Advanced Design System 2011.01

Feburary 2011
Signal Converters

© Agilent Technologies, Inc. 2000-2011
5301 Stevens Creek Blvd., Santa Clara, CA 95052 USA
No part of this documentation may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Acknowledgments

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Mentor products and processes are registered trademarks of Mentor Graphics Corporation. * Calibre is a trademark of Mentor Graphics Corporation in the US and other countries. "Microsoft®, Windows®, MS Windows®, Windows NT®, Windows 2000® and Windows Internet Explorer® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Oracle and Java and registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC. FLEXlm is a trademark of Globetrotter Software, Incorporated. Layout Boolean Engine by Klaas Holwerda, v1.7 http://www.xs4all.nl/~kholwerd/bool.html . FreeType Project, Copyright (c) 1996-1999 by David Turner, Robert Wilhelm, and Werner Lemberg. QuestAgent search engine (c) 2000-2002, JObjects. Motif is a trademark of the Open Software Foundation. Netscape is a trademark of Netscape Communications Corporation. Netscape Portable Runtime (NSPR), Copyright (c) 1998-2003 The Mozilla Organization. A copy of the Mozilla Public License is at http://www.mozilla.org/MPL/ . FFTW, The Fastest Fourier Transform in the West, Copyright (c) 1997-1999 Massachusetts Institute of Technology. All rights reserved.

The following third-party libraries are used by the NlogN Momentum solver:

"This program includes Metis 4.0, Copyright © 1998, Regents of the University of Minnesota", http://www.cs.umn.edu/~metis , METIS was written by George Karypis (karypis@cs.umn.edu).

Intel@ Math Kernel Library, http://www.intel.com/software/products/mkl

SuperLU_MT version 2.0 - Copyright © 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy). All rights reserved. SuperLU Disclaimer: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7-zip - 7-Zip Copyright: Copyright (C) 1999-2009 Igor Pavlov. Licenses for files are: 7z.dll: GNU LGPL + unRAR restriction, All other files: GNU LGPL. 7-zip License: This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful,but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. unRAR copyright: The decompression engine for RAR archives was developed using source code of unRAR program.All copyrights to original unRAR code are owned by Alexander Roshal. unRAR License: The unRAR sources cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified unRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver. 7-zip Availability: http://www.7-zip.org/

AMD Version 2.2 - AMD Notice: The AMD code was modified. Used by permission. AMD copyright: AMD Version 2.2, Copyright © 2007 by Timothy A. Davis, Patrick R. Amestoy, and Iain S. Duff. All Rights Reserved. AMD License: Your use or distribution of AMD or any modified version of AMD implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies.User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. AMD Availability: http://www.cise.ufl.edu/research/sparse/amd

UMFPACK 5.0.2 - UMFPACK Notice: The UMFPACK code was modified. Used by permission. UMFPACK Copyright: UMFPACK Copyright © 1995-2006 by Timothy A. Davis. All Rights Reserved. UMFPACK License: Your use or distribution of UMFPACK or any modified version of UMFPACK implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. UMFPACK Availability: http://www.cise.ufl.edu/research/sparse/umfpack UMFPACK (including versions 2.2.1 and earlier, in FORTRAN) is available at http://www.cise.ufl.edu/research/sparse . MA38 is available in the Harwell Subroutine Library. This version of UMFPACK includes a modified form of COLAMD Version 2.0, originally released on Jan. 31, 2000, also available at http://www.cise.ufl.edu/research/sparse . COLAMD V2.0 is also incorporated as a built-in function in MATLAB version 6.1, by The MathWorks, Inc. http://www.mathworks.com . COLAMD V1.0 appears as a column-preordering in SuperLU (SuperLU is available at http://www.netlib.org). UMFPACK v4.0 is a built-in routine in MATLAB 6.5. UMFPACK v4.3 is a built-in routine in MATLAB 7.1.

Qt Version 4.6.3 - Qt Notice: The Qt code was modified. Used by permission. Qt copyright: Qt Version 4.6.3, Copyright (c) 2010 by Nokia Corporation. All Rights Reserved. Qt License: Your use or distribution of Qt or any modified version of Qt implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the
terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies.User
documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. Qt Availability: http://www.qtsoftware.com/downloads Patches Applied to Qt can be found in the installation at: $HPEESOF_DIR/prod/licenses/thirdparty/qt/patches. You may also contact Brian Buchanan at Agilent Inc. at brian_buchanan@agilent.com for more information.

The HiSIM_HV source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code, is owned by Hiroshima University and/or STARC.

Errata The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

Warranty The material contained in this document is provided "as is", and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this documentation and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license. Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at http://systemc.org/ . This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

Restricted Rights Legend U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

About Signal Converters

The signal converter components provide conversions between the various signal types in ADS Ptolemy and process scalar or matrix data that are integer, double precision floating point (real), fixed-point (fixed), or complex values. Each component accepts a specific class of signal and outputs a resultant signal. These components do not accept any matrix class of signal.

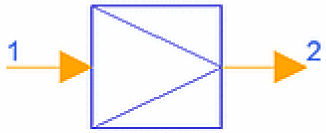
If a component receives another class of signal, the received signal is converted automatically to the signal class specified as the input of the component. Auto conversion from a higher to a lower precision signal class can result in loss of information.

The auto conversion from timed, complex or floating-point (real) signals to a fixed signal uses a default bit width of 54 bits with the minimum number of integer bits needed to represent the value in twos complement representation. For example, the auto conversion of the floating-point (real) value of 1.0 creates a fixed-point value with precision of 2.52; a value of 0.5 would create one of precision of 1.53. For details on conversions between different classes of signals, refer to *Conversion of Data Types* (ptolemy) in the *ADS Ptolemy Simulation* documentation.

Some components operate with fixed-point numbers. These components use one or more parameters that define the characteristics of the fixed-point processing. These parameters include: OverflowHandler, OutputPrecision, RoundFix, ReportOverflow, and others.

For details, refer to *Parameters for Fixed-Point Components* (ptolemy) in the *ADS Ptolemy Simulation* documentation.

