RX Family

Using Intrinsic Functions for Multiply-Accumulate Operations

R01AN0255EJ0100
Rev.1.00
Mar 14, 2011

Introduction

This document describes the use of DSP function instructions for intrinsic functions on RX Family MCUs.

Target Device

RX Family

Contents

1. Introduction

The RX Family CPU core (the RX) integrates a 16-bit × 16-bit multiply-accumulate unit. When executing a 32-bit × 32-bit integer-multiply instruction (MUL instruction), which is normally used for arithmetic expressions or address calculations employing multiplication, the lower 32 bits of the 64-bit result of the 32-bit × 32-bit operation are taken as the calculation result. In other words, it is assumed when using the MUL instruction that the calculation result will not exceed 32 bits. However, there is usually valid data contained in the high-order bits of the result of a multiply or multiply-accumulate operation when fixed-point expressions are used for numeric data (see reference [1], for example). As a result, only a narrow range of numeric data can be handled when using the MUL instruction for multiply or multiply-accumulate operations involving numeric data using fixed-point expressions, because the MUL instruction can only be used when 32 bits are sufficient to express the calculation result. As a solution to this problem, the RX incorporates a 48-bit accumulator that supports multiply-accumulate operation instructions (and multiply instructions), instructions that perform rounding operations on values stored in the accumulator, and transfer instructions for moving data between the accumulator and the general registers. By combining these multiply-accumulate operation instructions, rounding instructions, etc., a variety of operations involving numeric data using fixed-point expressions can be performed at high speed, resulting in data processing capacity rivaling that of a DSP. Explanations of the use of these instructions, interspersed with examples, are provided below. The description below covers multiply-accumulate operation instruction and related topics. For further details, refer to the *RX Family Software Manual*. The use of these multiply-accumulate operation instructions and rounding instructions is described in the application note *Using Multiply-Accumulate Operation Instructions* (R01AN0254EJ0100).

The present application note describes the use of intrinsic functions for multiply-accumulate operations, which are provided as extended functions of the RX Family C/C++ Compiler (the compiler) (intrinsic functions supporting the multiply-accumulation instruction are available at compiler version 1.01 or later). Intrinsic functions can be called in the same manner as other functions in programs written in C, and they are expanded inline by the compiler. By using intrinsic functions, it is possible easily to make efficient use of multiply-accumulate operations without having to use assembly language.

[1] Mori, Natori, Torii, *Iwanami kōza jōhōkagaku: 18 sūchi keisan* [Numerical Computation], pp. 1–27, Iwanami Shoten, 1982

2. Types of Intrinsic Functions

The following three intrinsic functions are provided as extended functions of the compiler.

- macw1
- macw2
- macl

The intrinsic functions **macw1** and **macw2** are intended for use with fixed-point data. The functions **macw1** and **macw2** each use the **RACW** instruction to perform rounding on the result of multiply-accumulate operations using 16-bit signed data values, returning the result of the rounding operation as a 16-bit signed data value. The difference between **macw1** and **macw2** is that the former uses a **RACW #1** instruction and the latter uses a **RACW #2** instruction. In contrast, the intrinsic functions **macl** is intended for use with integer data. The function **macl** returns as a 32-bit signed data value the result of multiply-accumulate operations using 16-bit signed data values.

2.1 macw1

The intrinsic function **macw1** performs multiply-accumulate operations on fixed-point data. The function **macw1** executes multiply-accumulate operation on 16-bit signed data values and returns the result as a 16-bit signed data value. Note that the operation result is rounded by using the **RACW #1** instruction.

The syntax used to call the intrinsic function **macw1** is as follows.

```
#include <machine.h>
short macw1(short* data1, short* data2, unsigned long count);
```

The arguments **data1** and **data2** each specify the start address of a 16-bit signed data array. The argument count specifies the length of (number of elements in) the arrays.

When the intrinsic function **macw1** is called, it is expanded inline into the following sequence of operations.

