

Reference Guide

The VEEOS library consists of importable objects and functions that can be utilized in your VEE programs. The functions are organized into various libraries that are automatically imported. The objects are accessed via the VEEOS menu picks. This guide provides details of the various functions and objects.

- [Libraries](#)
 - [Functions](#)
 - [Standalone Functions](#) (no matching object)
 - [Standalone Objects](#) (no matching function)
 - [VEEOS Error Codes](#)
-

Libraries

The end user will not generally interact with these directly.

- [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.
 - [scriptLib](#) - Enables support of scripting languages
 - [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.

NOTE that the actual function name adds a prefix ("o_") to the object names. For example the [polarResample](#) object has underlying function `o_polarResample()`

- [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
- [baseNtoDec](#) - Base conversion from any base to decimal.
- [brb](#) - Intentionally create a crash dialog (BRB)
- [buildField](#) - Creates a field with specified name and value.
- [checkStatus](#) - Queries the script server to determine if it is running and ready for use.
- [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
- [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.
- [decToBaseN](#) - Base conversion from decimal to any base.
- [dif](#) - calculates difference between successive data points
- [disconnectServer](#) - Closes the socket connection to the script server. The server remains running.
- [downloadLatestLibrary](#) - Downloads the latest version the VEEOS library.
- [dtmfPulse](#) - Creates a DTMF pulse consisting of two frequencies that identify the given character.
- [dtmfReceiver](#) - Converts a DTMF pulse or (pulse train) into the corresponding character(s)
- [dtmfSigGen](#) - Generates a pulse train that represents the DTMF signal corresponding to a given set of characters
- [dutyCycle](#) - Determines the duty cycle of an arbitrary signal
- [envelope](#) - Constructs an envelope for arbitrary signals.
- [enumToList](#) - Converts an enum into a text array.
- [eval](#) - Evaluates the given function at the given argument.
- [executeScript](#) - Instructs the server to run the previously specified script.
- [fallTime](#) - Determines the fall time of an arbitrary signal
- [findFile](#) - Determines the location(s) of a file within a given directory tree.
- [frequency](#) - Determines the dominant frequency of an arbitrary signal.
- [fromClipboard](#) - Reads text from the system clipboard.
- [funcFilter](#) - Implements an analog filter with transfer response specified by an arbitrary function.
- [funcGen](#) - A general purpose function generator.
- [getResult](#) - Queries the server for the requested output data from the last script

execution.

- [help](#) - brings up a local copy of the online help in a popup window
- [hexToInt](#) - Interprets input as hex value and outputs it as an integer
- [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
- [isPrime](#) - Determines if a number is prime.
- [listToEnum](#) - Converts a text array into an enum.
- [longLat](#) - returns the longitude and latitude for a given city
- [period](#) - Determines the period of an arbitrary signal.
- [PI](#) - Evaluates the product of a series.
- [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
- [printText](#) - Prints text to a specified printer
- [psftp](#) - Automates an SFTP session.
- [riseTime](#) - Determines the rise time of an arbitrary signal
- [randomInt](#) - Returns a random integer from specified range.
- [randomReorder](#) - scrambles the order of a string or array.
- [rootFilter](#) - Implements an analog filter with transfer response specified by poles and zeroes.
- [runInShell](#) - runs specified command in a cmd shell.
- [screenShot](#) - Takes snapshot of screen
- [scriptConfig](#) - Assists the user in controlling various settings related to scripting.
- [scriptInit](#) - Readies scripting by setting up the needed environment and starting the script server as needed.
- [scriptPro](#) - Provides a simple wrapper for executing arbitrary scripts
- [sendData](#) - Uploads the specified data to the script server.
- [sendScript](#) - Uploads the specified script to the script server.
- [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
- [setShutdownTimeout](#) - Sets the idle timeout after which the script server shuts down.
- [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
- [SIGMA](#) - Evaluates the sum of a series.

- [slewFilter](#) - implements an analog filter whose response is based on a given slew rate.
- [startServer](#) - Starts the script server, enabling scripting to proceed.
- [stopServer](#) - Shuts down the script server.
- [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.
- [timeZone](#) - Determines the current timezone of the user's computer.
- [toClipboard](#) - Writes text to the system clipboard.
- [toCoord](#) - Creates Coordinates from input data.
- [toEngr](#) - Formats a number in standard engineering format.
- [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
- [unixNow](#) - Determines the current timestamp as understood by the unix world.
- [unzip](#) - Uses standard ZIP to unpack zip bundles.
- [utcNow](#) - Determines the current timestamp referenced to UTC time rather than local time.
- [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.
- [viewPicture](#) - Displays chosen picture in a photo viewer.
- [wait](#) - Pauses execution for specified time in seconds
- [week](#) - Converts timestamp to week of the year.
- [wget](#) - Downloads a file from the web.
- [which](#) - Determines the location of an executable in the system path
- [whois](#) - Determines the registration information for a given domain.
- [winVersion](#) - Find the version of Windows which is currently running.
- [yDay](#) - Converts timestamp to day of the year.
- [yWeek](#) - Converts timestamp to week of the year.
- [ywToDate](#) - Converts year and week to timestamps at start and end of the week.
- [zeroCrossings](#) - Determines the points at which a function crosses the X axis: the zero crossings.
- [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
 - [referenceGuide](#) - local viewable copy of the Reference Guide
 - [script](#) - Provides a simple wrapper for executing arbitrary scripts
 - [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.

NOTE that the actual function name adds a prefix ("o_") to the names shown. For example the [polarResample](#) object has underlying function `o_polarResample()`

- [startFreq](#) - Extracts the beginning frequency of a spectrum.
- [stopFreq](#) - Extracts the end frequency from a spectrum.
- [timeSpan](#) - Extracts the time span from a waveform.
- [aAmp](#) - generic analog amplifier
- [aFilter](#) - generic classical analog filter
- [aMixer](#) - generic analog mixer
- [aRectifier](#) - generic analog rectifier
- [arrayData](#) - Extracts the array of values from a waveform or spectrum.
- [butterPoly](#) - generates Butterworth polynomials
- [chebyPoly](#) - generates Chebychev polynomials
- [comma2point](#) - Replaces commas with points.
- [dimSizes](#) - Returns the size(s) of the dimensions for an array.
- [dtmfCode](#) - Generates the DTMF frequency pair that defines the given input character.
- [dtmfDecode](#) - Converts a pair of DTMF frequencies into the corresponding character
- [dtmfRcvPulse](#) - Converts a DTMF signal pulse into the corresponding character

- [numDims](#) - Returns the dimensionality of a value.
- [point2comma](#) - Replaces points with commas.
- [raiseError](#) - raises an error message that halts program execution.
- [specArray](#) - Extracts the array of values from a spectrum.
- [startFreq](#) - Extracts the beginning frequency of a spectrum.
- [stopFreq](#) - Extracts the end frequency from a spectrum.
- [timeSpan](#) - Extracts the time span from a waveform.
- [waveArray](#) - Extracts the array of values from a waveform.
- [Xcoord](#) - Returns the X value of a coordinate
- [Ycoord](#) - Returns the Y value of a coordinate
- [Zcoord](#) - Returns the Z value of a coordinate

VEEOS Error Codes

Code	Text	Location	Description
6600	"no valid VEEOS Bootstrap Library found"	veeosInit	As part of normal function veeosInit looks in multiple place for an initial "bootstrap" library. If nothing appropriate is found veeosInit aborts. Likely there is a veeos installation issue or a networking issue.
6601	"VEEOS Bootstrap Library: init() function not found"	veeosInit	Once an appropriate library is found, veeosInit executes the "init()" function inside this library. If you get this error check your bootstrap library and make sure that you have included a function called "init". If you have not written a custom bootstrap library then most likely your veeos install has a problem.
6602	"Sorry- arrays cannot be joined. More than one dimension does not match."	mathLib.o_arrayConcat	You have asked to concatenate arrays that cannot be concatenated. For example, a 2x3 and a 2x4 can be concatenated while a 2x3 and a 4x5 cannot. Check your inputs.
6603	"cannot merge arrays - at least one face must match"	mathLib.o_concat3D	You have asked to concatenate arrays that cannot be concatenated. Specifically at least one face of the array must match. Check your inputs.
6604	"Arrays not commensurate"	mathLib.o_concat2D	You have asked to concatenate arrays that cannot be concatenated. For example, a 2x3

	- cannot concat"		and a 2x4 can be concatenated while a 2x3 and a 4x5 cannot. Check your inputs.
6605	"Sorry- cannot process arrays with more than 2 dimensions."	mathLib.o_randomReorder	As written randomReorder only works for arrays with 2 or less dimensions. The algorithm used is not scalable easily to more than 2D. Consider writing your own routine if you need 3 or more dimensions.
6606	"invalid character for DTMF"	sigLib.o_dtmfCode	The standard DTMF keypad has 16 specific characters. Your string input contains characters other than these.
6607	"Invalid DTMF row freq"	sigLib.o_dtmfDecode	There are only 4 valid low ("row") frequencies for DTMF signaling: 697,770,852,941. Your input did not match any of these.
6608	"Invalid DTMF col freq"	sigLib.o_dtmfDecode	There are only 4 valid high ("column") frequencies for DTMF signaling: 1209,1336,1477,1633. Your input did not match any of these.
6609	"Query failed-check ID and city - this may be intermittent"	netLib.o_currentHumidity	The net service used is sometimes a little fragile. Your specified city may not be found or your ID is not valid. This may also occur intermittently due to networking issues.
6610	"Query failed-check ID and city - this may be intermittent"	netLib.o_currentHumidity	The net service used is sometimes a little fragile. Your specified city may not be found or your ID is not valid. This may also occur intermittently due to networking issues.
	"Query failed-		The net service used is sometimes

6611	check ID and city - this may be intermittent"	netLib.o_currentTemp	a little fragile. Your specified city may not be found or your ID is not valid. This may also occur intermittently due to networking issues.
6612	"Query failed-check ID and city - this may be intermittent"	netLib.o_longLat	The net service used is sometimes a little fragile. Your specified city may not be found or your ID is not valid. This may also occur intermittently due to networking issues.
6613	"not enough transitions found to determine accurate duty cycle"	sigLib.o_dutyCycle	Check your signal. There are not enough clear transitions in it to be able to properly estimate a duty cycle. For instance a single pulse is not sufficient.
6614	"sorry - fit failed - try different method"	sigLib.o_envelope	The various methods used for envelope detection span a range of efficiency and robustness. Your signal is not compatible with the method chosen so try another method. In general higher numbers are more robust, lower numbers are faster.
6615	"sorry - fit failed - try different method"	sigLib.o_envelope	The various methods used for envelope detection span a range of efficiency and robustness. Your signal is not compatible with the method chosen so try another method. In general higher numbers are more robust, lower numbers are faster.
6616	"not enough transitions found to determine period"	sigLib.o_period	Check your signal. There are not enough clear transitions in it to be able to properly estimate a period. For instance a single pulse is not sufficient.

6617	"invalid data type for PCL header"	sysLib.o_printText	The PCL header that you specified is not valid. Check your syntax and data type.
6618	"no "printer" field found"	sysLib.o_printText	The input record that you used to specify a PCL header does not contain the needed "printer" field
6619	"invalid lines per inch"	sysLib.o_printText	PCL printers only allow a specific set of values for lines per inch. Your specification is not within this set.
6620	"invalid characters per inch"	sysLib.o_printText	The specified PCL character spacing is not within acceptable range. Check your input.
6621	"invalid top margin"	sysLib.o_printText	The specified PCL top margin is not within acceptable range. Check your input.
6622	"invalid lines per page"	sysLib.o_printText	The specified PCL lines per page is not within acceptable range. Check your input.
6623	"Coordinates not found"	sysLib.o_screenShot	No specification was found for the screen area to be captured. Check your input.
6624	"DTMF sample rate must be at least 3300"	sigLib.o_dtmfSigGen	In order to meet Nyquist sampling criterion, DTMF signals must have sufficient sample rate. Make sure that you sample rate is at least 3300 samples per second.
6625	"invalid input to dtmfReceiver"	sigLib.o_dtmfReceiver	The dtmf receiver can accept a single waveform or a record with specific fields. Your input is neither
6626	"Error reading RC File. Invalid data"	bootstrapLib.init()	A GVAR line in the RC file contains an invalid type specification.

type:"

6627 "Server failed
to start within X sec" scriptLib.o_startServer()

startServer() launches the server process and waits until it responds to a checkStatus() query. If this does not occur within a preset time limit then an error is assumed.

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\(\)](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 03 August 2021 11:33.

codeLib

Programming Aids

Functions

- [brb](#) - *Intentionally create a crash dialog (BRB)*
- [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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 04 August 2021 15:27.

dataLib

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

Functions

- [addField](#) - Adds to an existing record a field with specified name and value.
- [baseNtoDec](#) - Base conversion from any base to decimal.
- [buildField](#) - Creates a field with specified name and value.
- [bytes](#) - Converts data to the bytes as written on disk.
- [createConstant](#) - Creates a non-editable importable data object.
- [decToBaseN](#) - Base conversion from decimal to any base.
- [enumToList](#) - Converts an enum into a text array.
- [hexToInt](#) - Interprets input as hex value and outputs it as an integer
- [listToEnum](#) - Converts a text array into an enum.
- [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.
- [toEngr](#) - Formats a number in standard engineering format.
- [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

- [comma2point](#) - Replaces commas with points.
- [dimSizes](#) - Returns the size(s) of the dimensions for an array.
- [numDims](#) - Returns the dimensionality of a value.
- [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

Notes

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 31 July 2021 08:00.

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 03 August 2021 11:33.

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
- [isPrime](#) - Determines if a number is prime.
- [PI](#) - Evaluates the product of a series.
- [randomInt](#) - Returns a random integer from specified range.
- [randomReorder](#) - scrambles the order of a string or array.
- [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
- [SIGMA](#) - Evaluates the sum of a series.
- [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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 03 August 2021 11:33.

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
- [psftp](#) - Automates an SFTP session.
- [wget](#) - Downloads a file from the web.
- [whois](#) - Determines the registration information for a given domain.

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 30 July 2021 14:35.

scriptLib

Enable Scripting and provide various utilities based on scripting.

Functions

- [checkStatus](#) - Queries the script server to determine if it is running and ready for use.
- [disconnectServer](#) - Closes the socket connection to the script server. The server remains running.
- [executeScript](#) - Instructs the server to run the previously specified script.
- [findFile](#) - Determines the location(s) of a file within a given directory tree.
- [getResult](#) - Queries the server for the requested output data from the last script execution.
- [scriptConfig](#) - Assists the user in controlling various settings related to scripting.
- [scriptInit](#) - Readies scripting by setting up the needed environment and starting the script server as needed.
- [scriptPro](#) - Provides a simple wrapper for executing arbitrary scripts
- [sendData](#) - Uploads the specified data to the script server.
- [sendScript](#) - Uploads the specified script to the script server.
- [setShutdownTimeout](#) - Sets the idle timeout after which the script server shuts down.
- [startServer](#) - Starts the script server, enabling scripting to proceed.
- [stopServer](#) - Shuts down the script server.
- [which](#) - Determines the location of an executable in the system path

Internal Content

These are used within the library but may also be useful directly accessed from without.

- private functions
 - [findEngine](#) - Determines the location(s) of a scripting engine on a local computer.
 - [formatScript](#) - An aid to formulate the final script to be uploaded.
 - version - returns revision date of this library
- variables
 - none

Standalone Objects

- [script](#) - Provides a simple wrapper for executing arbitrary scripts

Standalone Functions

Notes

Scripting is supported under Windows 7 and beyond.

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:35.

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.
- [dtmfPulse](#) - Creates a DTMF pulse consisting of two frequencies that identify the given character.
- [dtmfReceiver](#) - Converts a DTMF pulse or (pulse train) into the corresponding character(s)
- [dtmfSigGen](#) - Generates a pulse train that represents the DTMF signal corresponding to a given set of characters
- [dutyCycle](#) - Determines the duty cycle of an arbitrary signal
- [envelope](#) - Constructs an envelope for arbitrary signals.
- [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.
- [funcGen](#) - A general purpose function generator.
- [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.
- [slewFilter](#) - implements an analog filter whose response is based on a given slew rate.
- [zeroCrossings](#) - Determines the points at which a function crosses the X axis: the zero crossings.

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
- [dtmfCode](#) - Generates the DTMF frequency pair that defines the given input character.
- [dtmfDecode](#) - Converts a pair of DTMF frequencies into the corresponding character
- [dtmfRcvPulse](#) - Converts a DTMF signal pulse into the corresponding character
- 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

Standalone Functions

- [specArray](#) - Extracts the array of values from a spectrum.
- [startFreq](#) - Extracts the beginning frequency of a spectrum.
- [stopFreq](#) - Extracts the end frequency from a spectrum.
- [timeSpan](#) - Extracts the time span from a waveform.
- [waveArray](#) - Extracts the array of values from a waveform.

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.

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 30 July 2021 08:59.

sysLib

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

Functions

- [fromClipboard](#) - Reads text from the system clipboard.
- [printText](#) - Prints text to a specified printer
- [raiseError](#) - raises an error message that halts program execution.
- [runInShell](#) - Runs specified code in a Windows shell
- [screenShot](#) - Takes snapshot of screen
- [toClipboard](#) - Writes text to the system clipboard.
- [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.
- [viewPicture](#) - Displays chosen picture in a photo viewer.
- [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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 03 August 2021 11:33.

veeosLib

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

Functions

- [help](#) - brings up a local copy of the online help in a popup window
- [time](#) - Converts timestamp to time of the day.
- [timeZone](#) - Determines the current timezone of the user's computer.
- [unixNow](#) - Determines the current timestamp as understood by the unix world.
- [utcNow](#) - Determines the current timestamp referenced to UTC time rather than local time.
- [yDay](#) - Converts timestamp to day of the year.
- [yWeek](#) - Converts timestamp to week of the year.
- [ywToDate](#) - Converts year and week to timestamps at start and end of the week.

