

IEEE Standard VHDL Mathematical Packages

Sponsor

**Design Automation Technical Committee
of the
IEEE Computer Society**

Reaffirmed 10 December 2002

Approved 19 September 1996

IEEE Standards Board

Abstract: The MATH_REAL package declaration, the MATH_COMPLEX package declaration, and the semantics of the standard mathematical definition and conventional meaning of the functions that are part of this standard are provided. Ways for users to implement this standard are given in an informative annex. Samples of the MATH_REAL and MATH_COMPLEX package bodies are provided in an informative annex on diskette as guidelines for implementors to verify their implementation of this standard. Implementors may choose to implement the package bodies in the most efficient manner available to them.

Keywords: IEEE, MATH_COMPLEX, MATH_REAL, VHDL, VHDL model interoperability

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1997 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1997. Printed in the United States of America.

ISBN 1-55937-894-8

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (508) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

(This introduction is not a part of IEEE Std 1076.2-1996, IEEE Standard VHDL Mathematical Packages.)

This set of packages provides a standard for the declaration of most frequently used real and complex elementary functions required for numerically oriented modeling applications. Use of these packages with their defined data types, constants, and functions is intended to provide a mechanism for writing VHDL models (compliant with IEEE Std 1076-1993) that are portable and interoperable with other VHDL models adhering to this standard. The standard serves a broad class of applications with reasonable ease of use and requires implementations that are of high quality.

The work of this committee is the culmination of efforts by several groups with the same goals working over a period of three and a half years. Between 1992 and this date, many individuals made valuable contributions to the development of this standard, and several commercial organizations donated material that was used as a starting point or building block for the final version of these packages. In addition, the following sources were used for some of the algorithms in the packages:

- Abramowitz, M. and Stegun, I., *Handbook of Mathematical Functions*, 9th ed. New York: Dover Publications, Inc., 1972.
- Hu, X., Harber, R., and Bass, S., “Expanding the range of convergence of the CORDIC algorithm,” *IEEE Transactions on Computers*, vol. 40, no. 1, pp. 13–21, 1991.
- L’Ecuyer, Pierre, “Efficient and Portable Combined Random Number Generators,” *Communications of the ACM*, vol. 31, no. 6, pp. 742–774, 1988.
- University of California, Berkeley. “C source for natural logarithm function,” */system/unix/4.3bsd-repo/lib/libm/common_source/log.c*, version 5.10 (Berkeley), 1993.
- Walther, J. S., “A unified algorithm for elementary functions,” *Proceedings of Spring Joint Computer Conference*, pp. 379–385, 1971.

The following persons were members of the IEEE P1076.2 Working Group:

Jose A. Torres, *Chair*

Donald F. Hanson

Charles Swart

Alex Zamfirescu

The following persons were on the balloting committee:

Mostapha Aboulhamid	James P. Hanna	Curtis Parks
Robert E. Anderson	William A. Hanna	Gary S. Porter
Peter J. Ashenden	Donald F. Hanson	Adam Postula
Mike Beaver	Randolph E. Harr	Jan Pukite
Jean-Michel Berge	Rich Hatcher	Fred Rose
J. Bhasker	Robert G. Hillman	Hemant G. Rotithor
William D. Billowitch	Yee-Wing Hsieh	Larry F. Saunders
Dennis B. Brophy	Yu-I Hsieh	Quentin Schmierer
Raul Camposano	Sylvie Hurat	Francesco Sforza
Harold W. Carter	Masaharu Imai	Charles Shelor
Mojj C. Chian	Mitsuaki Ishikawa	Raj Singh
Ernst Christen	Stephen Ives	David W. Smith
David Coelho	Navneet Kumar Jain	J. Dennis Soderberg
Alan Coppola	Choon B. Kim	Balsha R. Stanisic
W. Terry Coston	Chris Kingsley	Michael F. Sullivan
Joanne DeGroat	Jean Lebrun	Charles Swart
Antonie C. deJager	Rajeev Madhavan	Jose A. Torres
Allen Dewey	Serge Maginot	Alain Vachoux
Michael A. Dukes	Victor Martin	Ranganadha R. Vemuri
Douglas D. Dunlop	Paul J. Menchini	Venkat V. Venkataraman
William Fazakerly	Gerald T. Michael	Ronald Waxman
Jacques P. Flandrois	John T. Montague	Alan Whittaker
Richard Grisel	Chandra Moturu	John C. Willis
Michael J. Haney	Robert N. Newshutz	Mark Zwolinski

When the IEEE Standards Board approved this standard on 19 September 1996, it had the following membership:

