RL78 Family

R01AN1216ES0101
Rev.1.01
June 10 2013

RL78 Digital Signal Controller Library - Fixed point and Motor

Introduction

This document describes the usage of RL78 16bit-Fixed-point and Motor Control Library which is suitable for motor control S/W.

Target Device

RL78/G14 Group

Development environment

IDE Package:	CubeSuite+ for RL78,78K V1.02.00
Compiler Package:	CubeSuite+ RL78,78K0R Compiler CA78K0R V.1.4.0

Content

1. Digital Signal Controller Library

The Digital Signal Controller library contains 2 key components of motor control libraries namely

16-bit Fixed-point

Motor Control

The above can be found commonly in Motor Control software. This application note aims to explain the usage of both components as 16-bit Fixed-point or 16-bit Fixed-point with Motor Control since Fixed Point can be used in any applications.

1.1 "r_stdint.h"

This header file defines the following basic integer types.

```
typedef signed char int8_t;
typedef unsigned char uint8_t;
typedef signed short int16_t;
typedef unsigned short uint16_t;
typedef signed long int32_t;
typedef unsigned long uint32_t;
```

2. 16-bit Fixed-point Library

2.1 Overview

This library provides real-number operations using fixed-point format[1] for RL78/G14 Group. This library is focusing a motor control library released from RENESAS, but this is also useful for other application areas.

The 16-bit fixed-point library enables fast real-number operations, especially on CPU's without FPU.

This library supports the following functions.

1. Multiplication and division
2. Mathematical functions (sin, cos, atan, and sqrt)
3. Conversion between floating point data.

Multiplication functions support for fixed-point type with 8, 10, 12, 14 or 16 fraction bits. Use 8-bit, 10-bit, 12-bit, 14-bit or 16-bit depending on the required precision of your application. The library functions mainly support 12-bit precision.

In fixed-point arithmetic, the range of values is restricted compared with floating point. So appropriate precision should be selected according to the input/output values of each operations.

2.2 Format of Fixed-point Data

Following is the format of fixed-point data supported in this library.

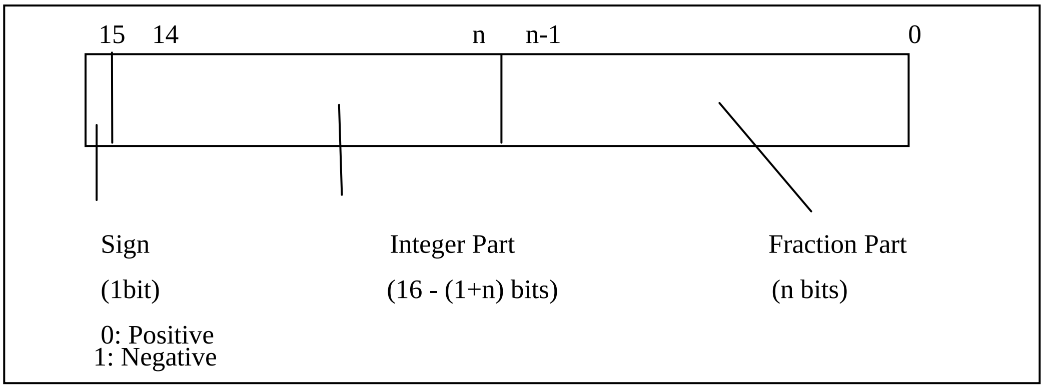


Figure 1. Fixed-point Data Format

According to the number of bits in fraction part, types FIX8, FIX10, FIX12, FIX14 and FIX16 are supported. The number indicates the number of bits in the fraction type.

Generic fixed-point type FIX is also supported, and generic fixed-point operations are supported for this type.

[1] Fixed-point format represents a real number by assuming a decimal point at some fixed bit position.

2.3 Library Files

2.3.1 16-bit Fixed-point library only

The following include file and library files are provided.

When using this library alone, include the file indicated in table 1, and link the library file (corresponding to the compiler option) indicated in table 2.

Table 1. Include File for Fixed-point Library

Library	Function	
Fixed-point library	Implements fixed-point operations	"r_fixmath.h"

Table 2. Fixed-point Libraries

Library name	Compiler Option
	cpu
R_dsp_rl78.lib	RL78/G14

Before using, copy these files into your local include or library directories.

```
include directory ── r_fixmath.h, r_stdint.h

library ──────────── R_dsp_rl78.lib
```

Figure 2. Sample Configuration

Example of Usage

The following example shows a program using FIX12 operation and how to specify the library under CubeSuite+.

[Source Program]

```
#include <stdio.h>
#include "r_fixmath.h"              // Necessary when using
                                    // fixed-point library
#define M_PI      (2048)  /* pi */
#define M_2PI_3   (1365)  /* 2*pi/3 */
#define M_PI_2    (1024)  /* pi/2 */
#define M_PI_4    ( 512)  /* pi/4 */

void print_sin()
{
    float r_flt;
    FIX12 r_fix12;

    r_fix12 = R_FIX_sin_int16(M_PI_2);  // computes sin
    r_flt = FIX12_tofloat(r_fix12);     // Convert back for printing
    printf("%f¥n", r_flt);
}
```

[How to specify the library under CubeSuite+]

Select [Property] of [CA78K0R] in project tree menu. In the dialog box [Property], select tab [Frequently Used Options (for Link)], and specify the library in "Using libraries" and the library path in "Additional library paths".