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

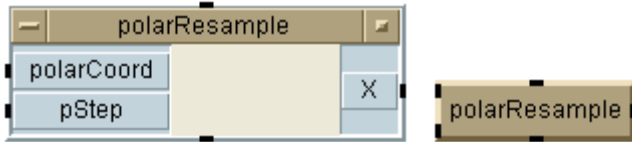
- [Help](#) - brings up a local copy of the online help in a popup window

Notes

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 31 July 2021 11:23.

polarResample(a)

Resamples polar data to ensure full 0-360 coverage.



Syntax

`o_polarResample(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

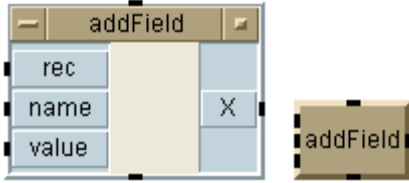
See Also

[polarRegress](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

addField(rec,name,value)

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



Syntax

`o_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

`o_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.

New for VEEOS 2: If the "rec" field is null or anything other than a record then a record is returned with just the one new field, That is, *addField* becomes [buildField](#) if the rec input is anything other than a record.

Reference

Supported On

VEE 7.0+, All platforms.

See Also

[buildField](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 05 August 2021 10:44.

arrayConcat(a,b)

Joins together arrays.



Syntax

`o_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

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

```
o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

arrayRev(a)

Reverses the order of any 1D array.



Syntax

`o_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

`o_arrayRev([1,2,3,4])` returns `[4,3,2,1]`

`o_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

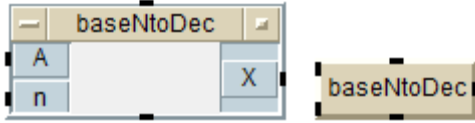
See Also

[arrayConcat](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

baseNtoDec(A,n)

Base conversion from any base to decimal.



Syntax

`o_baseNtoDec(a)`

A: scalar Text - value in base n as plain text

n: Int - base to be used

return: decimal (base 10) value

Usage

This implements a general base conversion between decimal and an arbitrary base. The numeric characters 0-9 are used up to base ten, then the alpha characters A-Z up to base 37 and then beyond that are ASCII characters.

Location

menu: veeos ==> Data ==> baseNtoDec

library: [dataLib](#)

Example

```
o_baseNtoDec("A36",27) returns 7377
```

```
o_baseNtoDec("A36",16) returns 2614
```

Notes

This is implemented without loops so is very fast. Only positive integer values are supported.

Reference

Dependencies

Supported On

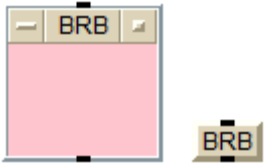
VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

brb()

Intentionally create a crash dialog (BRB)



Syntax

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

Usage

Use this to simulate a VEE crash by displaying the crash dialog (BRB).

Location

menu: veeos ==> Coding ==> BRB

library: [codeLib](#)

Example

Notes

Reference

Dependencies

Supported On

VEE 7.0+, Windows XP+

See Also

buildField(name,value)

Creates a field with specified name and value.



Syntax

o_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

o_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.

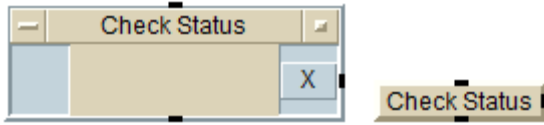
See Also

[addField](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 05 August 2021 10:44.

checkStatus()

Queries the script server to determine if it is running and ready for use.



Syntax

`o_checkStatus()`

no inputs

return: Text - "OK" means that the server is up and ready. Any other result is an error.

Usage

Use this to determine if the server is up and ready.

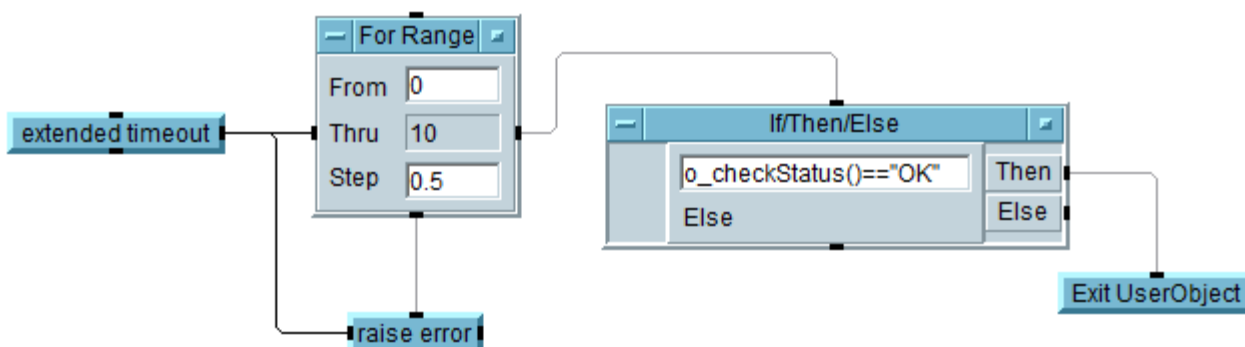
Location

menu: veeos ==> Scripting ==> Advanced ==> checkStatus

library: [scriptLib](#)

Example

The typical usage of checkStatus is a polling loop that waits for the server to reply with "OK".



Notes

This object never actually needs to be run, but it is advisable to check the server status before attempting to execute any script

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

See Also

[checkStatus](#), [disconnectServer](#), [executeScript](#), [getResult](#), [sendData](#), [sendScript](#), [setShutdownTimeout](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

colMax(a)

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



Syntax

o_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

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

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 26 July 2021 09:41.

colMean(a)

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



Syntax

`o_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

```
o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

colMedian(a)

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



Syntax

o_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

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

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

colMin(a)

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



Syntax

o_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

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

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

colMode(a)

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



Syntax

`o_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

```
o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

colRMS(a)

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



Syntax

o_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

```
o_colRMS([ 4 4 5 3
           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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 26 July 2021 09:41.

colSDev(a)

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



Syntax

o_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

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

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

colVari(a)

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



Syntax

`o_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

```
o_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

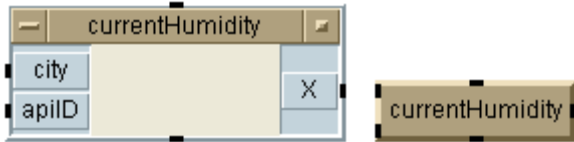
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

currentHumidity(city,apiID)

Returns the current humidity for a given city



Syntax

o_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 ==> Network ==> currentHumidity

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

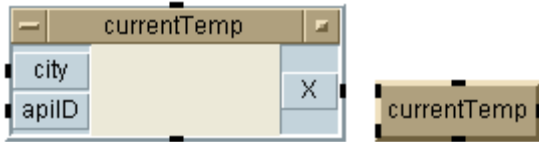
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 30 July 2021 10:36.

currentTemp(city,apiID)

Returns the current temperature for a given city



Syntax

o_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 ==> Network ==> currentTemp

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](#), [currentTemp](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 30 July 2021 10:36.

dcBlock(a)

Removes the DC component of a signal.



Syntax

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

decayFilter(a,decayRate)

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



Syntax

`o_decayFilter(a,decayRate)`

a: waveform - input waveform

decayRate: numeric - decay rate in percentage per sample

return: waveform- input transformed by the decay-based filter

Usage

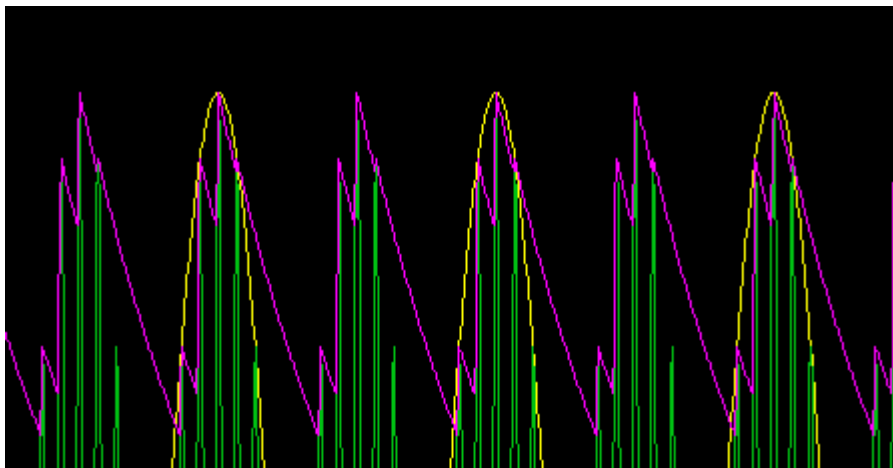
Typical usage is to model a diode-RC detector, but can be used in any case where the output signal is a decayed version of the input signal.

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

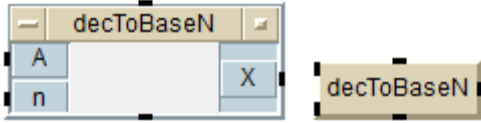
See Also

[analogFilter](#), [envelope](#), [slewFilter](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 29 July 2021 08:02.

decToBaseN(A,n)

Base conversion from decimal to any base.



Syntax

o_decToBaseN(a)

A: Int - value in base 10

n: Int - base to be used

return: an ascii string representing the value in base n

Usage

This implements a general base conversion between decimal and an arbitrary base. The numeric characters 0-9 are used up to base ten, then the alpha characters A-Z up to base 37 and then beyond that are ASCII characters.

Location

menu: veeos ==> Data ==> decToBaseN

library: [dataLib](#)

Example