(1) Accumulator initialization (setting to 0)
(2) Repeating loop containing multiply-accumulate operation on **data1** and **data2** (**MULLO**, **MULHI**, **MACLO**, and **MACHI** instructions)
(3) Rounding of operation result (accumulator) (**RACW #1** instruction)
(4) Extraction of operation result from accumulator (**MVFACHI** instruction)

Figure 1 illustrates this sequence of operations. The multiply-accumulate operations use the accumulator (ACC) and have 48-bit precision. After the operation result is rounded by the **RACW #1** instruction, the **MVFACHI** instruction is used to extract the upper 32 bits of the accumulator value as the return value.

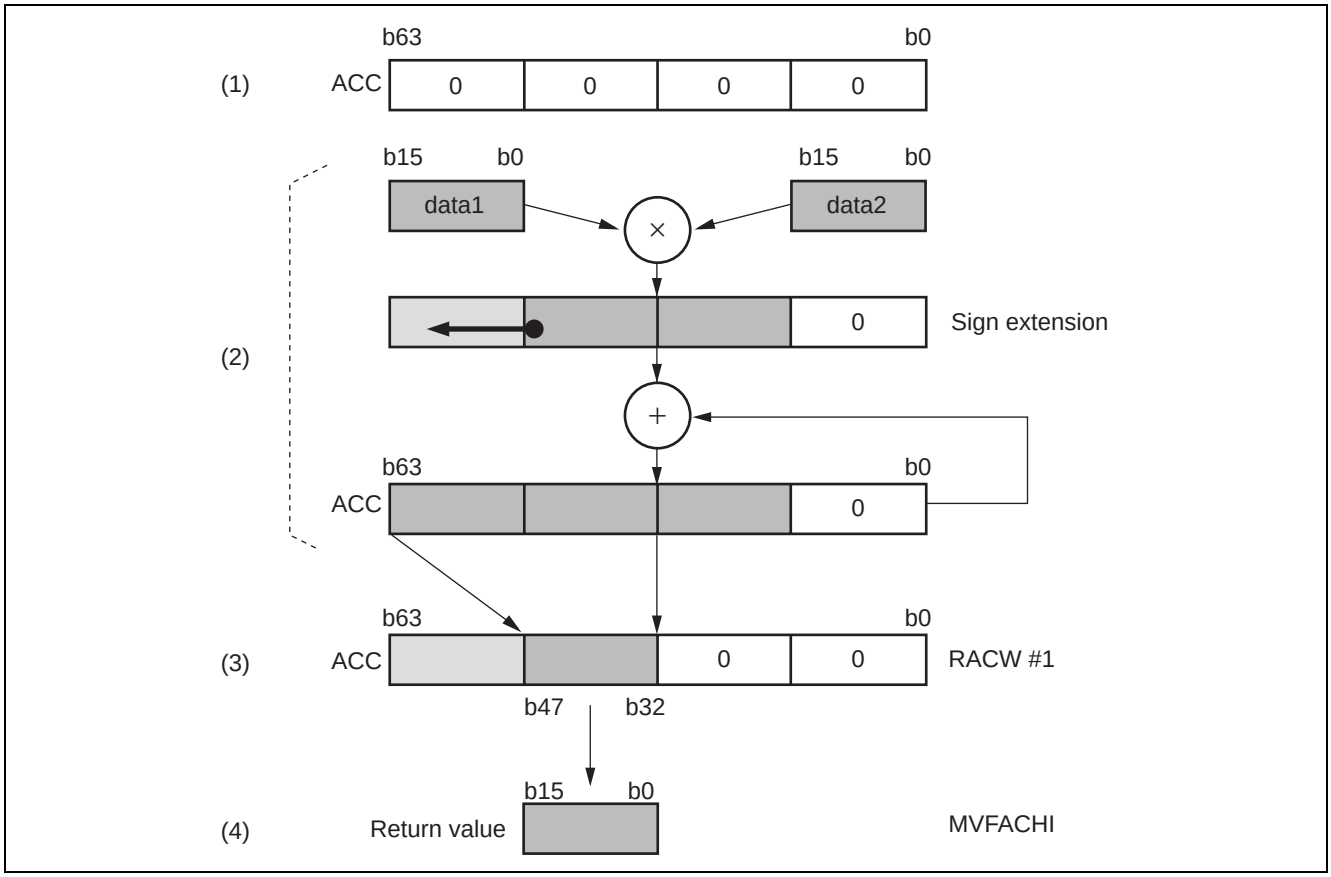

Figure 1 Operation Sequence of Intrinsic Function macw1

These operations can be represented in C language pseudocode as follows.

```
short macw1(short* data1, short* data2, unsigned long count)
{
    short ret;

    /* (1) */
    accumulator initialization ;
    /* (2) */
    while (count--) {
        multiply-accumulate operation(*data1, *data2);  // MULLO, MACLO,
        MACHI instructions
        data1++;
        data2++;
    }
    /* (3) */
    rounding ;  // RACW #1 instruction
    /* (4) */
    ret = operation result extracted from accumulator ;  // MVFACHI
    instruction
    return ret;
}
```

2.2 macw2

The intrinsic function **macw2** performs multiply-accumulate operations on fixed-point data. The function **macw2** executes multiply-accumulate operations on 16-bit signed integer data values and returns the result as a 16-bit signed integer data value. Note that the operation result is rounded by using the **RACW #2** instruction.

The syntax used to call the intrinsic function **macw2** is as follows.

```
#include <machine.h>
short macw2(short* data1, short* data2, unsigned long count);
```

The arguments **data1** and **data2** each specify the start address of a 16-bit signed data array. The argument count specifies the length of (number of elements in) the arrays.

When the intrinsic function **macw2** is called, it is expanded inline into the following sequence of operations.

(1) Accumulator initialization (setting to 0)
(2) Repeating loop containing multiply-accumulate operation on **data1** and **data2** (**MULLO**, **MULHI**, **MACLO**, and **MACHI** instructions)
(3) Rounding of operation result (accumulator) (**RACW #2** instruction)
(4) Extraction of operation result from accumulator (**MVFACHI** instruction)

Figure 2 illustrates this sequence of operations. The multiply-accumulate operations use the accumulator (ACC) and have 48-bit precision. After the operation result is rounded by the **RACW #2** instruction, the **MVFACHI** instruction is used to extract the upper 32 bits of the accumulator value as the return value.

