

Reference Guide

- [Libraries](#)
- [Functions](#)
- [Standalone Functions](#)
- [Standalone Objects](#)

Libraries

- [bootstrapLib](#) - *does the work needed to load and enable veeos*
- [codeLib](#) - *Programming aids*
- [dataLib](#) - *Create, modify, translate, analyze and otherwise manipulate data.*
- [eeLib](#) - *Functions related to Electrical Engineering.*
- [mathLib](#) - *General mathematics functions*
- [netLib](#) - *Functions that use the internet.*
- [sigLib](#) - *Signal processing components*
- [sysLib](#) - *Functions related to the operating system and the general computer environment.*
- [veeosLib](#) - *Functions that didn't fit elsewhere or that help manage the library.*

Functions

These standard functions correspond directly to a single object so that the identical functionality can be used in expressions or as objects.

- [addField](#) - *Adds to an existing record a field with specified name and value.*
- [arrayConcat](#) - *joins together arrays*
- [arrayRev](#) - *reverses the order of any 1D array*
- [buildField](#) - *Creates a field with specified name and value.*
- [colMax](#) - *Array statistics - determines the Maximum value of each column in an array*
- [colMean](#) - *Array statistics - determines the Mean value of each column in an array*
- [colMedian](#) - *Array statistics - determines the Median value of each column in an array*
- [colMin](#) - *Array statistics - determines the Minimum value of each column in an array*
- [colMode](#) - *Array statistics - determines the Mode of each column in an array*
- [colRMS](#) - *Array statistics - determines the RMS value of each column in an array*
- [colSDev](#) - *Array statistics - determines the Standard Deviation of each column in an array*
- [colVari](#) - *Array statistics - determines the Variance of each column in an array*
- [concat2D](#) - ~~joins together 2D arrays~~ - *deprecated in favor of arrayConcat*
- [concat3D](#) - ~~joins together 3D arrays~~ - *deprecated in favor of arrayConcat*
- [currentHumidity](#) - *returns the current humidity for a given city*
- [currentTemp](#) - *returns the current temperature for a given city*
- [dcBlock](#) - *- removes the DC component of a signal*
- [decayFilter](#) - *Implements a filter whose response is based on a given decay rate.*
- [dif](#) - *calculates difference between successive data points*
- [downloadLatestLibrary](#) - *Downloads the latest version the VEEOS library.*
- [dutyCycle](#) - *Determines the duty cycle of an arbitrary signal*
- [eval](#) - *Evaluates the given function at the given argument.*
- [fallTime](#) - *Determines the fall time of an arbitrary signal*
- [frequency](#) - *Determines the dominant frequency of an arbitrary signal.*
- [funcFilter](#) - *Implements an analog filter with transfer response specified by an arbitrary function.*

- [hilbert](#) - Computes the Hilbert transform of the input waveform.
- [ifThenEval](#) - Defines a function of multiple domains - is also a general Case statement.
- [importCLib](#) - Imports a Compiled Library
- [importULib](#) - Imports a User Library
- [isElement](#) - determines if a given element exists within a set
- [longLat](#) - returns the longitude and latitude for a given city
- [period](#) - Determines the period of an arbitrary signal.
- [polarFit](#) - Determines the unique circle defined by given three points.
- [polarRegress](#) - Determines the circle that best fits a set of points.
- [polarResample](#) - Resamples polar data to ensure full 0-360 coverage.
- [polyFilter](#) - Implements an analog filter with transfer response specified by numerator and denominator polynomials.
- [popupNotice](#) - Creates a pop-up message on the user's screen
- [ppCap](#) - Calculates Parallel Plate Capacitance
- [riseTime](#) - Determines the rise time of an arbitrary signal
- [rootFilter](#) - Implements an analog filter with transfer response specified by poles and zeroes.
- [setAND](#) - returns the set of elements that are in both of two sets
- [setComplement](#) - Returns the set of elements that are in one set but not in the other.
- [setOR](#) - returns the set of elements that are in either of two sets
- [setXOR](#) - returns the set of elements that are in either but not both of two sets
- [setVar](#) - Sets (or creates) a variable with given name and value
- [strReplace](#) - Replaces occurrences of a target string within the input string.
- [time](#) - Converts timestamp to time of the day.
- [timeStretch](#) - Stretches a series of data points by replicating them.
- [toCoord](#) - Creates Coordinates from input data.
- [toggleRadix](#) - Switches decimal radix between point and comma
- [toInt16](#) - Interprets input as hex value and outputs it as a 16-bit signed integer
- [toInt32](#) - Interprets input as hex value and outputs it as a 32-bit signed integer
- [toUInt8](#) - Interprets input as hex value and outputs it as an 8-bit unsigned integer
- [toBinStrN](#) - Formats a number as a binary string of arbitrary length.
- [unique](#) - returns unique subset
- [unzip](#) - Uses standard ZIP to unpack zip bundles.
- [uudecode](#) - Unpacks a coded ASCII file into the original format of the enclosed file.
- [uuencode](#) - Unpacks a coded ASCII file into the original format of the enclosed file.
- [wait](#) - Pauses execution for specified time in seconds
- [week](#) - Converts timestamp to week of the year.
- [winVersion](#) - Find the version of Windows which is currently running.
- [zip](#) - Uses standard ZIP to bundle whatever source files are specified.

Standalone Objects

These objects (or groups of objects) have no single underlying matching function.

- [analogAmp](#) - configurable analog amplifier
- [analogFilter](#) - configurable analog classical filter
- [bitStream](#) - display optimized to show bit pattern of digital stream
- [downConvert](#) - mixer-based down-conversion object
- [forEach](#) - Program construct for looping through a set of values.
- [negBridge](#) - negative polarity full-wave bridge rectifier
- [negRectifier](#) - negative polarity half-wave rectifier
- [numberStream](#) - display optimized to show streaming analog data
- [posBridge](#) - negative polarity half-wave rectifier

- [posRectifier](#) - *negative polarity full-wave bridge rectifier*
- [toBinStr8](#) - *Formats a number as a binary string of length 8.*
- [toBinStr16](#) - *Formats a number as a binary string of length 16.*
- [toBinStr32](#) - *Formats a number as a binary string of length 32.*
- [toBinStr64](#) - *Formats a number as a binary string of length 64.*
- [upConvert](#) - *mixer-based up-conversion object*
- [veeosInit](#) - *Initializes the loading of veeos*

Standalone Functions

These functions have no single overlying matching object but may of course still be used as desired.

- [aAmp](#) - *generic analog amplifier*
- [aFilter](#) - *generic classical analog filter*
- [aMixer](#) - *generic analog mixer*
- [aRectifier](#) - *generic analog rectifier*
- [butterPoly](#) - *generates Butterworth polynomials*
- [chebyPoly](#) - *generates Chebychev polynomials*
- [comma2point](#) - *Replaces commas with points.*
- [point2comma](#) - *Replaces points with commas.*
- [Xcoord](#) - *Returns the X value of a coordinate*
- [Ycoord](#) - *Returns the Y value of a coordinate*
- [Zcoord](#) - *Returns the Z value of a coordinate*

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 31 October 2015 20:34.

bootstrapLib

Does the work needed to load and enable veeos

Functions

- [init](#) - load and enable veeos

Extended Objects

Notes

BootstrapLib itself is just a library. All the work is actually done by a function inside it called [init\(\)](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:21. © 2015.

bootstrapLib: init()

Load and enable veeos.

Syntax

```
init()
    no inputs
    return: none
```

Usage

This function is designed to be invisible to the end user and to not be explicitly called. Instead it is run in the background by [veeosInit](#).

Location

library: [bootstrapLib](#)

Notes

Since init() is called by [veeosInit](#) which in turn is to be included in all VEE programs that use the veeos library, it is designed to be as small as possible and to have the absolute minimum hard-coded functionality. In essence it does nothing other than to find an RC file and then executes the directives included in that RC file.

In searching for an RC file, init() searches the below locations in series and concatenates all the directives found. This order is chosen for standard precedence: the most specific directive takes precedence if there is more than one conflicting definition.

1. Worldwide global - RC file is downloaded from veeos.org
2. VEE Install directory- searched for veeos.rc
3. Home directory - searched for veeos.rc
4. Local directory - searched for veeos.rc

Note that since the RC files are concatenated and then the directives in them are executed in order, the last RC file takes precedence. This conforms with standard precedence that says most-specific should win.

Note also that these above RC files can be totally bypassed by setting an environment variable MY_VEEOSRC specifying the RC file(s) to be loaded. As with the default precedence, last files loaded take precedence.

NOTE that due to this behavior, init() can run any RC file directives that the user desires. The end result is that although init() is part of veeos it can alternately be used for initiating the end user's customizations instead (or in addition). This enable the end user program to change functionality (i.e. overriding) depending upon the environment in which it is run.

Reference

The default veeos RC file is self documenting but the documentation is also included here for reference. Note that the RC file directives include loading libraries and setting variables. The also can be used to automatically execute functions.

```
### User Function Libraries to load. Space delimited. Field order dependent
```

```
# ulib <name> <file> [<function to execute>]

# examples
# ulib dt VEEROOT/veeos/lib/dtLib.vxe
# ulib dt "C:\program files\Agilent\Vee Pro 7.0\veeos\lib\dtLib.vxe" init()

### Compiled Function Libraries to load. Space delimited. Field order dependent.
# clib <name> <file> <definition file> [<function to execute>]

# example
# clib io VEEROOT/veeos/lib/ioLib.dll VEEROOT/veeos/lib/ioLib.h

### set up variables
# gvar <name> <type> <value>

# example
# gvar eps0 Real64 8.854187817e-12
```

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[veeosInit](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:21. © 2015.

codeLib

Programming Aids

Functions

- [importCLib](#) - Imports a Compiled Library
- [importULib](#) - Imports a User Library
- [popupNotice](#) - Creates a pop-up message on the user's screen
- [wait](#) - Pauses execution for specified time in seconds

Internal Content

These are used within the library but can often be useful directly accessed from without.

- private functions
 - `version` - returns revision date of this library
- variables
 - `none`

Standalone Objects

- [forEach](#) - Program construct for looping through a set of values.
- [bitStream](#) - display optimized to show bit pattern of digital stream
- [numberStream](#) - display optimized to show streaming analog data

Notes

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:22. © 2015.

importCLib(libName,DLL, def)

Imports a Compiled Library



Syntax

importCLib(libName, DLL, def)

libName: scalar Text - library name to use for the imported library

DLL: scalar text - the DLL file that is to be loaded

def: array 1D Text - the definition file contents

return: type description

Usage

This wraps the Import Library object within a function so that it can be used in expressions.

Location

menu: veeos ==> coding ==> importCLib

library: [codeLib](#)

Example

importCLib("games", "C:\temp\gameLib.dll", ["long __stdcall chess(int difficulty)","long __stdcall go(int difficulty)"]) imports the compiled library gameLib.dll with the library name "games".

Notes

As with any name, do not use spaces or punctuation within the library name. If you need spaces in the library file path be sure to quote them. Unless your library is within the default search path, you will need absolute path for the library file. Note that a definition file is not needed since the function definitions are included as an argument.

Reference

Supported On

VEE 7.0+, Windows XP+

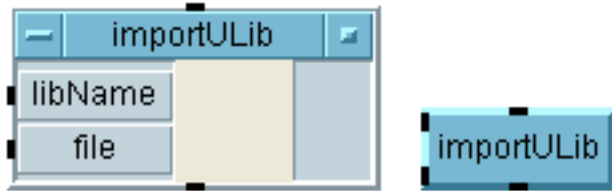
See Also

[importULib](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:25. © 2015.

importULib(libName,file)

Imports a User Library



Syntax

```
importULib(libName, file)
    libName: scalar Text - library name to use for the imported library
    file: scalar Text - the VEE library file
    return: none
```

Usage

This wraps the Import Library object within a function so that it can be used in expressions.

Location

menu: veeos ==> coding ==> importULib

library: [codeLib](#)

Example

`importULib("games", "C:\temp\gameLib.vee")` imports the user library `gameLib.vee` with the library name "games".

Notes

As with any name, do not use spaces or punctuation within the library name. If you need spaces in the library file path be sure to quote them. Unless your library is within the default search path, you will need absolute path for the library file.

Reference

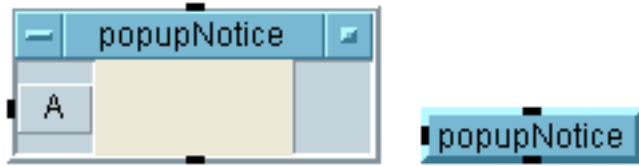
Supported On

VEE 7.0+, All platforms.

See Also

popupNotice(a)

Creates a pop-up message on the user's screen



Syntax

popupNotice(a)

a: scalar Text - message to be displayed

return: none

Usage

While many other mechanisms exist to place messages on the screen, this simplistic one allows asynchronous one-liners to pop up via a function call.

Location

menu: veeos ==> Coding ==> popupNotice

library: [codeLib](#)

Example

popupNotice("present status is RED") pops up a message on the user's screen

Notes

The popup used is asynchronous and will stay in place as long as the program is running. If you desire to eliminate it at any time, send the message "HIDENOTICE".

Reference

Dependencies

Supported On

Vee 7.0+, all platforms

See Also

wait(a)

Pauses execution for specified time in seconds



Syntax

- wait
(a)
a: scalar Real - number of seconds to sleep
return: none

Usage

Use this to delay execution of the next expression within a formula

Location

menu: veeos ==> Coding ==> wait

library: [codeLib](#)

Example

```
// now wait for 5 seconds before continuing
wait(5);
```

Notes

This is seemingly identical to the Delay object, but is a function so that it can be used in expressions. This works via a direct call to the Windows Kernel. As discussed on VRF ca. 2004 (Georg Nied et al), it can be useful for resolving CPU loading issues.

Reference

Dependencies

- [importCLib](#) used to import kernel's sleep function

Supported On

wait

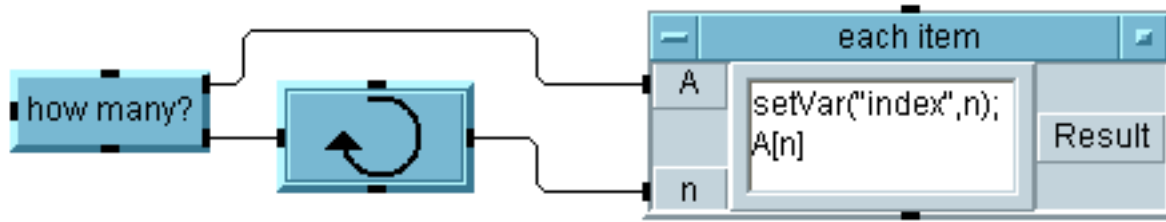
VEE 7.0+, Windows XP+

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:34. © 2015.

forEach

Program construct for looping through a set of values.



Syntax

forEach is a set of objects only - no underlying function

input: array of any type

output: one element of the array at a time

Usage

Drop this set of objects down into your program wherever you need to serially process each member of an array.

Location

menu: veeos ==> Coding ==> forEach

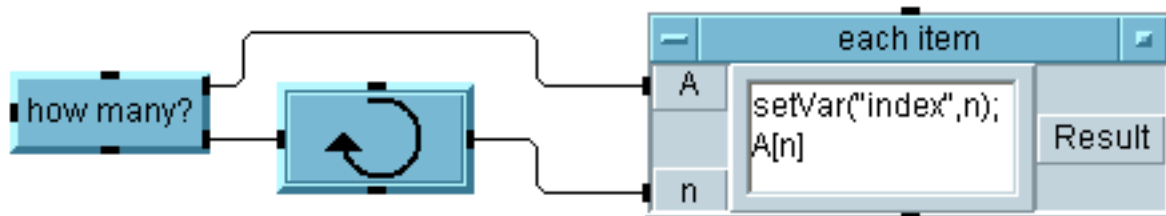
library: *none- objects only*

Example

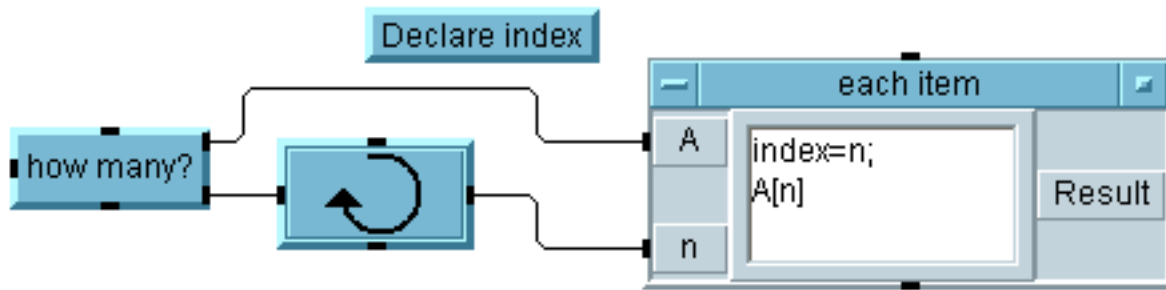
Notes

While this is functionally the same as the forEach native object introduced with VEE8, this rendition will work equally in any version of VEE and it provides an advantage in that the index is available for use.

In some cases it may well be desirable to create an index variable for use in other sections of the program. This allows you to know where in a given list you are currently indexed. One way to do this is with a [setVar\(\)](#), as in



or, if you would like to reduce the scope of the index variable, you can use a Declare Variable first, as in



Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 27 October 2015 11:53. © 2015.

bitstream

Display optimized to show bit pattern of digital stream



Syntax

bitStream is a set of objects only - no underlying function

input: array of any type

output: display of the streaming bit pattern

Usage

Drop this set of objects down into your program wherever you need to display the real-time signal of a streaming pattern of bits.

