking		SEC-DED (Single-Error Correction and Double-Error Detection Code: 64-bit data with 8-bit ECC code)			
Security		TrustZone Filter is integrated for memory access and SFR access. Access to the memory space is controlled by setting the memory Security Attribution (SA). And access to I/O space (SFR) space is controlled by setting the register Security Attribution (SA). See section 59.3.5. TrustZone Filter function.			

Note 1. When ECC function is disabled, it is used as a data region and can be accessed directly. When ECC function is enabled, direct access is disabled. When bypass is enabled, direct access to ECC bit is possible.

Figure 14. RA8 Dual-Core's Internal SRAM Memory Specification

ECC with SRAM: ECC (Error-Correcting Code) memory is used to detect and correct single-bit errors in SRAM. ECC memory is typically used in mission-critical applications where data integrity is of utmost importance.

There are some drawbacks to using ECC memory. ECC memory typically has higher latency compared to non-ECC memory. This is because ECC memory requires additional time to check and potentially correct data errors. ECC memory modules are more expensive than non-ECC memory. This can be a significant drawback, especially for cost-sensitive applications like consumer products, where data integrity is not as critical. ECC memory requires additional hardware and logic to implement error checking and correction. This overhead can result in slightly lower memory bandwidth and increased power consumption, although the impact is generally minimal.

ECC for SRAM can be enabled or disabled via the ECCMOD[1:0] bits in the SRAMCRn register. When enabled, it uses SEC-DED (Single-Error Correction, Double-Error Detection) with 8-bit check bits for every 64-bit data. ECC supports correction of 1-bit errors and detection of 2-bit errors. Depending on whether error checking is enabled, corresponding status bits (ERRn0, ERRn1) may be updated. When ECC is disabled, neither error correction nor detection occurs. After power-on or exiting Deep Software Standby mode, SRAM contains undefined data, so it's recommended to initialize used areas with 64-bit writes and ECC enabled (with or without error checking) to prevent ECC errors.

The SRAM interrupt source includes an ECC error and a TrustZone® filter error. ECC error can choose a non-maskable interrupt or reset by SRAMCR0.OAD or SRAMCR1.OAD bit. The SRAM interrupt occurs when one error status in the SRAMESR register is set to 1; the SRAM interrupt continues to occur until the SRAMESR register flag is cleared. When common memory errors occur (NMISR.CMST=1 or RSTSR1.CMSR=1), read SRAMESR and check the SRAM interrupt source.

Name	Interrupt Source	DTC Activation	DMAC Activation
ECCERR	ECC error (SRAMs with ECC)	Not possible	Not possible
TZFLT	TrustZone filter error	Not possible	Not possible

Figure 15. RA8 Dual-Core's SRAM Interrupt source

SRAM Wait State: When ICLK frequency is higher than 125 MHz, do not set the wait enable bit (0x00) in the SRAMWTSC register to insert a wait cycle. When the wait is not inserted, the operation is not guaranteed. Depending on the operating frequency of ICLK, the WAIT setting for SRAM access has the following conditions.

- 250 MHz ≥ ICLK > 125 MHz = 1 wait
- 125 MHz ≥ ICLK = No-wait

Instruction Fetch from SRAM:
When executing code from SRAM, it's important to properly initialize the SRAM area to ensure correct CPU instruction prefetching. If the CPU attempts to prefetch from uninitialized memory, ECC errors may occur. To prevent this, initialize an extra 12-byte region beyond the program's end address, aligned to an 8-byte boundary. Renesas recommends using NOP instructions for this initialization.

ECC Error Checking in SRAM:
Since SRAM contents are undefined after power-on, reading uninitialized memory can trigger ECC errors. SRAM reads occur in 8-byte (64-bit) units, so all memory should be initialized in 8-byte aligned segments before access when ECC error checking is enabled.

2.1.5 TCM (Tightly Coupled Memory)

TCM is a specialized, high-speed memory closely integrated with the CPU cores of the RA8 dual-core MCU. Strategically placed near the CPU core, TCM enables low-latency, deterministic access to critical program instructions and data, allowing for quick instruction and data fetch and execution. Its close physical proximity significantly reduces access delays and avoids contention that may occur with shared memory resources.

Note: To allocate and configure the TCM memory resources using FSP configurator, please refer to the application note *"Developing-with-ra8-dual-core-mcu-(R01AN7881)"*

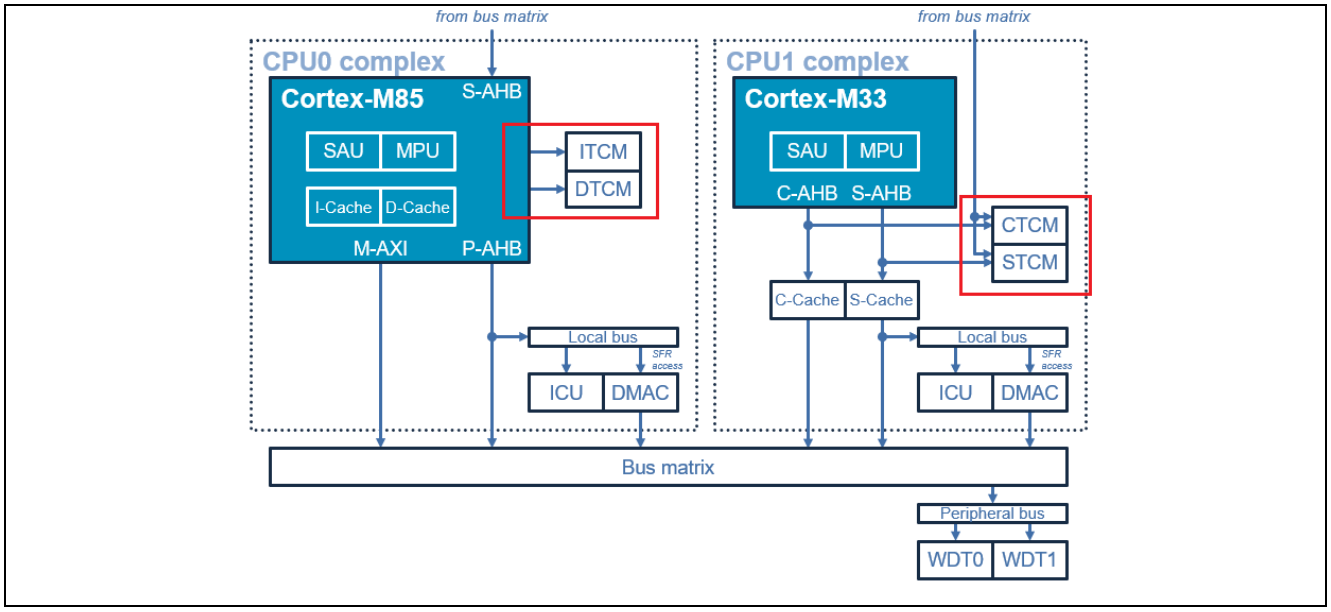


Figure 16. TCM Memory Architecture for M85 and M33 core in RA8 Dual-Core MCU

2.1.5.1 ITCM (Instruction Tightly Coupled Memory)

ITCM is a specialized, high-speed memory closely integrated with the CPU cores of the RA8 dual-core MCU. Strategically placed near the CPU core, ITCM enables low-latency, deterministic access to critical program instructions, enabling quick instruction fetches and execution. Its close physical proximity significantly reduces access delays and avoids contention that may occur with shared memory resources.

ITCM is ideal for storing time-sensitive code, such as ISRs, control loops, and high-frequency functions that demand consistent execution speed. To further enhance performance, TCM, encompassing both ITCM and DTCM, and cache memory can be used in tandem with Helium™ technology. TCM offers single-cycle, deterministic access, making it particularly effective for minimizing latency in real-time operations. Placing critical code and frequently accessed data in TCM ensures fast, predictable execution, which is essential for performance-driven embedded applications.

In the RA8 dual-core MCU, the CM85 core (CPU0) is equipped with a 128 KB ITCM organized into 16 blocks of 8 KB each and includes Error Correction Code (ECC) support. ECC functionality can be enabled via the OFS1. In the INITECCEN setting, the ITCM is activated by default through the ITCMCR.EN = 1 configuration. This ITCM is physically separate from the main Flash memory, reducing contention and ensuring faster, lower-latency instruction access than the relatively slower Flash interface.

For CPU0, ITCM is memory-mapped to:

- 0x0000_0000 – 0x0001_FFFF for secure access
- 0x1000_0000 – 0x1001_FFFF for non-secure access

Similarly, the CM33 core (CPU1) features a 64 KB Code Tightly Coupled Memory (CTCM), structured as 8 blocks of 8 KB each, also with ECC support. Like CPU0, ECC is controlled via OFS1.INITECCEN, and the memory is enabled by default with ITCMCR.EN = 1. Separating CTCM from Flash memory helps maintain consistent performance by avoiding Flash access delays.

For CPU1, the CTCM is mapped to:

- 0x0000_0000 – 0x0000_FFFF for secure access
- 0x1000_0000 – 0x1000_FFFF for non-secure access

This architecture ensures that both cores benefit from deterministic, high-speed instruction fetches without contention from shared Flash resources.

For MCU-based RTOS systems, the microkernel can be placed at the ITCM to improve the speed of the system. Motor control algorithms can be placed at the ITCM for improved(i.e., deterministic) execution.

ITCM instructions do not pass through the Cache and do not impose any wait-state restrictions. This allows for direct and deterministic access to instructions from this tightly coupled memory, improving execution speed for critical or time-sensitive code. Accessing instructions from ITCM typically incurs fewer or no wait states than accessing instructions from slower memory regions, such as Flash memory. This reduction in wait states enhances the MCU's real-time performance and responsiveness.

Note: TCM access is unavailable when the CPU is in Deep Sleep, Software Standby, or Deep Software Standby modes.

FSP Supplied ITCM initialization happens in `ra\fsp\src\bsp\cmsis\Device\RENESAS\Source\system.c`, inside the **`SystemInit().`**

```
/* Initialize ITCM RAM from ROM image. */
#if BSP_FEATURE_BSP_HAS_ITCM
   bsp_init_itcm();
#endif
```

The linker script has a section .itcm_data for the ITCM, users can place the data in the ITCM using the sample code in your application

```
#define ITCM_CODE       __attribute__((section(".itcm_data"), long_call))

ITCM_CODE void itcmfn() {

      // Place your code logic here
}
```

Users can refer to the *High Performance with RA8 MCU using CM85 core with Helium™-(R01AN7127)* application note and application project which showcases the performance improvement using the ITCM.

2.1.5.2 DTCM (Data Tightly Coupled Memory)

DTCM is another specialized memory on RA8 MCUs, closely integrated with the MCU's core processor, similar to ITCM. DTCM, however, is primarily used to store critical data that requires fast, deterministic access.

In the RA8 dual-core MCU, the CM85 core (CPU0) features a 128 KB DTCM, organized as 16 blocks of 8 KB, with Error Correction Code (ECC) support. DTCM provides fast, deterministic access to critical program data, making it ideal for variables and buffers that require low-latency, high-frequency access, such as those used in real-time processing.

Like ITCM, DTCM is physically located near the CPU core, separate from the main system RAM. This separation reduces memory access latency and avoids contention for shared memory resources, ensuring quick, predictable read/write operations. The deterministic nature of DTCM access is essential for real-time applications, where consistent timing is critical.

Due to its limited 128 KB capacity, developers must strategically allocate performance-critical data, such as control variables, sensor inputs, computation intermediates, or communication buffers, to DTCM. It is especially valuable in motor control, digital signal processing (DSP), and industrial control systems.

In the RA8 dual-core MCU, DTCM is enabled by default via DTCMCR.EN = 1, and ECC can be activated through the OFS1.INITECCEN setting. It is mapped to the following address ranges:

- 0x2000_0000 – 0x2001_FFFF for secure access
- 0x3000_0000 – 0x3001_FFFF for non-secure access

DTCM is connected to both the CPUSAHBI and DMAC/DTC buses, enabling efficient data exchange across the system without performance bottlenecks.

In contrast, the CM33 core (CPU1) includes a 64 KB Instruction TCM (ITCM), organized as 8 blocks of 8 KB, also with ECC support. Like the CM85's ITCM, it is separate from Flash memory to prevent access delays. ECC can be enabled via OFS1.INITECCEN, and it is active by default (ITCMCR.EN = 1). Its memory map is:

- 0x0000_0000 – 0x0000_FFFF for secure access
- 0x1000_0000 – 0x1000_FFFF for non-secure access

This architecture enables both cores to perform time-critical operations with reliable and consistent performance, owing to the use of dedicated, tightly integrated memory subsystems.

In Software Standby mode, the TCM data is retained. Whereas in the Deep software standby mode it is not retained.

DTCM data does not pass through the Cache and imposes no wait-state restrictions. This allows direct, deterministic access to data from this tightly coupled memory, improving access to critical or time-sensitive data. Accessing Data from DTCM typically incurs fewer or zero wait states compared to accessing data from external memories. This reduction in wait states enhances real-time performance by fast and deterministic access.

FSP Supplied DTCM initialization happens in ra\fsp\src\bsp\cmsis\Device\RENESAS\Source\system.c, inside the **SystemInit().**

```
/* Initialize DTCM RAM from ROM image and zero initialize DTCM RAM BSS section..*/
#if BSP_FEATURE_BSP_HAS_DTCM
   bsp_init_dtcm();
#endif
```

The linker script has a section .dtcm_data for the DTCM. Users can place the data in the DTCM RAM using the sample code in your application.

```
__attribute__((section(".dtcm_data"))) uint32_t my_array[64];
```

Users can refer to the *High Performance with RA8 MCU using CM85 core with Helium™-(R01AN7127)* application note and application project which showcases the performance improvement using the DTCM.

2.1.6 Cache

Cache memory on RA8 is a high-speed, volatile memory that stores frequently accessed data to reduce the time it takes to access that data from the slower main memory or storage. The primary purpose of RA8's cache is to improve the overall performance of the system by providing faster access to frequently used data. RA8 MCU provides 2 types of caches - Data Cache and Instruction Cache to improve the processing speed and reduce latency. For CPU0(CM85 core), it is implemented by ARM, and for CPU1(CM33 core) it is a Renesas Implementation, and it is called C-cache, which is for code/instruction, and S-cache which is called system cache, used for data

For CPU0, the Instruction cache: 16 KB with ECC and Data cache: 16 KB with ECC are inside the CPU0 core.

Whereas for the CPU1 Code-bus Cache (C-Cache): 16 KB with ECC on Cortex-M33 Code AHB interface (C-AHB), and another one is System-bus Cache (S-Cache): 16 KB with ECC on Cortex-M33 System AHB interface (S-AHB).

2.1.6.1 Data Cache

RA8 dual-core MCUs' CPU0 and CPU1 provide 16 KB of Data L1 Cache with ECC. L1 cache on the CPU side is used to store frequently accessed data to reduce the time it takes for the CPU to retrieve information from the main memory (RAM). This helps in accelerating program execution and data processing. The Cortex®-M85 processor L1 Data cache has the following features.

- It is a four-way set-associative cache.
- It has a cache line size of 32 bytes.
- It supports the following inner memory attributes and allocation hints for Non-shareable memory:
  - Write-Back and Write-Through Cacheable.
  - Read-Allocate and No Read-Allocate.
  - Write-Allocate and No Write-Allocate.
  - Transient and Non-transient. Clean cache lines that are associated with Transient memory are prioritized for eviction over lines that are associated with Non-transient memory.

Allocation into the L1 data cache depends on inner memory attributes only.

- The outer and inner memory attributes are exported on the Manager AXI (M-AXI) interface to support further system-level caching.
- The shareability attribute forces the region to be treated as non-cacheable, regardless of the inner memory attributes. This enables maintaining coherence at the system level.

2.1.6.2 Instruction Cache

RA8 dual-core MCUs' CPU0 and CPU1 provide 16KB of L1 Instruction Cache with ECC. L1 cache on CPU side is used to store frequently accessed instructions to reduce the time it takes for the CPU to retrieve information from the slave memory device used for retrieving instructions(code memory). This helps in accelerating program execution and data processing. The Cortex®-M85 processor L1 Data cache has the following features.

- It is a two-way set-associative cache.
- It has a cache line size of 32 bytes.
- It does not allow writes to be performed, except for allocations.
- It only supports read-allocate for inner cacheable memory. Write-Allocate, Write-Back, Write-Through, and Transient attribute hints are ignored. Allocation into the L1 data cache depends on inner memory attributes only.
- Outer and inner memory attributes are exported on the Manager-AXI (M-AXI) interface to support further system-level caching.
- The shareability attribute is ignored for instruction side accesses.
- The inner cache ability attributes are always respected.

Debug accesses from the Debug AHB (D-AHB) subordinate interface on the processor cannot read information from the instruction cache.

Software or a debugger must use the direct cache access registers to read the contents of RAM arrays. The instruction cache is logically organized into two sets of RAM arrays. The dimensions of these RAM arrays vary with the cache size and the inclusion of Error Correcting Code (ECC) logic.

2.1.6.3 How to Enable / Disable Cache Memory

By default, the RA8 dual-core device configuration enables the CM85 Instruction Cache (I-Cache) through the FSP, which also manages cache coherency as needed. The Data Cache (D-Cache) for CM85 can be optionally enabled via the BSP configuration settings, as shown in the Figure 17, but it remains disabled by default. When incorporating any form of cache into the system, it is important to consider cache coherency. For more in-depth information, refer to the Cortex-M85 Caches documentation.

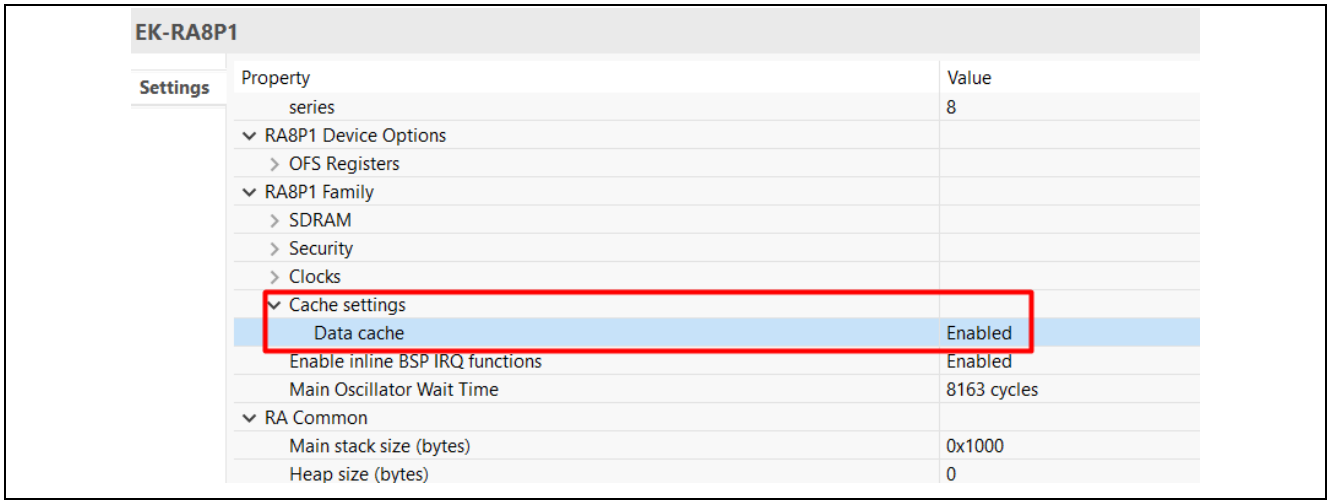


Figure 17. BSP Setting Data Cache (D-Cache) Enable

FSP provides a code in the file `RA8/ra/arm/CMSIS_5/CMSIS/Core/Include/cachel1_armv7.h` for enabling and disabling the Cache. So with FSP supplied code, this can be enable or disabled at the application level. I-Cache is enabled by FSP by default in `ra\fsp\src\bsp\cmsis\Device\RENESAS\Source\system.c`, inside the **`SystemInit().`**

```
/**
  \brief   Enable I-Cache
  \details Turns on I-Cache
  */
__STATIC_FORCEINLINE void SCB_EnableICache (void)

/**
  \brief   Disable I-Cache
  \details Turns off I-Cache
  */
__STATIC_FORCEINLINE void SCB_DisableICache (void)


/**
  \brief   Enable D-Cache
  \details Turns on D-Cache
  */
__STATIC_FORCEINLINE void SCB_EnableDCache (void)

/**
  \brief   Disable D-Cache
  \details Turns off D-Cache
  */
__STATIC_FORCEINLINE void SCB_DisableDCache (void)
```

2.1.6.4 Cache Memory and Performance Impact on the System

Using 16 KB of L1 Instruction Cache and 16 KB of L1 Data Cache memory present in the RA8 dual-core MCU's CPU0 and CPU1 can significantly impact system performance by reducing memory latency, increasing data access speed, and improving overall system throughput. However, deterministic operation is reduced. In general, though, the worst-case performance of the application can be analyzed by turning off each cache and benchmarking performance.

