User's Manual

32

RH850G4MH

Virtualization

User's Manual: Software

Renesas microcontroller
RH850 Family

Renesas Electronics
www.renesas.com

Rev. 1.10 Jun. 2021

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 Overview

1.1 Purpose of This User’s Manual

This user's manual describes the details of instructions related to the virtualization function[Note 1] available in the RH850G4MH. For details of other instructions, see the *RH850G4MH User's Manual: Software.*

For details of RH850 architecture, see the hardware manual of the product used.

Note 1. The virtualization function is supported only when Architecture Identifier bit is 07_H in the PID register of RH850 architecture.

Section 2 Instruction

This section describes the instructions (mnemonics) used for this CPU.

Only the changes involved with the addition of the virtualization support function are described below. For details of each instruction (mnemonic) for which any specification has not been changed, see the relevant section of the *RH850G4MH User's Manual: Software*.

2.1 Opcodes and Instruction Formats

This CPU has two types of instructions: CPU instructions, which are defined as basic instructions, and coprocessor instructions, which are defined according to the application.

2.1.1 CPU Instructions

Instructions classified as CPU instructions are allocated in the opcode area other than the area used in the format of the coprocessor instructions shown in **Section 2.1.2, Coprocessor Instructions**.

CPU instructions are basically expressed in 16-bit and 32-bit formats. There are also several instructions that use option data to add bits, enabling the configuration of 48-bit and 64-bit instructions. For details, see the opcode of the relevant instruction in **Section 2.2.3, Basic Instruction Set**.

Opcodes in the CPU instruction opcode area that do not define significant CPU instructions are reserved for future function expansion and cannot be used. For details, see **Section 2.1.3, Reserved Instructions**.

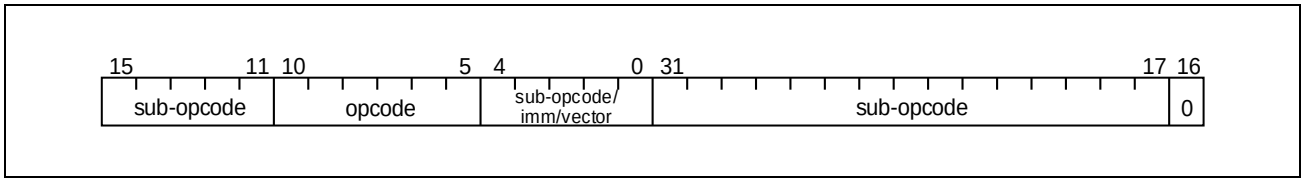
Only the instruction format of each instruction (mnemonic) changed with the addition of the virtualization support function is described below. For details of the instruction format of each instruction (mnemonic) for which any specification has not been changed, see the relevant section of the *RH850G4MH User's Manual: Software*.

(1) Extended Instruction Format 2 (Format X)

This is a 32-bit instruction format that has a 6-bit opcode field and uses the other bits as a sub- opcode field.

CAUTION

Extended instruction format 2 might use part of the general-purpose register specification field or the sub-opcode field as a system register number field, condition code field, immediate field, or displacement field. For details, see the description of each instruction in **Section 2.2.3, Basic Instruction Set**.

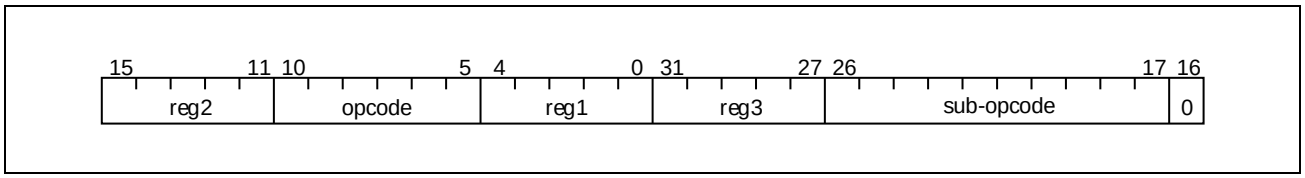


(2) Extended Instruction Format 3 (Format XI)

This is a 32-bit instruction format that has a 6-bit opcode field and three general-purpose register specification fields, and uses the other bits as a sub-opcode field.

