To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

APPLICATION NOTE

H8/300H Tiny Series

Conversion from Single-Precision Floating-Point to Signed 32-bit Binary (FKTR)

Introduction

Converts a single-precision floating-point number in general registers to a signed 32-bit binary number.

Target Device

H8/300H Tiny Series

Contents

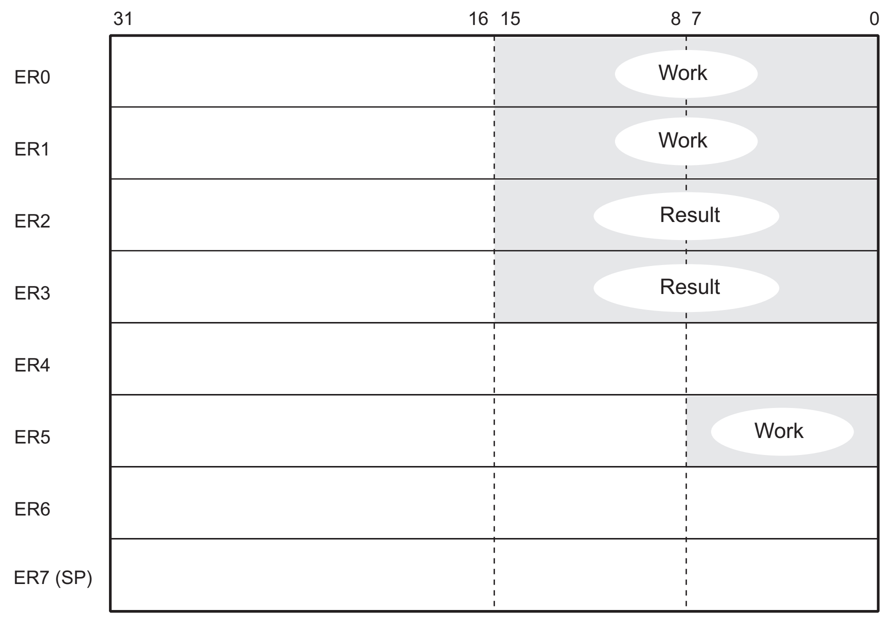
1. Function

1. Converts a single-precision floating-point number in general registers to a signed 32-bit binary number.
2. When the single-precision floating-point number is a zero, the output is also zero.
3. When the single-precision floating-point number has an absolute value greater than or equal to 2^{31}, the maximum value with the corresponding sign ($2^{31} - 1$ or -2^{31}) is output. Zero is output in response to absolute values less than one.

2. Arguments

Contents		Storage Location	Data Length (Bytes)
Input	Single-precision floating-point number	R0, R1	4
Output	Signed 32-bit binary number	R2, R3	4

3. Changes to Internal Registers and Flags



I	UI	H	U	N	Z	V	C
—	—	↕	—	↕	↕	↕	↕

—: No change
↕: Varies
0 : Fixed to 0
1 : Fixed to 1

4. Programming Specifications:

Program memory (bytes)	100
Data memory (bytes)	0
Stack (bytes)	0
Number of cycles	108
Re-entrant	Yes
Relocatable	Yes
Interrupts during execution	Yes

5. Note

The number of cycles given in the programming specifications is the value for execution of the example in figure 1.
For details on the floating-point data format, refer to Reference: Description of Single-Precision Floating-Point Formats.

6. Descriptions

6.1 Descriptions of Functions

1. The arguments are listed below.
   1) Set the input argument.
      R0: higher-order two bytes of the single-precision floating-point number
      R1: lower-order two bytes of the single-precision floating-point number
   2) The FKTR subroutine sets the output argument.
      R2: higher-order two bytes of the signed 32-bit binary number
      R3: lower-order two bytes of the signed 32-bit binary number

2. The following figure illustrates the execution of the FKTR subroutine. When the input argument is set as shown below, the subroutine places the result of conversion in R2 and R3.