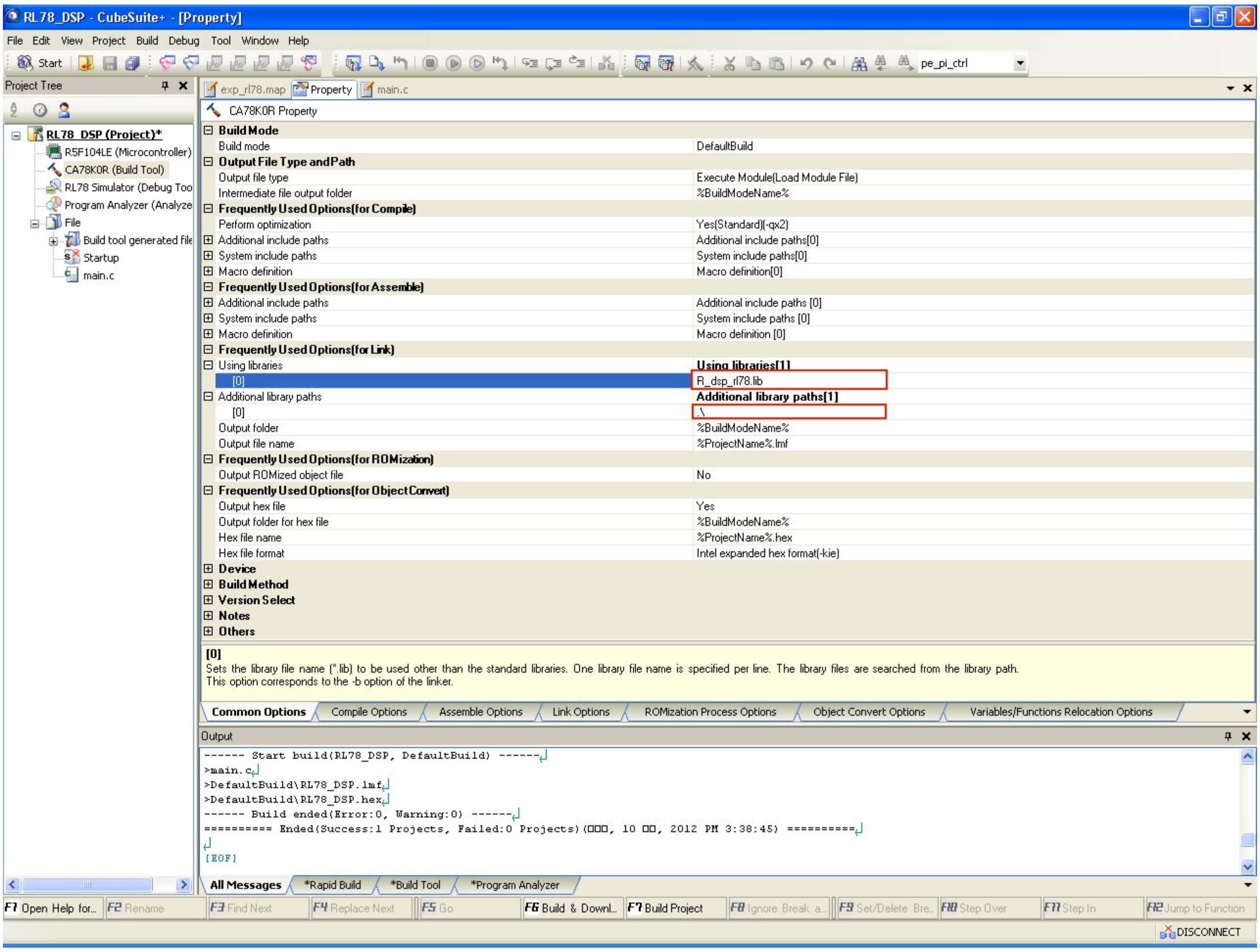


Figure 3. Specifying library

2.3.2 16-bit Fixed-point and Motor Control library

The following include file and library files are provided.

When using this library with motor control, include the file indicated in table 3, and link the library file (corresponding to the compiler option) indicated in table 4.

Table 3. Include File for Fixed-point Library

Library	Function	
Fixed-point library with motor control	Implements fixed-point and motor control operations	"r_dsp.h"

Table 4. Fixed-point Libraries

Library name	Compiler Option
	Cpu
R_dsp_rl78.lib	RL78/G14

Before using, copy these files into your local include or library directories.

```
include directory ——— r_dsp.h, r_stdint.h

library           ——————— R_dsp_rl78.lib
```

Figure 3. Sample Configuration

Example of Usage

The following example shows a program using FIX12 operation with motor and how to specify the library under CubeSuite+.

