

RX Family

R01AN0228EJ0100

Rev.1.00

Mar 14, 2011

Image Filter Programs Leveraging a DSP Instruction

Introduction

This document describes several examples of image filter programs which utilize the RX family's DSP instruction.

Target Device

RX Family

Contents

1. General.....	2
2. Image Filters.....	2
3. Image Filter Programs.....	7
4. Sample Programs.....	11

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. For details on the RX's multiply-accumulation instruction, refer to "RX Family User's Manual; Software" (REJ09B0435)". The application note "How to Use Multiply-Accumulation Instruction" (R01AN0254EJ) explains how to use these multiply-accumulation and rounding operation instructions. In addition, the application note "How to Use Intrinsic Functions for Multiply-Accumulation" (R01AN0255EJ) explains how to use these multiply-accumulation and rounding operation instructions through intrinsic functions that are extended functions of the RX Family C/C++ compiler (hereafter referred to as compiler).

The following are explanations of image filter programs which utilize the multiply-accumulation instruction supported by the RX family (for details on the theoretical aspects of an image filter, refer to text [2] below and other related documents). Note that the sample programs use the multiply-accumulation instruction through compiler intrinsic functions (intrinsic functions supporting the multiply-accumulation instruction are available at compiler version 1.01 or later).

Notes: [1] Mori, Natori and Torii; "Iwanami Koza Computer Science-18 Numerical Calculation," pp.1-27, Iwanami Shoten, (1982)

[2] R. C. Gonzalez et al.; "Digital Image Processing", 3rd edition, Pearson Education International, (2008)

2. Image Filters

This application note deals with image filters which multiply a total number of nine pixels by 3×3 filter mask coefficients. These pixels include pixel $f(x, y)$ and its surrounding pixels in image f as shown in figure 1.

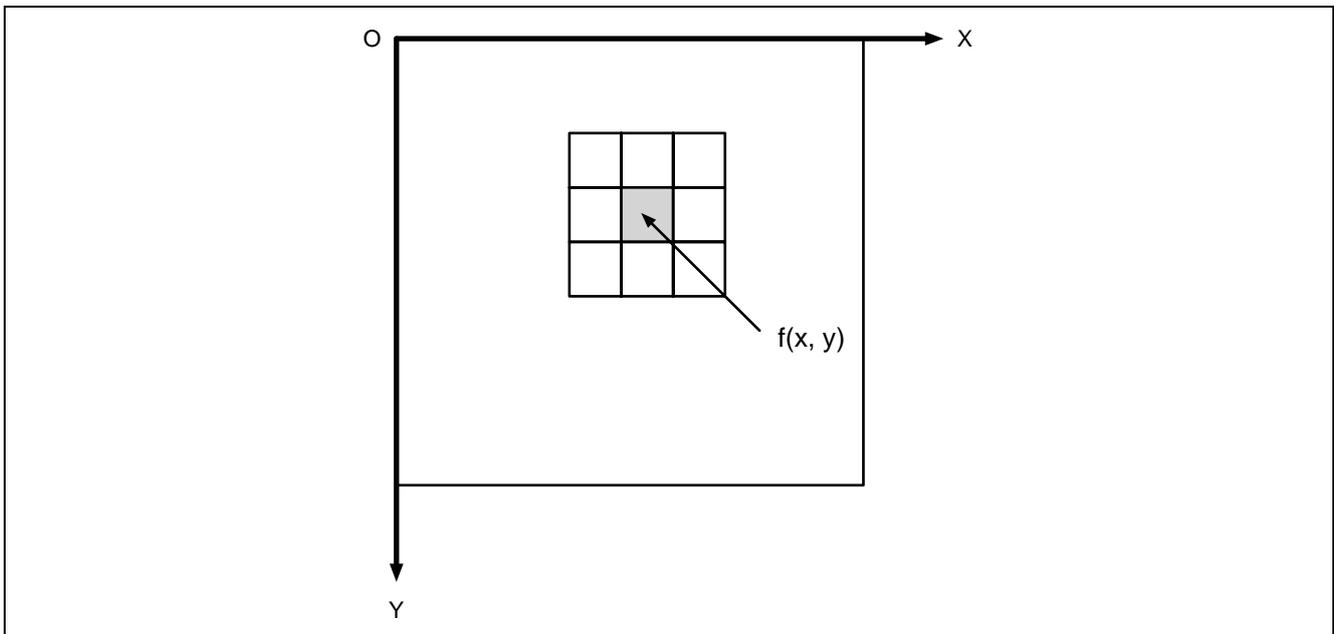


Figure 1 3×3 Filter Mask Applied to Image f

Figure 2 below illustrates the relationship between the pixels of image f and the coefficients of 3×3 filter mask w .