These caches can be used for both internal and external memory. Only the M-AXI interface accesses can be cached. TCM and P-AHB interface transactions or accesses cannot be cached.

Users can refer to the *High Performance with RA8 MCU using the CM85 core with Helium™* " application note and application project, which showcases the performance improvement using the D-cache.

2.2 External Memory

When the internal memory capacity proves insufficient, RA8 dual-core supports additional data and code memory via the external memory interface for QSPI flash, OSPI flash, OSPI RAM, and SDRAM. Additionally, the RA8 dual-core also supports 128 MB (16 *8) external CS area for memory-mapped devices. All the external memory accesses are in the non-secure region.

In RA8 dual-core, 128 MB SDRAM area (0x6800_0000 – 0x6FFF_FFFF) is exclusively dedicated for the SDRAM memory interface, whereas the 512 MB address space (0x7000_0000 – 0x9FFF_FFFF) is dedicated for OSPI flash, OSPI RAM, or QSPI Flash. There are two slots of 256 MB that can be used for OSPI flash and OSPI RAM, or two OSPI flash, or QSPI. The Chip selectable CS area of 128 MB (16MB * 8 slots) is at the address space 0x6000_0000 – 0x67FF_FFFF.

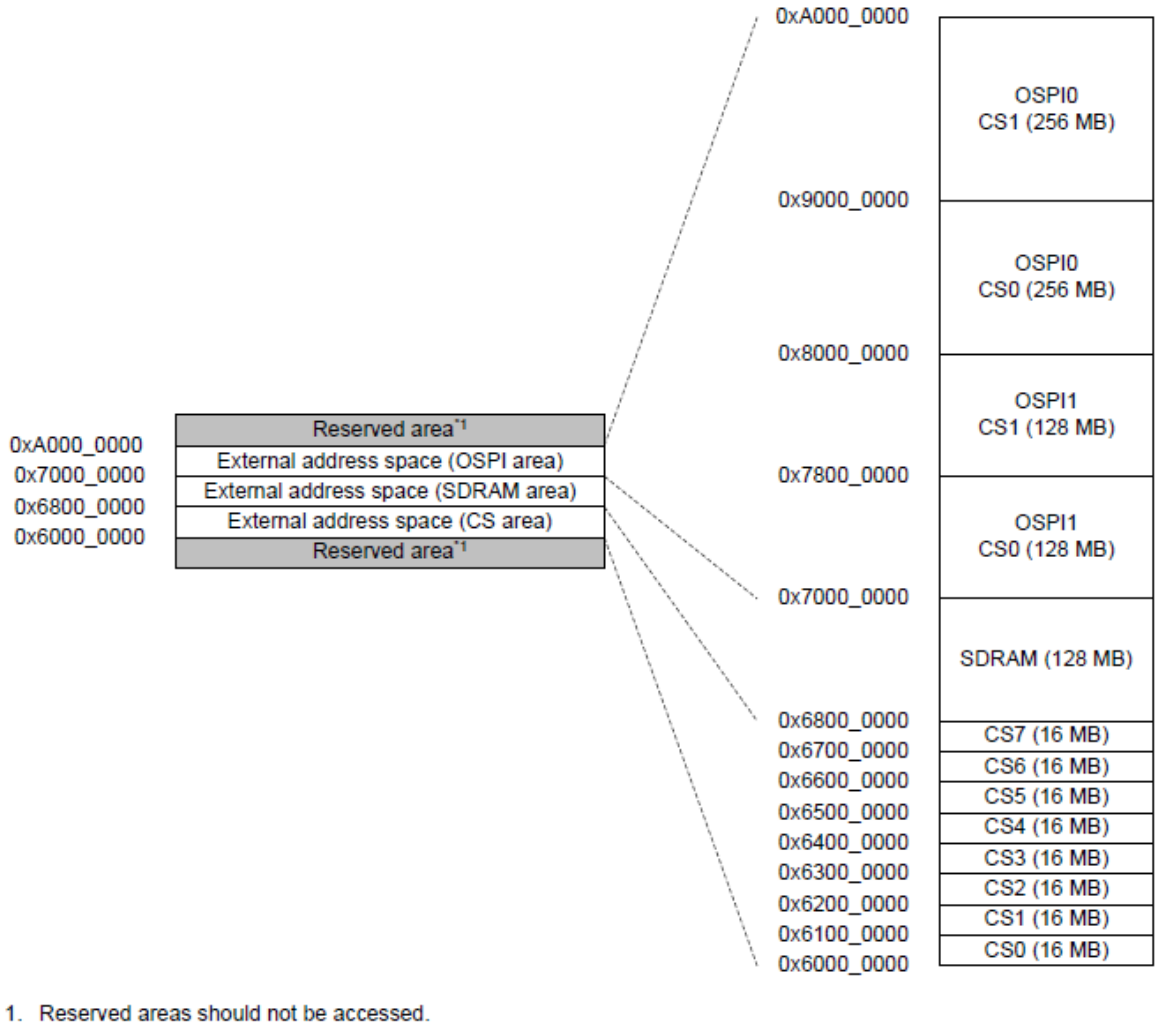


Figure 18. RA8 Dual-Core External Memory interface Address Map

The OSPI area is accessed via EOBI, whereas the CS area and SDRAM area are accessed via (ECBI).

Feature	Functional description
External buses	• CS area (ECBI): Connected to the external devices (external memory interface) • SDRAM area (ECBI): Connected to the SDRAM (external memory interface) • OSPI area (EOBI): Connected to the OSPI (external device interface)

Figure 19. RA8 Dual-Core External Memory Bus Interface

2.2.1 SDRAM

In the RA8 dual-core MCU, the SDRAM area is placed at an address range from 0x6800_0000 – 0x6FFF_FFFF. RA8 dual-core MCU's SDRAM Controller (SDRAMC) extends the memory capabilities of the chip by providing the interface to an external 8-bit, 16-bit, or 32-bit SDRAM device.

The SDRAM controller supports a CAS latency of 1, 2, or 3 cycles, thus optimizing the read access depending on the frequency. Self-refresh, power down, and Deep Power Down mode features can be used to minimize the power consumption of the SDRAM device. Selectable self-refresh and auto-refresh are provided and can multiplex the output of row address and column address (8, 9, 10, or 11 bits). SDRAMC operates in synchronization with SDCLK.

Note 1. BCLK and SDCLK must operate at the same frequency when the SDRAM is in use.

SDRAM access can be enabled or disabled using the SDC Control Register (SDCCR). The SDRAM bus width can also be set using BSIZE[1:0] in SDCCR. The refresh operation is available even when the operation of the SDRAM address space is disabled, as long as self-refresh or auto-refresh is enabled. This feature is particularly helpful when developing strategies for the application to reduce power consumption by transferring application execution into SRDAM.
By default, the Endian mode of the SDRAM is Little endian; the mode can be configured using the EMODE of the SDCMOD Register. SDRAM access mode can be set for continuous access using BE in SDAMOD: SDRAM Access Mode Register.

To control the SDRAM, the SDRAMC issues a command for each bus cycle. Commands are defined by a combination of the SDCS, RAS, CAS, WE, CKE, and other signals.

Name	Abbreviation	Command	SDCS	RAS	CAS	WE	CKE		BA1	BA0
							n-1	n		
DESL	DSL	Device deselect	H	x	x	x	H	x	x	x
ACTV	ACT	Bank active	L	L	H	H	H	x	V	V
READ	RD	Read	L	H	L	H	H	x	V	V
WRIT	WRI	Write	L	H	L	L	H	x	V	V
PRE	PRE	Precharge	L	L	H	L	H	x	V	V
PALL	PRA	All bank precharge	L	L	H	L	H	x	x	x
REF	RFA	Auto-refresh	L	L	L	H	H	x	x	x
MRS	MRS	Mode register set	L	L	L	L	H	x	L	L
SELF	RFS	Self-refresh entry	L	L	L	H	H	L	x	x
SELFX	RFX	Self-refresh end	H	x	x	x	L	H	x	x

Note: H = High level, L = Low level, V = Valid, x = Don't care.
n = Command issue cycle, n – 1 = 1 cycle before the command is issued.

Figure 20. RA8 Dual-Core MCU's External Memory- SDRAM Commands

Conditions for Setting the SDRAMC Registers: The SDRAMC registers must only be modified when all the conditions are satisfied as shown in Figure 21.

Function or operation	Registers	Conditions
Self-refresh	SDSELF*1	• SDRAM access is disabled (SDCCR.EXENB = 0*2) • Auto-refresh operation is enabled (SDRFEN.RFEN = 1).
Auto-refresh	SDRFCR	Self-refresh operation is disabled (SDSELF.SFEN = 0)
	SDRFEN	• SDRAM access is disabled (SDCCR.EXENB = 0*2) • Self-refresh operation is disabled (SDSELF.SFEN = 0).
Initialization sequence	SDIR*1	SDICR is not set yet, and the same conditions as for SDICR modification are satisfied
	SDICR*1	• Auto-refresh operation is disabled (SDRFEN.RFEN = 0) • Self-refresh operation is disabled (SDSELF.SFEN = 0).
Address register	SDADR	• SDRAM access is disabled (SDCCR.EXENB = 0*2) • Auto-refresh operation is disabled (SDRFEN.RFEN = 0) • Self-refresh operation is disabled (SDSELF.SFEN = 0).
Timing register	SDTR	• Self-refresh operation is in progress (SDSELF.SFEN = 1) or • SDRAM access is disabled (SDCCR.EXENB = 0*2) • Auto-refresh operation is disabled (SDRFEN.RFEN = 0) • Self-refresh operation is disabled (SDSELF.SFEN = 0).
Mode register	SDMOD*1	• SDRAM access is disabled (SDCCR.EXENB = 0*2) • Self-refresh operation is disabled (SDSELF.SFEN = 0).

Note 1. Before modifying this register, confirm that all the status bits in SDSR are 0.
Note 2. After writing 0 to the EXENB bit, confirm that it is cleared to 0.

Figure 21. RA8 Dual-Core - External Memory- Conditions for SDRAM register modification

Initialization Sequencer: The SDRAMC has a sequencer to issue SDRAM initialization commands. After a reset, the initialization sequencer must be activated without fail. Operation is not guaranteed if the SDRAM is not initialized.
The SDRAM initialization sequencer issues an all-bank pre-charge command followed by auto-refresh commands n times, where n = 1 to 15. The SDRAM initialization sequence timing can be set using the SDRAM Initialization Register (SDIR).
The SDRAM initialization sequence can be activated using the SDRAM Initialization Sequence Control Register (SDICR). These registers must be set only when the conditions listed in Figure 21 are satisfied.

SDRAMC Setting Procedure

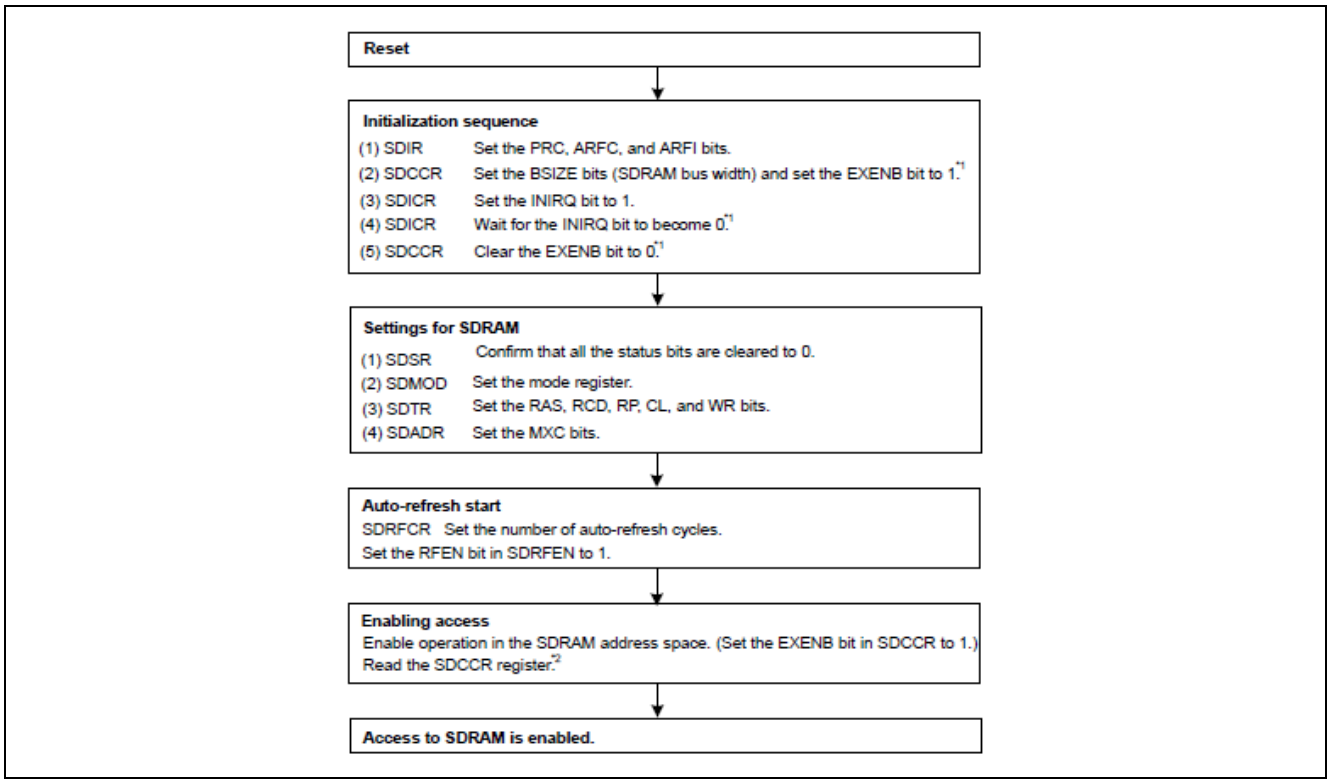


Figure 22. RA8 Dual-Core - External Memory SDRAMC Setting Procedure

The Renesas FSP provides C header files in the CMSIS data structure that map all of the external bus control registers. The Renesas FSP provides an example to initialize the SDRAM memory controller using direct register access for boards that interface the MCU and an SDRAM. Look for the *bsp_sdram_init* function in the file **`ra > board > board_name > board_sdram.c.`**

2.2.2 Octal SPI Memory

In the RA8 dual-core MCU, the OSPI area is mapped to the address range from (0x7000_0000 – 0x9FFF_FFFF). The OSPI memory interface is compliant with the xSPI protocol. The OSPI is compliant with JEDEC standard JESD251 (profiles 1.0 and 2.0), JESD251-1, and JESD252.

The OSPI interface can support two slave devices. But only one memory device can be accessed at a time. Data can be transferred at up to 200Mbytes/sec. A dedicated transfer target is provided (Channel 1: GLCDC Bus master, Channel 2: Other Bus Master), suitable for Graphical applications.

RA8 dual-core Series support xSPI compliant Octal SPI interface with support for Execute-In-Place and Decryption-on-the-fly (DOTF). The DOTF feature enables secure storage of application code or data on external Octal memory, and an encrypted code image stored in the external flash can be brought in securely via the OSPI interface. The primary advantage of using DOTF is that code execution and data read of the external memory is performed at full speed with seamless background decryption.

Decryption on the Fly (DOTF) is supported with memory-mapped reads. The data can be encrypted using a pre-stored, known key or a runtime-generated key.

The OSPI device can be erased and programmed using the OSPI APIs from the OSPI module support as well as the J-link driver that is integrated with IDE support.

The following operating modes are supported:

- Protocol modes:
  - 1/4/8pin with SDR/DDR (1S-1S-1S, 4S-4D-4D, 8D-8D-8D)
  - 2/4pin with SDR (1S-2S-2S, 2S-2S-2S, 1S-4S-4S, 4S-4S-4S)
- Configurable address length
- Configurable initial access latency cycle
- Execute in place (XIP) mode

The xSPI master interface has functions to issue transactions for external memory with xSPI slave interfaces. It allows writing to registers in external memory or reading from them. This xSPI master has two modes to issue the transaction. One is a manual-command mode where the software dynamically configures all fields of the xSPI frame and starts the transaction by software request. The other is a memory-mapping mode in which the xSPI master automatically converts the “system bus” for a pre-configured memory area into an xSPI transaction. It enables access from the system bus to the external memory area outside of the chip via the xSPI bus.
In memory-mapping mode operation, the payload of address and data field is delivered from the system bus signals. The information on the command field and size is delivered from the configured register bits. When using the FSP OSPI driver after the R_OSPI_Open API is executed successfully, access to the OSPI data area will be performed in a memory-mapped manner. Note that each memory-mapped region has an associated CS (channel selection on the OSPI FSP stack), and it is important to use the correct channel when calling the R_OSPI_Open. Figure 23 shows the register bits for memory mapping.

System bus transaction	Format change mode	Command	Command size	Address	Address size	Data	Data size	Latency cycle
Write for slave n memory area	Normal	CMCFG2CSn.WRCMD[15:8]	1 byte	SAWADDR[x:0]	CMCFG0CSn.ADDSIZE[1:0]	SWDATA	Up to SAWLEN and SAWSIZE	CMCFG2CSn.WRLATE[4:0]
	8D-8D-8D profile 1.0	CMCFG2CSn.WRCMD[15:0]	2 bytes					
	8D-8D-8D profile 2.0 Command Modifier	CMCFG2CSn.WRCMD[15:8]	1 byte	{SAWADDR[27:4], 0000000000000b, SAWADDR[3:1]}	5 bytes			
	8D-8D-8D profile 2.0 Extended Command Modifier	CMCFG2CSn.WRCMD[15:13]	3 bits	{0b, SAWADDR[31:4], 0000000000000b, SAWADDR[3:1]	45 bits			
Read for slave n memory area	Normal	CMCFG1CSn.RDCMD[15:8]	1 byte	SARADDR[x:0]	CMCFG0CSn.ADDSIZE[1:0]	SRDATA	Up to SARLEN and SARSIZE	CMCFG1CSn.RDLATE[4:0]
	8D-8D-8D profile 1.0	CMCFG1CSn.RDCMD[15:0]	2 bytes					
	8D-8D-8D profile 2.0 Command Modifier	CMCFG1CSn.RDCMD[15:8]	1 byte	{SARADDR[27:4], 0000000000000b, SARADDR[3:1]}	5 bytes			
	8D-8D-8D profile 2.0 Extended Command Modifier	CMCFG1CSn.RDCMD[15:13]	3 bits	{0b, SARADDR[31:4], 0000000000000b, SARADDR[3:1]}	45 bits			

Note: The MSByte of Address can be replaced with Address Replace Enable and Code bits (CMCFG0CSn.ADDRPEN[7:0]/ ADDRPCD[7:0]).

Figure 23. OSPI Memory-mapping Configuration for Memory Area Access (n = 0, 1) Specifications

The OSPI interface provides DOTF functionality when configured in memory map mode. This provides a strong layer of protection when storing information in the external SPI devices. The DOTF functionality supports the storage of encrypted data and code on the xSPI device. The data can be encrypted using a pre-stored, known key or a runtime-generated key. A dedicated AES engine supports transparent OSPI operation for data read and code execution. For the details on the operational flow of this feature, the user

can reference application note R11AN0773 "*Application Design using RA8 Series MCU Decryption on the Fly for OSPI*".