[Source Program]

```
#include <stdio.h>
#include "r_dsp.h"                // Necessary when using
                                     // fixed-point with motor library
#define M_PI       (2048)  /* pi */
#define M_2PI_3    (1365)  /* 2*pi/3 */
#define M_PI_2     (1024)  /* pi/2 */
#define M_PI_4     ( 512)  /* pi/4 */

void motor_uvw2dq()
{
    int16_t theta = 0;
    int16_t ia, ib;
    int16_t iu_ad, iw_ad;
    int16_t id_lpf, iq_lpf;

    while(1)
    {
        theta += M_PI_4;
        theta %= (2*M_PI);
        iu_ad = R_FIX_sin_int16(theta);              //simulate iu
        iw_ad = R_FIX_sin_int16(theta + M_2PI_3);    //simulate iw
        R_motor_uw2ab_int16( iu_ad, iw_ad, &ia, &ib );
        R_motor_ab2dq_int16( ia, ib, theta, &id_lpf, &iq_lpf ); //get feedback id and iq
        printf("%d¥n", id_lpf);
        printf("%d¥n", iq_lpf);
    }
}
```

[How to specify the library under CubeSuite+]

Select [Property] of [CA78K0R] in project tree menu. In the dialog box [Property], select tab [Frequently Used Options (for Link)], and specify the library in "Using libraries" and the library path in “Additional library paths”.

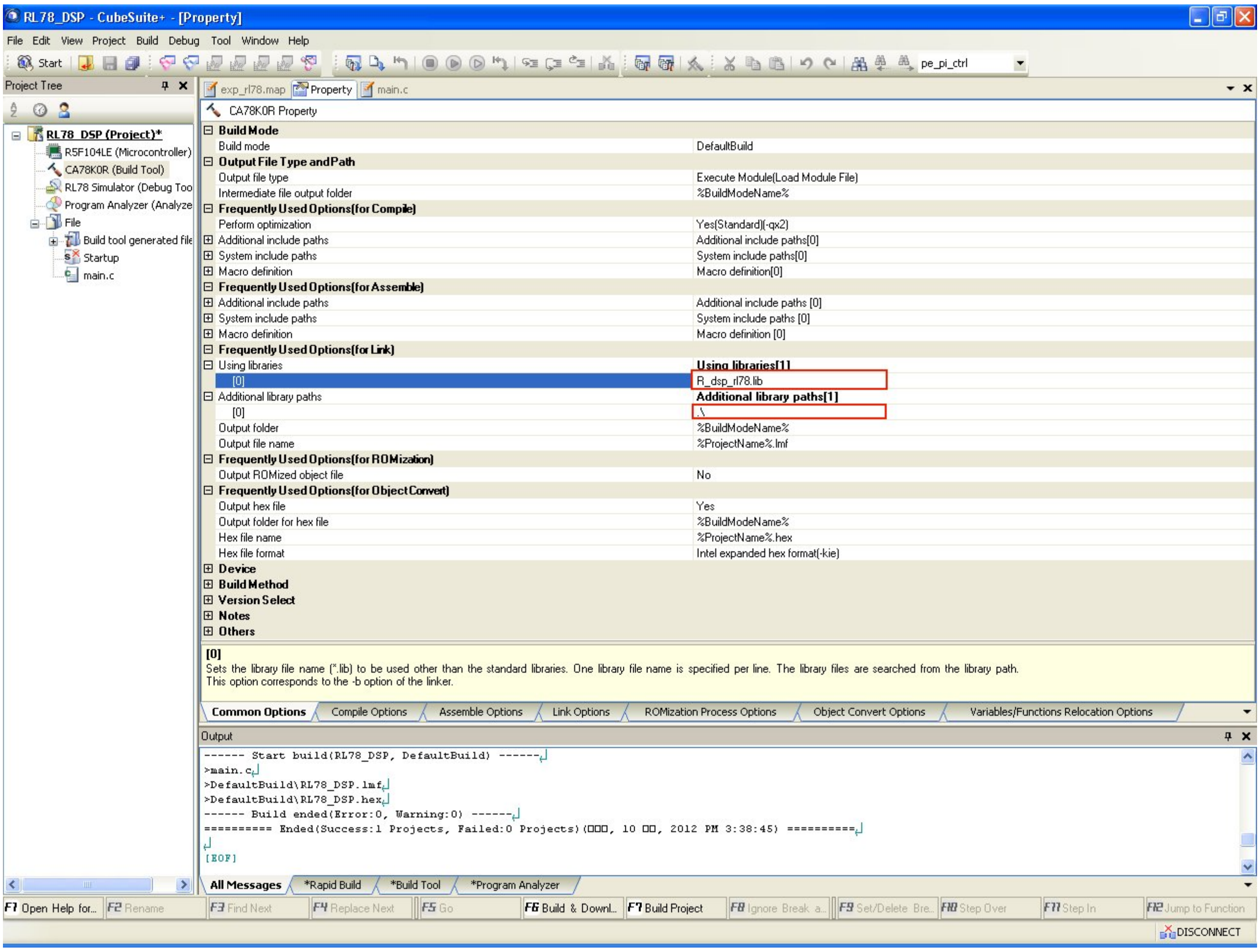


Figure 3. Specifying library

2.4 Notes on Library Usage

If the result of operation or conversion exceeds the range of fixed-point type, the result is not guaranteed.