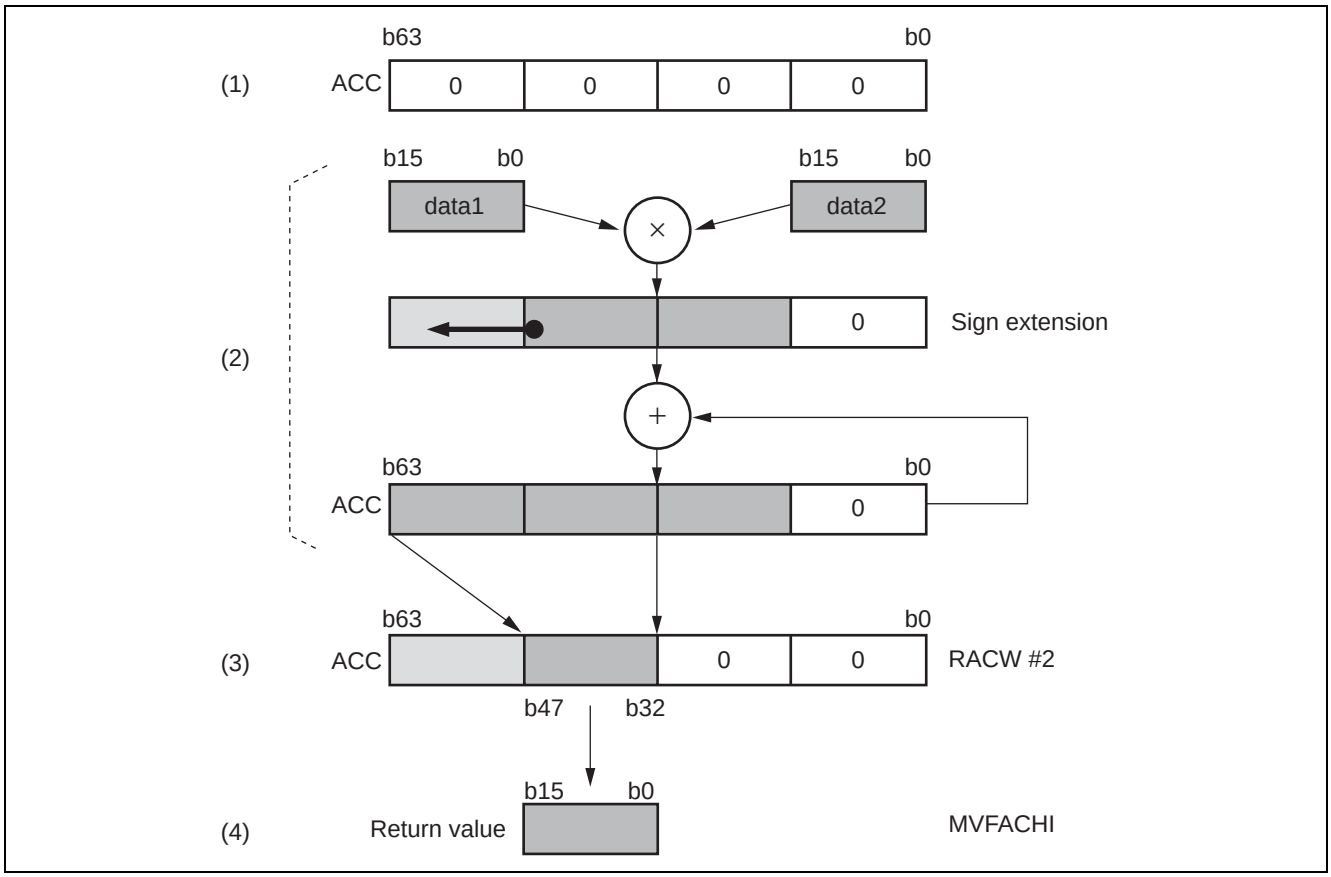

Figure 2 Operation Sequence of Intrinsic Function macw2

These operations can be represented in C language pseudocode as follows.

```
short macw2(short* data1, short* data2, unsigned long count)
{
    short ret;

    /* (1) */
    accumulator initialization ;
    /* (2) */
    while (count--) {
        multiply-accumulate operation(*data1, *data2);  // MULLO, MACLO,
        MACHI instructions
        data1++;
        data2++;
    }
    /* (3) */
    rounding ;  // RACW #2 instruction
    /* (4) */
    ret = operation result extracted from accumulator ;  // MVFACHI
    instruction
    return ret;
}
```

2.3 macl

The intrinsic function **macl** performs multiply-accumulate operations on integer data. The **macl** function executes multiply-accumulate operations on 16-bit signed data values and returns the result as a 32-bit signed data value.

The syntax used to call the intrinsic function **macl** is as follows.

```
#include <machine.h>
short macwl(short* data1, short* data2, unsigned long count);
```

The arguments **data1** and **data2** each specify the start address of a 16-bit signed data array. The argument count specifies the length of (number of elements in) the arrays.

When the intrinsic function **macl** is called, it is expanded inline into the following sequence of operations.

(1) Accumulator initialization (setting to 0)
(2) Repeating loop containing multiply-accumulate operation on **data1** and **data2** (**MULLO**, **MULHI**, **MACLO**, and **MACHI** instructions)
(3) Extraction of operation result from accumulator (**MVFACHI** instruction)

Figure 3 illustrates this sequence of operations. The multiply-accumulate operations use the accumulator (ACC) and have 48-bit precision. The **MVFACHI** instruction is used to extract the middle 32 bits of the accumulator value as the return value.

