RX Family

Fixed-point Library

Introduction

This document describes the usage of RX Family Fixed-point Library.

Target Device

RX Family
Content

1. Fixed-point Library .................................................................................................................................. 4
  1.1 Overview .............................................................................................................................................. 4
  1.1 Format of Fixed-point Data .................................................................................................................. 4
  1.3 Library Files ......................................................................................................................................... 4
  1.4 Example of Usage ............................................................................................................................... 5
  1.5 Notes on Library Usage ....................................................................................................................... 6
  2. Specification of Fixed-point Library ....................................................................................................... 7
  2.1 “fixmath.h” ........................................................................................................................................... 7
  2.2 Description of Functions ...................................................................................................................... 12
  2.2.1 Multiplication (macro) ..................................................................................................................... 12
  2.2.2 Division (macro) .............................................................................................................................. 12
  2.2.3 Conversion (macro) ........................................................................................................................ 13
  2.2.4 Multiplication ................................................................................................................................... 15
  2.2.5 Sine Function .................................................................................................................................. 15
  2.2.6 Cosine Function .............................................................................................................................. 16
  2.2.7 Arctangent Function ....................................................................................................................... 16
  2.2.8 Square Root Function .................................................................................................................... 17
  2.2.9 Multiplication (fraction part) (macro) ............................................................................................. 17
  2.2.10 Multiplication (saturated) (macro) ................................................................................................. 18
  2.2.11 Multiplication (FIX-type) (macro) .................................................................................................. 18
  2.2.12 Multiplication (FIX-type) ................................................................................................................ 19
  2.2.13 Multiplication (fraction part, FIX-type) (macro) ............................................................................. 19
  2.2.14 Multiplication (fraction part, FIX-type) (macro) ............................................................................. 20
  2.2.15 Multiplication (saturated, FIX-type) (macro) ................................................................................. 20
  2.2.16 Multiplication (saturated, FIX-type) .............................................................................................. 21
  3. Performance and Precision ................................................................................................................. 22
3.1 Evaluation Condition ................................................................. 22
3.2 Execution Cycles ......................................................................... 22
3.3 Precision ..................................................................................... 22
1. Fixed-point Library

1.1 Overview

This library provides real-number operations using fixed-point format for RX Family.

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

This library supports the following functions for fixed-point type with 16, 24, or 29 fraction bits.

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

Use 16-bit or 24-bit depending on the required precision of your application. 29-bit precision is supported as the most precise type which can represent the input range of trigonometric functions (-π~+π).

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. For this reason, this library supports multiplication, division, conversion for all the precision (from 1 to 31) of fixed-point type.

1.1 Format of Fixed-point Data

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

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>n</th>
<th>n-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sign (1bit) Integer Part (32 - (1+n) bits) Fraction Part (n bits)
```

Figure 1. Fixed-point Data Format

According to the number of bits in fraction part, types from FIX1 to FIX31 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.

1.3 Library Files

The following include file and library files are provided.

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

---

1 Fixed-point format represents a real number by assuming a decimal point at some fixed bit position.
### Table 1. Include File for Fixed-point Library

<table>
<thead>
<tr>
<th>Library Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed-point library</td>
<td>Implements fixed-point operations</td>
</tr>
<tr>
<td>“fixmath.h”</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2. Fixed-point Libraries

<table>
<thead>
<tr>
<th>Library name</th>
<th>Compiler Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXfixmath_big.lib</td>
<td>RX big</td>
</tr>
<tr>
<td>RXfixmath_little.lib</td>
<td>RX little</td>
</tr>
</tbody>
</table>

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

```
#include directory: fixmath.h
library directory: RXfixmath_big.lib
RXfixmath_little.lib
```

![Figure 2. Sample Configuration](image)

### 1.4 Example of Usage

The following example shows a program using FIX16 operation and how to specify the library undefined.