3. Specification of 16-bit Fixed-point Library

3.1 "r_fixmath.h"/"r_dsp.h"

This header file defines types and functions for fixed-point operations.
Table 5 shows the types defined in the file and supported functions (macros).

NOTATION: The notation `<n>` in type, function, or macro names represents a number from 1 to 16. The number in the function or macro name corresponds to the number in the type name.

Table 5. Types and Supported Functions

Type	Supported functions and macros
FIX8, FIX10, FIX12, FIX14, FIX16	R_FIX<n>_mul_int16, FIX<n>_tofloat, FIX<n>_fromfloat, FIX<n>_todouble, FIX<n>_fromdouble
FIX12	R_FIX_sin_int16, R_FIX_cos_int16, R_FIX_sqrt_int16, R_FIX_sqrt2_int16
int16_t / uint16_t	R_FIX_atan_int16, R_FIX_atan2_int16, R_FIX_limit_int16, R_FIX_ulimit_int16, R_FIX_div_int16
int32_t	R_FIX_mul32_int16

FIX8, FIX10, FIX12, FIX14, FIX16 and FIX are defined as short type.

When the operands and the result of an operation are the same type (FIX<n>), use the function corresponding to that type.

[Hints on Fixed-point Library Usage]

(1) Select one of the standard fixed-point type (FIX10 or FIX12) according to the requirement of your application.
(2) Compared with floating-point types, fixed-point types have limited range of values. It is recommended to select appropriate fixed-point types according to the range of input or intermediate result, or required precision of arithmetic.
(3) When converting data between different fixed-point types, use shift operator of C language.

Example: Conversion from FIX10 to FIX12

```
FIX10 x, FIX12 y;
x=y>>2;
```

(4) When adding or subtracting between data of the same fixed-point type, use integer addition or subtraction of the C language.

Example: Addition of FIX12.

```
FIX12 x, y, z;
z=x+y;
```

(5) Conversion between floating-point types and fixed-point types should be done only when required. Unnecessary conversions reduces the efficiency. But the conversion function applied to a constant generates a constant expression by expanding a macro, and fixed-point constant can be specified without any overhead.

Example: Fixed-point constant.

```
FIX12 x;
x=FIX12_fromfloat(3.14f);
```

Table 6 shows the representations and ranges of fixed types.

Table 6. Representation and Ranges of Fixed Types

Type	Size (byte)	**Alignment** (byte)	Sign	**Range** Minimum Value	Maximum Value
FIX8	2	2	signed	-2^{7}(-128.0)	$2^{7}-2^{-8}$ (127.99609375)
FIX10	2	2	signed	-2^{5}(-32.0)	$2^{5}-2^{-10}$ (31. 9990234375)
FIX12	2	2	signed	-2^{3}(-8.0)	$2^{3}-2^{-12}$ (7. 999755859375)
FIX14	2	2	signed	-2^{1}(-2.0)	$2^{1}-2^{-14}$ (1. 99993896484375)
FIX16	2	2	signed	-2^{-1}(-0.5)	$2^{-1}-2^{-16}$ (0. 4999847412109375)
FIX	2	2	signed	Represents one of above ranges, depending on the number of fraction bits assumed.	

The macros defined are listed in table 7.

Table 7. List of Macros

Category	Name	Parameter Type	Return Type	Description
Conversion	FIX<n>_tofloat	FIX<n> n=8,10,12,14,16	float	Converts FIX<n> to float.
	FIX<n>_fromfloat	float	FIX<n> n=8,10,12,14,16	Converts float to FIX<n>.
	FIX<n>_todouble	FIX<n> n=8,10,12,14,16	double	Converts FIX<n> to double.
	FIX<n>_fromdouble	double	FIX<n> n=8,10,12,14,16	Converts double to FIX<n>.

If the result of operation is outside the range of the data type, its value is not guaranteed.

The functions declared are listed in table 8.

Table 8. List of Functions

Category	Name	Parameter Type	Return Type	Description
Multiplication	R_FIX<n>_mul_int16	FIX<n> n=8,10,12,14,16	FIX<n> n=8,10,12,14,16	Computes multiplication of fixed-point data.
	R_FIX_mul32_int16	int16_t	int32_t	Computes multiplication of 16-bit integer data.
Division	R_FIX_div_int16	uint16_t	uint16_t	Computes division of unsigned 16-bit integer data
Sine	R_FIX_sin_int16	FIX12	FIX12	Computes sine of fixed-oint data (radian)
Cosine	R_FIX_cos_int16	FIX12	FIX12	Computes cosine of fixed-point data (radian).
Arctangent	R_FIX_atan_int16	FIX<n> n=16, 24, 29	FIX<n> n=16, 24, 29	Computes the principal radian value of arctangent of fixed-point data.
	R_FIX_atan2_int16	FIX<n> n=16, 24, 29	FIX<n> n=16, 24, 29	Computes the principal radian value of arctangent of y/x.
Square Root	R_FIX_sqrt_int16	uint16_t	uint16_t	Computes square root of fixed-point data
	R_FIX_sqrt2_int16	int16_t	uint16_t	Computes square root of $x^2 + y^2$
Limit	R_FIX_limit_int16	int16_t	int16_t	Computes limited value
	R_FIX_ulimit_int16	int16_t	uint16_t	Computes limited value greater than or equal to 0.