```
o_baseNtoDec(7377,27) returns A36
```

```
o_baseNtoDec(2614,16) returns A36
```

Notes

This is implemented without loops so is very fast. Only positive integer values are supported.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

dif(a)

Calculates difference between successive data points.



Syntax

`o_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

`o_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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

disconnectServer()

Closes the socket connection to the script server. The server remains running.



Syntax

`o_disconnectServer()`

no inputs

return: Text - "OK" means that the server has received this directive. Any other result is an error.

Usage

When the server is no longer needed it can be useful to disconnect from it.

Location

menu: veeos ==> Scripting ==> Advanced ==> Disconnect Server

library: [scriptLib](#)

Example

Notes

This object never actually needs to be run, but may be useful.

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

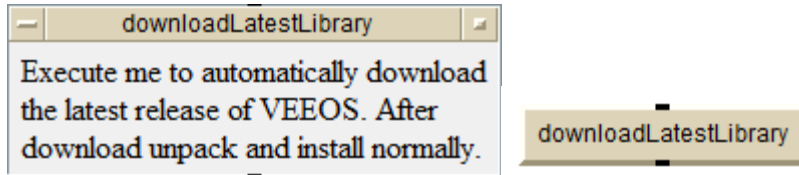
See Also

[checkStatus](#), [disconnectServer](#), [executeScript](#), [getResult](#), [sendData](#), [sendScript](#), [setShutdownTimeout](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

downloadLatestLibrary()

Downloads the latest version the VEEOS library.



Syntax

o_downloadLatestLibrary()

no inputs

return: none

Usage

Finds and downloads the latest version of the VEEOS library. The file is saved to disk automatically.

Location

menu: veeos ==> System ==> downloadLatestLibrary

library: [veeosLib](#)

Example

Notes

Once the release is downloaded, the user is notified of the location of the download.

Reference

Dependencies

network access

Supported On

VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 05 August 2021 15:50.

dtmfPulse(char,timeSpan,numPts)

Creates a DTMF pulse consisting of two frequencies that identify the given character.



Syntax

o_dtmfPulse(char,timeSpan,numPts)

char: scalar character string - character to be coded as a DTMF pulse

timeSpan: scalar real64 - duration of the desired DTMF pulse in seconds

numPts: scalar integer - number of time samples in the pulse

Usage

Use this to create a DTMF pulse for simulation within VEE or for real-world work. The output is a VEE waveform that can be transferred almost directly to an AWG or even to a DIO module.

Location

menu: veeos ==> signal processing ==> modules ==> dtmfPulse

library: sigLib

Example

Notes

In a real-world situation noise and distortion may well be present. This function generates pure tones, so anything else needs to be added later. If more than a single pulse (i.e. character) is needed, it is recommended to use [dtmfSigGen\(\)](#).

Reference

[DTMF is a very simple standard](#) consisting of two audio frequencies and developed to

enable simple data signaling over telephone lines.

Dependencies

Supported On

VEE 7.0+, All platforms.

See Also

[dtmfCode](#), [dtmfDecode](#), [dtmfRcvPulse](#), [dtmfReceiver](#), [dtmfSigGen](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

dtmfReceiver(dtmfPulse)

Converts a DTMF pulse or (pulse train) into the corresponding character(s)



Syntax

`o_dtmfReceiver(dtmfPulse)`

dtmfPulse: waveform - any DTMF-coded signal pulse

OR

dtmfPulse: scalar record that defines a pulse train. Three fields:

signal: waveform - the dtmf signal

pulseLen: real64 scalar - length of the pulses in seconds

nullLen: real64 scalar - length of the pulse train nulls in seconds

return: string - character(s) that correspond to the given pulse.

Usage

This function takes a general two-tone DTMF pulse train with or without noise and other signals and derives a best estimate of the intended encoded characters. If the input is a single waveform then it is assumed that a single pulse is present and a single character is derived. If the input is a record with the requisite three fields then a pulse train is processed and converted to a multiple character string.

Location

menu: veeos ==> signal processing ==> modules ==> dtmfReceiver

library: sigLib

Example

Notes

This function implements a DTMF receiver using a tuned narrow-band receiver and as such is highly noise resistant.

Reference

[DTMF is a very simple standard](#) consisting of two audio frequencies and developed to enable simple data signaling over telephone lines.

Dependencies

o_dtmfRcvPulse()

Supported On

VEE 7.0+, All platforms.

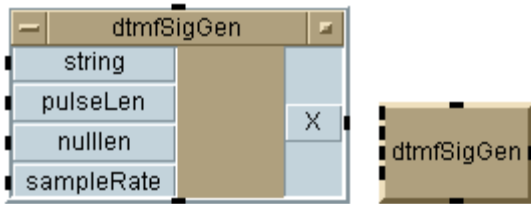
See Also

[dtmfCode](#), [dtmfDecode](#), [dtmfPulse](#), [dtmfRcvPulse](#), [dtmfSigGen](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

dtmfSigGen(string, pulseLen, NullLen, sampleRate)

Generates a pulse train that represents the DTMF signal corresponding to a given set of characters



Syntax

`o_dtmfSigGen(string, pulseLen, nullen, sampleRate)`

string: scalar string - characters to be coded into DTMF pulse train

pulseLen: scalar real64 - duration of the desired DTMF pulse in seconds

nullen: scalar real64 - duration of the nulls between pulses in seconds

sampleRate: scalar real64 - sample rate of the desired signal in samples per second

Return: waveform - the DTMF signal

Usage

This function is used to generate a DTMF pulse train from an arbitrary set of characters.

Location

menu: veeos ==> signal processing ==> modules ==> dtmfSigGen

library: sigLib

Example

Notes

In a real-world situation noise and distortion may well be present. This function generates pure tones, so anything else needs to be added later. If only a single pulse is needed, it is recommended to use [dtmfPulse\(\)](#).

Reference

[DTMF is a very simple standard](#) consisting of two audio frequencies and developed to enable simple data signaling over telephone lines.

Dependencies

o_dtmfPulse()

Supported On

VEE 7.0+, All platforms.

See Also

[dtmfCode](#), [dtmfDecode](#), [dtmfPulse](#), [dtmfRcvPulse](#), [dtmfReceiver](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

dutyCycle(a)

Determines the duty cycle of an arbitrary signal.



Syntax

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

envelope(a,opt)

Constructs an envelope for arbitrary signals.



Syntax

`o_envelope(a,opt)`

a: shape type - description

opt: scalar int- options: method 1: **0** for maxima, **1** for minima

method 2: **2** for maxima, **3** for minima

method 3: **4** for maxima, **5** for minima

return: type - coordinates of the envelope

Usage

The envelope of a signal is a line tracing the extreme points of the signal. Typically this represents a modulation signal on a carrier.

NOTE: use Method 1 for clean signals, Method 2 for noisy signals, Method 3 when 1 and 2 fail

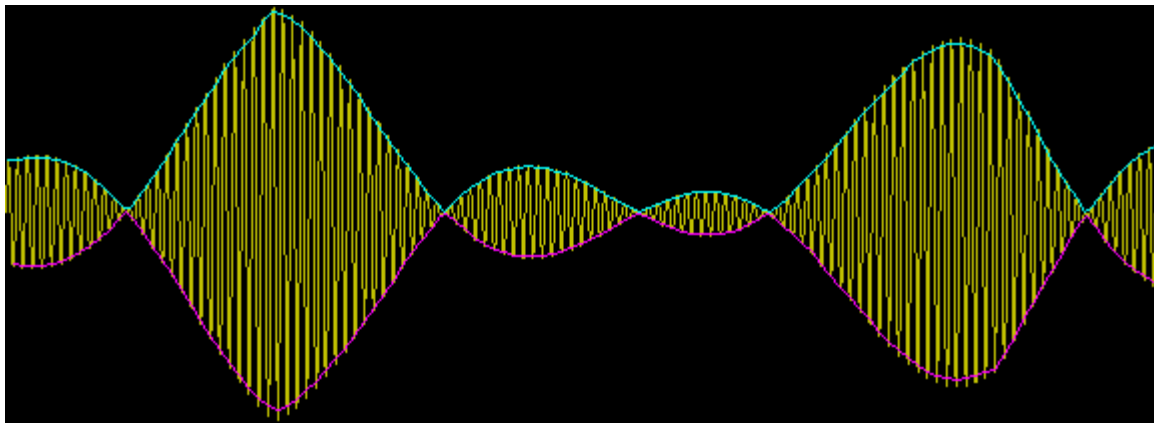
Location

menu: veeos ==> Signal Processing ==> envelope

library: sigLib

Example

See veeos examples . Typical signal and envelope are shown here:



`o_envelope(A,3)` returns minima using Method 2
`o_envelope(A,0)` returns maxima using Method 1

Notes

In a typical communication signal, a high frequency (carrier) is modulated by a low frequency (baseband) signal. The baseband signal contains the information to be sent and the carrier is chosen for transmission characteristics. Once modulated the baseband signal itself no longer exists, but rather it is represented by the fluctuations in the carrier. If specific assumptions are made for the modulated signal, then the baseband can be determined by basic signal processing. However for more complex signals- say a chirped carrier or freq hopping carrier- without specific information reconstructing the baseband can be very difficult. This function assumes nothing about the signal (except that it has a modulation) but rather looks directly at the time domain signal and ferrets out local extrema as best as it can.

Two methods are available in this function, selected by option value

Method 1 uses derivatives so is fragile when it comes to noise but also gives a very good estimate of correct peaks.

Method 2 eliminates derivatives so is more robust but still makes good estimate of correct peaks.

Method 3 is brute force and is reasonably noise-tolerant but makes only minimal effort to estimate peaks.

Another alternative that may give better results for some signals is [decayFilter\(\)](#) which has its own issues but will not fail and is what is commonly used in the real world.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

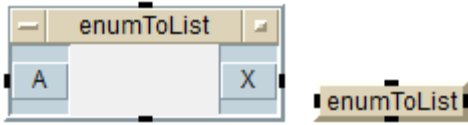
See Also

[decayFilter](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

enumToList(a)

Converts a text array into an enum.



Syntax

o_enumToList(a)

a: enum- value to be converted

return: Text array with values matching the enum

Usage

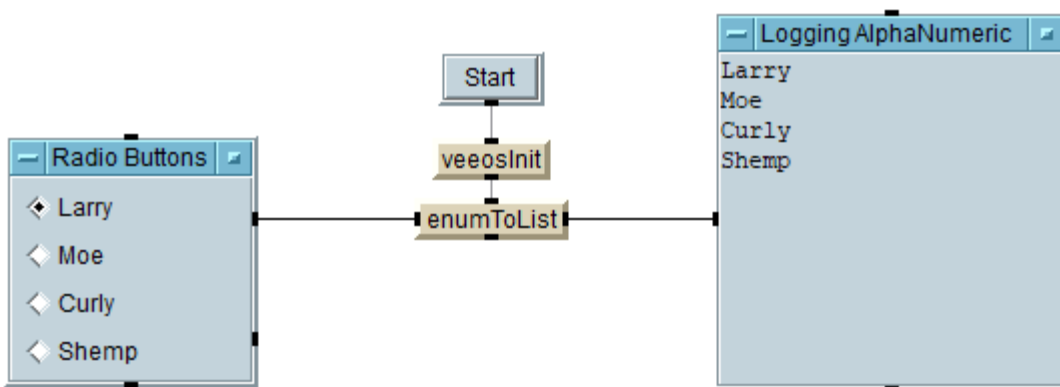
Use this to easily query enums to see what values they contain

Location

menu: veeos ==> Data ==> Conversion ==> enumToList

library: [dataLib](#)

Example



Notes

While an enum is very useful, it is also somewhat opaque. That is, given an arbitrary enum

there is no really easy way to see what the scope of possible values is. This function enables a very simple method by converting to a string array.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

eval(function,arg)

Evaluates the given function at the given argument.



Syntax

`o_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

`o_eval("2+a^2", 4)` returns 18

`o_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

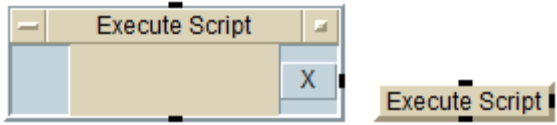
See Also

[ifThenEval](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

executeScript()

Instructs the server to run the previously specified script.



Syntax

`o_executeScript()`

no inputs

return: Text - "OK" means that the server has received this directive. Any other result is an error.

Usage

After data and a script have been uploaded, `o_executeScript()` instructs the script server to execute the script.

Location

menu: veeos ==> Scripting ==> Advanced ==> Execute Script

library: [scriptLib](#)

Example

Notes

This object needs to be run every time a script is to be executed. Since the script and data are persistent (see [sendData](#) and [sendScript](#)) it is completely possible to execute a script multiple times without sending more data.

Since the actual script execution may take a long time, this object should be considered to launch the script in the background. That is, the script is started but this object terminates immediately and does NOT wait for the script execution to complete. In a typical case one would use a polling loop with [checkStatus](#) to determine if the script execution is complete.

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

See Also

[checkStatus](#), [disconnectServer](#), [executeScript](#), [getResult](#), [sendData](#), [sendScript](#), [setShutdownTimeout](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

fallTime(a,margin)

Determines the fall time of an arbitrary signal



Syntax

`o_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

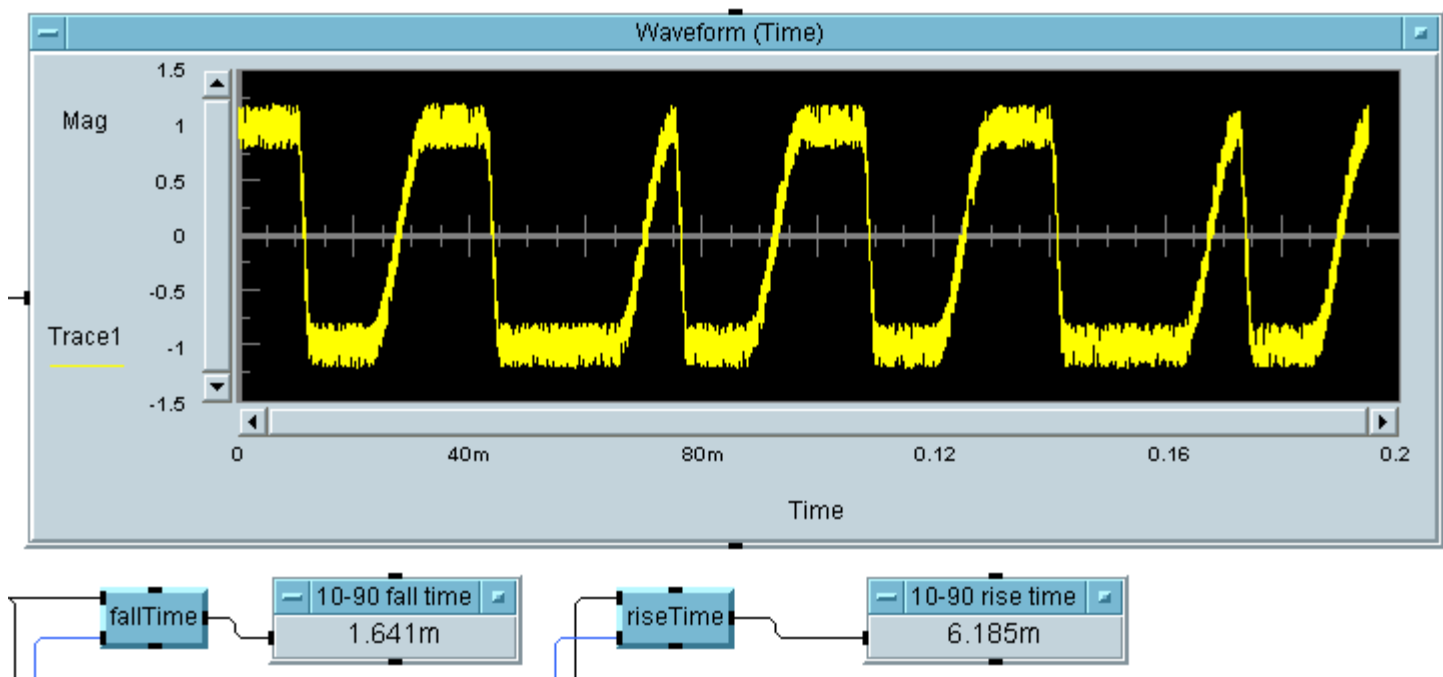
One common parameter for digital signals (pulses, square waves, etc.) is fall time. Fall time is the transition time from a lower extreme to an upper extreme. This is commonly defined as 10-90 time (time from 10% of the full transition to 90%) or 20-80 fall time. This function takes an arbitrary signal 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

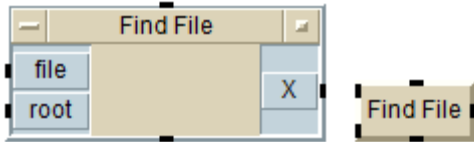
VEE 7.0+, all platforms

See Also

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

findFile(file,root)

Determines the location(s) of a file within a given directory tree.



Syntax

`o_findFile(file,root)`

file: Text - the full name of the file to be found

root: Text - the starting directory to search

return: scalar or array Text - the full path(s) to the file(s) found that match the specified name

Usage

Use this to find any which is in the file system.

Location

menu: veeos ==> Scripting ==> Script-Based Functions ==> Find File

library: [scriptLib](#)

Example

The typical usage of *findFile* occurs when the user knows that a file exists somewhere but is not sure where it is. For instance there may be multiple versions of Python installed and the user wants to find them all.

`o_findFile("python.exe","C:\")` finds several such files and returns an array

