User's Manual

32

RH850G4MH

Virtualization User's Manual: Hardware

Renesas microcontroller

RH850 Family

Renesas Electronics
www.renesas.com

Rev.1.40 Aug. 2023

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

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.

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

   Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to power supply or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.
5. 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.).
6. 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.
7. Power ON/OFF sequence

   In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current. The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

Table of Contents

Section 1 Virtualization support function

This CPU supports the Virtualization support function.

This section describes the characteristics of virtualization support function usable in this CPU.

1.1 Outline of virtualization support function

By using the virtualization support function of this CPU, it is easy to build a virtual machine with virtualization software.

CAUTIONS

1. **In this document, the software that manages virtual machines is called virtualization software. It is also referred to as a hypervisor or a virtual machine monitor (VMM: Virtual Machine Monitor).**
2. **This document describes the operations in the Virtualization Support Function enabled state, unless it is explicitly noted as disabled.**

1.1.1 Virtualization system

Figure 1.1 shows an example of the software operation on this CPU when virtualization support function is not used (Conventional system) and an example of the software operation on this CPU when virtual machine is built using this CPU's virtualization support function (Virtualization system).

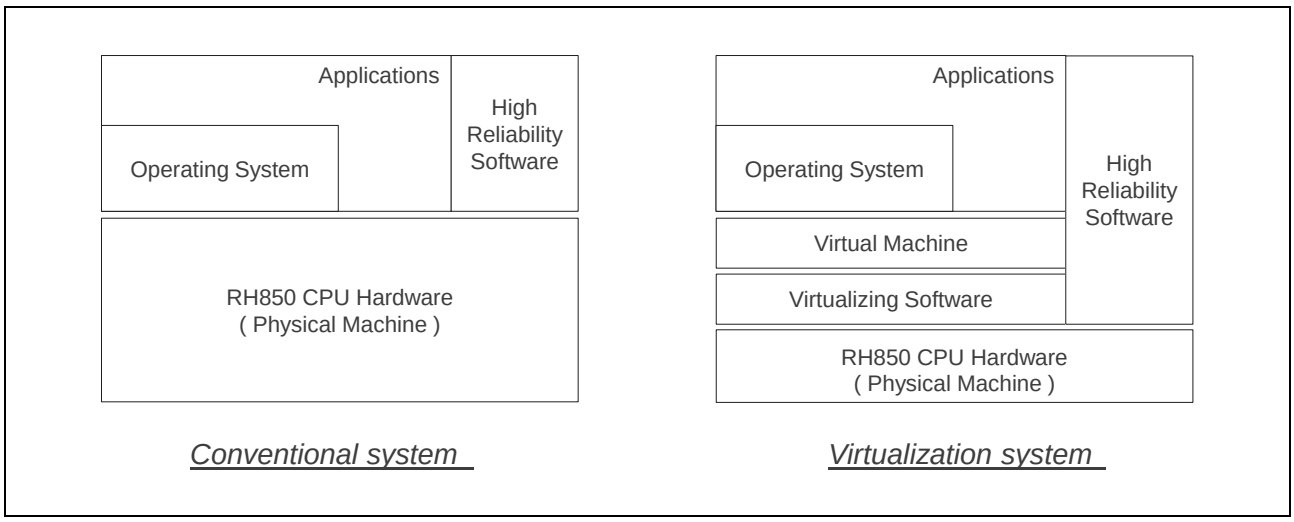


Figure 1.1 Comparison of Conventional system and Virtualization system

In a conventional system, management software such as operating system can operate all functions of CPU. Therefore, when the reliability of the management software is low (incomplete software, low level of security concerning external purchases, etc.), if the management software performs an illegal operation, not only the inside of the CPU becomes illegal, an illegal operation also spreads outside the CPU. As a result, it may be necessary to reset the control system that incorporates the CPU on which the management software is installed. This will adversely affect the stable operation of the system. As a result, a low reliability of the management software running on one of the CPUs constituting the control system reduces the reliability of the entire control system.

In one virtualization system, management software and application software that operates under its control operate on a virtual machine managed by virtualization software. From the management software running on the virtual machine, it looks like all the functions of the CPU can be operated like the conventional system. However, the virtualization software can use the virtualization support function to impose motion constraints on software running on the virtual machine. If the management software with low reliability does some illegal operations, the inside of the virtual machine is still in an incorrect state, but the influence of illegal operation to the outside of the CPU is suppressed by various constraints given by this virtualization software. Also, because the detected illegal operation is notified to the virtualization software, the virtualization software can take some actions such as stopping the management software that performed the illegal operation or restarting it. Resetting the system is unnecessary even if illegal operation is performed, leading to stable operation of the system. As a result, the unreliability of the management software running on the virtual machine of one of the CPUs constituting the control system does not affect the reliability of the entire control system.

In either system, software with high reliability that does not require operation constraints may be operating independently of management software and virtualization software.

In addition, in this CPU, it is also possible to build multiple virtual machines on one CPU. **Figure 1.2** shows a case where one virtual machine is built on this CPU and another case where two virtual machines are built on this CPU.

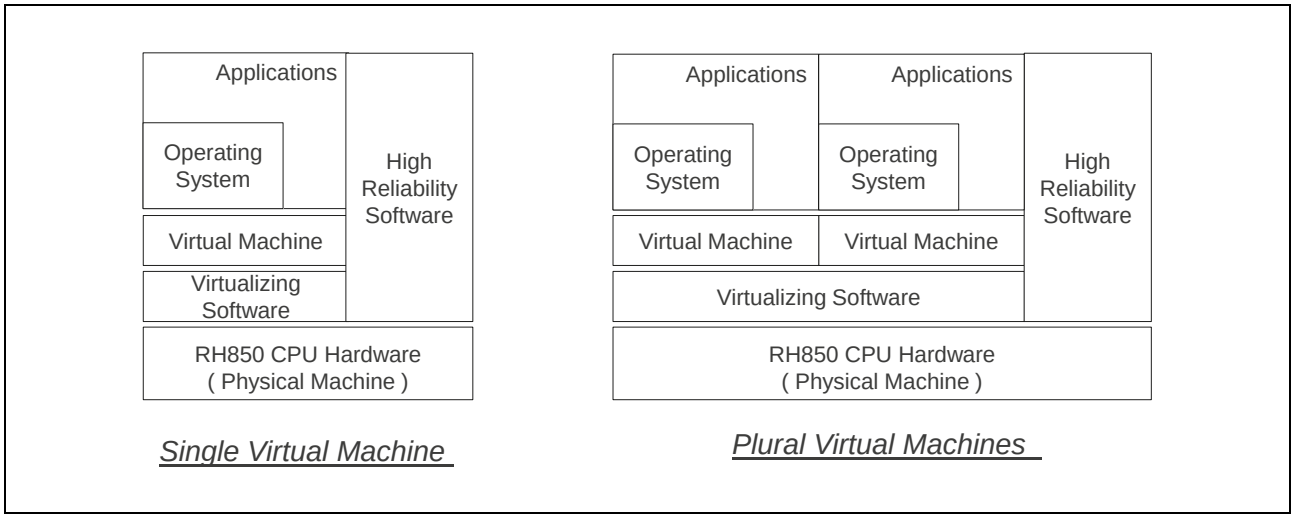


Figure 1.2 Construction of multiple virtual machines

Multiple virtual machines can be executed in parallel by the virtualization software switching the context of the virtual machine. In order to easily realize this, the virtualization support function of this CPU consists of a hardware identifier that can identify up to eight virtual machines, a memory management function for sharing an address space among multiple virtual machines or for exclusive use, and a context switching support function.

1.1.2 Paravirtualization and Partition Function

The virtualization system realized using the virtualization support function of this CPU is paravirtualization. Therefore, in order for the management software running on the conventional system compatible with this CPU to operate on the virtualization system of this CPU, it may be necessary to change the software. The necessary changes are out of the scope of this document since they depend on the CPU function used by the software to be changed and the function of the virtual machine built by the virtualization software.

To realize paravirtualization (resource separation and sharing), this CPU provides a partition function as a virtualization support function. The partition function enables a virtualization software environment without interference between virtual machines to be realized.

The partition function of this CPU imposes hardware-based restrictions on software operating on this CPU for the use of CPU resources. Virtualization software can use this partition function to realize the virtual machine as the operating environment of the software under this constraint.

1.1.3 Enabling virtualization support function

The virtualization support function of this CPU can be enabled or disabled by system register setting.

When the virtualization support function is disabled, this CPU can be used as a conventional system. When the virtualization support function is enabled, this CPU can be used as a virtualization system. For details on enabling and disabling virtualization support function, see **Section 2.1, CPU Operating Modes**.

1.1.4 Host mode and Guest mode

When virtualization support function of this CPU is enabled, Host mode and Guest mode can be used as new CPU operation modes. When virtualization support function is enabled, the CPU is in either Host mode or Guest mode.

Host mode and Guest mode are defined independently of the supervisor mode and user mode available in the conventional system. Therefore, there are four combinations of operation modes.

In Host mode, it is possible to operate all CPUs resources including virtualization support function. In Guest mode, most of the CPUs resources equivalent to the conventional system can be operated, but the virtualization support function or resources not related to the virtual machine can not be operated. Also, Host mode can restrict some operations of Guest mode, with the partition function provided by the virtualization support function. A guest partition is defined as the operating environment of the software at the Guest mode, constrained by the partition function.

For details of operation modes including Host mode and Guest mode, see **Section 2.1, CPU Operating Modes**.

1.2 Occupied and Shared resources

The allocation method of each resources is determined by the specification of virtualization software. This CPU provides separation means to allocate each resource to virtual machines. For general-purpose registers and other resources for which the CPU has no separation means, it may be necessary to replace the context at virtual machine switching.

Table 1.1 shows a list of resources of this CPU.

Table 1.1 Resource list

Resources	Location*1	Separation means
Flash memory	Outside CPU	MPU*2, Guard mechanism*3
Local memory	Inside CPU	MPU*2, Guard mechanism*3
Cluster memory	Outside CPU	MPU*2, Guard mechanism*3
Peripheral IO	Outside CPU	MPU*2, Guard mechanism*3
Interrupt channels	Outside CPU	MPU*2, Guard mechanism*3, INTC1 function*4
General purpose registers	Inside CPU	None
Vector registers	Inside CPU	Whether available or not can be specified*5
System registers	Inside CPU	Partial multiplexing*6

Note 1. For resources inside of CPU, specifications are defined in this document. Resources outside of CPU are basic resources enumerated as RH850 products, and specifications including mounting feasibility are not defined in this document. For details, see the hardware manual of the product used.

Note 2. The MPU (memory protection function) can restrict the address space that can be accessed. For details, see **Section 5, Memory Management**.

Note 3. Guard mechanism for access restriction may be implemented as specification of product. For details, including occupied and shared methods, see the hardware manual of the product used.

Note 4. Interrupt channel separation method is implemented as INTC (interrupt controller) specifications. For details, see the "Interrupts" section in the hardware manual of the product used.

Note 5. Host mode can specify whether to use in Guest mode. For details, see **Section 2.4.1, Coprocessor Use Permissions**.

Note 6. Some system registers are multiplexed in Host mode only or in Guest mode only. For details, see **Section 2.5.7, System Register Multiplexing**.

1.3 Functional difference based on activation of Virtualization Support Function

When the virtualization support function is enabled, there will be some functional differences compared to the case in which the function is disabled. **Table 1.2** shows the details of these differences.

Table 1.2 Functional difference based on activation of virtualization support function

Function	Overview	Reference
CPU function mode	Host mode and Guest mode are available.	**2.1**
Access authority	HV privilege to operate virtualization support function are available.	**2.1.3**
Access authority of system registers	There are the registers which access authority is changed.	**2.5.3**
Instruction execution authority	There are the instructions which execution authority is changed.	**5.2.6**
How to specify an exception handler address	There are restrictions on how to use table reference method interrupts in Host mode. There is a restriction on how to set the base address in Guest mode.	**4.2.1** **4.4.1**
Exception cause	Exceptions for effectively using the partition function can be used.	**4.1.2**
Exception acceptance condition	Conditions for accepting the above exception causes	**4.1.1**
Nesting memory protection function	Memory protection function dedicated to Guest mode that can be changed only in Host mode can be used.	**5.1.7**
Virtualization support function system register	System registers to control virtualization support functions can be used.	**3.10**
Host Context Register	Multiplexed system registers to improve efficiency of the context register switching when switching between Host mode and Guest mode can be used.	**3.11**
Guest Context Register		**3.12**
Memory protection function System register	Memory protection function for realizing partition function can be used.	**3.6**
Virtualization support function instruction	Instructions for efficiently realizing virtualization system can be used.	*[1]

Note 1. See the *RH850G4MH Virtualization User's Manual: Software*.

Section 2 Processor Model

This CPU adopts a processor model that has basic operation functions, registers, and an exception management function.

This section describes the unique features of the processor model of this CPU.

2.1 CPU Operating Modes

There are three independent operating modes in this CPU.

The first mode is the virtualization operating mode indicating the use state of the virtualization support function. The virtualization operating mode has two states: the conventional mode in which the virtualization support function is disabled and the virtualization mode in which the virtualization support function is enabled. The virtualization operating mode is indicated by the value of HVCFG.HVE.

- Conventional mode (HVCFG.HVE = 0): The virtualization support function is disabled.
- Virtualization mode (HVCFG.HVE = 1): The virtualization support function is enabled.

The second mode is the restricted operating mode indicating the operating state of the partition function. Restricted operating mode has two states: Host mode (HM) and Guest mode (GM). The restricted operating mode is indicated by the value of PSWH.GM.

- Host mode (PSWH.GM = 0): Partition function is disabled.
- Guest mode (PSWH.GM = 1): Partition function is enabled.

The third mode is the authority operating mode indicating the software operating authority. The authority operating mode has two states: Supervisor mode (SV) and User mode (UM). The authority operating mode is indicated by the value of PSW.UM.

- Supervisor mode (PSW.UM = 0): Access authority is high; CPU resources can be managed and operated.
- User mode (PSW.UM = 1): Access authority is low, and the use of CPU resources is restricted.

The restricted operating mode can be enabled only when the virtualization operating mode is virtualization mode. When the virtualization operating mode is the conventional mode, the function related to the state of the restricted operating mode can not be used. The authority operating mode can be enabled regardless of the state of other operating modes. However, detailed software operating authority is defined by combination with other operating modes. For details, see **Section 2.1.3, CPU Operating Mode and Privileges**.

2.1.1 Definition of CPU Operating Modes

2.1.1.1 Virtualization operating mode

See the hardware manual of the product used for the virtualization operating mode after reset release.

(1) Conventional mode

In the conventional mode, the virtualization support function can not be used. The RIE exception occurs when an instruction of the virtualization support function is about to be executed. The system registers of the virtualization support function are handled as undefined registers and operations are impossible. However, the HVCFG register indicating the state of the virtualization operating mode can be operated.

(2) Virtualization mode

The virtualization mode is a state in which virtualization support function can be used. The function related to the restricted operating mode is enabled only when the virtualization operating mode is virtualization mode. The operating restrictions applied to the software operating in virtualization mode depends on the state of the restricted operating mode and the authority operating mode.

2.1.1.2 Restricted operating mode

The initial mode of restricted operating mode is always Host mode.

(1) Host mode (HM)

In Host mode, the partition function is disabled. At this time, if the authority operating mode is supervisor mode, the software can operate all CPU resources including virtualization support function. In addition, it is possible to set an operating restriction that is enabled when changing to Guest mode.

(2) Guest mode (GM)

In Guest mode, the partition function is enabled. Operating restrictions which can not be changed from software running in Guest mode may be set.

2.1.1.3 Authority operating mode

The authority operating mode after reset release is always Supervisor mode.

(1) Supervisor mode (SV)

In Supervisor mode, the access authority is high and CPU resource can be managed and operated. Operable CPU resources depend on the state of the virtualization operating mode and restricted operating mode. For details, see **Section 2.1.3, CPU Operating Mode and Privileges**. In addition, it is possible to set an operating restriction that is enabled when changing to user mode.

(2) User mode (UM)

In User mode, the access authority is low and operation of CPU resources is restricted. Operating restrictions which can not be changed from software operating in User mode may be set. Software running in User mode can use the instructions and system registers defined as user resources.

2.1.2 CPU Operating Mode Transition

The CPU operating mode changes due to three events.

(1) Change due to Acknowledging an Exception

When an exception to change to Host mode is accepted in Guest mode, restricted operating mode transits to Host mode. In Guest mode, if an exception which does not change to the Host mode is accepted, the restricted operating mode will remain in Guest mode. In Host mode, if an exception is accepted, the restricted operating mode will remain in the Host mode. There is no exception to change from Host mode to Guest mode. See **Section 4.1.1, Exception Cause List** for details of exception to enter Host mode.

When an exception is acknowledged in User mode, the authority operating mode changes to Supervisor mode. When an exception is acknowledged in Supervisor mode, the authority operating mode remains in Supervisor mode. When accepting exceptions, the virtualization operating mode does not change.

(2) Change due to a Return Instruction

By executing a return instruction, the value of PSWH is restored according to the value of the corresponding bit saved in EIPSWH, FEPSWH. Restoring PSWH by a return instruction is only possible when the restricted operating mode is Host mode. Therefore, only the change from Host mode to Guest mode is possible for the change of the restricted operating mode by a return instruction.

Also, by executing a return instruction, the value of PSW is restored according to the value of the corresponding bit saved in EIPSW, FEPSW. Execution of a return instruction is possible when the software operating authority is HV privilege or SV privilege. This means the authority operating mode is Supervisor mode. Therefore, it is only possible to change from Supervisor mode to User mode for the change of authority operating mode by a return instruction.

For details of operating authority, see **Section 2.1.3, CPU Operating Mode and Privileges**. Note that the virtualization operating mode does not change by execution of a return instruction.

(3) Change due to a System Register Instruction

The virtualization operating mode changes by directly rewriting HVCFG.HVE by the LDSR instruction. In the conventional mode, the virtualization operating mode can be changed when the operating privilege is SV privilege and in the virtualization mode, it can be changed when the operating privilege is HV privilege. The authority operating mode is changed by directly writing PSW.UM by the LDSR instruction. On the other hand, the restricted operating mode can not be changed by directly writing PSWH.GM by the LDSR instruction.

CAUTIONS

1. **The state change in the virtualization operating mode has a big influence on the operation of software. Therefore, although HVCFG.HVE can be changed by the LDSR instruction, make the change immediately after reset release. Immediately after reset release is the first process in the reset handler, it indicates the timing before changing the setting of other system registers and memory access as an operand. The following operation is not supported: Dynamically changing HVCFG.HVE after this CPU completes the initialization process in the reset handler and the virtualization software or application software has started operating. Note that to change HVCFG.HVE by LDSR instruction, synchronization processing by SYNCI instruction is necessary. For details, see Section 7.3, Hazard Management after System Register Update.**
2. **In the state change of the restricted operating mode, synchronization processing similar to that of the SYNCM instruction is performed when accepting a cause exception and when executing a return instruction. As a result, state change is performed after all load and store operations which were executed before the state change are completed. For details of synchronization operation, see Section 7.3, Hazard Management after System Register Update.**
3. **The CPU operating mode cannot be changed in user mode because the higher-order 31 to 5 bits of the PSW register cannot be overwritten; it can be changed in supervisor mode. This CPU guarantees that if an LDSR instruction is used to update the PSW register, the new setting will be reflected when the subsequent instruction is executed. However, this CPU does not guarantee that the new setting will be reflected in the memory protection by the MPU for instruction fetch of the subsequent instruction. Therefore, for changing the higher-order 31 to 5 bits of the PSW register, it is recommended to use a return instruction. For details, see Section 7.3, Hazard Management after System Register Update.**

2.1.3 CPU Operating Mode and Privileges

In this CPU, the usable functions can be restricted according to usage permission settings for specific resources and the CPU operating mode. The execution of certain instructions and the operation of certain system registers can only be performed when there is the permission required for it. The permissions necessary to execute these specific instructions are called “privileges”. If the privilege for executing instructions or operating system registers is not given, these operations cannot be performed but an exception occurs.

The four types of authority defined by this CPU are as follows.

- Hypervisor (HV) privilege: HV privilege is the privilege required for execution of an instruction whose execution authority is HV privilege (HV privileged instruction) and operation of system registers whose access authority is HV privilege
- Supervisor (SV) privilege: SV privilege is the privilege required for execution of an instruction whose execution authority is SV privilege (SV privileged instruction) and operation of system registers whose access authority is SV privilege. Execution of HV privileged instruction and operation of system registers whose access authority is HV privilege can not be performed when CPU authority is SV privilege.
- User (UM) authority: UM authority is an authority that enables execution of instructions (user instructions) whose execution authority is not specially specified, and operation of system registers whose access authority is UM authority. Execution of HV privileged instruction or SV privileged instruction or operation of system registers whose access authority is HV privilege or SV privilege can not be executed when CPU authority is UM authority.
- Coprocessor use permissions: Permissions necessary to use a coprocessor

CAUTION

HV privilege also includes authority of SV privilege. Therefore, when CPU authority is HV privilege, software can execute SV privileged instructions and operate system registers whose access authority is SV privilege. On the other hand, coprocessor use permissions are independent of HV privilege and SV privilege. Therefore, even if the CPU authority is HV privilege or SV privilege, you can not use the coprocessor unless you have coprocessor use permissions.

Figure 2.1 shows the CPU operating mode transition and CPU authority transition.

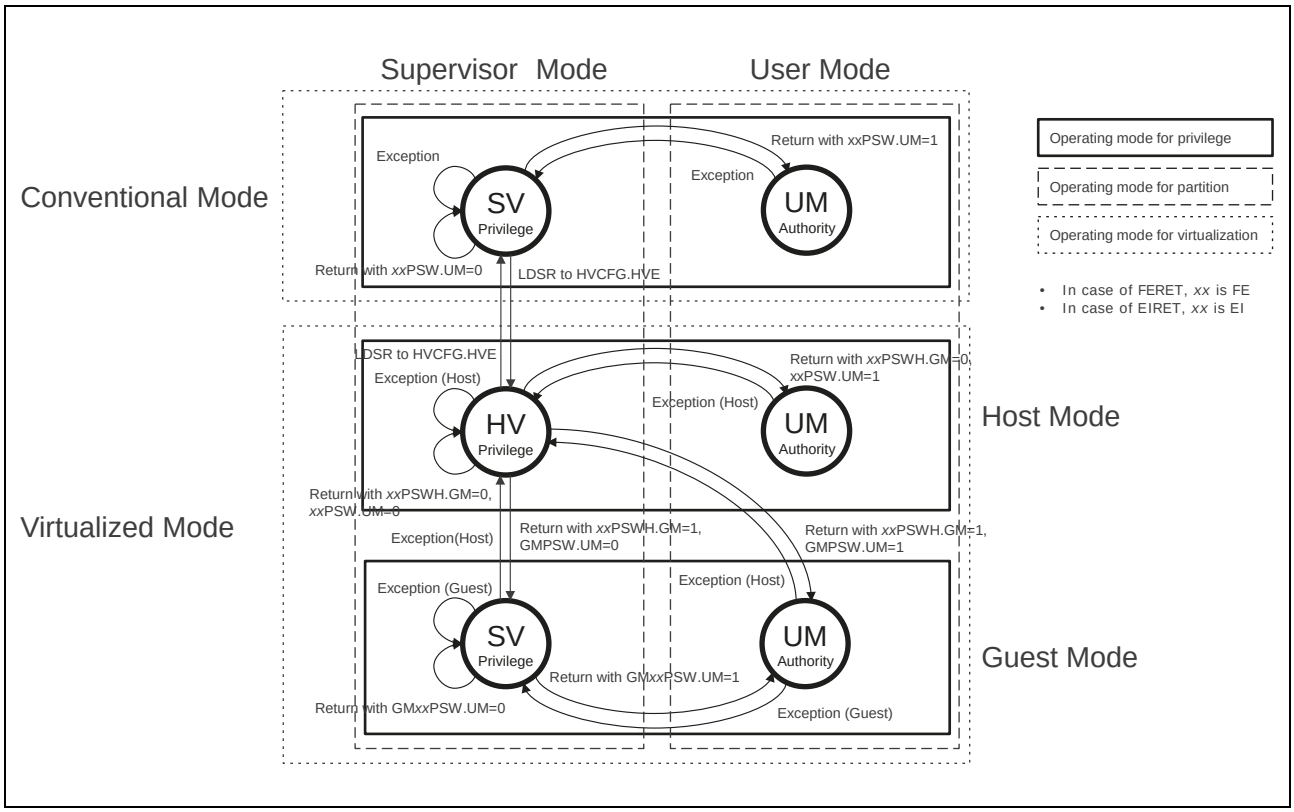


Figure 2.1 Relationship between CPU operating mode and privilege

Table 2.1 shows the relationship between the CPU operating modes and the authorities of the CPU defined accordingly.

Table 2.1 Definition of CPU operating mode and CPU authority

CPU operating mode			
HVCFG.HVE	**PSWH.GM**	**PSW.UM**	**CPU authority**
0	—	0	SV privilege
0	—	1	UM authority
1	0	0	HV privilege
1	0	1	UM authority*1
1	1	0	SV privilege
1	1	1	UM authority

Note 1. It operates as host mode for the function to be applied according to the state of the restricted operating mode. It operates as UM authority for functions applied according to CPU authority.

CAUTION

HV privilege is enabled only when the virtualization operating mode is virtualization mode. When the virtualization operating mode is the conventional mode, if an HV privilege instruction is executed, there are cases where RIE exceptions occur and cases where the instruction can be executed if the SV privilege is enabled. And when the virtualization operating mode is the conventional mode, if a system register that requires HV privilege as the access authority is operated, it may be handled as an undefined register, or it may be operable if there is SV privilege. For details, see each instructions and system register specifications.

(1) Hypervisor privilege (HV privilege)

The privilege necessary for operation of the virtualization support function is called hypervisor privilege (HV privilege). When the restricted operating mode is Host mode and the privilege operating mode is Supervisor mode, CPU authority is HV privilege. It is a privilege necessary for building and managing virtual machines and managing CPU resources outside the virtual machine. In HV privilege, virtualization software is assumed to be executed.

(2) Supervisor Privilege (SV Privilege)

The privilege necessary to perform the operation for important system resources, fatal error processing, and user-mode program execution management is called the supervisor privilege (SV privilege). This privilege is available in supervisor mode. The SV privilege is generally necessary to execute instructions used to perform the operation for important system resources, and these instructions are sometimes called SV privileged instructions.

In SV privilege, operations of operating system, exception handler, interrupt handler and so on, are assumed to be executed.

(3) User authority (UM authority)

UM authority is a privilege that can not perform the operation of the virtualization support function and other important system resources.

It can not execute HV privileged instructions or SV privileged instructions to perform those operations.

In User authority, operation of application software is assumed to be executed.

(4) Coprocessor Use Permissions

Regardless of the CPU operating mode, it is possible to specify whether coprocessors can be used.

The CU2 to CU0 bits in the PSW register are used by a supervisor to specify whether coprocessors can be used by each program. If the CU 2 to CU0 bits are not set to 1, a coprocessor unusable exception occurs when the corresponding coprocessor instruction is executed or the system register is accessed.

If no coprocessor is installed, it is not possible to set the corresponding CU bits to 1. The setting of the CU2 to CU0 bits is valid regardless of the CPU operating mode, and, even if the CPU authority is HV privilege or SV privilege, if the supervisor accesses coprocessor system registers, they must be allowed to be used by the setting of CU2-0 bit.

There are restrictions for changing the CU2-0 bit. For details, see **Section 2.4.1, Coprocessor Use Permissions**

(5) Operation when there is a Privilege Violation

When a privileged instruction is executed by someone who does not have the required privilege, or when a system register for which access permission is specified is accessed by someone who does not have the required permission, a RIE exception, PIE exception or UCPOP exception occurs.

The **Table 2.2** shows whether instructions can be executed depends on operating mode and use permission. The **Table 2.3** shows whether system registers can be accessed depends on operating mode and use permission.

Table 2.2 Operation when Execution Permission is Violated

Instruction		HVCFG	PSWH	PSW				
Execution Permission	Classification	HVE	GM	UM	CU2	CU1	CU0	Whether Instructions can be Executed
HV privilege*1	*2	0	—	—	—	—	—	Impossible (RIE exception occurs)
		1	0	0	—	—	—	Possible
		1	0	1	—	—	—	Impossible (PIE exception occurs)
		1	1	0	—	—	—	Impossible (PIE exception occurs)
		1	1	1	—	—	—	Impossible (PIE exception occurs)
SV privilege	*2	—	0	0	—	—	—	Possible
		—	0	1	—	—	—	Impossible (PIE exception occurs)
		—	1	0	—	—	—	Possible
		—	1	1	—	—	—	Impossible (PIE exception occurs)
User	Coprocessor 0 instruction	—	—	—	—	—	0	Impossible (UCPOP exception occurs)
		—	—	—	—	—	1	Possible
	Coprocessor 1 instruction	—	—	—	—	0	—	Impossible (UCPOP exception occurs)
		—	—	—	—	1	—	Possible
	Coprocessor 2 instruction	—	—	—	0	—	—	Impossible (UCPOP exception occurs)
		—	—	—	1	—	—	Possible
	Other than above	—	—	—	—	—	—	Possible

Note: —: 0 or 1

Note 1. Some operations of the CACHE instruction are HV privilege instructions when the virtualization operating mode is virtualization mode, but are SV privilege instructions when the virtualization operating mode is conventional mode. When these CACHE instructions are executed in the conventional mode, the RIE exception does not occur, and the execution authority is judged as the SV privileged instruction. For details, see **Section 5.2.7, Execution Privilege of the CACHE/PREF Instruction.**

Note 2. Coprocessor instructions with HV privilege or SV privilege are not defined in this CPU.

Table 2.3 Operation When Access Permission to System Registers is Violated

System Register		HVCFG	PSWH	PSW				
Access Permission	Classification	HVE	GM	UM	CU2	CU1	CU0	Whether Instructions can be Executed
HV privilege*1	*2	0	—	0	—	—	—	Accessible*3
		0	—	1	—	—	—	Inaccessible (PIE exception occurs)*3
		1	0	0	—	—	—	Accessible
		1	0	1	—	—	—	Inaccessible (PIE exception occurs)
		1	1	0	—	—	—	Inaccessible (PIE exception occurs)
		1	1	1	—	—	—	Inaccessible (PIE exception occurs)
SV privilege	Coprocessor 0 Permission	—	—	0	—	—	0	Inaccessible (UCPOP exception occurs)
		—	—	0	—	—	1	Accessible
		—	—	1	—	—	0	Inaccessible (UCPOP exception occurs)
		—	—	1	—	—	1	Inaccessible (PIE exception occurs)
	Coprocessor 1 Permission	—	—	0	—	0	—	Inaccessible (UCPOP exception occurs)
		—	—	0	—	1	—	Accessible
		—	—	1	—	0	—	Inaccessible (UCPOP exception occurs)
		—	—	1	—	1	—	Inaccessible (PIE exception occurs)
	Coprocessor 2 Permission	—	—	0	0	—	—	Inaccessible (UCPOP exception occurs)
		—	—	0	1	—	—	Accessible
		—	—	1	0	—	—	Inaccessible (UCPOP exception occurs)
		—	—	1	1	—	—	Inaccessible (PIE exception occurs)
	Other than above	—	—	0	—	—	—	Accessible
		—	—	1	—	—	—	Inaccessible (PIE exception occurs)
UM authority	Coprocessor 0 Permission	—	—	—	—	—	0	Inaccessible (UCPOP exception occurs)
		—	—	—	—	—	1	Accessible
	Coprocessor 1 Permission	—	—	—	—	0	—	Inaccessible (UCPOP exception occurs)
		—	—	—	—	1	—	Accessible
	Coprocessor 2 Permission	—	—	—	0	—	—	Inaccessible (UCPOP exception occurs)
		—	—	—	1	—	—	Accessible
	Other than above	—	—	—	—	—	—	Accessible

Note: —: 0 or 1

Note 1. In some system registers, the access authority is HV privilege when the virtualization operating mode is virtualization mode, but when the virtualization operating mode is conventional mode, the access authority is SV privilege. When these system registers are operated in the conventional mode, their execution authority is determined as system registers with access authority of the SV privilege. For details, see **Section 3, Register Set**.

Note 2. In this CPU, coprocessor system registers whose access authority is HV privilege are not defined.

Note 3. Operations on some system registers are handled as operations on undefined registers that require the SV privilege for accessing. For details, see **Section 3, Register Set**.

2.1.4 Halt State by a HALT Instruction

See the "CPU" section in the hardware manual of the product used.

2.1.5 Temporary Halt State by a SNOOZE instruction

See the "CPU" section in the hardware manual of the product used.

2.2 Instruction Execution

The instruction execution flow of this CPU is shown below.

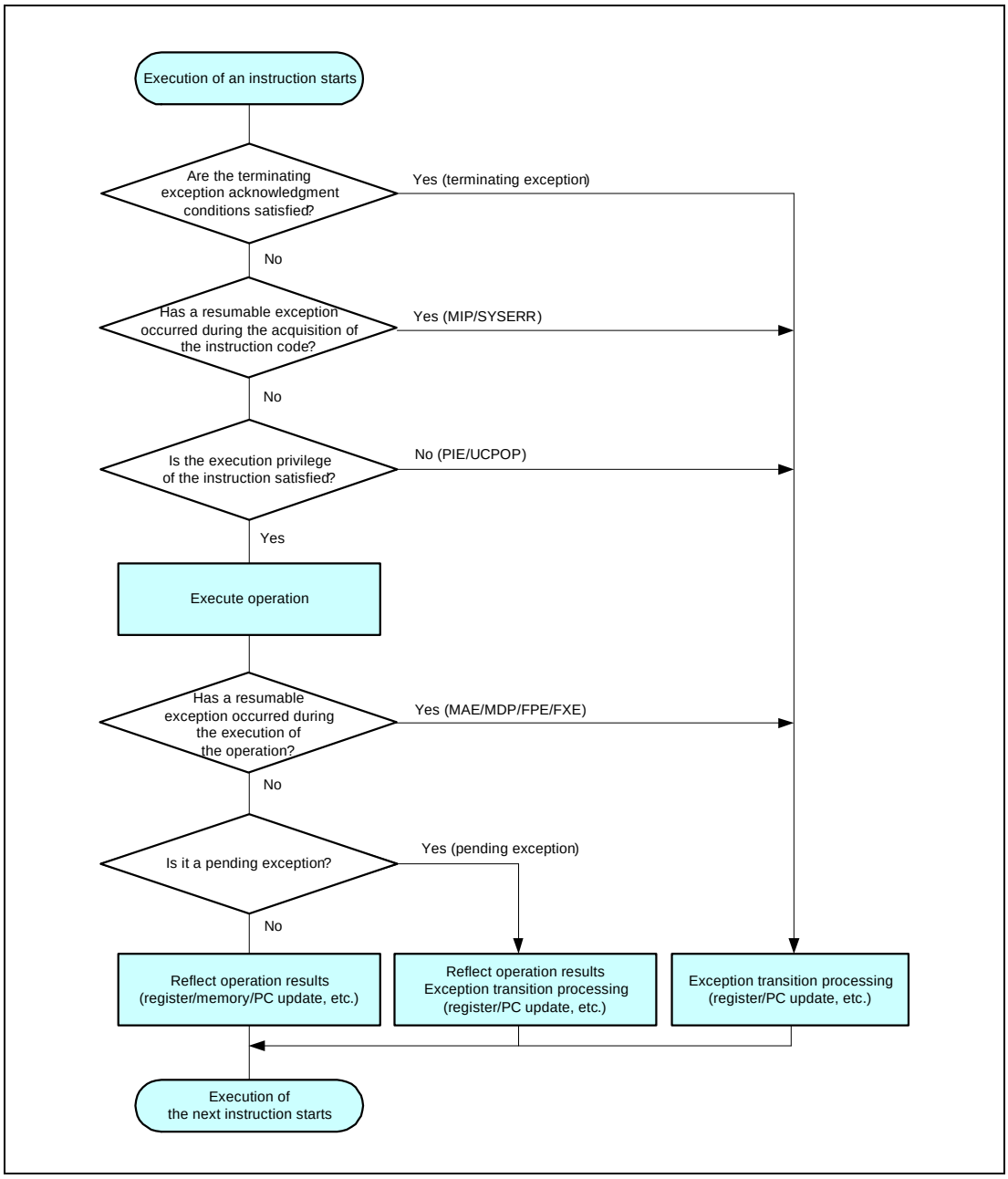


Figure 2.2 Instruction Execution Flow

When terminating exceptions can be acknowledged, if an exception is detected during the acquisition of instruction code, or if the execution privilege of the instruction is not satisfied, an exception occurs before the instruction is executed. If a resumable exception occurs while the CPU is executing the operation, it interrupts the execution of the operation and acknowledges the exception. In these cases, the result of instruction operation is, in principle, not reflected in registers or memory and the CPU retains its state that is established before executing the instruction.*1

For a pending exception such as a software exception, the exception is acknowledged after the result of instruction execution has been reflected.

For details of the types of exceptions, refer to **Section 2.3, Exceptions and Interrupts**.

Note 1. If an exception is acknowledged during the execution of the following instructions, intermediate results may be applied to memory or general-purpose registers. However, SP/EP is not updated.

- PREPARE, DISPOSE, PUSHSP, POPSP, STM.MP, LDM.MP, STM.GSR, LDM.GSR

2.3 Exceptions and Interrupts

Exceptions and interrupts are exceptional events that cause the branch from the executing program to another program. Exceptions and interrupts are triggered by various causes, including interrupts from peripherals and program errors.
For details, see **Section 4, Exceptions and Interrupts**.

2.3.1 Types of Exceptions

The exceptions of this CPU are divided into the following three types according to the purpose of the exceptions.

- Terminating exception
- Resumable exception
- Pending exception

Also, exceptions classified into these types are further classified into the following two according to the restricted operating mode in which the exception handler is processed.

- Exceptions handled in Host mode
- Exceptions handled in Guest mode

(1) Terminating Exception

See the "CPU" section in the hardware manual of the product used.

(2) Resumable Exception

See the "CPU" section in the hardware manual of the product used.

(3) Pending Exception

See the "CPU" section in the hardware manual of the product used.

(4) Exceptions handled in Host mode

The exception handler is handled in Host mode. There are cases for which the exception cause belongs to the host mode and cases for which the processing is specified to be performed in Host mode but the exception causes belong to guest mode. Exceptions that belong to the host mode are the resumable exceptions which are caused by instructions executed in host mode, the pending exceptions, and the terminating exceptions which occur regardless of restricted operating mode. Exceptions that are handled in Host mode although the exception cause belongs to Guest mode are those that occur when an HVTRAP instruction is executed in guest mode or if MIP, MDP, or SYSERR exception occurs when transition to the host mode is specified by the corresponding bit in GMCFG.

(5) Exceptions handled in Guest mode

The exception handler is handled in Guest mode. This exception applies to the case for which the exception cause belongs to Guest mode or to the case in which MIP, MDP, or SYSERR exception occurs when processing in guest mode is specified by the corresponding bit in GMCFG.

2.3.2 Exception Level

See the "CPU" section in the hardware manual of the product used.

2.4 Coprocessors

In this CPU, single-precision and double-precision floating-point unit (FPU) and extended floating point operation unit (FXU) are incorporated. Note that these coprocessors may not be available depending on the specification of the product.

2.4.1 Coprocessor Use Permissions

To execute a coprocessor instruction, permission to use the corresponding coprocessor instruction is necessary. Coprocessor use permissions are specified by the PSW.CU2 to PSW.CU0 bits, and, if an attempt is made to execute an instruction for which the corresponding coprocessor use permission is cleared to 0, a coprocessor unusable exception (UCPOP) occurs.

In the following cases, the values of the PSW.CU2 to CU0 bits are fixed at 0 and cannot be changed.

- Coprocessor functions are not incorporated in the product
- Coprocessor functions are made unavailable according to the functions of the product
- In guest mode, GMCFG.GCU2-0 is cleared (0), making it impossible to change GMPSW.CU2-0.

2.4.2 Correspondences between Coprocessor Use Permissions and Coprocessors

See the "CPU" section in the hardware manual of the product used.

2.4.3 Coprocessor Unusable Exceptions

See the "CPU" section in the hardware manual of the product used.

2.4.4 System Registers

See the "CPU" section in the hardware manual of the product used.

2.5 Registers

In this CPU, the program registers (general-purpose registers and the program counter PC) and system registers for controlling the status and storing exception information are defined.

2.5.1 Program Registers

The program registers include general-purpose registers (r0 to r31) and the program counter (PC).

In this CPU, the program registers are shared between Host mode and Guest mode, or between Guest partition. Therefore, when changing the restricted operating mode or changing Guest partition, proper replacement of program registers by software is required. For details, see **Section 4.1.7, Context switching**.

Table 2.4 Program Registers

Category	Access Permission	Name
Program counter	UM	PC
General-purpose registers	UM	r0 to r31

Note: Access to the registers with UM (user mode) access permission is always allowed.

For details about program registers, see **Section 3.1, Program Registers**.

2.5.2 System Registers

System registers are placed in dedicated address spaces defined based on two types of address information: selection ID and register number. Up to 32 selection ID can be defined, and one selection ID includes up to 32 system registers. Therefore, up to 1024 system registers can be defined in the address spaces for system registers. Basically this CPU allocates selection ID as shown below:

Selection ID 0 to 3, 10:	Registers related to basic functions
Selection ID 4 and 5:	Registers related to the memory management function
Selection ID 9:	Guest context registers
Selection ID 11, 14, 15	Registers related to counter function
Selection ID 12, 13:	Registers related to this CPU specific hardware functions
Other ID:	Reserved for future expansion of CPUs compatible with this CPU

For details about system registers, see the relevant sections in **Section 3, Register Set**.

In this CPU, there are system registers that are shared between Host mode and Guest mode, and system registers that are multiplexed. For details, see **Section 4.1.7, Context switching**. For multiplexed system registers, corresponding registers are automatically switched when the restricted operating mode is changed. On the other hand, when changing Guest partition, it is necessary to properly replace the registers by the software. For details, see **Section 2.5.7, System Register Multiplexing**.

2.5.3 Register Updating

See the "CPU" section in the hardware manual of the product used.

(1) LDSR and STSR

See the "CPU" section in the hardware manual of the product used.

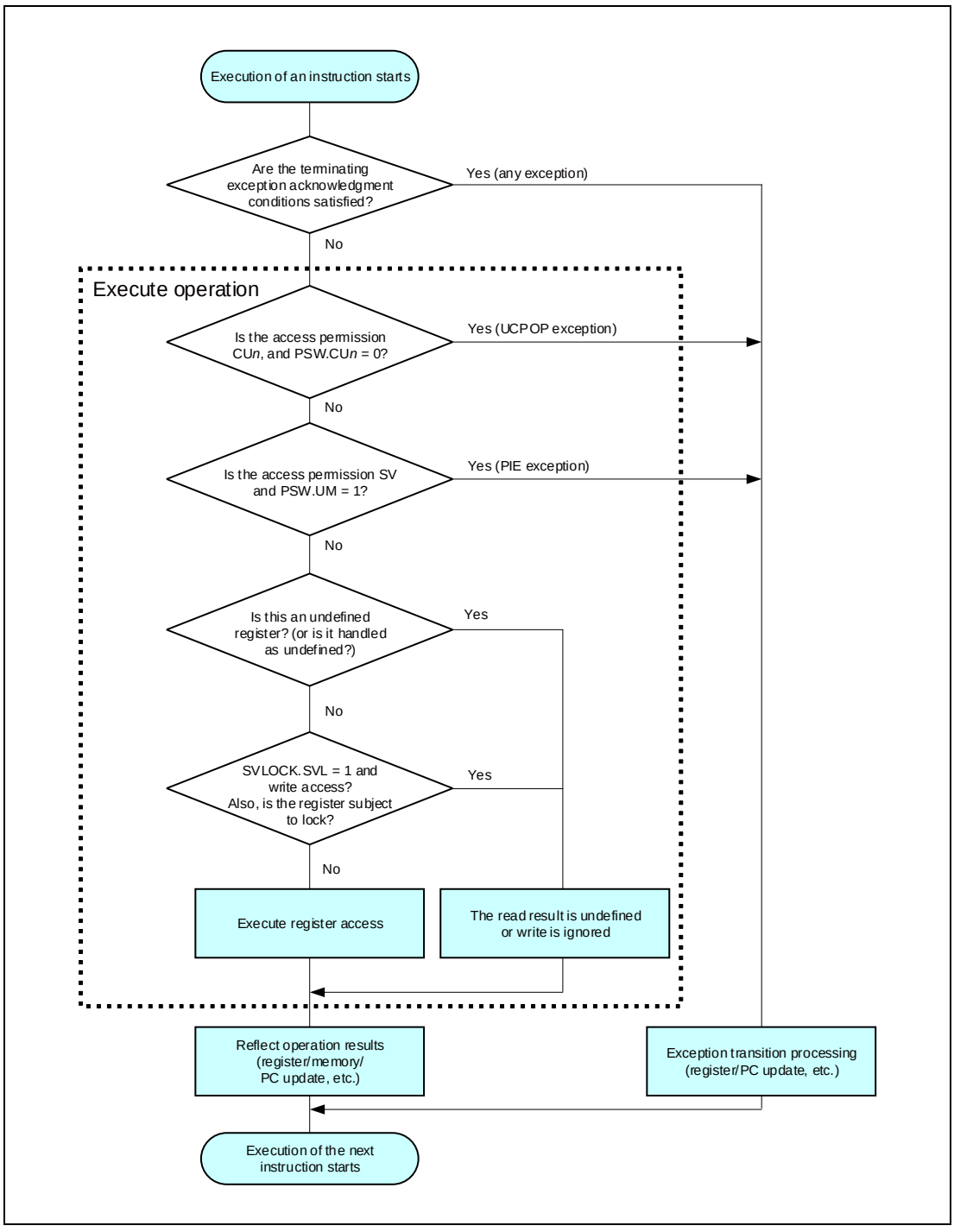


Figure 2.3 Flow of Executing the LDSR and STSR Instructions

(2) Changing the access authority of system register

Table 2.5 shows a list of system registers whose access authority is changed according to the state of the virtualization operating mode.

The setting of these system registers affects CPU operation, but they are not multiplexed. Also, when the virtualization operating mode is virtualization mode, SV privilege can be acquired on the guest partition side. Therefore, if these system registers are modified by the management software running on Guest partition, the virtualization software operating in host mode and the operation of other Guest partitions are affected. To prevent this, the access authority of these system registers is raised to HV privilege when the virtualization operating mode is virtualization mode. For multiplexing of the system registers, see **Section 2.5.7, System Register Multiplexing**.

Table 2.5 System registers whose access authority is changed according to the state of the virtualization operating mode (1/2)

Register number (regID, selID)	Name	Access authority				Description
		Conventional mode		Virtualization mode		
		Writing	Reading	Writing	Reading	
SR21,0	SNZCFG	SV	SV	HV	HV	
SR17,5	MPBK	SV	SV	HV	HV	
SR24,5	MPID0	SV	SV	HV*1	SV*1	
SR25,5	MPID1	SV	SV	HV*1	SV*1	
SR26,5	MPID2	SV	SV	HV*1	SV*1	
SR27,5	MPID3	SV	SV	HV*1	SV*1	
SR28,5	MPID4	SV	SV	HV*1	SV*1	
SR29,5	MPID5	SV	SV	HV*1	SV*1	
SR30,5	MPID6	SV	SV	HV*1	SV*1	
SR31,5	MPID7	SV	SV	HV*1	SV*1	
SR0,11	TSCOUNTL	SV*2	UM*2	HV*1	UM*2	
SR1,11	TSCOUNTH	SV*2	UM*2	HV*1	UM*2	
SR2,11	TSCTRL	SV	SV	HV	HV	
*3	*3	SV	SV	HV	HV	
SR9, 11	PMGMCTRL	SV	SV	HV	HV	
SR0,14	PMCTRL0	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR1,14	PMCTRL1	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR2,14	PMCTRL2	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR3,14	PMCTRL3	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR4,14	PMCTRL4	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR5,14	PMCTRL5	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR6,14	PMCTRL6	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR7,14	PMCTRL7	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR16,14	PMCOUNT0	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR17,14	PMCOUNT1	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR18,14	PMCOUNT2	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR19,14	PMCOUNT3	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR20,14	PMCOUNT4	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR21,14	PMCOUNT5	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR22,14	PMCOUNT6	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR23,14	PMCOUNT7	SV*4	SV*4	HV *4,*5	HV *4,*5	

Table 2.5 System registers whose access authority is changed according to the state of the virtualization operating mode (2/2)

Register number (regID, selID)	Name	Access authority				Description
		Conventional mode		Virtualization mode		
		Writing	Reading	Writing	Reading	
SR0,15	PMSUBCND0	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR1,15	PMSUBCND1	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR2,15	PMSUBCND2	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR3,15	PMSUBCND3	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR4,15	PMSUBCND4	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR5,15	PMSUBCND5	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR6,15	PMSUBCND6	SV*4	SV*4	HV *4,*5	HV *4,*5	
SR7,15	PMSUBCND7	SV*4	SV*4	HV *4,*5	HV *4,*5	

Note 1. Reading of these system registers is possible with SV privilege.

Note 2. Reading of these system registers is independent of the state of the virtualization operating mode and is possible with UM authority. Only writing authority changes.

Note 3. All system registers whose selID is 12 or 13, except LSCFG (regID = 2, selID = 12), L1RCFG (regID = 12, selID = 13) are applicable.

Note 4. Access authority can be changed to UM by PMUMCTRL setting.

Note 5. Access authority can be changed to SV by PMGMCTRL setting.

2.5.4 Accessing Undefined Registers

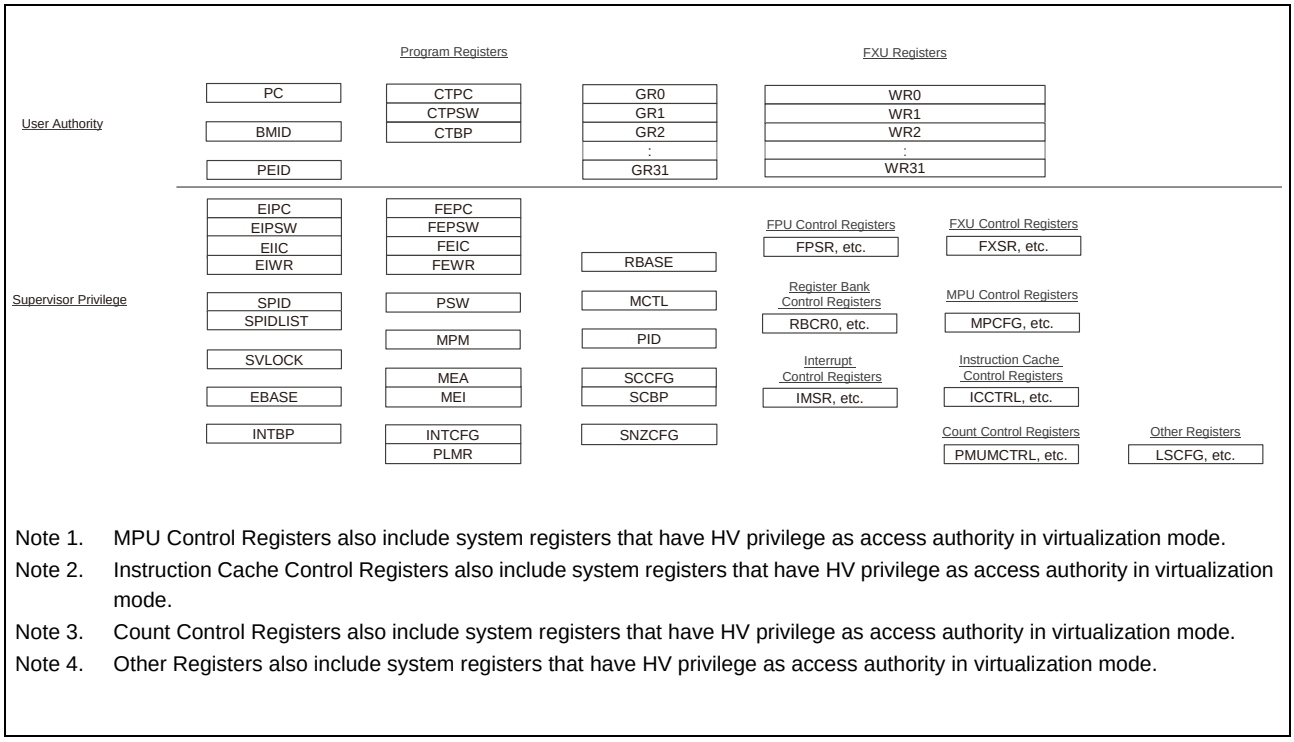
See the "CPU" section in the hardware manual of the product used.

