Math_Kernels_in_C6Accel

^ Up to main **C6Accel_Reference_guide** Table of Contents

This arcticle is part of a collection of articles describing the C6Accel included in DaVinci/OMAPL/OMAP3 devices. To navigate to the main page for the C6Accel reference guide click on the link above.

Contents

MATH based kernel API call Reference guide

The math kernels in the C6Accel project are a collection of highly-optimized and high-precision mathematical functions. The library is intended for ARM programmers to seamlessly port any floating-point algorithms into fixed-point code for execution on fixed point devices. These routines are typically used in computationally intensive real-time applications, where optimal execution speed and high accuracy are critical. By using these routines you can achieve execution speeds considerably faster than equivalent code written in standard C language. For more information on the IQ formats refer Media:IQMath_fixed_vs_floating.pdf

C6accel_FtoIQ(Float to IQ format conversion)

Function

```
 int C6accel_FtoIQ(C6accel_Handle hC6accel, float *F, int *R, unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- F : float variable
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : Fixed point equivalent of the inputs in float format

Description This function converts a floating-point constant or variable to the equivalent IQ value.

Special scenerio:

- Saturation: If the input is out of limits for a given Q value, the function returns 0 for positive input and 0x80000000 for negative input.

C6accel_IQN(integer to IQ format conversion)

Function

```
int C6accel_IQN( C6accel_Handle hC6accel, int *A,int *R, unsigned char  qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Signed integer value to be converted
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : Fixed point equivalent of the inputs in IQ format

Description This kernel converts an integer (short, char) to an equivalent IQ value

Note This kernel can also be used to convert floating-point numbers to IQ format. However, it is strongly advised not to do so because of the large performance overhead involved. This includes calling two floating-point RTS functions. The _FtoIQ() function should be used for this purpose.

C6accel_IQNtoF(IQ format to float conversion)

Function

```
int C6accel_IQNtoF( C6accel_Handle hC6accel, int  *A,float *R,unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Signed integer value to be converted
- qfmt : Q value for IQ format of the output
- n : number of inputs.
- R : floating point equivalent of the inputs in IQ format

Description This function converts a IQ number to equivalent floating-point value in IEEE 754 format

Usage This operation is typically used in cases where the user may wish to perform some operations in floating-point format or convert data back to floating-point for display purposes.

C6accel_IQNint(IQ format to integer conversion)

Function

```
int C6accel_IQNint( C6accel_Handle hC6accel, int *A,int *R,unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Signed integer value to be converted
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : integer part of the inputs in IQ format.

Description This This function returns the integer portion of IQ number

C6accel_IQXtoIQY(IQ format conversion from X to Y format)

Function

```
int C6accel_IQXtoIQY( C6accel_Handle hC6accel, int *A,int *R,int x, inty, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Value in input IQ format to be converted
- x : intput IQ format
- y : output Q format
- n : number of inputs
- R : Fractional part of the inputs in IQ format

Description This function converts Iq number in X format to IQ number in Y format.

C6accel_IQmpy(IQ format Multiplication)

Function

```
int C6accel_IQmpy(C6accel_Handle hC6accel, int *A, int *B,int *R, unsigned char qfmt, int n )
```

Parameters

- hC6accel: C6Accel handle
- A : 1st input to be multiplied (in IQ format)
- B : 2nd input to be multiplied (in IQ format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : integer part of the inputs in IQ format

Description This function multiplies two IQ number vectors. It does not perform saturation and rounding. In most cases, the multiplication of two IQ variables do not exceed the range of the IQ variable. Amongst all IQ multiply flavors available, this operation takes the least amount of cycles and code size.

C6accel_IQrmpy(IQ Multiplication with rounding)

Function

```
int C6accel_IQrmpy(C6accel_Handle hC6accel, int *A, int *B,int *R, unsigned char qfmt, int n )
```

Parameters

- hC6accel: C6Accel handle
- A : 1st input to be multiplied (in IQ format)
- B : 2nd input to be multiplied (in IQ format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : integer part of the inputs in IQ format

Description This function multiplies two IQ number and rounds the result. In cases where absolute accuracy is necessary, this operation performs the IQ multiply and rounds the result before storing back as an IQ number. This gives an additional 1/2 LSB of accuracy.

C6accel_IQrsmpy(IQ math Multiplication with rounding and saturation)

Function