Location

menu: veeos ==> Coding ==> bitStream

library: *none- objects only*

Example

Notes

The bitStream display was designed to display one bit from a stream of values. A typical use for this would be when streaming data through a serial IO device. By using a suitable number of bitstream displays one can simulate a simplistic bus analyzer. This example generates a sample stream of values and displays the corresponding bit patterns. For convenience the values are also shown.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

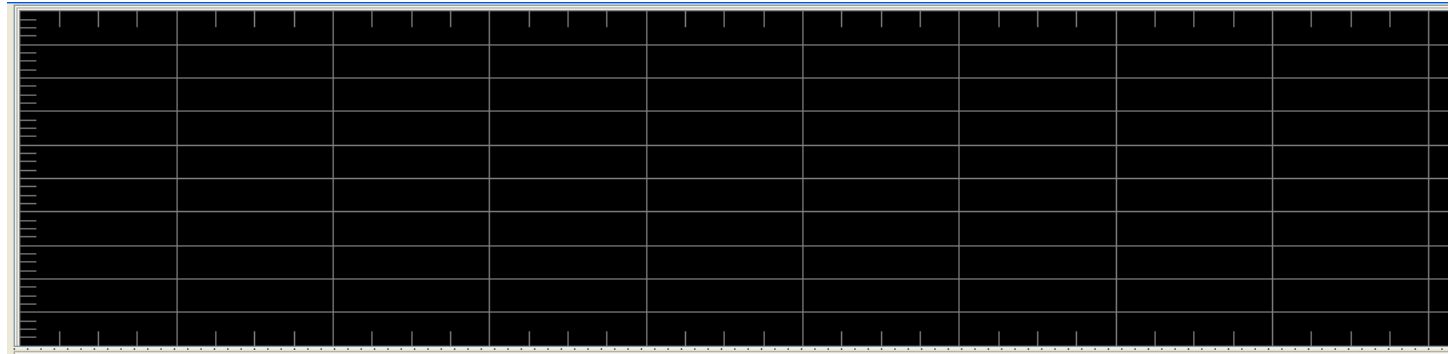
bitStream

[numberStream](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:20. © 2015.

numberStream

Display optimized to show streaming analog data



Syntax

numberStream is a set of objects only - no underlying function

input: array of numbers

output: display of the streaming analog value

Usage

Drop this set of objects down into your program wherever you need to display the real-time signal of a streaming set of values

Location

menu: veeos ==> Coding ==> numberStream

library: *none- objects only*

Example

Notes

The numberStream display was designed to graphically display a stream of values. A typical use for this would be when streaming data through a serial IO device and wanting to visualize the data. This example generates a sample stream of values and displays them using numberStream. For convenience the values are also shown textually. NOTE: Y scale is set at a default. reset as desired.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[bitStream](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:27. © 2015.

dataLib

Create, modify, translate, analyze and otherwise manipulate data.

Functions

- [addField](#) - Adds to an existing record a field with specified name and value.
- [buildField](#) - Creates a field with specified name and value.
- [comma2point](#) - Replaces commas with points.
- [hexToInt](#) - Interprets input as hex value and outputs it as an integer
- [point2comma](#) - Replaces points with commas.
- [polarFit](#) - Determines the unique circle defined by given three points.
- [polarRegress](#) - Determines the circle that best fits a set of points.
- [polarResample](#) - Resamples polar data to ensure full 0-360 coverage.
- [setVar](#) - Sets (or creates) a variable with given name and value
- [timeStretch](#) - Stretches a series of data points by replicating them.
- [toggleRadix](#) - Switches decimal radix between point and comma
- [toBinStrN](#) - Formats a number as a binary string of arbitrary length.
- [toCoord](#) - Creates Coordinates from input data.
- [toInt16](#) - Interprets input as hex value and outputs it as a 16-bit signed integer
- [toInt32](#) - Interprets input as hex value and outputs it as a 32-bit signed integer
- [toUInt8](#) - Interprets input as hex value and outputs it as an 8-bit unsigned integer

Internal Content

These are used within the library but can often be useful directly accessed from without.

- private functions
 - `version` - returns revision date of this library
- variables
 - `none`

Standalone Objects

- [toBinStr8](#) - Formats a number as a binary string of length 8.
- [toBinStr16](#) - Formats a number as a binary string of length 16.
- [toBinStr32](#) - Formats a number as a binary string of length 32.
- [toBinStr64](#) - Formats a number as a binary string of length 64.

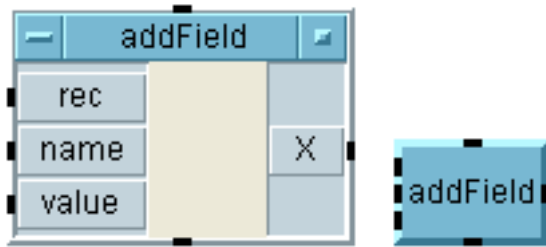
Standalone Functions

- [Xcoord](#) - Returns the X value of a coordinate
- [Ycoord](#) - Returns the Y value of a coordinate
- [Zcoord](#) - Returns the Z value of a coordinate

Notes

addField(rec,name,value)

Adds to an existing record a field with specified name and value.



Syntax

addField(rec,name,value)

rec: Record - an existing record

name: scalar Text - the name of a field that is to be added to the existing record

value: scalar Any - the value of a field that is to be added to the existing record

return: record - existing record with one additional field added

Usage

Creates a new field with specified name and value and adds it to an existing record.

Location

menu: veeos ==> Data ==> addField

library: [dataLib](#)

Example

addField (A, "new", 38.7) takes the record A and adds to it a field with name "new" and value of 38.7).

Notes

As with any record, one cannot add a field whose name matches an existing field.

Reference

Supported On

VEE 7.0+, All platforms.

See Also

[buildField](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:18. © 2015.

buildField(name,value)

Creates a field with specified name and value.



Syntax

buildField(name,value)

name: scalar Text - the name of a field that is to be created

value: scalar Any - the value of a field that is to be created

return: record - record with one field as specified

Usage

Creates a record with a single field with specified name and value.

Location

menu: veeos ==> Data ==> buildField

library: [dataLib](#)

Example

buildField ("new", 38.7) creates a record that has a single field with name "new" and value of 38.7).

Notes

As with any field, there must be no spaces or other punctuation in the name. Value though can be anything.

Reference

Supported On

VEE 7.0+, All platforms.

See Also

[addField](#)

comma2point(a)

Replaces commas with points.

Syntax

comma2point(a)

a: scalar Text - numbers as input string

return: Text - string with "," replaced by "."

Usage

Designed to be used when presenting results of numerical computations in text format.

Location

menu: *not included in menu*

library: [dataLib](#)

Example

comma2point("23,45 24,34 89,01") returns "23.45 24.34 89.01"

Notes

The world is split between two major groups that use a decimal radix of "," or ".". This function is provided as a quick way to switch between the two when presenting numbers in Text format. Note that strReplace() would also work but it uses loops so this one is faster.

Reference

Dependencies

Supported On

VEE7.0+, all platforms

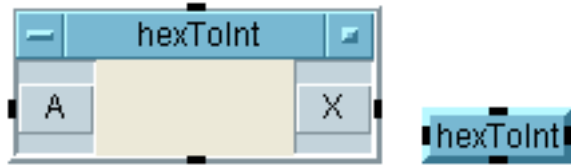
See Also

[point2Comma](#), [strReplace](#), [toggleRadix](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:22. © 2015.

hexToInt(a)

Interprets input as hex value and outputs it as an integer



Syntax

hexToInt(a)

a: Text - value that is assume to be hex

return: Int - input value converted to decimal integer

Usage

Convert values from hex to decimal integer

Location

menu: veeos ==> Data ==> Conversion ==> hexToInt

library: [dataLib](#)

Example

hexToInt("ee3425a") returns 249774682

Notes

Input can be a scalar or an array. Also can be text or numerical. This can be done with a From String, but since it is provided as a function it can be used in more places.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[toBinStr8](#), [toBinStr16](#), [toBinStr32](#), [toBinStr64](#), [toBinStrN](#), [toUInt8](#), [toInt16](#), [toInt32](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:25. © 2015.

point2comma(a)

Replaces points with commas.

Syntax

point2comma(a)

a: scalar Text - numbers as input string

return: Text - string with "." replaced by ","

Usage

Designed to be used when presenting results of numerical computations in text format.

Location

menu: *not included in menu*

library: [dataLib](#)

Example

comma2point("23.45 24.34 89.01") returns "23,45 24,34 89,01"

Notes

The world is split between two major groups that use a decimal radix of "," or ".". This function is provided as a quick way to switch between the two when presenting numbers in Text format. Note that strReplace() would also work but it uses loops so this one is faster.

Reference

Dependencies

Supported On

VEE7.0+, all platforms

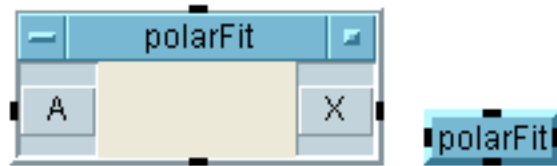
See Also

[comma2point](#), [strReplace](#), [toggleRadix](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](#). Last updated 20 October 2015 20:27. © 2015.

polarFit(a)

Determines the unique circle defined by given three points.



Syntax

polarFit(a)

a: array 1D coord - arbitrary three points in cartesian coords

return: scalar record - center and radius of circle

Usage

Given three points it is possible to define a unique circle. This function finds that circle and returns center and radius.

Location

menu: veeos ==> Data ==> Circles ==> polarFit

library: [dataLib](#)

Example

see veeos examples inside of VEE

Notes

Since 3 points on a plane uniquely define a circle (unless they are colinear of course), this function can operate with a closed-form expression and hence no loops. It is fast and precise normally, but if the points are nearly colinear then the result can be noisy.

Reference

Dependencies

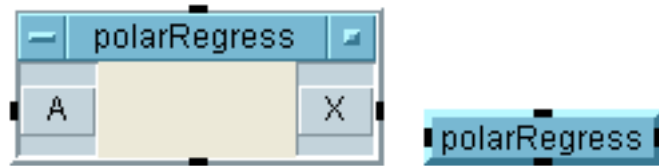
Supported On

VEE 7.0+, all platforms

See Also

polarRegress(a)

Determines the circle that best fits a set of points.



Syntax

polarFit(a)

a: array 1D coord - arbitrary points in cartesian coords
 return: scalar record with three fields
 center: best fit center as coord
 radius: best fit radius as real
 error: estimated error

Usage

This is designed designed to use in determining the best-fit center of a series of measured points.

Location

menu: veeos ==> Data ==> Circles ==> polarRegress

library: [dataLib](#)

Example

see veeos examples inside of VEE

Notes

3 points on a plane uniquely define a circle (unless they are colinear of course), but if one has more than three points that don't exactly lie on a unique circle then determining a best-fit circle has no closed-form solution. Suppose, for instance, that one want to determine a circle with multiple measurements. The coordinates of a series of points near the presumed circumference are made, and then this function can be used to determine the best-fit circle. The algorithm used here is very basic so should be quite robust. Rather than using any sort of optimization, determines analytically the circles uniquely defined (see [polarFit](#)) by sets of three points from within the measurements, and then averages the resulting circle centers and radii. Note that best results are thereby obtained if the measured points are widely spaced.

If there are only three points given then this reduces to [polarFit](#) since there is then a unique answer.

Note that there are other algorithms for finding a best-fit circle which may do better with noisy data, but a long as noise level si low this works quickly and well.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

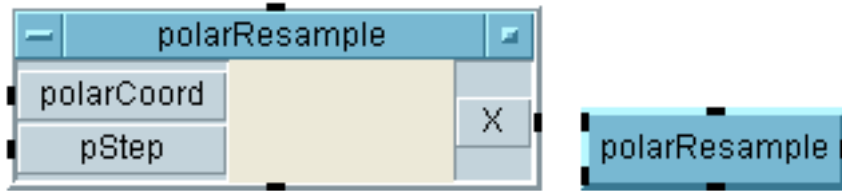
See Also

[polarFit](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 25 October 2015 11:22. © 2015.

polarResample(a)

Resamples polar data to ensure full 0-360 coverage.



Syntax

```
function(polarCoord,pStep)
    polarCoord: Array 1D Coord or PComplex - input data
                for Coords, phase is "X", mag is "Y".
    pStep: scalar real - desired step size of output data
    return: Array 1D Coord or PComplex - resampled data from 0 to 360-pStep degrees
```

Usage

Use this to fill in missing polar data by use of interpolation

Location

menu: veeos ==> Data ==> Circles ==> polarResample

library: [dataLib](#)

Example

see examples from veeos menu

Notes

This was written for the case of circular measurements wherein data is taken at multiple angles, but some angles are missing from the data. A typical case would be antenna measurements as in gain vs angle using a statistical measurement (i.e. random angles chosen). The data needs to be resampled for subsequent processing by a tool that requires constant angular steps. This function takes in data in either coordinates or as PComplex values, resamples, and then returns data in the same type as was input. No loops are used so this should be fairly fast. And since linear interpolation is used it should also be quite robust.

Importantly, the phase input needs to be monotonically increasing. Note that pStep is adjusted automatically such that an integer number of steps are created for 360 degrees.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[polarRegress](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 26 October 2015 12:08. © 2015.

setVar(name,value)

Sets (or creates) a variable with given name and value



Syntax

setVar(name,value)

name: scalar Text - name of variable to be set (or created)

value: any any - value of the variable to be set (or created)

return: *none*

Usage

Use this to set an existing variable or create a new one automatically.

Location

menu: veeos ==> Data ==> setVar

library: [dataLib](#)

Example

setVar("myvar", 567) sets the variable myvar to the value 567.

Notes

One can of course create a new variable using a Set Variable object, and one can set a value using an assignment operator. This function essentially combines both features by setting the named variable if it already exists but also automatically creating a new variable if needed.

Reference

Dependencies

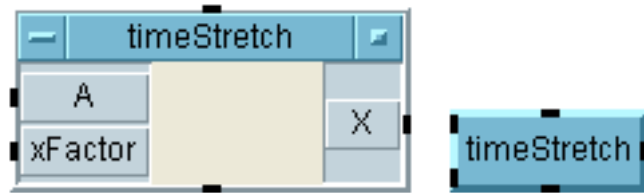
Supported On

VEE7.0+, all platforms

See Also

timeStretch(a)

Stretches a series of data points by replicating them.



Syntax

timeStretch(a)

a: Array1d number - an input array of numerical values

xFactor: scalar Int - number of times to repeat each input value

return: array 1D - input values repeated xFactor times

Usage

Designed for use in graphically displaying values. Since each value is repeated multiple times, graphs are simple made with levels shown rather than just data points.

Location

menu: veeos ==> Data ==> timeStretch

library: [dataLib](#)

Example

timeStretch([2 6 3 6 7],3) returns [2 2 2 6 6 6 3 3 3 6 6 6 7 7 7]

Notes

Where this is intended and is quite useful is in visualizing real-world signals that are controlled by discrete values. Classic case would be the time output of a DAC.

Reference

Dependencies

Supported On

VEE7.0+, all platforms

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:30. © 2015.

toggleRadix(a)

Changes points to commas- and vice versa.



Syntax

toggleRadix(a)

a: scalar Text - numbers as input string

return: Text - string with "," replaced by "." or vice versa depending upon inout

Usage

Designed to be used when presenting results of numerical computations in text format.

Location

menu: veeos ==> data ==> Formatting ==> toggleRadix

library: [dataLib](#)

Example

toggleRadix("23,45 24,34 89,01") returns "23.45 24.34 89.01" and

toggleRadix("23.45 24.34 89.01") returns "23,45 24,34 89,01"

Notes

The world is split between two major groups that use a decimal radix of "," or ".". This function is provided as a quick way to switch between the two when presenting numbers in Text format. Note that strReplace() would also work but it uses loops so this one is faster.

Reference

Dependencies

uses comma2point() and point2comma()

Supported On

VEE7.0+, all platforms

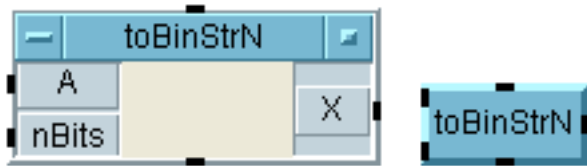
See Also

[comma2point](#), [point2Comma](#), [strReplace](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:32. © 2015.

toBinStrN(a,nBits)

Formats a number as a binary string of arbitrary length.



Syntax

toBinStrN(a,nBits)

a: scalar Int - value to be converted to binary string

b: scalar Int - number of digits to present

return: Text - input represented by a string of 1's and 0's

Usage

Use this to show in human readable format the binary representation of an integer.

Location

menu: veeos ==> Data ==> Conversion ==> toBinStrN

library: [dataLib](#)

Example

toBinStrN(38,8) returns "00100110"

toBinStrN(345,12) returns "000101011001"

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[toBinStr8](#), [toBinStr16](#), [toBinStr32](#), [toBinStr64](#)

toCoord(a)

Creates Coordinates from input data.



Syntax

toCoord(a)

a: scalar or Array 1D Complex
 or scalar or Array 1D PComplex
 or array 1D or array 2D numerical
 return: scalar or array 1D Coordinate

Usage

Use this to create coordinates within an expression.

