[Notes]
C/C++ Compiler Package for M16C Series and R8C Family,
C Compiler Package for M32C Series,
C Compiler Package for R32C Series

R20TS0750EJ0100
Rev.1.00
Oct. 01, 2021

Outline

When using any of the products in the title, note the following points.

1. Performing the addition/subtraction of single-precision floating-point data
2. Performing the addition/subtraction of double-precision floating-point data (1)
3. Performing the addition/subtraction of double-precision floating-point data (2)

1. Performing the addition/subtraction of single-precision floating-point data

1.1 Applicable Products

- C/C++ Compiler Package for M16C Series and R8C Family V.1.00 Release 1 to V.5.40 Release 00
- C Compiler Package for M32C Series V.1.00 Release 1 to V.5.42 Release 00
- C Compiler Package for R32C Series V.1.01 Release 00 to V.1.02 Release 01

1.2 Details

The result of adding/subtracting single-precision floating-point data may become ±1.175494e-38 (the normalized number that can be expressed in the single-precision floating-point format that has a minimum absolute value) under certain conditions. (Note 1)

Note 1: In C Compiler Package for M32C Series, the result may become ±1.175494e-38 or ±0 under certain conditions.

1.3 Conditions

The problem may arise if all the conditions from (1) to (3) are met.

(1) The code that performs the addition/subtraction of single-precision floating-point data is specified.

(2) The output code for (1) is the call of the runtime library function __f4add or __f4sub.

(3) When the operation in (1) is expressed as an addition (a+b or b+a), the sign, exponent, and mantissa parts for the applicable products, versions, and options meet the following tables.

- C/C++ Compiler Package for M16C Series and R8C Family V.1.00 Release 1 to V.5.30 Release 02

Either Condition 1 or 2 is met.

Condition	Sign of a and b	Exponent part of a	Mantissa part of a	Exponent part of b	Mantissa part of b
1	Different	0x19 to 0xFD	0x7FFFFF	A value one larger than the exponent part of a	0x000000
2	Different	0x18 to 0xFD	0x7FFFFE	A value one larger than the exponent part of a	0x000000

- C/C++ Compiler Package for M16C Series and R8C Family V.5.40 Release 00

Sign of a and b	Exponent part of a	Mantissa part of a	Exponent part of b	Mantissa part of b
Different	0x19 to 0xFD	0x7FFFFF	A value one larger than the exponent part of a	0x000000

- C Compiler Package for M32C Series V.1.00 Release 1 to V.5.42 Release 00

Sign of a and b	Exponent part of a	Mantissa part of a	Exponent part of b	Mantissa part of b
Different	0x19 to 0xFD	0x7FFFFF	A value one larger than the exponent part of a	0x000000

When the option -M82 (*1) or -M90 (*2) is used, the operation result may become ±0, not ±1.175494e-38.

*1: Applicable when -M82 is specified in V.5.20 Release 1 or later.

*2: Applicable when -M90 is specified in V.5.40 Release 00 or later.

- C Compiler Package for R32C Series V.1.01 Release 00 to V.1.02 Release 01

Sign of a and b	Exponent part of a	Mantissa part of a	Exponent part of b	Mantissa part of b
Different	0x19 to 0xFD	0x7FFFFF	A value one larger than the exponent part of a	0x000000

Applicable when the option -fuse_FPU is not used.

1.4 Example

Below is an example of an occurrence of the problem. The result of the addition/subtraction of float becomes -1.175494e-38, which is different from the expected value.

```
/* tp.c */
float a = 127.999992;  /* 0x42FFFFFF (3)*/
float b = -128;        /* 0x43000000 (3)*/
float c;
 void f1(void) {
  c = a + b;  /* (1), (2) */
}
```

1.5 Workaround

If you are using C/C++ Compiler Package for M16C Series and R8C Family, use V.5.42 Release 00 or later instead.

If degrading the precision of the operation result is permitted, you can prevent the problem by taking the following measure.

1.6 Countermeasure

You can reduce the error in the operation results by changing the values so that condition (3) is not met. The following shows the example in which the low order 2 bits of the mantissa part are masked so that the condition is not met.

```
float a = 127.999992; /* 0x42FFFFFF */
float b = -128;       /* 0xC3000000 */
float c;

float maskTwoBit(float x) {
  union {
    float flt;
    unsigned long ul;
  } u;
  u.flt = x;
```

```
  if ((u.ul & 0x7FFFFEUL) == 0x7FFFFEUL) {
    u.ul &= 0xFFFFFFFCUL;
  }
  return u.flt;
}

void main(void) {
  a = maskTwoBit(a);   //  added
  b = maskTwoBit(b);   //  added
  c = a + b;
}
```

In the example, the operation result (c) is -0.000000000000028421709430.

