お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

H8/300H Tiny シリーズ
単精度浮動小数点の乗算

要旨

汎用レジスタに設定された単精度浮動小数点の乗算を行ない，乗算結果を汎用レジスタに設定します。

動作確認デバイス

H8/300H Tiny シリーズ

目次

1. 機能 2
2. 引数 2
3. 内部レジスタ変化およびフラグ変化 2
4. プログラミング仕様 3
5. 注意事項 3
6. 説明 4
7. フローチャート 7
8. プログラムリスト 15
＜参考＞　単精度浮動小数点フォーマットについて 20

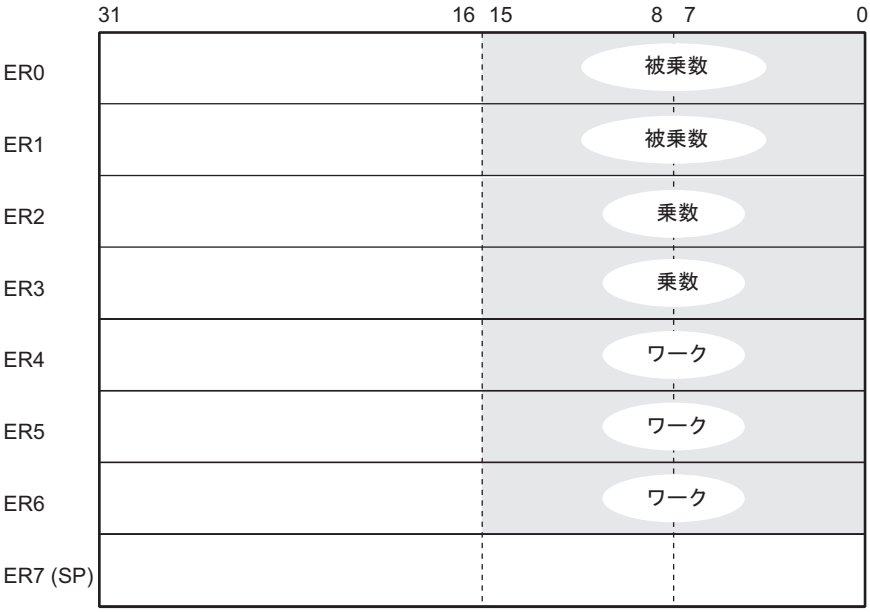
1. 機能

(1) 汎用レジスタに設定された単精度浮動小数点の乗算を行ない，乗算結果を汎用レジスタに設定します。
(2) 引数は，すべて単精度浮動小数点フォーマットです。

2. 引数

内容		格納場所	データ長
入力	被乗数	R0, R1	4
	乗数	R2, R3	4
出力	乗算結果	R0, R1	4

3. 内部レジスタ変化およびフラグ変化



I	UI	H	U	N	Z	V	C
–	–	↕	–	↕	↕	↕	↕

– ：不変
↕ ：変化
0 ：0固定
1 ：1固定

4. プログラミング仕様

プログラムメモリ (バイト)
348
データメモリ (バイト)
0
スタック (バイト)
16
ステート数
1078
リエントラント
可
リロケーション
可
途中割り込み
可

5. 注意事項

プログラミング仕様のステート数は，図 1 の例を実行した場合の値です。

浮動小数点フォーマットに関しては「＜参考＞　単精度浮動小数点フォーマットについて」を参照してください。

6. 説明

6.1 機能説明

(1) 引数の詳細は以下のとおりです。

(a) 入力引数は，次のように設定します。

R0：被乗数の上位 2 バイト
R1：被乗数の下位 2 バイト
R2：乗数の上位 2 バイト
R3：乗数の下位 2 バイト

(b) 出力引数は，次のように設定します。

R0：乗算結果の上位 2 バイト
R1：乗算結果の下位 2 バイト

(2) 図 1 にフトウェア FMUL の実行例を示します。
入力引数を図 1①のように設定すると，図 1②に示すように，乗算結果が R0, R1 に設定されます。

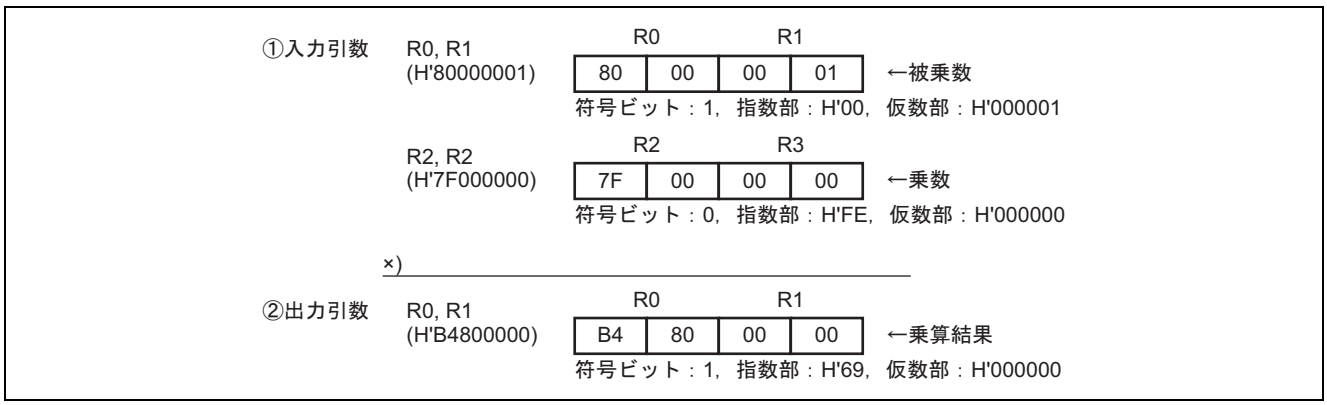


図 1　ソフトウェア FMUL の実行例

6.2 使用上の注意

(1) ソフトウェア FMUL で扱える最大値，最小値は，以下のとおりです。
正の最大値：H'7F80000
正の最小値：H'00000001
負の最大値：H'80000001
負の最小値：H'FF800000

(2) 単精度の正の数 H'7F800001～H'7FFFFFFF は，すべて最大値 (H'7F800000) として扱います。負の数 H'FF800000～H'FFFFFFFF は，すべて最小値 (H'FF800000) として扱います。

(3) 最大値は無限大 (∞) として取り扱いますので，∞ × 100 = ∞，∞ × (- 100) = -∞となります。(表 1 参照)

表 1 最大値を引数とした演算例

被乗数	乗数	結果
> H'7F800000 (+∞)	**正の数**	H'7F800000 (+∞)
	負の数	H'FF800000 (-∞)
< H'FF800000 (-∞)	**正の数**	H'FF800000 (-∞)
	負の数	H'7F800000 (+∞)
正の数	> H'7F800000 (+∞)	H'7F800000 (+∞)
	< H'FF800000 (-∞)	H'FF800000 (-∞)
負の数	> H'7F800000 (+∞)	H'FF800000 (-∞)
	< H'FF800000 (-∞)	H'7F800000 (+∞)

(4) H'80000000 は，H'00000000 (ゼロ) として扱います。
(5) ソフトウェア FMUL 実行後，汎用レジスタの被乗数，乗数データは破壊されます。
実行後も入力引数を必要とする場合は，あらかじめメモリ上に退避して下さい。