Donald C. Loughry, *Chair*

Richard J. Holleman, *Vice Chair*

Andrew G. Salem, *Secretary*

Gilles A. Baril	E. G. "Al" Kiener	Jose R. Ramos
Clyde R. Camp	Joseph L. Koepfinger*	Arthur K. Reilly
Joseph A. Cannatelli	Stephen R. Lambert	Ronald H. Reimer
Stephen L. Diamond	Lawrence V. McCall	Gary S. Robinson
Harold E. Epstein	L. Bruce McClung	Ingo Rüsck
Donald C. Fleckenstein	Marco W. Migliaro	John S. Ryan
Jay Forster*	Mary Lou Padgett	Chee Kiow Tan
Donald N. Heirman	John W. Pope	Leonard L. Tripp
Ben C. Johnson		Howard L. Wolfman

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
Alan H. Cookson
Chester C. Taylor

Mary Lynne Nielsen
IEEE Standards Project Editor

Contents

	PAGE
1. Overview.....	1
1.1 Scope.....	1
1.2 Constants, types, and functions provided	1
1.3 Conformance with this standard	3
2. References.....	3
3. Package declarations.....	4
3.1 MATH_REAL	4
3.2 MATH_COMPLEX.....	15
ANNEX	
Annex A (informative) Using the MATH_REAL and MATH_COMPLEX packages	34
A.1 Predefined data types, operators, and precision for MATH_REAL.....	34
A.2 Use of constants in MATH_REAL.....	34
A.3 Use and constraints of pseudo-random number generator in MATH_REAL	34
A.4 Precision across different platforms.....	34
A.5 Handling of overflow/underflow conditions.....	34
A.6 Testbench for the packages	35
A.7 Compatibility with VHDL analog standard.....	35
A.8 Overloading side effect	35
A.9 Synthesizability of functions.....	35
Annex B (informative) Package bodies	36

IEEE Standard VHDL Mathematical Packages

1. Overview

1.1 Scope

This standard is embodied in the MATH_REAL and MATH_COMPLEX package declarations, and in the semantics of the standard mathematical definition and the conventional meaning of the functions that are part of this standard, along with 1.3. The information in annex A is a guide to users and implementors and is not a normative part of this standard, but suggests ways in which one might use this set of packages. The information in annex B is provided as a guideline for implementors and is not a normative part of this standard, but suggests ways in which implementors may implement this standard. The functions in this set of packages were chosen because of their widespread utility, as well as because they are needed to support general floating-point usage and to build other generic packages.

1.2 Constants, types, and functions provided

The following constants of type REAL are provided:

MATH_E	MATH_LOG_OF_2	MATH_DEG_TO_RAD
MATH_1_OVER_E	MATH_LOG_OF_10	MATH_RAD_TO_DEG
	MATH_LOG2_OF_E	
MATH_PI	MATH_LOG10_OF_E	
MATH_2_PI		
MATH_1_OVER_PI	MATH_SQRT_2	
MATH_PI_OVER_2	MATH_1_OVER_SQRT_2	
MATH_PI_OVER_3	MATH_SQRT_PI	
MATH_PI_OVER_4		
MATH_3_PI_OVER_2		

The following functions/procedures of type REAL are provided:

SIGN(X)	EXP(X)	SINH(X)
CEIL(X)	LOG(X)	COSH(X)
FLOOR(X)	LOG2(X)	TANH(X)
ROUND(X)	LOG10(X)	
TRUNC(X)	LOG(X, BASE)	ARCSINH(X)
"MOD"(X, Y)		ARCCOSH(X)
	SIN(X)	ARCTANH(X)
REALMAX(X, Y)	COS(X)	
REALMIN(X, Y)	TAN(X)	
UNIFORM(SEED1, SEED2, X)		
SQRT(X)	ARCSIN(X)	
CBRT(X)	ARCCOS(X)	
"**"(X, Y)	ARCTAN(Y)	
	ARCTAN(Y, X)	

The following types and subtypes are provided:

COMPLEX	POSITIVE_REAL
COMPLEX_POLAR	PRINCIPAL_VALUE

The following constants of type COMPLEX are provided:

MATH_CBASE_1	MATH_CBASE_J	MATH_CZERO
--------------	--------------	------------

The following type conversion functions for COMPLEX and COMPLEX_POLAR are provided:

CMPLX(X, Y)	POLAR_TO_COMPLEX(Z)
COMPLEX_TO_POLAR(Z)	GET_PRINCIPAL_VALUE(X)

The following overloaded relational functions for type COMPLEX_POLAR are provided:

"="(L, R)	"/="(L, R)
-----------	------------

The following functions for type COMPLEX and COMPLEX_POLAR are provided:

