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.

H8/300L Super Low Power Series

Multiplication of 4-Digit BCD Numbers (MULD)

Introduction

The software MULD multiplies a 4-digit binary-coded decimal (BCD) number by another 4- digit BCD number and places the result (an 8-digit BCD number) in general-purpose registers.

Target Device

H8/38024

Contents

1. Arguments

Description		Memory area	Data length (bytes)
Input	Multiplicand	R1	2
	Multiplier	R0	2
Output	Result of multiplication	R2, R3	4

2. Changes to Internal Registers and Flags

R0	R1	R2	R3	R4	R5		R6		R7
×	—	○	○	—	—	×	×	×	—

I	U	H	U	N	Z	V	C
—	—	×	—	×	×	×	×

Legend
—: No change
×: Undefined
○: Result

3. Specifications

Program memory (bytes)
62
Data memory (bytes)
0
Stack (bytes)
0
Clock cycle count
1192
Reentrant
Possible
Relocation
Possible
Interrupt
Possible

4. Notes

The clock cycle count (1192) in the specifications is for multiplication of 9999 by 9999.

5. Description

5.1 Details of functions

1. The following arguments are used with the software MULD:
   R0: Sets a 4-digit BCD multiplier (16 bits long) as an input argument.
   R1: Sets a 4-digit BCD multiplicand (16 bits long) as an input argument.
   R2: The upper 4 digits of the result (an 8-digit BCD, 32 bits long) are set here as an input argument.
   R3: The lower 4 digits of the result (an 8-digit BCD, 32 bits long) are set here as an input argument.
2. The following figure illustrates the execution of the software MULD. When the input arguments are set as shown in (1), the result of multiplication is placed in R2 and R3 as shown in (2).

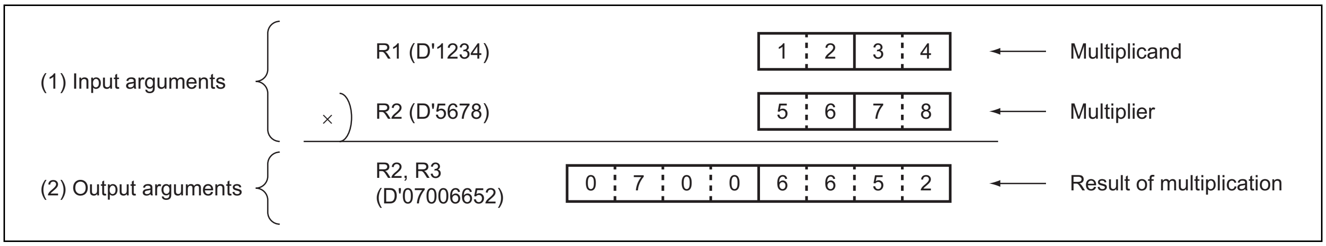


Figure 1 Example of Software MULD Execution

3. Table 1 lists the result of multiplication with 0 placed in input arguments.

Table 1 Result of Multiplication with 0 Placed in Input Arguments

Input arguments		Output arguments
Multiplicand (R1)	**Multiplier (R0)**	**Product (R2, R3)**
H'****	H'0000	H'0000
H'0000	H'****	H'0000
H'0000	H'0000	H'0000

Note: H'**** is a hexadecimal number.

5.2 Notes on usage

1. When the upper bits are not used (see figure 2), set them to 0; otherwise, a correct result cannot be obtained because multiplication is done on the numbers including indeterminate data placed in the upper bytes.

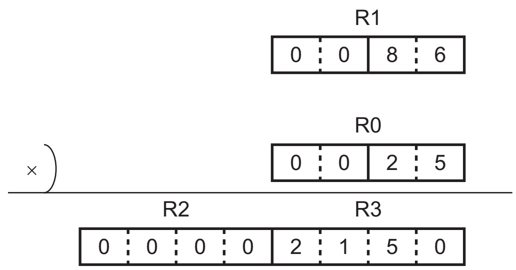


Figure 2 Example of Multiplication with Upper Bits Unused

2. After execution of the software MULD, the multiplier will be lost. When the multiplier is still needed after software MULD execution, save it in memory.

5.3 Data memory

The software MULD uses no data memory.