6.3 データメモリの説明

ソフトウェア FMUL では，データメモリを使用していません。

6.4 使用例

被乗数および乗数を設定し，ソフトウェア FMUL をサブルーチンコールします。

```
WORK1   .RES.W  2          ·····   被乗数を設定するデータメモリエリア
WORK2   .RES.W  2          ·····   乗数を設定するデータメモリエリア
WORK3   .RES.W  2          ·····   乗算結果を設定するデータメモリエリア
              .
              .
              .
        MOV.W  @WORK1,R0   ·····   ユーザプログラムで設定した被乗数を入力引数に設定
        MOV.W  @WORK1+2,R1
        MOV.W  @WORK2,R2   ·····   ユーザプログラムで設定した乗数を入力引数に設定
        MOV.W  @WORK2+2,R3
        JSR    @FMUL       ·····   ソフトウェアFMULをサブルーチンコール
        MOV.W  R0,@WORK3   ·····   出力引数に設定された乗算結果を格納します。
        MOV.W  R1,@WORK3+2
```

6.5 動作原理

単精度浮動小数点の乗算は，次のように行います。

(1) 被乗数，乗数が 0 であるか判定します。
被乗数，乗数のどちらかが 0 であれば，H'00000000 を出力値とします。
(2) 被乗数，乗数が無限大 (+∞, -∞) であるか判定します。
無限大 (+∞, -∞) であれば，表 1 に示す結果を出力値とします
(3) 被乗数，乗数の指数部の値を合わせます。
被乗数を R1 (符号ビット S1，指数部α1，仮数部β1) とし，乗数を R2 (符号ビット S2，指数部α2，仮数部β2) とすると

$$R_1 = (- 1)^{S1} \times 2^{\alpha 1 - 127} \times \beta_1$$
$$R_2 = (- 1)^{S2} \times 2^{\alpha 2 - 127} \times \beta_2$$

となり，乗算は，

$$R_1 \times R_2 = (- 1)^{S1 + S2} \times 2^{\alpha 1 + \alpha 2 - 127 - 127} \times \beta_1 \times \beta_2$$

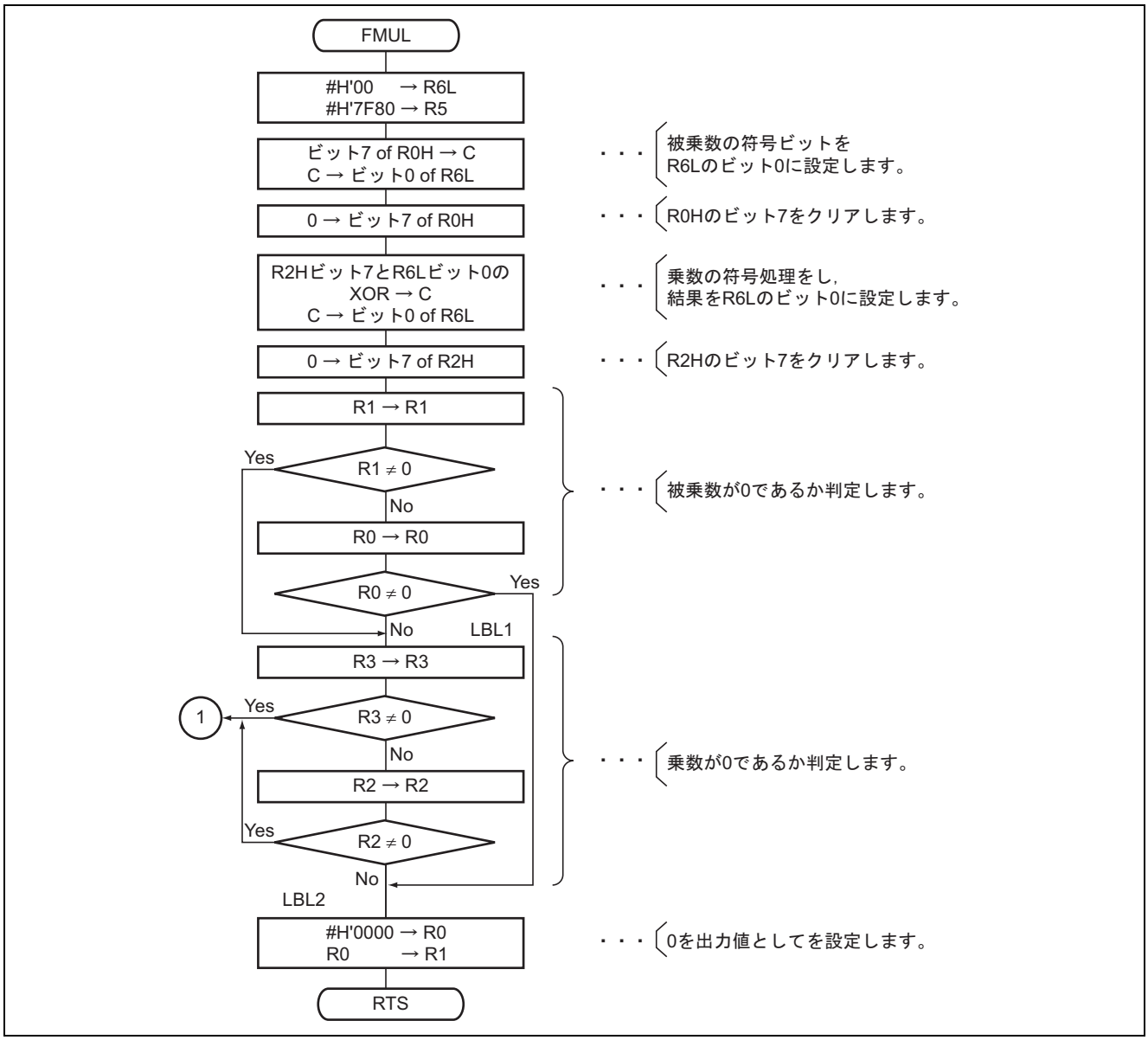
となります。浮動小数点フォーマットでは，指数部の乗算結果に H'7F (D'127) を加えるため，

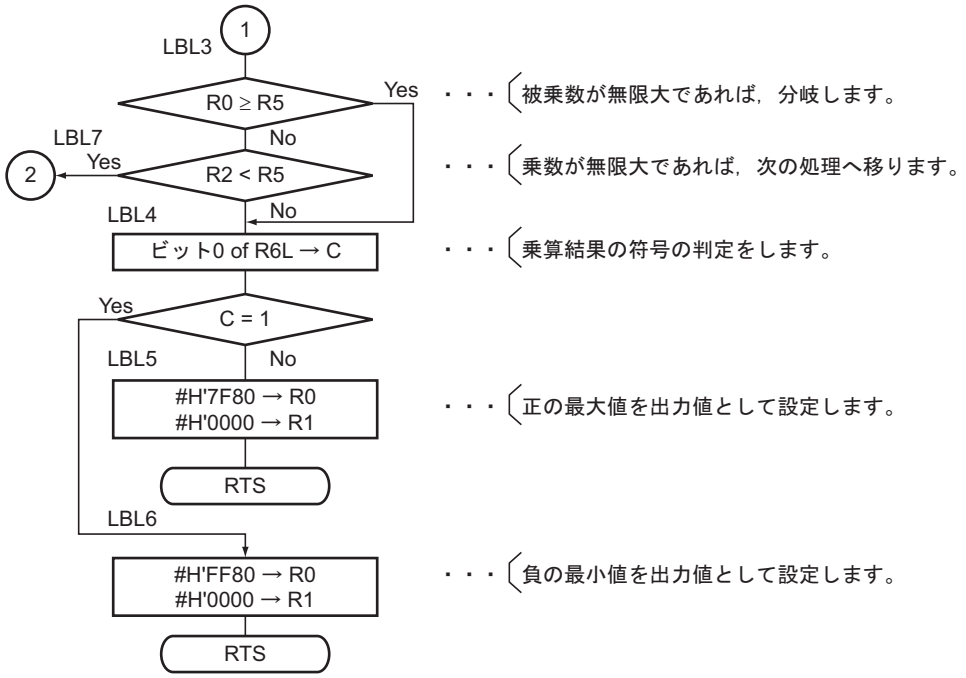
$$R_1 \times R_2 = (- 1)^{S1 + S2} \times 2^{\alpha 1 + \alpha 2 - 127} \times \beta_1 \times \beta_2$$