```
int C6accel_IQrsmpy(C6accel_Handle hC6accel, int *A, int *B,int *R, unsigned char qfmt, int n )
```

Parameters

- hC6accel: C6Accel handle
- A : 1st input to be multiplied (in IQ format)
- B : 2nd input to be multiplied (in IQ format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : integer part of the inputs in IQ format

Description This function multiplies two IQ number with rounding and saturation. In cases where the calculation may possibly exceed the range of the IQ variable, this operation rounds and then saturates the result to the maximum IQ value range before storing.

C6accel_IQmpyI32(Multiplication of IQnumber with an integer)

Function

```
int C6accel_IQmpyI32(C6accel_Handle hC6accel, int *A, int *B,int *R,unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : 1st input to be multiplied (in IQ format)
- B : 2nd input to be multiplied (in int format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : integer part of the inputs in IQ format

Description This macro multiplies an IQ number with an integer.

C6accel_IQmpyI32int(IQ math Multiplication with integral return)

Function

```
int C6accel_IQmpyI32int(C6accel_Handle hC6accel, int *A, int *B, int *R, unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : 1st input to be multiplied (in IQ format)
- B : 2nd input to be multiplied (in int format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : integer part of the inputs in int format

Description This function multiplies an IQ number with an integer and returns the integer part of the result.

C6accel_IQmpyI32frac(IQ math Multiplication with fractional return)

Function

```
int  C6accel_IQmpyI32frac(C6accel_Handle hC6accel, int *A, int *B,int *R, unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : 1st input to be multiplied (in IQ format)
- B : 2nd input to be multiplied (in int format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : fractional part of the inputs in IQ format

Description This function multiplies an IQ number with an integer and returns the fractional part of the result.

C6accel_IQmpyIQX(IQ multiplication of different IQ formats)

Function

```
int C6accel_IQmpyIQX(C6accel_Handle hC6accel, int *A, int N1, int *B, int N2, int *R)
```

Parameters

- hC6accel: C6Accel handle
- A : 1st input to be multiplied (in IQN1 format)
- B : 2nd input to be multiplied (in IQN2format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- Output: integer part of the inputs in IQ format

Description This function multiplies two IQ number that are represented in different IQ formats.

C6accel_IQdiv(IQ math division)

Function

```
int C6accel_IQdiv(C6accel_Handle hC6accel, int *A, int *B,int *R, unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : vector of Dividends (in IQN1 format)
- B : vector of Divisors (in IQN2format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : integer part of the inputs in IQ format

Description This module divides two IQN number and provides 32-bit results (IQN format) using the Newton-Raphson technique.

Note: Saturation This function saturates between maximum and minimum limits of Q format. No special handling of the divide by 0 has been implemented. Thus, the results are undefined if divide by 0 is attempted.

C6accel_IQsin(IQ math Sine function)

Function

```
int C6accel_IQsin(C6accel_Handle hC6accel, int  *A,int *R, unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Angle vector in radians (in IQN1 format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : sine value in IQ format of the angle being passed.

Description This function returns the sine of the input argument as a fixed-point number in IQN format (N=1:29).

Saturation: This function saturates between maximum and minimum limits of Q format.

C6accel_IQNsinPU(IQ math sine function with angle in per unit radian)

Function

```
int C6accel_IQNsinPU(C6accel_Handle hC6accel, int* A, int* R,int qfmt,int n)
```

Parameters

- hC6accel: C6Accel handle
- A : The input argument is in per-unit radians and represented as a fixed-point number in IQN format.(N=1:29)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : Output in Format IQN

Description This function returns the sine of the input argument as a fixed-point number in IQN format (N=1:29).

Saturation The function saturates between maximum and minimum limits of the Q format.

