SH7000 Series

Remainder of 32 Bit ÷ 32 Bit (Signed)

Label: DIVS32R

Functions Used: DIV0S Instruction
DIV1 Instruction

Contents

1. Function

Divides the dividend (signed 32 bits) by the divisor (signed 32 bits), and determines the remainder (signed 32 bits). Also indicates errors (division by 0) in the T bit.

2. Arguments

Description		Storage Location	Data Length (Bytes)
Input	Dividend (signed 32 bits)	R1	4
	Divisor (signed 32 bits)	R0	4
Output	Remainder (signed 32 bits)	R2	4
	Error (division by 0) generated/not generated (generated: T = 1, not generated: T = 0)	T bit (SR)	4

3. Internal Register Changes and Flag Changes

Register	(Before Execution)	→	(After Execution)
R0	Divisor (signed 32 bits)	→	No change
R1	Dividend (signed 32 bits)	→	Change
R2	Undefined	→	Remainder (signed 32 bits)
R3	Work		
R4	Work		
R5			
R6			
R7			
R8			
R9			
R10			
R11			
R12			
R13			
R14			
R15	(SP)		

T bit: *

— : No change
* : Change
0 : Fixed 0
1 : Fixed 1

4. Programming Specifications

Program memory (bytes)
182
Data memory (bytes)
0
Stack (bytes)
8
Number of states
87
Reentrant
Yes
Relocation
Yes
Intermediate interrupt
Yes

5. Notes

The number of states indicated in the programming specifications is the value when H'80000000 ÷ H'7FFFFFFF is calculated.

6. Description

(1) Function

Details of the arguments are as follows.

R0: Set the divisor (signed 32 bits) as the input argument.

R1: Set the dividend (signed 32 bits) as the input argument.

R2: Holds the remainder (signed 32 bits) as the output argument.

T bit (SR): Indicates whether an error (division by 0) has occurred.
T bit = 1: Indicates an error (division by 0) has occurred.
T bit = 0: Indicates no error (division by 0) has occurred.

Figure 1 shows a software DIVS32R execution example.

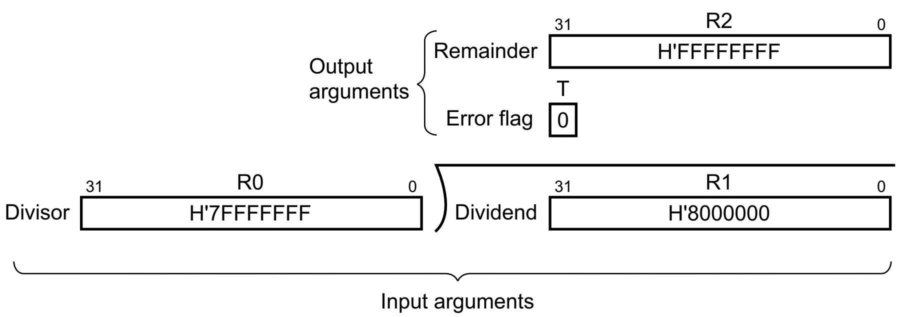


Figure 1 Software DIVS32R Execution Example

(2) Usage Notes

The value of R1, which is set to the dividend, is changed when software instruction DIVS32R is executed. If the value for the dividend will be needed after the software DIVS32R instruction is executed, it should be saved beforehand.

(3) RAM Used

No RAM is used by the software DIVS32R instruction.

(4) Usage Example

After the dividend and divisor are set in the input arguments, the software instruction DIVS32R is executed by a subroutine call.

```
          MOV.L   DATA1,R1      . . . Sets dividend (unsigned 32 bits) in input argument (R1)
          BSR     DIVS32R       . . . Subroutine call to software instruction DIVS32R
          MOV.L   DATA2,R0      . . . Sets divisor (unsigned 32 bits) in input argument (R0)
          BT      ERROR         . . . Branches to error processing subroutine if error (division by 0) occurs
                .
                .
                .
                .
         .align    4
DATA1    .data.l   H'80000000
DATA2    .data.l   H'7FFFFFFF
```

(5) Operating Principle

(a) Before division, the following initial settings are carried out.