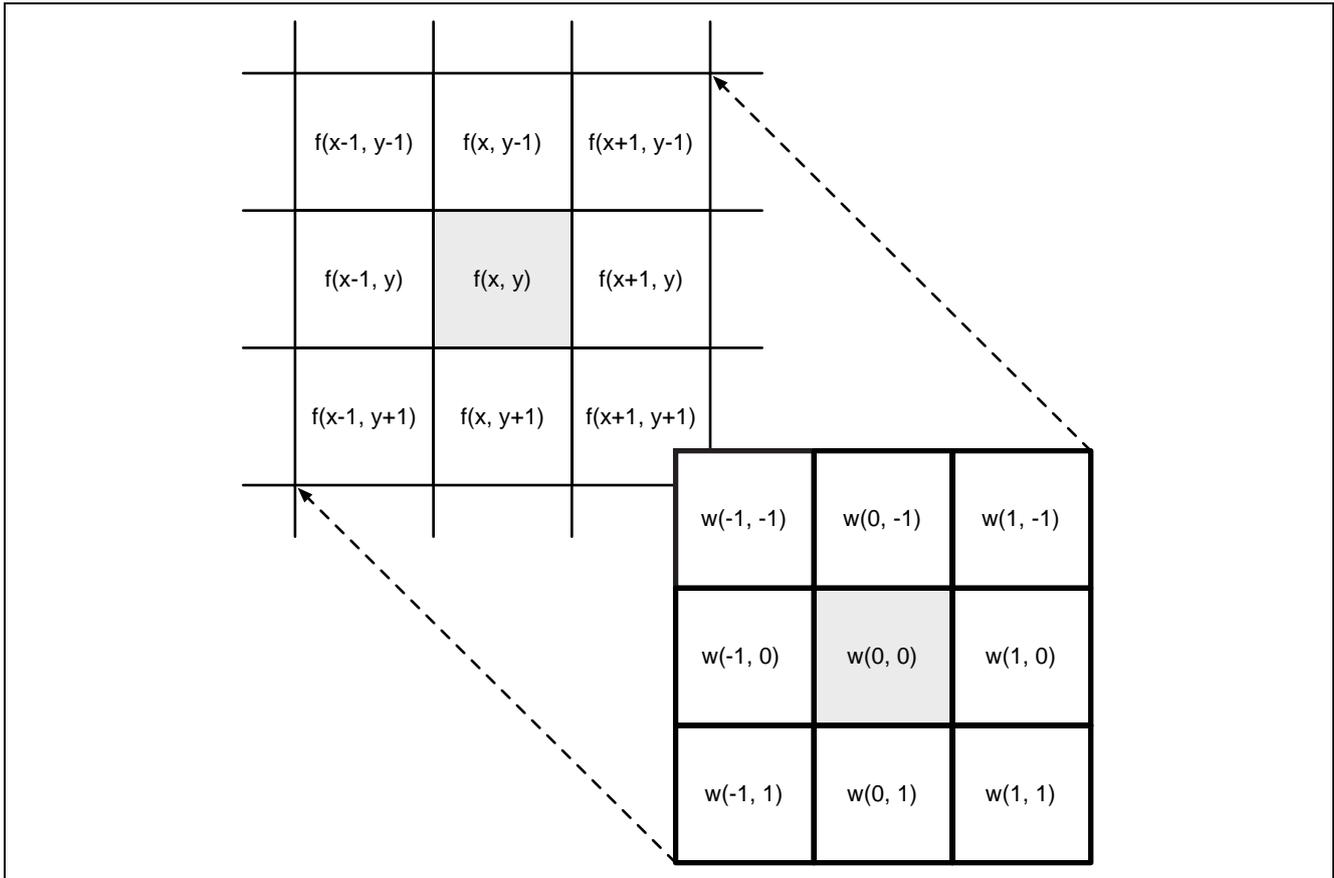


Figure 2 Relationship between Image f Pixel $f(x, y)$ and 3×3 Filter Mask Coefficient $w(s, t)$