BitsToInt



Description Bits to Integer
Library Signal Converters
Class SDFBitsToInt
C++ Code

Parameters

Name	Description	Default	Type	Range
nBits	number of bits read per execution	4	int	[1, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int

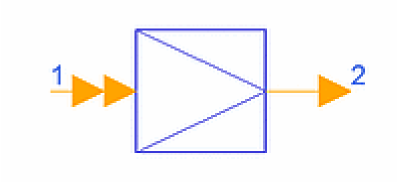
Pin Outputs

Pin	Name	Description	Signal Type
2	output		int

Notes/Equations

1. The integer input sequence is interpreted as a bit stream in which any nonzero value is interpreted as to mean a *one* bit. BitsToInt consumes nBits successive bits from the input, packs these into an integer, and outputs the resulting integer. The first received bit becomes the most significant bit of the output.
2. If nBits is larger than the integer word size, the first bits received is lost; if nBits is smaller than the wordsize minus one, the output integer is always non-negative.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

BusToNum



Description Bus to Number
Library Signal Converters
Class SDFBusToNum
C++ Code

Parameters

Name	Description	Default	Type
Previous	previous value of output	0	int

Pin Inputs

Pin	Name	Description	Signal Type
1	input		multiple int

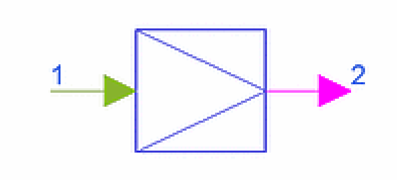
Pin Outputs

Pin	Name	Description	Signal Type
2	output		int

Notes/Equations

1. BusToNum accepts a number of input bit streams, where this number should not exceed the wordsize of an integer. Each bit stream has integer data with values 0, 3, or anything else; these are interpreted as binary 0, tri-state, or 1, respectively. When the component simulates, it reads one input bit from each input.
2. If any of the input bits is tri-stated, the output is the previous output (or the initial value of the Previous parameter if the simulation is the first one). Otherwise, the bits are assembled into an integer word, assuming twos complement encoding, and sign extended. The resulting signed integer is sent to the output.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

CxToFix



Description Complex to Fixed-Point
Library Signal Converters
Class SDFCxToFix
Derived From SDFFix
C++ Code

Parameters

Name	Description	Default	Type
OverflowHandler	output overflow characteristic: wrapped, saturate, zero_saturate, warning	wrapped	enum
ReportOverflow	simulation overflow error report option: DONT_REPORT, REPORT	REPORT	enum
RoundFix	fixed-point computations, assignments, and data type conversions option: TRUNCATE, ROUND	TRUNCATE	enum
OutputPrecision	precision of output in bits and accumulation	2.14	precision

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input complex type	complex

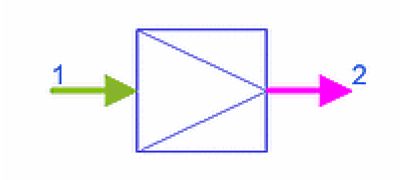
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output fix type	fix

Notes/Equations

1. CxToFix converts a complex input to a fixed-point output. The magnitude of the complex input is calculated and then converted to a fixed-point number with the given OutputPrecision.
2. If the fixed-point operations cannot fit into the precision specified, overflow occurs with the overflow characteristic specified by OverflowHandler. If ReportOverflow = REPORT, after the simulation has finished the number of overflow errors (if any) is reported. RoundFix identifies whether fixed-point calculations are truncate or round method. For details, refer to *Parameters for Fixed-Point Components* (ptolemy) in the *ADS Ptolemy Simulation* documentation.
3. This component uses twos-complement arithmetic; OutputPrecision parameter values must specify at least 1 bit to the left of the decimal place (used as a sign bit).
4. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

CxToFix_M



Description Complex to Fixed-Point Matrix
Library Signal Converters
Class SDFCxToFix_M
Derived From SDFFix

Parameters

Name	Description	Default	Type
OverflowHandler	output overflow characteristic: wrapped, saturate, zero_saturate, warning	wrapped	enum
ReportOverflow	simulation overflow error report option: DONT_REPORT, REPORT	REPORT	enum
RoundFix	fixed-point computations, assignments, and data type conversions option: TRUNCATE, ROUND	TRUNCATE	enum
OutputPrecision	precision of output in bits and accumulation	2.14	precision

Pin Inputs

Pin	Name	Description	Signal Type
1	input		complex matrix

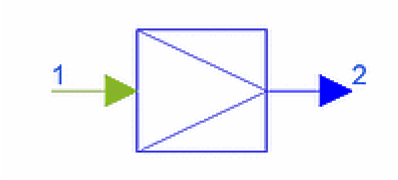
Pin Outputs

Pin	Name	Description	Signal Type
2	output		fix matrix

Notes/Equations

1. CxToFix_M converts an input complex matrix to a fixed-point matrix by calculating the magnitude of the elements in the input matrix and then converting the magnitudes to fixed-point numbers with the given OutputPrecision.
2. If the fixed-point operations cannot fit into the precision specified, overflow occurs with the overflow characteristic specified by OverflowHandler. If ReportOverflow = REPORT, after the simulation has finished the number of overflow errors (if any) is reported. RoundFix identifies whether fixed-point calculations are truncate or round method.
   For details, refer to *Parameters for Fixed-Point Components* (ptolemy) in the *ADS Ptolemy Simulation* documentation.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

CxToFloat



Description Complex to Floating-Point
Library Signal Converters
Class SDFCxToFloat
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input complex type	complex

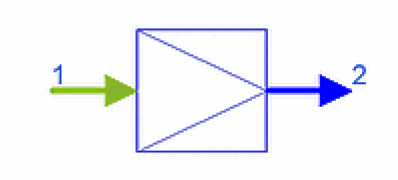
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output float type	real

Notes/Equations

1. CxToFloat converts a complex input to a floating-point (real) output. The output is the absolute of the input complex number.
   output – abs (input)
   Given a complex number
   $C = a + bj$
   the abs() function is defined as
   $abs(C) = \sqrt{a^2 + b^2}$
   The complex number is not converted to floating-point by discarding the imaginary component as might be expected.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

CxToFloat_M



Description Complex to Floating-Point(real) Matrix
Library Signal Converters
Class SDFCxToFloat_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		complex matrix

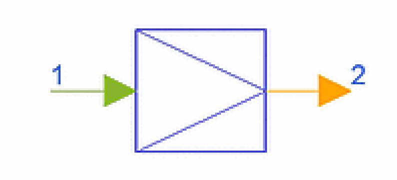
Pin Outputs

Pin	Name	Description	Signal Type
2	output		real matrix

Notes/Equations

1. CxToFloat_M converts an input complex matrix to a floating-point (real) matrix. The conversion results in a floating-point (real) matrix with entries that are the absolute value of each corresponding entry of the complex matrix being converted.
   $$FloatMatrix\ entry(i) = abs(ComplexMatrix\ entry(i))$$
   Where for a complex number
   $$C = a + bj$$
   the abs() function is defined as
   $$abs(C) = \sqrt{a^2 + b^2}$$
   The complex entries are not converted to floating-point by discarding the imaginary component as might be expected.
2. For general information regarding signal converter components, refer to the *About Signal Converters* (sigconverters).

