



Agilent Technologies

Signal Converters

August 2005

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms that apply to this software product is available upon request from your Agilent Technologies representative.

Restricted Rights Legend

Use, duplication or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DoD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

© Agilent Technologies, Inc. 1983-2005
395 Page Mill Road, Palo Alto, CA 94304 U.S.A.

Acknowledgments

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries.

Microsoft[®], Windows[®], MS Windows[®], Windows NT[®], and MS-DOS[®] 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.

Java[™] is a U.S. trademark of Sun Microsystems, Inc.

SystemC[®] is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission.

Contents

1 Signal Converters

Introduction.....	1-1
BitsToInt.....	1-2
BusToNum.....	1-4
CxToFix.....	1-6
CxToFix_M.....	1-8
CxToFloat.....	1-10
CxToFloat_M.....	1-12
CxToInt.....	1-14
CxToInt_M.....	1-15
CxToPolar.....	1-17
CxToRect.....	1-18
CxToTimed.....	1-19
CxToTimedIQ.....	1-21
FixToCx.....	1-22
FixToCx_M.....	1-23
FixToFloat.....	1-24
FixToFloat_M.....	1-25
FixToInt.....	1-26
FixToInt_M.....	1-27
FixToTimed.....	1-28
FloatIQToTimedIQ.....	1-30
FloatToCx.....	1-32
FloatToCx_M.....	1-33
FloatToFix.....	1-34
FloatToFix_M.....	1-36
FloatToInt.....	1-38
FloatToInt_M.....	1-39
FloatToTimed.....	1-40
IntToBits.....	1-41
IntToCx.....	1-42
IntToCx_M.....	1-43
IntToFix.....	1-44
IntToFix_M.....	1-46
IntToFloat.....	1-48
IntToFloat_M.....	1-49
IntToTimed.....	1-50
LogicToNRZ.....	1-51
NRZToLogic.....	1-52

NumToBus	1-53
PolarToCx	1-54
PolarToRect	1-55
RectToCx	1-56
RectToPolar	1-57
RFtoPower	1-58
TimedIQToCx	1-60
TimedIQToFloatIQ	1-61
TimedToCx	1-63
TimedToData	1-64
TimedToFix	1-66
TimedToFloat	1-68
TimedToInt	1-69
VltoPower	1-70

Index

Chapter 1: Signal Converters

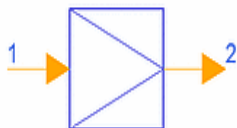
Introduction

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 automatically converted to the signal class specified as the input of the component. Auto conversion from a higher to a lower precision signal class may 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 32 bits with the minimum number of integer bits needed to represent the value. For example, the auto conversion of the floating-point (real) value of 1.0 creates a fixed-point value with precision of 2.30; a value of 0.5 would create one of precision of 1.31. For details on conversions between different classes of signals, refer to [“Conversion of Data Types”](#) in the *ADS Ptolemy Simulation* manual.

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”](#) in the *ADS Ptolemy Simulation* manual.

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

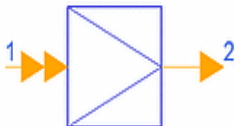
Pin	Name	Description	Signal Type
2	output		int

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 them 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 will be lost; if nBits is smaller than the wordsize minus one, the output integer will always be non-negative.

3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1.](#)

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

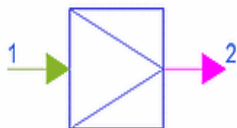
Notes/Equations

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 will be 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 two's

complement encoding, and sign extended. The resulting signed integer is sent to the output.

3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

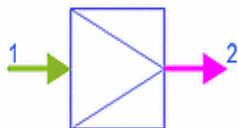
Pin Outputs

Pin	Name	Description	Signal Type
2	output	Output fix type	fix

Notes/Equations

1. CxToFix converts a complex input to a fixed-point output.
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) will be reported. RoundFix identifies whether fixed-point computations are truncate or round method. For details, refer to [“Parameters for Fixed-Point Components”](#) in the *ADS Ptolemy Simulation* manual.
3. This component uses two’s-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 the [“Introduction” on page 1-1](#).