Suppose that the input image pixel is $f(x, y)$ and the 3×3 filter mask coefficient is $w(s, t)$. Pixel $g(x, y)$ of the image output by the filter process can be expressed as follows.

$$g(x, y) = \sum_{s=-1}^1 \sum_{t=-1}^1 w(s, t) \times f(x + s, y + t)$$

This process simply means multiplying the pixels and mask coefficients together in sequence and adding up the products. Thus, it can utilize the RX family's multiply-accumulation instruction. This chapter explains several examples of an image filter which can be implemented using the mechanism above. The filter explanation is based on the sample image shown in figure 3.



Figure 3 Sample Image

2.1 Smoothing Filters

This section describes smoothing filters which smooth out gradations in images. Smoothing filters soften an image by averaging the value of a pixel with its surrounding pixels. They are available for purposes such as to obscure images and eliminate noise.

Figure 4 shows two typical examples of a smoothing filter mask. The filter on the left finds the arithmetic average value for the 3 × 3 mask in a straightforward manner. The filter on the right finds the average value by weighting the pixels in the center. Both filter masks need to divide the result of filtering by the total number of coefficients to calculate an average value.

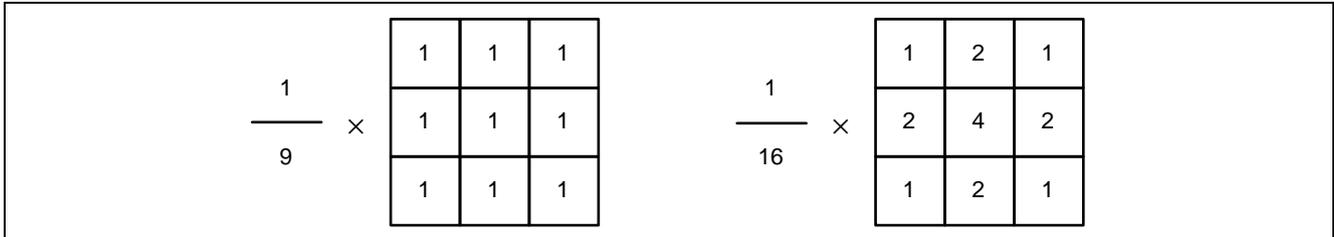


Figure 4 Smoothing Filter Masks (Left: Arithmetic Average, Right: Weighted Average)

Figure 5 shows examples of applying the smoothing filters above to the sample image.



Figure 5 Example Outputs of the Smoothing Filters (Left: Arithmetic Average, Right: Weighted Average)

2.2 Edge Detection Filters

This section describes the Sobel and Prewitt filters used for detecting edges in an image. This edge detection simply means image processing which highlights or extracts the area having steep gradations in an image.

Figure 6 shows the two Sobel filter masks. One is available for horizontal edge detection and the other for vertical edge detection.



Figure 6 Sobel Filter Masks (Left: Horizontal, Right: Vertical)

Figure 7 shows examples of applying the Sobel filters above to the sample image.

