The APU2 Math Library

APU2-Math-Library

Version	Details of Change	Author	Date
1.0	Initial Revision	Kien Pham	Nov 09, 2018

Contents

The APU2 Math Library provides basic functionality for developers to design their own imaging-based applications while taking advantage of NXP's massively parallel APEX architecture.

- Fractional multiplication with integer scale
- Fractional multiplication with scale Fixed point value
- Fractional multiplication with scale and round
- Signum
- Inverse square root
- Integer Square Root
- Square root
- ArcSinus
- Centesimal arctangent
- Tangent
- ArcCosinus
- ArcTangent
- Arctangent function with two parameters
- Hyperbolic Sinus
- Hyperbolic Cosine
- Hyperbolic Tangent
- Natural logarithm
- Natural exponent

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Module Documentation

2.1 UserAPI

Functions

- ALWAYS_INLINE vfxp_s1q15 vfxp_catan2 (vfxp_s1q15 y, vfxp_s1q15 x)

  Centesimal arctangent function with two parameters.
- ALWAYS_INLINE vec16s vfxp_qmult_scale (vec16s a, vec16s b, const int scale)

  Fractional multiplication with scale.
- ALWAYS_INLINE vec16s vfxp_qmult_vscale (vec16s a, vec16s b, vec16s scale)

  Fractional multiplication with scale.
- ALWAYS_INLINE vec16s vfxp_qmult_scale_round (vec16s a, vec16s b, const int scale)

  Fractional multiplication with scale and round.
- ALWAYS_INLINE vec16s vfxp_sgn (vec16s a)

  Signum.
- ALWAYS_INLINE vfxp_s2q14 vfxp_rsqrt (vfxp_s1q15 in)

  Inverse square root.
- ALWAYS_INLINE vfxp_s1q15 vfxp_sqrt (vfxp_s1q15 in)

  Square root.
- ALWAYS_INLINE vfxp_u32q0 vfxp_isqrt32 (vfxp_u32q0 x)

  Integer Square Root.
- ALWAYS_INLINE vfxp_s1q15 vfxp_tan (vfxp_s1q15 x)

  Tangent.
- ALWAYS_INLINE vfxp_s2q14 vfxp_asin (vfxp_s1q15 x)

  ArcSinus.
- ALWAYS_INLINE vfxp_s3q13 vfxp_acos (vfxp_s1q15 x)

  ArcCosinus.
- ALWAYS_INLINE vfxp_s1q15 vfxp_atan (vfxp_s1q15 x)

  ArcTangent.
- ALWAYS_INLINE vfxp_s3q13 vfxp_atan2 (vfxp_s1q15 y, vfxp_s1q15 x)

  Arctangent function with two parameters.
- ALWAYS_INLINE vfxp_s2q14 vfxp_sinh (vfxp_s1q15 x)

  Hyperbolic Sinus.
- ALWAYS_INLINE vfxp_s2q14 vfxp_cosh (vfxp_s1q15 x)

Hyperbolic Cosine.

- ALWAYS_INLINE vfxp_s1q15 vfxp_tanh (vfxp_s1q15 x)

  Hyperbolic Tangent.
- ALWAYS_INLINE vfxp_s2q14 vfxp_log (vfxp_s1q15 x)

  Natural logarithm.
- ALWAYS_INLINE vfxp_s3q13 vfxp_exp (vfxp_s1q15 x)

  Natural exponent.

2.1.1 Detailed Description

This is the group of enum, structure and functions needs to be exposed to APEX MATH library user

2.1.2 Function Documentation

2.1.2.1 ALWAYS_INLINE vfxp_s3q13 vfxp_acos (vfxp_s1q15 *x*)

ArcCosinus.

vfxp_acos(x) = V_S2Q14_PI_DIV_2 - vfxp_asin(x)

Parameters

in	*x*	Fixed point representation of the interval [-1, 1)

Returns

The output is between [pi, 0) - Q3.13

Note

Accuracy (ULP) -> Mean: 13.18, Worst: 11.03

2.1.2.2 ALWAYS_INLINE vfxp_s2q14 vfxp_asin (vfxp_s1q15 *x*)

ArcSinus.

The approximation is take from: "The Handbook Of Mathematical Functions" Chapter 4.4.46. arcsin(x) = pi/2 - sqrt(1 - x) ∗ poly(x) poly(x) = a∗x^7 + b∗x^6 + c∗x^5 + d∗x^4 + e∗x^3 + f∗x^2 + g∗x^1 + h a = 0.0012624911 b = 0.0066700901 c = -0.0170881256 d = 0.0308918810 e = -0.0501743046 f = 0.0889789874 g = -0.2145988016 h = 1.5707963050

Parameters

in	*x*	Fixed point representation of the interval [-1, 1)

Returns

The output is between [-pi/2, pi/2) - Q2.14

Note

Accuracy (ULP) -> Mean: 13.31, Worst: 11.23

2.1.2.3 ALWAYS_INLINE vfxp_s1q15 vfxp_atan (vfxp_s1q15 *x*)

ArcTangent.

