RX ファミリ
積和演算命令の活用方法

R01AN0254JJ0100
Rev.1.00
2011.06.17

要旨

この文書は、RX ファミリの DSP 機能命令の使用方法を説明します。

動作確認デバイス

RX ファミリ

目次

1. はじめに

RX ファミリ CPU コア（以下 RX と略）は 16 ビット×16 ビットの積和演算器を搭載しています。乗算を用いた演算式やアドレス計算で通常に用いる 32 ビット×32 ビットの整数乗算命令（MUL 命令）は、32 ビット×32 ビットの演算結果 64 ビットのうち下位 32 ビットをその演算結果とします。つまり、MUL 命令の使用にあたっては、演算結果が 32 ビットを超えないという前提があります。ところが、数値データを固定小数点表現（例えば、[1]を参照ください）で表す場合、乗算あるいは積和演算の結果のうち、有効なデータは上位側に位置するのが普通です。そのため、固定小数点表現を用いた数値データの乗算あるいは積和演算の場合に MUL 命令を用いていたのでは、演算結果が 32 ビット以内に納まる場合しか用いることができず、狭い範囲の数値データしか表現できなくなるという問題が発生します。この問題を解決するために、RX は、1 サイクルで実行可能な 48 ビットのアキュムレータによる積和演算命令（または乗算命令）、アキュムレータに格納された値の丸め演算を実行する命令、およびアキュムレータと汎用レジスタ間のデータの転送命令をサポートしています。これらの積和演算命令や丸め命令等を組み合わせることで、固定小数点表現を用いた数値データの種々の演算を高速に実現でき、DSP に匹敵するデータ処理能力を実現することができます。以下ではこれらの命令の使い方について、実例を交えながら説明します。なお、以下では積和演算命令等の説明を記載しますが、詳細は「RX ファミリ　ソフトウェアマニュアル」を参照ください。

[1] 森、名取、鳥居; "岩波講座　情報科学-18　数値計算", pp.1-27, 岩波書店, (1982)

2. DSP 機能命令の種類

RX では CPU 内蔵の 16 ビット×16 ビットの積和演算器を活用するための DSP 機能命令があります。これらを活用することで 16 ビット×16 ビットの積和演算を効率的に実現できます。これらの命令では 2 つの汎用レジスタの値を入力とし、演算結果はアキュムレータに格納されます。DSP 機能演算命令には、以下に示す 4 つの命令があります。

(1) MULHI src, src2

(2) MULLO src, src2

(3) MACHI src, src2

(4) MACLO src, src2

ここで、ニーモニックの先頭 3 文字の"MUL"は乗算を、つまり乗算結果をアキュムレータに格納することを示します。また、先頭 3 文字の"MAC"は積和演算を、つまり乗算結果とアキュムレータ値を加算して、その結果をアキュムレータに格納することを示します。一方、ニーモニックの後ろの 2 文字の"HI"は、ニーモニックの第一オペランドで示されたレジスタ値の上位（"HIgh"側）16 ビットと、第二オペランドで示されたレジスタ値の上位 16 ビットを被乗数として、乗算を実行することを示します。つまり、16 ビット×16 ビットの乗算を実行することを示します。また、ニーモニックの後ろの 2 文字の"LO"は、ニーモニックの第一オペランドで示されたレジスタ値の下位 16 ビットと、第二オペランドで示されたレジスタ値の下位（"LOw"側）16 ビットを被乗数として、乗算を実行することを示します。図 1 に上記 4 つの命令の機能を示します。

