# RX ファミリ
## DSP 機能命令を活用した色空間変換

R01AN0225JJ0100
Rev.1.00
2011.03.14

## 要旨

この文書は、RX ファミリの DSP 機能命令を使用した色空間変換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 ビットを越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 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。アプリケーションノート「積和演算命令の活用方法」(R01AN0254JJ) では、これらの積和演算命令や丸め命令の使い方を説明しています。以下ではこれらの命令を活用した色空間のデータの変換方法について説明します。

**【注】** [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. 色空間とは

色空間 (color space) とは、各色をいくつかの数値の組み合わせで表現する方法、あるいはいくつかの数値を組み合わせることで表現可能な色域のことです。色を定量的に表現するための体系を表色系と言います。理論上 3 つの値があれば全ての色を表現できるので、通常は 3 次元空間で表現します。つまり、基準とする 3 色の数値の組み合わせで全ての色を表現できることになります。

代表的な色空間には RGB があります。これは、赤 (Red)、緑 (Green)、青 (Blue) の光の 3 原色を利用した色空間です。コンピュータのモニタへの出力でお馴染みだと思います。他の色空間には、テレビで用いられている YCbCr/YPbPr、印刷分野で主流である CMYK などがあります。

ある色空間から別の色空間への変換は計算することで行うことができます。本アプリケーションノートでは RGB 表色系と YCbCr 表色系の相互変換に積和演算命令や丸め命令を活用する方法を示します。

**【注】 ある色空間から別の色空間への変換は計算で可能だと上では述べましたが、実際には変換先の色空間に変換元の色空間で表現していた色があるとは限らず、全ての色は変換可能だとは言えません。詳しくは、専門書を参照してください。**

## 3. RGB 表色系と YCbCr 表色系

本節では、色空間の相互変換の例に用いる RGB 表色系と YCbCr 表色系について簡単に説明します。

(1) RGB 表色系

RGB は光の三原色であり、数値を増すごとに白に近づき、数値を減らすごとに黒くなります。コンピュータのモニタで用いられるのもこの RGB です。コンピュータでは、通常 RGB 各 8 ビット、計 24 ビットを割り振ることで 1677 万 7216 色の表示を可能にしています。また、組込み機器で使用される LCD パネルでは同じく RGB 表色系を使用することが多いです。ただ、コンピュータのモニタとは異なり、RGB 合わせて 16 ビットで色を表現することが多いようです。

(2) YCbCr 表色系

輝度信号 Y と、2 つの色差信号 (Cb,Cr) を使って表現される色空間です。高画質アナログ映像信号の伝送や、デジタルビデオの記録方式として使用されています。Cb は B 信号から輝度 Y を差し引いた (B-Y) に特定の定数を掛けた値であり、Cr は R 信号から輝度 Y を差し引いた (R-Y) に特定の定数を掛けた値です。

人間の目は色の変化よりも明るさの変化に敏感なので、色差成分を減らしても不自然だと感じにくいという特性が人間の目にはあります。これを活用して、画像データのデータ量を減らす際には色差成分を間引かれます。

カラー静止画像の圧縮伸張に標準的に使用される JPEG[2][3]は YCbCr 表色系で表現された画像を処理対象としています。そのため、画像データを圧縮するためには、YCbCr 表色系で表現した画像データが必要になります。逆に、JPEG 画像を伸張した際は YCbCr 表色系で表現された画像データが得られます。

【注】 [2] JIS X 4301-1995 連続階調静止画像のディジタル圧縮及び符号処理
[3] ISO/IEC 10918-1 Information technology-Digital compression and coding of continuous-tone still images

## 4. RGB 表色系と YCbCr 表色系の相互変換

本アプリケーションノートでは、組込み機器で使用されることの多い、RGB 表色系で取り込んだデータを YCbCYr 表色系に変換して JPEG 圧縮する変換と、JPEG ファイルを伸張して得られた YCbCYr 表色系のデータを LCD に表示するための RGB 表色系への変換を取り上げます。

RGB 表色系から YCbCr 表色系への変換式 (以下では「RGB から YCbCr への変換」と呼ぶことにします) は以下のようになります。

$$Y = 0.29900 \times R + 0.58700 \times G + 0.11400 \times B$$
$$Cb = -0.16874 \times R - 0.33126 \times G + 0.50000 \times B + 128$$
$$Cr = 0.50000 \times R - 0.41869 \times G - 0.08131 \times B + 128$$