The functions takes advantage of the arctangent's property of odd symmetry. An absolute value is applied on the input data, the value is then used inside a high precision 5-th degree polynomial specially optimized for [0, 1] absX = abs(x) return poly(absX) poly(x) = a∗x^5 + b∗x^4 + c∗x^3 + d∗x^2 + e∗x + f a = -0.0542819708651922 b = 0.277205766796652 c = -0.463942427831086 d = 0.0290088104185372 e = 0.997378122323713 f = 5.73559304367514e-05

Parameters

in	*x*	Fixed point representation of the interval [-1, 1)

Returns

The output is between [-1, 1) - Q1.15

Note

Accuracy (ULP) -> Mean: 15.56, Worst: 13.20

2.1.2.4 ALWAYS_INLINE vfxp_s3q13 vfxp_atan2 (vfxp_s1q15 *y*, vfxp_s1q15 *x*)

Arctangent function with two parameters.

The function uses catan2.

Parameters

in	*y*	Fixed point representation of the abscissa [-1, 1)
in	*x*	Fixed point representation of the ordinate [-1, 1)

Returns

The output is between [-pi, pi] - Q3.13

Note

Accuracy (ULP) -> Mean: 13.18, Worst: 10.28

2.1.2.5 ALWAYS_INLINE vfxp_s1q15 vfxp_catan2 (vfxp_s1q15 *y*, vfxp_s1q15 *x*)

Centesimal arctangent function with two parameters.

The implementation is based on the CORDIC algorithm. See magatan2 for more details.

Parameters

in	*y*	Fixed point representation of the abscissa [-1, 1)
in	*x*	Fixed point representation of the ordinate [-1, 1)

Returns

The output is between [-0.5, 0.5], which is the equivalent of [-PI, PI].

Note

Accuracy (ULP) -> Mean: 15.90, Worst: 12.97

2.1.2.6 ALWAYS_INLINE vfxp_s2q14 vfxp_cosh (vfxp_s1q15 *x*)

Hyperbolic Cosine.

Polynomial approximation using the famous Taylor decomposition, the degree is 7 and to speed up the processing a Horner scheme is applied. Chebyshev approximations can be used also but the coefficient values are very big in comparison with the input data format. Big issue with precision !

Parameters

in	*x*	Fixed point representation of the interval [0, 1)

Returns

The output is between [1, 1.54) - Q2.14

Note

Accuracy (ULP) -> Mean: 14.33, Worst: 12.82

2.1.2.7 ALWAYS_INLINE vfxp_s3q13 vfxp_exp (vfxp_s1q15 *x*)

Natural exponent.

Computes the natural exponent using an 5-th degree polynomial

Parameters

in	*x*	Fixed point representation of the interval [0, 1)

Returns

The output is between [1, 2.7183) - Q3.13

Note

Accuracy (ULP) -> Mean: 13.36, Worst: 11.59

2.1.2.8 ALWAYS_INLINE vfxp_u32q0 vfxp_isqrt32 (vfxp_u32q0 *x*)

Integer Square Root.

Uses binary search to determine the square root of the given value. A trial multiplication is made whose result is compared to the argument; Depending on if it is greater or lesser, the binary search takes the upper or lower branch

Parameters

in	*x*	Integer value

Returns

floor(sqrt(x))

Note

Integer square root doesn't have an accuracy class

2.1.2.9 ALWAYS_INLINE vfxp_s2q14 vfxp_log (vfxp_s1q15 *x*)

Natural logarithm.

Polynomial approximation using best fit polynomial coefficients with constraints. The degree is 5 and to speed up the processing a Horner scheme is applied. Two multiply-scale operations are rounded to increase precision and it was the optimal number. Chebyshev approximations can be used also but the coefficient values are very big in comparison with the input data format. Big issue with precision !

Parameters

in	*x*	Fixed point representation of the interval [0.5, 1)

Returns

The output is between [-1, 0) - Q2.14

Note

Accuracy (ULP) -> Mean: 15.79, Worst: 14.02

2.1.2.10 ALWAYS_INLINE vec16s vfxp_qmult_scale (vec16s *a,* vec16s *b,* const int *scale*)

Fractional multiplication with scale.