CxToInt



Description Complex to Integer
Library Signal Converters
Class SDFCxToInt
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input complex type	complex

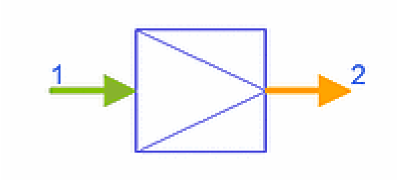
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output int type	int

Notes/Equations

1. CxToInt converts a complex input to an integer output. The output is generated by first converting the complex number to a floating-point (real) (see *CxToFloat* (sigconverters)) and then casting the result:
output = int (abs(input))
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

CxToInt_M



Description Complex to Integer Matrix
Library Signal Converters
Class SDFCxToInt_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		complex matrix

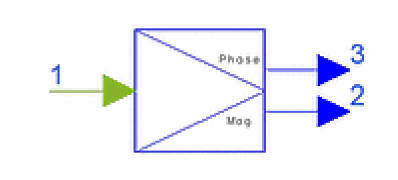
Pin Outputs

Pin	Name	Description	Signal Type
2	output		int matrix

Notes/Equations

1. CxToInt_M converts an input complex matrix to an integer matrix. The conversion results in an integer matrix with entries that are the integer portion of the absolute value of each corresponding entry of the complex matrix being converted.
   $IntegerMatrix\ entry(i) = int(abs(ComplexMatrix\ entry(i)))$
   Where, for a complex number
   $C = a + bj$
   the abs() function is defined as
   $abs(C) = \sqrt{a^2 + b^2}$
   The complex entries are not converted by discarding the imaginary component as might be expected.
2. For general information regarding signal converter components, refer to the *About Signal Converters* (sigconverters).

CxToPolar



Description Complex to Magnitude and Phase
Library Signal Converters
Class SDFCxToPolar
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input		complex

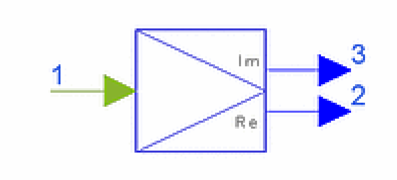
Pin Outputs

Pin	Name	Description	Signal Type
2	magnitude		real
3	phase		real

Notes/Equations

1. CxToPolar outputs the magnitude and angle of a complex number:
   Mag = | input|
   Phase = $\angle$ input (the angle is in radians)
2. For general information regarding signal converter components, refer to the *About Signal Converters* (sigconverters).

CxToRect



Description Complex to Real and Imaginary
Library Signal Converters
Class SDFCxToRect
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input		complex

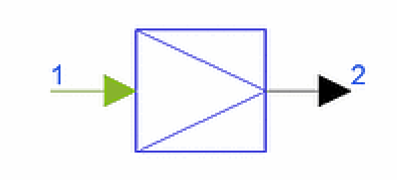
Pin Outputs

Pin	Name	Description	Signal Type
2	real		real
3	imag		real

Notes/Equations

1. CxToRect splits the input complex signal into its real (Re) and imaginary (Im) parts.

CxToTimed



Description Complex to Timed
Library Signal Converters
Class TSDFCxToTimed

Parameters

Name	Description	Default	Unit	Type	Range
TStep	output time step	0.0	sec	real	[0, ∞)
FCarrier	output carrier frequency	1.0e9	Hz	real	(0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	complex

Pin Outputs

Pin	Name	Description	Signal Type
2	output	output signal	timed

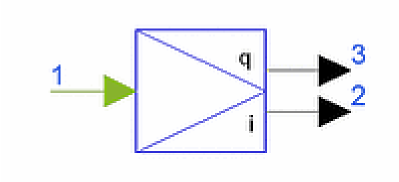
Parameters

Name	Description	Value Range
TStep	output time step	[0, +∞)
FCarrier	output carrier frequency	(0, +∞)

Notes/Equations

1. CxToTimed converts a complex signal $x[n]$ to a timed signal $y[nT]$, where T is equal to TStep.
   Given the complex number (a + jb) at input, the output is a complex envelope timed signal ((I + jQ), fc) where I = a, Q = b, and fc = FCarrier. Effectively, this converter is a modulator.
2. The timed output pin has an effective output resistance of 0 ohm.
3. Before the ADS 2002C release, this component enabled FCarrier to be 0 generating a complex envelope signal with characterization frequency equal to 0. Beginning with the ADS2002C release, FCarrier accepts positive values only. To assign a timestamp to a numeric complex signal so that it can be stored in a TimedSink and plotted in the data display with time as the X-axis, set FCarrier to any positive value.
4. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

CxToTimedIQ



Description complex to baseband timed I and Q
Library Signal Converters
Class TSDFCxToTimedIQ

Parameters

Name	Description	Default	Unit	Type	Range
TStep	output time step	0.0	sec	real	[0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	complex

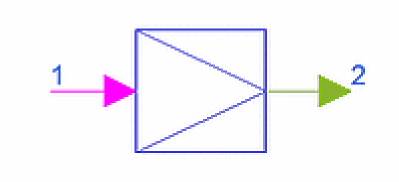
Pin Outputs

Pin	Name	Description	Signal Type
2	Iout	output I baseband signal	timed
3	Qout	output Q baseband signal	timed

Notes/Equations

1. CxToTimedIQ converts a complex input signal to two timed baseband signals. The time step of the output signals is specified by the TStep parameter. The real part of the input signal is output at the Iout pin; the imaginary part of the input signal is output at the Qout pin. This converter is equivalent to a CxToRect converter followed by a FloatIQToTimedIQ converter.
2. The timed output pins have an effective output resistance of 0 ohm.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FixToCx



Description Fixed-Point to Complex
Library Signal Converters
Class SDFFixToCx
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input fix type	fix

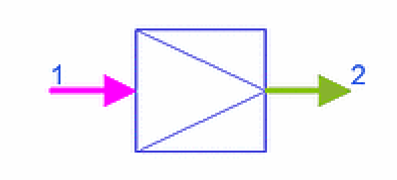
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output complex type	complex

Notes/Equations

1. FixToCx converts a fixed-point input to a complex output. The conversion results in a complex number with the real part the double-precision representation of the fixed-point input and imaginary part equal to 0.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FixToCx_M



Description Fixed-Point to Complex Matrix
Library Signal Converters
Class SDFFixToCx_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		fix matrix

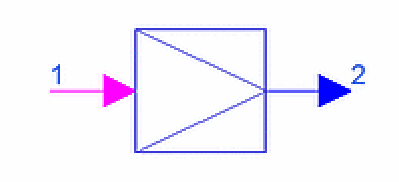
Pin Outputs

Pin	Name	Description	Signal Type
2	output		complex matrix

Notes/Equations

1. FixToCx_M converts an input fixed-point matrix to a complex matrix. The conversion results in a complex matrix with real value entries that are the double-precision representation of each corresponding entry of the fixed-point matrix. The imaginary values of the resulting complex matrix are 0.
2. For general information regarding signal converter components, refer to the *About Signal Converters* (sigconverters).