"ABS"(Z)	EXP(Z)	SIN(Z)
ARG(Z)	LOG(Z)	COS(Z)
	LOG2(Z)	
"-"(Z)	LOG10(Z)	SINH(Z)
CONJ(Z)	LOG(Z, BASE)	COSH(Z)
SQRT(Z)		

The following arithmetic functions for type COMPLEX and COMPLEX_POLAR are provided:

"+"	"*"
"_"	"/"

1.3 Conformance with this standard

The following conformance rules shall apply as they pertain to the use and implementation of this standard:

- a) The package declarations may be modified to include additional data required by tools, but modifications shall in no way change the external interfaces or simulation behavior of the description. It is permissible to add comments and/or attributes to the package declarations, but not to change or delete any original lines of the approved package declaration.
- b) The standard mathematical definition and conventional meaning of the mathematical functions that are part of this standard, together with the MATH_REAL and MATH_COMPLEX package declarations, represent the formal semantics of the implementation of the MATH_REAL and MATH_COMPLEX packages. An implementation is provided as a guideline in annex B. Implementors of these packages may choose to simply compile the package bodies provided in annex B, or they may choose to implement the package bodies in the most efficient form available to them. Implementations should conform to the semantics and minimum precision required by this standard.
- c) The MATH_REAL package shall be built on top of the standard data type and precision requirements for floating point operations defined in IEEE Std 1076-1993 (STD.STANDARD).
- d) The minimum precision required is that of IEEE Std 1076-1993. Because of this reason and the fact that the functions are implemented on digital computers with only finite precision, the functions and constants in this set of packages can, at best, only approximate the corresponding mathematically defined functions and constants. An implementation is allowed to provide a higher precision than the minimum required.
- e) For some functions, the implementation shall deliver “prescribed results” for certain special arguments, as defined in the comments for the functions in the function declaration. The purpose is to strengthen the accuracy requirements at special argument values. Prescribed results take precedence over maximum relative error requirements.
- f) The semantics of the standard require that all the functions in the packages detect and report invalid parameters (out of valid domain) through an assert statement. The domain of valid values is indicated in the MATH_REAL and MATH_COMPLEX package declarations. The default value of the severity level shall be Error.
- g) The semantics of the standard do not require detection of overflow or underflow. Therefore, detection of underflow/overflow is optional and implementation dependent.
- h) If an implementation chooses to provide any extensions beyond the minimum requirements of this standard (e.g., precision, overflow handling), then it shall document its behavior accordingly.
- i) The MATH_REAL and MATH_COMPLEX packages shall be compiled into a library symbolically named IEEE.

2. References

This standard shall be used in conjunction with the following publications:

IEEE Std 754-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic (ANSI).¹

IEEE Std 1076-1993, IEEE Standard VHDL Language Reference Manual (ANSI).

¹IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P. O. Box 1331, Piscataway, NJ 08855-1331, USA.

3. Package declarations

The declaration of each function includes the following information: description of the mathematical definition of the function; values to be returned by the function for special arguments; valid domain of values for the input arguments; error conditions; range of values into which the function maps the values in its domain; and notes on special accuracy situations, reachable values, usable domains, or algorithms to be used by an implementation.