32-bit multiplication with scaling

Parameters

in	*a*	Fixed point value
in	*b*	Fixed point value
in	*scale*	Const Integer

Returns

out = (a ∗ b) >> scale

2.1.2.11 ALWAYS_INLINE vec16s vfxp_qmult_scale_round (vec16s *a,* vec16s *b,* const int *scale*)

Fractional multiplication with scale and round.

32-bit multiplication with scaling and rounding

Parameters

in	*a*	Fixed point value
in	*b*	Fixed point value
in	*scale*	Const Integer

Returns

out = ((a ∗ b) + (1 << scale - 1)) >> scale

2.1.2.12 ALWAYS_INLINE vec16s vfxp_qmult_vscale (vec16s *a,* vec16s *b,* vec16s *scale*)

Fractional multiplication with scale.

32-bit multiplication with scaling

Parameters

in	*a*	Fixed point value
in	*b*	Fixed point value
in	*scale*	Fixed point value

Returns

out = (a ∗ b) >> scale

2.1.2.13 ALWAYS_INLINE vfxp_s2q14 vfxp_rsqrt (vfxp_s1q15 *in*)

Inverse square root.

The approximation method is based on Newton-Rhapson, which refines an initial estimate with each step of the algorithm xInit is calculated using a second order polynomial: x0(x) = a.∗ x.^2 + b.∗x + c; where a, b, c are calculated using the

least - square polynomial curve fitting technique: a = round(0.812139973184789 $*$ 2^Q); b = round(-2.02427236492453 $*$ 2^Q); c = round(2.21607718374657 $*$ 2^Q); Newton-Raphson step: x(n+1) = x(n) + x(n)$*$(1 - in $*$ x $*$ x) / 2

Parameters

in	*x*	Fixed point representation of the interval [0.5, 1)

Returns

The output is between (1, sqrt(2)] - Q2.14

Note

Accuracy (ULP) -> Mean: 15.41, Worst: 13.51

2.1.2.14 ALWAYS_INLINE vec16s vfxp_sgn (vec16s *a*)

Signum.

Extracts the sign information from the input variable (a == 0 ? 0 : (a $<$ 0 ? -1 : 1))

Parameters

in	*a*	Fixed point value

Returns

-1, 0, 1

Note

2.1.2.15 ALWAYS_INLINE vfxp_s2q14 vfxp_sinh (vfxp_s1q15 *x*)

Hyperbolic Sinus.

Polynomial approximation using the famous Taylor decomposition, the degree is 7 and to speed up the processing a Horner scheme is applied. Chebyshev approximations can be used also but the coefficient values are very big in comparison with the input data format. Big issue with precision !

Parameters

in	*x*	Fixed point representation of the interval [0, 1)

Returns

The output is between [0, 1.18) - Q2.14

Note

Accuracy (ULP) -> Mean: 14.16, Worst: 12.52

2.1.2.16 ALWAYS_INLINE vfxp_s1q15 vfxp_sqrt (vfxp_s1q15 *in*)

Square root.

The algorithm calculates sqrt(x) by using the following formula: sqrt(x) = x / sqrt(x). X is between [0,1) so a scaling is needed to place the value inside [0.5, 1). The scaled value will be sent to vfxp_rsqrt(), the results is then multiplied by x and the scale is compensated. The result of rsqrt will be weighted by sqrt(2) in case the scale is odd and rescaled by (scale - 1) / 2 if the scale is odd and scale / 2 if the scale is even.

Parameters

in	*x*	Fixed point representation of the interval [0, 1)

Returns

The output is between [0, 1) - Q1.15

Note

Accuracy (ULP) -> Mean: 15.52, Worst: 13.32

2.1.2.17 ALWAYS_INLINE vfxp_s1q15 vfxp_tan (vfxp_s1q15 *x*)

Tangent.

Polynomial approximation using the Least square polynomial approximation, the degree is 7 and to speed up the processing a Horner scheme is applied.

Parameters

in	*x*	Fixed point representation in RADIANS of the interval [0, pi/4)

Returns

The output is between [0, 1) - Q1.15

Note

Accuracy (ULP) -> Mean: 13.87, Worst: 12.56

2.1.2.18 ALWAYS_INLINE vfxp_s1q15 vfxp_tanh (vfxp_s1q15 *x*)

Hyperbolic Tangent.

Polynomial approximation using a 7th degree best fit polynomial, and to speed up the processing a Horner scheme is applied. The function uses the fact that tanh(x) has an odd symmetry, this means that the code calculates the tanh(x) over

Parameters

in	*x*	Fixed point representation of the interval [-1, 1)- Q1.15

Returns

The output is between [-0.76159, 0.76159) - Q1.15

Note

Accuracy (ULP) -> Mean: 15.96, Worst: 13.62

Index