```
C:\apps\python3.8\python.exe  
C:\Users\stan\AppData\Local\Microsoft\WindowsApps\python.exe  
C:\Program Files (x86)\Agilent\VEE Pro 7.0\veeos\python\python.exe
```

Notes

Since a typical system has many files, *findFile()* may take a long time to execute. If you

choose a very general path such as C:\ then the entire file system is searched and all matching files- including those in the Recycle Bin- will be found and reported.

Reference

Dependencies

This function is implemented with a Python script called via [scriptPro](#).

Supported On

VEE 7.0+, Windows 7+

See Also

[which](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

frequency(a)

Determines the dominant frequency of an arbitrary signal.



Syntax

o_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 $1/\text{period}$ 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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

fromClipboard(a)

Reads text from the system clipboard.



Syntax

o_fromClipboard()

return: text - content of the system clipboard if text, otherwise null string

Usage

Use this to bypass the usual "paste" operation (CTRL-V) needed in order to get text from the system clipboard.

Location

menu: veeos ==> sys ==> fromClipboard

library: [sysLib](#)

Example

If one brings up a text editor and types in "Hello World", then cuts it using CTRL-C, then in VEE o_fromClipboard() reads "Hello World" from the clipboard.

Notes

This can be useful in having VEE interact with other programs directly (use clipboard for IPC) or with the user in a semi-manual method. If the clipboard contains a graphic or other non-text content then a null string is returned.

Reference

Microsoft documents [the underlying system call](#) in their MSDN section.

Dependencies

This uses a dotNet system call to System.Windows.Forms and as such it may not be fully portable.

Supported On

VEE 7.0+, all platforms

See Also

[toClipboard](#), [printText](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

funcFilter(a,Hs)

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



Syntax

`o_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](#)

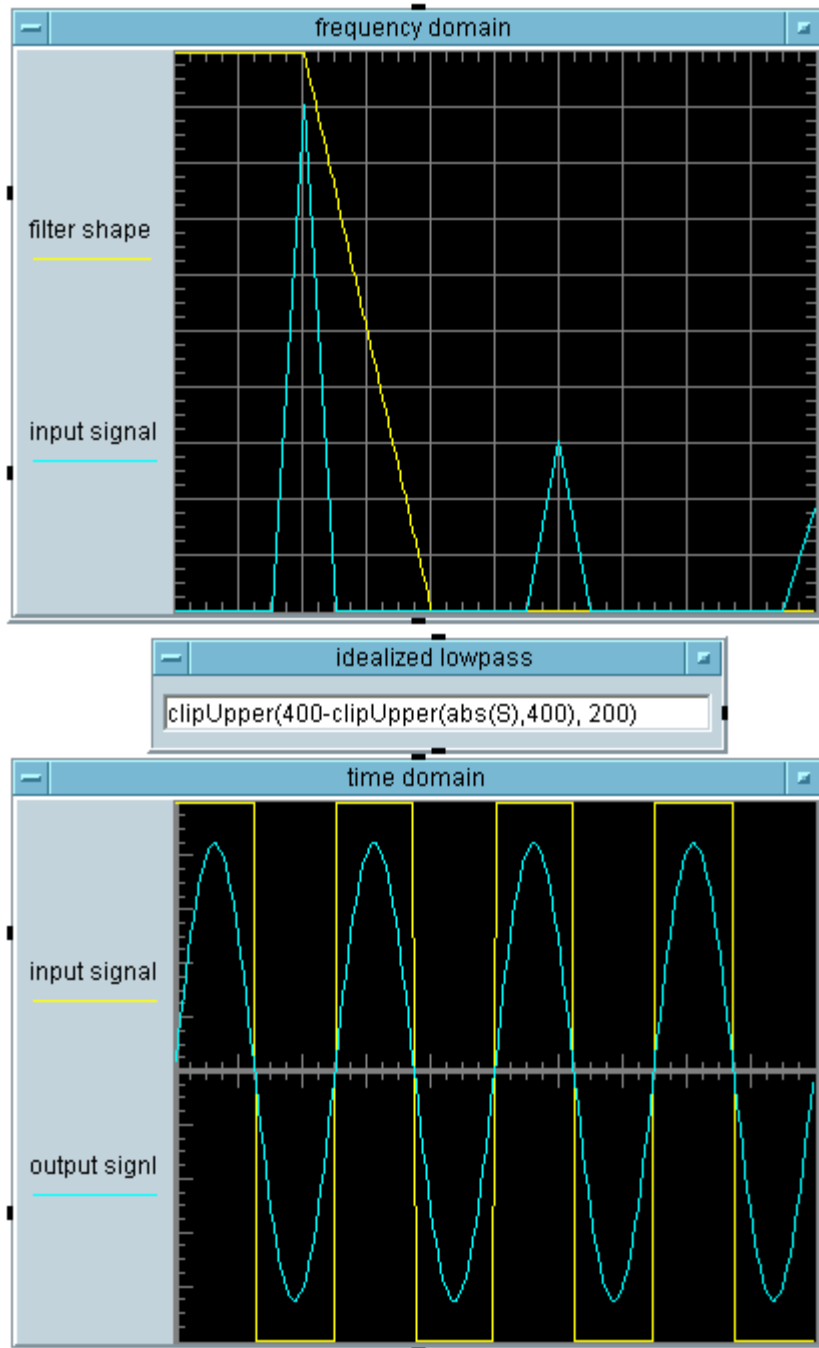
Example

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

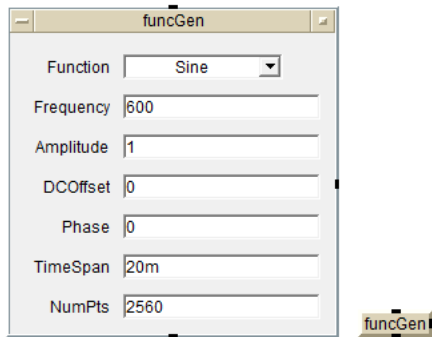
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 08:29.

funcGen(function,frequency,amplitude,DCOffset,Phase,TimeSpan,Numpts)

A general purpose function generator.



Syntax

o_funcGen(function,frequency,amplitude,DCOffset,Phase,TimeSpan,Numpts)

Function: Text - basic waveform function: Cosine, Sine, Triangle, Square, Ramp

Frequency: - Text - expression defining the frequency of the signal

Amplitude: - Text - expression defining the amplitude of the signal

DCOffset: - Text - expression defining the DC offset of the signal

Phase: - Text - expression defining the phase offset of the signal

TimeSpan: - Text - expression defining the duration of the signal

Numpts - Text - expression defining the number of samples of the signal

return: Waveform

Usage

This extends the capabilities of the built-in Function Generator by allowing arbitrary expressions for the various settings. By doing so complex waveforms may be generated easily.

`Frequency`, `Amplitude`, `DCOffset`, and `Phase` are variables that are mutually available. Hence the expression defining any of these may use the others as well as the `TimeSpan` and `NumPts` variables, as well as the `Time` variable defined by `NumPts` and `TimeSpan`

`TimeSpan` and `Numpts` can also be expressions but may NOT use these internal variables.

Location

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

library: [sigLib](#)

Example

Since expressions are possible, complex waveforms such as this amplitude-modulated square Chirp signal are easy to specify. Note the usage of the frequency and numPts variables.



Notes

If any of the defined parameters are arrays they MUST be of size numPts.

If you create circular definitions or too much cross-coupling of parameters expect errors! Keep it simple.

Reference

Dependencies

Supported On

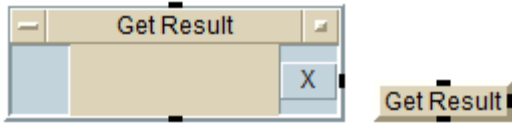
VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

getResult()

Queries the server for the requested output data from the last script execution.



Syntax

o_getResult()

no inputs

return: Record - the requested output variables as fields and an exit code and optionally error messages

Usage

After data and script are sent and the script is executed the server has on hand the values of the return variables that were specified in the script. getResult() returns those variables in the form of a record

Location

menu: veeos ==> Scripting ==> Advanced ==> Get Result

library: [scriptLib](#)

Example

Notes

This object needs to be run to return the variable values. If there are no return variables then this object does not need to run.

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

See Also

[checkStatus](#), [disconnectServer](#), [executeScript](#), [getResult](#), [sendData](#), [sendScript](#), [setShutdownTimeout](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

help

Brings up a local copy of the online help in a popup window



Syntax

- *none*

Usage

Use this as one way to access a local copy of the online help.

Location

menu: veeos ==> help

library: [veeosLib](#)

Example

Notes

The veeos install includes a copy of the Library Reference Guide from veeos.org. This is HTML code copied directly from the website, so needs an HTML browser to view. One can use their own browser directly (if it supports and is configured to allow a local file system) but for convenience, Help is offered. It works by brining up a browser control via ActiveX. Note that this is asynchronous so that you may leave it up independently of what you are doing in VEE.

NOTE that this uses ActiveX so may not work on all Windows installs due to installation path issues.

Reference

Supported On

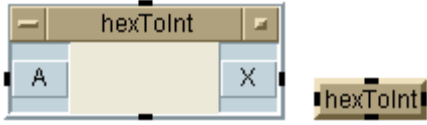
VEE 7.0+, Windows XP+

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 03 August 2021 11:34.

hexToInt(a)

Interprets input as hex value and outputs it as an integer



Syntax

o_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

o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

hilbert(a)

Computes the Hilbert transform of the input waveform.



Syntax

`o_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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

ifThenEval(cases,A)

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



Syntax

o_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 `o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

importCLib(libName,DLL, def)

Imports a Compiled Library



Syntax

o_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

o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

importULib(libName,file)

Imports a User Library



Syntax

o_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

o_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

[importCLib](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

isElement(set,element)

Determines if a given element exists within a set.



Syntax

`o_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

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

`o_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

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

isPrime(a)

Determines if a number is prime.



Syntax

o_isPrime(a)

a: scalar Integer - number to be tested

return: boolean - 0=not prime, 1=prime

Usage

Use whenever you would like to test a number to see if it is prime. A typical use would be to generate a set of primes for factorization.

Location

menu: veeos ==> Math ==> isPrime

library: [mathLib](#)

Example

o_isPrime(4) returns 0

o_isPrime(99961) returns 1

Notes

The algorithm used here is fairly brute-force but as such as about foolproof. While it is reasonably fast , it would not be appropriate to use for extremely large values.

Reference

Dependencies

Supported On

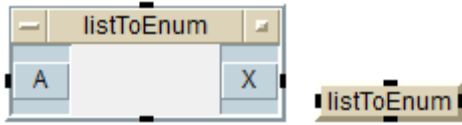
VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

listToEnum(a)

Converts a text array into an enum.



Syntax

o_listToEnum(a)

a: text array- value to be converted

return: enum with values matching the text array

Usage

Use this to easily generate enums, including within expressions.

Location

menu: veeos ==> Data ==> Conversion ==> listToEnum

library: [dataLib](#)

Example

o_listToEnum(["one", "two", "four"]) returns an enum with three values

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

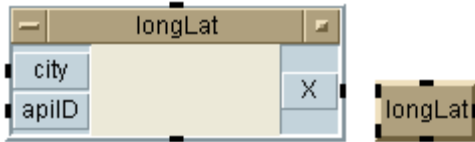
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

longLat(city,apiID)

Returns the longitude and latitude for a given city



Syntax

o_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 ==> Network ==> 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

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 30 July 2021 10:36.

period(a)

Determines the period of an arbitrary signal.



Syntax

o_period(a)

a: waveform - signal to be analyzed

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

Usage

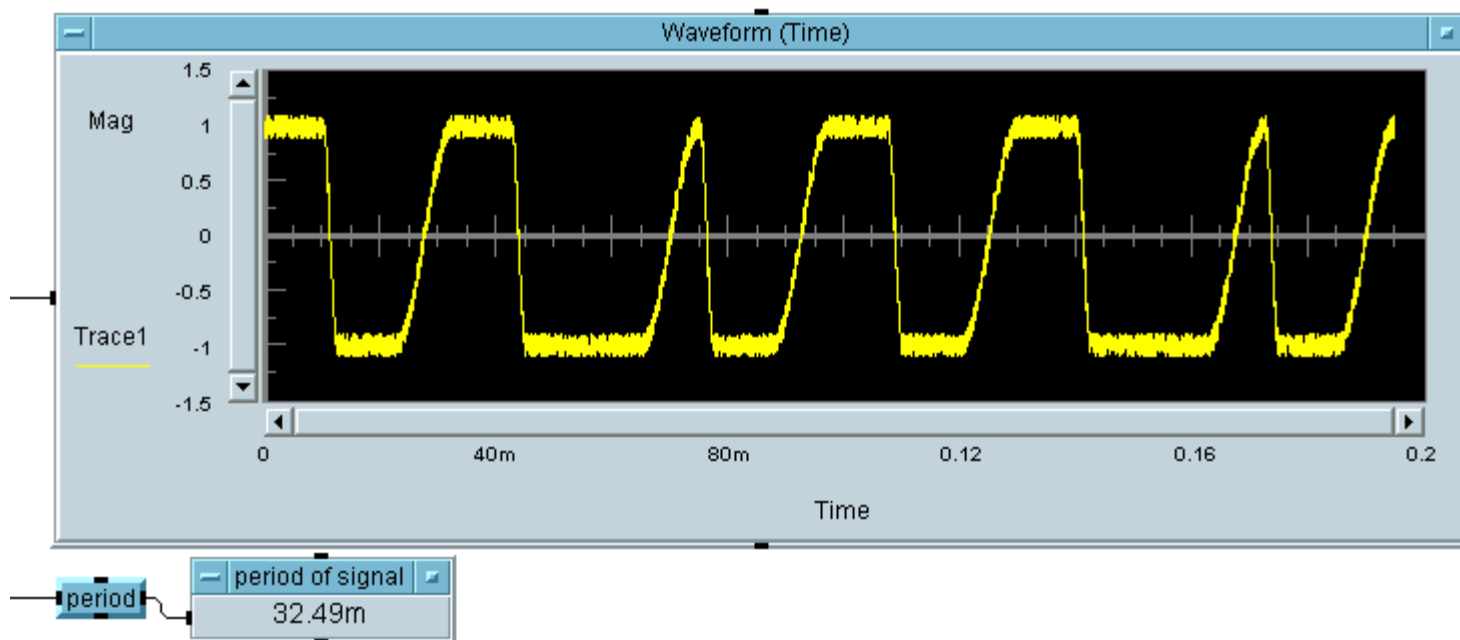
One common parameter for digital signals (pulses, square waves, etc.) is period. Period is the time between successive transitions of the signal. This function takes an arbitrary signal and analyzes it to determine the average period.

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](#)

PI(FofN,from,thru)

Evaluates the product of a series.



Syntax

`o_PI(FofN,from,thru)`

FofN: scalar Text - function to be evaluated with argument "n". *Optionally an array input can be used for a more complex series. See Notes below.*

from: scalar number - lower limit of "n"

thru: scalar number - upper limit of "n"

return: numeric - the product of terms F(n) for n=from through n=thru

Usage

An arbitrary function of "n" can be evaluated for a series. This implements a standard "PI" notation product.

Location

menu: veeos ==> Math ==> PI

library: mathLib

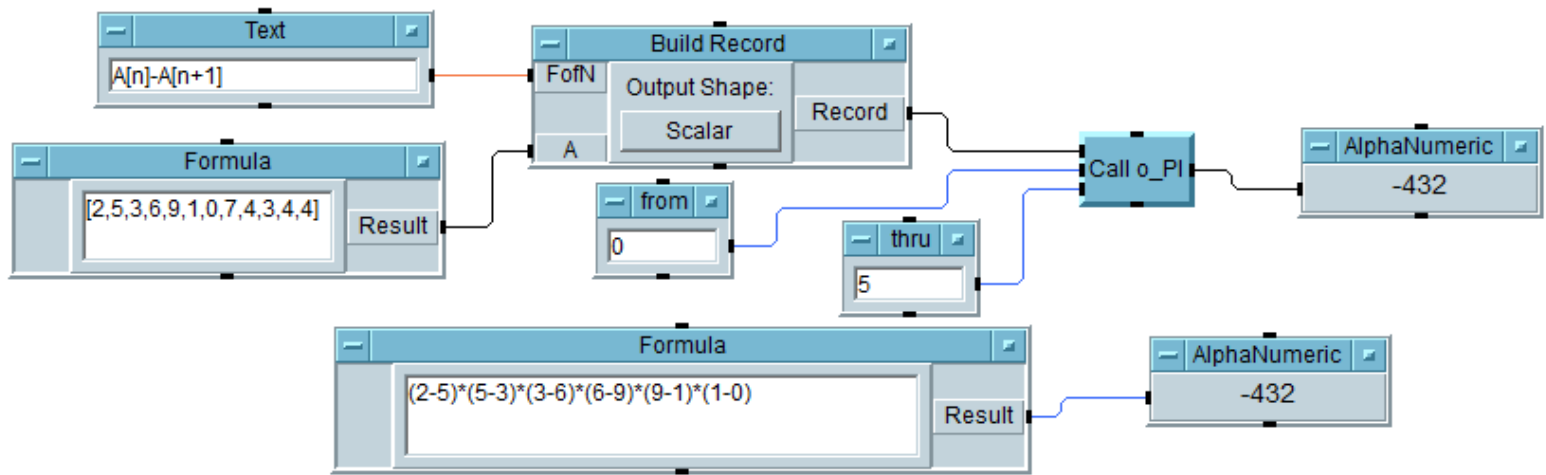
Example

`o_PI("n^2*(n-1)",3,5) = (9*2)*(16*3)*(25*4) = 86400`

Notes

A very common operation for series is the product of the various terms. This evaluates that product.

New for R2: instead of being limited to a single range of input values, a more complex series such as a Kalman filter may be evaluated by specifying a function and a series of values in the form of a record.



Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[SIGMA](#)

polarFit(a)

Determines the unique circle defined by given three points.



Syntax

o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

polarRegress(a)

Determines the circle that best fits a set of points.



Syntax

o_polarRegress(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 as long as noise level is low this works quickly and well.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[polarFit](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

polyFilter(a,num,den)

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



Syntax

`o_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

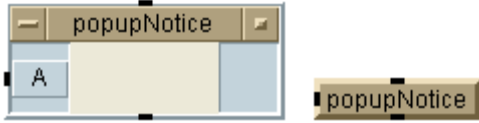
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

popupNotice(a)

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



Syntax

`o_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

`o_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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

ppCap(area, thickness, Er)

Calculates Parallel Plate Capacitance



Syntax

`o_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
`o_ppCap(1, 0.25, 9.9) = 0.35 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

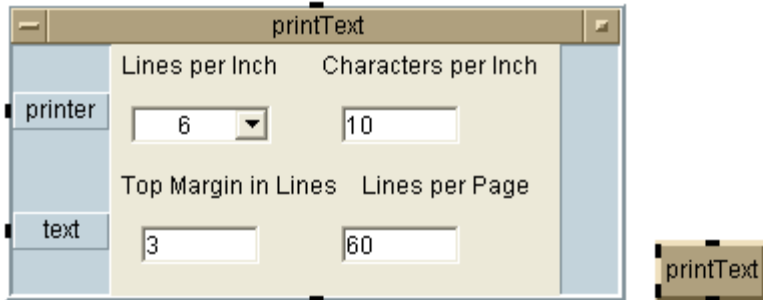
VEE 7.0+, All platforms.

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

printText(printer,text)

Sends arbitrary text to a networked printer



Syntax

`o_printText(printer,text)`

printer: scalar text - network name or address of desired printer

text: any text - text to be printed

Usage

Similar to the To Printer object but unlike To Printer this allows one to select a printer.

Location

menu: veeos ==> sys ==> printText

library: [sysLib](#)

Example

`o_printText("192.168.1.45",ramp(60,1,60))` prints a ramp of values to single page of text resulting in lines numbered 1 through 60.

Notes

By default, `o_printText()` prints uses a very "vanilla" set of print parameters: 6 lines per inch, 60 lines per page, 80 characters per line.

The open object version of this function allows the user to change these major settings as desired. For the advanced user these settings may also be changed programmatically. To

do this, enter a Record to the "printer" input instead of just a text constant. This record should have a field named "printer" and any combination of optional fields named "linesPerInch", "charsPerInch", "topMargin", and "linesPerPage".

Reference

This function writes directly to a network printer, bypassing Windows' printer queue and drivers. As such there is no need to install drivers on the machine being used. Note that only PCL printers are currently supported, though postscript could be added if there is demand. The expert user can include any desired PCL commands in the text to be printed if more control is desired. If there's enough demand the printer object could easily enough be given more capabilities.

Dependencies

none

Supported On

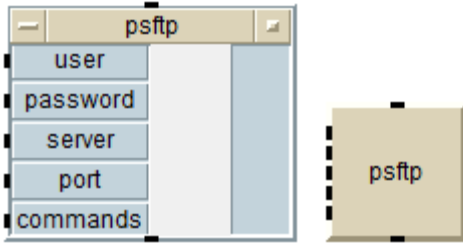
VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

psftp(user, password, server, port, commands)

Automates an SFTP session.



Syntax

`o_psftp(user, password, server, port, commands)`

user: Text - login name

password: Text - user's password

server: Text - IP address or server name

port: Int - port to use

commands: array Text - the exact commands to be executed

A scalar Text - domain name or IP address

return: array Text - registration information

Usage

Automates an FTP session

Location

menu: veeos ==> Network ==> psftp

library: [netLib](#)

Example