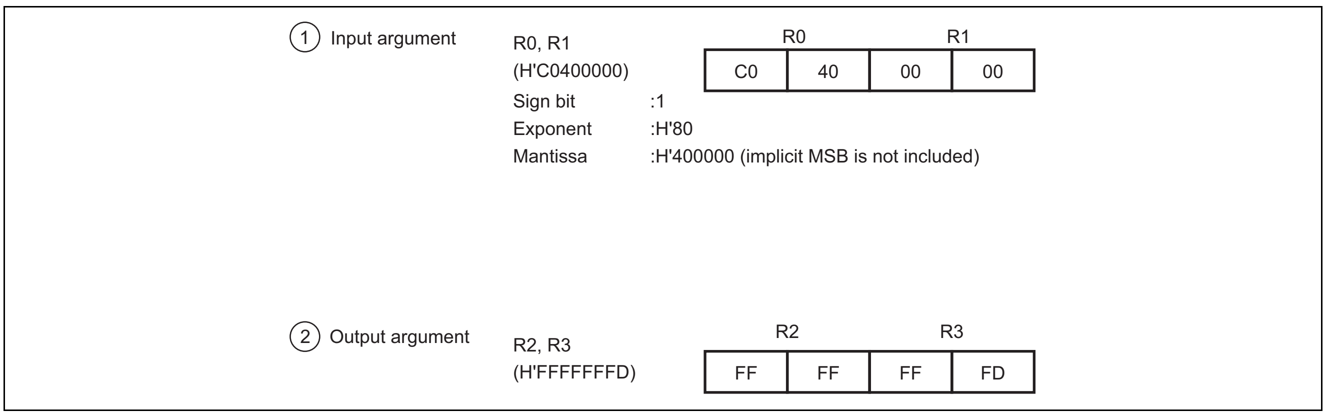


Figure 1 Example of FKTR Execution

6.2 Usage Notes

1. Zero is the output when the single-precision floating-point number is zero or has an absolute value smaller than one.
2. When the absolute value of the single-precision floating-point number is 2^{31} or greater, the maximum value with the same sign (H'7FFFFFFF or H'80000000) is output.
3. The input argument set in R0 and R1 is lost in the execution of FKTR. When you will still require the input argument, save it elsewhere in memory before executing FKTR.

6.3 Description of Data Memory

No data memory is used by FKTR.

6.4 Example of Usage

After setting a single-precision floating-point number in the general registers, call the FKTR subroutine.

