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

H8/300H Tiny Series

Square Root of a 32-Bit Binary Number (SQRT)

Introduction

Produces the square root of a 32-bit binary number as a 16-bit binary number.

Target Device

H8/300H Tiny Series

Contents

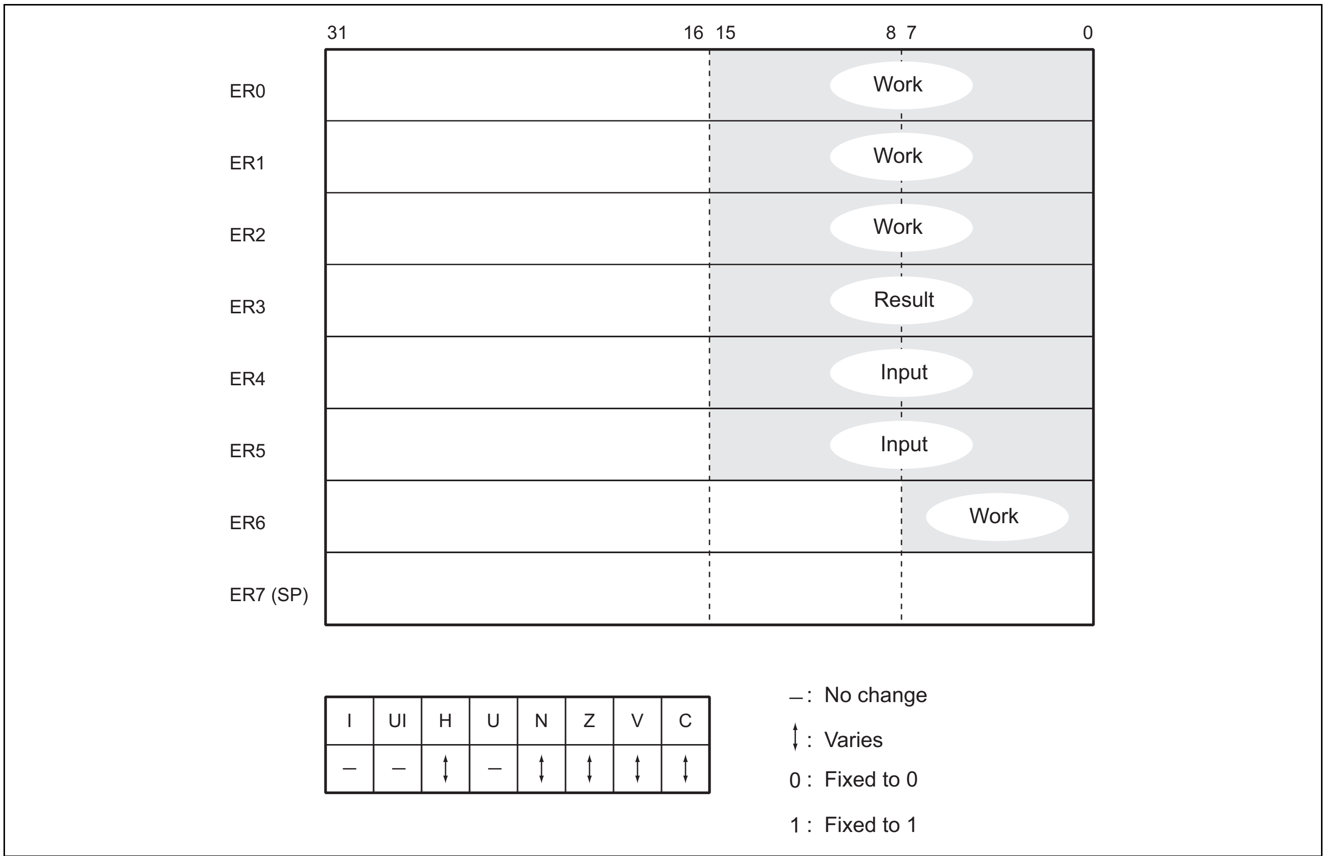
1. Function

1. Produces the square root of a 32-bit binary number as a 16-bit binary number.
2. Finds and outputs the square root of a 32-bit binary number as a 16-bit binary number.
3. The arguments are all unsigned integers. All data operations are on the general registers.