Location

menu: veeos ==> Data ==> Conversion ==> toCoord

library: [dataLib](#)

Example

toCoord((1,2)) creates a scalar coordinate with X=1 and Y=2
 toCoord((1,@90)) creates a scalar coordinate with X=0 and Y=1
 toCoord(<array 1D of complex>) creates a array of coordinates
 toCoord([1,2,3]) creates a scalar 3D coordinate (1,2,3)
 toCoord([[1,2,3],[4,5,6]]) creates an array 1D of 2D coordinates

Notes

While VEE syntax provides the ability to create and examine Complex and PComplex data within expressions, it lack the ability to do the same with coordinates. Hence one is left with the often-clumsy need to use the Build and Unbuild Coord objects instead. This function allows the user to convert Complex and PComplex data to 2D coordinates and also allows one to build 2D or 3D coordinates from scratch using an array as input.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

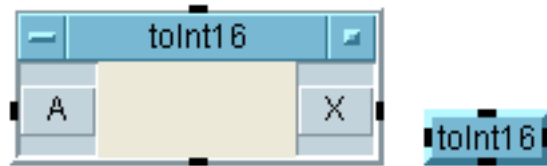
See Also

[Xcoord](#), [Ycoord](#), [Zcoord](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 26 October 2015 08:20. © 2015.

toInt16(a)

Interprets input as hex value and outputs it as a 16-bit signed integer



Syntax

toInt16(a)

a: number or text- value to be converted

return: Int16 - input value converted to signed 16-bit integer

Usage

Convert strings or numbers to signed 16-bit integers. Text inputs assumed to be in hex.

Location

menu: veeos ==> Data ==> Conversion ==> toInt16

library: [dataLib](#)

Example

toInt16("a34") returns 2612
 toInt16("4708") returns 18184
 toInt16(4708) returns 4708

Notes

Input can be a scalar or an array. Also can be text or numerical. This varies from the built-in function asInt16() in that it assumes that text inputs are hex.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

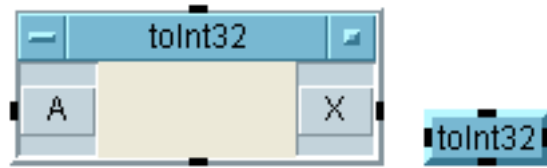
See Also

[toBinStr8](#), [toBinStr16](#), [toBinStr32](#), [toBinStr64](#), [toBinStrN](#), [hexToInt](#), [toUInt8](#), [toInt32](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:32. © 2015.

toInt32(a)

Interprets input as hex value and outputs it as a 32-bit signed integer



Syntax

toInt32(a)

a: number or text- value to be converted

return: Int32 - input value converted to signed 32-bit integer

Usage

Convert strings or numbers to signed 32-bit integers. Text inputs assumed to be in hex.

Location

menu: veeos ==> Data ==> Conversion ==> toInt32

library: [dataLib](#)

Example

toInt32("a34a") returns 41802

toInt32("478823") returns 4687907

toInt32(478823) returns 478823

Notes

Input can be a scalar or an array. Also can be text or numerical. This varies from the built-in function asInt32() in that it assumes that text inputs are hex.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

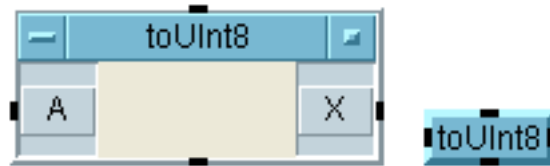
See Also

[toBinStr8](#), [toBinStr16](#), [toBinStr32](#), [toBinStr64](#), [toBinStrN](#), [hexToInt](#), [toUInt8](#), [toInt16](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:32. © 2015.

toUInt8(a)

Interprets input as hex value and outputs it as an 8-bit unsigned integer



Syntax

toUInt8(a)

a: number or text- value to be converted

return: UInt8 - input value converted to unsigned 8-bit integer

Usage

Convert strings or numbers to unsigned 8-bit integers. Text inputs assumed to be in hex.

Location

menu: veeos ==> Data ==> Conversion ==> toUInt8

library: [dataLib](#)

Example

toUInt8("a") returns 10

toUInt8("47") returns 71

toUInt8(47) returns 47

Notes

Input can be a scalar or an array. Also can be text or numerical. This varies from the built-in function asUInt8() in that it assumes that text inputs are hex.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[toBinStr8](#), [toBinStr16](#), [toBinStr32](#), [toBinStr64](#), [toBinStrN](#), [hexToInt](#), [toInt16](#), [toInt32](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:33. © 2015.

toBinStr8(a)

Formats a number as a binary string of length 8.



Syntax

- *no underlying function - use toBinStrN*
a: scalar Int - value to be converted to binary string
return: Text - input represented by a string of 8 1's and 0's

Usage

Use this to show in human readable format the binary representation of an integer.

Location

menu: veeos ==> Data ==> Conversion ==> toBinStr8

library: [dataLib](#)

Example

toBinStr8(38) returns "00100110"

Notes

This object is provided as a convenience only. It is a special case of toBinStrN()

Reference

Dependencies

uses toBinStrN()

Supported On

VEE 7.0+, all platforms

See Also

[toBinStr16](#), [toBinStr32](#), [toBinStr64](#), [toBinStrN](#), [hexToInt](#), [toUInt8](#), [toInt16](#), [toInt32](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:31. © 2015.

toBinStr16(a)

Formats a number as a binary string of length 16.



Syntax

- *no underlying function - use toBinStrN*
 - a: scalar Int - value to be converted to binary string
 - return: Text - input represented by a string of 16 1's and 0's

Usage

Use this to show in human readable format the binary representation of an integer.

Location

menu: veeos ==> Data ==> Conversion ==> toBinStr16

library: [dataLib](#)

Example

toBinStr16(85267) returns "0100110100010011"

Notes

This object is provided as a convenience only. It is a special case of toBinStrN()

Reference

Dependencies

uses toBinStrN()

Supported On

VEE 7.0+, all platforms

See Also

[toBinStr8](#), [toBinStr32](#), [toBinStr64](#), [toBinStrN](#), [hexToInt](#), [toUInt8](#), [toInt16](#), [toInt32](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:31. © 2015.

toBinStr32(a)

Formats a number as a binary string of length 32.



Syntax

- no underlying function - use toBinStrN*
 - a: scalar Int - value to be converted to binary string
 - return: Text - input represented by a string of 32 1's and 0's

Usage

Use this to show in human readable format the binary representation of an integer.

Location

menu: veeos ==> Data ==> Conversion ==> toBinStr32

library: [dataLib](#)

Example

toBinStr32(388775) returns "000000000000000001001011101110111"

Notes

This object is provided as a convenience only. It is a special case of toBinStrN()

Reference

Dependencies

uses toBinStrN()

Supported On

VEE 7.0+, all platforms

See Also

[toBinStr8](#), [toBinStr16](#), [toBinStr64](#), [toBinStrN](#), [hexToInt](#), [toUInt8](#), [toInt16](#), [toInt32](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:31. © 2015.

toBinStr64(a)

Formats a number as a binary string of length 64.



Syntax

- *no underlying function - use toBinStrN*
 - a: scalar Int - value to be converted to binary string
 - return: Text - input represented by a string of 64 1's and 0's

Usage

Use this to show in human readable format the binary representation of an integer.

Location

menu: veeos ==> Data ==> Conversion ==> toBinStr64

library: [dataLib](#)

Example

toBinStr64(34516786785267) returns "0000000000000000000011111011001001001000011010000101010111110011"

Notes

This object is provided as a convenience only. It is a special case of toBinStrN()

Reference

Dependencies

uses toBinStrN()

Supported On

VEE 7.0+, all platforms

See Also

[toBinStr8](#), [toBinStr16](#), [toBinStr32](#), [toBinStrN](#), [hexToInt](#), [toUInt8](#), [toInt16](#), [toInt32](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:31. © 2015.

Xcoord(a)

Returns the X value of a coordinate.

Function only- no corresponding object

Syntax

Xcoord(a)

a: scalar or Array1D Coord - coordinate to be examined

return: scalar or Array 1D - the X value of the input coordinate

Usage

Use this to access the X value from coordinate within an expression.

Location

menu: *no menu pick - to access use the Function and Object browser*

library: [dataLib](#)

Example

Given coordinate (1,2), Xcoord returns 1

Notes

While VEE syntax provides the ability to create and examine Complex and PComplex data within expressions, it lack the ability to do the same with coordinates. Hence one is left with the often-clumsy need to use the Build and Unbuild Coord objects instead. This function allows one to unbuild a coordinate within an expression.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[toCoord](#), [Ycoord](#), [Zcoord](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 26 October 2015 08:20. © 2015.

Ycoord(a)

Returns the Y value of a coordinate.

Function only- no corresponding object

Syntax

Xcoord(a)

a: scalar or Array1D Coord - coordinate to be examined

return: scalar or Array 1D - the Y value of the input coordinate

Usage

Use this to access the Y value from coordinate within an expression.

Location

menu: *no menu pick - to access use the Function and Object browser*

library: [dataLib](#)

Example

Given coordinate (1,2), Ycoord returns 2

Notes

While VEE syntax provides the ability to create and examine Complex and PComplex data within expressions, it lack the ability to do the same with coordinates. Hence one is left with the often-clumsy need to use the Build and Unbuild Coord objects instead. This function allows one to unbuild a coordinate within an expression.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[toCoord](#), [Xcoord](#), [Zcoord](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 26 October 2015 08:20. © 2015.

Zcoord(a)

Returns the Z value of a coordinate.

Function only- no corresponding object

Syntax

Zcoord(a)

a: scalar or Array1D Coord - coordinate to be examined

return: scalar or Array 1D - the Z value of the input coordinate

Usage

Use this to access the Z value from coordinate within an expression.

Location

menu: *no menu pick - to access use the Function and Object browser*

library: [dataLib](#)

Example

Given coordinate (1,2,3), Zcoord returns 3

Notes

While VEE syntax provides the ability to create and examine Complex and PComplex data within expressions, it lack the ability to do the same with coordinates. Hence one is left with the often-clumsy need to use the Build and Unbuild Coord objects instead. This function allows one to unbuild a coordinate within an expression.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[toCoord](#), [Xcoord](#), [Ycoord](#)

eeLib

Functions related to Electrical Engineering.

Functions

- [ppCap](#) - *Calculates Parallel Plate Capacitance*

Internal Content

These are used within the library but can often be useful directly accessed from without.

- private functions
 - version - returns revision date of this library
- variables
 - none*

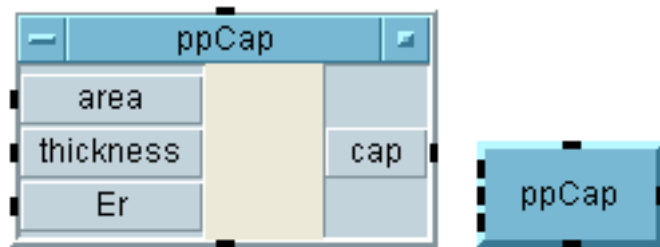
Standalone Objects

Notes

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:24. © 2015.

ppCap(area, thickness, Er)

Calculates Parallel Plate Capacitance



Syntax

ppCap(a,b)

area: scalar Real - area in mm²

thickness: scalar Real - thickness in mm

Er: scalar Real - relative dielectric constant (unitless)

return: Real - capacitance in Farads

Usage

Given an area and thickness plus the dielectric constant, basic physics provides the equivalent DC capacitance for two parallel conductors separated by a fixed amount.

Location

menu: veeos ==> Engineering Calculations ==> ppCap

library: [eeLib](#)

Example

A 1mm square pad on 10 mil high density alumina has an equivalent capacitance of $\text{ppCap}(1, 0.25, 9.9) = 0.35 \text{ pF}$

Notes

This function calculates C given the formula $C = \epsilon_0 \epsilon_r A / d$, which determines the classic DC parallel plate capacitance without taking into account any fringing effects. For capacitors whose electrical aspect ratio is small (very wide compared to thickness) this is a good approximation since fringing is minimal. For other capacitors one should start looking at EM techniques for a better answer.

Reference

- One source is [GSU Physics Department](#), although any beginning electrical field text will have this also.

Supported On

VEE 7.0+, All platforms.

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:28. © 2015.

mathLib

General mathematics functions

Functions

- [arrayConcat](#) - joins together arrays
- [arrayRev](#) - reverses the order of any 1D array
- [colMax](#) - Array statistics - determines the Maximum value of each column in an array
- [colMean](#) - Array statistics - determines the Mean value of each column in an array
- [colMedian](#) - Array statistics - determines the Median value of each column in an array
- [colMin](#) - Array statistics - determines the Minimum value of each column in an array
- [colMode](#) - Array statistics - determines the Mode of each column in an array
- [colRMS](#) - Array statistics - determines the RMS value of each column in an array
- [colSDev](#) - Array statistics - determines the Standard Deviation of each column in an array
- [colVari](#) - Array statistics - determines the Variance of each column in an array
- [concat2D](#) - joins together 2D arrays
- [concat3D](#) - joins together 3D arrays
- [dif](#) - calculates difference between successive data points
- [eval](#) - Evaluates the given function at the given argument.
- [hilbert](#) - Computes the Hilbert transform of the input waveform.
- [ifThenEval](#) - Defines a function of multiple domains - is also a general Case statement.
- [isElement](#) - determines if a given element exists within a set
- [setAND](#) - returns the set of elements that are in both of two sets
- [setComplement](#) - Returns the set of elements that are in one set but not in the other.
- [setOR](#) - returns the set of elements that are in either of two sets
- [setXOR](#) - returns the set of elements that are in either but not both of two sets
- [strReplace](#) - Replaces occurrences of a target string within the input string.
- [unique](#) - returns unique subset

Internal Content

These are used within the library but can often be useful directly accessed from without.

- private functions
 - [version](#) - returns revision date of this library
- variables
 - [none](#)

Standalone Objects

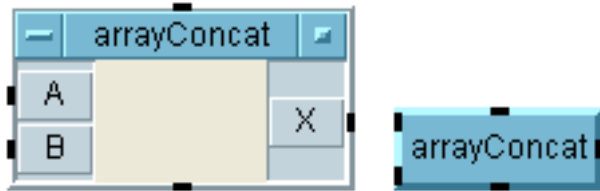
none

Notes

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:20. © 2015.

arrayConcat(a,b)

Joins together arrays.



Syntax

arrayConcat(a,b)

a: array any - first input array

b: array any - second input array

return: array any - the two input arrays concatenated together

Usage

Use this to join multiple arrays into a new array of same shape.

Location

menu: veeos ==> Math ==> Arrays ==> arrayConcat

library: [mathLib](#)

Example

```
arrayConcat(
  1 2 3 3 4
  4 5 6 , 3 4 ) yields
  7 8 9 3 4
```

```
arrayConcat(
  1 2 3 4
  4 5 , 3 4 ) yields
  7 8
  3 4
  3 4
```

Notes

While the built-in "concat()" function can join together arrays, it always flattens them into a 1D array. This function constructs a new array of same shape with the two input arrays joined side-by-side. Orientation is automatic (dimension that doesn't match gets extended. If all dimensions match then highest dim gets extended). Note that loops are not used so this function is fast.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[arrayRev](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 29 October 2015 09:40. © 2015.

arrayRev(a)

Reverses the order of any 1D array.



Syntax

arrayRev(a)

a: array 1D Any - 1D array to be processed

return: same type as input - array is in reverse order

Usage

Use any time you would like a 1D array in reverse order. Works with any data type.

Location

menu: veeos ==> Math ==> arrayRev

library: [mathLib](#)

Example

arrayRev([1,2,3,4]) returns [4,3,2,1]

arrayRev(["first","second","last"]) returns ["last" , "second" , "first"]

Notes

If the input is scalar no change is made. If the input is Waveform or Spectrum the data array is reversed but the time or freq mapping is unaffected. This uses no loops so is fast.

Reference

Dependencies

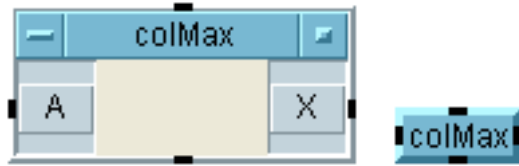
Supported On

VEE 7.0+, all platforms

See Also

colMax(a)

Array statistics - determines the Maximum value of each column in an array



Syntax

colMax(a)

a: array 2D numeric - input 2D array

return: array 2D - statistics of each column

Usage

When analyzing plentiful data, such as with a number of tests on many DUTs it can be desirable to examine the statistics of various parameters. A typical way to represent the data is in an array, with each row representing a test vector, each column a particular tested parameter. This function analyzes the Maximum value of each column- hence each parameter.

Location

menu: veeos ==> Math ==> Arrays ==> colMax

library: [mathLib](#)

Example

```
colMax([ 4 4 5 3
        3 5 4 5
        3 6 5 8 ]) returns [4 6 5 8]
```

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[colMean](#), [colMedian](#), [colMin](#), [colMode](#), [colRMS](#), [colSDev](#), [colVari](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:19. © 2015.

colMean(a)

Array statistics - determines the Mean value of each column in an array



Syntax

colmean(a)

a: array 2D numeric - input 2D array

return: array 2D - statistics of each column

Usage

When analyzing plentiful data, such as with a number of tests on many DUTs it can be desirable to examine the statistics of various parameters. A typical way to represent the data is in an array, with each row representing a test vector, each column a particular tested parameter. This function analyzes the Mean value of each column- hence each parameter.

Location

menu: veeos ==> Math ==> Arrays ==> colMean

library: [mathLib](#)

Example

```
colMean( [ 4 4 5 3
           3 5 4 5 ] returns [3.33 5 4.66 5.33]
           3 6 5 8
```

Notes

Reference

Uses no loops so is fast.

