RZ/A2M Group

Example of booting from HyperFlash™ using HyperBus™ controller

Introduction

This application note describes an example of booting from the HyperFlash via the HyperBus™ controller of RZ/A2M by using the boot mode 7 (HyperFlash™ boot 2 1.8-V product).

Target Device

RZ/A2M

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Note: HyperBus™/HyperFlash™/HyperRAM™ are trademarks of Cypress Semiconductor Corporation.

Contents

1. Specification

1.1 Booting from HyperFlash

In boot mode 7, RZ/A2M boots from HyperFlash allocated to the HyperFlash space. Figure 1.1 shows the Conceptual diagram of boot mode 7.

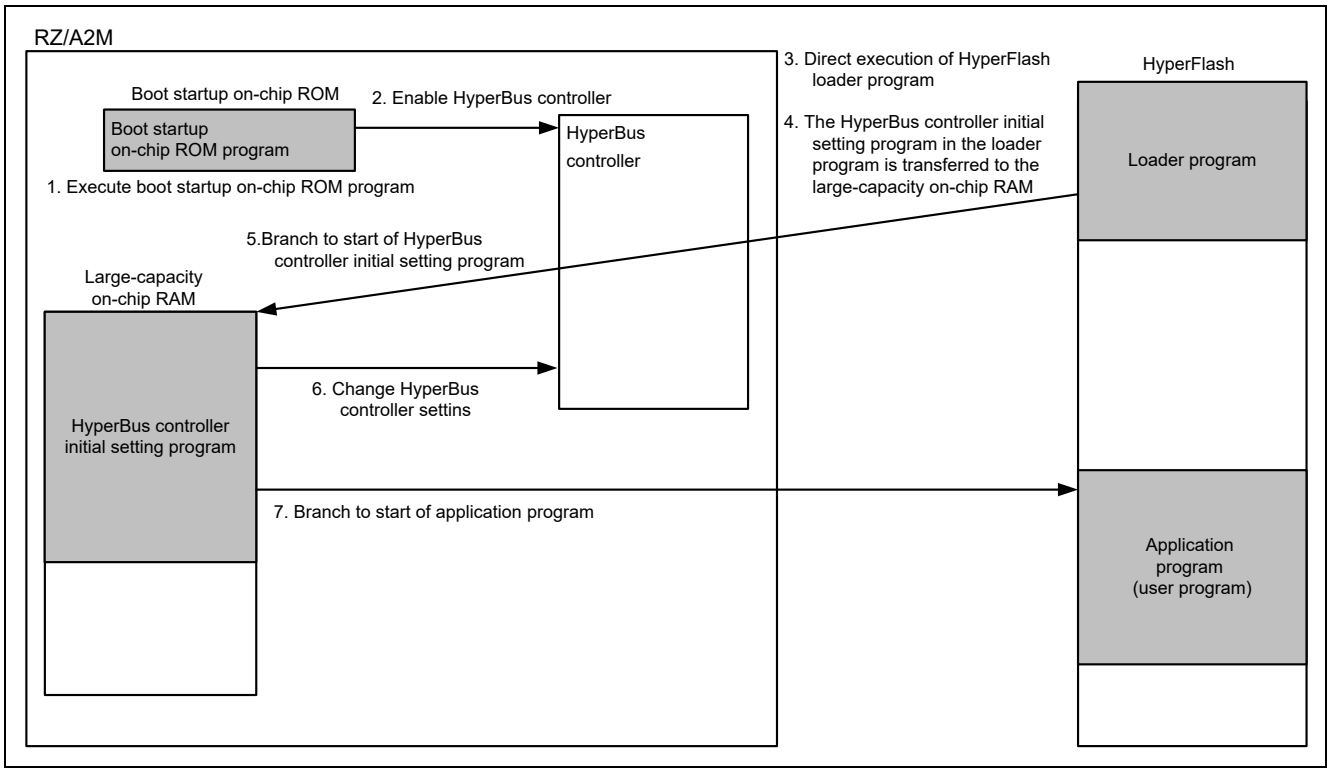


Figure 1.1 Conceptual diagram of boot mode 7

The conceptual diagram of boot mode 7 operation is described below.

1 When RZ/A2M starts up by boot mode 7, the boot startup on-chip ROM program runs after power-on reset is canceled.
2 The boot startup on-chip ROM program sets the CPG register, selects the HyperBus-related pins for use to enable to directly run programs allocated to the HyperFlash space.
3 The HyperFlash loader program is executed.
4 Both the HyperBus controller and HyperFlash initial setting program are transferred from HyperFlash to the large-capacity on-chip RAM.
5 The execution branches to the destination RAM address of transferred program.
6 Both the HyperBus controller and HyperFlash settings are changed via the HyperBus controller initial setting program.
7 The execution branches to the start address of the application program.

Immediately after the boot startup on-chip ROM program is executed, RZ/A2M is set to perform low speed access of HyperFlash, so it is necessary for the user program to provide the settings to enable high-speed access to HyperFlash. For the purpose of this setting, this application note describes how to allocate the loader program which provides optimal settings to the HyperFlash used by the customer to the start address (H'3000_0000) of the HyperFlash space branched by the boot startup on-chip ROM program, and then branch to the customer-created application program (user program) after the setting process to enable high-speed access to the HyperFlash in the loader program has been executed.

1.2 Peripheral Functions Used

This sample code not only configures the HyperBus controller but also initializes the clock pulse generator, interrupt controller, general-purpose input/output ports, memory management unit, primary cache (L1 cache), and secondary cache (L2 cache).

In this application note, the Clock pulse generator is referred to as the CPG, the Interrupt controller as the INTC, the OS timer as the OSTM, the Serial communication interface with FIFO as the SCIFA, the General I/O ports as the GPIO, the Power-down modes as the STB, and the Memory management unit as the MMU.

Table 1.1 summarizes Peripheral functions and their applications, and Figure 1.2 shows Operating environment for the sample code.

Table 1.1 Peripheral functions and their applications

Peripheral function	Application
HyperBus controller	In the memory-map read mode, the CPU generates a signal for direct read from HyperFlash connected to the HyperFlash space.
Clock pulse generator (CPG)	Generates the operating clocks of several frequencies for RZ/A2M.
Interrupt controller (INTC)	Used to control interrupts for OSTM channel 0, OSTM channel 2, and SCIFA channel 4
OS timer (OSTM)	Uses OSTM channel 0 and channel 2 • OSTM channel 0 Generates the intervals at which an LED is turned on and off. • OSTM channel 2 Used for time management via OS Abstraction Layer
Serial communication interface with FIFO (SCIFA)	Used for communication with host PC via SCIFA channel 4
General I/O ports (GPIO)	Used to switch the multiplexed pin functions for SCIFA channel 4, and to control a pin for turning on and turning off the LED.
Power-down modes (STB)	Used to cancel the module standby state of the RZ/A2M's peripheral I/O, and to enable writing to the on-chip data retention RAM.
Memory management unit (MMU), L1 cache, and L2 cache	Generates translation tables such as specification of the valid area of L1 cache or specification of the memory type in the RZ/A2M external address space, and sets the L1 cache and L2 cache enabled.

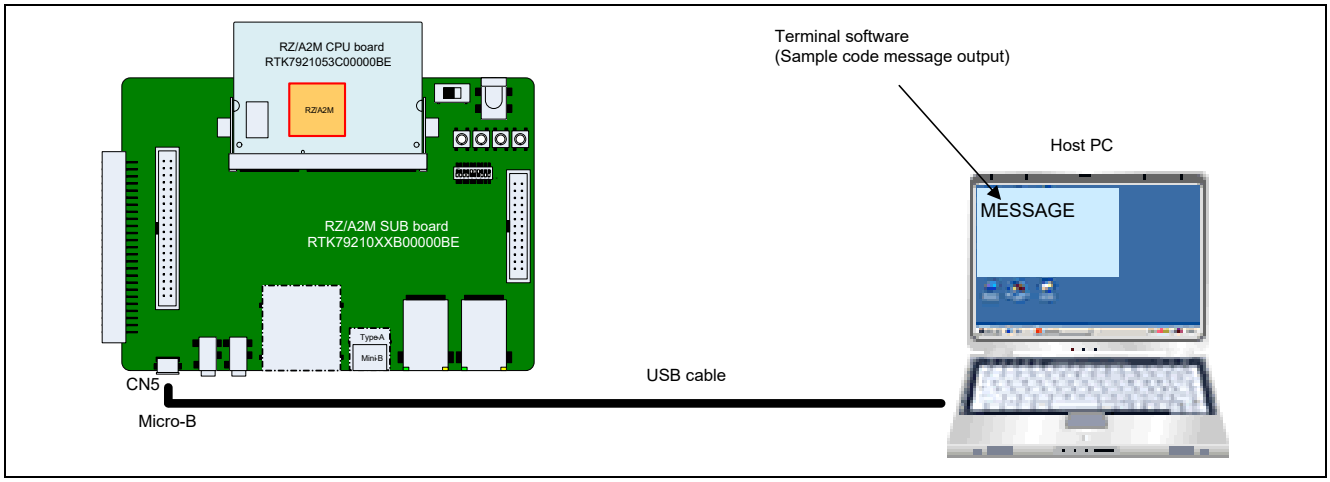


Figure 1.2 Operating environment

2. Operation Confirmation Conditions

The operation of the sample code accompanying this application note has been confirmed under the conditions below.

Table 2.1 Operation confirmation conditions (1/2)

Item	Description
MCU used	RZ/A2M
Operating frequency*	CPU clock (Iϕ): 528 MHz Image processing clock (Gϕ): 264 MHz Internal bus clock (Bϕ): 132 MHz Peripheral clock 1 (P1ϕ): 66 MHz Peripheral clock 0 (P0ϕ): 33 MHz HM_CK/HM_CK#: 132 MHz CKIO: 132 MHz
Operating voltage	Power supply voltage (I/O): 3.3 V Power supply voltage (PVcc_HO): 1.8 V Power supply voltage (internal): 1.2 V
Integrated development environment	e^2 studio V7.6.0
C compiler	GNU Arm Embedded Toolchain 6-2017-q2-update
	Compiler option (addition of directory path excluded) Release configuration: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Os -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -Wstack-usage=100 -fabi-version=0 Hardware Debug configuration: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Og -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -g3 -Wstack-usage=100 -fabi-version=0

Note: The operating frequency used in clock mode 1 (Clock input of 24 MHz from EXTAL pin)

Table 2.2 Operation confirmation conditions (2/2)

Item	Description
Operating mode	HyperFlash boot 7 (HyperFlash Booting 2 1.8-V product)
Communication setting of the terminal software	• Baud rate: 115200 bps • Data length: 8 bits • Parity: None • Stop bits: 1 bit • Flow control: None
Boards used	RZ/A2M CPU board RTK7921053C00000BE RZ/A2M SUB board RTK79210XXB00000BE
Devices used (functions used on the board)	• HyperBus MCP (HyperFlash: 64 Mbytes, HyperRAM: 8 Mbytes) • Manufacturer: Cypress, Product name: S71KS512SC0BHV000 • RL78/G1C (Converts between USB communication and serial communication to communicate with the host PC.) • LED1

3. Reference Application Note

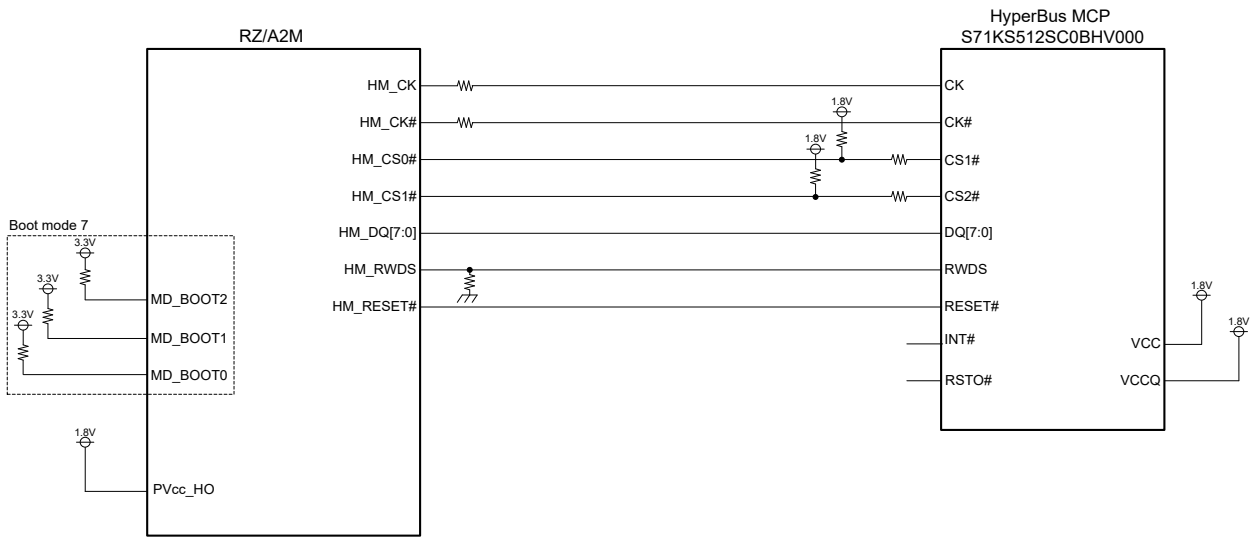
For additional information associated with this document, refer to the following application note.

- RZ/A2M Group: Example of Initialization (R01AN4321)

4. Hardware

4.1 Hardware Configuration

In the HyperFlash boot example introduced in this application note, processing is performed by the programs stored in the HyperFlash connected to the HyperFlash space using boot mode 7. Figure 4.1 shows the Connection example for booting from HyperFlash in boot mode 7.



Notes: 1. # is the symbol that indicates negative logic (or active low).
2. S71KS512SC0BHV000 includes both 64 MB HyperFlash and 8 MB HyperRAM. In boot mode 7, the HyperFlash connected to the HyperBus controller's HyperFlash space is accessed. In the processing of the application program in this sample code, the HyperRAM connected to the HyperBus controller's HyperRAM space can be accessed.

Figure 4.1 Connection example for booting from HyperFlash in boot mode 7

4.2 Used Pins List

Table 4.1 shows the Pins used and their functions.

Table 4.1 Pins used and their functions

Pin name	I/O	Function
MD_BOOT2	Input	Select boot mode (set to boot mode 7) MD_BOOT2: "H", MD_BOOT1: "H", MD_BOOT0: "H"
MD_BOOT1	Input	
MD_BOOT0	Input	
HM_CK	Output	HyperBus controller differential clock
HM_CK#	Output	HyperBus controller differential clock
HM_CS0#	Output	HyperBus controller chip select 0 (for HyperFlash)
HM_CS1#	Output	HyperBus controller chip select 1 (for HyperRAM)
HM_DQ[7:0]	Input and output	HyperBus controller data
HM_RWDS	Input and output	HyperBus controller read/write data mask
HM_RESET#	Output	Reset from HyperBus controller
P6_0	Output	Turns LED on and off
RxD4(P9_1)	Input	Serial receive data signal
TxD4(P9_0)	Output	Serial transmit data signal

Note: # is the symbol that indicates negative logic (or active low).

5. Software

5.1 Operation Overview

This section provides an overview of the sample code operation presented in this application note.

5.1.1 Terms Related to HyperFlash Boot

Table 5.1 lists the Terms related to HyperFlash boot in this application note.

Table 5.1 Terms related to HyperFlash boot

Term	Description
Boot startup on-chip ROM program	This program provides settings to directly execute the programs stored in the HyperFlash connected to the HyperFlash space when started up in boot mode 7 (HyperFlash boot 2 1.8-V product). RZ/A2M branches to the address of H'3000_0000 which is the start address of the HyperFlash space after the boot startup on-chip ROM program has been executed. Since this program is stored in the on-chip ROM of RZ/A2M, it does not need to be created by the customer.
Loader program	This program is executed after the boot startup on-chip ROM program process has completed. The loader program makes settings to the HyperBus controller and the registers in the HyperFlash corresponding to the HyperFlash used by the customer, and then branches to the start address of the application program. The loader program should be created by the customer according to the specifications of the HyperFlash to be used while referring to this application note. In the sample code, the initial settings are optimized for use with the Cypress HyperBus MCP (S71KS512SC0BHV000).
Application program (User program)	This program should be created by customers according to their system to be used.

