

Fixed Point

This library has been modified from [cubit](#) library by [influenceth](#) and adjusted to match with other fixed point implementations. This API provides basic some operations for signed fixed point numbers. Fixed point numbers are represented as a struct with a magnitude and a sign.

The magnitude represents the absolute value of the number, and the sign indicates whether the number is positive or negative.

...

```
Copy struct FP8x23{ mag:u32, sign:bool }
```

...

Data types

Orion supports currently these fixed point types:

Data type dtype Q8.23 FP8x23 Q16.16 FP16x16 Q32.32 FP32x32 Q64.64 FP64x64

Fixed Trait

...

```
Copy use orion::numbers::fixed_point::core::FixedTrait;
```

...

Fixed trait defines the operations that can be performed on a fixed point.

function description [fp.new](#) Constructs a new fixed point instance. [fp.new_unscaled](#) Creates a new fixed point instance with the specified unscaled magnitude and sign. [fp.from_felt](#) Creates a new fixed point instance from a felt252 value. [fp.abs](#) Returns the absolute value of the fixed point number. [fp.ceil](#) Returns the smallest integer greater than or equal to the fixed point number. [fp.exp](#) Returns the value of e raised to the power of the fixed point number. [fp.exp2](#) Returns the value of 2 raised to the power of the fixed point number. [fp.floor](#) Returns the largest integer less than or equal to the fixed point number. [fp.ln](#) Returns the natural logarithm of the fixed point number. [fp.log2](#) Returns the base-2 logarithm of the fixed point number. [fp.log10](#) Returns the base-10 logarithm of the fixed point number. [fp.pow](#) Returns the result of raising the fixed point number to the power of another fixed point number. [fp.round](#) Rounds the fixed point number to the nearest whole number. [fp.sqrt](#) Returns the square root of the fixed point number. [fp.acos](#) Returns the arccosine (inverse of cosine) of the fixed point number. [fp.acos_fast](#) Returns the arccosine (inverse of cosine) of the fixed point number faster with LUT. [fp.asin](#) Returns the arcsine (inverse of sine) of the fixed point number. [fp.asin_fast](#) Returns the arcsine (inverse of sine) of the fixed point number faster with LUT. [fp.atan](#) Returns the arctangent (inverse of tangent) of the input fixed point number. [fp.atan_fast](#) Returns the arctangent (inverse of tangent) of the input fixed point number faster with LUT. [fp.cos](#) Returns the cosine of the fixed point number. [fp.cos_fast](#) Returns the cosine of the fixed point number fast with LUT. [fp.sin](#) Returns the sine of the fixed point number. [fp.sin_fast](#) Returns the sine of the fixed point number faster with LUT. [fp.tan](#) Returns the tangent of the fixed point number. [fp.tan_fast](#) Returns the tangent of the fixed point number faster with LUT. [fp.acosh](#) Returns the value of the inverse hyperbolic cosine of the fixed point number. [fp.asinh](#) Returns the value of the inverse hyperbolic sine of the fixed point number. [fp.atanh](#) Returns the value of the inverse hyperbolic tangent of the fixed point number. [fp.cosh](#) Returns the value of the hyperbolic cosine of the fixed point number. [fp.sinh](#) Returns the value of the hyperbolic sine of the fixed point number. [fp.tanh](#) Returns the value of the hyperbolic tangent of the fixed point number. [fp.sign](#) Returns the element-wise indication of the sign of the input fixed point number.

Arithmetic & Comparison operators

FixedType implements arithmetic and comparison traits. This allows you to perform basic arithmetic operations using the associated operators. (+, +=, -, -=, *, *=, /, /=), as well as relational operators (>, >=, <, <=, ==, !=).

Examples

...

```
Copy fn add_fp_example() { // We instantiate two fixed point from here. // a = 1 // b = 2 let a = Fixed::new_unscaled(1, false); let b = Fixed::new_unscaled(2, false);
```

```
// We can add two fixed point as follows. let result = a + b;
```

```
assert(result == Fixed::new_unscaled(3), "invalid result"); }
```

...

...

```
Copy fncompare_fp_example()->bool{ // We instantiate two fixed point from here. // a = 42 // b = -10
leta=Fixed::new_unscaled(42,false); letb=Fixed::new_unscaled(10,true);

// We can compare two fixed point as follows. returna>b; }

    true
```

...

[Previous Numbers](#) [Next fp.new](#)

Last updated2 months ago