2.5.5 Supervisor Lock Setting

See the "CPU" section in the hardware manual of the product used.

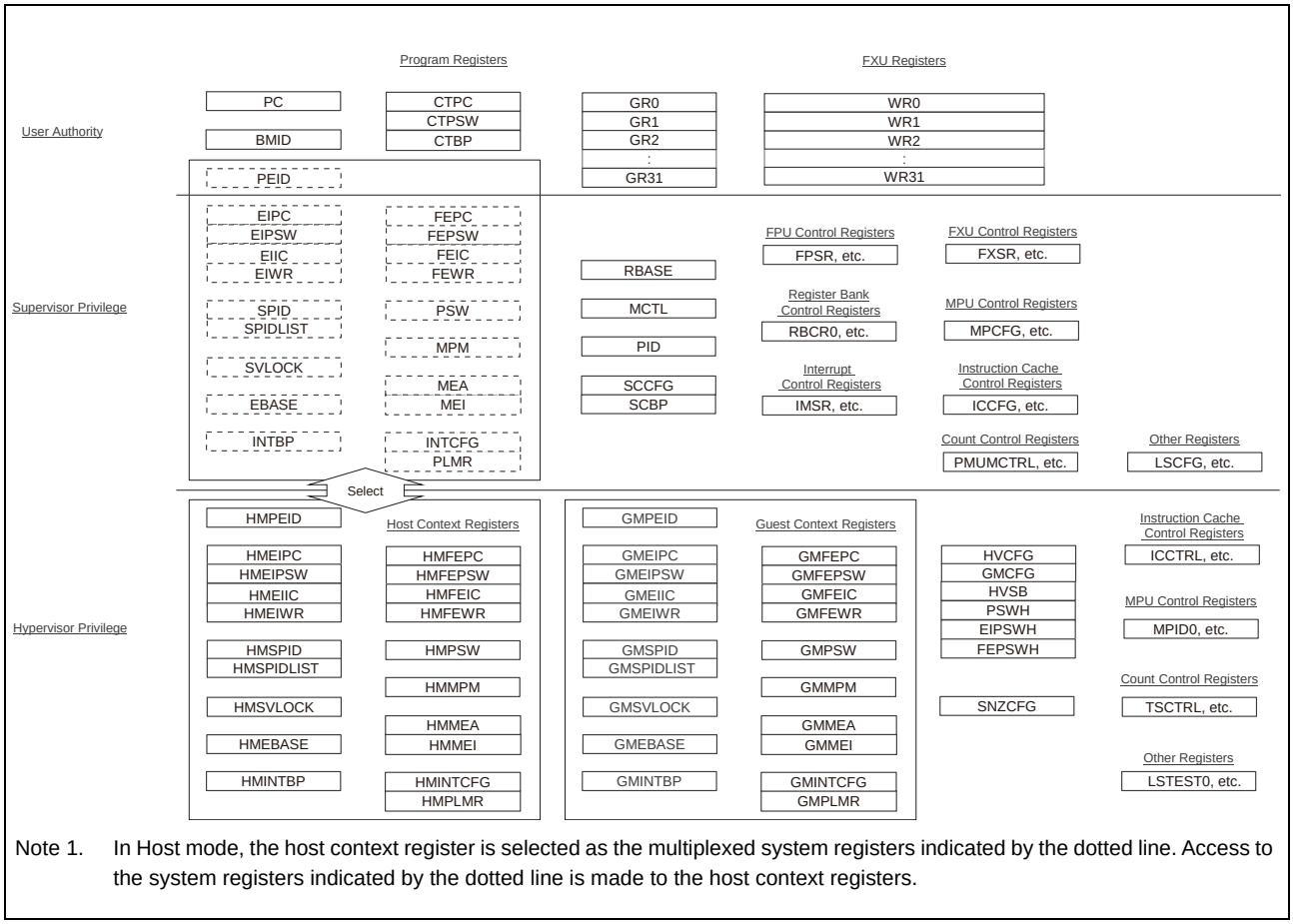
2.5.6 Change in Register Model

When the state of the virtualization operating mode and the virtualization operating mode are the virtualization mode, the register model to be referenced changes according to the state of the restricted operating mode. For multiplexing of the system registers, see **Section 2.5.7, System Register Multiplexing**.



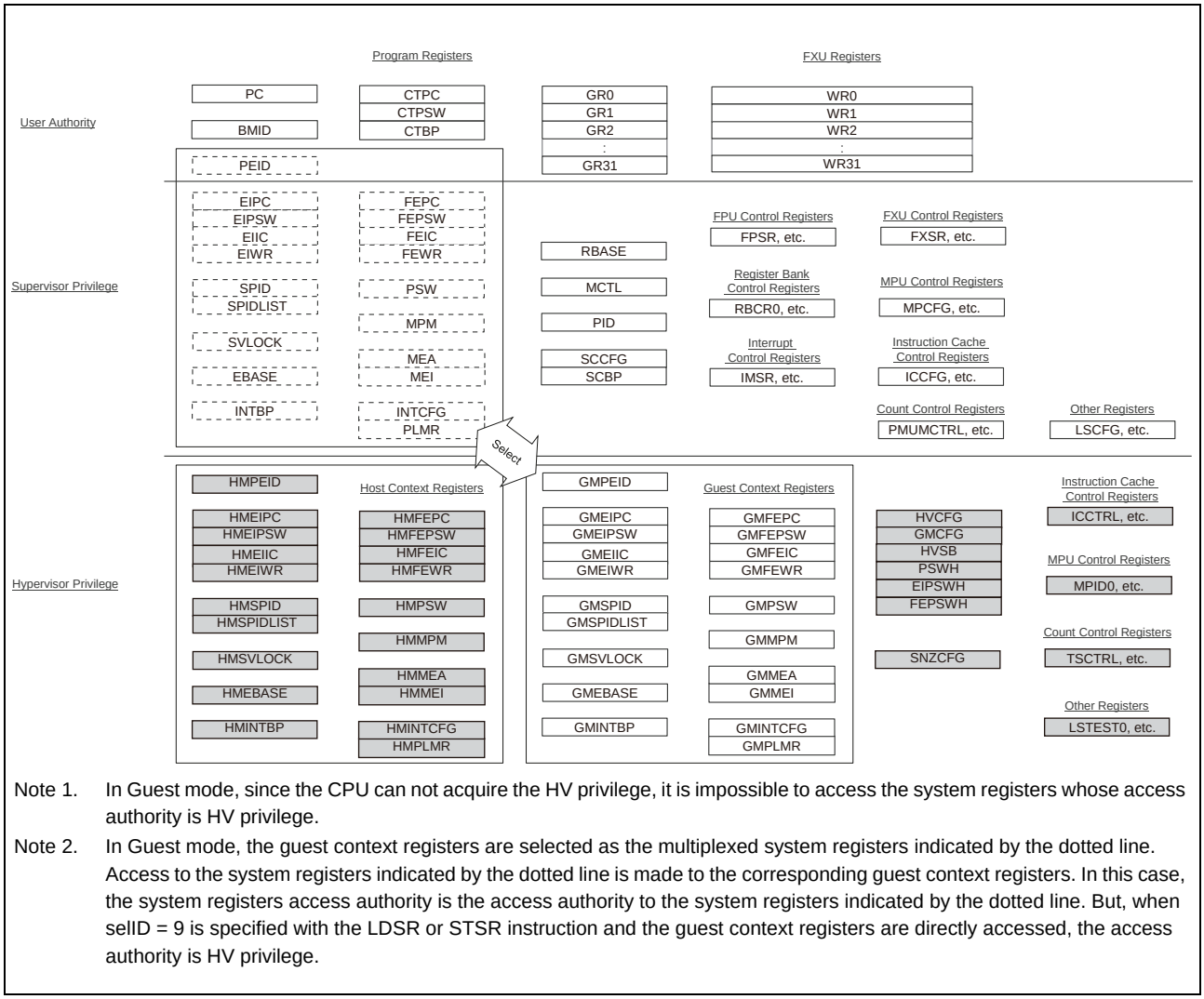
Note 1. MPU Control Registers also include system registers that have HV privilege as access authority in virtualization mode.
Note 2. Instruction Cache Control Registers also include system registers that have HV privilege as access authority in virtualization mode.
Note 3. Count Control Registers also include system registers that have HV privilege as access authority in virtualization mode.
Note 4. Other Registers also include system registers that have HV privilege as access authority in virtualization mode.

Figure 2.4 Register model in conventional mode



Note 1. In Host mode, the host context register is selected as the multiplexed system registers indicated by the dotted line. Access to the system registers indicated by the dotted line is made to the host context registers.

Figure 2.5 Register model in Host mode



Note 1. In Guest mode, since the CPU can not acquire the HV privilege, it is impossible to access the system registers whose access authority is HV privilege.

Note 2. In Guest mode, the guest context registers are selected as the multiplexed system registers indicated by the dotted line. Access to the system registers indicated by the dotted line is made to the corresponding guest context registers. In this case, the system registers access authority is the access authority to the system registers indicated by the dotted line. But, when selID = 9 is specified with the LDSR or STSR instruction and the guest context registers are directly accessed, the access authority is HV privilege.

Figure 2.6 Register model in Guest mode

2.5.7 System Register Multiplexing

In Guest mode, in the case an interrupt (EIINTn) to be processed in Host mode is accepted during operation, or in the case a memory protection exception occurred while operating in Guest mode but the exception handler was set to be processed in Host mode by GMCFG, mainly an exception with a change from Guest mode to Host mode may occur irrespective of the intention of the running software. In order to efficiently perform these exception acceptance and exception handling, this CPU incorporates system registers related to exception acceptance and exception handling separately for Host mode operation and Guest mode operation. This is called System Register Multiplexing. In System Register Multiplexing, when referring to or updating the original system registers, among the system registers multiplexed and mounted, the register corresponding to the state of restricted operating mode is automatically selected by hardware.

Among the system registers multiplexed and mounted, the set of system registers used during Host mode operation is called the host context register. The set of system registers used during Guest mode operation is called Guest context register.

The host context registers and guest context registers are used exclusively because they are automatically selected according to the state of the restricted operating mode.

This selection is not only explicitly done for system register operation by the LDSR instruction and STSR instruction but is also done for implicit use of system registers for instruction execution, information storage at the time of exception occurrence and so on; it is done whenever the system registers are used.

Also, when the restricted operating mode is Host mode, the value of the guest context registers do not affect the operation of the CPU. When the restricted operating mode is Guest mode, the value of the host context registers do not affect the operation of the CPU.

Note that, when the restricted operating mode is Host mode, each system register belonging to the guest context registers, uses the registers number (regID, selID) different from the original system registers to enable operation by LDSR instruction and STSR instruction. On the other hand, when the restricted operating mode is Guest mode, operation of each system register belonging to the host context register is impossible.

When virtualization software switches guest partitions, the guest context registers must be replaced. At that time, the virtualization software is in Host mode when executing LDSR or STSR instructions and switching the guest context registers by using the register number for operating the guest context registers is possible. For details on replacing the guest context register, see **Section 4.1.7, Context switching**.

Table 2.6 shows a list of multiplexed system registers.

Table 2.6 List of multiplexed system registers

Register number (regID, selID)*1	Name of original system register	Name of host context register*3	Name of guest context register	Register number (regID, selID)*2
SR 0, 0	EIPC	HMEIPC	GMEIPC	SR 0, 9
SR 1, 0	EIPSW	HMEIPSW	GMEIPSW	SR 1, 9
SR 2, 0	FEPC	HMFEPC	GMFEPC	SR 2, 9
SR 3, 0	FEPSW	HMFEPSW	GMFEPSW	SR 3, 9
SR 5, 0	PSW	HMPSW	GMPSW	SR 5, 9
SR13, 0	EIIC	HMEIIC	GMEIIC	SR13, 9
SR14, 0	FEIC	HMFEIC	GMFEIC	SR14, 9
SR28, 0	EIWR	HMEIWR	GMEIWR	SR28, 9
SR29, 0	FEWR	HMFEWR	GMFEWR	SR29, 9
SR 0, 1	SPID	HMSPID	GMSPID	SR16, 9
SR 1, 1	SPIDLIST	HMSPIDLIST	GMSPIDLIST	SR17, 9
SR 3, 1	EBASE	HMEBASE	GMEBASE	SR19, 9
SR 4, 1	INTBP	HMINTBP	GMINTBP	SR20, 9
SR 8, 1	SVLOCK	HMSVLOCK	GMSVLOCK	SR24, 9
SR 0, 2	PEID	HMPEID	GMPEID	SR30, 9
SR 6, 2	MEA	HMMEA	GMMEA	SR6, 9
SR 8, 2	MEI	HMMEI	GMMEI	SR8, 9
SR13, 2	INTCFG	HMINTCFG	GMINTCFG	SR21, 9
SR14, 2	PLMR	HMPLMR	GMPLMR	SR22, 9
SR 0, 5	MPM	HMMPM	GMMPM	SR25, 9

Note 1. It is the register number for operating the original system registers.

Note 2. When the restricted operating mode is Host mode, it is the register number for operating each system register belonging to the guest context registers.

Note 3. The original system register and the corresponding host context register use the same resources.

2.6 Data Types

2.6.1 Data Formats

See the "CPU" section in the hardware manual of the product used.

2.6.2 Data Representation

See the "CPU" section in the hardware manual of the product used.

2.6.3 Data Alignment

In this CPU, misaligned data allocation is inhibited. When the result of address calculation is a misaligned address, a misalignment exception (MAE) occurs.

Misaligned access indicates the accesses to the data size with the addresses listed below:

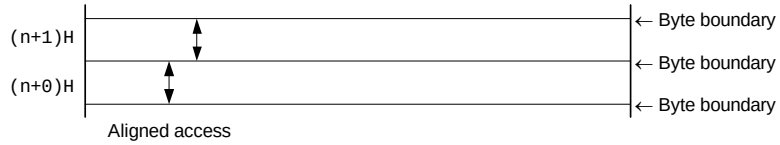
- Halfword size: The access to an address that is not at the halfword boundary (where LSB of the address = 0)
- Word size: The access to an address that is not at the word boundary (where the lowest two bits of the address = 0).
- Double-word size: The access to an address that is not at the double-word boundary (where the lowest three bits of the address = 0).
- Quad-word size: The access to an address that is not at the quad-word boundary (where the lowest four bits of the address = 0)

For the double-word format only, a misaligned access exception does not occur when data is placed at the word boundary but not at the double-word boundary, and data can be normally accessed in double word.

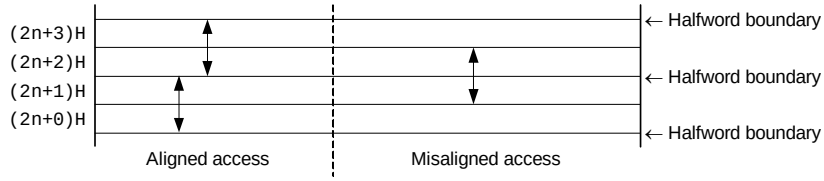
CAUTIONS

1. **The following instructions might possibly cause misaligned access. For details, see the relevant descriptions in the *RH850G4MH User's Manual: Software*.**
   - **LD.H, LD.HU, LD.W, LD.DW**
   - **SLD.H, SLD.HU, SLD.W**
   - **ST.H, ST.W, ST.DW**
   - **SST.H, SST.W**
   - **LDL.HU, LDL.W, STC.H, STC.W, CAXI**
   - **LDV.W, LDV.DW, LDV.QW, STV.W, STV.DW, STV.QW**
   - **LDVZ.H4, STVZ.H4**
2. **The following instructions do not cause misaligned access, because the address is rounded according to the instruction specification when a misaligned address is specified.**
   - **PREPARE, DISPOSE**
   - **PUSHSP, POPSP**
   - **STM.MP, LDM.MP**
   - **STM.GSR, LDM.GSR**

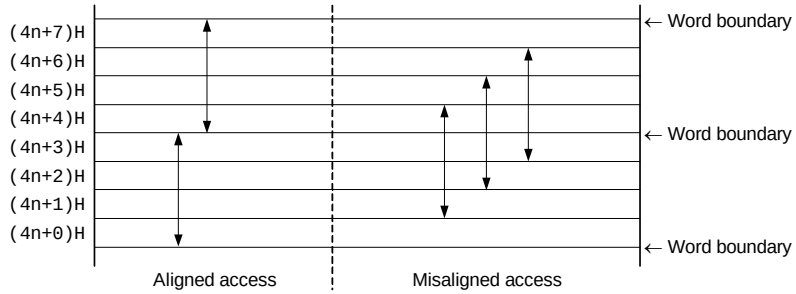
(a) Byte Access



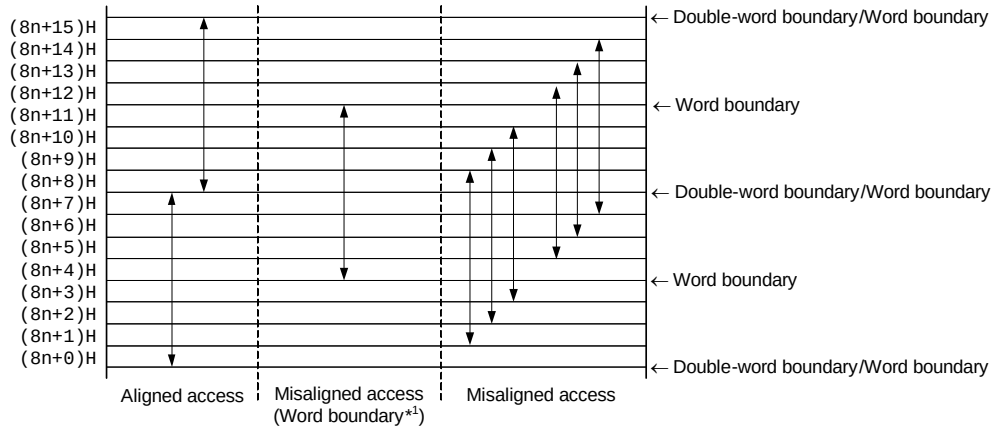
(b) Halfword Access



(c) Word Access



(d) Double-word Access



Note 1. No misalignment exception (MAE) occurs for LD.DW and ST.DW instructions. For details, see LD.DW and ST.DW in the *RH850G4MH User's Manual: Software*. An MAE occurs for other double-word access instructions.

Figure 2.7 Example of Data Placement for Misaligned Access (1/2)

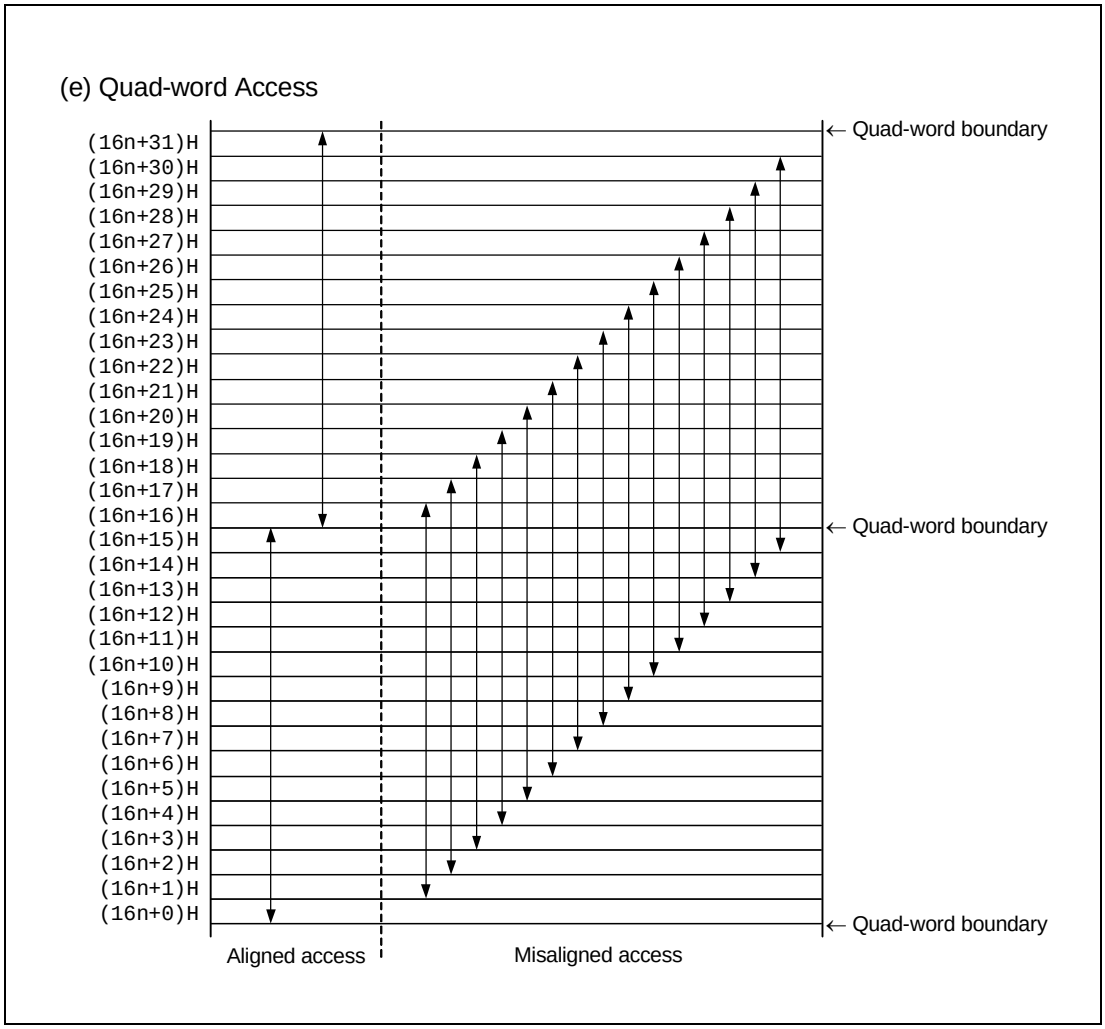


Figure 2.7 Example of Data Placement for Misaligned Access (2/2)

2.7 Address Space

See the "CPU" section in the hardware manual of the product used.

2.7.1 Memory Map

See the "CPU" section in the hardware manual of the product used.

2.7.2 Instruction Addressing

See the "CPU" section in the hardware manual of the product used.

2.7.3 Data Addressing

The following methods can be used to access the target registers or memory when executing an instruction.

If the result of address calculation exceeds the positive maximum value FFFF $FFFF_H$ by addition, it is wrapped around to 0000 0000_H. If the result of address calculation falls below the positive minimum value 0000 0000_H by subtraction, it is wrapped around to FFFF $FFFF_H$.

(1) Relative Addressing (PC Relative)

See the "CPU" section in the hardware manual of the product used.

(2) Register Addressing (Register Indirect)

See the "CPU" section in the hardware manual of the product used.

(3) Based Addressing

See the "CPU" section in the hardware manual of the product used.

(4) Bit Addressing

See the "CPU" section in the hardware manual of the product used.

(5) Post Index Increment/Decrement Addressing

See the "CPU" section in the hardware manual of the product used.

(6) Other Addressing

The target memory is accessed using a value specified by an instruction as the operand address. How a value is specified is explained in [Operation] or [Description] of each instruction.

The SWITCH, CALLT, SYSCALL, PREPARE, DISPOSE, PUSHSP, POPSP, STM.MP, LDM.MP, STM.GSR, and LDM.GSR are used with this type of addressing.

2.8 Execution Timing of a Store Instruction

See the "CPU" section in the hardware manual of the product used.

2.9 Memory Ordering

This CPU guarantees that memories are accessed in the programmed order. However, in the system that incorporates multiple bus masters such as the bus system with DMA or multi-core, the order of accesses to memories needs to be considered. For these cases, see **Section 7.1, Synchronization Processing**.

Also in the state change of the restricted operating mode, when accepting of the exception or executing of the return instruction, synchronization processing similar to the SYNCM instruction is performed. Therefore, the memory accesses performed in Host mode and memory accesses performed in Guest mode are not mixed in the bus system. For details, see **Section 2.1.2, CPU Operating Mode Transition** and **Section 7.1, Synchronization Processing**.

2.10 Acquiring the CPU Number

This CPU provides a method for identifying CPUs in a multi-processor system.

In the multi-processor configuration, you can identify which CPU core is running a program by referencing the PEID register. A unique number within a multi-processor system is assigned to the PEID register according to the specification of the product.

2.11 System Protection Identifier (SPID)

In this CPU, memory resources and peripheral devices are managed by system protection groups. By specifying the group to which the program being executed belongs, you can assign accessible memory resources and peripheral devices to the program.

The program being executed belongs to the group specified by the SPID, and whether the memory resources and peripheral devices are accessible is decided using the SPID. A value can be set to the SPID register by the supervisor.

CAUTION

According to the value of the SPID, how operations are assigned to memory resources and peripheral devices is determined by the specifications of the product.

2.12 Timestamp Counter

This CPU has a 64-bit timestamp counter. It can measure a long period of time, and so can be used to obtain specific information for time identification.

Since the timestamp counter is allocated to system registers, it can be accessed quickly by using a LDSR/STSR instruction.

If an overflow occurs during counting operation, no exception will occur.

2.12.1 How to Operate the Timestamp Counter

The value of the timestamp counter is initialized to 0 by a reset. Therefore, if you want to retain the value of the counter across a reset, then before the reset, save the value of the counter in a memory that is not initialized by the reset, restore the value to the counter after the reset, and restart counting.

The counter is 64-bit width, so the counter consists of two 32-bit system registers, TSCOUNTL and TSCOUNTH. Both registers need to be accessed by using LDSR/STSR instructions.

When the counter is not running (the value of the TSCTRL.CEN bit is 0), no special care is needed to access the two registers.

However, if the counter is running (the value of the TSCTRL.CEN bit is 1), it is not recommended to update the counter by using LDSR instructions. In this case, the timing of update of the counter is not guaranteed. It is recommended to update the counter when it is not running.

Also, when reading the value of the counter by using STSR instructions while it is running, it is recommended to follow the procedure below.

```
TSCNTRD:
        STSR    1, r21, 11      – Read the upper side of the counter
        STSR    0, r20, 11      – Read the lower side of the counter
        STSR    1, r22, 11      – Read again the upper side of the counter
        CMP      r21, r22       – Compare the two values read from the upper side of the counter
        BNE      TSCNTRD        – If they are not identical, a carry has occurred. Read again.
```

Even if the CPU core is not in operation after the execution of a HALT or SNOOZE instruction, the timestamp counter continues counting.

2.13 Performance Measurement Function

This CPU has the performance measurement function. The performance measurement function can measure the performance of programs executed, the effects of interrupts generated during operation, etc. by counting the occurrence of the event specified by the PMCTRLn.CND bit.

The system registers used by the performance measurement function can be accessed only in supervisor mode after a reset. However, it can be accessed in user mode by changing the setting of the PMUMCTRL register.

The performance measurement function itself works even in user mode regardless of the setting of the PMUMCTRL register. Even if all performance measurement channels are made inaccessible in user mode by using the PMUMCTRL register, configuration in supervisor mode allows performance measurement during operation in user mode.

This CPU has eight channels of system register set for the performance measurement function.

2.14 Debug Target Limitation

Debugger software, which provides a CPU debug function, is assumed to be developed separately from Host mode management software (hypervisor). Therefore, the DBGEN register is provided as a function used by a hypervisor to notify debugger software of debug target operating mode limitations when the virtualization support function is enabled.

While DBGEN register is a system register related to the debugging function, the access authority of the register is HV privilege and the register can be updated by a hypervisor. The DBGEN register is assumed to be set to appropriate values by a hypervisor which knows full details of all virtual machines to operate before a transition to Guest mode and the start of operation of the virtual machines.

To update the DBGEN values, be sure to use the following instruction flow.

```
SYNCM              // Waits for the completion of all previous instructions.
LDSR    r20, 0, 3  // Changes the DBGEN setting.
SYNCI              // Reflects the DBGEN setting to the instruction fetch side.
```

The setting change may not be reflected for the SYNCI instruction in the above instruction flow. Do not set a breakpoint in the SYNCI instruction. The DBGEN setting change is surely reflected in the instructions following the SYNCI instruction.

Section 3 Register Set

This chapter describes the program register and system register mounted on this CPU.

In this section, only changes related to addition of virtualization support function are described. For the registers whose specifications have not been changed, see the "CPU" section in the hardware manual of the product used.

Some of the system registers are separately mounted in Host mode operation and Guest mode operation. This is called System Registers Multiplexing. Among the system registers multiplexed, the set of system registers used during Host mode operation is called the host context register.

In System Register Multiplexing, when referring to or updating the conventional system registers, among the system registers multiplexed, the registers corresponding to the state of the restricted operating mode is automatically selected by the hardware. For details on system registers multiplexing, see **Section 2.5.7, System Register Multiplexing**.

In addition, access authority of some registers changes according to the state of virtualization operating mode.

For details, see **Section 2.5.3 (2) Table 2.5, System registers whose access authority is changed according to the state of the virtualization operating mode**.

3.1 Program Registers

See the "CPU" section in the hardware manual of the product used.

3.2 Basic System Registers

In this section, the basic system registers whose specifications have been changed are described. For registers whose specifications have not been changed, see the "CPU" section in the hardware manual of the product used.

Table 3.1 Basic System Registers

Register Number (regID, selID)	Symbol	Function	Access Authority	
			HVE = 0	HVE = 1
SR0, 0	EIPC	Status save registers when acknowledging EI level exception	SV	SV
SR1, 0	EIPSW	Status save registers when acknowledging EI level exception	SV	SV
SR2, 0	FEPC	Status save registers when acknowledging FE level exception	SV	SV
SR3, 0	FEPSW	Status save registers when acknowledging FE level exception	SV	SV
SR5, 0	PSW	Program status word	UM*1	UM*1
SR13, 0	EIIC	EI level exception cause	SV	SV
SR14, 0	FEIC	FE level exception cause	SV	SV
SR16, 0	CTPC	CALLT execution status save register	UM	UM
SR17, 0	CTPSW	CALLT execution status save register	UM	UM
SR20, 0	CTBP	CALLT base pointer	UM	UM
SR21, 0	SNZCFG	SNOOZE control register	SV	HV
SR28, 0	EIWR	EI level exception working register	SV	SV
SR29, 0	FEWR	FE level exception working register	SV	SV
SR0, 1	SPID	System protection identifier	SV	SV
SR1, 1	SPIDLIST	List of system protection identifiers that can be specified in SPID	SV	SV
SR2, 1	RBASE	Reset vector base address	SV	SV
SR3, 1	EBASE	Exception handler vector address	SV	SV
SR4, 1	INTBP	Base address of the interrupt handler “address” table	SV	SV
SR5, 1	MCTL	CPU control	SV	SV
SR6, 1	PID	Processor ID	SV	SV
SR8, 1	SVLOCK	Supervisor lock	SV	SV
SR11, 1	SCCFG	SYSCALL operation setting	SV	SV
SR12, 1	SCBP	SYSCALL base pointer	SV	SV
SR0, 2	PEID	Processor element identifier	UM	UM
SR1, 2	BMID	Bus master identifier	UM	UM
SR6, 2	MEA	Memory error address	SV	SV
SR8, 2	MEI	Memory error information	SV	SV
SR15, 2	RBCR0	Register bank control 0	SV	SV
SR16, 2	RBCR1	Register bank control 1	SV	SV
SR17, 2	RBNR	Register bank number	SV	SV
SR18, 2	RBIP	Register bank initial pointer	SV	SV

Note 1. The access permission differs depending on the bit. For details, see the "CPU" section in the hardware manual of the product used.

(1) EIPC — Status Save Register when Acknowledging EI Level Exception

When an EI level exception is acknowledged, the address of the instruction that was being executed when the EI level exception occurred, or of the next instruction.

When restricted operating mode is Host mode, HMEIPC is selected.

When restricted operating mode is Guest mode, GMEIPC is selected.

(2) EIPSW — Status Save Register when Acknowledging EI Level Exception

When an EI level exception is acknowledged, the current PSW setting is saved to the HMEIPSW register.

When restricted operating mode is Host mode, HMEIPSW is selected.

When restricted operating mode is Guest mode, GMEIPSW is selected.

(3) FEPC — Status Save Register when Acknowledging FE Level Exception

When an FE level exception is acknowledged, the address of the instruction that was being executed when the FE level exception occurred, or of the next instruction.

When restricted operating mode is Host mode, HMFEPC is selected.

When restricted operating mode is Guest mode, GMFEPC is selected.

(4) FEPSW — Status Save Register when Acknowledging FE Level Exception

When an FE level exception is acknowledged, the current PSW setting is saved to the FEPSW register.

When restricted operating mode is Host mode, HMFEPSW is selected.

When restricted operating mode is Guest mode, GMFEPSW is selected.

(5) PSW — Program Status Word

PSW (program status word) is a set of flags that indicate the program status (instruction execution result) and bits that indicate the operation status of the CPU (flags are bits in the PSW that are referenced by a condition instruction (Bcond, CMOV, etc.)).

When restricted operating mode is Host mode, HMPSW is selected.

When restricted operating mode is Guest mode, GMPSW is selected.

(6) EIIC — EI Level Exception Cause

The EIIC register retains the cause of any EI level exception that occurs.

When restricted operating mode is Host mode, HMEIIC is selected.

When restricted operating mode is Guest mode, GMEIIC is selected.

(7) FEIC — FE Level Exception Cause

The FEIC register retains the cause of any FE level exception that occurs.

When restricted operating mode is Host mode, HMFEIC is selected.

When restricted operating mode is Guest mode, GMFEIC is selected.

(8) CTPC — Status Save Register when Executing CALLT

When a CALLT instruction is executed, the address of the next instruction after the CALLT instruction is saved to CTPC.

For the CTPC, see the "CPU" section in the hardware manual of the product used.

(9) CTPSW — Status Save Register when Executing CALLT

When a CALLT instruction is executed, some of the PSW (program status word) settings are saved to CTPSW.

For the CTPSW, see the "CPU" section in the hardware manual of the product used.

(10) CTBP — CALLT Base Pointer

The CTBP register is used to specify table addresses of the CALLT instruction and generate target addresses.

For the CTBP, see the "CPU" section in the hardware manual of the product used.

(11) SNZCFG — SNOOZE Configuration

The SNZCFG register is used to configure the operation of the SNOOZE instruction.

For the SNZCFG, see the "CPU" section in the hardware manual of the product used.

(12) EIWR — EI Level Exception Working Register

The EIWR register is used as a working register when an EI level exception has occurred.

When restricted operating mode is Host mode, HMEIWR is selected.

When restricted operating mode is Guest mode, GMEIWR is selected.

(13) FEWR — FE Level Exception Working Register

The FEWR register is used as a working register when an FE level exception has occurred.

When restricted operating mode is Host mode, HMFEWR is selected.

When restricted operating mode is Guest mode, GMFEWR is selected.

(14) SPID — System Protection Identifier

The SPID register holds the system protection identifier of the CPU.

When restricted operating mode is Host mode, HMSPID is selected.

When restricted operating mode is Guest mode, GMSPID is selected.

(15) SPIDLIST — Legitimate System Protection Identifier List

The SPIDLIST register contains a list of system protection identifiers that can be set to the SPID register.

When restricted operating mode is Host mode, HMSPIDLIST is selected.

When restricted operating mode is Guest mode, GMSPIDLIST is selected.

(16) RBASE — Reset Vector Base Address

This register indicates the reset vector address when there is a reset.

For the RBASE, see the "CPU" section in the hardware manual of the product used.

(17) EBASE — Exception Handler Vector Address

This register indicates the exception handler vector address.

When restricted operating mode is Host mode, HMEBASE is selected.

When restricted operating mode is Guest mode, GMEBASE is selected.

(18) INTBP — Base Address of the Interrupt Handler Address Table

This register indicates the base address of the table when the table reference method is selected as the interrupt handler address selection method.

When restricted operating mode is Host mode, HMINTBP is selected.

When restricted operating mode is Guest mode, GMINTBP is selected.

(19) MCTL — Machine Control

The MCTL register is used to control the CPU.

For the MCTL, see the "CPU" section in the hardware manual of the product used.

(20) PID — Processor ID

The PID register retains a processor identifier that is unique to the CPU. The PID register is a read-only register.

CAUTION

The PID register indicates information used to identify the equipped CPU core and CPU core configuration. Note that not all the CPU functions can be identified. In case of changing a software behavior based on the PID register information, whether the target behavior can be identified or not should be noted.

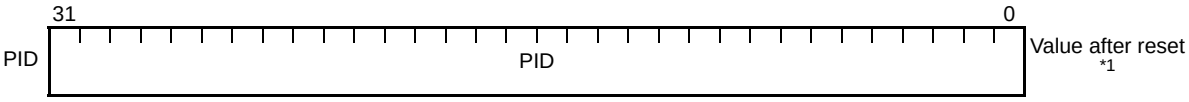


Table 3.2 PID Register Contents

Bit Position	Bit Name	Function	R/W	Value After Reset
31 to 24	PID	Architecture Identifier This identifier indicates the architecture of the processor.	R	*1
23 to 8		Function Identifier This identifier indicates the functions of the processor. These bits indicate whether or not functions defined per bit are implemented 0: Not implemented 1: Implemented	R	*1
7 to 0		Version Identifier This identifier indicates the version of the processor.	R	*1

Bit 23	Virtualization support function (HV)
Bit 22 to 19	Reserved
Bit 18	Register bank
Bit 17 to 12	Reserved
Bit 11	Extended floating-point operation function
Bit 10	Double-precision floating-point operation function
Bit 9	Single-precision floating-point operation function
Bit 8	Memory protection unit (MPU) function

Note 1. For details, see the hardware manual of the product used.

(21) SVLOCK — Supervisor Lock

The SVLOCK register is used to restrict the CPU operation in supervisor mode.

When restricted operating mode is Host mode, HMSVLOCK is selected.

When restricted operating mode is Guest mode, GMSVLOCK is selected.

(22) SCCFG — SYSCALL Operation Setting

This register is used to set operations related to the SYSCALL instruction.

For the SCCFG, see the "CPU" section in the hardware manual of the product used.

(23) SCBP — SYSCALL Base Pointer

The SCBP register is used to specify a table address of the SYSCALL instruction and generate a target address.

For the SCBP, see the "CPU" section in the hardware manual of the product used.

(24) PEID — Processor Element Identifier

The PEID register indicates the processor element identifier.

When restricted operating mode is Host mode, HMPEID is selected.

When restricted operating mode is Guest mode, GMPEID is selected.

(25) BMID — Bus Master Identifier

The BMID register indicates the bus master identifier.

For the BMID, see the "CPU" section in the hardware manual of the product used.

(26) MEA — Memory Error Address

The MEA register holds the address in which an MAE (misalignment) or MPU violation occurred.

When restricted operating mode is Host mode, HMMEA is selected.

When restricted operating mode is Guest mode, GMMEA is selected.

(27) MEI — Memory Error Information

The MEI register holds the information about the instruction that caused a misalignment exception (MAE) or memory protection exception (MDP). The information can be used as hint information for the emulation by software.

When restricted operating mode is Host mode, HMMEI is selected.

When restricted operating mode is Guest mode, GMMEI is selected.

(28) RBCR0 — Register Bank Control 0

The PBCR0 specifies the register bank operation.

When the virtualization support function is enabled and in Host mode (HVCFG.HVE = 1, PSWH.GM = 0), the settings of this register are disabled and register banks can not be used.

For the RBCR0, see the "CPU" section in the hardware manual of the product used.

(29) RBCR1 — Register Bank Control 1

The RBCR1 register controls the operation of the register bank function.

For the RBCR1, see the "CPU" section in the hardware manual of the product used.

(30) RBNR — Register Bank Number

The RBNR register indicates the number of the register bank to be used next.

For the RBNR, see the "CPU" section in the hardware manual of the product used.

(31) RBIP — Register Bank Initial Pointer

The RBIP register indicates the start address of the memory area where the register bank is located.

For the RBIP, see the "CPU" section in the hardware manual of the product used.

3.3 Interrupt Function Registers

3.3.1 Interrupt Function system Registers

Among interrupt function system registers and register bank control registers, the registers whose specifications have been changed from the "CPU" section in the hardware manual of the product used are described in this section. For registers whose specifications have not been changed, see the "CPU" section in the hardware manual of the product used.

Table 3.3 Interrupt Function System Registers

Register Number (regID, selID)	Symbol	Function	Access Authority	
			HVE = 0	HVE = 1
SR10, 2	ISPR	Priority of interrupt being serviced	SV	SV
SR11, 2	IMSR	Interrupt mask status	SV	SV
SR12, 2	ICSR	Interrupt control status	SV	SV
SR13, 2	INTCFG	Interrupt function setting	SV	SV
SR14, 2	PLMR	Interrupt priority masking	SV	SV

(1) ISPR — Priority of Interrupt being Serviced

The ISPR register holds the interrupt priority of the EI level interrupt (EIINTn) being processed by the CPU for each priority. Priority ceiling by interrupt priority is performed when multiplexed interrupts occur.

For the ISPR, see the "CPU" section in the hardware manual of the product used.

(2) IMSR — Interrupt Mask Status

The IMSR register is a register that indicates that interrupts notified to the CPU are masked for acceptance by the mask function in the CPU. For details of interrupt requests for updating each bit, see **Section 3.3.1 (2), (a) Interrupt request for updating IMSR**.

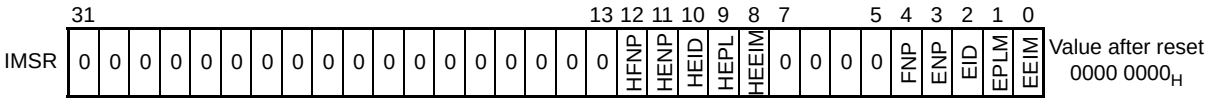


Table 3.4 IMSR Register Contents (1/2)

Bit Position	Bit Name	Function	R/W	Value After Reset
31 to 13	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
12	HFNP	In Guest mode, indicates that FEINT exits for which acceptance is masked by host context PSW.NP. 0: FEINT masked by host context PSW.NP does not exist. 1: FEINT masked by host context PSW.NP exists. In Host mode or when HVCFG.HVE is cleared (0), the value of HFNP is always 0.	R	0
11	HENP	In Guest mode, indicates that EIINT, BGFEINT, BGEIINT exists for which acceptance is masked by host context PSW.NP. 0: EIINT, BGFEINT, BGEIINT masked by host context PSW.NP do not exist. 1: EIINT, BGFEINT, BGEIINT masked by host context PSW.NP exist. In Host mode or when HVCFG.HVE is cleared (0), the value of HENP is always 0.	R	0
10	HEID	In Guest mode, indicates that EIINT, BGFEINT, BGEIINT exists for which acceptance is masked by host context PSW.ID. 0: EIINT, BGFEINT, BGEIINT masked by host context PSW.ID do not exist. 1: EIINT, BGFEINT, BGEIINT masked by host context PSW.ID exist. In Host mode or when HVCFG.HVE is cleared (0), the value of HEID is always 0.	R	0
9	HEPLM	In Guest mode, indicates that EIINT exists for which acceptance is masked by host context PLMR.PLM. 0: EIINT masked by host context PLMR.PLM does not exist. 1: EIINT masked by host context PLMR.PLM exists. In Host mode or when HVCFG.HVE is cleared (0), the value of HEPLM is always 0.	R	0
8	HEEIM	In Guest mode, indicates that ISPR.ISP exists when INTCFG.EPL of the host context is cleared (0), and EIINT exists for which acceptance is masked by host context PSW.EIMASK when host context INTCFG.EPL is set (1). 0: ISPR.ISP or EIINT masked by host context PSW.EIMASK does not exist. 1: ISPR.ISP or EIINT masked by host context PSW.EIMASK exists. In Host mode or when HVCFG.HVE is cleared (0), the value of HEEIM is always 0.	R	0
7 to 5	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
4	FNP	In Host mode, indicates that FEINT exists for which acceptance is masked by PSW.NP. In Guest mode, indicates that GMFEINT and BGFEINT exist for which acceptance is masked by GMPSW.NP. 0: FEINT masked by PSW.NP, and GMFEINT and BGFEINT masked by GMPSW.NP do not exist. 1: FEINT masked by PSW.NP, and GMFEINT and BGFEINT masked by GMPSW.NP exist.	R	0
3	ENP	In Host mode, indicates that EIINT exists for which acceptance is masked by PSW.NP. In Guest mode, indicates that GMEIINT and BGEIINT exist for which acceptance is masked by GMPSW.NP. 0: EIINT masked by PSW.NP, and GMEIINT and BGEIINT masked by GMPSW.NP do not exist. 1: EIINT masked by PSW.NP, and GMEIINT and BGEIINT masked by GMPSW.NP exist.	R	0

Table 3.4 IMSR Register Contents (2/2)

Bit Position	Bit Name	Function	R/W	Value After Reset
2	EID	In Host mode, indicates that EIINT exists for which acceptance is masked by PSW.ID. In Guest mode, indicates that GMEIINT and BGEIINT exist for which acceptance is masked by GMPSW.ID. 0: EIINT masked by PSW.ID, and GMEIINT and BGEIINT masked by GMPSW.ID do not exist. 1: EIINT masked by PSW.ID, and GMEIINT and BGEIINT masked by GMPSW.ID exist.	R	0
1	EPLM	In Host mode, indicates that EIINT exists for which acceptance is masked by PLMR.PLM. In Guest mode, indicates that GMEIINT and BGEIINT exist for which acceptance is masked by GMPLMR.PLM. 0: EIINT masked by PLMR.PLM, and GMEIINT and BGEIINT masked by GMPLMR.PLM do not exist. 1: EIINT masked by PLMR.PLM, and GMEIINT and BGEIINT masked by GMPLMR.PLM exist.	R	0
0	EEIM	The specification of this bit is the same as ICSR.PMEI. In Host mode, indicates that ISPR.ISP exists when INTCFG.EPL is cleared (0), and EIINT exists for which acceptance is masked by PSW.EIMASK when INTCFG.EPL is set (1). In Guest mode, indicates that GMEIINT and BGEIINT exist for which acceptance is masked by GMPSW.EIMASK. 0: ISPR.ISP or EIINT masked by PSW.EIMASK, and GMEIINT and BGEIINT masked by GMPSW.EIMASK do not exist. 1: ISPR.ISP or EIINT masked by PSW.EIMASK, and GMEIINT and BGEIINT masked by GMPSW.EIMASK exist.	R	0

(a) Interrupt request for updating IMSR

Interrupt requests for updating each bit of IMSR are as follows depending on the operating mode.

Table 3.5 Interrupt request for updating each bit of IMSR*1

Bits of IMSR	HVCFG.HVE=0	HVCFG.HVE=1	
		Host mode*2	Guest mode*2
HFNP	Fixed to 0	Fixed to 0	FEINT
HENP	Fixed to 0	Fixed to 0	EIINT, BGFEINT*3, BGEIINT
HEID	Fixed to 0	Fixed to 0	
HEPLM	Fixed to 0	Fixed to 0	EIINT
HEEIM	Fixed to 0	Fixed to 0	
FNP	FEINT	FEINT	GMFEINT, BGFEINT*3
ENP	EIINT	EIINT	GMEIINT, BGEIINT
EID			
EPLM			
EEIM			

Note 1. Each interrupt request is requested after acceptance judgment by bind of interrupts. Immediately after the CPU changes from Guest mode to Host mode, when the request selection on the interrupt controller side was not immediately changed and the CPU was notified of an interrupt request bound to Guest mode, all IMSR bits related to that interrupt request are cleared (0).

Note 2. Indicates the CPU operating mode when these interrupts are accepted.

Note 3. BGFEINT is an FE level exception in Guest mode, but as it is an EI level exception in Host mode, the exception level of the bit to be set differs.

(b) Order of updating IMSR

Acceptance conditions are confirmed in the order of ISPR or PSW.EIMASK (selected by INTCFG.EPL), PLMR, PSW.ID, PSW.NP in the exception bound to the conventional mode or host mode. In the exception bound to Guest mode, the reception conditions are confirmed in the order of GMPSW.EIMASK, GMPLMR, GMPSW.ID, GMPSW.NP. Note that when an acceptance condition is set to a value that masks an interrupt, the interrupt request is masked there, and the acceptance conditions that are in the subsequent order are not confirmed. Conditions that are not considered as acceptance conditions depending on the type of exception are not confirmed.

Since EI level interrupt has multiple acceptance conditions, it is set one bit at a time according to the order of confirmation of reception conditions as follows.

Table 3.6 Bits of IMSR set (1) by EI level interrupt (1/2)

	HENP	HEID	HEPLM	HEEIM
HEMP is set (1)	1	0*1	0*1	0*1
HEID is set (1)	0*2	1	0*1	0*1
HEPLM is set (1)	0*2	0*2	1	0*1
HEEIM is set (1)	0*2	0*2	0*2	1

Table 3.7 Bits of IMSR set (1) by EI level interrupt (2/2)

	ENP	EID	EPLM	EEIM
ENP is set (1)	1	0*1	0*1	0*1
EID is set (1)	0*2	1	0*1	0*1
EPLM is set (1)	0*2	0*2	1	0*1
EEIM is set (1)	0*2	0*2	0*2	1

Note 1. When an interrupt is masked under an acceptance condition, it is unmasked in acceptance conditions confirmed before that condition. The bits corresponding to such unmasked acceptance conditions are cleared (0).

Note 2. When an interrupt has already been masked with acceptance conditions that is confirmed in the order before a certain acceptance condition, acceptance is not confirmed under the acceptance conditions confirmed in the order after the acceptance condition masking the interrupt. Therefore, even if an acceptance condition for which acceptance is not confirmed is set to the value that masks the interrupt, the bit corresponding to such acceptance condition is cleared (0).

BGFEINT and BGEIINT are masked by the host context register (PSW) and guest context register (GMPSW, GMPLMR). When BGFEINT or BGEIINT occurs, each bit of IMSR becomes the following value for each interrupt acceptance condition setting.