2.2.3 CS Addressable Memory Space

The Chip Selectable (CS) area of 8, 16MB slots of Address spaces are available for memory and memory mapped peripherals as well. The address space for the CS0 – CS7 ranges from (0x60000000 – 0x67FFFFFF).

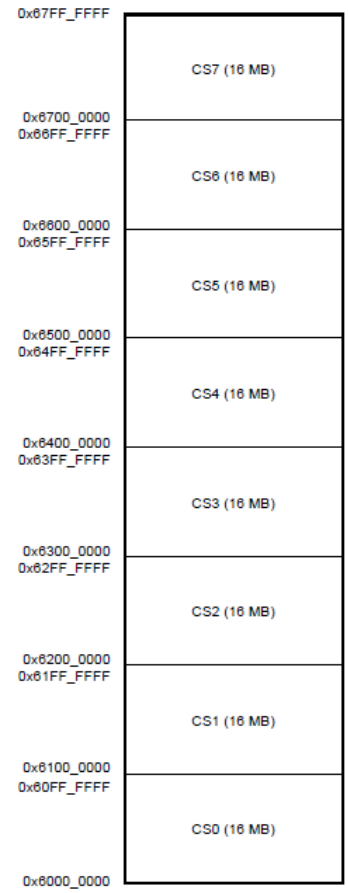


Figure 24. RA8 Dual-Core CS Area Memory Map

The use case for the external CS area is to provide additional memory capacity beyond the internal memory of an MCU. This external memory space is useful in various use cases where more memory is needed than the built-in memory can provide. Additional code storage, such as Flash NAND or NOR flash storage, and data storage, such as additional high-speed RAM storage, are examples of simple use cases.

In multi-processing systems, the CS area can be used to add an FPGA or SOC implementation, which acts as a subsystem for the MCU. Here, the MCU can be interfaced to the subsystems. For example, the MCU can be connected to the FPGA subsystem, and the MCU can upgrade the FPGA image. The MCU can have a dual-port memory through which the processed data from the FPGA can be accessed by the MCU for application-level processing.

The RA8 dual-core MCU's CS area can operate in Separate Bus and Address Data Multiplexed Bus modes.

Separate Bus mode:
In Separate Bus mode, the CS area controller (CSC) operates in synchronization with the external bus clock, BCLK. Operation cycles, such as wait cycles, specified in the CSC register, are counted on BCLK.

Normal Access: When the PRENB and PWENB bits in the CSnMOD registers are set to 0 to disable page read and page write access, all bus accesses are treated as normal read and write operations. Even when these bits are set to 1 to enable page read and page write access, a bus access other than the page access takes the form of normal read and write operations.

Page Access: When the PRENB and PWENB bits in CSnMOD are set to 1 to enable page read and page write access, the bus access for page access operations becomes page read and write. Page access can only occur when two or more rounds of external bus access are required for a single transfer request from the bus master. However, normal access is made when split accesses are not aligned or access extends across the 32-bit boundary.

Address/Data Multiplexed Bus mode:
When the address/data Multiplexed I/O Interface Select bit (MPXEN) in CSnCR is set to 1, addresses and data can be multiplexed input/output to/from the D15 to D00 pins in the corresponding area. Using this function enables direct connection of this MCU to peripherals of the MCU requiring address/data multiplexing. When an 8-bit width is selected with the BSIZE[1:0] bits in CSnCR, D7 to D00 are multiplexed with A07 to A00. When a 16-bit width is selected, D15 to D00 are multiplexed with A15 to A00. In the address/data multiplexed I/O space, accesses are controlled with the ALE, RD, WRn, and BCn signals. Byte strobe mode or single-write strobe mode is selectable in the same way as for a separate bus. However, with regard to the BCn signals within the address cycle, the byte-control signal is output for the data being read or written.
During the address/data multiplexed I/O space access, after the number of wait cycles specified by the Address Cycle Wait Select bits (AWAIT[1:0]) in CSnWCR2 is inserted in the address output cycle, data access is performed.

CS Area provides the External Wait Function. Wait cycles can be extended by the WAIT signal beyond the length of the normal access cycle wait specified in the CSRWAIT[4:0] and CSWWAIT[4:0] bits in CSnWCR1, and the page access cycle wait specified in the CSPRWAIT[2:0] and CSPWWAIT[2:0] bits in CSnWCR1. When external wait is enabled (EWENB = 1 in CSnMOD), wait cycles are inserted while the WAIT signal is held low. When the external wait is disabled (EWENB = 0 in CSnMOD), the WAIT signal has no effect. All wait cycles specified in CSnWCR1 are inserted independently of the WAIT signal. Configuring the WAIT signal enables the MCU to meet the specifications of the externally interfaced device.

With the Insertion of Recovery Cycles, recovery cycles can be inserted between consecutive rounds of external bus access by setting the Recovery Cycle Insertion Enable bit in CSRECEN to 1.

CS Area provides Write Buffer Function (External Bus). In write access, the main bus is released by writing data to the write buffer before the access is complete. This allows the next round of bus access to start. However, suppose the next access is to an external address space or to a register of the external bus controller. In that case, it is suspended until the external bus operations are in progress and complete.

Note: For a detailed external bus read-write timing diagram, refer to the HW UM of the respective MCUs.

The CS Area control register is used to configure operation with different external bus widths and endianness modes when choosing Separate or Multiplexed Address/Data Bus. CSnCR: CSn Control Register (n = 0 to 7).

Bit	Symbol	Function	R/W
0	EXENB	Operation Enable 0: Disable operation 1: Enable operation	R/W
3:1	—	These bits are read as 0. The write value should be 0.	R/W
5:4	BSIZE[1:0]	External Bus Width Select 0 0: 16-bit bus space 0 1: 32-bit bus space 1 0: 8-bit bus space Others: Setting prohibited	R/W
7:6	—	These bits are read as 0. The write value should be 0.	R/W
8	EMODE	Endian Mode 0: Little endian 1: Big endian	R/W
11:9	—	These bits are read as 0. The write value should be 0.	R/W
12	MPXEN	Address/Data Multiplexed I/O Interface Select 0: Separate bus interface is selected for area n. 1: Address/data multiplexed I/O interface is selected for area n.	R/W
15:13	—	These bits are read as 0. The write value should be 0.	R/W

Note: S-TYPE-3, P-TYPE-3
Note 1. When n = 0, the value after reset is 1.

Figure 25. RA8 Dual-Core - External Memory- CS Area Control register modification

EXENB bit (Operation Enable)
The EXENB bit enables operation of the associated CS area. On MCU reset, operation is enabled only for area 0 (EXENB = 1). Operation in other regions is disabled (EXENB = 0). Attempts to access disabled areas are ineffective. When the CSC and SDRAMC are used simultaneously, EBCLK and SDCLK must operate at the same frequency.

BSIZE[1:0] bits (External Bus Width Select)
The BSIZE[1:0] bits specify the data bus width for the associated area.

EMODE bit (Endian Mode)
The EMODE bit specifies the endianness for the associated area. The Arm® Cortex®-M85 core is fixed at little-endian order, so instruction code can only be allocated to external spaces with little-endian specified. If an area is specified as big endian, no instruction code can be allocated to it. Only CPU, DMAC, and DTC can access the big-endian area. The memory type of the big-endian area must be Device-Memory.

MPXEN bit (Address/Data Multiplexed I/O Interface Select)
The MPXEN bit specifies a separate bus interface or an address/data multiplexed I/O interface for each area.

The data alignment control for the CS area, for 32-bit, 16-bit, and 8-bit formats in little-endian and big-endian, is shown in detail in the “Endian and Data Alignment” section of the HW UM.

Note: BCLK (external bus clock): 125 MHz (max.) (The CSC (CS area controller) operates in synchronization with the BCLK).

BCLK pin output: The frequency matches the default BCLK frequency. Half of BCLK can be supplied by setting the EBCLK pin and the output select bit (BCKCR.BCLKDIV) in the External Bus Clock Control register. For details, see section, Clock Generation Circuit.

2.3 External Memory Bus Interface

The External Memory Bus interface facilitates the transfer of data between the processor and external memory devices, such as RAM or non-volatile memory, such as Flash memory. The External Memory Bus interface typically includes control signals, address lines, and data lines to establish communication and transfer data between the processor and the memory devices.

Parameter	Specifications
External address space	• The external address space is divided into 8 CS areas (CS0 to CS7) and the SDRAM area (SDCS) for management. • Chip select signals can be output for each area. • The bus width can be set for each area. – Separate bus: Selectable to 8-bit, 16-bit or 32-bit bus space – Address/data multiplexed bus: Selectable to 8-bit or 16-bit bus space • Endian mode can be specified for each area. Big endian mode has the following restrictions. See section 15.2.5.1. Endianness Constraint for details.
CS area controller	• Recovery cycles can be inserted: – Read recovery: Up to 15 cycles – Write recovery: Up to 15 cycles • Cycle wait function: Wait for up to 31 cycles (for page access, up to 7 cycles) • Use wait control to set up: – Assertion and negation timing of chip select signals (CS0 to CS7) – Assertion timing of the read signal (RD) and write signals (WR0/WR and WR1) – Timing of data output starts and ends • Write access modes: – Single-write strobe mode and byte strobe mode Separate bus or address/data multiplexed bus can be set for each area.
SDRAM area controller	• Multiplexed output of row address and column address (8, 9, 10, or 11 bits) • Self-refresh and auto-refresh selectable • CAS latency can be specified from 1 to 3 cycles.
FIFO	FIFO number : 16 for Read Data and Write Data
Frequency	• The CS area controller (CSC) operates in synchronization with the external bus clock (BCLK)*1 • The frequency of the EBCLK pin output is the same as BCLK by default. Half of the BCLK cycles can be supplied by setting the EBCLK Pin Output Select bit, BCKCR.BCLKDIV, in the External Bus Clock Control Register. For more information, see section 9, Clock Generation Circuit. • The SDRAM area controller (SDRAMC) operates in synchronization with the SDRAM clock (SDCLK).

Note 1. BCLK and SDCLK must operate at the same frequency when the SDRAM is in use.

Figure 26. RA8 Dual-Core - External Memory Bus interface

The external bus controller arbitrates requests for bus access on the external address space from the CPU M-AXI bus, DMAC/DTC bus, EDMAC (Ether) bus, GLCDC0 bus, GLCDC1 bus, DRW0 bus, DRW1 bus, MIPI bus, and CEU bus. For more details, see section "Arbitration" for the priority and arbitration method of each master to the “External Bus” section of the UM of the RA8 dual-core MCU.

The physical signals required for the RA8 dual-core to interface with external memory are listed below in Figure 27. Refer to section “External Bus” in the UM for more details, and specifically for the RA8 dual-core MCUs.

Pin name	I/O	Related functions	Description
EBCLK, SDCLK*1	Output	CSC, SDRAMC	Clock output pin
A23 to A00*2	Output	CSC, SDRAMC	Address output pins
D31 to D00 DQ31 to DQ00	I/O	CSC, SDRAMC	D31 to D00 are CSC data input/output pins. DQ31 to DQ00 are SDRAMC data input/output pins. • D31 to D00, DQ31 to DQ00 pins are enabled when the 32-bit bus space is specified. • D15 to D00, DQ15 to DQ00 pins are enabled when the 16-bit bus space is specified. • D07 to D00, DQ07 to DQ00 pins are enabled when the 8-bit bus space is specified.
BC0	Output	CSC	• Strobe signal that indicates (when low) that D07 to D00 are valid during access to an external address space in single-write strobe mode, active-low. • When an 8-bit bus space is specified, this output pin is always held low regardless of the write access mode.
BC1	Output	CSC	• Strobe signal that indicates (when low) that D15 to D08 are valid during access to an external address space in single-write strobe mode, active-low. • This pin is not used when the 8-bit bus space is specified.
BC2	Output	CSC	• Strobe signal that indicates (when low) that D23 to D16 are valid during access to an external address space in single-write strobe mode, active-low. • This pin is not used when the 8- or 16-bit bus space is specified.
BC3	Output	CSC	• Strobe signal that indicates (when low) that D31 to D24 are valid during access to an external address space in single-write strobe mode, active-low. • This pin is not used when the 8- or 16-bit bus space is specified.

Figure 27. RA8 Dual-Core - External Memory Bus Pins and their Functions

Pin name	I/O	Related functions	Description
CS1*3	Output	CSC	Chip select signal for area 1 (CS1), active-low
CS2*3	Output	CSC	Chip select signal for area 2 (CS2), active-low
CS3*3	Output	CSC	Chip select signal for area 3 (CS3), active-low
CS4	Output	CSC	Chip select signal for area 4 (CS4), active-low
CS5	Output	CSC	Chip select signal for area 5 (CS5), active-low
CS6	Output	CSC	Chip select signal for area 6 (CS6), active-low
CS7	Output	CSC	Chip select signal for area 7 (CS7), active-low
RD	Output	CSC	Strobe signal that indicates that a read from an external address space (CS0 to CS7) is in progress, active-low.
WR0/WR*4	Output	CSC	• WR0 signal is a strobe signal that indicates that a write to an external address space is in progress in byte strobe mode, and D07 to D00 are valid, active-low. • WR signal is a strobe signal that indicates that a write to an external address space is in progress in single-write strobe mode, active-low. • When an 8-bit bus space is specified, this output pin is held low during a write access regardless of the write access mode.
WR1	Output	CSC	• Strobe signal that indicates that D15 to D08 are valid during a write to an external address space in byte strobe mode, active-low. • This signal is invalid in single-write strobe mode. • This pin is not used when the 8-bit bus space is specified.
WR2	Output	CSC	• Strobe signal that indicates that D23 to D16 are valid during a write to an external address space in byte strobe mode, active-low. • This signal is invalid in single-write strobe mode. • This pin is not used when the 8- or 16-bit bus space is specified.
WR3	Output	CSC	• Strobe signal that indicates that D31 to D24 are valid during a write to an external address space in byte strobe mode, active-low. • This signal is invalid in single-write strobe mode. • This pin is not used when the 8- or 16-bit bus space is specified.
ALE	Output	CSC	Address latch signal when address/data multiplexed bus is selected.
WAIT	Input	CSC	Wait request signal used when accessing the external address space (CS0 to CS7), active-low
CKE	Output	SDRAMC	Clock enable signal
SDCS	Output	SDRAMC	Chip select signal, active-low
RAS	Output	SDRAMC	Row address strobe signal, active-low
CAS	Output	SDRAMC	Column address strobe signal, active-low
WE	Output	SDRAMC	Write enable signal, active-low
DQM0	Output	SDRAMC	I/O data mask enable signal for DQ07 to DQ00
DQM1	Output	SDRAMC	I/O data mask enable signal for DQ15 to DQ08
DQM2	Output	SDRAMC	I/O data mask enable signal for DQ23 to DQ16
DQM3	Output	SDRAMC	I/O data mask enable signal for DQ31 to DQ24

Figure 28. RA8 Dual-Core - External Memory Bus Pins and their Functions

Note 1: The EBCLK and the SDCLK pin functions are shared by the CS area controller (CSC) and the SDRAM area controller (SDRAMC). When using the CSC and the SDRAMC simultaneously, the SDCLK pin function is valid.

Note 2. The A23-to-A00 pin functions are shared between the CSC and the SDRAMC.
When using the CSC only: The A00 and BC0 pin functions share the same pin, and either becomes valid according to the area, with the function being A00 in byte strobe mode and BC0 in single-write strobe mode. Setting the 8-bit external bus width is prohibited in single-write strobe mode.
When using the SDRAMC only, The A15 to A00 pin functions are valid. The A00 and DQM1 pin functions share the same pin, and either pin function becomes valid according to the external bus width. When selecting an 8-bit bus width, the pin function is A00. When selecting a 16-bit bus width, the pin function is DQM1. When using the CSC and the SDRAMC simultaneously, the A23 to A16

pin functions are valid for CSC. The A15 to A00 pin functions are shared by the CSC and the SDRAMC. In the SDRAMC functions, the A00 and the DQM1 pin functions work as described. In the CSC functions, the A00 and the BC0 pin functions work as described.

Note 3: The CS0 to CS3 (CSC) and SDRAMC pin functions share the same pin. When using the CSC and the SDRAMC simultaneously, the CS0 to CS3 pin functions are invalid.

Note 4. The WR0 signal and WR signal are identical. The WR0 signal is referred to as WR in single-write strobe mode.

The external bus has a data alignment function to control which byte of the data bus (D31 to D24, D23 to D16, D15 to D08, D07 to D00) is used when accessing the external address space (the CS and SDRAM areas). Alignment is based on the bus specifications of the area to be accessed (8-bit, 16-bit, or 32-bit bus space), the data size, and the endian order.

FSP Configurator View
From the FSP perspective, the same signals can also be seen on the configurator **Pins** tab > **Peripherals** > **ExBus:Bus** > **Bus**.

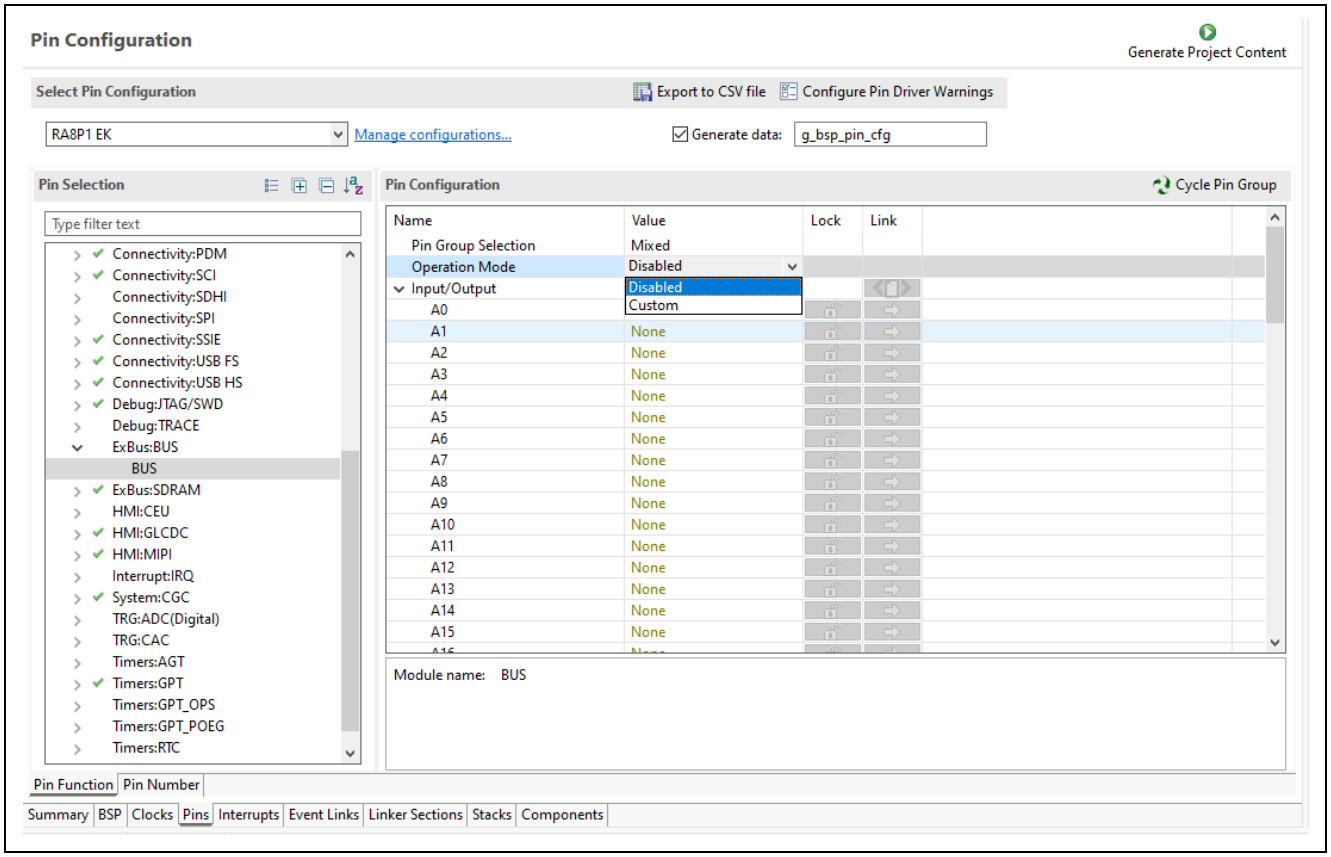


Figure 29. RA8 Dual-Core - External Memory Bus Pins Using FSP Configurator

For the SDRAM pins, from the FSP perspective, the same signals can also be seen on the configurator **Pins** tab > **Peripherals** > **ExBus: SDRAM** > **SDRAM**.