```
#include <stdio.h>
#include "fixmath.h" // Necessary when using
// fixed-point library

void main()
{
    float r_flt;
    FIX16 d_fix16, r_fix16;
    d_fix16 = FIX16_fromfloat(3.14f); // convert float type constant
    // to FIX16
    r_fix16 = FIX16_sin(d_fix16); // computes sin
    r_flt = FIX16_tofloat(r_fix16); // Convert back for printing
    printf("%f\n", r_flt);
}
```
Select [RX Standard Toolchain …] in [build] menu. In the dialog box [RX Standard Toolchain …], select tab [optimizing linker], and specify "input" for [category] and specify "library file" for [option item], and click [add] button. In the dialog box [Add library file], select the library to be linked.

![Figure 3. Specifying library](image)

1.5 Notes on Library Usage

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

2.1 "fixmath.h"

This header file defines types and functions for fixed-point operations. Table 3 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 31. The number in the function or macro name corresponds to the number in the type name.

<table>
<thead>
<tr>
<th>Type</th>
<th>Supported functions and macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIX1-FIX31 (except FIX16, FIX24 or FIX29)</td>
<td>FIX(n)_mul_short, FIX(n)_mul, FIX(n)_div, FIX(n)_tofloat, FIX(n)_fromflop, FIX(n)_todouble, FIX(n)_fromdouble, FIX(n)_mul_frac, FIX(n)_mul_sat</td>
</tr>
<tr>
<td>FIX16, FIX24, FIX29</td>
<td>FIX(n)_mul_short, FIX(n)_mul, FIX(n)_div, FIX(n)_tofloat, FIX(n)_fromflop, FIX(n)_todouble, FIX(n)_fromdouble, FIX(n)_mul_frac, FIX(n)_mul_sat, FIX(n)_sin, FIX(n)_cos, FIX(n)_atan, FIX(n)_sqrt</td>
</tr>
<tr>
<td>FIX</td>
<td>FIX(n)_mul_scale(n), FIX(n)_mul_frac_scale(n), FIX(n)_mul_sat_scale(n), FIX(n)_mul_scale, FIX(n)_mul_frac_scale, FIX(n)_mul_sat_scale</td>
</tr>
</tbody>
</table>
These types are defined as long type.

When the operands and the result of an operation are the same type (\(\text{FIX<n>}\)), use the function corresponding to that type. Otherwise, use a function corresponding to the generic fixed-point type \(\text{FIX}\).

**[Hints on Fixed-point Library Usage]**

1. Select one of the standard fixed-point type (FIX16 or FIX24) 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 FIX16 to FIX24

   ```c
   FIX16 x, FIX24 y;
   x=y>>8;
   ```

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 FIX16.

   ```c
   FIX16 x, y, z;
   z=x+y;
   ```

5. Conversion between floating-point types and fixed-point types should be done only when required. Unnecessary conversions reduce 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.

   ```c
   FIX16 x;
   x=FIX16_fromfloat(3.14f);
   ```
Table 4 shows the representations and ranges of fixed types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (byte)</th>
<th>Alignment (byte)</th>
<th>Sign</th>
<th>Range</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIX1</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-1073741824.0)</td>
<td>2**(4)(1073741823.5)</td>
</tr>
<tr>
<td>FIX2</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-536870912.0)</td>
<td>2**(4)(536870911.75)</td>
</tr>
<tr>
<td>FIX3</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-268435456.0)</td>
<td>2**(4)(268435455.875)</td>
</tr>
<tr>
<td>FIX4</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-134217728.0)</td>
<td>2**(4)(134217727.9375)</td>
</tr>
<tr>
<td>FIX5</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-67108864.0)</td>
<td>2**(4)(67108863.96875)</td>
</tr>
<tr>
<td>FIX6</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-33554432.0)</td>
<td>2**(4)(33554431.984375)</td>
</tr>
<tr>
<td>FIX7</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-16777216.0)</td>
<td>2**(4)(16777215.9921875)</td>
</tr>
<tr>
<td>FIX8</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-8388608.0)</td>
<td>2**(4)(8388607.99609375)</td>
</tr>
<tr>