```
o_psftp("demo","password","test.rebex.net",22,["lcd c:\temp","get readme.txt",  
"bye"])
```

 logs onto a server and downloads a readme file.

Notes

The FTP protocol is unforgiving so the syntax of your command file must be precise. It is

strongly advised to manually execute the FTP transactions and carefully record them before attempting to automate.

ALSO since this is using secure comm you most likely will need to accept an encryption key manually before anything else will work.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[longLat](#), [currentTemp](#), [currentHumidity](#), [whois](#), [wget](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

riseTime(a,margin)

Determines the rise time of an arbitrary signal.



Syntax

`o_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

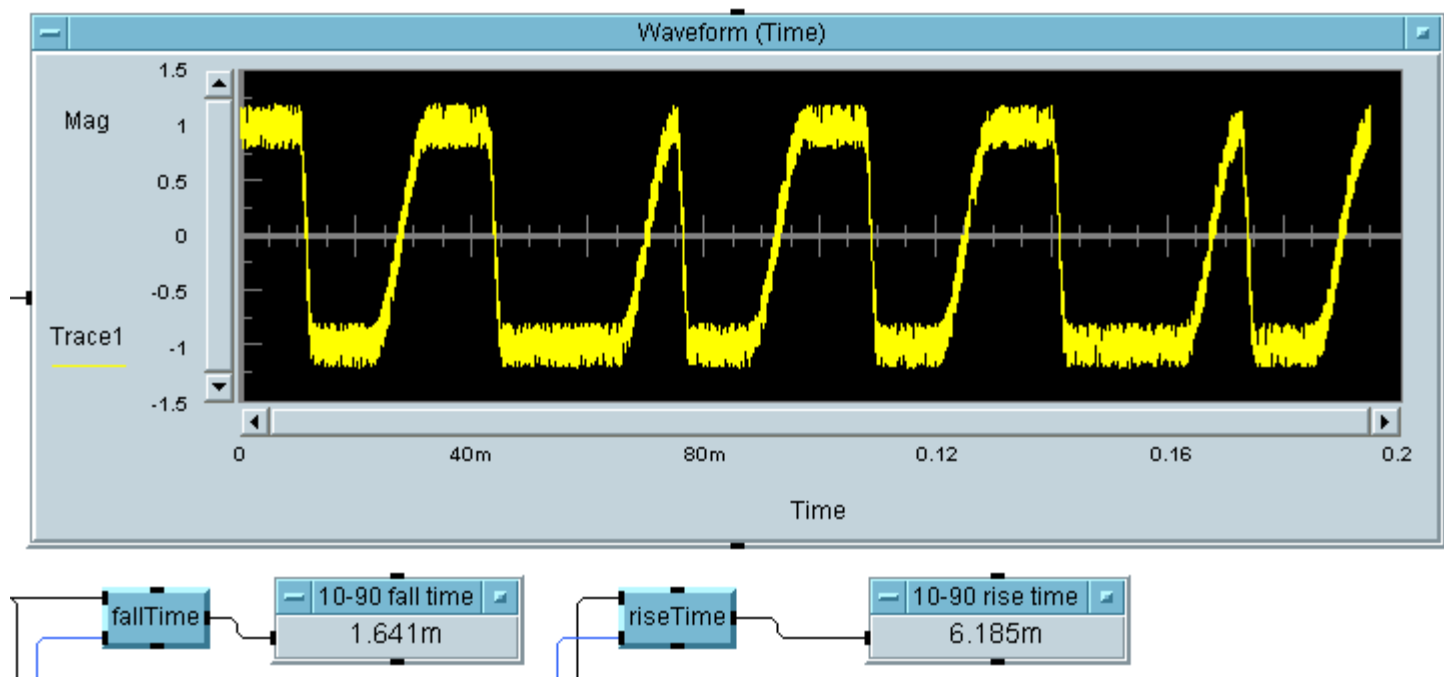
One common parameter for digital signals (pulses, square waves, etc.) is rise time. Rise time is the transition time from a lower extreme to an upper extreme. This is commonly defined as 10-90 time (time from 10% of the full transition to 90%) or 20-80 rise time. This function takes an arbitrary signal 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](#)

randomInt(min,max)

Returns a random integer from specified range.



Syntax

`o_randomInt(min,max)`

return: $\min \leq \text{random integer} \leq \max$

Usage

Use this to generate a random integer.

Location

menu: veeos ==> Math ==> randomInt

library: [mathLib](#)

Example

`o_randomInt(0,34)` is equally likely to return any of the 35 integers in the inclusive range of 0 to 34

Notes

This is similar to the built-in `random()` function, but the built-in returns real64's, this returns integers.

In looking at this function, several approaches were tried, as noted in [evalRandomInt.vee](#). Note that it isn't hard to end up with biased result, and that some approaches are slower than others. Also, as it turns out, the "best" solution in this case is to read the Help and note that the built-in `random()` function uses a non-inclusive range $[A,B)$ so can be used almost directly.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[randomReorder](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

randomReorder(a)

Scrambles the order of a string or array.



Syntax

`o_randomReorder(a)`

a: array or string - the set of values to be scrambled

return: same as original but scrambled into a random order

Usage

Use this when you have a set of values but would like to scramble the order of those values.

Location

menu: veeos ==> Math ==> Arrays ==> randomReorder

library: [mathLib](#)

Example

`o_randomReorder("this string")` returns any string such as "rgnts_hstii" that has all the same characters but in a random order

`o_randomReorder([1,2,3 4])` could return [1,4,3,2] or [4,1,3,2] or any number of other random reordering.

Notes

This should work with any data type and scalar, 1D or 2D arrays.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[randomInt](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

rootFilter(a,zeroes,poles)

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



Syntax

`o_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 $S-S+67$ or just 67.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

runInShell(a)

Runs specified code in a Windows shell



Syntax

`o_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

`o_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

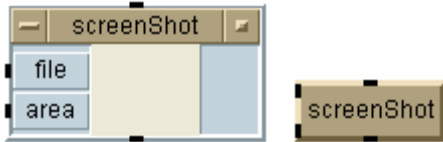
VEE 7.0+, Windows XP+

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

screenShot(file,area)

Takes snapshot of screen



Syntax

`o_screenShot(file,area)`

file: scalar text- file in which to store the screenshot

area: array 1D of coord or scalar text- two coordinates in pixels that define the area of the screen to be captured

Usage

Use this to programmatically capture all or part of the current screen display.

Location

menu: veeos ==> system ==> screenShot

library: sysLib

Example

`o_screenShot("C:/temp/capture.png", "")` captures the entire screen as a PNG file

`o_screenShot("", "(0,0) (500,300)")` captures a rectangular area to the clipboard.

Notes

- File Name
 - must have suffix of BMP or PNG. Other graphics types are not supported at this time.
 - If the file name is a null string then the screen is captured to the clipboard
- Area
 - Coordinates are referenced to the upper left of the screen and are in pixels.

These may be specified by 2 2D coordinates in a coord array or simply written as text.

- If the area field is a null string then the entire screen is captured.

Reference

Dependencies

Utilizes external compiled code which is supplied automatically with VEEOS installs. This code is open source and can also be used outside of VEE if desired.

Supported On

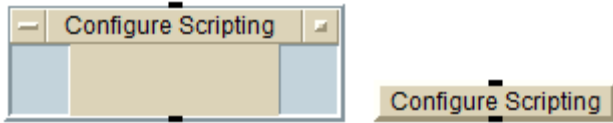
VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

scriptConfig()

Assists the user in controlling various settings related to scripting.



Syntax

o_scriptConfig()

no inputs

no outputs

Usage

As with other VEEOS settings, the script configuration settings are stored in an RC file. As detailed in the [bootstrap documentation](#) there is a hierarchy of RC files. This function helps the user edit the RC file in their home directory by providing a simple file editor. All setting other than those needed for scripting are left as is.



The various sections of this dialog are explained below in the general order in which they should be used. Note that the location of the RC file being edited is indicated in the upper notes.

1. The affected engine is selected (InstalledPython is the default)
2. A text box is provided for the location of the executable needed for a given engine. This is initially populated with an estimated value. If this value is correct great, otherwise fill it in manually or use one of the assists. As a hint, the dialog provides the name of the executable for the chosen engine.
3. BROWSE brings up a standard file browser. Wander by wherever your engine is

installed.

4. SEARCH attempts to find your engine and returns whatever paths it finds that may be correct.
5. Once you have the correct path indicated, choose ACCEPT PATH
6. CLEAR clears the text box and deletes any associated setting that you may have
7. Once everything looks good choose WRITE CHANGES AND EXIT. This writes the appropriate settings to your RC file.

Location

menu: veeos ==> Scripting ==> Configure Scripting

library: [scriptLib](#)

Example

Notes

This object never actually needs to be run since the RC file is plain text and can be edited using any plain text editor (notepad is recommended). However this does help with the RC file syntax and it is exercised automatically as needed by the [script](#) object.

Reference

Dependencies

Supported On

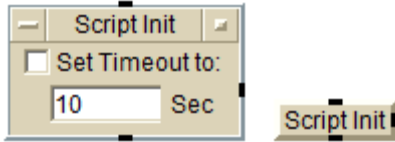
VEE 7.0+, Windows 7+

See Also

[script](#)

scriptInit()

Readies scripting by setting up the needed environment and starting the script server as needed. NOTE that this must be run at least once before any scripting or usage of objects that depend upon scripting.



Syntax

o_scriptInit()

no inputs

return: Text - confirmation that scripting is ready.

Usage

This object needs to run before any scripting is available. Its primary purpose is to make sure that the script server is running. If the server is not running and the server is local then the server is started. If the server is not running and is remote, then the user is notified that it needs to be started.

When selected, the "Set Timeout" option sets the length of inactivity in seconds after which the server will shut down. See [Set Shutdown Timeout](#)

Location

menu: veeos ==> Scripting ==> Advanced ==> scriptInit

library: [scriptLib](#)

Example

Notes

Reference

Dependencies

Supported On

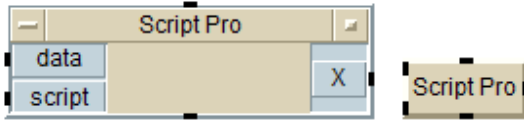
VEE 7.0+, Windows 7+

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

scriptPro(data,script)

The ScriptPro object provides a simple wrapper for executing arbitrary scripts. This object is meant to handle most of the overhead associated with actually running the script so that executing scripts is as simple as possible for the end user.



Syntax

o_scriptPro(data,script)

data: scalar Record - the field name are taken as input variables and the field values are the corresponding variable values.

script: array Text - the script to be executed, including language and return variable directives

return: Record - the requested output variables as fields and an exit code and optionally error messages

Usage

There are 4 pieces of information that the user provides:

1. **The script** as entered in the provided text area. This script includes three pieces in the following order:
 - a. The **Language Shebang** - The language in which the script is written. The default is an embedded python that is provided as part of VEEOS. For any other choice the user needs to install and configure that desired language before accessing it through the script object. See the Supported Languages page for more details.

```
#! <language> [full path to the language engine]
```

- b. Choice of **Output Variables** - a comma-delimited list of variables. *Note that these variables must conform with the script and language requirements. In particular most languages are case-sensitive so be sure and follow language rules.*

```
returnToVEE [variables]
```

- c. The **Script** itself written in chosen language. - *Note that this script is NOT a VEE script so the syntax requirements must follow the language in which the script is*

written. Since editing in the script object is highly limited, it is recommended that for anything more than very simple scripts editing be done in a proper editor and then copied into the script object.

2. **Input Variables** - the input variables as a **record**. Here the **variables** are defined by the field names and the values by the field values. Note that records are NOT supported as field values. Also note that as with scripts, the variable names must conform with the requirements of the chosen language.

Field name	Value
A	Text field
B	1.25

And there is a single **output** that returns the results of the script in the form of a record. As with the Input Variables, this record will have fields associated with each Output Variable selected. There will typically be other fields also such as a status or associated error messages.

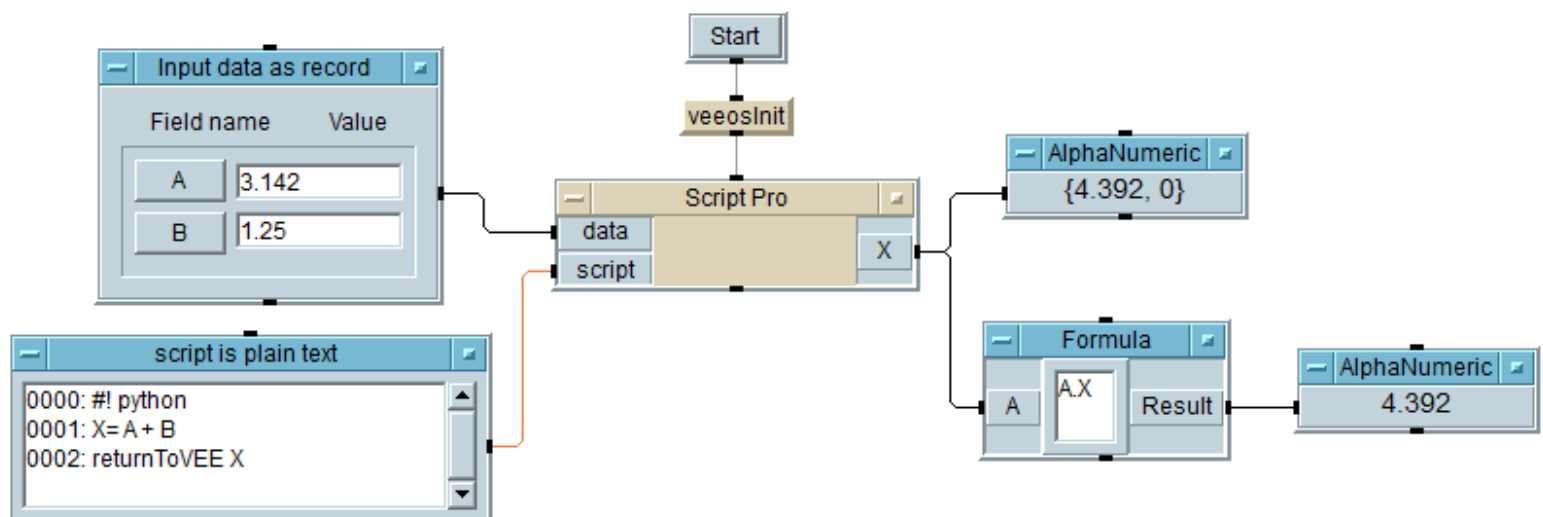
Location

menu: veeos ==> Scripting ==> Script Pro

library: scriptLib

Example

Note the input and output variable records.



Notes

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

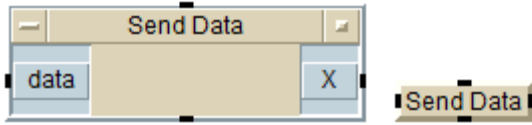
See Also

[Script](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

sendData(data)

Uploads the specified data to the script server.



Syntax

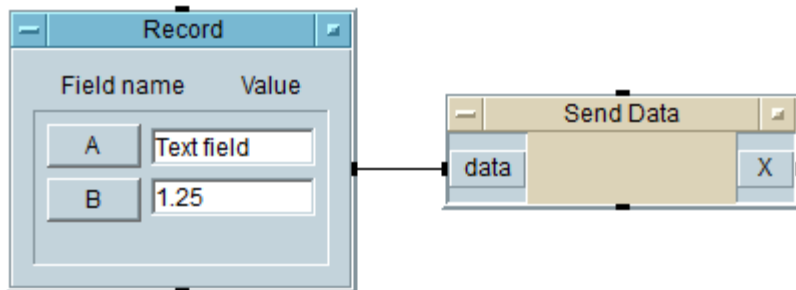
o_sendData(data)

data: Record - specifies both name and values of the variables to be used in the specified script

return: Text - "OK" means that the server has received this directive. Any other result is an error.

Usage

The data input must be a Record. The **variables** are defined by the field names and the *values* by the field values. *Note that records are NOT supported as field values. Also note that as with scripts, the variable names must conform with the requirements of the chosen language.*



Location

menu: veeos ==> Scripting ==> Advanced ==> Send Data

library: [scriptLib](#)

Example

Notes

This object must be run at least once so that the server knows what data to use.

The script and data are persistent and independent. As such a script may be uploaded once then executed many times with different data. Similarly a dataset may be uploaded once then processed using various scripts.

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

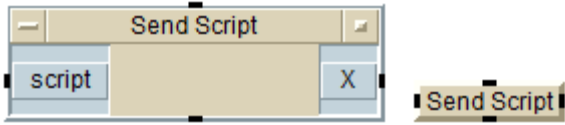
See Also

[checkStatus](#), [disconnectServer](#), [executeScript](#), [getResult](#), [sendData](#), [sendScript](#), [setShutdownTimeout](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

sendScript(script)

Uploads the specified script to the script server.



Syntax

`o_sendScript(script)`

script: text array - the specified script to be run

return: Text - "OK" means that the server has received this directive. Any other result is an error.

Usage

Before being executed, a script needs to be uploaded to the script server. This object automates that process.

Location

menu: veeos ==> Scripting ==> Advanced ==> Send Script

library: [scriptLib](#)

Example

Notes

This object must be run at least once so that the server knoww what script to execute.

The script and data are persistent and independent. As such a script may be uploaded once then executed many times with different data. Similarly a dataset may be uploaded once then processed using various scripts.

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

See Also

[checkStatus](#), [disconnectServer](#), [executeScript](#), [getResult](#), [sendData](#), [sendScript](#), [setShutdownTimeout](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

setAND(a,b)

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



Syntax

`o_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

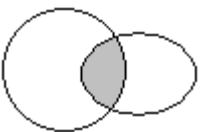
library: [mathLib](#)

Example

`o_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

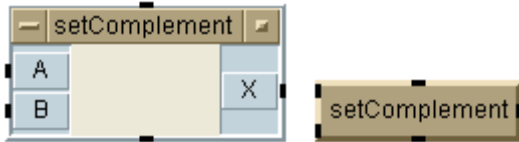
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

setComplement(a,b)

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



Syntax

`o_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

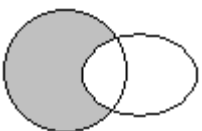
library: [mathLib](#)

Example

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

setOR(a,b)

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



Syntax

o_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

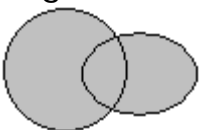
library: [mathLib](#)

Example

o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

setShutdownTimeout()

Sets the idle timeout after which the script server shuts down. Must be run before uploading data or scripts.



Syntax

`o_setShutdownTimeout()`

A: Int32 - the timeout in seconds

return: Text - "OK" means that the server has received this directive. Any other result is an error.

Usage

By default the script server shuts itself down after a period of inactivity. This is done to minimize the impact on the user's system. The downside is that the next script invocation has the overhead of restarting the server. By setting this timeout the user can avoid this downside.

Location

menu: veeos ==> Scripting ==> Advanced ==> Set Shutdown Timeout

library: [scriptLib](#)

Example

`o_setShutdownTimeout(300)` sets the timeout to 5 minutes.

Notes

This object never actually needs to be run, but may be useful.

Since setting the timeout requires a new session on the server, any existing data or scripts or results are lost when this object executes. As such it should be run BEFORE uploading scripts or data.

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

See Also

[checkStatus](#), [disconnectServer](#), [executeScript](#), [getResult](#), [sendData](#), [sendScript](#), [setShutdownTimeout](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

setXOR(a,b)

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



Syntax

`o_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

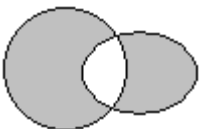
library: [mathLib](#)

Example

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

setVar(name,value)

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



Syntax

`o_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

`o_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

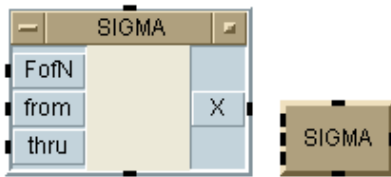
See Also

[createConstant](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:13.

SIGMA(FofN,from,thru)

Evaluates the sum of a series.



Syntax

`o_SIGMA(FofN,from,thru)`

FofN: scalar Text - function to be evaluated with argument "n" *Optionally an array input can be used for a more complex series. See Notes below.*

from: scalar number - lower limit of "n"

thru: scalar number - upper limit of "n"

return: numeric - the sum of terms $F(n)$ for $n=\text{from}$ through $n=\text{thru}$

Usage

An arbitrary function of "n" can be evaluated for a series. This implements a standard "SIGMA" notation sum.

Location

menu: veeos ==> Math ==> SIGMA

library: mathLib

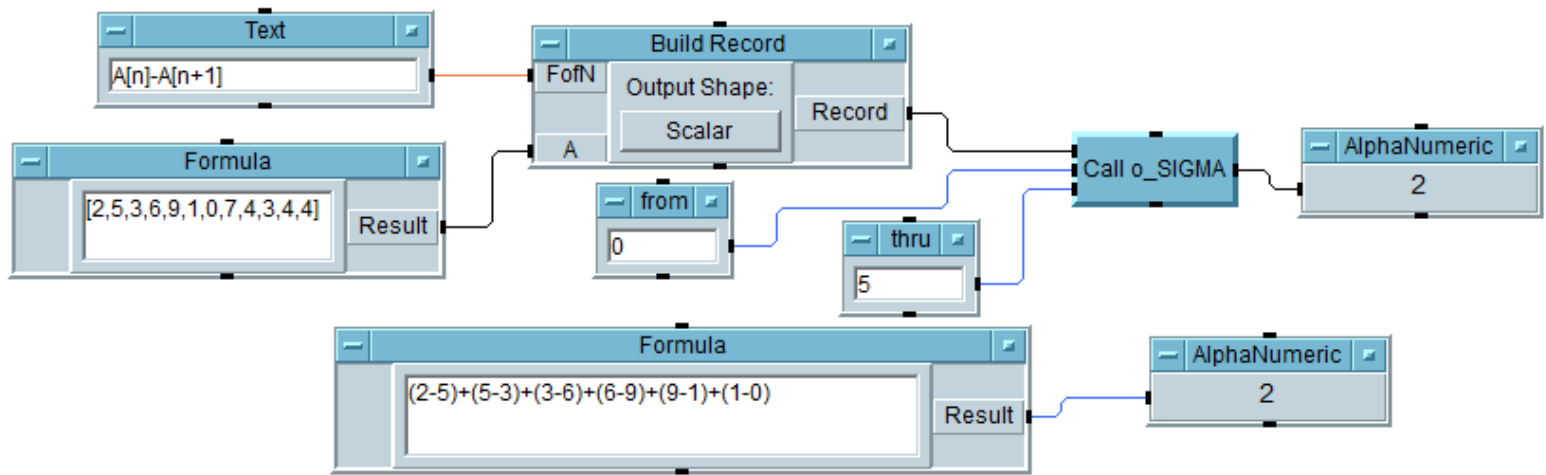
Example

`o_SIGMA("n^2*(n-1)",3,5) = (9*2)+(16*3)+(25*4) = 166`

Notes

A very common operation for series is the sum of the various terms. This evaluates that sum.

New for R2: instead of being limited to a single range of input values, a more complex series such as a Kalman filter may be evaluated by specifying a function and a series of values in the form of a record. FofN and A must be record fields as shown below.



Reference

Dependencies

Supported On

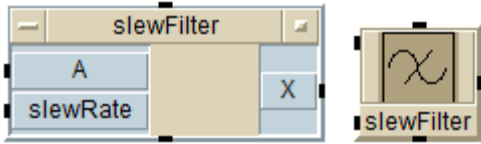
VEE 7.0+, all platforms

See Also

[PI](#)

slewFilter(a,slewRate)

Implements an analog filter whose response is based on a given slew rate.



Syntax

`o_slewFilter(a,slewRate)`

a: waveform - input waveform

slewRate: numeric - slew rate in 1/s - scalar for bidirectional, 1D array for separate rates - [pos rate, neg rate]

return: waveform- input transformed by the slew-based filter

Usage

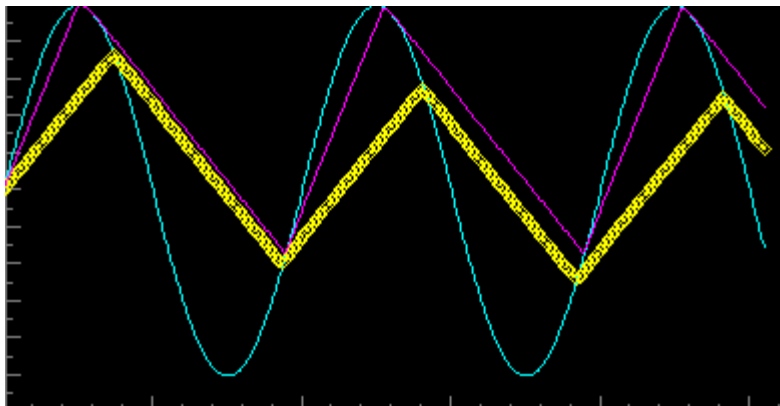
Typical usage is to model a system that can only change at up to a given speed due to finite drive ability. Classic case is a capacitor being charged at the maximum available current which limits that change in voltage per second.

Location

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

library: [sigLib](#)

Example



The original signal is in cyan. The yellow shows slew in both directions at the same rate (`o_slewRate(A,20)`), the magenta is slewed in both directions (`o_slewRate(A,[40,20])`) but at different rates.

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

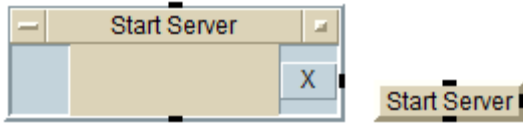
See Also

[analogFilter](#), [envelope](#), [decayFilter](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

startServer()

Starts the script server, enabling scripting to proceed.



Syntax

o_startServer()

no inputs

return: Text - "OK" means that the server has started successfully. Any other result is an error.

Usage

[Scripting](#) works via a script server. This object starts the server if it is not already started.

Location

menu: veeos ==> Scripting ==> Advanced ==> Start Server

library: [scriptLib](#)

Example

Notes

This object needs to be run at some time in order to start the server. When using Script or Script Pro this object is run automatically. When using the Advanced scripting objects it needs to be run first before the other pieces can proceed.

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

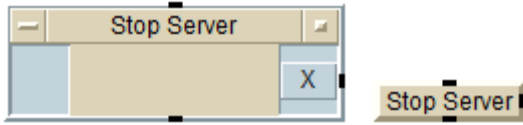
See Also

[checkStatus](#), [disconnectServer](#), [setShutdownTimeout](#), [stopServer](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

stopServer()

Shuts down the script server.



Syntax

o_startServer()

no inputs

return: Text - "OK" means that the server acknowledged the directive to shut itself down. Any other result is an error.

Usage

[Scripting](#) works via a script server. This object shuts down the server, typically when it is no longer needed.

Location

menu: veeos ==> Scripting ==> Advanced ==> Stop Server

library: [scriptLib](#)

Example

Notes

Since the server automatically shuts itself down after a [timeout](#) this object does not need to be run unless it is desired to shut the server down before it would normally quit anyhow.

If you would prefer to disconnect from the server rather than shut it down, see [disconnectServer](#)

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

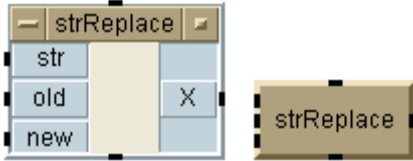
See Also

[checkStatus](#), [disconnectServer](#), [setShutdownTimeout](#), [startServer](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

strReplace(str,old,new)

Replaces occurrences of a target string within the input string.



Syntax

`o_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