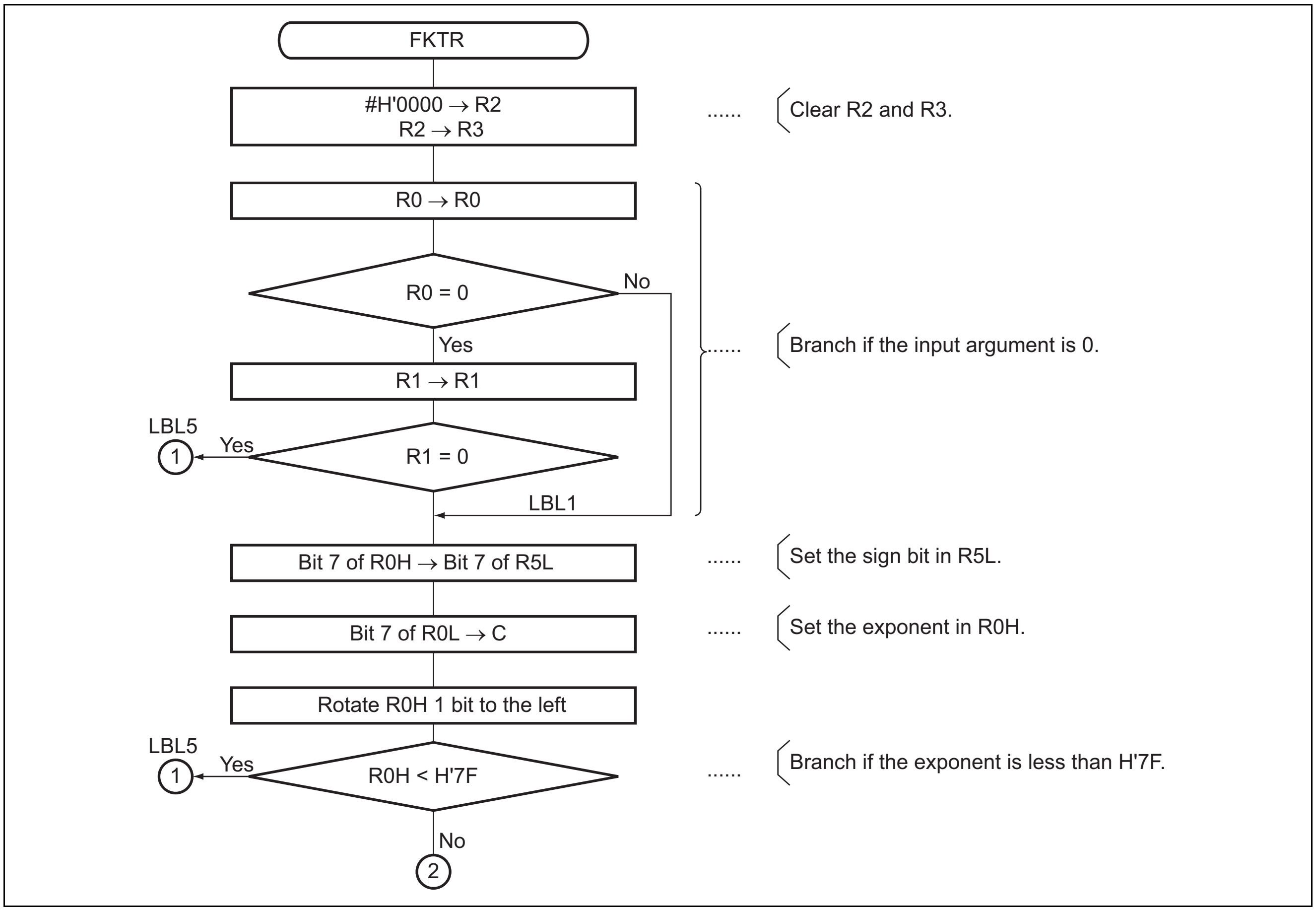
```
WORK1    . RES. W 2,0          ········· Reservation of the data memory area for setting of the single-precision floating point number by the
                                         user program.
WORK2    . RES. W 2,0          ········· Reservation of the data memory area where the signed 32-bit binary number will be placed for the
            .                            user program.
            .
            .
         MOV. W @WORK1, R0     ········· Sets the single-precision floating point number specified by the user program as the input
                                         argument
         MOV. B @WORK1+2, R1

         JSR    @FKTR          ········· Subroutine call of the software FKTR

         MOV. W R2, @WORK2     ········· Transfers the signed 32-bit binary number from the output argument to the data memory area.

         MOV. W R3, @WORK2+2
```