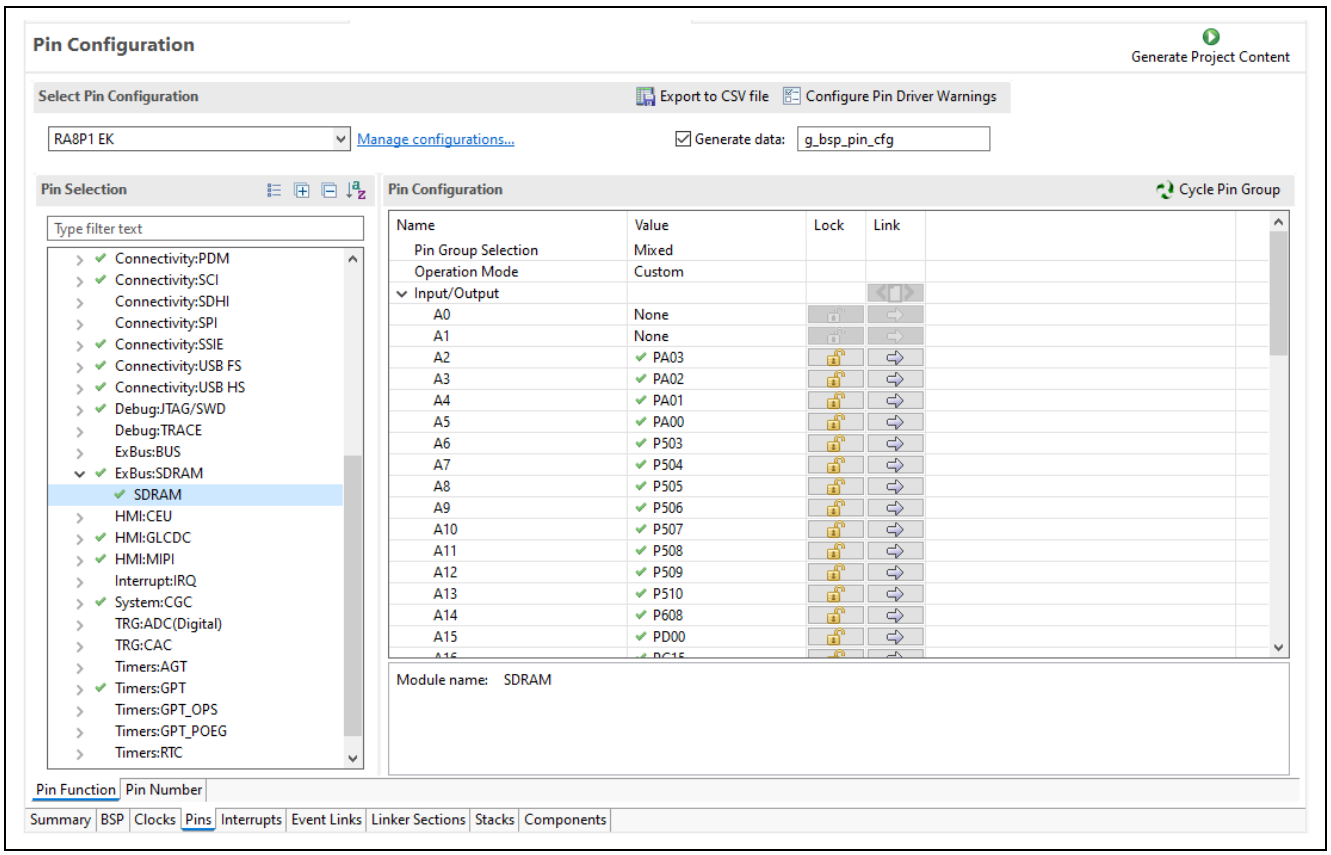


Figure 30. RA8 Dual-Core - External Memory Bus Pins and their Functions

2.4 How to Interface External Memory

The typical external memories that can be interfaced to the RA8 dual-core MCU are SDRAM, OSPI Flash, and QSPI Flash. RA8 dual-core provides an additional 128 MB (16 MB X 8) CS area, using which additional RAM can be added to the system using a parallel interface for faster access to data in a system where it goes beyond 1 MB.

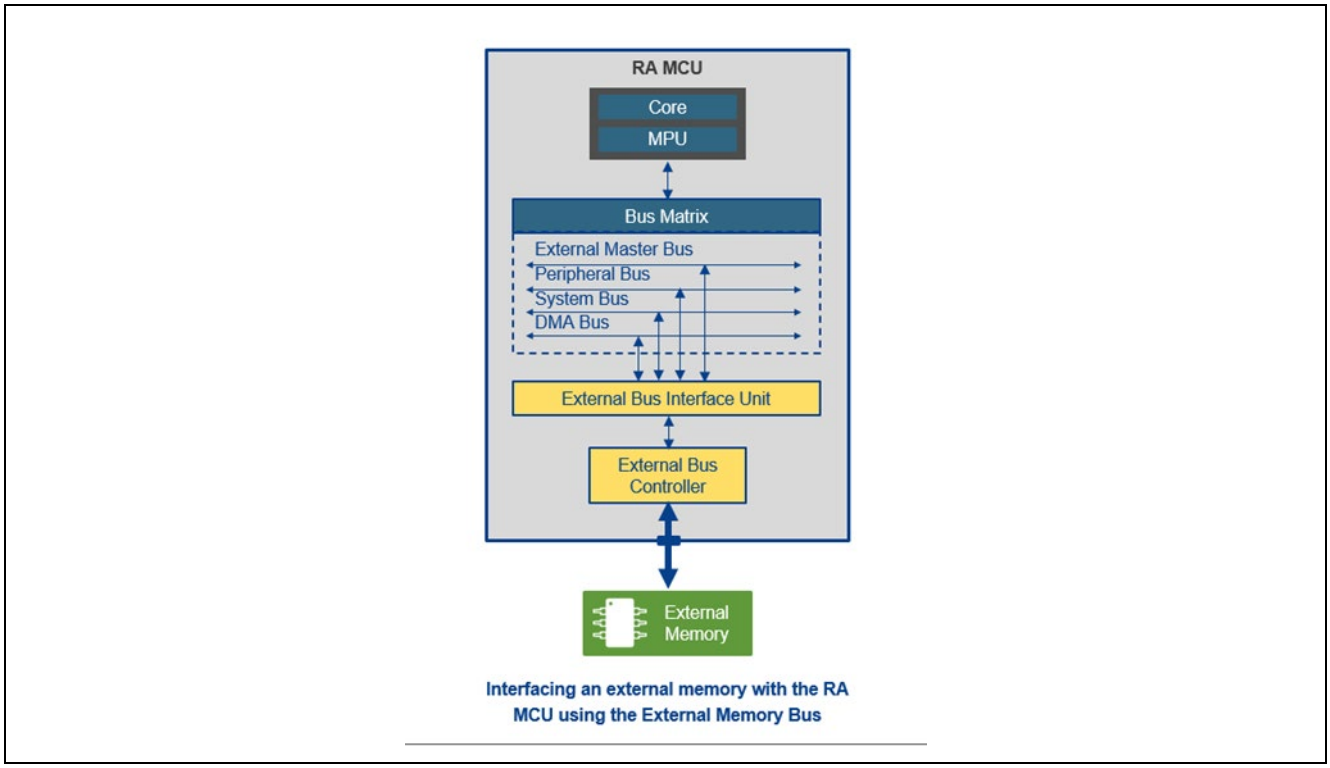


Figure 31. RA8 Dual-Core - External Memory Interface Overview

The RA8 dual-core has a dedicated SDRAM controller and OSPI bus interface unit with a bus controller through which the external SDRAM, OSPI RAM, OSPI Flash, or QSPI flash memory can be interfaced.

For interfacing the OSPI Flash to the RA8 dual-core MCU, it uses the External Octal Bus interface (EOBI), which is a serial interface. The Figure 32 shows the high-level block diagram of the external bus interface to the OSPI external memory

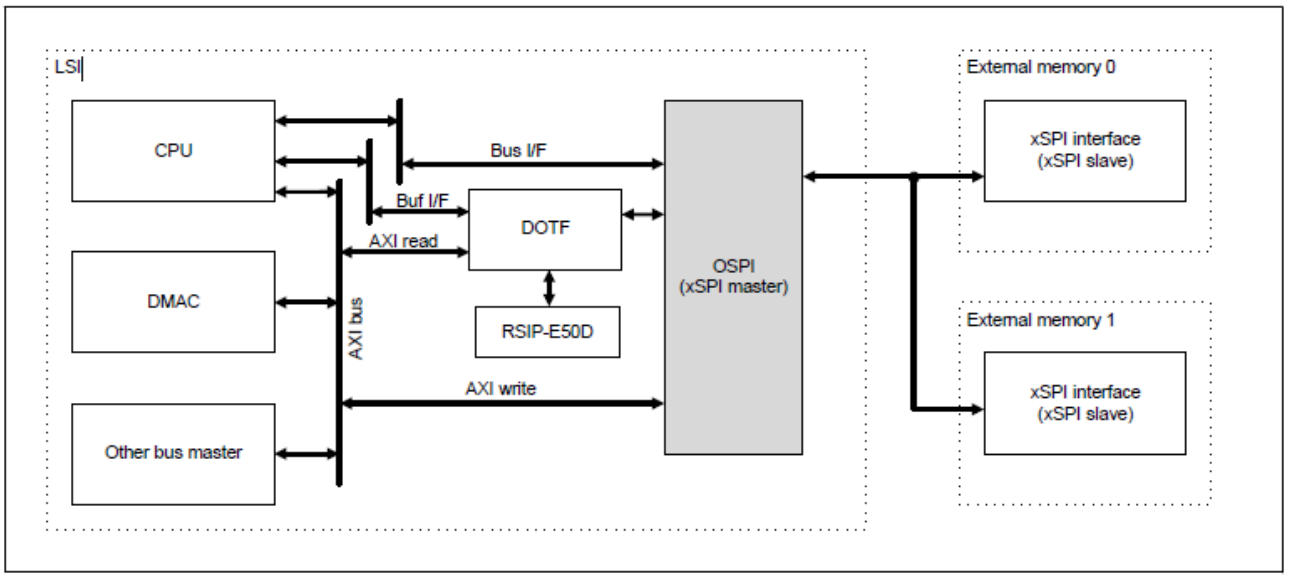


Figure 32. RA8 Dual-Core - External Memory - OSPI Interface Overview

Figure 33 shows the pin name and functions that are used for connecting the MCU to the OSPI memory.

Pin name	I/O	Function
OM_n_SCLK	Output	Clock positive
OM_n_SCLKN	Output	Clock negative
OM_n_CS0	Output	Chip select for slave 0
OM_n_CS1	Output	Chip select for slave 1
OM_n_DQS	I/O	Read data strobe/write data mask
OM_n_SIO0	I/O	Data 0 input/output
OM_n_SIO1	I/O	Data 1 input/output
OM_n_SIO2	I/O	Data 2 input/output
OM_n_SIO3	I/O	Data 3 input/output
OM_n_SIO4	I/O	Data 4 input/output
OM_n_SIO5	I/O	Data 5 input/output
OM_n_SIO6	I/O	Data 6 input/output
OM_n_SIO7	I/O	Data 7 input/output
OM_n_RESET	Output	Master reset status for slave 0, 1
OM_n_RSTO1	Input	Slave reset status for slave 1
OM_n_ECSINT1	Input	Interrupt for slave 1/error correction status for slave 1
OM_n_WP1	Output	Write protect for slave 1

Figure 33. RA8 Dual-Core - External Memory OSPI Bus Pins and Functions

The sample snapshot of the OSPI interface to EK-RA8P1 is shown in Figure 34. For a more detailed example on interfacing the OSPI flash to the RA8P1 MCU, please refer to the schematics of the EK-RA8P1 evaluation kit.

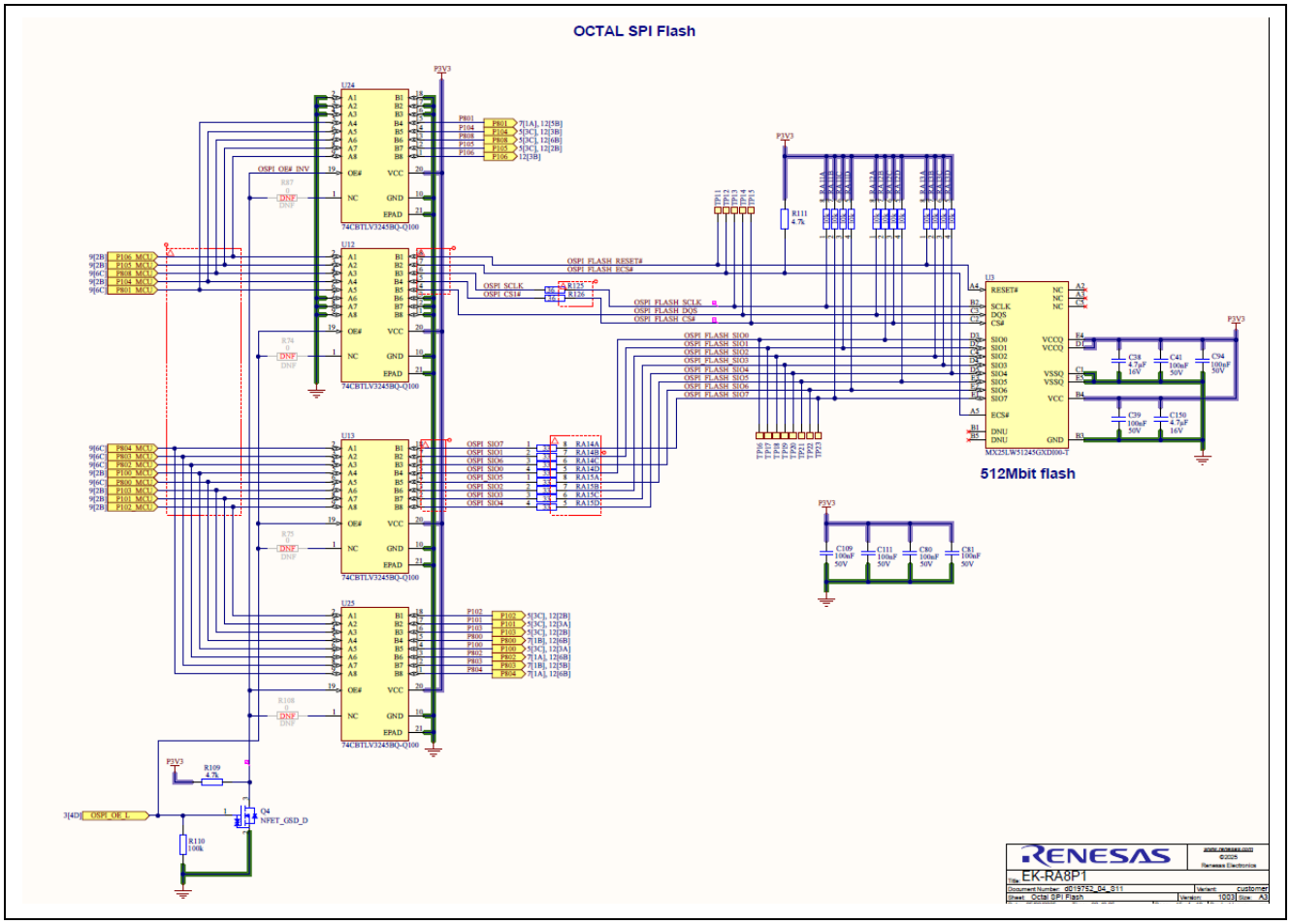


Figure 34. RA8P1 External Memory OSPI Interface Example

For interfacing the SDRAM, the RA8P1 MCU uses the External Memory CSC and SDRAM Bus Interface (ECBI), a parallel bus interface, which means you are required to connect a set of data, address, and control lines between the microcontroller and the SDRAM chip. The specific pins and connections required will depend on the Data bit mode you are trying to operate the SDRAM. RA8P1 MCU supports 8,16, and 32 bit modes. Refer to the UM of the MCU for pin assignments and connection details.

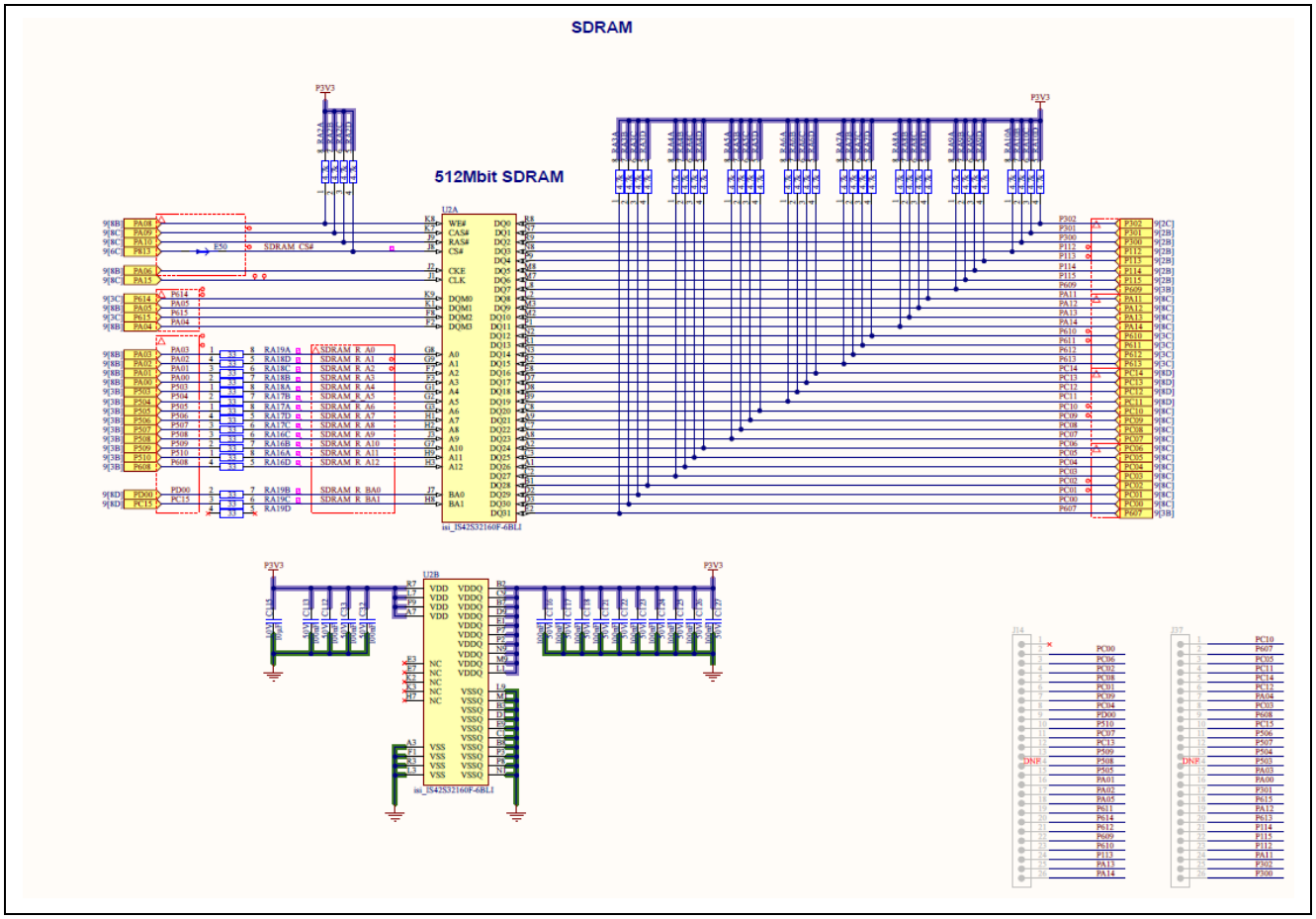


Figure 35. RA8P1 SDRAM Memory Interface Example

2.5 Supported External Memory Types on RA8 Dual-Core MCUs

RA8 dual-core Series MCUs support xSPI-compliant Octal SPI (OSPI) interface with support for Execute-In-Place and Decryption-on-the-fly (DOTF). The OSPI interface is compliant with JEDEC standard JESD251 (Profile 1.0 and 2.0), JESD251-1, and JESD252.