```
o_strReplace("I thunk 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

VEE 7.0+. All platforms.

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

time(aDate,option)

Converts timestamp to time of the day.



Syntax

```
o_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

`o_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.

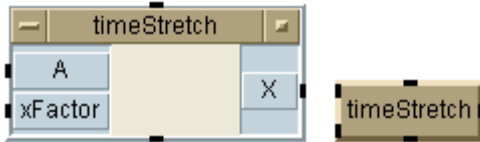
See Also

[week](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

timeStretch(a)

Stretches a series of data points by replicating them.



Syntax

o_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

o_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

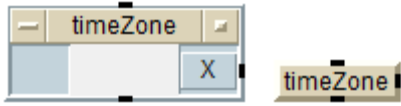
VEE7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

timeZone()

Determines the current timezone of the user's computer.



Syntax

```
o_timeZone()  
    no inputs  
    return: Int
```

Usage

Determines the current timezone of the user's computer.

Location

menu: veeos ==> Date/Time ==> timeZone

library: [veeosLib](#)

Example

Notes

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

Reference

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

Supported On

VEE 7.0+, All platforms.

See Also

[time](#), [yDay](#), [yWeek](#), [utcNow](#), [unixNow](#), [ywTodate](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

toClipboard(a)

Writes text to the system clipboard.



Syntax

o_toClipboard(a)
a: any text - description

Usage

Use this to bypass the usual "cut" operation (CTRL-C) needed in order to get text into the system clipboard for pasting into another application.

Location

menu: veeos ==> sys ==> toClipboard

library: [sysLib](#)

Example

o_toClipboard("Hello World") places the string "Hello World" onto the clipboard. A subsequent "paste" operation in a text editor places "Hello World"

Notes

Since any data type in VEE can be automatically converted to Text, this function can accept any data type. However since the conversion may not proceed with the formatting that you like it is suggested that you only input Text.

Reference

Microsoft documents [the underlying system call](#) in their MSDN section.

Dependencies

This uses a dotNet system call to System.Windows.Forms and as such it may not be fully portable.

Supported On

VEE 7.0+, all platforms

See Also

[fromClipboard](#), [printText](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

toCoord(a)

Creates Coordinates from input data.



Syntax

`o_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

`o_toCoord((1,2))` creates a scalar coordinate with X=1 and Y=2
`o_toCoord((1,@90))` creates a scalar coordinate with X=0 and Y=1
`o_toCoord(<array 1D of complex>)` creates a array of coordinates
`o_toCoord([1,2,3])` creates a scalar 3D coordinate (1,2,3)
`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

toEngr(a,numdig)

Formats a number in standard engineering format.



Syntax

`o_toEngr(a,numdig)`

a: scalar numeric - value to be formatted

numdig: scalar int - number of fractional digits to include

return: scalar text - formatted value

Usage

One of the few things that a To String cannot do is to write in Engineering standard format. This function adds that ability.

Location

menu: veeos ==> data ==> formatting ==> toEngr

library: dataLib

Example

`o_toEngr(100*pi,3)` returns 314.159

`o_toEngr(1000*pi,3)` returns 3.142 k

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

toggleRadix(a)

Changes points to commas- and vice versa.



Syntax

`o_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

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

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

toInt16(a)

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



Syntax

`o_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

`o_toInt16("a34")` returns 2612

`o_toInt16("4708")` returns 18184

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

toInt32(a)

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



Syntax

`o_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

`o_toInt32("a34a")` returns 41802

`o_toInt32("478823")` returns 4687907

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

toUInt8(a)

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



Syntax

`o_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

`o_toUInt8("a")` returns 10

`o_toUInt8("47")` returns 71

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

toBinStrN(a,nBits)

Formats a number as a binary string of arbitrary length.



Syntax

`o_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

`o_toBinStrN(38,8)` returns "00100110"

`o_toBinStrN(345,12)` returns "000101011001"

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

unique(a)

Returns unique subset.



Syntax

`o_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

`o_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

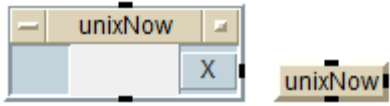
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

unixNow()

Determines the current timestamp as understood by the unix world.



Syntax

o_unixNow()
no inputs
return: scalar Real32

Usage

Determines the current timestamp as understood by the unix world.

Location

menu: veeos ==> Date/Time ==> unixNow

library: [veeosLib](#)

Example

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

Notes

Very similar to the builtin `now()` but that is Windows time as opposed to unix time. Windows uses a 64-bit clock referenced to January 1 of the year 1. Unix on the other hand uses a 32-bit clock references to 1 January 1970.

Reference

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

Supported On

VEE 7.0+, All platforms.

See Also

[time](#), [yDay](#), [yWeek](#), [timeZone](#), [utcNow](#), [ywToDate](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

unzip(zipfile,dest)

Uses standard ZIP to unpack zip bundles.



Syntax

o_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

o_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+

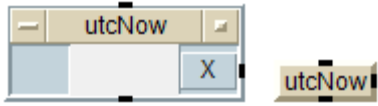
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

utcNow()

Determines the current timestamp referenced to UTC time rather than local time.



Syntax

```
o_utcNow()  
    no inputs  
    return: timestamp
```

Usage

Determines the current timestamp referenced to UTC time rather than local time.

Location

menu: veeos ==> Date/Time ==> utcNow

library: [veeosLib](#)

Example

Notes

Very similar to the builtin `now()` but `now()` is referenced to local time.

Reference

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

Supported On

VEE 7.0+, All platforms.

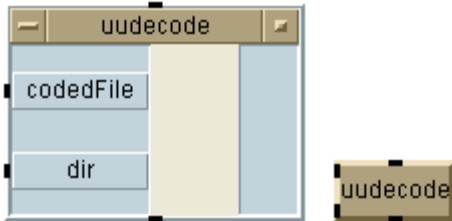
See Also

[time](#), [yDay](#), [yWeek](#), [timeZone](#), [unixNow](#), [ywTodate](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

uudecode(codedFile,dir)

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



Syntax

`o_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

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 ==> uudecode

library: [sysLib](#)

Example

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

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](#), [uuencode](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

uuencode(codedFile,dir)

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



Syntax

`o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 26 July 2021 09:41.

viewPicture(a)

Displays chosen picture in a photo viewer.



Syntax

o_viewPicture(a)

a: scalar Text - full path to picture to be viewed

return: int - exit code

Usage

Brings up the indicated picture in the user's default picture viewer.

Location

menu: veeos ==> system ==> viewPicture

library: [sysLib](#)

Example

o_viewPicture("c:\temp\screen.bmp") uses the default photo viewer to display the indicated picture.

Notes

This function uses the default picture viewer configured on the user's system. In general this will allow the user to view all picture within the chosen directory.

Reference

Supported On

VEE 7.0+, Windows XP+

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

wait(a)

Pauses execution for specified time in seconds



Syntax

`o_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
o_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

VEE 7.0+, Windows XP+

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

week(aDate)

Converts timestamp to week of the year.



Syntax

```
o_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

`o_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.

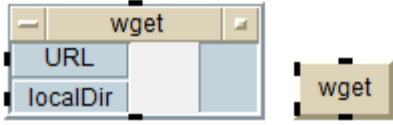
See Also

[time](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 29 July 2021 13:07.

wget(URL, localDir)

Downloads a file from the web.



Syntax

o_wget(A)

URL: scalar Text - URL of the file to be downloaded

localDir: scalar text - the local directory into which the URL should be downloaded

return: none

Usage

Use this to download an arbitrary file from the web.

Location

menu: veeos ==> Network ==> wget

library: [netLib](#)

Example

Notes

This can be used to download much more than a single file, so be a little careful!

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

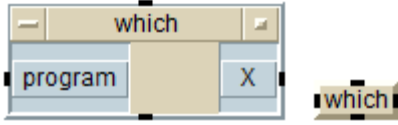
See Also

[longLat](#), [currentTemp](#), [currentHumidity](#), [whois](#), [psftp](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

which(program)

Determines the location of an executable in the system path.



Syntax

`o_which(program)`

program: Text - the name of the executable

return: Text - the full path to the first executable found that matches the specified name

Usage

Use this to find an executable which is in the system path. Executables not found in the path are not reported.

Location

menu: veeos ==> Scripting ==> Script-Based Functions ==> Which

library: [scriptLib](#)

Example

The typical usage of *which* occurs when the user has several executables in their path and wants to determine which one gets used. For example there may be several versions of Python installed. When the user executes "python" via Execute Program Python runs fine but it may be unclear which version runs.

`o_which("python.exe")` returns `C:\apps\python3.8\python.exe`

Notes

Under the assumption that it is likely an oversight, the typical ".exe" suffix is automatically appended if it is not specified. As such if you are looking for an executable without the ".exe" suffix, `o_which()` may return something other than what you are looking for

Reference

Dependencies

This function is implemented with a Python script called via scriptPro.

Supported On

VEE 7.0+, Windows 7+

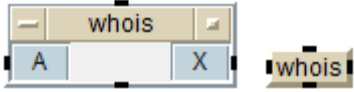
See Also

[findFile](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

whois(A)

Determines the registration information for a given domain.



Syntax

o_whois(A)

A scalar Text - domain name or IP address

return: array Text - registration information

Usage

Queries the web for registration information for a given domain or IP address.

Location

menu: veeos ==> Network ==> whois

library: [netLib](#)

Example

o_whois("veeos.org") will determine the domain registration of veeos.org

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[longLat](#), [currentTemp](#), [currentHumidity](#), [whois](#), [wget](#), [psftp](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

winVersion()

Find the version of Windows which is currently running.



Syntax

o_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

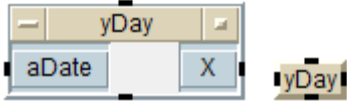
VEE 7.0+. Windows 3.1+

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

yDay(aDate)

Converts timestamp to day of the year.



Syntax

o_yDay(aDate)

aDate: scalar Timestamp

return: Int

Usage

Determines the day of the year given specified timestamp.

Location

menu: veeos ==> Date/Time ==> yDay

library: [veeosLib](#)

Example

o_yDay(now()) returns the current day 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. January 1 is taken as day 1.

Reference

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

Supported On

VEE 7.0+, All platforms.

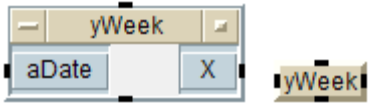
See Also

[time](#), [yWeek](#), [timeZone](#), [utcNow](#), [unixNow](#), [ywTodate](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

yWeek(aDate)

Converts timestamp to week of the year.



Syntax

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

Usage

Determines the week of the year given specified timestamp.

Location

menu: veeos ==> Date/Time ==> yWeek

library: [veeosLib](#)

Example

`o_yWeek(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.

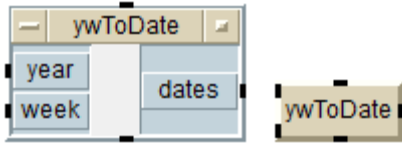
See Also

[time](#), [yDay](#), [timeZone](#), [utcNow](#), [unixNow](#), [ywToDate](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

ywToDate(year,week)

Converts year and week to timestamps at start and end of the week.



Syntax

`o_ywToDate(year,week)`

year: Int

week: Int

return: array Timestamp with the start and end time for the specified week.

Usage

Use this when you have use for the start and stop timestamps for a given week.

Location

menu: veeos ==> Date/Time ==> ywToDate

library: [veeosLib](#)

Example

`o_ywToDate(2024,14)` returns the timestamps of the start and stop of the 14th week of 2024.

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. Derived from original work 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

[time](#), [yDay](#), [yWeek](#), [timeZone](#), [utcNow](#), [unixNow](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:32.

zeroCrossings(a)

Determines the points at which a function crosses the X axis: the zero crossings.