Dependencies

Supported On

VEE 7.0+, all platforms

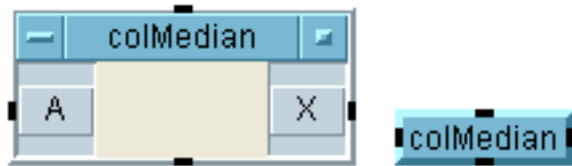
See Also

[colMax](#), [colMedian](#), [colMin](#), [colMode](#), [colRMS](#), [colSDev](#), [colVari](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:19. © 2015.

colMedian(a)

Array statistics - determines the Median value of each column in an array



Syntax

colMedian(a)

a: array 2D numeric - input 2D array

return: array 2D - statistics of each column

Usage

When analyzing plentiful data, such as with a number of tests on many DUTs it can be desirable to examine the statistics of various parameters. A typical way to represent the data is in an array, with each row representing a test vector, each column a particular tested parameter. This function analyzes the Median value of each column- hence each parameter.

Location

menu: veeos ==> Math ==> Arrays ==> colMedian

library: [mathLib](#)

Example

```
colMedian([ 4 4 5 3
            3 5 4 5 ] returns [3 5 5 5]
            3 6 5 8
```

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[colMax](#), [colMean](#), [colMin](#), [colMode](#), [colRMS](#), [colSDev](#), [colVari](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:18. © 2015.

colMin(a)

Array statistics - determines the Minimum value of each column in an array



Syntax

colMin(a)

a: array 2D numeric - input 2D array

return: array 2D - statistics of each column

Usage

When analyzing plentiful data, such as with a number of tests on many DUTs it can be desirable to examine the statistics of various parameters. A typical way to represent the data is in an array, with each row representing a test vector, each column a particular tested parameter. This function analyzes the Minimum value of each column- hence each parameter.

Location

menu: veeos ==> Math ==> Arrays ==> colMin

library: [mathLib](#)

Example

```
colMin([ 4 4 5 3
        3 5 4 5
        3 6 5 8 ]) returns [3 4 4 3]
```

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[colMax](#), [colMean](#), [colMedian](#), [colMode](#), [colRMS](#), [colSDev](#), [colVari](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:18. © 2015.

colMode(a)

Array statistics - determines the Mode of each column in an array



Syntax

colMode(a)

a: array 2D numeric - input 2D array

return: array 2D - statistics of each column

Usage

When analyzing plentiful data, such as with a number of tests on many DUTs it can be desirable to examine the statistics of various parameters. A typical way to represent the data is in an array, with each row representing a test vector, each column a particular tested parameter. This function analyzes the Mode (most common value) of each column- hence each parameter.

Location

menu: veeos ==> Math ==> Arrays ==> colMode

library: [mathLib](#)

Example

```
colMode([ 4 4 5 3
          3 5 4 5
          3 6 5 8]) returns [3 4 5 3]
```

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[colMax](#), [colMean](#), [colMedian](#), [colMin](#), [colRMS](#), [colSDev](#), [colVari](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:17. © 2015.

colRMS(a)

Array statistics - determines the RMS (root mean square) value of each column in an array



Syntax

colRMS(a)

a: array 2D numeric - input 2D array

return: array 2D - statistics of each column

Usage

When analyzing plentiful data, such as with a number of tests on many DUTs it can be desirable to examine the statistics of various parameters. A typical way to represent the data is in an array, with each row representing a test vector, each column a particular tested parameter. This function analyzes the Maximum value of each column- hence each parameter.

Location

menu: veeos ==> Math ==> Arrays ==> colRMS

library: [mathLib](#)

Example

```

      4 4 5 3
colRMS([ 3 5 4 5 ] returns [3.37 5.07 4.69 5.72]
      3 6 5 8

```

Notes

Uses no loops so is fast.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[colMax](#), [colMean](#), [colMedian](#), [colMin](#), [colMode](#), [colSDev](#), [colVari](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:18. © 2015.

colSDev(a)

Array statistics - determines the Standard Deviation of each column in an array



Syntax

colSDev(a)

a: array 2D numeric - input 2D array

return: array 2D - statistics of each column

Usage

When analyzing plentiful data, such as with a number of tests on many DUTs it can be desirable to examine the statistics of various parameters. A typical way to represent the data is in an array, with each row representing a test vector, each column a particular tested parameter. This function analyzes the Standard Deviation of each column- hence each parameter.

Location

menu: veeos ==> Math ==> Arrays ==> colSDev

library: [mathLib](#)

Example

```
colSDev([ 4 4 5 3
          3 5 4 5
          3 6 5 8 ]) returns [.82 1.41 .82 3.56]
```

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[colMax](#), [colMean](#), [colMedian](#), [colMin](#), [colMode](#), [colRMS](#), [colVari](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:17. © 2015.

colVari(a)

Array statistics - determines the Variance of each column in an array



Syntax

colVari(a)

a: array 2D numeric - input 2D array

return: array 2D - statistics of each column

Usage

When analyzing plentiful data, such as with a number of tests on many DUTs it can be desirable to examine the statistics of various parameters. A typical way to represent the data is in an array, with each row representing a test vector, each column a particular tested parameter. This function analyzes the Variance of each column- hence each parameter.

Location

menu: veeos ==> Math ==> Arrays ==> colVari

library: [mathLib](#)

Example

```
colVari([ 4 4 5 3
          3 5 4 5 ] returns [ .67 2 .67 12.67 ]
          3 6 5 8
```

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[colMax](#), [colMean](#), [colMedian](#), [colMin](#), [colMode](#), [colRMS](#), [colSDev](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:17. © 2015.

concat2D(a,b)

Joins together 2D arrays.

***NOTE** - this function is maintained for backward-compatibility but is now deprecated in favor of [arrayConcat\(a,b\)](#)*



Syntax

concat2D(a,b)

a: array 2D any - first input array

b: array 2D any - second input array

return: array 2D any - the two input arrays concatenated together

Usage

Use this to join multiple 2D arrays into a new 2D array.

Location

menu: veeos ==> Math ==> concat2D

library: [mathLib](#)

Example

```
concat2D ( 1 2 3 3 4
           4 5 6 , 3 4 ) yields 1 2 3 3 4
           7 8 9 3 4           4 5 6 3 4
                               7 8 9 3 4
```

```
concat2D ( 1 2 3 4
           4 5 , 3 4 ) yields 1 2
           7 8               4 5
                               7 8
                               3 4
                               3 4
```

Notes

While the built-in "concat()" function can join together 2D arrays, it always flattens them into a 1D array. This function maintains the 2D shape and automatically determines the correct orientation. If number of rows match then columns are joined, and if number of columns match then rows are joined. If neither match then an error is raised. Note that loops are not used so this function is fast.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[concat3D](#), [arrayConcat](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:19. © 2015.

concat3D(a,b)

Joins together 3D arrays.

***NOTE** - this function is maintained for backward-compatibility but is now deprecated in favor of [arrayConcat\(a,b\)](#)*



Syntax

concat3D(a,b)

a: array 3D any - first input array

b: array 3D any - second input array

return: array 3D any - the two input arrays concatenated together

Usage

Use this to join multiple 3D arrays into a new 3D array.

Location

menu: veeos ==> Math ==> concat3D

library: [dataLib](#)

Example

See example in veeos menu.

Notes

While the built-in "concat()" function can join together 3D arrays, it always flattens them into a 1D array. This function maintains the 3D shape and automatically determines the correct orientation. Note that two of the dimensions need to match before it is feasible to join the arrays. If no two match then an error is raised. Note that loops are not used so this function is fast.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[concat2D](#), [arrayConcat](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 28 October 2015 11:19. © 2015.

dif(a)

Calculates difference between successive data points.



Syntax

dif(a)

a: array numeric - array of data values

return: array numeric - difference between successive data points

Usage

Replaces an array with the difference between successive points. This is the "delta" in difference equations. In order to maintain the same qty of values, the first data point is kept.

Location

menu: veeos ==> Math ==> dif

library: [mathLib](#)

Example

dif([1 4 3 6 7 8]) returns [1 3 -1 3 1 1]

Notes

When creating difference equations, say $\text{dif}(a) / (\text{dif}(a) + 1)$ one of course needs a dif function, just as one needs an integral function for differential equations. This rendition of *dif* fits that bill. Note that *dif* should actually return one less point than the input, but in order to make successive operations easy the input array length is maintained by arbitrarily keeping the first data point in place. This will of course cause end effects.

Reference

Dependencies

Supported On

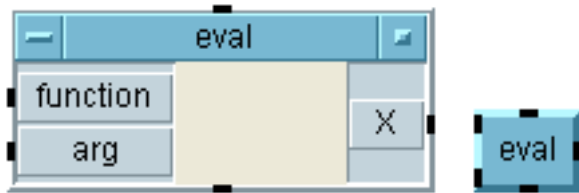
VEE 7.0+, all platforms

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:23. © 2015.

eval(function,arg)

Evaluates the given function at the given argument.



Syntax

eval(function,arg)

a: scalar Text - function of A

b: any - value of A

return: function evaluated at the given value of A

Usage

This is designed for use in cases where the function is determined at runtime.

Location

menu: veeos ==> Math ==> eval

library: [mathLib](#)

Example

eval("2+a^2", 4) returns 18

eval("3*A+15", "4") returns 44415

Notes

When a formula is known at programming time it of course can be coded into a formula and then evaluated at runtime. However in cases such as filters where the function to be evaluated is itself determined only at runtime, one needs the ability to evaluate that function. eval(function,arg) provides a convenient way to do this

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[ifThenEval](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:24. © 2015.

hilbert(a)

Computes the Hilbert transform of the input waveform.



Syntax

hilbert(a)

a: waveform - the input waveform

return: waveform - Hilbert transform of the input waveform

Usage

Computes the Hilbert transform.

Location

menu: veeos ==> Math ==> hilbert

library: [mathLib](#)

Example

For given waveform A, hilbert(a) returns the Hilbert transform of A

Notes

The current code is based on a contribution to VRF by Tom Sanders in 1999.

Reference

Hilbert transform is well described at [Wikipedia](#) and is a very commonly used function in Signal processing.

Dependencies

none.

Supported On

VEE 7.0+, All platforms.

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:25. © 2015.

ifThenEval(cases,A)

Defines a function of multiple domains - is also a general Case statement.



Syntax

ifThenEval(cases,A)

cases: Array 1D Record - defines the cases to be considered

A: any - defines the independent variable

return: any - for each case the result of the input formula

Usage

Use this where you would normally use a Case statement. The input "cases" is a array of records. Each record has two fields, each of type Text and each one defining a formula to be evaluated. The "if" field of the first record (case) is evaluated. If True then the "eval" field is evaluated and the result is returned. If False then the next record (case) is evaluated the same way. As soon as any condition is matched the function exits. If no condition matches then the original value is returned.

Location

menu: veeos ==> Math ==> ifThenEval

library: [mathLib](#)

Example

Define a function of two domains: $A > 0$ and $A < 5$ and $A > 5$ and two accompanying formulas $1/A$ and $2*A$.
For the ifThenEval we then create a record array with two fields such that

```
rec[0].if = "A>0 and A<5";
rec[0].eval="1/A";
rec[1].if = "A>5";
rec[1].eval = "2*A";
```

Feeding values A 0,1,2,3,4,5,6,7 to *ifThenEval(rec,A)* we obtain an output of 0,1,1/2,1/3,1/4,5,16,14

Notes

VEE never had a Case statement but has always needed one. Originally there was a good reason that Case was not included but, although that reason went away many years ago, Case was never added. This function was developed to fill that gap.

Reference

Dependencies

Use [eval\(\)](#) extensively

Supported On

VEE 7.0+, all platforms

See Also

[eval](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:25. © 2015.

isElement(set,element)

Determines if a given element exists within a set.



Syntax

isElement(set,element)

a: array any - input set

element: scalar any - element to look for in the set

return: scalar int - "1" if element is in set, "0" if not

Usage

Use this to determine if a given element exists within a set.

Location

menu: veeos ==> Math ==> Sets ==> isElement

library: [mathLib](#)

Example

isElement([2 3 4 3 5 5 3 1], 6) returns 0 ("false")

isElement([2 3 4 3 5 5 3 1], 2) returns 1 ("true")

Notes

When dealing with sets it is often useful to know if an element exists within a given set. A typical case is when looking for a specific test result in an array of results, based on a specific detail like an operator name.

Reference

Dependencies

Supported On

isElement

VEE 7.0+, all platforms

See Also

[setAND](#), [setComplement](#), [setOR](#), [setXOR](#), [unique](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 24 October 2015 19:30. © 2015.

setAND(a,b)

Returns the set of elements that are in both of two sets.



Syntax

setAND(a,b)

a: array any - input set

b: array any - input set

return: array 1D any - array that includes all elements that are part of both A and B

Usage

Use this to perform set operation "intersection"

Location

menu: veeos ==> Math ==> Sets ==> setAND

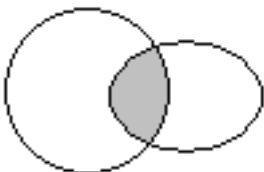
library: [mathLib](#)

Example

setAND([2 3 4 3 5 5 3 1], [6 3 4 7 1]) returns [3 4 1]

Notes

setAND is equivalent to an intersection of two sets with duplicate elements removed, as in the intersecting area of a Venn diagram.



Reference

Dependencies

uses unique(A)

Supported On

VEE 7.0+, all platforms

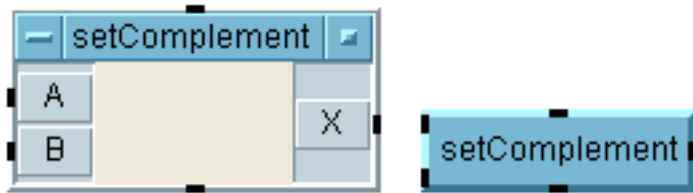
See Also

[isElement](#), [setComplement](#), [setOR](#), [setXOR](#), [unique](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 24 October 2015 19:30. © 2015.

setComplement(a,b)

Returns the set of elements that are in one set but not in the other.



Syntax

setComplement(a,b)

a: array any - input set

b: array any - input set

return: array 1D any - array that includes all elements that are part of A but not part of B

Usage

Use this to perform set operation "relative complement", which is analogous to a subtraction.

Location

menu: veeos ==> Math ==> Sets ==> setComplement

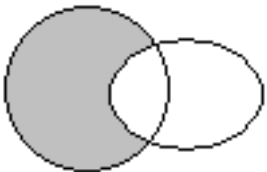
library: [mathLib](#)

Example

setComplement([2 3 4 3 5 5 3 1], [6 3 4 7 1]) returns [2 5]

Notes

setComplement is equivalent to a subtraction of two sets, as in the area of a Venn diagram that is part of set A but not part of set B.



Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[isElement](#), [setAND](#), [setOR](#), [unique](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 24 October 2015 19:36. © 2015.

setOR(a,b)

Returns the set of elements that are in either of two sets.



Syntax

setOR(a,b)

a: array any - input set

b: array any - input set

return: array 1D any - array that includes all elements that are part of either A or B

Usage

Use this to perform set operation "union"

Location

menu: veeos ==> Math ==> Sets ==> setOR

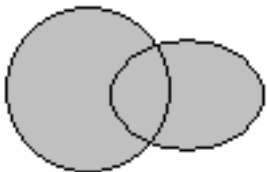
library: [mathLib](#)

Example

setOR([2 3 4 3 5 5 3 1], [6 3 4 7 1]) returns [2 5 6 3 4 7 1]

Notes

setOR is equivalent to a union of two sets with duplicate elements removed, as in a Venn diagram.



Reference

Dependencies

uses unique(A)

Supported On

VEE 7.0+, all platforms

See Also

[isElement](#), [setAND](#), [setComplement](#), [setXOR](#), [unique](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 24 October 2015 19:30. © 2015.

setXOR(a,b)

Returns the set of elements that are in either but not both of two sets.



Syntax

setXOR(a,b)

a: array any - input set

b: array any - input set

return: array 1D any - array that includes all elements that are part of either A or B but not both A and B

Usage

Use this to perform set operation "union - intersection"

Location

menu: veeos ==> Math ==> Sets ==> setXOR

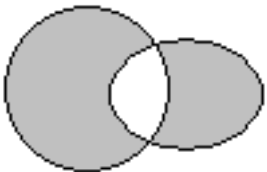
library: [mathLib](#)

Example

setXOR([2 3 4 3 5 5 3 1], [6 3 4 7 1]) returns [2 5 6 7]

Notes

setXOR is equivalent to a union of two sets minus the intersection of those sets, as in the area of a Venn diagram outside of the intersecting area.



Reference

Dependencies

Supported On

VEE 7.0+, all platforms

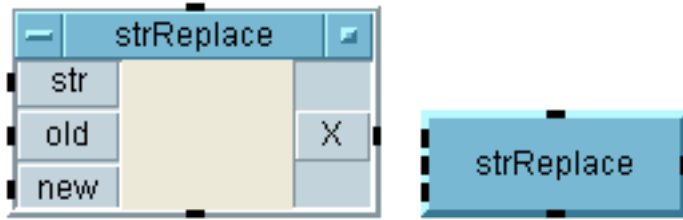
See Also

[isElement](#), [setAND](#), [setComplement](#), [setOR](#), [unique](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 24 October 2015 19:30. © 2015.

strReplace(str,old,new)

Replaces occurrences of a target string within the input string.



Syntax

strreplace(str,old,new)

str: scalar Text - input string

old: scalar Text - target string

new: scalar Text - replacement string

return: Text with substituted string

Usage

The input string is searched for the target string. When found the replacement string is inserted in place of the target string. The resulting string is then processed in the same way until target string is no longer found.

Location

menu: veeos ==> math ==> strReplace

library: [mathLib](#)

Example

```
strReplace("I think this is an unterestung strung","un","in")
```

returns "I think this is an interesting string"

Notes

This objects recursively looks through the string to find and replace strings. Hence it may be possible to have unintended results if one replacement triggers another. In the future may want to rewrite to take this into account.