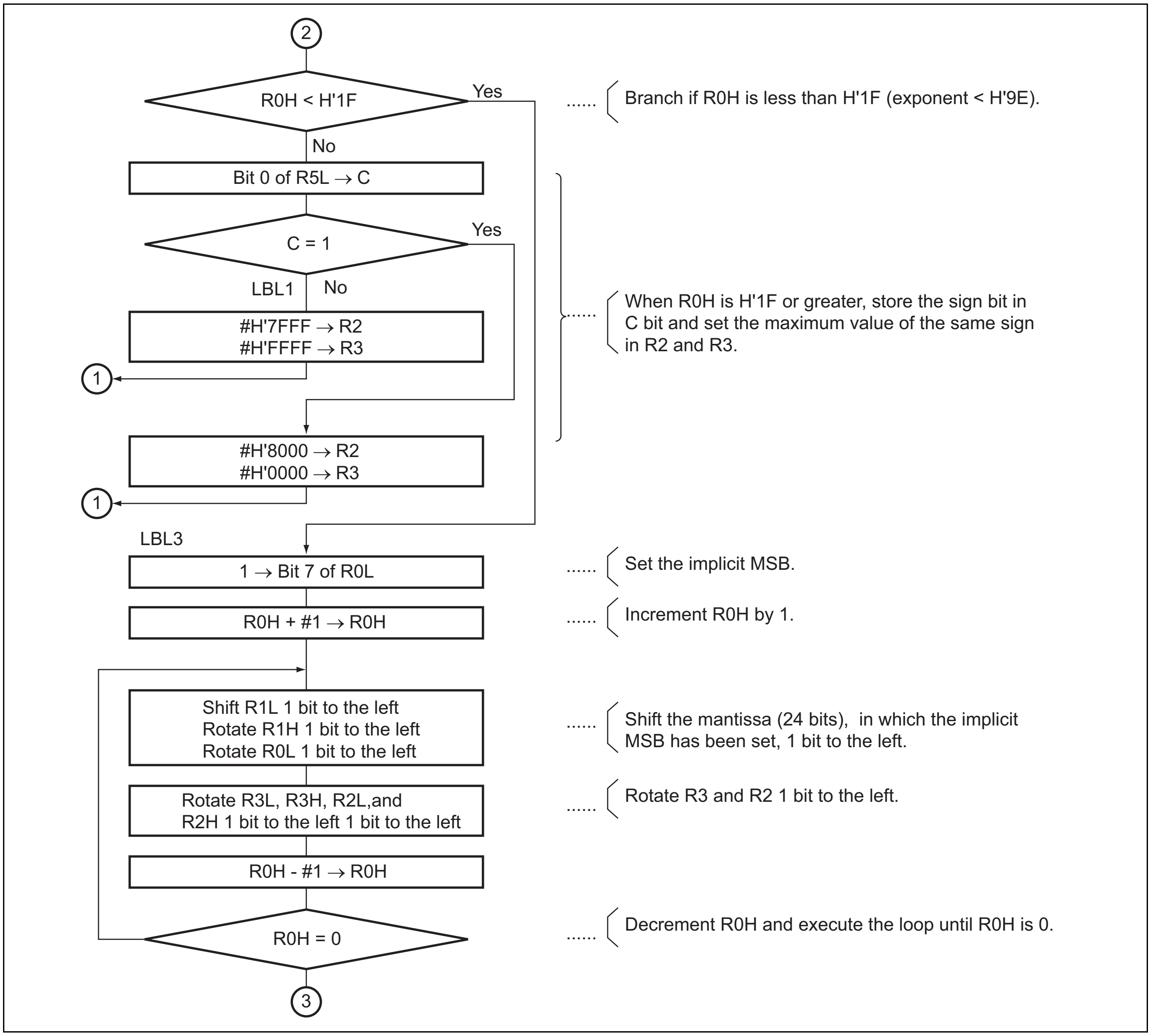
6.5 Principles of Operation

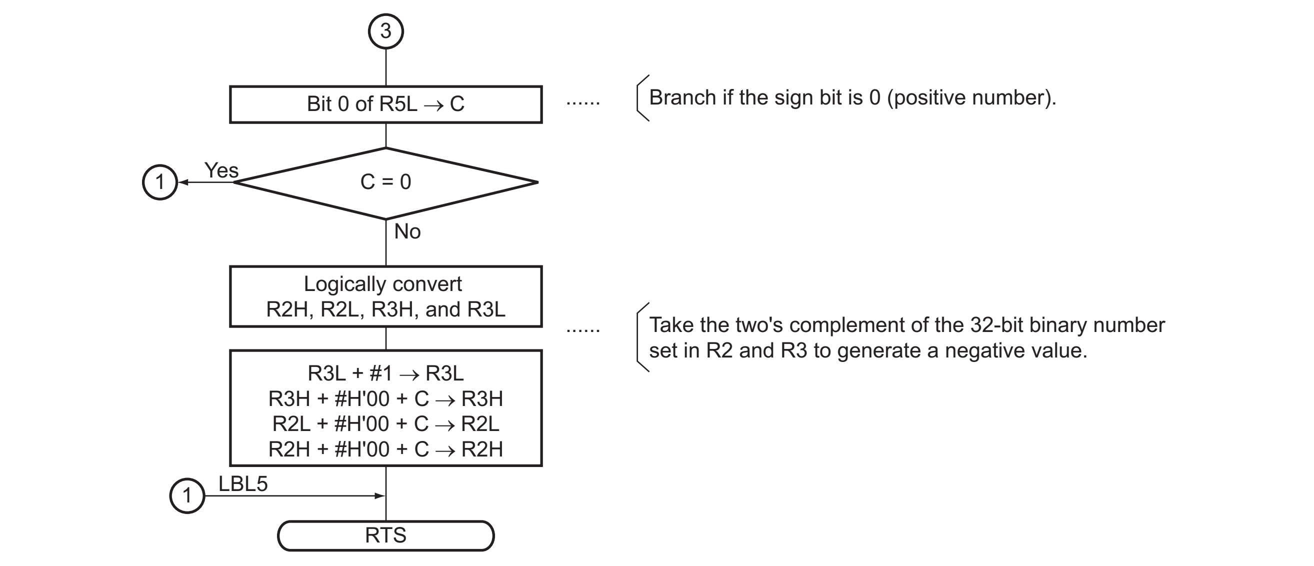
The FKTR subroutine converts the single-precision floating-point number to a signed 32-bit binary number in the following sequence.

1. Firstly, FKTR checks the input argument.
   1) If the single-precision floating-point number is zero, zero is output.
   2) If the exponent is less than H'7F, zero is output.
   3) If the exponent is H'9E or greater, the maximum value with the same sign is output.
2. When the input argument is not zero and its absolute value is one or greater (i.e., exponent is at least H'7F) but less than 2^{31} (i.e., exponent is less than H'9E), the subroutine:
   1) sets the implicit MSB;
   2) shift the mantissa (24 bits), in which the implicit MSB has been set, 1 bit to the left;
   3) rotates R3and R2 one bit to the left;
   4) repeats steps 2) and 3) (R0H +1) times;
   5) tests the sign bit—if it is negative, takes the two's complement to make the number negative.

7. Flowchart


FKTR