Syntax

o_zeroCrossings(a)

a: array 1D real OR Waveform

return: array Real - the estimated points where the array or waveform cross the X axis

 waveform input > output is in time units

 mapped array input > output is in mapped units

 unmapped array > output is in index units

Usage

The real roots of a function lie on the X axis and can be estimated by this function since it locates zero crossings.

Location

menu: veeos ==> Signal Processing ==> Zero Crossings

library: [sigLib](#)

Example

see veeos example

Notes

For maximum robustness the underlying algorithm uses a grid search and then linear interpolation to find the zero crossing. Either direction of crossing is found.

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:33.

zip(zipfile,source)

Uses standard ZIP to bundle whatever source files are specified.



Syntax

o_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

o_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+

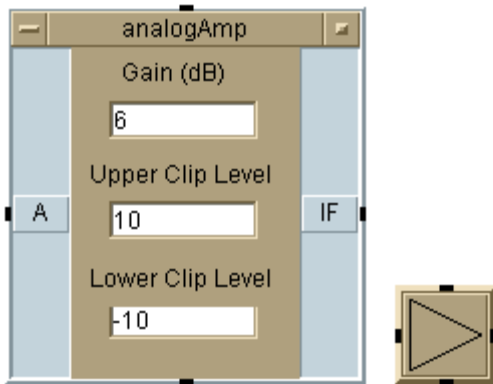
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

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

Underlying function is aAmp.

Supported On

VEE 7.0+, all platforms

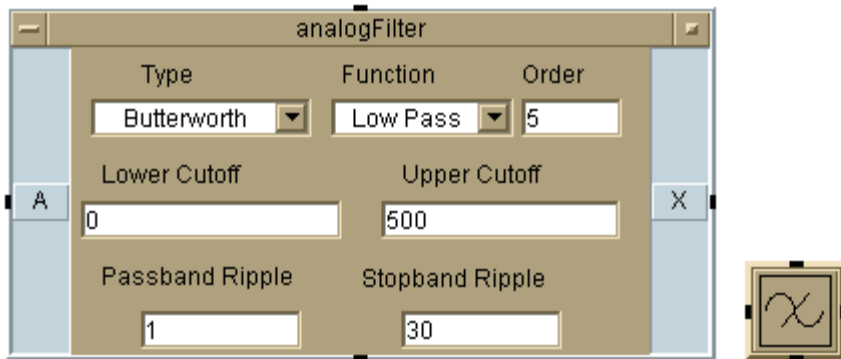
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 26 July 2021 09:41.

analogFilter

Configurable analog classical filter.



Syntax

None- object only.

Usage

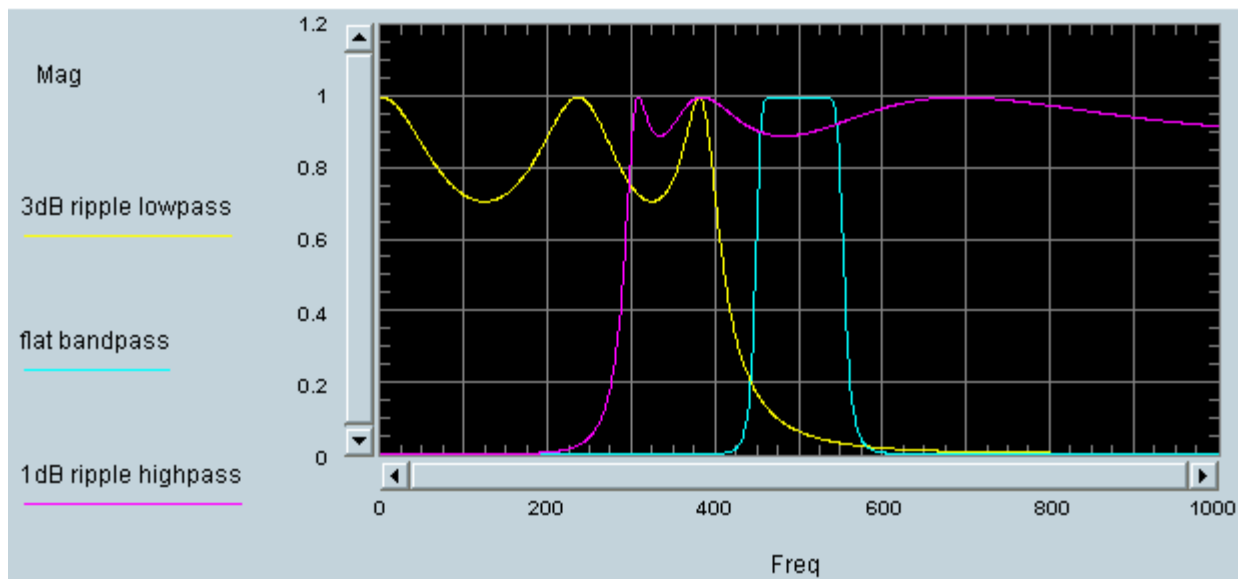
Drop this down in order to implement analog filtering on a signal. Filter type, order, cutoff and so on are configurable.

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

VEE 7.0+, all platforms

See Also

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

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 bitsream 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

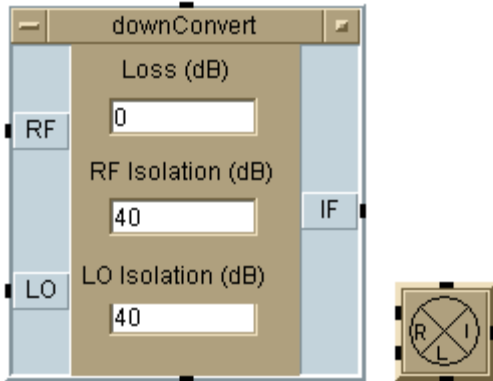
VEE 7.0+, all platforms

See Also

[numberStream](#)

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

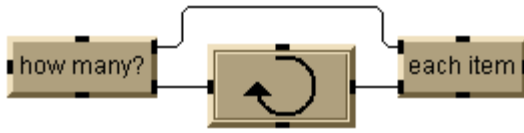
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

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

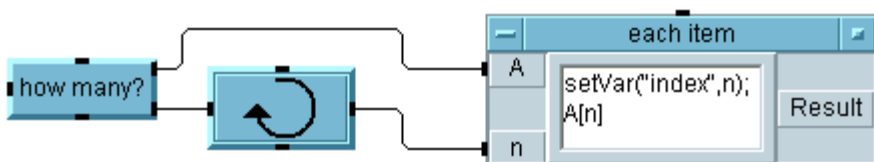
library: *none- objects only*

Example

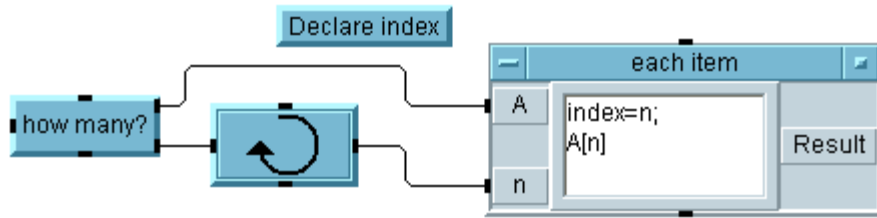
Notes

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

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

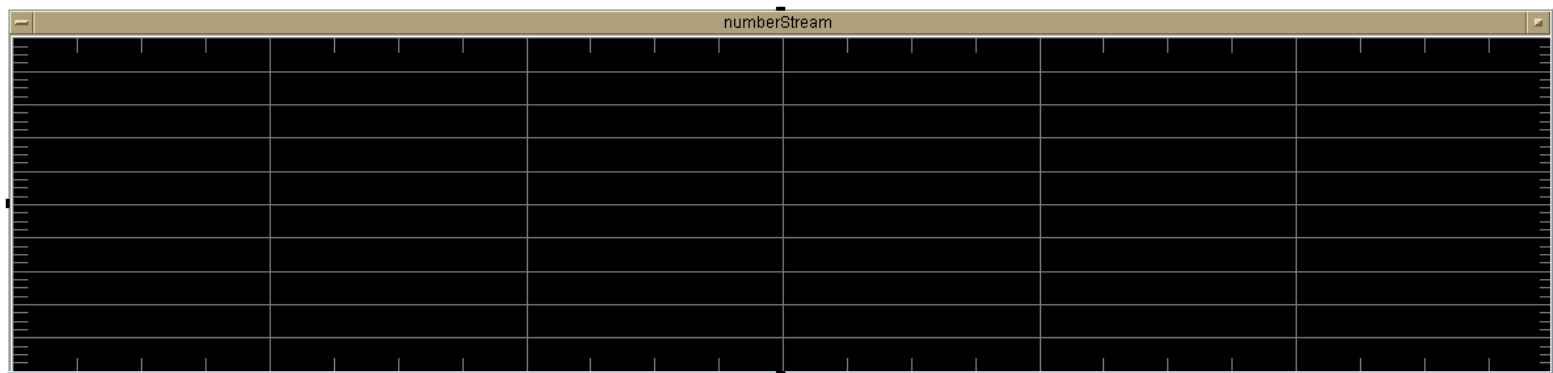
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

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](#)

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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

referenceGuide

Provides local viewable version of the entire reference guide.

Syntax

None- object only.

Usage

This can be dropped down in any program if you like, and is called by the [help](#) function

Location

menu: none

library: none

Example

Notes

At present this uses an HTML activeX object to view a local copy of the documentation. In the future something more sophisticated should be developed.

Reference

Dependencies

Supported On

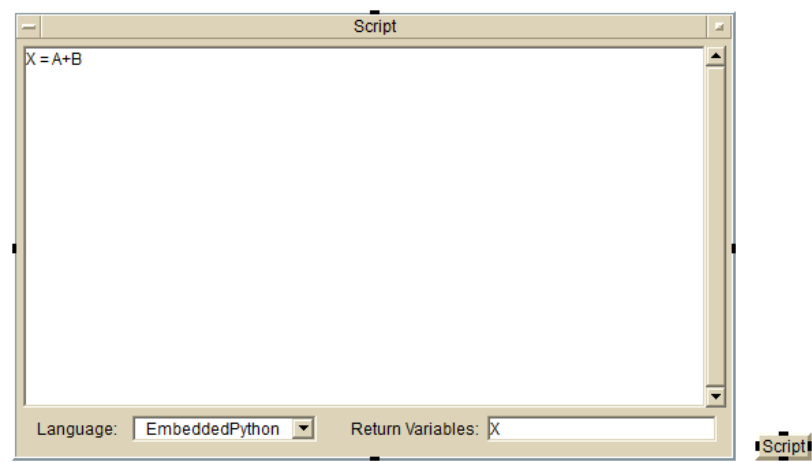
VEE 7.0+, all platforms

See Also

[help](#)

script

The Script object provides a simple wrapper for executing arbitrary scripts. This object is meant to take care of all the overhead associated with actually running the script so that executing scripts is as simple as possible for the end user.



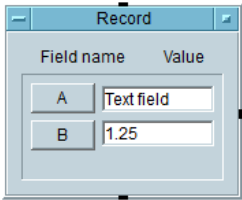
Syntax

None- object only.

Usage

There are 4 pieces of information that the user provides:

1. **The script** as entered in the provided text area. *Note that this script is NOT a VEE script so the syntax requirements must follow the language in which the script is written. Since editing in the script object is highly limited, it is recommended that for anything more than very simple scripts editing be done in a proper editor and then copied into the script object.*
2. The **Language** - The language in which the script is written. The default is an embedded python that is provided as part of VEEOS. For any other choice the user needs to install and configure that desired language before accessing it through the script object. See the Supported Languages page for more details.
3. **Input Variables** - the input variables as a **record**. Here the **variables** are defined by the field names and the *values* by the field values. *Note that records are NOT supported as field values. Also note that as with scripts, the variable names must conform with the requirements of teh chosen language.*



4. Choice of **Output Variables** - a comma-delimited list of variables. *Note that these variables must conform with the script and language requirements. In particular most languages are case-sensitive so be sure and follow language rules.*

And there is a single **output** that returns the results of the script in the form of a record. As with the Input Variables, this record will have fields associated with each Output Variable selected. There will typically be other fields also such as a status or associated error messages.

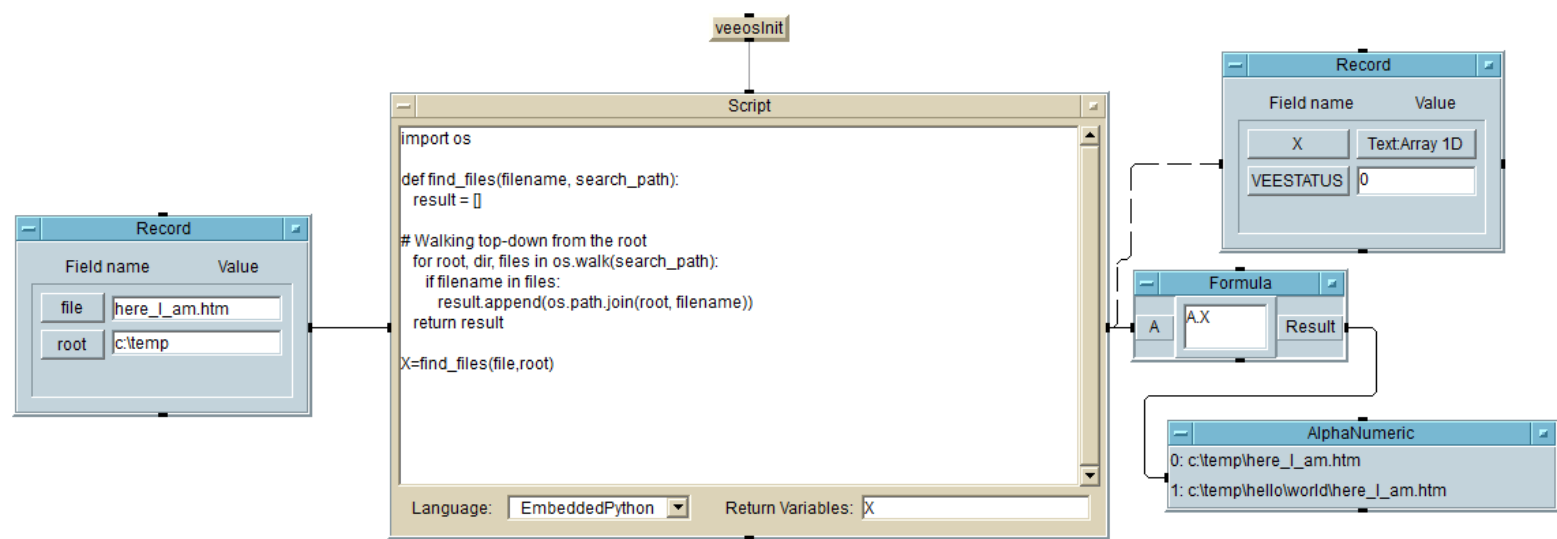
Location

menu: veeos ==> Scripting ==> Script

library: scriptLib

Example

Note the input and output variable records.



Notes

Reference

Dependencies

This function uses [ScriptPro](#) for the actual execution

Supported On

VEE 7.0+, Windows 7+

See Also

[ScriptPro](#)

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](#), [toEngr](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

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](#), [toEngr](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

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 "00000000000000001001011101110111"

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](#), [toEngr](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

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
"000000000000000000000111110110010010010000110100001010101111110011"

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

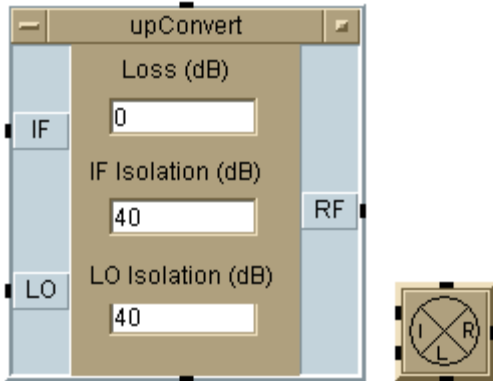
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

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

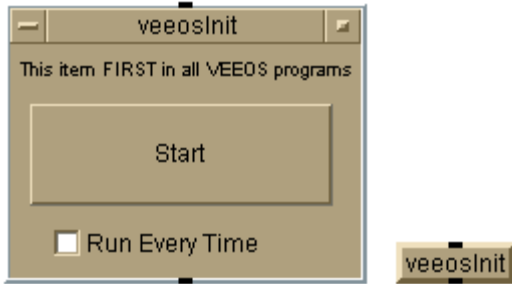
See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

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](#)

startFreq(a)

Extracts the beginning frequency of a spectrum.

Function only- no corresponding object

Syntax

o_startFreq(a)

a: spectrum

return: Real64- *the start frequency*

Usage

This is the same as using the Unbuild Spectrum object but has the advantage of being useful inside expressions.

Location

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

library: [sigLib](#)

Example

Notes

Reference

Dependencies

Supported On

VEE7.0+, all platforms

See Also

[dimSizes](#), [numDims](#), [arrayData](#), [stopFreq](#), [timeSpan](#)

stopFreq(a)

Extracts the end frequency from a spectrum.

Function only- no corresponding object

Syntax

o_stopFreq(a)
a: spectrum
return: Real64- *the stop frequency*

Usage

This is the same as using the Unbuild Spectrum object but has the advantage of being useful inside expressions.

Location

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

library: [sigLib](#)

Example

Notes

Reference

Dependencies

Supported On

VEE7.0+, all platforms

See Also

[dimSizes](#), [numDims](#), [arrayData](#), [startFreq](#), [stopFreq](#), [timeSpan](#)

timeSpan(a)

Extracts the time span from a waveform.

Function only- no corresponding object

Syntax

o_timeSpan(a)

a: waveform or spectrum

return: Real64- *the timespan*

Usage

This is the same as using the Unbuild Waveform object but has the advantage of being useful inside expressions.