5.4 Example of usage

Set a multiplicand and a multiplier in the registers and call the software MULD as a subroutine.

```
WORK1     . RES. W     1          ---------- Reserve a data memory area in which the user program places
                                             a 4-digit BCD multiplicand.

WORK2     . RES. W     1          ---------- Reserve a data memory area in which the user program places
                                             a 4-digit BCD multiplier.

WORK3     . RES. W     2          ---------- Reserve a data memory area in which the user program places
               .                             the result of multiplication (an 8-digit BCD number).
               .
               .
          MOV. W       @WORK1, R1     ------- Place the 4-digit BCD multiplicand set by the user program
                                              in the input argument (R1).

          MOV. W       @WORK2, R0     ------- Place the 4-digit BCD multiplier set by the user program
                                              in the input argument (R0).

          JSR          @MULD          ------- Call the software MULD as a subroutine.

          MOV. W R2,   @WORK3         ------- Place the result set in the output argument
          MOV. W R3,   @WORK3+2               in the data memory of the user program.
               .
               .
               .
```

5.5 Operation

1. Multiplication of decimal numbers can be done by performing a series of additions and shifts. Figure 3 shows an example of the multiplication (5678 × 1234).

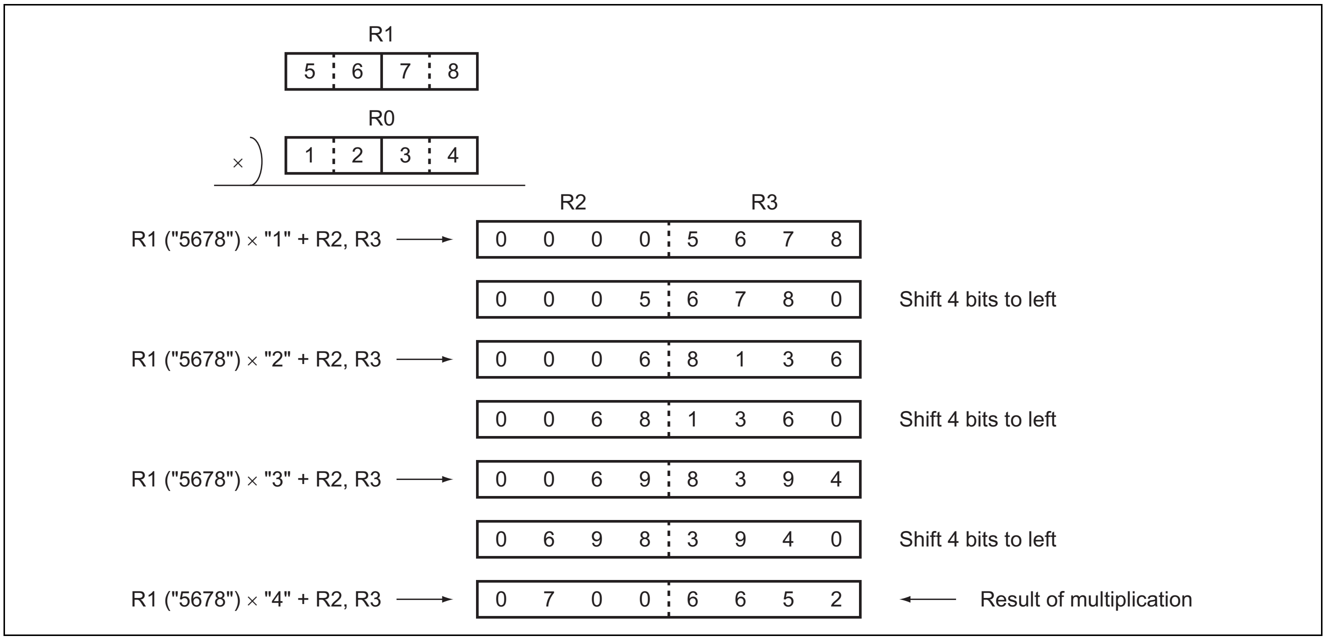
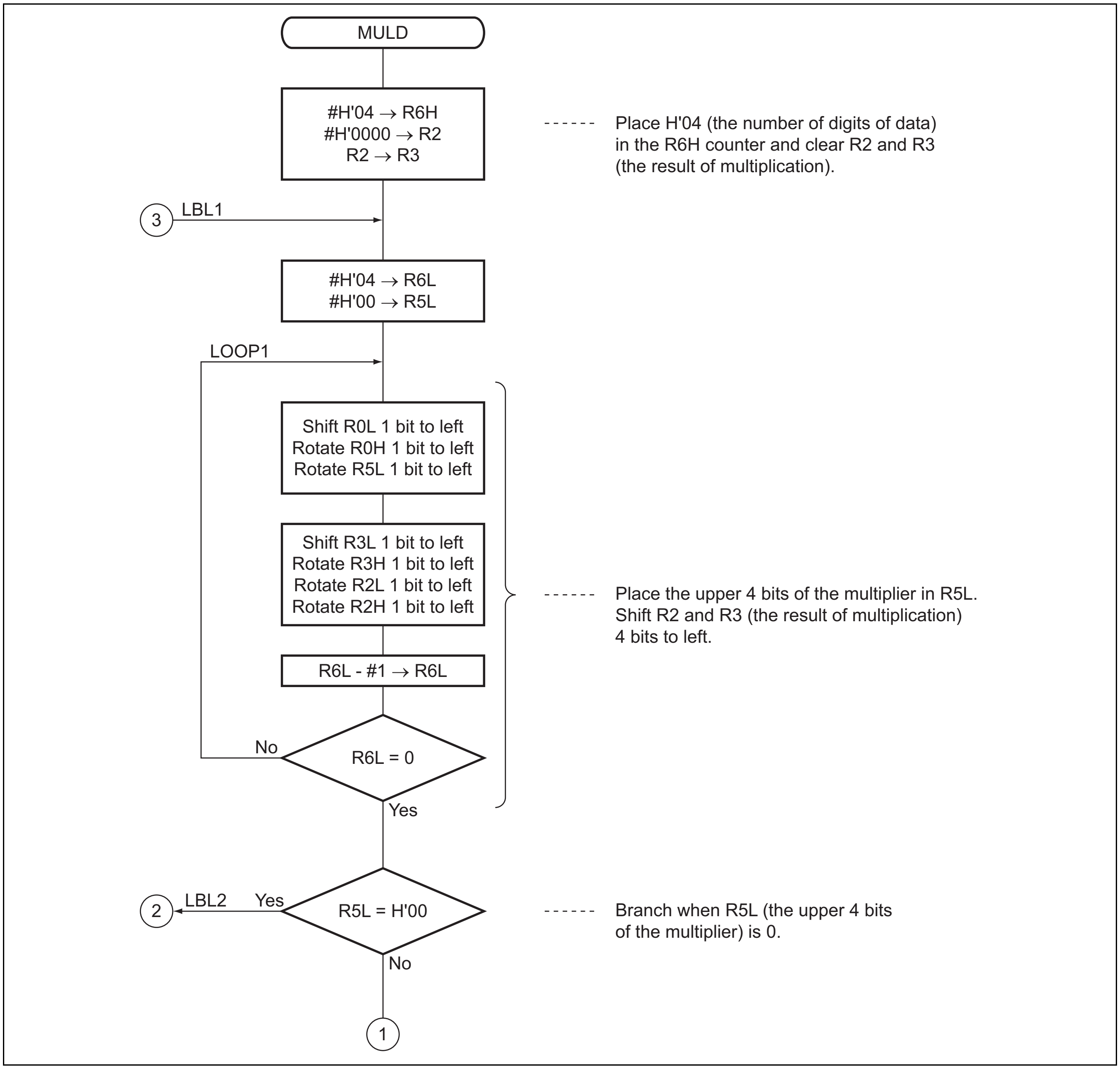