FixToFloat



Description Fixed-Point to Floating-Point
Library Signal Converters
Class SDFFixToFloat
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input fix type	fix

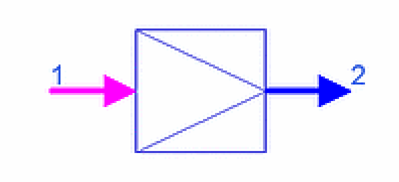
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output float type	real

Notes/Equations

1. FixToFloat converts a fixed-point input to a floating-point (real) output.
2. For general information regarding signal converter components, refer to the *About Signal Converters* (sigconverters).

FixToFloat_M



Description Fixed-Point to Floating-Point Matrix
Library Signal Converters
Class SDFFixToFloat_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		fix matrix

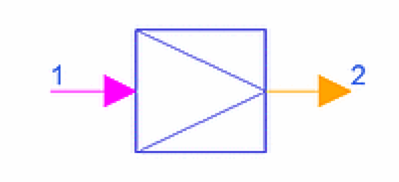
Pin Outputs

Pin	Name	Description	Signal Type
2	output		real matrix

Notes/Equations

1. FixToFloat_M converts a fixed-point matrix to a floating-point (real) matrix. The conversion results in a floating-point (real) matrix with entries that are the double-precision representation of each corresponding entry of the fixed-point matrix.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FixToInt



Description Fixed-Point to Integer
Library Signal Converters
Class SDFFixToInt
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input fix type	fix

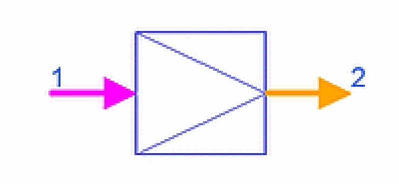
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output int type	int

Notes/Equations

1. FixToInt converts a fixed-point input to an integer output.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FixToInt_M



Description Fixed-Point to Integer Matrix
Library Signal Converters
Class SDFFixToInt_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		fix matrix

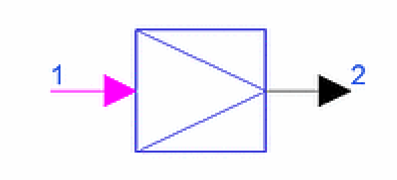
Pin Outputs

Pin	Name	Description	Signal Type
2	output		int matrix

Notes/Equations

1. FixToInt_M converts a fixed-point matrix and to an integer matrix. The conversion results in a integer matrix with entries that are the integer representation of each corresponding entry of the fixed-point matrix.
2. For general information regarding signal converter components, refer to the *Introduction* (sigconverters).

FixToTimed



Description Fixed-Point to Timed
Library Signal Converters
Class TSDFFixToTimed

Parameters

Name	Description	Default	Unit	Type	Range
TStep	output time step	0.0	sec	real	[0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input		fix

Pin Outputs

Pin	Name	Description	Signal Type
2	output		timed

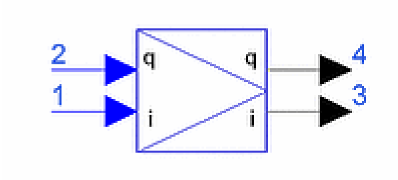
Parameters

Name	Description	Value Range
TStep	output time step	[0, +∞)

Notes/Equations

1. FixToTimed converts a fixed-point input signal $x[n]$ to a timed signal $y[nT]$ where T is equal to TStep. The output y is a real baseband timed signal.
2. The timed output pin has an effective output resistance of 0 ohm.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FloatIQToTimedIQ



Description floating-point I and Q to baseband timed I and Q
Library Signal Converters
Class TSDFFloatIQToTimedIQ

Parameters

Name	Description	Default	Unit	Type	Range
TStep	output time step	0.0	sec	real	[0, ∞)

Pin Outputs

Pin	Name	Description	Signal Type
1	Iin	input I signal	real
2	Qin	input Q signal	real

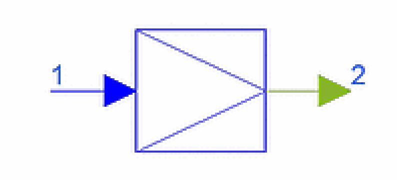
Pin Inputs

Pin	Name	Description	Signal Type
3	Iout	output I signal	timed
4	Qout	output Q signal	timed

Notes/Equations

1. FloatIQToTimedIQ converts the two floating-point (real) input signals to two timed baseband signals. The time step of the output signals is specified by the TStep parameter. The floating-point (real) input signal at pin Iin is converted to a timed baseband signal at pin Iout and the floating-point (real) input signal at pin Qin is converted to a timed baseband signal at pin Qout.
   This converter is equivalent to two FloatToTimed converters in parallel (one connected between pins Iin and Iout and the other connected between pins Qin and Qout).
2. The timed output pins have an effective output resistance of 0 ohm.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FloatToCx



Description Floating-Point to Complex
Library Signal Converters
Class SDFFloatToCx
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input float type	real

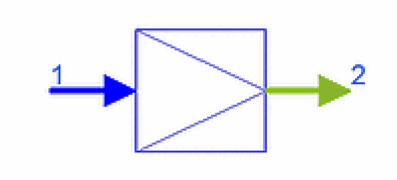
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output complex type	complex

Notes/Equations

1. FloatToCx converts a floating-point (real) input to a complex output. The conversion results in a complex number with the real part equal to input and imaginary part equal to 0.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FloatToCx_M



Description Floating-Point to Complex Matrix
Library Signal Converters
Class SDFFloatToCx_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		real matrix

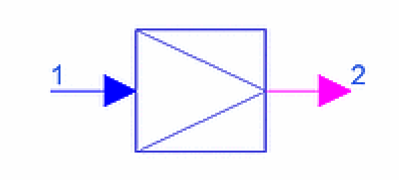
Pin Outputs

Pin	Name	Description	Signal Type
2	output		complex matrix

Notes/Equations

1. FloatToCx_M converts a floating-point (real) matrix to a complex matrix. The conversion results in a complex matrix with real value entries that are the corresponding entries of the floating-point (real) matrix. The imaginary values of the resulting complex matrix are 0.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FloatToFix



Description Floating-Point to Fixed-Point
Library Signal Converters
Class SDFFloatToFix
Derived From SDFFix
C++ Code

Parameters

Name	Description	Default	Type
OverflowHandler	output overflow characteristic: wrapped, saturate, zero_saturate, warning	wrapped	enum
ReportOverflow	simulation overflow error report option: DONT_REPORT, REPORT	REPORT	enum
RoundFix	fixed-point computations, assignments, and data type conversions option: TRUNCATE, ROUND	TRUNCATE	enum
OutputPrecision	precision of output in bits and accumulation	2.14	precision

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input float type	real

Pin Outputs

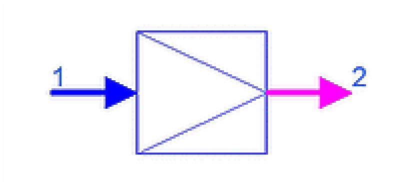
Pin	Name	Description	Signal Type
2	output	Output fix type	fix

Notes/Equations

1. FloatToFix converts a floating-point (real) input to a fixed-point output with the given OutputPrecision.
2. If the fixed-point operations cannot fit into the precision specified, overflow occurs with the overflow characteristic specified by OverflowHandler. If ReportOverflow = REPORT, after the simulation has finished the number of overflow errors (if any) is reported. RoundFix identifies whether fixed-point calculations are truncate or round method.
   For details, refer to Parameters for Fixed Point Components, in the *ADS Ptolemy Simulation* documentation.
3. This component uses twos-complement arithmetic; OutputPrecision parameter values must specify at least one bit to the left of the decimal place (used as a sign bit).
4. If unsigned arithmetic is desired, use a FloatToFixSyn component instead of FloatToFix.

5. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FloatToFix_M



Description Floating-Point to Fixed-Point Matrix
Library Signal Converters
Class SDFFloatToFix_M
Derived From SDFFix

Parameters

Name	Description	Default	Type
OverflowHandler	output overflow characteristic: wrapped, saturate, zero_saturate, warning	wrapped	enum
ReportOverflow	simulation overflow error report option: DONT_REPORT, REPORT	REPORT	enum
RoundFix	fixed-point computations, assignments, and data type conversions option: TRUNCATE, ROUND	TRUNCATE	enum
OutputPrecision	precision of output in bits and accumulation	2.14	precision

Pin Inputs

Pin	Name	Description	Signal Type
1	input		real matrix

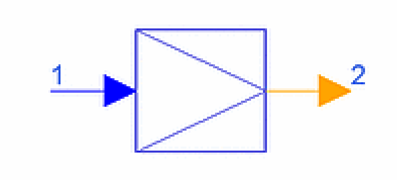
Pin Outputs

Pin	Name	Description	Signal Type
2	output		fix matrix

Notes/Equations

1. FloatToFix_M converts a floating-point (real) matrix to a fixed-point matrix by converting the floating-point numbers in the input matrix to fixed-point numbers with the given OutputPrecision.
2. If the fixed-point operations cannot fit into the precision specified, overflow occurs with the overflow characteristic specified by OverflowHandler. If ReportOverflow = REPORT, after the simulation has finished the number of overflow errors (if any) is reported. RoundFix identifies whether fixed-point calculations are truncate or round method.
   For details, refer to *Parameters for Fixed-Point Components* (ptolemy) in the *ADS Ptolemy Simulation* documentation.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FloatToInt



Description Floating-Point to Integer
Library Signal Converters
Class SDFFloatToInt
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input float type	real

Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output int type	int

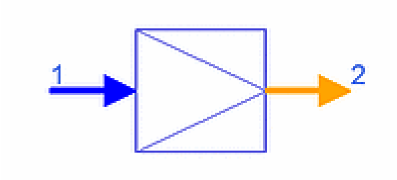
Notes/Equations

1. FloatToInt converts a floating-point (real) input to an integer output. The conversion occurs by taking the integer portion (truncating the fractional part) of the floating-point number. For example:

   1.1, 1.221, 1.5, 1.673, 1.8961, and 1.974 are all converted to 1;
   −2.03, −2.324, −2.59, −2.71, −2.85, and −2.97 are all converted to −2.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

anchor:1105296:TopicAlias:FloatToInt_M}

FloatToInt_M



Description Floating-Point to Integer Matrix
Library Signal Converters
Class SDFFloatToInt_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		real matrix

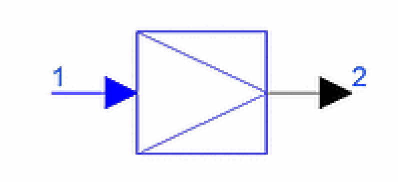
Pin Outputs

Pin	Name	Description	Signal Type
2	output		int matrix

Notes/Equations

1. FloatToInt_M converts a floating-point (real) matrix to an integer matrix. The conversion results in an integer matrix with entries that are the integer portion of each corresponding entry of the floating-point (real) matrix.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

FloatToTimed



Description Floating-Point to Timed
Library Signal Converters
Class TSDFFloatToTimed

Parameters

Name	Description	Default	Unit	Type	Range
TStep	output time step	0.0	sec	real	[0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	real

Pin Outputs

Pin	Name	Description	Signal Type
2	output	output signal	timed

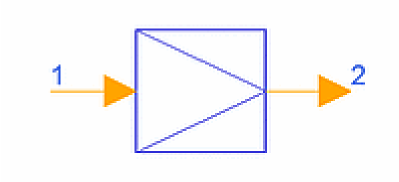
Notes/Equations

1. FloatToTimed converts a floating-point (real) signal to a timed signal. Given the floating-point (real) number $x[n]$ at input, the output is a real baseband timed signal $y(t)$ with

   $$y(nT) = x[n]$$

   where T is the input signal time step and $n = 1, 2, \ldots$
2. The timed output pin has an effective output resistance of 0 ohm.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

IntToBits



Description Integer to Bits
Library Signal Converters
Class SDFIntToBits
C++ Code

Parameters

Name	Description	Default	Type	Range
nBits	number of bits written per execution	4	int	[1, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int

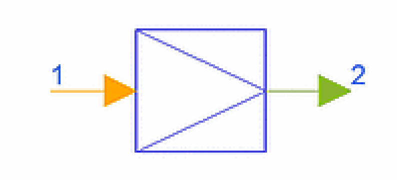
Pin Outputs

Pin	Name	Description	Signal Type
2	output		int

Notes/Equations

1. IntToBits reads the least significant nBits from an integer input, and outputs the bits as integers serially on the output, most significant bit first.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

IntToCx



Description* Integer to Complex
Library Signal Converters
Class SDFIntToCx
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input integer type	int

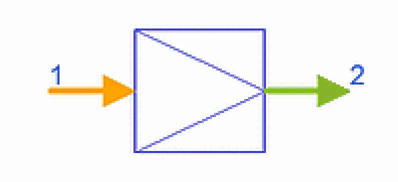
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output complex type	complex

Notes/Equations

1. IntToCx converts an integer input to a complex output. The conversion results in a complex number with the real part equal to double-precision representation of input and the imaginary part equal to 0.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

IntToCx_M



Description Integer to Complex Matrix
Library Signal Converters
Class SDFIntToCx_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int matrix

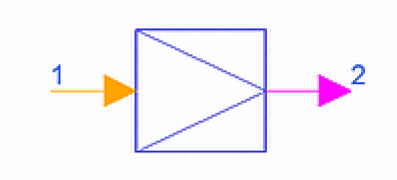
Pin Outputs

Pin	Name	Description	Signal Type
2	output		complex matrix

Notes/Equations

1. IntToCx_M converts an input integer matrix to a complex matrix, where real value entries are the double-precision representation of the corresponding entries of the integer matrix. The imaginary values of the resulting complex matrix are 0.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

IntToFix



Description* Integer to Fixed-Point
Library Signal Converters
Class SDFIntToFix
Derived From SDFFix
C++ Code

