Date: Jan. 24, 2022

RENESAS TECHNICAL UPDATE

TOYOSU FORESIA, 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan
Renesas Electronics Corporation

Product Category	MPU/MCU		Document No.	TN-RH8-B0373A/E	Rev.	1.00
Title	Additional restrictions for the location of RH850 G3M branch instructions		Information Category	Technical Notification		
Applicable Product	RH850/C1H, C1M RH850/D1x RH850/F1M, F1H, F1H-100 RH850/P1M, P1M-E, P1x-C	Lot No. All	Reference Document	See related documents below.		

1. <u>Precautions for use : Restrictions for the location of branch instructions.</u>

When the specific instruction combinations and the specific internal state are met, the CPU cannot continue the execution.

Therefore, instruction combinations that satisfy the following restrictions of 1, 2, 3, and 4 at the same time, are prohibited.

Restriction 1 : A “br” instruction (Instruction C[Note1]) that is backward (which means branching to the smaller address).

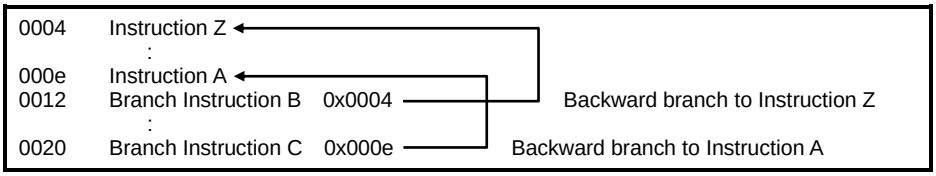
Restriction 2 : The branch destination address of Restriction 1 “br” instruction is 0x10*n+0xE, and this branch destination instruction (Instruction A[Note2]) is a 32-bit instruction with bit[26:25]=11.

Restriction 3 : A 16-bit/32-bit branch instruction (Instruction B[Note1, Note2, Note3]) that is backward just after the instruction of Restriction 2.

Restriction 4 : Instructions A, B, and C are located at the address range of 0x0000_0000 to 0x01FF_FFFF.

When CPU is in the state that execution cannot be continued, only a device initialization by a reset signal (e.g., reset by a watchdog timer) can release this state.

<Sample code>



Note 1: There is a possibility that instruction B and instruction C is the same instruction.

Note 2: Please refer to “7. Applicable Instructions”.

Note 3: In case of a "loop" instruction only, “Backward” includes branching to itself.

2. Descriptions of this phenomenon

When both static conditions and dynamic conditions shown below are met, the CPU may be in the state that the execution cannot be continued.

I. Static conditions

When meeting the following two conditions (a) and (b).

(a) When the instruction type, instruction location and branch destination of the instructions A, B, and C in the below table are met.

	Instruction type	Instruction location (Address)	Branch destination
Instruction A	32bit instruction and bit[26:25]=11[Note5]	0x10 * n + 0xe	-
Instruction B[Note4]	16bit/32bit branch instruction[Note5]	The next address of Instruction A	Backward[Note6]
Instruction C[Note4]	br instruction	Higher address than instruction A	Instruction A

<Sample code>

```
0004   Instruction Z
              :
000e   Instruction A
0012   Branch Instruction B   0x0004      Backward branch to Instruction Z
              :
0020   Branch Instruction C   0x000e      Backward branch to Instruction A
```

Note 4: There is a possibility that instruction B and instruction C is the same instruction.

Note 5: Please refer to “7. Applicable Instructions”.

Note 6: “Backward” means branching to a smaller address. In case of a "loop" instruction only, “Backward” includes branching to itself.

(b) Instructions A, B, and C are located at the address range of 0x0000_0000 to 0x01FF_FFFF.

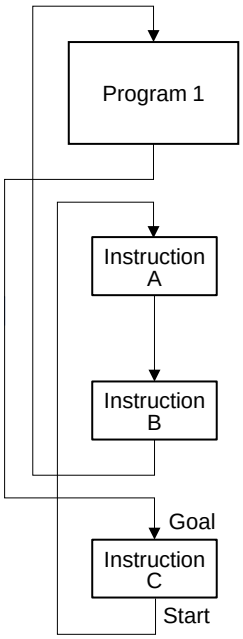
RENESAS

II. Dynamic conditions

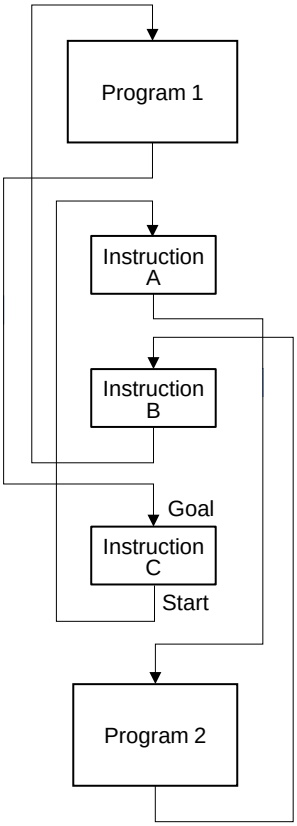
When meeting **neither following conditions (a) nor (b)** while the period between Instruction C execution and the next Instruction C execution (Program 1 or Program 2 shown in below figures).