<td>FIX9</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-4194304.0)</td>
<td>2**(4)(4194303.998046875)</td>
</tr>
<tr>
<td>FIX10</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-2097152.0)</td>
<td>2**(4)(2097151.999234375)</td>
</tr>
<tr>
<td>FIX11</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-1048576.0)</td>
<td>2**(4)(1048575.9995117875)</td>
</tr>
<tr>
<td>FIX12</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-524288.0)</td>
<td>2**(4)(524287.999755859375)</td>
</tr>
<tr>
<td>FIX13</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-262144.0)</td>
<td>2**(4)(262143.9998779296875)</td>
</tr>
<tr>
<td>FIX14</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-131072.0)</td>
<td>2**(4)(131071.99993896484375)</td>
</tr>
<tr>
<td>FIX15</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-65536.0)</td>
<td>2**(4)(65535.99996482421875)</td>
</tr>
<tr>
<td>FIX16</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-32768.0)</td>
<td>2**(4)(32767.9999847412109375)</td>
</tr>
<tr>
<td>FIX17</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-16384.0)</td>
<td>2**(4)(16383.999999237060546875)</td>
</tr>
<tr>
<td>FIX18</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-8192.0)</td>
<td>2**(4)(8191.9999996185302734375)</td>
</tr>
<tr>
<td>FIX19</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-4096.0)</td>
<td>2**(4)(4095.9999998026513671875)</td>
</tr>
<tr>
<td>FIX20</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-2048.0)</td>
<td>2**(4)(2047.9999999432568359375)</td>
</tr>
<tr>
<td>FIX21</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-1024.0)</td>
<td>2**(4)(1023.9999999523162841796875)</td>
</tr>
<tr>
<td>FIX22</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-512.0)</td>
<td>2**(4)(511.9999997615841208984375)</td>
</tr>
<tr>
<td>FIX23</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-256.0)</td>
<td>2**(4)(255.99999988079070144921875)</td>
</tr>
<tr>
<td>FIX24</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-128.0)</td>
<td>2**(4)(127.999999903955355224609375)</td>
</tr>
<tr>
<td>FIX25</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-64.0)</td>
<td>2**(4)(63.999999701976776123046875)</td>
</tr>
<tr>
<td>FIX26</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-32.0)</td>
<td>2**(4)(31.9999998509883880615234375)</td>
</tr>
<tr>
<td>FIX27</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-16.0)</td>
<td>2**(4)(15.99999992549419403076171875)</td>
</tr>
<tr>
<td>FIX28</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-8.0)</td>
<td>2**(4)(7.999999962747097015380859375)</td>
</tr>
<tr>
<td>FIX29</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-4.0)</td>
<td>2**(4)(3.9999999813735485076904296875)</td>
</tr>
<tr>
<td>FIX30</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-2.0)</td>
<td>2**(4)(1.99999999068677425384521484375)</td>
</tr>
<tr>
<td>FIX31</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>-2**(4)</td>
<td>-2**(4)(-1.0)</td>
<td>2**(4)(0.999999995343387126922607421875)</td>
</tr>
</tbody>
</table>

Table 4. Representation and Ranges of Fixed Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (byte)</th>
<th>Alignment (byte)</th>
<th>Sign</th>
<th>Range</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIX</td>
<td>4</td>
<td>4</td>
<td>signed</td>
<td>Represents one of above ranges, depending on the number of fraction bits assumed.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The macros defined are listed in table 5.

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Parameter Type</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication</td>
<td>FIX&lt;n&gt;_mul_short</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>Computes multiplication of fixed-point data. (If the multiplication result exceeds 32-bits, the result is not guaranteed).</td>
</tr>
<tr>
<td></td>