Table 3.8 The value of IMSR when BGFEINT is masked

GMPSW.NP	PSW.ID	PSW.NP	FNP	HEID	HENP
0	0	0	0	0	0
0	0	1	0	0	1
0	1	d/c	0	1	0
1	0	0	1	0	0
1	0	1	1	0	1
1	1	d/c	1	1	0

Table 3.9 The value of IMSR when BGEIINT is masked

GMPSW.EIMASK	GMPLMR	GMPSW.ID	GMPSW.NP	PSW.ID	PSW.NP	EEIM	EPLM	EID	ENP	HEID	HENP
1*1	d/c	d/c	d/c	d/c	d/c	1	0	0	0	0	0
0*1	1*2	d/c	d/c	d/c	d/c	0	1	0	0	0	0
0*1	0*2	1	d/c	0	0	0	0	1	0	0	0
0*1	0*2	1	d/c	0	1	0	0	1	0	0	1
0*1	0*2	1	d/c	1	d/c	0	0	1	0	1	0
0*1	0*2	0	1	0	0	0	0	0	1	0	0
0*1	0*2	0	1	0	1	0	0	0	1	0	1
0*1	0*2	0	1	1	d/c	0	0	0	1	1	0
0*1	0*2	0	0	0	0	0	0	0	0	0	0
0*1	0*2	0	0	0	1	0	0	0	0	0	1
0*1	0*2	0	0	1	d/c	0	0	0	0	1	0

Note 1. 1 when GMPSW.EIMASK satisfies the mask condition or 0 otherwise.

Note 2. 1 when GMPLMR satisfies the mask condition or 0 otherwise.

(3) ICSR — Interrupt Control Status

This register indicates the status of interrupt control inside the CPU.

ICSR can be used only when HVCFG.HVE = 0. When HVCFG.HVE = 1, ICSR is an undefined register.

For details of ICSR, see the "CPU" section in the hardware manual of the product used.

(4) INTCFG — Interrupt Function Setting

This register is used to specify settings related to the CPU's internal interrupt function.

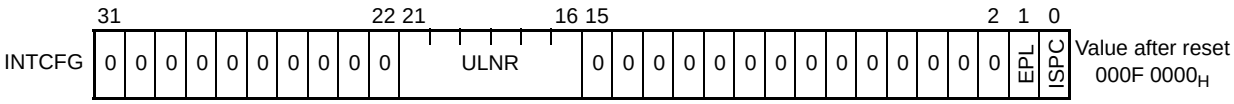


Table 3.10 INTCFG Register Contents (1/2)

Bit Position	Bit Name	Function	R/W	Value After Reset
31 to 22	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
21 to 16	ULNR	Specifying the maximum value of available register bank numbers. If the value of the RBNR.BN is bigger than the ULNR, or the value of the RBNR.BN is 63; and the interrupt (EIINTn) whose register bank function is enable occurs, the SYSERR exception will occur. Note that the interrupt (EIINTn) is not accepted and is held. When the HVCFG.HVE bit is set (1), updating of this bit is possible but register bank can not be used, so the value of this bit does not affect the operation of the CPU.	R/W	$0F_H$
15 to 2	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
1	EPL	For the interrupt (EIINTn), specify whether to enable interrupt priority level extension function. 0: Interrupt priority level extension function is disabled 1: Interrupt priority level extension function is enabled The following is the overview of interrupt operation by setting this bit. For details, see **Section 4.1.5, Interrupt Exception Priority and Priority Masking** , **Section 4.4, Exception Handler Address**, **Section 4.5, Register Bank Function**.	R/W	0

Interrupt function		Value of EPL = 0	Value of EPL = 1
Number of interrupt priority level		16	64
Mask Function	ISPR	Function is enabled	Function is disabled
	PSW.EIMASK	Function is disabled	Function is enabled
	PLMR	Function is enabled	
	PSW.NP	Function is enabled	
	PSW.ID	Function is enabled	
INTCFG.ISPC		Settings can be changed	Fixed 0
ICSR.PMEI		Function is enabled	
IMSR		Function is enabled	
Handler generation	Direct vector method	Each vector can be used by each priority.	Each vector can be used by priority 0 to 14. The priority 15 vector is shared with the priority 15 or greater.
	Table reference method	Available	
Register bank		The usage can be set by each priority.	Priority 0 to 14 can be individually set. The priority 15 setting is shared with the priority 15 or greater.

Table 3.10 INTCFG Register Contents (2/2)

Bit Position	Bit Name	Function	R/W	Value After Reset
0	ISPC	This bit changes how the ISPR register is written. 0: The ISPR register is automatically updated. Updates triggered by the program (via execution of LDSR instruction) are ignored. 1: The ISPR register is not automatically updated. Updates triggered by the program (via execution of LDSR instruction) are performed. If this bit is cleared to 0, the bits of the ISPR register are automatically set to 1 when an interrupt (EIINT*n*) is acknowledged, and cleared to 0 when the EIRET instruction is executed. In this case, the bits are not updated by an LDSR instruction executed by the program. If this bit is set to 1, the bits of the ISPR register are not updated by the acknowledgement of an interrupt (EIINT*n*) or by execution of the EIRET instruction. In this case, the bits can be updated by an LDSR instruction executed by the program. In normal cases, the ISPC bit should be cleared. When performing software-based priority control, however, set this bit (1). For details, see **(2), Interrupt Priority Mask** in **Section 4.1.5, Interrupt Exception Priority and Priority Masking**.	R/W	0

(5) PLMR — Interrupt Priority Level Mask

This register masks the interrupts (EIINT*n*) whose priority level is not higher than the level specified by these bits.

When restricted operating mode is Host mode, HMPLMR is selected.

When restricted operating mode is Guest mode, GMPLMR is selected.

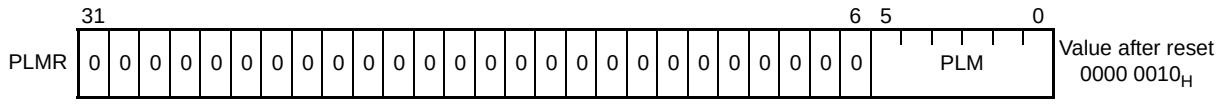


Table 3.11 PLMR Register Contents

Bit Position	Bit Name	Function	R/W	Value After Reset
31 to 6	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
5 to 0	PLM	These bits are used to mask the interrupts (EIINT*n*) whose priority level is not higher than the level specified by these bits. When an interrupt (EIINT*n*) is masked by this register, it is not accepted. The correspondence between the value of the PLM bit and the highest priority of interrupts to be masked is shown below. **Value of PLM Bit** / **Highest Priority of Interrupts to be Masked** 0 / Priority 0 (All priorities are not acceptable) 1 / Priority 1 (Only priority 0 is acceptable) : 14 / Priority 14 (Only priorities 13 or higher are acceptable) 15 / Priority 15 (Only priorities 14 or higher are acceptable) : 62 / Priority 62 (Only priorities 61 or higher are acceptable) 63 / Priority 63 (Only priorities 62 or higher are acceptable) Since the highest priority level of an interrupt that is defined for this CPU is 0, if 0 is specified in the PLM bit, all interrupts (EIINT*n*) are masked by this register. Since the lowest interrupt priority level defined by this CPU is 63, the interrupts (EIINTn) with the interrupt priority of 63 are always masked. Regardless of INTCFG.EPL setting, the interrupt mask by PLMR is done by the value of the PLM bit and the value of the interrupt priority notified from the interrupt controller*1. If the value of PLMR is altered by the LDSR instruction, the new PLMR value is reflected in the instructions following that LDSR instruction.	R/W	10_H

Note 1. Even if INTCFG.EPL is cleared (0), it is possible to receive an interrupt (EIINTn) with priority 16 or less from the interrupt controller. Even if INTCFG.EPL is cleared (0), PLMR will not be automatically reduced to 4 bits, and PLMR always masks interrupts for interrupt priority level of up to 64 level.

3.4 FPU Function Registers

See the "CPU" section in the hardware manual of the product used.

3.5 FXU Function Registers

See the "CPU" section in the hardware manual of the product used.

3.6 MPU Function Registers

3.6.1 MPU Function System Registers

Among the system registers of the memory protection function, this section describes the registers whose specifications have been changed from the "CPU" section in the hardware manual of the product used. For registers whose specifications have not been changed, see the "CPU" section in the hardware manual of the product used.

Table 3.12 MPU Function System Registers

Register Number (regID, selID)	Symbol	Function	Access Authority	
			HVE = 0	HVE = 1
SR0, 5	MPM	Memory protection operation mode setting	SV	SV
SR2, 5	MPCFG	MPU configuration	SV	HV*1
SR8, 5	MCA	Memory protection setting check address	SV	SV
SR9, 5	MCS	Memory protection setting check size	SV	SV
SR10, 5	MCC	Memory protection setting check command	SV	SV
SR11, 5	MCR	Memory protection setting check result	SV	SV
SR12, 5	MCI	Memory protection setting check SPID	SV	SV
SR16, 5	MPIDX	Index of memory protection setting registers to be accessed	SV	SV
SR17, 5	MPBK	MPU Bank Setting	SV	HV
SR20, 5	MPLA	Protection area minimum address	SV	SV
SR21, 5	MPUA	Protection area maximum address	SV	SV
SR22, 5	MPAT	Protection area attribute	SV	SV
SR24, 5	MPID0	SPID which can access protection area	SV	HV*1
SR25, 5	MPID1	SPID which can access protection area	SV	HV*1
SR26, 5	MPID2	SPID which can access protection area	SV	HV*1
SR27, 5	MPID3	SPID which can access protection area	SV	HV*1
SR28, 5	MPID4	SPID which can access protection area	SV	HV*1
SR29, 5	MPID5	SPID which can access protection area	SV	HV*1
SR30, 5	MPID6	SPID which can access protection area	SV	HV*1
SR31, 5	MPID7	SPID which can access protection area	SV	HV*1

Note 1. Reading is possible with SV privilege.

(1) MPM — Memory Protection Operation Mode

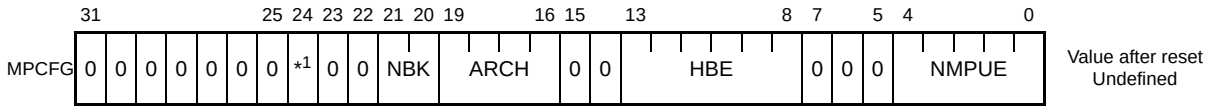
The memory protection operation mode register is used to define the basic operation mode of the memory protection function.

When restricted operating mode is Host mode, HMMPM is selected.

When restricted operating mode is Guest mode, GMMPM is selected.

(2) MPCFG — MPU Configuration

This register holds the information about the configuration of the MPU.



Note 1. Undefined

Table 3.13 MPCFG Register Contents

Bit Position	Bit Name	Function	R/W	Value After Reset
31 to 25	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
24	—	(Reserved for future expansion. Be sure to set to 0.)	R	Undefined
23, 22	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
21, 20	NBK	Indicates the "number of banks - 1" of the MPU bank *1 equipped in this CPU. Since the MPU bank of this CPU is equipped with one bank, 0 is read.	R	0*2
19 to 16	ARCH	These bits indicate the version of the MPU architecture specifications. 1: MPU specification corresponding to RH850 version 2.0 2 : MPU specification corresponding to RH850 version 2.1 Values other than the above are reservations for the future. These bits are not read out by the CPU corresponding to the RH850v2 architecture.	R	2
15, 14	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
13 to 8	HBE	Indicates the first entry number of the host management entry. As a result, the entry from 0 to "HBE - 1" is the guest management entry, and the entry from HBE to NMPUE is the host management entry. Note that when the value of HBE is 0, it indicates that no guest management entry is set. And when the value of HBE is NMPUE+1, it indicates that no host management entry is set. When HVCFG.HVE = 0, this bit has no specific function and automatically cleared (0).	R/W	0
7 to 5	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
4 to 0	NMPUE	These bits indicate the “number of entries -1” of MPU entries implemented in this CPU. A value of 31 is read since this CPU incorporates 32 MPU entries.	R	31

Note 1. The MPU bank is the name when the number of MPU entries indicated by the NMPUE are handled as one pair. If two banks are equipped, the number of MPU entries twice as large as the value indicated by the NMPUE are equipped. However, each MPU bank is used exclusively. Only the MPU bank indicated by the MPBK register is used for memory protection. Therefore, even if multiple banks are equipped, the maximum number of the MPU entries that can be used simultaneously for memory protection is up to the value indicated by the NMPUE.

Note 2. The value of this bit is changed according to the hardware manual of the product used. In this CPU it is fixed to 0.

(3) MCA — Memory Protection Setting Check Address

This register is used to specify the base address of the area for which a memory protection setting check is to be performed.

For details of MCA, see the "CPU" section in the hardware manual of the product used.

(4) MCS — Memory Protection Setting Check Size

This register is used to specify the size of the area for which a memory protection setting check is to be performed.

For details of MCS, see the "CPU" section in the hardware manual of the product used.

(5) MCC — Memory Protection Setting Check Command

This command register is used to start a memory protection setting check.

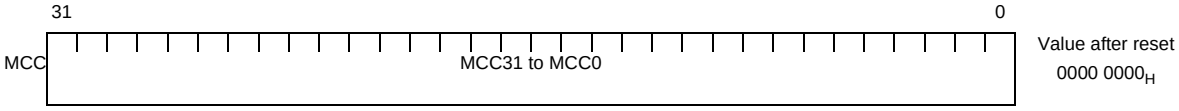


Table 3.14 MCC Register Contents

Bit Position	Bit Name	Function	R/W	Value After Reset
31 to 0	MCC31 to MCC0	When any value is written to the MCC register, a memory protection setting check starts. By setting up the MCA / MCS register and then writing to the MCC register, results are stored in MCR*1. Because the check is started by any written value, a check can be started by using r0 as the source register without using any unnecessary registers. Note that, for the check, the results are applied according to each area setting regardless of the state of the PSW.UM bit. This check is performed for the MPU bank memory protection setting indicated by MPBK. When the MCC register is read, value 0000 0000_H is always returned.	R/W	0

Note 1. Even if CPU operating mode is Host mode or Guest mode, the check result of the memory protection setting for the current operating mode is stored in the MCR register.

(6) MCR — Memory Protection Setting Check Result

This register is used to store the results of a memory protection setting check.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCR	0	0	HSXE	HSWE	HSRE	HUXE	HUWE	HURE	0	0	GSXE	GSWE	GSRE	GUXE	GUWE	GURE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	OV	0	0	SXE	SWE	SRE	UXE	UWE	URE	Value after reset Undefined

Table 3.15 MCR Register Contents (1/2)

Bit Position	Bit Name	Function	R/W	Value After Reset
31, 30	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
29	HSXE	When the specified area is within one of the protected areas of the host management entry and the protected area is supervisor execution enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
28	HSWE	When the specified area is within one of the protected areas of the host management entry and the protected area is supervisor write enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
27	HSRE	When the specified area is within one of the protected areas of the host management entry and the protected area is supervisor read enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
26	HUXE	When the specified area is within one of the protected areas of the host management entry and the protected area is user execution enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
25	HUWE	When the specified area is within one of the protected areas of the host management entry and the protected area is user write enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
24	HURE	When the specified area is within one of the protected areas of the host management entry and the protected area is user read enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
23, 22	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
21	GSXE	When the specified area is within one of the protected areas of the guest management entry and the protected area is supervisor execution enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
20	GSWE	When the specified area is within one of the protected areas of the guest management entry and the protected area is supervisor write enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
19	GSRE	When the specified area is within one of the protected areas of the guest management entry and the protected area is supervisor read enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
18	GUXE	When the specified area is within one of the protected areas of the guest management entry and the protected area is user execution enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined

Table 3.15 MCR Register Contents (2/2)

Bit Position	Bit Name	Function	R/W	Value After Reset
17	GUWE	When the specified area is within one of the protected areas of the guest management entry and the protected area is user write enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
16	GURE	When the specified area is within one of the protected areas of the guest management entry and the protected area is user write enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored.	R/W	Undefined
15 to 9	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
8	OV	If the specified area includes 0000 0000_H or 7FFF $FFFF_H$, 1 is stored in this bit. In other cases, 0 is stored in this bit.	R/W	Undefined
7, 6	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
5	SXE	When the specified area is within one of the protected areas and the protected area is supervisor execution enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored. When HVCFG.HVE = 1 and SXE is 1, both HSXE and GSXE are 1. When SXE is 0, either HSXE or GSXE or both are 0.	R/W	Undefined
4	SWE	When the specified area is within one of the protected areas and the protected area is supervisor write enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE is cleared (0), 0 is always stored. When HVCFG.HVE = 1 and SWE is 1, both HSWE and GSWE are 1. When SWE is 0, either HSWE or GSWE or both are 0.	R/W	Undefined
3	SRE	When the specified area is within one of the protected areas and the protected area is supervisor read enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE = 1 and SRE is 1, both HSRE and GSRE are 1. When SRE is 0, either HSRE or GSRE or both are 0.	R/W	Undefined
2	UXE	When the specified area is within one of the protected areas and the protected area is use execution enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE = 1 and UXE is 1, both HUXE and GUXE are 1. When UXE is 0, either HUXE or GUXE or both are 0.	R/W	Undefined
1	UWE	When the specified area is within one of the protected areas and the protected area is user write enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE = 1 and UWE is 1, both HUWE and GUWE are 1. When UWE is 0, either HUWE or GUWE or both are 0.	R/W	Undefined
0	URE	When the specified area is within one of the protected areas and the protected area is user read enable, 1 is stored. Otherwise, 0 is stored. When HVCFG.HVE = 1 and URE is 1, both HURE and GURE are 1. When URE is 0, either HURE or GURE or both are 0.	R/W	Undefined

(7) MCI — Memory Protection Setting Check SPID

This register is used to specify the SPID for which a memory protection settings check is to be performed.

For details of MCI, see the "CPU" section in the hardware manual of the product used.

(8) MPIDX — Index of Memory Protection Setting Registers to be Accessed

This register is used to specify the index of memory protection setting registers to be accessed.

For details of MPIDX, see the "CPU" section in the hardware manual of the product used.

(9) MPBK — MPU Bank Setting

The MPBK register selects the MPU bank.

For details of MPBK, see the "CPU" section in the hardware manual of the product used.

(10) MPLA — Protection Area Minimum Address

This register indicates the minimum address of a protection area.

For details of MPLA, see the "CPU" section in the hardware manual of the product used.

(11) MPUA — Protection Area Maximum Address

This register indicates the maximum address of a protection area.

For details of MPUA, see the "CPU" section in the hardware manual of the product used.

(12) MPAT — Protection Area Attribute

This register indicates the attribute of a protection area.

For details of MPAT, see the "CPU" section in the hardware manual of the product used.

(13) MPIDn —SPID which can Access Protection Area

This register specifies the SPID which can access protection area.

For details of MPIDn, see the "CPU" section in the hardware manual of the product used.

3.7 Cache Operation Function Registers

3.7.1 Cache Control Function System Registers

Cache control function system registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID.

Table 3.16 Cache Control Function System Registers

Register Number (regID, selID)	Symbol	Function	Access Authority	
			HVE = 0	HVE = 1
SR16, 4	ICTAGL	Instruction cache tag Lo access	SV	HV
SR17, 4	ICTAGH	Instruction cache tag Hi access	SV	HV
SR18, 4	ICDATL	Instruction cache data Lo access	SV	HV
SR19, 4	ICDATH	Instruction cache data Hi access	SV	HV
SR24, 4	ICCTRL	Instruction cache control	SV	HV*1
SR26, 4	ICCFG	Instruction cache configuration	SV	SV
SR28, 4	ICERR	Instruction cache error	SV	SV

Note 1. Reading is possible with SV privilege.

(1) ICTAGL — Instruction Cache Tag Lo Access

This register is used by the CIST/CILD instruction in relation to the instruction cache. During execution of the CIST instruction, values that are stored to the tag RAM for the instruction cache are stored. During execution of the CILD instruction, values read from the tag RAM for the instruction cache are stored.

For details of ICTAGL, see the "CPU" section in the hardware manual of the product used.

(2) ICTAGH — Instruction Cache Tag Hi Access

This register is used by the CIST/CILD instruction in relation to the instruction cache. During execution of the CIST instruction, values that are stored to the tag RAM for the instruction cache are stored. During execution of the CILD instruction, values read from the tag RAM for the instruction cache are stored.

For details of ICTAGH, see the "CPU" section in the hardware manual of the product used.

(3) ICDATL — Instruction Cache Data Lo Access

This register is used by the CIST/CILD instruction in relation to the instruction cache. During execution of the CIST instruction, values that are stored to the data RAM for the instruction cache are stored. During execution of the CILD instruction, values read from the data RAM for the instruction cache are stored.

For details of ICDATL, see the "CPU" section in the hardware manual of the product used.

(4) ICDATH — Instruction Cache Data Hi Access

This register is used by the CIST/CILD instruction in relation to the instruction cache. During execution of the CIST instruction, values that are stored to the data RAM for the instruction cache are stored. During execution of the CILD instruction, values read from the data RAM for the instruction cache are stored.

For details of ICDATH, see the "CPU" section in the hardware manual of the product used.

(5) ICCTRL — Instruction Cache Control

This register is used to control the instruction cache.

For details of ICCTRL, see the "CPU" section in the hardware manual of the product used.

(6) ICCFG — Instruction Cache Configuration

This register indicates the instruction cache configuration.

For details of ICCFG, see the "CPU" section in the hardware manual of the product used.

(7) ICERR — Instruction Cache Error

This register is used to store cache error information for the instruction cache.

For details of ICERR, see the "CPU" section in the hardware manual of the product used.

3.8 Count Function Registers

3.8.1 Count Function System Registers

Count function system registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID.

Table 3.17 Count Function System Registers

Register Number (regID, selID)	Symbol	Function	Access Authority*1	
			HVE = 0	HVE = 1
SR0, 11	TSCOUNTL	Timestamp count L register	SV*4	HV*4
SR1, 11	TSCOUNTH	Timestamp count H register	SV*4	HV*4
SR2, 11	TSCTRL	Timestamp count control register	SV	HV
SR8, 11	PMUMCTRL	Performance counter User mode control register	SV	SV
SR9, 11	PMGMCTRL	Performance counter Guest mode control register	SV	HV
SR0, 14	PMCTRL0	Performance count control 0 register	SV*2	HV*2*3
SR1, 14	PMCTRL1	Performance count control 1 register	SV*2	HV*2*3
SR2, 14	PMCTRL2	Performance count control 2 register	SV*2	HV*2*3
SR3, 14	PMCTRL3	Performance count control 3 register	SV*2	HV*2*3
SR4, 14	PMCTRL4	Performance count control 4 register	SV*2	HV*2*3
SR5, 14	PMCTRL5	Performance count control 5 register	SV*2	HV*2*3
SR6, 14	PMCTRL6	Performance count control 6 register	SV*2	HV*2*3
SR7, 14	PMCTRL7	Performance count control 7 register	SV*2	HV*2*3
SR16, 14	PMCOUNT0	Performance count 0 register	SV*2	HV*2*3
SR17, 14	PMCOUNT1	Performance count 1 register	SV*2	HV*2*3
SR18, 14	PMCOUNT2	Performance count 2 register	SV*2	HV*2*3
SR19, 14	PMCOUNT3	Performance count 3 register	SV*2	HV*2*3
SR20, 14	PMCOUNT4	Performance count 4 register	SV*2	HV*2*3
SR21, 14	PMCOUNT5	Performance count 5 register	SV*2	HV*2*3
SR22, 14	PMCOUNT6	Performance count 6 register	SV*2	HV*2*3
SR23, 14	PMCOUNT7	Performance count 7 register	SV*2	HV*2*3
SR0, 15	PMSUBCND0	Performance count subcondition 0 register	SV*2	HV*2*3
SR1, 15	PMSUBCND1	Performance count subcondition 1 register	SV*2	HV*2*3
SR2, 15	PMSUBCND2	Performance count subcondition 2 register	SV*2	HV*2*3
SR3, 15	PMSUBCND3	Performance count subcondition 3 register	SV*2	HV*2*3
SR4, 15	PMSUBCND4	Performance count subcondition 4 register	SV*2	HV*2*3
SR5, 15	PMSUBCND5	Performance count subcondition 5 register	SV*2	HV*2*3
SR6, 15	PMSUBCND6	Performance count subcondition 6 register	SV*2	HV*2*3
SR7, 15	PMSUBCND7	Performance count subcondition 7 register	SV*2	HV*2*3

Note 1. The access authority may be different depending on the value of HVCFG.HVE.
Note 2. By setting PMUMCTRL register, access with user authority is enabled.
Note 3. By setting the PMGMCTRL register, access in guest mode is enabled.
Note 4. Regardless of whether the virtualization support function is enabled or disabled, reading can be done with UM authority. Only writing authority changes.

(1) TSCOUNTL — Timestamp Count L

This register constitutes the time stamp counter.

For details of TSCOUNTL, see the "CPU" section in the hardware manual of the product used.

(2) TSCOUNTH — Timestamp Count H

This register constitutes the time stamp counter.

For details of TSCOUNTH, see the "CPU" section in the hardware manual of the product used.

(3) TSCTRL — Timestamp Count Control

This register constitutes the time stamp counter.

For details of TSCTRL, see the "CPU" section in the hardware manual of the product used.

(4) PMUMCTRL — Performance Counter User Mode Control

In User mode, this register specifies whether or not access to system registers of the performance measurement function is available.

For details of PMUMCTRL, see the "CPU" section in the hardware manual of the product used.

(5) PMGMCTRL — Performance Counter Guest Mode Control

When HVCFG.HVE is set (1), this register specifies whether or not access to the system registers of the performance measurement function in Guest mode is available. For each channel of the performance measurement function, specify whether the corresponding system registers can be accessible.

Accessing these system registers when access in Guest mode is disabled causes the PIE exception.

When HVCFG.HVE is cleared (0), this register is treated as an undefined register.

	31																							8	7	6	5	4	3	2	1	0	
PMGMCTRL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PMGE7	PMGE6	PMGE5	PMGE4	PMGE3	PMGE2	PMGE1	PMGE0	Value after reset 0000 0000$_H$

Table 3.18 PMGMCTRL Register Contents

Bit Position	Bit Name	Function	R/W	Value After Reset
31 to 8	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
7 to 0	PMGEn n = 7 to 0	Specify whether PMCOUNTn register, PMCTRLn register and PMSUBCNDn register can be accessed in Guest mode. 0: Access to PMCOUNTn/PMCTRLn/PMSUBCNDn in guest mode is disabled. 1: Access to PMCOUNTn/PMCTRLn/PMSUBCNDn in guest mode is enabled.	R/W	0

(6) PMCTRLn — Performance Count Control

This register controls the counting operation of the PMCOUNTn register. This CPU has 8 channels (n = 0 to 7) of performance count control registers.

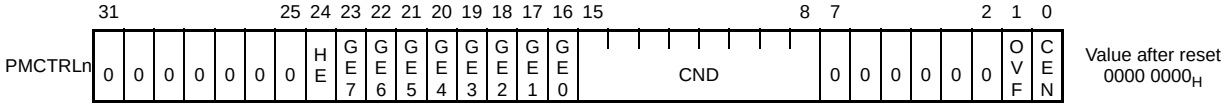


Table 3.19 PMCTRLn Register Contents (1/2)

Bit Position	Bit Name	Function	R/W	Value After Reset
31 to 25	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
24	HE	In Host mode, indicates whether or not counting is possible. 0: In Host mode, it does not count even if an event occurs. 1: In Host mode, it counts when event occurs. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0
23	GE7	In Guest mode, indicates whether or not counting is possible when PSWH.GPID = 7. 0: In Guest mode, it does not count even if an event occurs when PSWH.GPID = 7. 1: In Guest mode, it counts if an event occurs when PSWH.GPID = 7. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0
22	GE6	In Guest mode, indicates whether or not counting is possible when PSWH.GPID = 6. 0: In Guest mode, it does not count even if an event occurs when PSWH.GPID = 6. 1: In Guest mode, it counts if an event occurs when PSWH.GPID = 6. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0
21	GE5	In Guest mode, indicates whether or not counting is possible when PSWH.GPID = 5. 0: In Guest mode, it does not count even if an event occurs when PSWH.GPID = 5. 1: In Guest mode, it counts if an event occurs when PSWH.GPID = 5. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0
20	GE4	In Guest mode, indicates whether or not counting is possible when PSWH.GPID = 4. 0: In Guest mode, it does not count even if an event occurs when PSWH.GPID = 4. 1: In Guest mode, it counts if an event occurs when PSWH.GPID = 4. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0
19	GE3	In Guest mode, indicates whether or not counting is possible when PSWH.GPID = 3. 0: In Guest mode, it does not count even if an event occurs when PSWH.GPID = 3. 1: In Guest mode, it counts if an event occurs when PSWH.GPID = 3. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0
18	GE2	In Guest mode, indicates whether or not counting is possible when PSWH.GPID = 2. 0: In Guest mode, it does not count even if an event occurs when PSWH.GPID = 2. 1: In Guest mode, it counts if an event occurs when PSWH.GPID = 2. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0
17	GE1	In Guest mode, indicates whether or not counting is possible when PSWH.GPID = 1. 0: In Guest mode, it does not count even if an event occurs when PSWH.GPID = 1. 1: In Guest mode, it counts if an event occurs when PSWH.GPID = 1. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0
16	GE0	In Guest mode, indicates whether or not counting is possible when PSWH.GPID = 0. 0: In Guest mode, it does not count even if an event occurs when PSWH.GPID = 0. 1: In Guest mode, it counts if an event occurs when PSWH.GPID = 0. When HVCFG.HVE is cleared (0), this bit is reserved. Be sure to set to 0.	R/W	0

Table 3.19 PMCTRLn Register Contents (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th><th>R/W</th><th>Value After Reset</th></tr>
<tr><td>15 to 8</td><td>CND</td><td>The following events have descriptions that are different from the "CPU" section in the hardware manual of the product used. For other events, see the "CPU" section in the hardware manual of the product used.
<table>
<tr><th>Number</th><th>Details of Event</th><th>Speculative Operation</th></tr>
<tr><td>10_H</td><td>Number of executions of all instructions*1 *2</td><td>None</td></tr>
<tr><td>18_H</td><td>Number of instructions that cause branch*2</td><td>None</td></tr>
<tr><td>20_H</td><td>Counts the number of times EI level interrupts are accepted. But it is not counted by accepting BGFEINT and BGEIINT.
When counting in Host mode, counts the number of times EIINT is accepted.
When counting in Guest mode, counts the number of times GMEIINT is accepted.</td><td>None</td></tr>
<tr><td>21_H</td><td>Counts the number of times FE level interrupts are accepted. But it is not counted by accepting BGFEINT and BGEIINT.
When counting in Host mode, counts the number of times FEINT is accepted. When counting in Guest mode, counts the number of times GMFEINT is accepted.</td><td>None</td></tr>
<tr><td>22_H</td><td>Counts the number of times terminating type exceptions are accepted. Note that there are exceptions where the count object changes depending on the CPU operating mode.
When counting in Host mode, FEINT and EIINT, BGFEINT and BGEIINT, SYSERR of terminating type that occurred during Host mode, and GMCFG.GSYSE in guest mode, when these was set (1), counts the number of times of acceptance of the terminating type SYSERR that occurred.
These are not counted even if counting is enabled in Guest mode.
When counting in Guest mode, when GMFEINT and GMEIINT, GMCFG.GSYSE is cleared (0) in Guest mode, in this case counts the number of times of accepting the terminating type SYSERR that occurred. These are not counted even if counting is enabled in Host mode.</td><td>None</td></tr>
<tr><td>23_H</td><td>Counts the number of times resumable type exception and completion type exception are accepted*3. Note that there are exceptions where the count object changes depending on the CPU operating mode.
When counting in Host mode, when GMCFG.HMP or GMCFG.GMP is set (1), in this case count memory protection error exception that occurred. And in this case these are not counted even if counting is enabled in Guest mode.
When counting in Guest mode, when GMCFG.HMP or GMCFG.GMP is cleared (0), in this case count memory protection error exception that occurred. And in this case these are not counted even if counting is enabled in Host mode.</td><td>None</td></tr>
<tr><td>24_H</td><td>Number of times where Guest mode of the GPID specified by PMCTRLn.GE 0-PMCTRLn.GE7 was terminated by BGFEINT or BGEIINT.</td><td>None</td></tr>
<tr><td>25_H</td><td>Number of times BGFEINT or BGEIINT bound to the GPID specified by PMCTRLn.GE 0-PMCTRLn.GE7 occurred.</td><td>None</td></tr>
<tr><td>28_H</td><td>Number of clock cycles during which no interrupt is processed*4.</td><td>None</td></tr>
<tr><td>29_H</td><td>Number of clock cycles during which no interrupt is processed and interrupts are disabled*5.</td><td>None</td></tr>
</table></td><td>R/W</td><td>0</td></tr>
<tr><td>7 to 2</td><td>—</td><td>(Reserved for future expansion. Be sure to set to 0.)</td><td>R</td><td>0</td></tr>
<tr><td>1</td><td>OVF</td><td>This bit serves as the overflow flag.</td><td>R/W</td><td>0</td></tr>
<tr><td>0</td><td>CEN</td><td>This bit enables or disables the count operation of the PMCOUNTn register.
0: Disables count operation.
1: Enables count operation.</td><td>R/W</td><td>0</td></tr>
</table>

Note 1. The notes described in the "CPU" section in the hardware manual of the product used. are applied as they are. And when an instruction which causes a transition from Guest mode to Host mode is executed, the CPU changes to Host mode, but this event is counted according to the operating mode at the time the instruction is executed. Therefore, if PSWH.GPID count setting when those instructions are executed in Guest mode is set to count enabled, this event is counted. Even if counting is enabled in Host mode, if the CPU is in Guest mode and PSWH.GPID count setting at that time is not set to count enabled, even if these instructions are executed, this event is not counted. When counting is enabled in Host mode and these instructions are executed in Host mode, this event is counted.

Note 2. The notes described in the "CPU" section in the hardware manual of the product used. are applied as they

are. And when the return instruction for transitioning from Host mode to Guest mode is executed, when counting is enabled in Host mode, this event is counted. Even if counting of the value set to PSWH.GPID is enabled after changing to Guest mode, if counting in Host mode is not enabled, this event is not counted by execution of the return instruction to change to Guest mode.

Note 3. When an instruction which causes a transition from Guest mode to Host mode is executed, when counting is enabled in Host mode, this event is counted. When these instructions are executed, even if counting is enabled in Guest mode, this event is not counted.

Note 4. When the INTCFG.EPL bit is cleared (0), the period during which the value of ISPR.ISP is 0000h is counted. Therefore, even if the INTCFG.EPL bit is cleared (0), if the INTCFG.ISPC bit is set (1) and ISPR.ISP is not automatically updated, this event is not measured correctly. And even if the INTCFG.EPL bit and the INTCFG.ISPC bit are both cleared (0), when an interrupt request with a priority lower than 16 is accepted and ISPR.ISP is not set (1), also this event is not measured correctly. When the INTCFG.EPL bit is set (1), the period during which the value of PSW.EIMASK is FF_H is counted.

Note 5. For the definition of the period during which interrupt processing is not in progress, the value of CND is the same as 28_H. In addition, as the period of interrupt disable, the period during which the PSW.ID bit is set (1) is counted. Interrupt disable setting by PSW.NP bit and PLMR register is not counted as a period of interrupt disabled.

(7) PMCOUNTn — Performance Count

In the performance measurement function, this register counts the number of occurrences of various events specified by the PMCTRLn register.

For details of PMCOUNTn, see the "CPU" section in the hardware manual of the product used.

(8) PMSUBCNDn — Performance Count Sub condition

This register specifies the subcondition to be used according to the setting of PMCTRLn.CND.

For details of PMSUBCNDn, see the "CPU" section in the hardware manual of the product used.

3.9 Hardware Function Registers

See the "CPU" section in the hardware manual of the product used.

3.10 Virtualization support function system registers

Virtualization support function system registers are the system registers for controlling the virtualization support function. There are newly added system registers which realize the virtualization support function and the system registers whose specifications have been changed. Reading and writing to virtualization support function system registers are performed by specifying the system registers number consisting of register number and selection identifier by LDSR instruction and STSR instruction. Some registers are treated as an undefined register when HVE = 0. The initial value of such a register after reset release with HVE = 0 is not guaranteed. When HVE is changed from 0 to 1, write an appropriate value to the register.

Table 3.20 List of virtualization support function system registers

Register Number (regID, selID)	Name	Function	Access Authority*1	
			HVE = 0	HVE = 1
SR16, 1	HVCFG	Setting virtualization support function	SV	HV
SR17, 1	GMCFG	Setting Guest mode operation	SV	HV
SR15, 0	PSWH	Program status extended words	SV	HV*2
SR18, 0	EIPSWH	PSWH saving register when EI level exception is accepted	SV	HV
SR19, 0	FEPSWH	PSWH saving register when FE level exception is accepted	SV	HV
SR20, 1	HVSB	Information notification register to Guest mode	SV	HV*2
SR0, 3	DBGEN	Control of debugging with Host mode software	SV	HV

Note 1. The access authority may be different depending on the value of HVCFG.HVE.

Note 2. In Host mode, reading with SV privilege or UM authority is possible. In Guest mode, only reading with SV privilege or UM authority is possible. In Guest mode, writing is disabled.

(1) HVCFG

This register controls the CPU virtualization support function.

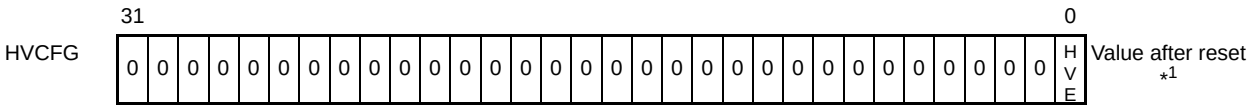


Table 3.21 HVCFG Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 1	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
0	HVE	Enables or disables the virtualization support function. *2 0: Virtualization support function is disabled 1: Virtualization support function is enabled When this bit is cleared (0), virtualization support function can not be used. And depending on the setting of this bit, the execution authority of the instruction and the access authority of the system registers may change. For details, see **Section 1, Virtualization support function**. When PID.HV is cleared (0) and the virtualization support function is not implemented, this bit is always cleared (0) and can not be set (1).	R/W	*1

Note 1. For details, see the hardware manual of the product used.

Note 2. Although the value of this bit can be changed by software, the value of this bit greatly affects CPU operation such as state management, access authority judgment, interrupt handling and so on. Therefore, when the value of this bit needs to be changed from the initial value, the change must be made before the settings of other functions are changed, at the beginning of the CPU initialization process after releasing reset. Do not change the value of this bit during user program operation.

(2) GMCFG

This register sets the operating mode of Guest mode. This register is enabled when HVCFG.HVE is set (1). The setting of this register affects only the operation in Guest mode. Setting of this register has no effect on operation in Host mode. When HVCFG.HVE is cleared (0), this register is treated as an undefined register. In that case, the value of this register is not automatically cleared (0), but the setting value does not affect the operation of the CPU.

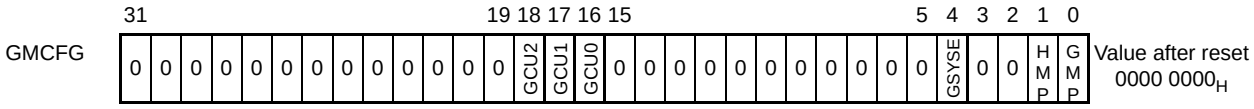


Table 3.22 GMCFG Register Contents (1/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 19	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
18	GCU2	Specifies whether GMPSW.CU2 can be changed. 0: GMPSW.CU2 can not be changed. 1: GMPSW.CU2 can be changed. However, this bit is a reserved function for the future CPUs compatible with this CPU. The value of this bit cannot be changed. In this CPU, the value of GMPSW.CU2 is fixed to (0).	R	0
17	GCU1	Specify availability of changing GMPSW.CU1. 0: GMPSW.CU1 can not be changed. (CU1 is automatically cleared (0).) 1: GMPSW.CU1 can be changed. When this bit is set (1) and while GMPSW.CU1 is set (1), if this bit is cleared (0), the value of GMPSW.CU1 is cleared (0). When this bit is cleared (0), since GMPSW.CU1 is always cleared (0), extended floating point arithmetic function (FXU) can not be used at all in Guest mode.	R/W	0
16	GCU0	Specify availability of changing GMPSW.CU0. 0: GMPSW.CU0 can not be changed. (CU0 is automatically cleared (0).) 1: GMPSW.CU0 can be changed. When this bit is set (1) and while GMPSW.CU0 is set (1), if this bit is cleared (0), the value of GMPSW.CU0 is cleared (0). When this bit is cleared (0), since GMPSW.CU0 is always cleared (0), floating point arithmetic function (FPU) can not be used at all in Guest mode.	R/W	0
15 to 5	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
4	GSYSE	Specifies the operating mode of the SYSERR exception handler when the SYSERR exception occurs in Guest mode. 0: Operation in Guest mode 1: Operation in Host mode If this bit is set (1), when the SYSERR exception occurs, the PSWH.GM bit is automatically cleared (0), and the mode changes to Host mode.	R/W	0
3, 2	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
1	HMP	Specifies the operating mode of the exception handler when the guest management entry is accessible but the host management entry is not accessible and the MPU exception occurs. 0: Operation in Guest mode 1: Operation in Host mode If this bit is set (1), when the target MPU exception occurs, the PSWH.GM bit is automatically cleared (0), and the mode changes to Host mode.	R/W	0

Table 3.22 GMCFG Register Contents (2/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
0	GMP	Specifies the operating mode of the exception handler if an access is not available in the guest management entry and the MPU exception occurs. 0: Operation in Guest mode 1: Operation in Host mode If this bit is set (1), when the target MPU exception occurs, the PSWH.GM bit is automatically cleared (0), and the mode changes to Host mode.	R/W	0

(3) PSWH

When HVCFG.HVE is set (1), it becomes an extended register on the higher bit side of PSW (program status word).

The value of this register can not be changed with LDSR instruction. This register is updated by executing exception return instructions (EIRET, FERET, DBRET).

When HVCFG.HVE is cleared (0), this register is treated as an undefined register.

PSWH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GPID			0	0	0	0	0	0	0	0

Value after reset
0000 0000$_H$

Table 3.23 PSWH Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31	GM	Indicates the CPU operating mode when HVCFG.HVE is set (1). 0: Host mode 1: Guest mode The value of this bit can not be changed with the LDSR instruction. The value of this bit is cleared (0) by the occurrence of an exception that changes from Guest mode to Host mode. And the value of this bit is set (1) by execution of a return instruction that causes the change from Host mode to Guest mode. Note that when HVCFG.HVE is cleared (0), the value of this bit automatically becomes 0.	R	0
30 to 11	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
10 to 8	GPID	When HVCFG.HVE is set (1), it indicates Guest mode partition ID. Note that when HVCFG.HVE is cleared (0), the value of this bit automatically becomes 0.	R	0
7 to 0	—	(Reserved for future expansion. Be sure to set to 0.)	R	0

(4) EIPSWH

When HVCFG.HVE is set (1), accepting EI level exception saves the contents of PSWH at that time. And when the EIRET instruction is executed in Host mode, the value of this register is transferred to PSWH. But when the EIRET instruction is executed in Guest mode, the value of this register is not transferred to PSWH.

When HVCFG.HVE is cleared (0), this register is treated as an undefined register. In this case, the value of this register is not automatically cleared (0). And even if the EIRET instruction is executed, the value of this register is not transferred to PSWH.

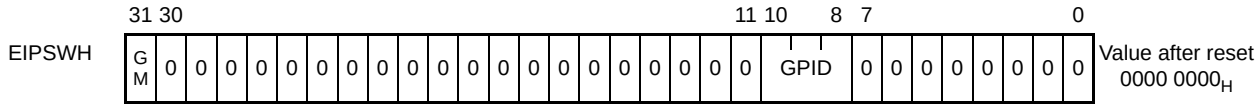


Table 3.24 EIPSWH Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31	GM	When EI level exception is accepted, the PSWH.GM bit is saved. While this bit is set (1) in Host mode, if the EIRET instruction is executed, it changes to Guest mode after returning.	R/W	0
30 to 11	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
10 to 8	GPID	When EI level exception is accepted, the PSWH.GPID is saved.	R/W	0
7 to 0	—	(Reserved for future expansion. Be sure to set to 0.)	R	0

(5) FEPSWH

When HVCFG.HVE is set (1), accepting FE level exception saves the contents of PSWH at that time. And when the FERET instruction is executed in Host mode, the value of this register is transferred to PSWH. But when the FERET instruction is executed in Guest mode, the value of this register is not transferred to PSWH.

When HVCFG.HVE is cleared (0), this register is treated as an undefined register. In this case, the value of this register is not automatically cleared (0). And even if the FERET instruction is executed, the value of this register is not transferred to PSWH.

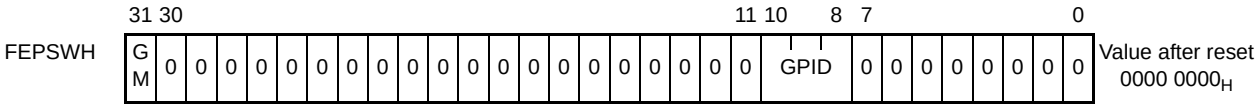


Table 3.25 FEPSWH Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31	GM	When FE level exception is accepted, the PSWH.GM bit is saved. While this bit is set (1) in Host mode, if the FERET instruction is executed, it changes to Guest mode after returning.	R/W	0
30 to 11	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
10 to 8	GPID	When FE level exception is accepted, the PSWH.GPID is saved.	R/W	0
7 to 0	—	(Reserved for future expansion. Be sure to set to 0.)	R	0

(6) HVSB

When HVCFG.HVE is set (1), this register notifies information to Guest mode. Host mode management software can use this register to notify any information such as software state of Host mode.

This register is used only for notification of information on software. And the hardware behavior will not change depending on the value set in this register. In Guest mode, it only can be read.

When HVCFG.HVE is cleared (0), this register is treated as an undefined register.

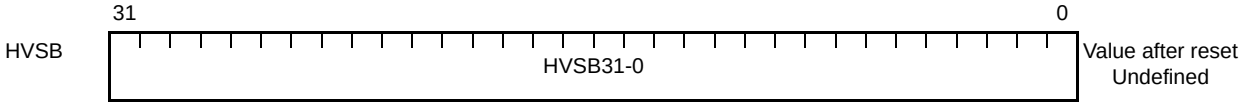


Table 3.26 HVSB Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	HVSB	Notifies any information to Guest mode.	R/W	Undefined

(7) DBGEN

When HVCFG.HVE is set (1), this register controls event detection with breakpoint setting. Although this register is a system register related to the debugging function, it can be accessed using the LDSR and STSR instructions without debugging authority.

When HVCFG.HVE is cleared (0), this register is treated as an undefined register. In this case, the setting of this register is not automatically cleared (0) and the value of this register does not affect event detection with breakpoint setting.

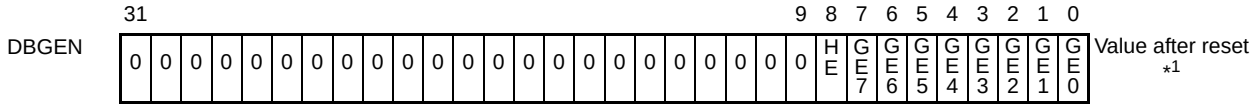


Table 3.27 DBGEN Register Contents (1/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 9	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
8	HE	In Host mode, indicates whether or not event detection is allowed. 0: In Host mode, event detection is not allowed. 1: In Host mode, event detection is allowed.	R/W	*1
7	GE7	In Guest mode, indicates whether or not event detection is allowed when PSWH.GPID = 7. 0: When GPID = 7, event detection is not allowed. 1: When GPID = 7, event detection is allowed.	R/W	*1
6	GE6	In Guest mode, indicates whether or not event detection is allowed when PSWH.GPID = 6. 0: When GPID = 6, event detection is not allowed. 1: When GPID = 6, event detection is allowed.	R/W	*1
5	GE5	In Guest mode, indicates whether or not event detection is allowed when PSWH.GPID = 5. 0: When GPID = 5, event detection is not allowed. 1: When GPID = 5, event detection is allowed.	R/W	*1

Table 3.27 DBGEN Register Contents (2/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
4	GE4	In Guest mode, indicates whether or not event detection is allowed when PSWH.GPID = 4. 0: When GPID = 4, event detection is not allowed. 1: When GPID = 4, event detection is allowed.	R/W	*1
3	GE3	In Guest mode, indicates whether or not event detection is allowed when PSWH.GPID = 3. 0: When GPID = 3, event detection is not allowed. 1: When GPID = 3, event detection is allowed.	R/W	*1
2	GE2	In Guest mode, indicates whether or not event detection is allowed when PSWH.GPID = 2. 0: When GPID = 2, event detection is not allowed. 1: When GPID = 2, event detection is allowed.	R/W	*1
1	GE1	In Guest mode, indicates whether or not event detection is allowed when PSWH.GPID = 1. 0: When GPID = 1, event detection is not allowed. 1: When GPID = 1, event detection is allowed.	R/W	*1
0	GE0	In Guest mode, indicates whether or not event detection is allowed when PSWH.GPID = 0. 0: When GPID = 0, event detection is not allowed. 1: When GPID = 0, event detection is allowed.	R/W	*1

Note 1. For details, see the hardware manual of the product used.

3.11 Host Context Register

The host context registers are the system registers for Host mode that is multiplexed with the guest context to realize virtualization support function. For details, see **Section 2.5.7, System Register Multiplexing**.

The access authority when accessing the host context registers is the same as that of the system register of the original function definition when HVCFG.HVE is set (1). For details of the access authority, see **Section 3.2, Basic System Registers**, **Section 3.3, Interrupt Function Registers**, and **Section 3.6, MPU Function Registers**.

Note that the host context registers are enabled only when HVCFG.HVE is set (1). When HVCFG.HVE is cleared (0), the system registers are not multiplexed and become the register model in the conventional mode. For details of the change of register model accompanying virtualization, see **Section 2.5.6, Change in Register Model**.

(1) HMEIPC

When EI level exception handled in Host mode is acknowledged, the address of the instruction that was being executed when the EI level exception occurred, or of the next instruction, is saved to the HMEIPC register (see **Section 4.1.3, Types of Exceptions**).

Because there is only one pair of EI level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

Be sure to set an even-numbered address to the HMEIPC register. An odd-numbered address must not be specified.

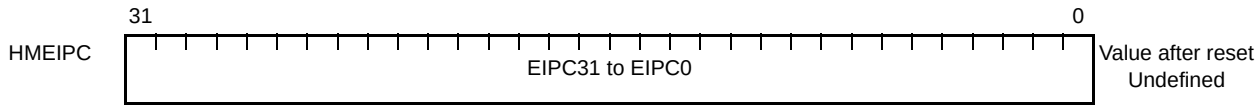


Table 3.28 HMEIPC Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 1	EIPC31 to EIPC1	These bits indicate the PC saved when an EI level exception is acknowledged.	R/W	Undefined
0	EIPC0	This bit indicates the PC saved when an EI level exception is acknowledged. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the EIRET instruction is executed is 0.	R/W	Undefined

(2) HMEIPSW — Status Save Register when Acknowledging EI Level Exception

When EI level exception handled in Host mode is acknowledged, the current PSW setting is saved to the HMEIPSW register.