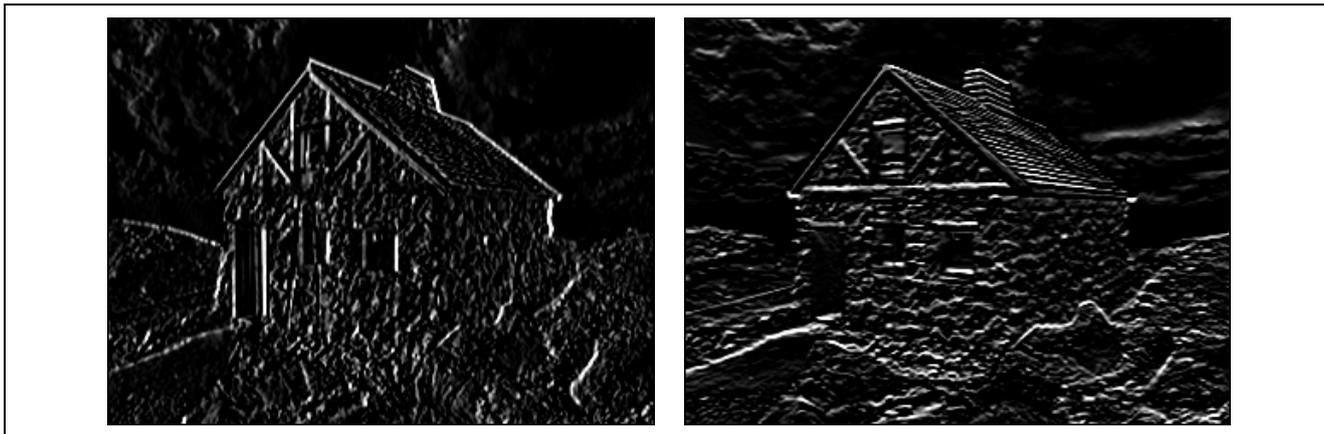


Figure 7 Example Outputs of the Sobel Filters (Left: Horizontal, Right: Vertical)

Figure 8 shows the two Prewitt filter masks. One is available for horizontal edge detection and the other for vertical edge detection like the Sobel filter masks.

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

Figure 8 Prewitt Filter Masks (Left: Horizontal, Right: Vertical)

Figure 8 shows examples of applying the Prewitt filters above to the sample image.

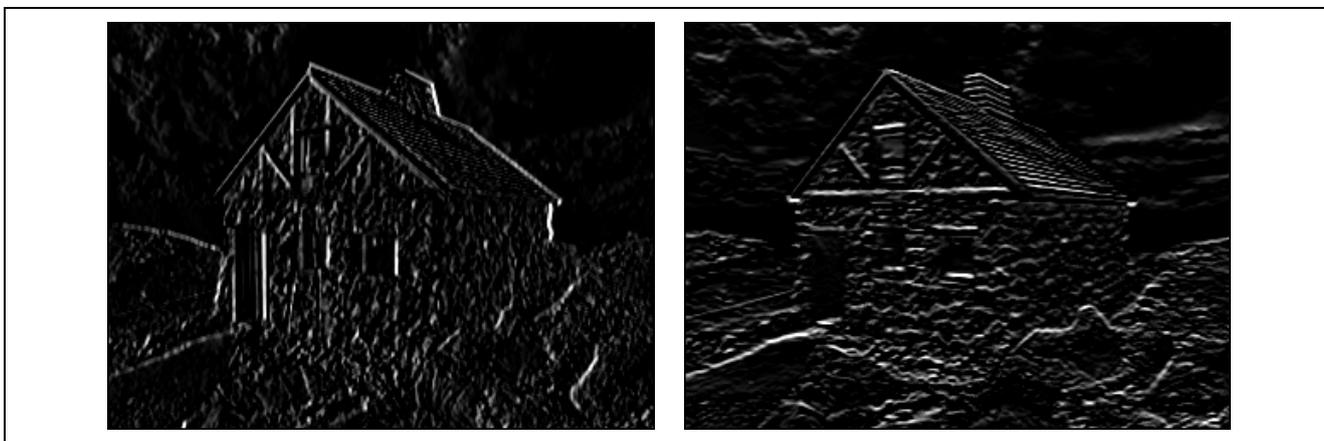


Figure 9 Example Outputs of the Prewitt Filters (Left: Horizontal, Right: Vertical)

2.3 Sharpening Process (Laplacian Filter)

This section describes the image sharpening process (unsharp masking) with the Laplacian filter. Figure 10 shows an example of the Laplacian filter mask.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 10 Laplacian Filter Mask

Figure 11 shows an example of applying the Laplacian filter above to the sample image. Note that the pixel values for the image shown are properly scaled.



Figure 11 Example Output of the Laplacian Filter (after Scaling)

Simply put, the Laplacian filter extracts gradation changes (contours) by subtracting the average value (equal to the result of smoothing) of the surrounding pixels from the pixel value of an original image. These contours can be added to the original image to sharpen the image. When writing a program, note that the output of the Laplacian filter might fall outside the range between the maximum and minimum pixel values of an original image and that this output, when added to the original image, might also exceed the maximum pixel value.

Figure 12 shows examples of the Sharpening Process with the Laplacian filter applied to the sample image.



Figure 12 Examples of the Sharpening Process (Left: Original Image, Right: Sharpened Image)

3. Image Filter Programs

This chapter describes the image filter programs which utilize the RX family's multiply-accumulation instruction.