Parameters

Name	Description	Default	Type
OverflowHandler	output overflow characteristic: wrapped, saturate, zero_saturate, warning	wrapped	enum
ReportOverflow	simulation overflow error report option: DONT_REPORT, REPORT	REPORT	enum
RoundFix	fixed-point computations, assignments, and data type conversions option: TRUNCATE, ROUND	TRUNCATE	enum
OutputPrecision	precision of output in bits and accumulation	16.0	precision

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input integer type	int

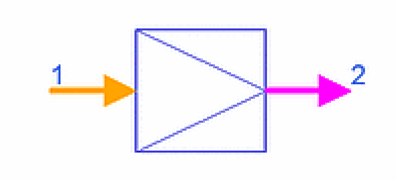
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output fix type	fix

Notes/Equations

1. IntToFix converts an integer input to a fixed-point output with the given OutputPrecision.
2. If fixed-point operations cannot fit into the precision specified, overflow occurs with the characteristic specified by OverflowHandler. If ReportOverflow = REPORT, the number of overflow errors (if any) is reported after simulation. RoundFix identifies whether fixed-point calculations are truncate or round.
   For details, refer to*Parameters for Fixed-Point Components* (ptolemy) in the *ADS Ptolemy Simulation* documentation.
3. This component uses twos-complement arithmetic; OutputPrecision parameter values must specify at least one bit to the left of the decimal place (used as a sign bit).
4. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

IntToFix_M



Description Integer to Fixed-Point Matrix
Library Signal Converters
Class SDFIntToFix_M
Derived From SDFFix

Parameters

Name	Description	Default	Type
OverflowHandler	output overflow characteristic: wrapped, saturate, zero_saturate, warning	wrapped	enum
ReportOverflow	simulation overflow error report option: DONT_REPORT, REPORT	REPORT	enum
RoundFix	fixed-point computations, assignments, and data type conversions option: TRUNCATE, ROUND	TRUNCATE	enum
OutputPrecision	precision of output in bits and accumulation	16.0	precision

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int matrix

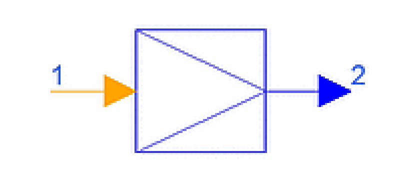
Pin Outputs

Pin	Name	Description	Signal Type
2	output		fix matrix

Notes/Equations

1. IntToFix_M converts an input integer matrix to a fixed-point matrix by converting the integer numbers in the input matrix to fixed-point numbers with the given OutputPrecision.
2. If fixed-point operations cannot fit into the precision specified, overflow occurs with the characteristic specified by OverflowHandler. If ReportOverflow = REPORT, the number of overflow errors (if any) is reported after simulation. RoundFix identifies whether fixed-point calculations are truncate or round.
   For details, refer to *Parameters for Fixed-Point Components* (ptolemy) in the *ADS Ptolemy Simulation* documentation.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

IntToFloat



Description Integer to Floating-Point
Library Signal Converters
Class SDFIntToFloat
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input	Input integer type	int

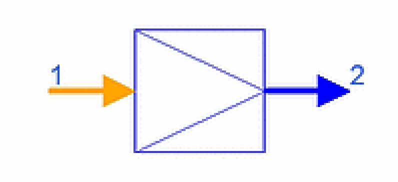
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output float type	real

Notes/Equations

1. IntToFloat converts an integer input to a floating-point (real) output.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

IntToFloat_M



Description Integer to Floating-Point Matrix
Library Signal Converters
Class SDFIntToFloat_M
Derived From MatrixBase

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int matrix

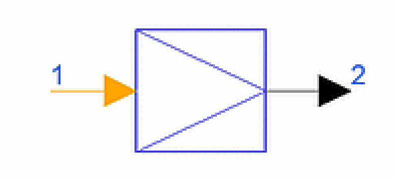
Pin Outputs

Pin	Name	Description	Signal Type
2	output		real matrix

Notes/Equations

1. IntToFloat_M converts an input integer matrix to a floating-point (real) matrix, where each element of the output matrix is the double-precision floating-point representation of the corresponding element in the input matrix.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

IntToTimed



Description Integer to Timed
Library Signal Converters
Class TSDFIntToTimed

Parameters

Name	Description	Default	Unit	Type	Range
TStep	output time step	0.0	sec	real	[0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	int

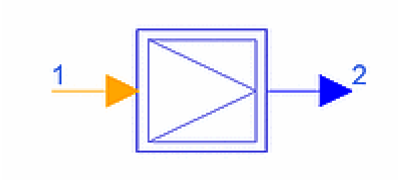
Pin Outputs

Pin	Name	Description	Signal Type
2	output	output signal	timed

Notes/Equations

1. IntToTimed converts an integer signal to a timed signal. Given the integer number $x[n]$ at input, the output is a real baseband timed signal $y(t)$ with
   $y(nT) = x[n]$
   where T is the input signal time step and $n = 1, 2, \ldots$
2. The timed output pin has an effective output resistance of 0 ohm.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

LogicToNRZ



Description Logic to NRZ Format
Library Signal Converters
Class SDFLogicToNRZ

Parameters

Name	Description	Default	Type	Range
Amplitude	amplitude of NRZ signal	1.0	real	(-∞, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int

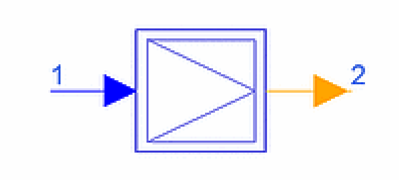
Pin Outputs

Pin	Name	Description	Signal Type
2	output		real

Notes/Equations

1. Converts a logic level to NRZ level. An input Logic 0 produces a -Amplitude output; an input Logic 1 produces a +Amplitude output.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

NRZToLogic



Description NRZ to Logic Format
Library Signal Converters
Class SDFNRZToLogic

Parameters

Name	Description	Default	Type	Range
Amplitude	Amplitude	1.0	real	(-∞, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input		real

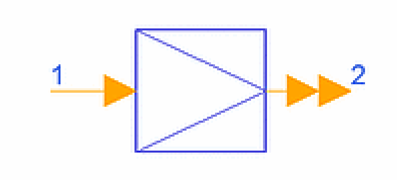
Pin Outputs

Pin	Name	Description	Signal Type
2	output		int

Notes/Equations

1. Converts an NRZ level to Logic level. An input ≥0 produces an output Logic 1; an input <0 produces an output Logic 0.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

NumToBus



Description Number to Bus
Library Signal Converters
Class SDFNumToBus
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int