The RA8 dual-core Series is compatible with all flash devices that are JESD251 standard compliant and support both Octal Flash & Octal RAM and HyperFlash & HyperRAM devices that are compatible with the Infineon HyperBus specification. RA8 dual-core's xSPI guarantees operation with JESD251 (xSPI for Non-Volatile Memory) compliant memory.

The RA8 dual-core series MCUs have been tested and are confirmed to be compatible with these memory P/Ns.

Category	Supplier	Part Number
RAM	JSC	JSC28SSU8AGDY
	Cypress	S27KL0641
Flash	Infineon	S28HS512TGABHI01
	ISSI	IS25LX032/64/128
	Macronix	MX25LM51245G
	Macronix	MX25UW512454G
	Cypress	S26KL512S
	Micron (XccelaFlash)	MT25QL128ABA MT35XL512ABA
	Cypress	S25FS512S
	Macronix	MX25R1635F

3. Memory Considerations in System Design

Designing an MCU-based system with limited memory resources requires careful integration of both internal and external memory to maximize performance and storage efficiency. Internal memory resides within the MCU, while external memory, such as SDRAM and QSPI/OSPI flash, provides additional capacity for larger data storage needs.

The choice of memory significantly influences the system's performance, cost, and overall capabilities. Important factors to evaluate when selecting memory for MCU-based applications include the type and size of memory, speed and access time, limited write cycles, power consumption, support for error correction (ECC), electrical characteristics, security features, compatibility, and compliance with relevant regulations.

3.1 Internal Memory Selection in System Design

Internal memory provides several advantages over external memory, mainly due to its proximity to the CPU, which enables faster read and write operations and enhances overall system performance. It is readily available at power-up, eliminating the latency typically associated with initializing external memory interfaces. Selecting the appropriate internal memory configuration is a crucial design decision that directly influences the system's performance, cost, security, and functionality. Since internal memory is built into the MCU, it removes the need for extra components and complex PCB routing, resulting in a more compact and cost-effective design. Additionally, when external memory is not required, the unused interface pins can be reassigned for other purposes, such as GPIO, allowing for the use of smaller MCU packages.
Internal memory also consumes less power than external alternatives, making it well-suited for low-power and battery-operated devices. Furthermore, many MCUs incorporate memory protection mechanisms that enhance security by preventing unauthorized access and minimizing the risk of data corruption.
Despite its benefits, internal memory has some limitations. Its capacity is fixed and may be insufficient for applications requiring extensive storage or complex firmware. Expanding memory typically requires switching to a higher memory specification MCU. Additionally, embedded code memory used for program storage has limited write/erase cycles, necessitating careful management to ensure long-term reliability.

Ultimately, the decision to use only internal memory depends on the application's requirements in terms of performance, power efficiency, storage capacity, and cost. In many cases, a hybrid memory approach, combining internal and external resources, offers a more balanced solution.

For small to medium-scale applications, particularly those without demanding graphics or image processing needs, internal memory alone may be sufficient. With up to 1 MB of program memory and 2 MB of SRAM available, such systems can be effectively implemented using internal memory.

Internal Memory Type	Size	Usage description, Features
Code memory	1MB	To store the bootloader code, Program Code (Firmware), Read-only Data.
SRAM	1664KB	SRAM memory is used for storing data, variables, and buffers required for runtime operations. It allows the MCU to read, write, and manipulate data during program execution. Part of the internal memory is also used for the stack.
ITCM	128KB	Tightly coupled Memory for Instruction, ITCM, which is linked directly to the CPU0 core. ITCMs support critical routines, such as interrupt handling routines or real-time tasks, ensuring the highest level of responsiveness with no cache latency.
CTCM	64 KB	Tightly coupled Memory for code, CTCM, which is linked directly to the CPU1 core. CTCMs support critical routines, such as interrupt handling routines or real-time tasks, ensuring the highest level of responsiveness with no cache latency.
DTCM	128KB	Tightly coupled Memory for Data, DTCM, which is linked directly to the CPU0 core. For faster, predictable, and deterministic data access. Critical buffers can be placed in the DTCM. Systems with

		rapid context switching can leverage DTCM for storing and managing the context switching data efficiently.
STCM		Tightly coupled Memory for data, STCM, which is linked directly to the CPU1 core. For faster, predictable, and deterministic data access. Critical buffers can be placed in the STCM. Systems with rapid context switching can leverage STCM for storing and managing the context switching data efficiently.
Instruction Cache	16KB	To improve the execution speed and efficiency of the MCU by storing frequently accessed program instructions in a small, high-speed memory close to the CPU0 (CM85) core.
Code cache	16KB	To improve the execution speed and efficiency of the MCU by storing frequently accessed program code in a small, high speed memory close to the CPU1 (CM33) core.
Data Cache	16KB	Faster Data Access Reduce latency. Support write-back or write-through policies, allowing the CPU0 to efficiently manage data writes to external memory while maintaining data coherency.
S-Cache	16KB	Faster Data Access Reduce latency. Support write-back or write-through policies, allowing the CPU1 to efficiently manage data writes to external memory while maintaining data coherency.

3.2 External Memory Selection in System Design

Depending on the specific application requirements, a system may function efficiently using only internal memory or may necessitate a combination of internal and external memory. For applications with relatively modest code and data needs, internal memory typically provides adequate resources. However, in more demanding scenarios, such as those involving large codebases or intensive data processing, internal code flash and SRAM may prove insufficient.

For instance, in image processing or graphical applications utilizing graphics accelerator modules, the internal SRAM may only support frame buffers for lower-resolution displays. When working with high-resolution LCDs like XGA (1024 × 768) or higher, external SDRAM is often required to handle the larger frame buffer sizes. This hybrid memory approach helps maintain a balance between performance and cost.

In advanced graphical use cases, large image files can be stored externally in QSPI or OSPI flash memory, freeing up internal code flash for critical application code. This setup enhances storage capacity and flexibility. For added security, encrypted data can be stored in external memory and decrypted on-the-fly (DOTF) during read operations from QSPI/OSPI flash.

When incorporating external memory such as SDRAM or QSPI/OSPI flash, access latency becomes an important consideration, as it can impact system performance. To address this, performance can be further optimized using features like DMA transfers, instruction cache (I-Cache), and data cache (D-Cache).

Table 4. RA8 Dual-Core External Memory selection options

Internal Memory Type	Size	Usage description, Features
OSPI Flash memory interface*	256MB *2	EOBI OSPI memory interface can support up to 2 slots of 256MB devices. Which can be used for the OSPI flash
OSPI RAM memory interface*	256MB *2	EOBI OSPI memory interface can support up to 2 slots of 256MB devices. Which can be used for OSPI RAM
QSPI Flash memory interface*	256MB *2	EOBI OSPI memory interface can support up to 2 slots of 256MB devices. Which can be used for QSPI flash
SDRAM Memory	128 MB	Dedicated SDRAM memory interface to an external 8-bit, 16-bit, or 32-bit SDRAM device

CS Area	8 * 16 MB	8, 16MB slots of Address spaces are available for memory and memory-mapped peripherals as well. The address space for the CS0 – CS7 ranges from (0x60000000 – 0x67FFFFFF).

Note: *There are only 2 slots available here, and it can have either OSPI flash, OSPI RAM, or QSPI Flash when physically mounted.

4. TrustZone and Secure Memory Management

Arm TrustZone feature on RA8 dual-core divides the system into secure and non-secure parts. Secure applications can use both secure and non-secure resources. Non-secure applications can only use non-secure resources. Each one must use the correct address type based on its level.

The system also has two access levels: 1) Privileged and 2) Unprivileged. Privileged mode can access everything. Unprivileged mode can only access limited areas.

The TrustZone implementation for Armv8-M architecture includes two key components: 1) the Security Attribution Unit (SAU) and 2) the Implementation Defined Attribution Unit (IDAU). Together, they partition the 4GB addressable memory space into Secure (S) and Non-Secure (NS) regions.
Within the Secure region, memory is further categorized into:

- Secure (S): Reserved for memory and peripherals that can only be accessed by trusted (secure) software or secure masters.
- Non-Secure Callable (NSC): A specialized secure region that contains Secure Gateway (SG) instructions. These instructions enable controlled transitions from non-secure to secure states.
- Non-Secure (NS): Memory and peripherals accessible by both secure and non-secure software running on the system.

Implementation Defined Attribution Unit (IDAU)

The Implementation Defined Attribution Unit (IDAU) uses address bit [28] to distinguish between secure alias and non-secure alias regions for code, SRAM, and peripherals. Both the secure code and secure SRAM regions are assigned the Non-Secure Callable (NSC) security attributes. The memory security layout defined by the IDAU is hardcoded in hardware and cannot be modified through software.

Master Security Attribution Unit (MSAU)

The Memory Security Attribution Unit (MSAU) serves as the IDAU for non-CPU masters, defining the system specific security address mapping for these components. It distinguishes between secure and non-secure alias regions, but does not specify Non-Secure Callable (NSC) regions or assign region numbers. Non-CPU masters can perform secure transactions using secure alias addresses defined by the MSAU, but non-secure masters are restricted from accessing secure regions via secure alias addresses. The security map set by the MSAU is hardwired and cannot be altered through software.

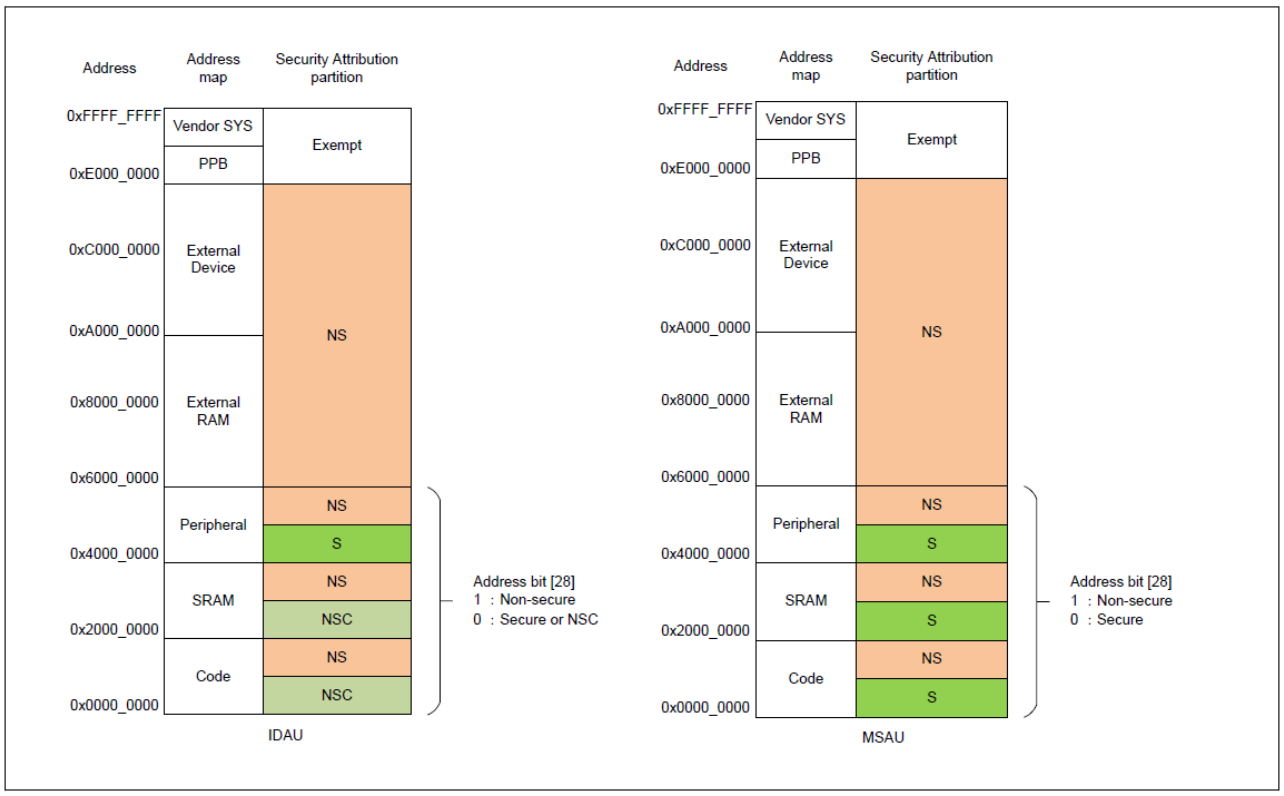


Figure 36. RA8 Dual-Core - IDAU and MSAU defined security map

Armv8-M TrustZone Security Overview

- Security Attribution Units:
  - IDAU and SAU (8 regions) are implemented to define secure/non-secure regions.
  - MSAU handles security for non-CPU masters.
- Memory Region Security Settings:
  - Code MRAM, SiP Flash, CM33 TCM, SRAM, and VBATT backup registers:
    Each supports up to 2 regions (secure and non-secure).
  - Peripherals:
    Security settings can be applied individually per unit or channel.
- Predefined Non-Secure Regions:
  - CSC, SDRAM, OPSI0, and OPSI1 (only in Standard products) are set as non-secure by default.

MRAM Protection for TrustZone

MRAM protection within TrustZone is available only on security-enabled devices. To comply with the TrustZone specification, MRAM implements four types of protection mechanisms that prevent unauthorized access from the non-secure domain. These protections focus on additional operations handled by the MRAM sequencer and include:

- Protection for the MRAM area (Program access)
- Protection for the extra MRAM area (Program access)
- Protection for the MRAM area (Read access)
- Protection for MRAM control registers

TrustZone Filter for SRAM

SRAM registers can be assigned a Security Attribution (SA) to define access as either secure or non-secure.

- If the SA marks the SRAM registers as secure, any non-secure access is blocked by the TrustZone Filter, which detects the violation and prevents access.
- Similarly, if the SA marks them as non-secure, any secure access is also blocked by the TrustZone Filter.

Note: A single SA setting applies commonly to all SRAM registers and it cannot be set individually for each.

SA	Transaction	Write access	Read access
Secure	Secure	Permitted	Permitted
	Non-secure	Protected (TrustZone Filter error)	Protected (TrustZone Filter error)
Non-secure	Secure	Protected (TrustZone Filter error)	Protected (TrustZone Filter error)
	Non-secure	Permitted	Permitted

Figure 37. Register Protection

SRAM0, SRAM1, SRAM2, and SRAM3 regions can each be independently configured as secure or non-secure. The access permissions for these regions are as follows. TrustZone Filter error for SRAM memory generates an error notification.

SA	Transaction	Write access	Read access
Secure	Secure	Permitted	Permit
	Non-secure	Protected (TrustZone Filter error)	Protected (TrustZone Filter error)
Non-secure	Secure	Protected (TrustZone Filter error)	Protected (TrustZone Filter error)
	Non-secure	Permitted	Permitted

Figure 38. Memory protection

Note: More details on the usage of TrustZone and memory partition in application development using dual-core can be found in “TrustZone Application Note”

5. Low Power Mode and Memory Management

RA8 dual-core MCUs offer various low-power modes designed to reduce power consumption. In software standby mode, the MCU core halts, peripherals are disabled, and clock sources are shut down. Despite this, the MCU remains powered, allowing it to retain critical state information such as register contents and on-chip SRAM data. This mode offers quick wake-up times, though it does not achieve the absolute minimum power consumption.

In contrast, deep software standby mode powers down the entire MCU, including the on-chip SRAM, enabling the lowest possible power usage. However, this also means that the SRAM contents are lost upon entering this mode. To preserve the MCU’s state, the data must first be checkpointed and saved to non-volatile memory before entering deep sleep. Upon waking, the system restores this data back into SRAM, allowing execution to resume. However, due to the relatively high latency and energy cost of flash memory operations, this approach introduces additional overhead.

5.1 Operating State of Memory Modules in Low Power Mode

Table 5. State of RA8 Dual-Core - Memory Modules in low power modes

Module	Software Standby Mode(SSTBY)	Deep Software Standby Mode (DSTBY)		
	SSTBY	DSTBY1	DSTBY2	DSTBY3
External Bus (EBCLK)	Stop (Retained)	Stop (Retained)		
CPU	Stop (Retained)	Stop (Undefined)		
TCM (SRAM)	Stop (Retained)*15	Stop (Undefined)		
User SRAM	Stop (Retained)*13	Stop (Undefined)		
Backup register	Stop (Retained)	Stop (Retained)		
MRAM	Stop (Retained)	Stop (Retained)		

Note 13. If PDRAMSCR0.RKEEPn bit is set to 0, the contents of the target User SRAM are not retained.
Note 14. If PDRAMSCR1.RKEEPn bit is set to 0, the contents of the target TCM are not retained.

5.2 Operating State of Processor Low Power Mode

Table 6. State of RA8 Dual-Core Processor in low power modes

Item	CPUn Sleep mode		CPUn Deep Sleep mode	
	n = 0 (CPU0)	n = 1 (CPU1)	n = 0 (CPU0)	n = 1 (CPU1)
Transition condition	WFI instruction after set CPU0.SCR. SLEEPDEEP = 0.	WFI instruction after set CPU1.SCR. SLEEPDEEP = 0.	WFI instruction after set CPU0.SCR. SLEEPDEEP = 1	WFI instruction after set CPU1.SCR. SLEEPDEEP = 1
Canceling method	All interrupts. Any reset available in the mode.	All interrupts. Any reset available in the mode.	All interrupts. Any reset available in the mode.	All interrupts. Any reset available in the mode.
State after cancellation by an interrupt	Program execution state (interrupt processing)	Program execution state (interrupt processing)	Program execution state (interrupt processing)	Program execution state (interrupt processing)
State after cancellation by a reset	Reset state	Reset state	Reset state	Reset state
CPU0	Stop (Retained)	—	Stop (Retained)	—
CPU0 TCM (SRAM)	—	—	Stop (Retained)*4	—
DMA Controller 0 (DMAC0)	Selectable	—	Selectable	—
Data Transfer Controller 0 (DTC0)	Selectable	—	Selectable	—
Watchdog Timer 0 (WDT0)	Selectable*1	—	Selectable*1	—
CPU1	—	Stop (Retained)	—	Stop (Retained)
DMA Controller 1 (DMAC1)	—	Selectable	—	Selectable

5.3 Power Gating and Clock Gating for Memory Modules

Power consumption can be reduced in multiple ways, including setting clock dividers, controlling EBCLK output, controlling SDCLK output, stopping modules, power gating control, selecting operating power control modes in normal operation, and transitioning to low power modes and processor low power modes.
SRAM Power Domain Standby Control Register PDRAMSCR0/1 is used for configuring the RAM retention.
Module Stop Control Register MSTPCRA/B/C/D/E has bits to control the module stop functionality for memory and other peripherals.

Table 7. State of RA8 Dual-Core - Memory Modules in low power modes

Parameter	Specification
Reducing power consumption by switching clock signals	The frequency division ratio can be selected independently for the system clock, peripheral module clock, external bus clock, and MRAM interface clock.*1
EBCLK output control	BCLK output or high-level output can be selected.
SDCLK output control	SDCLK output or high-level output can be selected.
Module-stop	Functions can be stopped independently for each peripheral module.
Power gating control	This function can be controlled the power state of the power domain. • Control the turning On/OFF for the power domain • Control the retention of specific circuits during power gating
Processor low power modes	• CPU Sleep mode • CPU Deep Sleep mode
Low power modes	• Software Standby mode*2 • Deep Software Standby mode1, 2, 3*2

Note 1. For details, see section 9, Clock Generation Circuit.
Note 2. This mode is not supported in external VDD mode.