(i) R2 is used for the upper 32 bits to sign extend the dividend to 64 bits. (Figure 2-(1))

(ii) If the dividend is negative, it is converted to the complement of 1 for handling by the one-step division instruction. (Figure 2-(2))

(iii)The M, Q, and T bits used in one-step division are set to signed division values (M = divisor sign, Q = dividend sign, T = quotient sign). (Figure 2-(3))

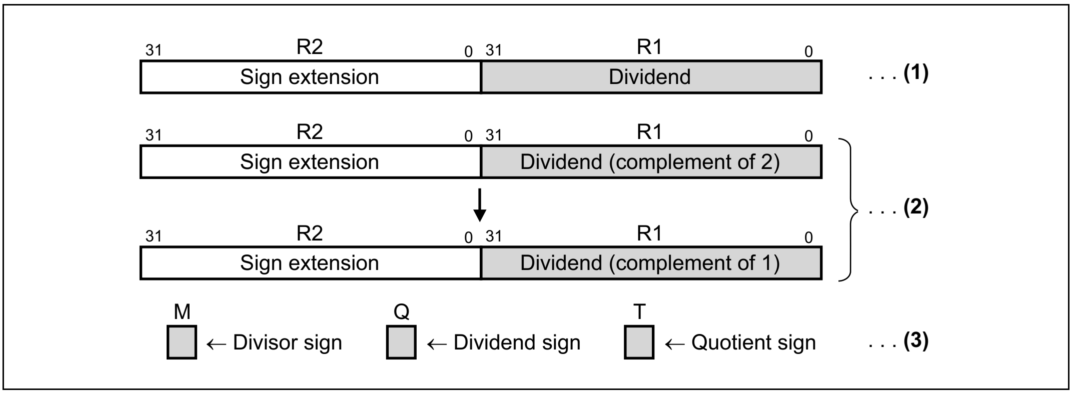


Figure 2 Initial Settings

(b) As shown in figure 3, the division operation is repeated through the number of divisor bits (32 times) using the ROCTL and DIV1 instructions.

