

# RENESAS TECHNICAL UPDATE

1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan  
Renesas Electronics Corporation

Product Category	MCU		Document No.	MCCY-ZB-14-0025-1	Rev.	1.00
Title	R7F0C014 Usage Notifications of The Divide Instruction (DIVHU, DIVWU)		Information Category	Technical Notification		
Applicable Products	R7F0C014 (R7F0C014B2DFP-C, R7F0C014L2DFA-C, R7F0C014L2DFB-C)	Lot No	Reference Document	R7F0C014B2D, R7F0C014L2D User's Manual: Hardware Rev. 2.00 R01UH0442EJ0200		
		All lot				

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.7

## Revision History

Revision history of restrictions

Document Number	Date issued	Description
MCCY-ZB-14-0025-1	November 6, 2014	First edition issued List of usage restriction: No. 1 (this document)

**1. About the restriction**

<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 (8<sup>th</sup> or 9<sup>th</sup> cycle for DIVHU, 16<sup>th</sup> or 17<sup>th</sup> cycle for DIVWU).

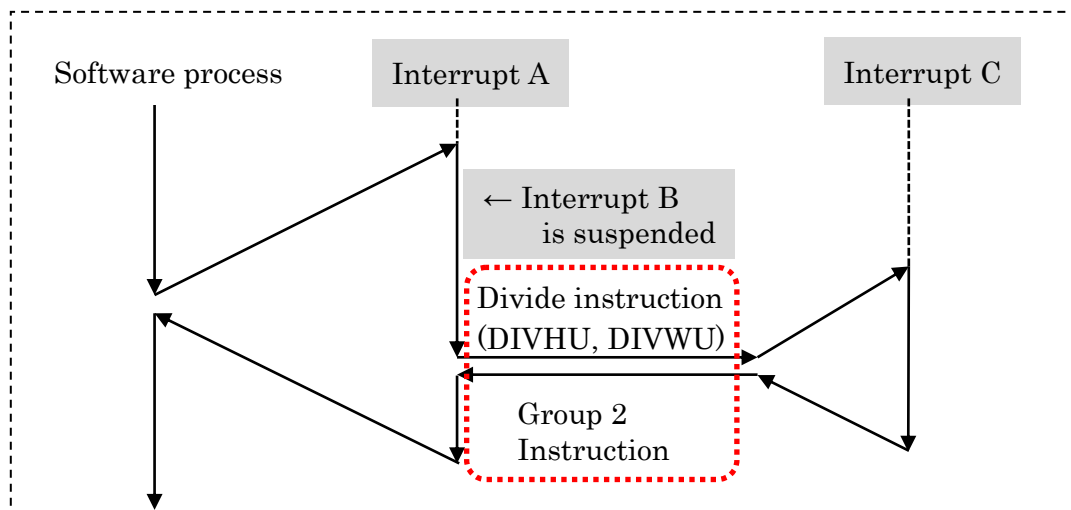


Figure1. Behavior subject to the restriction

- 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<sup>Note</sup>.

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	Instruction	Operand	Instruction	Operand
XCH	A, saddr	INC	saddr	CALL	All
	A, sfr	INC	!addr16	CALLT	All
	A, !addr16	INC	[HL+byte]	BRK	-
	A, [DE]	DEC	saddr	RET	-
	A, [DE+byte]	DEC	!addr16	RETI	-
	A, [HL]	DEC	[HL+byte]	RETB	-
	A, [HL+byte]	INCW	saddrp	BR	All
	A, [HL+B]	INCW	!addr16	BC	All
	A, [HL+C]	INCW	[HL+byte]	BNC	All
ADD	saddr, #byte	DECW	saddrp	BZ	All
ADDC	saddr, #byte	DECW	!addr16	BNZ	All
SUB	saddr, #byte	DECW	[HL+byte]	BH	All
SUBC	saddr, #byte	MOV1	saddr.bit, CY	BNH	All
AND	saddr, #byte	MOV1	sfr.bit, CY	BT	All
OR	saddr, #byte	MOV1	[HL].bit, CY	BF	All
XOR	saddr, #byte	SET1	saddr.bit	BTCLR	All
		SET1	sfr.bit	HALT	-
		SET1	!addr16.bit	STOP	-
		SET1	[HL].bit		
		CLR1	saddr.bit		
		CLR1	sfr.bit		
		CLR1	!addr16.bit		
		CLR1	[HL].bit		