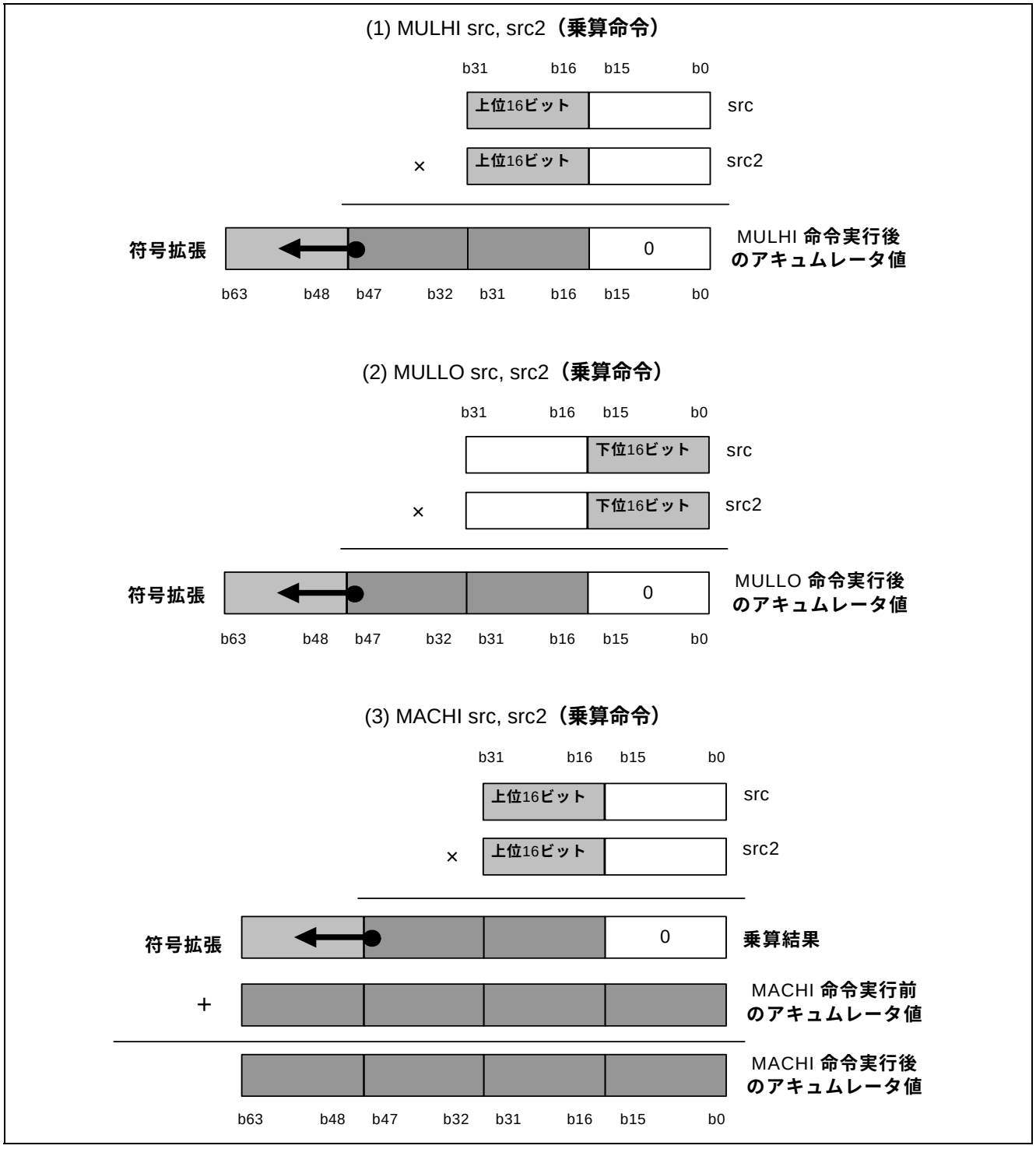


図 1　16 ビット×16 ビットの積和演算命令の機能

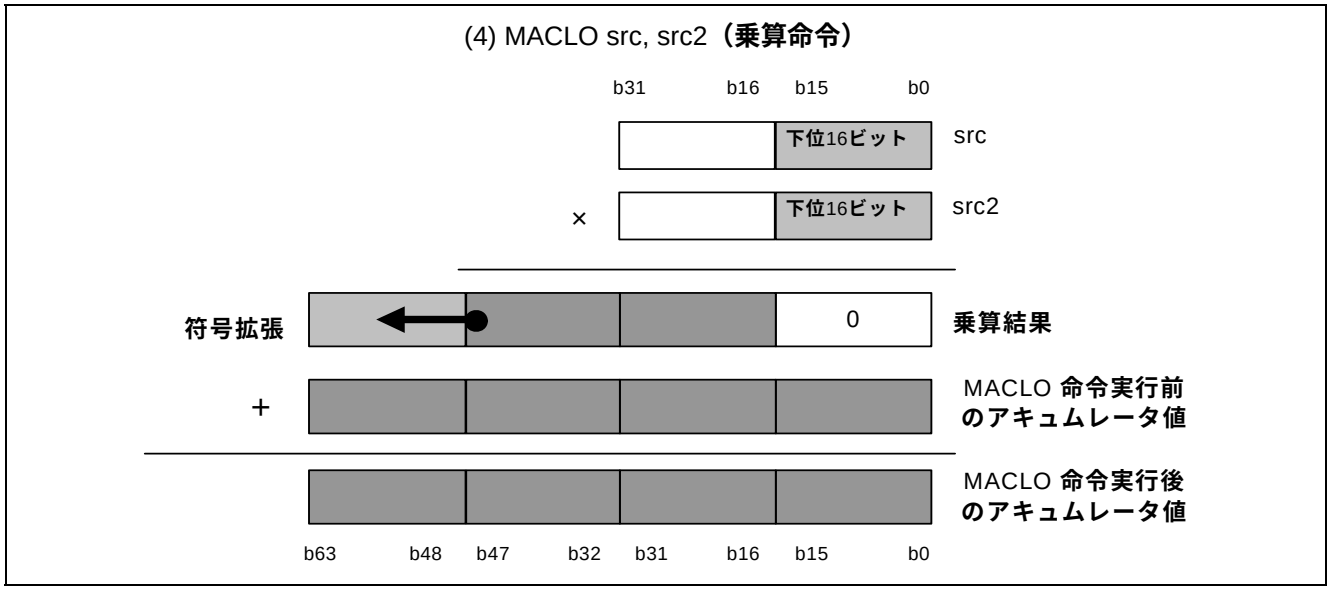


図 1　16 ビット×16 ビットの積和演算命令の機能（続き）

3. 積和演算命令とデータのパッキング

RX の搭載する積和演算命令は 16 ビットのワードデータを使用することになります。一方で、RX の汎用レジスタのビット長は 32 ビットです。そのため、16 ビットのワードデータを転送命令で汎用レジスタに転送して、あらかじめ汎用レジスタに格納されたもう一方のデータと乗算する場合を考えてみます。以下では式