5.1.2 Operation Overview of Sample Code Overall

The sample code consists of the loader program and the application program executed after completing the process of boot startup on-chip ROM program.

1 Loader program (Project name: rza2m_hyperflash_boot_loader_gcc)
The loader program provides optimal settings to the HyperFlash used (Cypress HyperBus MCP (S71KS512SC0BHV000)). The loader program should be located at the start address (H'3000_0000) of the HyperFlash space branched from the boot startup on-chip ROM program. After the loader program is executed, validating signature in the application program, is performed, and the execution branches to the start address of the application program.
The start address of the application program is specified by the symbol definition "__application_base_address" of the linker script "linker_script.ld".

2 Application program (Project name: rza2m_hyperflash_boot_sample_osless_gcc)
This is a program to be executed after optimal settings for HyperFlash are provided in the loader program. Change the allocation address so that the VECTOR_TABLE section of the application program matches the address that is specified in "__application_base_address."
In the sample code, the application program is located at the address H'3004_0000.

Figure 5.1 shows the Operation overview of sample code in this application note.

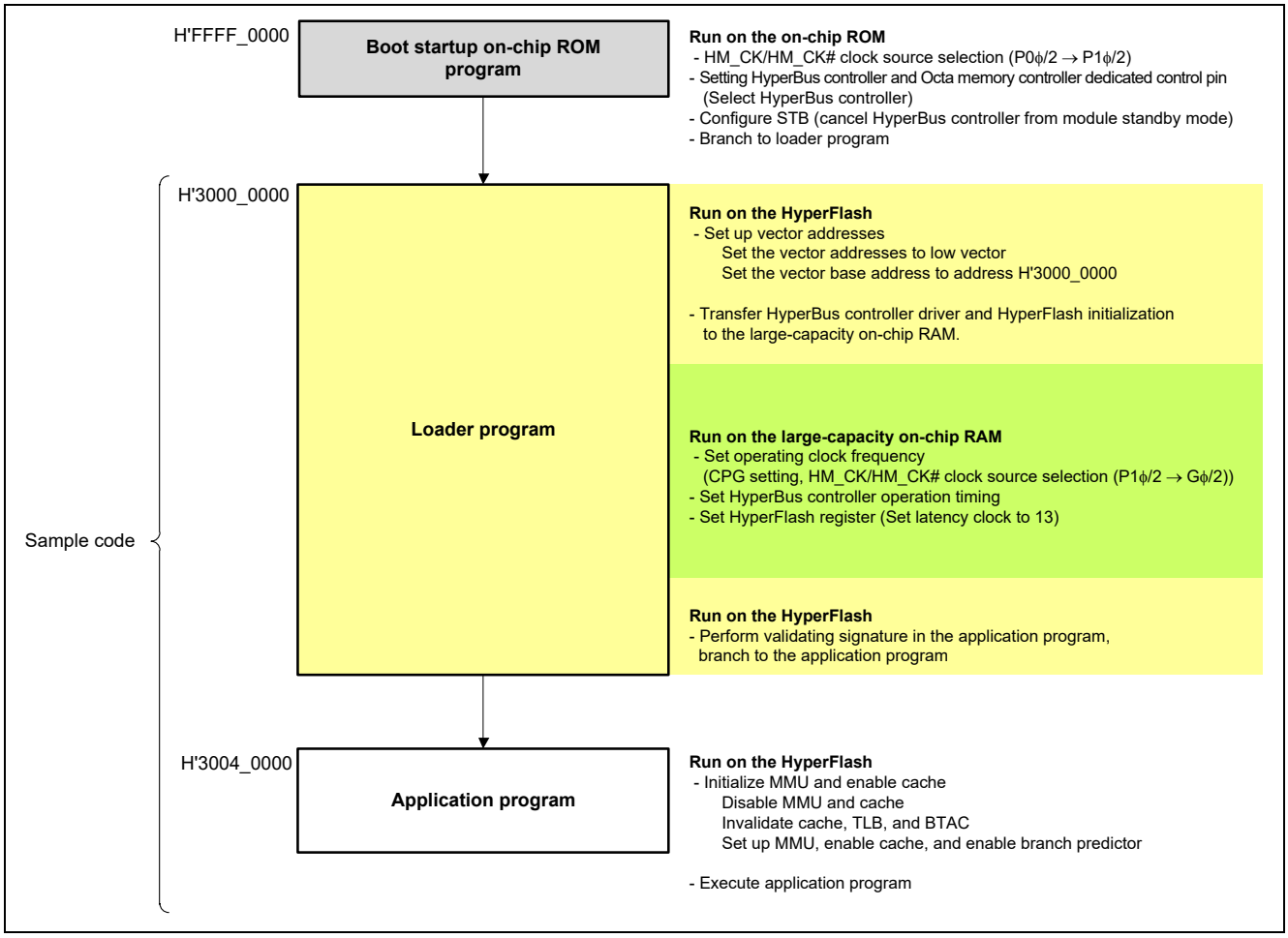


Figure 5.1 Operation overview of sample code

5.1.3 Operation Overview of Loader Program

The loader program is executed after completing the process of the boot startup on-chip ROM program. The loader program should be located at the start address (H'3000_0000) of the HyperFlash space branched from the boot startup on-chip ROM program.

The boot startup on-chip ROM program makes setting enable the HyperBus controller to access HyperFlash. The settings enable reading for the HyperFlash connected to RZ/A2M, and enable the direct execution of programs located in the HyperFlash space. RZ/A2M is set to allow for general access to HyperFlash, therefore it is necessary to execute processes to optimize access to the HyperFlash used in the customer's loader program (such as change of clock frequencies, and latency change of process to match the HyperFlash specifications).

Because the loader program's HyperBus controller setting process and the HyperFlash register change process cannot be set by a program allocated to the HyperFlash space, these processes must be transferred to the large-capacity on-chip RAM and executed there.

Refer to Table 5.2 for the settings after execution of the boot startup on-chip ROM program and loader program.

Table 5.2 shows the settings made by the boot startup on-chip ROM program and the loader program.

After the loader program makes the settings shown in Table 5.2, it branches to the start address of the application program. In the sample code, the application program is located at the address H'3004_0000.

Table 5.2 Settings for the boot startup on-chip ROM program and loader program

	Item	After execution of boot startup on-chip ROM program	After execution of loader program
CPG setting	Operating clock settings Clock input of 24 MHz from EXTAL pin in clock mode 1	Iϕ = 132 MHz Gϕ = 264 MHz Bϕ = 132 MHz P1ϕ = 66 MHz P0ϕ = 33 MHz	Iϕ = 528 MHz Gϕ = 264 MHz Bϕ = 132 MHz P1ϕ = 66 MHz P0ϕ = 33 MHz
	HYPCLK selection SCLKSEL.HYMCR[1:0]	Select P1ϕ B'01	Select Gϕ B'11
HyperBus controller setting	Maximum time setting enable: MCR0.MAXEN	HM_CS0# signal Low time is not set 0	
	Maximum time setting: MCR0.MAXLEN[8:0]	MCR0.MAXEN is set to "0", so maximum time setting is disabled —	
	Read chip select timing setting: MTR0.RCSHI[3:0]	Time from when HM_CS0# signal has been negated until the start of the next read access B'0000 (1.5 clock cycles)	
	Write chip select timing setting: MTR0.WCSHI[3:0]	Time from when HM_CS0# signal has been negated until the start of the next write access B'0000 (1.5 clock cycles)	
	Read chip select set up timing setting: MTR0.RCSS[3:0]	Time from when HM_CS0# signal has been asserted until the start of the next read access B'0000 (1 clock cycle)	
	Write chip select set up timing setting: MTR0.WCSS[3:0]	Time from when HM_CS0# signal has been asserted until the start of the next write access B'0000 (1 clock cycle)	
	Read chip select hold timing setting: MTR0.RCSH[3:0]	Time from when read access was completed until the HM_CS0# signal is negated. B'0000 (1 clock cycle)	
	Write chip select hold timing setting: MTR0.RCSH[3:0]	Time from when write access was completed until the HM_CS0# signal is negated. B'0000 (1 clock cycle)	
HyperFlash register setting	Volatile Configuration Register	16 clock read latency xVCR[7:4] = B'1011	13 clock read latency xVCR[7:4] = B'1000
Other	CP15 system control register vector bit	1 (High vector)	0 (Low vector)
	CP15 vector-based address register (VBAR)*	—	H'3000_0000

Note: It is possible to allocate either a high vector (H'FFFF_0000) or low vector (H'0000_0000) to the vector base address depending on the setting of the CP15 system control register's V bit, and if set to low vector, it is possible to set the vector base address via VBAR.

5.1.4 Application Program

(1) Operation of the application program

This application program shows an example of access to the HyperRAM connected to RZ/A2M, by setting the HyperRAM space's HyperBus controller and HyperRAM register, then performing write and read processes for the area allocated to the HyperRAM space.

After a reset is cancelled, the boot startup on-chip ROM program and loader program are executed in that order. It then branches to the startup processing for the application program that is allocated to address H'3004_0000.

In the startup process, after the stack pointer has been set, as shown in Table 5.3, the HyperBus controller and HyperRAM register settings are implemented to enable access to HyperRAM*. The settings for the MMU and FPU are executed, and the section initialization is performed, after that, the execution branches to the resetprg function.

In the resetprg function, after RTC and USB unused channel initialization processing is executed, L1 cache and L2 cache are enabled and INTC initialization is performed. The large-capacity on-chip RAM address is set in VBAR to enable high-speed interrupt processing, IRQ interrupt and FIQ interrupt are enabled, and the main function is called.

In the main function for this application program, the CPG, OSTM channel 0, SCIFA channel 4, and GPIO initial setting processing is performed. As a result of this initialization processing, the main function outputs the character strings (startup message) to the terminal on the host PC connected with the serial interface and sets the OSTM channel 0 timer to interval timer mode to activate the timer. It generates the OSTM channel 0 interrupt with a cycle of 500 ms and repeats turning on/off the LED on the CPU board every 500 ms using such interrupt.

For details on the initialization executed by the application program, refer to the application note "RZ/A2M Group Example of Initialization".

In addition, in the main function of this application program, a write to the HyperRAM cache disable area is performed.

Note: Setting of the HyperBus controller corresponding with HyperRAM space is performed before section initialization to allow the HyperRAM to be used as work RAM area, and access to the HyperRAM register is performed before the MMU enable setting (Access to the HyperRAM register requires setting the memory attribute to the strongly-ordered attribute or the device attribute's HyperRAM area).

Table 5.3 shows the HyperBus controller and HyperRAM register settings for the application program.

Table 5.3 HyperBus controller and HyperRAM register settings

	Item	After execution of application program
HyperBus controller setting	Maximum time setting enable: MCR1.MAXEN	HM_CS1# signal Low time is set 0
	Maximum time setting: MCR1.MAXLEN[8:0]	MCR1.MAXEN is set to "0", so maximum time setting is disabled —
	Access target: MCR1.CRT	Access target during read/write 0 (memory base)
	Device type: MCR1.DEVTYPE	1 (HyperRAM)
	Read chip select timing setting: MTR1.RCSHI[3:0]	Time from when HM_CS1# signal has been negated until the start of the next read access B'0000 (1.5 clock cycles)
	Write chip select timing setting: MTR1.WCSHI[3:0]	Time from when HM_CS1# signal has been negated until the start of the next write access B'0000 (1.5 clock cycles)
	Read chip select set up timing setting: MTR1.RCSS[3:0]	Time from when HM_CS1# signal has been asserted until the start of the next read access B'0000 (1 clock cycle)
	Write chip select set up timing setting: MTR1.WCSS[3:0]	Time from when HM_CS1# signal has been asserted until the start of the next write access B'0000 (1 clock cycle)
	Read chip select hold timing setting: MTR1.RCSH[3:0]	Time from when read access was completed until the HM_CS1# signal is negated. B'0000 (1 clock cycle)
	Write chip select hold timing setting: MTR1.RCSH[3:0]	Time from when write access was completed until the HM_CS1# signal is negated. B'0000 (1 clock cycle)
	Latency setting: MTR1.LTCY[3:0]	B'0000 (5 clock cycles)
HyperRAM register setting	Configuration Register 0	CR0[7:4] = B'0000 (5 clock latency)

(2) Notes to be observed when creating an application program

The application program should be allocated to the address branched from the loader program. Note that the application program should be allocated to the different sector in HyperFlash from the one in the loader program.

The sector size of HyperFlash in the Cypress HyperBus MCP (S71KS512SC0BHV000) mounted on the RZ/A2M CPU board is 256 KB. In the sample code, the application program is allocated to the address of H'3004_0000 in sector 01.

Figure 5.2 shows the Sample code program allocation.

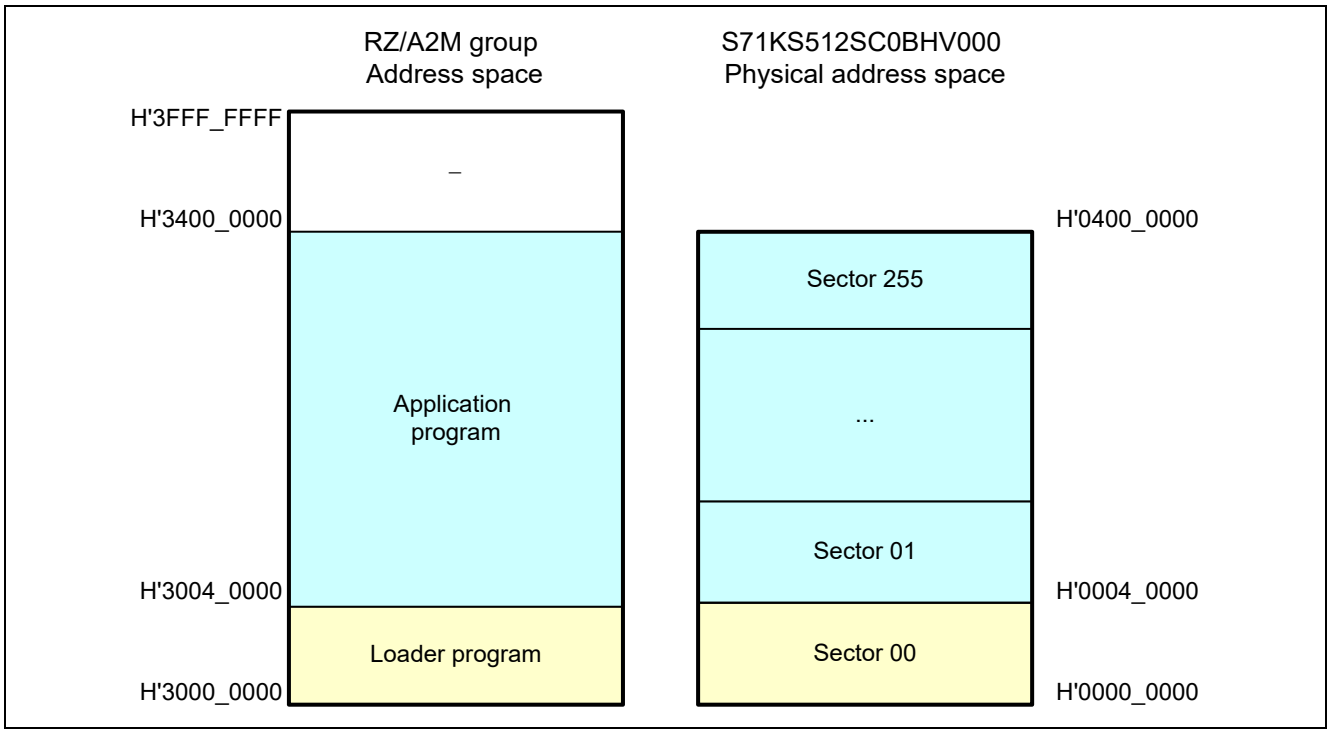


Figure 5.2 Sample code program allocation

The start address of the application program can be changed by making the following changes:

- Project for the loader program
  The branch to the starting address of the application program is executed by the loader program (reset_handler.asm). Specify the destination of branch with the symbol definition "__application_base_address" in the linker script "linker_script.ld".
- Project for the application program
  Change the allocation address so that the VECTOR_TABLE section of the application program matches the address that is specified in "__application_base_address".