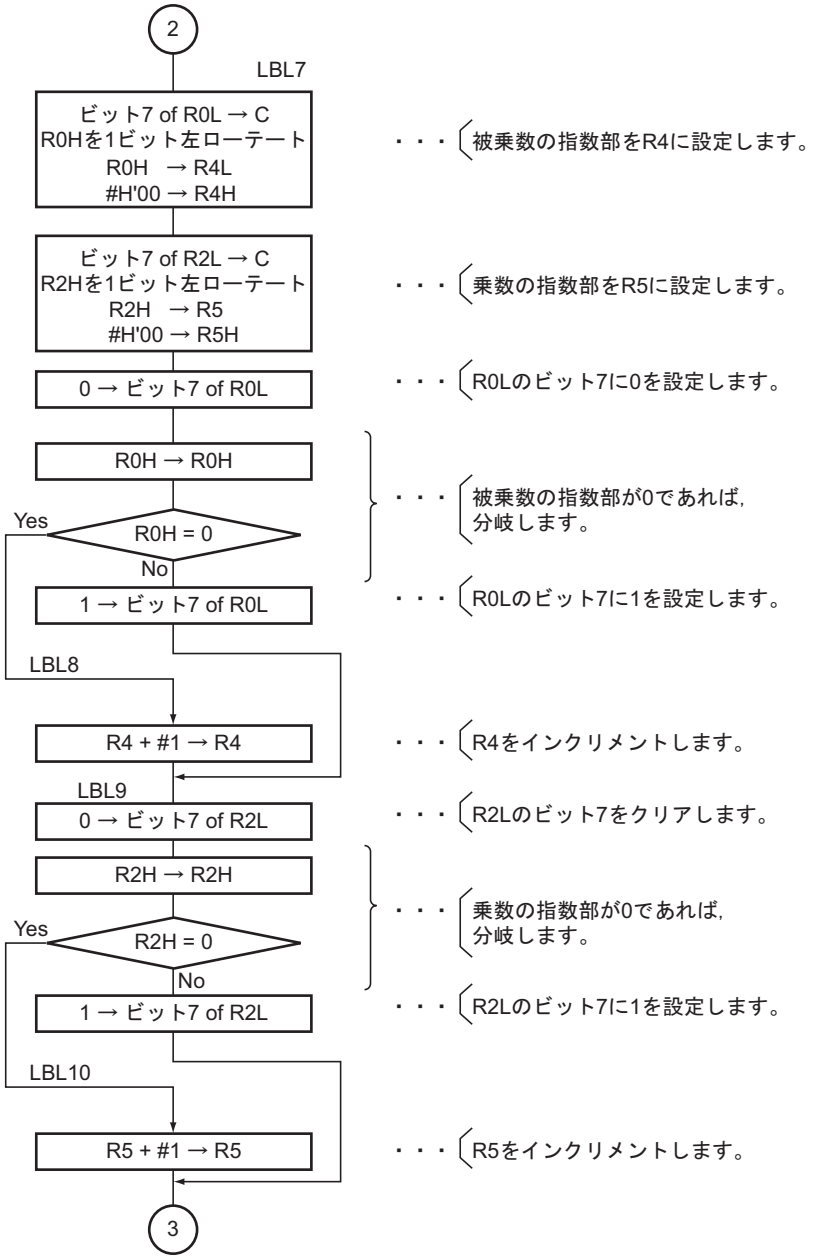
となります。よって，処理は次のように行います。

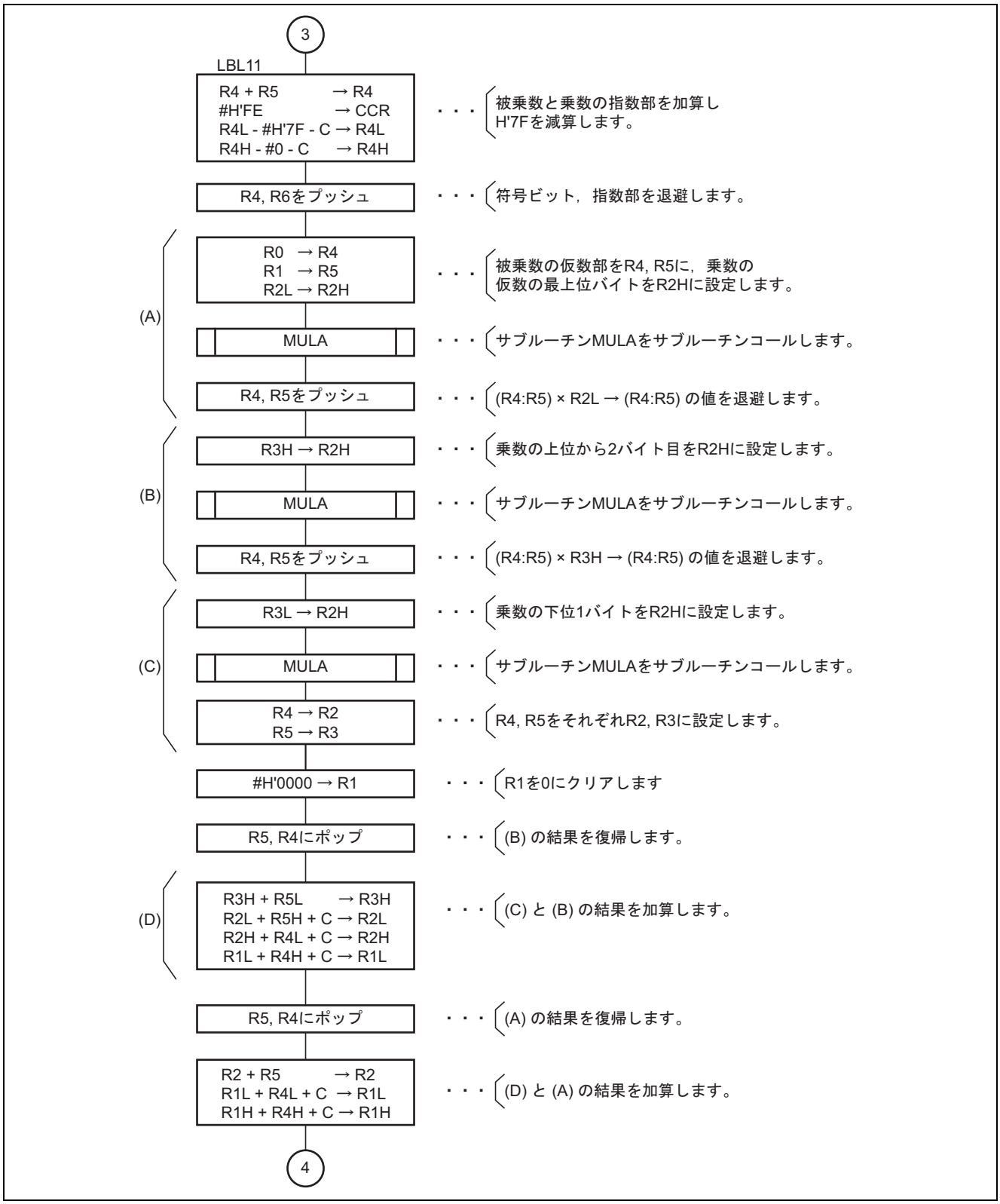
(a) 指数部を加算処理します。
α1, α2 は，ともに浮動小数点フォーマットに沿って H'7F (D'127) が加算されています。
乗算結果には H'7F (D'127) を加えるため，
$(\alpha_1 - H'7F) + (\alpha_2 - H'7F) + H'7F = \alpha_1 + \alpha_2 - H'7F$ とします。
(非正規化表現の場合，指数部にあらかじめ 1 を加算して処理します。)
(b) 仮数部を乗算処理します。
インプリシット MSB も含めて処理します。
(非正規化表現の場合，仮数部のインプリシット MSB は 0 として処理します。)
(c) 演算結果を浮動小数点フォーマットにします。