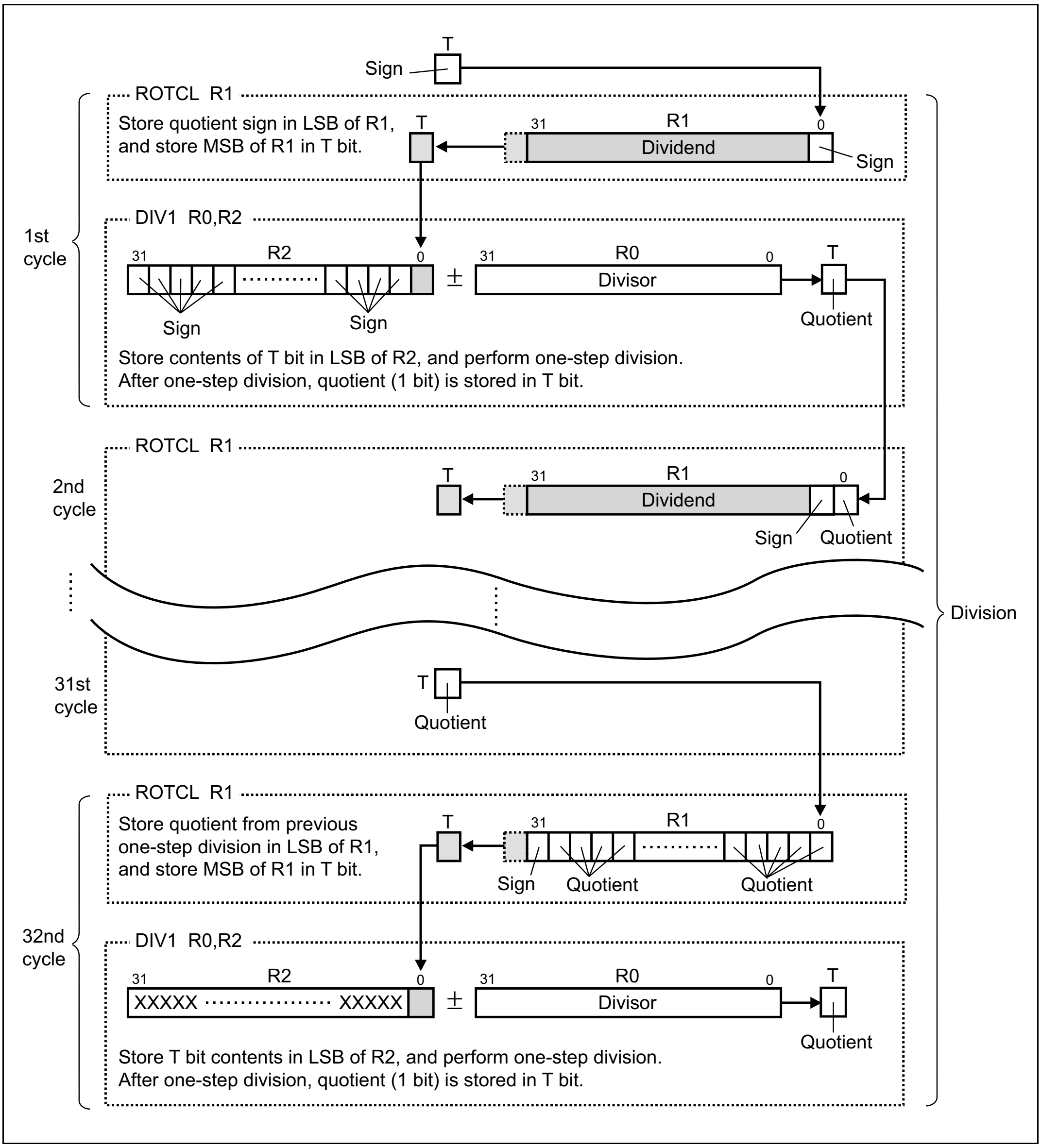


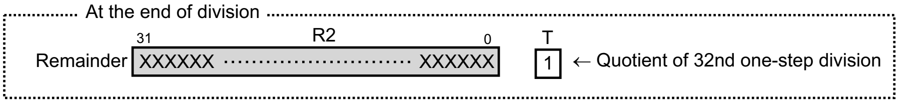
Figure 3 Operation Example

(c) - (i) As shown in figure 4, the way of determining the remainder differs depending on the dividend sign and the value of the T bit (quotient of 32nd one-step division).

- **If dividend is positive**

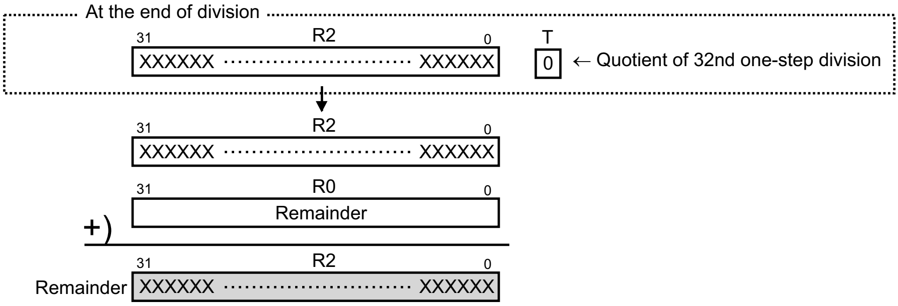
  T = 0

  The contents of R2 at the end of the division are the remainder.