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

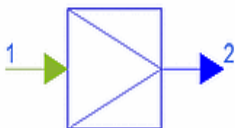
Notes/Equations

1. CxToFix_M converts an input complex matrix to a fixed-point matrix by constructing a new fixed-point matrix 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) will be reported. RoundFix identifies whether fixed-point computations are truncate or round method.

For details, refer to [“Parameters for Fixed-Point Components”](#) in the *ADS Ptolemy Simulation* manual.

3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

$$C = a + bj$$

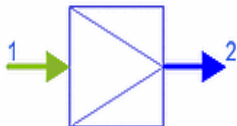
the `abs()` function is defined as

$$abs(C) = \sqrt{a^2 + b^2}$$

(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 the [“Introduction” on page 1-1](#).

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.

$$\text{FloatMatrix entry}(i) = \text{abs}(\text{ComplexMatrix entry}(i))$$

Where for a complex number

$$C = a + bj$$

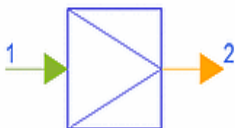
the abs() function is defined as

$$\text{abs}(C) = \sqrt{a^2 + b^2}$$

(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 [“Introduction” on page 1-1](#).

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

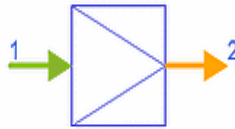
Notes/Equations

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) and then casting the result:

$$output = int(abs(input))$$

2. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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.

$$\text{IntegerMatrix entry}(i) = \text{int}(\text{abs}(\text{ComplexMatrix entry}(i)))$$

Where, for a complex number

$$C = a + bj$$

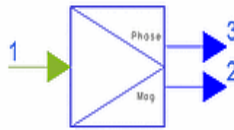
the abs() function is defined as

$$\text{abs}(C) = \sqrt{a^2 + b^2}$$

(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 [“Introduction” on page 1-1](#).

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

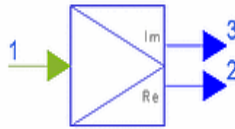
1. CxToPolar outputs the magnitude and angle of a complex number:

$$\text{Mag} = |\text{input}|$$

$$\text{Phase} = \angle \text{input} \text{ (the angle is in radians)}$$

2. For general information regarding signal converter components, refer to the ["Introduction" on page 1-1](#).

CxToRect



Description Complex to Real and Imaginary

Library Signal Converters

Class SDFCxToRect

C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input		complex

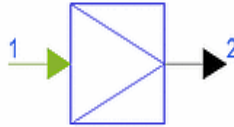
Pin Outputs

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	$[0, \infty)$
FCarrier	output carrier frequency	1.0e9	Hz	real	$(0, \infty)$

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	$[0, +\infty)$
FCarrier	output carrier frequency	$(0, +\infty)$

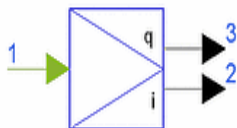
Notes/Equations

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. Before the ADS 2002C release, this component allowed $FCarrier$ to be 0 generating a complex envelope signal with characterization frequency equal to 0. Beginning with the ADS2002C release, $FCarrier$ now 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.
3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

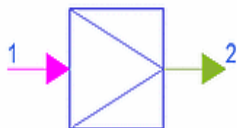
Pin Outputs

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. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

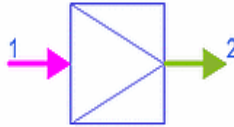
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

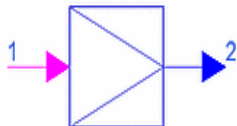
Pin Outputs

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 [“Introduction” on page 1-1](#).

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

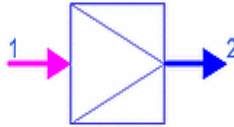
Pin Outputs

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 [“Introduction” on page 1-1](#).