CAUTION

Extended instruction format 3 might use part of the general-purpose register specification field or the sub-opcode field as a system register number field, condition code field, immediate field, or displacement field. For details, see the description of each instruction in **Section 2.2.3, Basic Instruction Set**.



2.1.2 Coprocessor Instructions

For detailed of Coprocessor Instructions, see the relevant section of the *RH850G4MH User's Manual: Software*.

2.1.3 Reserved Instructions

For detailed of Reserved Instructions, see the relevant section of the *RH850G4MH User's Manual: Software*.

2.2 Basic Instructions

2.2.1 Overview of Basic Instructions

Only the changes involved with the addition of the virtualization support function are described below. For details of each instruction (mnemonic) for which any specification has not been changed, see the relevant section of the *RH850G4MH User's Manual: Software*.

(16) Special Instructions

The following instructions (mnemonics) are provided.

- EIRET : Return from EI level trap or interrupt
- FERET : Return from FE level trap or interrupt
- LDM.MP : Load Multiple MPU entries from memory
- STM.MP : Store Multiple MPU entries to memory

2.2.2 Special Operations

For detailed of Special Operations, see the relevant section of the *RH850G4MH User's Manual: Software*.

2.2.3 Basic Instruction Set

This section explains the following items of each mnemonic (in alphabetical order).

- Instruction format: Indicates how the instruction is written and its operand(s) (for symbols, see **Table 2.1**).
- Operation: Indicates the function of the instruction (for symbols, see **Table 2.2**).
- Format: Indicates the instruction format (see **Section 2.1, Opcodes and Instruction Formats**).
- Opcode: Indicates the bit field of the instruction opcode (for symbols, see **Table 2.3**).
- Flag: Indicates the change of flags of PSW (program status word) after the instruction execution. "0" is to clear (reset), "1" to set, and "—" to remain unchanged.
- Description: Describes the operation of the instruction.
- Supplement: Provides supplementary information on the instruction.
- Caution: Provides precautionary notes.

Table 2.1 Conventions of Instruction Format

Symbol	Meaning
reg1	General-purpose register (as source register)
reg2	General-purpose register (primarily as destination register with some as source registers)
reg3	General-purpose register (primarily used to store the remainder of a division result and/or the higher 32 bits of a multiplication result)
bit#3	3-bit data to specify bit number
imm ×	×-bit immediate data
disp ×	×-bit displacement data
regID	System register number
selID	System register selection ID
vector ×	Data to specify vector (× indicates the bit size)
cond	Condition code (see **Table 2.4**)
cccc	4-bit data to specify condition code (see **Table 2.4**)
sp	Stack pointer (r3)
ep	Element pointer (r30)
list12	Lists of registers
rh-rt	Indicates multiple general-purpose registers, from the general-purpose register indicated by *rh* to the general-purpose register indicated by *rt*.
eh-et	Indicates multiple system registers of MPU entry (MPLA, MPUA, MPAT), from the entry number indicated by *eh* to the entry number indicated by *et*.
[ ]+	Post increment addressing
[ ]−	Post decrement addressing

Table 2.2 Conventions of Operation (1/2)

Symbol	Meaning
←	Assignment
GR [a]	Value stored in general-purpose register *a*
SR [a, b]	Value stored in system register (RegID = *a*, SelID = *b*)
(n:m)	Bit selection. Select from bit *n* to bit *m*.
CheckException(a)	Checks the conditions for generating the exception "a" and, if one is detected, suspends the instruction execution and performs exception processing.