  T = 1

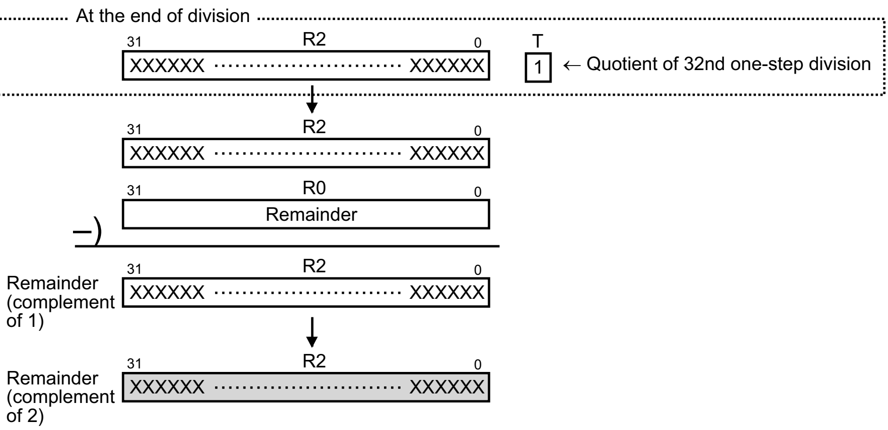
  The contents of R2 at the end of division, due to internal processing of the 32nd one-step division, are a value reflecting a one-time oversubtraction of the divisor from the remainder. Therefore, the divisor is added to R2



- **If dividend is negative**

  T = 1

  The contents of R2 at the end of division, due to internal processing of the 32nd one-step division, is the remainder plus one divisor over. Therefore, the divisor is subtracted from R2. Since the content of R2 becomes a complement of 1, it must then be changed to a complement of 2.



  T = 0

  The contents of R2 at the end of division is a remainder that is a complement of 1, so it is changed to a complement of 2.

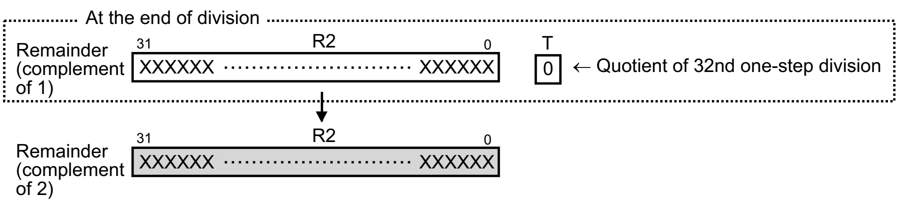


Figure 4 Remainder

(ii) The software instruction DIVS32R performs the processing described in (i) as follows. Note that R3 stores H'00000000, and the sign bit of the dividend is stored in the LSB of R4.

- The initialization instruction (DIV0S) of unsigned division is used to store the remainder's sign bit in the T bit and to store the remainder's sign bit in the T bit to R3.

```
DIV0S   R3,R2  } Remainder sign bit → R3
MOVT    R3
```

- Since the remainder sign bit and dividend sign bit become the same sign, the dividend sign (R4) and remainder sign (R3) are exclusively ORed to determine whether the two signs match. If the signs are different, the remainder is the value with the divisor added or subtracted one time too many.

Table 1 Remainder Signs

Dividend Sign	Remainder Sign
Positive	Positive
Negative	Negative

```
XOR     R4,R3
ROTCR   R3         } Dividend sign = remainder sign?
BF      DIVS32R1
```