3.1 Image Data Structure

For simplicity, the explanation below assumes that the image filter programs handle a grayscale image which is 320 pixels wide \times 240 pixels high. Each pixel, expressed with a 16-bit signed integer for utilizing the RX family's multiply-accumulation instruction, can have any values ranging from 0 through 255 (0: Black, 255: White). The following code fragment defines the image size:

```
/* constant(s) */
#define WIDTH 320 /* image width */
#define HEIGHT 240 /* image height */
```

3.2 Filter Mask Structure

Like image data, each filter mask is expressed with an array of 16-bit signed data for using the RX family's multiply-accumulation instruction. The filter masks are of 3×3 matrix so that there are in all nine coefficients to be stored in an array. However, the set of coefficients on the same line starts at a 32-bit boundary, and because of this, padding is inserted at the end of each line. This is shown in the following pseudo code:

```
int16_t w[12] = {
    w(-1,-1), w(0,-1), w(1,-1), /* padding */ 0,
    w(-1, 0), w(0, 0), w(1, 0), /* padding */ 0,
    w(-1, 1), w(0, 1), w(1, 1), /* padding */ 0,
};
```

Note that the coefficients can be considered 16-bit signed integers or fixed-point data depending on the type of filter. Fixed-point data are expressed as 16-bit signed data which include a decimal point between b15 and b14 as shown in figure 13. The coefficients can be set to any value from -1.0 to 1.0 .

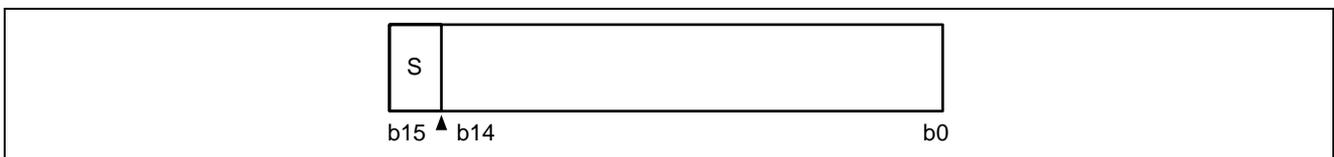


Figure 13 Fixed-point Data Format

3.3 Filter Functions

A filter function takes in an input image and filter mask as arguments, applies the filter mask to the image and puts out a different image. This section separately shows the program codes using integer coefficients of a filter mask and those using fixed-point coefficients.

Below is filter function `filter_macw1` which uses fixed-point coefficients of a filter mask. This function creates an image by applying filter mask to image `f` and outputs the created image to `g`.

```

/* Apply filter mask to image f and output the result to g (using the fixed-
point coefficients). */
#include <machine.h>

void filter_macw1(int16_t *f, int16_t *g, int16_t mask[12])
{
    int x, y;
    int16_t *r1 = f;
    int16_t *r2 = r1 + WIDTH;
    int16_t *r3 = r2 + WIDTH;
    int16_t *ro = g + WIDTH;

    /* Note: Do not filter the pixels on the edges (four sides) of the image.
*/
    for (y = 0; y < HEIGHT - 2; y++) {
        for (x = 0; x < WIDTH - 2; x++) {
            ro[x + 1] = (int16_t) (macw1(r1 + x, mask, 3)
                                   + macw1(r2 + x, mask + 4, 3)
                                   + macw1(r3 + x, mask + 8, 3));
        }
        r1 += WIDTH;
        r2 += WIDTH;
        r3 += WIDTH;
        ro += WIDTH;
    }
}

```

Below is filter function filter_mac1 which uses integer coefficients of a filter mask. filter_mac1 creates an image by applying filter mask to image f and outputs the created image to g like filter_macw1.

```

/* Apply filter mask to image f and output the result to g (using the integer
coefficients). */
void filter_mac1(int16_t *f, int16_t *g, int16_t mask[12])
{
    int x, y;
    int16_t *r1 = f;
    int16_t *r2 = r1 + WIDTH;
    int16_t *r3 = r2 + WIDTH;
    int16_t *ro = g + WIDTH;

    /* Note: Do not filter the pixels on the edges (four sides) of the image.
*/
    for (y = 0; y < HEIGHT - 2; y++) {
        for (x = 0; x < WIDTH - 2; x++) {
            ro[x + 1] = (int16_t) (mac1(r1 + x, mask, 3)
                                   + mac1(r2 + x, mask + 4, 3)
                                   + mac1(r3 + x, mask + 8, 3));
        }
        r1 += WIDTH;
        r2 += WIDTH;
        r3 += WIDTH;
        ro += WIDTH;
    }
}

```

Note that the filter functions above do not filter the pixels on the edges (four sides) of an input image. Filtering these pixels is impossible unless required surrounding pixels are provided in some way. The process is simplified to avoid complexity of the programs.