(a) When executing Instruction E[Note7] even once.

(b) When executing 4 or more Branch instructions D[Note7] that are Backward[Note8] and located on different addresses.



Sequence Example 1
(Instruction A is not a branch instruction)



Sequence Example 2
(Instruction A is a branch instruction)

Note 7: Please refer to “7. Applicable Instructions”.

Note 8: “Backward” means branching to smaller address. In case of a "loop" instruction only, “Backward” includes branching to itself.

3. How to Judge

Please check whether static conditions are met, and please use the Check Tool of Static Condition that Renesas prepares to check the static conditions easier.

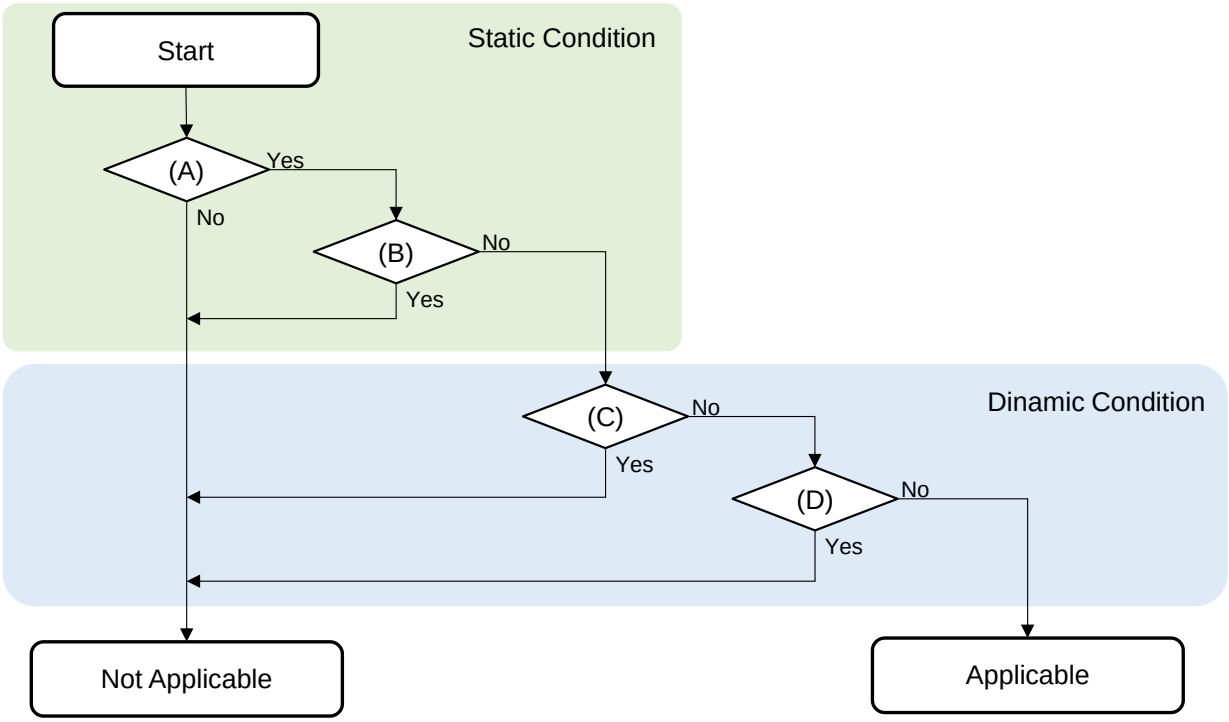
However, there is a possibility that the Check Tool of the Static Condition detects a point as applicable that is NOT applicable. So please confirm whether this detected point is a false detection or not.

If static conditions are met, please check whether either the dynamic conditions (a) or (b) are met. If either of these conditions is met, this phenomenon does not occur.

File format that Check Tool of Static Condition supports, is shown the below.