If the result of operation is outside the range of the data type, its value is not guaranteed.

3.2 Description of Functions

3.2.1 Conversion (macro)

(1) Conversion from float type to fixed-point

[Interface]FIX<n> FIX<n>_fromfloat(float x)

n: 8,10,12,14,16

[Description] Converts float type data to fixed-point type.

[Header] "r_fixmath.h"

[Return Value] Result of conversion

[Parameters] x: Source of conversion

[Example]

```
#include "r_fixmath.h"
float x;
FIX12 ret;

ret = FIX12_fromfloat(x);
```

(2) Conversion from double type to fixed-point

[Interface]FIX<n> FIX<n>_fromdouble(double x)

n: 8,10,12,14,16

[Description] Converts double type data to fixed-point type.

[Header] "r_fixmath.h"

[Return Value] Result of conversion

[Parameters] x: Source of conversion

[Example]

```
#include "r_fixmath.h"
double x;
FIX12 ret;

ret = FIX12_fromdouble(x);
```

(3) Conversion from fixed-point type to float

[Interface]float FIX<n>_tofloat(FIX<n> x)

n: 8,10,12,14,16

[Description] Converts fixed-point data to float.

[Header] "r_fixmath.h"

[Return Value] Result of conversion

[Parameters] x: Source of conversion

[Example]

```
#include "r_fixmath.h"
FIX12 x;
float ret;

ret = FIX12_tofloat(x);
```

(4) Conversion from fixed-point type to double

[Interface]double FIX<n>_todouble(FIX<n> x)

n: 8,10,12,14,16

[Description] Converts fixed-point data to double.

[Header] "r_fixmath.h"

[Return Value] Result of conversion

[Parameters] x: Source of conversion

[Example]

```
#include "fixmath.h"
FIX12 x;
double ret;

ret = FIX12_todouble(x);
```

3.2.2 Multiplication

(1) Multiplication of fixed-point data

[Interface]FIX<n> R_FIX<n>_mul_int16(FIX<n> x, FIX<n> y)

n: 8,10,12,14,16

[Description] Computes the multiplication of two fixed-point data of FIX<n> type. 32-bit intermediate result is used. Supposing fraction part both of x and y is n-bit, computes the product of two fixed-point data and the values of x and y are multiplied as long data, and shifted n bits to the right.

[Header] "r_fixmath.h"

[Return Value] Result of multiplication

[Parameters] x: Fixed-point data.
y: Fixed-point data

[Example]

```
#include "r_fixmath.h"
FIX12 x, y, ret;

ret = R_FIX12_mul_int16(x, y);
```

(2) Multiplication of 16-bit integer data

[Interface]int32_t R_FIX_mul32_int16(int16_t x, int16_t y)

[Description] Computes the multiplication of two 16-bit integer data and computes the product of two data. The result is 32-bit integer data.

[Header] "r_fixmath.h"

[Return Value] 32-bit integer result of multiplication

[Parameters] x: 16-bit integer data.
y: 16-bit integer data

[Example]

```
#include "r_fixmath.h"
int16_t x, y ;
Int32_t ret;

ret = R_FIX_mul32_int16(x, y);
```

3.2.3 Division

[Interface]uint16_t R_FIX_div_int16(uint16_t a, uint16_t b)

[Description] Computes the value (a * 65536U) / b and returns the quotient.

[Header] "r_fixmath.h"

[Return Value] The quotient of division

[Parameters] x: unsigned 16-bit integer data.
y: unsigned 16-bit integer data

[Example]

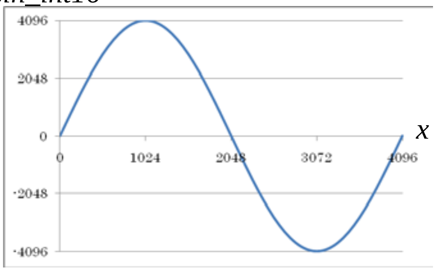
```
#include "r_fixmath.h"
uint16_t x, y, ret;

ret = R_FIX_div_int16(x, y);
```

3.2.4 Sine Function

[Interface]int16_t R_FIX_sin_int16 (int16_t x)

[Description] Computes the sine function of FIX12 fixed-point data (radian value).
For given input “x”, computes 4096 * *sin* (2π * x / 4096)

R_FIX_sin_int16



[Header] "r_fixmath.h"

[Return Value] Result of sine in the FIX12 fixed-point data.

[Parameters] x: Fixed-point data (radian)

[Example]

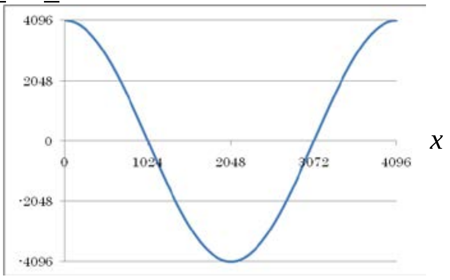
```
#include "r_fixmath.h"
FIX12 x, ret;

ret= R_FIX_sin_int16 (x);
```

3.2.5 Cosine Function

[Interface]int16_t R_FIX_cos_int16(int16_t x)

[Description] Computes the cosine function of FIX12 fixed-point data (radian value).
For given input "x", computes 4096 * *cos* (2π * x / 4096)

R_FIX_cos_int16



[Header] "r_fixmath.h"

[Return Value] Result of cosine in the FIX12 fixed-point data.