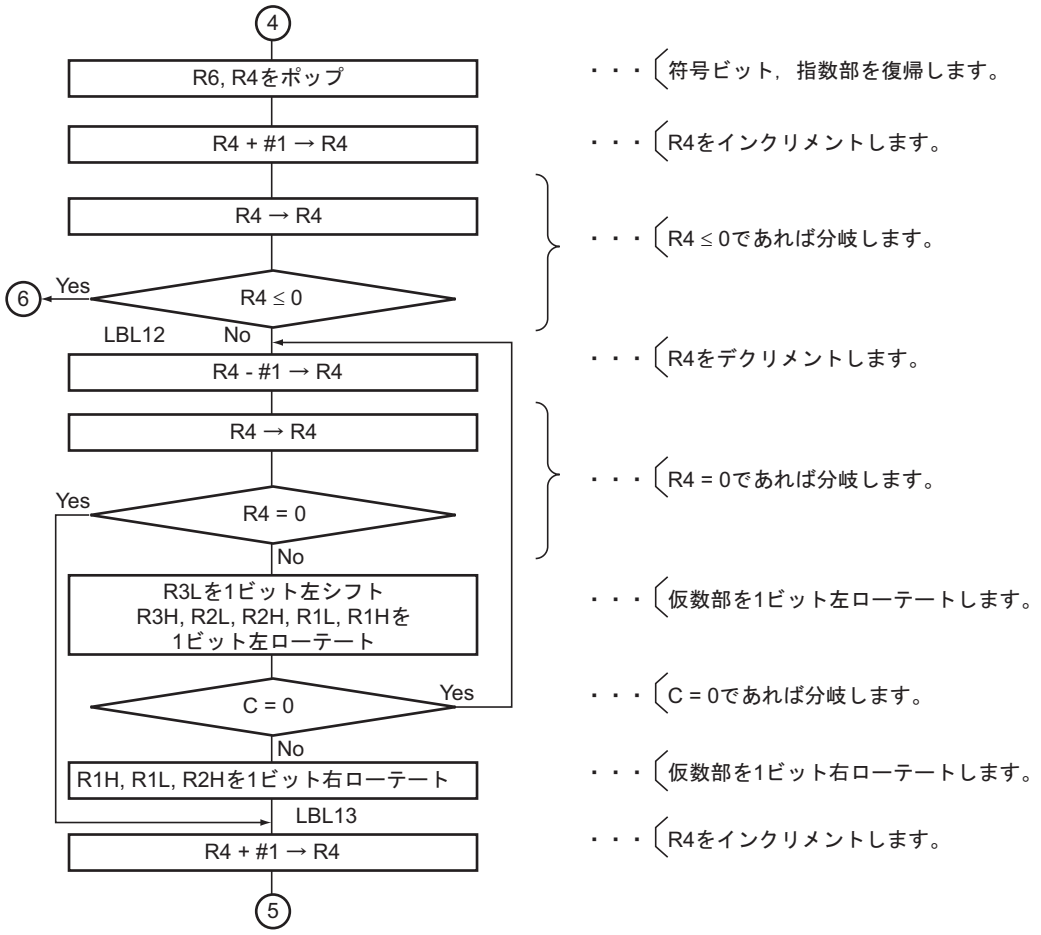
7. フローチャート

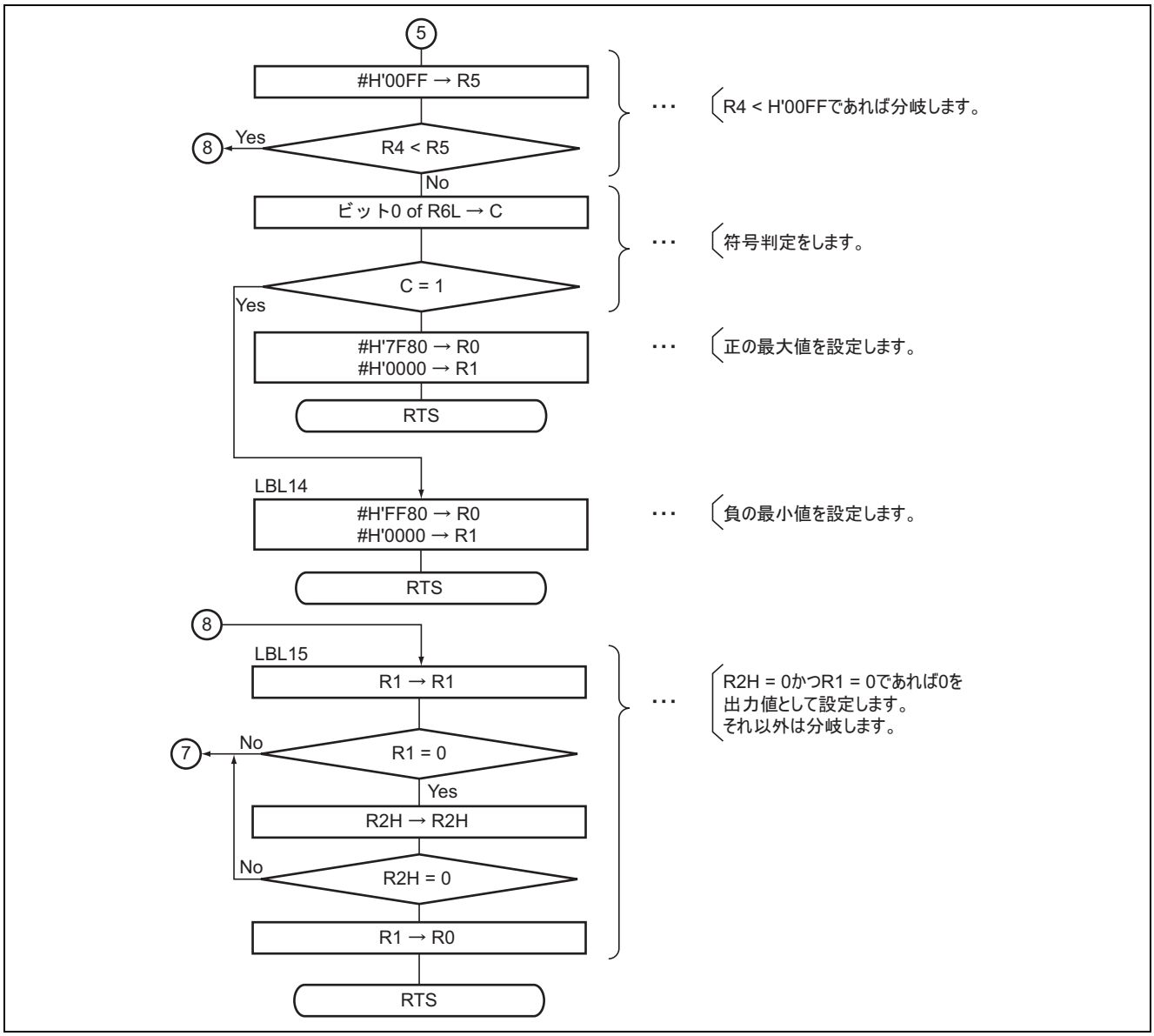

FMUL
#H'00 → R6L
#H'7F80 → R5
ビット7 of R0H → C
C → ビット0 of R6L
0 → ビット7 of R0H
R2Hビット7とR6Lビット0の
XOR → C
C → ビット0 of R6L
0 → ビット7 of R2H
R1 → R1
Yes
R1 ≠ 0
No
R0 → R0
R0 ≠ 0
Yes
No
LBL1
R3 → R3
1
Yes
R3 ≠ 0
No
R2 → R2
Yes
R2 ≠ 0
No
LBL2
#H'0000 → R0
R0 → R1
RTS
・・・ 被乗数の符号ビットを
R6Lのビット0に設定します。
・・・ R0Hのビット7をクリアします。
・・・ 乗数の符号処理をし，
結果をR6Lのビット0に設定します。
・・・ R2Hのビット7をクリアします。
・・・ 被乗数が0であるか判定します。
・・・ 乗数が0であるか判定します。
・・・ 0を出力値としてを設定します。