3.4 Pixel Scaling Function

Below is function equalize which properly scales the values of image pixels. This function is used in the image sharpening process.

```
/* Scale the pixel values of image image to values within the range of 0 to
255. */
void equalize(int16_t *image)
{
    int16_t *p;
    int min, max;

    min = max = image[0];
    for (p = image + 1; p < image + WIDTH * HEIGHT; p++) {
        if (*p < min) {
            min = *p;
        }
        if (*p > max) {
            max = *p;
        }
    }
    for (p = image; p < image + WIDTH * HEIGHT; p++) {
        *p = (int16_t) (((*p - min) * 255) / (max - min));
    }
}
```

3.5 Image Data Add-up Function

Below is function image_add which adds up pixel values of two images. This function is used in the image sharpening process like function equalize described in the previous section.

```
/* Add the pixel values of image g to the pixel values of image f. */
void image_add(int16_t *f, const int16_t *g)
{
    int i;

    for (i = 0; i < WIDTH * HEIGHT; i++) {
        *f++ += *g++;
    }
}
```

4. Sample Programs

This chapter shows samples of the image filter programs described in the previous chapter.

4.1 Environment for Executing the Sample Programs

The environment for executing the sample programs is as follows.

```
/* local variable(s) */
static int16_t buf[WIDTH * HEIGHT];
static int16_t test_image[WIDTH * HEIGHT] = {
#include "image.h"
};

/* bitmap and palette */
uint8_t bitmap[WIDTH * HEIGHT];
const uint32_t palette[256] = {
#include "palette.h"
};

/* Output image img to bitmap. */
void put_image(const int16_t *img)
{
    int i, c;

    for (i = 0; i < WIDTH * HEIGHT; i++) {
        c = img[i];
        if (c < 0) {
            c = 0;
        } else if (c > 255) {
            c = 255;
        }
        bitmap[i] = (uint8_t) c;
    }
}
```

Variables `buf` and `test_image` store image data. Variable `buf` is used as a working image buffer. It mainly stores the output of the image filter. Variable `test_image` stores the sample image data (initial data are defined in header file "image.h").

Variables `bitmap` and `palette` constitute a bitmap (8-bit index color) for displaying images in the integrated development environment (High-performance Embedded Workshop). The sample program calls function `put_image` which then converts the specified image data and writes the conversion results into the bitmap. Variable `palette` stores grayscale color data in RGB888 format which are associated with indexes 0 through 255 (initial data are defined in the header file "palette.h").

The image written into the bitmap can be displayed in the High-performance Embedded Workshop. For details of how to display it in the High-performance Embedded Workshop, refer to the next section.

4.2 Bitmap Image Display

To display the bitmap image (bitmap) in the integrated development environment (High-performance Embedded Workshop)'s window, follow these steps:

1. Select "Screen" > "Graphic" > "Image" menu from the High-performance Embedded Workshop's menu bar.
2. The "Image Property" dialog box opens. (Refer to figure 14.)
3. Select "RGB" as "Mode" in "Color Data" in the dialog box.
4. Select "8 bits (Index Color)" from the "Bit/Pixel" menu in the dialog box.
5. Select the symbol "_bitmap" from the "Data address" menu in the dialog box.
6. Select the symbol "_palette" from the "Palette address" menu in the dialog box.
7. Type 320 (bitmap width) in the "Width" edit box in the dialog box.
8. Type 240 (bitmap height) in the "Height" edit box in the dialog box.
9. Click the OK button to close the dialog box.
10. The High-performance Embedded Workshop's window opens to display the image.