Table 2.2 Conventions of Operation (2/2)

Symbol	Meaning
zero-extend (n)	Zero-extends “n” to word
sign-extend (n)	Sign-extends “n” to word
load-memory (a, b)	Reads data of size *b* from address *a*
store-memory (a, b, c)	Writes data *b* of size *c* to address *a*
extract-bit (a, b)	Extracts value of bit *b* of data *a*
set-bit (a, b)	Sets value of bit *b* of data *a*
not-bit (a, b)	Inverts value of bit *b* of data *a*
clear-bit (a, b)	Clears value of bit *b* of data *a*
saturated (n)	Performs saturated processing of “n”. If n ≥ 7FFF FFFF$_H$, n = 7FFF FFFF$_H$. If n ≤ 8000 0000$_H$, n = 8000 0000$_H$.
clip (a, b, c)	Performs saturated processing on the word data “a” assuming the sign “b” and converts it to data of the size “c”. ● If the sign “b” is Sign and the size “c” is Byte: When 0000 007F$_H$ < a ≤ 7FFF FFFF$_H$, the result is 0000 007F$_H$. When 8000 0000$_H$ ≤ a < FFFF FF80$_H$, the result is FFFF FF80$_H$. ● If the sign “b” is Unsign and the size “c” is Byte: When 0000 00FF$_H$ < a, the result is 0000 00FF$_H$. ● If the sign “b” is Sign and the size “c” is Halfword: When 0000 7FFF$_H$ < a ≤ 7FFF FFFF$_H$, the result is 0000 7FFF$_H$. When 8000 0000$_H$ ≤ a < FFFF 8000$_H$, the result is FFFF 8000$_H$. ● If the sign “b” is Unsign and the size “c” is Halfword: When 0000 FFFF$_H$ < a, the result is 0000 FFFF$_H$.
result	Outputs results on flag
Byte	Byte (8 bits)
Halfword	Halfword (16 bits)
Word	Word (32 bits)
==	Comparison (true upon a match)
!=	Comparison (true upon a mismatch)
+	Add
−	Subtract
||	Bit concatenation
×	Multiply
÷	Divide
%	Remainder of division results
AND	AND
OR	OR
XOR	Exclusive OR
NOT	Logical negate
logically shift left by	Logical left-shift
logically shift right by	Logical right-shift
arithmetically shift right by	Arithmetic right-shift
P-TYPE_Addressing()	Handles post index increment/decrement addressing.

Table 2.3 Conventions of Opcode

Symbol	Meaning
R	1-bit data of code specifying reg1 or regID
r	1-bit data of code specifying reg2
w	1-bit data of code specifying reg3
D	1-bit data of displacement (indicates higher bits of displacement)
d	1-bit data of displacement
I	1-bit data of immediate (indicates higher bits of immediate)
i	1-bit data of immediate
V	1-bit data of code specifying vector (indicates higher bits of vector)
v	1-bit data of code specifying vector
cccc	4-bit data for condition code specification (See **Table 2.4**)
bbb	3-bit data for bit number specification
L	1-bit data of code specifying general-purpose register in register list
S	1-bit data of code specifying EIPC/FEPC, EIPSW/FEPSW in register list
P	1-bit data of code specifying PSW in register list

Table 2.4 Condition Codes

Condition Code (cccc)	Condition Name	Condition Formula
0000	V	OV = 1
1000	NV	OV = 0
0001	C/L	CY = 1
1001	NC/NL	CY = 0
0010	Z	Z = 1
1010	NZ	Z = 0
0011	NH	(CY or Z) = 1
1011	H	(CY or Z) = 0
0100	S/N	S = 1
1100	NS/P	S = 0
0101	T	Always (Unconditional)
1101	SA	SAT = 1
0110	LT	(S xor OV) = 1
1110	GE	(S xor OV) = 0
0111	LE	((S xor OV) or Z) = 1
1111	GT	((S xor OV) or Z) = 0

2.2.3.1 EIRET

<Special instruction>

Return from EI level trap or interrupt

EIRET

Return from EI level exception

[Instruction format] EIRET

[Operation]

```
if ( PSWH.GM==1 )
then
    if ( GMPSW.UM==0 )
    then
        PC ← GMEIPC
        GMPSW ← GMEIPSW
    else
else
    if ( HMPSW.UM==0 )
    then
        PC ← HMEIPC
        HMPSW ← HMEIPSW
        PSWH ← EIPSWH
    else
```

[Format] Format X

[Opcode]

15 0	31 16
0000011111100000	0000000101001000

[Flags]

CY	Value read from EIPSW.CY is set
OV	Value read from EIPSW.OV is set
S	Value read from EIPSW.S is set
Z	Value read from EIPSW.Z is set
SAT	Value read from EIPSW.SAT is set

[Description]

Returns execution from an EI level exception.