5.4 Low Power Modes and SDRAM

Before entering Software Standby or Deep Software Standby, the R_BSP_SdramSelfRefreshEnable() function must be called to switch the SDRAM into Self-Refresh mode, preserving its contents during the low-

power state. After this call, SDRAM must not be accessed, as doing so will cause a fault. To avoid issues during low-power transitions, users should not place FSP code, data, or wakeup interrupt handlers in SDRAM. When exiting Software Standby, call R_BSP_SdramSelfRefreshDisable() to switch back to Auto-Refresh mode and re-enable SDRAM access.

If resuming from Deep Software Standby (or any condition where SDRAM contents must be retained), users should first call R_BSP_SdramInit(false) before pin initialization, and R_BSP_SdramSelfRefreshDisable() after pin configuration to fully restore SDRAM operation.

6. Configuring the Dual-Core MCU Memory Using FSP Configurator

The Renesas FSP offers intuitive memory configuration support through its configuration tools. Users can easily configure the various memory types such as Option Memory, SDRAM, Cache Memory, OSPI memory, and external bus for memory-based access. Additionally, for RA8 dual-core, the **solution.xml** file created as part of the “Renesas RA FSP Solution” project creation provides a graphical representation of available memory. This graphical interface allows developers to assign and configure memory regions for each core, making it easier to manage and optimize memory usage in dual-core applications. This user-friendly approach greatly simplifies memory setup and enhances usability for developers working with complex memory architectures.

6.1 FSP-based Memory Configuration for RA Dual-Core MCU

The BSP tab in the FSP Configurator allows developers to easily configure SDRAM settings, including timing parameters, initialization options, support for enable/disable, endian mode, continuous access mode, and bus width selection. In addition, optional memory settings can be customized using dedicated Optional Memory Register configurations. The configurator tool also provides control to enable or disable the data cache, enhancing performance tuning. For application development, the configurator offers options to define stack and heap sizes, ensuring memory allocation tailored to your application’s needs.

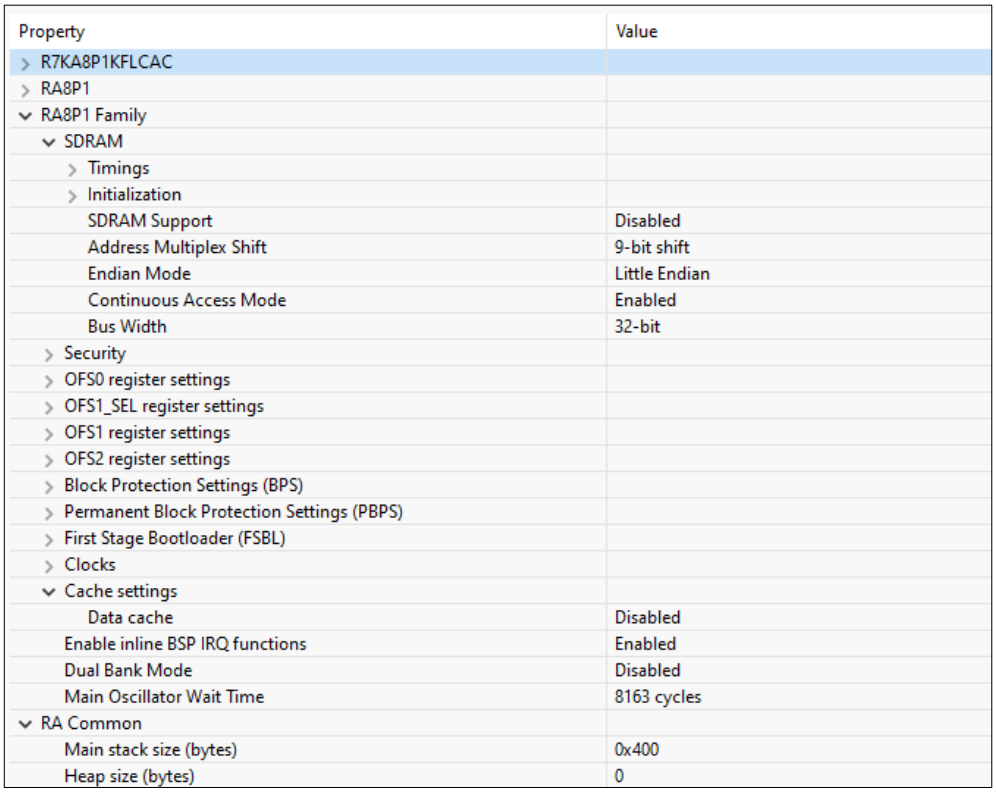


Figure 39. Memory configuration using BSP Tab

Also under the Stacks tab, the OSPI memory can be configured with features such as memory prefetch, XiP (Execute in Place), DMAC, autocalibration, and DOTF support. The configurator also enables detailed HAL driver settings, including mode selection, channel configuration, XiP entry/exit code setup, and decryption-related settings like DOTF key, key length, key format, and decryption address range. These options offer

fine-grained control for supported OSPI memories, as illustrated in the snapshot below. For more details, refer to the Example project for the OSPI and Application Project for the OSPI with DOTF.

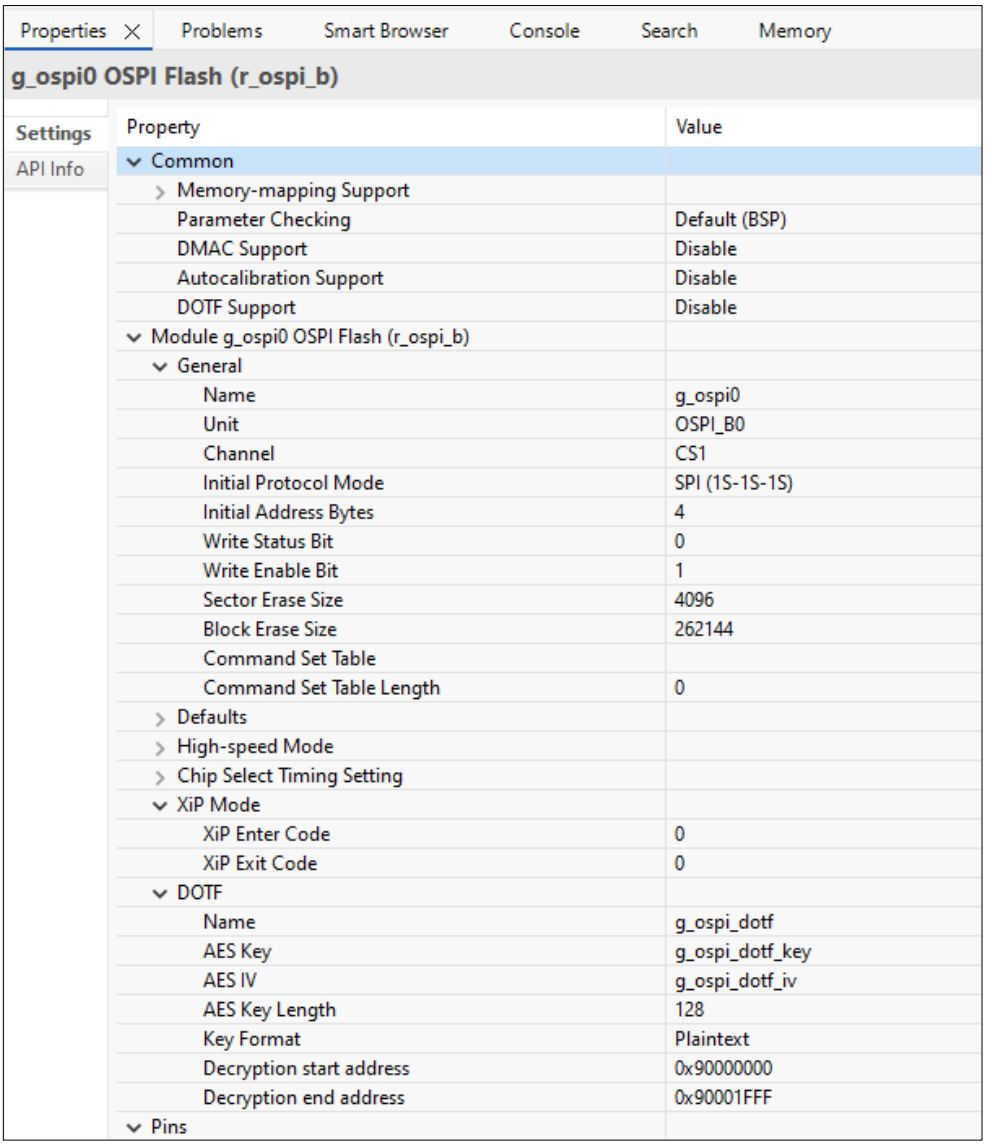


Figure 40. OSPI Memory configuration using Stacks Tab

Additionally, the Pins tab allows configuration of settings related to external memory interfaces such as SDRAM, OSPI, and the external memory bus. Users can assign pins for address and data buses, chip select, clock, read/write signals, and other memory-specific functions as needed.

6.2 SDRAM Memory Startup Configuration and Initialization

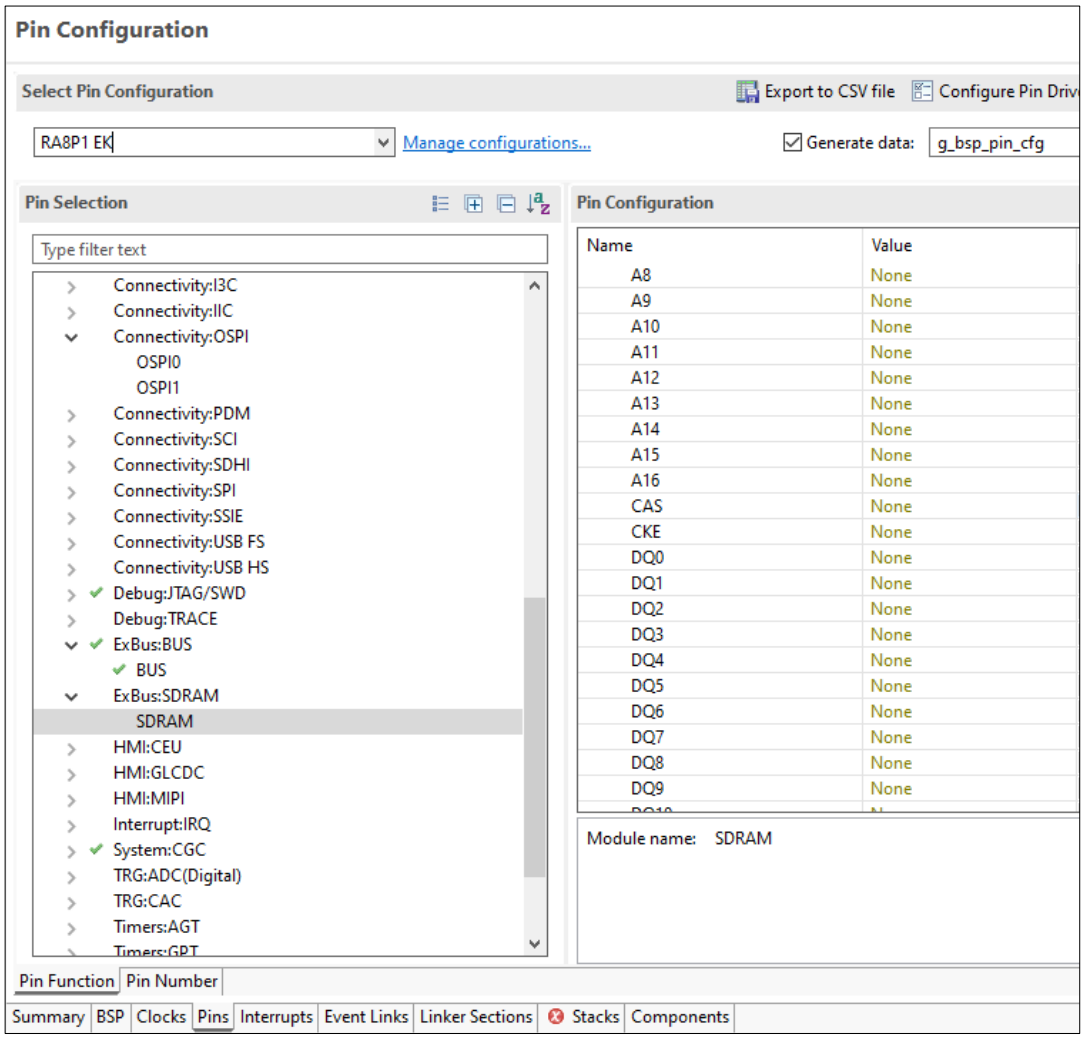


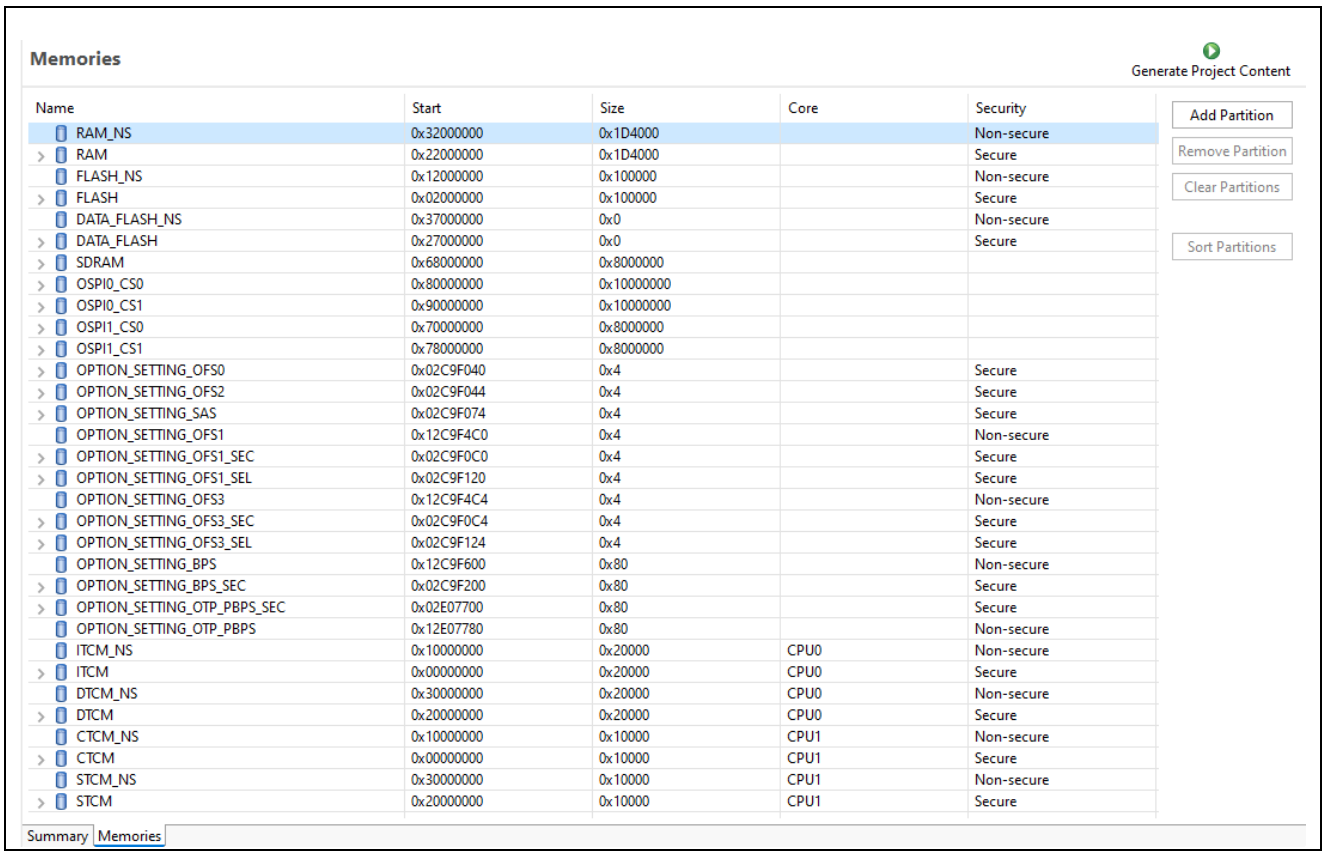
Figure 41. SDRAM Memory configuration using Pins Tab

The Board Support Package (BSP) supports the use of external SDRAM modules on the RA8 dual-core that include SDRAM functionality. SDRAM can be enabled and configured through the BSP tab in the RA Configuration Editor. By default, initialization is handled in the 'Post C' warm start callback. However, the call to R_BSP_SdramInit() can be moved elsewhere in the code as long as it occurs after clock and pin initialization, and only once per reset.

The BSP does not initialize any memory sections within SDRAM. It is the user's responsibility to initialize any code or data stored in SDRAM.

6.3 Dual-Core Memory Selection and its Configuration

e² studio provides a GUI-based memory configuration option where users can add/modify the memory partition for the 2 cores and the application that requires selectable memory areas.



Name	Start	Size	Core	Security
RAM_NS	0x32000000	0x1D4000		Non-secure
RAM	0x22000000	0x1D4000		Secure
FLASH_NS	0x12000000	0x100000		Non-secure
FLASH	0x02000000	0x100000		Secure
DATA_FLASH_NS	0x37000000	0x0		Non-secure
DATA_FLASH	0x27000000	0x0		Secure
SDRAM	0x68000000	0x8000000		
OSPI0_CS0	0x80000000	0x10000000		
OSPI0_CS1	0x90000000	0x10000000		
OSPI1_CS0	0x70000000	0x8000000		
OSPI1_CS1	0x78000000	0x8000000		
OPTION_SETTING_OFS0	0x02C9F040	0x4		Secure
OPTION_SETTING_OFS2	0x02C9F044	0x4		Secure
OPTION_SETTING_SAS	0x02C9F074	0x4		Secure
OPTION_SETTING_OFS1	0x12C9F4C0	0x4		Non-secure
OPTION_SETTING_OFS1_SEC	0x02C9F0C0	0x4		Secure
OPTION_SETTING_OFS1_SEL	0x02C9F120	0x4		Secure
OPTION_SETTING_OFS3	0x12C9F4C4	0x4		Non-secure
OPTION_SETTING_OFS3_SEC	0x02C9F0C4	0x4		Secure
OPTION_SETTING_OFS3_SEL	0x02C9F124	0x4		Secure
OPTION_SETTING_BPS	0x12C9F600	0x80		Non-secure
OPTION_SETTING_BPS_SEC	0x02C9F200	0x80		Secure
OPTION_SETTING_OTP_PBPS_SEC	0x02E07700	0x80		Secure
OPTION_SETTING_OTP_PBPS	0x12E07780	0x80		Non-secure
ITCM_NS	0x10000000	0x20000	CPU0	Non-secure
ITCM	0x00000000	0x20000	CPU0	Secure
DTCM_NS	0x30000000	0x20000	CPU0	Non-secure
DTCM	0x20000000	0x20000	CPU0	Secure
CTCM_NS	0x10000000	0x10000	CPU1	Non-secure
CTCM	0x00000000	0x10000	CPU1	Secure
STCM_NS	0x30000000	0x10000	CPU1	Non-secure
STCM	0x20000000	0x10000	CPU1	Secure

Figure 42. RA8 Dual-Core - Memory Partition using IDE

The FSP Configurator offers a graphical interface to visualize and configure memory allocation for dual-core MCUs, supporting both CPU0 and CPU1. The generated solution.xml from the solution project can be opened to view and modify it. Users can specify the start address and size of memory regions assigned to each core individually.
The configuration view includes memory types such as SRAM, FLASH (MRAM), SDRAM, Option Settings, QSPI, OSPI, ITCM, and DTCM, all categorized into secure and non-secure regions and their accessibility by each core.
Before building the solution or application, users can define how memory is divided between CPU0 and CPU1. These settings form the foundation for code compilation and linking, with the generated code referencing the appropriate linker scripts for each core within their respective projects.