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

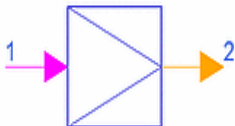
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

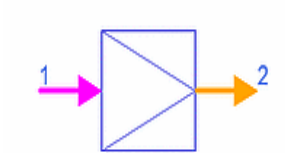
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

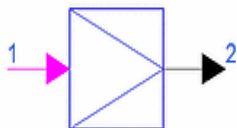
Pin Outputs

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” on page 1-1](#).

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

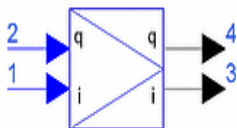
Name	Description	Value Range
TStep	output time step	[0, +∞)

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 the [“Introduction” on page 1-1.](#)

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 Inputs

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

Pin Outputs

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

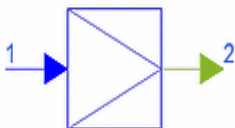
Notes/Equations

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. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

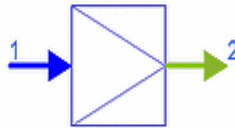
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

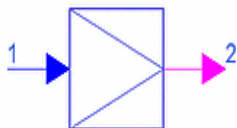
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

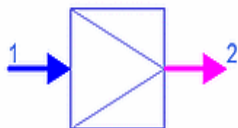
Notes/Equations

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) will be reported. RoundFix identifies whether fixed-point computations are truncate or round method.

For details, refer to [“Parameters for Fixed-Point Components”](#) in the *ADS Ptolemy Simulation* manual.

3. This component uses two’s-complement arithmetic; OutputPrecision parameter values must specify at least 1 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 the [“Introduction” on page 1-1](#).

FloatToFix_M



Description Floating-Point to Fixed-Point Matrix

Library Signal Converters

Class SDFFloatToFix_M

Derived From SDDFix

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

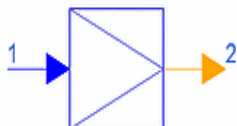
Notes/Equations

1. FloatToFix_M converts a floating-point (real) matrix to a fixed-point matrix by constructing a new fixed-point matrix 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) will be reported. RoundFix identifies whether fixed-point computations are truncate or round method.

For details, refer to [“Parameters for Fixed-Point Components”](#) in the *ADS Ptolemy Simulation* manual.

3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

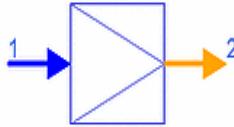
Pin Outputs

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.
2. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

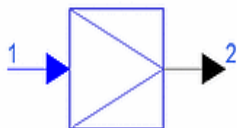
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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 timed signal $y[nT]$ where T is TStep. The output y will be a real baseband timed signal.

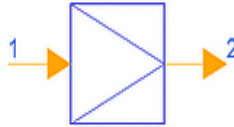
$$y[nT] = x[n \times T]$$

where

$$n = 1, 2, \dots$$

2. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

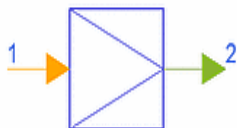
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

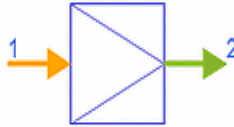
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

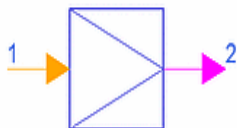
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

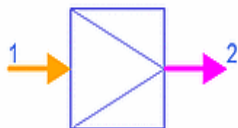
Notes/Equations

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) will be reported after simulation. RoundFix identifies whether fixed-point computations are truncate or round.

For details, refer to [“Parameters for Fixed-Point Components”](#) in the *ADS Ptolemy Simulation* manual.

3. This component uses two’s-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 the [“Introduction” on page 1-1](#).

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

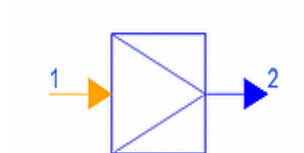
Notes/Equations

1. IntToFix_M converts an input integer matrix to a fixed-point matrix by constructing a new fixed-point matrix 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) will be reported after simulation. RoundFix identifies whether fixed-point computations are truncate or round.