2. Arguments

	Contents	Storage Location	Data Length (Bytes)
Input	Original number (binary number for square-root extraction)	R4, R5	4
Output	Square root	R3	2

3. Changes to Internal Registers and Flags



4. Programming Specifications

Program memory (bytes)
94
Data memory (bytes)
0
Stack (bytes)
0
Number of cycles
1340
Re-entrant
Yes
Relocatalbe
Yes
Interrupts during execution
Yes

5. Note

The number of cycles in the programming specifications is the value in the execution of SQRT as shown in figure 1.

6. Description

6.1 Description of Functions

1. The arguments are as follows.
   R4: Set the higher-order word of the original number in 32-bit binary here, as an input argument.
   R5: Set the lower-order word of the original number here.
   R3: The square root (16-bit binary) is set here, as an output argument.
2. The following figure illustrates the execution of the SQRT subroutine. When the input arguments are set as shown below, the corresponding square root is set in R3.

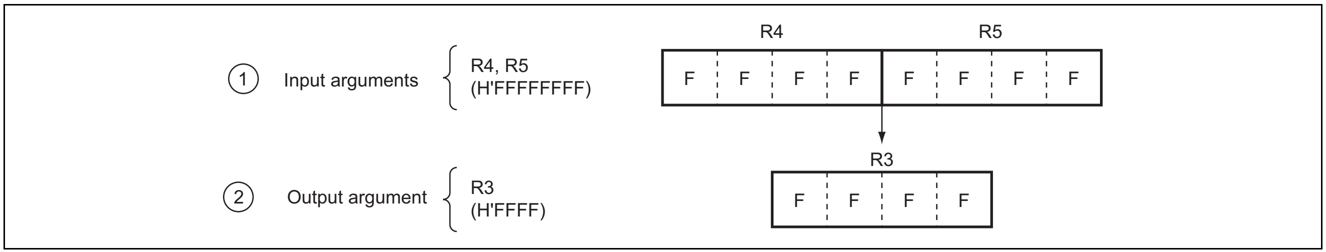


Figure 1 Example of SQRT Execution

6.2 Usage Notes

1. Any higher-order bits of the input argument that are unused must be explicitly set to "0", as shown in figure 2. Otherwise, the correct result might not be obtained because undefined data in the higher-order bits is included in computation of the square root.

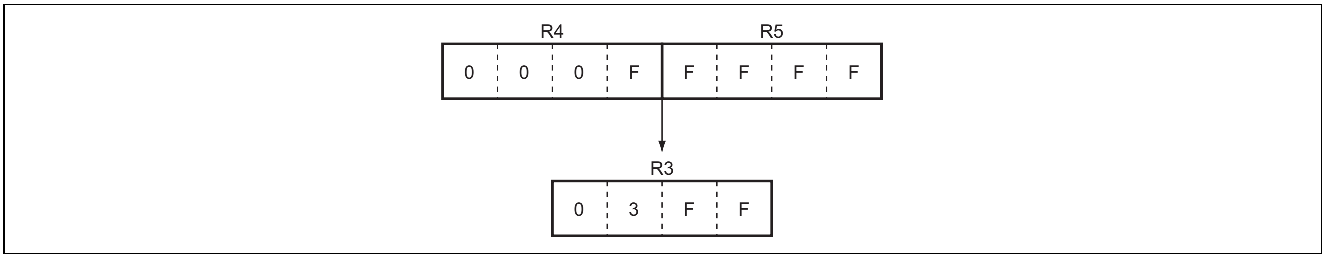


Figure 2 Example when Higher-order Bits are not Used

2. The fractional part of the result is discarded.

6.3 Description of Data Memory

No data memory is used by SQRT.