<td>FIX&lt;n&gt;_mul_frac</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>Computes fractional part f of the fixed-point multiplication (0&lt;=f&lt;1.0).</td>
</tr>
<tr>
<td>Division</td>
<td>FIX&lt;n&gt;_div</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>Computes division of fixed-point data.</td>
</tr>
<tr>
<td>Conversion</td>
<td>FIX&lt;n&gt;_tofloat</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>float</td>
<td>Converts FIX&lt;n&gt; to float.</td>
</tr>
<tr>
<td></td>
<td>FIX&lt;n&gt;_fromfloat</td>
<td>float</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>Converts float to FIX&lt;n&gt;.</td>
</tr>
<tr>
<td></td>
<td>FIX&lt;n&gt;_todouble</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>double</td>
<td>Converts FIX&lt;n&gt; to double.</td>
</tr>
<tr>
<td></td>
<td>FIX&lt;n&gt;_fromdouble</td>
<td>double</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>Converts double to FIX&lt;n&gt;.</td>
</tr>
<tr>
<td>Multiplication of generic fixed-point</td>
<td>FIX_mul_scale&lt;n&gt;</td>
<td>FIX</td>
<td>FIX</td>
<td>Computes multiplication of generic fixed-point data.</td>
</tr>
<tr>
<td></td>
<td>FIX_mul_frac_scale&lt;n&gt;</td>
<td>FIX</td>
<td>FIX</td>
<td>Computes fractional part f of the generic fixed-point multiplication (0&lt;=f&lt;1.0).</td>
</tr>
<tr>
<td></td>
<td>FIX_mul_sat_scale&lt;n&gt;</td>
<td>FIX</td>
<td>FIX</td>
<td>Computes multiplication of generic fixed-point data. When overflow occurs, the result is maximum or minimum value of the range.</td>
</tr>
<tr>
<td></td>
<td>FIX_mul_frac_scale</td>
<td>FIX</td>
<td>FIX</td>
<td>Computes fractional part f of the generic fixed-point multiplication (0&lt;=f&lt;1.0).</td>
</tr>
</tbody>
</table>

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 6.

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Parameter Type</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication</td>
<td>FIX_mul</td>
<td>FIX</td>
<td>FIX</td>
<td>Computes multiplication of fixed-point data.</td>
</tr>
<tr>
<td></td>
<td>FIX&lt;n&gt;_mul_sat</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>FIX&lt;n&gt; n=1~31</td>
<td>Computes multiplication of fixed-point data. When overflow occurs, the result is maximum or minimum value of the range.</td>
</tr>
<tr>
<td>Sine</td>
<td>FIX&lt;n&gt;_sin</td>
<td>FIX&lt;n&gt; n=16, 24, 29</td>
<td>FIX&lt;n&gt; n=16, 24, 29</td>
<td>Computes sine of fixed-point data (radian)</td>
</tr>
<tr>
<td>Cosine</td>
<td>FIX&lt;n&gt;_cos</td>
<td>FIX&lt;n&gt; n=16, 24, 29</td>
<td>FIX&lt;n&gt; n=16, 24, 29</td>
<td>Computes cosine of fixed-point data (radian).</td>
</tr>
<tr>
<td>Arctangent</td>
<td>FIX&lt;n&gt;_atan</td>
<td>FIX&lt;n&gt; n=16, 24, 29</td>
<td>FIX&lt;n&gt; n=16, 24, 29</td>
<td>Computes radian value of arctangent of fixed-point data.</td>
</tr>
<tr>
<td>Square Root</td>
<td>FIX&lt;n&gt;_sqrt</td>
<td>FIX&lt;n&gt; n=16, 24, 29</td>
<td>FIX&lt;n&gt; n=16, 24, 29</td>
<td>Computes square root of fixed-point data</td>
</tr>
<tr>
<td>Multiplication of generic fixed-point</td>
<td>FIX_mul_scale</td>
<td>FIX</td>
<td>FIX</td>
<td>Computes multiplication of generic fixed-point data.</td>
</tr>
<tr>
<td></td>
<td>FIX_mul_sat_scale</td>
<td>FIX</td>
<td>FIX</td>
<td>Computes multiplication of generic fixed-point data. When overflow occurs, the result is maximum or minimum value of the range.</td>
</tr>
</tbody>
</table>

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

2.2.1 Multiplication (macro)

- **Interface**: \texttt{FIX\textless{}n\textgreater{} \_mul\_short\textless{}FIX\textless{}n\textgreater{} \ x, \ FIX\textless{}n\textgreater{} \ y)}
  - n: 1-31

- **Description**: Two 32-bit data are multiplied and shifted right by n bits. This computes the multiplication of two fixed-point data of FIX\textless{}n\textgreater{} type.

- **Header**: "fixmath.h"

- **Return Value**: Result of multiplication

- **Parameters**: 
  - x: Fixed-point data.
  - y: Fixed-point data