This instruction loads the return PC, return PSW, and return PSWH from the EIPC, EIPSW, and EIPSWH, sets the values in the PC, PSW, and PSWH, and passes control to the return PC address.

When the EIRET instruction is executed while EIPSWH.GM is set (1) in the host mode, PSWH.GM is restored to the value and a transition to the guest mode occurs. In the transition to the guest mode, in addition, control is passed to the return PC address loaded from the HMEIPC. For this reason, to execute the EIRET instruction which causes a mode transition, it is necessary to set a value appropriate for the guest mode in the HMEIPC in advance. When the EIRET instruction which causes a mode transition is executed, the HMPSW is restored to the value of the HMEIPSW. A transition to the guest mode causes the destination referenced as the PSW to be changed to the GMPSW. The value of the GMPSW is not restored by executing the EIRET instruction which causes a mode transition. For this reason, it is necessary to set a value appropriate for the guest mode in the GMPSW before executing the EIRET instruction in advance.

When the EIRET instruction is executed in the guest mode, the PSWH is not restored to the EIPSWH. In the guest mode, a mode transition cannot be caused by executing the EIRET instruction.

When the EIRET instruction is executed while the INTCFG.EPL, INTCFG.ISPC, and PSW.EP are all cleared (0), the corresponding bits of the ISPR register are cleared.

[Supplement]

This instruction is a supervisor-privileged instruction.

2.2.3.2 FERET

<Special instruction>

FERET	Return from FE level trap or interrupt Return from FE level exception

[Instruction format] FERET

[Operation]

```
if ( PSWH.GM==1 )
then
    if ( GMPSW.UM==0 )
    then
        PC ← GMFEPC
        GMPSW ← GMFEPSW
    else
else
    if ( HMPSW.UM==0 )
    then
        PC ← HMFEPC
        HMPSW ← HMFEPSW
        PSWH ← FEPSWH
    else
```

[Format] Format X

[Opcode]

15 0	31 16
0000011111100000	0000000101001010

[Flags]

CY	Value read from FEPSW.CY is set
OV	Value read from FEPSW.OV is set
S	Value read from FEPSW.S is set
Z	Value read from FEPSW.Z is set
SAT	Value read from FEPSW.SAT is set

[Description]

Returns execution from an FE level exception.

This instruction loads the return PC, return PSW, and return PSWH from the FEPC, FEPSW, and FEPSWH, sets the values in the PC, PSW, and PSWH, and passes control to the return PC address.

When the FERET instruction is executed while FEPSWH.GM is set (1) in the host mode, PSWH.GM is restored to the value, a transition to the guest mode occurs. In the transition to the guest mode, in addition, control is passed to the return PC address loaded from the HMFEPC. For this reason, to execute the FERET instruction which causes a mode transition, it is necessary to set a value appropriate for the guest mode in the HMFEPC in advance. When the FERET instruction which causes a mode transition is executed, the HMPSW is restored to the value of the HMFEPSW. A transition to the guest mode causes the destination referenced as the PSW to be changed to the GMPSW. The value of the GMPSW is not restored by executing the FERET instruction which causes a mode transition. For this reason, it is necessary to set a value appropriate for the guest mode in the GMPSW before executing the FERET instruction in advance.

When the FERET instruction is executed in the guest mode, the PSWH is not restored to the FEPSWH. In the guest mode, a mode transition cannot be caused by executing the FERET instruction.

[Supplement]

This instruction is a supervisor-privileged instruction.

CAUTION

The FERET instruction can also be used as a hazard barrier instruction when the CPU's operating status (PSW) is changed by a control program such as the OS. Use the FERET instruction to clarify the program blocks on which to effect the hardware function (mainly the memory management function) associated with the UM bit in the PSW when these bits are changed to accord with the mounted CPU. The hardware function that operates in accordance with the PSW value updated by the FERET instruction is guaranteed to be effected from the instruction indicated by the return address of the FERET instruction.