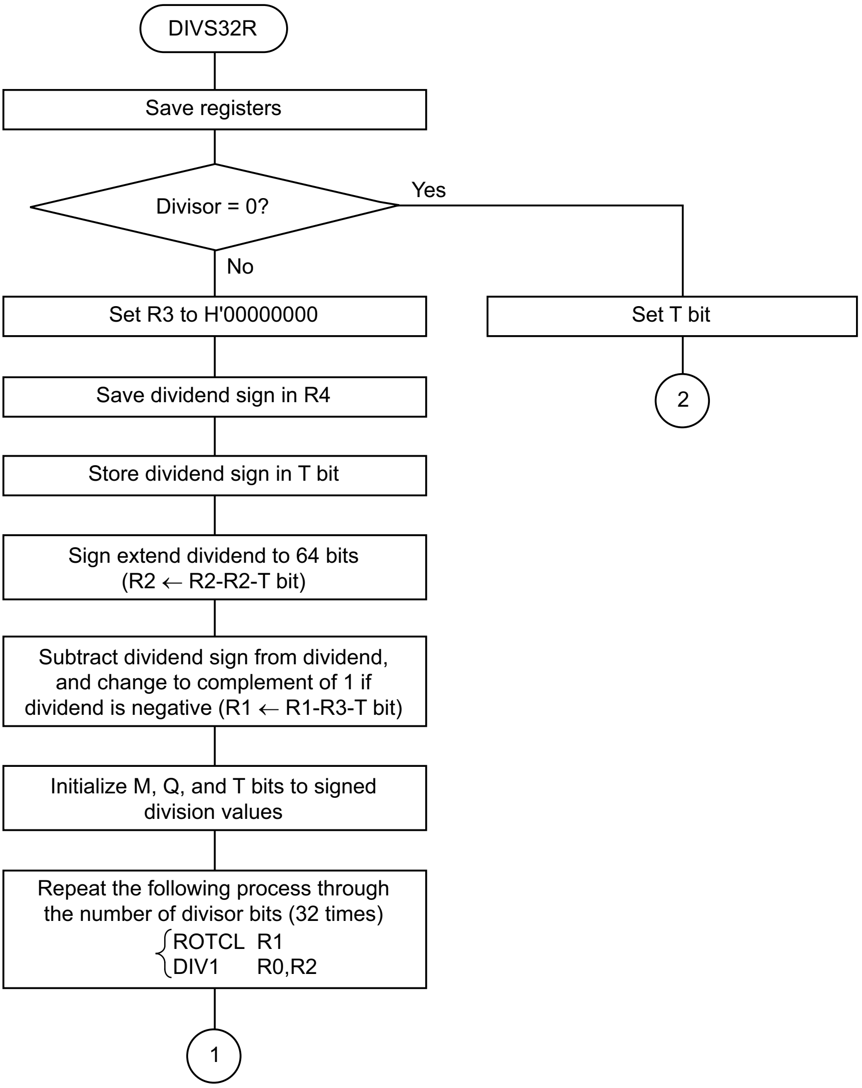
- The DIV1 instruction is used to correct for oversubtraction or overaddition.

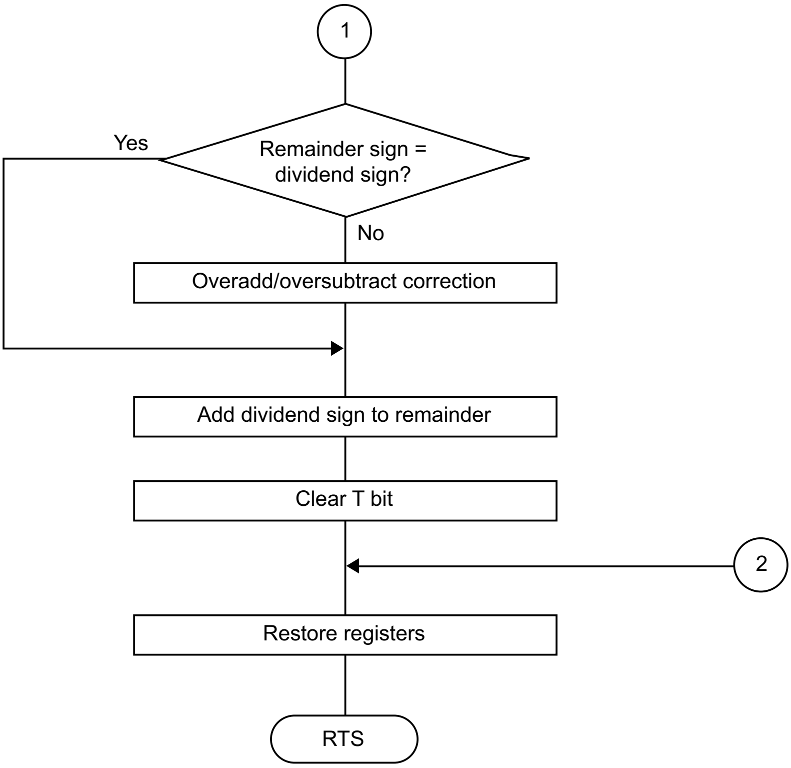
```
DIV0S   R0,R2   : Initializes signed division
SHAR    R2      : Remainder (R2) is halved due to execution of next DIV1
                  instruction, so R2 is doubled
DIV1    R0,R2   : Remainder is added when oversubtraction occurs due to DIV1
                  internal processing; remainder is subtracted when overaddition
                  occurs
```