5.2 Peripheral Function Setting and Memory Allocation when Sample Code is Executed

5.2.1 Setting for Peripheral Functions

Table 5.4 lists the Setting for peripheral functions during execution of the sample code.

Table 5.4 Setting for peripheral functions

Module	Setting
CPG	CPU clock: Set to 1/2 the PLL circuit frequency Internal bus clock: Set to 1/8 the PLL circuit frequency Peripheral clock 1 (P1ϕ): Set to 1/16 the PLL circuit frequency If the input clock is 24 MHz in clock mode 1 (divider 1: ×1/2, PLL circuit: ×88), set to the following frequencies • CPU clock (Iϕ): 528 MHz • Image processing clock (Gϕ): 264 MHz • Internal bus clock (Bϕ): 132 MHz • Peripheral clock 1 (P1ϕ): 66 MHz • Peripheral clock 0 (P0ϕ): 33 MHz • HM_CK/HM_CK#: 132 MHz (When Gϕ selected) • CKIO clock: 132 MHz (When Bϕ selected)
HyperBus controller	The CPU generates the signal for the direct read from the HyperRAM connected to the HyperRAM space and from the HyperFlash connected to the HyperFlash space
STB	Write permission to on-chip data retention RAM and provision of clock to peripheral functions Clock is supplied to OSTM0, OSTM2, SCIFA4, and the HyperBus controller with STBCR3, STBCR4, and STBCR9
GPIO	PORT6 and PORT9 shared pin functions are set • P6_0: Turns LED on and off • P9_1: RxD4, P9_0: TxD4
OSTM	Sets the channel 0 and the channel 2 in interval timer mode • Channel 0 Sets the timer count to have interrupt request generated every 500 ms when P1ϕ = 66 MHz. Generate the intervals at which the LEDs are turned on and off. • Channel 2 Sets the timer count to have interrupt request generated every 1 ms when P1ϕ = 66 MHz. Used for time management via OS Abstraction Layer.
INTC	Initializes INTC, registers and executes OSTM channel 0 interrupt (interrupt ID is 88) handler, OSTM channel 2 interrupt (interrupt ID is 90) handler and SCIFA channel 4 interrupt (interrupt ID is 322 or 323) handler respectively
SCIFA	Sets the channel 4 in asynchronous communication mode • Data length: 8 bits • Stop bits: 1 bit • Parity: None • Data transfer direction: LSB first transfer Sets the clock source without frequency dividing, the baud rate generator to double speed mode, and the basic clock at 8 times the bit rate when P1ϕ is 66 MHz. Set the bit rate parameter to 71 so that the bit rate is 115200 bps. (The bit rate error is -0.53 %)

5.2.2 Memory Mapping

Figure 5.3 shows the RZ/A2M Group address space and RZ/A2M CPU board memory map.

In the sample code, the code and data that use the ROM area are assigned to the HyperFlash connected to the HyperFlash space, and the code and data that use the RAM area are assigned to the large-capacity on-chip RAM.

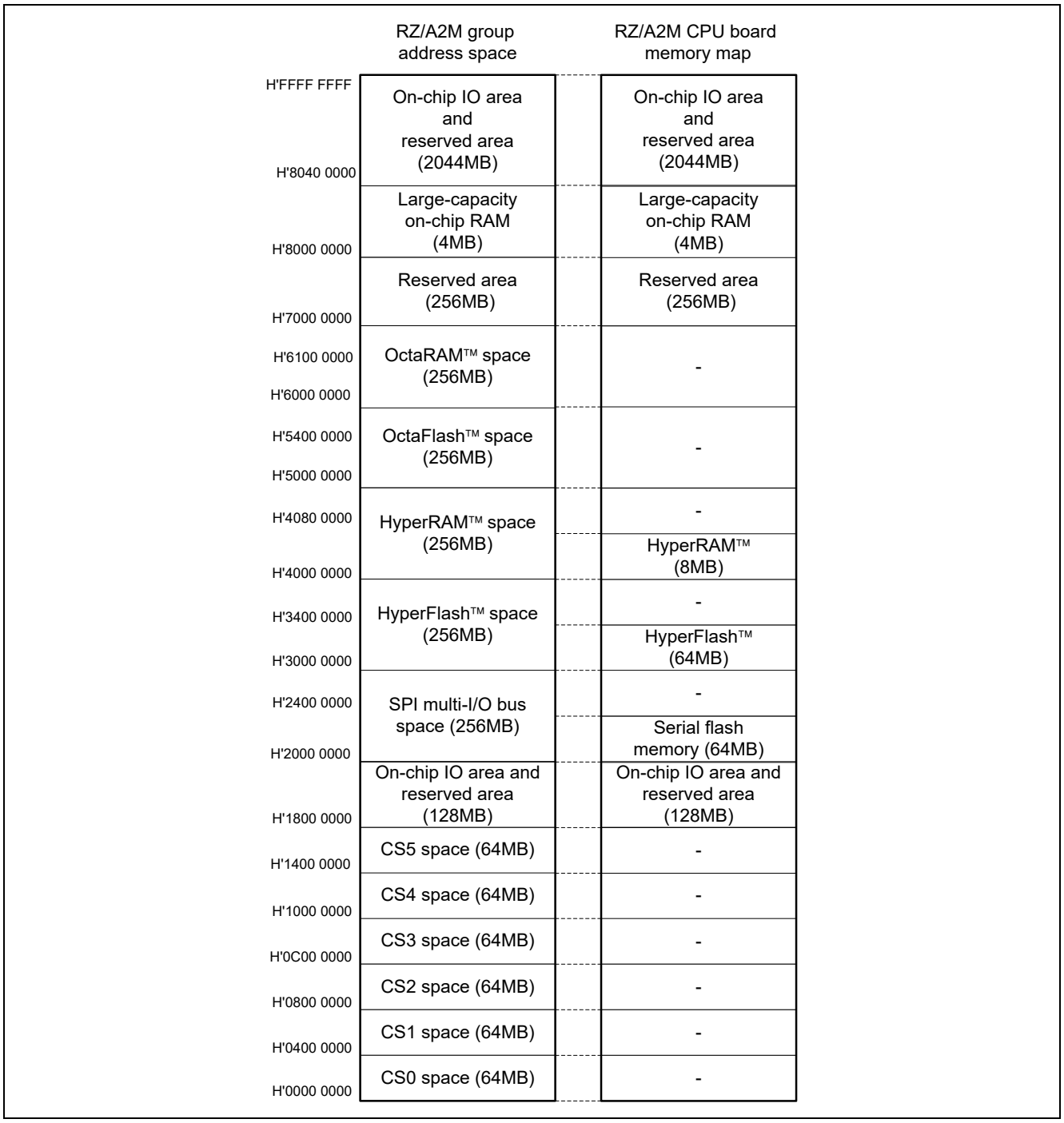


Figure 5.3 Memory mapping

5.2.3 Virtual Address Space Used in the Application Program.

In the application program, MMU is set to enabled, and virtual address space that can be accessed by the CPU is prepared, as shown in Figure 5.4 and Figure 5.5. However, the loader program operates in an MMU disabled state.

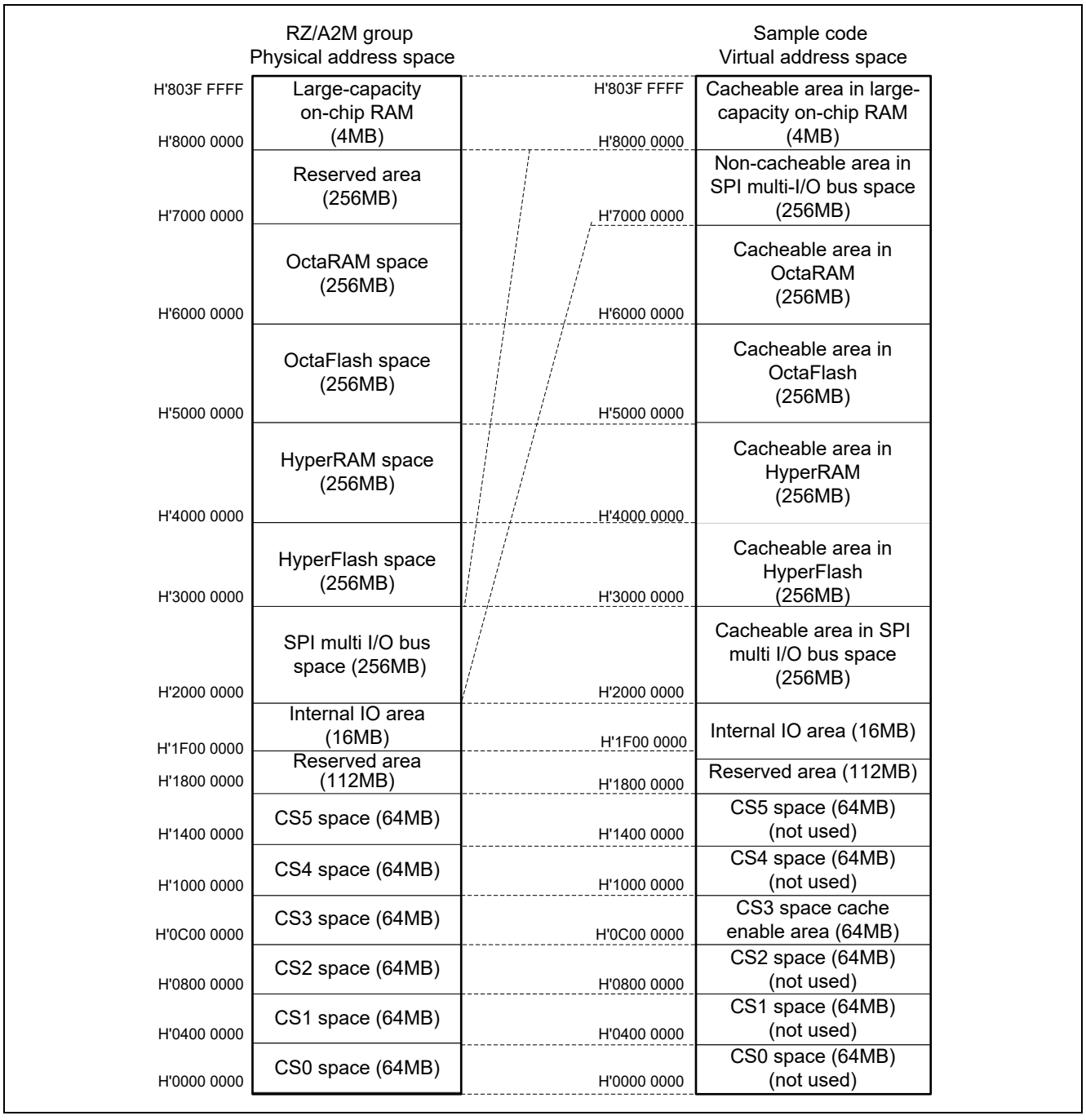
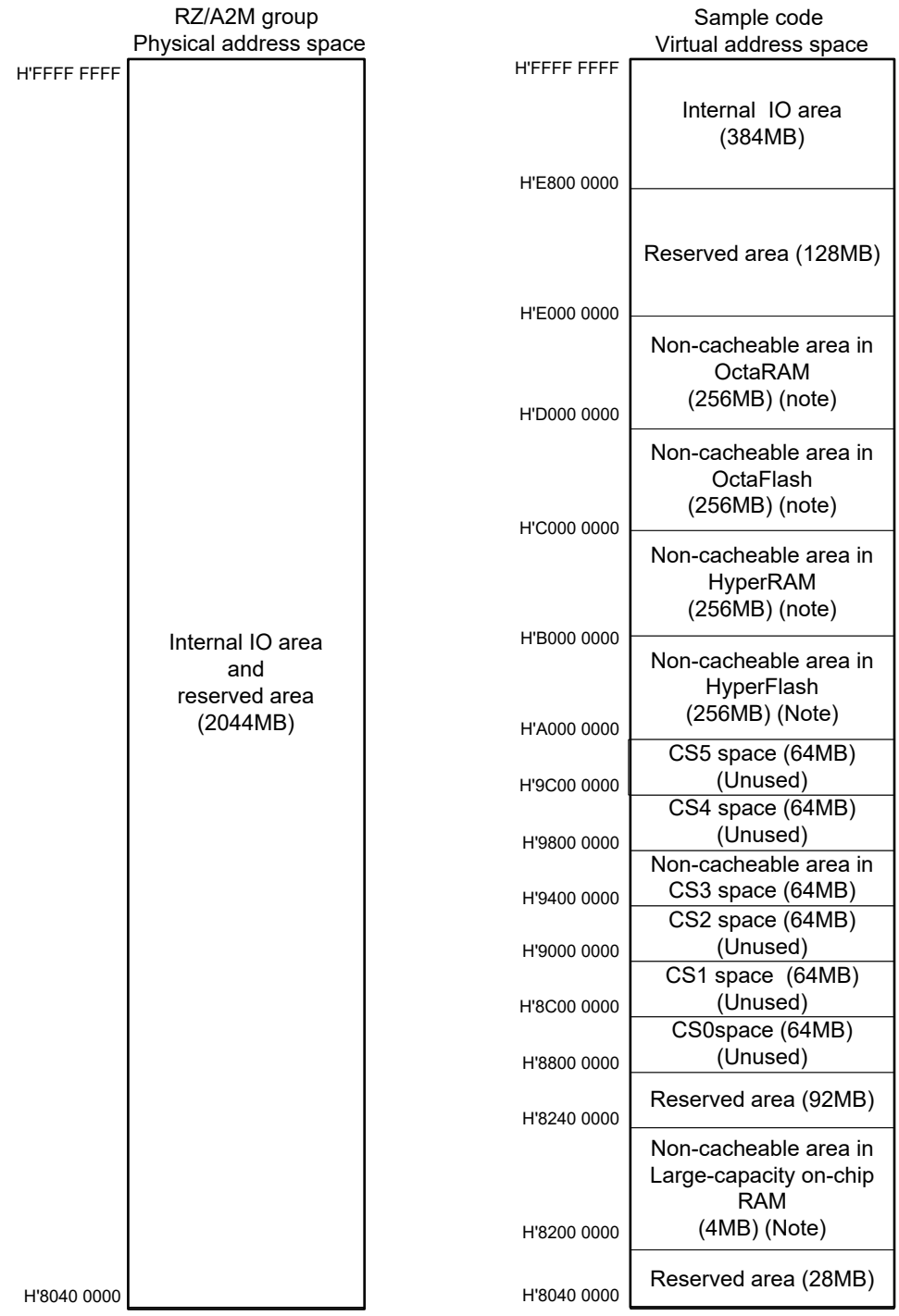


Figure 5.4 Virtual address space used in the application program (1/2)



Note: For each of the spaces for the large-capacity on-chip RAM, HyperFlash, HyperRAM, OctaFlash, and OctaRAM, an area usable where cache is disabled is prepared. The areas where cache is disabled of HyperFlash and OctaFlash set the MMU to the strongly-ordered attribute.

Figure 5.5 Virtual address space used in the application program (2/2)

5.2.4 Section Assignment in Sample Code

Table 5.5 shows Sections and objects to be used in the loader program, and

Table 5.6 to

Table 5.7 shows Sections and objects to be used in the application program.