3.1 MATH_REAL

```
-----  
--  
-- Copyright 1996 by IEEE. All rights reserved.  
--  
-- This source file is an essential part of IEEE Std 1076.2-1996, IEEE Standard  
-- VHDL Mathematical Packages. This source file may not be copied, sold, or  
-- included with software that is sold without written permission from the IEEE  
-- Standards Department. This source file may be used to implement this standard  
-- and may be distributed in compiled form in any manner so long as the  
-- compiled form does not allow direct decompilation of the original source file.  
-- This source file may be copied for individual use between licensed users.  
-- This source file is provided on an AS IS basis. The IEEE disclaims ANY  
-- WARRANTY EXPRESS OR IMPLIED INCLUDING ANY WARRANTY OF MERCHANTABILITY  
-- AND FITNESS FOR USE FOR A PARTICULAR PURPOSE. The user of the source  
-- file shall indemnify and hold IEEE harmless from any damages or liability  
-- arising out of the use thereof.  
--  
-- Title:      Standard VHDL Mathematical Packages (IEEE Std 1076.2-1996,  
--            MATH_REAL)  
--  
-- Library:    This package shall be compiled into a library  
--            symbolically named IEEE.  
--  
-- Developers: IEEE DASC VHDL Mathematical Packages Working Group  
--  
-- Purpose:    This package defines a standard for designers to use in  
--            describing VHDL models that make use of common REAL constants  
--            and common REAL elementary mathematical functions.  
--  
-- Limitation: The values generated by the functions in this package may  
--            vary from platform to platform, and the precision of results  
--            is only guaranteed to be the minimum required by IEEE Std 1076-  
--            1993.  
--  
-- Notes:      No declarations or definitions shall be included in, or  
--            excluded from, this package.  
--            The "package declaration" defines the types, subtypes, and  
--            declarations of MATH_REAL.  
--            The standard mathematical definition and conventional meaning  
--            of the mathematical functions that are part of this standard  
--            represent the formal semantics of the implementation of the  
--            MATH_REAL package declaration. The purpose of the MATH_REAL  
--            package body is to provide a guideline for implementations to  
--            verify their implementation of MATH_REAL. Tool developers may  
--            choose to implement the package body in the most efficient  
--            manner available to them.
```

```

--
-- -----
-- Version      : 1.5
-- Date         : 24 July 1996
-- -----

package MATH_REAL is
    constant CopyrightNotice: STRING
        := "Copyright 1996 IEEE. All rights reserved.";

    --
    -- Constant Definitions
    --
    constant MATH_E : REAL := 2.71828_18284_59045_23536;
                                -- Value of e
    constant MATH_1_OVER_E : REAL := 0.36787_94411_71442_32160;
                                -- Value of 1/e
    constant MATH_PI : REAL := 3.14159_26535_89793_23846;
                                -- Value of pi
    constant MATH_2_PI : REAL := 6.28318_53071_79586_47693;
                                -- Value of 2*pi
    constant MATH_1_OVER_PI : REAL := 0.31830_98861_83790_67154;
                                -- Value of 1/pi
    constant MATH_PI_OVER_2 : REAL := 1.57079_63267_94896_61923;
                                -- Value of pi/2
    constant MATH_PI_OVER_3 : REAL := 1.04719_75511_96597_74615;
                                -- Value of pi/3
    constant MATH_PI_OVER_4 : REAL := 0.78539_81633_97448_30962;
                                -- Value of pi/4
    constant MATH_3_PI_OVER_2 : REAL := 4.71238_89803_84689_85769;
                                -- Value 3*pi/2
    constant MATH_LOG_OF_2 : REAL := 0.69314_71805_59945_30942;
                                -- Natural log of 2
    constant MATH_LOG_OF_10 : REAL := 2.30258_50929_94045_68402;
                                -- Natural log of 10
    constant MATH_LOG2_OF_E : REAL := 1.44269_50408_88963_4074;
                                -- Log base 2 of e
    constant MATH_LOG10_OF_E : REAL := 0.43429_44819_03251_82765;
                                -- Log base 10 of e
    constant MATH_SQRT_2 : REAL := 1.41421_35623_73095_04880;
                                -- square root of 2
    constant MATH_1_OVER_SQRT_2 : REAL := 0.70710_67811_86547_52440;
                                -- square root of 1/2
    constant MATH_SQRT_PI : REAL := 1.77245_38509_05516_02730;
                                -- square root of pi
    constant MATH_DEG_TO_RAD : REAL := 0.01745_32925_19943_29577;
                                -- Conversion factor from degree to radian
    constant MATH_RAD_TO_DEG : REAL := 57.29577_95130_82320_87680;
                                -- Conversion factor from radian to degree

    --
    -- Function Declarations
    --
    function SIGN (X: in REAL ) return REAL;
        -- Purpose:
        --     Returns 1.0 if X > 0.0; 0.0 if X = 0.0; -1.0 if X < 0.0
        -- Special values:
        --     None
        -- Domain:

```

```

--      X in REAL
-- Error conditions:
--      None
-- Range:
--      ABS(SIGN(X)) <= 1.0
-- Notes:
--      None

function CEIL (X : in REAL ) return REAL;
-- Purpose:
--      Returns smallest INTEGER value (as REAL) not less than X
-- Special values:
--      None
-- Domain:
--      X in REAL
-- Error conditions:
--      None
-- Range:
--      CEIL(X) is mathematically unbounded
-- Notes:
--      a) Implementations have to support at least the domain
--          ABS(X) < REAL(INTEGER'HIGH)

function FLOOR (X : in REAL ) return REAL;
-- Purpose:
--      Returns largest INTEGER value (as REAL) not greater than X
-- Special values:
--      FLOOR(0.0) = 0.0
-- Domain:
--      X in REAL
-- Error conditions:
--      None
-- Range:
--      FLOOR(X) is mathematically unbounded
-- Notes:
--      a) Implementations have to support at least the domain
--          ABS(X) < REAL(INTEGER'HIGH)

function ROUND (X : in REAL ) return REAL;
-- Purpose:
--      Rounds X to the nearest integer value (as real). If X is
--      halfway between two integers, rounding is away from 0.0
-- Special values:
--      ROUND(0.0) = 0.0
-- Domain:
--      X in REAL
-- Error conditions:
--      None
-- Range:
--      ROUND(X) is mathematically unbounded
-- Notes:
--      a) Implementations have to support at least the domain
--          ABS(X) < REAL(INTEGER'HIGH)

function TRUNC (X : in REAL ) return REAL;
-- Purpose:
--      Truncates X towards 0.0 and returns truncated value
-- Special values:
--      TRUNC(0.0) = 0.0