C6accel_IQNcos(IQ math cosine function)

Function

```
int C6accel_IQNcos(C6accel_Handle hC6accel, int  *A,int *R, unsigned char qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Angle vector in radians (in IQN1 format)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : cosine vector in IQ format of the angle vector being passed.

Description This function returns the cosine of the input argument as a fixed-point number in IQN format (N=1:29).

Saturation: This function saturates between maximum and minimum limits of Q format.

C6accel_IQNcosPU(IQ math cosine function with angle in per unit radian)

Function

```
int C6accel_IQNcosPU(C6accel_Handle hC6accel, int* A, int* R, int qfmt,,int n)
```

Parameters

- hC6accel: C6Accel handle

- A : The input vector argument in per-unit radians and represented as a fixed-point number in IQN format.(N=1:29)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : Output cosine value in Format IQN

Description This function returns the cosine of the input argument as a fixed-point number in IQN format (N=1:29).

Saturation: This function saturates between maximum and minimum limits of Q format.

C6accel_IQasin(IQ math arc sine)

Function

```
int C6accel_IQasin(C6accel_Handle hC6accel,  int*  A, int* R,int qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : The input vector of sine value argument represented as a fixed-point number in IQN format.(N=1:29)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : Output angle value in radians in Format IQN

Description This function returns the inverse sine of the input argument as a fixed-point number in IQN format (N=1:29).

C6accel_IQacos(IQ math arc cosine function)

Function

```
int C6accel_IQacos( C6accel_Handle hC6accel, int* A, int* R,int qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : The input vector of cosine value argument represented as a fixed-point number in IQN format.(N=1:29)
- qfmt : Q value for IQ format of the output
- n : numberof inputs
- R : Output angle value in radians in Format IQN

Description This function returns the cosine of the input argument as a fixed-point number in IQN format (N=1:29).

Saturation: This function saturates between maximum and minimum limits of Q format.

C6accel_IQatan2(IQ math arc tan function)

Function

```
int C6accel_IQatan2(C6accel_Handle hC6accel,  int* A, int* B, int R,int qfmt, int n)
```

Parameters

- A : Vector of inputs
- hC6accel: C6Accel handle
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This module computes the 4-quadrant arctangent. Module output in radians and varies from -pi to pi.

C6accel_IQatan2PU(IQ math arc tan with angle in per unit radian)

Function

```
int C6accel_IQatan2PU(C6accel_Handle hC6accel,  int* A, int* B, int* R,int qfmt,int n)
```

Parameters

- hC6accel: C6Accel handle
- A, B : Input vectors to compute inverse tan
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This module computes the 4-quadrant arctangent. Module output is in per-unit radians and varies from 0 (0 radians) to 1 (2p radians).

C6accel_IQsqrt(IQ math squareroot function)

Function

```
int C6accel_IQsqrt( C6accel_Handle hC6accel, int * A,int * R, int qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Input vector in IQ format whose square roots have to be computed
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This module computes the square root of the input using table look-up and Newton-Raphson approximation.

C6accel_IQisqrt(IQ math inverse squareroot)

Function

```
int C6accel_IQisqrt(C6accel_Handle hC6accel,  int * A,int qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Input vector in IQ format whose inverse square roots have to be computed
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This module computes the inverse square root of the input using table look-up and Newton-Raphson approximation.

C6accel_IQmag(IQ math Magnitude of Complex numbers)

Function

```
int C6accel_IQmag( C6accel_Handle hC6accel, int*  A, int  *B,int *R,int qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A,B : real and imaginary vectors of complex terms in IQ format
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This function calculates the magnitude of two orthogonal vectors as follows: Mag = sqrt(A2 + B2). This operation achieves better accuracy and avoids overflow problems that may be encountered by using the _IQsqrt function. This is because the internal computations (A2 + B2) are maintained at 64-bit accuracy.

C6accel_IQexp(Exponential IQ math function)

Function

```
int C6accel_IQexp( C6accel_Handle hC6accel, int  *A,int *R,int qfmt, int n)
```

Parameters

- hC6accel: C6Accel handle
- A : Input vector in IQ format
- R : Result vector of exponential values IQ Format
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This function calculates the fixed-point exponential of a value A.