x*a+y*b

を用いて、積和演算命令の使い方とデータのパッキングについて説明します。

RX の汎用レジスタ上位 16 ビットと汎用レジスタ上位 16 ビットを用いて乗算あるいは積和する積和演算命令を用いることで、使用されていない汎用レジスタ上位 16 ビットを活用できないかを検討します。16 ビットデータ x と y を 32 ビットデータの上位 16 ビットと下位 16 ビットに、16 ビットデータ a と b を 32 ビットデータの上位 16 ビットと下位 16 ビットにそれぞれ配置したデータを準備し、この 32 ビットデータをそれぞれ異なる汎用レジスタに転送すれば、x*a の部分式の計算に MULHI 命令を使用できます。このように 2 つの 16 ビットデータを 1 つの 32 ビットデータとして格納することをデータのパッキングと呼びます。これにより、使用されていない汎用レジスタ上位 16 ビットを有効に使用することができるようになります。

またデータをパッキングすることでデータのレジスタへの転送に 32 ビットのロングワードデータの転送命令を使用することでき、これにより 2 回のワードデータの転送が 1 回で済むようになります。この方法を用いれば、上で 6 命令で実現した（x*a+y*b）の式の計算が下に示すような 4 命令で実行できるようになります。

```
mov.l  [Rs_x], Rsrc1   ; x,y の転送
mov.l  [Rs_a], Rsrc2   ; a,b の転送
mullo  Rsrc1, Rsrc2   ;  x*a （積和演算命令）
maclo  Rsrc1, Rsrc2   ;  +y*b（積和演算命令）
```

つまり、16 ビットデータ x, y と a, b をそれぞれ 32 ビットデータの上位と下位に設定し、32 ビットデータの転送命令を使用することで、転送命令が 2 命令不要になります。16 ビット×16 ビットの積和演算命令を使用した積和演算のフローを図 2 に示します。16 ビット×16 ビットの積和演算命令を使用する際に注意すべきことは、RX に用意された積和演算命令を使用できるように、2 つの被乗数を汎用レジスタの上位同士、あるいは下位同士にそろえることです。なお、エンディアンによっては、x, y と a, b のレジスタでの格納位置は異なりますのでご注意ください。