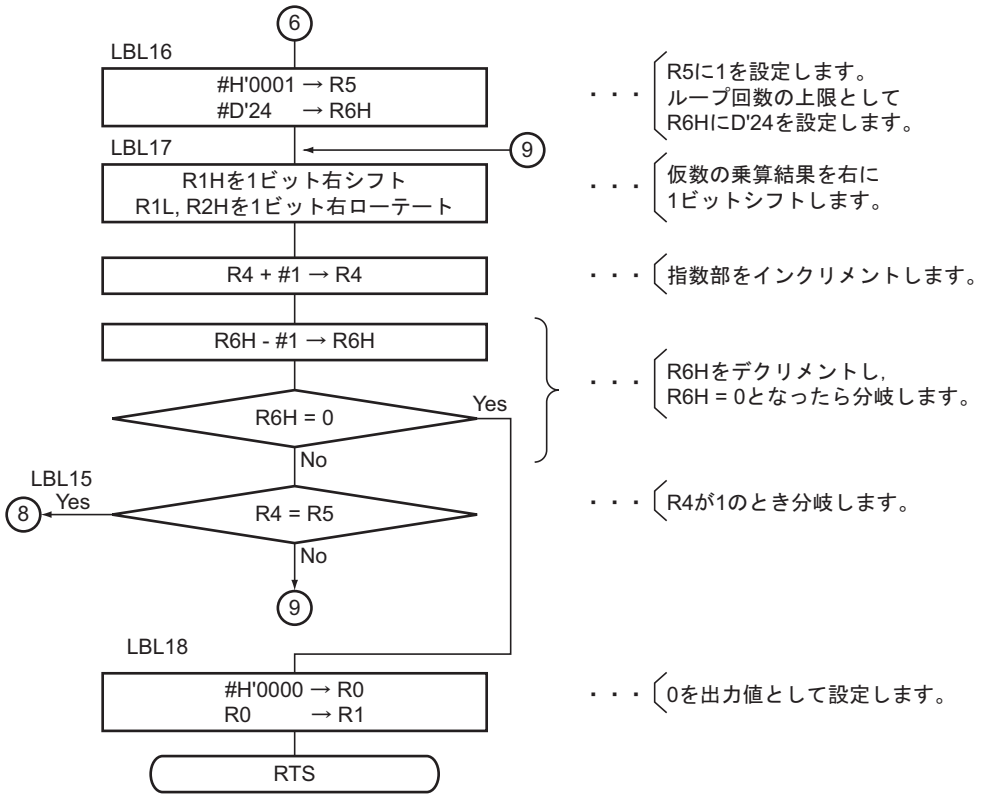

1
LBL3
R0 ≥ R5
Yes
・・・被乗数が無限大であれば，分岐します。
No
LBL7
Yes
2
R2 < R5
・・・乗数が無限大であれば，次の処理へ移ります。
No
LBL4
ビット0 of R6L → C
・・・乗算結果の符号の判定をします。
Yes
C = 1
No
LBL5
#H'7F80 → R0
#H'0000 → R1
・・・正の最大値を出力値として設定します。
RTS
LBL6
#H'FF80 → R0
#H'0000 → R1
・・・負の最小値を出力値として設定します。
RTS

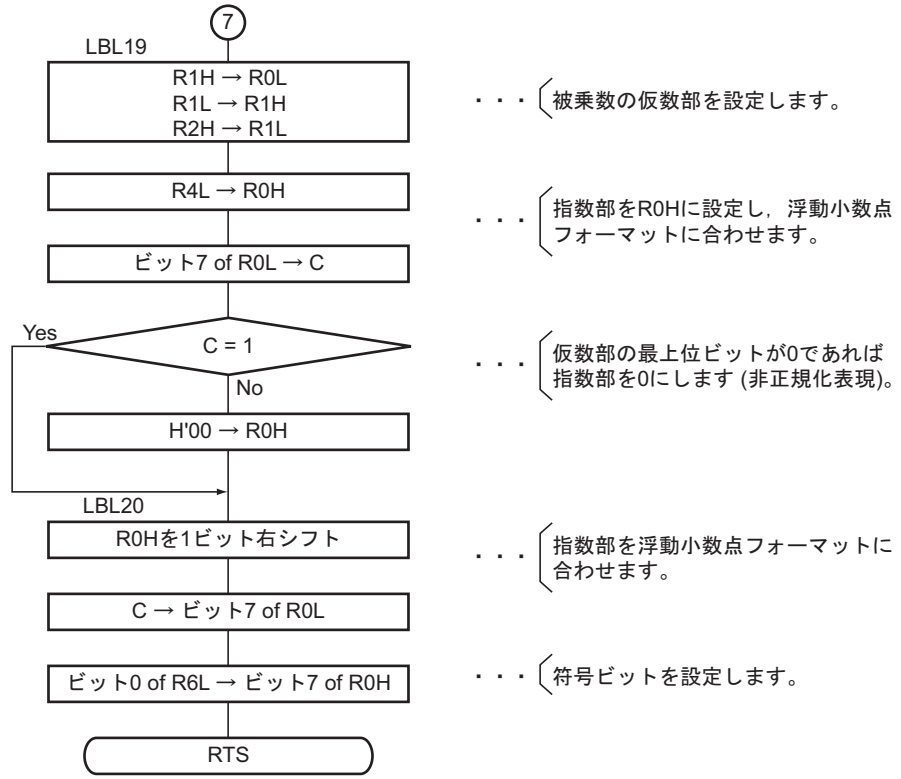

2
LBL7
ビット7 of R0L → C
R0Hを1ビット左ローテート
R0H → R4L
#H'00 → R4H
・・・被乗数の指数部をR4に設定します。
ビット7 of R2L → C
R2Hを1ビット左ローテート
R2H → R5
#H'00 → R5H
・・・乗数の指数部をR5に設定します。
0 → ビット7 of R0L
・・・R0Lのビット7に0を設定します。
R0H → R0H
Yes
R0H = 0
No
・・・被乗数の指数部が0であれば,
分岐します。
1 → ビット7 of R0L
・・・R0Lのビット7に1を設定します。
LBL8
R4 + #1 → R4
・・・R4をインクリメントします。
LBL9
0 → ビット7 of R2L
・・・R2Lのビット7をクリアします。
R2H → R2H
Yes
R2H = 0
No
・・・乗数の指数部が0であれば,
分岐します。
1 → ビット7 of R2L
・・・R2Lのビット7に1を設定します。
LBL10
R5 + #1 → R5
・・・R5をインクリメントします。
3