[Parameters] x: Fixed-point data (radian)

[Example]

```
#include "r_fixmath.h"
FIX12 x, ret;

ret = R_FIX_cos_int16(x);
```

3.2.6 Arc tangent Function

[Interface]int16 R_FIX_atan_int16(int16_t x)

[Description] Computes the principal of arc tangent. The result is radian value.

[Header] "r_fixmath.h"

[Return Value] Result of arc tangent (in radian),
where 0 <= R_FIX_atan_int16(x) < 0x2000 (corresponding to pi/4).

[Parameters] x: integer, where 0 <= x <= 255

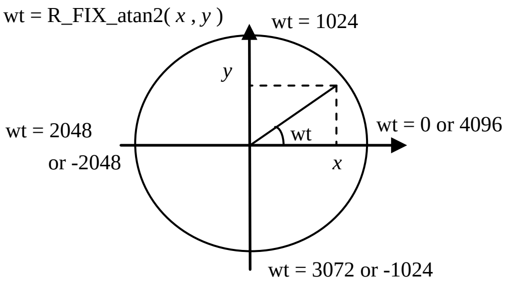
[Example]

```
#include "r_fixmath.h"
Int16_t x, ret;

   ret = R_FIX_atan_int16(x);
```

3.2.7 Arc tangent Function of two variables

[Interface]int16 R_FIX_atan2_int16(int16_t x, int16_t y)

[Description] Computes the principal value of the arc tangent of y/x. The result is radian value.



[Header] "r_fixmath.h"

[Return Value] Result of arctangent (in radian),
where -2048 <= R_FIX_atan2_int16(x, y) <= 2048 (corresponding to pi).
The following are the return value of singular point:

x	y	Return value
0	+	1024
0	−	-1024
0	0	512
+	0	0
−	0	-2048

[Parameters] x: integer

[Example]

```
#include "r_fixmath.h"
Int16_t x, y, ret;

   ret = R_FIX_atan2_int16(x, y);
```

3.2.8 Square Root Function

[Interface]uint16_t R_FIX_sqrt_int16(uint16_t x)

[Description] Computes the square root of FIX12 fixed-point data “ax”.

[Header] "fixmath.h"

[Return Value] Result of square root of ax in FIX12 format. Its range is [0, 16383].

[Parameters] x: FIX12 fixed-point data.

[Example]

```
#include "fixmath.h"
FIX12 x, ret;

ret = (FIX12)R_FIX_sqrt_int16((uint16_t)x);
```

3.2.9 Square Root of sum of squares

[Interface]uint16_t R_FIX_sqrt2_int16(int16_t x, int16_t y)

n: 1~31

[Description] Computes the square root of $x^2 + y^2$.

[Header] "fixmath.h"

[Return Value] Result of square root of ax in FIX12 format. Its range is [0, 16383].

[Parameters] x: FIX12 fixed-point data.
y: FIX12 fixed-point data.

[Example]

```
#include "fixmath.h"
FIX12 x, y, ret;

ret = (FIX12)R_FIX_sqrt2_int16(x, y);
```

3.2.10 Limit Function

(1) Limit function

[Interface] `int16_t R_FIX_limit_int16(int16_t x, uint16_t limit)`

[Description] Computes the limited value of given data “x”.

[Header] "fixmath.h"

[Return Value] Result of the limited value. Its range is [-limit, limit].

[Parameters] x: 16-bit integer data to be limited.
limit: limit value at positive side.

[Example]

```
#include "fixmath.h"
int16_t x, ret;

ret = R_FIX_limit_int16(x, 4096U);
```

(2) Limit function greater than equal to zero

[Interface] `uint16_t R_FIX_ulimit_int16(int16_t x, uint16_t limit)`

[Description] Computes the limited value of given data “x”. The result is greater than equal to 0.

[Header] "fixmath.h"

[Return Value] Result of the limited value. Its range is [0, limit].

[Parameters] x: 16-bit integer data to be limited.
limit: limit value at positive side.

[Example]

```
#include "fixmath.h"
int16_t x;
uint16_t ret;

ret = R_FIX_ulimit_int16 (x, 4096U);
```

4. Specification of Motor Fixed-point Library

4.1 ”r_dsp.h”

This header file defines types and functions for motor control and fix point operations.

The motor control functions declared are listed in table 9.