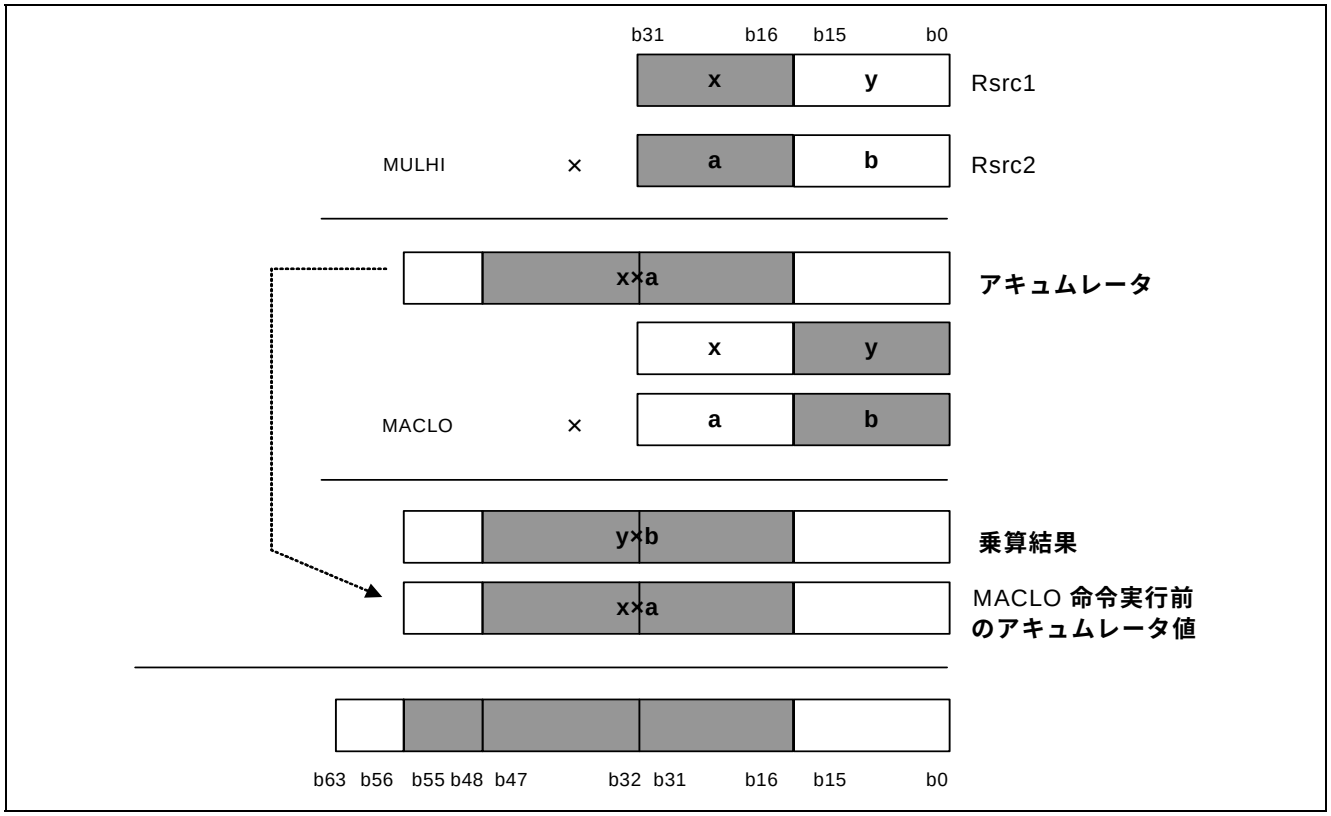


図 2　積和演算命令を使用した積和演算フロー

4. 丸め処理

前節で見たように、積和演算結果はアキュムレータに格納されます。この演算結果をアキュムレータから汎用レジスタに転送する命令 MVFACHI（アキュムレータ上位 32 ビットからの転送）あるいは MVFACMI（アキュムレータ中央 32 ビットからの転送）を用いて必要な部分を取り出し活用することになります。計算の仕方によっては積和演算結果に丸め処理（四捨五入）を実施したい場合があります。このような用途のために RX では RACW（16 ビット符号付きアキュムレータ丸め処理）が準備されています。RACW 命令を用いることで、アキュムレータ内で図 3 に示すような丸め処理が行われ、丸め処理の結果を MVFACHI 命令を用いて汎用レジスタに取り出すことができます。

丸め処理を使用する際は、RACW 命令がどうような丸め処理を行うかをよく理解した上で使用する必要があります。つまり、実施したい計算内容と RACW 命令の行う丸め処理が合致している必要があります。図 4 に RACW 命令の処理内容を示します。

- アキュムレータの値に対してワードサイズで丸めを行い、その結果をアキュムレータに格納します。
  以下に動作概要図を示します

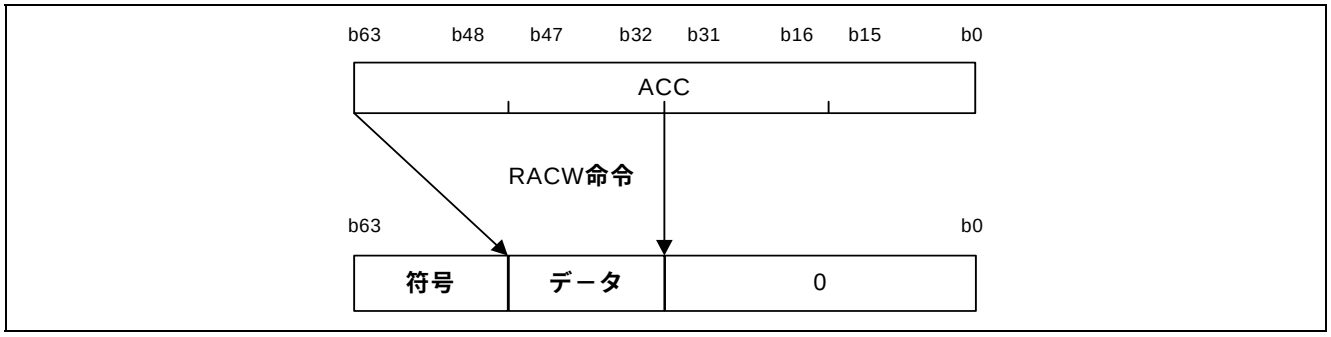


図 3　RACW 命令の処理概要

- RACW 命令は、以下のような手順で実行されます。
  処理 1.　アキュムレータの値を、src で指定したビット数分（1 ビットまたは 2 ビット）、左シフトします。