Because there is only one pair of EI level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

HMEIPSW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	UM	0	0	0	0	EIMASK						0	CU2 to CU0			EBV	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z

Value after reset: 0000 0020$_H$

Table 3.29 HMEIPSW Register Contents

Bit Position	Bit Name	Description	R/W	Value after Reset
31	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
30	UM	This bit stores the HMPSW.UM bit setting when an EI level exception is acknowledged.	R/W	0
29 to 26	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
25 to 20	EIMASK	These bits store the HMPSW.EIMASK bit setting when an EI level exception is acknowledged*1.	R/W	0
19	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
18 to 16	CU2 to CU0	These bits store the HMPSW.CU2-0 field setting when an EI level exception is acknowledged*2.	R/W	0
15	EBV	This bit stores the HMPSW.EBV bit setting when an EI level exception is acknowledged.	R/W	0
14 to 8	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
7	NP	This bit stores the HMPSW.NP bit setting when an EI level exception is acknowledged.	R/W	0
6	EP	This bit stores the HMPSW.EP bit setting when an EI level exception is acknowledged.	R/W	0
5	ID	This bit stores the HMPSW.ID bit setting when an EI level exception is acknowledged.	R/W	1
4	SAT	This bit stores the HMPSW.SAT bit setting when an EI level exception is acknowledged.	R/W	0
3	CY	This bit stores the HMPSW.CY bit setting when an EI level exception is acknowledged.	R/W	0
2	OV	This bit stores the HMPSW.OV bit setting when an EI level exception is acknowledged.	R/W	0
1	S	This bit stores the HMPSW.S bit setting when an EI level exception is acknowledged.	R/W	0
0	Z	This bit stores the HMPSW.Z bit setting when an EI level exception is acknowledged.	R/W	0

Note 1. Only if HMINTCFG.EPL is set to 1, the value other than 0 can be set in this field. If INTCFG.EPL is cleared to 0, the value of this field becomes 0. Note that If HMINTCFG.EPL is cleared to 0 when the value of this field is other than 0, the value of this field becomes 0.

Note 2. CU2 is reserved for future CPUs that are to be made compatible with this CPU. It is always set to 0 in this CPU.

(3) HMFEPC

When FE level exception handled in Host mode is acknowledged, the address of the instruction that was being executed when the FE level exception occurred, or of the next instruction, is saved to the HMFEPC register (see **Section 4.1.3, Types of Exceptions**).

Because there is only one pair of FE level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

Be sure to set an even-numbered address to the HMFEPC register. An odd-numbered address must not be specified.

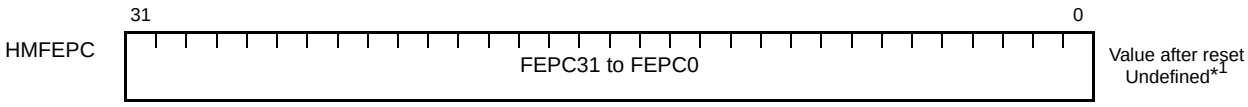


Table 3.30 HMFEPC Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 1	FEPC31 to FEPC1	These bits indicate the PC saved when an FE level exception is acknowledged.	R/W	Undefined*1
0	FEPC0	This bit indicates the PC saved when an FE level exception is acknowledged. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the FERET instruction is executed is 0.	R/W	Undefined*1

Note 1. When a reset occurs, among the instructions that completed execution before the reset occurred the value of the program counter of the instruction that was executed last is saved. If there is no execution completed instruction before the reset occurs, the value after reset is undefined. There is no information that identifies whether the value after reset is the value of the program counter of the execution completed instruction or undefined.

(4) HMFEPSW

When FE level exception handled in Host mode is accepted, the current PSW setting is saved to the HMFEPSW register.

Because there is only one pair of FE level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

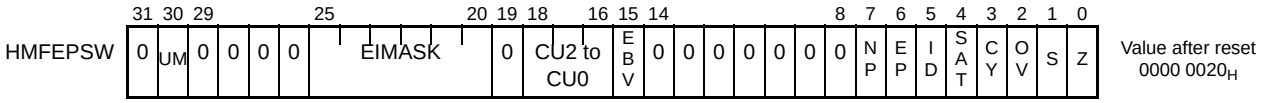


Table 3.31 HMFEPSW Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
30	UM	This bit stores the HMPSW.UM bit setting when an FE level exception is acknowledged.	R/W	0
29 to 26	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
25 to 20	EIMASK	These bits store the HMPSW.EIMASK field setting when an FE level exception is acknowledged*1.	R/W	0
19	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
18 to 16	CU2 to CU0	These bits store the HMPSW.CU2-0 field setting when an FE level exception is acknowledged*2.	R/W	0
15	EBV	This bit stores the HMPSW.EBV bit setting when an FE level exception is acknowledged.	R/W	0
14 to 8	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
7	NP	This bit stores the HMPSW.NP bit setting when an FE level exception is acknowledged.	R/W	0
6	EP	This bit stores the HMPSW.EP bit setting when an FE level exception is acknowledged.	R/W	0
5	ID	This bit stores the HMPSW.ID bit setting when an FE level exception is acknowledged.	R/W	1
4	SAT	This bit stores the HMPSW.SAT bit setting when an FE level exception is acknowledged.	R/W	0
3	CY	This bit stores the HMPSW.CY bit setting when an FE level exception is acknowledged.	R/W	0
2	OV	This bit stores the HMPSW.OV bit setting when an FE level exception is acknowledged.	R/W	0
1	S	This bit stores the HMPSW.S bit setting when an FE level exception is acknowledged.	R/W	0
0	Z	This bit stores the HMPSW.Z bit setting when an FE level exception is acknowledged.	R/W	0

Note 1. Only if HMINTCFG.EPL is set to 1, the value other than 0 can be set in this field. If HMINTCFG.EPL is cleared to 0, the value of this field becomes 0. Note that If HMINTCFG.EPL is cleared to 0 when the value of this field is other than 0, the value of this field becomes 0.

Note 2. CU2 is reserved for future CPUs that are to be made compatible with this CPU. It is always set to 0 in this CPU.

(5) HMPSW

The HMPSW register is PSW (Program Status Word) used for Host mode.

PSW (program status word) is a set of flags that indicate the program status (instruction execution result) and bits that indicate the operation status of the CPU (flags are bits in the PSW that are referenced by a condition instruction (Bcond, CMOV, etc.)).

CAUTIONS

1. **When the LDSR instruction is used to change the contents of this register, the changed contents become valid from the subsequent instruction. For details, see Section 7.3, Hazard Management after System Register Update.**
2. **The access permission for the PSW register differs depending on the bit. All bits can be read, but some bits can only be written under certain conditions. See Table 3.32 for the access permission for each bit.**

Table 3.32 Access Permission for HMPSW Register

Bit		Access Permission when Reading		Access Permission when Writing	
		HVE = 0	HVE = 1	HVE = 0	HVE = 1
30	UM	UM	UM	SV*1	SV*1
25 to 20	EIMASK			SV*1	SV*1
18 to 16	CU2-0			SV*1	SV*1
15	EBV			SV*1	SV*1
7	NP			SV*1	SV*1
6	EP			SV*1	SV*1
5	ID			SV*1	SV*1
4	SAT			UM	UM
3	CY			UM	UM
2	OV			UM	UM
1	S			UM	UM
0	Z			UM	UM

Note 1. The access permission for the whole HMPSW register is UM, so the PIE exception does not occur even if the register is written by using an LDSR instruction when HMPSW.UM is 1. In this case, writing is ignored.

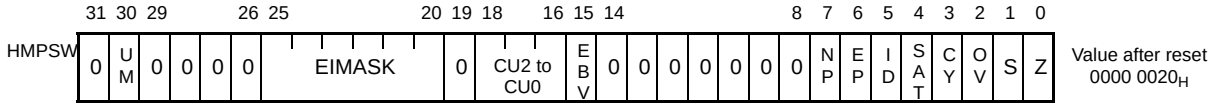


Table 3.33 HMPSW Register Contents (1/2)

Bit Position	Bit Name	Description	R/W	Value after Reset
31	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
30	UM	This bit indicates that the CPU is in user mode (in UM mode). 0: Supervisor mode 1: User mode	R/W	0
29 to 26	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
25 to 20	EIMASK	This field indicates the interrupt priority that becomes the boundary between enabling and disabling an acknowledgement of an interrupt (EIINTn). For an interrupt (EIINTn) with higher priority than the value set in this field, the acknowledgement is enabled. For an interrupt (EIINTn) with priority less than or equal to the value set in this field, the acknowledgement is disabled. 0: Interrupt acceptance of all priority levels is disabled. 1: Interrupt acceptance of priority level 1 or lower is disabled. (0 is enabled) 2: Interrupt acceptance of priority level 2 or lower is disabled. (1 or more are enabled) … 62: Interrupt acceptance of priority level 62 or lower is disabled. (61 or higher is enabled) 63: Interrupt acceptance of priority level 63 or lower is disabled. (62 or more are enabled) Only if the HMINTCFG.EPL is set to 1, interrupt acknowledgment control is performed by the value of this field. If the HMINTCFG.EPL is cleared to 0, interrupt acknowledgment control by the value of this field is not performed*1. If the HMINTCFG.EPL is set to 1 and an interrupt (EIINTn) is acknowledged, the interrupt priority is saved in this field as part of HMPSW change due to acknowledgment of interrupt (EIINTn). The interrupt (EIINTn) with a priority of 63 is always disabled. However, the CPU halt state due to HALT or SNOOZE is released by an interrupt (EIINTn) with priority 63.	R/W	0
19	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
18 to 16	CU2 to CU0	These bits indicate the coprocessor use permissions. When the bit corresponding to the coprocessor is 0, a coprocessor unusable exception occurs if an instruction for the coprocessor is executed or a coprocessor resource (system register) is accessed. Bit 18 (CU2): Fixed to 0*2. Bit 17 (CU1): FXU Bit 16 (CU0): FPU CU2 to CU0 are fixed to 0 in the devices that do not have corresponding coprocessors.	R/W	000
15	EBV	This bit indicates the reset vector and exception vector operation. See **Section 3.2 (16), RBASE — Reset Vector Base Address** and **Section 3.2 (17), EBASE — Exception Handler Vector Address.**	R/W	0
14 to 8	—	(Reserved for future expansion. Be sure to set to 0.)	R	0

Table 3.33 HMPSW Register Contents (2/2)

Bit Position	Bit Name	Description	R/W	Value after Reset
7	NP	This bit disables the acknowledgement of FE level exception. When an FE level exception is acknowledged, this bit is set to 1 to disable the acknowledgement of EI level and FE level exceptions. As for the exceptions for which the NP bit disables the acknowledgment, see **Table 4.1, Exception Cause List**. 0: The acknowledgement of FE level exception is enabled. 1: The acknowledgement of FE level exception is disabled.	R/W	0
6	EP	This bit indicates that an exception other than an interrupt is being serviced. It is set to 1 when the corresponding exception occurs. This bit does not affect acknowledging an exception request even when it is set to 1. 0: An exception other than an interrupt is not being serviced. 1: An exception other than an interrupt is being serviced.	R/W	0
5	ID	This bit disables the acknowledgement of EI level exception. When an EI level or FE level exception is acknowledged, this bit is set to 1 to disable the acknowledgement of EI level exception. As for the exceptions for which the ID bit disables the acknowledgment, see **Table 4.1, Exception Cause List** This bit is also used to disable EI level exceptions from being acknowledged as a critical section while an ordinary program or interrupt is being serviced. It is set to 1 when the DI instruction is executed, and cleared to 0 when the EI instruction is executed. The change of the ID bit by the EI or ID instruction will be enabled from the next instruction. 0: The acknowledgement of EI level exception is enabled. 1: The acknowledgement of EI level exception is disabled.	R/W	1
4	SAT*3	This bit indicates that a saturation arithmetic operation instruction resulted in overflow and saturation processing is applied to the result. This is a cumulative flag, that is, it is set (1) once a saturation occurs and not cleared (0) by subsequent instructions with unsaturated results. This bit is cleared by the LDSR instruction. Note that execution of an arithmetic operation instruction neither sets nor clears this flag. 0: The result was not saturated 1: The result was saturated	R/W	0
3	CY	This bit indicates whether a carry or borrow has occurred in the operation result. 0: Carry and borrow have not occurred. 1: Carry or borrow has occurred.	R/W	0
2	OV*3	This bit indicates whether or not an overflow has occurred during an operation. 0: Overflow has not occurred. 1: Overflow has occurred.	R/W	0
1	S*3	This bit indicates whether or not the result of an operation is negative. 0: Result of operation is positive or 0. 1: Result of operation is negative.	R/W	0
0	Z	This bit indicates whether or not the result of an operation is 0. 0: Result of operation is not 0. 1: Result of operation is 0.	R/W	0

Note 1. Only if HMINTCFG.EPL is set to 1, the value other than 0 can be set in this field. If HMINTCFG.EPL is cleared to 0, the value of this field becomes 0. If HMINTCFG.EPL is cleared to 0, since the interrupt acknowledgment controlled by the value of this field is not performed, even if the value of this field is 0, if another interrupt acknowledgment condition is satisfied, an interrupt (EIINTn) can be acknowledged. Note that If HMINTCFG.EPL is cleared to 0 when the value of this field is other than 0, the value of this field becomes 0.

Note 2. The coprocessor use permission CU2 is reserved for future CPUs that are to be made compatible with this CPU.

Note 3. Saturation processing is applied to the operation result in accordance with the contents of the OV and S flags. The SAT flag is set (1) only when the OV flag is set (1) in the saturation arithmetic operation.

(6) HMEIIC

When an EI-level exception handled in Host mode occurs, the HMEIIC register stores the cause of the exception. The value retained in this register is an exception code corresponding to a specific exception cause (see **Table 4.1, Exception Cause List**).

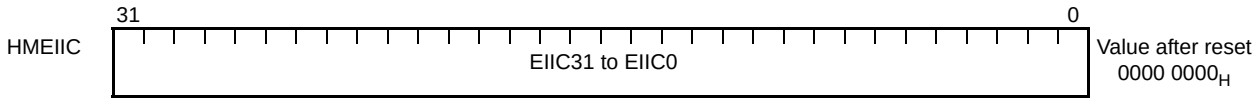


Table 3.34 HMEIIC Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	EIIC31 to EIIC0	These bits store the exception cause code when an EI level exception occurs. The EIIC15-0 field stores the exception cause codes shown in **Table 4.1**. The EIIC31-16 field stores detailed exception cause codes defined individually for each exception. If there is no particular definition, these bits are set to 0.	R/W	0

(7) HMFEIC

When a FE-level exception handled in Host mode occurs, the HMFEIC register stores the cause of the exception. The value retained in this register is an exception code corresponding to a specific exception cause (see **Table 4.1, Exception Cause List**).

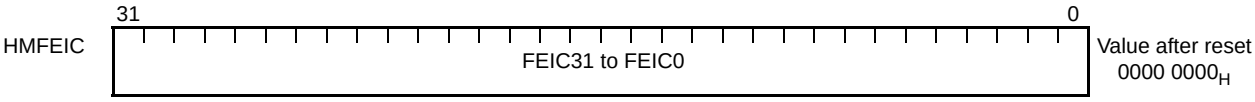


Table 3.35 HMEIIC Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	FEIC31 to FEIC0	These bits store the exception cause code when an FE level exception occurs. The FEIC15-0 field stores the exception cause codes shown in **Table 4.1**. The FEIC31-16 field stores detailed exception cause codes defined individually for each exception. If there is no particular definition, these bits are set to 0.	R/W	0

(8) HMEIWR

The HMEIWR register is the working register for when EI level exception occurred in Host mode.

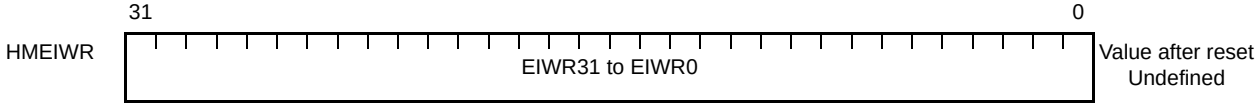


Table 3.36 EIWR Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	EIWR31 to EIWR0	These bits constitute a working register that can be used for any purpose during the processing of an EI level exception. Use this register for purposes such as storing the values of general-purpose registers.	R/W	Undefined

(9) HMFEWR

The HMFEWR register is the working register for when FE level exception occurred in Host mode.

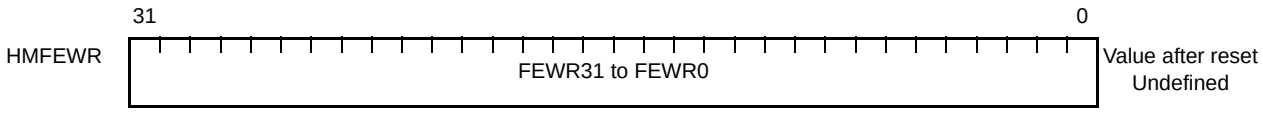


Table 3.37 HMFEWR Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	FEWR31 to FEWR0	These bits constitute a working register that can be used for any purpose during the processing of an FE level exception. Use this register for purposes such as storing the values of general-purpose registers.	R/W	Undefined

(10) HMSPID

The HMSPID register is a register that indicates the system protection number of CPU used in Host mode.

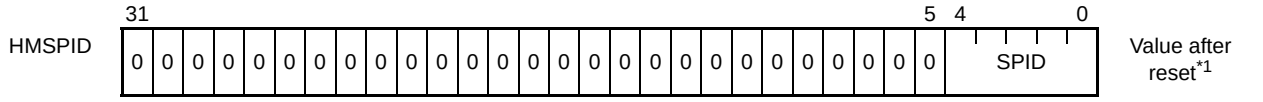


Table 3.38 HMSPID Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 5	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
4 to 0	SPID	These bits indicate the system protection identifier. The system protection identifier is a variable ID that is used for access protection in a product which consists of two or more bus masters including this CPU. For its uses and constraints on its value, see the hardware manual of the product used. Within this CPU, the SPID is used to check for area matching by the MPU. It allows the system specifications defined for the product to be reflected in the MPU's protection feature. The settable system protection identifiers are given by the HMSPIDLIST register. If an attempt is made to set an illegal system protection identifier, the HMSPID register is not updated and retains the original value.	R/W	*1

Note 1. For details, see the hardware manual of the product used.

(11) HMSPIDLIST

In Host mode, the HMSPIDLIST register contains a list of system protection identifiers that can be set to the HMSPID register.

The bits corresponding to the settable system protection identifiers are set to 1. The bits corresponding to illegal system protection identifiers are cleared to 0. These values are set outside the CPU as system specifications and cannot be altered by this CPU.

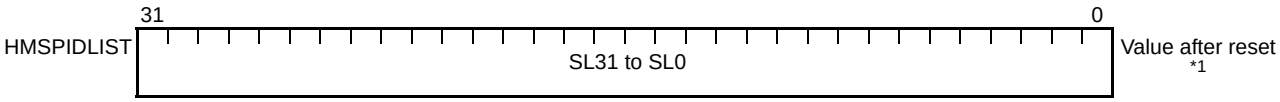


Table 3.39 HMSPIDLIST Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31	SL31	This bit indicates whether or not 31 can be set as a system protection identifier.	R	*1
30	SL30	This bit indicates whether or not 30 can be set as a system protection identifier.	R	*1
29	SL29	This bit indicates whether or not 29 can be set as a system protection identifier.	R	*1
28	SL28	This bit indicates whether or not 28 can be set as a system protection identifier.	R	*1
27	SL27	This bit indicates whether or not 27 can be set as a system protection identifier.	R	*1
26	SL26	This bit indicates whether or not 26 can be set as a system protection identifier.	R	*1
25	SL25	This bit indicates whether or not 25 can be set as a system protection identifier.	R	*1
24	SL24	This bit indicates whether or not 24 can be set as a system protection identifier.	R	*1
23	SL23	This bit indicates whether or not 23 can be set as a system protection identifier.	R	*1
22	SL22	This bit indicates whether or not 22 can be set as a system protection identifier.	R	*1
21	SL21	This bit indicates whether or not 21 can be set as a system protection identifier.	R	*1
20	SL20	This bit indicates whether or not 20 can be set as a system protection identifier.	R	*1
19	SL19	This bit indicates whether or not 19 can be set as a system protection identifier.	R	*1
18	SL18	This bit indicates whether or not 18 can be set as a system protection identifier.	R	*1
17	SL17	This bit indicates whether or not 17 can be set as a system protection identifier.	R	*1
16	SL16	This bit indicates whether or not 16 can be set as a system protection identifier.	R	*1
15	SL15	This bit indicates whether or not 15 can be set as a system protection identifier.	R	*1
14	SL14	This bit indicates whether or not 14 can be set as a system protection identifier.	R	*1
13	SL13	This bit indicates whether or not 13 can be set as a system protection identifier.	R	*1
12	SL12	This bit indicates whether or not 12 can be set as a system protection identifier.	R	*1
11	SL11	This bit indicates whether or not 11 can be set as a system protection identifier.	R	*1
10	SL10	This bit indicates whether or not 10 can be set as a system protection identifier.	R	*1
9	SL9	This bit indicates whether or not 9 can be set as a system protection identifier.	R	*1
8	SL8	This bit indicates whether or not 8 can be set as a system protection identifier.	R	*1
7	SL7	This bit indicates whether or not 7 can be set as a system protection identifier.	R	*1
6	SL6	This bit indicates whether or not 6 can be set as a system protection identifier.	R	*1
5	SL5	This bit indicates whether or not 5 can be set as a system protection identifier.	R	*1
4	SL4	This bit indicates whether or not 4 can be set as a system protection identifier.	R	*1
3	SL3	This bit indicates whether or not 3 can be set as a system protection identifier.	R	*1
2	SL2	This bit indicates whether or not 2 can be set as a system protection identifier.	R	*1
1	SL1	This bit indicates whether or not 1 can be set as a system protection identifier.	R	*1
0	SL0	This bit indicates whether or not 0 can be set as a system protection identifier.	R	*1

Note 1. For details, see the hardware manual of the product used.

(12) HMEBASE

In Host mode, the HMEBASE register indicates the vector address of the exception handler. The settings of this register are valid when the HMPSW.EBV bit is 1.

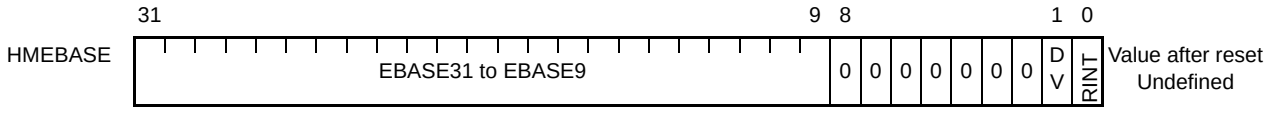


Table 3.40 HMEBASE Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 9	EBASE31 to EBASE9	The address of the exception handler routine is changed to the address calculated by adding the offset address of each exception to the base address specified in this register. The EBASE8 to EBASE0 bits are implicitly set to 0.	R/W	Undefined
8 to 2	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
1	DV	When the DV bit is set, the exception handler address for interrupt is determined by using the direct vector method. For details, see **Section 4.4.1 (2), Table Reference Method**.	R/W	Undefined
0	RINT	When the RINT bit is set, the exception handler address for interrupt processing is reduced. See **Section 4.4.1 (1), Direct Vector Method**.	R/W	Undefined

(13) HMINTBP

In Host mode, the HMINTBP register indicates the base address of the table when the table reference method is selected as the interrupt handler address selection method.

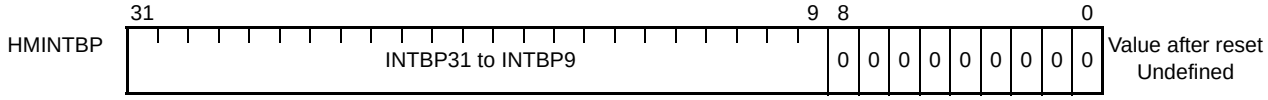


Table 3.41 HMINTBP Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 9	INTBP31 to INTBP9	These bits indicate the base pointer address for an interrupt when the table reference method is used. The value indicated by these bits is the first address in the table used to determine the exception handler when the interrupt specified by the table reference method (EIINT*n*) is acknowledged. The INTBP8 to INTBP0 bits are not assigned as names because these bits are always 0.	R/W	Undefined
8 to 0	—	(Reserved for future expansion. Be sure to set to 0.)	R	0

(14) HMINTCFG

In Host mode, the HMINTCFG register is used to specify settings related to the CPU's internal interrupt function.

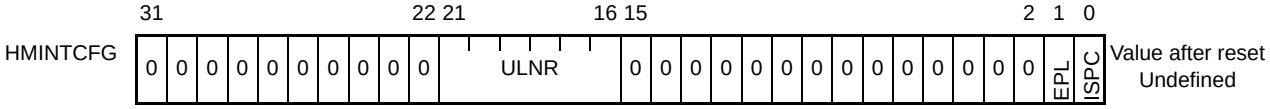


Table 3.42 HMINTCFG Register Contents (1/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 22	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
21 to 16	ULNR	Specifies the maximum value of available register bank numbers. If the value of the RBNR.BN is bigger than the ULNR, or the value of the RBNR.BN is 63, when the interrupt (EIINTn) using the register bank function occurs, the SYSERR exception will occur. Note that the interrupt (EIINTn) is not accepted and is put on hold.	R/W	$0F_H$
15 to 2	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
1	EPL	For the interrupt (EIINTn), specifies whether to enable interrupt priority level extension function. 0: Interrupt priority level extension function is disabled 1: Interrupt priority level extension function is enabled The following is the overview of interrupt operation by setting this bit. For details, see **Section 4.1.5, Interrupt Exception Priority and Priority Masking**, **Section 4.4, Exception Handler Address**, **Section 4.5, Register Bank Function**.	R/W	0

Interrupt function		Value of EPL = 0	Value of EPL = 1
Number of interrupt priority level		16	64
Mask Function	ISPR	Function is enabled	Function is disabled
	PSW.EIMASK	Function is disabled	Function is enabled
	PLMR	Function is enabled	
	PSW.NP	Function is enabled	
	PSW.ID	Function is enabled	
INTCFG.ISPC		Settings can be changed	Fixed 0
ICSR.PMEI		Function is enabled	
IMSR		Function is enabled	
Handler generation	Direct vector method	Each vector can be used by each priority.	Each vector can be used by priority 0 to 14. The priority 15 vector is shared with the priority 15 or greater.
	Table reference method	Available	
Register bank		Availability can be set by each priority.	Priority 0 to 14 can be individually set. The priority 15 setting is shared with the priority 15 or greater.

Table 3.42 HMINTCFG Register Contents (2/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
0	ISPC	This bit changes how the ISPR register is written. 0: The ISPR register is automatically updated. Updates triggered by the program (via execution of LDSR instruction) are ignored. 1: The ISPR register is not automatically updated. Updates triggered by the program (via execution of LDSR instruction) are performed. If this bit is cleared to 0, the bits of the ISPR register are automatically set to 1 when an interrupt (EIINT*n*) is acknowledged, and cleared to 0 when the EIRET instruction is executed. In this case, the bits are not updated by an LDSR instruction executed by the program. If this bit is set to 1, the bits of the ISPR register are not updated by the acknowledgement of an interrupt (EIINT*n*) or by execution of the EIRET instruction. In this case, the bits can be updated by an LDSR instruction executed by the program. In normal cases, the ISPC bit should be cleared. When performing software-based priority control, however, set this bit (1). For details, see **Section 4.1.5 (2), Interrupt Priority Mask**.	R/W	0

(15) HMPLMR

In Host mode, the HMPLMR register masks the interrupts (EIINTn) whose priority level is not higher than the level specified by these bits.

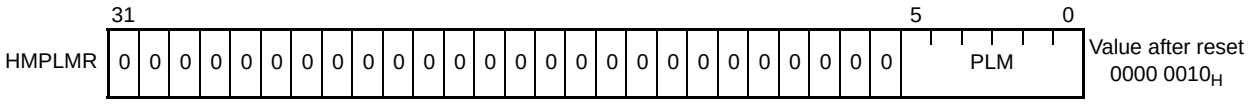


Table 3.43 HMPLMR Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 6	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
5 to 0	PLM	These bits are used to mask the interrupts (EIINT*n*) whose priority level is not higher than the level specified by these bits. When an interrupt (EIINT*n*) is masked by this register, it is not accepted. The correspondence between the value of the PLM bit and the highest priority of interrupts to be masked is shown below. **Value of PLM Bit / Highest Priority of Interrupts to be Masked** 0 — Priority 0 (All priorities are not acceptable) 1 — Priority 1 (Only priority 0 is acceptable) : 14 — Priority 14 (Only priorities 13 or higher are acceptable) 15 — Priority 15 (Only priorities 14 or higher are acceptable) : 62 — Priority 62 (Only priorities 61 or higher are acceptable) 63 — Priority 63 (Only priorities 62 or higher are acceptable) Since the highest priority level of an interrupt that is defined for this CPU is 0, if 0 is specified in the PLM bit, all interrupts (EIINT*n*) are masked by this register. Since the lowest interrupt priority level defined by this CPU is 63, the interrupts (EIINTn) with the interrupt priority of 63 are always masked. Regardless of INTCFG.EPL setting, the interrupt mask by PLMR is done by the value of the PLM bit and the value of the interrupt priority notified from the interrupt controller*1. If the value of PLMR is altered by the LDSR instruction, the new PLMR value is reflected in the instructions following that LDSR instruction.	R/W	10_H

Note 1. Even if INTCFG.EPL is cleared (0), it is possible to receive an interrupt (EIINTn) with priority 16 or less from the interrupt controller. Even if INTCFG.EPL is cleared (0), PLMR will not be automatically reduced to 4 bits, and PLMR always masks interrupts for interrupt priority level of up to 64 level.

(16) HMSVLOCK

In Host mode, the HMSVLOCK register is used to restrict the CPU operation in supervisor mode.

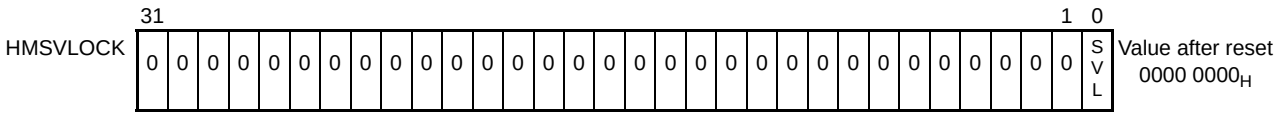


Table 3.44 HMSVLOCK Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 1	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
0	SVL	This bit specifies whether to restrict the CPU operation in supervisor mode. 0: Does not restrict the CPU operation in supervisor mode. 1: Restrict the CPU operation in supervisor mode. If the SVL bit is set to 1, the following system registers*1 cannot be updated even when the CPU is in supervisor mode: HMSPID, HMMPM, MPLA, MPUA, MPAT, MPIDn, MPBK For details, see **Section 2.5.5, Supervisor Lock Setting**.	R/W	0

Note 1. The target system registers are those registers that are associated with memory accessing. This register prevents these registers from being rewritten carelessly and unintentional memory access from being performed outside the CPU.

(17) HMMEA

The HMMEA register is MEA used for Host mode.

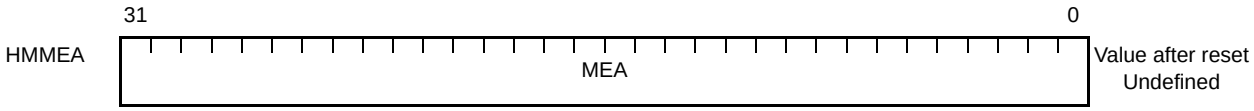


Table 3.45 HMMEA Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	MEA	These bits holds the address in which an MAE (misalignment) or MPU violation occurred.	R/W	Undefined

(18) HMMEI

In Host mode, the HMMEI register holds the information about the instruction that caused a misalignment exception (MAE) or memory protection exception (MDP). The information can be used as hint information for the emulation by software.

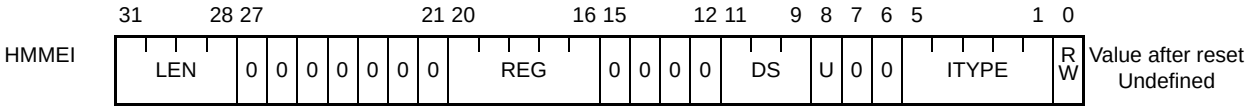


Table 3.46 HMMEI Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 28	LEN	These bits indicate the code size of the instruction that caused the exception. 0: Non-instruction factor 2: 16 bits 4: 32 bits 6: 48 bits 8:64 bits Values other than those listed above are reserved for future use and never stored here. For details, see **Table 3.47**.	R/W	Undefined
27 to 21	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
20 to 16	REG	These bits indicate the source register number or destination register number of the instruction that caused the exception. For details, see **Table 3.47**.	R/W	Undefined
15 to 12	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
11 to 9	DS	These bits indicate the data type of the instruction that caused the exception*1. 0: Byte (8 bits) 1: Halfword (16 bits) 2: Word (32 bits) 3: Double-word (64 bits) 4: Quad-word (128 bits) Values other than those listed above are reserved for future use and never stored here. For details, see **Table 3.47**.	R/W	Undefined
8	U	This bit indicates the sign extension method of the instruction that caused the exception. 0: Signed 1: Unsigned For details, see **Table 3.47**.	R/W	Undefined
7 to 6	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
5 to 1	ITYPE	These bits indicate the instruction that caused the exception. For details, see **Table 3.47**.	R/W	Undefined
0	RW	This bit indicates whether the operation performed by the instruction that caused the exception is a read (Load-memory) or a write (Store-memory). 0: Read (Load-memory) 1: Write (Store-memory) For details, see **Table 3.47**.	R/W	Undefined

Note 1. Even if the data is divided and access is made several times due to the specifications of the hardware, the original data type indicated by the instruction is stored.

Table 3.47 Instructions Causing Exceptions and Values of HMMEI Register (1/2)

Instruction	LEN	REG	DS	U	RW	ITYPE
SLD.B	2 (16 bits)	dst	0 (Byte)	0 (Signed)	0 (Read)	00000_B
SLD.BU	2 (16 bits)	dst	0 (Byte)	1 (Unsigned)	0 (Read)	00000_B
SLD.H	2 (16 bits)	dst	1 (Half-word)	0 (Signed)	0 (Read)	00000_B
SLD.HU	2 (16 bits)	dst	1 (Half-word)	1 (Unsigned)	0 (Read)	00000_B
SLD.W	2 (16 bits)	dst	2 (Word)	0 (Signed)	0 (Read)	00000_B
SST.B	2 (16 bits)	src	0 (Byte)	0 (Signed)	1 (Write)	00000_B
SST.H	2 (16 bits)	src	1 (Half-word)	0 (Signed)	1 (Write)	00000_B
SST.W	2 (16 bits)	src	2 (Word)	0 (Signed)	1 (Write)	00000_B
LD.B (disp16)	4 (32 bits)	dst	0 (Byte)	0 (Signed)	0 (Read)	00001_B
LD.BU (disp16)	4 (32 bits)	dst	0 (Byte)	1 (Unsigned)	0 (Read)	00001_B
LD.H (disp16)	4 (32 bits)	dst	1 (Half-word)	0 (Signed)	0 (Read)	00001_B
LD.HU (disp16)	4 (32 bits)	dst	1 (Half-word)	1 (Unsigned)	0 (Read)	00001_B
LD.W (disp16)	4 (32 bits)	dst	2 (Word)	0 (Signed)	0 (Read)	00001_B
ST.B (disp16)	4 (32 bits)	src	0 (Byte)	0 (Signed)	1 (Write)	00001_B
ST.H (disp16)	4 (32 bits)	src	1 (Half-word)	0 (Signed)	1 (Write)	00001_B
ST.W (disp16)	4 (32 bits)	src	2 (Word)	0 (Signed)	1 (Write)	00001_B
LD.B (disp23)	6 (48 bits)	dst	0 (Byte)	0 (Signed)	0 (Read)	00010_B
LD.BU (disp23)	6 (48 bits)	dst	0 (Byte)	1 (Unsigned)	0 (Read)	00010_B
LD.H (disp23)	6 (48 bits)	dst	1 (Half-word)	0 (Signed)	0 (Read)	00010_B
LD.HU (disp23)	6 (48 bits)	dst	1 (Half-word)	1 (Unsigned)	0 (Read)	00010_B
LD.W (disp23)	6 (48 bits)	dst	2 (Word)	0 (Signed)	0 (Read)	00010_B
LD.DW (disp23)	6 (48 bits)	dst	3 (Double-word)	0 (Signed)	0 (Read)	00010_B
ST.B (disp23)	6 (48 bits)	src	0 (Byte)	0 (Signed)	1 (Write)	00010_B
ST.H (disp23)	6 (48 bits)	src	1 (Half-word)	0 (Signed)	1 (Write)	00010_B
ST.W (disp23)	6 (48 bits)	src	2 (Word)	0 (Signed)	1 (Write)	00010_B
ST.DW (disp23)	6 (48 bits)	src	3 (Double-word)	0 (Signed)	1 (Write)	00010_B
LD.B (+)	4 (32 bits)	dst	0 (Byte)	0 (Signed)	0 (Read)	00100_B
LD.BU (+)	4 (32 bits)	dst	0 (Byte)	1 (Unsigned)	0 (Read)	00100_B
LD.H (+)	4 (32 bits)	dst	1 (Half-word)	0 (Signed)	0 (Read)	00100_B
LD.HU (+)	4 (32 bits)	dst	1 (Half-word)	1 (Unsigned)	0 (Read)	00100_B
LD.W (+)	4 (32 bits)	dst	2 (Word)	0 (Signed)	0 (Read)	00100_B
ST.B (+)	4 (32 bits)	src	0 (Byte)	0 (Signed)	1 (Write)	00100_B
ST.H (+)	4 (32 bits)	src	1 (Half-word)	0 (Signed)	1 (Write)	00100_B
ST.W (+)	4 (32 bits)	src	2 (Word)	0 (Signed)	1 (Write)	00100_B
LD.B (−)	4 (32 bits)	dst	0 (Byte)	0 (Signed)	0 (Read)	00101_B
LD.BU (−)	4 (32 bits)	dst	0 (Byte)	1 (Unsigned)	0 (Read)	00101_B
LD.H (−)	4 (32 bits)	dst	1 (Half-word)	0 (Signed)	0 (Read)	00101_B
LD.HU (−)	4 (32 bits)	dst	1 (Half-word)	1 (Unsigned)	0 (Read)	00101_B
LD.W (−)	4 (32 bits)	dst	2 (Word)	0 (Signed)	0 (Read)	00101_B
ST.B (−)	4 (32 bits)	src	0 (Byte)	0 (Signed)	1 (Write)	00101_B
ST.H (−)	4 (32 bits)	src	1 (Half-word)	0 (Signed)	1 (Write)	00101_B
ST.W (−)	4 (32 bits)	src	2 (Word)	0 (Signed)	1 (Write)	00101_B
LDL.BU	4 (32 bits)	dst	0 (Byte)	1 (Unsigned)	0 (Read)	00111_B
LDL.HU	4 (32 bits)	dst	1 (Half-word)	1 (Unsigned)	0 (Read)	00111_B

Table 3.47 Instructions Causing Exceptions and Values of HMMEI Register (2/2)

Instruction	LEN	REG	DS	U	RW	ITYPE
LDL.W	4 (32 bits)	dst	2 (Word)	0 (Signed)	0 (Read)	00111_B
STC.B	4 (32 bits)	src	0 (Byte)	0 (Signed)	1 (Write)	00111_B
STC.H	4 (32 bits)	src	1 (Half-word)	0 (Signed)	1 (Write)	00111_B
STC.W	4 (32 bits)	src	2 (Word)	0 (Signed)	1 (Write)	00111_B
CAXI	4 (32 bits)	dst*1	2 (Word)*1	0 (Signed)*1	0 (Read)*2	01000_B
SET1	4 (32 bits)	0*1	0 (Byte)*1	0 (Signed)*1	0 (Read)*2	01001_B
CLR1	4 (32 bits)	0*1	0 (Byte)*1	0 (Signed)*1	0 (Read)*2	01001_B
NOT1	4 (32 bits)	0*1	0 (Byte)*1	0 (Signed)*1	0 (Read)*2	01001_B
TST1	4 (32 bits)	0*1	0 (Byte)*1	0 (Signed)*1	0 (Read)	01001_B
PREPARE	*5,*6	src*1	2 (Word)*1	0 (Signed)*1	1 (Write)	01100_B
DISPOSE	4 (32 bits)	dst*1	2 (Word)*1	0 (Signed)*1	0 (Read)	01100_B
PUSHSP	4 (32 bits)	src*1	2 (Word)*1	0 (Signed)*1	1 (Write)	01101_B
POPSP	4 (32 bits)	dst*1,*3	2 (Word)*1	0 (Signed)*1	0 (Read)	01101_B
STM.GSR	4 (32 bits)	0*1	2 (Word)*1	0 (Signed)*1	1 (Write)	01110_B
LDM.GSR	4 (32 bits)	0*1	2 (Word)*1	0 (Signed)*1	0 (Read)	01110_B
STM.MP	4 (32 bits)	0*1	2 (Word)*1	0 (Signed)*1	1 (Write)	01111_B
LDM.MP	4 (32 bits)	0*1	2 (Word)*1	0 (Signed)*1	0 (Read)	01111_B
SWITCH	2 (16 bits)	0*1	1 (Half-word)*1	0 (Signed)*1	0 (Read)	10000_B
CALLT	2 (16 bits)	0*1	1 (Half-word)*1	1 (Unsigned)*1	0 (Read)	10001_B
SYSCALL	4 (32 bits)	0*1	2 (Word)*1	0 (Signed)*1	0 (Read)	10010_B
CACHE	4 (32 bits)	0*1	0 (Byte)*1	0 (Signed)*1	0 (Read)	10100_B
Interrupt (table reference method)*4	0 (Non-instruction)	0*1	2 (Word)*1	0 (Signed)*1	0 (Read)	10101_B
Save onto register bank	0 (Non-instruction)	0*1	2 (Word)*1	0 (Signed)*1	1 (Write)	10110_B
RESBANK	4 (32 bits)	0*1	2 (Word)*1	0 (Signed)*1	0 (Read)	10110_B
LDV.W (disp16)	6 (48 bits)	dst	2 (Word)	0 (Signed)	0 (Read)	11101_B
LDV.DW (disp16)	6 (48 bits)	dst	3 (Double-word)	0 (Signed)	0 (Read)	11101_B
LDV.QW (disp16)	6 (48 bits)	dst	4 (Quad-word)	0 (Signed)	0 (Read)	11101_B
STV.W (disp16)	6 (48 bits)	src	2 (Word)	0 (Signed)	1 (Write)	11101_B
STV.DW (disp16)	6 (48 bits)	src	3 (Double-word)	0 (Signed)	1 (Write)	11101_B
STV.QW (disp16)	6 (48 bits)	src	4 (Quad-word)	0 (Signed)	1 (Write)	11101_B
LDVZ.H4 (disp16)	6 (48 bits)	dst	3 (Double-word)	0 (Signed)	0 (Read)	11111_B
STVZ.H4 (disp16)	6 (48 bits)	src	3 (Double-word)	0 (Signed)	1 (Write)	11111_B

Note: dst: Destination register number; src: Source register number

Note 1. Specific to this CPU.

Note 2. This exception occurs when the instruction executes a read access.

Note 3. When the destination is r3, 0 is stored.

Note 4. When an exception occurs during a table reference that is triggered by an interrupt for which the table reference method is selected,

Note 5. In instruction format (1), “4 (32 bits)” is set.

Note 6. In instruction format (2), the value is selected as follows according to the value of the ff field in the instruction code:

ff = 00_B: 4 (32 bits)
ff = 01_B: 6 (48 bits)
ff = 10_B: 6 (48 bits)
ff = 11_B: 8 (64 bits)

(19) HMMPM

The HMMPM register is a register that determines the MPU operation state in Host mode.

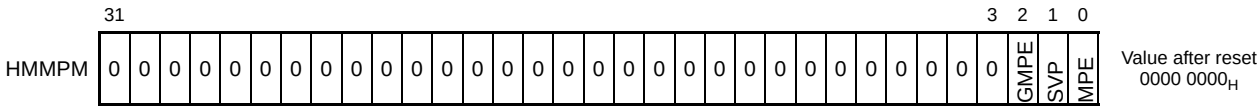


Table 3.48 HMMPM Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 3	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
2	GMPE	It is disabled in Host mode because it is a function for Guest mode. In Host mode, this bit can not be set.	R	0
1	SVP	In SV mode (HMPSW.UM = 0) in Host mode, enables or disables the memory protection function by host management entry. 0: Memory protection function is disabled in SV mode. 1: Memory protection function is enabled in SV mode. However, in order for the memory protection function in SV mode to be enabled, the MPE bit must be set (1).	R/W	0
0	MPE	Enables or disables the memory protection function by host management entry in Host mode. 0: Memory protection setting is disabled in Host mode. 1: Memory protection setting is enabled in Host mode. When setting the value of this bit to 1, be sure to set at least one host management entry.	R/W	0

(20) HMPEID

The HMPEID register is a register that indicates the processor element number in Host mode.

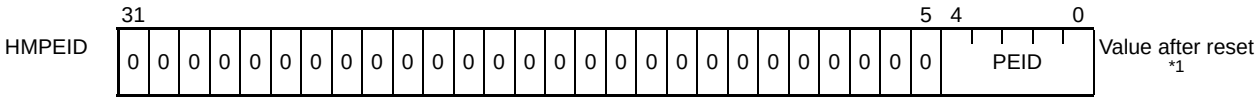


Table 3.49 HMPEID Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 5	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
4 to 0	PEID	These bits indicate the processor element identifier.	R	*1

Note 1. The processor element identifier of the CPU which is defined by the product specification is read. Writing to these bits is not possible. For details of the value after reset of HMPEID register, see the hardware manual of the product used.

3.12 Guest Context Register

The guest context registers are the system registers used for Guest mode and multiplexed with the host context to realize the virtualization support function. For details, see **Section 2.5.7, System Register Multiplexing**.

Table 3.50 shows the access authority for directly accessing the guest context. In this case, 9 is specified as the selection identifier (selID). In this way all become HV privilege, so the guest context can not be directly accessed in Guest mode. In Guest mode, the access authority when accessing the system register of the conventional function definition corresponding to the guest context is the access authority originally defined in each system registers.

Note that the guest context registers are enabled only when HVCFG.HVE is set (1). When HVCFG.HVE is cleared (0), it is treated as an undefined register. The initial values of these registers are not guaranteed after reset release with HVE = 0. When HVE is changed from 0 to 1, write an appropriate value to each of these registers.

Table 3.50 List of Guest Context Register

Register Number (regID, selID)	Name	Function	Access authority	
			HVE = 0	HVE = 1
SR0, 9	GMEIPC	EIPC register used for Guest mode	SV	HV
SR1, 9	GMEIPSW	EIPSW register used for Guest mode	SV	HV
SR2, 9	GMFEPC	FEPC register used for Guest mode	SV	HV
SR3, 9	GMFEPSW	FEPSW register used for Guest mode	SV	HV
SR5, 9	GMPSW	PSW register used for Guest mode	SV	HV
SR6, 9	GMMEA	MEA register used for Guest mode	SV	HV
SR8, 9	GMMEI	MEI register used for Guest mode	SV	HV
SR13, 9	GMEIIC	EIIC register used for Guest mode	SV	HV
SR14, 9	GMFEIC	FEIC register used for Guest mode	SV	HV
SR16, 9	GMSPID	SPID register used for Guest mode	SV	HV
SR17, 9	GMSPIDLIST	SPIDLIST register used for Guest mode	SV	HV
SR19, 9	GMEBASE	EBASE register used for Guest mode	SV	HV
SR20, 9	GMINTBP	INTBP register used for Guest mode	SV	HV
SR21, 9	GMINTCFG	INTCFG register used for Guest mode	SV	HV
SR22, 9	GMPLMR	PLMR register used for Guest mode	SV	HV
SR24, 9	GMSVLOCK	SVLOCK register used for Guest mode	SV	HV
SR25, 9	GMMPM	MPM register used for Guest mode	SV	HV
SR28, 9	GMEIWR	EIWR register used for Guest mode	SV	HV
SR29 9	GMFEWR	FEWR register used for Guest mode	SV	HV
SR30, 9	GMPEID	PEID register used for Guest mode	SV	HV

(1) GMEIPC

When EI level exception handled in Guest mode is accepted, the address of the instruction that was executing when the EI level exception occurred or the address of the next instruction are saved.

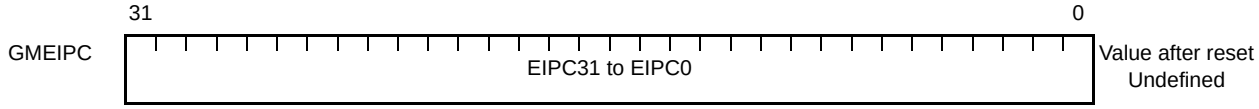


Table 3.51 GMEIPC Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 1	EIPC31 to EIPC1	These bits Indicate the returning PC when EI level exception is accepted.	R/W	Undefined
0	EIPC0	This bit Indicates the returning PC when EI level exception is accepted. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the EIRET instruction is executed is 0.	R/W	Undefined

(2) GMEIPSW

When EI level exception handled in Guest mode is accepted, the contents of PSW at that time are saved.

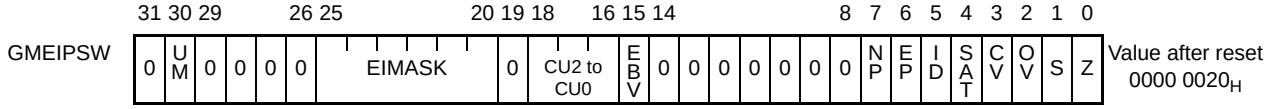


Table 3.52 GMEIPSW Register Contents (1/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
31	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
30	UM	This bit stores the GMPSW.UM bit setting when an EI level exception is acknowledged.	R/W	0
29 to 26	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
25 to 20	EIMASK	These bits store the GMPSW.EIMASK bits setting when an EI level exception is acknowledged.*1	R/W	0
19	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
18 to 16	CU2 to CU0	These bits store the GMPSW.CU2-0 bits setting when an EI level exception is acknowledged.*2	R/W	0
15	EBV	This bit stores the GMPSW.EBV bit setting when an EI level exception is acknowledged.	R/W	0
14 to 8	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
7	NP	This bit stores the GMPSW.NP bit setting when an EI level exception is acknowledged.	R/W	0
6	EP	This bit stores the GMPSW.EP bit setting when an EI level exception is acknowledged.	R/W	0
5	ID	This bit stores the GMPSW.ID bit setting when an EI level exception is acknowledged.	R/W	1
4	SAT	This bit stores the GMPSW.SAT bit setting when an EI level exception is acknowledged.	R/W	0
3	CY	This bit stores the GMPSW.CY bit setting when an EI level exception is acknowledged.	R/W	0

Table 3.52 GMEIPSW Register Contents (2/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
2	OV	This bit stores the GMPSW.OV bit setting when an EI level exception is acknowledged.	R/W	0
1	S	This bit stores the GMPSW.S bit setting when an EI level exception is acknowledged.	R/W	0
0	Z	This bit stores the GMPSW.Z bit setting when an EI level exception is acknowledged.	R/W	0

Note 1. Only if GMINTCFG.EPL is set (1), the value other than 0 can be set in this field. And if GMINTCFG.EPL is cleared (0), the value of this field becomes 0. Note that if GMINTCFG.EPL is cleared (0) when the value of this field is other than 0, the value of this field becomes 0.

Note 2. The CU2-0 field is a reservation function for future CPUs compatible with this CPU in the future. This field is fixed to 0 in this CPU.

(3) GMFEPC

When FE level exception handled in Guest mode is accepted, the address of the instruction that was executing when the FE level exception occurred or the address of the next instruction are saved.

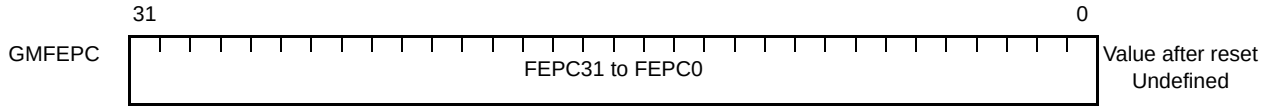


Table 3.53 GMFEPC Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 1	FEPC31 to FEPC1	These bits Indicate the returning PC when FE level exception is accepted.	R/W	Undefined
0	FEPC0	This bit Indicates the returning PC when FE level exception is accepted. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the FERET instruction is executed is 0.	R/W	Undefined

(4) GMFEPSW

When FE level exception handled in Guest mode is accepted, the contents of PSW at that time are saved.