Table 5.5 Sections and objects to be used in the loader program

Output section name	Input section name Input object name	Description	Loading area	Execution area
LOAD_MODULE1	VECTOR_TABLE	Exception processing vector table	FLASH	FLASH
LOAD_MODULE2	*/r_cpg/*.o (.text .rodata)	CPG settings processing	FLASH	LRAM
	*/rza_io_regrw.o (.text .rodata)	I/O register access processing		
	/r_hyperbus/.o (.text .rodata)	HyperBus controller settings processing		
	/hwsetup.o (.text .rodata)	Hardware Setup settings processing		
	* (.data)	Data area with default initial values		
LOAD_MODULE3	RESET_HANDLER	Reset processing	FLASH	FLASH
	INIT_SECTION */sections.o	Section initialization processing		
	* (.text)	Code area for defaults		
	* (.rodata)	Constant data area for defaults		
.data.memclk_setup	*/r_memclk_setup.o (.text .rodata .data)	Memory clock setting processing	FLASH	LRAM
	*/r_*_memclk_setup.o (.text .rodata .data)	Memory clock setting processing for each driver		
.bss.memclk_setup	*/r_memclk_setup.o (.bss COMMON)	Data area without initial values for memory clock settings processing	—	LRAM
	*/r_*_memclk_setup.o (.bss COMMON)	Data area without initial values for memory clock settings processing for each driver		
.stack	None	Stack area for SVC mode	—	LRAM
.bss	* (.bss .bss.*) * (COMMON)	Data area without default initial values	—	LRAM
.heap	None	Heap area	—	LRAM

Note: "FLASH" and "LRAM" shown in Load Area and Execution Area indicate the HyperFlash area and the large-capacity on-chip RAM area respectively.

Table 5.6 Sections and objects to be used in the application program (1/2)

Output section name	Input section name Input object name	Description	Loading area	Execution area
LOAD_MODULE1	VECTOR_TABLE	Exception processing vector table	FLASH	FLASH
LOAD_MODULE2	*/r_cpg/*.o (.text .rodata .data)	CPG settings processing	FLASH	LRAM
	*/rza_io_regrw.o (.text .rodata .data)	I/O register access processing		
	/r_hyperbus/.o (.text .rodata .data)	HyperBus controller settings processing		
	/hwsetup.o (.text .rodata .data)	Hardware Setup settings processing		
LOAD_MODULE3	*/r_cpg/*.o (.bss)	Data area without initial values for CPG settings processing	—	LRAM
	*/rza_io_regrw.o (.bss)	Data area without initial values for I/O register access processing		
LOAD_MODULE4	RESET_HANDLER	Reset processing	FLASH	FLASH
	INIT_SECTION */sections.o	Section initialization processing		
.data.memclk_setup	*/r_memclk_setup.o (.text .rodata .data)	Memory clock setting processing	FLASH	LRAM
	*/r_*_memclk_setup.o (.text .rodata .data)	Memory clock setting processing for each driver		
.bss.memclk_setup	*/r_memclk_setup.o (.bss COMMON)	Data area without initial values for memory clock settings processing	—	LRAM
	*/r_*_memclk_setup.o (.bss COMMON)	Data area without initial values for memory clock settings processing for each driver		

Table 5.7 Sections and objects to be used in the application program (2/2)

Output section name	Input section name Input object name	Description	Loading area	Execution area
.data	VECTOR_MIRROR_TABLE	Exception processing vector table	FLASH	LRAM
	*/r_intc_*.o (.text .rodata .data)	Code area for INTC driver processing		
	IRQ_FIQ_HANDLER	IRQ/FIQ handler processing		
.bss	None	None	—	LRAM
.uncached_RAM	*/r_cache_*.o (.bss)	Data area without initial values for L1 and L2 cache setting processing*2	—	LRAM
	UNCACHED_BSS	Data area without initial values (a setting in which cache is disabled)		
.uncached_RAM2	*/r_cache_*.o (.text .rodata .data)	L1 and L2 cache setting processing*2	FLASH	LRAM
	UNCACHED_DATA	Data area with initial values (a setting in which cache is disabled)		
.mmu_page_table	None	MMU translation table area	—	LRAM
.stack	None	Stack area for system mode	—	LRAM
		Stack area for IRQ mode		
		Stack area for FIQ mode		
		Stack area for SVC mode		
		Abort (ABT) mode stack area		
.text2	* (.text .text.*)	Code are for defaults	FLASH	FLASH
	* (.rodata .rodata.*)	Constant data area for defaults		
.data2	* (.data .data.*)	Date area with default initial values	FLASH	LRAM
.bss2	* (.bss .bss.*) * (COMMON)	Area for data with default initial values	—	LRAM
.heap	None	Heap area	—	LRAM

Notes: 1. "FLASH" and "LRAM" shown in Load Area and Execution Area indicate the HyperFlash area and the large-capacity on-chip RAM area respectively.
2. This section must be allocated to an area where cache is disabled.

5.3 Interrupts Used

Interrupts are not used in the loader program.

For interrupts used in the application program, refer to the application note "RZ/A2M Group Example of Initialization".

5.4 Data Types

Table 5.8 shows the Data types used in the sample code.

Table 5.8 Data types used in the sample code

Symbol	Description
char_t	8-bit character
bool_t	Boolean type. Value is true (1), false (0).
int_t	Fast integer, signed, 32-bit integer in this sample code.
int8_t	8-bit integer, signed (defined in standard library stdint.h)
int16_t	16-bit integer, signed (defined in standard library stdint.h)
int32_t	32-bit integer, signed (defined in standard library stdint.h)
int64_t	64-bit integer, signed (defined in standard library stdint.h)
uint8_t	8-bit integer, unsigned (defined in standard library stdint.h)
uint16_t	16-bit integer, unsigned (defined in standard library stdint.h)
uint32_t	32-bit integer, unsigned (defined in standard library stdint.h)
uint64_t	64-bit integer, unsigned (defined in standard library stdint.h)
float32_t	32-bit float
float64_t	64-bit float
float128_t	128-bit float

5.5 Constants Used by the Loader Program

Table 5.9 to

Table 5.11 lists the constants used in the sample code.

For the constants used in the application program, refer to the application note "RZ/A2M Group Example of Initialization".

Table 5.9 Constants used in the loader program (1/3)

Constant	Setting value	Description
DRV_SUCCESS	(0)	Normal end
DRV_ERROR	(-1)	Error end
		Specifies the memory space subject to control (CS0/CS1)
HYPERBUS_CS0_AREA	(0)	Specifies the CS0 space control
HYPERBUS_CS1_AREA	(1)	Specifies the CS1 space control
		Specifies the area (memory or register) when the HyperRAM connected to the CS1 space is accessed
HYPERBUS_MEMORY_SPACE	(0)	Sets access to the memory area
HYPERBUS_REGISTER_SPACE	(1)	Sets access to the register area
		Specifies the program to initialize the HyperBus controller and connected devices
HYPERBUS_NO_INIT	(0)	Initialization is not performed
HYPERBUS_INIT_AT_LOADER	(1)	Performs initialization in loader program
HYPERBUS_INIT_AT_APP	(2)	Performs initialization in application program
HYPERBUS_MAXEN_OFF	(0)	Disables specifying HM_CSn# signal Low time using the MCR0 or MCR1 MAXLEN[8:0] bits
HYPERBUS_MAXEN_ON	(1)	Specifies HM_CSn# signal Low time using the MCR0 or MCR1 MAXLEN[8:0] bits
		Specifies the minimum wait time from when HM_CSn# signal has been negated until the start of the next access.
HYPERBUS_CSHI_1P5_CYCLE	(0)	1.5 clock cycles
HYPERBUS_CSHI_2P5_CYCLE	(1)	2.5 clock cycles
HYPERBUS_CSHI_3P5_CYCLE	(2)	3.5 clock cycles
HYPERBUS_CSHI_4P5_CYCLE	(3)	4.5 clock cycles
HYPERBUS_CSHI_5P5_CYCLE	(4)	5.5 clock cycles
HYPERBUS_CSHI_6P5_CYCLE	(5)	6.5 clock cycles
HYPERBUS_CSHI_7P5_CYCLE	(6)	7.5 clock cycles
HYPERBUS_CSHI_8P5_CYCLE	(7)	8.5 clock cycles
HYPERBUS_CSHI_9P5_CYCLE	(8)	9.5 clock cycles
HYPERBUS_CSHI_10P5_CYCLE	(9)	10.5 clock cycles
HYPERBUS_CSHI_11P5_CYCLE	(10)	11.5 clock cycles
HYPERBUS_CSHI_12P5_CYCLE	(11)	12.5 clock cycles
HYPERBUS_CSHI_13P5_CYCLE	(12)	13.5 clock cycles
HYPERBUS_CSHI_14P5_CYCLE	(13)	14.5 clock cycles
HYPERBUS_CSHI_15P5_CYCLE	(14)	15.5 clock cycles
HYPERBUS_CSHI_16P5_CYCLE	(15)	16.5 clock cycles

Table 5.10 Constants used in the loader program (2/3)

Constant	Setting value	Description
HYPERBUS_CSS_1_CYCLE	(0)	Specifies the HM_CSn# set up time from when the HM_CSn# signal is asserted 1 clock cycle
HYPERBUS_CSS_2_CYCLE	(1)	2 clock cycles
HYPERBUS_CSS_3_CYCLE	(2)	3 clock cycles
HYPERBUS_CSS_4_CYCLE	(3)	4 clock cycles
HYPERBUS_CSS_5_CYCLE	(4)	5 clock cycles
HYPERBUS_CSS_6_CYCLE	(5)	6 clock cycles
HYPERBUS_CSS_7_CYCLE	(6)	7 clock cycles
HYPERBUS_CSS_8_CYCLE	(7)	8 clock cycles
HYPERBUS_CSS_9_CYCLE	(8)	9 clock cycles
HYPERBUS_CSS_10_CYCLE	(9)	10 clock cycles
HYPERBUS_CSS_11_CYCLE	(10)	11 clock cycles
HYPERBUS_CSS_12_CYCLE	(11)	12 clock cycles
HYPERBUS_CSS_13_CYCLE	(12)	13 clock cycles
HYPERBUS_CSS_14_CYCLE	(13)	14 clock cycles
HYPERBUS_CSS_15_CYCLE	(14)	15 clock cycles
HYPERBUS_CSS_16_CYCLE	(15)	16 clock cycles

Table 5.11 Constants used in the loader program (3/3)

Constant	Setting value	Description
		Specifies the HM_CSn# negate hold time from access completion
HYPERBUS_CSH_1_CYCLE	(0)	1 clock cycle
HYPERBUS_CSH_2_CYCLE	(1)	2 clock cycles
HYPERBUS_CSH_3_CYCLE	(2)	3 clock cycles
HYPERBUS_CSH_4_CYCLE	(3)	4 clock cycles
HYPERBUS_CSH_5_CYCLE	(4)	5 clock cycles
HYPERBUS_CSH_6_CYCLE	(5)	6 clock cycles
HYPERBUS_CSH_7_CYCLE	(6)	7 clock cycles
HYPERBUS_CSH_8_CYCLE	(7)	8 clock cycles
HYPERBUS_CSH_9_CYCLE	(8)	9 clock cycles
HYPERBUS_CSH_10_CYCLE	(9)	10 clock cycles
HYPERBUS_CSH_11_CYCLE	(10)	11 clock cycles
HYPERBUS_CSH_12_CYCLE	(11)	12 clock cycles
HYPERBUS_CSH_13_CYCLE	(12)	13 clock cycles
HYPERBUS_CSH_14_CYCLE	(13)	14 clock cycles
HYPERBUS_CSH_15_CYCLE	(14)	15 clock cycles
HYPERBUS_CSH_16_CYCLE	(15)	16 clock cycles
		Specifies the read latency during the HyperFlash space access and read/write latency during the HyperRAM space access
HYPERBUS_LTCY_5_CYCLE	(0)	5 clock latency
HYPERBUS_LTCY_6_CYCLE	(1)	6 clock latency
HYPERBUS_LTCY_7_CYCLE	(2)	7 clock latency
HYPERBUS_LTCY_8_CYCLE	(3)	8 clock latency
HYPERBUS_LTCY_9_CYCLE	(4)	9 clock latency
HYPERBUS_LTCY_10_CYCLE	(5)	10 clock latency
HYPERBUS_LTCY_11_CYCLE	(6)	11 clock latency
HYPERBUS_LTCY_12_CYCLE	(7)	12 clock latency
HYPERBUS_LTCY_13_CYCLE	(8)	13 clock latency
HYPERBUS_LTCY_14_CYCLE	(9)	14 clock latency
HYPERBUS_LTCY_15_CYCLE	(10)	15 clock latency
HYPERBUS_LTCY_16_CYCLE	(11)	16 clock latency

Note: The HM_CSn# in the table indicates HM_CS0# or HM_CS1#.

5.6 Structures/Unions

Table 5.12 to

Table 5.13 lists the HyperBus controller initial setting structures used by the loader program.

Table 5.12 Structure for configuring HyperBus controller initial settings (st_hyperbus_cfg_t) (1/2)

Space	Member	Description
CS0	e_hyperbus_init_control init_flag0	Specifies the initialization method for the CS0 space HyperBus controller and HyperFlash HYPERBUS_NO_INIT: Not initialized HYPERBUS_INIT_AT_LOADER: Initialized in loader project HYPERBUS_INIT_AT_APP: Initialized in application project
	e_hyperbus_maxen_t maxen0	Sets whether the HM_CS0# signal Low time setting is controlled by maxlen0. HYPERBUS_MAXEN_OFF: No setting HYPERBUS_MAXEN_ON: Set
	uint16_t maxlen0	Sets HM_CS0# signal maximum read/write process time. 0: 1 clock cycle … 511: 512 clock cycles
	e_hyperbus_cshi_t rcshi0	Sets the time from when HM_CS0# signal has been negated until the start of the next read access. HYPERBUS_CSHI_1P5_CYCLE: 1.5 clock cycles … HYPERBUS_CSHI_16P5_CYCLE: 16.5 clock cycles
	e_hyperbus_cshi_t wcshi0	Sets the time from when HM_CS0# signal has been negated until the start of the next write access. HYPERBUS_CSHI_1P5_CYCLE: 1.5 clock cycles … HYPERBUS_CSHI_16P5_CYCLE: 16.5 clock cycles

Table 5.13 Structure for configuring HyperBus controller initial settings (st_hyperbus_cfg_t) (2/2)

Space	Member	Description
CS0	e_hyperbus_css_t rcss0	Sets the time from when HM_CS0# signal has been asserted until the start of the next read access. HYPERBUS_CSS_1_CYCLE: 1 clock cycle … HYPERBUS_CSS_16_CYCLE: 16 clock cycles
	e_hyperbus_css_t wcss0	Sets the time from when HM_CS0# signal has been asserted until the start of the next write access. HYPERBUS_CSS_1_CYCLE: 1 clock cycle … HYPERBUS_CSS_16_CYCLE: 16 clock cycles
	e_hyperbus_csh_t rcsh0	Sets the time from when read access was completed until the HM_CS0# signal is negated. HYPERBUS_CSH_1_CYCLE: 1 clock cycle … HYPERBUS_CSH_16_CYCLE: 16 clock cycles
	e_hyperbus_csh_t wcsh0	Sets the time from when write access was completed until the HM_CS0# signal is negated. HYPERBUS_CSH_1_CYCLE: 1 clock cycle … HYPERBUS_CSH_16_CYCLE: 16 clock cycles
	e_hyperbus_ltcy_t operate_ltcy0	Sets the read latency during operation. HYPERBUS_LTCY_5_CYCLE: 5 clock latency … HYPERBUS_LTCY_6_CYCLE: 16 clock latency

Table 5.14 to

Table 5.15 lists the HyperBus controller initial setting structures used by the application program.