図 4　RACW 命令における丸め処理

処理 2.　1 ビットまたは 2 ビットの左シフトを行った 64 ビットの値に従って、アキュムレータの値が変化します。

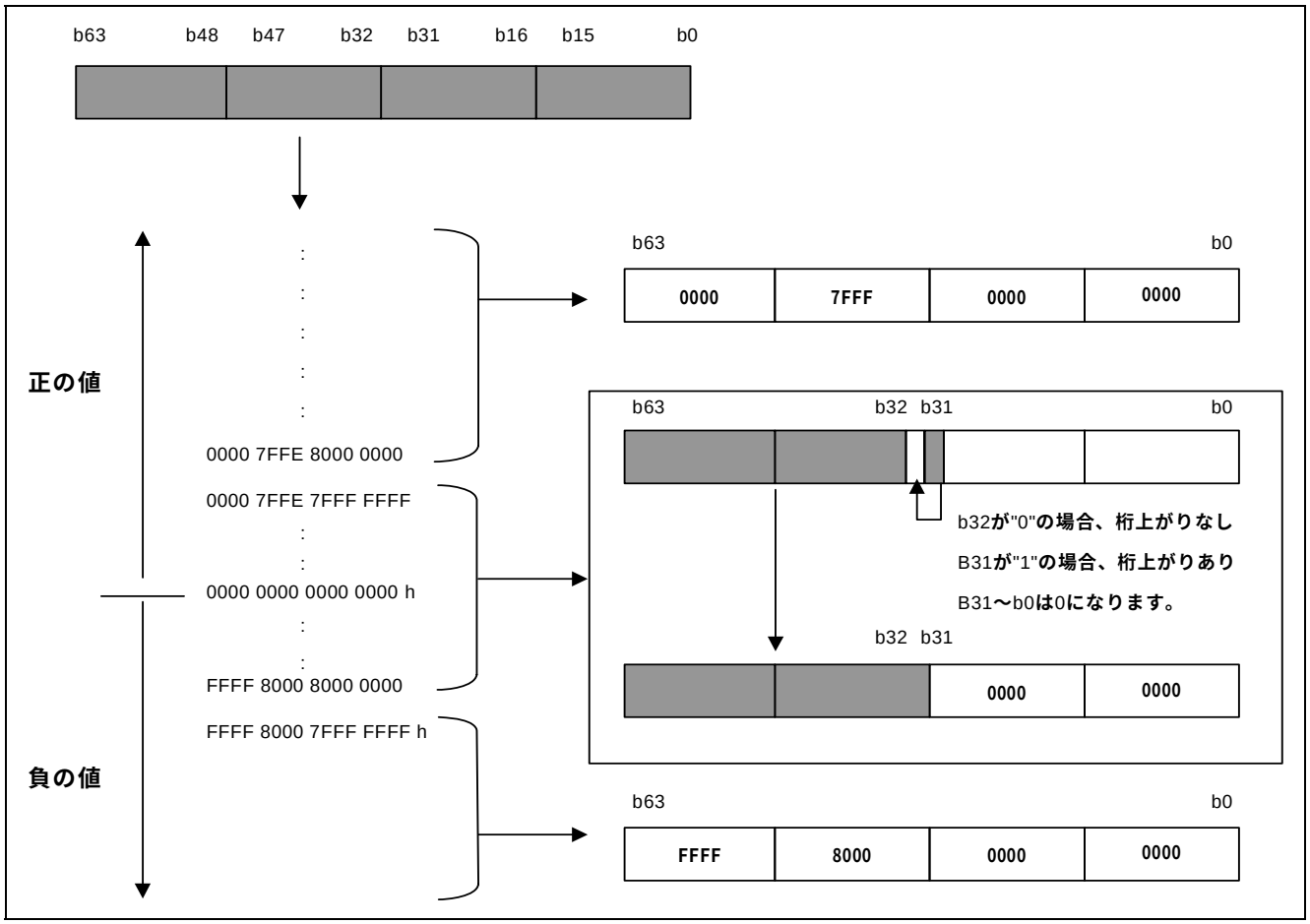


図 4　RACW 命令における丸め処理（続き）

RACW　src

と記述した場合、アキュムレータの値を、src で指定したビット数分（1 ビットまたは 2 ビット）、左シフトした後、b31 の値（0 か 1 か）で丸めを行うところがポイントです。

5. 固定小数点表現と丸め処理

RX の丸め命令"RACW"は、src で指定したビット数分（1 ビットまたは 2 ビット）、アキュムレータを左シフトした後にアキュムレータの bit31 の値に応じて丸め処理を行います（図 4 参照）。つまり丸め命令を使用する際には、積和演算命令の結果であるアキュムレータの値は bit31 と bit30 の間（RACW 命令の src に 1 を指定した場合）、あるいは bit30 と bit29 の間（RACW 命令の src に 2 を指定した場合）に小数点がある固定小数点表現をとった値である必要があります。したがって、アキュムレータの bit30 から bit16 までの 15 ビット（RACW 命令の src に 1 を指定した場合）、あるいは bit29 から bit16 までの 14 ビット（RACW 命令の src に 2 を指定した場合）が小数点以下の桁と見なせます。16 ビット×16 ビットの積和演算命令の場合、乗算結果をアキュムレータの bit47 から bit16 に格納するため、丸め演算を使用するためには、乗算結果の下位 15 ビットあるいは 14 ビットが小数点部分となるように被乗数の小数点を考える必要があります。演算を使用するための被乗数の表現例を図 5 に示します。具体例は 7 章で示します。