- **Example**
  ```
  #include "fixmath.h"
  fix16 x, y, ret;
  ret=FIX16\_mul\_short(x, y);
  ```

- **Note**: Short multiplication uses 32-bit integer arithmetic. The result is not guaranteed if the intermediate result exceeds 32 bits.

2.2.2 Division (macro)

- **Interface**: \texttt{FIX\textless{}n\textgreater{} \_div\textless{}FIX\textless{}n\textgreater{} \ x, \ FIX\textless{}n\textgreater{} \ y)}
  - n: 1-31

- **Description**: Computes the quotient of two fixed-point data.

- **Header**: "fixmath.h"

- **Return Value**: Result of division

- **Parameters**: 
  - x: Dividend.
  - y: divisor

- **Example**
  ```
  #include "fixmath.h"
  fix16 x, y, ret;
  ret=FIX16\_div(x, y);
  ```
2.2.3 Conversion (macro)

(1) Conversion from float type to fixed-point

[Interface]  
FIX<n> FIX<n>_fromfloat(float x)
n: 1~31

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

[Header] "fixmath.h"

[Return Value] Result of conversion

[Parameters] x: Source of conversion

[Example]
#include "fixmath.h"
float x;
FIX16 ret;

    ret=FIX16_fromfloat(x);

(2) Conversion from double type to fixed-point

[Interface]  
FIX<n> FIX<n>_fromdouble(double x)
n: 1~31

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

[Header] "fixmath.h"

[Return Value] Result of conversion

[Parameters] x: Source of conversion

[Example]
#include "fixmath.h"
double x;
FIX16 ret;

    ret=FIX16_fromdouble(x);
(3) Conversion from fixed-point type to float

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

n: 1~31

[Description] Converts fixed-point data to float.

[Header] "fixmath.h"

[Return Value] Result of conversion

[Parameters] x: Source of conversion

[Example]
#include "fixmath.h"
FIX16 x;
float ret;

    ret=FIX16_tofloat(x);

(3) Conversion from fixed-point type to double

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

n: 1~31

[Description] Converts fixed-point data to double.

[Header] "fixmath.h"

[Return Value] Result of conversion

[Parameters] x: Source of conversion

[Example]
#include "fixmath.h"
FIX16 x;
double ret;

    ret=FIX16_todouble(x);
2.2.4 Multiplication

[Interface]  
FIX FIX_mul(FIX x, FIX y, int n)
  
n: 1~31

[Description] Computes the multiplication of two fixed-point data of FIX type. 64-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] "fixmath.h"

[Return Value] Result of multiplication

[Parameters]  
x: Fixed-point data.
y: Fixed-point data
n: bit size of Fraction Part

[Example]  
#include "fixmath.h"
FIX16 x, y, ret;

  ret=(FIX16)FIX_mul((FIX)x, (FIX)y, 16);

2.2.5 Sine Function

[Interface]  
FIX<n> FIX<n>_sin(FIX<n> x)
  
n: 16, 24, 29

[Description] Computes the sine function of FIX<n> fixed-point data (radian value).

[Header] "fixmath.h"

[Return Value] Result of sine.

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

[Example]  
#include "fixmath.h"
FIX16 x, ret;

  ret=FIX16_sin(x);
2.2.6 Cosine Function

[Interface]  FIX<n>  FIX<n>_cos(FIX<n>  x)
             n: 16, 24, 29

[Description] Computes the cosine function of FIX<n> fixed-point data (radian value).

[Header] "fixmath.h"

[Return Value] Result of cosine.

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

[Example]
#include "fixmath.h"
FIX16 x, ret;
ret=FIX16_cos(x);

2.2.7 Arctangent Function

[Interface]  FIX<n>  FIX<n>_atan(FIX<n>  x)
             n: 16, 24, 29

[Description] Computes the arctangent function of FIX<n> fixed-point data. The result is radian value.

[Header] "fixmath.h"

[Return Value] Result of arctangent (in radian).

[Parameters] x: Fixed-point data.

[Example]
#include "fixmath.h"
FIX16 x, ret;
ret=FIX16_atan(x);
2.2.8 Square Root Function

[Interface] \[\text{FIX<n> FIX<n>\_sqrt(FIX<n> x)}\]
\[n: 16, 24, 29\]

[Description] Computes the square root of FIX<n> fixed-point data.

[Header] "fixmath.h"

[Return Value] Result of square root.

[Parameters] x: Fixed-point data.

[Example]