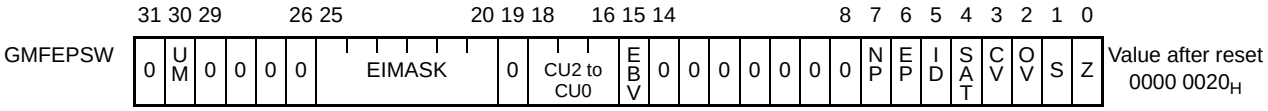


Table 3.54 GMFEPSW Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
30	UM	This bit stores the GMPSW.UM bit setting when an FE level exception is acknowledged.	R/W	0
29 to 26	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
25 to 20	EIMASK	These bits store the GMPSW.EIMASK bits setting when an FE level exception is acknowledged.*1	R/W	0
19	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
18 to 16	CU2 to CU0	These bits store the GMPSW.CU2-0 bits setting when an FE level exception is acknowledged.*2	R/W	0
15	EBV	This bit stores the GMPSW.EBV bit setting when an FE level exception is acknowledged.	R/W	0
14 to 8	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
7	NP	This bit stores the GMPSW.NP bit setting when an FE level exception is acknowledged.	R/W	0
6	EP	This bit stores the GMPSW.EP bit setting when an FE level exception is acknowledged.	R/W	0
5	ID	This bit stores the GMPSW.ID bit setting when an FE level exception is acknowledged.	R/W	1
4	SAT	This bit stores the GMPSW.SAT bit setting when an FE level exception is acknowledged.	R/W	0
3	CY	This bit stores the GMPSW.CY bit setting when an FE level exception is acknowledged.	R/W	0
2	OV	This bit stores the GMPSW.OV bit setting when an FE level exception is acknowledged.	R/W	0
1	S	This bit stores the GMPSW.S bit setting when an FE level exception is acknowledged.	R/W	0
0	Z	This bit stores the GMPSW.Z bit setting when an FE level exception is acknowledged.	R/W	0

Note 1. Only if GMINTCFG.EPL is set (1), the value other than 0 can be set in this field. And if GMINTCFG.EPL is cleared (0), the value of this field becomes 0. Note that if GMINTCFG.EPL is cleared (0) when the value of this field is other than 0, the value of this field becomes 0.

Note 2. The CU2-0 field is a reservation function for future CPUs compatible with this CPU in the future. This field is fixed to 0 in this CPU.

(5) GMPSW

The GMPSW register is PSW (Program Status Word) used for Guest mode.

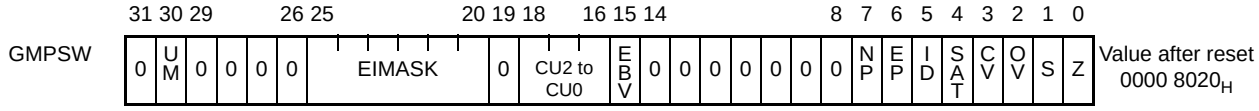


Table 3.55 GMPSW Register Contents (1/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
31	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
30	UM	This bit indicates that the CPU is in user mode (in UM mode). 0: Supervisor mode 1: User mode	R/W	0
29 to 26	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
25 to 20	EIMASK	This field indicates the interrupt priority that becomes the boundary between enabling and disabling an acknowledgement of an interrupt (EIINTn). For an interrupt (EIINTn) with higher priority than the value set in this field, the acknowledgement is enabled. For an interrupt (EIINTn) with priority less than or equal to the value set in this field, the acknowledgement is disabled. 0: Interrupt acknowledgment of all priorities is disabled. 1: Interrupt acknowledgment of priority 1 or lower is disabled (0 is enabled) 2: Interrupt acknowledgment of priority 2 or lower is disabled (1 or more are enabled) ... 62: Interrupt acknowledgment of priority 62 or lower is disabled (61 or higher is enabled) 63: Interrupt acknowledgment of priority 63 or lower is disabled (62 or more are enabled) Only if the GMINTCFG.EPL is set to 1, interrupt acknowledgment control is performed by the value of this field. If the GMINTCFG.EPL is cleared to 0, interrupt acknowledgment control by the value of this field is not performed*1. If the GMINTCFG.EPL is set to 1 and an interrupt (EIINTn) is acknowledged, the interrupt priority is saved in this field as part of PSW change due to acknowledgment of interrupt (EIINTn). The interrupt (EIINTn) with a priority of 63 is always disabled. However, the CPU halt state due to HALT or SNOOZE is also released by an interrupt (EIINTn) with priority 63.	R/W	0
19	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
18 to 16	CU2 to CU0	These bits indicate the coprocessor use permissions. When the bit corresponding to the coprocessor is 0, a coprocessor unusable exception occurs if an instruction for the coprocessor is executed or a coprocessor resource (system register) is accessed. Bit 18 (CU2): Fixed to 0*2. Bit 17 (CU1): FXU Bit 16 (CU0): FPU CU2 to CU0 are fixed to 0 in the devices that do not have corresponding coprocessors.	R/W	000
15	EBV	This bit is always set (1) in Guest mode and can not be cleared (0).	R	1
14 to 8	—	(Reserved for future expansion. Be sure to set to 0.)	R	0

Table 3.55 GMPSW Register Contents (2/2)

Bit Position	Bit Name	Description	R/W	Value After Reset
7	NP	This bit disables the acknowledgement of FE level exception. When an FE level exception is acknowledged, this bit is set to 1 to disable the acknowledgement of EI level and FE level exceptions. As for the exceptions for which the NP bit disables the acknowledgment, see **Table 4.1, Exception Cause List**. 0: The acknowledgement of FE level exception is enabled. 1: The acknowledgement of FE level exception is disabled.	R/W	0
6	EP	This bit indicates that an exception other than an interrupt is being serviced. It is set to 1 when the corresponding exception occurs. This bit does not affect acknowledging an exception request even when it is set to 1. 0: An exception other than an interrupt is not being serviced. 1: An exception other than an interrupt is being serviced.	R/W	0
5	ID	This bit disables the acknowledgement of EI level exception. When an EI level or FE level exception is acknowledged, this bit is set to 1 to disable the acknowledgement of EI level exception. As for the exceptions for which the ID bit disables the acknowledgment, see **Table 4.1, Exception Cause List** This bit is also used to disable the acknowledgement of EI level exceptions as a critical section while an ordinary program or interrupt is being serviced. It is set to 1 when the DI instruction is executed, and cleared to 0 when the EI instruction is executed. The change of the ID bit by the EI or ID instruction will be enabled from the next instruction. 0: The acknowledgement of EI level exception is enabled. 1: The acknowledgement of EI level exception is disabled.	R/W	1
4	SAT*[3]	This bit indicates that a saturation arithmetic operation instruction resulted in overflow and saturation processing is applied to the result. This is a cumulative flag, that is, it is set (1) once a saturation occurs and not cleared (0) by subsequent instructions with unsaturated results. This bit is cleared by the LDSR instruction. Note that execution of an arithmetic operation instruction neither sets nor clears this flag. 0: The result was not saturated 1: The result was saturated	R/W	0
3	CY	This bit indicates whether a carry or borrow has occurred in the operation result. 0: Carry and borrow have not occurred. 1: Carry or borrow has occurred.	R/W	0
2	OV*[3]	This bit indicates whether or not an overflow has occurred during an operation. 0: Overflow has not occurred. 1: Overflow has occurred.	R/W	0
1	S*[3]	This bit indicates whether or not the result of an operation is negative. 0: Result of operation is positive or 0. 1: Result of operation is negative.	R/W	0
0	Z	This bit indicates whether or not the result of an operation is 0. 0: Result of operation is not 0. 1: Result of operation is 0.	R/W	0

Note 1. Only if GMINTCFG.EPL is set to 1, the value other than 0 can be set in this field. If GMINTCFG.EPL is cleared to 0, the value of this field becomes 0. If GMINTCFG.EPL is cleared to 0, since the interrupt acknowledgment controlled by the value of this field is not performed, even if the value of this field is 0, if another interrupt acknowledgement condition is satisfied, an interrupt (EIINTn) can be acknowledged. Note that If GMINTCFG.EPL is cleared to 0 when the value of this field is other than 0, the value of this field becomes 0.

Note 2. The coprocessor use permission CU2 is reserved for future CPUs that are to be made compatible with this CPU.

Note 3. Saturation processing is applied to the operation result in accordance with the contents of the OV and S flags. The SAT flag is set (1) only when the OV flag is set (1) in the saturation arithmetic operation.

(6) GMEIIC

When EI level exception handled in Guest mode is accepted, the cause of the exception is saved. The values held by the GMEIIC register are the exception cause code corresponding to each exception cause.

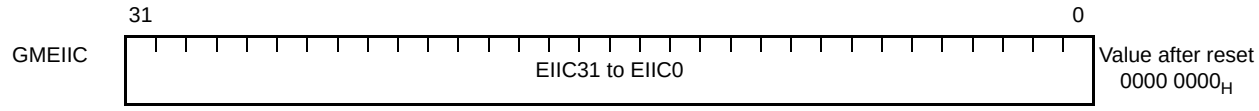


Table 3.56 GMEIIC Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	EIIC31 to EIIC0	When EI level exception is accepted, the exception cause code is saved. In EIIC15 to 0, the exception cause code shown in **Table 4.1** is saved. In EIIC31 to 16, detailed exception cause code defined for each exception is saved. When the function related to the exception is not defined in particular, 0 is set.	R/W	0000 0000H

(7) GMFEIC

When FE level exception handled in Guest mode is accepted, the cause of the exception is saved. The values held by the GMFEIC register are the exception cause code corresponding to each exception cause.

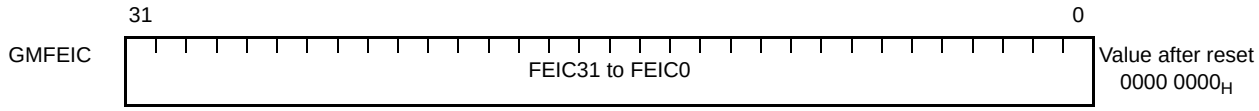


Table 3.57 GMFEIC Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	FEIC31 to FEIC0	When FE level exception is accepted, the exception cause code is saved. In FEIC15 to 0, the exception cause code shown in **Table 4.1** is saved. In FEIC31 to 16, detailed exception cause code defined for each exception is saved. When the function related to the exception is not defined in particular, 0 is set.	R/W	0000 0000H

(8) GMEIWR

The GMEIWR register is the working register for when EI level exception occurred in Guest mode.

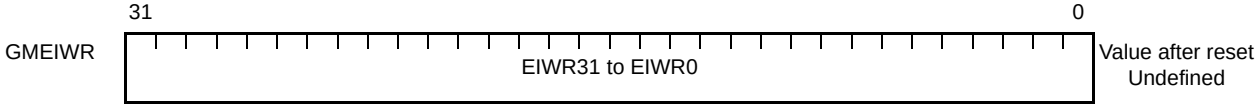


Table 3.58 GMEIWR Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	EIWR31 to EIWR0	This register is an arbitrarily available working register which is used during EI level exception handling and it can be used for temporary saving of general purpose registers.	R/W	Undefined

(9) GMFEWR

The GMFEWR register is the working register for when FE level exception occurred in Guest mode.

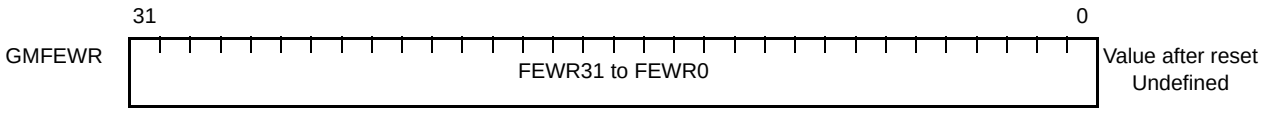


Table 3.59 GMFEWR Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	FEWR31 to FEWR0	This register is an arbitrarily available working register to be used during FE level exception handling and it can be used for temporary saving of general purpose registers.	R/W	Undefined

(10) GMSPID

The GMSPID register is a register that indicates the system protection number of CPU used in Guest mode.

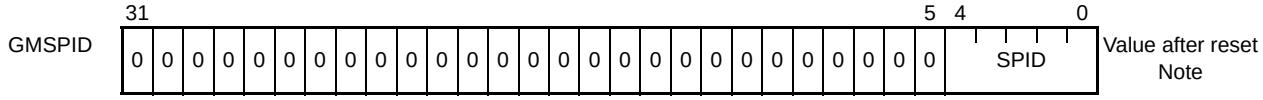


Table 3.60 GMSPID Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 5	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
4 to 0	SPID	This field indicates the system protection number used in Guest mode. The system protection number is a variable ID used for access protection etc. in products composed of multiple bus masters including this CPU. See the specifications of the product for actual usage, restrictions on setting values and so on. In the scope of this CPU, it is used for area matching of the MPU. And it is possible to reflect the system specification defined in the product to the MPU protection function. In addition the system protection number that can be set is given by the SPIDLIST register.*3 Note that when attempting to write a system protection number which can not be set, the value of the GMSPID register is not updated, the original value is kept.	R/W*2	*1

Note 1. For details, see the hardware manual of the product used.

Note 2. Even if SVLOCK.SVL is set (1), updating is possible. However, when updating to SPID when GMSPID is mapped to SPID by changing to Guest mode, the update limitation by SVLOCK.SVL applies.

Note 3. Any update by the LDSR instruction or the LDM.GSR instruction is restricted by the value of the SPIDLIST register. Updating the GMSPID register is not limited by the GMSPIDLIST register. When the CPU operating mode is Guest mode and the GMSPID register mapped to the SPID register is operated by the LDSR instruction, it is restricted by the GMSPIDLIST register mapped to the SPIDLIST register. However, this is the actual operation of the register, and the architecture specification does not change because SPID register update is restricted by SPIDLIST register.

(11) GMSPIDLIST

This register is mapped to the SPIDLIST register when CPU operating mode is Guest mode. In that case, this register shows the list of system protection numbers that can be set in the SPID register. When the system protection number is available for setting, the corresponding bit is set (1). And when the system protection number is not available for setting, the corresponding bit is cleared (0). The GMSPIDLIST register does not directly restrict the update of the GMSPID register.

The value of the GMSPIDLIST can be set in Host mode, but can not be set in Guest mode. And when setting the value of the GMSPIDLIST in Host mode, only the bits of the GMSPIDLIST corresponding to the bit set (1) by the SPIDLIST can be set (1). As a result, the values that can be set for the GMSPID are limited by the SPIDLIST from the outside of the CPU as the system specification, and furthermore, they are the values restricted by the GMSPIDLIST by the hypervisor operating in Host mode.

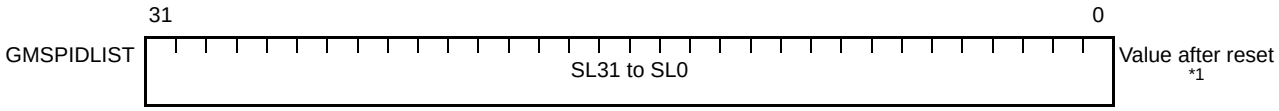


Table 3.61 GMSPIDLIST Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	SL31 to SL0	This field indicates whether the number corresponding to bit n (n = 0 - 31) can be set to the GMSPID. Bit n=0: Setting n to the GMSPID is not possible Bit n=1: Setting n to the GMSPID is possible	R/W	Undefined*1

Note 1. For details, see the hardware manual of the product used.

(12) GMEBASE

This register indicates the method of specifying the handler address of the exceptions or interrupts. Since GMPSW.EBV is always set (1) in Guest mode, the handler address is always specified by the GMEBASE.

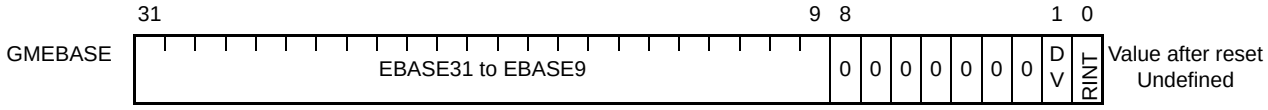


Table 3.62 GMEBASE Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 9	EBASE31 to EBASE9	This field specifies the base address of the handler address. The handler address is the address obtained by adding the exception and interrupt offset address to this base address. EBASE8-0 implicitly uses 0.	R/W	Undefined
8 to 2	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
1	DV	This bit indicates the method of selecting the EIINT interrupt handler address. 0: Follow the specification method accompanying the EIINT request 1: Not follow the specification method accompanying EIINT request, always follow the vector method	R/W	Undefined
0	RINT	This bit specifies reduction of handler address when handler address of EIINT interrupt is the vector method. 0: Not reduce handler address 1: Reduce handler address	R/W	Undefined

(13) GMINTBP

This register is the register that indicates the table base address when table reference method is selected as the EIINT interrupt handler address selection method.

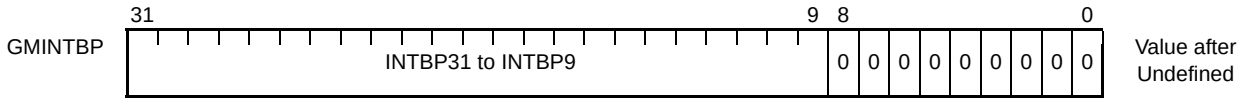


Table 3.63 GMINTBP Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 9	INTBP31 to INTBP9	This field indicates the base address of the EIINT interrupt table reference method. It becomes the first address of the table. INTBP8-0 implicitly uses 0.	R/W	Undefined
8 to 0	—	(Reserved for future expansion. Be sure to set to 0.)	R	0

(14) GMINTCFG

This register indicates settings related to the interrupt function.

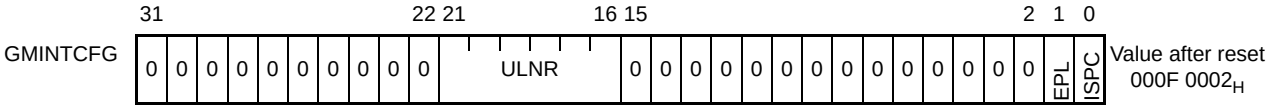


Table 3.64 GMINTCFG Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 22	—	(Reserved for future expansion. Be sure to set to 0.)	R	Undefined
21 to 16	ULNR	This field specifies the maximum value of the available register bank number. If the interrupt using the register bank occurs when the value of RBNR.BN is larger than the ULNR or when the value of the RBNR.BN is 63, the interrupt is not accepted but held, and the SYSERR exception occurs.	R/W	$0F_H$
15 to 2	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
1	EPL	This bit sets the interrupt priority level extension function. 0: Interrupt priority extension is disabled. 1: Interrupt priority extension is enabled. In guest mode, the interrupt priority expansion is always enabled.	R	1
0	ISPC	This bit sets the method of changing the write to the ISPR register. 0: The ISPR is updated automatically. 1: The ISPR is not updated automatically. However, in Guest mode, the interrupt priority extension is always enabled, so interrupt control by the ISPR register is not performed. The value of this bit is always cleared (0).	R	0

(15) GMPLMR

This register inhibits the EIINT interrupt acceptance below the specified interrupt priority level.

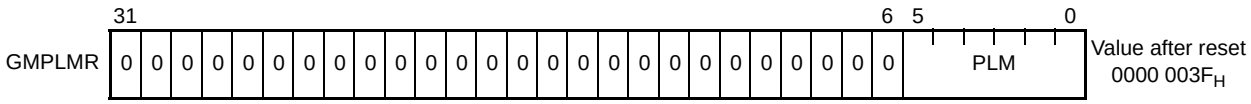


Table 3.65 GMPLMR Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 6	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
5 to 0	PLM	This field prohibits the EIINT interrupt acceptance below the set interrupt priority level. PLM / Priority to prohibit acceptance 0 — Prohibit accepting all priorities 1 — Priority 1 or less (Only 0 can be accepted) : 62 — Priority 62 or less (61 or more is possible) 63 — Priority 63 or less (62 or more is possible) Setting PLMR to 0 prohibits acceptance of all priority EIINT interrupts. This has the same effect as setting the PSW.ID (1). Note that acceptance of the EIINT interrupt with priority 63 is always prohibited, regardless of the setting value of PLMR.	R/W	$3F_H$

(16) GMSVLOCK

This register suppresses the operating authority in the supervisor mode.

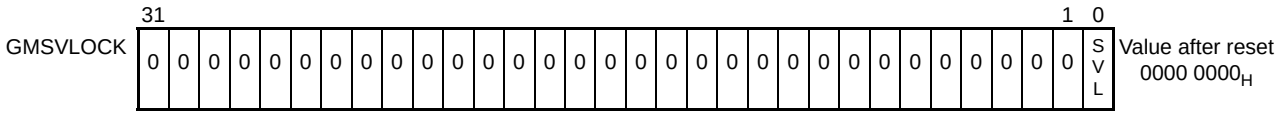


Table 3.66 GMSVLOCK Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 1	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
0	SVL	This bit specifies suppression of the operating authority in the supervisor mode. 0: Operating authority in the supervisor mode is not suppressed 1: Operating authority in the supervisor mode is suppressed When the SVL bit is set (1), the following system registers *1 will not be updated even in the supervisor mode. GMSPID, GMMPM, MPLA, MPUA, MPAT, MPIDn, MPBK	R/W	0

Note 1. The target system registers are the registers related to memory access. The GMSVLOCK register prevents these system registers from being unexpectedly rewritten and unintended memory accesses to be made outside the CPU.

(17) GMMEA

This register is the MEA register for Guest mode.

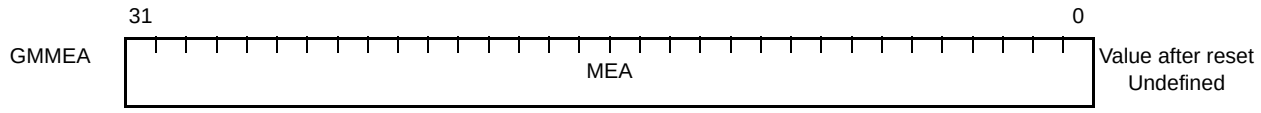


Table 3.67 GMMEA Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 0	MEA	When the MAE and MDP violation occur, the address at that time is saved.	R/W	Undefined

(18) GMMEI

This register is the MEI register for Guest mode.

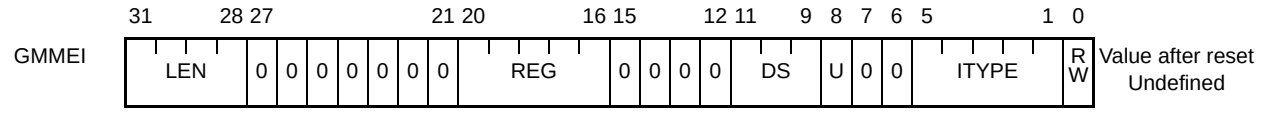


Table 3.68 GMMEI Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 28	LEN	This field indicates the instruction code size of the instruction that caused the exception.	R/W	Undefined
27 to 21	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
20 to 16	REG	This field indicates the source register number or the destination register number of the instruction that caused the exception.	R/W	Undefined
15 to 12	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
11 to 9	DS	This field indicates the data type of the instruction that caused the exception*1.	R/W	Undefined
8	U	This bit indicates the sign extension method of the instruction that caused the exception.	R/W	Undefined
7, 6	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
5 to 1	ITYPE	This field indicates the instruction that caused the exception.	R/W	Undefined
0	RW	This bit indicates the operation of the instruction that caused the exception.	R/W	Undefined

Note 1. Even when the access is divided by hardware, the data type indicated by the instruction is saved.

The instruction that caused the exception and the value of the GMMEI register are the same as the specification of the HMMEI. For details, see **Section 3.11 (18), HMMEI**.

(19) GMMPM

This register determines the MPU operating state in Guest mode.

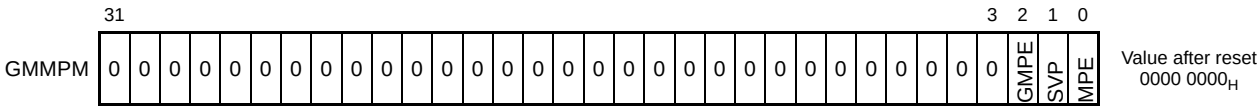


Table 3.69 GMMPM Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 3	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
2	GMPE	This bit enables memory protection by the host management entry in Guest mode. 0: In Guest mode memory protection setting by the host management entry is disabled. 1: In Guest mode memory protection setting by the host management entry is enabled. When memory protection by the host management entry in Guest mode is disabled, only the setting of the guest management entry is used to determine memory protection. Note that changing of GMPE is not possible in Guest mode.	R/W	0
1	SVP	This bit specifies whether to enable or disable the memory protection function by the guest management entry in the SV mode (PSW.UM = 0) in Guest mode. 0: Memory protection is disabled in SV mode. 1: Memory protection is enabled in SV mode.	R/W*1	0
0	MPE	This bit specifies whether to enable or disable the memory protection function by the guest management entry in Guest mode. 0: Memory protection function by the guest management entry is disabled in Guest mode. 1: Memory protection function by the guest management entry is enabled in Guest mode. When setting the value of this bit to 1, be sure to set at least one guest management entry. When memory protection by the guest management entry in Guest mode is disabled, only the setting of the host management entry is used to determine memory protection. When changing to Guest mode and mapping to MPM, MPM and MPE can be updated in Guest mode.	R/W*1	0

Note 1. It can be updated even if SVLOCK.SVL is set (1). However, when the GMMPM is mapped to the MPM after changing to Guest mode and it updates as the MPM, the update limitation by the SVLOCK.SVL is effective.

(20) GMPEID

This register indicates the processor element number. And this register is not subject to save and return by the STM.GSR or the LDM.GSR instruction.

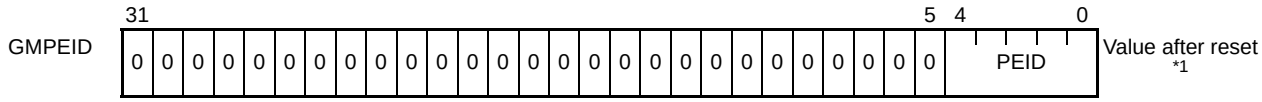


Table 3.70 GMPEID Register Contents

Bit Position	Bit Name	Description	R/W	Value After Reset
31 to 5	—	(Reserved for future expansion. Be sure to set to 0.)	R	0
4 to 0	PEID	This field indicates the processor element number. The value of this field is rewritable. When the virtual processor element number is set in Host mode, its virtual processor element number is read from the PEID register in Guest mode. However, the virtual processor element number set in this register is not used for processing outside the CPU. When the function that the processor element number is used outside the CPU is implemented, in the function external to the CPU, whatever the value of this register is, the processor element number specified by the product specification which was given as an initial value to this register is always used. When this register is accessed as the PEID in Guest mode, this register is not updated by writing to PEID.	R/W	*1

Note 1. For details, see the hardware manual of the product used.

Section 4 Exceptions and Interrupts

An exception is a particular event that forces branching of operation from the current program to another program.

The program at the branch destination of a given exception is called an “exception handler”.

CAUTION

This CPU handles interrupts as types of exception.

4.1 Outline of Exceptions

This section describes the elements that assign properties to exceptions, and shows how exceptions work.

4.1.1 Exception Cause List

This CPU supports the following exceptions.

Table 4.1 Exception Cause List

Exception	Name	Source	Type[*1]	Return/ Restoration	Priority Order[*2]		Restricted operating mode when the exception occurred[*5]	Restricted operating mode when exception handling is done[*5]
					Priority Level	Priority		
RESET	Reset	Reset input	Terminating	—	1	—	Operating mode independent	Host mode
MDP	Memory protection exception (access privilege)	Memory protection violation due to an interrupt using the table reference method	Terminating	Yes	2[*3]	—	[*8]	[*8]
FENMI	FENMI interrupt	Interrupt input terminal	Terminating	No	3	1	Operating mode independent	Host mode
FEINT	FEINT interrupt	Interrupt input terminal	Terminating	Yes	3	2	Operating mode independent	Host mode
EIINT0-2047	User interrupt	Interrupt input terminal	Terminating	Yes	4	[*4]	Operating mode independent	Host mode
BGFEINT	BGFEINT exception	Background specification of GMFEINT interrupt	Terminating	Yes	5	1	Guest mode	Host mode
BGEIINT	BGEIINT exception	Background specification of GMEINT interrupt	Terminating	Yes	5	2	Guest mode	Host mode
GMFEINT	GMFEINT interrupt	Interrupt input terminal	Terminating	Yes	6	1	Guest mode	Guest mode
SYSERR	System error	Error due to context saving to the register bank	Terminating	No[*6]	6	2	Guest mode	[*9]
GMEIINT	User interrupt	Interrupt input terminal	Terminating	Yes	7	—	Guest mode	Guest mode
MIP	Memory protection exception (execution privilege)	Memory protection violation due to instruction fetching	Resumable	Yes	8	1	[*8]	[*8]
SYSERR	System error	Error due to instruction fetching	Resumable	No[*6]	8	2	Host mode	Host mode
							Guest mode	[*9]
UCPOP	Coprocessor unusable exception	Execution of a coprocessor instruction/access permission violation	Resumable	Yes	8	3	Operating mode independent	Same mode as when it occurred
RIE	Reserved instruction exception	Execution of a reserved instruction	Resumable	Yes	8	4	Operating mode independent	Same mode as when it occurred
PIE	Privilege instruction exception	Execution of a privileged instruction/access permission violation	Resumable	Yes	8	5	Operating mode independent	Same mode as when it occurred
SYSERR	System error	Error prior to context restoration from the register bank	Resumable	No[*6]	8	6	Guest mode[*10]	[*9]
MAE	Misalignment exception	Misaligned access occurrence	Resumable	Yes	9	[*7]	Operating mode independent	Same mode as when it occurred
MDP	Memory protection exception (access privilege)	Memory protection violation due to operand access	Resumable	Yes	9		[*8]	[*8]
FPE	FPU exception (precise)	Execution of an FPU instruction	Resumable	Yes	9		Operating mode independent	Same mode as when it occurred
FXE	FXU exception (precise)	Execution of an FXU instruction	Resumable	Yes	9		Operating mode independent	Same mode as when it occurred
HVTRAP	Hypervisor trap	Execution of HVTRAP instruction	Pending	Yes	9		Operating mode independent	Host mode
SYSCALL	System call	Execution of the SYSCALL instruction	Pending	Yes	9		Operating mode independent	Same mode as when it occurred
FETRAP	FE level trap	Execution of the FETRAP instruction	Pending	Yes	9		Operating mode independent	Same mode as when it occurred
TRAP0	EI level trap 0	Execution of the TRAP instruction	Pending	Yes	9		Operating mode independent	Same mode as when it occurred
TRAP1	EI level trap 1	Execution of the TRAP instruction	Pending	Yes	9		Operating mode independent	Same mode as when it occurred

Note 1. For details, see **Section 4.1.3, Types of Exceptions**.

Note 2. The acknowledgment priority for exceptions is checked by the priority level, and then priority. A smaller value has a higher priority. For details, see **Section 4.1.4, Exception Acknowledgment Conditions and Priority Order**.

Note 3. The case in which an MDP exception occurs during the processing (table read or automatic context saving onto a register bank) which is performed after a table reference

method interrupt (EIINTn) is selected as the result of priority determination. The occurrence of this type of exceptions takes precedence over that of the terminating-type exceptions except the reset. For details, see **Section 4.1.2, Overview of Exception Causes**.

Note 4. EIINT0 to 2047 are selected according to the channel. For details, see **Section 4.1.5, Interrupt Exception Priority and Priority Masking**.

Note 5. The restricted operating mode when an exception occurred and the restricted operating mode which handles that exception are shown. Handling of an exception that occurred in guest mode may be performed after transition to host mode. The fact that the exception occurrence does not depend on the operating mode shows that the exception occurs in both host mode and guest mode and can be accepted if the acknowledgment conditions is satisfied.

Note 6. It is impossible to return/restore to the original program where the SYSERR exception occurred. However, if, for example, a SYSERR exception occurs during the execution of an application program, management software such as an operating system as an exception handler processing can make the execution of that application program stop, initiate a software reset, or execute other application program. Also, as shown in **Table 4.3**, if a SYSERR exception occurs in guest mode and GMCFG.GSYSE is set (1), the exception handler is processed in host mode. Therefore, it is possible to make the virtual machine that caused the SYSERR exception by the virtualization software stop, initiate a software reset on the virtual machine, or make other virtual machine's operation start.

Note 7. Since it occurs exclusively, there is no priority difference within the same priority level.

Note 8. For any cause of memory protection violation, the restricted operating mode at exception handling is determined by the condition shown by **Table 4.2**.

Note 9. For any cause in guest mode, the restricted operating mode at exception handling is determined by the condition shown by **Table 4.3**.

Note 10. The register bank function and RESBANK instruction cannot be used in host mode. Therefore, when the restricted operating mode is the host mode, a SYSERR exception does not occur prior to context restoration from the register bank. For details, see **Section 4.5.3, Context Restoration**.

Table 4.2 Relationship between the occurrence condition of memory protection violation exception (MIP, MDP) and restricted operating mode at exception handling

Restricted operating mode when the exception occurred	Occurrence condition			Restricted operating mode at exception handling
	Restricted operating mode to which the exception occurrence cause belongs*1	MPU entry for which a violation is detected*2	GMCFG	
Host mode	Host mode	Host management entry	—	Host mode
Guest mode	Host mode	Host management entry	—	Host mode*3
	Guest mode	Host management entry	HMP = 1	Host mode*3
			HMP = 0	Guest mode
		Guest management entry	GMP = 1	Host mode
			GMP = 0	Guest mode

Note 1. Among the causes (instruction fetch, operand access) that make a memory protection violation exception occur, the case that the restricted operating mode to which the exception occurrence cause belongs and the restricted operating mode at the time of exception occurrence are different is only when the interrupt (EIINTn) that generates interrupt handler address by table reference method to be generated occurs in guest mode and a memory protection violation is detected in the table read.

Note 2. For details of the difference in violation detection depending on the type of MPU entry and the state of restricted operating mode see **Section 5.1, Memory Protection Unit (MPU)**.

Note 3. If an interrupt (EIINTn) for generating an interrupt handler address by the table reference method occurred in the guest mode, but a memory protection violation is detected in the table read, the host mode is entered and the MDP exception handler is processed. Also, when an interrupt (GMEIINTn) that generates an interrupt handler address by the table reference method occurred in the guest mode, and a memory violation of the table read in host management entry is detected when GMCFG.HMP bit is set (1), the host mode is entered and the MDP exception handler is processed. When MDP exception (terminating type) occurs under these two conditions, FEPSWH.GM is set (1) in either case and 009DH is stored in the lower-order 16 bits of HMFEIC. Therefore, with only these register values, it is not possible to distinguish whether MDP exception (terminating type) occurred in host mode or guest mode interrupt. it is necessary to identify from the violation address stored in the MEA.

Table 4.3 Relationship between occurrence condition of system error occurring in guest mode and restricted operating mode at exception handling

System error	Occurrence condition	Restricted operating mode at exception handling
	GMCFG	
Error due to context saving to the register bank Error due to instruction fetching Error prior to context restoration from the register bank	GSYSE = 1	Host mode
	GSYSE = 0	Guest mode

Table 4.4 Exception Acknowledgment Condition*1

		Check the guest partition		Host context register					Guest context register			
				HMINTCFG.EPL								
				0	1							
		PSWH		ISPR	HMPSW	HMPLMR	HMPSW		GMPSW	GMPLMR	GMPSW	
Exception	Type	GM	GPID	ISP	EIMASK	PLM	ID	NP	EIMASK	PLM	ID	NP
RESET	Terminating	x	x	x	x	x	x	x	x	x	x	x
MDP	Terminating	x	x	x	x	x	x	x	x	x	x	x
FENMI	Terminating	x	x	x	x	x	x	x	x	x	x	x
FEINT	Terminating	x	x	x	x	x	x	0	x	x	x	x
EIINT0-2047	Terminating	x	x	Enabled	Enabled	Enabled	0	0	x	x	x	x
BGFEINT	Terminating	1	x	x	x	x	0	0	x	x	x	0
BGEIINT	Terminating	1	x	x	x	x	0	0	Enabled	Enabled	0	0
GMFEINT	Terminating	1	Match	x	x	x	x	x	x	x	x	0
SYSERR*2	Terminating	x	x	x	x	x	x	0	x	x	x	x
SYSERR*3	Terminating	x	x	x	x	x	x	x	x	x	x	0
GMEIINT	Terminating	1	Match	x	x	x	x	x	Enabled	Enabled	0	0
MIP	Resumable	x	x	x	x	x	x	x	x	x	x	x
SYSERR*4	Resumable	x	x	x	x	x	x	x	x	x	x	x
UCPOP	Resumable	x	x	x	x	x	x	x	x	x	x	x
RIE	Resumable	x	x	x	x	x	x	x	x	x	x	x
PIE	Resumable	x	x	x	x	x	x	x	x	x	x	x
SYSERR*5	Resumable	x	x	x	x	x	x	x	x	x	x	x
MAE	Resumable	x	x	x	x	x	x	x	x	x	x	x
MDP	Resumable	x	x	x	x	x	x	x	x	x	x	x
FPE	Resumable	x	x	x	x	x	x	x	x	x	x	x
FXE	Resumable	x	x	x	x	x	x	x	x	x	x	x
HVTRAP	Pending	x	x	x	x	x	x	x	x	x	x	x
SYSCALL	Pending	x	x	x	x	x	x	x	x	x	x	x
FETRAP	Pending	x	x	x	x	x	x	x	x	x	x	x
TRAP0	Pending	x	x	x	x	x	x	x	x	x	x	x
TRAP1	Pending	x	x	x	x	x	x	x	x	x	x	x

Note: x: Not an acknowledgment condition

Note 1. For the exception acknowledgment condition, the relationship between the exception cause and the guest partition is first confirmed. If it is an exception for host mode, the state of the current guest partition does not affect exception acknowledgment, and exception acknowledgment condition will be confirmed by the order of ISPR or HMPSW.EIMASK (selected by HMINTCFG.EPL), HMPLMR, HMPSW.ID, HMPSW.NP. In exceptions for guest mode, the state of the current guest partition may affect exception acceptance. There are exceptions for which the acknowledgment condition is that the restricted operating mode is the guest mode and other exceptions for which, in addition to this acknowledgment condition, the guest partition ID specified by the exception cause side and the current guest partition ID (PSWH.GPID) must match. When the exception for guest mode can acknowledge the state of the current guest partition, the exception acknowledgment condition will be confirmed in the order of GMPSW.EIMASK, GMPLMR, GMPSW.ID, GMPSW.NP.
If an exception is masked under a certain acknowledgment condition, the acknowledgment conditions that are in the subsequent order are not confirmed. Additionally, depending on the type of exception, the conditions which do not become acknowledgment conditions are not confirmed.

Note 2. The case in which the restricted operating mode at exception occurrence is guest mode and the restricted operating mode at exception handling is host mode.

Note 3. The case in which the restricted operating mode at exception occurrence is guest mode and the restricted operating mode at exception handling is guest mode.

Note 4. The acknowledgment condition of the SYSERR exception due to the instruction fetch is unconditional, and it is irrelevant to the state of the restricted operating mode at the exception occurrence and at exception handling.

Note 5. The acknowledgment condition of the SYSERR exception prior to context restoration from the register bank is unconditional, and it is irrelevant to the state of the restricted operating mode at the exception occurrence and at exception handling.

Table 4.5 Resource update by exception acceptance (1/2)

Exception	Restricted operating mode when exception handling is done*1	Saved Resource	Exception Cause Code*2	Host context register										Guest context register					
					INTCFG.EPL														
					0		1												
				PSWH	ISPR	HMPSW	ISPR	HMPSW						GMPSW					
				GM	ISP	EIMASK	ISP	EIMASK	UM	ID	NP	EP	EBV	EIMASK	UM	ID	NP	EP	EBV
RESET*3	Host	HMFE	None	0	0000_H	00_H	0000_H	00_H	0	1	0	0	0	00_H	0	1	0	0	0
MDP	Host	HMFE	*4	0	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	*4	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s
FENMI	Host	HMFE	$E0_H$	0	s	s	s	s	0	1	1	0	s	s	s	s	s	s	s
FEINT	Host	HMFE	$F0_H$-FF_H*5	0	s	s	s	s	0	1	1	0	s	s	s	s	s	s	s
SYSERR	Host	HMFE	*6	0	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
EIINT0-2047	Host	HMEI	1000_H-$17FF_H$*5	0	*8	s	s	*9	0	1	s	0	s	s	s	s	s	s	s
BGFEINT	Host	HMEI	$D800_H$-$D80F_H$*5	0	s	s	s	s	0	1	s	1	s	s	s	s	s	s	s
BGEIINT	Host	HMEI	$D000_H$-$D7FF_H$*5	0	s	s	s	s	0	1	s	1	s	s	s	s	s	s	s
GMFEINT	Guest	GMFE	$F0_H$-FF_H*5	s	s	s	s	s	s	s	s	s	s	s	0	1	1	0	s
SYSERR	Host	HMFE	*6	0	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	*6	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s
GMEIINT	Guest	GMEI	1000_H-$17FF_H$*5	s	s	s	s	s	s	s	s	s	s	*9	0	*10	s	0	s
MIP	Host	HMFE	*4	0	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	*4	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s
SYSERR	Host	HMFE	*6	0	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	*6	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s
UCPOP	Host	HMFE	80_H-82_H*7	s	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	80_H-82_H*7	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s
RIE	Host	HMFE	60_H	s	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	60_H	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s
PIE	Host	HMFE	$A0_H$	s	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	$A0_H$	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s
SYSERR	Host	HMFE	*6	0	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	*6	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s
MAE	Host	HMFE	$C0_H$	s	s	s	s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	$C0_H$	s	s	s	s	s	s	s	s	s	s	s	0	1	1	1	s

Table 4.5 Resource update by exception acceptance (2/2)

Exception	Restricted operating mode when exception handling is done*1	Saved Resource	Exception Cause Code*2	Host context register										Guest context register					
					INTCFG.EPL														
					0			1											
				PSWH	ISPR	HMPSW		ISPR	HMPSW					GMPSW					
				GM	ISP	EIMASK		ISP	EIMASK	UM	ID	NP	EP	EBV	EIMASK	UM	ID	NP	EP	EBV
MDP	Host	HMFE	*4	0	s	s		s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	*4	s	s	s		s	s	s	s	s	s	s	s	0	1	1	1	s
FPE	Host	HMEI	71_H	s	s	s		s	s	0	1	s	1	s	s	s	s	s	s	s
	Guest	GMEI	71_H	s	s	s		s	s	s	s	s	s	s	s	0	1	s	1	s
FXE	Host	HMEI	75_H	s	s	s		s	s	0	1	s	1	s	s	s	s	s	s	s
	Guest	GMEI	75_H	s	s	s		s	s	s	s	s	s	s	s	0	1	s	1	s
HVTRAP	Host	HMEI	$F000_H$-$F01F_H$	0	s	s		s	s	0	1	s	1	s	s	s	s	s	s	s
SYSCALL	Host	HMEI	8000_H-$80FF_H$	s	s	s		s	s	0	1	s	1	s	s	s	s	s	s	s
	Guest	GMEI	8000_H-$80FF_H$	s	s	s		s	s	s	s	s	s	s	s	0	1	s	1	s
FETRAP	Host	HMFE	31_H-$3F_H$	s	s	s		s	s	0	1	1	1	s	s	s	s	s	s	s
	Guest	GMFE	31_H-$3F_H$	s	s	s		s	s	s	s	s	s	s	s	0	1	1	1	s
TRAP0	Host	HMEI	40_H-$4F_H$	s	s	s		s	s	0	1	s	1	s	s	s	s	s	s	s
	Guest	GMEI	40_H-$4F_H$	s	s	s		s	s	s	s	s	s	s	s	0	1	s	1	s
TRAP1	Host	HMEI	50_H-$5F_H$	s	s	s		s	s	0	1	s	1	s	s	s	s	s	s	s
	Guest	GMEI	50_H-$5F_H$	s	s	s		s	s	s	s	s	s	s	s	0	1	s	1	s

Note: s: Retained

Note 1. There are cases in which the restricted operating mode at the time an exception occurs and at the time of exception handling are different. For details, see **Table 4.1, Exception Cause List**

Note 2. Represents lower-order 16 bits of the exception cause code. The higher-order 16 bits of the exception cause code are loaded with a detail code which is defined for each exception. The code is 0000_H unless it is specifically described in the individual functional descriptions.

Note 3. When a reset occurs, the system registers are initialized, but only FEPC performs a special operation of storing the PC of the last instruction which has completed just before.

Note 4. The value depends on the type of memory protection violation and the type of MPU entry for which a memory protection violation was detected. For details, see **Section 4.1.2, Overview of Exception Causes**.

Note 5. With these exception causes, the values within the specified range are stored as exception cause code. These values correspond to the channel number of each exception cause.

Note 6. The value depends on the occurrence cause of system error. For details, see **Section 4.1.2, Overview of Exception Causes**.

Note 7. 80_H-82_H correspond to the coprocessor use permissions (CU0-CU2), respectively.

Note 8. The bit corresponding to the priority of the acknowledged interrupt (EIINTn) is set (1).

Note 9. The priority of the acknowledged interrupt (EIINTn) is stored.

Note 10. There are cases in which the PSW.ID bit is set to 0 for interrupts of table reference method in which the register bank is used. For details, see **Section 4.5.2, Automatic Context Saving**.

4.1.2 Overview of Exception Causes

The following is an overview of the exception causes handled by the CPU.

(1) RESET

For details, see the “CPU” section in the hardware manual of the product used.

(2) FENMI, FEINT, and EIINT

These are the interrupts for host mode among the interrupts generated by interrupt signals from the interrupt controller to activate a certain program. The interrupt handlers are handled in host mode. For details about the interrupt function, see **Section 3.3, Interrupt Function Registers** and the hardware manual of the product used.

(3) GMFEINT, GMEIINT

These are the interrupts for guest mode among the interrupts generated by interrupt signals from the interrupt controller to activate a certain program. Along with the interrupt request, the guest partition ID for which the interrupt request should be processed is also notified. For an interrupt to be acknowledged, the restricted operating mode is guest mode and the value of the current guest partition ID (PSWH.GPID) must match the value of the guest partition ID notified from the interrupt controller . For details about interrupt function, see **Section 3.3, Interrupt Function Registers** and the hardware manual of the product used.

(4) BGFEINT, BGEIINT

These are the background interrupts for guest mode among the interrupts generated by interrupt signals from the interrupt controller to activate a certain program. The causes of the background interrupts are the interrupt causes (GMFEINT, GMEIINTn) for guest mode. They occur when the specific condition is satisfied at the interrupt controller side and the value of the guest partition ID to be processed by the interrupt managed by the interrupt controller and the value of the current guest partition ID (PSWH.GPID) are different from each other. When the background interrupt is acknowledged in guest mode, the restricted operating mode is transitioned to host mode and is handled as an EI level exception. For details about background interrupt, see **Section 4.1.9, Background Interrupts**. For details about the interrupt controller, see the hardware manual of the product used.

Table 4.6 shows the exception cause codes that are loaded in the HMEIIC when acknowledging background interrupt

Table 4.6 Exception cause code of background interrupt

Background Interrupt	HMEIIC bit*[1]	Stored content
BGFEINT	[18:16]	The guest partition ID in which the generation cause GMFEINT is to be processed.
	[15:8]	$D8_H$
	[3:0]	Interrupt channel number of generation cause GMFEINT.
BGEIINT	[18:16]	The guest partition ID in which the generation cause GMEIINT is to be processed.
	[15:12]	D_H
	[11]	0_B
	[10:0]	Interrupt channel number of generation cause GMEIINT.

Note 1. The other bits are padded with 0.

(5) SYSERR

This is a system error exception.

An error occurring during automatic context saving using the register bank function is notified as a terminating-type SYSERR exception. In this case, as with the case of acknowledging an interrupt, the PC of the instruction that is interrupted when the exception occurred is loaded in the FEPC and the PSW at that time in the FEPSW, respectively. An error occurring during the restoration of the context with the RESBANK instruction is notified as a resumable-type SYSERR exception. In this case, the PC of the RESBANK instruction is loaded in the FEPC and the PSW at that time in the FEPSW, respectively.

An error that occurs at an instruction fetch access is notified as a resumable-type SYSERR exception. In this case, the PC of the instruction to be fetched is loaded in the FEPC and the PSW at that time in the FEPSW, respectively.

Table 4.7 lists the exception cause codes that are loaded in the lower-order 16 bits of the FEIC register when SYSERR exceptions occur. The higher-order 16 bits of the exception cause code are padded with 0s.

Table 4.7 Lower-order 16 Bits of the Exception Cause Codes Associated with the SYSERR Exception

Exception Cause Code	Cause
11_H	A response error occurred in a bus slave at the time of instruction fetching.*1
13_H	An error within the scope of safety functions, such as an ECC error or parity error, occurred at the time of instruction fetching.*1,*2
$1C_H$	An error occurred when automatically saving context to the register bank.
$1D_H$	An error occurred at the time of context restoration from the register bank (during the execution of the RESBANK instruction).

Note 1. For details, see the hardware manual of the product used.

Note 2. When an ECC or parity error is found in fetching from the instruction cache, it is handled as a cache miss so a SYSERR exception does not occur.

All the causes that generate a SYSERR exception for this CPU are listed in **Table 4.7**. An error that is detected externally to the CPU does not generate a SYSERR exception.

(6) FPE

For details, see the "CPU" section in the hardware manual of the product used.

(7) FXE

For details, see the "CPU" section in the hardware manual of the product used.

(8) MIP and MDP

These are exceptions that occur when the MPU detects a violation. Detecting an exception is performed when the address at which the instruction will access the memory is calculated. For details, see **Section 5.1, Memory Protection Unit (MPU)**.

There are cases in which an MDP exception is detected during a table read or automatic context saving onto the register bank when a table reference method interrupt (EIINT*n*) is selected as the result of determining the priority level of the terminating-type exception. In such a case, the execution of the instruction is interrupted and an MDP exception (terminating-type) is generated as with an ordinary interrupt. At this time, if the MDP exception is handled in the host mode, the PC of the interrupted instruction is saved to HMFEPC, and the HMPSW before interrupt acceptance is saved to HMFEPSW. When an MDP exception is handled in guest mode, the PC of the interrupted instruction is saved to GMFEPC, and the GMPSW before interrupt acceptance is saved to GMFEPSW, respectively. For the table reference method, see **Section 4.4, Exception Handler Address**.

Table 4.8 lists the exception cause codes that are loaded in the lower-order 16 bits of the FEIC register when memory protection violation exceptions (MIP, MDP) occur.

Table 4.8 Lower-order 16 Bits of the Exception Cause Codes Associated with the Memory Protection Exception

Access setting*1		Type of memory protection violation exceptions		
Host management entry	Guest management entry	MIP	MDP (operand access)	MDP (table read*2)
Enabled	Enabled	Does not occur*3	Does not occur*3	Does not occur*3
Enabled	Disabled	90_H	91_H	95_H
Disabled*4	Enabled*4	98_H	99_H	$9D_H$
Disabled	Disabled	90_H	91_H	95_H

Note 1. After considering the setting of all MPU entries, it indicates whether access to the target memory access is enabled or disabled.

Note 2. When the memory protection violation is detected in the table read of an interrupt (EIINT*n*, GMEIINT*n*) using the table reference method

Note 3. Since memory access is enabled, no memory protection violation exception occurs.

Note 4. In host mode, the values of host management entry is "disabled" and guest management entry is "enabled".