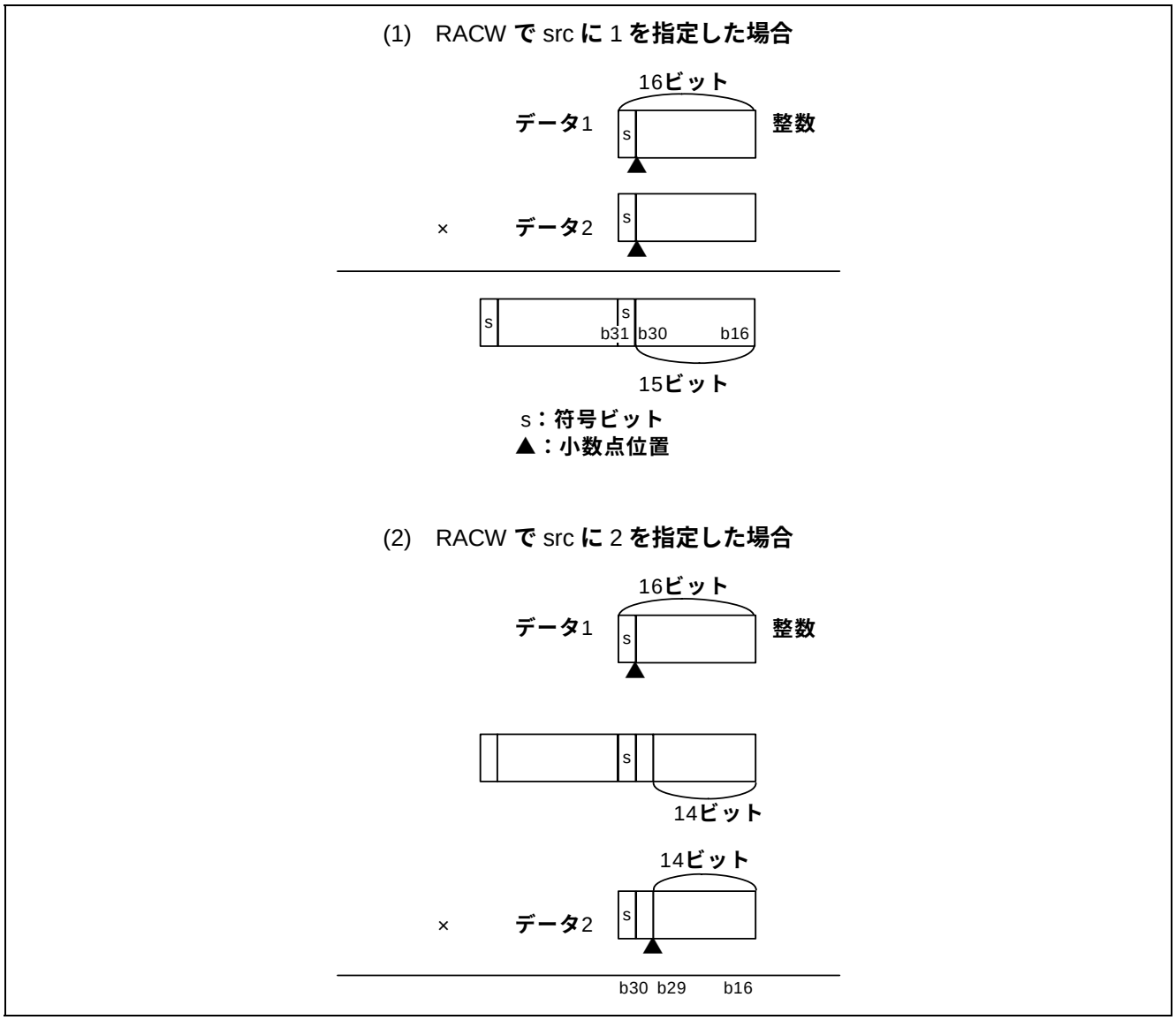


図 5　丸め処理を使用するための被乗数の表現例

6. DSP 機能命令を使用した場合の処理フロー

本章では、DSP 機能命令を使用した基本的な場合の処理フローについて説明します。具体的には、固定小数点表現で表されたデータを用いて積和演算を実行し、その後丸め処理を実行、アキュムレータから汎用レジスタへ所望のデータを転送するまでの一連の処理について説明します。

6.1 積和演算命令に用いる命令群

積和演算命令を用いる場合の一連の処理、つまり、積和演算、丸め演算、アキュムレータから汎用レジスタへのデータ転送では、下に示す命令群を一組として用います。

- MULHI（乗算命令）
- MULLO（乗算命令）
- MACHI（積和命令）
- MACLO（積和命令）
- RACW（アキュムレータの丸め命令）
- MVFACHI（アキュムレータから汎用レジスタへの転送命令）
- MVFACMI（アキュムレータから汎用レジスタへの転送命令）
- MVTACHI（汎用レジスタからアキュムレータへの転送命令）
- MVTACLO（汎用レジスタからアキュムレータへの転送命令）

6.2　演算の流れ

積和演算命令を用いる場合、以下の手順で演算を進めます。

(1) アキュムレータの初期化
(2) 積和演算
(3) 丸め処理
(4) アキュムレータから汎用レジスタへの演算結果の転送

図 6 に典型的な積和演算の流れを示します。図 6 の(1)では MULHI を用いて初めに乗算を行っています。これにより乗算結果がアキュムレータに設定されます。これによりアキュムレータが初期化されます。もし、アキュムレータの初期値を乗算結果ではなく、特定の値に設定したい場合は汎用レジスタの内容をアキュムレータに転送する MVTACHI 命令と MVTACLO 命令を使用してアキュムレータの値を初期化する必要があります。(2)の積和演算は図では 1 度しか実行していませんが、複数回の実行ももちろん可能です。その後、図では小数点位置に応じて丸め処理を実行する場合の流れを説明しています。丸め処理を実行する必要がない場合は、(3)の処理は不要で、アキュムレータから必要なデータを汎用レジスタに転送します。丸め処理を行わない場合は、アキュムレータの上位 32 ビットが必要なのか、アキュムレータ中央 32 ビットが必要なのかに応じて MVFACHI 命令か MVFACMI 命令を使い分ける必要があることを留意してください。