For details, refer to [“Parameters for Fixed-Point Components”](#) in the *ADS Ptolemy Simulation* manual.

3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

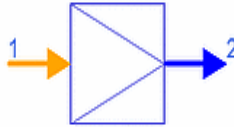
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

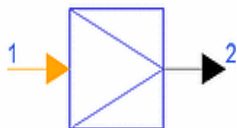
Pin Outputs

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 the double-length floating-point entries correspond to the integer entries of the input matrix.
2. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

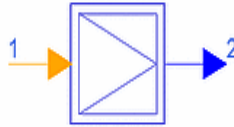
- IntToTimed converts an integer signal to a timed signal. Given the integer number $x[n]$ at input, the output is a timed signal $y[nT]$ where T is the given TStep. The output y is a real baseband timed signal.

$$y[nT] = x[n \times T]$$

$$n = 1, 2, \dots$$

- For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

LogicToNRZ



Description Logic to NRZ Format
Library Signal Converters

Parameters

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

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int

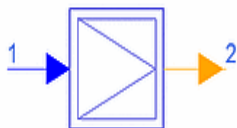
Pin Outputs

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 the [“Introduction” on page 1-1](#).

NRZToLogic



Description NRZ to Logic Format

Library Signal Converters

Parameters

Name	Description	Default	Type	Range
Amplitude	Amplitude	1.0	real	$(-\infty, \infty)$

Pin Inputs

Pin	Name	Description	Signal Type
1	input		real

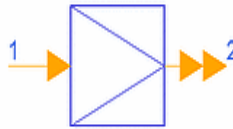
Pin Outputs

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 the [“Introduction” on page 1-1](#).

NumToBus



Description Number to Bus

Library Signal Converters

Class SDFNumToBus

C++ Code

Pin Inputs

Pin	Name	Description	Signal Type
1	input		int

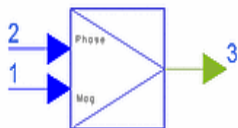
Pin Outputs

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.
2. For example, integer values from 0 to 7 are input. BusSplit4 is used to access the 4 output bits. Integer values 0 to 7 are then converted into the binary forms of 0000, 0001, 0010, 0011, 0100, 0101, 0110, and 0111.
3. Also see BusSplit components.
4. For general information regarding signal converter components, refer to the ["Introduction" on page 1-1](#).

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 the ["Introduction" on page 1-1](#).

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

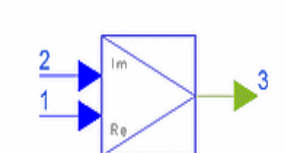
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

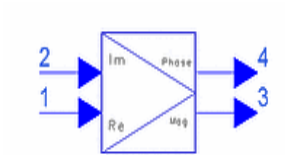
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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 the [“Introduction” on page 1-1](#).

RFtoPower



Description RF signal envelope to power converter

Library Signal Converters

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

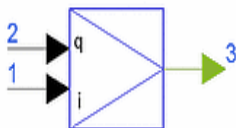
Pin	Name	Description	Signal Type
2	P	power value in watts	real

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 $2 * \text{RefR}$ 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 the *Example Project > PtolemyDocExamples > Timed_RF_Subsystems_prj*. Open the networks design *RF_PAE_example.dsn* and push into RF_PAE_TestFixture that uses RFtoPower.
5. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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

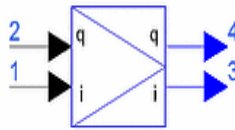
Pin Outputs

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 the [“Introduction” on page 1-1](#).