Reference

Supported On

strReplace

VEE 7.0+. All platforms.

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:29.

unique(a)

Returns unique subset.



Syntax

unique(a)

a: array any - input set

return: array 1D any - array with all duplicated elements removed

Usage

Use this to remove duplicate elements in an array.

Location

menu: veeos ==> Math ==> Sets ==> unique

library: [mathLib](#)

Example

unique([2 3 4 3 5 5 3 1]) returns [2 4 5 3 1]

Notes

When dealing with sets it is often useful to remove duplicate entries. A typical case is when processing all elements individually and not wanting to perform the processing twice for a duplicated element. Note that in the case of duplicates, the *last* identical element is retained.

Reference

Dependencies

use isElement(set,element)

Supported On

VEE 7.0+, all platforms

See Also

[isElement](#), [setAND](#), [setComplement](#), [setOR](#), [setXOR](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 24 October 2015 19:30. © 2015.

netLib

Functions that use the internet.

Functions

- [longLat](#) - returns the longitude and latitude for a given city
- [currentTemp](#) - returns the current temperature for a given city
- [currentHumidity](#) - returns the current humidity for a given city

Internal Content

These are used within the library but can often be useful directly accessed from without.

- private functions
 - version - returns revision date of this library
- variables
 - none*

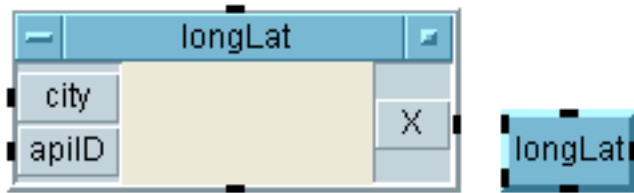
Standalone Objects

Notes

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:26. © 2015.

longLat(city,apiID)

Returns the longitude and latitude for a given city



Syntax

longLat(city,apiID)

city: scalar Text - name of city

apiID: scalar text - API ID from website

return: coord - longitude and latitude

Usage

Queries the web for longitude and latitude coordinates. City can be specified by full name, or for the USA, by zip code.

Location

menu: veeos ==> Web ==> longLat

library: [netLib](#)

Example

Notes

This function takes advantage of the published API from [openweathermap.org](#). Obtain your user ID by registering with [openweathermap](#). The ID is free and will be usable for all API calls from [openweathermap.org](#). Since this depends upon web services and the concomitant network connectivity, plus potential non-availability of specific location data, expect occasional intermittent behavior.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

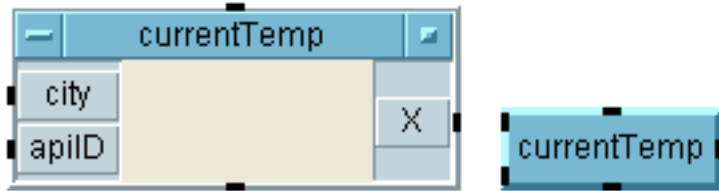
longLat

[currentHumidity](#), [currentTemp](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:26. © 2015.

currentTemp(city,apiID)

Returns the current temperature for a given city



Syntax

currentTemp(city)

city: scalar Text - name of city

apiID: scalar text - API ID from website

return: real - temperature in deg F

Usage

Queries the web for current temperature. City can be specified by full name, or for the USA, by zip code.

Location

menu: veeos ==> web ==> currentTemp

library: netLib

Example

Notes

This function takes advantage of the published API from openweathermap.org. Obtain your user ID by registering with [openweathermap](http://openweathermap.org). The ID is free and will be usable for all API calls from openweathermap.org. Since this depends upon web services and the concomitant network connectivity, plus potential non-availability of specific location data, expect occasional intermittent behavior.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

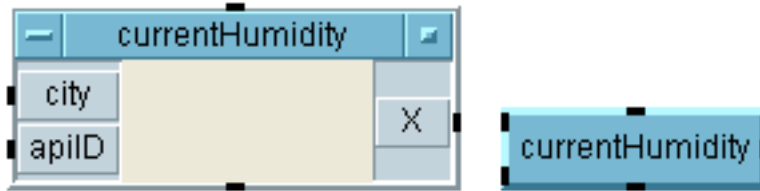
currenttemp

[longLat](#), [currentTemp](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:22. © 2015.

currentHumidity(city,apiID)

Returns the current humidity for a given city



Syntax

currentTemp(city)

city: scalar Text - name of city

apiID: scalar text - API ID from website

return: real - relative humidity in %

Usage

Queries the web for current humidity. City can be specified by full name, or for the USA, by zip code.

Location

menu: veeos ==> web ==> currentHumidity

library: netLib

Example

Notes

This function takes advantage of the published API from openweathermap.org. Obtain your user ID by registering with [openweathermap](http://openweathermap.org). The ID is free and will be usable for all API calls from openweathermap.org. Since this depends upon web services and the concomitant network connectivity, plus potential non-availability of specific location data, expect occasional intermittent behavior.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

currentHumidity

[longLat](#), [currentTemp](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:22. © 2015.

sigLib

Signal Processing Components

Functions

- [dcBlock](#) - removes the DC component of a signal
- [decayFilter](#) - Implements a filter whose response is based on a given decay rate.
- [dutyCycle](#) - Determines the duty cycle of an arbitrary signal
- [fallTime](#) - Determines the fall time of an arbitrary signal
- [frequency](#) - Determines the dominant frequency of an arbitrary signal.
- [funcFilter](#) - Implements an analog filter with transfer response specified by an arbitrary function.
- [period](#) - Determines the period of an arbitrary signal.
- [polyFilter](#) - Implements an analog filter with transfer response specified by numerator and denominator polynomials.
- [riseTime](#) - Determines the rise time of an arbitrary signal
- [rootFilter](#) - Implements an analog filter with transfer response specified by poles and zeroes.

Internal Content

These are used within the library but can often be useful directly accessed from without.

- private functions
 - [aAmp](#) - generic analog amplifier
 - [aFilter](#) - generic classical analog filter
 - [aMixer](#) - generic analog mixer
 - [aRectifier](#) - generic analog rectifier
 - [butterPoly](#) - generates Butterworth polynomials
 - [chebyPoly](#) - generates Chebychev polynomials
 - version - returns revision date of this library
- variables
none

Standalone Objects

- [analogAmp](#) - configurable analog amplifier
- [analogFilter](#) - configurable analog classical filter
- [downConvert](#) - mixer-based down-conversion object
- [negBridge](#) - An ideal negative polarity full-wave rectifier.
- [negRectifier](#) - An ideal negative polarity half-wave rectifier.
- [posBridge](#) - An ideal positive polarity full-wave rectifier.
- [posRectifier](#) - An ideal positive polarity half-wave rectifier.
- [upConvert](#) - mixer-based up-conversion object

Notes

Since VEE was originally designed as a Signal Processing toolbench, it has many capabilities in that area that have been mostly neglected over the years. These signal processing components are designed to encourage quick and easy nonlinear signal analysis, generally in near-real-time, that in many cases is far simpler than using common circuit analysis tools. As with circuit tools, VEE circuits are strung together using components with

more-or-less direct real world analogs, but without the often-complicated setup and batch processing that comes with circuit analysis tools. Given this, VEE can also be extended to incorporate circuit analysis tools in a way that hides them from the end user, ending up with the full power of circuit analysis but with the ease-of-use of graphical programming.

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 31 October 2015 20:34. © 2015.

dcBlock(a)

Removes the DC component of a signal.



Syntax

dcBlock(a)

a: waveform - input signal

return: waveform - same signal but with the DC component removed

Usage

Use wherever a DC block would normally be required in a circuit. Typical use is to isolate bias voltages or to protect sensitive inputs.

Location

menu: veeos ==> Signal processing ==> Devices ==> dcBlock

library: [sigLib](#)

Example

Notes

A classical DC block is in practice usually implemented as a low-leakage series capacitor. As such it always has a lower frequency cutoff due to the finite capacitance and a higher frequency cutoff due to the physical size and concomitant resonances that occur. In this case the dcBlock is implemented mathematically so is ideal, having none of the limitations of a capacitor.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[negBridge](#), [posBridge](#), [negRectifier](#), [posRectifier](#)

decayFilter(a,decayRate)

Implements a filter whose response is based on a given decay rate.



Syntax

```
decayFilter(a,decayRate)
a: waveform - input waveform
decayRate: numeric - decay rate in percentage
return: waveform- input transformed by the decay-based filter
```

Usage

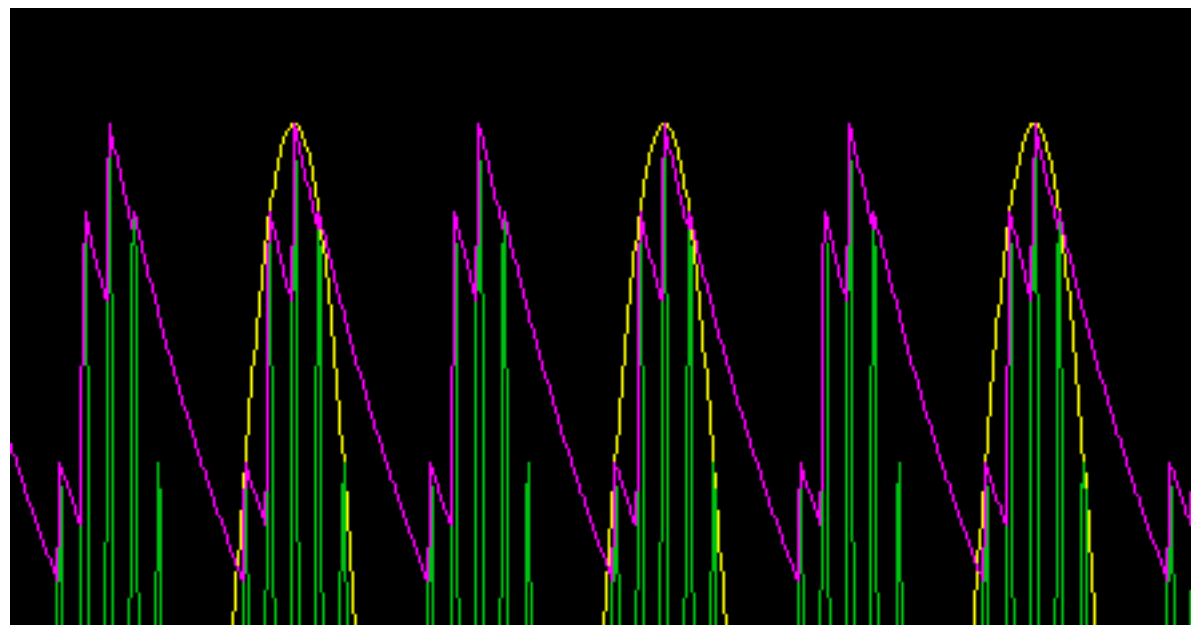
Typical usage is to model a diode-RC detector, but can be used in any case where the output signal is a decayed version of the input signal.

Location

menu: veeos ==> Signal processing ==> Modules ==> decayFilter

library: [sigLib](#)

Example



Notes

A very basic nonlinear filter can be implemented with a diode and a capacitor. The capacitor provides charge storage (i.e. memory) and the diode provides charges to the capacitor in one direction only (i.e. charges cap but doesn't discharge it). This provides a non-linear filter such that it follows a rising input but the output decays at a specified rate(so-called slew rate). A typical use for this is for recovering an envelope from an AM-modulated signal.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

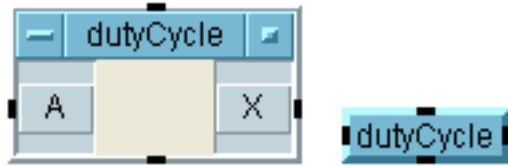
See Also

[analogFilter](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 01 November 2015 08:12. © 2015.

dutyCycle(a)

Determines the duty cycle of an arbitrary signal.



Syntax

dutyCycle(a)

a: waveform - signal to be analyzed

return: scalar Real- duty cycle of the (presumed) periodic signal

Usage

Returns the duty cycle of a signal, as defined by 10-90 transitions between a "high" and "low" value.

Location

menu: veeos ==> Signal Processing ==> dutyCycle

library: [sigLib](#)

Example

see example in veeos menu

Notes

One common parameter for digital signals is duty cycle: the relative time spent "high" vs "low". This is normally presented as a ratio, with "1" being "high all the time" and "0" being "low all the time". This function uses that standard and defines "up" as within 10% of highest value and "down" as within 10% of the lowest value. Given these margins the function is resistant to noise in the signal, but of course gets best results with many cycles.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

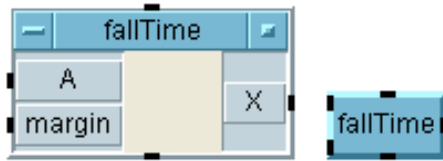
See Also

[fallTime](#), [frequency](#), [period](#), [riseTime](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 30 October 2015 16:45. © 2015.

fallTime(a,margin)

Determines the fall time of an arbitrary signal



Syntax

fallTime(a,margin)

a: waveform - signal to be analyzed

margin: scalar numeric - margin to be used for fall time, in percent

return: scalar Real- fall time of signal

Usage

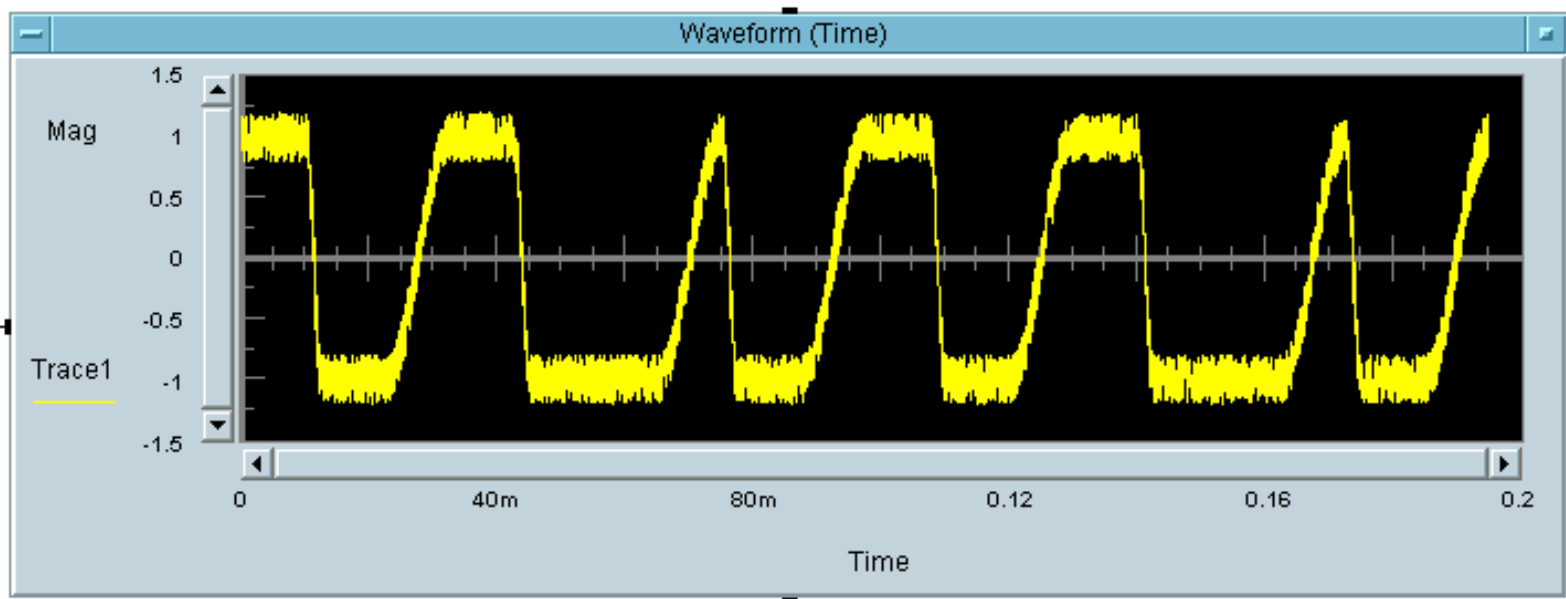
One common parameter for digital signals (pulses, square waves, etc.) is fall time. Fall time is the transition time from a lower extreme to an upper extreme. This is commonly defined as 10-90 time (time from 10% of the full transition to 90%) or 20-80 fall time. This function takes an arbitrary signal and analyzes it to determine the average fall time. The input *margin* is used to define both the upper and lower transition limits. Hence margin=10 is the same as 10-90 fall time.

Location

menu: veeos ==> Signal Processing ==> fallTime

library: [sigLib](#)

Example



Notes

This function is very general in nature so any signal may be used. It finds all applicable transitions in the given signal and then averages them to give an estimate of the signal's fall time. Note that it does not treat any particular fall time as more important than any other so if you are looking for transitions in very noisy signals you may want to examine the signal differently. Since *margin* can be specified you are free to define fall as you like.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

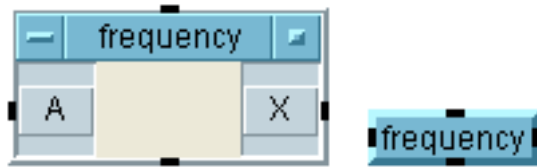
See Also

[dutyCycle](#), [frequency](#), [period](#), [riseTime](#),

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 01 November 2015 08:13. © 2015.

frequency(a)

Determines the dominant frequency of an arbitrary signal.



Syntax

frequency(a)

a: waveform - signal to be analyzed

return: scalar Real- frequency of the (presumed) periodic signal

Usage

Think of this as a noise-resistant frequency counter. It examines the signal and returns a representative frequency for the signal.

