RX Family

Color Space Conversion Using the DSP Instructions

R01AN0225EJ0100
Rev.1.00
Mar 14, 2011

Introduction

The purpose of this document is to explain how to use the color space conversion feature using the DSP instructions of the RX family microcomputers.

Target Device

RX Family

Contents

1. General

The RX family CPU core (hereafter referred to as RX) incorporates a 16 × 16-bit multiply-accumulator. The result of executing a typical 32 × 32-bit integer multiplication instruction (MUL instruction) that is used for multiplicative expressions or address calculations is given by the lower 32 bits of the 64-bit result of multiplying two 32-bit numbers. Accordingly, it is assumed that the result of using an MUL instruction does not exceed 32 bits. However, when a numerical value is expressed as a fixed-point number (For example, refer to [1].), it is common that the valid data of the result of a multiplication or a multiply-accumulation is assigned to the upper bits. Therefore, if a multiplication or a multiply-accumulation of fixed-point numbers is carried out using a MUL instruction, the result must be within 32 bits and only a very limited range of numerical values can be dealt with. To solve this problem, the RX supports the instructions to perform the following: multiply-accumulation (or multiplication) by a 48-bit accumulator, rounding operation of the value stored in an accumulator, and data transfer between an accumulator and a general-purpose register. The combination of these multiply-accumulation and rounding operation instructions allows several high-speed operations on fixed-point numbers and data processing performance equal to DSPs. The application note "How to Use Multiply-Accumulation Instruction" (R01AN0254EJ) explains how to use these multiply-accumulation instructions and rounding operation instructions. This document explains the procedure for converting color space data using these instructions.

Note: [1] Mori, Natori, Torii, Iwanami Course: Information Science Vol. 18, Numeric Computation, pp.1-27, Iwanamishoten, (1982)

2. What is a Color Space?

A color space is a method of representing each color in terms of a combination of several numeric values or a color gamut that can be represented by combining several numeric values. A system for representing colors quantitatively is called a color system. Since all the colors can theoretically be represented by three values, a color system is usually represented by a three-dimensional space. That is, all colors can be represented by combining the numeric values of three reference colors.

One of the typical color spaces is RGB. This is a color space that makes use of the light's three primary colors of red, green, and blue. You will be familiar with this color space on a computer monitor. The other color spaces include YCbCr/YPbPr which are used in TV broadcasting and CMYK which is predominant in the printing field.

Conversion of one color space to another color space can be accomplished by performing computations. This application note introduces a method of performing mutual conversion between the RGB and YCbCr color systems using multiply-accumulation and rounding operation instructions.

Note: It is mentioned above that conversion from one color space to another is possible with computation. In practice, however, colors that are represented in the source color space do not always exist in the destination color space; it cannot be said that all colors can be converted. Refer to technical books for details.

3. RGB Color System and YCbCr Color System

This chapter provides a brief description of the RGB and YCbCr color systems which are used in the examples of mutual color space conversions contained in this document.

(1) RGB color system

RGB are light's three primary colors in which the color moves toward white as their numeric values get greater and toward black as their values get smaller. It is RGB that is adopted for computer displays. On a computer, each of RGB is normally assigned 8 bits, totaling in 24 bits, to provide the computer with the capability to represent 16777216 colors. The same RGB color system is often used on the LCD panel which is used in embedded devices. Unlike the computer monitor, however, many models of LCD panels seemingly use a total of 16 bits of information to represent all RGB colors.

(2) YCbCr color system

The YCbCr color system is a color space that is represented by the brightness signal Y and two color difference signals (Cb, Cr). It is used to transmit high image quality analog video signals and used as a recording system for digital videos. Cb is the results of subtracting the level of the brightness Y from the level of the B signal (B-Y) multiplied by a specific constant whereas Cr is the results of subtracting the level of the brightness Y from the level of the R signal (R-Y) multiplied by a specific constant.

Since the human eyes are more sensitive to the changes in brightness than to the changes in colors, they have characteristics of hardly feeling unnatural when the color difference components are reduced. By taking advantage of this, the color difference components are thinned out to reduce the volume of image data.