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	Instruction	Operand	Instruction	Operand	Instruction	Operand
MOV	A, !addr16	MOVW	AX, !addr16	ADD ADDC SUB SUBC AND OR XOR	A, !addr16	ADDW	AX, !addr16
	A, [DE]		AX, [DE]		A, [HL]		AX, [HL+byte]
	A, [DE+byte]		AX, [DE+byte]		A, [HL+byte]		AX, ES:!addr16
	A, [HL]		AX, [HL]		A, [HL+B]		AX, ES:[HL+byte]
	A, [HL+byte]		AX, [HL+byte]		A, [HL+C]		AX, !addr16
	A, [HL+B]		AX, word[B]		A, ES:!addr16		AX, [HL+byte]
	A, [HL+C]		AX, word[C]		A, ES:[HL]		AX, ES:!addr16
	A, word[B]		AX, word[BC]		A, ES:[HL+byte]		AX, ES:[HL+byte]
	A, word[C]		BC, !addr16		A, ES:[HL+B]		AX, ES:[HL+byte]
	A, word[BC]		DE, !addr16		A, ES:[HL+C]		AX, !addr16
	B, !addr16	HL, !addr16	A, !addr16	AX, [HL+byte]			
	C, !addr16	AX, ES:!addr16	A, [HL]	AX, ES:!addr16			
	X, !addr16	AX, ES:[DE]	A, [HL+byte]	AX, ES:[HL+byte]			
	A, ES:!addr16	AX, ES:[DE+byte]	A, [HL+B]	AX, ES:[HL+byte]			
	A, ES:[DE]	AX, ES:[HL]	A, [HL+C]	CY, [HL].bit			
	A, ES:[DE+byte]	AX, ES:[HL+byte]	!addr16, #byte	CY, ES:[HL].bit			
	A, ES:[HL]	AX, ES:word[B]	A, ES:!addr16	CY, [HL].bit			
	A, ES:[HL+byte]	AX, ES:word[C]	A, ES:[HL]	CY, ES:[HL].bit			
	A, ES:[HL+B]	AX, ES:word[BC]	A, ES:[HL+byte]	CY, [HL].bit			
	A, ES:[HL+C]	BC, ES:!addr16	A, ES:[HL+B]	CY, ES:[HL].bit			
A, ES:word[B]	DE, ES:!addr16	A, ES:[HL+C]	CY, [HL].bit				
A, ES:word[C]	HL, ES:!addr16	ES:!addr16, #byte	CY, ES:[HL].bit				
B, ES:!addr16		!addr16	BT	ES:[HL].bit, \$addr20			
C, ES:!addr16		ES:!addr16	BF	ES:[HL].bit, \$addr20			
X, ES:!addr16		X, [HL+byte]					
		X, ES:[HL+byte]					

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
- Interrupt mask flag register  
MK0L, MK0H, MK1L, MK1H, MK2L, MK2H
- Priority specification flag register  
PR00L, PR00H, PR01L, PR01H, PR02L, PR02H,  
PR10L, PR10H, PR11L, PR11H, PR12L, PR12H

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

Instruction	Operand	Instruction	Operand	Instruction	Operand
MOV	sfr, #byte	XCH	A, sfr	INC	!addr16
MOV	!addr16, #byte	XCH	A, !addr16	INC	[HL+byte]
MOV	sfr, A	XCH	A, [DE]	DEC	!addr16
MOV	!addr16, A	XCH	A, [DE+byte]	DEC	[HL+byte]
MOV	[DE], A	XCH	A, [HL]	INCW	!addr16
MOV	[DE+byte], #byte	XCH	A, [HL+byte]	INCW	[HL+byte]
MOV	[DE+byte], A	XCH	A, [HL+B]	DECW	!addr16
MOV	[HL], A	XCH	A, [HL+C]	DECW	[HL+byte]
MOV	[HL+byte], #byte	ONEB	!addr16	MOV1	sfr.bit, CY
MOV	[HL+byte], A	CLRB	!addr16	MOV1	[HL].bit, CY
MOV	[HL+B], A	MOVW	[HL+byte], X	SET1	sfr.bit
MOV	[HL+C], A	MOVW	sfrp, #word	SET1	!addr16.bit
MOV	word[B], #byte	MOVW	sfrp, AX	SET1	[HL].bit
MOV	word[B], A	MOVW	!addr16, AX	CLR1	sfr.bit
MOV	word[C], #byte	MOVW	[DE], AX	CLR1	!addr16.bit
MOV	word[C], A	MOVW	[DE+byte], AX	CLR1	[HL].bit
MOV	word[BC], #byte	MOVW	[HL], AX	BTCLR	sfr.bit, \$addr20
MOV	word[BC], A	MOVW	[HL+byte], AX	BTCLR	[HL].bit, \$addr20
		MOVW	word[B], AX		
		MOVW	word[C], AX		
		MOVW	word[BC], AX		

Inquiry

Renesas Electronics (China) Co., Ltd.

Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China

Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.

Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333

Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited

Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong

Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics India Pvt. Ltd.

No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India

Tel. +91-80-67208700, Fax. +91-80-67208777