C6accel_IQlog(Logarithmic IQ math function)

Function

```
int C6accel_IQlog( C6accel_Handle hC6accel, int  *A, int *R,int qfmt, int n )
```

Parameters

- hC6accel: C6Accel handle
- A : Input vector in IQ format
- R : Result vector of log values in IQ format
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This function calculates the fixed-point natural logarithm

C6accel_IQpow(IQmath Power function)

Function

```
int C6accel_IQpow( C6accel_Handle hC6accel, int  *A, int *B, int *R,int qfmt, int n )
```

Parameters

- hC6accel: C6Accel handle
- A : Base vector in IQ vector
- B : Power Vector in IQ vector
- R : Result vector in IQ format
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This function calculates the magnitude of two orthogonal vectors as follows: Mag = sqrt(A2 + B2). This operation achieves better accuracy and avoids overflow problems that may be encountered by using the _IQsqrt function. This is because the internal computations (A2 + B2) are maintained at 64-bit accuracy.

C6accel_IQabs(Absolute value IQmath function)

Function

```
int C6accel_IQabs(C6accel_Handle hC6accel,  int  *A, int *R,int qfmt, int n )
```

Parameters

- hC6accel: C6Accel handle
- A : Input vector in IQ format
- R : Result vector in IQ format
- qfmt : Q value for IQ format of the output
- n : numberof inputs

Description This function calculates the absolute value of an IQ number

C6accel_addsp_i: Single precision floating-point addition

Function:

```
int C6accel_addsp_i(C6accel_Handle hC6accel,float *x, float *y,float *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x,y : input float vectors
- r : Resultant float vector (x+y)
- n : Number of elements in the vectors

Description: The sum of two input 32-bit floating-point number is generated

Special Cases:

- Zero input return zero output
- Underflow and overflow is checked only in the DEBUG mode

C6accel_subsp_i: Single precision floating point subtraction

Function:

```
int C6accel_subsp_i(C6accel_Handle hC6accel,float *x, float *y, float *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x,y : input float vectors
- r : Resultant float vector (x-y)
- n : Number of elements in the vectors

Description: The difference of two single precision floating point numbers

Special Cases:

- Underflow and overflow is checked in DEBUG mode

C6accel_uintsp_i: Convert 32-bit unsigned integer to single precision floating point

Function:

```
int C6accel_uintsp_i(C6accel_Handle hC6accel,unsigned int *x, float *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: A 32-bit unsigned integer is converted to a single precision floating point number

C6accel_divsp_i: Single-precision floating-point division

Function:

```
int C6accel_divsp_i(C6accel_Handle hC6accel,float *x, float *y, float *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x,y : input float vectors
- r : Resultant float vector (x/y)
- n : Number of elements in the vectors

Description: The quotient for division of two 32-bit floating-point numbers is generated

Special Cases:

- Underflow and Overflow of the quotient is checked only in the DEBUG mode
- Zero divided by Zero returns 1.#NAN
- Non-zero over zero returns infinity

C6accel_intsp_i: Convert 32-bit signed integer to single-precision floating-point

Function:

```
int C6accel_intsp_i(C6accel_Handle hC6accel,int *x, float *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: An input 32-bit signed integer is converted to a 32-bit single precision floating point number

C6accel_mpysp_i: Single precision floating-point multiplication

Function:

```
 int C6accel_mpysp_i(C6accel_Handle hC6accel,float *x, float *y, float *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x,y : input float vectors
- r : Resultant float vector (x*y)
- n : Number of elements in the vectors

Description: The product of two 32-bit floating point numbers is generated

C6accel_recipsp_i: Single precision floating point reciprocal

Function:

```
int C6accel_recipsp_i(C6accel_Handle hC6accel,float *x, float *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x : input float vector
- r : Resultant float vector (1/x)
- n : Number of elements in the vectors

Description: The reciprocal of an input 32-bit floating point number is generated

Special Cases:

- Underflow and overflow is checked only in DEBUG mode
- The reciprocal of zero returns infinity

C6accel_spint_i: Single precision floating point to 32-bit signed integer

Function:

```
int C6accel_spint_i(C6accel_Handle hC6accel,float *x, int *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x : input float vector
- r : Resultant float vector r=(float)x
- n : Number of elements in the vectors