JPEG[2][3] which is normally used in compression and extension of color still pictures is intended to process images that are represented in the YCbCr color system. Consequently, to compress image data, image data that is represented in the YCbCr color system is required. Conversely, image data that is represented in the YCbCr color system is obtained when a JPEG image is extended.

Notes: [2] JIS X 4301-1995 Digital compression and coding of continuous-tone still images
[3] ISO/IEC 10918-1 Information technology-Digital compression and coding of continuous-tone still images

4. Interconversion between the RGB Color System and YCbCr Color System

This application note discusses the conversion of data in the RGB color system which is frequently employed in embedded devices to the YCbCYr color system for JPEG compression and the conversion of YCbCYr color system data which is obtained by extending a JPEG file to the RGB color system for display on an LCD.

The equations for the conversion from RGB color system to YCbCr color system (called "conversion from RGB to YCbCr" in the rest of this document) are given below.

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

The equations for the conversion from YCbCr color system to RGB color system (called "conversion from YCbCr to RGB" in the rest of this document) are given below.

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

where the values of Cb and Cr are assumed to be smaller than 128.

The values of Cb and Cr are assumed to be smaller than 128 in the conversion from YCbCr to RGB. In other words, it is necessary to subtract 128 from the unsigned Cb and Cr data, respectively, that take on a value from 0 to 255, then substitute the results into the above equations during computation.

5. Sample Programs

This chapter explains how to use the multiply-accumulation instructions in color space conversion programs.

5.1 Conversion from RGB to YCbCr

The maximum value of the nine coefficients in the conversion equations is 0.58700 and the minimum value is -0.41869. Since all the coefficients take on some values between -1.0 and 1.0, we adopt a data representation in which the bits from bit positions bit14 to bit0 of 16-bit data make up the fractional part, assuming a decimal point between bit15 and bit14. To implement this data representation, we multiply the floating-point number by the value of 2^{15} to yield fixed-point word data. For example, +0.58700 leads to $0.58700 \times 2^{15} = 19235$, i.e., 0x4B23 in hexadecimal notation. In this way, by representing all coefficients with word data, we will be all set for using the multiply-accumulation instructions

Figure 1 shows how to place the RGB data to be handled as input data and the YCbCr data to be handled as output data in memory. It is assumed that the RGB data is 1-byte unsigned data and that, for each pixel, R/G/B data is stored in memory in that order. The YCbCr data which is derived from RGB data is also 1-byte unsigned data and Y/Cb/Cr data is stored in memory in that order for each pixel. Since byte data serves as input in this program, no data packing is carried out on the RGB data as explained in the application note "How to Use Multiply-accumulation Instruction" (XXX). On the other hand, it is decided that the coefficient data is subjected to data packing because 1) they can be prepared in advance, 2) two coefficients out of the nine coefficients have the same value (0.50000) and therefore no more than eight coefficients need be prepared, and 3) only four long words will serve the purpose if data is packed.