Location

menu: veeos ==> Signal Processing ==> frequency

library: [sigLib](#)

Example

see examples in veeos menu

Notes

One common parameter for periodic signals is frequency. This is generally the dominant frequency and can be determined by various methods. This function looks at the spectrum and determines the peak. As such it is very tolerant to noise. For amplitude modulated signals this will give a better frequency estimate than using 1/[period](#) since the latter may well see the modulation rather than the carrier.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[dutyCycle](#), [fallTime](#), [period](#), [riseTime](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](#). Last updated 30 October 2015 16:44. © 2015.

funcFilter(a,Hs)

Implements an analog filter with transfer response specified by an arbitrary function.



Syntax

funcFilter(aHs)

a: Waveform or Spectrum - inputSignal

Hs: scalar Text - any function with argument "s"

return: Waveform or Spectrum - output signal formed by passing through the filter defined by Hs

Usage

Analog filters are generally specified by a rational polynomial. However there's no reason why this can't be generalized by allowing most any transfer function $H(s)$. Use this filter for when you would like to use any $H(s)$.

Location

menu: veeos ==> Signal Processing ==> Modules ==> funcFilter

library: [sigLib](#)

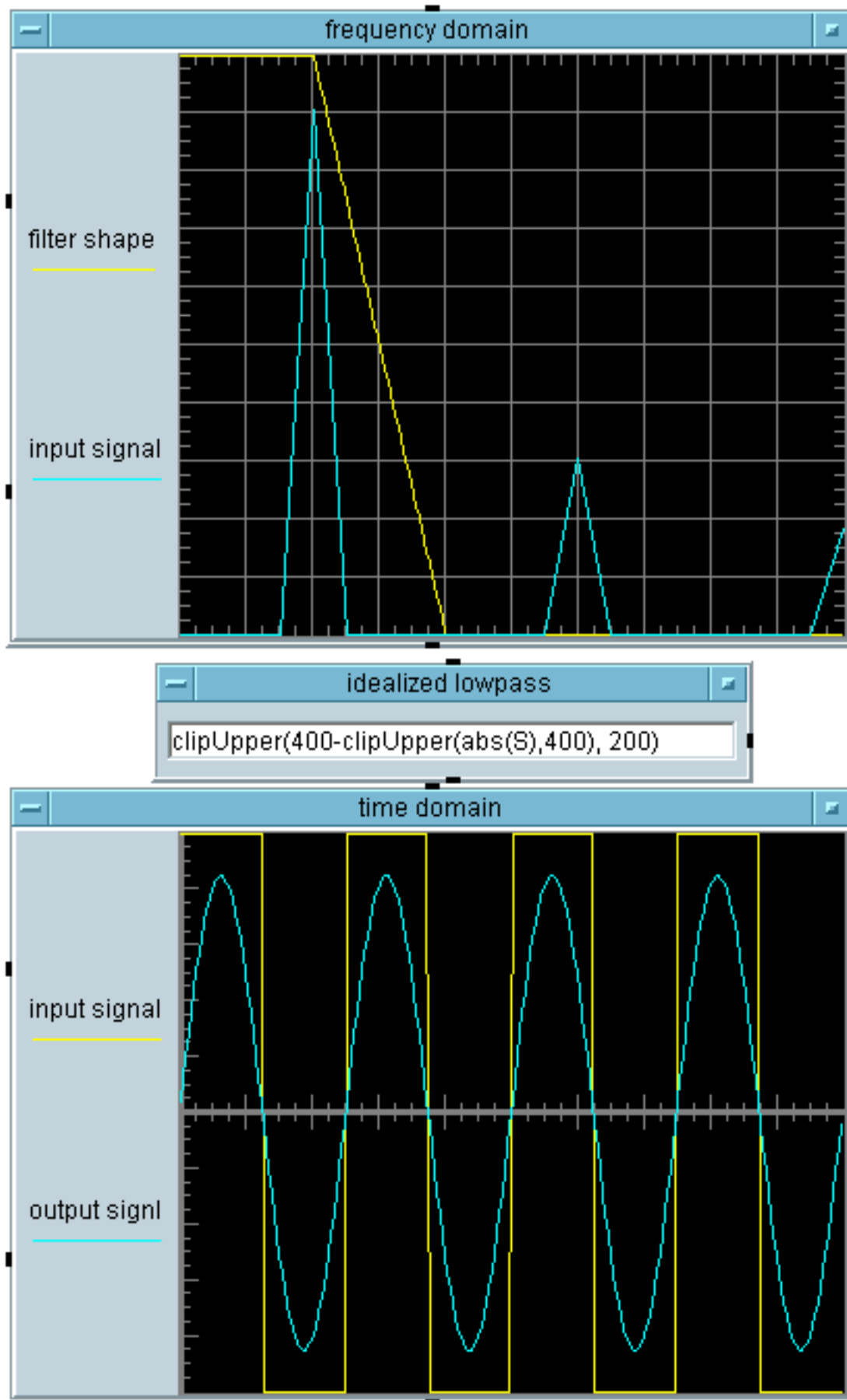
Example

Brickwall filter: The ultimate lowpass filter would remove all frequencies above a given limit. This is not realizable (is the limit of an infinite number of elements) but can certainly be modeled. For instance

$H(s) = (\text{im}(s) < 450 ? 1 : 0)$ would remove everything above 450Hz and allow everything below it to pass unimpeded

Phase shift: a pure phase shift is also not achievable in the real world except for at single frequency, but in the mathematical world is quite doable. In this case

$H(s) = \exp(S)$ shifts the phase of everything by 90 degrees (e^s becomes e^{jF} and hence 90 degree shift).



Notes

Rational polynomials are used for a very good reason to define filters: they are guaranteed to be stable (if poles are in LHP) , causal, and physically realizable. However there's nothing that says one can't use most any function to define the frequency response.

NOTE that the transfer function $H(s)$ has a complex argument and that this filter filter operates on the jF axis (real frequencies) so any function that you write must be valid for complex numbers. Since some VEE functions do not support complex arguments you do need to be a little careful here.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[aFilter](#), [AnalogFilter](#), [rootFilter](#), [funcFilter](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 01 November 2015 08:16. © 2015.

period(a)

Determines the period of an arbitrary signal.



Syntax

period(a)

a: waveform - signal to be analyzed

return: scalar Real- period of the (presumed) periodic signal

Usage

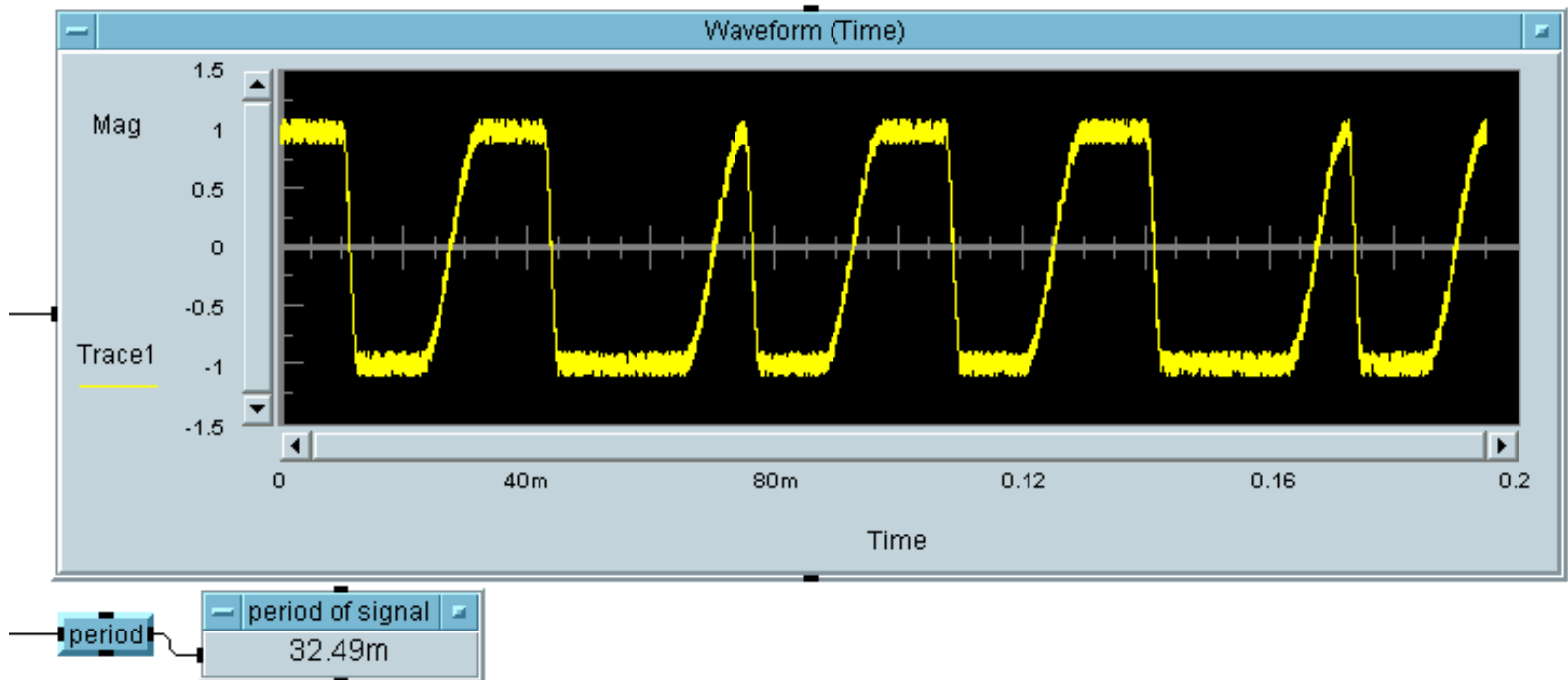
One common parameter for digital signals (pulses, square waves, etc.) is period. Period is the time between successive transitions of the signal. This function takes an arbitrary signal and analyzes it to determine the average period.

Location

menu: veeos ==> Signal Processing ==> period

library: [sigLib](#)

Example



Notes

This function is very general in nature so any signal may be used. It finds all applicable times between transitions in the given signal and then averages them to give an estimate of the signal's period. This function defines transitions as 10-90% so can handle a fair amount of noise in the signal. However, also note that this function would not be suitable for amplitude modulated signals.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

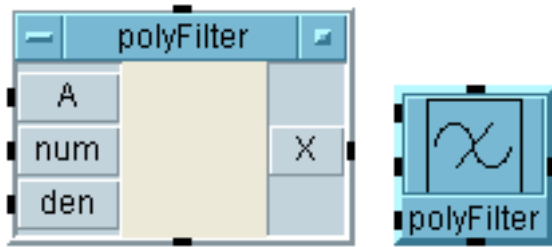
See Also

[fallTime](#), [frequency](#), [riseTime](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](#). Last updated 01 November 2015 08:12. © 2015.

polyFilter(a,num,den)

Implements an analog filter with transfer response specified by numerator and denominator polynomials.



Syntax

polyFilter(a,num,den)

a: Waveform or Spectrum - inputSignal

num: array 1D numeric - coefficients of numerator polynomial (0 thru n)

den: array 1D numeric - coefficients of denominator polynomial (0 thru n)

return: Waveform or Spectrum - output signal formed by passing through defined filter

Usage

Analog filters are generally specified by a rational polynomial. Use this filter when you know the coefficients of the defining polynomial.

Location

menu: veeos ==> Signal Processing ==> Modules ==> polyFilter

library: [sigLib](#)

Example

see example in veeos menu

Notes

In general a linear (analog) filter may be defined by the rational polynomial $H(S)$ where $\text{Output}(S) = \text{Input}(S) * H(S)$. A typical $H(S)$ rational polynomial may be expressed as:

$$(n[0]*S^0 + n[1]*S^1 + \dots + n[i]*S^i) / (p[0]*S^0 + p[1]*S^1 + \dots + p[i]*S^i)$$

where n and p are the coefficient vectors that define these polynomials. This function inputs those two vectors, determines $H(jF)$, and filters the input signal appropriately.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[aFilter](#), [AnalogFilter](#), [funcFilter](#), [rootFilter](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 31 October 2015 20:34. © 2015.

riseTime(a,margin)

Determines the rise time of an arbitrary signal.



Syntax

riseTime(a,margin)

a: waveform - signal to be analyzed

margin: scalar numeric - margin to be used for rise time, in percent

return: scalar Real- rise time of signal

Usage

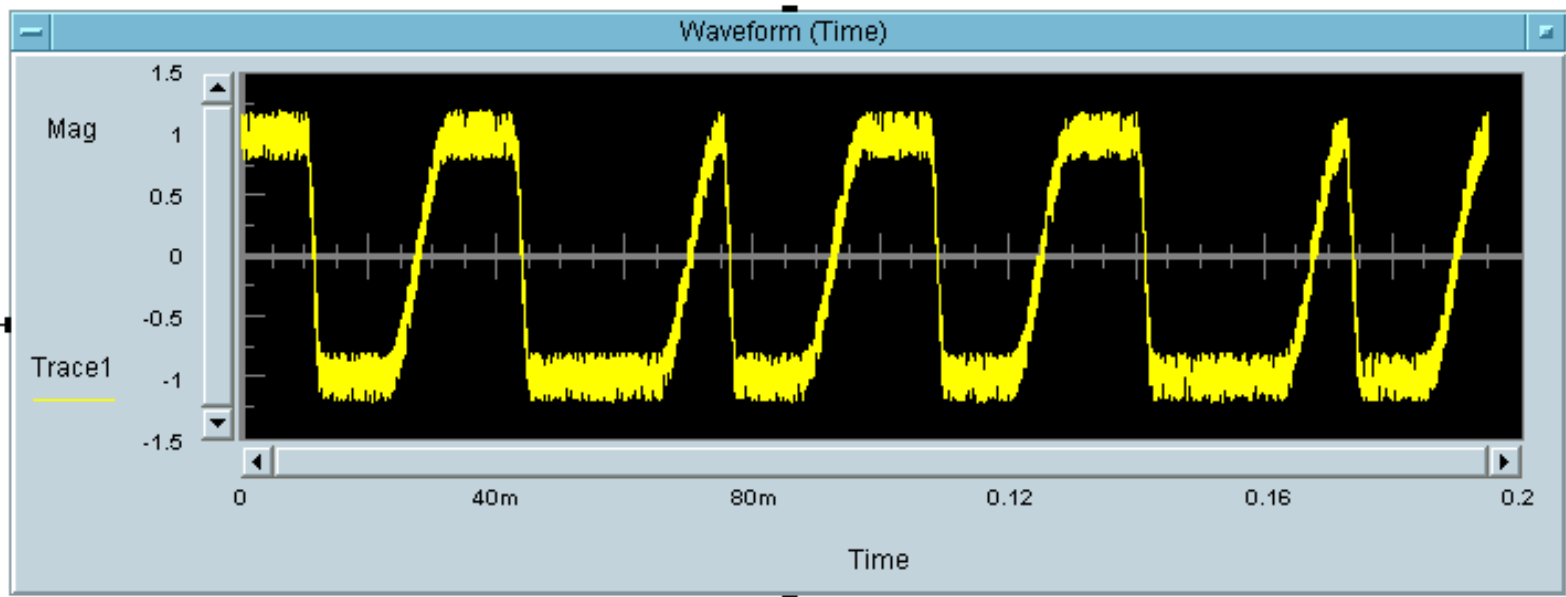
One common parameter for digital signals (pulses, square waves, etc.) is rise time. Rise time is the transition time from a lower extreme to an upper extreme. This is commonly defined as 10-90 time (time from 10% of the full transition to 90%) or 20-80 rise time. This function takes an arbitrary signal and analyzes it to determine the average rise time. The input *margin* is used to define both the upper and lower transition limits. Hence margin=10 is the same as 10-90 rise time.

Location

menu: veeos ==> Signal Processing ==> riseTime

library: [sigLib](#)

Example



Notes

This function is very general in nature so any signal may be used. It finds all applicable transitions in the given signal and then averages them to give an estimate of the signal's rise time. Note that it does not treat any particular rise time as more important than any other so if you are looking for transitions in very noisy signals you may want to examine the signal differently. Since *margin* can be specified you are free to define rise as you like.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

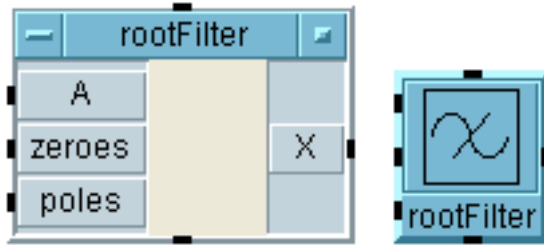
See Also

[dutyCycle](#), [fallTime](#), [frequency](#), [period](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 01 November 2015 08:12. © 2015.

rootFilter(a,zeroes,poles)

Implements an analog filter with transfer response specified by poles and zeroes.



Syntax

rootFilter(a,zeroes,poles)

a: Waveform or Spectrum - inputSignal

zeroes: array 1D Complex - zeroes of numerator polynomial

poles: array 1D Complex - zeroes of denominator polynomial

return: Waveform or Spectrum - output signal formed by passing through defined filter

Usage

Analog filters are generally specified by a rational polynomial. Use this filter when you know the roots (zeroes and poles) of the defining polynomial.

Location

menu: veeos ==> Signal Processing ==> Modules ==> rootFilter

library: [sigLib](#)

Example

see example in veeos menu

Notes

In general a linear (analog) filter may be defined by the rational polynomial $H(S)$ where $\text{Output}(S) = \text{Input}(S) * H(S)$. A typical $H(S)$ rational polynomial may be expressed as:

$$\left((S-z[0]) * (S-z[1]) * \dots (S-z[i]) \right) / \left((S-p[0]) * (S-p[1]) * \dots (S-p[i]) \right)$$

where z and p are the roots of the numerator and denominator polynomials. This function inputs those two vectors, determines $H(jF)$, and filters the input signal appropriately.