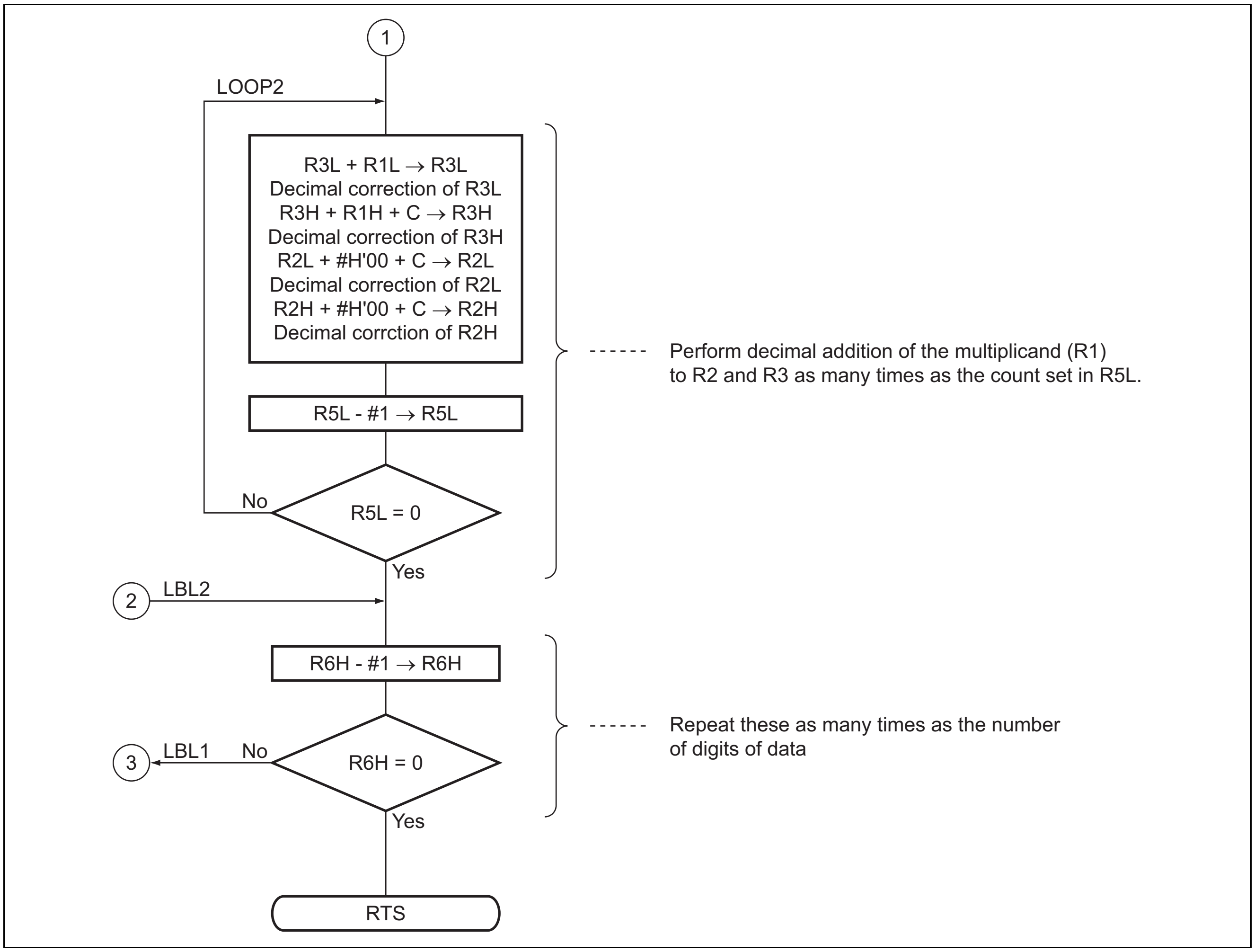
Figure 3 Example of Multiplication (5678 × 1234)