```

```

-- Domain:
--      X in REAL
-- Error conditions:
--      None
-- Range:
--      TRUNC(X) is mathematically unbounded
-- Notes:
--      a) Implementations have to support at least the domain
--          ABS(X) < REAL(INTEGER'HIGH)

function "MOD" (X, Y: in REAL ) return REAL;
-- Purpose:
--      Returns floating point modulus of X/Y, with the same sign as
--      Y, and absolute value less than the absolute value of Y, and
--      for some INTEGER value N the result satisfies the relation
--      X = Y*N + MOD(X,Y)
-- Special values:
--      None
-- Domain:
--      X in REAL; Y in REAL and Y /= 0.0
-- Error conditions:
--      Error if Y = 0.0
-- Range:
--      ABS(MOD(X,Y)) < ABS(Y)
-- Notes:
--      None

function REALMAX (X, Y : in REAL ) return REAL;
-- Purpose:
--      Returns the algebraically larger of X and Y
-- Special values:
--      REALMAX(X,Y) = X when X = Y
-- Domain:
--      X in REAL; Y in REAL
-- Error conditions:
--      None
-- Range:
--      REALMAX(X,Y) is mathematically unbounded
-- Notes:
--      None

function REALMIN (X, Y : in REAL ) return REAL;
-- Purpose:
--      Returns the algebraically smaller of X and Y
-- Special values:
--      REALMIN(X,Y) = X when X = Y
-- Domain:
--      X in REAL; Y in REAL
-- Error conditions:
--      None
-- Range:
--      REALMIN(X,Y) is mathematically unbounded
-- Notes:
--      None

procedure UNIFORM(variable SEED1,SEED2:inout POSITIVE; variable X:out REAL);
-- Purpose:
--      Returns, in X, a pseudo-random number with uniform
--      distribution in the open interval (0.0, 1.0).

```

```
-- Special values:
--     None
-- Domain:
--     1 <= SEED1 <= 2147483562; 1 <= SEED2 <= 2147483398
-- Error conditions:
--     Error if SEED1 or SEED2 outside of valid domain
-- Range:
--     0.0 < X < 1.0
-- Notes:
--     a) The semantics for this function are described by the
--         algorithm published by Pierre L'Ecuyer in "Communications
--         of the ACM," vol. 31, no. 6, June 1988, pp. 742-774.
--         The algorithm is based on the combination of two
--         multiplicative linear congruential generators for 32-bit
--         platforms.
--
--     b) Before the first call to UNIFORM, the seed values
--         (SEED1, SEED2) have to be initialized to values in the range
--         [1, 2147483562] and [1, 2147483398] respectively. The
--         seed values are modified after each call to UNIFORM.
--
--     c) This random number generator is portable for 32-bit
--         computers, and it has a period of  $\sim 2.30584 \times (10^{18})$  for each
--         set of seed values.
--
--     d) For information on spectral tests for the algorithm, refer
--         to the L'Ecuyer article.
```

```
function SQRT (X : in REAL ) return REAL;
-- Purpose:
--     Returns square root of X
-- Special values:
--     SQRT(0.0) = 0.0
--     SQRT(1.0) = 1.0
-- Domain:
--     X >= 0.0
-- Error conditions:
--     Error if X < 0.0
-- Range:
--     SQRT(X) >= 0.0
-- Notes:
--     a) The upper bound of the reachable range of SQRT is
--         approximately given by:
--         SQRT(X) <= SQRT(REAL'HIGH)
```

```
function CBRT (X : in REAL ) return REAL;
-- Purpose:
--     Returns cube root of X
-- Special values:
--     CBRT(0.0) = 0.0
--     CBRT(1.0) = 1.0
--     CBRT(-1.0) = -1.0
-- Domain:
--     X in REAL
-- Error conditions:
--     None
-- Range:
--     CBRT(X) is mathematically unbounded
-- Notes:
```