S-record file : Compiler-independent

<Judgment Flow>



(A)	Is there a point that the Check Tool detects as Applicable?
(B)	Are all points that the Check Tool detects as Applicable, a false detection? (8. False detection for S-record files by Check Tool)
(C)	In points that the Check Tool detects, is the Instruction E[Note9] executed even once whithin the period between Instruction C execution and the next Instruction C execution?
(D)	In points that the Check Tool detects, are 4 or more Branch Instructions D[Note9] that are Backward[Note10] and are located on different addresses, executed whithin the period between Instruction C execution and the next Instruction C execution?

Note 9 : Please refer to “7. Applicable Instructions”.

Note 10: “Backward” means branching to smaller address. In case of "loop" instruction only, “Backward” includes branching to itself.

4. Workarounds

When the branch destination of Instruction C is backward, “jr” instruction is used instead of “br”,

Alternatively, lowest 4-bit address of all Instruction A in applicable points must be different from 0xe by adjusting address of modules at linking time. In addition, there is a possibility that new applicable points occur by adjusting address of modules at linking time. Therefore, if new applicable points occur, please adjust the address again that these new points also become NOT applicable.

5. Future Action

We plan to apply these workarounds shown above by compiler modifications.

Please contact the supplier of the compiler about the schedule of the compiler modification.

6. Related documents

Document Title	Document Number
RH850/C1x Group User’s Manual: Hardware Rev.1.60	R01UH0414EJ0160
RH850/D1L/D1M Group User’s Manual: Hardware Rev.2.20	R01UH0451EJ0220
RH850/F1H Group User’s Manual: Hardware Rev 1.12	R01UH0445EJ0112
RH850/F1H PREMIUM 100pin Version User’s Manual: Hardware Rev1.00	R01UH0631EJ0100
RH850/F1M Group User’s Manual: Hardware Rev 1.03	R01UH0518EJ0103
RH850/P1L-C Group User’s Manual: Hardware Rev.1.10	R01UH0592EJ0110
RH850/P1M-E Group User’s Manual: Hardware Rev.1.10	R01UH0585EJ0110
RH850/P1x Group User’s Manual: Hardware Rev.1.40	R01UH0436EJ0140
RH850/P1x-C Group User’s Manual: Hardware Rev.1.30	R01UH0517EJ0130

7. Applicable Instructions

#	Instruction A (And bit[26:25]=11)	Instruction B	Instruction C
1	bcond disp17	bcond disp9 (including “br” instruction)	br disp9 (one of bcond disp9)
2	jarl disp22, reg2	bcond disp17	
3	jr disp22	jr disp22	
4	loop reg1, disp16	loop reg1, disp16	
5	addi imm16, reg1, reg2		
6	andi imm16, reg1, reg2		
7	clr1 bit#3, disp16[reg1]		
8	dispose imm5, list12		
9	dispose imm5, list12, [reg1]		
10	ld.b disp16[reg1], reg2		
11	ld.bu disp16[reg1], reg2		
12	ld.h disp16[reg1], reg2		
13	ld.hu disp16[reg1], reg2		
14	ld.w disp16[reg1], reg2		
15	movea imm16, reg1, reg2		
16	movhi imm16, reg1, reg2		
17	mulhi imm16, reg1, reg2		
18	not1 bit#3, disp16[reg1]		
19	ori imm16, reg1, reg2		
20	prepare list12, imm5		
21	prepare list12, imm5, sp/imm		
22	satsubi imm16, reg1, reg2		
23	set1 bit#3, disp16[reg1]		
24	st.b reg2, disp16[reg1]		
25	st.h reg2, disp16[reg1]		
26	st.w reg2, disp16[reg1]		
27	tst1 bit#3, disp16[reg1]		
28	xori imm16, reg1, reg2		

#	Instruction D	Instruction E
1	bcond disp9 (including “br” instruction)	eiret
2	bcond disp17	feret
3	jr disp22	synci
4	loop reg1, disp16	trap (only when executed in UM mode)
5	jr disp32	fetrap (only when executed in UM mode)
6		syscall (only when executed in UM mode)
7		rie (only when executed in UM mode)
8		ldsr reg1, PSW (only when changed the value of PSW.UM)

8. False detection for S-record files by Check Tool

Since it is impossible to distinguish between data and instruction code in case of an S-record file, there is a possibility that the Check Tool of Static Condition detects wrongly a point that is NOT applicable as applicable, due to the following factors.

This Check Tool searches the instruction code corresponding to “br” instruction (16-bit) of Instruction C by dividing ROM data into 16-bits unit. Therefore, there is a possibility that the Check Tool detects wrongly, by regarding a part of 32-bit / 48-bit / 64-bit instructions or table data originally as “br” instruction of Instruction C.

To determine whether it is false detection or not, please confirm using the debugger to display C source code and assembly code at the same time.