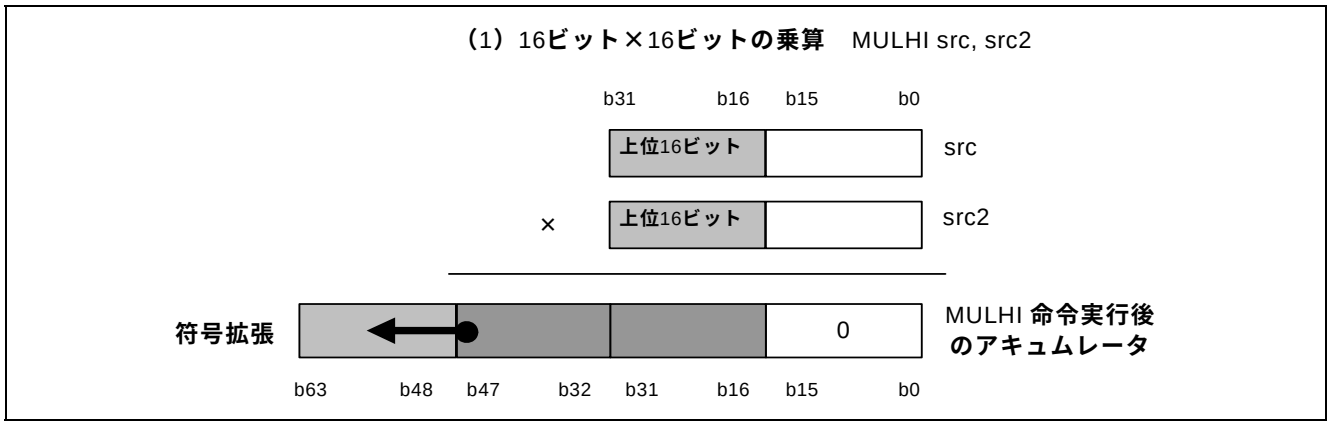


図 6　積和演算命令を用いる場合の演算の流れ

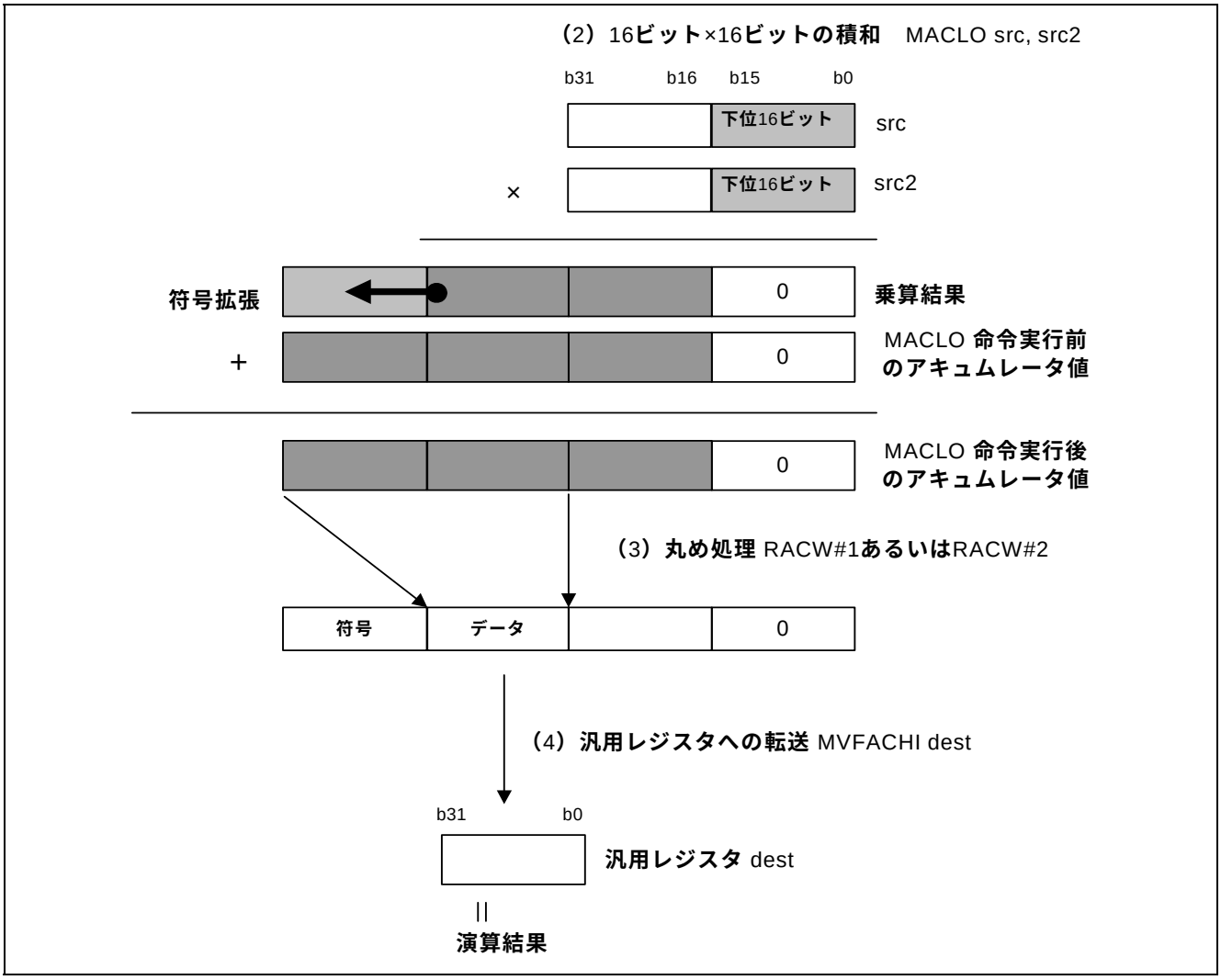


図 6　積和演算命令を用いる場合の演算の流れ（続き）

7. サンプルプログラム

積和演算命令を用いたプログラム例を行列の掛け算を用いて説明します。

7.1 3 行 4 列の行列と 4 行 1 列の行列の掛け算

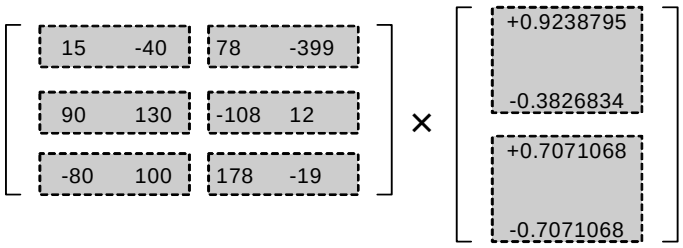
以下の 3 行 4 列の行列と 4 行 1 列の行列の掛け算を行うプログラムを考えることにします。

$$\begin{pmatrix} 16 & -40 & 78 & -399 \\ 90 & 130 & -108 & 12 \\ -80 & 100 & 178 & -19 \end{pmatrix} \times \begin{pmatrix} +0.9238795 \\ -0.3826834 \\ +0.7071068 \\ -0.7071068 \end{pmatrix}$$

行列の要素をワードデータとして扱うため、4 行 1 列の要素を固定小数点で表す必要があります。今回のデータは-1.0 より大きく 1.0 より小さい範囲の値ですので、16 ビットデータの bit15 と bit14 の間に小数点があると考え、bit14 から bit0 までを小数点以下の桁と見なすデータ表現を採用します。そしてこのデータ表現にするためには、浮動小数点数に 2^{15} の値を乗じてワードデータの固定小数点データとします。たとえば、+0.9238795 は $0.9238795 \times 2^{15}=30274$、つまり 16 進表示で 0x7642 となります。このようにすべてのデータをワードデータとすることで積和命令を使用する準備が整います。

7.2 データのパッキング

今回の行列の掛け算では、行×列のひとまとまりの計算単位がワードデータ 4 つです。つまり、ワードデータの 4 回の積和計算をすることになります。そのため、2 つのワードデータを 32 ビット・データにパッキングすることで、データの転送回数を少なくし、かつ 32 ビットデータの上位同士、下位同士を乗算する命令を活用することができます。次の図に、上の例において 32 ビット・データにパッキングされるワードデータのペアをグレイのボックスで示します。



7.3 サンプルプログラム

7.1 節および 7.2 節で述べた考え方で作成したプログラム例を次に示します。

共用体 matrix は 3 行 4 列の行列を表しています。また、共用体 vector は 4 行 1 列の行列を表しています。typedef された dataUnit は、メンバーword を使ってワードデータでデータの初期化を行い、積和命令を使用する際にはメンバーlongWord を使ってロング・ワードデータでデータを転送するために定義したものです。

関数 matrixmultiply4 で 4 回の積和計算を実施し、その結果に丸め処理を行い、最終結果を関数の戻り値として返します。本サンプルプログラムでは　関数 matrixmultiply4 を積和命令を用いて実現しています。関数 matrixmultiply4 はアセンブリ記述関数をインライン展開しています。そのため*#pragma inline_asm* を使用しています。

```
/***********************************************************************/
/*                                                                     */
/*  FILE       :RXmatrix.c                                             */
/*  DATE       :Mon, Jul 19, 2010                                       */
/*  DESCRIPTION :Main Program                                           */