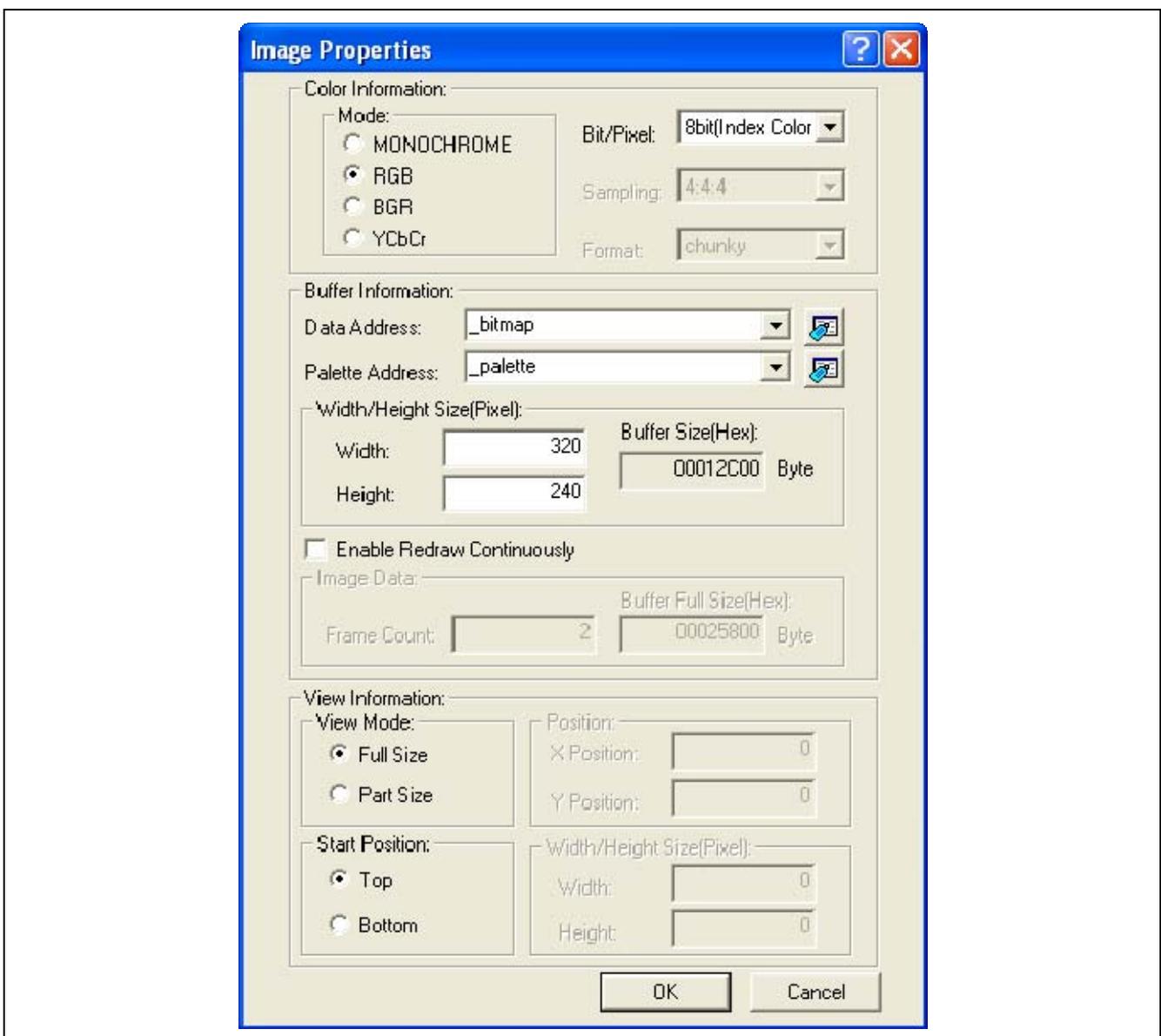


Figure 14 "Image Property" Dialog Box

4.3 Sample Smoothing Filters

Below are programs serving as smoothing filter masks. `smoothing_mask_1` calculates an arithmetic average. `smoothing_mask_2` calculates an weighted average.

```
int16_t smoothing_mask_1[12] = {
    3640, 3640, 3640, /* padding */ 0,
    3640, 3640, 3640, /* padding */ 0,
    3640, 3640, 3640, /* padding */ 0,
};

int16_t smoothing_mask_2[12] = {
    2048, 4096, 2048, /* padding */ 0,
    4096, 8192, 4096, /* padding */ 0,
    2048, 4096, 2048, /* padding */ 0,
};
```

The coefficients for these masks are 16-bit signed fixed-point data. Smoothing filters need to divide the result of filtering by the total number of coefficients to calculate an average value. In this example, a separate division process is already performed for each coefficient.