Description: A single precision floating point number is converted to a 32-bit signed integer

C6accel_spuint_i: Single precision floating point to 32-bit unsigned integer

Function:

```
int C6accel_spuint_i(C6accel_Handle hC6accel,float *x, unsigned int *r, int n)
```

Parameters

- hC6accel: C6Accel handle
- x,y : input float vectors
- r : Resultant int vector r = (unsigned int)x
- n : Number of elements in the vectors

Description: A single precision floating point number is converted to 32-bit unsigned integer

Special Cases:

- Numbers less than 1.0 returns zero
- Results greater than 32 bits generate the following saturation values:
- 0xffff_ffff for positive numbers
- 0x0000_0000 for negative numbers

Floating point FastRTS kernels in C6Accel

C6accel_atandp (Double-Precision Polar Arc Tangent)

Function call:

```
        Int C6accel_atandp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_atandp function return the arc tangent of a floating-point argument z. The return value is an angle in the range [-?/2, ?/2] radians.

Special Cases:

- If | z | < 1.49e-8 = 2- 26, then the return value is z for small angles.

C6accel_atansp (Single-Precision Polar Arc Tangent)

Function:

```
        Int C6accel_atansp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_atandp function return the arc tangent of a floating-point argument z. The return value is an angle in the range [-?/2, ?/2] radians.

Special Cases:

- If | z | < 2.44e-4 = 2- 12, then the return value is z for small angles.

C6accel_cosdp (Double-Precision Cosine)

Function

```
Int C6accel_cosdp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_cosdp function return the cosine of a floating-point argument z. The angle z is expressed in radians. The return value is in the range of [-1.0 and +1.0]. An argument with a large magnitude may produce a result with little or no significance.

Special Cases:

- If | W | < 9.536743e-7 = 2- 20, then the return value is W for small angles.
- If | W | > 1.0737e+9 = 2+30, then the return value is zero for large angles.

C6accel_cossp (Single-Precision Cosine)

Function:

```
Int C6accel_cossp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_cossp function return the cosine of a floating-point argument z. The angle z is expressed in radians. The return value is in the range of [-1.0 and +1.0]. An argument with a large magnitude may produce a result with little or no significance.

Special Cases:

- If | W | < 2.44e-4 = 2- 12, then the return value is W for small angles.
- If | W | > 1.04858e+6 = 2+20, then the return value is zero for large angles.

C6accel_sindp (Double-Precision Sine)

Function

```
Int C6accel_sindp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_sindp function return the sine of a floating-point argument z. The angle z is expressed in radians. The return value is in the range of [-1.0 and +1.0]. An argument with a large magnitude may produce a result with little or no significance.

Special Cases:

- If | W | < 9.536743e-7 = 2- 20, then the return value is W for small angles.

- If | W | > 1.0737e+9 = 2+30, then the return value is zero for large angles.

C6accel_sinsp (Single-Precision Sine)

Function:

```
Int C6accel_sinsp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_sinsp function return the sine of a floating-point argument z. The angle z is expressed in radians. The return value is in the range of [-1.0 and +1.0]. An argument with a large magnitude may produce a result with little or no significance.

Special Cases:

- If | W | < 2.44e-4 = 2- 12, then the return value is W for small angles.
- If | W | > 1.04858e+6 = 2+20, then the return value is zero for large angles.

C6accel_expdp (Double-Precision Exponential)

Function

```
Int C6accel_expdp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_expdp function returns the exponential function of a real floating-point argument z. The return value is the number e raised to power z. If the magnitude of z is too large, the maximum double-precision floating-point number (1.797693e+308 = 2+1024) is returned.

Special Cases:

- If | z | < 1.11e-16 = 2- 53, then the return value is 1.0 for small arguments.
- If z < -708.3964 = minimum log e (2.225e-308 = 2- 1022), then the return value is 0.0.
- If z > +709.7827 = maximum log e (1.797693e+308 = 2+1024), then the return value is 1.797693e+308 = 2+1024(maximum double-precision floating-point

number).

C6accel_expsp (Single-Precision Exponential)

Function:

```
Int C6accel_expsp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_expdp function returns the exponential function of a real floating-point argument z. The return value is the number e raised to power z. If the magnitude of z is too large, the maximum double-precision floating-point number (1.797693e+308 = 2+1024) is returned.