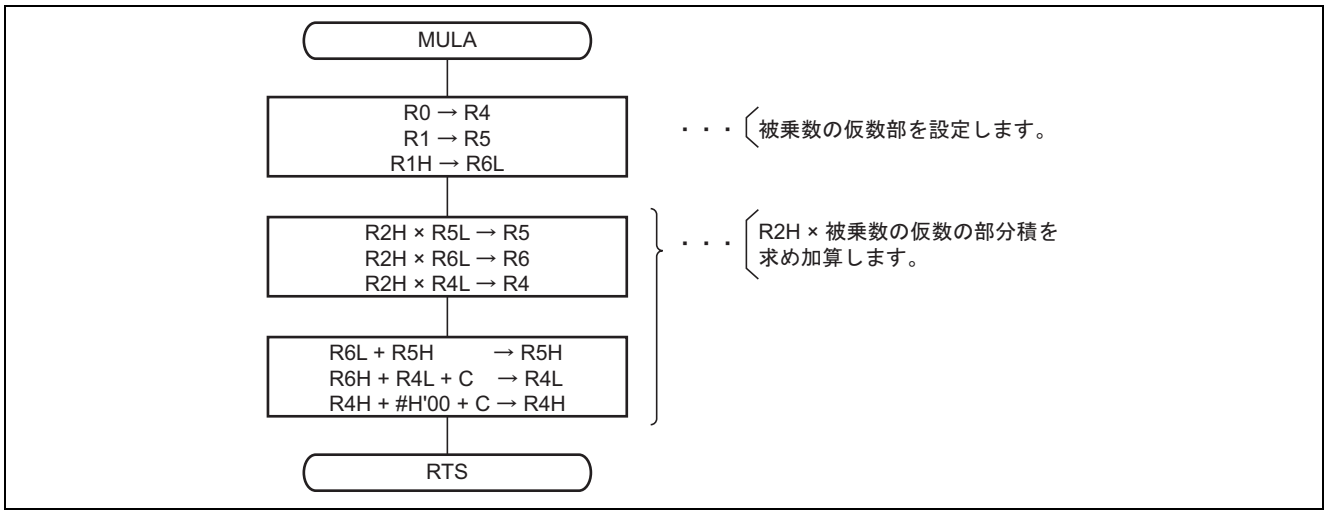

3
LBL11
R4 + R5 → R4
#H'FE → CCR
R4L - #H'7F - C → R4L
R4H - #0 - C → R4H
・・・ 被乗数と乗数の指数部を加算し H'7Fを減算します。
R4, R6をプッシュ
・・・ 符号ビット，指数部を退避します。
(A)
R0 → R4
R1 → R5
R2L → R2H
・・・ 被乗数の仮数部をR4, R5に，乗数の 仮数の最上位バイトをR2Hに設定します。
MULA
・・・ サブルーチンMULAをサブルーチンコールします。
R4, R5をプッシュ
・・・ (R4:R5) × R2L → (R4:R5) の値を退避します。
(B)
R3H → R2H
・・・ 乗数の上位から2バイト目をR2Hに設定します。
MULA
・・・ サブルーチンMULAをサブルーチンコールします。
R4, R5をプッシュ
・・・ (R4:R5) × R3H → (R4:R5) の値を退避します。
(C)
R3L → R2H
・・・ 乗数の下位1バイトをR2Hに設定します。
MULA
・・・ サブルーチンMULAをサブルーチンコールします。
R4 → R2
R5 → R3
・・・ R4, R5をそれぞれR2, R3に設定します。
#H'0000 → R1
・・・ R1を0にクリアします
R5, R4にポップ
・・・ (B) の結果を復帰します。
(D)
R3H + R5L → R3H
R2L + R5H + C → R2L
R2H + R4L + C → R2H
R1L + R4H + C → R1L
・・・ (C) と (B) の結果を加算します。
R5, R4にポップ
・・・ (A) の結果を復帰します。
R2 + R5 → R2
R1L + R4L + C → R1L
R1H + R4H + C → R1H
・・・ (D) と (A) の結果を加算します。
4


4
R6, R4をポップ
R4 + #1 → R4
R4 → R4
6
Yes
R4 ≤ 0
LBL12
No
R4 - #1 → R4
R4 → R4
Yes
R4 = 0
No
R3Lを1ビット左シフト
R3H, R2L, R2H, R1L, R1Hを
1ビット左ローテート
C = 0
Yes
No
R1H, R1L, R2Hを1ビット右ローテート
LBL13
R4 + #1 → R4
5
・・・符号ビット，指数部を復帰します。
・・・R4をインクリメントします。
・・・R4 ≤ 0であれば分岐します。
・・・R4をデクリメントします。
・・・R4 = 0であれば分岐します。
・・・仮数部を1ビット左ローテートします。
・・・C = 0であれば分岐します。
・・・仮数部を1ビット右ローテートします。
・・・R4をインクリメントします。


5
#H'00FF → R5
8
Yes
R4 < R5
No
ビット0 of R6L → C
C = 1
Yes
#H'7F80 → R0
#H'0000 → R1
RTS
LBL14
#H'FF80 → R0
#H'0000 → R1
RTS
8
LBL15
R1 → R1
7
No
R1 = 0
Yes
R2H → R2H
No
R2H = 0
R1 → R0
RTS
R4 < H'00FFであれば分岐します。
符号判定をします。
正の最大値を設定します。
負の最小値を設定します。
R2H = 0かつR1 = 0であれば0を
出力値として設定します。
それ以外は分岐します。


6
LBL16
#H'0001 → R5
#D'24 → R6H
・・・ R5に1を設定します。
ループ回数の上限として
R6HにD'24を設定します。
LBL17
9
R1Hを1ビット右シフト
R1L, R2Hを1ビット右ローテート
・・・ 仮数の乗算結果を右に
1ビットシフトします。
R4 + #1 → R4
・・・ 指数部をインクリメントします。
R6H - #1 → R6H
R6H = 0
Yes
No
・・・ R6Hをデクリメントし,
R6H = 0となったら分岐します。
LBL15
Yes
8
R4 = R5
No
・・・ R4が1のとき分岐します。
9
LBL18
#H'0000 → R0
R0 → R1
・・・ 0を出力値として設定します。
RTS


7
LBL19
R1H → R0L
R1L → R1H
R2H → R1L
・・・ 被乗数の仮数部を設定します。
R4L → R0H
ビット7 of R0L → C
・・・ 指数部をR0Hに設定し，浮動小数点
フォーマットに合わせます。
Yes
C = 1
No
・・・ 仮数部の最上位ビットが0であれば
指数部を0にします (非正規化表現)。
H'00 → R0H
LBL20
R0Hを1ビット右シフト
・・・ 指数部を浮動小数点フォーマットに
合わせます。
C → ビット7 of R0L
ビット0 of R6L → ビット7 of R0H
・・・ 符号ビットを設定します。
RTS


MULA
R0 → R4
R1 → R5
R1H → R6L
・・・被乗数の仮数部を設定します。
R2H × R5L → R5
R2H × R6L → R6
R2H × R4L → R4
・・・R2H × 被乗数の仮数の部分積を
求め加算します。
R6L + R5H → R5H
R6H + R4L + C → R4L
R4H + #H'00 + C → R4H
RTS