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

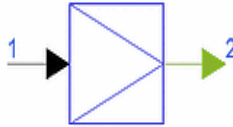
1. TimedIQToFloatIQ converts the two timed baseband input signals to two floating-point (real) signals. The timed baseband input signal at pin Iin is converted to a floating-point (real) signal at pin Iout and the timed baseband input signal at pin Qin is converted to a floating-point (real) signal at 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 the [“Introduction” on page 1-1.](#)

TimedToCx



Description Timed to Complex

Library Signal Converters

Class TSDFTimedToCx

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

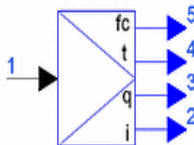
Pin Outputs

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, the output will be $(I + jQ)$. Likewise, when the timed signal is a real baseband signal, the output will be 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 the [“Introduction” on page 1-1](#).

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:

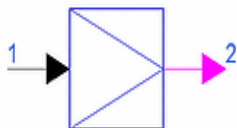
$$x(t) = I(t)$$

with $Q=0$ and $F_c=0$.

2. This component has infinite input impedance.

3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1.](#)

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:

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

for baseband:

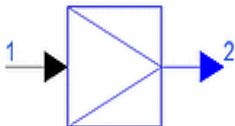
$$y[nt] = (\text{fix})\{x(t)\}.$$

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) will be reported. RoundFix identifies whether fixed-point computations are truncate or round method.

For details, refer to [“Parameters for Fixed-Point Components”](#) in the *ADS Ptolemy Simulation* manual.

3. This component uses two’s-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 the [“Introduction” on page 1-1](#).

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:

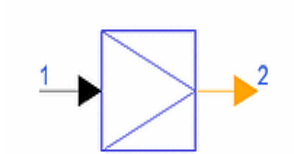
$$y[n] = I(t) \cos(2\pi F_c t) - Q(t) \sin(2\pi F_c t)$$

for baseband:

$$y[n] = x(t).$$

2. This component has infinite input impedance.
3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

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:

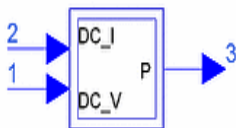
$$y[n] = (int)\{I(t)\cos(2\pi F_c t) - Q(t)\sin(2\pi F_c t)\}$$

for baseband:

$$y[n] = (int)x(t).$$

2. This component has infinite input impedance.
3. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

VltoPower



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

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. VltoPower 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 the *Example Project > PtolemyDocExamples > Timed_RF_Subsystems_prj*. Open the networks design *RF_PAE_example.dsn* and push into RF_PAE_TestFixture that uses VItoPower.
5. For general information regarding signal converter components, refer to the [“Introduction” on page 1-1](#).

Index

B

BitsToInt, 1-2
BusToNum, 1-4

C

CxToFix, 1-6
CxToFix_M, 1-8
CxToFloat, 1-10
CxToFloat_M, 1-12
CxToInt, 1-14
CxToInt_M, 1-15
CxToPolar, 1-17
CxToRect, 1-18
CxToTimed, 1-19
CxToTimedIQ, 1-21

F

FixToCx, 1-22
FixToCx_M, 1-23
FixToFloat, 1-24
FixToFloat_M, 1-25
FixToInt, 1-26
FixToInt_M, 1-27
FixToTimed, 1-28
FloatIQToTimedIQ, 1-30
FloatToCx, 1-32
FloatToCx_M, 1-33
FloatToFix, 1-34
FloatToFix_M, 1-36
FloatToInt, 1-38
FloatToInt_M, 1-39
FloatToTimed, 1-40

I

IntToBits, 1-41
IntToCx, 1-42
IntToCx_M, 1-43
IntToFix, 1-44
IntToFix_M, 1-46
IntToFloat, 1-48
IntToFloat_M, 1-49
IntToTimed, 1-50

L

LogicToNRZ, 1-51

N

NRZToLogic, 1-52
NumToBus, 1-53

P

PolarToCx, 1-54
PolarToRect, 1-55

R

RectToCx, 1-56
RectToPolar, 1-57
RFtoPower, 1-58

T

TimedIQToCx, 1-60
TimedIQToFloatIQ, 1-61
TimedToCx, 1-63
TimedToData, 1-64
TimedToFix, 1-66
TimedToFloat, 1-68
TimedToInt, 1-69

V

VltoPower, 1-70