(9) RIE

For details, see the "CPU" section in the hardware manual of the product used.

(10) PIE

For details, see the “CPU” section in the hardware manual of the product used.

(11) UCPOP

For details, see the “CPU” section in the hardware manual of the product used.

(12) MAE

For details, see the “CPU” section in the hardware manual of the product used.

(13) TRAP, FETRAP, and SYSCALL

For details, see the “CPU” section in the hardware manual of the product used.

(14) HVTRAP

These are exceptions that occur according to the result of HVTRAP instruction execution. When the HVTRAP instruction is executed in guest mode, the restricted operating mode is transitioned to host mode and is handled as an EI level exception. For details, see the *RH850G4MH Virtualization User's Manual: Software.*

4.1.3 Types of Exceptions

This CPU divides exceptions into the following three types according how they are executed.

- Terminating exceptions
- Resumable exceptions
- Pending exceptions

Also, exceptions classified into these types are further classified into the following two according to the restricted operating mode, in which the exception handler is processed respectively.

- Exception handled in host mode
- Exception handled in guest mode

(1) Terminating Exceptions

For details, see the “CPU” section in the hardware manual of the product used.

CAUTIONS

1. **For details, see the “CPU” section in *the hardware manual of the product used.***
2. **A terminating exception may be accepted during the execution of an instruction that performs multiple memory accesses. In this case, although the execution of the instruction is terminated, the result of memory accesses that have been already completed are not canceled. For example, memory is updated by the PREPARE instruction and the general-purpose registers are updated by the DISPOSE instruction. However, it is guaranteed that the PC and SP retain the original values required to re-execute the instruction. For details, see the *RH850G4MH User’s Manual: Software*. The relevant instructions are listed below.**
   - **PREPARE, DISPOSE, PUSHSP, POPSP, RESBANK, STM.MP, LDM.MP, STM.GSR, LDM.GSR**

(2) Resumable Exceptions

For details, see the “CPU” section in the hardware manual of the product used.

(3) Pending Exceptions

For details, see the “CPU” section in the hardware manual of the product used.

(4) Exception handled in host mode

Exception handler is processed in host mode. There are cases for which the exception cause belongs to the host mode and cases for which the designated processing is performed in host mode but the exception causes belong to guest mode. At exceptions belongs to the host mode, instruction was executed in host mode, pending exception, and terminating exception which occurs regardless of restricted operating mode are corresponding. Although the exception cause belongs to the guest mode, for exceptions handled in the host mode include HVTRAP instruction executed in guest mode. This is

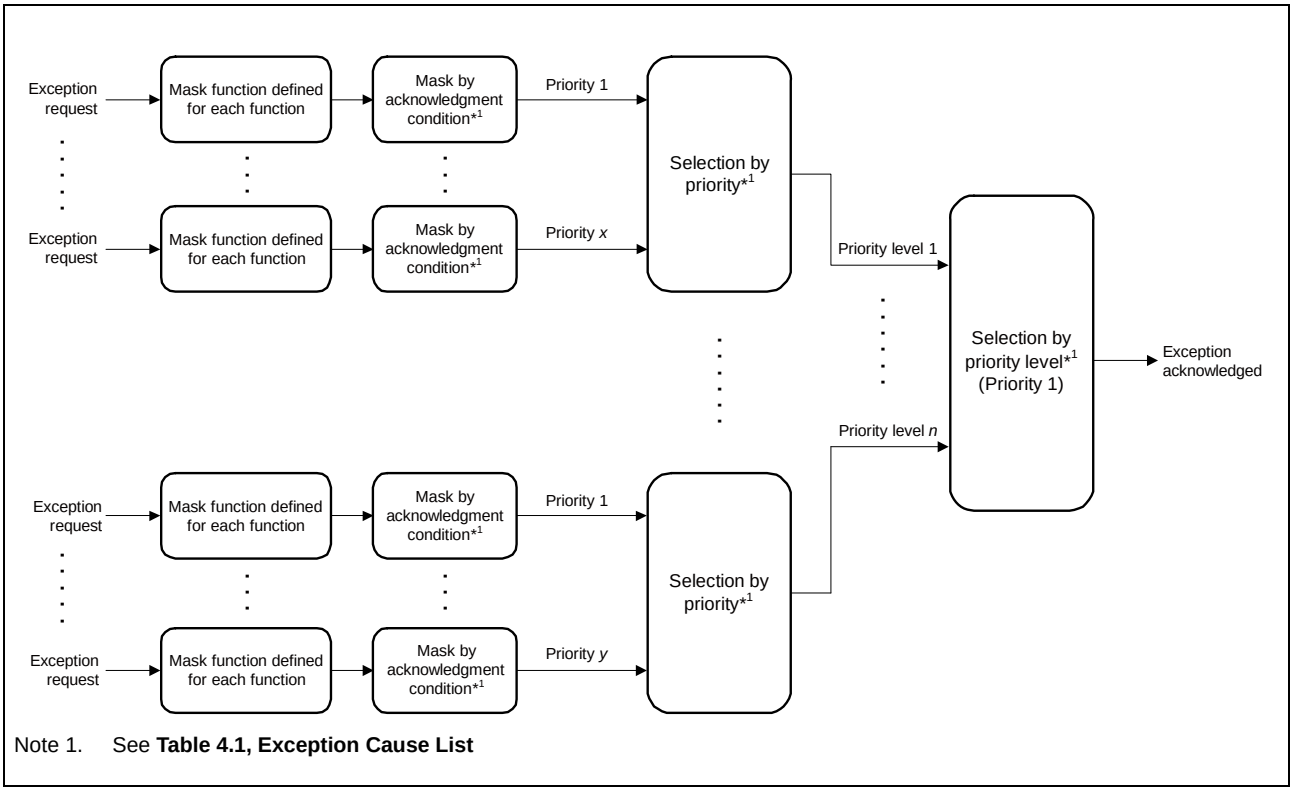
the case when the transition to the host mode was specified by the corresponding bit of GMCFG at the time MIP, MDP, SYSERR exceptions occur.

(5) Exception handled in guest mode

The exception handler is processed in guest mode. This exception is applied to the case for which the exception cause belongs to Guest mode or to the case in which MIP, MDP, or SYSERR exception occurs when handling in guest mode is specified by the corresponding bit in GMCFG.

4.1.4 Exception Acknowledgment Conditions and Priority Order

For details, see the “CPU” section in the hardware manual of the product used.



Note 1. See **Table 4.1, Exception Cause List**

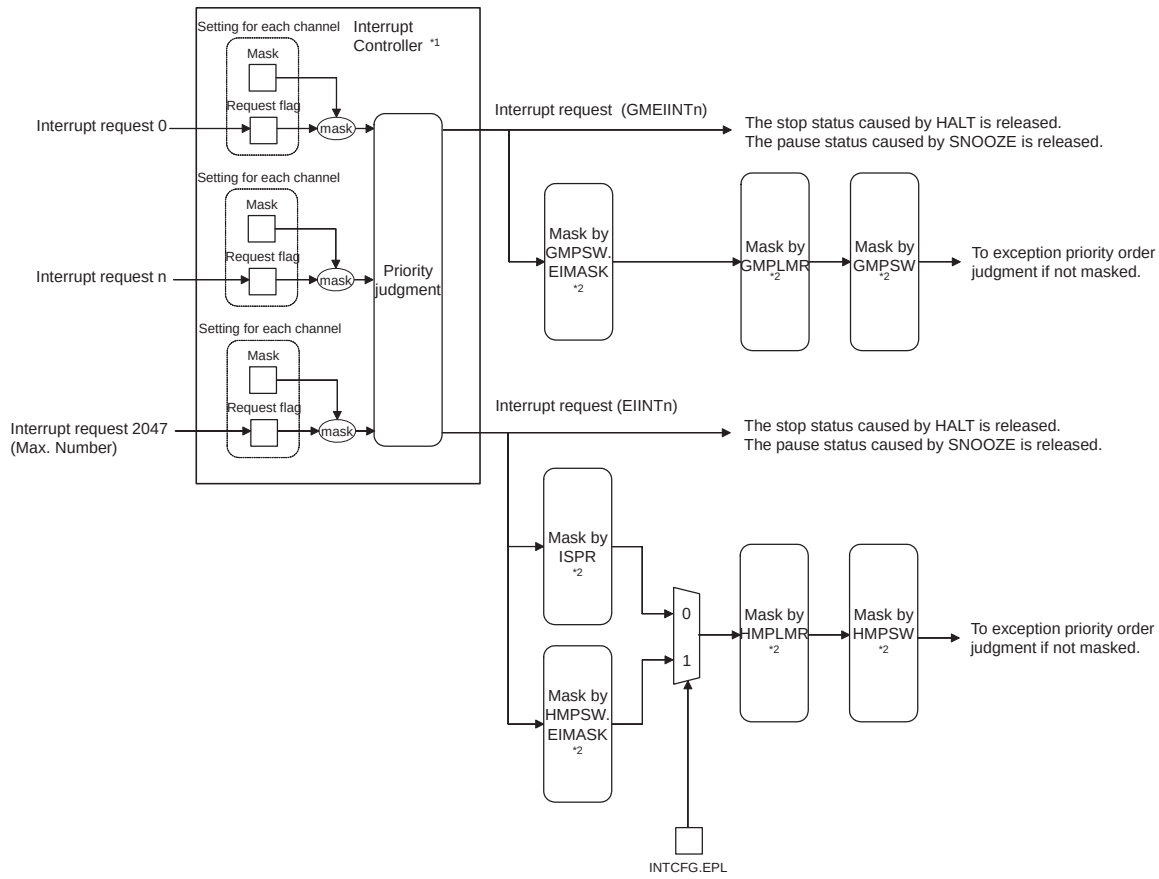
Figure 4.1 Exception Acknowledgment Conditions and Priority Order

For detailed descriptions, see the “CPU” section in the hardware manual of the product used. Additionally, only when the CPU acknowledged BGFEINT or BGEIINT, this CPU does not return acceptance response to the requesting module. Interrupt requests that became cause of background interrupts are retained on the interrupt controller side. Therefore, an interrupt request as GMFEINT or GMEIINT is triggered if the mode transitions to guest mode of the guest partition ID in which the interrupt request is to be processed.

4.1.5 Interrupt Exception Priority and Priority Masking

An interrupt (EIINT*n*, GMEIINT*n*) can be masked for each exception priority or interrupt priority by setting registers. This function allows more flexible software structure and an interrupt ceiling with no maintenance.

Figure 4.2 shows an overview of the functions of interrupt exception priority and priority masking.



Note 1. For details about the interrupt controller, see *the hardware manual of the product used.*

Note 2. EIINTn which is masked by each masking function can be checked by each bit of IMSR or ICSR.PMEI.

Figure 4.2 Interrupt Exception Priority and Priority Masking

(1) Interrupt Priority

This CPU supports below shown priority levels for interrupts (EIINTn) :

Maximum 16 priority level : HMINTCFG.EPL is cleared to 0

Maximum 64 priority level : HMINTCFG.EPL is set to 1

The priority of the interrupt (GMEIINTn) acknowledgeable by this CPU is 64 levels at maximum because the interrupt priority level extension function is always enabled in the guest mode (GMINTCFG.EPL is always set (1)) .

For the details on the procedure to set interrupt priority in interrupt controller, see the hardware manual of the product used.

(a) When an Interrupt which is less than or equal to priority level 16 occurs (when HMINTCFG.EPL is cleared 0)

The function for acknowledgment of interrupt (EIINTn) works in a limited way. Interrupt can be masked by ISPR and HMPLMR. If either one of bit for ISPR is set to 1, all interrupts (EIINTn) for which the priority level is less than or equal to 16 is masked. HMPLMR always works as specified regardless the setting of HMINTCFG.EPL. On the other hand, the priority information of acknowledged interrupt is not saved in ISPR and PSW.EIMASK.

As mentioned above, its use is not recommended because the priority information is not handled by CPU even if the acknowledge of interrupt (EIINTn) is possible.

(b) Constraints for interrupts with priority level 16 to 62

Regardless the setting of HMINTCFG.EPL, below shown constraints are applied for interrupts (EIINTn, GMEIINTn) with priority level 16 to 62.

- When the priority of the acknowledged interrupt (EIINTn) is 16 to 62 and the exception handler address is generated by the direct vector address method, the offset address is 1F0H that is the same as for interrupt with priority level 15. Thus, the same exception handler address is used for all interrupts with a priority level of less than or equal to 15. Refer to **Section 4.4, Exception Handler Address** for more details
- When the priority of the acknowledged interrupt (GMEIINTn) is 16 to 62 in guest mode, the availability of register bank is specified by RBCR0[15] that is the same with the interrupt which is less than equal 15 priority level. For all interrupts (EIINTn) with a priority level of less than or equal to 15, availability of register bank cannot be specified for each priority level. Refer to **Section 4.5.2, Automatic Context Saving**.
  Additionally, in host mode, the register bank function cannot be used when an interrupt (EIINTn) is acknowledged regardless of the value of RBCR0.BE.
- When the priority of the acknowledged interrupt (GMEIINTn) is 16 to 62 in guest mode, the value of the PSW.ID bit after saving to the register bank is specified by RBCR1.NC[15] which is the same as for priority 15. All interrupts (GMEIINTn) which is less than equal 15 priority level can not be specified its availability by each priority level. Refer to **Section 4.5.2 (2), Suppressing the Update of the GMPSW.ID Bit**.
  Additionally, in host mode, the register bank function cannot be used when an interrupt (EIINTn) is acknowledged regardless of the value of RBCR0.BE.

(c) Constraint for interrupt with priority level 63

If HMINTCFG.EPL is cleared to 0 (= Interrupt priority level extension function is disabled), the interrupt (EIINTn) for which the priority level is 63 (lowest priority level) must be masked by HMPLMR. If HMINTCFG.EPL is set to 1 (= Interrupt priority level extension function is enabled), the interrupt (EIINTn) for which priority level is 63 (lowest priority level) must be masked by HMPSW.EIMASK, GMPSW.EIMASK. As mentioned above, interrupt (EIINTn, GMEIINTn) with priority level 63 is not acknowledged regardless of HMINTCFG.EPL. But both stop status caused by HALT and SNOOZE instruction can be released by occurrence of interrupt with priority level 63 (EIINTn, GMEIINTn).

(2) Interrupt Priority Mask

If interrupt priority level extension function is disabled (HMINTCFG.EPL is cleared to 0), interrupt (EIINTn) acknowledgment is judged by ISPR and PLMR. If interrupt priority level extension function in host mode is enabled (HMINTCFG.EPL is set to 1), interrupt (EIINTn) acknowledgment is judged by HMPSW.EIMASK and HMPLMR. In the guest mode, the interrupt priority level extension function is always enabled (GMINTCFG.EPL is always set (1)), and for the interrupt (GMEIINTn), the acknowledgment is judged by GMPSW.EIMASK and GMPLMR.

(a) ISPR

Interrupt acknowledgment judged by ISPR is only done when interrupt priority level extension function is disabled. In other words, ISPR can be used only when the restricted operating mode is host mode.

For the ISPR register, the bit corresponding to the priority is set to 1 when the hardware acknowledges an interrupt, and interrupts with the same or lower priority are masked. When the EIRET instruction corresponding to the interrupt is executed, the corresponding bit of the ISPR register is cleared to 0 to clear the mask.

This automatic interrupt ceiling makes multiple interrupts servicing easy without using software control.

The function of the HMINTCFG register allows you to disable automatic update of the ISPR register upon acknowledgment of and return from an interrupt. To perform interrupt ceiling control by using software without using the function of the ISPR register, set the ISPC bit in the HMINTCFG register to 1, clear the ISPR register, and then control the ceiling value with software by using the HMPLMR register.

(b) HMPSW.EIMASK, GMPSW.EIMASK

Interrupt acknowledge judged by HMPSW.EIMASK and GMPSW.EIMASK are only done when interrupt priority level extension function is enabled.

HMPSW.EIMASK, GMPSW.EIMASK mask the interrupt (EIINTn, GMEIINTn) which has an equal or lower priority than the set value. When CPU acknowledges an interrupt (EIINTn, GMEIINTn), its priority is stored to them. When EIRET or FERET instruction is executed, the value of EIPSW.EIMASK or FEPSW.EIMASK is stored to them. PSW.EIMASK can be changed by LDSR instruction.

(c) HMPLMR, GMPLMR

HMPLMR, GMPLMR mask the interrupt (EIINTn, GMEIINTn) which has which has an equal or lower priority than the set value.

The HMPLMR, GMPLMR registers allows you to mask specific interrupt priorities with software. Use them to raise the priority level of the interrupt ceiling temporarily in a program. The mask setting specified by the ISPR register or HMPSW.EIMASK, GMPSW.EIMASK and the mask setting of HMPLMR, GMPLMR might overlap, and an interrupt is masked if it is masked with one of them. Normally, use the HMPLMR, GMPLMR registers to raise the ceiling value from the ceiling value of the ISPR register or HMPSW.EIMASK, GMPSW.EIMASK.

4.1.6 Return and Restoration

For details, see the “CPU” section in the hardware manual of the product used.

4.1.7 Context switching

To save the current program sequence when an exception occurs, appropriately save the following resources according to the function definitions. There are resources that are automatically saved by hardware and resources that need to be saved by software.

- Program counter (PC)
- Program status word (HMPSW, GMPSW)
- Exception cause code (HMEIIC, HMFEIC, GMEIIC, GMFEIC)
- Work system register (HMEIWR, HMFEWR, GMEIWR, GMFEWR)

The resource to be used as the saving destination is determined according to the exception type. Saved resource determination is described below.

For exceptions that use the register bank function, specific resources are automatically saved. For details, see **Section 4.5, Register Bank Function**.

(1) Context Saving

When exception is acknowledged, the pending bits (HMPSW.ID, NP and GMPSW.ID, NP bits) are automatically set. New exceptions with certain acknowledgment conditions might not be acknowledged, based on these pending bits.

To enable multiple exception handling which makes exceptions of the same level acceptable again, the return registers and certain information about the corresponding exception causes must be saved, such as to a stack. This information that must be saved is called the “context”.

In principle, it is necessary to make sure that no exceptions of the same level can occur before saving the context.

The working system registers that can be used in the work of saving contexts and those for which the values are saved to enable the handling of multiple exceptions as required are referred to as the basic context registers.

These basic context registers are provided for each exception level. For this reason, it is possible to precisely return from the current exception since its context will not be overwritten when an exception of a different level occurs before saving the current context.

Table 4.9 Basic Context Registers

Exception Level	Basic Context Registers	
	Host context register	Guest context register
EI level	HMEIPC, HMEIPSW, HMEIIC, HMEIWR	GMEIPC, GMEIPSW, GMEIIC, GMEIWR
FE level	HMFEPC, HMFEPSW, HMFEIC, HMFEWR	GMFEPC, GMFEPSW, GMFEIC, GMFEWR

About an exception level, see **Section 2.3.2, Exception Level**.

(2) Context Save and Restore Instructions

Table 4.10 shows the list of save and return instructions. When using the save instructions, context can be saved to memory at once by a single instruction. When using the restore instruction, context can be restored from memory at once by a single instruction. Context save/return can be performed in a shorter time with a smaller code size, if context save/return instruction is used, as compared with execution of a combination of multiple instructions.

Table 4.10 Context save/return instruction by resource.

Resource	Save instruction	Return instruction	Specification Method for Operation Target
General-purpose registers	PUSHSP*1	POPSP*1	Specify start and end numbers for consecutive register numbers.
MPU setting entry	STM.MP	LDM.MP	Specify start and end numbers for consecutive entry numbers. *2
Specific System Register	STM.GSR	LDM.GSR	For details, see **Table 4.11**.

Note 1. PUSHSP and POPSP instructions differ from the STM.GSR, LDM.GSR instructions, etc. in the method of generating the access destination address. Therefore, when multiple instructions are required for address generation, it is preferable to use the necessary number of ST.W, LD.W instructions rather than save/return general-purpose registers using the PUSHSP and POPSP instructions; context save/return can be performed in a shorter time.
In comparison of code sizes, PUSHSP and POPSP instructions are somewhat smaller than multiple ST.W, LD.W instructions when context can be saved/returned with 1 instruction. However, when an additional address generation instruction is required, the number of instructions is added to the PUSHSP and POPSP instructions, so that compared with the case where multiple ST.W and LD.W instructions are used, superiority or inferiority varies depending on the number of general-purpose registers to be saved/returned.

Note 2. The MPU setting entry consists of one set of three system registers: MPLA, MPUA, and MPAT. Therefore, when one entry is specified as a target to be processed by the STM.MP, LDM.MP instruction, three system registers are saved/returned.

Table 4.11 List of operation target system registers of STM.GSR, LDM.GSR instruction

Save/Return Order*1	Address offset*2	selID	regID	Registers Name*3, *4
1	00_H	0	6	FPSR
2	04_H	0	7	FPEPC
3	08_H	0	16	CTPC
4	$0C_H$	0	17	CTPSW
5	10_H	0	20	CTBP
6	14_H	1	5	MCTL
7	18_H	1	11	SCCFG
8	$1C_H$	1	12	SCBP
9	20_H	2	15	RBCR0
10	24_H	2	16	RBCR1
11	28_H	2	17	RBNR
12	$2C_H$	2	18	RBIP
13	30_H	5	8	MCA
14	34_H	5	9	MCS
15	38_H	5	11	MCR
16	$3C_H$	5	12	MCI
17	40_H	5	16	MPIDX
18	44_H	9	0	GMEIPC
19	48_H	9	1	GMEIPSW
20	$4C_H$	9	2	GMFEPC
21	50_H	9	3	GMFEPSW
22	54_H	9	5	GMPSW
23	58_H	9	6	GMMEA
24	$5C_H$	9	8	GMMEI
25	60_H	9	13	GMEIIC
26	64_H	9	14	GMFEIC
27	68_H	9	16	GMSPID
28	$6C_H$	9	17	GMSPIDLIST
29	70_H	9	19	GMEBASE
30	74_H	9	20	GMINTBP
31	78_H	9	21	GMINTCFG
32	$7C_H$	9	22	GMPLMR
33	80_H	9	24	GMSVLOCK
34	84_H	9	25	GMMPM
35	88_H	9	28	GMEIWR
36	$8C_H$	9	29	GMFEWR

Note 1. Processing starts from No. 1 in the save/return order.

Note 2. When the target system register is saved / restored, the value obtained by adding the address offset to the value of the base address register becomes the memory address to be accessed.

Note 3. When the STM.GSR, LDM.GSR instruction is executed, all these system registers are saved and restored.

Note 4. GMPEID is not an operation target of the STM.GSR, LDM.GSR instruction.

4.1.8 Exception to transition from guest mode to host mode

As shown in **Table 4.1**, **Table 4.2**, **Table 4.3**, there are exceptions and conditions that do not change the restricted operating mode at exception occurrence and exception handling, but there are exceptions and conditions which cause the transition of restricted operating mode to host mode even if the restricted operating mode is guest mode at exception occurrence.

For exception in which restricted operating mode does not change from guest mode at exception occurrence and exception handling, the exception handling is performed within the guest partition that caused the exception. This means that exceptions that occur within a virtual machine are handled inside the virtual machine.

On the other hand, when the restricted operating mode at exception occurrence is guest mode, if the restricted operating mode at exception handling is host mode, the exception can be categorized into two. The first type requests the virtualization software running in host mode to perform processing that can not be handled in the guest mode like the HVTRAP instruction. The second is an exception for the host mode, which is independent of guest mode operation.

When an exception for which the restricted operating mode at exception handling is host mode is acknowledged, PSWH.GM indicating the restricted operating mode and PSWH.GPID indicating the guest partition are saved to FEPSWH or EIPSWH, depending on the level of the exception that occurred. The transition of the restricted operating mode from the guest mode to the host mode with the exception acknowledgment can be judged by referring to the value of FEPSWH.GM or EIPSWH.GM depending on the exception level. If these values are not changed by an exception handler, when a return instruction is executed at the end of exception handling, it is possible to return to the restricted operating mode at the time the exception occurred. When the restricted operating mode at the time of exception occurrence is guest mode, it is possible to return to the guest partition at the time the exception occurred.

The details of the system register update contents at the time of exception acknowledgement are summarized in the **Table 4.5**, **Table 4.12** and **Figure 4.3** show the relationship between the system register name updated with exception acknowledgment and the restricted operating mode at exception occurrence and exception handling.

When the restricted operating mode at exception occurrence is guest mode and the restricted operating mode at exception handling is host mode, the update of the system register is performed against the host context register.

Table 4.12 Relationship between system registers updated by exception acknowledgment and states of restricted operating mode at exception occurrence and exception handling

Restricted operating mode at exception occurrence	Restricted operating mode at exception handling	Exception Level	PC save location	PSWH save location	HMPSW save location	GMPSW save location	Exception cause save location
Host mode	Host mode	FE	HMFEPC	FEPSWH	HMFEPSW	—	HMFEIC
		EI	HMEIPC	EIPSWH	HMEIPSW	—	HMEIIC
Guest mode	Host mode	FE	HMFEPC	FEPSWH	HMFEPSW*1	—	HMFEIC
		EI	HMEIPC	EIPSWH	HMEIPSW*1	—	HMEIIC
Guest mode	Guest mode	FE	GMFEPC	—	—	GMFEPSW	GMFEIC
		EI	GMEIPC	—	—	GMEIPSW	GMEIIC

Note 1. When the restricted operating mode at exception occurrence is guest mode and the restricted operating mode at exception handling is host mode, HMPSW.ID, NP are updated. Therefore, HMPSW before the exception acceptance must be saved. When returning from exception handling, since HMPSW is restored as it is before the acceptance of exception, exception can be accepted again.

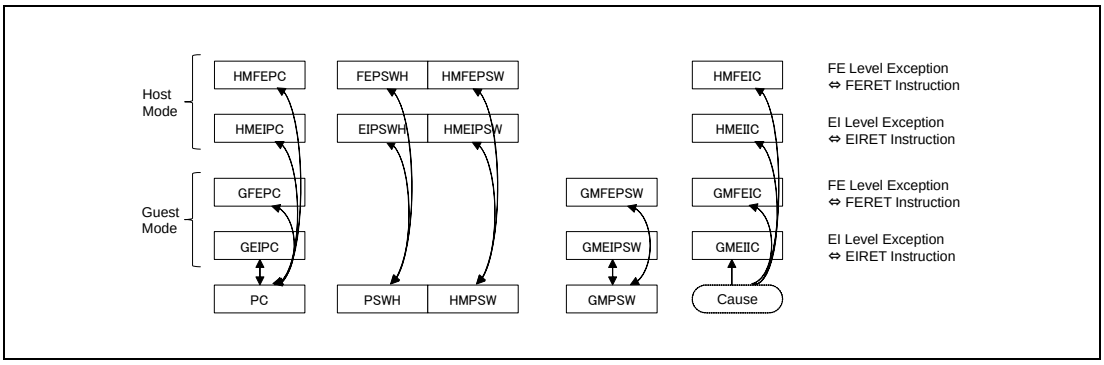


Figure 4.3 Relationship between system registers updated by exception acceptance and state of restricted operating mode during exception handling

Additionally, there is no exception by which the restricted operating mode transits from host mode to guest mode.

In the state transition of the restricted operating mode, the same synchronization processing as the SYNCM instruction is performed at the time of reception of the cause exception and execution of the return instruction. Consequently, all the load and store processing that were executed before the state transition are completed, and state transition is performed with the completion of detection of SYSERR (terminating type) accompanying operand access. For details about synchronization processing see **Section 7.4, Synchronizing for restricted operating mode transition**.

4.1.9 Background Interrupts

(1) Overview of Background Interrupt

For guest mode interrupts (GMFEINT, GMEIINTn), the guest partition ID for which these should be handled is specified. One of the acknowledgment conditions for these interrupts is that the value of this specified guest partition ID matches the value of the CPU's guest partition ID (PSWH.GPID).

Even when restricted operating mode is guest mode, if the guest partition IDs do not match, the interrupt for the guest mode (GMFEINT, GMEIINTn) will not occur. The background interrupt is a process to generate interrupt for that guest partition even if such guest partition IDs do not match.

If the interrupt (GMFEINT, GMEIINTn) cause for the guest mode has been generated by the interrupt controller, but the value of the guest partition ID assigned in these interrupts does not match the value of the CPU's guest partition ID (PSWH.GPID), the interrupt controller changes the interrupt for the guest mode to the background interrupt if the specific condition separately defined by the interrupt controller is satisfied. If GMFEINT is the cause, BGFEINT is generated, and if GMEIINT is the cause, BGEIINT is generated. For details about generation of background interrupt, see the interrupt controller section in the hardware manual of the product used.

When the background interrupt is acknowledged in guest mode, the restricted operating mode transitions to host mode and is handled as an EI level exception. In addition, if a background interrupt is acknowledged, the cause of the interrupt (GMFEINT, GMEIINTn) for the guest mode, which is the cause of the generation, will not be cleared.

Because the interrupt controller generates the background interrupt, the restricted operating mode and guest partition ID in CPU may be changed during processing by the interrupt controller. **Table 4.13** indicates the relationship between acceptance of the background interrupt, the values of the guest partition ID assigned to the interrupt (GMFEINT, GMEIINTn) that is the cause of the background interrupt, the value of the guest partition ID in CPU and the restricted operating mode when the background interrupt occurs.

Table 4.13 Relationship between the restricted operating mode and acknowledgment of background interrupt

State when background interrupt occurs		Acknowledgment of background interrupt
Restricted operating mode	**The guest partition ID**	
host mode	—	Background interrupt is not acknowledged
guest mode	mismatch	If another condition is satisfied, the background interrupt is acknowledged.
	match	The background interrupt is not acknowledged.

(2) Examples of Using Background Interrupts and Notes

Even if a background interrupt is acknowledged, unlike the acknowledgement of an interrupt (GMFEINT, GMEIINTn) for guest mode, which is the cause of the generating the background interrupt, an EI level exception handler handled in host mode is executed. This document describes an example of using background interrupts.

Figure 4.4 shows an example of interrupt handling by background interrupt.

<1> By acknowledgment of the background interrupt, the restricted operating mode transitions to host mode.

<2> When a background interrupt occurs, it means that an interrupt request has been issued for a guest partition different from the guest partition that had been running until that time. In this CPU, by acknowledging background interrupt, it does not directly transition to the guest partition where the interrupt that is the cause of that generation should be processed, but processing is first passed to the exception handler of host mode. Therefore, even if a background interrupt is acknowledged, it can be left to the judgment of the virtualization software whether or not to transition to the guest partition to process the interrupt, which is the cause of that generation. For example, it is possible to decide whether guest partitions can be transitioned based on the number of acknowledgments of background interrupts in a certain period of time, the execution time of guest partitions to be transitioned, and so forth.
Once the virtualization software decides to transition to the guest partition to handle the interrupt which is the cause of generating the background interrupt, the context is switched from the guest partition that has been operating until then to the next guest partition. For details about context switching, see **4.1.7, Context switching**.

<3> The CPU transitions to the guest partition that should process the interrupt that is the cause of generating the background interrupt. In accordance with this, the interrupt controller generates interrupts (GMFEINT, GMEIINTn) that is the cause of the generating background interrupts.

<4> The interrupt that are no longer background interrupt (GMFEINT, GMEIINTn) is acknowledged. However, whether this interrupt can be acknowledged depends on the state of the interrupt acknowledgment condition. If this interrupt is not acknowledged and the execution time of the guest partition specified by the virtualization software has expired and the transition to another guest partition is made, this interrupt (GMFEINT, GMEIINTn) may generate a background interrupt again.

<5> Interrupt handling is performed within the guest partition.

<6> At the termination of the interrupt handling, the control is passed to the virtualization software by the HVTRAP instruction. However, the interrupt handler can not directly recognize that the interrupt handling was performed via the background interrupt. For details about how to provide such process, see **Section 4.1.9 (3), Application example of HVSB with background interrupt**.

<7> The virtualization software schedules the next guest partition to run. This example shows an example of transitioning to a guest partition whose execution was interrupted by a background interrupt.

<8> The guest partition that was interrupted by the background interrupt is re-executed.

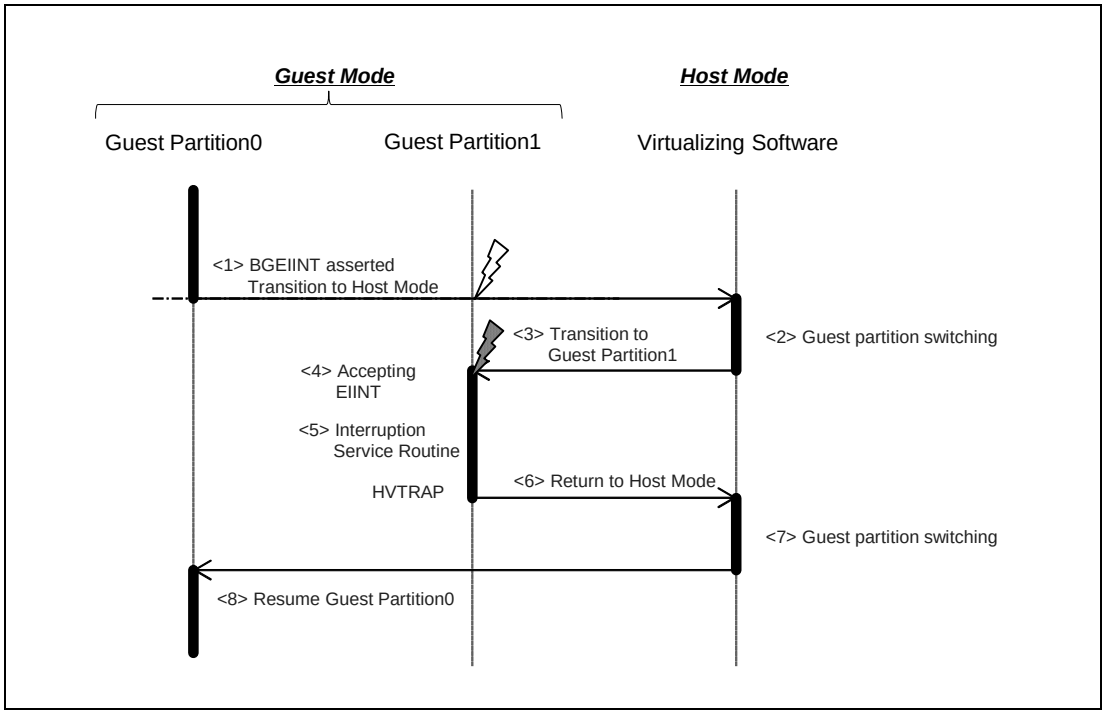


Figure 4.4 Example of Interrupt handling by background interrupt

If the virtualization software does not make a transition to the guest partition that should process the interrupt which is the cause of generating the background interrupt in <2> and makes the transition to

the guest partition where the execution was interrupted by the background interrupt, or if transition is made to another guest partition even though the interrupts are not acknowledged in <4>, a background interrupt will be generated again due to the same interrupt cause.

In order to prevent frequent occurrence of such background interrupts, as shown in <2>, the number of occurrences of background interrupts is managed, and if it exceeds a certain number, it is necessary to operate the interrupt controller to disable the background interrupt.

Also note that the following behavior may occur if BGFEINT and BGEIINT are used in different guest partitions at the same time.

- BGEIINT occurs after the transition to the guest partition at <2>, where the interrupt that generates BGFEINT should be processed.
- BGFEINT occurs after the transition to the guest partition at <2>, where the interrupt that generates BGEIINT should be processed.

In this case, neither GMEIINT nor GMFEIINT can be acknowledged in any guest partitions.

In order to prevent such a behavior, be sure to disable occurrence of BGEIINT by operating the interrupt controller at <2> before the transition to the guest mode when using BGFEINT and BGEIINT in different guest partitions at the same time.

(3) Application example of HVSB with background interrupt

The HVSB can be written with HV authority and can only be read by SV authority or UM authority. Virtualization software can use HVSB to convey information to software operating in guest mode.

For example, there is no direct method to recognize the guest partition that started the operation due to acknowledgment of the background interrupt, whether it is due to a background interrupt or another normal scheduling. It is possible for the virtualization software to recognize the state even in the guest partition by writing the value for identifying the state in the HVSB and then transitioning to the guest partition. Consequently, after acknowledging an interrupt which is the cause of generating background interrupt, HVSB is checked at the time of returning, and if it is a transition by background interrupt, the control is returned to the virtualization software by HVTRAP instruction without executing a return instruction.

Note that this is just an example of operation. How the software interface between virtualization software and software operating in guest mode is defined and how the HVSB is applied depends on the specification of virtualization software.

4.2 Operation when Acknowledging an Exception

Check whether each exception that is reported during instruction execution is acknowledged according to the priority. The procedure for exception-specific acknowledgment operation is shown below.

<1> Check whether the acknowledgment conditions are satisfied and whether exceptions are acknowledged according to their priority.

<2> Calculate the exception handler address according to the current HMPSW, GMPSW values*1

<3> For FE level exceptions which occur in the host mode or the guest mode and is handled in the host mode, the following processing is performed.

- Saving the PC to HMFEPC
- Saving the HMPSW to HMFEPSW
- Saving the PSWH to FEPSWH
- Storing the exception cause code in HMFEIC
- Updating the HMPSW*2
- Storing the exception handler address calculated in (2) in the PC, and then passing control to the exception handler.

<4> For EI level exceptions which occur in the host mode or guest mode and is handled in the host mode, the following processing is performed.

- Saving the PC to HMEIPC
- Saving the HMPSW to HMEIPSW
- Saving the HPSW to EIPSWH
- Storing the exception cause code in HMEIIC
- Updating the HMPSW*2
- Storing the exception handler address calculated in (2) in the PC, and then passing control to the exception handler.

<5> For FE level exceptions which occur in the guest mode and is handled in the guest mode, the following processing is performed.

- Saving the PC to GMFEPC
- Saving the GMPSW to GMFEPSW
- Storing the exception cause code in GMFEIC
- Updating the GMPSW*2
- Store the exception handler address calculated in (2) in the PC, and then pass control to the exception handler.

<6> For EI level exceptions which occur in the guest mode and is processed in the guest mode, the following processing is performed.

- Saving the PC to GMEIPC
- Saving the GMPSW to GMEIPSW
- Storing the exception cause code in GMEIIC
- Updating the GMPSW*2
- Storing the exception handler address calculated in (2) in the PC, and then passing control to the exception handler.
- When the exception is an interrupt that uses the register bank, save the context automatically*3.

Note 1. For details, see **Section 4.4, Exception Handler Address**.

Note 2. For the values to be updated, see **Table 4.1, Exception Cause List**.

Note 3. For details on register banks, see **Section 4.5, Register Bank Function**.

Figure 4.5 shows the steps <1> to <4>.

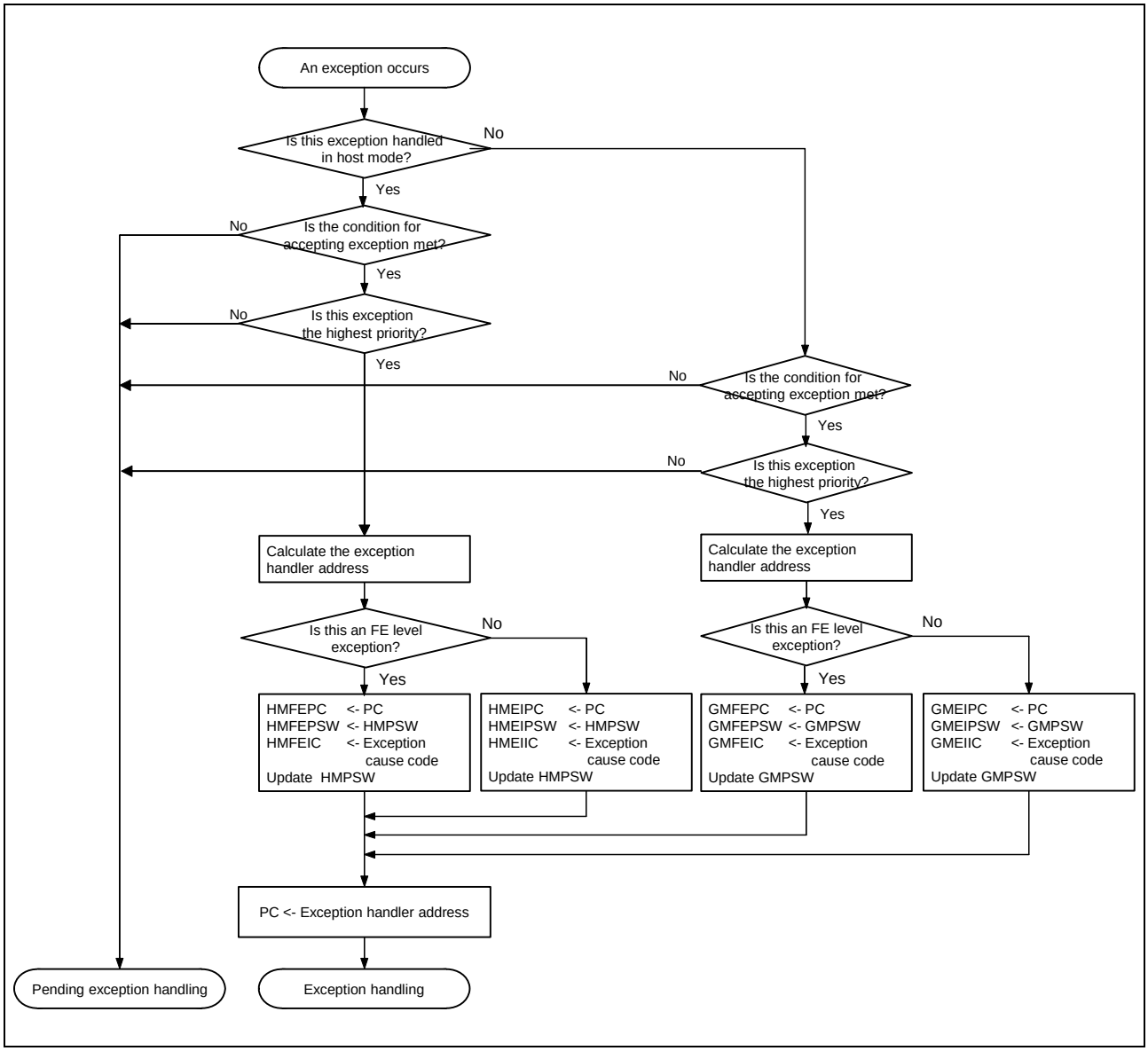


Figure 4.5 Operation When Acknowledging an Exception

4.2.1 Special Operations

(1) EP Bit of PSW Register

If an interrupt is acknowledged, the PSW.EP bit is cleared to 0. If an exception other than an interrupt is acknowledged, the PSW.EP bit is set to 1.

The operation of the CPU when executing the EIRET and FERET instructions depends on the state of the EP bit. If the EP bit is cleared to 0, the CPU notifies the external interrupt controller of the termination of the exception handling routine. This function is necessary to properly control the request flag in the interrupt controller and other resources upon return from the interrupt. When the EP bit is cleared to 0, and the EIRET instruction is executed, the bit with the highest priority (0 is the highest) among the bits set to 1 in ISPR.ISP15 to ISPR.ISP0 is cleared to 0.

To return from an interrupt, be sure to execute the return instruction with the EP bit cleared to 0.

(2) Coprocessor Unusable Exception

For coprocessor unusable exceptions, the opcodes that cause the exception depend on the status of the CU bit of the PSW register.
When a coprocessor is not included in the product or it is not usable, if an attempt is made to execute a coprocessor instruction corresponding to the coprocessor, a coprocessor unusable exception (UCPOP) immediately occurs. If an LDSR or STSR instruction attempts to access a system register of the coprocessor, a coprocessor unusable exception (UCPOP) immediately occurs too.
For details, see **Section 2.4.3, Coprocessor Unusable Exceptions**.

(3) Reserved Instruction Exception

If an opcode that is reserved for future function extension and for which no instruction is defined is executed, a reserved instruction exception (RIE) occurs.

The opcode that always generates a reserved instruction exception is defined as the RIE instruction.

(4) Reset

Reset is performed in the same way as exception handling, but it is not regarded as EI level exception or FE level exception. The reset operation is the same as that of an exception without acknowledgment conditions, but the value of each register is changed to the value after reset. In addition, returning to the original program from the reset is not possible.

All exceptions that have occurred at the same time as CPU initialization are canceled and not acknowledged even after CPU initialization.

For details, see **Section 8, Reset**.

(5) The register bank function is unusable in host mode

With the interrupt (EIINTn) handled in host mode, the register bank function is unusable. Even if the interrupt request (EIINTn) with the priority corresponding to the bit set to 1 of RBCR0.BE is acknowledged, the register bank function is unusable. **Table 4.14** shows whether the register bank function can be used when an interrupt with priority i (EIINTn, GMEIINTn) is acknowledged.

Table 4.14 Whether the register bank function can be used when an interrupt (EIINTn, GMEIINTn) with priority(i) is acknowledged

The restricted operating mode in which interrupts are handled	RBCR0.BE[*i*]	Whether the register bank function can be used
Host mode	0	Unusable
	1	Unusable
Guest mode	0	Unusable
	1	Usable

4.3 Return from Exception Handling

To return from exception handling, execute the return instruction (EIRET or FERET) corresponding to the relevant exception level.

When a context has been saved, such as to a stack, the context must be restored before executing the return instruction. When execution is returned from an unrestorable exception, the status before the exception occurred in the original program cannot be restored. Consequently, the execution result might differ from that when the exception does not occur.

The EIRET instruction is used to return from EI level exception handling and the FERET instruction is used to return from FE level exception handling.

When the EIRET or FERET instruction is executed, the CPU performs the following processing and then passes control to the return PC address.

<1> If the PSWH.GM is set to 0, the host context register is used, and if the PSWH.GM is set to 1, the guest context register is used to judge processing.

<2> If the PSWH.GM bit is set to 0 and the HMPSW.EP bit is set to 0, the CPU notifies the interrupt controller of the termination of the exception routine.
If the PSWH.GM bit is set to 1 and the GMPSW.EP bit is set to 0, the CPU notifies the interrupt controller of the termination of the exception routine.

<3> When the EIRET instruction is executed while PSWH.GM = 0, if PSW.EP = 0 and INTCFG.ISPC = 0, the CPU updates the ISPR register. When the EIRET instruction is executed while PSWH.GM = 1 or when the FERET instruction is executed, the CPU does not update the ISPR register.

<4> When the EIRET instruction is executed while PSWH.GM = 0, the value set to the return PC, HMPSW, and PSWH are loaded from the HMEIPC, HMEIPSW, and EIPSWH registers, respectively.
When the EIRET instruction is executed while PSWH.GM = 1, the value set to the return PC and GMPSW are loaded from the GMEIPC and GMEIPSW registers, respectively.
When the FERET instruction is executed while PSWH.GM = 0, the value set to the return PC, HMPSW, and PSWH are loaded from the HMFEPC, HMFEPSW, and FEPSWH registers, respectively.
When the FERET instruction is executed while PSWH.GM = 1, the value set to the return PC and GMPSW are loaded from the GMFEPC and GMFEPSW registers, respectively.

<5> Transit to the taken PC address.

Figure 4.6 shows the flow for returning from exception handling using the EIRET or FERET instruction.

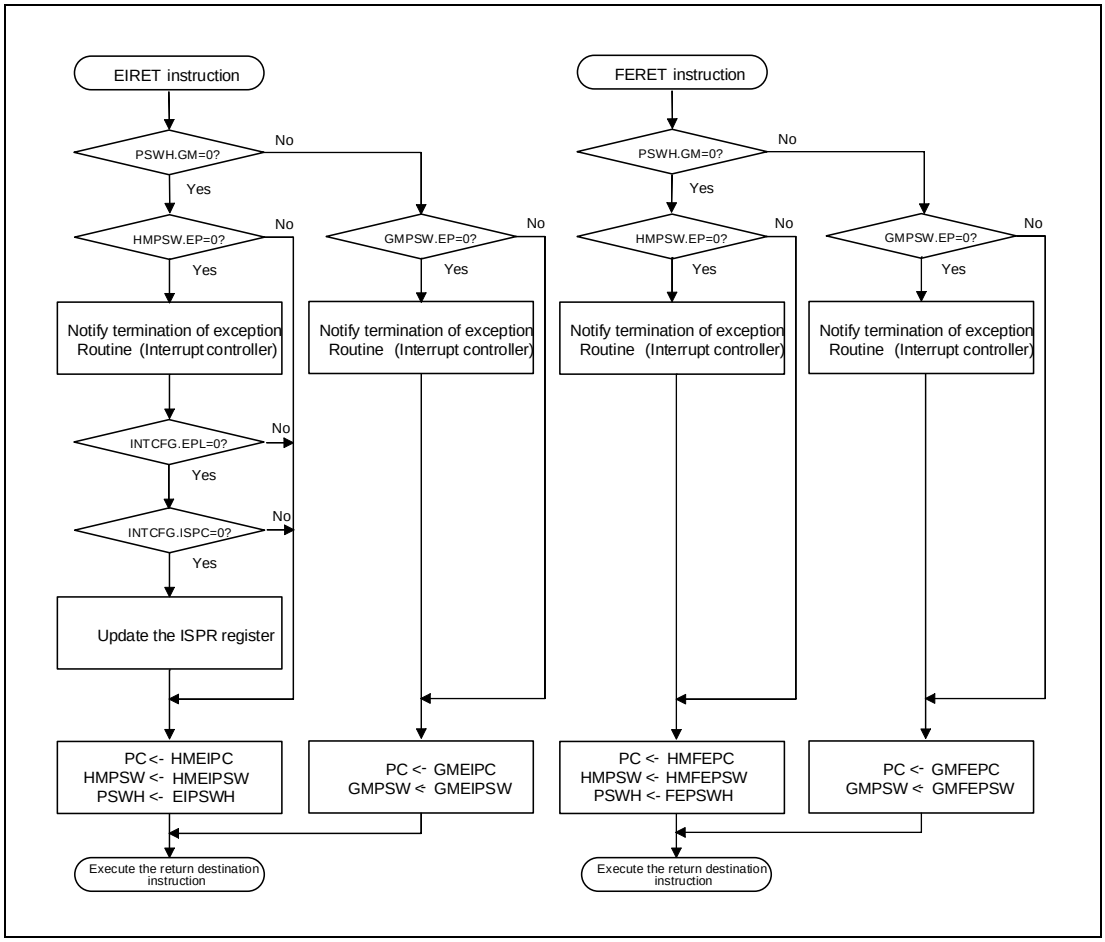


Figure 4.6 Return Instruction-Based Exception Return Flow

4.4 Exception Handler Address

For this CPU, the exception handler address used for execution during reset input, exception acknowledgment, or interrupt acknowledgment can be changed according to the settings.

4.4.1 Resets, Exceptions, and Interrupts

For resets and exceptions handled in the host mode, the exception handler address is determined by using the direct vector method, in which the reference point of the exception handler address can be changed by using the HMPSW.EBV bit, RBASE register, and HMEBASE register. For exception handled in guest mode, the reference point of the exception handler address is changed by the GMEBASE register. Since the reference point of the exception handler address can be set independently for the host mode and guest mode, the exception handler address unique to the guest partition can be used.

For interrupts, the direct vector method and table reference method can be specified. If the table reference method is selected, execution can branch to the address indicated by the exception handler table allocated in the memory.

(1) Direct Vector Method

For the exception handled in host mode, this CPU use the results of adding offset address in **Table 4.15, Selection of Base Register/Offset Address** to the base address indicated by the RBASE register or HMEBASE register as the exception handler address. For CPU exceptions handled in guest mode, this CPU uses the GMEBASE register as the base address.