Table 9. List of Functions

Category	Name	Parameter Type	Return Type	Description
Clarke	R_motor_uvw2ab_int16	int16_t u, int16_t v, int16_t w, int16_t *a, int16_t *b	None	3-axis stationary frame to 2-axis stationary frame transform
Clarke	R_motor_uw2ab_int16	int16_t u, int16_t w, int16_t *a, int16_t *b	None	3-axis stationary frame to 2-axis stationary frame transform. (Equation used when v =1 - u - w is considered)
Clarke	R_motor_uv2ab_int16	int16_t u, int16_t v, int16_t *a, int16_t *b	None	3-axis stationary frame to 2-axis stationary frame transform (Equation used when w =1 - u - v is considered)
Inverse Clarke	R_motor_ab2uvw_int16	int16_t a, int16_t b, int16_t *u, int16_t *v, int16_t *w	None	2-axis stationary frame to 3-axis stationary frame transform
Park	R_motor_ab2dq_int16	int16_t a, int16_t b, int16_t wt, int16_t *d, int16_t *q	None	2-axis stationary frame to 2-axis rotating frame transform
Inverse Park	R_motor_dq2ab_int16	int16_t d, int16_t q, int16_t wt, int16_t *a, int16_t *b	None	2-axis rotating frame to 2-axis stationary frame transform
Coordinate Transform	R_motor_xy2ra_int16*	int16_t x, int16_t y, int16_t *r, int16_t *a	None	Orthogonal coordinate to rotation coordinate transform.
Coordinate Transform	R_motor_ra2xy_int16*	int16_t r, int16_t a, int16_t *x, int16_t *y	None	Rotation coordinate to orthogonal coordinate transform
PID	R_motor_PI_int16	int16_t err, int16_t max, int16_t kp, int16_t ki, int32_t *integ	None	PI Controller

4.2 Function Specifications

4.2.1 R_motor_uvw2ab_int16

[Interface] void R_motor_uvw2ab_int16(int16_t u, int16_t v, int16_t w, int16_t *a, int16_t *b)

[Description] 3-axis stationary frame to 2-axis stationary frame transform. The equation is as follows.

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

[Header] “r_dsp.h”

[Return Value] None.

[Parameters] u: integer, where -16384 <= u <= 16383
v: integer, where -16384 <= v <= 16383
w: integer, where -16384 <= w <= 16383
*a: integer, where |*a| ＜ sqrt(2/3) * 32768
*b: integer, where |*b| ＜ sqrt(1/2) * 32768

[Example]

```
#include "r_dsp.h"
int16_t u, v, w, a, b;

R_ motor_uvw2ab_int16(u, v, w, &a, &b);
```

4.2.2 R_motor_uw2ab_int16

[Interface] void R_motor_uw2ab_int16 (int16_t u, int16_t w, int16_t *a, int16_t *b)

[Description]3-axis stationary frame to 2-axis stationary frame transform. The equation is as follows.

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{3/2} & 0 \\ -\sqrt{2}/2 & -\sqrt{2} \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix}$$

[Header] “r_dsp.h”

[Return Value] None.

[Parameters] u: integer, where -10922 <= u <= 10922
w: integer, where -10922 <= w <= 10922
*a: integer, where |*a| ＜ sqrt(2/3) * 32768
*b: integer, where |*b| ＜ sqrt(1/2) * 32768

[Example]

```
#include "r_dsp.h"
int16_t u, w, a, b;

R_motor_uw2ab_int16(u, w, &a, &b);
```

4.2.3 R_motor_uv2ab_int16

[Interface] void R_motor_uv2ab_int16 (int16_t u, int16_t v, int16_t *a, int16_t *b)

[Description]3-axis stationary frame to 2-axis stationary frame transform. The equation is as follows.

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{3/2} & 0 \\ \sqrt{2}/2 & \sqrt{2} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

[Header] “r_dsp.h”

[Return Value] None.

[Parameters] u: integer, where -10922 <= u <= 10922
v: integer, where -10922 <= v <= 10922
*a: integer, where |*a| ＜ sqrt(2/3) * 32768
*b: integer, where |*b| ＜ sqrt(1/2) * 32768

[Example]

```
#include "r_dsp.h"
int16_t u, v, a, b;

R_motor_uv2ab_int16(u, v, &a, &b);
```

4.2.4 R_motor_ab2uvw_int16

[Interface] void R_motor_ab2uvw _int16 (int16_t a, int16_t b, int16_t *u, int16_t *v, int16_t *w)

[Description]2-axis stationary frame to 3-axis stationary frame transform. The equation is as follows.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 \\ -1/2 & \sqrt{3}/2 \\ -1/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

[Header] “r_dsp.h”

[Return Value] None.

[Parameters] a: integer, where -32768 <= a <= 32767
b: integer, where -32768 <= b <= 32767
*u: integer, where |*u| ＜ sqrt(2/3) * 32768
*v: integer, where |*v| ＜ sqrt(1/2) * 32768
*w: integer, where |*w| < sqrt(1/2) * 32768

[Example]

```
#include "r_dsp.h"
int16_t u, v, w, a, b;

R_motor_ab2uvw_int16(a, b, &u, &v, &w);
```

4.2.5 R_motor_ab2dq_int16

[Interface] void R_motor_ab2dq_int16 (int16_t a, int16_t b, int16_t wt, int16_t *d, int16_t *q)

[Description]2-axis stationary frame to 2-axis rotating frame transform. The equation is as follows.

$$\begin{bmatrix} d \\ q \end{bmatrix} = \begin{bmatrix} \cos\omega t & \sin\omega t \\ -\sin\omega t & \cos\omega t \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

[Header] “r_dsp.h”

[Return Value] None.