2.2.3.3 LDM.MP

<Special instruction>

LDM.MP

Load Multiple MPU entries from memory

Load MPU entries

[Instruction format] LDM.MP [reg1], eh-et

[Operation]

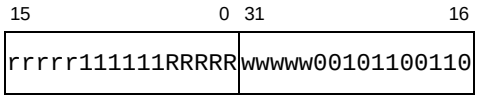
```
if ( PSW.UM==0 )
then
    if ( eh ≤ et )
    then
    cur ← eh
    end ← et
    tmp ← reg1
    while (cur ≤ end) {
        adr ← tmp Note 1, Note 2
        CheckException(MDP)
        MPLA[cur] ← Load-memory (adr, Word)
        tmp ← tmp + 4
        adr ← tmp Note 1, Note 2
        CheckException(MDP)
        MPUA[cur] ← Load-memory (adr, Word)
        tmp ← tmp + 4
        adr ← tmp Note 1, Note 2
        CheckException(MDP)
        MPAT[cur] ← Load-memory (adr, Word)
        tmp ← tmp + 4
        cur ← cur + 1
    }
    else
else
```

Note 1. The lower 2 bits of adr are masked by 0.

Note 2. An MDP exception may occur as a result of address calculation.

[Format] Format XI

[Opcode]



rrrrr indicates eh.
wwwww indicates et.
RRRRR indicates reg1.

[Flags]

CY	—
OV	—
S	—
Z	—
SAT	—

[Descriptions]

The word data is read from the address generated from the word data of the general-purpose register reg1 and stored to the MPU protection area setting system registers (MPLA, MPUA, and MPAT) according to the specified order. Word size is added to the address each time the read data is stored to the system register. The contents of these system registers is processed in ascending order, regardless of the value of MPIDX, from the entry number starting from eh to that starting from et (eh, eh+1, eh+2, …, et). The bank specified by MPBK is only to be processed.

[Supplement]

This instruction stores data directly from memory to multiple target system registers. This instruction can perform the operation more effectively than by reading memory into a general-purpose register by LD.W instructions, and storing it to the system register by LDSR instructions.

The lower 2-bit address generated from the general-purpose register reg1 is masked by 0 and aligned on a word boundary. The general-purpose register reg1 retains the original value after the instruction execution is complete.

This instruction is an SV privilege instruction.

When this instruction is executed in guest mode and host managed entries are included for specified entries, data is read from the corresponding memory, whose data is discarded without being stored to the system register. In this case, an exception accompanied by memory access occurs though a PIE exception does not occur.

CAUTIONS

1. **When an exception or an interrupt occurs during instruction execution and even if data from memory has not been stored to all system registers, the instruction execution can be aborted and the exception or interrupt can be accepted, when the acceptance condition is satisfied. When the execution is suspended, it is impossible to know to which system registers data from memory has been stored. After the return from exception processing, the suspended LDM.MP instruction can be precisely re-executed as long as resources related to execution of the LDM.MP instruction are not changed during exception processing, for the return PC from an exception is considered to be PC of this LDM.MP instruction. This instruction re-execution restarts the LDM.MP instruction processing from the start.**
2. **When this instruction is executed, memory protection violation is detected with the MPU settings updated. This instruction as hardware function does not automatically stop memory protection violation detection. Therefore, it is necessary to avoid the occurrence of unintended memory protection violation during execution of this instruction that memory protection function be disabled in advance or entries that configured the memory protection settings for memory access be excluded from the processing target. Memory protection by host managed entries is always enabled.**

2.2.3.4 STM.MP

<Special instruction>

STM.MP

Store Multiple MPU entries to memory

Store MPU entries

[Instruction format] STM.MP eh-et, [reg1]

[Operation]