YCbCr 表色系から RGB 表色系への変換式 (以下では「YCbCr から RGB への変換」と呼ぶことにします) は以下のようになります。

$$R = Y + 1.40200 \times Cr$$
$$G = Y - 0.34414 \times Cb - 0.71414 \times Cr$$
$$B = Y + 1.77200 \times Cb$$

ただし、Cb と Cr は 128 より小さい値とする。

YCbCr から RGB への変換の際には、Cb と Cr は 128 より小さい値とする、つまり 0 から 255 の範囲の値を取る符号無しの Cb と Cr のデータからそれぞれ 128 を引いてのち、上の式に代入して計算する必要があります。

# 5. サンプルプログラム

色空間変換プログラムへの積和演算命令の活用方法を説明します。

## 5.1 RGB から YCbCr への変換

変換式の 9 個の係数は、最大のデータが 0.58700 で最小のデータが–0.41869 です。すべての係数は–1.0 と 1.0 の間のデータですので、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 を小数点部分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。例えば、+0.58700 は $0.58700 \times 2^{15} = 19235$、つまり 16 進表示で 0x4B23 となります。このように全ての係数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。

図 1 に入力データとなる RGB データと出力データとなる YCbCr データをどうのようにメモリに配置するかを示します。RGB データは符号無しの 1 バイトデータで、1 画素につき R/G/B データがこの順番に格納されているものとします。RGB データから変換された YCbCr データも符号無しの 1 バイトデータで、1 画素につき Y/Cb/Cr データがこの順番に格納されているものとします。今回のデータはバイトデータが入力となるので、RGB データにつてはアプリケーションノート「積和演算命令の活用方法」 (R01AN0254JJ) で示したようなデータのパッキングはしません。一方、係数データについては事前に準備できること、9 つの係数のうち 2 つの係数 (0.50000) が同じ値であるため実際 8 つの係数を準備できればいいのでデータをパッキングしておけば 4 ロングワードで済むことからデータのパッキングをすることにします。