6.4 Example of Usage

After setting the number for which the square root is to be extracted, call the SQRT subroutine.

```
WORK1   .RES.W 2              ......... Reservation of the data memory area for setting by the user program of the 32-bit binary
                                        number for square-root extraction.
WORK2   .RES.W 2              ......... Reservation of a data memory area to hold the 16-bit binary square root for the user program.
          .
          .
          .
        MOV.W @WORK1, R4      ......... Sets, as the input argument, the 32-bit binary number for square-root extraction that is specified
                                        by the user program.
        MOV.W @WORK1+2, R5
      | JSR    @SQRT |        ......... Subroutine call of SQRT.
        MOV.W R3, @WORK 2     ......... Transfers , to the data memory of the user program, the 16-bit binary square root the output by
          .                             the subroutine.
          .
          .
```

6.5 Principles of Operation

1. The following figure shows the method used to calculate the square root (H'05) of the hexadecimal number H'22.

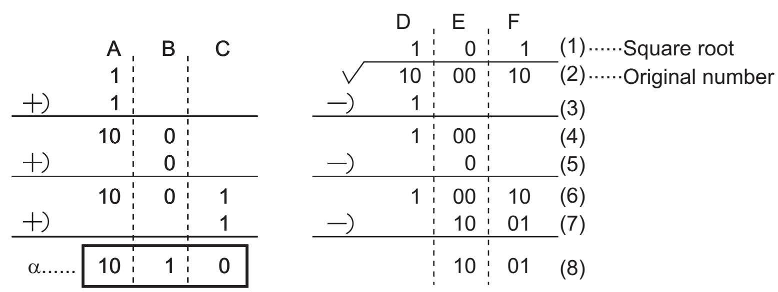
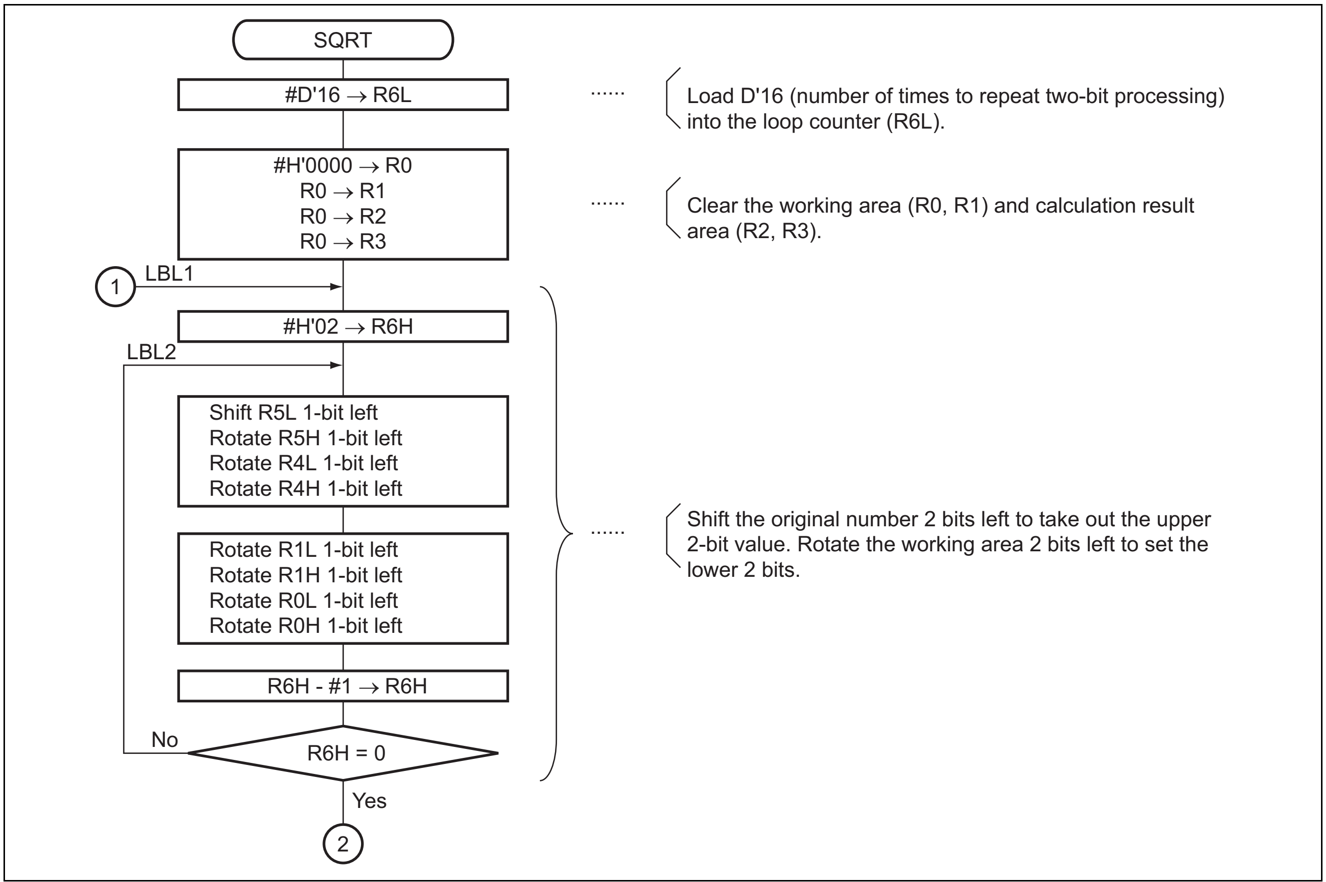