1.7 Schedule for Fixing the Problem

We do not plan to fix it.

2. Performing the addition/subtraction of double-precision floating-point data (1)

2.1 Applicable Products

- C/C++ Compiler Package for M16C Series and R8C Family V.1.00 Release 00 to V.5.40 Release 00
- C Compiler Package for M32C Series V.1.00 Release 1 to V.5.40 Release 00

2.2 Details

If the result of adding/subtracting double-precision floating-point data meets certain conditions, the error in the operation result may become greater than expected. Note if the subtraction of a close value results in cancellation of significant digits, the error becomes a maximum of ±1.00e-13.

2.3 Conditions

The problem may arise if all the conditions from (1) to (3) are met.

(1) The code that performs the addition/subtraction of double-precision floating-point data is specified.

(2) The output code for (1) is the call of the runtime library function __f8add or __f8sub.

(3) One of the conditions from (3-1) to (3-3) is met.

(3-1) When the operation in (1) is expressed as an addition (a+b or b+a), the conditions in (3-1-1) and (3-1-2) are met.

(3-1-1) The sign and exponent parts of a and b are the same.

(3-1-2) The following bits are 1 in the mantissa part of a or b.

- C/C++ Compiler Package for M16C Series and R8C Family

  Bit 1(*) to bit 8

- C Compiler Package for M32C Series

  Bit 0 to bit 7

  *The bits are numbered so that the LSB is bit 0. This applies throughout this document.

(3-2) When the operation in (1) is expressed as an addition (a+b or b+a), in which the exponent part of a is smaller than that of b and their difference is expressed as d, the conditions in (3-2-1) and (3-2-2) are met.

(3-2-1) The value of d meets one of the following.

- In the range from 1 to 7
- In the range from 9 to 15
- In the range from 17 to 23
- In the range from 25 to 31
- In the range from 33 to 39
- In the range from 41 to 45

(3-2-2) The following bits are 1 in the mantissa part of a or b (including the implicit leading bit).

- C/C++ Compiler Package for M16C Series and R8C Family

  Bit d to bit (d+7)

- C Compiler Package for M32C Series

  Bit (d-1) to bit (d+7)

(3-3) When the operation in (1) is expressed as an addition (a+b or b+a), in which the exponent part of a is smaller than that of b and their difference is expressed as d, the conditions in (3-3-1) and (3-3-2) are met.

(3-3-1) The sign of a and b are the same.

(3-3-2) The mantissa part of a, which includes the implicit leading bit, is expressed as a1m. The mantissa part of b, which does not include the implicit leading bit, is expressed as b0m. The calculation result of ((a1m >> d) + b0m) or ((a1m >> d) + b0m + 1) becomes the following.

- C/C++ Compiler Package for M16C Series and R8C Family

  The eight bits from bit 1 to bit 8 and bit 52 are 1.

- C Compiler Package for M32C Series

  The nine bits from bit 0 to bit 8 and bit 52 are 1.

2.4 Example

Below is an example of an occurrence of the problem.

```
void main(void) {
  /* Condition (3-1) */
  double a = 255.99999999999997; /* 0x406FFFFFFFFFFFF */
  double b = 128.0; /* 0x4060000000000000 */
  double c = a + b; /* (1)(2) */
  /* Although the expected value of c is 384.000…(0x4078000000000000),
   * the result is 383.99999999998545(0x4077FFFFFFFFF00). */

  /* Condition (3-2) */
  a = 127.999999999999992; /* 0x405FFFFFFFFFFFF */
  b = -128.0; /* 0xC060000000000000 */
  c = a + b; /* (1)(2) */
  /* Although the expected value of c is
   * -0.000000000000014210…(0xBD10000000000000), the result is
   * -0.000000000007275957…(0xBDA0000000000000).   */

  /* Condition (3-3) */
  a = 127.99999999999955; /* 0x405FFFFFFFFFFE0 */
  b = 192.0000000000004;  /* 0x4068000000000000E */
  c = a + b; /* (1)(2) */
  /* Although the expected value of c is
   * 319.99999999999994(0x4073FFFFFFFFFFF), the result is
   * 319.99999999998545(0x4073FFFFFFFFF00).   */
}
```

2.5 Workaround

If you are using C/C++ Compiler Package for M16C Series and R8C Family, use V5.42 Release 00 or later instead.