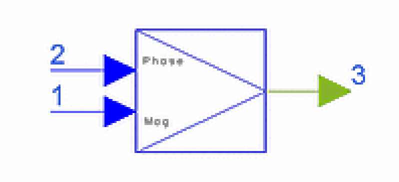
Pin Outputs

Pin	Name	Description	Signal Type
2	output		multiple int

Notes/Equations

1. NumToBus converts an integer to binary form in terms of bits to be output on the parallel bus. The most significant bit is the sign bit. The output can be observed by connecting a BusSplit component.
   For example, integer values from 0 to 7 are input. BusSplit4 is used to access the 4 output bits. Then integer values 0 to 7 are converted into the binary forms of 0000, 0001, 0010, 0011, 0100, 0101, 0110, and 0111.
2. Also see: BusSplit components
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

PolarToCx



Description Magnitude and Phase to Complex
Library Signal Converters
Class SDFPolarToCx
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	magnitude		real
2	phase		real

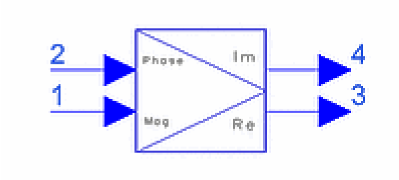
Pin Outputs

Pin	Name	Description	Signal Type
3	output		complex

Notes/Equations

1. PolarToCx converts a complex number from its polar representation to its Cartesian form. The angle must be specified in radians.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

PolarToRect



Description* Magnitude and Phase to Rectangular
Library Signal Converters
Class SDFPolarToRect
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	magnitude		real
2	phase		real

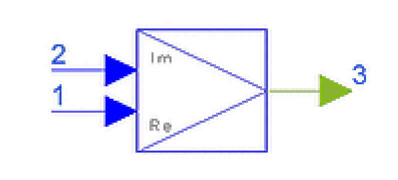
Pin Outputs

Pin	Name	Description	Signal Type
3	x		real
4	y		real

Notes/Equations

1. PolarToRect converts two floating-point (real) inputs representing a complex number in polar form to two floating-point (real) outputs representing a complex number in the rectangular form. The phase is assumed to be in radians.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

RectToCx



Description Real and Imaginary to Complex
Library Signal Converters
Class SDFRectToCx
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	real		real
2	imag		real

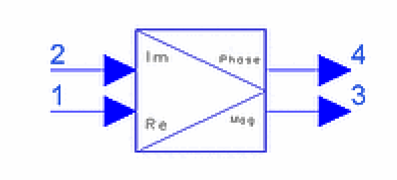
Pin Outputs

Pin	Name	Description	Signal Type
3	output		complex

Notes/Equations

1. RectToCx converts its real (Re) and imaginary (Im) inputs to a complex output.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

RectToPolar



Description Rectangular to Polar
Library Signal Converters
Class SDFRectToPolar
C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	x		real
2	y		real

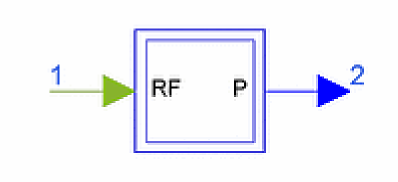
Pin Outputs

Pin	Name	Description	Signal Type
3	magnitude		real
4	phase		real

Notes/Equations

1. RectToPolar converts two floating-point (real) inputs representing a complex number in rectangular form to two floating-point (real) outputs representing a complex number in polar form (magnitude and phase). The phase output is in the range $-\pi$ to π.
2. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

RFtoPower



(Description RF signal envelope to power converter
Library Signal Converters
Class TSDFRFtoPower

Parameters

Name	Description	Default	Unit	Type	Range
RefR	RF signal reference resistance	50	Ohm	real	(0, ∞)
NumStart	sample number to start integration for power in watts	0		int	[0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	Env	RF signal complex envelope	complex

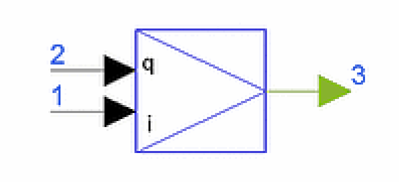
Pin Outputs

Pin	Name	Description	Signal Type
2	P	power value in watts	real

Notes/Equations

1. RFtoPower converts an input complex signal representing an RF signal I, Q envelope to average power in Watts.
2. The input RF complex envelope is squared and divided by 2 × RefR to obtain the instantaneous power.
3. Output P is the integrated instantaneous power (with integration beginning as sample NumStart) divided by the number of samples recorded.
4. Use of this component is demonstrated in (access from ADS Main window) *File* > *Open* > *Example* > PtolemyDocExamples > Timed_RF_Subsystems_wrk_ . Open the networks design *RF_PAE_example* and push into RF_PAE_TestFixture that uses RFtoPower.
5. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

TimedIQToCx



Description baseband timed I and Q to complex
Library Signal Converters
Class TSDFTimedIQToCx

Pin Inputs

Pin	Name	Description	Signal Type
1	Iin	input I baseband signal	timed
2	Qin	input Q baseband signal	timed

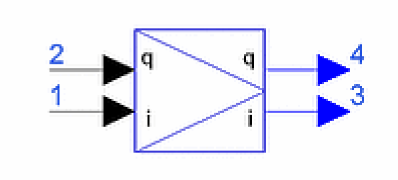
Pin Outputs

Pin	Name	Description	Signal Type
3	output	output signal	complex

Notes/Equations

1. TimedIQToCx converts the two timed baseband input signals to a complex signal. The signal at pin Iin becomes the real part of the complex output signal and the signal at pin Qin becomes the imaginary part of the complex output signal. This converter is equivalent to a TimedIQToFloatIQ converter followed by a RectToCx converter.
2. This component has infinite input impedance.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

TimedIQToFloatIQ



Description baseband timed I and Q to floating-point I and Q
Library Signal Converters
Class TSDFTimedIQToFloatIQ

Pin Inputs

Pin	Name	Description	Signal Type
1	Iin	input I signal	timed
2	Qin	input Q signal	timed

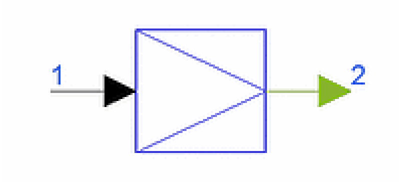
Pin Outputs

Pin	Name	Description	Signal Type
3	Iout	output I signal	real
4	Qout	output Q signal	real

Notes/Equations

1. TimedIQToFloatIQ converts the two timed baseband input signals to two floating-point (real) signals. The timed baseband input signal at pin Iin is converted to a floating-point (real) signal at pin Iout and the timed baseband input signal at pin Qin is converted to a floating-point (real) signal at pin Qout.
   This converter is equivalent to two TimedToFloat converters in parallel (one connected between pins Iin and Iout and the other connected between pins Qin and Qout). The difference between this converter and a pair of TimedToFloat converters in parallel is that the TimedToFloat converters can accept timed RF and signals and convert them to floating-point (real) signals, where the TimedIQToFloatIQ converter accepts timed baseband signals only at its inputs.
2. This component has infinite input impedance.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