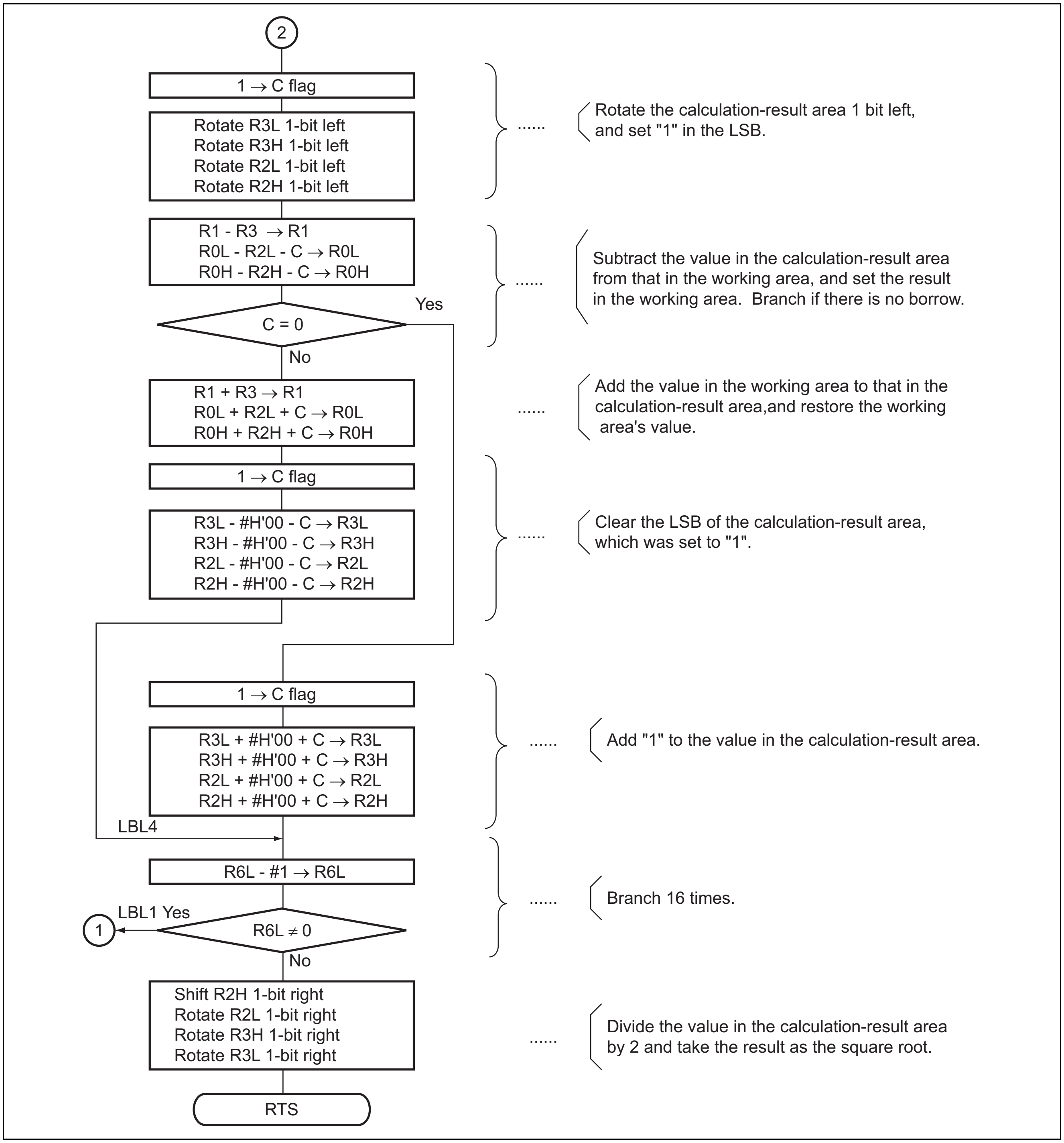
Figure 3 Calculation of Square Root

1) As shown in the figure, the square root can be found by processing every two-bit unit, from highest to lowest order, of the original number.
2) The square root (1) is equivalent to the quotient when α is divided by two. Parameter α is found through the operations A, B, and C in the figure. The SQRT subroutine finds the square root by calculating the value of α and then dividing it by two.

2. Details on the program are given below.
   The program:
   a) sets D'16 in R6L, which is the number of two-bit units in a 32-bit binary number;
   b) clears the area for storage of the square root (R2, R3) and the working area (R0, R1);
   c) extracts the two highest-order bits of the input binary number to R0 and R1, by rotating R4, R5, R0, and R1 two bits to the left;
   d) places "1" in R2, R3;
   e) subtracts R2, R3 from R0, R1 to find the difference (D, (2), (3), and (4)), then sets difference thus found in R0, R1; and
   f) if the result is positive, increments R2, R3 (A to (4)); if the result is negative, decrements R2, R3, and adds R2, R3 to R0, R1 (D, E, and (6)).
3. In SQRT, R6 is decremented each time steps c) through f) above are performed, and this processing is repeated until R6 has reached "0".

7. Flowchart


SQRT
#D'16 → R6L
Load D'16 (number of times to repeat two-bit processing) into the loop counter (R6L).
#H'0000 → R0
R0 → R1
R0 → R2
R0 → R3
Clear the working area (R0, R1) and calculation result area (R2, R3).
1
LBL1