Below are sample programs which use the smoothing filter masks above. These masks contain fixed-point coefficients. Thus, the filter function used is `filter_macw1` for fixed-point data.

```
/* Test image smoothing (1) */
memset(buf, 0, sizeof buf);
filter_macw1(test_image, buf, smoothing_mask_1);
put_image(buf);

/* Test image smoothing (2) */
memset(buf, 0, sizeof buf);
filter_macw1(test_image, buf, smoothing_mask_2);
put_image(buf);
```

4.4 Sample Edge Detection Filters

Below are programs serving as the Sobel filter masks. `sobel_h_mask` is a horizontal filter mask. `sobel_v_mask` is a vertical filter mask.

```
/* Sobel (horizontal) edge detection filter */
int16_t sobel_h_mask[12] = {
    -1,  0,  1, /* padding */ 0,
    -2,  0,  2, /* padding */ 0,
    -1,  0,  1, /* padding */ 0,
};

/* Sobel (vertical) edge detection filter */
int16_t sobel_v_mask[12] = {
    -1, -2, -1, /* padding */ 0,
    0,  0,  0, /* padding */ 0,
    1,  2,  1, /* padding */ 0,
};
```

Below are sample programs which use the filter masks above. These filter masks contain integer coefficients. Thus, the filter function used is `filter_mac1` for integers.

```
/* Test image edge detection (Sobel, horizontal) */
memset(buf, 0, sizeof buf);
filter_macl(test_image, buf, sobel_h_mask);
put_image(buf);

/* Test image edge detection (Sobel, vertical) */
memset(buf, 0, sizeof buf);
filter_macl(test_image, buf, sobel_v_mask);
put_image(buf);
```

Below are sample programs serving as the Prewitt filter masks. `prewitt_h_mask` is a horizontal filter mask. `prewitt_v_mask` is a vertical filter mask.

```
/* Prewitt (horizontal) edge detection filter */
int16_t prewitt_h_mask[12] = {
    -1,  0,  1, /* padding */ 0,
    -1,  0,  1, /* padding */ 0,
    -1,  0,  1, /* padding */ 0,
};

/* Prewitt (vertical) edge detection filter */
int16_t prewitt_v_mask[12] = {
    -1, -1, -1, /* padding */ 0,
     0,  0,  0, /* padding */ 0,
     1,  1,  1, /* padding */ 0,
};
```

Below are sample programs which use the filter masks above. These masks contain integer coefficients. Thus, the filter function used is `filter_macl` for integers.

```
/* Test image edge detection (Prewitt, horizontal) */
memset(buf, 0, sizeof buf);
filter_macl(test_image, buf, prewitt_h_mask);
put_image(buf);

/* Test image edge detection (Prewitt, vertical) */
memset(buf, 0, sizeof buf);
filter_macl(test_image, buf, prewitt_v_mask);
put_image(buf);
```

4.5 Sample Sharpening Process (Laplacian Filter)

Below is a sample program serving as the Laplacian filter mask.

```
/* Laplacian filter */
int16_t laplacian_mask[12] = {
    -1, -1, -1, /* padding */ 0,
    -1,  8, -1, /* padding */ 0,
    -1, -1, -1, /* padding */ 0,
};
```

Below is a sample sharpening program which uses the filter mask above. This mask contains integer coefficients. Thus, the filter function used is `filter_mac1` for integers. This sample program scales the output of the filter, adds the scaled output to the original image, and then performs the scaling process again.

```
/* Test image sharpening (Laplacian filter) */
memset(buf, 0, sizeof buf);
filter_mac1(test_image, buf, laplacian_mask);
equalize(buf);
put_image(buf);
image_add(buf, test_image);
equalize(buf);
put_image(buf);
```

Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.

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.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
 2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
 3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
 4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
 5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
 6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
 7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
"Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
 8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
 9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
 10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
 11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
 12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



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 Laviend' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141