- R4, which contains the remainder and dividend sign bit in the LSB, is added to the remainder (R2). The remainder and dividend sign bits indicate the same sign, so if the remainder is positive, 0 is added and the remainder does not change. If the remainder is negative, 1 is added and it becomes a complement of 2.

```
ADD   R4,R2
```

7. Flowchart


DIVS32R
Save registers
Divisor = 0?
Yes
No
Set R3 to H'00000000
Set T bit
2
Save dividend sign in R4
Store dividend sign in T bit
Sign extend dividend to 64 bits
(R2 ← R2-R2-T bit)
Subtract dividend sign from dividend,
and change to complement of 1 if
dividend is negative (R1 ← R1-R3-T bit)
Initialize M, Q, and T bits to signed
division values
Repeat the following process through
the number of divisor bits (32 times)
ROTCL R1
DIV1 R0,R2
1


1
Remainder sign = dividend sign?
Yes
No
Overadd/oversubtract correction
Add dividend sign to remainder
Clear T bit
2
Restore registers
RTS

8. Program Listing

```
                     ;****************************************************************
                     ;*                                                              *
                     ;*      NAME ; RESIDUAL OF 32 BIT SIGNED DIVISION (DIVS32R)    *
                     ;*                                                              *
                     ;****************************************************************
                     ;*                                                              *
                     ;*      ENTRY : R1 (DIVIDEND)                                   *
                     ;*              R0 (DIVISOR)                                    *
                     ;*    RETURNS : R2 (RESIDUAL)                                   *
                     ;*              T BIT (ERROR -> TRUE;T=1,FALSE;T=0              *
                     ;*                                                              *
                     ;****************************************************************
00001000                    .SECTION A,CODE,LOCATE=H'1000
         00001000    DIVS32R .EQU    $          ; Entry point
00001000 2F36               MOV.L   R3,@-R15   ; Escape register
00001002 2F46               MOV.L   R4,@-R15   ;
00001004 2008               TST     R0,R0      ; Divisor = 0?
00001006 8952               BT      DIVS32R2   ; Yes
00001008 233A               XOR     R3,R3      ; R3 <- H'00000000
0000100A 2137               DIV0S   R3,R1      ; T bit <- sign of Dividend
0000100C 0429               MOVT    R4         ; R4 <- T bit
0000100E 322A               SUBC    R2,R2      ; R2 sign extend
00001010 313A               SUBC    R3,R1      ;
00001012 2207               DIV0S   R0,R2      ; Divide as signed
                                               ;
00001014 4124               ROTCL   R1         ; Divide 1 step
00001016 3204               DIV1    R0,R2      ;
00001018 4124               ROTCL   R1         ;
0000101A 3204               DIV1    R0,R2      ;
0000101C 4124               ROTCL   R1         ;
0000101E 3204               DIV1    R0,R2      ;
00001020 4124               ROTCL   R1         ;
00001022 3204               DIV1    R0,R2      ;
00001024 4124               ROTCL   R1         ;
00001026 3204               DIV1    R0,R2      ;
00001028 4124               ROTCL   R1         ;
0000102A 3204               DIV1    R0,R2      ;
0000102C 4124               ROTCL   R1         ;
0000102E 3204               DIV1    R0,R2      ;
00001030 4124               ROTCL   R1         ;
00001032 3204               DIV1    R0,R2      ;
                                               ;
00001034 4124               ROTCL   R1         ;
00001036 3204               DIV1    R0,R2      ;