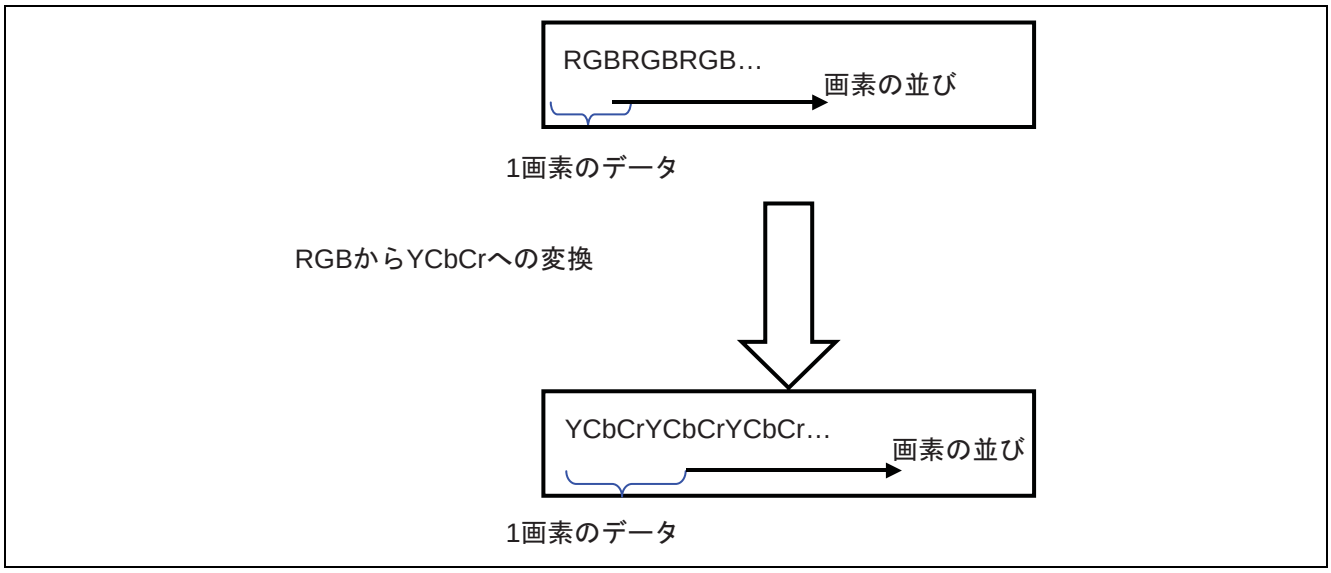


図 1　RGB データと YCbCr データのメモリ配置

あと、色空間の変換で注意すべきことは、変換の計算後に計算結果が符号無しの 1 バイトデータの範囲、つまり 0 以上 255 以下になっているかの確認が必要なことです。もし 0 より小さな値の場合は計算結果を 0 に、255 より大きな値の場合は計算結果を 255 にするという飽和演算と呼ばれる処理を行う必要があります。飽和演算を行うには RX の min/max 命令を使用すると高速に実行できます。

RGB から YCbCr への変換のサンプルプログラムを以下に示します。　関数 rgb2ycc では積和命令と丸め処理命令を用いて変換の計算を実現しています。関数 rgb2ycc 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。また、0 より小さな値の場合は計算結果を 0 に、255 より大きな値の場合は計算結果を 255 にするという飽和演算には条件分岐命令ではなく min/max 命令を使用して高速に処理できるようにしています。

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :rgb2ycc.c                                             */
/*  DATE        :Sun, Jul 25, 2010                                     */
/*  DESCRIPTION :Main Program                                          */
/*  CPU TYPE    :Other                                                 */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.4.50).    */
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                    */
/*                                                                     */
/***********************************************************************/



//#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

#include <stdio.h>
#include <stdint.h>

#define WIDTH  (64)
#define HEIGHT (56)

/* Input Image Data in RGB format */
uint8_t rgbData[3*WIDTH*HEIGHT] = {
#include "flowerbutton.h"
};

/* Output Image Data in YCbCr format */
uint8_t yccData[3*WIDTH*HEIGHT];

#define NUM_COEFF (8)
typedef union {
  int16_t word[NUM_COEFF];
  int32_t longWord[NUM_COEFF/2];
} CoeffData;

CoeffData coeffData = {
  0x4b23,    // +0.58700
  0x2646,    // +0.29900
  0x0e98,    // + 0.11400
  0xea68,    // -0.16874
  0xd59a,    // - 0.33126
  0x4000,    // + 0.50000
  0xca69,    // - 0.41869
  0xf599,    // - 0.08131
};
```

```
#pragma inline_asm rgb2ycc
void rgb2ycc(uint8_t *, uint8_t *);

void main(void)
{
  int i, j;
  uint8_t *in, *out;

  in = &rgbData[0];
  out = &yccData[0];
  for (i=0; i<HEIGHT; ++i) {
    for (j=0; j<WIDTH; ++j, in+=3, out+=3) {
       rgb2ycc(in, out);
    }
  }
  return;
}

/*
  RGB から YCｂCr への変換
 */
void rgb2ycc(uint8_t *rgb, uint8_t *ycc)
{
  push.l  r6
  push.l  r7

;  レジスタ r1:  rgb
;  レジスタ r2:  ycc

  movu.b  [r1+], r3; r
  movu.b  [r1+], r4; g
  movu.b  [r1], r5 ; b
  mov.l   #_coeffData, r1

  /*   Y = 0.29900 * R + 0.58700 * G + 0.11400 * B   */
  mov.l   [r1+], r6
  mov.l   [r1+], r7
  mullo   r6, r4
  shlr #16, r6      ; 上位 16bit にある係数を使用するために下位 16bit にシフト
  maclo   r6, r3
  maclo   r7, r5
  shlr #16, r7      ; 下位 16bit に-0.16874 がくる。この係数は Cb の計算に使用する。
  racw #1
  mvfachi r6
  min     #000000FFH, r6     ; 飽和演算
  max      #00H,r6
  mov.b    r6, [r2+]

  /*   Cｂ = -0.16874 * R - 0.33126 * G + 0.50000 * B + 128   */
  mov.l   [r1+], r6
  mullo   r7, r3
  maclo   r6, r4
  shlr #16, r6      ; 下位 16bit に+0.50000 がくる。この係数は Cr の計算にも使用する。
  maclo   r6, r5
  racw #1
  mvfachi r7
  add  #128, r7
  min     #000000FFH, r7 ; 飽和演算
```