TimedToCx



Description Timed to Complex
Library Signal Converters
Class TSDFTimedToCx

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

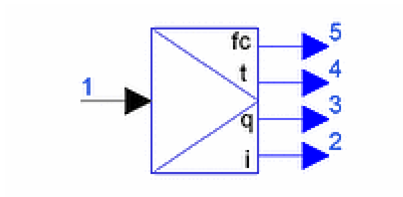
Pin Outputs

Pin	Name	Description	Signal Type
2	output	output signal	complex

Notes/Equations

1. TimedToCx converts a timed signal to a complex signal. If the timed signal is a complex envelope signal, the output is (I + jQ). Likewise, when the timed signal is a real baseband signal, the output is the complex number (R + jS) where R is the real baseband signal and S is 0.
2. This component has infinite input impedance.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

TimedToData



Description Timed to data: i, q, time, fc
Library Signal Converters
Class TSDFTimedToData

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

Pin Outputs

Pin	Name	Description	Signal Type
2	i	i envelope	real
3	q	q envelope	real
4	t	time	real
5	fc	characterization frequency	real

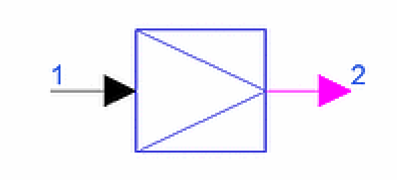
Notes/Equations

1. TimedToData converts a timed input signal $x(t)$ to its constituent data members $\{i, q, t, F_c\}$. An input timed RF signal is represented as:

   $$x(t) = I(t)\cos(2\pi F_c t) - Q(t)\sin(2\pi F_c t)$$

   An input baseband timed signal is simply:
   $x(t) = I(t)$
   with $Q = 0$ and $F_c = 0$.
2. This component has infinite input impedance.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

TimedToFix



Description Timed to Fixed
Library Signal Converters
Class TSDFTimedToFix
Derived From TSDFFix

Parameters

Name	Description	Default	Type
OverflowHandler	output overflow characteristic: wrapped, saturate, zero_saturate, warning	wrapped	enum
ReportOverflow	simulation overflow error report option: DONT_REPORT, REPORT	REPORT	enum
RoundFix	fixed-point calculations, assignments, and data type conversion option: TRUNCATE, ROUND	TRUNCATE	enum
OutputPrecision	precision of output in bits and accumulation	2.14	precision

Pin Inputs

Pin	Name	Description	Signal Type
1	input		timed

Pin Outputs

Pin	Name	Description	Signal Type
2	output		fix

Notes/Equations

1. TimedToFix converts a timed input signal $x(t) = \{I(t), Q(t), F_c\}$ (either baseband or complex envelope flavors) to a fixed-point output $y[n]$ with the given OutputPrecision. The conversion rule is
for complex envelope:

   $$y[n] = (\mathrm{fix})\{\mathrm{I}(nt) \times cos(2\pi F_c nT) - Q(nT) \times \sin(2\pi F_c nT)\}$$

   where T is the input signal time step

   for baseband:

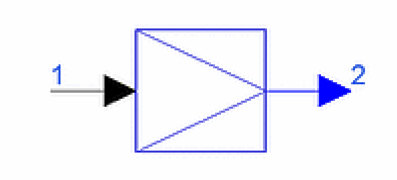
   $$y[n] = (\mathrm{fix})\{x(nT)\}$$

   where T is the input signal time step
2. If the fixed-point operations cannot fit into the precision specified, overflow occurs with the overflow characteristic specified by OverflowHandler. If ReportOverflow = REPORT, after the simulation has finished the number of overflow errors (if any) is

reported. RoundFix identifies whether fixed-point calculations are truncate or round method.
For details, refer to *Parameters for Fixed-Point Components* (ptolemy) in the *ADS Ptolemy Simulation* documentation.

3. This component uses twos-complement arithmetic; OutputPrecision values must specify at least 1 bit to the left of the decimal place (used as a sign bit).
4. This component has infinite input impedance.
5. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

TimedToFloat



Description Timed to Floating-Point
Library Signal Converters
Class TSDFTimedToFloat

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

Pin Outputs

Pin	Name	Description	Signal Type
2	output	output signal	real

Notes/Equations

1. TimedToFloat converts a timed input signal $x(t) = \{I(t), Q(t), F_c\}$ to a floating-point (real). The conversion rule is:
   for complex envelope:

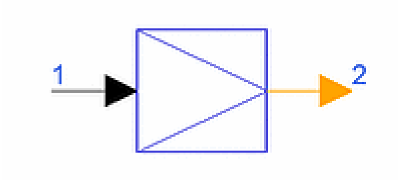
   $$y[n] = I(nT) \times \cos(2\pi F_c nT) - Q(nT) \times \sin(2\pi F_c nT)$$

   where T is the input signal time step
   for baseband:

   $$y[n] = x(nT)$$

   where T is the input signal time step
2. This component has infinite input impedance.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

TimedToInt



Description Timed to Integer
Library Signal Converters
Class TSDFTimedToInt

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

Pin Outputs

Pin	Name	Description	Signal Type
2	output	output signal	int

Notes/Equations

1. TimedToInt converts a timed input signal $x(t) = \{I(t), Q(t), F_c\}$ to an integer output $y[n]$. The conversion rule is:
   for complex envelope:

   $$y[n] = (int)\{I(nT) \times \cos(2\pi F_c nT) - Q(nT) \times \sin(2\pi F_c nT)\}$$

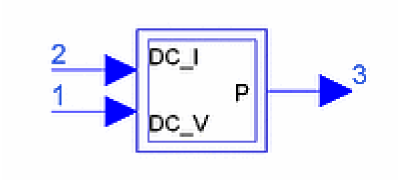
   where T is the input signal time step

   for baseband:

   $$y[n] = (int)\{x(nT)\}$$

   where T is the input signal time step
2. This component has infinite input impedance.
3. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).

VItoPower



Description Baseband voltage and current signal to power converter
Library Signal Converters
Class TSDFVItoPower

Parameters

Name	Description	Default	Type	Range
NumStart	sample number to start integration for power in watts	0	int	[0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	V	voltage value	real
2	I	current value	real

Pin Outputs

Pin	Name	Description	Signal Type
3	P	power value in watts	real

Notes/Equations

1. VItoPower converts input values representing voltage and current to average power in Watts.
2. Inputs V and I are multiplied to obtain the instantaneous power.
3. Output P is the integrated instantaneous power (with integration beginning at sample NumStart) divided by the number of samples recorded.
4. Use of this component is demonstrated in (access from the ADS Main window) *File* > *Open* > *Example* > PtolemyDocExamples > Timed_RF_Subsystems_wrk_ . Open the networks design *RF_PAE_example* and push into RF_PAE_TestFixture that uses VItoPower.
5. For general information regarding signal converter components, refer to *About Signal Converters* (sigconverters).