6.4 Ensuring Data Coherency in a RA8 Dual-Core

In RA8 dual-core, shared memory regions are areas of memory accessible by CPU0 and CPU1 cores. When these cores read and write to the same memory region simultaneously, race conditions can occur, this means the system's behavior depends on the unpredictable timing of those accesses, potentially leading to data corruption or inconsistent results. To prevent this, careful management is needed through:

- Synchronization mechanisms like mutexes, semaphores, or inter-core interrupts to ensure that only one core accesses the shared region at a time or that accesses are properly ordered.
- Atomic operations, where possible, to perform read-modify-write actions safely.

These techniques are essential during inter-core communication, where consistency and correctness of shared data are critical.

7. Inter-processor communication (IPC) and Shared memory

In RA8 dual-core, IPC enables efficient data exchange and synchronization between the two cores typically CPU0 and CPU1. IPC mechanisms are essential for sharing resources and ensuring real-time responsiveness in multicore systems. IPC in RA8 dual-core includes features like shared memory, hardware semaphores, interrupts, and message passing. These allow the cores to communicate without interfering with each other's execution, while maintaining data integrity and system performance. IPC is especially important in real-time embedded applications where workload distribution and inter-core coordination are critical.

7.1 Communication Mechanisms Between Cores

To ensure exclusive access to shared memory, IPC provides multiple semaphore-based locking mechanisms. Developers can choose to implement either separate or shared hardware semaphores between the two CPUs, depending on the application's needs.

For efficient data exchange, IPC also includes mechanisms like FIFO buffers. For example, CPU0 can use a write-only FIFO while CPU1 reads from it, and vice versa. This allows straightforward, unidirectional data transfer between cores.

7.2 Shared Memory Settings and Limitations

The IPC has the registers to perform a semaphore for synchronization and exclusive control between CPU0 and CPU1. This register only indicates status and does not have any hardware protection of shared memory. Thus, exclusive control must be done by software that uses this register.

The security of this register depends on IPCSAR. The privilege of this register depends on IPCPAR.
Note: For more details of the IPC and shared memory, please refer to the App note "*Inter-processor Communication in RA8 Dual-Core MCUs*"

8. How to interpret the electrical characteristics of the RA8 Dual-Core MCU

The electrical characteristics of the RA8 dual-core MCU and its peripherals are crucial considerations when designing embedded systems. These characteristics include voltage levels, current requirements, timing specifications, and operating temperature, etc. These electrical characteristics can impact system performance, reliability, and compatibility. HW UM contains a detailed table for electrical characteristics. Here in this section, some common electrical characteristics used for the MCU and its peripherals are explained.

Voltage Levels: Supply Voltage (**Vcc/Vdd**): The voltage level required to power the RA8 dual-core and its peripherals. It is usually specified in volts (V) and should be within the MCU's acceptable range.

I/O Voltage Levels: **VIH, VIL. VOH, VIL** are the voltage levels at which the MCU's input and output pins operate. RA8 dual-core MCU supports different I/O voltage levels (e.g., 3.3V or 5V). It's important to ensure compatibility with external devices.

Current Requirements: Supply Current (**Icc/Idd**): The current consumed by the MCU and its peripherals during operation. This includes both static (quiescent) and dynamic (active) current.

I/O Pin Current: The maximum current that can be sourced or sunk by each I/O pin. This specification is crucial for driving external devices or interfacing with other components.

Output Drive Strength: Output Current Drive: The maximum current that an I/O pin can source or sink. Higher drive strength is needed for driving loads with higher impedance.

Timing Specifications:

Setup and Hold Times: Timing requirements for data setup and hold times, particularly relevant for synchronous communication interfaces like SPI/QSPI/OSPI and I2C.

Access Time: The time it takes for the memory to respond to a read or write operation. Access time can affect the MCU's overall performance, especially in real-time applications.

Power Management: Low-Power Modes:- Information about different low-power modes and their associated current consumption.

IO Schmitt Trigger Inputs: Specifications for Schmitt trigger inputs, which are commonly used for noisy input signals.

Erase Time: The time it takes to erase a block or sector of Flash memory. Erase times can vary depending on the MCU and the memory technology used.

Operating Temperature Range: The range of temperatures within which the MCU and its peripherals are guaranteed to operate reliably.

8.1 How to read the Electrical Characteristics specified in the MCU UM

RA8 dual-core MCU UM has a section for electrical characteristics, which comprises many categories. In this document, we have given a brief description of the electrical characteristics and how to read and infer from the data provided in the UM.

- Absolute Maximum Ratings
- DC Characteristics
  - Tj/Ta Definition
  - I/O V_{IH}, V_{IL}
  - I/O I_{OH}, I_{OL}
  - I/O V_{OH}, V_{OL}, and other characteristics
  - Operating and Standby Current
  - VCC Rise and Fall Gradient and Ripple Frequency
  - Thermal Characteristics
- AC Characteristics
  - Frequency
  - Clock timing
  - Reset timing
  - Wakeup Timing
  - Bus Timing
  - OSPI Timing and many more
- Peripheral (such as USB, MIP ADC, flash memory) characteristics and so on.

Absolute Maximum Ratings: Specifications in the electrical characteristics of an MCU that define the maximum limits of various electrical and environmental parameters beyond which the MCU may be damaged or may not operate correctly. For the RA8 dual-core, the Absolute Maximum ratings are given below. Here, you can see that in a typical VCC of 3.3V, for the MCU, the VCC value can go up to 4.0V; above 4.0V can damage the device. Similarly, the operating and Storage temperatures can be in the range of -55 to 125 °C. Similarly, the absolute rating for other input voltage sources is also listed in the Table 8.

Table 8. RA8 Dual-Core Absolute Maximum Ratings

Parameter	Symbol		Value	Unit
Power supply voltage	VCC, VCC_DCDC*2		−0.3 to +4.0	V
	VCC2	Standard Product	−0.3 to +4.0	V
		SiP Product	−0.3 to +2.5	V
External power supply voltage	VCL		−0.3 to +1.2	V
VBATT power supply voltage	VBATT		−0.3 to +4.0	V
Input voltage (except for 5 V-tolerant ports *1)	V_{in}		−0.3 to VCC + 0.3, −0.3 to VCC2 + 0.3, −0.3 to VCC_USB + 0.3 or −0.3 to VBATT_R + 0.3	V
Input voltage (5 V-tolerant ports *1)	V_{in}		−0.3 to + VCC + 4.0 (max. 5.8) or −0.3 to + VCC2 + 4.0 (max. 5.8)	V
Reference power supply voltage	VREFH/VREFH0		−0.3 to AVCC0 + 0.3	V
USBFS power supply voltage	VCC_USB		−0.3 to +4.0	V
USBHS power supply voltage	VCC_USBHS		−0.3 to +4.0	V
USBHS analog power supply voltage	AVCC_USBHS		−0.3 to +4.0	V
MIPI PHY analog power supply voltage	AVCC_MIPI		−0.3 to +4.0	V
MIPI PHY power supply voltage	VCC18_MIPI		−0.3 to +2.5	V
Analog power supply voltage	AVCC0		−0.3 to +4.0	V
Analog input voltage	V_{AN}		−0.3 to AVCC0 + 0.3	V
Operating junction temperature *3 *4 *5	T_{opj}		0 to 95 or −40 to +105	°C
Storage temperature	T_{stg}		−55 to +125	°C

Note 1. Ports P204, P205, P303, P407 to P413, P511, P512, P514, P515 and P708 to P715 are 5 V tolerant.
Note 2. Connect VCC_DCDC to VCC.
Note 3. See section 70.2.1. Tj/Ta Definition.
Note 4. Contact a Renesas Electronics sales office for information on derating operation when Tj = +95 °C to +125 °C. Derating is the systematic reduction of load for improved reliability.
Note 5. The lower and upper limit of operating junction temperature depend on the product. For details, see section 1.3. Part Numbering.

Even though the MCU can take up to Absolute maximum ratings, for some peripherals, in order to perform reliably, it is also highly recommended to operate the MCU within the recommended operating voltage levels. Table 9 shows the recommended operating voltage ranges for the ETHERC/IIC/USB/SDRAM.

Table 9. RA8 Dual-Core Recommended Operating Rating

Parameter	Symbol			Min	Typ	Max	Unit
Power supply voltages	VCC, VCC_DCDC	Other than the following		1.62	—	3.63	V
		When ESWM is used		2.30	—	3.63	V
		When SDRAM is used		3.00	—	3.63	V
	VCC2	Standard Product	Other than the following	1.62	—	3.63	V
			When ESWM is used	2.30	—	3.63	V
			When SDRAM is used	3.00	—	3.63	V
		SiP Product		1.70	—	2.00	V
	VCL	When external VDD is used*2	voltage range 1	0.92	—	0.99	V
			voltage range 2	0.87	—	0.99	V
		When DCDC is used (High-speed mode)	VSCR_1	—	0.95	—	V
			VSCR_2	—	0.925	—	V
		When DCDC is used (Software Standby mode)	SVSCR_1	—	0.95	—	V
			SVSCR_2	—	0.925	—	V
			SVSCR_3	—	0.825	—	V
			SVSCR_4	—	0.765	—	V
			SVSCR_5	—	0.715	—	V
	VSS, VSS_DCDC			—	0	—	V
USB power supply voltages	VCC_USB, VCC_USBHS, AVCC_USBHS	When USB is not used		1.62	—	3.63	V
		When USB is used		3.00	—	3.60	V
	VSS_USB, VSS1_USBHS, VSS2_USBHS			—	0	—	V
MIPI PHY power supply voltages	VCC18_MIPI			1.65	1.80	1.95	V
	AVCC_MIPI			2.90	—	3.60	V
	VSS_MIPI			—	0	—	V
VBATT power supply voltage	VBATT			1.62	—	3.63	V
Analog power supply voltages	AVCC0*1	Other than the following		1.62	—	3.63	V
		When Channel dedicated sample-and-hold circuit is used		2.70	—	3.63	V
	AVSS0			—	0	—	V

Note 1. When the A/D converter, the D/A converter and the High-Speed Analog Comparator are not in use, do not leave the AVCC0, VREFH/VREFH0, AVSS0, and VREFL/VREFL0 pins open. Connect the AVCC0 and VREFH/VREFH0 pins to VCC, and the AVSS0 and VREFL/VREFL0 pins to VSS, respectively.
Note 2. VCL voltage must never be higher than VCC voltage.

DC Characteristics

Tj/Ta Definition The permissible operating junction temperature (also referred to as **Tj** or **Tj(max)**) is an important parameter in the DC characteristics of an MCU. It defines the maximum temperature at which the internal semiconductor components of the MCU can operate safely and reliably. Exceeding this temperature can lead to thermal stress, reduced performance, and potential damage to the MCU.

Table 10. RA8 Dual-Core Tj/Ta ratings

Parameter	Symbol	Typ	Max	Unit	Test conditions
Permissible operating junction temperature	T_j	—	125*1	°C	High-speed mode

I/O V_{IH}, V_{IL}: V_{IH} (Voltage Input High) and **V_{IL}** (Voltage Input Low) refer to the voltage levels that define the logical high and logical low states for a digital input signal. These voltage levels are crucial for ensuring reliable communication between digital devices.

Table 11. I/O V_{IH}, V_{IL} ratings

Parameter		VCC/VCC2/ AVCC0/ VCC_USB	Symbol	Min	Typ	Max	Unit
Peripheral function pin	EXTAL (external clock input), WAIT, SPI*1 (except RSPCK)	1.62 V or above	V_{IH}	VCC × 0.8	—	—	V
			V_{IL}	—	—	VCC × 0.2	
	SPI*2 (except RSPCK)	1.62 V or above	V_{IH}	VCC2 × 0.8	—	—	
			V_{IL}	—	—	VCC2 × 0.2	
	OSPI (except OM_0_RSTO1, OM_0_ECSINT 1, OM_1_RSTO1 and OM_1_ECSINT 1)	2.70 V or above	V_{IH}	VCC2 × 0.8	—	—	
			V_{IL}	—	—	VCC2 × 0.2	
		1.62 to 2.00V	V_{IH}	VCC2 × 0.7	—	VCC2 + 0.3	
			V_{IL}	VSS − 0.3	—	VCC2 × 0.3	
	SD*3	2.70 V or above	V_{IH}	VCC × 0.625	—	VCC + 0.3	
			V_{IL}	VSS − 0.3	—	VCC × 0.25	
		1.70 to 1.95 V	V_{IH}	1.27	—	2	
			V_{IL}	VSS − 0.3	—	0.58	
	SD*4	2.70 V or above	V_{IH}	VCC2 × 0.625	—	VCC2 + 0.3	
			V_{IL}	VSS − 0.3	—	VCC2 × 0.25	
		1.70 to 1.95 V	V_{IH}	1.27	—	2	
			V_{IL}	VSS − 0.3	—	0.58	

Here, in the case of OSPI Peripheral for **V_{IH}**, the minimum Voltage to recognize it as high is 0.8 times VCC2, and for **V_{IL}**, it is 0.2 times the VCC2

I/O I_{OH}, I_{OL}: I_{OH} (Output High Current) and **I_{OL}** (Output Low Current) are electrical characteristics that specify the maximum current a digital output pin can source (**I_{OH}**) or sink (**I_{OL}**) while maintaining proper voltage levels.

Table 12. I/O I_{OH}, I_{OL} ratings

Parameter			VCC/VCC2/ AVCC0/ VCC_USB	Symbol	Min	Typ	Max	Unit
Permissible output current (average value per pin)	Ports P000 to P015, P201	—	—	I_{OH}	—	—	−2.0	mA
				I_{OL}	—	—	2.0	mA
	Ports P204, P205, P303, P407 to P413, P511, P512, P514, P515, P708 to P715, PA15 (total 23 pins)	Low drive*1	—	I_{OH}	—	—	−2.0	mA
				I_{OL}	—	—	2.0	mA
		Middle drive*2	—	I_{OH}	—	—	−4.0	mA
				I_{OL}	—	—	4.0	mA
		High drive*3	—	I_{OH}	—	—	−16	mA
				I_{OL}	—	—	20.0	mA
		High-speed high drive*4	—	I_{OH}	—	—	−20	mA
				I_{OL}	—	—	20.0	mA
	Other output pins*5	Low drive*1	—	I_{OH}	—	—	−2.0	mA
				I_{OL}	—	—	2.0	mA
		Middle drive*2	—	I_{OH}	—	—	−4.0	mA
				I_{OL}	—	—	4.0	mA
		High drive*3	—	I_{OH}	—	—	−16	mA
				I_{OL}	—	—	16.0	mA
		High-speed high drive*4	—	I_{OH}	—	—	−20	mA
				I_{OL}	—	—	20.0	mA

Here, in the case of port output as an example, when configured as Low drive, IOH is -2.0mA and IOL is 2.0mA; for Port configured as Middle drive, **I_{OH}** is -4.0mA and **I_{OL}** is 4.0mA. For a port configured as High drive, the IOH is -16mA and the **I_{OL}** is 16.0mA.

Caution: To protect the reliability of the MCU, the output current values should not exceed the values in this Table 12. The average output current indicates the average value of the current measured during 100 µs.

RA8 dual-core MCU UM provides details on the permissible output current for various ports and pins. For specific details on the ports and other peripheral pins, refer to the UM.

I/O V_{OH}, V_{OL}: V_{OH} (Output High Voltage) and **V_{OL}** (Output Low Voltage) are electrical characteristics that specify the voltage levels of a digital output signal when it is in the high (**V_{OH}**) or low (**V_{OL}**) logic state.

Here, in the case of the SD peripheral as an example for **V_{OL}**, the minimum Voltage to recognize it as logic low when VCC is 2.70V or higher, it is 0.125 times the VCC for (I_{OL} = 3.0mA), and for V_{OH} it is 0.75 times the VCC for (I_{OL} = -2.0mA). In the case of VCC between 1.70V and 1.95V, VOH is 1.4V (with IOL = -2.0mA), and it is 0.45V for (I_{OL} = 2.0mA).

Table 13. I/O V_{OH}, V_{OL} ratings

Parameter		VCC/VCC2/ AVCC0	Symbol	Min	Typ	Max	Unit	Test conditions
Output voltage	IIC	2.70 V or above	V_{OL}	—	—	0.4	V	I_{OL} = 3.0 mA
			V_{OL}	—	—	0.6		I_{OL} = 6.0 mA
		1.68 V or above	V_{OL}	—	—	VCC × 0.2		I_{OL} = 3.0 mA
			V_{OL}	—	—	0.6		I_{OL} = 6.0 mA
	SD	2.70 V or above	V_{OH}	VCC × 0.75	—	—		I_{OH} = −2.0 mA
			V_{OL}	—	—	VCC × 0.125		I_{OL} = 3.0 mA
			V_{OH}	VCC2 × 0.75	—	—		I_{OH} = −2.0 mA
			V_{OL}	—	—	VCC2 × 0.125		I_{OL} = 3.0 mA
		1.70 V to 1.95 V	V_{OH}	1.4	—	—		I_{OH} = −2.0 mA
			V_{OL}	—	—	0.45		I_{OL} = 2.0 mA

Operating and Standby Current

Operating current is the current drawn by the device during normal operation, when it is executing tasks, processing data, and actively communicating with other components or devices. The operating current can vary depending on factors such as clock speed, workload, input/output activity, and other operational parameters.

Standby current, also known as quiescent current or sleep current, is the current consumed by a device or component when it is in a low-power or idle state. In this state, the device is not actively performing its primary functions but is still operational and ready to be awakened or respond to an external event.

Balancing the need for high performance during active operation with minimal power consumption during standby or idle periods is a fundamental aspect of energy-efficient system design. The RA8 dual-core UM gives details of the Operating and Standby current for different modes.

In the below

Table 14 You can see the operating current for different clock frequencies with VCC_DCDC greater than 2.5V and less than 2.5V. The higher the operating frequency, the higher the current consumption. From the table, you can get the approximate current consumption for your system and the design. For calculating this current consumption, there are other parameters that come into the picture. For more details and specific current consumption, users are required to refer to the MCU UM.

Table 14. Current of high-speed mode, maximum condition (DCDC mode)

Parameter			Symbol	Typ	Max		Unit	Test conditions
					95 °C	105 °C		
Supply current *1*2*6	—		I_{CC}	3.85	6.27	6.69	mA	—
	CPUCLK0 = 1 GHz CPUCLK1 = 250 MHz VSCR_1	VCC_DCDC ≥ 2.5 V	I_{CC_DCDC} *4	205	390	—	mA	VCC_DCDC = 3.3 V NPUCLK = 500 MHz, MRICLK = 250 MHz, MRPCLK = 125 MHz, ICLK = 250 MHz, BCLK = 125 MHz, PCLKA = 125 MHz, PCLKB = 62.5 MHz, PCLKC = 125 MHz, PCLKD = 250 MHz, PCLKE = 250 MHz
			I_{DD} *3	525	1000 *5	—		
		VCC_DCDC < 2.5 V	I_{CC_DCDC} *4	400	760	—		VCC_DCDC = 1.8 V Clock settings are the same as above
			I_{DD}	525	1000 *5	—		
	CPUCLK0 = 800 MHz CPUCLK1 = 200 MHz VSCR_1	VCC_DCDC ≥ 2.5 V	I_{CC_DCDC} *4	171	—	390	mA	VCC_DCDC = 3.3 V NPUCLK = 400 MHz, MRICLK = 200 MHz, MRPCLK = 100 MHz, ICLK = 200 MHz, BCLK = 100MHz, PCLKA = 100 MHz, PCLKB = 50 MHz, PCLKC = 100 MHz, PCLKD = 200 MHz, PCLKE = 200 MHz
			I_{DD} *3	438	—	1000 *5		
		VCC_DCDC < 2.5 V	I_{CC_DCDC} *4	333	—	760		VCC_DCDC = 1.8 V Clock settings are the same as above
			I_{DD}	438	—	1000 *5		
	CPUCLK0 = 600 MHz CPUCLK1 = 150 MHz VSCR_2	VCC_DCDC ≥ 2.5 V	I_{CC_DCDC} *4	133	313	344	mA	VCC_DCDC = 3.3 V NPUCLK = 300 MHz, MRICLK = 150 MHz, MRPCLK = 75 MHz, ICLK = 150 MHz, BCLK = 75 MHz, PCLKA = 75 MHz, PCLKB = 37.5 MHz, PCLKC = 75 MHz, PCLKD = 150 MHz, PCLKE = 150 MHz
			I_{DD} *3	348	821	901		
		VCC_DCDC < 2.5 V	I_{CC_DCDC} *4	260	612	672		VCC_DCDC = 1.8 V Clock settings are the same as above
			I_{DD}	348	821	901		

Bus Timing: The timing specifications for communication and data transfer between peripheral devices or components on a bus.

