Date: October 14, 2014

RENESAS TECHNICAL UPDATE

1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan
Renesas Electronics Corporation
Inquiry http://japan.renesas.com/contact/
E-mail: csc@renesas.com

Product Category	MPU & MCU		Document No.	TN-RL*-A035A/E	Rev.	1.00
Title	RL78/L1C Limitation		Information Category	Technical Notification		
Applicable Products	RL78/L1C R5F110xx,R5F111xx	Lot No All lot	Reference Document	RL78/L1C User's Manual: Hardware Rev.2.00 R01UH0409EJ0200 (Feb. 2014)		

The restriction below applies to the divide instruction (DIVHU or DIVWU) in the above mentioned Applicable Products.

List of Restrictions to be notified

Item	New restrictions	Corresponding page
1	Restriction of the divide instruction (DIVHU, DIVWU)	p.2-p.6

Revision History

Revision history of RL78/L1C restrictions

Document Number	Date issued	Description
TN-RL*-A035A/E	October 14, 2014	First edition issued List of usage restriction: No. 1 (this document)

1. About the restrication

<Usage subject to the restriction>

If the software code applies to ALL of the four conditions below, the code is subject to the restriction.

1) A divide instruction (DIVHU or DIVWU) is executed in an interrupt service routine.
   A divide instruction (DIVHU or DIVWU) is defined as Group 1 instruction.
2) Multiple interrupts are enabled in the interrupt service routine in which the divide instruction (DIVHU or DIVWU) is executed.
3) More than one interrupts with different interrupt priorities occur during the process of the interrupt service routine mentioned in 2) above.
   Please refer to Table 1 for the detail of the priorities of the corresponding interrupts.
4) The divide instruction (DIVHU or DIVWU) is followed by a Group 2 instruction.
   Please refer to Item 5. "The List of Group 2 Instruction" for the details of Group 2 instructions. Please note, that any instruction is classified as "Group 2" if the preceding divide instruction is executed in RAM.

2. Details of the restriction

There is a possibility of unintended operation when branching from Interrupt A to Interrupt C, or branching from Interrupt C to Interrupt A.

I. A "Group 1" instruction (DIVHU, DIVWU) and a "Group 2" instruction are consecutive in Interrupt A in which multiple interrupts are enabled.
II. Interrupt B, whose request occurs during the process of Interrupt A, is suspended.
III. Interrupt C is generated during the two clock cycles just before the MCU completes the execution of the divide instruction (8th or 9th cycle for DIVHU, 16th or 17th cycle for DIVWU).

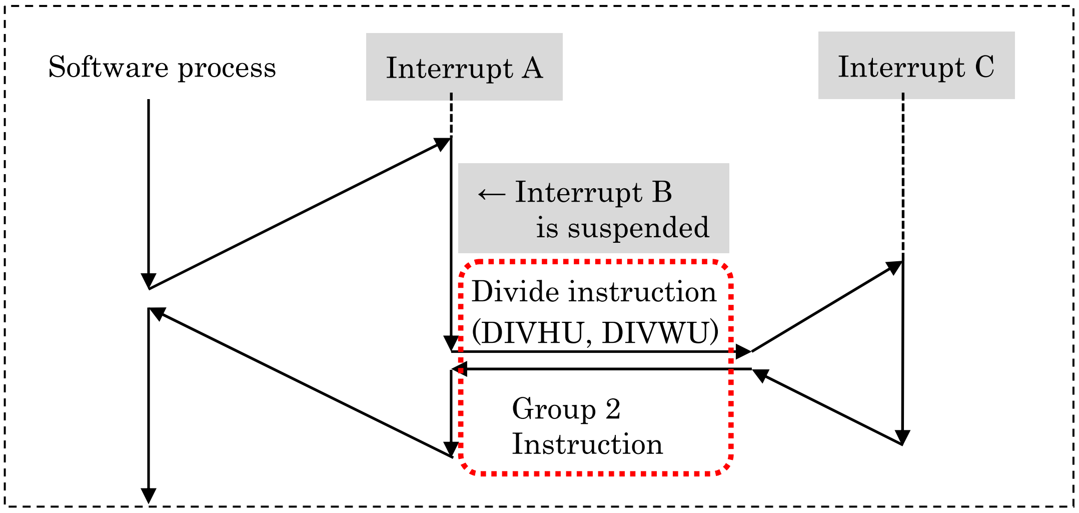

Figure1. Behavior subject to the restriction

Note 1: Please refer to Item 5. "The List of Group 2 Instruction" for the details of Group 2 instruction.