Special Cases:

- If | z | < 9.313e-10 = 2- 30, then the return value is 1.0 for small arguments.
- If z < -87.3365 = minimum log e (1.175e-38 = 2- 126), then the return value is 0.0.
- If z > +88.7228 = maximum log e (3.402823e+38 = 2+128), then the return value is 3.402823e+38 = 2+128 (maximum single-precision floating-point number).

C6accel_exp2dp (Double-Precision Exponent of 2)

Function

```
Int C6accel_exp2dp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_exp2dp function returns the exponential function of a real floating-point argument z. The return value is the number 2 raised to power z. If the magnitude of z is too large, the maximum double-precision floating-point number (1.797693e+308 = 2+1024) is returned.

Special Cases:

- If | z | < 1.11e-16 = 2- 53, then the return value is 1.0 for small arguments.
- If z < -708.3964 = minimum log 2 (2.225e-308 = 2- 1022), then the return value is 0.0.
- If z > +709.7827 = maximum log 2 (1.797693e+308 = 2+1024), then the return value is 1.797693e+308 = 2+1024(maximum double-precision floating-point

number).

C6accel_exp2sp (Single-Precision Exponent of 2)

Function:

```
Int C6accel_exp2sp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_exp2dp function returns the exponential function of a real floating-point argument z. The return value is the number 2 raised to power z. If the magnitude of z is too large, the maximum double-precision floating-point number (1.797693e+308 = 2+1024) is returned.

Special Cases:

- If | z | < 9.313e-10 = 2- 30, then the return value is 1.0 for small arguments.
- If z < -87.3365 = minimum log 2 (1.175e-38 = 2- 126), then the return value is 0.0.
- If z > +88.7228 = maximum log 2 (3.402823e+38 = 2+128), then the return value is 3.402823e+38 = 2+128 (maximum single-precision floating-point number).

C6accel_exp10dp (Double-Precision Exponent of 10)

Function

```
Int C6accel_exp10dp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_exp10dp function returns the exponential function of a real floating-point argument z. The return value is the number 10 raised to power z. If the magnitude of z is too large, the maximum double-precision floating-point number (1.797693e+308 = 2+1024) is returned.

Special Cases:

- If | z | < 1.11e-16 = 2- 53, then the return value is 1.0 for small arguments.
- If z < -708.3964 = minimum log 2 (2.225e-308 = 2- 1022), then the return value is 0.0.
- If z > +709.7827 = maximum log 2 (1.797693e+308 = 2+1024), then the return value is 1.797693e+308 = 2+1024(maximum double-precision floating-point

number).

C6accel_exp10sp (Single-Precision Exponent of 10)

Function:

```
Int C6accel_exp10sp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_exp2dp function returns the exponential function of a real floating-point argument z. The return value is the number 10 raised to power z. If the magnitude of z is too large, the maximum double-precision floating-point number (1.797693e+308 = 2+1024) is returned.

Special Cases:

- If | z | < 9.313e-10 = 2- 30, then the return value is 1.0 for small arguments.
- If z < -87.3365 = minimum log 10 (1.175e-38 = 2- 126), then the return value is 0.0.
- If z > +88.7228 = maximum log 10 (3.402823e+38 = 2+128), then the return value is 3.402823e+38 = 2+128 (maximum single-precision floating-point number).

C6accel_logdp (Double-Precision Logarithm)

Function

```
Int C6accel_logdp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_logdp function returns the logarithm function of a real floating-point argument z. The return value is the number logarithmic value of z. If the magnitude of z is too large, the maximum double-precision floating-point number (1.797693e+308 = 2+1024) is returned.

Special Cases:

- If | z | < 1.11e-16 = 2- 53, then the return value is 1.0 for small arguments.
- If z < -708.3964 = minimum log e (2.225e-308 = 2- 1022), then the return value is 0.0.
- If z > +709.7827 = maximum log e (1.797693e+308 = 2+1024), then the return value is 1.797693e+308 = 2+1024(maximum double-precision floating-point

number).

C6accel_logsp (Single-Precision Logarithm)

Function:

```
Int C6accel_logsp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vectorlogarithm
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_logdp function returns the logarithm function of a real floating-point argument z. The return value is the number logarithmic value of z. If the magnitude of z is too large, the maximum double-precision floating-point number (1.797693e+308 = 2+1024) is returned.