For the exception handled in host mode, whether to use the RBASE register or the HMEBASE register as the base address is selected according to the HMPSW.EBV. If the HMPSW.EBV bit is set to 1, the value of the HMEBASE register is used as the base address If the bit is cleared to 0, the value of the RBASE register is used as the base address For exceptions handled in guest mode, since the GMPSW.EBV bit is always set to 1, the value of the GMEBASE register is used as the base address.

However, reset input always refers to the RBASE register.

In addition, user interrupts (EIINT*n*, GMEIINT*n*) refer to the RINT bit of the selected base register, and reduce the offset address according to the value of the bit. If the RBASE.RINT bit, HMEBASE.RINT bit or GMEBASE.RINT bit is set to 1, all user interrupts are handled using an offset address of 100_H. If the bit is cleared to 0, the offset address is determined according to **Table 4.15, Selection of Base Register/Offset Address**.

Figure 4.7 shows the flow of selecting the method of generating a handler address, **Figure 4.8** shows the flow of generating a handler address for the Direct Vector Method.

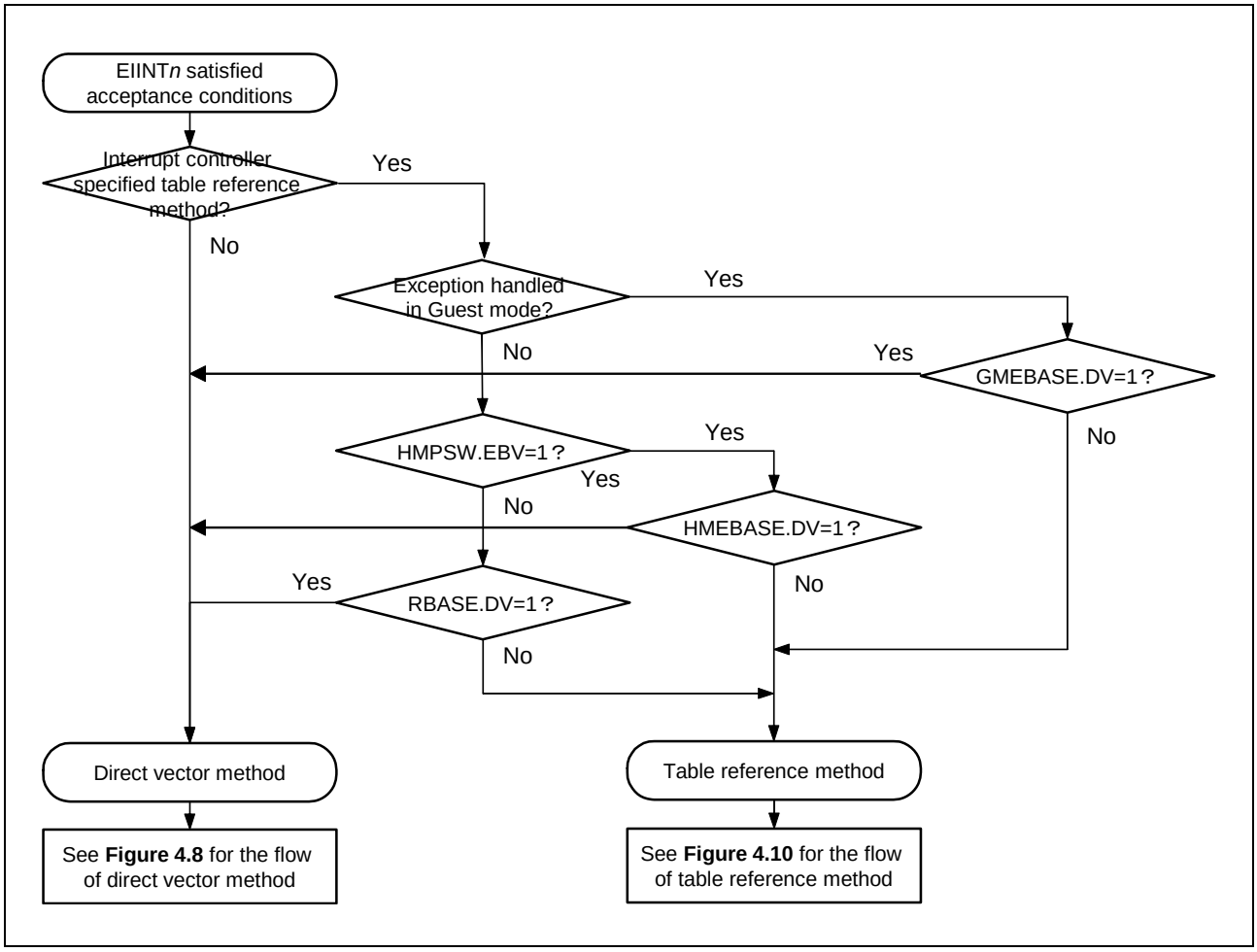
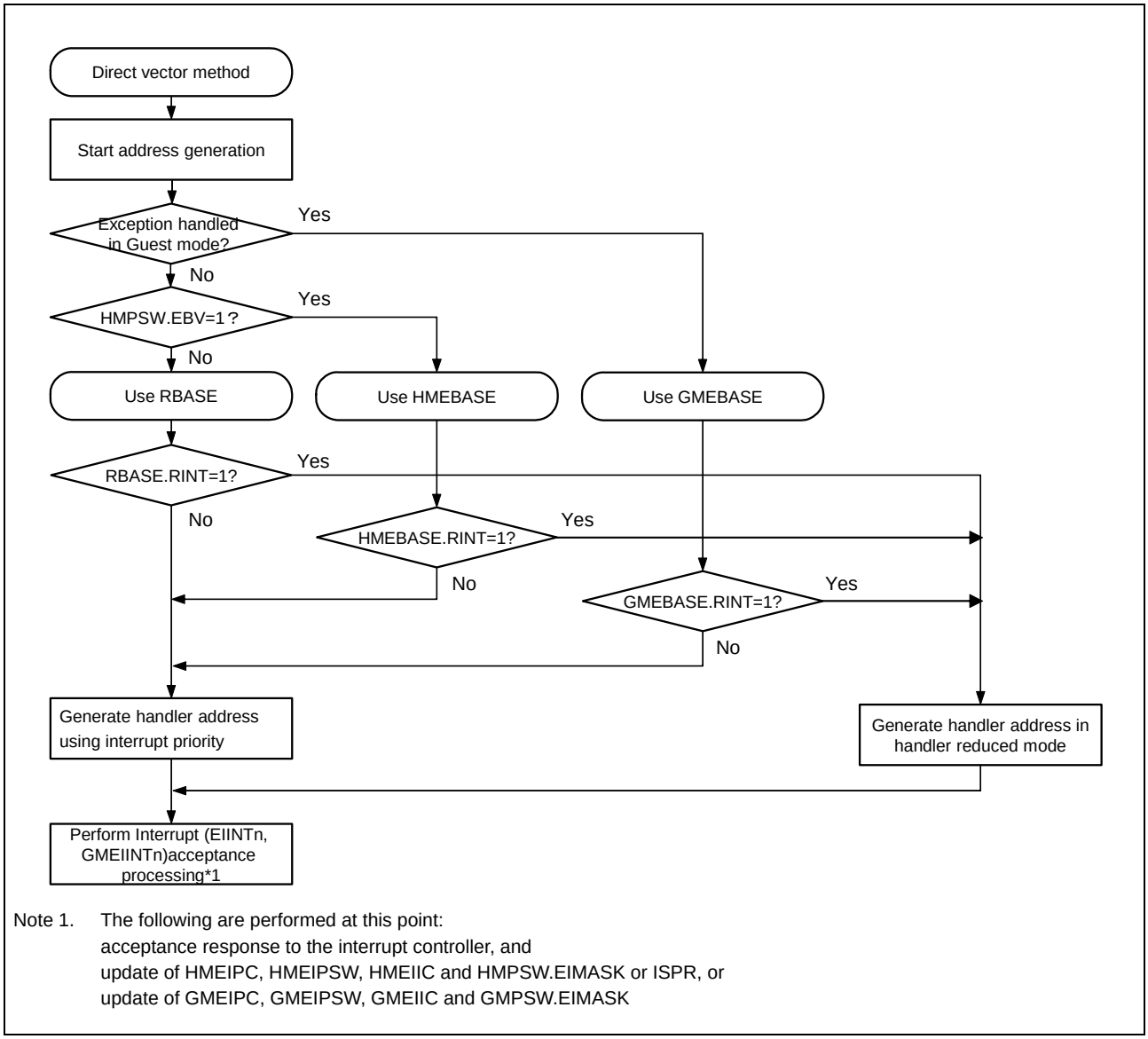
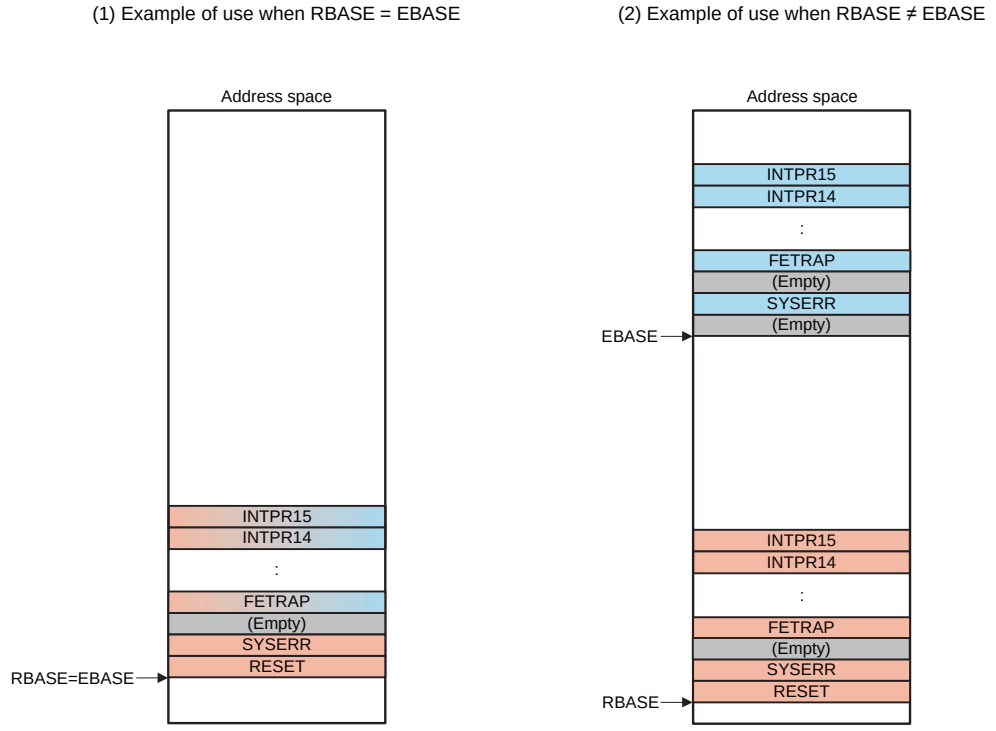


Figure 4.7 The Flow of Selecting the Method of Generating a Handler Address



Note 1. The following are performed at this point:
acceptance response to the interrupt controller, and
update of HMEIPC, HMEIPSW, HMEIIC and HMPSW.EIMASK or ISPR, or
update of GMEIPC, GMEIPSW, GMEIIC and GMPSW.EIMASK

Figure 4.8 The flow of Selecting the Method of Generating a Handler Address



Note 1. INTPRx is the same as EIINT*n* (priority x) in **Table 4.15**.

Figure 4.9 Example of using the Direct Vector Method in exception handled in Host Mode

The table below shows how base register is selected and offset address for each exception. The value of the HMPSW, GMPSW register used to determine the exception handler address is the value after being updated due to the acknowledgment of an exception.

Table 4.15 Selection of Base Register/Offset Address

	Base Register Selection			Offset Address Selection	
	Exception handled in Host Mode		Exception handled in Guest Mode		
Exception Cause	HMPSW.EBV = 0	HMPSW.EBV = 1	GMPSW.EBV=1	RINT = 0	RINT = 1
RESET	RBASE	None*1	None*2	000_H	000_H
SYSERR		HMEBASE	GMEBASE*3	010_H	010_H
HVTRAP			None*2	020_H	020_H
FETRAP			GMEBASE	030_H	030_H
TRAP0				040_H	040_H
TRAP1				050_H	050_H
RIE				060_H	060_H
FPE/FXE				070_H	070_H
UCPOP				080_H	080_H
MIP/MDP			GMEBASE*3	090_H	090_H
PIE			GMEBASE	$0A0_H$	$0A0_H$
(R.F.U.)				$0B0_H$	$0B0_H$
MAE				$0C0_H$	$0C0_H$
BGFEINT/BGEIINT			None*2	$0D0_H$	$0D0_H$
FENMI			None*2	$0E0_H$	$0E0_H$
FEINT/GMFEINT			GMEBASE	$0F0_H$	$0F0_H$
EIINTn, GMEIINTn (priority 0)				100_H	100_H
EIINTn, GMEIINTn (priority 1)				110_H	
EIINTn, GMEIINTn (priority 2)				120_H	
EIINTn, GMEIINTn (priority 3)				130_H	
EIINTn, GMEIINTn (priority 4)				140_H	
EIINTn, GMEIINTn (priority 5)				150_H	
EIINTn, GMEIINTn (priority 6)				160_H	
EIINTn, GMEIINTn (priority 7)				170_H	
EIINTn, GMEIINTn (priority 8)				180_H	
EIINTn, GMEIINTn (priority 9)				190_H	
EIINTn, GMEIINTn (priority 10)				$1A0_H$	
EIINTn, GMEIINTn (priority 11)				$1B0_H$	
EIINTn, GMEIINTn (priority 12)				$1C0_H$	
EIINTn, GMEIINTn (priority 13)				$1D0_H$	
EIINTn, GMEIINTn (priority 14)				$1E0_H$	
EIINTn, GMEIINTn (priority 15 or less*4)				$1F0_H$	

Note 1. Since it is an exception generated to update HMPSW.EBV to 0, it is not used.

Note 2. Since it is an exception generated to update PSWH.GM to 0, it is not used

Note 3. This case is processed in guest mode by setting of the corresponding bit of GMCFG. If these exceptions are handled in host mode, the handler address is determined by HMPSW.EBV and RBASE or HMEBASE.

Note 4. Interrupt (EIINTn, GMEIINTn) whose priority level is less than or equal to 16 uses the same offset address as the interrupt (EIINTn, GMEIINTn) with priority level 15. Since the same interrupt handler is used, software process is necessary for recognition of interrupt factor. By this constraint, the increasing of memory area

which is used for interrupt handler by direct vector method can be avoided even if interrupt priority level extension function is enabled.

The user interrupt offset address reduction function is used to reduce the memory size required by the exception handler for specific operating modes of the system. The main purpose of this is to minimize the amount of memory consumed in operating modes that use only the minimum functionality, for example, during system maintenance and diagnosis.

(2) Table Reference Method

In the direct vector method, there is one user-interrupt exception handler for each interrupt priority, and user-interrupts with the same priority branch to the same interrupt handler, but some users might want to use code areas that differ from the start time for each interrupt handler.

This CPU defines a table reference method to accommodate to such uses.

Figure 4.10 shows the flow of generating a handler address for the table reference method.

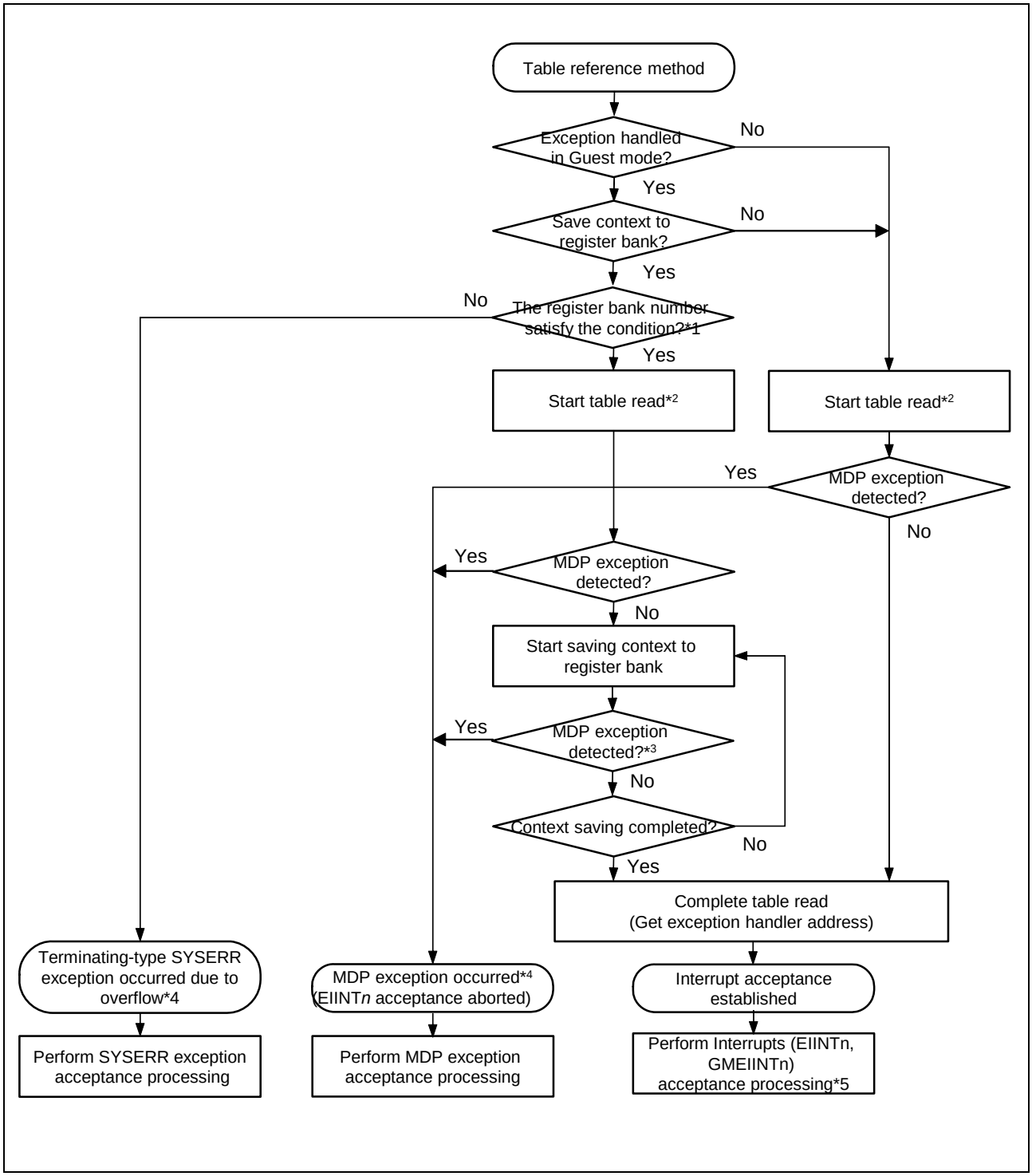


Figure 4.10 Flow of Generating a Handler Address of Table Reference Method (1/2)

Note 1.	For automatic context saving onto the register bank, the operation condition judgment is necessary. For details, see **Figure 4.12**.
Note 2.	The PC of the instruction interrupted by the interrupt and the value of the PSW at that time are saved to the HMEIPC and the HMEIPSW or GMEIPC and GMEIPSW respectively.
Note 3.	Detection of an MDP exception is performed every time an individual context is saved. An MDP exception occurs when an MDP violation is detected when saving an individual context rather than after all context saving is completed.
Note 4.	The PC of the instruction interrupted by the interrupt and the value of the PSW at that time are saved to the HMFEPC and the HMFEPSW or GMFEPC or GMFEPSW respectively. These values are the same as the values saved to the HMEIPC and the HMEIPSW or GMEIPC and GMEIPSW at the timing of Note 2. Therefore, when returning from the exception handler, it is possible to return exactly to the interrupted instruction. In addition, because no acceptance response has been reported to the interrupt controller, unless the interrupt request is canceled in the exception handler, the interrupt request will remain held and interrupt acceptance processing will start again.
Note 5.	Notification of the acceptance to the interrupt controller and the update of the HMEIIC and HMPSW.EIMASK or ISPR, or GMEIIC and GMPSW.EIMASK are performed at this timing.

Figure 4.10 Flow of Generating a Handler Address of Table Reference Method (2/2)

<1> In any of the following cases, the exception handler address is determined by using the direct vector method.

- When the interrupt channel setting is not the table reference method
- When in interrupts handled in host mode, HMPSW.EBV = 0 and RBASE.DV = 1
- When in interrupts handled in host mode, HMPSW.EBV = 1 and HMEBASE.DV = 1
- When In interrupts handled in guest mode, GMEBASE.DV = 1

<2> In cases other than <1>, the table read position is calculated.
Exception handler address read position handled in host mode
HMINTBP register + channel number × 4 bytes
Exception handler address read position handled in guest mode
GMINTBP register + channel number × 4 bytes

<3> Word data is read starting at the exception handler address read position calculated in <2>.

<4> Word data read in <3> is used as the exception handler address.

<5> The acceptance of the interrupt is confirmed at the point when the exception handler address is obtained. In addition to the exception acceptance processing shown in **Figure 4.5, Operation When Acknowledging an Exception**, the CPU returns an acceptance response to the interrupt controller and updates the ISPR or HMPSW.EIMASK or GMPSW.EIMASK.

CAUTIONS

1. **For details about the interrupt channel settings, see the hardware manual of the product used.**
2. **There are cases in which a memory protection exception (MDP) occurs while reading word data from the exception handler address read position depending on the memory protection settings. In such a case, the acceptance of the interrupt is temporarily cancelled. Since no acceptance response is returned to the interrupt controller, the interrupt request is held pending and becomes acceptable again when execution is returned from the memory protection exception handling.**
3. **If a memory protection exception occurs during a word data read from the exception handler address read position, the PC of the instruction that is interrupted by the interrupt is saved in the HMFEPC or GMFEPC. Since the acceptance of the interrupt is cancelled, the value of the HMPSW or GMPSW established when the interrupt occurred (which should have been saved in the HMEIPSW or GMEIPSW) is saved in the HMFEPSW or GMFEPSW.**

Exception handler address read positions corresponding to interrupt channels and the placement of the interrupt handler address table in memory are shown below. Since the interrupt handler address table can be set independently for the host mode and guest mode, the interrupt handler address unique to the guest partition can be used.

Table 4.16 Exception Handler Address Read Position

Type	Exception Handler Address Read Position	
	Interrupt handled in host mode	Interrupt handled in guest mode
EIINT, GMEIINT interrupt channel 0	HMINTBP + 0 × 4	GMINTBP + 0 × 4
EIINT, GMEIINT interrupt channel 1	HMINTBP + 1 × 4	GMINTBP + 1 × 4
:	:	:
EIINT, GMEIINT interrupt channel 2046	HMINTBP + 2046 × 4	GMINTBP + 2046 × 4
EIINT, GMEIINT interrupt channel 2047	HMINTBP + 2047 × 4	GMINTBP + 2047 × 4

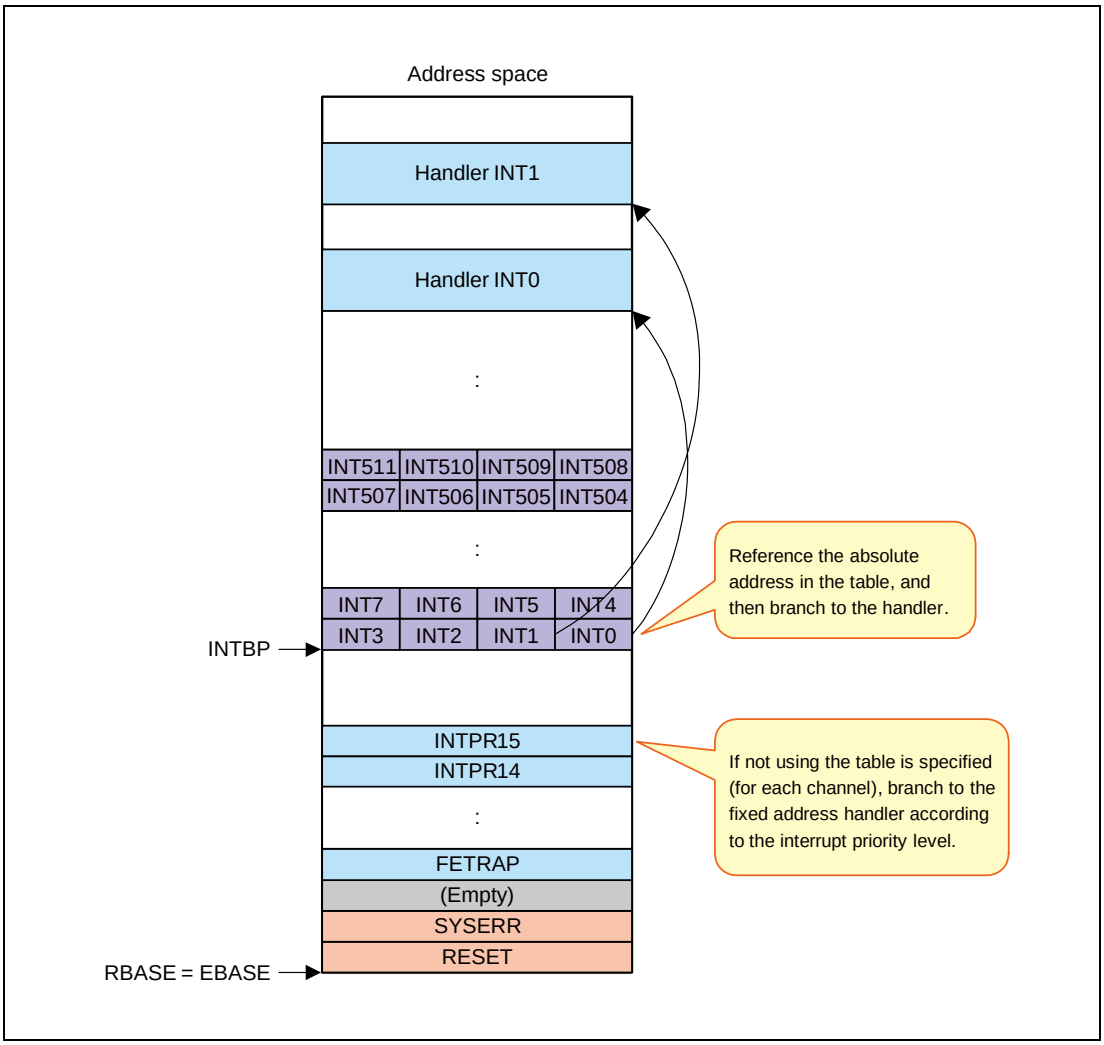


Figure 4.11 Image when using Table Reference Method in interrupts handled in host mode

For details about the exception handler address selection method settings for each interrupt channel, see the hardware manual of the product used.

4.4.2 System Calls

For details, see the “CPU” section in the hardware manual of the product used.

4.5 Register Bank Function

This CPU provides the register bank function that automatically saves the context when it accepts an interrupts (GMEIINT*n*). Since the saving of the context proceeds in parallel with the interrupt acceptance processing, this function enables the CPU to make high-speed interrupt response.

4.5.1 Outline of the Register Bank Function

The register bank function that this CPU provides has the following features:

- Automatically saves the context upon acceptance of an interrupt (GMEIINTn) *1meeting certain conditions (see **Section 4.5.2, Automatic Context Saving**).
- The destination of the context saving is not dedicated memory but an area of ordinary memory installed in this CPU (specified by the RBIP register). The context to be saved can be selected from two groups (specified by the RBCR0.MD bit).
- Supports a maximum of 64 levels of multiple interrupts (the RBNR.BN bits indicate the number of accepted interrupts).
- The context is restored from the register bank by executing the RESBANK instruction (no automatic restoration function).

Note 1. In host mode, the register bank function cannot be used when an interrupt(EIINTn) is acknowledged.

4.5.2 Automatic Context Saving

Automatic context saving is carried out when the following conditions are satisfied for an interrupt request (GMEIINTn) notified:

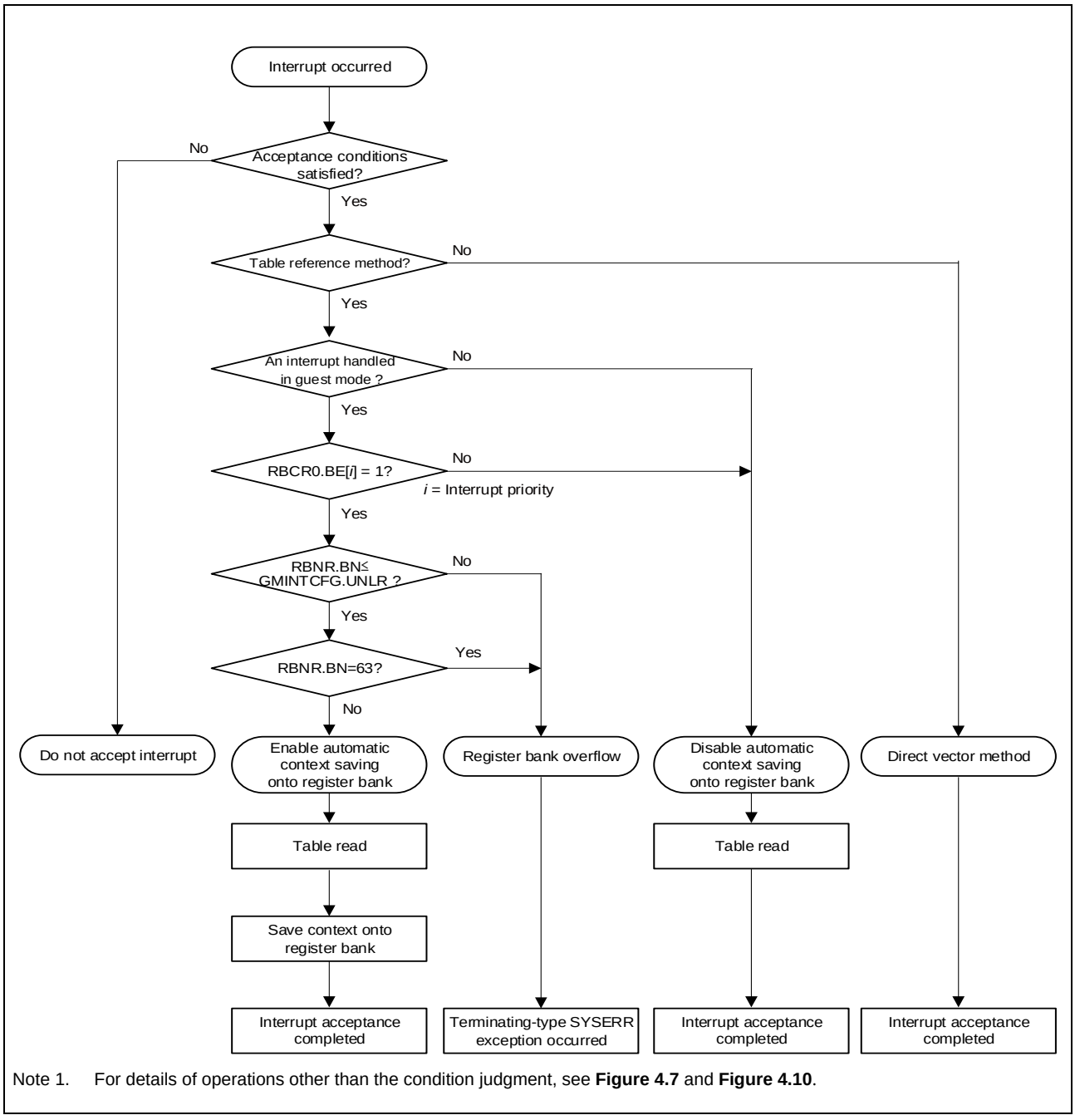
- Table reference method is specified for the interrupt request*1.
- The use of the register bank is specified by the RBCR0.BE[*i*] bit which is associated with the priority (*i*) of the interrupt request*2 (the automatic context saving onto the register bank is enabled).
- The RBNR.BN bits have a value no greater than the value of GMINTCFG.ULNR*3 (check the number of the register banks in use).

Note 1. See the hardware manual of the product used for the procedure to specify the table reference method.
Note 2. Availability of context auto saving for all interrupts (EIINTn) whose priority level is less than or equal to 16 is specified by RBCR0.BE[15].
Note 3. If the value of RBNR.BN is bigger than GMINTCFG.ULNR or the value of RBNR.BN is 63, the context auto saving is not done. In this case, SYSERR exception whose exception cause code (lower 16bits) is $1C_H$ is issued.

Automatic context saving is not carried out if the table reference method is not specified for the interrupt request (GMEIINT*n*) or the use of the register bank is not specified by the RBCR0.BE[*i*] bit associated with the interrupt priority (*i*).

In addition, no automatic context saving is carried out if the value of the RBNR.BN bits are greater than GMINTCFG.ULNR, or if the value of RBNR.BN is 63. In such a case, it is assumed that an unexpected interrupt acceptance processing is being performed and a terminating-type SYSERR exception is notified for error processing.

Figure 4.12 shows the flow of checking the conditions for automatic context saving.



Note 1. For details of operations other than the condition judgment, see **Figure 4.7** and **Figure 4.10**.

Figure 4.12 Flow of Checking the Conditions for Automatic Context Saving

(1) Automatic Context Saving

Shown below is the processing of automatic context saving onto the register bank.

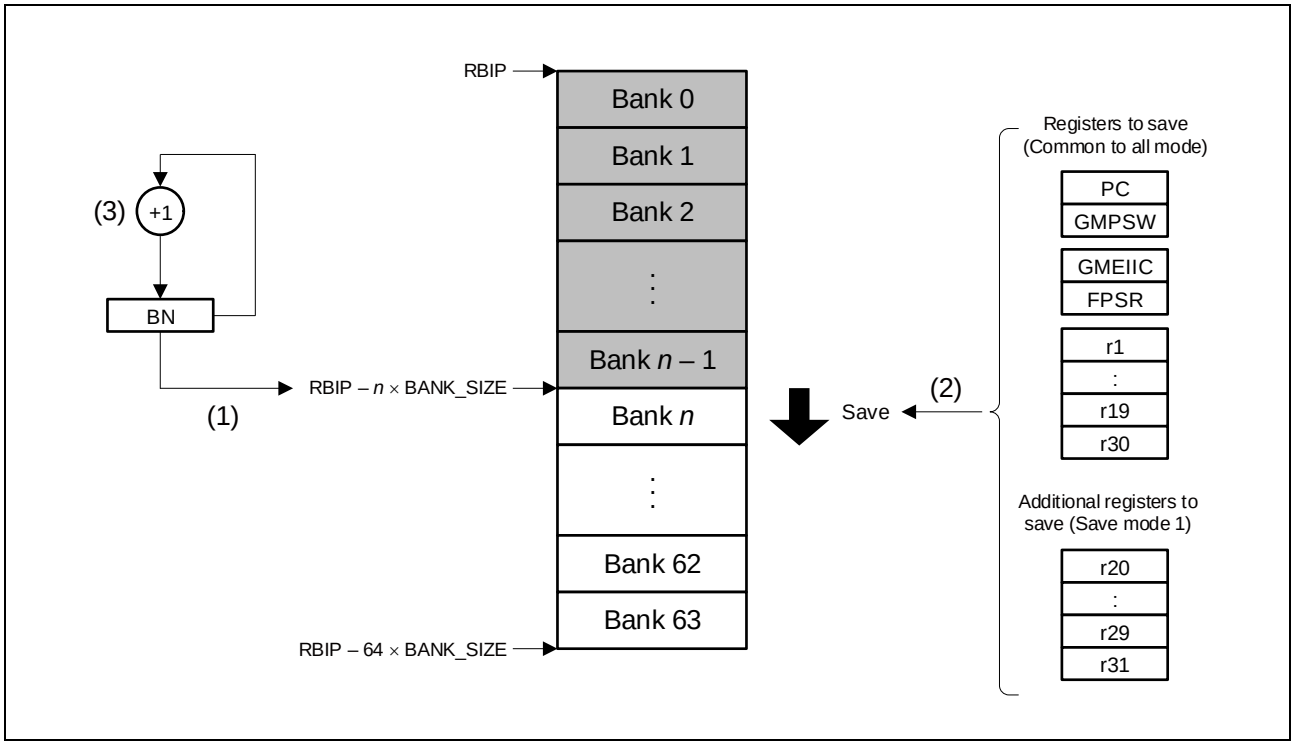


Figure 4.13 Processing of Automatic Context Saving

1. Calculate the start address of register bank n on which the context is to be saved based on the value of the register bank initial pointer (RBIP) and the value (*n*) of the BN bits of the register bank number register (RBNR) established upon acceptance of the interrupt. The size (BANK_SIZE) of the register bank differs depending on the save mode that is selected.
   Start address:
   RBIP – *n* × BANK_SIZE
   BANK_SIZE:
   Save mode 0: 60_H
   Save mode 1: 90_H
2. Save the target registers onto the register bank n specified by the RBNR.BN bits according to the save mode specified by the RBCR0.MD bit*1. **Table 4.17** shows a list of the registers to save, their addresses, and save order.
3. Increment the value of the RBNR.BN bits by 1. This terminates the register bank saving processing*2 and initiates the execution of the interrupt handler*3.

Note 1. MDP exceptions (terminating-type) might be caused by memory accesses made during the register bank saving processing. The other types of interrupts and exceptions cannot be accepted.

Note 2. The saving of the registers into the memory area specified as the register bank may not be completed. To ensure the completion of register saving, this CPU can employ a procedure similar to the one with the ordinary store instruction. For details, see **Section 7.2, Guaranteeing the Completion of Store Instruction**.

Note 3. Exceptions of higher priorities can be accepted after the saving onto the register bank is finished.

Table 4.17 List of Registers to Save, Addresses, Save Order, and Restore Order

Addresses RBIP − n × BANK_SIZE		Save mode 0 (MD = 0)	Save mode 1 (MD = 1)	Save Order *1, *2	Restore Order *1, *3
	−04_H	PC	PC	1	24 / 35
	−08_H	GMPSW	GMPSW	2	23 / 34
	−$0C_H$	GMEIIC	GMEIIC	3	22 / 33
	−10_H	FPSR	FPSR	4	21 / 32
	−14_H	r1	r1	5	20 / 31
	−18_H	r2	r2	6	19 / 30
	−$1C_H$	r3	r3	7	18 / 29
	−20_H	r4	r4	8	17 / 28
	−24_H	r5	r5	9	16 / 27
	−28_H	r6	r6	10	15 / 26
	−$2C_H$	r7	r7	11	14 / 25
	−30_H	r8	r8	12	13 / 24
	−34_H	r9	r9	13	12 / 23
	−38_H	r10	r10	14	11 / 22
	−$3C_H$	r11	r11	15	10 / 21
	−40_H	r12	r12	16	9 / 20
	−44_H	r13	r13	17	8 / 19
	−48_H	r14	r14	18	7 / 18
	−$4C_H$	r15	r15	19	6 / 17
	−50_H	r16	r16	20	5 / 16
	−54_H	r17	r17	21	4 / 15
	−58_H	r18	r18	22	3 / 14
	−$5C_H$	r19	r19	23	2 / 13
	−60_H	r30	r20	24	1 / 12
	−64_H	*4	r21	25	11
	−68_H		r22	26	10
	−$6C_H$		r23	27	9
	−70_H		r24	28	8
	−74_H		r25	29	7
	−78_H		r26	30	6
	−$7C_H$		r27	31	5
	−80_H		r28	32	4
	−84_H		r29	33	3
	−88_H		r30	34	2
	−$8C_H$		r31	35	1
	−90_H		*5		
	−94_H		*6		

Note 1. Register saving and restoration start sequentially at sequence number 1.

Note 2. The target registers to be saved in save mode 0 (RBCR0.MD = 0) are the registers up to r30 with a save sequence number of 24.

Note 3. The restoration sequence numbers shown on the left side are for save mode 0 and those shown on the right side are for save mode 1.

Note 4. In save mode 0, the next bank (n + 1) starts from this address.

Note 5. In save mode 1, no register is saved in this address. This address is not used.

Note 6. In save mode 1, the next bank (n + 1) starts from this address.

(2) Suppressing the Update of the GMPSW.ID Bit

When an interrupt (GMEIINTn) is accepted in the normal state, the GMPSW.ID bit is set to 1 and interrupts of the same EI level cannot be accepted unless the acceptance of interrupts is enabled explicitly with the EI instruction.

For interrupts that make use of the register bank (satisfying the automatic context saving conditions), on the other hand, by clearing the RBCR1.NC[*i*] bit associated with the interrupt priority (*i*) to 0, the GMPSW.ID bit keeps its value 0 without being set to 1 when an interrupt is accepted. This makes it possible to accept interrupts of higher priorities without software intervention after the end of automatic context saving. Since necessary registers are automatically saved, the CPU can return to the original interrupt handling precisely.

NOTE

All interrupts (EIINTn) whose priority level is less than or equal to 16 is specified by the value of PSW.ID bit after interrupt acknowledge defined by RBCR1.NC[15].

4.5.3 Context Restoration

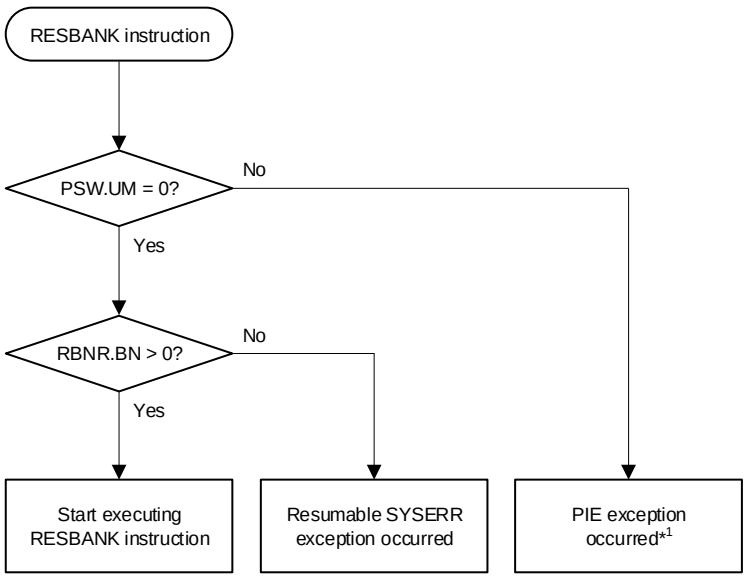
The context automatically saved onto the register bank needs to be restored before returning from the interrupt handling to the original program. Although the context saving onto the register bank is automatically done, its restoration needs to be accomplished explicitly by software executing the RESBANK instruction.

(1) Conditions for Executing the RESBANK Instruction

The RESBANK instruction must be executed to restore the context from the register bank. The RESBANK instruction is a supervisor-privileged instruction. A PIE exception will occur if it is executed in user mode.

A resumable-type SYSERR exception will occur if the RESBANK instruction is executed when the value of the RBNR.BN bits are 0. 0 in the RBNR.BN bits means that no context has automatically been saved onto the register bank. Accordingly, an invalid value will be restored if the RESBANK instruction is executed in such case.

The above-mentioned operating conditions are summarized in **Figure 4.14**.



Note 1. The process for detecting MIP and other exceptions that are not caused directly by the RESBANK instruction is omitted.

Figure 4.14 Flow of Checking the Conditions for Executing the RESBANK Instruction

Additionally, that register bank functions can not be used for interrupts handled in host mode, but the RESBANK instruction can be executed while the restricted operating mode is host mode. However, in this case, the context saved in the register bank by an interrupt handled in guest mode is restored as the context for host mode. Since such a usage is not assumed, do not execute the RESBANK instruction in host mode.

(2) Context Restoration

The figure below shows the processing of the RESBANK instruction for restoring the context from the register bank.

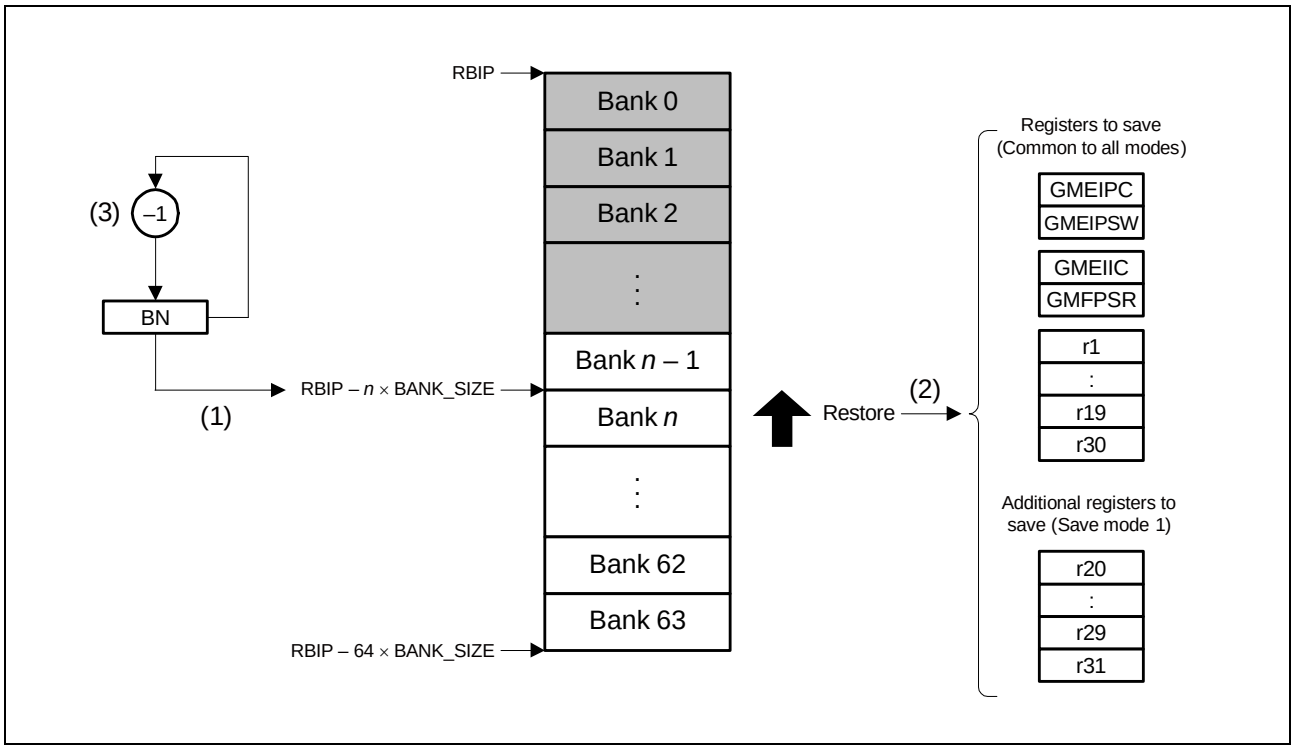


Figure 4.15 Processing of Context Restoration

1. Calculate the start address of register bank n-1 from which the context is to be restored based on the value of the register bank initial pointer (RBIP) and the value (n) of the BN bits of the register bank number register (RBNR) established upon acceptance of the interrupt. Unlike in the case of saving the context, the start address to be referenced when restoring the context is the lower-limit side of the register bank addresses. The size (BANK_SIZE) of the register bank differs depending on the save mode that is selected.
   Start address:
   RBIP – n × BANK_SIZE
   BANK_SIZE:
   Save mode 0: 60_H
   Save mode 1: 90_H
2. Restore the context from the register bank n-1 in the target registers according to the save mode specified by the RBCR0.MD bit*1. **Table 4.17** shows a list of the registers to restore, their addresses, and restore order.
3. Decrement the value of the RBNR.BN bits by 1*2. This terminates the register bank restoration processing and completes the execution of the RESBANK instruction*3.

Note 1. The CPU can accept terminating-type exceptions while restoring the context from the register bank. MDP exceptions (resumable-type) might be caused by memory accesses made during the register bank restoration processing.

Note 2. When the CPU accepts an exception before the value of RBNR.BN is updated, it stops the execution of the RESBANK instruction even if the restoration of all registers is not yet

completed. In this case, though there are some registers for which restoration has been completed, the CPU cannot know which registers have been restored. Since the return PC of the exception is the PC of this RESBANK instruction, the CPU can re-execute precisely the RESBANK instruction if none of the resources related to the RESBANK instruction are altered during the exception handling.

Note 3. After the RESBANK instruction is executed, execute the EIRET instruction to return from the interrupt handler.

4.6 List of Memory Access Exceptions

Table 4.18 shows a list of instructions and exceptions that make memory accesses and the exceptions that can be detected. Exceptions identified by the symbol "✓" are detected for instructions and exceptions that access memory. Exceptions identified by the symbol "×" are not detected.

Table 4.18 List of Memory Access Exceptions

Instruction/Exception	MAE	MDP	Instruction/Exception	MAE	MDP
SLD.B	×	✓	PREPARE	×	✓
SLD.BU	×	✓	DISPOSE	×	✓
SLD.H	✓	✓	PUSHSP	×	✓
SLD.HU	✓	✓	POPSP	×	✓
SLD.W	✓	✓	STM.MP	×	✓
SST.B	×	✓	LDM.MP	×	✓
SST.H	✓	✓	STM.GSR	×	✓
SST.W	✓	✓	LDM.GSR	×	✓
LD.B	×	✓	SWITCH	×	✓
LD.BU	×	✓	CALLT	×	✓
LD.H	✓	✓	SYSCALL	×	✓
LD.HU	✓	✓	LDV.W	✓	✓
LD.W	✓	✓	LDV.DW	✓	✓
LD.DW	✓	✓	LDV.QW	✓	✓
ST.B	×	✓	STV.W	✓	✓
ST.H	✓	✓	STV.DW	✓	✓
ST.W	✓	✓	STV.QW	✓	✓
ST.DW	✓	✓	LDVZ.H4	✓	✓
LDL.BU	×	✓	STVZ.H4	✓	✓
LDL.HU	✓	✓	CACHE (CHBII)	×	✓
LDL.W	✓	✓	CACHE (CIBII)	×	×
STC.B	×	✓	CACHE (CFALI)	×	✓
STC.H	✓	✓	CACHE (CISTI)	×	×
STC.W	✓	✓	CACHE (CILDI)	×	×
CAXI	✓	✓	CACHE (other commands)	×	×
SET1	×	✓	PREF (PREFI)	×	×*1
CLR1	×	✓	PREF (other commands)	×	×
NOT1	×	✓	Table reference type EIINTn	×	✓*2
TST1	×	✓	Register bank saving processing	×	✓*2
			RESBANK	×	✓

Note 1. When a violation is detected during the execution of the PREF instruction, no MDP exception is generated but the memory access is suppressed. Read access is not performed.

Note 2. Both of table read accesses due to table reference method interrupts and write accesses for automatically saving the context onto the register bank occur independently of the execution of an instruction. This CPU handles any MDP exception that is caused by these accesses as a terminating-type exception. The exception cause code of this MDP exception differs from that of resumable-type MDP exceptions. For details see **Table 4.1**.

Section 5 Memory Management

This CPU provides the following functions for managing memory.

- Memory protection unit (MPU)
- Instruction cache function

5.1 Memory Protection Unit (MPU)

Memory protection functions are provided in an MPU (memory protection unit) to maintain a smooth system by detecting and preventing unauthorized use of system resources by unreliable programs, runaway events, etc. In the virtualization mode, the following three functions are extended to the memory protection function.

- Layered Memory Protection Function
- Memory Protection Setting Bank Function
- Memory Protection Setting High Speed Save and Return Function

5.1.1 Features

(1) Memory Access Control

See the "CPU" section in the hardware manual of the product used.

(2) Access Management for Each CPU Operation Mode

See the "CPU" section in the hardware manual of the product used.