Note 2: Whether the MCU accepts an interrupt or not depends on the combination of the priority levels (0 to 3) of the interrupts. Table 1 shows the combinations subject to the restriction.

Table1. Combinations subject to the restriction

Priority level of Interrupt A	Priority level of Interrupt B	Priority level of Interrupt C	Restriction
Level 0	Level 1/ Level 2/ Level 3	Level 0	Your code could be subject to the restriction.
Level 1	Level 1/ Level 2/ Level 3	Level 0	
Level 2	Level 2/ Level 3	Level 0/ Level 1	
Level 3	Level 3	Level 0/ Level 1/ Level 2	
Other than above			The code is not subject to the restriction.

3. Software Workaround

Please implement one of the following software workaround.

（A）Disable interrupts during a divide or modulo operation.

Example:

__asm(“push PSW”);

DI();

Divide or modulo operation in C

__asm(“pop PSW”);

（B）Insert a NOP instruction immediately after the divide instruction.
Also, if the divide instruction (DIVHU or DIVWU) is executed in RAM, move it to code flash.

Example:

```
DIVWU        ; Divide instruction
NOP          ; Insert a NOP instruction
RET          ; Group 2 instruction
```

In the case of using a High-level language including C, compilers may generate the instructions subject to this restriction. In this case, take the workaround (A).

Note: In the case of Renesas compiler CA78K0, “#pragma di” should be declared in the code to use DI();

4. Permanent Measure

We will implement the software workaround into Renesas compiler CA78K0R V1.7.1.

Detail of the implementation:

CA78K0R V1.7.1 always inserts a NOP instruction immediately after each DIVWU / DIVHU instruction when building. This Implementation eliminates the need for the software workaround mentioned in Item 3. Software Workaround”.$^{\text{Note}}$

V1.7.1 Release Schedule: November 18, 2014

Note: If a divide instruction (DIVHU or DIVWU) is executed in RAM, code modification is required.

5. List of Group2 instruction

In the case a divide instruction (DIVHU or DIVWU) is followed by an instruction of Group2, it is subject to the restriction mentioned in this report. Instruction meeting one of the following conditions (Condition 1 to 3) is subject to Group2.

Condition 1: Instruction whose execution cycles are 2 or more.

Instruction	Operand
XCH	A, saddr
	A, sfr
	A, !addr16
	A, [DE]
	A, [DE+byte]
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]
ADD	saddr, #byte
ADDC	saddr, #byte
SUB	saddr, #byte
SUBC	saddr, #byte
AND	saddr, #byte
OR	saddr, #byte
XOR	saddr, #byte

Instruction	Operand
INC	saddr
INC	!addr16
INC	[HL+byte]
DEC	saddr
DEC	!addr16
DEC	[HL+byte]
INCW	saddrp
INCW	!addr16
INCW	[HL+byte]
DECW	saddrp
DECW	!addr16
DECW	[HL+byte]
MOV1	saddr.bit, CY
MOV1	sfr.bit, CY
MOV1	[HL].bit, CY
SET1	saddr.bit
SET1	sfr.bit
SET1	!addr16.bit
SET1	[HL].bit
CLR1	saddr.bit
CLR1	sfr.bit
CLR1	!addr16.bit
CLR1	[HL].bit

Instruction	Operand
CALL	All
CALLT	All
BRK	-
RET	-
RETI	-
RETB	-
BR	All
BC	All
BNC	All
BZ	All
BNZ	All
BH	All
BNH	All
BT	All
BF	All
BTCLR	All
HALT	-
STOP	-

Condition 2: Instruction reading the code flash memory or the mirror area.

Instruction in the tables below of reading the code flash memory or the mirror area is subject to "Group 2".

Instruction	Operand
MOV	A, !addr16
	A, [DE]
	A, [DE+byte]
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]
	A, word[B]
	A, word[C]
	A, word[BC]
	B, !addr16
	C, !addr16
	X, !addr16
	A, ES:!addr16
	A, ES:[DE]
	A, ES:[DE+byte]
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]
	A, ES:word[B]
	A, ES:word[C]
	A, ES:word[BC]
	B, ES:!addr16
	C, ES:!addr16
	X, ES:!addr16

Instruction	Operand
MOVW	AX, !addr16
	AX, [DE]
	AX, [DE+byte]
	AX, [HL]
	AX, [HL+byte]
	AX, word[B]
	AX, word[C]
	AX, word[BC]
	BC, !addr16
	DE, !addr16
	HL, !addr16
	AX, ES:!addr16
	AX, ES:[DE]
	AX, ES:[DE+byte]
	AX, ES:[HL]
	AX, ES:[HL+byte]
	AX, ES:word[B]
	AX, ES:word[C]
	AX, ES:word[BC]
	BC, ES:!addr16
	DE, ES:!addr16
	HL, ES:!addr16