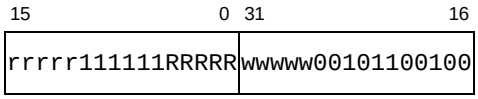
```
if ( PSW.UM==0 )
then
    if ( eh ≤ et )
    then
        cur ← eh
        end ← et
        tmp ← reg1
        while (cur ≤ end) {
            adr ← tmp Note 1, Note 2
            CheckException(MDP)
            Store-memory (adr, MPLA[cur], Word)
            tmp ← tmp + 4
            adr ← tmp Note 1, Note 2
            CheckException(MDP)
            Store-memory (adr, MPUA[cur], Word)
            tmp ← tmp + 4
            adr ← tmp Note 1, Note 2
            CheckException(MDP)
            Store-memory (adr, MPAT[cur], Word)
            tmp ← tmp + 4
            cur ← cur + 1
        }
    else
else
```

Note 1. The lower 2 bits of adr are masked by 0.

Note 2. An MDP exception may occur as a result of address calculation.

[Format] Format XI

[Opcode]



rrrrr indicates eh.
wwwww indicates et.
RRRRR indicates reg1.

[Flags]

CY	—
OV	—
S	—
Z	—
SAT	—

[Descriptions]

The word data of the MPU protection area setting system registers (MPLA, MPUA, and MPAT) is stored to the address generated from the word data of the general-purpose register reg1 according to the specified order. Word size is added to the address each time the word data of the system register is stored. The contents of these system registers is processed in ascending order, regardless of the value of MPIDX, from the entry number starting from eh to that starting from et (eh, eh+1, eh+2, …, et). The bank specified by MPBK is only to be processed.

[Supplement]

This instruction stores the target MPU protection area setting directly to memory. This instruction can perform the operation more effectively than by specifying the entry via MPIDX, reading the value of system register into a general-purpose register by STSR instructions, and storing it to memory by ST.W instructions.

The lower 2-bit address generated from the general-purpose register reg1 is masked by 0 and aligned on a word boundary. The general-purpose register reg1 retains the original value after the instruction execution is complete.

This instruction is an SV privilege instruction.

When this instruction is executed in guest mode and host managed entries are included for specified entries, the contents is stored to memory.

CAUTION

When an exception or an interrupt occurs during instruction execution and even if the contents of all system registers has not been stored to the memory, instruction execution can be aborted and exceptions or interrupts can be accepted, as long as the acceptance condition is satisfied. When the execution is suspended, it is impossible to know the contents of which system registers has been stored to the memory. After the return from exception processing, the suspended STM. MP instruction can be precisely re-executed as long as resources related to execution of the STM.MP instruction are not changed during exception processing, for the return PC from an exception is considered to be the PC of STM.MP instruction. This instruction re-execution restarts the STM.MP instruction processing from the start.

2.3 Cache Instructions

For detailed of Cache Instructions, see the relevant section of the *RH850G4MH User's Manual: Software*.

2.4 Floating-Point Instructions

For detailed of Floating-Point Instructions, see the relevant section of the *RH850G4MH User's Manual: Software*.

2.5 Extended Floating-Point Instructions

For detailed of Extended Floating-Point Instructions, see the relevant section of the *RH850G4MH User's Manual: Software*.

2.6 Virtualization Support Instructions

2.6.1 Overview of Virtualization Support Instructions

This CPU supports virtualization support instructions to help you build a virtual machine with virtualization software.

The following virtualization support instructions (mnemonics) are available:

- HVTRAP : Hypervisor EI-level Trap
- LDM.GSR : Load Multiple Guest System Registers from memory
- STM.GSR : Store Multiple Guest System Registers to memory

2.6.2 Virtualization Support Instruction Set

This section details each instruction, dividing each mnemonic (in alphabetical order) into the following items.

- Instruction format: Indicates how the instruction is written and its operand(s).
- Operation: Indicates the function of the instruction.
- Format: Indicates the instruction format.
- Opcode: Indicates the bit field of the instruction opcode..
- Description: Describes the operation of the instruction.
- Supplement: Provides supplementary information on the instruction.