/*  CPU TYPE   :RX610                                                  */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.4.50).    */
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                     */
/*                                                                     */
/***********************************************************************/

#include <stdint.h>

//#include "typedefine.h"
#ifdef __cplusplus
//#include <ios>                        // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt;       // Remove the comment when you use ios
#endif

void main(void);
#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

#define CONST_BITS  15
#define ONE ((long int) 1)
#define CONST_SCALE (ONE << CONST_BITS)

#define MATRIX_SIZE (4)

typedef union {
  int16_t word[MATRIX_SIZE];
  int32_t longWord[MATRIX_SIZE/2];
} dataUnit;

dataUnit vector = {
  0x7642,       // +0.9238795
  0xcf05,       // -0.3826834
  0x5a82,       // +0.7071068
  0xa57f        // -0.7071068
};

#define COLUMN (3)

dataUnit matrix[COLUMN] = {
  {16, -40, 78, -399},
  {90, 130, -108, 12},
  {-80, 100, 178, -19},
};

#pragma inline_asm matrixmultiply4
int16_t matrixmultiply4(dataUnit *, dataUnit *);

void main(void)
{
```

```
  int i;
  int16_t results[COLUMN];

  for (i=0; i<COLUMN; ++i) {
   results[i] = matrixmultiply4(&matrix[i], &vector);
  }
  return;
}

int16_t matrixmultiply4(dataUnit *a, dataUnit *b)
{
   mov.l [r1+], r3   ; 2 つのワードデータの転送
   mov.l [r2+], r4   ; 2 つのワードデータの転送
   mulhi r3, r4
   maclo r3, r4
   mov.l [r1+], r3   ; 2 つのワードデータの転送
   mov.l [r2+], r4   ; 2 つのワードデータの転送
   machi r3, r4
   maclo r3, r4
   racw  #1    ; 丸め処理
   mvfachi  r1
}

#ifdef __cplusplus
void abort(void)
{

}
#endif
```

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.06.17	—	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0