```c
#include "fixmath.h"
FIX16 x, ret;

ret=FIX16\_sqrt(x);
```

2.2.9 Multiplication (fraction part) (macro)

[Interface] \[\text{FIX<n> FIX<n>\_mul\_frac(FIX<n> x, FIX<n> y)}\]
\[n: 1\sim31\]

[Description] Computes the fraction part of the product. The result will always be positive \((0\leq\text{result}<1.0)\).

[Header] "fixmath.h"

[Return Value] Fractional part of the product.

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

[Example]

```c
#include "fixmath.h"
FIX16 x, y, ret;

ret=FIX16\_mul\_frac(x, y);
```
2.2.10 Multiplication (saturated) (macro)

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

n: 1~31

[Description] Computes the product of two fixed-point data. When the result overflows, the return value will be the maximum or minimum value, according to the sign of the result.

[Header] "fixmath.h"

[Return Value] Product of fixed point data.

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

[Example]  
#include "fixmath.h"
FIX16 x, y, ret;

    ret=FIX16_mul_sat(x, y);

2.2.11 Multiplication (FIX-type) (macro)

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

n: 1~31

[Description] Computes the product of two generic fixed-point data. The values of x and y are multiplied as long data, and shifted n bits to the right.

[Header] "fixmath.h"

[Return Value] Product of fixed point data.

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

[Example]  
#include "fixmath.h"
FIX x, y, ret;

    ret=FIX_mul_scale16(x, y);

[Note] If the result cannot be represented in 32-bit, its value is not guaranteed. When multiplying FIX<n1> and FIX<n2> to get FIX<n3> type, shift count is n1+n2-n3, and you can specify this value (1~31) as <n>. 
### 2.2.12 Multiplication (FIX-type)

**Interface**

```c
FIX FIX_mul_scale(FIX x, FIX y, int n)
n: 1~31
```

**Description**

Computes the product of two generic fixed-point data. The values of x and y are multiplied as long data, and shifted n bits to the right.

**Header**

"fixmath.h"

**Return Value**

Product of fixed point data.

**Parameters**

- `x`: Fixed-point data.
- `y`: Fixed-point data.

**Example**

```c
#include "fixmath.h"
FIX x, y, ret;
int n=16;
ret=FIX_mul_scale(x, y, n);
```

**Note**

If the result cannot be represented in 32-bit, its value is not guaranteed. When multiplying `FIX<n_1>` and `FIX<n_2>` to get `FIX<n_3>` type, shift count is `n_1+n_2-n_3`, and you can specify this value (1~31) as `<n>`.

### 2.2.13 Multiplication (fraction part, FIX-type) (macro)

**Interface**

```c
FIX FIX_mul_frac_scale<n>(FIX x, FIX y)
n: 1~31
```

**Description**

Computes the fraction part of the product of two generic fixed-point data. The values of x and y are multiplied as long data, shifted n bits to the right, and lower n-bits are returned.

**Header**

"fixmath.h"

**Return Value**

Product of fixed point data.

**Parameters**

- `x`: Fixed-point data.
- `y`: Fixed-point data.

**Example**