Special Cases:

- If | z | < 9.313e-10 = 2- 30, then the return value is 1.0 for small arguments.
- If z < -87.3365 = minimum log e (1.175e-38 = 2- 126), then the return value is 0.0.
- If z > +88.7228 = maximum log e (3.402823e+38 = 2+128), then the return value is 3.402823e+38 = 2+128 (maximum single-precision floating-point number).

All the Fast RTS MATH kernels are part of C6Accel release 1.00.00.01. Documentation For these kernels will be added in the next Release. For updates please refer Wiki documentation.

C6accel_recipdp (Double-Precision Reciprocal)

Function

```
Int C6accel_logdp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_recipdp function returns the reciprocal function of a real floating-point argument z. The return value is the number reciprocal value of z.

Special Cases:

- If | z | < 2.225e-308 = 2- 1022, then the return value for small arguments is NaN = Not-a-Number (exponent and mantissa are all ones) > maximum double precision floating point value +/- 1.797693e+308 = +/-1* 2+1024.
- If | z | > 1.797693e+308 = 2+1024, then the return value is zero for large arguments.

C6accel_recipsp (Single-Precision Reciprocal)

Function:

```
Int C6accel_recipsp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_recipdp function returns the reciprocal function of a real floating-point argument z. The return value is the number reciprocal value of z.

Special Cases:

- If | z | < 1.1755e-38 = 2- 126, then the return value for small arguments is NaN = Not-a-Number (exponent and mantissa are all ones) > the maximum single precision floating point value +/- 3.402823e+38 = +/-1 * 2+128.

- If | z | > 3.402823e+38 = 2+128, then the return value is zero for large arguments.

C6accel_rsqrtdp (Double-Precision Reciprocal square root)

Function

```
Int C6accel_rsqrtdp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_rsqrtdp function returns the reciprocal squareroot function of a real floating-point argument z. The return value is the number reciprocal value of z.

Special Cases:

- If | z | < 2.225e-308 = 2- 1022, then the return value for small arguments is NaN = Not-a-Number (exponent and mantissa are all ones) > the maximum double-precision floating point value 1.797693e+308 = 2+1024.
- If | z | > 1.797693e+308 = 2+1024, then the return value is zero for large arguments.

C6accel_rsqrtsp (Single-Precision Reciprocal square root)

Function:

```
Int C6accel_rsqrtsp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_rsqrtsp function returns the reciprocal square root function of a real floating-point argument z. The return value is the number reciprocal value of z.

Special Cases:

- If | z | < 1.1755e-38 = 2- 126, then the return value for small arguments is NaN = Not-a-Number (exponent and mantissa are all ones) > maximum single precision floating point value 3.402823e+38 = 2+128.
- If | z | > 3.402823e+38 = 2+128, then the return value is zero for large arguments.

C6accel_sqrtdp (Double-Precision Square root)

Function

```
Int C6accel_sqrtdp(C6accel_Handle hC6accel,double *z, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input double vector
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_sqrtdp function returns the square root function of a real floating-point argument z. The return value is the number reciprocal value of z.

Special Cases:

- If | z | < 2.225e-308 = 2- 1022, then the return value for small arguments is NaN = Not-a-Number (exponent and mantissa are all ones) > the maximum double-precision floating point value 1.797693e+308 = 2+1024.
- If | z | > 1.797693e+308 = 2+1024, then the return value is zero for large arguments.

C6accel_sqrtsp (Single-Precision Square root)

Function:

```
Int C6accel_sqrtsp(C6accel_Handle hC6accel,float *z, float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- z : input float vector
- r : Resultant float vector r = (float)x
- n : Number of elements in the vectors

Description: The C6accel_sqrtsp function returns the square root function of a real floating-point argument z. The return value is the number reciprocal value of z.