#H'0000 → R2
R2 → R3
Clear R2 and R3.
R0 → R0
R0 = 0
No
Yes
R1 → R1
LBL5
Yes
1
R1 = 0
LBL1
Branch if the input argument is 0.
Bit 7 of R0H → Bit 7 of R5L
Set the sign bit in R5L.
Bit 7 of R0L → C
Set the exponent in R0H.
Rotate R0H 1 bit to the left
LBL5
Yes
1
R0H < H'7F
Branch if the exponent is less than H'7F.
No
2


2
R0H < H'1F
Yes
No
Bit 0 of R5L → C
C = 1
Yes
LBL1
No
#H'7FFF → R2
#H'FFFF → R3
1
#H'8000 → R2
#H'0000 → R3
1
LBL3
1 → Bit 7 of R0L
R0H + #1 → R0H
Shift R1L 1 bit to the left
Rotate R1H 1 bit to the left
Rotate R0L 1 bit to the left
Rotate R3L, R3H, R2L,and
R2H 1 bit to the left 1 bit to the left
R0H - #1 → R0H
R0H = 0
3
Branch if R0H is less than H'1F (exponent < H'9E).
When R0H is H'1F or greater, store the sign bit in C bit and set the maximum value of the same sign in R2 and R3.
Set the implicit MSB.
Increment R0H by 1.
Shift the mantissa (24 bits), in which the implicit MSB has been set, 1 bit to the left.
Rotate R3 and R2 1 bit to the left.
Decrement R0H and execute the loop until R0H is 0.