The table below shows the bus timings for the various parameters with different voltage level conditions.

Table 15. Bus timings with different conditions

Condition 1: When using the CS area controller (CSC).	
	VCC = VCC_DCDC = VCC_USB = VBATT = 1.68 V to 3.6 V, VCC2 = 1.65 V to 3.63 V
	BCLK = 8 to 120 MHz, BCLKA = 8 to 120 MHz, EBCLK = 8 to 120 MHz (When VCC = VCC_USB = VBATT = 2.70 to 3.63 V)
	BCLK = 8 to 60 MHz, EBCLK = 8 to 30 MHz (When VCC = VCC_USB = VBATT = 1.68 to 3.63 V)

	Output load conditions: VOH = VCC × 0.5, VOL = VCC × 0.5, C = 30 pF
	EBCLK: High drive output is selected in the port drive capability bit in the PmnPFS register.
	Others: Middle drive output is selected in the port drive capability bit in the PmnPFS register.

Condition 2: When using the SDRAM area controller (SDRAMC).	
	BCLK = SDCLK = 8 to 125 MHz, BCLKA = SDCLK = 8 to 133 MHz
	VCC = VCC2 = VCC_DCDC = VCC_USB = VBATT = 3.0 to 3.63 V, VCC2 = 1.62V to 3.63V
	Output load conditions: VOH = VCC × 0.5, VOL = VCC × 0.5, C = 15 pF
	SDCLK: High-speed high drive output is selected in the port drive capability bit in the PmnPFS register.
	Others: High drive output is selected in the port drive capability bit in the PmnPFS register.

Condition 3: When using the SDRAM area controller (SDRAMC) and CS area controller (CSC) simultaneously.	
	BCLK = SDCLK = 8 to 66 MHz, BCLKA = SDCLK = 8 to 66 MHz
	VCC = VCC2 = VCC_DCDC = VCC_USB = VBATT = 3.0 to 3.63 V, VCC2 = 1.62V to 3.63V
	Output load conditions: VOH = VCC × 0.5, VOL = VCC × 0.5, C = 15 pF
	EBCLK/SDCLK: High drive output is selected in the port drive capability bit in the PmnPFS register.
	Others: Middle drive output is selected in the port drive capability bit in the PmnPFS register.

The typical bus timings and their acceptable ranges are given in the following table. This timing data is important when choosing external peripherals such as OSPI flash or SDRAM. While designing an external memory interface, it is highly recommended to refer to the electrical characteristics of the memory devices and check that the timing parameters are in the range specified in the UM for the proper working of the system.

Table 16. Bus Timings

Parameter	Condition	VCC/VCC2	Symbol	Min	Max	Unit
Address delay	Condition1	2.70 V or above	t_{AD}	1.0	12.5	ns
		1.62 V or above		1.0	12.5	ns
	Condition3	3.0 V or above		1.0	10.8	ns
Byte control delay	Condition1	2.70 V or above	t_{BCD}	1.0	12.5	ns
		1.62 V or above		1.0	12.5	ns
	Condition3	3.0 V or above		1.0	10.8	ns
CS delay	Condition1	2.70 V or above	t_{CSD}	1.0	12.5	ns
		1.62 V or above		1.0	12.5	ns
	Condition3	3.0 V or above		1.0	10.8	ns
ALE delay time	Condition1	2.70 V or above	t_{ALED}	1.0	12.5	ns
		1.62 V or above		1.0	12.5	ns
	Condition3	3.0 V or above		1.0	10.8	ns
RD delay	Condition1	2.70 V or above	t_{RSD}	1.0	12.5	ns
		1.62 V or above		1.0	12.5	ns
	Condition3	3.0 V or above		1.0	10.8	ns
Read data setup time	Condition1	2.70 V or above	t_{RDS}	12.5	—	ns
		1.62 V or above		20.5	—	ns
	Condition3	3.0 V or above		10.8	—	ns
Read data hold time	Condition1	2.70 V or above	t_{RDH}	0	—	ns
		1.62 V or above		0	—	ns
	Condition3	3.0 V or above		0	—	ns
WR/WRn delay	Condition1	2.70 V or above	t_{WRD}	1.0	12.5	ns
		1.62 V or above		1.0	12.5	ns
	Condition3	3.0 V or above		1.0	10.8	ns
Write data delay	Condition1	2.70 V or above	t_{WDD}	—	12.5	ns
		1.62 V or above		—	12.5	ns
	Condition3	3.0 V or above		1.0	10.8	ns

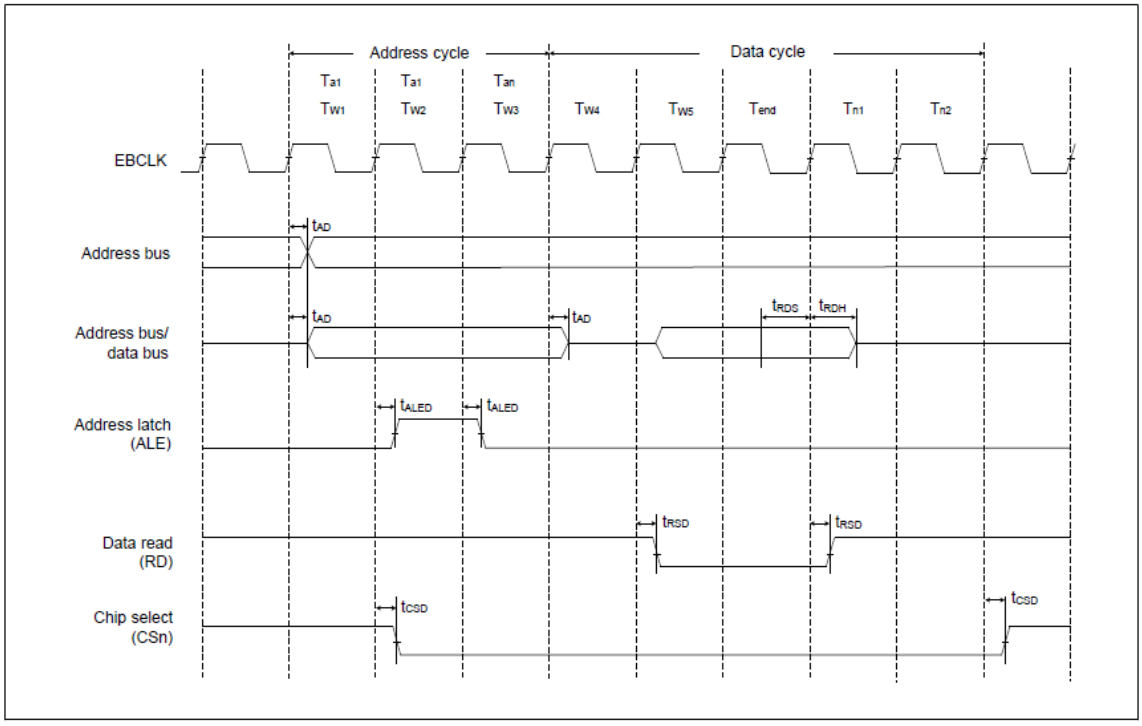


Figure 43. Address/data multiplexed bus read access timing

Table 17. Bus Timings

Parameter	Condition	VCC/VCC2	Symbol	Min	Max	Unit
Address delay 2 (SDRAM)	Condition 2	3.0 V or above	t_{AD2}	0.8	6.0	ns
	Condition 3	3.0 V or above		0.8	10	
CS delay 2 (SDRAM)	Condition 2	3.0 V or above	t_{CSD2}	0.8	6.0	ns
	Condition 3	3.0 V or above		0.8	10	
DQM delay (SDRAM)	Condition 2	3.0 V or above	t_{DQMD}	0.8	6.0	ns
	Condition 3	3.0 V or above		0.8	10	
CKE delay (SDRAM)	Condition 2	3.0 V or above	t_{CKED}	0.8	6.0	ns
	Condition 3	3.0 V or above		0.8	10	
Read data setup time 2 (SDRAM)	Condition 2	3.0 V or above	t_{RDS2}	2.1	—	ns
	Condition 3	3.0 V or above		6.1	—	
Read data hold time 2 (SDRAM)	Condition 2	3.0 V or above	t_{RDH2}	1.5	—	ns
	Condition 3	3.0 V or above		1.5	—	
Write data delay 2 (SDRAM)	Condition 2	3.0 V or above	t_{WDD2}	—	6.0	ns
	Condition 3	3.0 V or above		—	10	
Write data hold time 2 (SDRAM)	Condition 2	3.0 V or above	t_{WDH2}	0.8	—	ns
	Condition 3	3.0 V or above		0.8	—	
WE delay (SDRAM)	Condition 2	3.0 V or above	t_{WED}	0.8	6.0	ns
	Condition 3	3.0 V or above		0.8	10	
RAS delay (SDRAM)	Condition 2	3.0 V or above	t_{RASD}	0.8	6.0	ns
	Condition 3	3.0 V or above		0.8	10	
CAS delay (SDRAM)	Condition 2	3.0 V or above	t_{CASD}	0.8	6.0	ns
	Condition 3	3.0 V or above		0.8	10	

SDRAM Single read timing diagram is shown as an example where the minimum and maximum delay for different parameters, such as address delay, chip select delay, write data delay, and other parameters, are listed. This is just a guide in selecting the SDRAM devices. When choosing the SDRAM, the device's electrical characteristics will also have a minimum and maximum range of timing details for different parameters. For proper operation, they need to be within the range for proper data read, write, and other operations.

For instance, CS delay2 (SDRAM) t_{csd2}, the minimum delay is 0.8ns and the maximum is 6.0ns, which means the SDRAM device must also be in accordance with the limit specified. So while choosing the SDRAM for the design, the electrical characteristics of the device UM also need to be considered for all the matching characteristics.

In the below Figure 44, if the SDRAM clock is 120MHZ (8.3ns) the t_{csd2} within the range of 0.8ns to 6.8 ns. Similar to other parameters, their timings can be measured and verified before selecting the device.

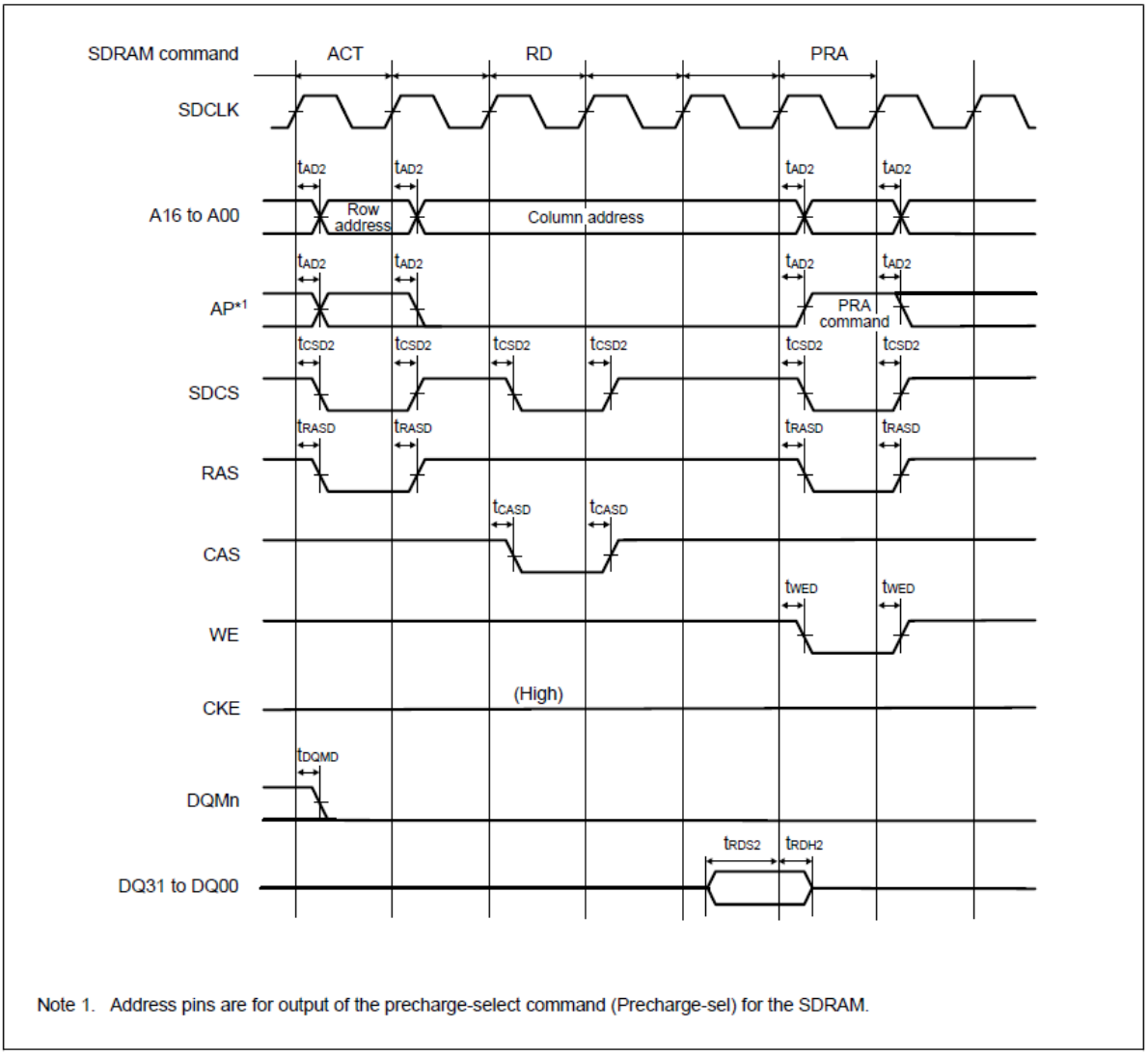


Figure 44. SDRAM single read cycle timing

9. Example Projects, Reference Applications, and Application Notes

This section provides reference example projects to implement memory architecture configurations using FSP in real-world scenarios.

9.1 Reference Example for OSPI

For understanding the operation of the external OSPI interface, the xSPI Example Project using EK-RA8P1, EK-RA8M2, and EK-RA8D2 can be referred to. In this Example project, the FSP OSPI driver is used to show the operation of the OSPI module. The Project uses the onboard OSPI device to demonstrate the 1S-1S-1S and 8D-8D-8D modes. The example project also gives the FSP configuration for the OSPI module. Users can refer to this example project to start developing OSPI applications.

The EK-RA8P1 OSPI Example Projects can be found at the location ra-fsp-examples/example_projects/ek_ra8p1 at master renesas/ra-fsp-examples GitHub

The EK-RA8M2 OSPI Example Projects can be found at the location ra-fsp-examples/example_projects/ek_ra8m2 at master renesas/ra-fsp-examples GitHub

The EK-RA8D2 OSPI Example Projects can be found at the location ra-fsp-examples/example_projects/ek_ra8d2 at master renesas/ra-fsp-examples GitHub

9.2 Reference Application project for Decryption on the fly for OSPI

For understanding the operation of the DOTF for OSPI, the reference application Project 'Application_Design_using_RA8_Series_MCU_Decryption_On_the_Fly_for_OSPI' using EK-RA8P1 can be referred to. In this Application project, the FSP OSPI driver, along with DOTF, is used to show the operation of the OSPI module and DOTF feature.

The OSPI Application Projects can be found at the location ra-fsp-examples/application_projects/r11an0773/Application_Design_using_RA8_Series_MCU_Decryption_On_the_Fly_for_OSPI/.

9.3 Reference Example for SDRAM

The SDRAM Example project for RA8P1 will have an option to store the data under SDRAM and retrieve it. It can be found at the location ra-fsp-examples/example_projects/ek_ra8p1 at master · renesas/ra-fsp-examples · GitHub.

The SDRAM Example project for RA8M2 will have an option to store the data under SDRAM and retrieve it. It can be found at the location ra-fsp-examples/example_projects/ek_ra8m2 at master · renesas/ra-fsp-examples · GitHub.

The SDRAM Example project for RA8D2 will have an option to store the data under SDRAM and retrieve it. It can be found at the location ra-fsp-examples/example_projects/ek_ra8d2 at master · renesas/ra-fsp-examples · GitHub.

9.4 Reference Example for MRAM

The MRAM Example project (mram_ek_ra8p1_ep) RA8P1 will have an option to validate the basic functionality of the MRAM driver. The user can interact with the application via RTT input to execute write, read, and erase operations, starting at a fixed MRAM address. It can be found at the location ra-fsp-examples/example_projects/ek_ra8p1 at master renesas/ra-fsp-examples GitHub.

The MRAM Example project (mram_ek_ra8m2_ep)RA8M2 will have an option to validate the basic functionality of the MRAM driver. The user can interact with the application via RTT input to execute write, read, and erase operations, starting at a fixed MRAM address. It can be found at the location ra-fsp-examples/example_projects/ek_ra8m2 at master renesas/ra-fsp-examples GitHub.

The MRAM Example project (mram_ek_ra8d2_ep)RA8D2 will have an option to validate the basic functionality of the MRAM driver. The user can interact with the application via RTT input to execute write, read, and erase operations, starting at a fixed MRAM address. It can be found at the location ra-fsp-examples/example_projects/ek_ra8d2 at master renesas/ra-fsp-examples GitHub.

10. References

- Renesas FSP User's Manual: https://renesas.github.io/fsp
- Flash Memory Programming: https://www.renesas.com/us/en/document/apn/flash-memory-programming
- Renesas RA MCU Datasheets: See http://renesas.com/ra and select the relevant MCU
- RA8P1, RA8M2, RA8T2, and RA8D2 Example Projects on Renesas RA GitHub: https://github.com/renesas/ra-fsp-examples
- RA8P1, RA8M2, RA8T2, and RA8D2 Quick Design Guide
- Renesas RA Family RA8P1 User's Manual: Hardware (R01UH1064)
- Renesas RA Family RA8M2 User's Manual: Hardware (R01UH1066)
- Renesas RA Family RA8D2 User's Manual: Hardware (R01UH1065)
- Renesas RA Family RA8T2 User's Manual: Hardware (R01UH1067)

- Renesas RA Family RA8 Quick Design Guide (R01AN7087)
- *High Performance with RA8 MCU using CM85 core with Helium™-(R01AN7127)*
- EK-RA8P1, MCK-RA8T2, EK-RA8M2, and EK-RA8D2 Board Design Schematics.
- AMBA® AXI and ACE Protocol Specification
- Arm® AMBA® 5 AHB Protocol Specification
- AMBA® APB Protocol Version 2.0 Specification
- AMBA® ATB Protocol Specification
- ARM® Cortex®-M33 Processor Technical Reference Manual (ARM 100230)
- TrustZone® technology for ARM®v8-M Architecture (ARM 100690_0100_00_en)
- ARM® Cortex®-M85 Processor Technical Reference Manual (ARM 101924_0000_02_en)

11. Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

EK-RA8P1 Resources	www.renesas.com/ek-ra8p1
MCK-RA8T2 Resources	www.renesas.com/mck-ra8t2
EK-RA8M2 Resources	www.renesas.com/ek-ra8m2
EK-RA8D2 Resources	www.renesas.com/ek-ra8d2
RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.30.25	-	Initial version
1.01	Oct.15.25		Added MCU support for RA8T2, RA8M2, RA8D2

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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/.