If you have a zeroes at infinity then the numerator will be just a constant. In this case you may be better off using [polyFilter](#). However, perhaps you really do want to work with the poles separately, in which case you can still proceed by noting that the polynomial has factors of the form $(S-z[0])$ and that you are free to specify anything you like for $z[0]$. In this case you can, for instance, choose $z[0] = "S-67"$ so that $S-z[0]$ becomes $S-S$.

rootFilter

+67 or just 67.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[aFilter](#), [AnalogFilter](#), [funcFilter](#), [rootFilter](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 01 November 2015 08:12. © 2015.

aAmp(a,specs)

Generic analog amplifier

Function only- no corresponding object

Syntax

aAmp(a,specs)

a: Waveform - input signal

specs: scalar Record - record that sets specs for the amplifier

3 fields: gain in dB, upperClip, lowerClip

return: Waveform - output signal

Usage

Drop this in to model an idealized non-linear amplifier with saturation.

Location

menu: *no menu pick - to access use the Function and Object browser*

library: [sigLib](#)

Example

Notes

This implements a mathematically ideal saturating amplifier. Saturation is taken as a hard limit. It is normally hidden behind [analogAmp](#), providing the needed functionality, but can also be used standalone.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[aFilter](#), [aMixer](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](#). Last updated 20 October 2015 20:17. © 2015.

aFilter(a,specs)

Generic classical analog filter

Function only- no corresponding object

Syntax

```
function(a,b)
    a: shape type - description
    b: shape type - description
    return: type - description
```

Usage

Location

menu: *no menu pick - to access use the Function and Object browser*

library: funcLib

Example

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[aAmp](#), [aFilter](#), [aMixer](#), [polyFilter](#), [rootFilter](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 31 October 2015 14:43. © 2015.

aMixer(a,b,specs)

Generic analog mixer

Function only- no corresponding object

Syntax

function(a,b)

a: Waveform - input signal 1

b: Waveform - input signal 2

specs: scalar Record - specifies behavior

fields: cl (conversion loss in dB), AXi (A signal isolation in dB), BXi (B signal isolation in dB)

return: Waveform - the result of mixing the two input signals and adding in leakage.

Usage

Location

menu: *no menu pick - to access use the Function and Object browser*

library: [sigLib](#)

Example

Notes

This implements a mathematically ideal mixer with the addition of leakage signals. It is normally hidden behind [upConvert](#) and [downConvert](#), providing the needed functionality, but can also be used standalone.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[aAmp](#), [aFilter](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:19. © 2015.

aRectifier(a,specs)

Generic analog rectifier

Function only- no corresponding object

Syntax

aRectifier(a,specs)
a: shape type - description
b: shape type - description
return: type - description

Usage

Location

menu: *no menu pick - to access use the Function and Object browser*

library: [sigLib](#)

Example

Notes

Reference

Dependencies

Supported On

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](#). Last updated 20 October 2015 20:20. © 2015.

butterPoly(specs)

Generates Butterworth polynomials

Function only- no corresponding object

Syntax

butterPoly(specs)
 specs: scalar record with fields:
 function: enum (HP,BP,LP,BS)
 Fl: scalar Real - lower cutoff frequency
 Fu: scalar Real - upper cutoff frequency
 return: Text - polynomial in S

Usage

Given frequency specs this calculates a polynomial that is intended to be placed in a formula to generate a frequency domain transfer function. That way signals can be passed through this formula at will.

Location

menu: *no menu pick - to access use the Function and Object browser*

library: [sigLib](#)

Example

Given LP function, Fu=100, 3rd order, the resulting polynomial transfer function in S is:

$$(-1*(-49.999999999999999, 86.60254037844388)*-1*(-100, 6.123031769111886E-015)*-1*(-49.999999999999999, -86.60254037844388))/((S-(-49.999999999999999, 86.60254037844388))*(S-(-100, 6.123031769111886E-015))*(S-(-49.999999999999999, -86.60254037844388)))$$

Notes

Reference

Dependencies

Supported On

See Also

[chebyPoly](#)

chebyPoly(specs)

Generates Chebyshev polynomials

Function only- no corresponding object

Syntax

chebyPoly(specs)

specs: scalar record with fields:

function: enum (HP,BP,LP,BS)

Fl: scalar Real - lower cutoff frequency

Fu: scalar Real - upper cutoff frequency

Pr: scalar Real - passband ripple

return: Text - polynomial in S

Usage

Given frequency specs this calculates a polynomial that is intended to be placed in a formula to generate a frequency domain transfer function. That way signals can be passed through this formula at will.

Location

menu: *no menu pick - to access use the Function and Object browser*

library: [sigLib](#)

Example

Given LP function, Fu=100, 35th order, , 1dB ripple, the resulting polynomial transfer function in S is:

```
(-1*(-8.945836220019011, 99.01071120033895)*-1*(-23.42050328179965, 61.19198477210937)*-1*(-28.94933412356129,
6.374444838711282E-015)*-1*(-23.42050328179965, -61.19198477210936)*-1*(-8.945836220019015, -99.01071120033895)))/((S-(-
8.945836220019011, 99.01071120033895))*(S-(-23.42050328179965, 61.19198477210937))*(S-(-28.94933412356129, 6.374444838711282E-
015))*(S-(-23.42050328179965, -61.19198477210936))*(S-(-8.945836220019015, -99.01071120033895)))
```

Notes

Reference

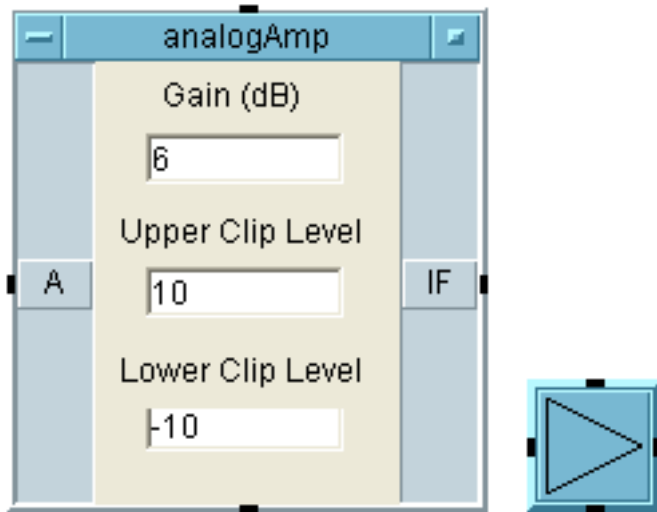
Dependencies

Supported On

See Also

analogAmp

A generic configurable nonlinear amplifier



Syntax

None- object only.

Usage

Use this to simulate an idealized saturating amplifier.

Location

menu: veeos ==> Signal Processing ==> Modules ==> analogAmp

library: [sigLib](#)

Example

See veeos examples

Notes

This implements a mathematically ideal saturating amplifier. Saturation is taken as a hard limit.

Reference

Dependencies

analogAmp

Underlying function is aAmp.

Supported On

VEE 7.0+, all platforms

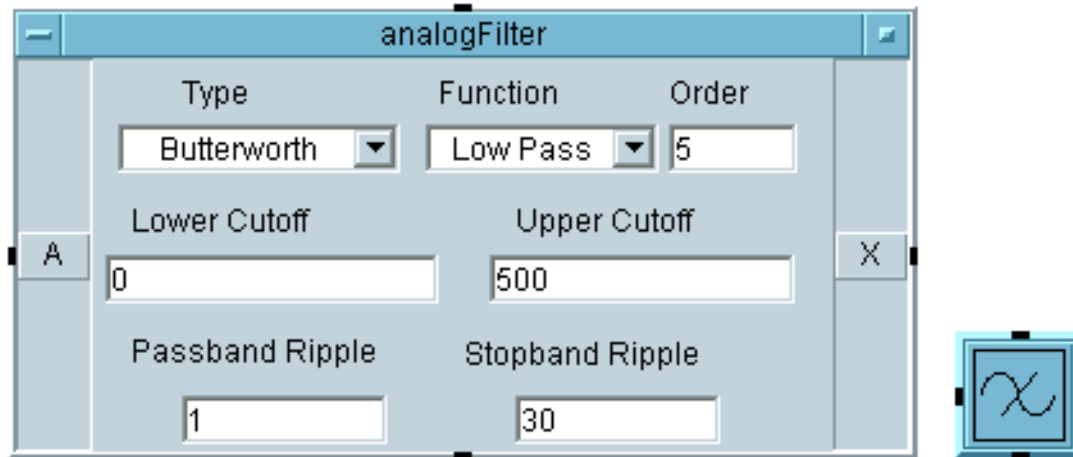
See Also

[analogFilter](#), [downConvert](#), [upConvert](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:20. © 2015.

analogFilter

Configurable analog classical filter.



Syntax

None- object only.

Usage

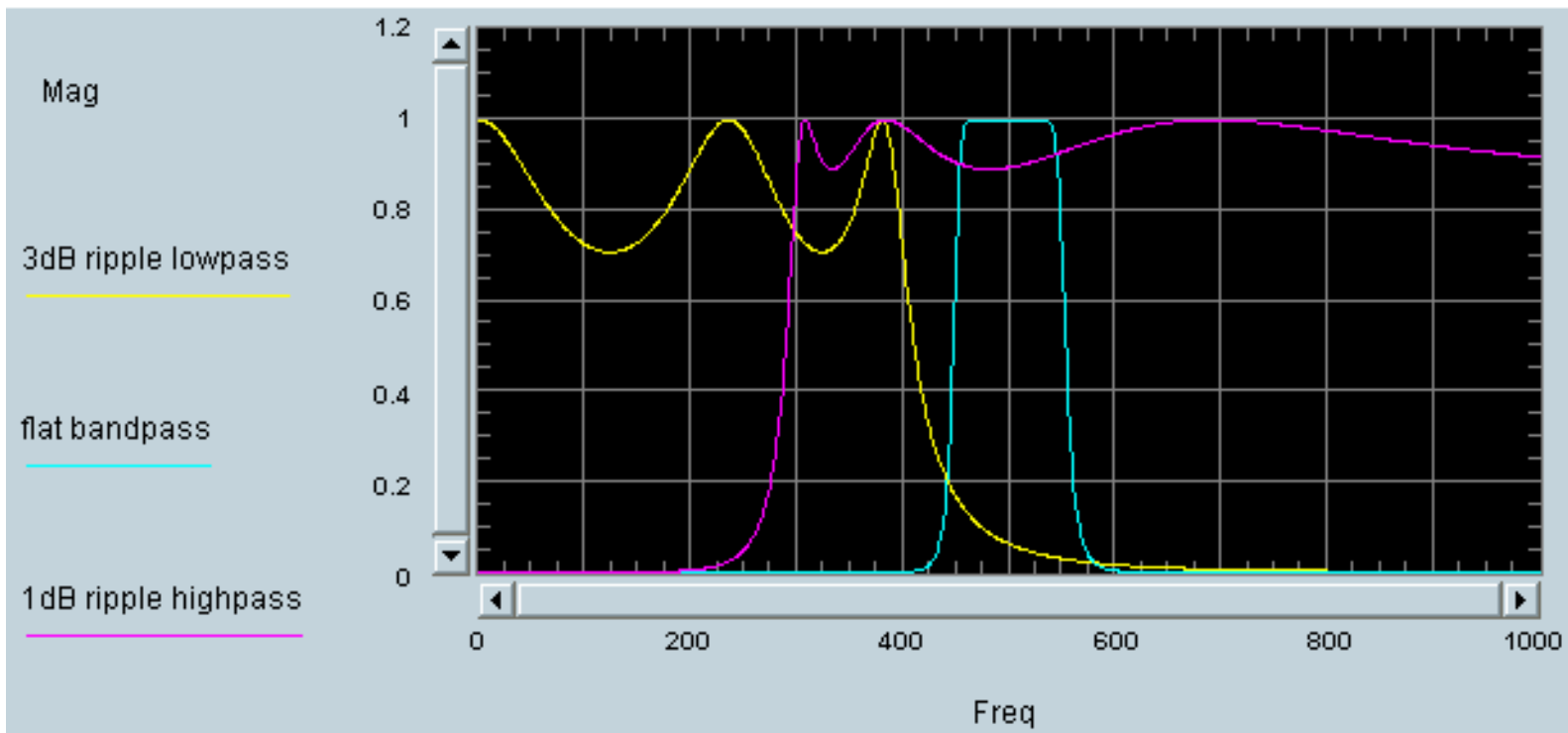
Drop this down in order to implement analog filtering on a signal. Filter type, order, cutoff and so on are configurable.

Location

menu: veeos ==> Signal Processing ==> Modules ==> analogFilter

library: [sigLib](#)

Example



Notes

This object is a wrapper around the `aFilter` function which does the actual work. The wrapper provides an easy way to make sure that the configuration inputs of `aFilter` are valid. In operation, the signal is converted to frequency domain and then multiplied by the transfer function defined by the chosen filter specifications.

Reference

The filters implemented include the classical polynomial definitions of Chebychev (equal ripple) and Butterworth (maximally flat) filters.

Dependencies

Uses `aFilter` which in turn requires `ButterPoly` and `ChebyPoly`

Supported On

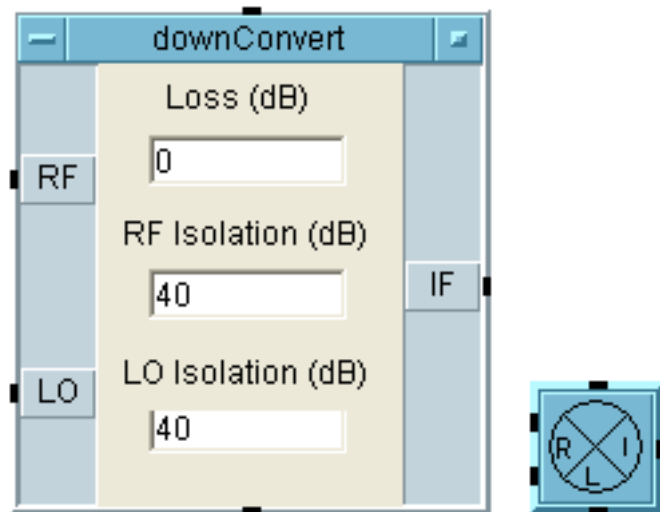
See Also

[analogAmp](#), [downConvert](#), [upConvert](#), [decayFilter](#), [polyFilter](#), [rootFilter](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 01 November 2015 08:12. © 2015.

downConvert

Mixer-based down-conversion object



Syntax

None- object only.

Usage

Use this to mix two signals together in a down-converter configuration, translating an RF frequency to a lower IF frequency by mixing with an LO. Note that RF-IF and LO-IF isolation may be specified.

Location

menu: veeos ==> Signal Processing ==> Modules ==> downConvert

library: [sigLib](#)

Example

Notes

In communications it is very common to place a baseband signal on carrier that is at much higher frequency than the baseband. This allows the broadband baseband signal to be treated as a narrowband signal, moves it to a convenient frequency for transmission and allows many other possible processing steps. Conversely it is common to reverse this up-conversion with a down-conversion to retrieve the original baseband signal. This object uses a mixer to mathematically model such frequency conversion. Note that while leakage is allowed within the model, it is not specifically accounted for as higher-level mixing products.

Reference

Any basic communications text will talk about this subject.

Dependencies

Uses aMixer to simulate the actual mixing action.

Supported On

VEE 7.0+, all platforms

See Also

[analogAmp](#), [analogFilter](#), [upConvert](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:23. © 2015.

negBridge

An ideal negative polarity full-wave rectifier.



Syntax

none - object only

Usage

To simulate a full-wave rectifier.

Location

menu: veeos ==> Signal processing ==> Devices ==> negBridge

library: [sigLib](#)

Example

Notes

In a real world circuit, a full-wave rectifier is typically produced by using a transformer and a bridge configuration of 4 diodes. This effectively allows negative going signals through untouched and inverts positive-going signals so that all outputs are the same polarity. When used for power supplies or signal recovery, the full-wave rectifier is more complex than a half wave rectifier but at the same time is far more efficient since all of the signal passes through.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[dcBlock](#), [posBridge](#), [negRectifier](#), [posRectifier](#)

negRectifier

An ideal positive negative half-wave rectifier.



Syntax

none - object only

Usage

To simulate a half-wave rectifier.

Location

menu: veeos ==> Signal processing ==> Devices ==> negRectifier

library: [sigLib](#)

Example

Notes

In a real world circuit, a half-wave rectifier is typically produced by using a single diode. This effectively allows one polarity of signal to pass, but block the opposite polarity. When used for power supplies or signal recovery, the half-wave rectifier is very simple and cheap to implement, but is less efficient than a full-wave rectifier since half of the signal is blocked.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[dcBlock](#), [negBridge](#), [posBridge](#), [posRectifier](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:26. © 2015.

posBridge

An ideal positive polarity full-wave rectifier.



Syntax

None - object only

Usage

To simulate a full-wave rectifier.

Location

menu: veeos ==> Signal processing ==> Devices ==> posBridge

library: [sigLib](#)

Example

Notes

In a real world circuit, a full-wave rectifier is typically produced by using a transformer and a bridge configuration of 4 diodes. This effectively allows positive going signals through untouched and inverts negative-going signals so that all outputs are the same polarity. When used for power supplies or signal recovery, the full-wave rectifier is more complex than a half wave rectifier but at the same time is far more efficient since all of the signal passes through.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[dcBlock](#), [negBridge](#), [negRectifier](#), [posRectifier](#)

posRectifier

An ideal positive polarity half-wave rectifier.



Syntax

none - object only

Usage

To simulate a half-wave rectifier.