Table 5.14 Structure for configuring HyperBus controller initial settings (st_hyperbus_cfg_t) (1/2)

Space	Member	Description
CS1	e_hyperbus_init_control init_flag1	Specifies the initialization method for the CS1 space HyperBus controller and HyperRAM HYPERBUS_NO_INIT: Not initialized HYPERBUS_INIT_AT_LOADER: Initialized in loader project HYPERBUS_INIT_AT_APP: Initialized in application project
	e_hyperbus_maxen_t maxen1	Sets whether the HM_CS1# signal Low time setting is controlled by maxlen1. HYPERBUS_MAXEN_OFF: No setting HYPERBUS_MAXEN_ON: Set
	uint16_t maxlen1	Sets HM_CS1# signal maximum read/write process time. 0: 1 clock cycle … 511: 512 clock cycles
	e_hyperbus_cshi_t rcshi1	Sets the time from when HM_CS1# signal has been negated until the start of the next read access. HYPERBUS_CSHI_1P5_CYCLE: 1.5 clock cycles … HYPERBUS_CSHI_16P5_CYCLE: 16.5 clock cycles
	e_hyperbus_cshi_t wcshi1	Sets the time from when HM_CS1# signal has been negated until the start of the next write access. HYPERBUS_CSHI_1P5_CYCLE: 1.5 clock cycles … HYPERBUS_CSHI_16P5_CYCLE: 16.5 clock cycles

Table 5.15 Structure for configuring HyperBus controller initial settings (st_hyperbus_cfg_t) (2/2)

Space	Member	Description
CS1	e_hyperbus_css_t rcss1	Sets the time from when HM_CS1# signal has been asserted until the start of the next read access. HYPERBUS_CSS_1_CYCLE: 1 clock cycle … HYPERBUS_CSS_16_CYCLE: 16 clock cycles
	e_hyperbus_css_t wcss1	Sets the time from when HM_CS1# signal has been asserted until the start of the next write access. HYPERBUS_CSS_1_CYCLE: 1 clock cycle … HYPERBUS_CSS_16_CYCLE: 16 clock cycles
	e_hyperbus_csh_t rcsh1	Sets the time from when read access was completed until the HM_CS1# signal is negated. HYPERBUS_CSH_1_CYCLE: 1 clock cycle … HYPERBUS_CSH_16_CYCLE: 16 clock cycles
	e_hyperbus_csh_t wcsh1	Sets the time from when write access was completed until the HM_CS1# signal is negated. HYPERBUS_CSH_1_CYCLE: 1 clock cycle … HYPERBUS_CSH_16_CYCLE: 16 clock cycles
	e_hyperbus_ltcy_t operate_ltcy1	Sets the read/write latency during operation. HYPERBUS_LTCY_5_CYCLE: 5 clock latency HYPERBUS_LTCY_6_CYCLE: 6 clock latency

5.7 Variables

Table 5.16 shows the List of variables for loader program.

Table 5.16 List of variables for loader program

Variable name	Description	Comment
st_hyperbus_cfg_t HYPERBUS_CFG_TABLE[0]	Settings table data for the HyperBus controller	Refer to Table 6.1

HYPERBUS_CFG_TABLE[0] uses table data for HyerRAM access even in the application program.

5.8 Functions

The sample code comprises interface functions (API functions) for using peripheral functions, user-defined functions (functions called by API functions) which must be prepared by the user for the purpose of the target system, and sample functions which are necessary for the sample code to operate

For the sample code of the loader program, Table 5.17 lists the Sample functions, Table 5.18 lists the API functions, Table 5.19 lists the HyperFlash sample functions, Table 5.20 lists the HyperRAM sample functions, and Table 5.21 lists the User-defined functions.

Table 5.17 Sample functions

Function	Description
reset_handler	Reset handler processing (assembler function)
INITSCT	Program section initialization (assembler function)
R_SC_HardwareSetup	HyperBus controller initial setting
r_memclk_setup	Memory clock setting processing

Table 5.18 API functions

Function	Description
R_HYPERBUS_Setup	Initial setting for the HyperBus controller and the devices connected to HyperBus
R_HYPERBUS_SelectSpace	HyperBus controller access area selection
R_CPG_InitialiseHwIf	CPG initial setting

Table 5.19 HyperFlash sample functions

Function	Description
HyperFlash_ReadVCR	Read function for HyperFlash Volatile Configuration Register (VCR)
HyperFlash_WriteVCR	Write function for HyperFlash Volatile Configuration Register (VCR)
HyperFlash_EraseSect	Erase function for specified HyperFlash sector
HyperFlash_WriteWord	Write function to specified HyperFlash address (1 word (16 bits) unit)
HyperFlash_ReadROMInfo	Read function for HyperFlash device ID and Common Flash Interface (CFI) information

Table 5.20 HyperRAM sample functions

Function	Description
HyperRAM_ReadID0	Read function for HyperRAM Identification Register 0 (ID0)
HyperRAM_ReadID1	Read function for HyperRAM Identification Register 1 (ID1)
HyperRAM_ReadCR0	Read function for HyperRAM Configuration Register 0 (CR0)
HyperRAM_WriteCR0	Write function for HyperRAM Configuration Register 0 (CR0)
HyperRAM_ReadCR1	Read function for HyperRAM Configuration Register 1 (CR1)
HyperRAM_WriteCR1	Write function for HyperRAM Configuration Register 1 (CR1)

Table 5.21 User-defined functions

Function	Description
Userdef_PreHardwareSetup	Necessary hardware initialization processing before the HyperBus controller initialization process
Userdef_PostHardwareSetup	Necessary hardware initialization processing after the HyperBus controller initialization process

5.9 Function Specification

Specifications of the functions of the sample code loader program are listed below.

reset_handler	
Outline	Reset handler processing
Declaration	reset_handler
Description	The entry function of the loader program and the application program. The setting process to enable high-speed access to HyperFlash is performed in the loader program. The setting process to enable high-speed access to HyperRAM is performed in the application program.
Argument	None
Return Value	None

INITSCT		
Outline	Program section initialization	
Declaration	void INITSCT(void)	
Description	The data with initial values which must be transferred to the RAM area (including code and constant data that must be executed in the RAM area) is transferred from the ROM area, and the initialization of the RAM area data with no initial values.	
Argument	p_dtbl	: Pointer to the area where the section information for data with initial values is stored
	p_btbl	: Pointer to the area where the section information for data with no initial values is stored
Return Value	None	

R_SC_HardwareSetup	
Outline	HyperBus controller initial setting
Declaration	void R_SC_HardwareSetup(void)
Description	The loader program makes the optimal settings for the used HyperFlash. — CPU and peripheral clock setting — Makes the HyperBus controller and HyperFlash register settings to enable high-speed access to HyperFlash. The application program makes the optimal settings for the used HyperRAM. — Makes the HyperBus controller and HyperRAM register settings to enable high-speed access to HyperRAM.
Argument	None
Return Value	None
Precautions	This function cannot be assigned to execute from HyperFlash or HyperRAM. This function must be assigned to an area other than HyperFlash or HyperRAM.

r_memclk_setup	
Outline	Memory clock setting processing
Declaration	void r_memclk_setup (void)
Description	Before the R_SC_HardwareSetup function is executed, the memory clock setting is performed. For HyperFlash boot, memory clock setting is not performed here.
Argument	None
Return Value	None
Precautions	This function cannot be assigned to execute from HyperFlash. This function must be assigned to an area other than HyperFlash.

R_HYPERBUS_Setup	
Outline	Initial setting for the HyperBus controller and the devices connected to HyperBus
Declaration	void R_HYPERBUS_Setup (void)
Description	In the loader program, the HYPERBUS controller and HyperFlash register initial setting is performed. In the application program, the HYPERBUS controller and HyperRAM register initial setting is performed.
Argument	None
Return Value	None
Precautions	This function cannot be assigned to execute from HyperFlash or HyperRAM. This function must be assigned to an area other than HyperFlash or HyperRAM.

R_HYPERBUS_SelectSpace		
Outline	HyperBus controller access area selection	
Declaration	int_t R_HYPERBUS_SelectSpace(e_hyperbus_access_area_t area, e_hyperbus_space_select_t space)	
Description	HyperBus controller access area is selected. When accessing the HyperRAM register, specify HYPERBUS_REGISTER_SPACE to the argument space to call this function. When accessing the HyperRAM memory area, specify HYPERBUS_MEMORY_SPACE to the argument space.	
Argument	e_hyperbus_access_area_t area	: Specifies space accessed by HyperBus HYPERBUS_CS0_AREA (Use prohibited) HYPERBUS_CS1_AREA
	e_hyperbus_space_select_t space	: Specifies the HyperRAM access target area HYPERBUS_MEMORY_SPACE HYPERBUS_REGISTER_SPACE
Return Value	DRV_SUCCESS	: Normal end
	DRV_ERROR	: Error end
Precautions	This function cannot be assigned to execute from HyperRAM. This function must be assigned to an area other than HyperRAM. This function can only be set for the HyperRAM space.	

R_CPG_InitialiseHwIf		
Outline	CPG initial setting	
Declaration	int_t R_CPG_InitialiseHwIf(void)	
Description	Uses the r_cpg_drv_sc_cfg.h CPG configuration data to make CPG register (FRQCR, CKIOSEL, SCLKSEL) settings. In this sample code, CPG settings are made so that the operating frequency in "Operating clock settings" and "HYPLCK selection" in "Table 5.2 Settings for the boot startup on-chip ROM program and loader program" is used.	
Argument	None	None
Return Value	DRV_SUCCESS	: Normal end
	DRV_ERROR	: r_cpg_drv_sc_cfg.h CPG configuration data is wrong
Precautions	This function must be assigned to the large-capacity on-chip RAM.	

HyperFlash_ReadVCR		
Outline	Read function for the HyperFlash Volatile Configuration Register (VCR)	
Declaration	uint16_t HyperFlash_ReadVCR(uint32_t baddr)	
Description	Reads the HyperFlash Volatile Configuration Register (VCR), and responds with the read value as the return value.	
Argument	uint32_t baddr	HyperFlash space base address
Return Value	Value read from the Volatile Configuration Register	
Precautions	This function cannot be assigned to execute from HyperFlash. This function must be assigned to an area other than HyperFlash. If this function is called when MMU is enabled, specify an address of the HyperFlash space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.	

HyperFlash_WriteVCR		
Outline	Write function for the HyperFlash Volatile Configuration Register (VCR)	
Declaration	int_t HyperFlash_WriteVCR (uint32_t baddr, uint16_t wdata)	
Description	Writes the value specified in the wdata argument to the HyperFlash Volatile Configuration Register.	
Argument	uint32_t baddr	HyperFlash space base address
	uint16_t wdata	Value written to the Volatile Configuration Register
Return Value	DRV_SUCCESS	: Normal end
Precautions	This function cannot be assigned to execute from HyperFlash. This function must be assigned to an area other than HyperFlash. If this function is called when MMU is enabled, specify an address of the HyperFlash space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.	

HyperFlash_EraseSect		
Outline	Erase function for specified HyperFlash sector	
Declaration	int_t HyperFlash_EraseSect (uint32_t baddr, uint32_t saddr)	
Description	Erases the HyperFlash sector specified in the saddr argument.	
Argument	uint32_t baddr	HyperFlash space base address
	uint32_t saddr	Address for Hyper Flash sector to be erased
Return Value	DRV_SUCCESS	: Normal end
	DRV_ERROR	: Error end
Precautions	This function cannot be assigned to execute from HyperFlash. This function must be assigned to an area other than HyperFlash. If this function is called when MMU is enabled, specify an address of the HyperFlash space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.	

HyperFlash_WriteWord		
Outline	Write function to specified HyperFlash address (1 word (16bits) unit)	
Declaration	uint16_t HyperFlash_WriteWord (uint32_t baddr, uint32_t waddr, uint16_t wdata)	
Description	Writes the value specified in the wdata argument to the HyperFlash address specified in the waddr argument.	
Argument	uint32_t baddr	HyperFlash space base address
	uint32_t waddr	HyperFlash address to be written
	uint16_t wdata	Data written to HyperFlash
Return Value	DRV_SUCCESS	: Normal end
Precautions	This function cannot be assigned to execute from HyperFlash. This function must be assigned to an area other than HyperFlash. If this function is called when MMU is enabled, specify an address of the HyperFlash space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.	

HyperFlash_ReadROMInfo		
Outline	Read function for HyperFlash device ID and Common Flash Interface (CFI) information	
Declaration	void HyperFlash_ReadROMInfo (uint32_t baddr, uint16_t *idbuf)	
Description	Reads the HyperFlash device ID ad CFI information.	
Argument	uint32_t baddr	HyperFlash space base address
	uint16_t *idbuf	Start address of the buffer (16 bits × 256 words) that stores the device ID and CFI information
Return Value	None	
Precautions	This function cannot be assigned to execute from HyperFlash. This function must be assigned to an area other than HyperFlash. If this function is called when MMU is enabled, specify an address of the HyperFlash space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument. For details on the device ID and CFI information, refer to the data sheet for the used HyperFlash.	

HyperRAM_ReadID0	
Outline	Read function for HyperRAM Identification Register 0 (ID0)
Declaration	uint16_t HyperRAM_ReadID0(uint32_t baddr)
Description	Reads the HyperRAM Identification Register 0, and responds with the read value as the return value.
Argument	uint32_t baddr HyperRAM space base address
Return Value	Value read from the Identification Register 0
Precautions	This function cannot be assigned to execute from HyperRAM. This function must be assigned to an area other than HyperRAM. If this function is called when MMU is enabled, specify an address of the HyperRAM space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.

HyperRAM_ReadID1	
Outline	Read function for HyperRAM Identification Register 1 (ID1)
Declaration	uint16_t HyperRAM_ReadID1(uint32_t baddr)
Description	Reads the HyperRAM Identification Register 1, and responds with the read value as the return value.
Argument	uint32_t baddr HyperRAM space base address
Return Value	Value read from the Identification Register 1
Precautions	This function cannot be assigned to execute from HyperRAM. This function must be assigned to an area other than HyperRAM. If this function is called when MMU is enabled, specify an address of the HyperRAM space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.

HyperRAM_ReadCR0	
Outline	Read function for HyperRAM Configuration Register 0 (CR0)
Declaration	uint16_t HyperRAM_ReadCR0(uint32_t baddr)
Description	Reads the HyperRAM Configuration Register 0, and responds with the read value as the return value.
Argument	uint32_t baddr HyperRAM space base address
Return Value	Value read from the Configuration Register 0
Precautions	This function cannot be assigned to execute from HyperRAM. This function must be assigned to an area other than HyperRAM. If this function is called when MMU is enabled, specify an address of the HyperRAM space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.

HyperRAM_WriteCR0		
Outline	Write function for HyperRAM Configuration Register 0 (CR0)	
Declaration	int_t HyperRAM_WriteCR0 (uint32_t baddr, uint16_t wdata)	
Description	Writes the data specified in the wdata argument to the HyperRAM Configuration Register 0.	
Argument	uint32_t baddr	HyperRAM space base address
	uint16_t wdata	Data written to the Configuration Register 0
Return Value	DRV_SUCCESS	: Normal end
Precautions	This function cannot be assigned to execute from HyperRAM. This function must be assigned to an area other than HyperRAM. If this function is called when MMU is enabled, specify an address of the HyperRAM space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.	

HyperRAM_ReadCR1		
Outline	Read function for HyperRAM Configuration Register 1 (CR1)	
Declaration	uint16_t HyperRAM_ReadCR1(uint32_t baddr)	
Description	Reads the HyperRAM Configuration Register 1, and responds with the read value as the return value.	
Argument	uint32_t baddr	HyperRAM space base address
Return Value	Value read from the Configuration Register 1	
Precautions	This function cannot be assigned to execute from HyperRAM. This function must be assigned to an area other than HyperRAM. If this function is called when MMU is enabled, specify an address of the HyperRAM space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.	

HyperRAM_WriteCR1		
Outline	Write function for HyperRAM Configuration Register 1 (CR1)	
Declaration	int_t HyperRAM_WriteCR1 (uint32_t baddr, uint16_t wdata)	
Description	Writes the data specified in the wdata argument to the HyperRAM Configuration Register 1.	
Argument	uint32_t baddr	HyperRAM space base address
	uint16_t wdata	Data written to the Configuration Register 1
Return Value	DRV_SUCCESS	: Normal end
Precautions	This function cannot be assigned to execute from HyperRAM. This function must be assigned to an area other than HyperRAM. If this function is called when MMU is enabled, specify an address of the HyperRAM space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.	