Instruction	Operand
ADD ADDC SUB SUBC AND OR XOR	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]
CMP	A, !addr16
	A, [HL]
	A, [HL+byte]
	A, [HL+B]
	A, [HL+C]
	!addr16, #byte
	A, ES:!addr16
	A, ES:[HL]
	A, ES:[HL+byte]
	A, ES:[HL+B]
	A, ES:[HL+C]
	ES:!addr16, #byte
CMP0	!addr16
	ES:!addr16
CMPS	X, [HL+byte]
	X, ES:[HL+byte]

Instruction	Operand
ADDW	AX, !addr16
	AX, [HL+byte]
	AX, ES:!addr16
	AX, ES:[HL+byte]
SUBW	AX, !addr16
	AX, [HL+byte]
	AX, ES:!addr16
	AX, ES:[HL+byte]
CMPW	AX, !addr16
	AX, [HL+byte]
	AX, ES:!addr16
	AX, ES:[HL+byte]
MOV1	CY, [HL].bit
	CY, ES:[HL].bit
AND1	CY, [HL].bit
	CY, ES:[HL].bit
OR1	CY, [HL].bit
	CY, ES:[HL].bit
XOR1	CY, [HL].bit
	CY, ES:[HL].bit
BT	ES:[HL].bit, $addr20
BF	ES:[HL].bit, $addr20

Condition 3 : Instruction suspending interrupt requests.

Instruction listed in the table below that suspends interrupt requests, is subject to Group2.

Instruction	Operand
MOV	PSW, #byte
MOV	PSW, A
MOV1	PSW.bit, CY
SET1	PSW.bit
CLR1	PSW.bit
RETB	-
RETI	-
POP	PSW
BTCLR	PSW.bit,$addr20
EI	-
DI	-
SKC	-
SKNC	-
SKZ	-
SKNZ	-
SKH	-
SKNH	-

Instruction writing to the registers below is subject to Group2 since it suspends interrupt requests.

Writing to the registers below by register addressing is also subject to the condition3.

- Interrupt request flag register

  IF0L, IF0H, IF1L, IF1H, IF2L, IF2H, IF3L

- Interrupt mask flag register

  MK0L, MK0H, MK1L, MK1H, MK2L, MK2H, MK3L

- Priority specification flag register

  PR00L, PR00H, PR01L,PR01H, PR02L, PR02H, PR03L,

  PR10L, PR10H, PR11L, PR11H, PR12L, PR12H, PR13L

The table below shows the instruction writing to the registers listed above。

Instruction	Operand
MOV	sfr, #byte
MOV	!addr16, #byte
MOV	sfr, A
MOV	!addr16, A
MOV	[DE], A
MOV	[DE+byte], #byte
MOV	[DE+byte], A
MOV	[HL], A
MOV	[HL+byte], #byte
MOV	[HL+byte], A
MOV	[HL+B], A
MOV	[HL+C], A
MOV	word[B], #byte
MOV	word[B], A
MOV	word[C], #byte
MOV	word[C], A
MOV	word[BC], #byte
MOV	word[BC], A

Instruction	Operand
XCH	A, sfr
XCH	A, !addr16
XCH	A, [DE]
XCH	A, [DE+byte]
XCH	A, [HL]
XCH	A, [HL+byte]
XCH	A, [HL+B]
XCH	A, [HL+C]
ONEB	!addr16
CLRB	!addr16
MOVS	[HL+byte], X
MOVW	sfrp, #word
MOVW	sfrp, AX
MOVW	!addr16, AX
MOVW	[DE], AX
MOVW	[DE+byte], AX
MOVW	[HL], AX
MOVW	[HL+byte], AX
MOVW	word[B], AX
MOVW	word[C], AX
MOVW	word[BC], AX

Instruction	Operand
INC	!addr16
INC	[HL+byte]
DEC	!addr16
DEC	[HL+byte]
INCW	!addr16
INCW	[HL+byte]
DECW	!addr16
DECW	[HL+byte]
MOV1	sfr.bit, CY
MOV1	[HL].bit, CY
SET1	sfr.bit
SET1	!addr16.bit
SET1	[HL].bit
CLR1	sfr.bit
CLR1	!addr16.bit
CLR1	[HL].bit
BTCLR	sfr.bit, $addr20
BTCLR	[HL].bit, $addr20