Figure 3 illustrates the method used to find the product by repeatedly adding the multiplicand, multiplied by the respective digits of the multiplier from the leftmost digit, to the result and then shifting the result.
Firstly, the highest-order BCD digit (four bits) of the multiplier is taken, and the multiplicand is added this number of times. The result is then shifted four bits to left (one BCD digit). Next, the second highest digit is taken from the multiplier, the multiplicand is added to the preceding result this number of times, and so on. The final result is found by repeating the processes as many times as there are BCD digits (four times).

2. Details of the program are given below.
   The program:
   a. places H'04 in R6H as a counter that indicates the number of BCD digits in the data;
   b. clears R2 and R3, where the result of multiplication is to be stored;
   c. shifts R2 and R3 four bits (one BCD digit) to left;
   d. loads one BCD digit from the higher-order end of the multiplier to R5L, and branches to step g. when R5L is "0";
   e. BCD-adds the multiplicand to R2, R3 the same number of times as the value in R5L;
   f. decrements R6H; and
   g. repeats steps c to f above until R6H has become "0".

6. Flowchart


MULD
#H'04 → R6H
#H'0000 → R2
R2 → R3
Place H'04 (the number of digits of data) in the R6H counter and clear R2 and R3 (the result of multiplication).
3
LBL1
#H'04 → R6L
#H'00 → R5L
LOOP1
Shift R0L 1 bit to left
Rotate R0H 1 bit to left
Rotate R5L 1 bit to left
Shift R3L 1 bit to left
Rotate R3H 1 bit to left
Rotate R2L 1 bit to left
Rotate R2H 1 bit to left
Place the upper 4 bits of the multiplier in R5L. Shift R2 and R3 (the result of multiplication) 4 bits to left.
R6L - #1 → R6L
No
R6L = 0
Yes
2
LBL2
Yes
R5L = H'00
Branch when R5L (the upper 4 bits of the multiplier) is 0.
No
1