00001036 4124               ROTCL   R1         ;
0000103A 3204               DIV1    R0,R2      ;
0000103C 4124               ROTCL   R1         ;
0000103E 3204               DIV1    R0,R2      ;
00001040 4124               ROTCL   R1         ;
```

```
    00001042 3204          DIV1     R0,R2       ;
    00001044 4124          ROTCL    R1          ;
    00001046 3204          DIV1     R0,R2       ;
    00001048 4124          ROTCL    R1          ;
    0000104A 3204          DIV1     R0,R2       ;
    0000104C 4124          ROTCL    R1          ;
    0000104E 3204          DIV1     R0,R2       ;
    00001050 4124          ROTCL    R1          ;
    00001052 3204          DIV1     R0,R2       ;
                                                ;
    00001054 4124          ROTCL    R1          ;
    00001056 3204          DIV1     R0,R2       ;
    00001058 4124          ROTCL    R1          ;
    0000105A 3204          DIV1     R0,R2       ;
    0000105C 4124          ROTCL    R1          ;
    0000105E 3204          DIV1     R0,R2       ;
    00001060 4124          ROTCL    R1          ;
    00001062 3204          DIV1     R0,R2       ;
    00001064 4124          ROTCL    R1          ;
    00001066 3204          DIV1     R0,R2       ;
    00001068 4124          ROTCL    R1          ;
    0000106A 3204          DIV1     R0,R2       ;
    0000106C 4124          ROTCL    R1          ;
    0000106E 3204          DIV1     R0,R2       ;
    00001070 4124          ROTCL    R1          ;
    00001072 3204          DIV1     R0,R2       ;
                                                ;
    00001074 4124          ROTCL    R1          ;
    00001076 3204          DIV1     R0,R2       ;
    00001078 4124          ROTCL    R1          ;
    0000107A 3204          DIV1     R0,R2       ;
    0000107C 4124          ROTCL    R1          ;
    0000107E 3204          DIV1     R0,R2       ;
    00001080 4124          ROTCL    R1          ;
    00001082 3204          DIV1     R0,R2       ;
    00001084 4124          ROTCL    R1          ;
    00001086 3204          DIV1     R0,R2       ;
    00001088 4124          ROTCL    R1          ;
    0000108A 3204          DIV1     R0,R2       ;
    0000108C 4124          ROTCL    R1          ;
    0000108E 3204          DIV1     R0,R2       ;
    00001090 4124          ROTCL    R1          ;
    00001092 3204          DIV1     R0,R2       ;
                                                ;
    00001094 2237          DIV0S    R3,R2       ; R2 : keep sign
    00001096 0329          MOVT     R3          ;
    00001098 234A          XOR      R4,R3       ; (R4 Xor R3)--1? oversub or overadd
    0000109A 4325          ROTCR    R3          ;
    0000109C 8B02          BF       DIVS32R1    ; T bit = 0?
    0000109E 2207          DIV0S    R0,R2       ; Clear oversub or overadd
    000010A0 4221          SHAR     R2          ;
```

```
101 000010A2 3204        101         DIV1     R0,R2      ;
102 000010A4             102   DIVS32R1                  ;
103 000010A4 324C        103         ADD      R4,R2      ;
104 000010A6 0008        104         CLRT                ; T bit <- No error
105 000010A8 64F6        105         MOV.L    @R15+,R4   ; Return register
106 000010AA 000B        106         RTS                 ;
107 000010AC 63F6        107         MOV.L    @R15+,R3   ;
108 000010AE             108   DIVS32R2                  ;
109 000010AE 0018        109         SETT                ; T bit <- Error
110 000010B0 64F6        110         MOV.L    @R15+,R4   ; Return register
111 000010B2 000B        111         RTS                 ;
112 000010B4 63F6        112         MOV.L    @R15+,R3   ;
113 000010B4 63F6        113         .END
*****TOTAL ERRORS    0
```