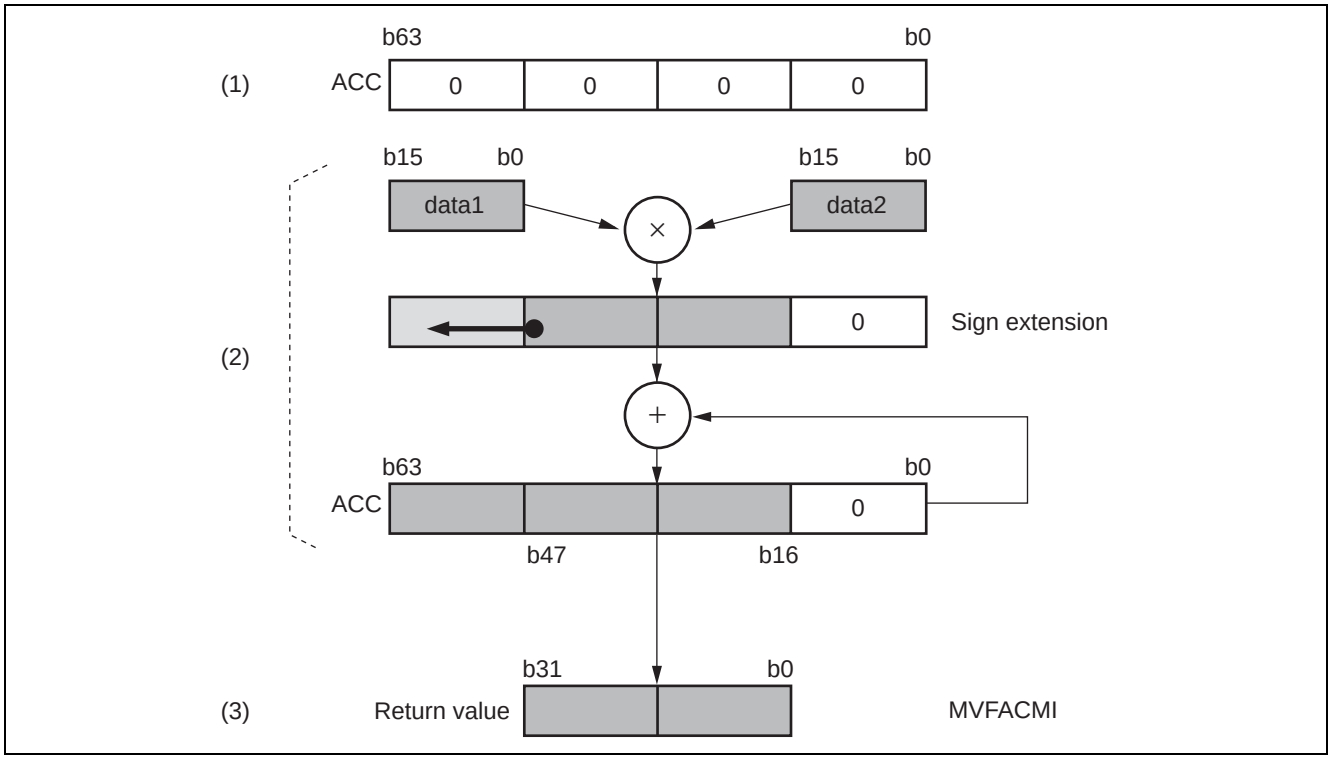

Figure 3 Operation Sequence of Intrinsic Function macl

These operations can be represented in C language pseudocode as follows.

```
short macl(short* data1, short* data2, unsigned long count)
{
    long ret;

    /* (1) */
    accumulator initialization ;
    /* (2) */
    while (count--) {
        multiply-accumulate operation(*data1, *data2);  // MULLO, MACLO,
        MACHI instructions
        data1++;
        data2++;
    }
    /* (3) */
    ret = operation result extracted from accumulator ;  // MVFACHI
    instruction
    return ret;
}
```

3. Notes on Use of Intrinsic Functions

This section describes points to be borne in mind when using the intrinsic functions (**macw1**, **macw2**, and **macl**).