2.6.2.1 HVTRAP

<Special instruction>

Hypervisor EI-level Trap

HVTRAP

[Instruction format] HVTRAP vector5

[Operation]

```
if ( HVCFG.HVE==1 )
then
    if ( PSW.UM==0 )
    then
        HMEIPC ← PC+4 (return PC)
        HMEIPSW ← HMPSW
        EIPSWH ← PSWH
        HMEIIC ← vector5(cause code)
        PSWH.GM ← 0
        HMPSW.UM ← 0
        HMPSW.EP ← 1
        HMPSW.ID ← 1
        PC ← exception handler address
    else
else
```

[Format] Format X

[Opcode]

15 0	31 16
00000111111vvvvv	0000000100010000

vvvvv indicates vector5.

[Flags]

CY	—
OV	—
S	—
Z	—
SAT	—

[Description]

When the HVTRAP instruction is executed while HVCFG.HVE is set (1), the CPU saves the return PC, current HMPSW and PSWH values in the HMEIPC, HMEIPSW, and EIPSWH, respectively, stores the exception cause code in the HMEIIC register, and updates the HMPSW and PSWH according to operation description. When the HVTRAP instruction is executed in the guest mode, the CPU enters the host mode. When the HVTRAP instruction is executed in the host mode, the CPU remains in the host mode.

[Supplement]

This instruction can be executed only when the virtualization support function is enabled. In addition, the instruction is an SV privilege instruction.

2.6.2.2 LDM.GSR

<Special instruction>

Load Multiple Guest System Registers from memory

LDM.GSR

Load Guest System Registers

[Instruction format] LDM.GSR [reg1]

[Operation]

```
if ( HVCFG.HVE==1 )
then
    if ( PSWH.GM==0 )
    then
        if ( PSW.UM==0 )
        then
            tmp ← reg1
            foreach (all system registers in the pre-defined list) {
                adr ← tmp Note 1, Note 2
                CheckException(MDP)
                SR[in list] ← Load-memory (adr, Word)
                tmp ← tmp + 4
            }
        else
    else
else
```

Note 1. The lower 2 bits of adr are masked by 0.

Note 2. An MDP exception may occur as a result of address calculation.

[Format] Format X

[Opcode]

15 0	31 16
000001111111RRRRR	1001100101100000

RRRRR indicates reg1.

[Flags]

CY	—
OV	—
S	—
Z	—
SAT	—

[Description]

This instruction reads word data from the address generated from the word data of the general-purpose register reg1 and sequentially stores it in the pre-defined system registers. Each time read data is stored in a system register, the word size is added to the address. For details of target system registers, see the hardware manual of the product used.

This instruction manipulates coprocessor system registers. Even when this instruction is executed if you do not have the corresponding coprocessor use permission, no coprocessor unusable exception occurs. Read data is stored in the coprocessor system registers.

[Supplement]

This instruction stores data directly from memory to multiple target system registers. This instruction can perform the operation more effectively than by reading memory into a general-purpose register by LD.W instructions, and storing it to the system register by LDSR instructions.

The lower 2-bit address generated from the general-purpose register reg1 is masked by 0 and aligned on a word boundary. The general-purpose register reg1 retains the original value after the instruction execution is complete.

This instruction can be executed only when the virtualization support function is enabled. In addition, the instruction is an HV privilege instruction.

CAUTION

When an exception or an interrupt occurs during instruction execution and even if not all of processing has been completed, the instruction execution can be aborted and the exception or interrupt can be accepted, when the acceptance condition is satisfied. When the execution is suspended, it is impossible to know to which system registers data from memory has been stored. After the return from exception processing, the suspended LDM.GSR instruction can be precisely re-executed as long as resources related to execution of the LDM.GSR instruction are not changed during exception processing, for the return PC from an exception is considered to be PC of this LDM.GSR instruction. This instruction re-execution restarts the LDM.GSR instruction processing from the start.

2.6.2.3 STM.GSR

<Special instruction>

Store Multiple Guest System Registers to memory

STM.GSR

Store of Guest System Registers

[Instruction format] STM.GSR [reg1]

[Operation]

```
if ( HVCFG.HVE==1 )
then
    if ( PSWH.GM==0 )
    then
        if ( PSW.UM==0 )
        then
            tmp ← reg1
            foreach (all system registers in the pre-defined list) {
                adr ← tmp Note 1, Note 2
                CheckException(MDP)
                Store-memory (adr, SR[in list], Word)
                tmp ← tmp + 4
            }
        else
    else
else
```

Note 1. The lower 2 bits of adr are masked by 0.