(3) Protection with the System Protection Identifier (SPID)

See the "CPU" section in the hardware manual of the product used.

(4) Protection per Restricted Operating Mode

In the virtualization mode, memory access control corresponding to the restricted operating mode is provided. For details, see **5.1.7, Layered Memory Protection Function**.

5.1.2 Protection Area Settings

(1) Protection Area Settings

See the "CPU" section in the hardware manual of the product used.

(2) Separation of memory protection setting entry

Each setting entry of the memory protection function is classified into the following two based on the value set in the MPCFG.HBE.

- Host management entry — The entry that can be set only in Host mode. Assuming that the hypervisor specifies the access permission area to the virtual machine.
- Guest management entry — The entry that can be set also in Guest mode. Assuming that the software of the virtual machine itself specifies the access permission area.

Table 5.1 Partition of the MPU entry

Value of the MPCFG.HBE	Guest management entry	Host management entry
0	None	All entries
0 < HBE ≤ MPCFG.NMPUE	0 ≤ Entry number < HBE	HBE ≤ Entry number ≤ MPCFG.NMPUE
MPCFG.NMPUE < HBE	All entries	None

The MPU entry configured as host management entry can not be written in Guest mode. The operation when the host management entry is specified with the MPIDX register is as follows.

Table 5.2 Accessibility of the MPU entry

	Host mode		Guest mode	
Entry Type	Read	Write	Read	Write
Guest management entry	Available	Available	Available	Available
Host management entry	Available	Available	Available	Unavailable *1

Note 1. Ignore write operations. The PIE exception does not occur.

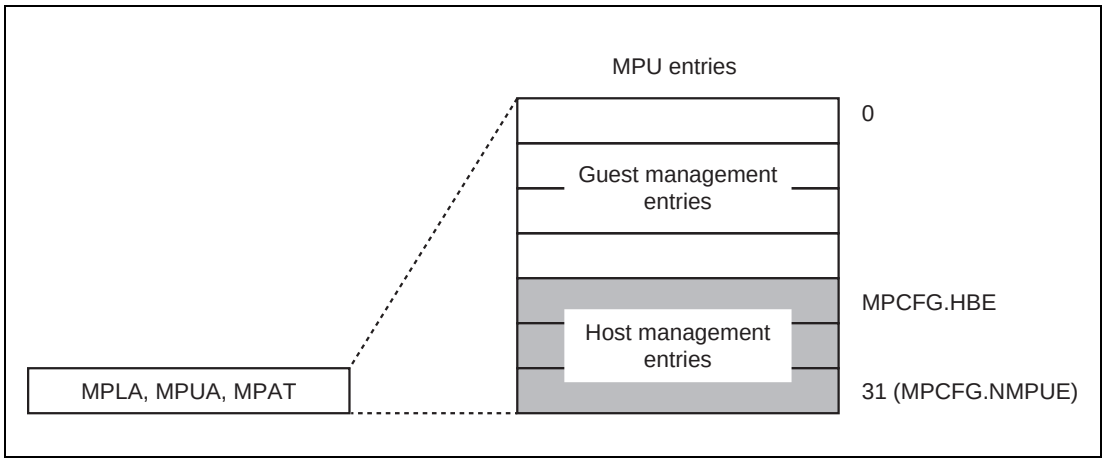


Figure 5.1 Separation of MPU entries

5.1.3 Precautions for Protection Area Setup

(1) Crossing Protection Area Boundaries

See the "CPU" section in the hardware manual of the product used.

(2) Invalid Protection Area Settings

See the "CPU" section in the hardware manual of the product used.

(3) Lower- and Upper-Limit Addresses Referenced during Protection Violation Checks

See the "CPU" section in the hardware manual of the product used.

(4) Memory Access Spanning Contiguous Protection Areas

See the "CPU" section in the hardware manual of the product used.

(5) Memory Access which Spans Over Address 0000 0000_H

See the "CPU" section in the hardware manual of the product used.

(6) Memory access that crosses the host management entry violation area and the guest management violation area

Since the minimum unit of memory protection setting is the word size, if one time memory access size is bigger than the word size, it may cross different memory protection settings. And in the memory access in Guest mode, if the area where the host management entry detects the violation and the area where the violation is detected by the guest management entry are near each other, the exception cause detected when one time memory access crosses the area depends on the memory access method.

In the instruction fetch with the instruction length of 48 bits or 64 bits, the exception cause code corresponding to the management entry that detected the memory protection violation against the address on the lower side (close to 0) is stored in the FEIC. For example, for the instruction of 64 bit length, when the host management entry detects the memory protection violation on its lower side (access is permitted in the guest management entry) and the guest management entry detects the memory protection violation on its upper side, it is determined that the violation has been detected in the host management entry and 98H is stored in the lower 16 bits of the exception cause code.

With the memory access instruction whose memory access size is 64 bits or 128 bits, access destinations are not the word size and are judged collectively as the one area. Therefore, even if the area where the violation is detected by the host management entry and the area where the violation is detected by the guest management entry do not overlap, if it is included in one area of 64 bits or 128 bits, the same judgment as in **Table 5.3** is made and it is detected as the violation in the guest management entry. For example, when the 64-bit memory access instruction (LD.DW, ST.DW, etc.) accesses the area where the lower 32 bits are detected as violation in the host management entry and the upper 32 bits are detected as violation in the guest management entry, it is judged that the violation is detected in the guest management entry, and 91H is stored in the lower 16 bits of the exception cause code.

For instructions that make multiple memory accesses, there are instructions that automatically add addresses and instructions that automatically subtract addresses, but in both cases, the exception cause code corresponding to the management entry that first detected the memory protection violation is stored in FEIC. For example, the PUSHSP is the instruction which store the general purpose registers in memory by ascending order of numbers while subtracting addresses by 4 bytes at a time. When the PUSHSP is executed, if the host management entry detects the memory protection violation with r5 memory access and the guest management entry detects the memory protection violation with r6 memory access (this is the lower side as the address), it is judged that the MDP is detected in the host management entry, and 99H is stored in the lower 16 bits of the exception cause code.

The reason why the violation detection operation differs depending on the memory access method as described above is as follows.

- The instruction length is unknown unless the instruction code is fetched, but if the memory protection violation is detected, the correct instruction code may not be fetched. Therefore, the instruction length can not be considered for judgment of the memory protection violation and it is only possible to check whether memory protection violation can be detected sequentially from the lower order side of the address.
- With the instructions that access 64-bit or 128-bit data, unlike instruction fetch, the size of the memory access is determined by the instruction code. On the other hand, when a part of the access target causes memory protection violation, if the instruction execution is completed as it is, a part of the memory access can not be performed. Therefore, when a part of the access target causes memory protection violation, an exception occurs as in the case that all of the access target causes memory protection violation.
- The instructions that access multiple memories will access a larger memory area when summed, but each memory access size is the word size.

5.1.4 Access Control

See the "CPU" section in the hardware manual of the product used.

5.1.5 Violations and Exceptions

In this CPU, violations are detected during instruction fetch access or operand access according to the protection area settings, and an exception is generated.

- Execution protection violation (during instruction fetch access)
- Data protection violation (during operand access)

(1) Execution Protection Violation (MIP Exception)

See the "CPU" section in the hardware manual of the product used.

(2) Data Protection Violation (MDP Exception)

See the "CPU" section in the hardware manual of the product used.

(3) Exception Cause Code and Exception Address

When an execution protection violation or data protection violation has been detected, the exception cause code is determined as shown in **Table 5.3**, **Table 5.5**, and **Table 5.6**. The determined exception cause code is set to the FEIC register.

The MEA register is used to store either the PC of the instruction that detected the execution protection violation or the access address used when the data protection violation occurred. The MEA register is shared by MIP and MDP exceptions since these exceptions do not occur simultaneously. Also, when a data protection violation occurs, the information of the instruction or event that caused the violation is stored in the MEI register.

(a) MPU exception cause code (lower 16 bits) in Guest mode

In the MPU violation exception of Guest mode, the exception cause differs depending on which management entry the violation occurred. When access is denied by both the host management entry and the guest management entry, the cause code in case of denial in the guest management entry is stored.

Table 5.3 shows the lower 16 bits of the exception cause code that is obtained when the setting of the host management entry and the guest management entry are available and unavailable for access.

Table 5.3 MPU exception cause code (lower 16 bits) in Guest mode

Access setting of the host management entry	Access setting of the guest management entry	MIP (Instruction fetch)	MDP (Operand access)	MDP (Table reference interrupt)
Available	Available	It does not occur.	It does not occur.	It does not occur.
Available	Unavailable	90_H	91_H	95_H
Unavailable	Available	98_H	99_H	$9D_H$
Unavailable	Unavailable	90_H	91_H	95_H

The transition destination mode at exception occurrence can be selected according to the detected level by setting of the GMCFG register. When both the management entries are unavailable, the transition destination is determined according to the setting of the GMCFG.GMP.

Table 5.4 MPU exception detection transition destination mode in Guest mode

Violation entry	GMCFG.GMP	GMCFG.HMP	Transition destination
Guest management entry	0	x	Guest mode
	1	x	Host mode
Host management entry	x	0	Guest mode
	x	1	Host mode

(b) MPU exception cause code (lower 16 bits) in Host mode

In the MPU violation exception of Host mode, the lower 16 bits of the exception cause codes are the values shown in **Table 5.5**.

Table 5.5 MPU exception cause code (lower 16 bits) in Host mode

Access setting of the host management entry	MIP (Instruction fetch)	MDP (Operand access)	MDP (Table reference interrupt)
Available	It does not occur.	It does not occur.	It does not occur.
Unavailable	98_H	99_H	$9D_H$

(c) MPU exception cause code (upper 16 bits)

There is no difference between the guest management entry and the host management entry on the upper 16 bit side of the cause code. **Table 5.6** shows the upper 16 bits of the exception cause codes.

Table 5.6 MPU exception Cause Code (upper 16 bits) of Memory Protection Violation

Exception	Operation Mode at Violation	Bit Number and Bit Name										
		31-25	24	23	22	21	20	19	18	17	16	15-0
		—	MS	BL	RMW	SX*6	SW*6	SR*6	UX*6	UW*6	UR*6	—
MIP	User mode	0	0	0	0	0	0	0	1	0	0	
	Supervisor mode	0	0	0	0	1	0	0	0	0	0	
MDP	User mode	0	*5	*4	*3	0	0	0	0	*2	*1	
	Supervisor mode	0				0	*2	*1	0	0	0	

Note 1. When the read violation occurs in the instruction including the read operation, the SR or UR bit is set (1).

Note 2. When the write violation occurs in the instruction including the write operation, the SW or UW bit is set (1).

Note 3. When the instruction causing the violation contains read-modify-write operation (SET1, NOT1, CLR1, CAXI), it is set (1).

Note 4. When the instruction causing the violation performs block transfer (PREPARE, DISPOSE, PUSHSP, POPSP), it is set (1).

Note 5. When the instruction that caused the violation makes the misaligned access, it is set (1). Misaligned access is applicable when the LD.DW or ST.DW accesses the word boundary address.

Note 6. In Guest mode, when MPU violation is detected in both the host management entry and the guest management entry, only the violation detection result in the guest management entry is stored and the violation detection result in the host management entry is not stored. Even if the MPU violation is detected only in the guest management entry, the violation detection result in the guest management entry is stored in these bits. And when the MPU violation is detected only in the host management entry, the violation detection result in the host management entry is stored in these bits. In addition, in Host mode, since the MPU violation is detected only in the host management entry, the violation detection result in the host management entry is stored in these bits.

Note: UR: A violation is detected during read operation in user mode (PSW.UM = 1).
UW: A violation is detected during write operation in user mode (PSW.UM = 1).
UX: A violation is detected when the instruction executed in user mode (PSW.UM = 1).
SR: A violation is detected during read operation in supervisor mode (PSW.UM = 0).
SW: A violation is detected during write operation in supervisor mode (PSW.UM = 0).
SX: A violation is detected when the instruction executed in supervisor mode (PSW.UM = 0).

5.1.6 Memory Protection Setting Check Function

For the programs, such as OS, that provide services, this CPU provides a memory protection setting check function to enable implementation of a service protection function that checks in advance whether or not the address area to be used for the requested operations is within an area that is accessible by the source that requested the service. By using this function, when verifying the validity of the user-supplied parameters for a system service, the OS can complete the verification in a shorter time than by repeating reading and checking the area settings by software. In the virtualization mode, the memory protection check function can be used in both Host mode and Guest mode. The check results for each operating mode are stored.

(1) Check Details

See the "CPU" section in the hardware manual of the product used.

(2) Checking Procedure

See the "CPU" section in the hardware manual of the product used.

(3) Sample Code

See the "CPU" section in the hardware manual of the product used.

(4) Method of Calculating Address Areas

See the "CPU" section in the hardware manual of the product used.

(5) Checking Results of the Memory Protection Setting Checking Function

The check results are updated according to the following table.

The following abbreviations are used in the table. Each bit of X, W, R becomes 0 or 1 depending on the protection setting. If 0 or 1 is described for the abbreviation type, it means that, under that condition, all three bits of X, W, R become either 0 or 1 as described.

SV mode : Supervisor mode (PSW.UM=0)

UM mode : User mode (PSW.UM=1)

HS*E	Check result for SV mode in the host management entry of MCR HSXE, HSWE, HSRE
HU*E	Check result for UM mode in the host management entry of MCR HUXE, HUWE, HURE
GS*E	Check result for SV mode in the guest management entry of MCR GSXE, GSWE, GSRE
GU*E	Check result for UM mode in the guest management entry of MCR GUXE, GUWE, GURE

S*E	Final check result for SV mode of MCR	SXE, SWE, SRE
U*E	Final check result for UM mode of MCR	UXE, UWE, URE

Table 5.7 Memory protection check result when the MPM.GMPE = 0 in Guest mode

MPM (GMMPM)		Address	OV	HS*E*1	HU*E*1	GS*E	GU*E	S*E	U*E
MPE	SVP								
0	-	-	0	1	1	1	1	1	1
1	0	It crosses 00000000_H or $7FFFFFFF_H$	1	1	1	1	0	1	0
		It does not cross boundary.	0	1	1	1	*2	1	*2
	1	It crosses 00000000_H or $7FFFFFFF_H$	1	1	1	0	0	0	0
		It does not cross boundary.	0	1	1	*2	*2	*2	*2

Note 1. In Guest mode and when the MPM.GMPE = 0, the memory protection by the host management entry is not performed, so the values of the HS * E and HU * E are always 1.

Note 2. The check results of the guest management entry are reflected in each bit of X, W, R. However, when all MPU entries are the host management entries, the value of each bit of X, W, R will be 0.

Table 5.8 Memory protection check results when the MPM.GMPE = 1 in Guest mode

MPM (GMMPM)		Address	OV	HS*E	HU*E	GS*E	GU*E	S*E	U*E
MPE	SVP								
0	-	It crosses 00000000_H or $7FFFFFFF_H$	1	0	0	1	1	0	0
		It does not cross boundary.	0	*1	*1	1	1	*3	*3
1	0	It crosses 00000000_H or $7FFFFFFF_H$	1	0	0	1	0	0	0
		It does not cross boundary.	0	*1	*1	1	*2	*3	*3
	1	It crosses 00000000_H or $7FFFFFFF_H$	1	0	0	0	0	0	0
		It does not cross boundary.	0	*1	*1	*2	*2	*3	*3

Note 1. The check results of the host management entry are reflected in each bit of X, W, R. However, if all MPU entries are the guest management entries, the value of each bit of X, W, R will be 0.

Note 2. The check results of the guest management entry are reflected in each bit of X, W, R. However, if all MPU entries are the host management entries, the value of each bit of X, W, R will be 0.

Note 3. The value obtained by combining both the check results of the host management entry and the check results of the guest management entry are reflected in each bit of X, W, R. They become 1 when both the corresponding bit values of both entries are 1. They become 0 when one of the values is 0.

Table 5.9 Memory protection check results in Host mode

MPM (GMMPM)									
MPE	SVP	Address	OV	HS*E	HU*E	GS*E*1	GU*E*1	S*E	U*E
0	-	-	0	1	1	1	1	1	1
1	0	It crosses 00000000_H or $7FFFFFFF_H$	1	1	0	1	1	1	0
		It does not cross boundary.	0	1	*2	1	1	1	*2
	1	It crosses 00000000_H or $7FFFFFFF_H$	1	0	0	1	1	0	0
		It does not cross boundary.	0	*2	*2	1	1	*2	*2

Note 1. In Host mode, the memory management by the guest management entry is not performed, so the values of the GS * E and GU * E are always 1.

Note 2. The check results of the host management entry are reflected in each bit of X, W, R. However, when all MPU entries are the guest management entries, the value of each bit of X, W, R will be 0.

5.1.7 Layered Memory Protection Function

In the virtualization mode, the layered memory protection function can be used in Guest mode.

In the layered memory protection function, the memory protection setting is set to two types of entries. They are the memory protection entry (the host management entry) that can be set only in Host mode and the memory protection entry (the guest management entry) that can be set also in Guest mode.

And the memory access in Guest mode is only available when both the memory protection entries are set to access enabled. When one of the protection settings is set to access disabled, the memory protection violation and the exception occur.

This CPU is assumed to manage the guest mode memory access in two layers, the host mode management software (Hypervisor) and the guest mode management software (Operating System etc.). For example, the usage which in the host management entry the entire memory area usable by the virtual machine is set by the Hypervisor, and the usage which in the guest management entry the memory area in which the application software operating in each virtual machine can be used is set by the Operating System, are supported. As a result, even when the memory management setting by the management software in Guest mode is unsuitable and access to the memory area for which access from the virtual machine is disabled is enabled in the guest management entry, the memory access to that area can be denied by the memory protection setting of the host management entry. As a result, it is possible to prevent abnormal operation of a virtual machine from influencing the Host mode and other virtual machines.

In addition, memory protection functions (such as access guard function on the slave side) not defined by this CPU exist on the bus, and protection by them may be effective. For details, see the hardware manual of the product used.

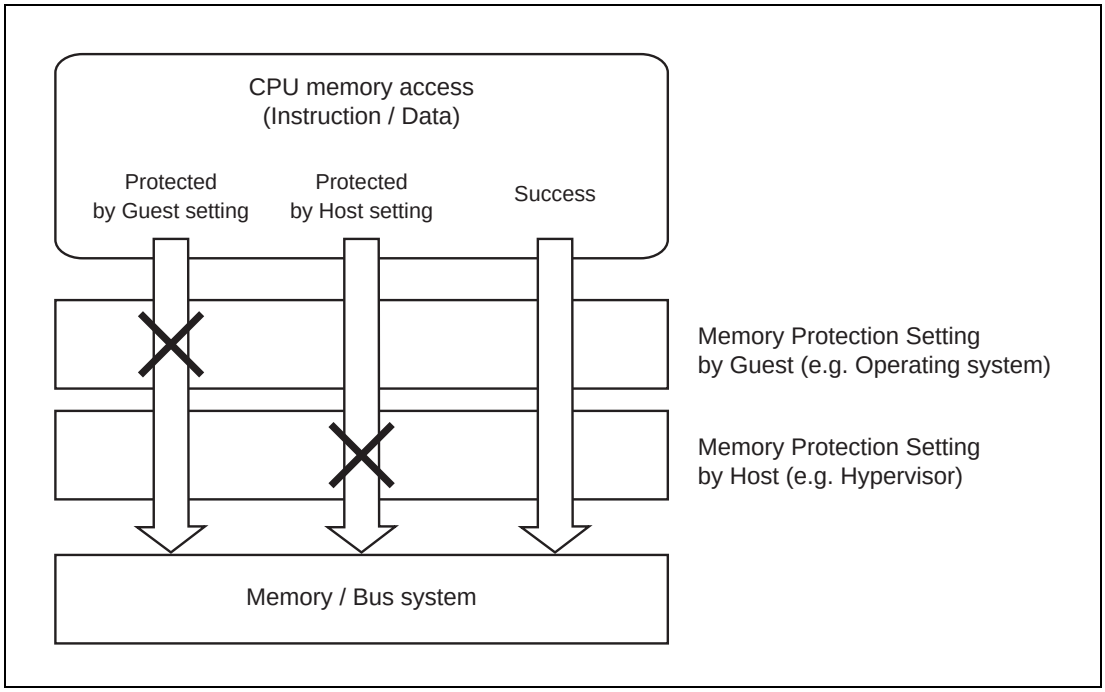


Figure 5.2 Virtualization memory protection

(1) Memory protection in Guest mode

Memory access in Guest mode is permitted when the target address is enabled in both entries according to the following judgment flowchart.

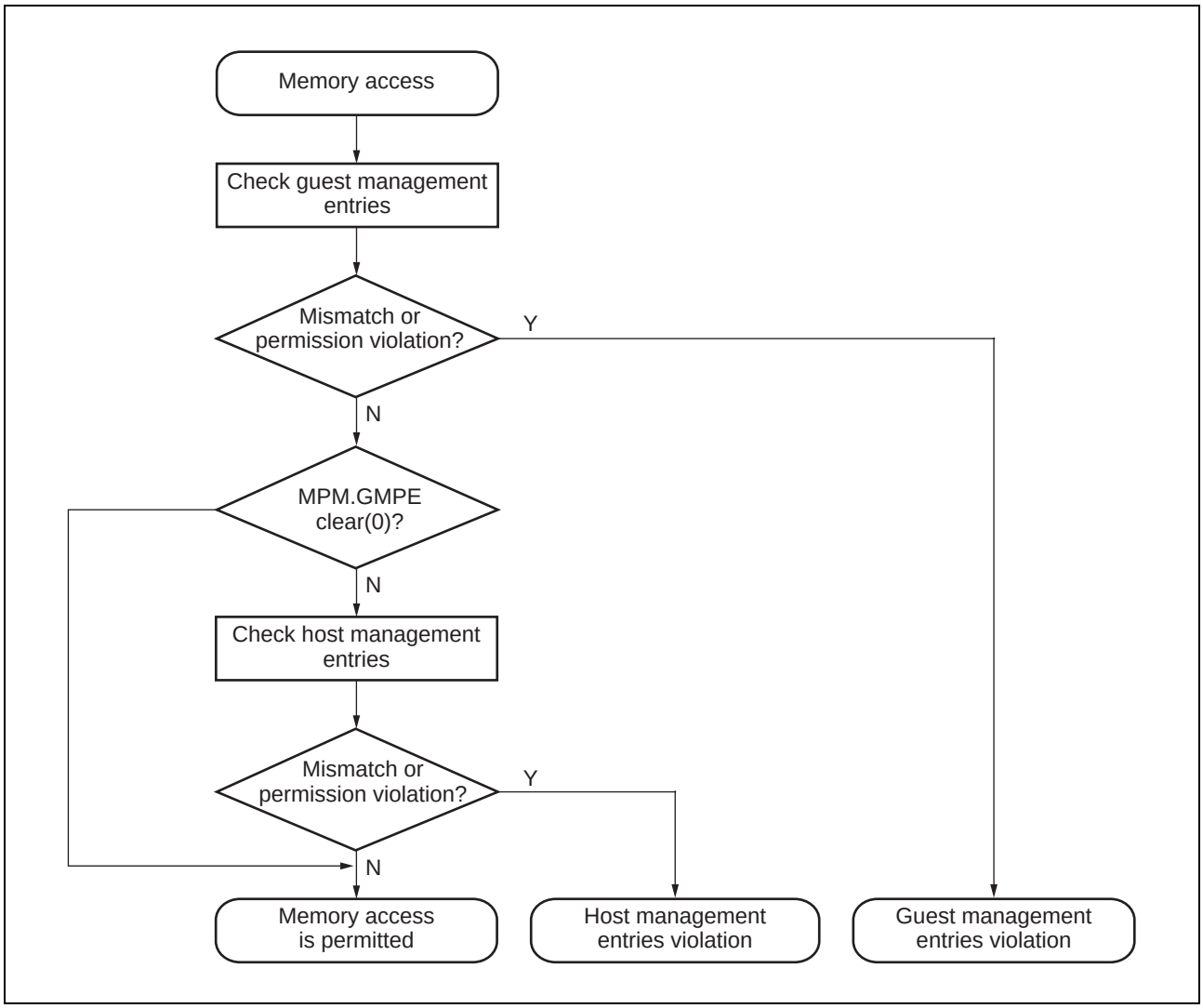


Figure 5.3 Memory protection judgment in Guest mode

Since it is not possible to manipulate the value of the MPIDn register in Guest mode, it is necessary to set the MPU entry based on the value set in Host mode.

(a) Check guest management entries

The memory protection setting of the MPU entry classified as the guest management entry is used to determine whether memory access is enabled or not. Except for the conditions listed below, the method of judging whether memory access is enabled or not is the same as in the case where the hierarchical memory protection function is disabled in the conventional mode. For details, see **Section 5.1.2(1), Protection Area Settings** .

– The entry used for the determination is limited to the guest management entry.

(b) Check host management entries

The memory protection setting of the MPU entry classified as the host management entry is used to determine whether memory access is enabled or not. Except the conditions listed below, the method of judging whether memory access is enabled or not is the same as in the case which the layered memory protection function is disabled in the conventional mode. For details, see **Section 5.1.2(1), Protection Area Settings** .

– The entry used for the determination is limited to the host management entry.
– Whether memory protection is enabled or disabled is specified, not by the MPM.MPE but by the MPM.GMPE.
– When the MPM.GMPE is set (1), memory protection is always performed even in SV mode.

(2) Memory protection in Host mode

Memory access in Host mode is permitted when the target address is enabled in the host management entry according to the following judgment flowchart.

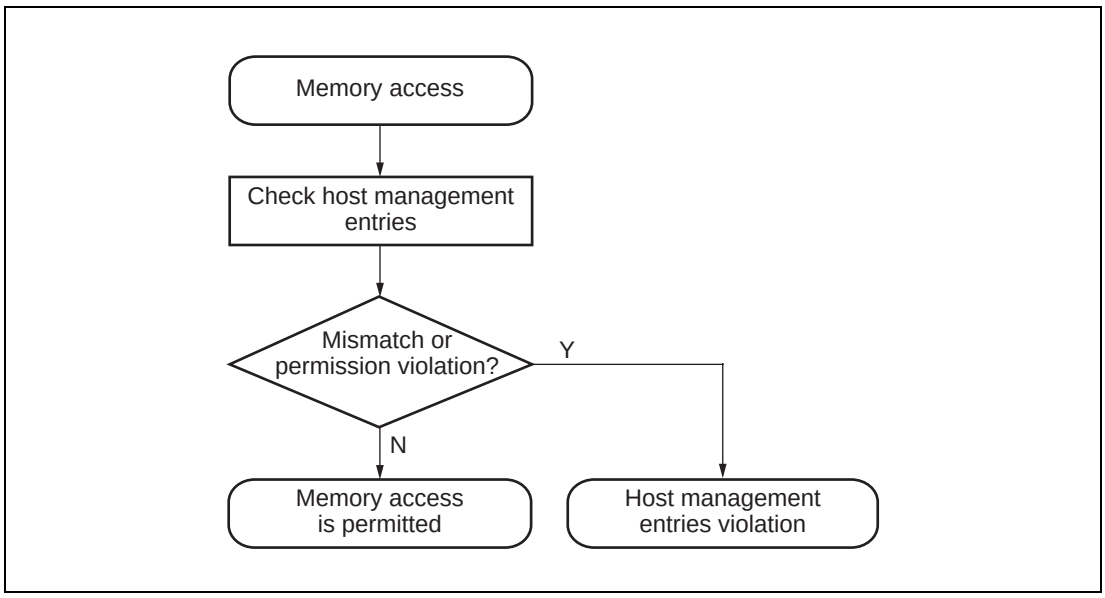


Figure 5.4 Memory protection judgment in Host mode

(a) Check host management entries

The memory protection setting of the MPU entry classified as the host management entry is used to determine whether memory access is enabled or not. Except for the conditions listed below, the method of judging whether memory access is enabled or not is the same as in the case for which the layered memory protection function is disabled in the conventional mode. For details, see **Section 5.1.2 (1), Protection Area Settings**.

– The entry used for the determination is limited to the host management entry.

And the following is different from memory protection by the host management entry in Guest mode. Both of these are the same operation as memory protection in the conventional mode.

– Whether memory protection is enabled or disabled is specified, not by the MPM.GMPE but by the MPM.MPE.

– Validity or invalidity of memory protection in SV mode is performed by the MPM.SVP.

5.1.8 Memory Protection Setting Bank Function

This CPU has maximum of 2 banks of MPU protection setting entries. The MPU bank is specified by the MPBK register. The number of the MPU banks is indicated by the MPCFG.NBK in the system register.

For the operation using each MPU entry below, the MPU bank specified by the MPBK register is used. The value of the protection setting entry of the MPU bank that is not specified by the MPBK has no effect on CPU operation.

- Protection information to perform memory access when MPM.MPE = 1 and MPM.GMPE = 1
- Access by LDSR/STSR instruction (MPAT, MPUA, MPLA specified by MPIDX)
- Memory protection check function
- MPU entry save/restore instructions (STM.MP/LDM.MP)

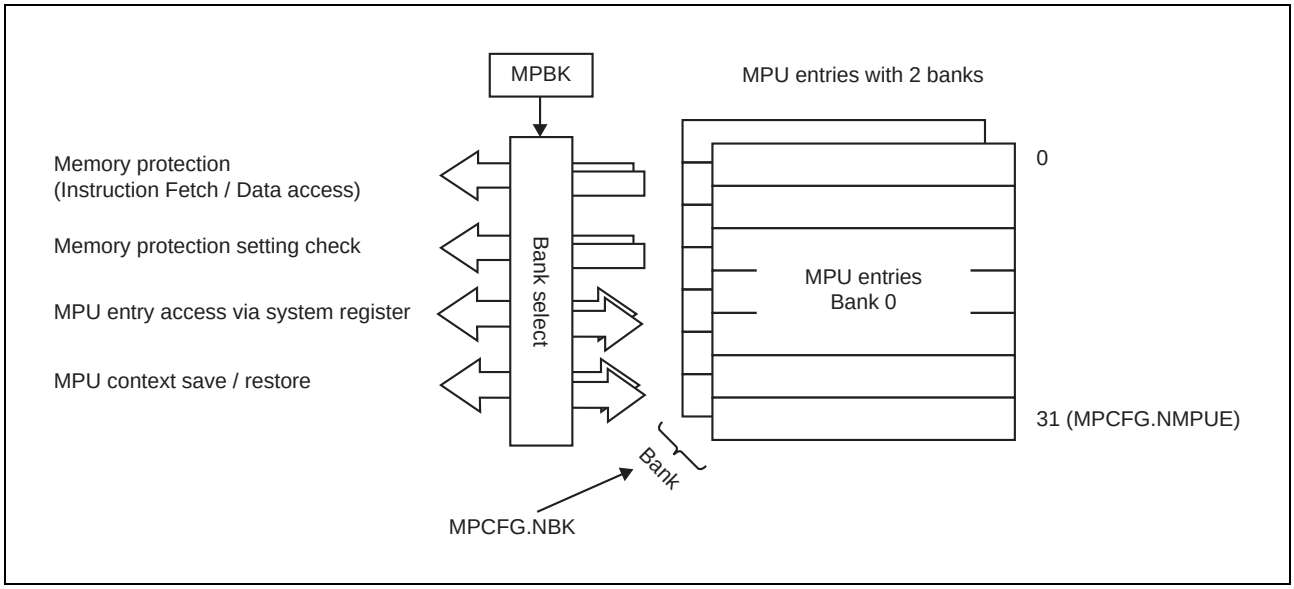


Figure 5.5 MPU Bank changing function

In the MPU bank, since the entry itself is provided as hardware which is banked, it is possible to switch the MPU bank quickly only by changing the bank specification with the MPBK register. This makes it possible to shorten the switch of the MPU context in the time required for virtual machine context switch.

However, since the number of MPU banks is limited, it is necessary for the specification designer of the system to decide the appropriate allocation. In addition, when operating more virtual machines than the number of the MPU banks, it is necessary to save and restore to memory by software in the same way as other system registers.

(1) Notes on Bank Switching

Since the value of the MPIDn register does not change at the time of switching the MPU bank, it is necessary to ensure that there is no inconsistency between the MPIDn assignment setting of the MPAT register and the setting of the MPIDn register.

For example, it is not necessary to manipulate the MPIDn register, especially when the setting of the MPIDn register expected by each entry before and after the switching of the MPU bank is the same.

On the other hand, when the setting of the MPIDn register that each entry expects differs before and after switching the MPU bank, unintended memory protection settings will be applied after switching the MPU bank as it is. Therefore, in such a case, the MPIDn register must also be treated as a virtual machine context, and processing such as saving and restoring must be performed when changing the MPU bank.

5.1.9 Memory protection setting High speed save and restore function

Apart from the MPU bank, dedicated instructions for high speed saving and restore of the MPU entry can be used.

- STM.MP: Store Multiple MPU entries to memory
- LDM.MP: Load Multiple MPU entries from memory

These instructions save and restore the entries of the MPU bank indicated by the MPBK register to memory. For saving and restoring, all entries in the bank are targeted. Entries of the MPU bank that the MPBK register does not indicate are not operated at all.

(1) MPU entry restore instruction in Guest mode

When the STM.MP instruction is executed in Guest mode, the host management entry is also saved. Also, when the LDM.MP instruction is executed, the memory area corresponding to the host management entry is read, but the host management entry is not updated. At this time, the read data is discarded as it is. Even in memory access for the host management entry, when the protection violation or the bus error etc. are detected, the exception corresponding to them will be generated.

(2) Notes on executing the MPU entry save and restore instruction

Execute the MPU entry restore instruction (LDM.MP) by checking that memory access is possible with setting of memory protection invalid or restoring. Since memory protection setting switches during execution of the LDM.MP instruction, it may cause unintended operation.

5.2 Cache

This section describes the function to control the cache memory that is installed in this CPU.

In normal operation, the cache memory is used implicitly in instruction fetching by the CPU. However, if the setting is to be changed or the state of the cache memory is directly manipulated by the CACHE instruction, consistency with the normal operation of CPU must be maintained. In many cases, the hardware makes adjustments so that inconsistencies between the normal operation of the CPU and the manipulation of the cache do not arise. However, to make sure that the results of cache manipulation are the intended state, take care that the manipulation does not affect the implicit operation.

For a detailed description of the cache memory that is installed in the individual products, see the hardware manual of the product used.

5.2.1 Features

See the "CPU" section in the hardware manual of the product used.

5.2.2 Cache Operation Registers

See the "CPU" section in the hardware manual of the product used.

5.2.3 Change Cache Use Mode

See the "CPU" section in the hardware manual of the product used.

5.2.4 Cache Operations Using CACHE Instruction

See the "CPU" section in the hardware manual of the product used.

5.2.5 Cache Operation by the PREF Instruction

See the "CPU" section in the hardware manual of the product used.

5.2.6 Cache Index Specification Method

See the "CPU" section in the hardware manual of the product used.

5.2.7 Execution Privilege of the CACHE/PREF Instruction

Because the CACHE instruction directly manipulates the contents of the cache memory, privileges are specified according to the type of operation. When the CACHE instruction is executed without the privilege required for the CACHE operation, a privilege instruction exception (PIE) occurs.

On the other hand, the PREF instruction provides information for speculative execution, so it can be executed in any mode.

The privileges required by the different operations performed by the CACHE instruction are shown below.

(a) Operations allowed with the user authority

Among address specification method operations, operations without a cache lock (CHBI) can be executed in any operation mode.

(b) Operations requiring the supervisor privilege

When the virtualization operating mode is the conventional mode, among address specification method operations, operations with a cache lock (CFAL) require the supervisor privilege.

In addition, index (CIBII, CISTI, CILDI) specification method operations also require the supervisor privilege.

(c) Operations requiring the Hypervisor privileges

When the virtualization operating mode is the virtualization mode, the Hypervisor privilege is required for operations with cache lock (CFAL) and operations with index specification method (CIBII, CISTI, CILDI) among the operations of the addressing method.

Table 5.10 Relationship between the state of virtualization operating mode and execution authority of CACHE and PREF instruction

Instruction	Address/Index	Instruction execution authority		Remarks
		Conventional mode	Virtualization mode	
CHBII	Address	UM	UM	
CIBII	Index	SV	HV	When these operations are executed with the SV privilege in virtualization mode, the cache data used by Host mode and software which is running on other guest partitions can be changed. Therefore, the execution authority in virtualization mode is changed to HV privilege.
CFALI	Address	SV	HV	
CISTI	Index	SV	HV	
CILDI	Index	SV	HV	
PREF	Address	UM	UM	

5.2.8 Memory Protection for the CACHE and PREF Instructions

See the "CPU" section in the hardware manual of the product used.

5.2.9 Example of Using the CACHE Instruction to Manipulate Cache Memory

See the "CPU" section in the hardware manual of the product used.

5.2.10 Configuration of Instruction Cache

See the "CPU" section in the hardware manual of the product used.

5.2.11 Data Buffer Function

See the "CPU" section in the hardware manual of the product used."

Section 6 Coprocessor

6.1 Floating-Point Operation

See the "CPU" section in the hardware manual of the product used.

6.2 Extended Floating-Point Operation

See the "CPU" section in the hardware manual of the product used.

Section 7 Hazard Control

In this CPU, software-based hazard control may be required in order to allow subsequent instructions to refer to the correct operation results of memory or system register.

- Dedicated instruction for Synchronization Processing
- Guaranteeing the Completion of Store Instruction
- Hazard Management after System Register Update
- Restricted Operating Mode Transition

7.1 Synchronization Processing

For details, see the "CPU" section in the hardware manual of the product used.

7.2 Guaranteeing the Completion of Store Instruction

For details, see the "CPU" section in the hardware manual of the product used.

7.3 Hazard Management after System Register Update

If an LDSR instruction is executed to update the setting of a system register before an STSR instruction is executed to read the system register or before a CALLT instruction or the like, which uses the system register, is executed, then the new setting must be reflected in order to perform the desired operation.

This CPU guarantees that if an LDSR instruction is used to update system registers shown in **Table 7.1**, the new register setting will be applied when the subsequent instruction is executed. However, it does not guarantee that the new setting will be applied in instruction fetching. In this case, synchronization process is required. Also, the execution of an EI or DI instruction is treated in the same way as the update of PSW by an LDSR instruction; that is, it is guaranteed that the values updated by an EI or DI instruction is applied to the subsequent instruction.

Table 7.1 System Registers that Guarantee Reflection of their Updates by LDSR Instruction to the Subsequent Instruction

selID[*1]	System Register[*2]							
0	EIPC	EIPSW	FEPC	FEPSW	PSW	FPSR	FPEPC	FPST
	FPCC	FPCFG	EIIC	FEIC	PSWH	CTPC	CTPSW	EIPSWH
	FEPSWH	CTBP	SNZCFG	EIWR	FEWR			
1	SPID	SPIDLIST	RBASE	EBASE	INTBP[*3]	MCTL	PID	SVLOCK
	SCCFG	SCBP	HVCFG	GMCFG	HVSB			
2	PEID	BMID	MEA	MEI	ISPR[*3]	IMSR	ICSR	INTCFG[*3]
	PLMR[*3]	RBCR0[*4]	RBCR1[*4]	RBNR[*4]	RBIP[*4]			
5	MPM	MPCFG	MCA	MCS	MCC	MCR	MCI	MPIDX[*5]
10	FXSR	FXST	FXINFO	FXCFG	FXXC	FXXP		
12	LSCFG							
13		L1RCFG				RDBCR		

Note 1. In the representation of LDSR and STSR instructions, selection ID are represented as selID.

Note 2. This table includes registers whose value cannot be updated and registers that have bits whose values cannot be updated.

Note 3. If these registers, which control the acceptance of interrupts, are updated, interrupts will be accepted with the new register settings if interrupt requests are present at the time of executing the subsequent instruction.

Note 4. If these registers, which control the register bank function, are updated, interrupts will be accepted with the new register settings if interrupt requests are present at the time of executing the subsequent instruction. However, when the RESBANK instruction is executed after updating the RBCR0.MD, a synchronization process is required.

Note 5. If an LDSR instruction is executed to update the MPLA, MPUA, or MPAT register immediately after another LDSR instruction is executed to update the MPIDX register, it is guaranteed that the new setting of MPIDX is reflected in the subsequent instructions. In order to read the MPLA, MPUA, or MPAT register by using a STSR instruction after the update of the MPIDX register, a synchronization process is required.

If the settings related to instruction fetch or a system register that is not shown in **Table 7.1** is updated, the new register setting can be reflected by performing any of the following synchronization processes immediately after the LDSR instruction. The appropriate synchronization process should be decided according to details of the new register setting and the subsequent operation.

For SYNCI or SYNCP instruction, that is used for synchronization process, see **Section 7.1, Synchronization Processing**.

7.3.1 Updating the Settings Related to Instruction Fetching

For details, see the "CPU" section in the hardware manual of the product used.

7.3.2 Updating the Memory Protection Settings of MPU

For details, see the "CPU" section in the hardware manual of the product used.

7.3.3 Updating Interrupt-Related System Registers

For details, see the "CPU" section in the hardware manual of the product used.

7.3.4 Updating Register Bank Function-Related System Registers

For details, see the "CPU" section in the hardware manual of the product used.

7.3.5 Reading a System Register by Using an STSR Instruction

For details, see the "CPU" section in the hardware manual of the product used.

7.3.6 Referencing a System Register by the Subsequent Instruction

For details, see the "CPU" section in the hardware manual of the product used.

7.3.7 Use of EIRET and FERET Instructions in Synchronization Process

For details, see the "CPU" section in the hardware manual of the product used.

7.3.8 Updating PSW.EBV and EBASE

EBASE is a register that indicates the vector address of the exception handler. This is enabled when PSW.EBV is set (to 1). The recommended updating procedure is as follows.

1. Set PSW.ID to 1
2. Clear PSW.EBV to 0 (updating at the same time with PSW.ID is possible)*1
3. Update EBASE
4. Set PSW.EBV to 1*1
5. Clear PSW.ID to 0 (updating at the same time with PSW.EBV is possible)

Note 1. In Guest Mode, PSW.EBV=1 is fixed, so this procedure is unnecessary.

For updating PSW.UM in the procedure above as well, use of an EIRET or FERET instruction described in the hardware manual of the product used is recommended.

7.3.9 Synchronization processing of STM.MP, LDM.MP, STM.GSR, LDM.GSR instructions

The STM.MP, LDM.MP, STM.GSR and LDM.GSR instructions manipulate the values of the system registers. Therefore, for instructions executed before and after these instruction executions, these instructions perform the following synchronization process.

- STM.GSR instruction waits for the execution completion of all preceding FPU instructions executed. Consequently, the values of FPSR and FPECP updated by the FPU instructions are certainly saved.
- STM.MP and STM.GSR instructions wait for the execution completion of all preceding LDSR instructions executed. Consequently, the update results of the system registers which are not listed in **Table 7.1** are also certainly saved.
- All instructions following LDM.MP and LDM.GSR wait for the completion of LDM.MP and LDM.GSR executed in advance. Consequently, subsequent instructions can operate with the update results of the system registers due to these instructions. However, regarding the update of the MPU function register due to the LDM.MP instruction, it may not be precisely reflected in the memory protection function related to instruction fetch. For details, see **Section 7.3.1, Updating the Settings Related to Instruction Fetching**.
- If an interrupt is acknowledged or an exception occurs during the execution of the LDM.MP instruction or the LDM.GSR instruction, those acknowledges wait for the completion of all load processing in the middle of execution. Consequently, this ensures that the system registers operated by these instructions are not updated after the interrupt handler or exception handler starts processing

7.4 Synchronizing for restricted operating mode transition

The state of restricted operating mode differs greatly from the viewpoint of CPU authority. Therefore, if CPU authority and authority of bus access being executed in the bus system are different, there might be a risk of causing unnecessary authority conflict.

In order to prevent such risk in advance, in case the state transition of the restricted operating mode is necessary in this CPU, the same synchronization processing as the SYNCM instruction is performed at the time of acknowledging of causing exception and execution of the return instruction. Consequently, all the load and store processing that were executed before the state transition are completed. For details of SYNCM instruction, that are used for synchronization processing, see **Section 7.1, Synchronization Processing**.

If the transition cause of the restricted operating mode is a terminating exception such as interrupt (EIINTn), which exception will be acknowledged is determined according to the judgment of acknowledgment priority order between terminating exceptions. For details of the determination of acknowledgment priority order, see **Section 4.1.1, Exception Cause List**. If SYSERR (terminating-type) is not acknowledged, the request for that exception is made pending. SYSERR (terminating type) which is detected during guest mode operation will not occur after transitioning to host mode.

Section 8 Reset

8.1 Status of Registers After Reset

If a reset signal is input by a method defined by the hardware specifications, the program registers and system registers are placed in the status shown by the value after reset of each register in **Section 3, Register Set**, and program execution is started. Set the contents of each register to an appropriate value in the program.

The CPU starts execution of a program from the reset address specified in **Section 4.4, Exception Handler Address** by reset.

Note that because the PSW.ID bit is set to 1 immediately after a reset, conditional EI level exceptions will not be acknowledged. To acknowledge conditional EI level exceptions, clear the PSW.ID bit to 0.

Section 9 Virtualization of Interrupt

This section describes the virtualization function provided by the interrupt controller. For details of interrupt controller registers, see the "Interrupts" section in the hardware manual of the product used.

9.1 Interrupt Binding

In the interrupt controller, all EI level and FE level interrupt channels can be allocated to Host mode or Guest mode. This allocation of interrupt channels to operating modes is called interrupt binding. An interrupt is bound by setting the CPU operating mode and partition ID (valid only when the CPU operating mode is Guest mode) in the EIBDn and FIBDn registers.

When an interrupt request of an interrupt channel bound to Host mode is accepted by the CPU, the exception handler is processed in Host mode. In the interrupt controller after reset, all interrupt channels are bound to Host mode.

When an interrupt request of an interrupt channel bound to Guest mode is accepted by the CPU, the exception handler is processed in Guest mode. To bind an interrupt channel to Guest mode, use a partition ID (GPID) to specify a Guest mode partition (PSWH.GPID) for binding. By binding an interrupt channel to Guest mode in this way, a virtual machine operating in Guest mode can directly accept an interrupt to handle the interrupt quickly.

9.2 Notification of an Interrupt Request

Table 9.1, Interrupt Binding and Interrupt Request lists interrupt binding settings and interrupt requests of which the CPU is notified according to the CPU operating mode. The CPU always notifies the interrupt controller of the CPU operating mode (PSWH.GM) and partition ID (PSWH.GPID) used by the interrupt controller to check interrupt channel binding.

Table 9.1 Interrupt Binding and Interrupt Request

Interrupt Input	CPU Operating Mode	Interrupt Channel Bind Destination		Background Interrupt	Interrupt Request of Which the CPU Is Notified
		Operating Mode	GPID		
FEINT	Host mode	Host mode	—	—	FEINT
		Guest mode	—	—	None
	Guest mode	Host mode	—	—	FEINT
		Guest mode	Match	—	GMFEINT
			Mismatch	Allowed*1	BGFEINT
				Not allowed*1	None
EIINT	Host mode	Host mode	—	—	EIINT
		Guest mode	—	—	None
	Guest mode	Host mode	—	—	EIINT
		Guest mode	Match	—	GMEIINT
			Mismatch	Allowed*2	BGEIINT
				Not allowed*2	None

Note 1. Allowed when FIBG.BGEn corresponding to the channel is 1. Not allowed when it is 0.

Note 2. Allowed when the interrupt priority is higher than EIBG.BGPR. Not allowed when it is not.

When the CPU is in Host mode, the interrupt controller notifies the CPU of only the interrupt requests bound to Host mode (FEINT/EIINT). The interrupt requests bound to Guest mode are treated as masked.

When the CPU is in Guest mode, the interrupt controller notifies the CPU of the interrupt requests bound to Guest mode for which the PSWH.GPID and EIBDn/FIBDn setting match (GMFEINT/GMEIINT). The interrupt requests for which the PSWH.GPID and setting do not match are treated as masked. However, when a priority for which a background interrupt is allowed is specified by the EIBG/FIBG setting, the interrupt requests for which the PSWH.GPID and setting do not match are not treated as masked and the CPU is notified of the interrupt requests as background interrupts (BGFEINT/BGEIINT).

Notification priority is determined for interrupt requests for which the PSWH.GPID and setting match and interrupt requests treated as background interrupts together. Therefore, even when there is an interrupt request treated as a background interrupt, when there is an interrupt request with higher priority for which the PSWH.GPID and setting match, the CPU is notified of GMFEINT/GMEIINT.

When the CPU is in Guest mode, it is also notified of an interrupt request bound to Host mode (FEINT/EIINT). When the CPU accepts the interrupt request, it changes to Host mode. The CPU is notified of an interrupt request bound to Host mode with any interrupt priority by giving priority over any interrupt request bound to Guest mode. For this reason, if an interrupt acceptance condition is not satisfied in Host mode, the interrupt request bound to Host mode is not accepted, and the CPU is not notified of the interrupt request bound to Guest mode. To avoid this status, to change the CPU from Host mode to Guest mode, set the Host mode acceptance condition in advance so that all interrupt requests bound to Host mode can be accepted. To accept the interrupt requests, the following settings are required.

- For operation when INTCFG.EPL is cleared (0), clear all ISPR bits (0).
- For operation when INTCFG.EPL is set (1), set PSW.EIMASK to $3F_H$.
- Set the PLMR to $3F_H$.
- Clear (0) PSW.ID.
- Clear (0) PSW.NP.
- Set the priority of interrupts bound to Host mode to a value smaller than $3F_H$.

To clear the ISPR (0), set INTCFG.ISPC (1) and write 0 to the ISPR. To change the CPU from Host mode to Guest mode by using the EIRET instruction, set values for EIPSW.EIMASK, NP, and ID as shown above to change the PSW bits, and then execute the EIRET instruction.

9.3 Restriction on IHVCFG.IHVE Operation

Be sure to set the same value in IHVCFG.IHVE and HVCFG.HVE. When different values are set in IHVCFG.IHVE and HVCFG.HVE, enabling the interrupt function is prohibited.

9.4 Restriction on Operation in Guest Mode with Interrupt Controller INTC1

When the CPU is in Guest mode, it can operate only the channels bound to PSWH.GPID to update INTC1 registers. Accessing a channel not bound to PSWH.GPID causes an access violation to HV privilege. Operating the IMR to update interrupt masks at a time can update only the bits corresponding to the channels bound to PSWH.GPID. The values of the bits corresponding to the channels not bound to PSWH.GPID are retained.

9.5 Restriction on Operating EICn and EEICN Registers

The interrupt controller determines priority with 64 priority levels regardless of the EPL value in the CPU. At this time, the interrupt priority of each channel is determined according to the EEICn.EIP[7:4] and EEICn.EIP[3:0]*1 values. For this reason, when each channel is operated using the EICn register, the determined priority may not be as expected depending on the EEICn.EIP[7:4] value. To operate each channel when the interrupt priority level extension function is disabled (INTCFG.EPL = 0), be sure to set the EEICn.EIP[7:4] value to 0, and then use the EICn register. To operate each channel when the interrupt priority level extension function is enabled (INTCFG.EPL = 1), be sure to use the EEICn register.

Note 1. The update of EICn.EIP[3:0] is reflected to EEICn.EIP[3:0]. The update of EEICn.EIP[3:0] is reflected to EICn.EIP[3:0].

RH850G4MH Virtualization
User's Manual: Hardware

Publication Date: Rev.0.90 April 17, 2020
Rev.1.40 August 31, 2023

Published by: Renesas Electronics Corporation

RH850G4MH Virtualization

Renesas Electronics Corporation

R01UH0865EJ0140