3.1 Alignment of Data Address Boundaries

How to arrange data used by multiply-accumulate operations in C programs such that it is aligned with 4-byte address boundaries is described below.

When two or more units of data are used for multiply-accumulate operations using an intrinsic function (**macw1**, **macw2**, or **macl**), two 16-bit data values are packed into a single 32-bit (4-byte) unit of data and loaded into a register. The RX can read 32-bit data units from any address boundary, so the data may be assigned to any address in memory. However, if the data is aligned with a 4-address boundary, the RX can load the full 32 bits in a single memory access. This is an effective way of speeding up data loading.

The technique used in C to align a unit of data with a 4-byte boundary in C involves wrapping the data to the same union as a dummy long integer member. The following declaration is an example of this.

```
/* Example of aligning u1.data with a 4-byte address boundary */
union { int16_t data[10]; int32_t dummy; } u1;
```

In this example, the compiler will ensure that the start address of the short type array (**u1.data[10]**) is aligned with a 4-byte address boundary.

It is possible to initialize the first member of a union in ANSI standard C, so defining initialization data in the following manner presents no problems.

```
/* Example of aligning u2.data with a 4-byte address boundary and setting its
initial value */
const union { int16_t data[10]; int32_t dummy; } u2 = {
    { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
};
```

In the above example, the initial value string is enclosed in two sets of curly brackets, and both are required. The outer set of curly brackets corresponds to the union and the inner set of curly brackets corresponds to the short type array.

3.2 Multiply-Accumulate Operation Repetition Count

The third argument of the intrinsic function (**macw1**, **macw2**, or **macl**) specifies the number of times the multiply-accumulate operation is repeated. The repetition count can be specified either with an expressions containing variables and with a constant expression. However, the sequence of instructions used by the compiler when an intrinsic function is called differs depending on whether or not the repetition count argument contains variables or is a constant expression.

Generally speaking, when all other conditions are the same, shorter and faster code will be generated when a constant expression is used for the repetition count rather than an expressions containing variables. Therefore, when the number of times the multiply-accumulate operation is to be repeated is known in advance, specifying the repetition count with a constant expression can be expected to result in the generation of more efficient code.

3.3 Saving and Restoring the Contents of the Accumulator in Interrupt Functions

In an application that uses the intrinsic functions (**macw1**, **macw2**, and **macl**), it is necessary to ensure that the contents of the accumulator (ACC) are not overwritten by an interrupt function. In other words, if there is a possibility that an interrupt function may overwrite the contents of the accumulator, the accumulator value must be saved at the start of the interrupt function and restored at the end of the interrupt function.

For details on saving and restoring the accumulator contents in interrupt functions, see the descriptions of the **save_acc** compiler option and **#pragma interrupt** extension specification in the *RX Family C/C++ Compiler Package User's Manual* (the **save_acc** compiler option is available at compiler version 1.01 or later).

4. Sample Program

This section describes the use of intrinsic functions in the sample program with reference to coordinate rotation.

4.1 Coordinate Rotation (Coordinate Conversion)

Let us consider an example of conversion involving rotation of two-dimensional coordinates (x, y) with the origin as the center, as shown in figure 4.