1
LOOP2
R3L + R1L → R3L
Decimal correction of R3L
R3H + R1H + C → R3H
Decimal correction of R3H
R2L + #H'00 + C → R2L
Decimal correction of R2L
R2H + #H'00 + C → R2H
Decimal corrction of R2H
Perform decimal addition of the multiplicand (R1) to R2 and R3 as many times as the count set in R5L.
R5L - #1 → R5L
No
R5L = 0
Yes
2
LBL2
R6H - #1 → R6H
Repeat these as many times as the number of digits of data
3
LBL1
No
R6H = 0
Yes
RTS

7. Program List

```
*** H8/300 ASSEMBLER VER 1.0B ** 08/18/92 10:01:29
PROGRAM NAME =
                                  ;*************************************************************
                                  ;*
                                  ;*    00 - NAME  :DECIMAL MULTIPLICATION
                                  ;*                (MULD)
                                  ;*
                                  ;*************************************************************
                                  ;*
                                  ;*    ENTRY      :R1 (MULTIPLICAND)
                                  ;*                R0 (MULTIPLIER)
                                  ;*
                                  ;*    RETURNS    :R2 (UPPER WORD OF RESULT)
                                  ;*                R3 (LOWER WORD OF RESULT)
                                  ;*
                                  ;*************************************************************
                                  ;
MULD_cod C    0000                     .SECTION            MULD_code,CODE,ALIGN=2
                                       .EXPORT  MULD
                                  ;
MULD_cod C          00000000      MULD .EQU $              ;Entry point
MULD_cod C    0000  F604               MOV.B    #H'04,R6H   ;Set bit counter1
MULD_cod C    0002  79020000           MOV.W    #H'0000,R2  ;Clear R2
MULD_cod C    0006  0D23               MOV.W    R2,R3       ;Clear R3
MULD_cod C    0008                LBL1
MULD_cod C    0008  FE04               MOV.B    #H'04,R6L   ;Set bit counter2
MULD_cod C    000A  FD00               MOV.B    #H'00,R5L   ;Clear R5L
MULD_cod C    000C                LOOP1
MULD_cod C    000C  1008               SHLL.B   R0L         ;Shift multiplier 1 bit left
MULD_cod C    000E  1200               ROTXL.B  R0H
MULD_cod C    0010  120D               ROTXL.B  R5L
MULD_cod C    0012  100B               SHLL.B   R3L         ;Shift result 1 bit left
MULD_cod C    0014  1203               ROTXL.B  R3H
MULD_cod C    0016  120A               ROTXL.B  R2L
MULD_cod C    0018  1202               ROTXL.B  R2H
MULD_cod C    001A  1A0E               DEC.B    R6L         ;Decrement bit counter 2
MULD_cod C    001C  46EE               BNE      LOOP1       ;Branch if Z=0
MULD_cod C    001E  AD00               CMP.B    #H'00,R5L
MULD_cod C    0020  4714               BEQ      LBL2        ;Branch if Z=1
MULD_cod C    0022                LOOP2
MULD_cod C    0022  089B               ADD.B    R1L,R3L     ;R1L + R3L -> R1L
MULD_cod C    0024  0F0B               DAA.B    R3L         ;Decimal adjust R3L
MULD_cod C    0026  0E13               ADDX.B   R1H,R3H     ;R1H + R3H + C -> R1H
MULD_cod C    0028  0F03               DAA.B    R3H         ;Decimal adjust R3H
MULD_cod C    002A  9A00               ADDX.B   #H'00,R2L   ;R2L + #H'00 + C -> R2L
MULD_cod C    002C  0F0A               DAA.B    R2L         ;Decimal adjust R2L
MULD_cod C    002E  9200               ADDX.B   #H'00,R2H   ;R2H + #H'00 + C -> R2H
MULD_cod C    0030  0F02               DAA.B    R2H         ;Decimal adjust R2H
MULD_cod C    0032  1A0D               DEC.B    R5L         ;Clear bit 0 of R5L
MULD_cod C    0034  46EC               BNE      LOOP2       ;Branch if Z=0
```

```
MULD_cod C    0036              LBL2
MULD_cod C    0036  1A06              DEC.B     R6H           ;Decrement bit counter1
MULD_cod C    0038  46CE              BNE       LBL1          ;Branch if Z=0
                                ;
MULD_cod C    003A  5470              RTS
                                      .END
*****TOTAL ERRORS 0
*****TOTAL WARNINGS 0
```

Website and Support

Renesas Technology Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sep.18.03	—	First edition issued
2.00	Nov.30.06	All pages	Content correction