Userdef_PreHardwareSetup	
Outline	Necessary hardware initialization processing before the HyperBus controller initialization process
Declaration	void Userdef_PreHardwareSetup (void)
Description	This is the user definable function to describe the hardware initialization process which is required to be executed before the HyperBus controller initialization. It is called at the start of the R_SC_HardwareSetup function. In the loader program, the CPG initial setting function is called. In the application program, the IOKEEP bit is cleared, and the pin state which was saved when recovering from deep standby mode is cancelled.
Argument	None
Return Value	None
Precautions	If this function contains processes that cannot be executed from HyperFlash or HyperRAM, this function must be assigned to an area other than HyperFlash or HyperRAM area.

Userdef_PostHardwareSetup	
Outline	Necessary hardware initialization processing after the HyperBus controller initialization process
Declaration	void Userdef_PostHardwareSetup (void)
Description	This is the user definable function to describe the hardware initialization process which is required to be executed after the HyperBus controller initialization. It is called at the end of the R_SC_HardwareSetup function. Nothing is performed in the loader program. In the application program, the write enable for each page of the on-chip data retention RAM is set to write enabled.
Argument	None
Return Value	None
Precautions	If this function contains processes that cannot be executed from HyperFlash or HyperRAM, this function must be assigned to an area other than HyperFlash or HyperRAM area.

5.10 Loader Program Flowcharts

5.10.1 Loader Program – Overall

Figure 5.6 shows the Flowchart of loader program (overall).

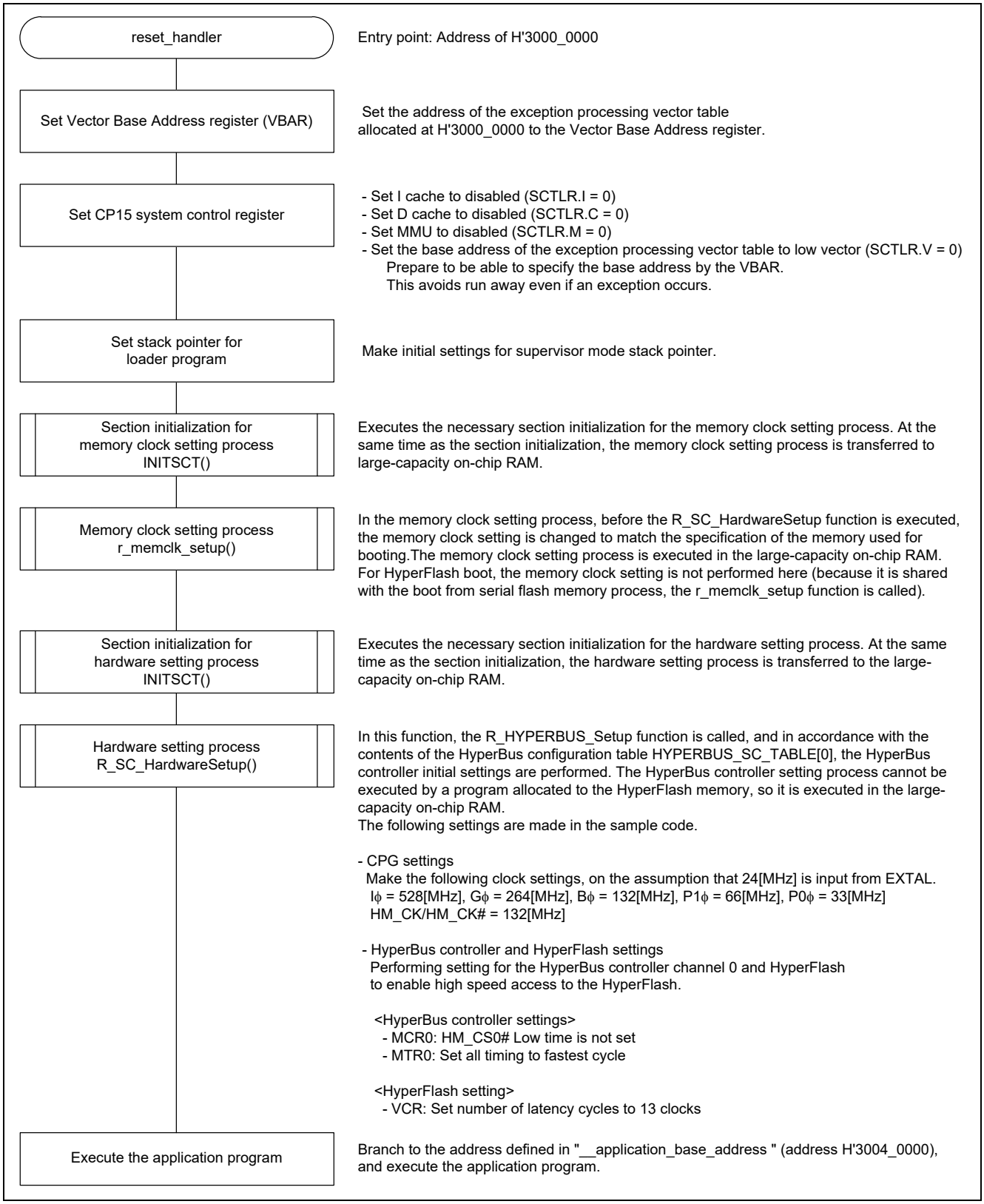


Figure 5.6 Flowchart of loader program (overall)

5.10.2 R_SC_HardwareSetup Function

This function performs the hardware initial setting. In the loader program, the settings of HyperBus controller and HyperFlash are performed to enable high-speed access to HyperFlash. The clocks supplied to HyperFlash (HM_CK/HM_CK#) are also set.

Because a program allocated to the HyperFlash space cannot perform the setting process of HyperBus controller registers and HyperFlash registers, the program is loaded into the large-capacity on-chip RAM and executed from there.

Figure 5.7 shows the R_SC_HardwareSetup (loader program) flowchart.

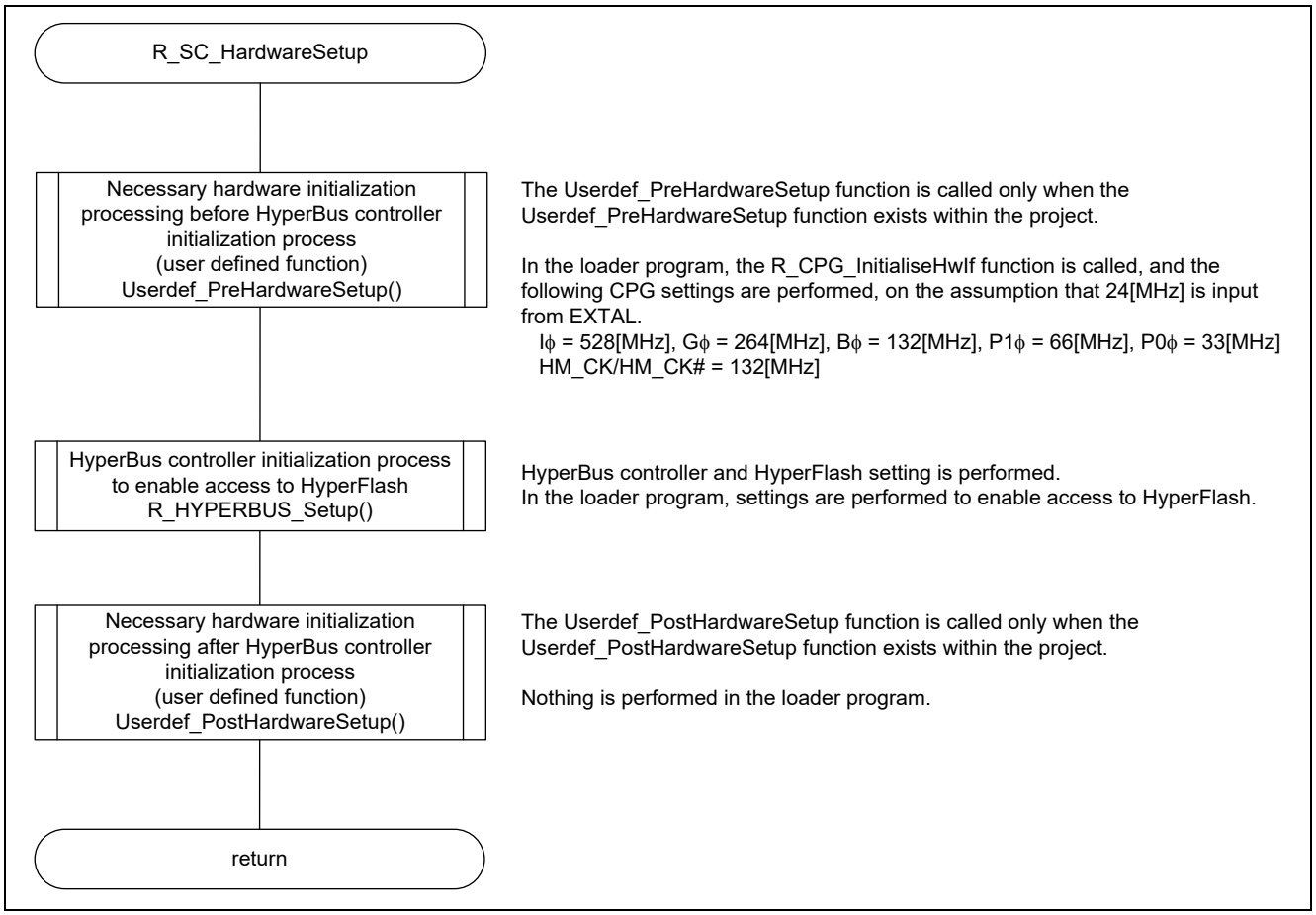


Figure 5.7 R_SC_HardwareSetup (loader program) flowchart

5.10.3 HyperBus Controller Initial Setting

Figure 5.8 shows the R_HYPERBUS_Setup (loader program) flowchart.

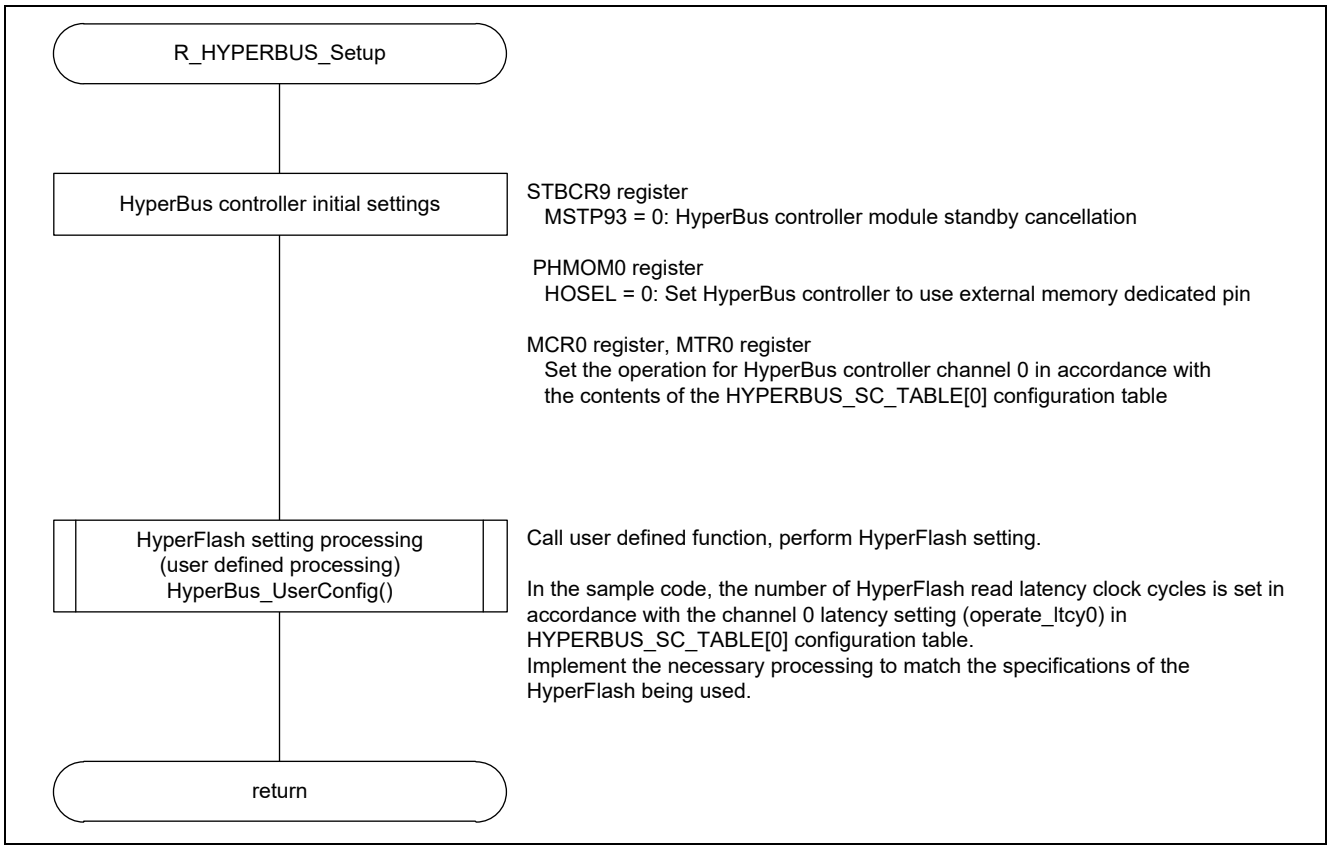


Figure 5.8 R_HYPERBUS_Setup (loader program) flowchart

5.10.4 HyperFlash setting

Figure 5.9 shows the HyperBus_UserConfig function (loader program) flowchart. This function is a user-defined function. Implement the necessary processing to match the specifications of the HyperFlash being used.

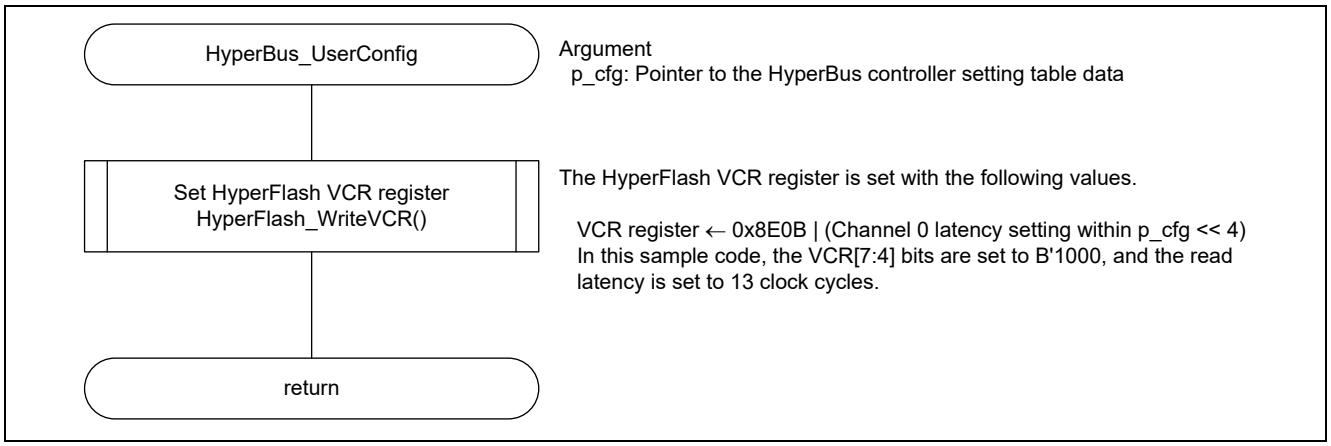


Figure 5.9 HyperBus_UserConfig function (loader program) flowchart

5.11 Application Program Flowchart

5.11.1 Application Program – Overall

For details regarding the application program, refer to the application note "RZ/A2M Group Example of Initialization".