[Parameters] a: integer, where -23168 <= a <= 23168
b: integer, where -23168 <= b <= 23168
wt: integer, where -32768 <= wt <= 32767, wt is a parameter that performs arithmetic operations $2\pi = 4096$.
*d: integer, where -32768 <= *d <= 32767
*q: integer, where -32768 <= *q <= 32767

[Example]

```
#include "r_dsp.h"
int16_t a, b, wt, d, q;

R_motor_ab2dq_int16(a, b, wt, &d, &q);
```

4.2.6 R_motor_dq2ab_int16

[Interface] void R_motor_dq2ab_int16 (int16_t d, int16_t q, int16_t wt, int16_t *a, int16_t *b)

[Description] 2-axis rotating frame to 2-axis stationary frame transform. The equation is as follows.

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \cos \omega t & -\sin \omega t \\ \sin \omega t & \cos \omega t \end{bmatrix} \begin{bmatrix} d \\ q \end{bmatrix}$$

[Header] "r_dsp.h"

[Return Value] None.

[Parameters]
- d: integer, where -16384 <= d <= 16383
- q: integer, where -16384 <= q <= 16383
- wt: integer, where -32768 <= wt <= 32767, wt is a parameter that performs arithmetic operations $2\pi = 4096$.
- *a: integer, where -32768 <= *a <= 32767
- *b: integer, where -32768 <= *b <= 32767

[Example]

```
#include "r_dsp.h"
int16_t a, b, wt, d, q;

R_motor_dq2ab_int16(d, q, wt, &a, &b);
```

4.2.7 R_motor_xy2ra_int16

[Interface] void R_motor_xy2ab_int16 (int16_t x, int16_t y, int16_t *r, int16_t *a)

[Description] Orthogonal coordinate to rotation coordinate transform. The equation is as follows.

$$\begin{cases} r = \sqrt{x^2 + y^2} \\ \alpha = \tan^{-1} y / x \end{cases}$$

[Header] “r_dsp.h”

[Return Value] None.

[Parameters]
x: integer, where -16384 <= x <= 16383
y: integer, where -16384 <= y <= 16383
*r: integer, where -32768 <= *r <= 32767
*a: integer, where -2048 <= *a <= 2047, *a is a parameter that performs arithmetic operations 2 π =4096.

[Example]

```
#include "r_dsp.h"
int16_t x, y, r, a;

R_motor_xy2ab_int16(x, y, &r, &a);
```

4.2.8 R_motor_ra2xy_int16

[Interface] void R_motor_ra2xy_int16 (int16_t r, int16_t a, int16_t *x, int16_t *y)

[Description] Rotation coordinate to orthogonal coordinate transform. The equation is as follows.

$$\begin{cases} x = r\cos\alpha \\ y = r\sin\alpha \end{cases}$$

[Header] "r_dsp.h"

[Return Value] None.

[Parameters] r: integer, where -32768 <= r <= 32767
a: integer, where -32768 <= a <= 32767
*x: integer, where -32768 <= *x <= 32767
*y: integer, where -32768 <= *y <= 32767

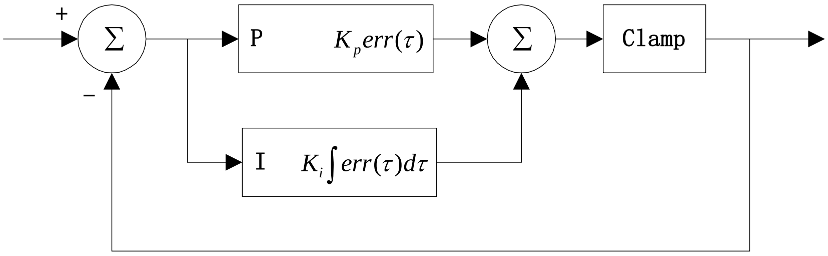
[Example]

```
#include "r_dsp.h"
int16_t x, y, r, a;

R_motor_ra2xy_int16(r, a, &x, &y);
```

4.2.9 R_motor_PI_int16

[Interface] int16_t void R_motor_PI_int16 (int16_t err, int16_t max, int16_t kp, int16_t ki, int16_t *integ)

[Description] The block diagram of PI controller.



[Header] "r_dsp.h"

[Return Value] None.

[Parameters] err: integer, where -32768 <= err <= 32767
max: integer, where 0 <= max <= 32767
kp: integer, where -32768 <= kp <= 32767
ki: integer, where -32768 <= ki <= 32767
*integ: integer, where -max <= *integ <= max

[Example]

```
#include "r_dsp.h"
int16_t err, max, kp, ki, integral;

R_motor_PI_int16(err, max, kp, ki, &integral);
```

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jun.15.12	—	First edition issued
1.01	Jun.10.13	—	Workspace’s DSC Library updated to accommodate all RL78/G14 devices

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 manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

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 one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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: +44-1628-651-700, Fax: +44-1628-651-804

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.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

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 2.2