3
Bit 0 of R5L → C
...... Branch if the sign bit is 0 (positive number).
Yes
1
C = 0
No
Logically convert
R2H, R2L, R3H, and R3L
...... Take the two's complement of the 32-bit binary number
set in R2 and R3 to generate a negative value.
R3L + #1 → R3L
R3H + #H'00 + C → R3H
R2L + #H'00 + C → R2L
R2H + #H'00 + C → R2H
1
LBL5
RTS

8. Program Listing

```
                ;************************************************************
                ;*                                                          *
                ;*          NAME:  CHANGE FLOATING POINT TO 32 BIT BINARY   *
                ;*                 (FKTR)                                   *
                ;*                                                          *
                ;************************************************************
                ;*                                                          *
                ;*        ENTRY:   R0      SINGLE PREC. NO. (UPPER 2 BYTES) *
                ;*                 R1                       (LOWER 2 BYTES) *
                ;*                                                          *
                ;*       RETURNS: R2     SIGNED 32-BIT NO. (UPPER 2 BYTES)  *
                ;*                 R3                        (LOWER 2 BYTES) *
                ;*                                                          *
                ;************************************************************
                ;
                        .CPU        300HN
0000                    .SECTION    FKTR_code,CODE,ALIGN=2
                        .EXPORT     FKTR
                ;
     00000000   FKTR    .EQU        $           ;Entry point
0000 79020000           MOV.W       #H'0000,R2  ;Clear R2
0004 0D23               MOV.W       R2,R3       ;Clear R3
                ;
0006 0D00               MOV.W       R0,R0
0008 4604               BNE         LBL1
000A 0D11               MOV.W       R1,R1
000C 4754               BEQ         LBL5        ;Branch if R0=R1=0
000E            LBL1
000E 7770               BLD         #7,R0H
0010 670D               BST         #0,R5L      ;Set sign bit to bit0 of R5L
0012 7778               BLD         #7,R0L
0014 1200               ROTXL.B     R0H         ;Set exponent
0016 F57F               MOV.B       #H'7F,R5H
0018 1850               SUB.B       R5H,R0H
001A 4546               BCS         LBL5        ;Branch if R0<H'7F
001C A01F               CMP.B       #H'1F,R0H
001E 4518               BCS         LBL3        ;Branch if R0<H'1F
0020 770D               BLD         #0,R5L
0022 450A               BCS         LBL2        ;Branch if sign bit = 1
0024 79027FFF           MOV.W       #H'7FFF,R2
0028 7903FFFF           MOV.W       #H'FFFF,R3  ;Set H'7FFFFFFF
002C 4034               BRA         LBL5        ;Always branch
002E            LBL2
002E 79028000           MOV.W       #H'8000,R2
0032 79030000           MOV.W       #H'0000,R3  ;Set H'80000000
0036 402A               BRA         LBL5
                ;
0038            LBL3
0038 7078               BSET        #7,R0L      ;Set implicit MSB
003A 8001               ADD.B       #1,R0H      ;R0H + #1 -> R0H
003C            LBL4
003C 1009               SHLL.B      R1L         ;Shift mantissa 1 bit left
```

```
 53   003E 1201               53           ROTXL.B     R1H
 54   0040 1208               54           ROTXL.B     R0L
 55                           55     ;
 56   0042 120B               56           ROTXL.B     R3L         ;Rotate 32 bit binary 1 bit left
 57   0044 1203               57           ROTXL.B     R3H
 58   0046 120A               58           ROTXL.B     R2L
 59   0048 1202               59           ROTXL.B     R2H
 60   004A 1A00               60           DEC.B       R0H         ;Decrement R0H
 61   004C 46EE               61           BNE         LBL4        ;Branch if Z=0
 62                           62     ;
 63   004E 770D               63           BLD         #0,R5L      ;Bit load sign bit to C flag
 64   0050 4410               64           BCC         LBL5        ;Branch if C=0
 65   0052 1702               65           NOT         R2H         ;2's complement 32 bit binary
 66   0054 170A               66           NOT         R2L
 67   0056 1703               67           NOT         R3H
 68   0058 170B               68           NOT         R3L
 69   005A 8B01               69           ADD.B       #H'01,R3L
 70   005C 9300               70           ADDX.B      #H'00,R3H
 71   005E 9A00               71           ADDX.B      #H'00,R2L
 72   0060 9200               72           ADDX.B      #H'00,R2H
 73                           73     ;
 74   0062                    74     LBL5
 75   0062 5470               75           RTS
 76                           76     ;
 77                           77           .END
*****TOTAL ERRORS       0
*****TOTAL WARNINGS     0
```