In this application program, in addition to the above, the HyperRAM initial setting is performed and a write to the HyperRAM area where cache is disabled is performed.

5.11.2 R_SC_HardwareSetup Function

This function performs the hardware initial setting. In the application program, the HyperBus controller and HyperRAM settings are performed to enable access to the HyperRAM connected to the HyperBus controller.

This function is loaded into the large-capacity on-chip RAM and executed from there.

Figure 5.10 shows the R_SC_HardwareSetup (application program) flowchart.

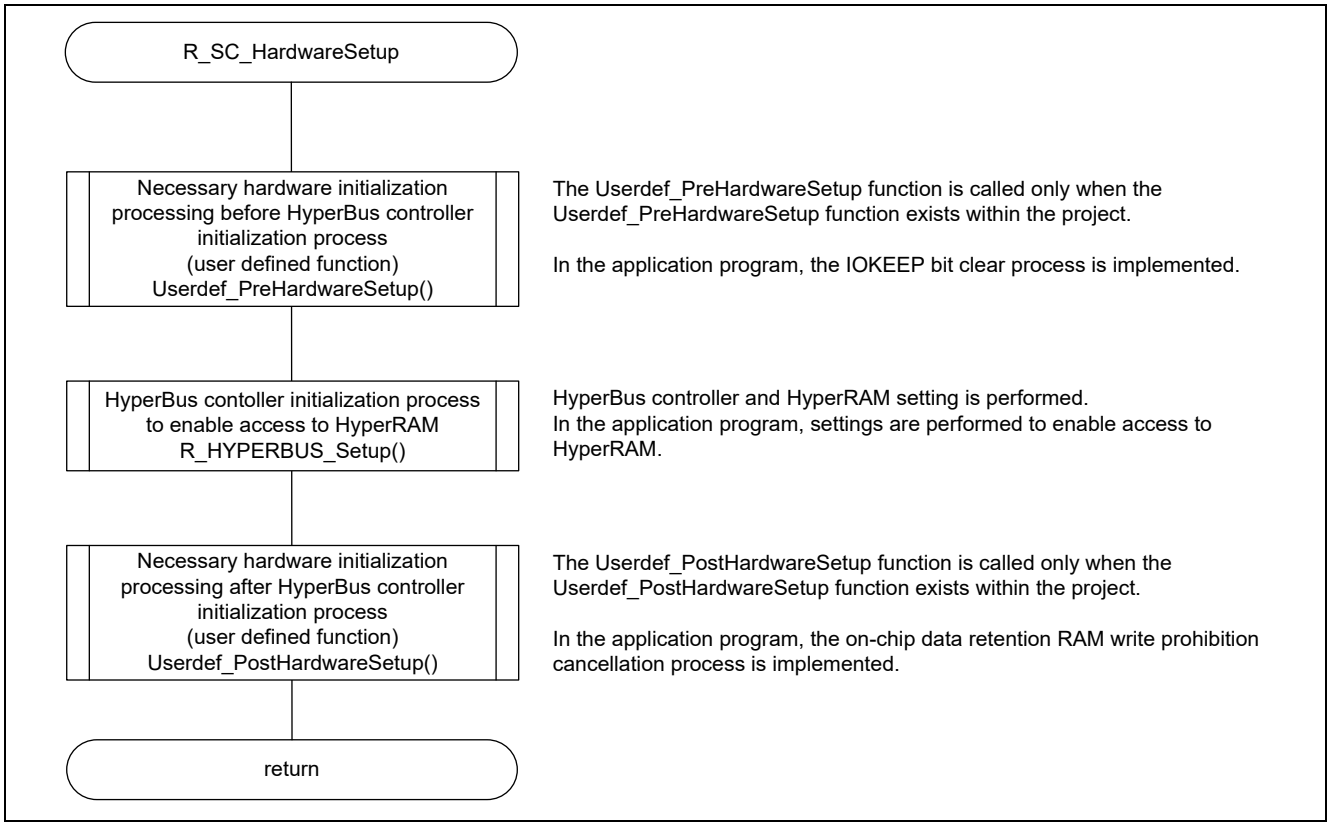


Figure 5.10 R_SC_HardwareSetup (application program) flowchart

5.11.3 HyperBus Controller Initial Setting

Figure 5.11 shows the R_HYPERBUS_Setup (application program) flowchart.

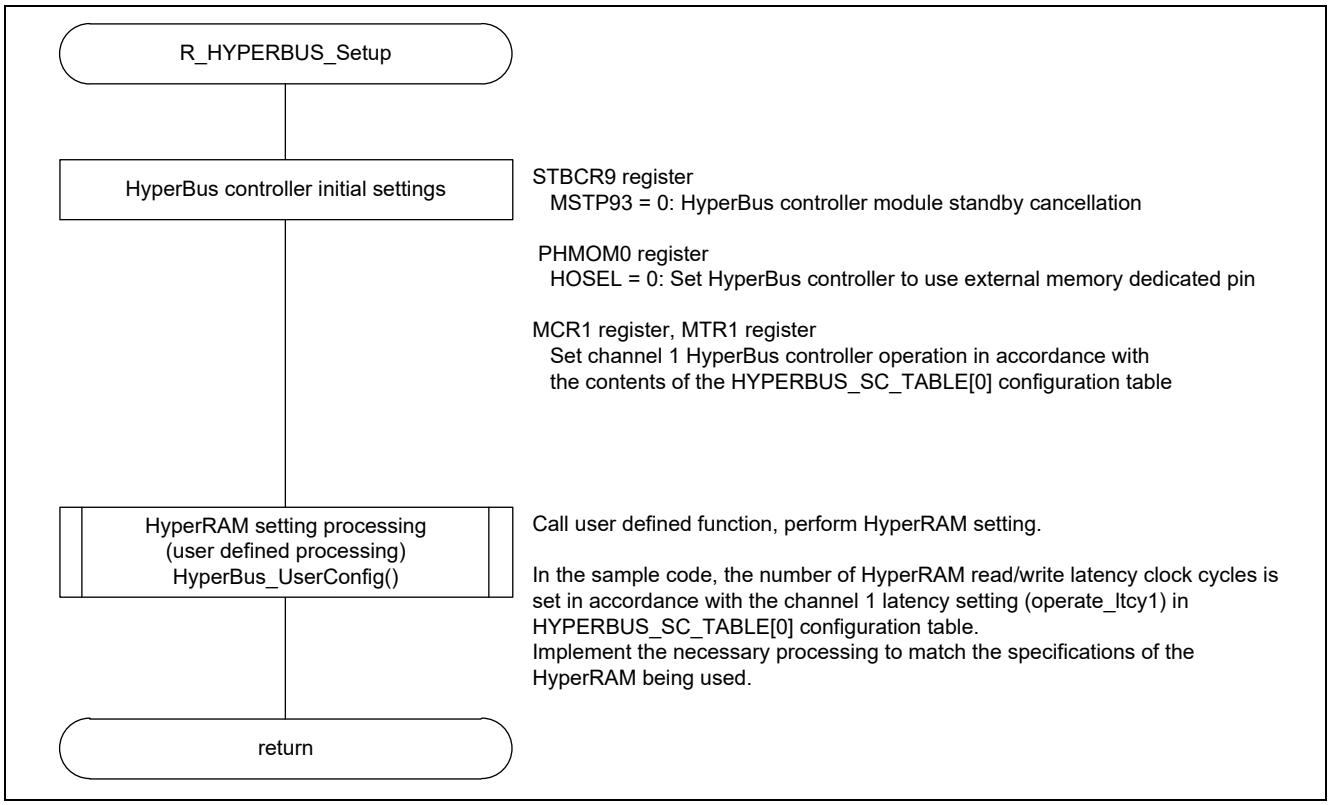


Figure 5.11 R_HYPERBUS_Setup (application program) flowchart

5.11.4 HyperRAM Initial Setting

Figure 5.12 shows the HyperBus_UserConfig function (application program) flowchart. This function is a user-defined function. Implement the necessary processing to match the specifications of the HyperRAM being used.

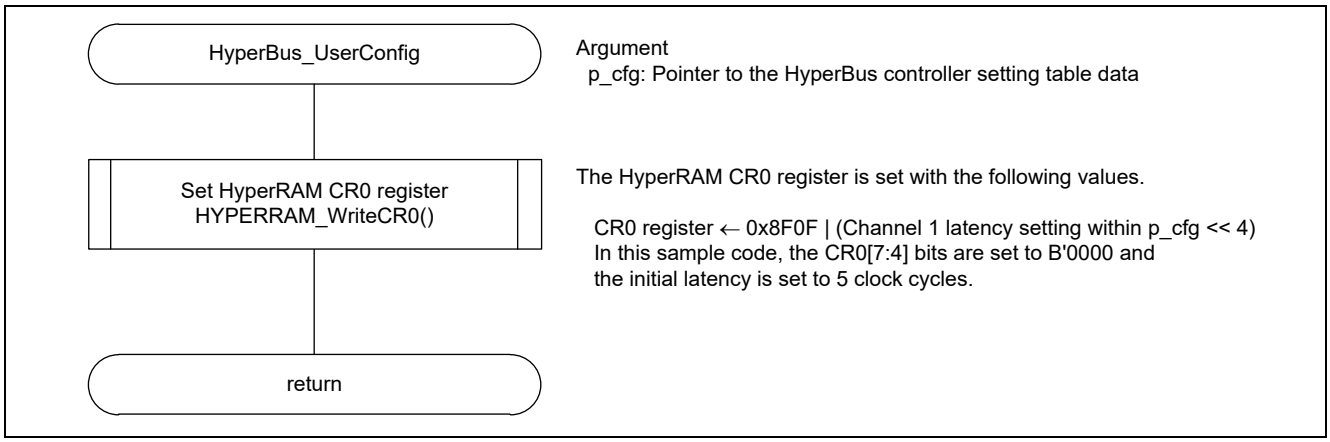
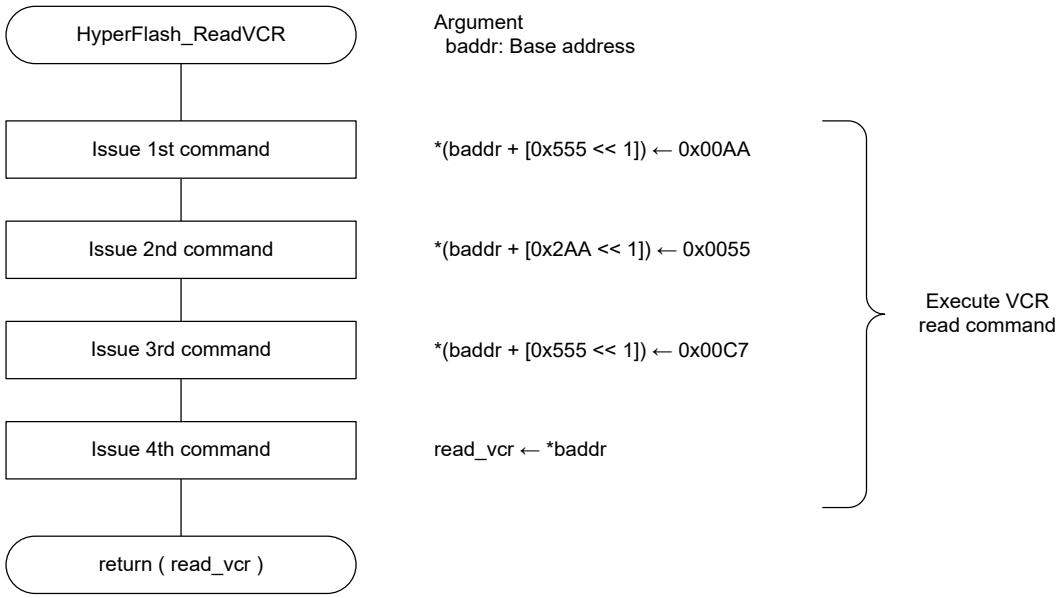


Figure 5.12 HyperBus_UserConfig function (application program) flowchart

5.12 Function for HyperFlash and HyperRAM Access Flowchart

5.12.1 Read function for HyperFlash Volatile Configuration Register

The flow chart for Read function for HyperFlash Volatile Configuration Register (VCR) is shown in Figure 5.13.

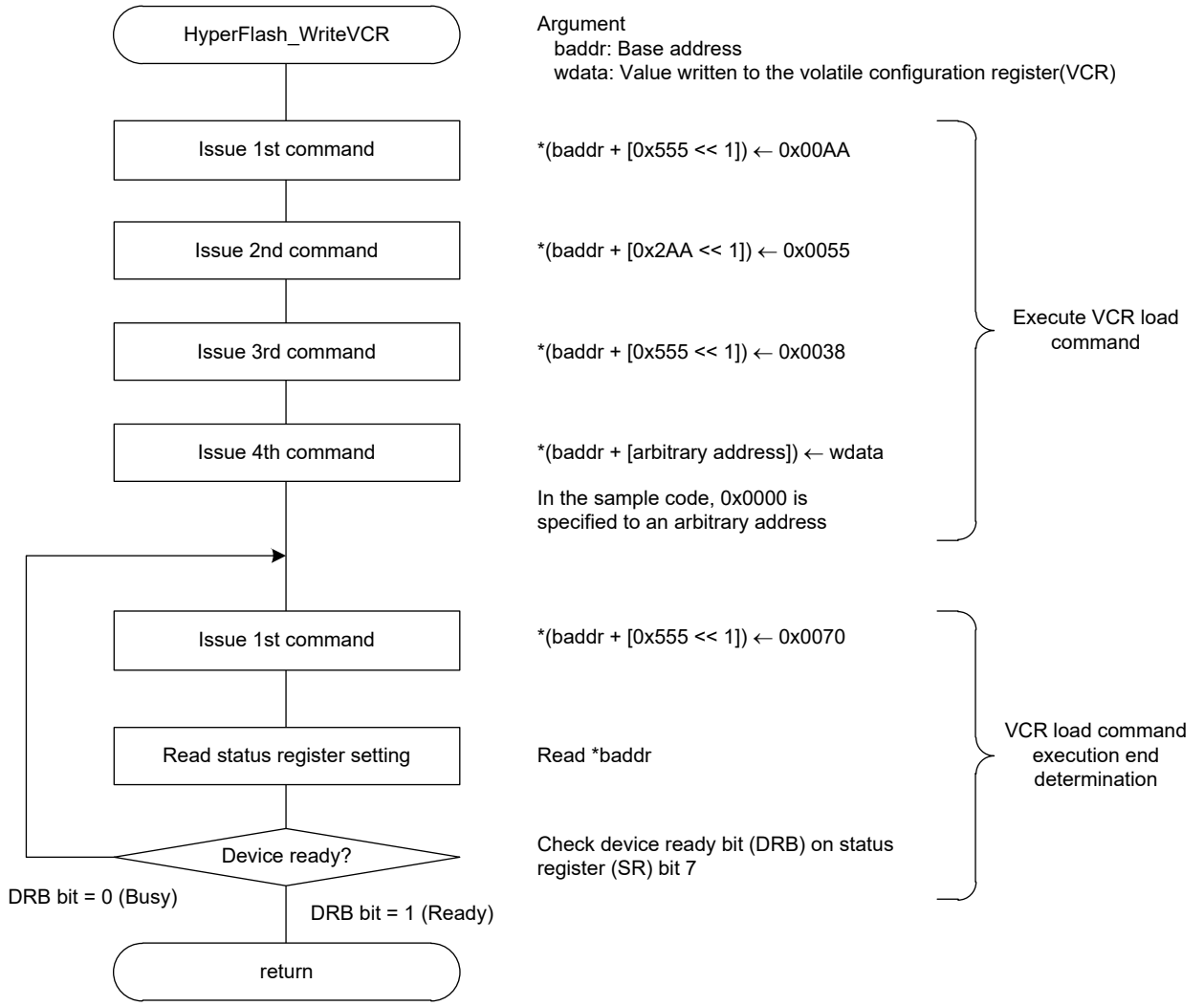


Note: If this function is called when MMU is enabled, specify an address of the HyperFlash space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.