```
  max     #00H, r7
  mov.b   r7, [r2+]
  mov.l   [r1+], r7

  /*  Cr = 0.50000 * R - 0.41869 * G - 0.08131 * B + 128  */
  mullo   r6, r3
  maclo   r7, r5
  shlr#16, r7
  maclo   r7, r4
  racw#1
  mvfachi r7
  add     #128, r7
  min     #000000FFH, r7     ; 飽和演算
  max     #00H, r7
  mov.b   r7, [r2]

    pop  r7
  pop  r6
}

#ifdef __cplusplus
void abort(void)
{

}
#endif
```

## 5.2　YCbCr から RGB への変換

5.1 節とは逆方向の変換を行います。要領は同じです。変換式の係数は 4 個で、最大の係数値は+1.77200 です。最大の係数は 1.0 と 2.0 の間のデータで、16 ビットデータのbit14 とbit13 の間に小数点があると考え、bit13 からbit0 を小数点部分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、浮動小数点に $2^{14}$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。例えば、+1.77200 は $1.77200 \times 2^{14} = 29032$、つまり 16 進表示で 0x7168 となります。bit14 とbit13 の間に小数点があると考えるので、丸め命令はracw #2 を使用する点がポイントとなります。係数データは 4 個なので 2 つのワードデータを 32 ビット・データにパッキングします。

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :ycc2rgb.c                                             */
/*  DATE        :Sun, Jul 25, 2010                                     */
/*  DESCRIPTION :Main Program                                          */
/*  CPU TYPE    :Other                                                 */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.4.50).    */
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                    */
/*                                                                     */
/***********************************************************************/



//#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

#include <stdio.h>
#include <stdint.h>

#define WIDTH  (64)
#define HEIGHT (56)

/* Input Image Data in YCbCr format */
uint8_t yccData[3*WIDTH*HEIGHT] = {
#include "flowerbutton_ycc.h"
};

/* Output Image Data in RGB format */
uint8_t rgbData[3*WIDTH*HEIGHT];

#define NUM_COEFF (4)
typedef union {
  int16_t word[NUM_COEFF];
  int32_t longWord[NUM_COEFF/2];
} CoeffData;

CoeffData coeffData = {
  0x59ba,       // +1.40200
  0xe9fb,       // -0.34414
  0xd24d,       //  -0.71414
  0x7168,       //+1.7720
};

#pragma inline_asm ycc2rgb
void ycc2rgb(uint8_t *, uint8_t *);

void main(void)
```

```
{
  int i, j;
  uint8_t *in, *out;

  in = &yccData[0];
  out = &rgbData[0];
  for (i=0; i<HEIGHT; ++i) {
   for (j=0; j<WIDTH; ++j, in+=3, out+=3) {
      ycc2rgb(in, out);
   }
  }
}

/*
  YCbCr から RGB への変換
 */
void ycc2rgb(uint8_t *ycc, uint8_t *rgb)
{
  push.l  r6
  push.l  r7

;  レジスタ r1:  ycc
;  レジスタ r2:  rgb

  movu.b  [r1+], r3; y
  movu.b  [r1+], r4; cb
  movu.b  [r1], r5 ; cr
  add  #-128, r4
  add  #-128, r5
  mov.l   #_coeffData, r1

  /*   R = Y + 1.40200 * Cr  */
  mov.l   [r1+], r6
  mullo   r6, r5
  shlr #16, r6      ; 上位 16bit にある係数を使用するために下位 16bit にシフト
  racw #2
  mvfachi r7
  add  r3, r7
  min     #000000FFH, r7     ; 飽和演算
  max      #00H,r7
  mov.b    r7, [r2+]

  /*   G = Y - 0.34414 * Cb - 0.71414 * Cr  */
  mov.l   [r1+], r7
  mullo   r6, r4
  maclo   r7, r5
  shlr #16, r7      ; 上位 16bit にある係数を使用するために下位 16bit にシフト
  racw #2
  mvfachi r6
  add  r3, r6
  min     #000000FFH, r6 ; 飽和演算
  max      #00H, r6
  mov.b    r6, [r2+]

  /*   B = Y + 1.77200 * Cb  */
  mullo   r7, r4
  racw #2
  mvfachi r6
```

```
  add  r3, r6
  min     #000000FFH, r6     ; 飽和演算
  max      #00H, r6
  mov.b    r6, [r2]

  pop  r7
  pop  r6
}

#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 | 2011.03.14 | — | 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