If you are using C Compiler Package for M32C Series, use V.5.41 Release 00 or later instead.

Note that "3. Performing the addition/subtraction of double-precision floating-point data (2)" may apply. Read the following sections as well.

2.6 Schedule for Fixing the Problem

We do not plan to fix it.

3. Performing the addition/subtraction of double-precision floating-point data (2)

3.1 Applicable Products

- C/C++ Compiler Package for M16C Series and R8C Family V.5.42 Release 00 to V.6.00 Release 00
- C Compiler Package for M32C Series V.5.41 Release 00 to V.5.42 Release 00
- C Compiler Package for R32C Series V.1.01 Release 00 to V.1.02 Release 01

3.2 Details

The result of adding/subtracting double-precision floating-point data may become ±2.2250738585072014e-308 (the minimum value that can be expressed in the double-precision floating-point format).

3.3 Conditions

The problem may arise if all the conditions from (1) to (3) are met.

(1) The code that performs the addition/subtraction of double-precision floating-point data is specified.

(2) The output code for (1) is the call of the runtime library function __f8add or __f8sub.

(3) When the operation in (1) is expressed as an addition (a+b or b+a), the sign, exponent, and mantissa parts for the applicable products and versions meet the following tables.

- C/C++ Compiler Package for M16C Series and R8C Family V.5.42 Release 00 to V.6.00 Release 00

Either Condition 1 or 2 is met.

Condition	Sign of a and b	Exponent part of a	Mantissa part of a	Exponent part of b	Mantissa part of b
1	Different	0x36 to 0x7FD	0xFFFFFFFFFFFF	A value one larger than the exponent part of a	0x0000000000000
2	Different	0x35 to 0x7FD	0xFFFFFFFFFFFE	A value one larger than the exponent part of a	0x0000000000000

- C Compiler Package for M32C Series V.5.41 Release 00 to V.5.42 Release 00

Sign of a and b	Exponent part of a	Mantissa part of a	Exponent part of b	Mantissa part of b
Different	0x36 to 0x7FD	0xFFFFFFFFFFFF	A value one larger than the exponent part of a	0x0000000000000

- C Compiler Package for R32C Series V.1.01 Release 00 to V.1.02 Release 01

Sign of a and b	Exponent part of a	Mantissa part of a	Exponent part of b	Mantissa part of b
Different	0x36 to 0x7FD	0xFFFFFFFFFFFF	A value one larger than the exponent part of a	0x0000000000000

3.4 Example

Below is an example of an occurrence of the problem.

```
/* tp.c */
double a = 127.99999999999999;  /* 0x405FFFFFFFFFFFF (3)*/
double b = -128;                /* 0xC060000000000000 (3)*/
double c;
void f1(void) {
  c = a + b;  /* (1), (2) */
}
```

In the example, the operation result (variable c) of 127.99999999999999 (variable a) + -128.0 (variable b) should be -0.000000000000142108547152..., but the result becomes -2.2250738585072014e-308.

3.5 Countermeasure

You can reduce the error in the operation results by changing the values so that condition (3) is not met. The following shows the example in which the low order 2 bits of the mantissa part are masked so that the condition is not met.

```
double a = 127.99999999999999;
double b = -128;
double c;

double maskTwoBit(double x) {
  union {
    double dbl;
    unsigned long long ull;
  } u;
  u.dbl = x;
  if ((u.ull & 0xFFFFFFFFFFFFEULL) == 0xFFFFFFFFFFFFEULL) {
    u.ull &= 0xFFFFFFFFFFFFFFFCULL;
  }
  return u.dbl;
}

void main(void) {
  a = maskTwoBit(a);   //  added
  b = maskTwoBit(b);   //  added
  c = a + b;
}
```

In the example, the operation result (c) is -0.000000000000028421709430.

3.6 Schedule for Fixing the Problem

We do not plan to fix it.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.01.21	-	First edition issued

Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

The past news contents have been based on information at the time of publication. Now changed or invalid information may be included.

The URLs in the Tool News also may be subject to change or become invalid without prior notice.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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/

TS Colophon 4.2