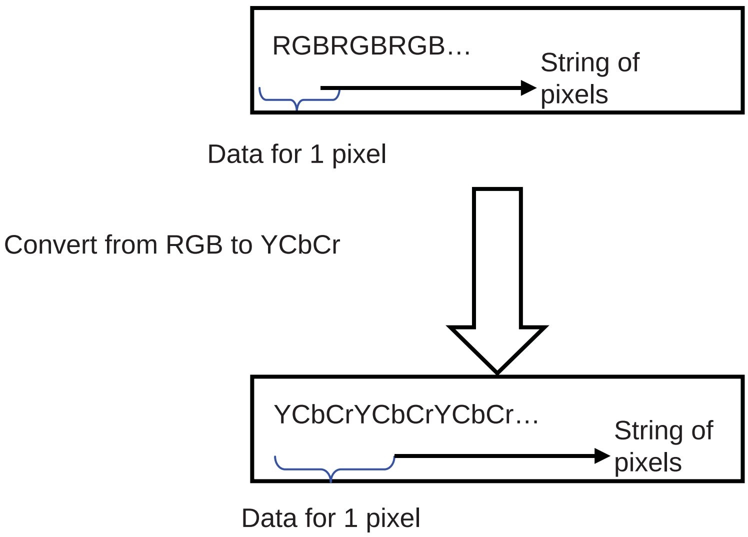

Figure 1 Placement of RGB and YCbCr Data in Memory

Another point to notice with respect to color space conversion is that it is necessary to verify that the results of conversion computation fall within the value range of 1-byte unsigned data, i.e., greater than or equal to 0 and not greater than 255. Processing called saturate calculation need be carried out in which the calculation results are set to 0 for any values smaller than 0 and to 255 for any values greater than 255. The saturate calculation can be executed at high speed using the RX's min/max instructions.

A sample program for conversion from RGB to YCbCr is given below. The function rgb2ycc carries out the conversion computation using the multiply-accumulation and rounding operation instructions. The function rgb2ycc is an inline expansion of an assembly coded function. Consequently, #pragma inline_asm is used. For the saturate calculation in which the calculation results are set to 0 for any values smaller than 0 and to 255 for any values greater than 255, not the conditional branch instructions but the min/max instructions are used for high-speed execution.

```
/***********************************************************************/
/*                                                                     */
/*  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;
}

/*
  Converts from RGB to YCbCr
 */
void rgb2ycc(uint8_t *rgb, uint8_t *ycc)
{
  push.l r6
  push.l r7

; register r1: rgb
; register 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     ; Shift into lower 16 bits to use coefficient in upper 16
bits
  maclo  r6, r3
  maclo  r7, r5
  shlr #16, r7     ; -0.16874 is placed in lower 16 bits. This coefficient is
used to compute Cb.
  racw #1
  mvfachi r6
  min    #000000FFH, r6    ; Saturate calculation
  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     ; +0.50000 is placed in lower 16 bits. This coefficient is
used to compute Cr.
  maclo  r6, r5
  racw #1
```

```
    mvfachi r7
    add  #128, r7
    min     #000000FFH, r7 ; Saturate calculation
    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     ; Saturate calculation
    max      #00H, r7
    mov.b    r7, [r2]

       pop   r7
    pop  r6
  }

  #ifdef __cplusplus
  void abort(void)
  {

  }
  #endif
```

5.2 Conversion from YCbCr to RGB

The sample program introduced in this section performs conversion in the direction opposite to that which is described in section 5.1. The procedure is similar. There are four coefficients used in the conversion equations. The maximum coefficient value is +1.77200. The value of the maximum coefficient falls between 1.0 and 2.0. Accordingly, a data representation in which the bits from bit positions bit13 to bit0 of 16-bit data make up the fractional part, assuming a decimal point between bit14 and bit13. To implement this data representation, we multiply the floating-point number by the value of 2^{14} to yield fixed-point word data. For example, +1.77200 leads to 1.77200×2^{14}=29032, i.e., 0x7168 in hexadecimal notation. Since the decimal point is considered to lie between bit14 and bit13, using the rounding operation instruction in the form of racw #2 will become the key point. Since the number of coefficients used is 4, two pieces of word data are packed into 32-bit data.

```
/***********************************************************************/
/*                                                                     */
/*  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);
   }
  }
}

/*
  Converts from YCbCr to RGB
 */
void ycc2rgb(uint8_t *ycc, uint8_t *rgb)
{
  push.l  r6
  push.l  r7

; register r1: ycc
; register 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     ; Shift into lower 16 bits to use coefficient in upper 16
bits
  racw #2
  mvfachi r7
  add  r3, r7
  min     #000000FFH, r7     ; Saturate calculation
  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     ; Shift into lower 16 bits to use coefficient in upper 16
bits
  racw #2
  mvfachi r6
  add  r3, r6
  min     #000000FFH, r6 ; Saturate calculation
  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    ; Saturate calculation
    max      #00H, r6
    mov.b    r6, [r2]

    pop  r7
    pop  r6
}

#ifdef __cplusplus
void abort(void)
{

}
#endif
```

Website and Support

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

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

Revision Record

Rev.	Date	Description: Page	Description: Summary
1.00	Mar 14, 2011	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1