<Reference> Description of Single-Precision Floating-Point Formats

Single-Precision Floating-Point Formats:

1. Internal Representation of Single-Precision Floating Point Numbers
   One of the following formats is used depending on the value of the single-precision floating-point data in this application note (a real number is indicated as R).
   1) Internal Representation When R=0

31	30	29	......	2	1	0
0	0	0	......	0	0	0

   All the 32 bits are 0.
   2) Normalized Format

31	30 … 23	22 … 0
S	α	β

   α is an index number with an 8-bit-long field. β is a mantissa with a 23-bit-long field. Here, the R value can be represented by the expression below (when $1 \le \alpha \le 254$).

   ↓Implicit MSB

   $R = 2^S \times 2^{\alpha-126} \times (1 + 2^{-1} \times \beta_{22} + 2^{-2} \times 21 + \ldots\ldots + 2^{-23} \times \beta_0)$

   where, βi is the value of the i-th bit of β ($0 \le i \le 22$), and S is the sign bit.
   3) Denormalized Format

31	30 … 23	22 … 0
S	0 0 0 0 0 0 0 0	β

   β is a mantissa with a 23-bit-long field. This format is used to represent a real number that is too small to be represented by the normalized format.
   Here, the R value can be represented by the expression below.

   $R = 2^S \times 2^{-126} \times (2^{-1} \times \beta_{22} + 2^{-2} \times 21 + \ldots\ldots + 2^{-23} \times \beta_0)$
   4) Infinity

31	30 … 23	22 … 0
S	1 1 1 1 1 1 1 1	β

   β is a mantissa with a 23-bit-long field. Note that if all the bits in the index part are 1, the R value is handled as follows, in this application note.

   When S = 0: Plus infinity
   $R = +\infty$
   When S = 1: Minus infinity
   $R = -\infty$

2. Internal Representation Examples
   S = B'0 (binary)
   α = B'10000011 (binary)
   β = B'1011100……0 (binary)
   Under the above conditions, the corresponding R value is represented as follows.

   $R = 2^0 \times 2^{131-126} \times (1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5})$
   $= 16 + 8 + 2 + 1 + 0.5 = 27.5$
   1) Maximum and Minimum Values
   Here, the maximum and minimum values are absolute values. The maximum value is indicated as R_{MAX} and the minimum value is indicated as R_{MIN}. Up to the following values can be represented.

   $R_{MAX} = 2^{254-127} \times (1 + 2^{-1} + 2^{-2} + 2^{-3} + \ldots\ldots + 2^{-23})$
   $\approx 3.27 \times 10^{38}$
   $R_{MIN} = 2^{-126} \times 2^{-23} = 2^{-140} \approx 1.40 \times 10^{-45}$

Revision Record

Rev.	Date	Description	
		Page	Summary
2.00	Feb.28.06	—	Format has been changed from Hitachi version to Renesas version.

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.