Special Cases:

- If | z | < 1.1755e-38 = 2- 126, then the return value for small arguments is NaN = Not-a-Number (exponent and mantissa are all ones) > maximum single precision floating point value 3.402823e+38 = 2+128.
- If | z | > 3.402823e+38 = 2+128, then the return value is zero for large arguments.

C6accel_powdp (Double-Precision Raise to a Power)

Function

```
Int C6accel_powdp(C6accel_Handle hC6accel,double *x,double *y, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- x : input double vector(base)
- y : input double vector (power)
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_powdp function returns the x power of y function The return value is the number x^y.

Special Cases: The following order of tests are observed:

- If y = 0, return 1.0 (x is ignored).
- If | x | > 8.9885e+307 = 2+1023, the return value is Infinity (y is ignored).
- If | x | < 2.225e-308 = 2- 1022, then the return value is 0 (y is ignored).
- If x < 0, and y is not an integer value, then NaN is returned.
- If x < 0, and y is a 32-bit integer value, -1y* |x| y is formed.

Form W = y * log e (|x|).

- If W > 709.089 = maximum log e (+8.988e+307 = 2+1023), Infinity is returned.
- If W < -708.396 = minimum log e (+2.225e-307 = 2- 1022), then 0 is returned.

Otherwise, exp e (W) = exp e (y * log e (x)) = x y is returned.

C6accel_powsp (Single-Precision Raise to a Power)

Function:

```
Int C6accel_powsp(C6accel_Handle hC6accel,float *x, ,float *y,  float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- x : input float vector(base)
- y : input float vector(power)
- r : Resultant float vector r = (float)x
- n : Number of elements in the vectors

Description: The C6accel_powsp function returns the x power of y function . The return value is the x power of y.

Special Cases:

- The following order of tests are observed:
- If y = 0, return 1.0 (x is ignored).
- If | x | > 1.701e+38 = 2+127, the return value is Infinity (y is ignored).
- If | x | < 1.175e-38 = 2- 126, then the return value is 0 (y is ignored).
- If x < 0, and y is not an integer value, then NaN is returned.
- If x < 0, and y is a 32-bit integer value, -1y* |x| y is formed.
- Form W = y * logf e (|x|).

If W > 88.02969 = maximum logf e (+1.701e+38 = 2+127), Infinity is returned. If W < -87.3365 = minimum logf e (+1.175e-38 = 2- 126), then 0 is returned. Otherwise, expf e (W) = expf e (y * logf e (x)) = x y is returned.

C6accel_divdp (Double-Precision Division)

Function

```
Int C6accel_divdp(C6accel_Handle hC6accel,double *x,double *y, double *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- x : input double vector(dividend)
- y : input double vector (divisor)
- r : Resultant double vector r= (float)x
- n : Number of elements in the vectors

Description: The C6accel_divdp function returns the division function of a real floating-point argument x by y. The return value is the number resulting from x/y.

Special Cases:

- If | y | < 2.225e-308 = 2- 1022, then the return value is NaN = Not-a-Number(exponent and mantissa are all ones) > +/- 1.797693e+308 = +/-1 * 2+1024(largest double-precision floating-point number) with the sign of x.

C6accel_divsp (Single-Precision Division)

Function:

```
Int C6accel_divsp(C6accel_Handle hC6accel,float *x, ,float *y,  float *r, int n)
```

Parameters:

- hC6accel: C6Accel handle
- x : input float vector(dividend)
- y : input float vector(divisor)
- r : Resultant float vector r = (float)x
- n : Number of elements in the vectors

Description: The C6accel_divsp functions return the quotient of two real floating-point arguments x and y. The return value is x / y.

Special Cases:

- If | y| < 1.1755e-38 = 2- 126, then the return value is NaN = Not-a-Number (exponent and mantissa are all ones) > +/- 3.402823e+38 = +/- 1 * 2+128 (largest single-precision floating-point number) with the sign of x.

Return to C6Accel Documentation

Click here.