Figure 5.13 Read function for HyperFlash Volatile Configuration Register (VCR)

5.12.2 Write Function for HyperFlash Volatile Configuration Register (VCR)

The flow chart for Write function for HyperFlash Volatile Configuration Register (VCR) is shown in Figure 5.14.

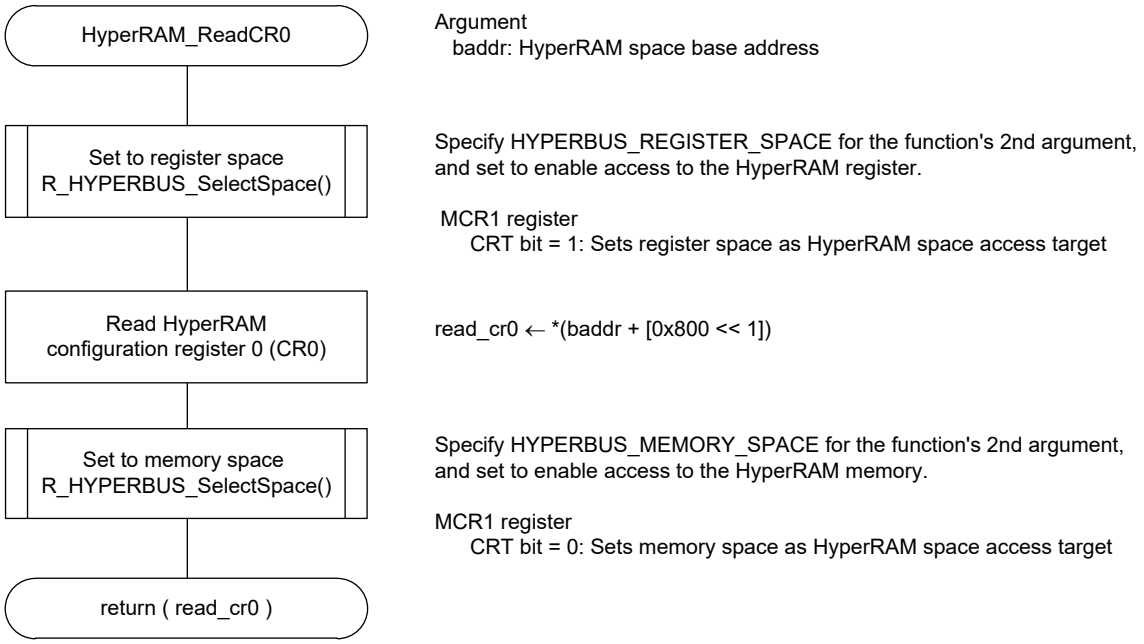


Note: If this function is called when MMU is enabled, specify an address of the HyperFlash space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.

Figure 5.14 Write function for HyperFlash Volatile Configuration Register (VCR)

5.12.3 Read Function for HyperRAM Configuration Register 0 (CR0)

The flow chart for Read function for HyperRAM Configuration Register 0 (CR0) is shown in Figure 5.15.

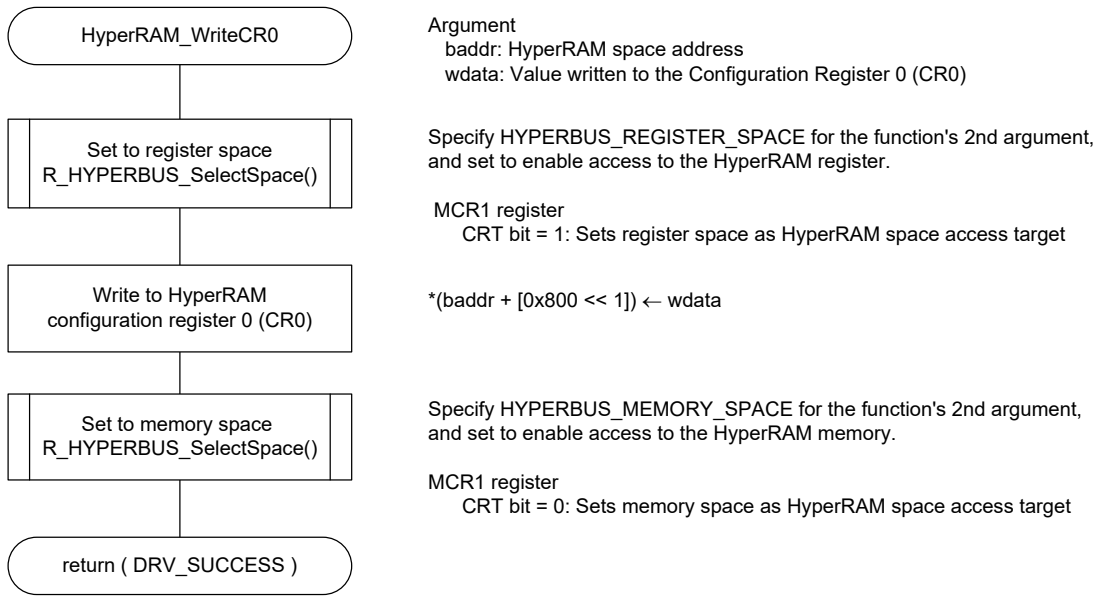


Note: If this function is called when MMU is enabled, specify an address of the HyperRAM space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.

Figure 5.15 Read function for HyperRAM Configuration Register 0 (CR0)

5.12.4 Write Function for HyperRAM Configuration Register 0 (CR0)

The flow chart for Write function for HyperRAM Configuration Register 0 (CR0) is shown in Figure 5.16.



Note: If this function is called when MMU is enabled, specify an address of the HyperRAM space where the strongly-ordered attribute or device attribute is set as the memory attribute in the baddr argument.

Figure 5.16 Write function for HyperRAM Configuration Register 0 (CR0)

5.12.5 Function to Specify Area (Memory or Register) when Accessing HyperRAM

The flow chart for Function to specify area (memory or register) when accessing HyperRAM is shown in Figure 5.17.

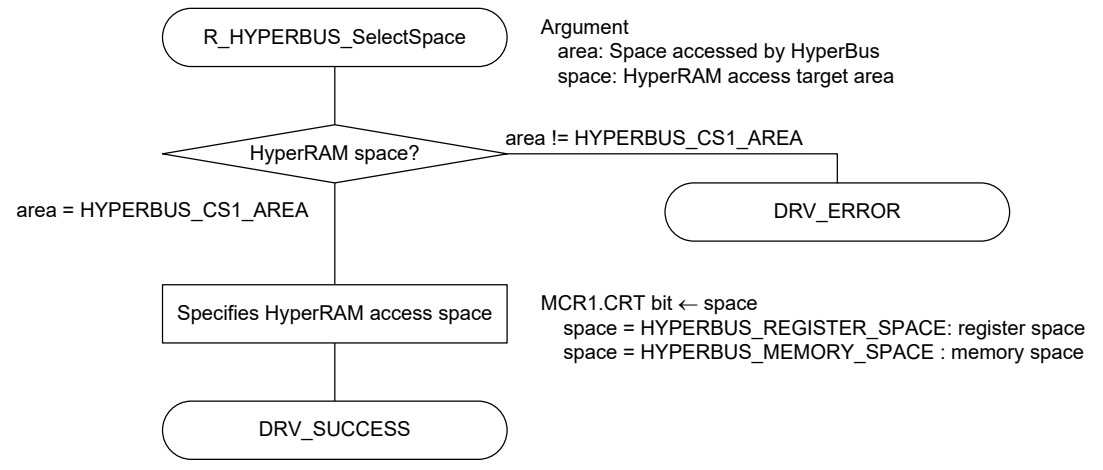


Figure 5.17 Function to specify area (memory or register) when accessing HyperRAM

6. Application Example

6.1 HyperBus Controller Setting

The contents of the HyperBus controller setting table data are referenced and set to the HyperBus controller. Table 6.1 shows the Setting values for HyperBus controller setting table data (HYPERBUS_SC_TABLE[0]) of the sample code. The Hyperbus controller initial setting function (R_HYPERBUS_Setup) uses this table data, and performs the settings for CS0 memory configuration register (MCR0), CS1 memory configuration register (MCR1), CS0 memory timing register (MTR0), and CS1 memory timing register (MTR1), which are related to the Hyperbus controller operation timing.

Implement the necessary settings to match the specifications of the HyperFlash and HyperRAM being used.

Table 6.1 Setting values for HyperBus controller setting table data (HYPERBUS_SC_TABLE[0])

Space	Member	Setting value	Supported register
HyperFlash	e_hyperbus_init_control init_flag0	HYPERBUS_INIT_AT_LOADER: Initialized in loader program	—
	e_hyperbus_maxen_t maxen0	HYPERBUS_MAXEN_OFF: No setting	MCR0.MAXEN
	uint16_t maxlen0	0: 1 clock cycle	MCR0.MAXLEN[8:0]
	e_hyperbus_cshi_t rcshi0	HYPERBUS_CSHI_1_5: 1.5 clock cycles	MTR0.RCSHI[3:0]
	e_hyperbus_cshi_t wcshi0	HYPERBUS_CSHI_1_5: 1.5 clock cycles	MTR0.WCSHI[3:0]
	e_hyperbus_css_t rcss0	HYPERBUS_CSS_1: 1 clock cycle	MTR0.RCSS[3:0]
	e_hyperbus_css_t wcss0	HYPERBUS_CSS_1: 1 clock cycle	MTR0.WCSS[3:0]
	e_hyperbus_csh_t rcsh0	HYPERBUS_CSH_1: 1 clock cycle	MTR0.RCSH[3:0]
	e_hyperbus_csh_t wcsh0	HYPERBUS_CSH_1: 1 clock cycle	MTR0.WCSH[3:0]
	e_hyperbus_ltcy_t operate_ltcy0	HYPERBUS_LTCY_13: 13 clock latency	—
HyperRAM	e_hyperbus_init_control init_flag1	HYPERBUS_INIT_AT_APP: Initialized in application program	—
	e_hyperbus_maxen_ t maxen1	HYPERBUS_MAXEN_OFF: No setting	MCR1.MAXEN
	uint16_t maxlen1	0: 1 clock cycles	MCR1.MAXLEN[8:0]
	e_hyperbus_cshi_t rcshi1	HYPERBUS_CSHI_1_5: 1.5 clock cycles	MTR1.RCSHI[3:0]
	e_hyperbus_cshi_t wcshi1	HYPERBUS_CSHI_1_5: 1.5 clock cycles	MTR1.WCSHI[3:0]
	e_hyperbus_css_t rcss1	HYPERBUS_CSS_1: 1 clock cycle	MTR1.RCSS[3:0]
	e_hyperbus_css_t wcss1	HYPERBUS_CSS_1: 1 clock cycle	MTR1.WCSS[3:0]
	e_hyperbus_csh_t rcsh1	HYPERBUS_CSH_1: 1 clock cycle	MTR1.RCSH[3:0]
	e_hyperbus_csh_t wcsh1	HYPERBUS_CSH_1: 1 clock cycle	MTR1.WCSH[3:0]
	e_hyperbus_ltcy_t operate_ltcy1	HYPERBUS_LTCY_5: 5 clock latency	MTR1.LTCY[3:0]

6.2 HyperFlash Latency Clock Setting

Table 6.2 shows the RZ/A2M CPU board's on-board HyperFlash latency clock and maximum operating frequency relationship. In the sample code, the setting values of xVCR[7:4] bits in the Volatile Configuration Register (VCR) are changed, and the HyperFlash latency is set to 13 cycles to achieve a shortest latency at 132 MHz of the operating frequency in the loader program processing.

Set the HyperBus controller and HyperFlash latency clock to match the specifications of the used HyperFlash.

Table 6.2 HyperFlash latency clock and maximum operating frequency

Latency code xVCR[7:4]	Latency clock	Maximum operating frequency (MHz)
0000	5	52
0001	6	62
0010	7	72
0011	8	83
0100	9	93
0101	10	104
0110	11	114
0111	12	125
1000	13	135
1001	14	145
1010	15	156
1011	16	166 (factory setting)
1110 to 1111	Reserved	Not applicable

Note: The latency codes shown in the table are the setting values for the HyperFlash VCR/NVCR bits 7 to 4.

6.3 HyperRAM Latency Clock Setting

Table 6.3 shows the RZ/A2M CPU board's on-board HyperRAM latency clock and maximum operating frequency relationship. In the sample code, the setting values of CR0[7:4] bits are changed, and the HyperRAM latency is set to 5 cycles to achieve a shortest latency at 132 MHz of the operating frequency in the application program processing.

Set the HyperBus controller and HyperRAM latency clock to match the specifications of the used HyperRAM.

Table 6.3 HyperRAM latency clock and maximum operating frequency

Latency code CR0[7:4]	Latency clock	Maximum operating frequency (MHz)
0000	5	133
0001	6	166 (factory setting)
0010 to 1101	Reserved	Not applicable
1110	3	83
1111	4	100

Note: The latency codes shown in the table are the setting values for the Configuration Register 0 (CR0) bits 7 to 4 in the HyperRAM register base.

7. Sample Code Precautions

7.1 HyperFlash and HyperRAM Register Access

If the register for a device connected to the HyperBus controller (HyperFlash or HyperRAM) is accessed when MMU is enabled, it is necessary to access the space for each device set as strongly-ordered attribute or device attribute as the memory attribute.

In the sample code, an area where cache is enabled and an area where cache is disabled are prepared for each of the HyperFlash space and the HyperRAM space.

In the HyperRAM space, an area where cache is enabled and an area where cache is disabled are prepared as normal memory attribute areas. Even when one of the areas is accessed, the HyperRAM register is not accessed correctly. When the HyperRAM register is accessed, prepare an area set as strongly-ordered attribute or device attribute in the HyperRAM space and then access.

In the HyperFlash space, an area where cache is enabled is prepared in the normal memory attribute and an area where cache is disabled is prepared in the strongly-ordered attribute. When accessing the HyperFlash register, access the area where cache is disabled in the HyperFlash space (address H'A000 0000 to H'AFFF FFFF).

Furthermore, even when writing to the memory area of HyperFlash, same as with the register access, it is necessary to access the areas set with the strongly-ordered attribute or device attribute for the HyperFlash space.

8. Sample Code

Sample code can be downloaded from the Renesas Electronics Website.

9. Reference Documents

- User's Manual: Hardware
  RZ/A2M Group User's Manual: Hardware
  (The latest version can be downloaded from the Renesas Electronics Website.)

  RTK7921053C00000BE (RZ/A2M CPU board) User's Manual
  (The latest version can be downloaded from the Renesas Electronics Website.)

  RTK79210XXB00000BE (RZ/A2M SUB board) User's Manual
  (The latest version can be downloaded from the Renesas Electronics Website.)

  Arm Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
  (The latest version can be downloaded from the Arm Website.)

  Arm Cortex™-A9 Technical Reference Manual Revision: r4p1
  (The latest version can be downloaded from the Arm Website.)

  Arm Generic Interrupt Controller Architecture Specification - Architecture version2.0
  (The latest version can be downloaded from the Arm Website.)

  Arm CoreLink™ Level 2 Cache Controller L2C-310 Technical Reference Manual Revision: r3p3
  (The latest version can be downloaded from the Arm Website.)
- Technical Update/Technical News
  (The latest information can be downloaded from the Renesas Electronics Website.)
- User's Manual: Integrated development environment
  The e^2 studio Integrated Development Environment user's manual can be downloaded from the Renesas Electronics Website.
  (The latest version can be downloaded from the Renesas Electronics Website.)

Revision History

Rev.	Date	Description	
		Page	Summary
Rev.1.00	Dec.20.19	—	First edition issued

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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. 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.4.0-1 November 2017)

Corporate Headquarters

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

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/.

Trademarks

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