Note 2. An MDP exception may occur as a result of address calculation.

[Format] Format X

[Opcode]

15 0	31 16
00000111111RRRRR	1001000101100000

RRRRR indicates reg1.

[Flags]

CY	—
OV	—
S	—
Z	—
SAT	—

[Descriptions]

This instruction sequentially stores the word data of pre-defined system registers in the address generated from the word data of the general-purpose register reg1. Each time one of these system registers is stored, the word size is added to the address. For details of target system registers, see the hardware manual of the product used.

This instruction manipulates coprocessor system registers. Even when this instruction is executed if you do not have the corresponding coprocessor use permission, no coprocessor unusable exception occurs. The contents of the coprocessor system registers are stored in the memory.

[Supplement]

This instruction stores data directly from multiple target system registers to memory. The instruction can perform the operation more effectively than by reading the values of system registers into a general-purpose register by STSR instructions and storing them to memory by ST.W instructions.

The lower 2-bit address generated from the general-purpose register reg1 is masked by 0 and aligned on a word boundary. The general-purpose register reg1 retains the original value after the instruction execution is complete.

This instruction can be executed only when the virtualization support function is enabled. In addition, the instruction is an HV privilege instruction.

CAUTION

When an exception or an interrupt occurs during instruction execution and even if data from memory has not been stored to all system registers, the instruction execution can be aborted and the exception or interrupt can be accepted, when the acceptance condition is satisfied. When the execution is suspended, it is impossible to know which system register's data has been stored to memory. After the return from exception processing, the suspended STM.GSR instruction can be precisely re-executed as long as resources related to execution of the STM.GSR instruction are not changed during exception processing, for the return PC from an exception is considered to be PC of this STM.GSR instruction. This instruction re-execution restarts the STM.GSR instruction processing from the start.

Appendix A Number of Instruction Execution Clocks

A.1 Numbers of Clock Cycles for Execution

For detailed of Numbers of Clock Cycles for Execution, see the relevant section of the *RH850G4MH User's Manual: Software*.

A.2 Number of G4MH Instruction Execution Clocks

Legend of Execution Clocks

Symbol	Description
issue	When the other instruction is executed immediately after the execution of the current instruction
repeat	When the same instruction is repeated immediately after the execution of the current instruction
latency	When the following instruction uses the result of the current instruction

Types of Instructions	Mnemonics	Operand	Instruction Length (Number of Bytes)	Number of Execution Clocks		
				issue	repeat	latency
Special instruction	LDM.GSR	[reg1]	4	26 Note 1, Note 2	26 Note 1, Note 2	26 Note 1, Note 2
	LDM.MP	[reg1], eh-et	4	N+8 Note 2, Note 3	N+8 Note 2, Note 3	N+8 Note 2, Note 3
	STM.GSR	[reg1]	4	19	19	19
	STM.MP	eh-et, [reg1]	4	N+2 Note 2, Note 3	N+2 Note 2, Note 3	N+2 Note 2, Note 3
Special instruction (with branching)	HVTRAP	vector5	4	8	8	8

Note 1. If there are no wait states (cycles of waiting) associated with the memory access.

Note 2. Performs processing to synchronize pipeline.

Note 3. N depends on the total number of MPU entries specified in eh-et. Each entry has 3 registers, and since up to two registers are processed in one cycle, the value if there are no wait states will be as follows.

N = int (Number of saved and restored MPU entries x 1.5 + 0.5); however, N is in the range of 0 to 32.

RH850G4MH Virtualization
User's Manual: Software

Publication Date: Rev.0.90 April 17, 2020
Rev.1.10 June 30, 2021

Published by: Renesas Electronics Corporation

RH850G4MH Virtualization

Renesas Electronics Corporation

R01US0432EJ0110