#H'02 → R6H
LBL2
Shift R5L 1-bit left
Rotate R5H 1-bit left
Rotate R4L 1-bit left
Rotate R4H 1-bit left
Rotate R1L 1-bit left
Rotate R1H 1-bit left
Rotate R0L 1-bit left
Rotate R0H 1-bit left
Shift the original number 2 bits left to take out the upper 2-bit value. Rotate the working area 2 bits left to set the lower 2 bits.
R6H - #1 → R6H
No
R6H = 0
Yes
2


2
1 → C flag
Rotate R3L 1-bit left
Rotate R3H 1-bit left
Rotate R2L 1-bit left
Rotate R2H 1-bit left
...... Rotate the calculation-result area 1 bit left, and set "1" in the LSB.
R1 - R3 → R1
R0L - R2L - C → R0L
R0H - R2H - C → R0H
C = 0
Yes
No
...... Subtract the value in the calculation-result area from that in the working area, and set the result in the working area. Branch if there is no borrow.
R1 + R3 → R1
R0L + R2L + C → R0L
R0H + R2H + C → R0H
...... Add the value in the working area to that in the calculation-result area,and restore the working area's value.
1 → C flag
R3L - #H'00 - C → R3L
R3H - #H'00 - C → R3H
R2L - #H'00 - C → R2L
R2H - #H'00 - C → R2H
...... Clear the LSB of the calculation-result area, which was set to "1".
1 → C flag
R3L + #H'00 + C → R3L
R3H + #H'00 + C → R3H
R2L + #H'00 + C → R2L
R2H + #H'00 + C → R2H
...... Add "1" to the value in the calculation-result area.
LBL4
R6L - #1 → R6L
LBL1 Yes
1
R6L ≠ 0
No
...... Branch 16 times.
Shift R2H 1-bit right
Rotate R2L 1-bit right
Rotate R3H 1-bit right
Rotate R3L 1-bit right
...... Divide the value in the calculation-result area by 2 and take the result as the square root.
RTS