Location

menu: veeos ==> Signal processing ==> Devices ==> negRectifier

library: [sigLib](#)

Example

Notes

In a real world circuit, a half-wave rectifier is typically produced by using a single diode. This effectively allows one polarity of signal to pass, but block the opposite polarity. When used for power supplies or signal recovery, the half-wave rectifier is very simple and cheap to implement, but is less efficient than a full-wave rectifier since half of the signal is blocked.

Reference

Dependencies

Supported On

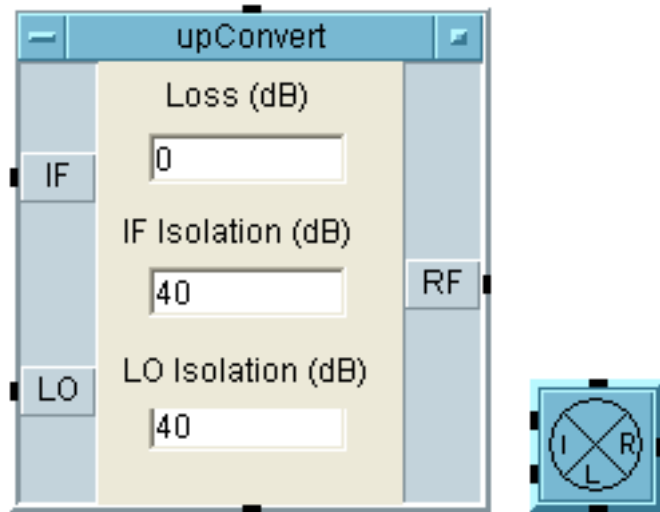
VEE 7.0+, all platforms

See Also

[dcBlock](#), [negBridge](#), [posBridge](#), [negRectifier](#)

upConvert

Mixer-based up-conversion object



Syntax

None- object only.

Usage

Use this to mix two signals together in a up-converter configuration, translating a baseband or IF frequency to a higher RF frequency by mixing with an LO. Note that RF-IF and LO-RF isolation may be specified.

Location

menu: veeos ==> Signal Processing ==> Modules ==> upConvert

library: [sigLib](#)

Example

Notes

In communications it is very common to place a baseband signal on carrier that is at much higher frequency than the baseband. This allows the broadband baseband signal to be treated as a narrowband signal, moves it to a convenient frequency for transmission and allows many other possible processing steps. Conversely it is common to reverse this up-conversion with a down-conversion to retrieve the original baseband signal. This object uses a mixer to mathematically model such frequency conversion. Note that while leakage is allowed within the model, it is not specifically accounted for as higher-level mixing products.

Reference

Any basic communications text will talk about this subject.

Dependencies

Uses aMixer to simulate the actual mixing action.

Supported On

VEE 7.0+, all platforms

See Also

[analogAmp](#), [analogFilter](#), [downConvert](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:33. © 2015.

sysLib

Functions related to the operating system and the general computer environment.

Functions

- [runInShell](#) - Runs specified code in a Windows shell
- [unzip](#) - Uses standard ZIP to unpack zip bundles.
- [uudecode](#) - Unpacks a coded ASCII file into the original format of the enclosed file.
- [uuencode](#) - Unpacks a coded ASCII file into the original format of the enclosed file.
- [winVersion](#) - Find the version of Windows which is currently running.
- [zip](#) - Uses standard ZIP to bundle whatever source files are specified.

Internal Content

These are used within the library but can often be useful directly accessed from without.

- private functions
 - version - returns revision date of this library
- variables
 - none

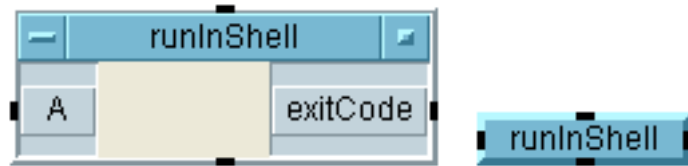
Standalone Objects

Notes

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:29. © 2015.

runInShell(a)

Runs specified code in a Windows shell



Syntax

`runInShell(a)`

a: scalar Text - full command line to execute

return: int - exit code

Usage

Input needs to be the full command as you would enter it directly in a windows CMD shell. If spaces are included in path or file names, they need to be quoted.

Location

menu: veeos ==> system ==> runInShell

library: [sysLib](#)

Example

`runInShell("copy C:\\temp\\somefile.txt C:\\temp\\newname.txt")` uses the file system executable "copy" to copy the file somefile.txt to a new file newname.txt

Notes

This function brings up a CMD shell, executes the given command line, and then exits (cmd /C option). The present working directory is used as the "working directory", so absolute paths are generally needed. This function provides a simple way to run programs from within a formula or other expression. If more flexibility is needed then use the underlying native VEE "Execute Program" object

Reference

cmd options can be found via "help cmd" at a command prompt or can be found at [Microsoft](#).

Supported On

VEE 7.0+, Windows XP+

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:28. © 2015.

unzip(zipfile,dest)

Uses standard ZIP to unpack zip bundles.



Syntax

unzip(zipfile,dest)

zipfile: scalar Text - zip file that is to be unpacked

dest: scalar Text - directory into which the zip file should be unpacked

return: none

Usage

This uses the standard [ZIP specification](#) to unpack a zip file into the contained set of files and directories into a single package. Zip is a long time industry standard and is universally available.

Location

menu: veeos ==> system ==> unzip

library: [sysLib](#)

Example

unzip("C:\temp\veebackups.zip" , "C:\temp\") unpacks all of veebackups.zip into the directory C:\temp.

Notes

This function uses the present working directory as a reference so absolute paths are generally needed. Note that as with any unpackaging it is possible for this to inadvertently overwrite existing files. Hence be careful what you specify.

Reference

- [ZIP specification](#)

Supported On

VEE 7.0+, Windows XP+

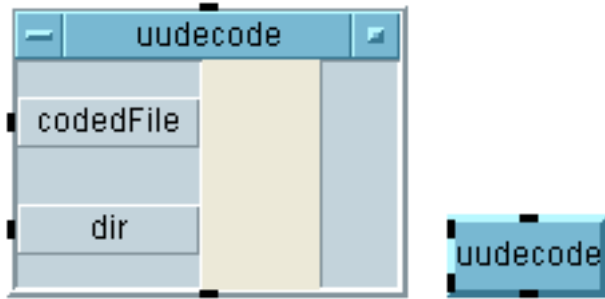
See Also

[zip](#), [uudecode](#), [uuencode](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:33. © 2015.

uudecode(codedFile,dir)

Unpacks a coded ASCII file into the original format of the enclosed file.



Syntax

uudecode(codedFile,dir)

codedFile: scalar Text - the uuencoded file to be decoded

dir: scalar Text - the directory in which to place the file encoded within codedFile

return: none

Usage

Uencode/uudecode have been used for many decades to enable binary files to be placed within ASCII documents such as usenet forums or emails. By placing binary files as ASCII one can bypass virtually all filtering and ensure that the file arrives intact.

Location

menu: veeos ==> system ==> uudecode

library: [sysLib](#)

Example

uudecode("C:\temp\mydata.uue" , "C:\temp") unpacks whatever files was encoded into mydata.uue into the C:\temp directory

Notes

Uencode/uudecode use a very widespread and long-stable standard. Do not use spaces or other punctuation in file names for uuencoding

Reference

- See [Wikipedia](#) for a discussion of the uencode standard

Supported On

uudecode

VEE 7.0+, Windows XP+

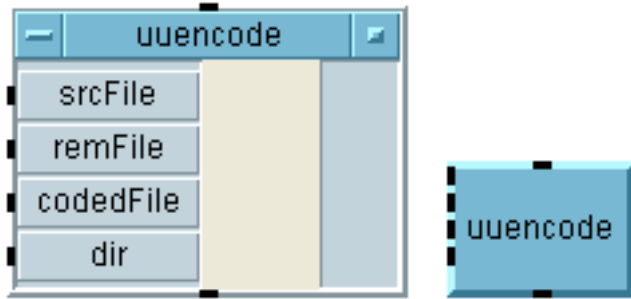
See Also

[zip](#), [unzip](#), [uuencode](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:33. © 2015.

uuencode(codedFile,dir)

Unpacks a coded ASCII file into the original format of the enclosed file.



Syntax

uuencode(srcFile, remFile, codedFile, dir)

srcFile: scalar Text - the file that is to be encoded

remFile: scalar Text - the name under which srcFile will be called when it is decoded

codedFile: scalar Text - the file into which the encoded srcFile is to be placed

dir: scalar Text - the working directory to be used

return: none

Usage

Uuencode/uudecode have been used for many decades to enable binary files to be placed within ASCII documents such as usenet forums or emails. By placing binary files as ASCII one can bypass virtually all filtering and ensure that the file arrives intact.

Location

menu: veeos ==> system ==> uuencode

library: [sysLib](#)

Example

uuencode("junk.dat", "dataFromStan.dat", "mydata.uue" , "C:\temp") cd's to the C:\temp directory and then encodes the file junk.dat into the package mydata.uue.

Notes

Uuencode/uudecode use a very widespread and long-stable standard. Do not use spaces or other punctuation in file names for uuencoding.

Reference

- See [Wikipedia](#) for a discussion of the uuencode standard

Supported On

VEE 7.0+, Windows XP+

See Also

[zip](#), [unzip](#), [uudecode](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:34. © 2015.

winVersion()

Find the version of Windows which is currently running.



Syntax

winVersion()

return: record contains Windows version information including description, version name and service pack.

Usage

This function queries the running Windows kernel for version information and then decodes it based on Windows standards.

Location

menu: Veeos ==> System ==> winVersion

library: [sysLib](#)

Example

Running on XP/32 returns:

version	Version 5.1 Build 2600
windowsName	Windows XP
servicePack	Service Pack 3

Notes

This is an updated version of the code written and contributed to VRF by Georg Nied in 2003. In some cases there is an ambiguity since kernels are sometimes shared between versions. For instance, Windows XP 64 shares a kernel with Windows 2003 so this kernel query cannot distinguish between the two.

Reference

- Windows version numbers are [detailed by Microsoft](#).

Supported On

VEE 7.0+. Windows 3.1+

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:35. © 2015.

zip(zipfile,source)

Uses standard ZIP to bundle whatever source files are specified.



Syntax

zip(zipfile,source)

zipfile: scalar Text - zip file that is to be created

source: scalar Text - list of file(s) that are to be added to the zip file

return: none

Usage

This uses the standard [ZIP specification](#) to bundle an arbitrary set of files and directories into a single package. Zip is a long time industry standard and is universally available.

Location

menu: veeos ==> system ==> zip

library: [sysLib](#)

Example

zip("C:\temp\veebackups.zip" , "C:\temp*.bak") grabs all of the file in C:\temp that have a bak extension and bundles them into the file veebackups.zip.

Notes

This function uses the present working directory as a reference so absolute paths are generally needed.

Reference

- [ZIP specification](#)

Supported On

VEE 7.0+, Windows XP+

See Also

[unzip](#), [uudecode](#), [uuencode](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:35. © 2015.

veeosLib

Functions that didn't fit elsewhere or that help manage the library.

Functions

- [time](#) - Converts timestamp to time of the day.
- [week](#) - Converts timestamp to week of the year.
- [downloadLatestLibrary](#) - Downloads the latest version the VEEOS library.

Internal Content

These are used within the library but can often be useful directly accessed from without.

- private functions
 - version - returns revision date of this library
- variables
 - none*

Standalone Objects

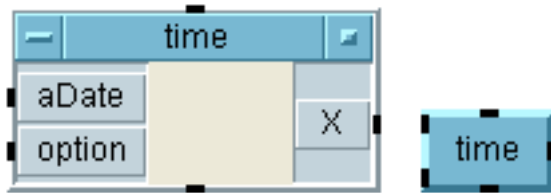
- Help - brings up a local copy of the online help in a popup window

Notes

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:34. © 2015.

time(aDate,option)

Converts timestamp to time of the day.



Syntax

```
time(aDate, option)
  aDate: scalar Timestamp
  option: scalar Int - 0=12hr, 1=24hr, 2=seconds
  return: Text
```

Usage

Separates out the time alone from a given timestamp.

Location

menu: veeos ==> misc ==> time

library: [veeosLib](#)

Example

time(now(),0) returns the current time of day in 12 hour format

Notes

VEE has built-in functions for day, month, year, but not time. This helps complete the set of Time/Date functions.

Reference

Supported On

VEE 7.0+, All platforms.

See Also

[week](#)

week(aDate)

Converts timestamp to week of the year.



Syntax

```
week(aDate)
    aDate: scalar Timestamp
    return: Int
```

Usage

Determines the week of the year given specified timestamp.

Location

menu: veeos ==> misc ==> week

library: [veeosLib](#)

Example

week(now()) returns the current week of the year.

Notes

VEE has built-in functions for day, month, year, but not time. This helps complete the set of Time/Date functions. Week in this case is defined by ISO-8601. Original version posted to VRF by Delmar Fryd in November 2012.

Reference

- [ISO-8601](#) is an international standard for time and date representation.

Supported On

VEE 7.0+, All platforms.

See Also

week

[time](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:34. © 2015.

downloadLatestLibrary()

Downloads the latest version the VEEOS library.



Syntax

```
downloadLatestLibrary()
    no inputs
    return: byte array - zip file
```

Usage

Finds and downloads the latest version of the VEEOS library. This is a zip file so save it wherever you like using To File object with BINARY write.

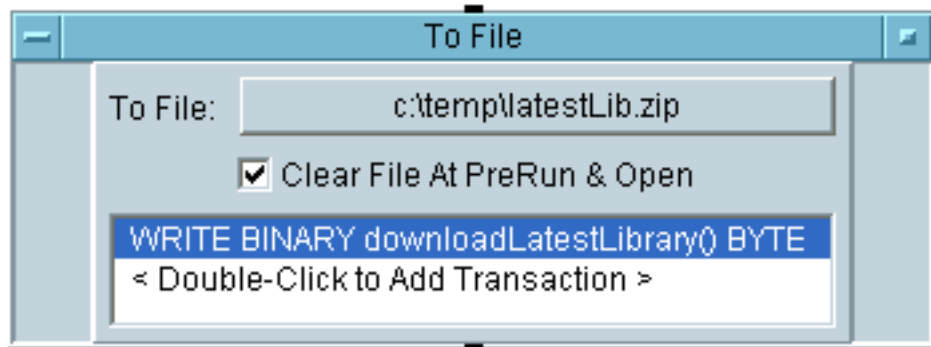
Location

menu: veeos ==> System ==> downloadLatestLibrary

library: veeosLib

Example

To download and save the latest version of the library:



Notes

Since the (mandatory) veeosInit object already determines the latest version available, all that's left is to wander by the web and download the latest version. This uses a loop so will be slow but should also be quite robust.

Reference

Dependencies

network access

Supported On

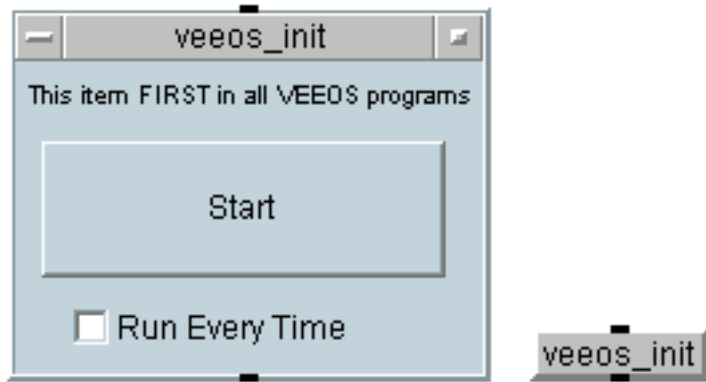
VEE 7.0+, all platforms

See Also

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:24. © 2015.

veeosInit

Initializes the loading of veeos



Syntax

- veeosInit has no underlying function

Usage

Place this object at the beginning of any program that uses the veeos library. Set "Run Every Time" to cause this object to execute each time your program is run.

Location

menu: veeos ==> Veeos Init

Notes

Since the veeosInit object is to be included in all VEE programs that use the veeos library, it is designed to be as small as possible and to have the absolute minimum hard-coded functionality. In essence it does nothing other than to find and import a library called [bootstrapLib](#) and then run a function from that library called init. It also only runs the very first time a program is run and then takes no action later, hence minimizing overhead.

In searching for bootstrapLib itself, veeosInit searches the below locations in series and stopping as soon as the first is found. This order is chosen for standard precedence: the most specific definition takes precedence.

1. Environment variable - if it exists, the variable MY_BOOTSTRAPLIB is used to identify the library file to load.
2. Local directory - searched for bootstrapLib.vxe
3. Home directory - searched for bootstrapLib.vxe
4. VEE Install directory- searched for bootstrapLib.vxe

Hence Hence the default behavior is to use the bootstrapLib that comes with veeos, but any other alternative takes precedence.

NOTE that due to this behavior, veeosInit can initiate any functionality that the end user desires. All one needs to do is to create their own library and point to it. the only requirement is that there is a function called init() in this library. The end result is that although veeosInit is part of veeos it

can alternately be used for initiating the end user's customizations instead (or in addition).

By default, veeosInit runs the very first time that a program containing it executes. On subsequent runs veeosInit skips its internal functionality in order to minimize the time it takes to run. To override this behavior, the Run Every Time option can be selected. This causes veeosInit to execute all of its functionality at every program run. This can be useful when doing development work since it causes a fresh start of veeos configuration. Run Every Time is best left unchecked after development is done.

Reference

Dependencies

[bootstrapLib](#)

Supported On

VEE 7.0+, all platforms

See Also

[bootstrapLib](#)

© 2015. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 20 October 2015 20:34. © 2015.