8. プログラムリスト

```
1                    1  ;*********************************************************************
2                    2  ;*                                                                   *
3                    3  ;*         NAME:    FLOATING POINT MULTIPLICATION (FMUL)            *
4                    4  ;*                                                                   *
5                    5  ;*********************************************************************
6                    6  ;*                                                                   *
7                    7  ;*       ENTRY:     R0    (HIGHER WORD OF MULTIPLICAND)             *
8                    8  ;*                  R1    (LOWER WORD OF MULTIPLICAND)              *
9                    9  ;*                  R2    (HIGHER WORD OF MULTIPLIER)               *
10                  10  ;*                  R3    (LOWER WORD OF MULTIPLIER)                *
11                  11  ;*                                                                   *
12                  12  ;*      RETURNS:    R0    (HIGHER WORD OF RESULT)                   *
13                  13  ;*                  R1    (LOWER WORD OF RESULT)                    *
14                  14  ;*                                                                   *
15                  15  ;*********************************************************************
16                  16  ;
17                  17              .CPU     300HN
18  0000            18              .SECTION FMUL_code,CODE,ALIGN=2
19                  19              .EXPORT  FMUL
20                  20  ;
21        00000000  21  FMUL        .EQU     $                ;Entry point
22  0000 FE00       22              MOV.B    #H'00,R6L        ;Clear R6L
23  0002 79057F80   23              MOV.W    #H'7F80,R5       ;Set H'7F80
24                  24  ;
25  0006 7770       25              BLD      #7,R0H           ;Set sign bit of multiplicand
26  0008 670E       26              BST      #0,R6L           ;  to bit 0 of R6L
27  000A 7270       27              BCLR     #7,R0H           ;Bit clear bit 7 of R0H
28                  28  ;
29  000C 7772       29              BLD      #7,R2H           ;
30  000E 750E       30              BXOR     #0,R6L           ;Set sign bit of result
31  0010 670E       31              BST      #0,R6L           ;  to bit 0 of R6L
32  0012 7272       32              BCLR     #7,R2H           ;Bit clear bit 7 of R2H
33                  33  ;
34  0014 0D11       34              MOV.W    R1,R1            ;
35  0016 4604       35              BNE      LBL1
36  0018 0D00       36              MOV.W    R0,R0
37  001A 4708       37              BEQ      LBL2             ;Branch if R1=R=0
38  001C            38  LBL1
39  001C 0D33       39              MOV.W    R3,R3            ;
40  001E 460C       40              BNE      LBL3             ;Branch if not R3 = 0
41  0020 0D22       41              MOV.W    R2,R2
42  0022 4608       42              BNE      LBL3             ;Branch if not R2 = 0
43  0024            43  LBL2
44  0024 79000000   44              MOV.W    #H'0000,R0       ;Set 0 as result
45  0028 0D01       45              MOV.W    R0,R1
46  002A 5470       46              RTS
47  002C            47  LBL3
48  002C 1D05       48              CMP.W    R0,R5
49  002E 4304       49              BLS      LBL4             ;Branch if R0>=R5
50  0030 1D25       50              CMP.W    R2,R5
51  0032 4218       51              BHI      LBL7             ;Branch if R2>=R5
52                  52  ;
53  0034            53  LBL4
54  0034 770E       54              BLD      #0,R6L           ;Load sign bit
55  0036 450A       55              BCS      LBL6             ;Branch if C=1
```

```
 56  0038               56 LBL5
 57  0038 79007F80      57            MOV.W   #H'7F80,R0      ;Set #H'7F800000 as result
 58  003C 79010000      58            MOV.W   #H'0000,R1
 59  0040 5470          59            RTS
 60                     60 ;
 61  0042               61 LBL6
 62  0042 7900FF80      62            MOV.W   #H'FF80,R0      ;Set #H'FF800000 as result
 63  0046 79010000      63            MOV.W   #H'0000,R1
 64  004A 5470          64            RTS
 65                     65 ;
 66  004C               66 LBL7
 67  004C 7778          67            BLD     #7,R0L
 68  004E 1200          68            ROTXL   R0H
 69  0050 0C0C          69            MOV.B   R0H,R4L         ;Set exponent of multiplicand in R4
 70  0052 F400          70            MOV.B   #H'00,R4H
 71                     71 ;
 72  0054 777A          72            BLD     #7,R2L
 73  0056 1202          73            ROTXL   R2H
 74  0058 0C2D          74            MOV.B   R2H,R5L         ;Set exponent of multiplier in R5
 75  005A F500          75            MOV.B   #H'00,R5H
 76                     76 ;
 77  005C 7278          77            BCLR    #7,R0L          ;Clear bit 7 of R0L
 78  005E 0C00          78            MOV.B   R0H,R0H
 79  0060 4704          79            BEQ     LBL8            ;Branch if multiplier is denormalized
 80  0062 7078          80            BSET    #7,R0L          ;Set implicit MSB
 81  0064 4004          81            BRA     LBL9            ;Branch always
 82  0066               82 LBL8
 83  0066 79140001      83            ADD.W   #1,R4
 84  006A               84 LBL9
 85  006A 727A          85            BCLR    #7,R2L          ;Clear bit 7 of R2L
 86  006C 0C22          86            MOV.B   R2H,R2H
 87  006E 4704          87            BEQ     LBL10           ;Branch if multiplier is denormalized
 88  0070 707A          88            BSET    #7,R2L          ;Set implicit MSB
 89  0072 4004          89            BRA     LBL11           ;Branch always
 90  0074               90 LBL10
 91  0074 79150001      91            ADD.W   #1,R5
 92                     92 ;
 93  0078               93 LBL11
 94  0078 0954          94            ADD.W   R5,R4           ;addition exponents
 95  007A 06FE          95            ANDC    #H'FE,CCR       ;Clear C flag of CCR
 96  007C BC7F          96            SUBX.B  #H'7F,R4L       ;R4L - #H'7F - C -> R4L
 97  007E B400          97            SUBX.B  #H'00,R4H
 98                     98 ;
 99  0080 6DF4          99            PUSH    R4              ;Push R4
100  0082 6DF6         100            PUSH    R6              ;Push R6
101                    101 ;
102  0084 0D04         102            MOV.W   R0,R4
103  0086 0D15         103            MOV.W   R1,R5
104                    104 ;
105  0088 0CA2         105            MOV.B   R2L,R2H
106  008A 5E000000     106            JSR     @MULA           ;R2L * (R0L:R1) -> (R4:R5)
107  008E 6DF4         107            PUSH    R4              ;Push R4
108  0090 6DF5         108            PUSH    R5              ;Push R5
109                    109
110  0092 0C32         110            MOV.B   R3H,R2H         ;
111  0094 5E000000     111            JSR     @MULA           ;R3L * (R0L:R1) -> (R4:R5)
112  0098 6DF4         112            PUSH    R4              ;Push R4
```

```
113   009A 6DF5         113           PUSH    R5              ;Push R5
114                     114 ;
115   009C 0CB2         115           MOV.B   R3L,R2H         ;R3L * (R0L:R1) -> (R4:R5)
116   009E 5E000000     116           JSR     @MULA           ;Push R4
117   00A2 0D42         117           MOV.W   R4,R2           ;Push R5
118   00A4 0D53         118           MOV.W   R5,R3
119                     119 ;
120   00A6 79010000     120           MOV.W   #H'0000,R1      ;Clear R1
121   00AA 6D75         121           POP     R5              ;Pop   R5
122   00AC 6D74         122           POP     R4              ;Pop   R4
123                     123 ;
124   00AE 08D3         124           ADD.B   R5L,R3H         ;R3H + R5L + C -> R3H
125   00B0 0E5A         125           ADDX.B  R5H,R2L         ;R2L + R5H + C -> R2L
126   00B2 0EC2         126           ADDX.B  R4L,R2H         ;R2H + R4L + C -> R2H
127   00B4 0E49         127           ADDX.B  R4H,R1L         ;R1L + R4H + C -> R1L
128                     128 ;
129   00B6 6D75         129           POP     R5              ;Pop   R5
130   00B8 6D74         130           POP     R4              ;Pop   R4
131   00BA 0952         131           ADD.W   R5,R2           ;R2  + R5     -> R2L
132   00BC 0EC9         132           ADDX.B  R4L,R1L         ;R1L + R4L + C -> R1L
133   00BE 0E41         133           ADDX.B  R4H,R1H         ;R1H + R4H + C -> R1H
134                     134 ;
135   00C0 6D76         135           POP     R6              ;Pop   R6
136   00C2 6D74         136           POP     R4              ;Pop   R4
137   00C4 79140001     137           ADD.W   #1,R4
138   00C8 0D44         138           MOV.W   R4,R4
139                     139 ;
140   00CA 474E         140           BEQ     LBL16           ;Branch if R4=0
141   00CC 4B4C         141           BMI     LBL16           ;Branch if R4<0
142   00CE              142 LBL12
143   00CE 79340001     143           SUB.W   #1,R4
144   00D2 0D44         144           MOV.W   R4,R4
145   00D4 4714         145           BEQ     LBL13           ;Branch if R4=0
146   00D6 100B         146           SHLL    R3L             ;Shift mantissa 1 bit left
147   00D8 1203         147           ROTXL   R3H
148   00DA 120A         148           ROTXL   R2L
149   00DC 1202         149           ROTXL   R2H
150   00DE 1209         150           ROTXL   R1L
151   00E0 1201         151           ROTXL   R1H
152   00E2 44EA         152           BCC     LBL12           ;Branch if C=0
153   00E4 1301         153           ROTXR   R1H             ;Rotate mantissa 1 bit right
154   00E6 1309         154           ROTXR   R1L
155   00E8 1302         155           ROTXR   R2H
156   00EA              156 LBL13
157   00EA 79140001     157           ADD.W   #1,R4
158                     158 ;
159   00EE 790500FF     159           MOV.W   #H'00FF,R5
160   00F2 1D45         160           CMP.W   R4,R5
161   00F4 4418         161           BCC     LBL15           ;Branch if R5>R4
162   00F6 770E         162           BLD     #0,R6L          ;Load sign bit
163   00F8 450A         163           BCS     LBL14           ;Branch if C=1
164   00FA 79007F80     164           MOV.W   #H'7F80,R0      ;Set H'7F800000 to result
165   00FE 79010000     165           MOV.W   #H'0000,R1
166   0102 5470         166           RTS
167                     167 ;
168   0104              168 LBL14
169   0104 7900FF80     169           MOV.W   #H'FF80,R0      ;Set H'FF800000 to product
```

```
170  0108 79010000     170           MOV.W   #H'0000,R1
171  010C 5470         171           RTS
172  010E              172 LBL15
173  010E 0D11         173           MOV.W   R1,R1
174  0110 462A         174           BNE     LBL19           ;Branch if not R1=0
175  0112 0C22         175           MOV.B   R2H,R2H
176  0114 4626         176           BNE     LBL19           ;Branch if not R2H=0
177  0116 0D10         177           MOV.W   R1,R0
178  0118 5470         178           RTS
179                    179 ;
180  011A              180 LBL16
181  011A 79050001     181           MOV.W   #H'0001,R5      ;Set #H'0001 to R5
182  011E F618         182           MOV.B   #D'24,R6H       ;Set bit counter
183  0120              183 LBL17
184  0120 1101         184           SHLR    R1H             ;Shift mantissa 1 bit right
185  0122 1309         185           ROTXR   R1L
186  0124 1302         186           ROTXR   R2H
187  0126 79140001     187           ADD.W   #1,R4           ;Increment exponent
188  012A 1A06         188           DEC.B   R6H             ;Decrement bit counter
189  012C 4706         189           BEQ     LBL18           ;Branch if Z=1
190  012E 1D54         190           CMP.W   R5,R4
191  0130 47DC         191           BEQ     LBL15           ;Branch if R5=R4
192  0132 40EC         192           BRA     LBL17           ;Branch always
193  0134              193 LBL18
194  0134 79000000     194           MOV.W   #H'0000,R0      ;Clear result
195  0138 0D01         195           MOV.W   R0,R1
196  013A 5470         196           RTS
197                    197 ;
198  013C              198 LBL19
199  013C 0C18         199           MOV.B   R1H,R0L
200  013E 0C91         200           MOV.B   R1L,R1H
201  0140 0C29         201           MOV.B   R2H,R1L
202                    202 ;
203  0142 0CC0         203           MOV.B   R4L,R0H
204  0144 7778         204           BLD     #7,R0L
205  0146 4502         205           BCS     LBL20           ;Branch if C=1
206  0148 F000         206           MOV.B   #H'00,R0H
207  014A              207 LBL20                             ;Correct into floating-point format
208  014A 1100         208           SHLR    R0H
209  014C 6778         209           BST     #7,R0L
210  014E 770E         210           BLD     #0,R6L
211  0150 6770         211           BST     #7,R0H
212  0152 5470         212           RTS
213                    213 ;
214                    214 ;---------------------------------------------------------------------
215                    215 ;
216  0154              216 MULA                              ;R2H * (R0L:R1) -> (R4:R5)
217  0154 0D04         217           MOV.W   R0,R4           ;R0  -> R4
218  0156 0D15         218           MOV.W   R1,R5           ;R1  -> R5
219  0158 0C1E         219           MOV.B   R1H,R6L         ;R1H -> R6L
220                    220 ;
221  015A 5025         221           MULXU   R2H,R5          ;R2H * R5L -> R5
222  015C 5026         222           MULXU   R2H,R6          ;R2H * R6L -> R6
223  015E 5024         223           MULXU   R2H,R4          ;R2H * R4L -> R4
224                    224 ;
225  0160 08E5         225           ADD.B   R6L,R5H         ;R5H + R6L      -> R5H
226  0162 0E6C         226           ADDX.B  R6H,R4L         ;R4L + R6H   + C -> R4L
```