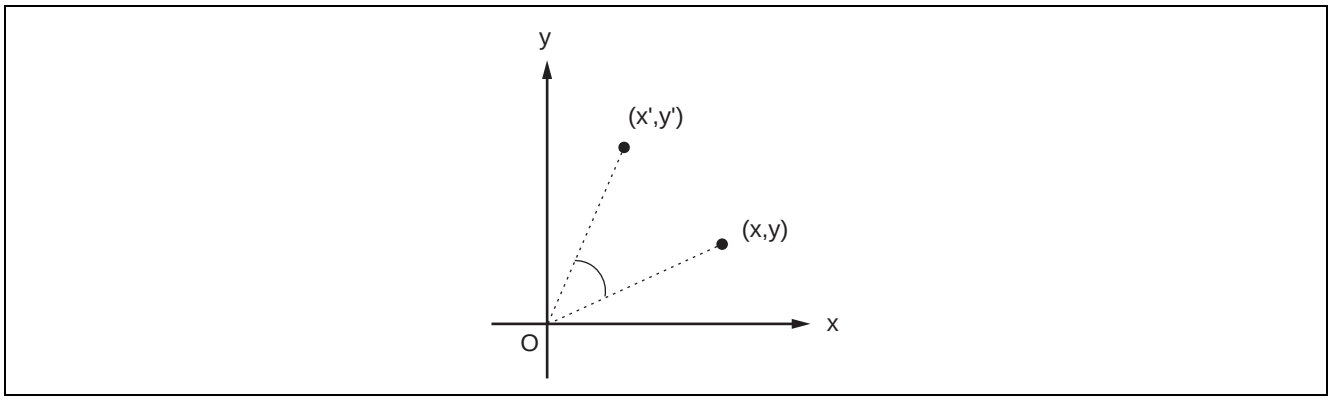

Figure 4 Coordinate Rotation (Coordinate Conversion)

The coordinate conversion can be expressed as a matrix multiplication, as follows, where θ is the angle of rotation and (x', y') are the destination coordinates.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$

In this example, the angle of rotation is specified as 75°, so the following conversion expression is used.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0.25882 & -0.96593 \\ 0.96593 & 0.25882 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$

The data values of the elements of the 4-by-4 square matrix (the conversion matrix) in the above expression range from –1.0 to 1.0, and they can be represented as 16-bit fixed-point data with the decimal point between bits 15 and 14. In this data expression bits 14 to 0 are the fractional portion of the value, so a value of 0.25882 is represented as follows.

$$0.25882 \times 2^{15} = 8481$$

Thus, it becomes 0x2121 hexadecimal notation. In this way, the elements of the conversion matrix are converted into 16-bit data values as preparation for using multiply-accumulate operations.

4.2 Data Structure and Intrinsic Function Selection

First, the 16-bit fixed-point data from the conversion matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is expressed as a one-dimensional array, and the individual elements are stored in an array in the following sequence (left to right).

$$a, b, c, d$$

Next, the two-dimensional coordinates (x, y) are expressed as a (two-element) array of 16-bit signed integers, with the first element being the x coordinate and the second element being the y coordinate.

Note that in this example the input contains fixed-point data, so it is necessary to adjust the decimal point position in order to express the operation result as a 16-bit signed integer. Specifically, the values of the conversion matrix elements each have the decimal point between bits 15 and 14, so the operation result must be shifted one bit to the left

before rounding is performed. Therefore, **macw1** has been selected as the intrinsic function for the multiply-accumulate operations.

4.3 Sample Program

A sample program written according to the thinking described in 4.1 and 4.2 is shown below.

```
#include <machine.h>

/*
  75° rotation conversion
  rotates coordinates (xy_in[0], xy_in[1]) 75°
  and stores the destination coordinate values in (xy_out[0], xy_out[1])

 */
void rotate75(int16_t *xy_in, int16_t *xy_out)
{
    static union { int16_t matrix[4]; int32_t dummy; } u = {
        {
            0x2121,  0x845d,  /* 0.25882, -0.96593 */
            0x7ba3,  0x2121,  /* 0.96593,  0.25882 */
        }
    };

    xy_out[0] = (int16_t)macw1(xy_in, u.matrix, 2);
    xy_out[1] = (int16_t)macw1(xy_in, u.matrix + 2, 2);
}

void main(void)
{
    union { int16_t coord[2]; int32_t dummy; } u1, u2;

    /* 75° rotation of coordinates (128, 64) */
    u1.coord[0] = 128;
    u1.coord[1] = 64;
    rotate75(u1.coord, u2.coord);  /* result is assigned to u2.coord */
}
```

Website and Support

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

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

Revision Record

		Description	
Rev.	**Date**	**Page**	**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.

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.

RENESAS

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