8. Program Listing

```
                      ;*********************************************************************
                      ;*                                                                   *
                      ;*          NAME : 32 BIT SQUARE ROOT (SQRT)                         *
                      ;*                                                                   *
                      ;*********************************************************************
                      ;*                                                                   *
                      ;*        ENTRY  : R4,R5  (32 BIT BINARY)                            *
                      ;*                                                                   *
                      ;*       RETURNS : R3     (SQUARE ROOT)                              *
                      ;*                                                                   *
                      ;*********************************************************************
                      ;
                              .CPU       300HN
0000                          .SECTION   SQRT_code,CODE,ALIGN=2
                              .EXPORT    SQRT
                      ;
      00000000        SQRT    .EQU       $              ;Entry point
0000  FE10                    MOV.B      #D'16,R6L      ;Set shift counter
0002  79000000                MOV.W      #H'0000,R0     ;Clear R0
0006  0D01                    MOV.W      R0,R1          ;Clear R1
0008  0D02                    MOV.W      R0,R2          ;Clear R2
000A  0D03                    MOV.W      R0,R3          ;Clear R3
000C                  LBL1
000C  F602                    MOV.B      #H'02,R6H
000E                  LBL2
000E  100D                    SHLL.B     R5L            ;Shift 32 bit binary 1 bit left
0010  1205                    ROTXL.B    R5H
0012  120C                    ROTXL.B    R4L
0014  1204                    ROTXL.B    R4H
0016  1209                    ROTXL.B    R1L
0018  1201                    ROTXL.B    R1H
001A  1208                    ROTXL.B    R0L
001C  1200                    ROTXL.B    R0H
001E  1A06                    DEC.B      R6H            ;Decrement R6H
0020  46EC                    BNE        LBL2           ;Branch if Z=0
0022  0401                    ORC.B      #H'01,CCR      ;Set C flag of CCR
0024  120B                    ROTXL.B    R3L            ;Rotate square root
0026  1203                    ROTXL.B    R3H
0028  120A                    ROTXL.B    R2L
002A  1202                    ROTXL.B    R2H
002C  1931                    SUB.W      R3,R1          ;R1  - R3       -> R1
002E  1EA8                    SUBX.B     R2L,R0L        ;R0L - R2L - C -> R0L
0030  1E20                    SUBX.B     R2H,R0H        ;R0H - R2H - C -> R0H
0032  4412                    BCC        LBL3           ;Branch if C=0
0034  0931                    ADD.W      R3,R1          ;R1  + R3       -> R1
0036  0EA8                    ADDX.B     R2L,R0L        ;R0L + R2L + C -> R0L
0038  0E20                    ADDX.B     R2H,R0H        ;R0H + R2H + C -> R0H
003A  0401                    ORC.B      #H'01,CCR      ;Bit set C flag of CCR
003C  BB00                    SUBX.B     #H'00,R3L      ;R3L - #H'00 - C -> R3L
003E  B300                    SUBX.B     #H'00,R3H      ;R3H - #H'00 - C -> R3H
0040  BA00                    SUBX.B     #H'00,R2L      ;R2L - #H'00 - C -> R2L
0042  B200                    SUBX.B     #H'00,R2H      ;R2H - #H'00 - C -> R2H
```

```
53     0044  400A        53          BRA         LBL4           ;
54     0046              54   LBL3
55     0046  0401        55          ORC.B       #H'01,CCR      ;Bit set C flag of CCR
56     0048  9B00        56          ADDX.B      #H'00,R3L      ;R3L + #H'00 + C -> R3L
57     004A  9300        57          ADDX.B      #H'00,R3H      ;R3H + #H'00 + C -> R3H
58     004C  9A00        58          ADDX.B      #H'00,R2L      ;R2L + #H'00 + C -> R2L
59     004E  9200        59          ADDX.B      #H'00,R2H      ;R2H + #H'00 + C -> R2H
60     0050              60   LBL4
61     0050   1A0E       61          DEC.B       R6L            ;Decrement shift counter
62     0052  46B8        62          BNE         LBL1           ;Branch if
63     0054  1102        63          SHLR.B      R2H
64     0056  130A        64          ROTXR.B     R2L
65     0058  1303        65          ROTXR.B     R3H            ;Rotate square root
66     005A  130B        66          ROTXR.B     R3L
67     005C  5470        67          RTS
68                       68   ;
69                       69          .END
*****  TOTAL ERRORS   0
*****  TOTAL WARNINGS 0
```

Revision Record

Rev.	Date	Description	
		Page	Summary
2.00	Jun.12.06	—	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.