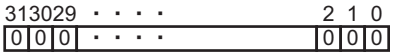
```
227   0164 9400          227            ADDX.B   #H'00,R4H        ;R4H + #H'00 + C -> R4H
228   0166 5470          228            RTS
229                      229 ;
230                      230            .END
*****TOTAL ERRORS      0
*****TOTAL WARNINGS    0
```

＜参考＞　単精度浮動小数点フォーマットについて

(1) 単精度浮動小数点の内部表現について

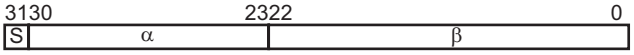
本アプリケーションノートでは，単精度浮動小数点の値により，以下の表現方法を用います。
(実数を R とする。)

(a) R = 0 のときの内部表現



32ビットのすべてのビットが0です。

(b) 正規表現 (Normalized Format)



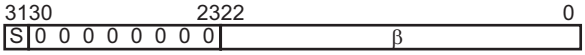
ここでαはフィールド長が 8 ビットの指数です。βはフィールド長が 23 ビットの仮数です。
この時の R の値は，次の式で表現できます。 (1 ≤ α ≤ 254 の場合)

↓インプリシット MSB

$$R = 2^{S} \times 2^{\alpha - 126} \times (1 + 2^{-1} \times \beta_{22} + 2^{-2} \times 21 + \cdots\cdots + 2^{-23} \times \beta_{0})$$

ここで，βi とはβの i (0 ≤ i ≤ 22) 番目の 1 ビットの値です。S は符号ビットです。

(c) 特異表現 (Denormalized Format)

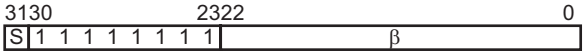


ここでβはフィールド長が 23 ビットの仮数です。この表現は正規表現では表現できない小さい実数を表現するときに用います。

この時の R の値は，次の式で表現できます。

$$R = 2^{S} \times 2^{-126} \times (2^{-1} \times \beta_{22} + 2^{-2} \times 21 + \cdots\cdots + 2^{-23} \times \beta_{0})$$

(d) 無限大



ここでβはフィールド長が 23 ビットの仮数です。ただし本アプリケーションノートでは，指数部がすべて 1 の場合，次のように扱います。

S = 0 の場合：正の無限大

$R = +\infty$

S = 1 の場合：負の無限大

$R = -\infty$

(2) 内部表現の例

S = B'0 (2 進)
α = B'10000011 (2 進)
β = B'1011100・・・・・・0 (2 進)

としたとき，これに対応した実数は次のようになります。

$R = 20 \times 2131 - 126 \times (1 + 2 - 1 + 2 - 3 + 2 - 4 + 2 - 5)$
$= 16 + 8 + 2 + 1 + 0.5 = 27.5$

(a) 最大値と最小値

ここでは絶対値としての最大，最小値について述べます。最大値を R_{MAX}，最小値を R_{MIN} とします。以下の数値まで表わすことができます。

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
2.00	2006.02.28	-	日立版からルネサス版へフォーマット変更
3.00	2006.06.12	4, 6	誤記修正

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。