```c
#include "fixmath.h"
FIX x, y, ret;
ret=FIX_mul_frac_scale16(x, y);
```

**Note**

This function can be used to compute the fractional part of `FIX<n>` type by multiplying `FIX<n+d>` type and `FIX<n-d>` type.
2.2.14 Multiplication (fraction part, FIX-type) (macro)

[Interface]  FIX FIX_mul_frac_scale(FIX x, FIX y, int n)
              n: 1~31

[Description] Computes the fraction part of the product of two generic fixed-point data. The values of x and y are
multiplied as long data, shifted n bits to the right, and lower n-bits are returned.

[Header]  "fixmath.h"

[Return Value] Product of fixed point data.

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

[Example]
#include "fixmath.h"
FIX x, y, ret;
int n=16;
ret=FIX_mul_frac_scale(x, y, n);

[Note] This function can be used to compute the fractional part of FIX<n> type by multiplying FIX<n+d>
type and FIX<n-d> type.

2.2.15 Multiplication (saturated, FIX-type) (macro)

[Interface]  FIX FIX_mul_sat_scale<n>(FIX x, FIX y)
              n: 1~31

[Description] Computes the product of two generic fixed-point data. The values of x and y are multiplied as long
data, and shifted n bits to the right. When the result overflows, the return value will be the maximum
or minimum value, according to the sign of the result.

[Header]  "fixmath.h"

[Return Value] Product of fixed point data.

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

[Example]
#include "fixmath.h"
FIX x, y, ret;
ret=FIX_mul_sat_scale16(x, y);

[Note] If the result cannot be represented in 32-bit, its value is not guaranteed. When multiplying FIX<n1> and FIX<n2>
to get FIX<n3> type, shift count is n1+n2-n3, and you can specify this value (1~31) as <n>.
2.2.12 Multiplication (saturated, FIX-type)

[Interface] FIX FIX_mul_sat_scale(FIX x, FIX y, int n)
              n: 1~31

[Description] Computes the product of two generic fixed-point data. The values of x and y are multiplied as long
data, and shifted n bits to the right. When the result overflows, the return value will be the maximum
or minimum value, according to the sign of the result.

[Header] "fixmath.h"

[Return Value] Product of fixed point data.

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

[Example]

#include "fixmath.h"
FIX x, y, ret;
int n=16;

        ret=FIX_mul_sat_scale(x, y, n);

[Note] If the result cannot be represented in 32-bit, its value is not guaranteed. When multiplying FIX<\text{n}_1> and FIX<\text{n}_2> to get FIX<\text{n}_3> type, shift count is \text{n}_1+\text{n}_2-\text{n}_3, and you can specify this value (1~31) as \text{n}. 
3. Performance and Precision

3.1 Evaluation Condition

Compiler: Renesas RX Standard Compiler V.1.0.0.0
Configuration: See table 7 for library configuration.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Compilation Options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cpu</td>
</tr>
<tr>
<td>1</td>
<td>RX600</td>
</tr>
<tr>
<td>2</td>
<td>RX600</td>
</tr>
</tbody>
</table>

3.2 Execution Cycles

The execution cycles of fixed-point mathematical functions are shown in table 8.

<table>
<thead>
<tr>
<th>CPU</th>
<th>Configuration (see table 7)</th>
<th>RX600</th>
<th>RX600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sine</td>
<td>FIX16_sin</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>FIX24_sin</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>FIX29_sin</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>Cosine</td>
<td>FIX16_cos</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>FIX24_cos</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>FIX29_cos</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>Arctangent</td>
<td>FIX16_atan</td>
<td>118</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>FIX24_atan</td>
<td>118</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FIX29_atan</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>Square Root</td>
<td>FIX16_sqrt</td>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>FIX24_sqrt</td>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>FIX29_sqrt</td>
<td>81</td>
<td></td>
</tr>
</tbody>
</table>

[Note] The numbers are in cycles. Measurement may include some error.

3.3 Precision

The maximum error of these mathematics functions is ±2 in the last place except FIX29_sqrt. The precision of FIX29_sqrt is ±3 in the last place.
Website and Support

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

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com
## Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Dec.28.10</td>
<td>-</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Oct.30.12</td>
<td>-</td>
<td>A bug which FIX&lt;n&gt;_atan(-1) returns pai/4 is fixed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The document number was changed from R20AN0228 to R21AN0003</td>
</tr>
</tbody>
</table>
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. 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 you 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” Products: Office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronics 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 implants etc.) or may cause serious property damage (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 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, dispose 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 liability 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.