Location

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

library: [sigLib](#)

Example

Notes

Reference

Dependencies

Supported On

VEE7.0+, all platforms

See Also

[dimSizes](#), [numDims](#), [arrayData](#), [startFreq](#), [stopFreq](#)

aAmp(a,specs)

Generic analog amplifier

Function only- no corresponding object

Syntax

o_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](#), [aRectifier](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 28 July 2021 21:03.

aFilter(a,specs)

Generic classical analog filter

Function only- no corresponding object

Syntax

o_aFilter(a,specs)

a: waveform - input signal to be filtered

b: scalar record - specifications for the filter, with fields

type: scalar int - 0 = butterworth, 1= chebychev

Function: scalar int - 0= low pass, 1= band pass, 2= band stop, 3= high pass

Fl: real scalar - lower cutoff frequency

Fu: real scalar - Upper cutoff frequency

Pr: real scalar - passband ripple in dB

Sr: real scalar - stopband ripple in dB

order: scalar int - classical filter order

return: waveform - filtered signal

Usage

Use this to filter a given signal using classical polynomial-based filter functions.

Location

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

library: sigLib

Example

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

aMixer(a,b,specs)

Generic analog mixer

Function only- no corresponding object

Syntax

o_aMixer(a,b,specs)

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](#), [aRectifier](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

aRectifier(a,specs)

Generic analog rectifier

Function only- no corresponding object

Syntax

o_aRectifier(a,specs)

a: waveform - input signal

specs: scalar record - specifications including fields:

 pol: scalar enum - polarity ("pos" or "neg")

 type: scalar enum - "full" or "half"

return: waveform - output signal

Usage

Simulates an ideal analog rectifier, either full or half-wave and either positive or negative polarity. Used inside various functions but can also be used directly if desired.

Location

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

library: [sigLib](#)

Example

Notes

Reference

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

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

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

arrayData(a)

Extracts the array of values from a waveform.

Function only- no corresponding object

Syntax

```
o_waveArray(a)  
  a: waveform  
  return: array Real64- the array of values of this waveform
```

Usage

This is the same as using the Unbuild Waveform object but has the advantage of being useful inside expressions.

Location

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

library: [sigLib](#)

Example

Notes

Reference

Dependencies

Supported On

VEE7.0+, all platforms

See Also

[dimSizes](#), [numDims](#), [startFreq](#), [stopFreq](#), [specArray](#), [timeSpan](#)

butterPoly(specs)

Generates Butterworth polynomials

Function only- no corresponding object

Syntax

o_butterPoly(specs)

specs: scalar record with fields:

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

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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

chebyPoly(specs)

Generates Chebbychev polynomials

Function only- no corresponding object

Syntax

o_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:

$$\begin{aligned} &(-1*(-8.945836220019011, 99.01071120033895)*-1*(-23.42050328179965, \\ &61.19198477210937)*-1*(-28.94933412356129, 6.374444838711282E-015)*-1*(- \\ &23.42050328179965, -61.19198477210936)*-1*(-8.945836220019015, - \\ &99.01071120033895))/((S-(-8.945836220019011, 99.01071120033895))*(S-(- \\ &23.42050328179965, 61.19198477210937))*(S-(-28.94933412356129, \\ &6.374444838711282E-015))*(S-(-23.42050328179965, -61.19198477210936))*(S-(- \\ &8.945836220019015, -99.01071120033895))) \end{aligned}$$

Notes

Reference

Dependencies

Supported On

See Also

[butterPoly](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

comma2point(a)

Replaces commas with points.

Function only- no corresponding object

Syntax

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

library: [dataLib](#)

Example

o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 26 July 2021 09:41.

dimSizes(a)

Returns the size(s) of the dimensions for an array.

Function only- no corresponding object

Syntax

o_dimSizes(a)

a: any - any value

return: Int32 array - *the size of each dimension*

Usage

This is the same as using the Unbuild Array object but has the advantage of being useful inside expressions.

Location

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

library: [dataLib](#)

Example

`A=o_dimSizes(A);A[0]` returns the size of the first dimension of A

Notes

Reference

Dependencies

Supported On

VEE7.0+, all platforms

See Also

[numDims](#), [arrayData](#), [startFreq](#), [stopFreq](#), [timeSpan](#)

dtmfCode(char)

Generates the DTMF frequency pair that defines the given input character.

Function only- no corresponding object

Syntax

o_dtmfCode(char)

char: scalar string - single character

return: real64 array - 2 elements defining the two frequencies

Usage

This function allows one to easily find the two frequencies that define a given character in DTMF

Location

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

library: sigLib

Example

o_dtmfCode("A") returns [697,1633]

o_dtmfCode("#") returns [941,1477]

Notes

Reference

[DTMF is a very simple standard](#) consisting of two audio frequencies and developed to enable simple data signaling over telephone lines.

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[dtmfDecode](#), [dtmfPulse](#), [dtmfRcvPulse](#), [dtmfReceiver](#), [dtmfSigGen](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

dtmfDecode(freqs)

Converts a pair of DTMF frequencies into the corresponding character

Function only- no corresponding object

Syntax

dtmfDecode(freqs)

freqs: real64 array - set of two frequencies

return: char - character that corresponds to the given frequencies.

Usage

Once one has determined the dual tones that define a given DTMF pulse, the next step is to decode this into the corresponding character. This object accomplishes that task.

Location

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

library: sigLib

Example

o_dtmfCode([697,1633]) returns "A"

o_dtmfCode([941,1477]) returns "#"

Notes

Reference

[DTMF is a very simple standard](#) consisting of two audio frequencies and developed to enable simple data signaling over telephone lines.

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[dtmfCode](#), [dtmfPulse](#), [dtmfRcvPulse](#), [dtmfReceiver](#), [dtmfSigGen](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

dtmfRcvPulse(dtmfPulse)

Converts a DTMF signal pulse into the corresponding character

Function only- no corresponding object

Syntax

dtmfRcvPulse(dtmfPulse)

dtmfPulse: waveform - any DTMF-coded signal pulse

return: char - character that corresponds to the given pulse.

Usage

This function takes a general two-tone DTMF pulse with or without noise and other signals and derives a best estimate of the intended encoded character.

Location

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

library: sigLib

Example

Notes

This function implements a DTMF receiver using a tuned narrow-band receiver and as such is highly noise tolerant. This function is used within the more general dtmfReceiver but may also be used as-is if you have a single DTMF pulse.

Reference

[DTMF is a very simple standard](#) consisting of two audio frequencies and developed to enable simple data signaling over telephone lines.

Dependencies

Supported On

VEE 7.0+, all platforms

See Also

[dtmfCode](#), [dtmfDecode](#), [dtmfPulse](#), [dtmfReceiver](#), [dtmfSigGen](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

numDims(a)

Returns the dimensionality of a value.

Function only- no corresponding object

Syntax

o_numDims(a)

a: any - any value

return: Int32 - *the number of dimensions. 0 for scalar 1 or more for an array*

Usage

This is the same as using the Unbuild Array object but has the advantage of being useful inside expressions.

Location

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

library: [dataLib](#)

Example

`(o_numDims(A)?A:[A])` returns the input unchanged if it is an array, but converts a scalar input into a 1D array. Hence regardless of the input, the result is an array.

Notes

Reference

Dependencies

Supported On

VEE7.0+, all platforms

See Also

[dimSizes](#), [arrayData](#), [startFreq](#), [stopFreq](#), [timeSpan](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

point2comma(a)

Replaces points with commas.

Function only- no corresponding object

Syntax

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

library: [dataLib](#)

Example

o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

raiseError(message,code)

Provides a callable Raise Error capability.

Function only- no corresponding object

Syntax

o_raiseError(message,code)

message: scalar text - error message to be displayed

code: int16 scalar - error code to be displayed

return: none

Usage

VEE natively provides a Raise Error object for user-defined error states. This is used to inform user of the error and to halt execution. This function encapsulates the Raise Error object in a function so that it can be called in formulas or anywhere else.

Location

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

library: sysLib

Example

A formula with

```
A=10;(A<=9?"you did well": o_raiseError("you messed up",666) )
```

raises an error and halts operation since A is out of a defined range.

Notes

Reference

Dependencies

none

Supported On

VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

specArray(a)

Extracts the array of values from a spectrum.

Function only- no corresponding object

Syntax

o_specArray(a)

a: spectrum

return: array Complex- *the array of values of this spectrum*

Usage

This is the same as using the Unbuild Spectrum object but has the advantage of being useful inside expressions.

Location

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

library: [sigLib](#)

Example

Notes

Reference

Dependencies

Supported On

VEE7.0+, all platforms

See Also

[dimSizes](#), [numDims](#), [startFreq](#), [stopFreq](#), [timeSpan](#), [waveArray](#)

Xcoord(a)

Returns the X value of a coordinate.

Function only- no corresponding object

Syntax

o_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), o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 26 July 2021 09:41.

Ycoord(a)

Returns the Y value of a coordinate.

Function only- no corresponding object

Syntax

o_Ycoord(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), o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

Zcoord(a)

Returns the Z value of a coordinate.

Function only- no corresponding object

Syntax

o_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), o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:33.

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. Directory from which VEE is launched - searched for veeos.rc
5. Directory where the VEE program is located - 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

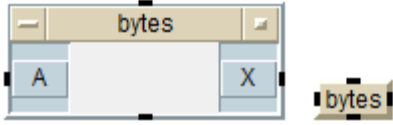
VEE 7.0+, all platforms

See Also

[veeosInit](#)

bytes(a)

Converts data to the bytes as written on disk.



Syntax

`o_bytes(a)`

a: scalar any

return: an array of bytes matching what would be written in binary on disk

Usage

At times it is useful to examine the details of binary data. Traditionally in VEE the process would be to write the data to a file on disk in binary format and then read in the bytes. Using file IO like this can be fragile and is slow at best. This object bypasses the file IO and creates the bytes directly within VEE. And since it does not use loops, it is very fast.

Location

menu: veeos ==> Data ==> bytes

library: dataLib

Example

```
o_bytes(asReal64(pi)) returns 24 45 68 84 251 33 9 64
o_bytes(asUInt8(5)) returns 5
o_bytes(asInt16(5)) returns 5 0
```

Notes

This function is likely only useful in rare cases, but it was done largely as a challenge to see if it was feasible to accomplish without using loops.

Reference

Dependencies

Supported On

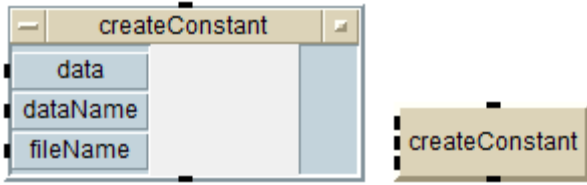
VEE 7.0+, all platforms

See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

createConstant(data,dataName,fileName)

Creates a non-editable importable data object.



Syntax

`o_createConstant(data, dataName, fileName)`

data: any valid VEE data

dataName: Text - the name assigned to the constant object that is created

fileName: Text - name of the file into which the constant object is placed

return: none

Usage

This automates the process of creating a secured object containing a data constant.

Location

menu: veeos ==> Data ==> createConstant

library: [dataLib](#)

Example

`o_createConstant(8.85418782E-012 , "eps0", "c:/temp/eps0.vxe")` creates a file containing a single object that in turn contains the value 8.85418782E-012



Once imported this object looks like: . Note that this is read-only.

Notes

This object does NOT create a constant for use within the program calling

createConstant() but rather in a file on disk.

Reference

Dependencies

Supported On

VEE 7.0+, Windows 7+

See Also

[setVar](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:20.

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

`o_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

```
o_concat2D ( 

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

 , 

|   |   |
|---|---|
| 3 | 4 |
| 3 | 4 |
| 3 | 4 |

 ) yields 

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 3 | 3 | 4 |
| 4 | 5 | 6 | 3 | 4 |
| 7 | 8 | 9 | 3 | 4 |


```

```
o_concat2D ( 

|   |   |
|---|---|
| 1 | 2 |
| 4 | 5 |
| 7 | 8 |

 , 

|   |   |
|---|---|
| 3 | 4 |
| 3 | 4 |
| 3 | 4 |

 ) yields 

|   |   |
|---|---|
| 1 | 2 |
| 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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

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

o_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](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 22 July 2021 09:45.

findEngine(engine)

Determines the location(s) of a scripting engine on a local computer.

Function only - no corresponding object.

Syntax

`o_findEngine(engine)`

engine: Text - the name of the script engine to find

return: scalar or array Text - the full path(s) to the file(s) found that match the specified name

Usage

Use this to find potential locations of a script engine on the local computer.

Location

menu: *none*

library: [scriptLib](#)

Example

The typical usage of *findEngine* occurs within the [scriptConfig](#) object to assist the user in finding installed versions of a given engine.

`o_findEngine("python")` finds several possibilities and returns an array

```
C:\apps\python3.8\python.exe  
C:\Users\stan\AppData\Local\Microsoft\WindowsApps\python.exe  
C:\Program Files (x86)\Agilent\VEE Pro 7.0\veeos\python\python.exe
```

Notes

This object never needs to be run but rather is included as a convenience. If the user knows where their install is, then that can be configured. One possible use is to find old stale installs of a given scripting engine. Since a typical system has many files, *findEngine()* may take a while to execute.

Reference

Dependencies

This function is implemented with a Python script called via [scriptPro](#).

Supported On

VEE 7.0+, Windows 7+

See Also

[which](#), [findFile](#)

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

formatScript(rawScript,engine,returnVars)

An aid to formulate the final script to be uploaded.

Function only - no corresponding object.

Syntax

o_formatScript(rawScript,engine,returnVars)

rawScript: Text array - the script itself

engine: Text - the name of the script engine

returnVars: Text - comma-delimited list of variables to be returned

return: array Text - the full script

Usage

Used by the script object to formulate the final script to be uploaded to the server. Adds the appropriate *shebang* and *returnToVEE* lines.

Note that this script is NOT a VEE script so the syntax requirements must follow the language in which the script is written. Since editing in the script object is highly limited, it is recommended that for anything more than very simple scripts editing be done in a proper editor and then copied into the script object.

Location

menu: *none*

library: [scriptLib](#)

Example

Notes

This object never needs to be run but rather is used internally. However it may also be useful when manually creating scripts.

Reference

Dependencies

This function is implemented with a Python script called via [scriptPro](#).

Supported On

VEE 7.0+, Windows 7+

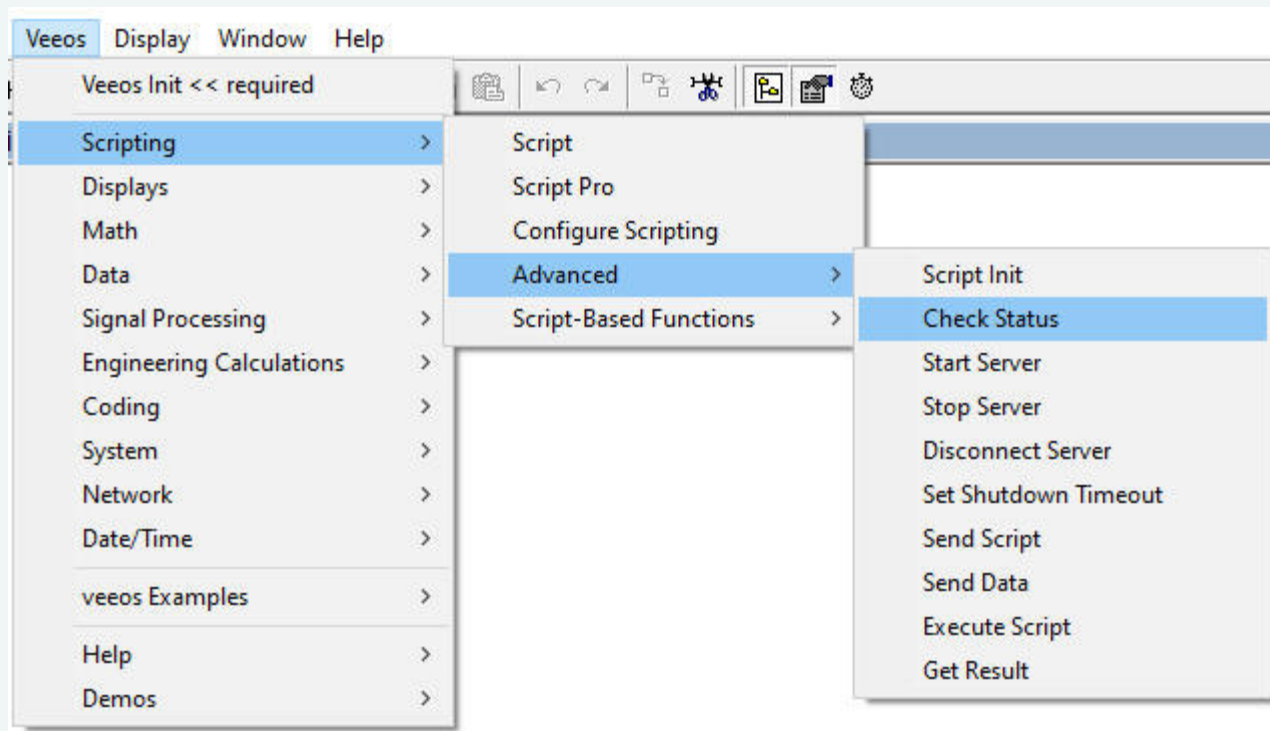
See Also

© 2015-2021. All Rights Reserved. Stan Bischof (stan@worldbadminton.com). Last updated 09 August 2021 18:25.

VEEOpenSource Scripting

The scripting library provides the ability to transparently access programs written in a variety of scripting languages. This includes sharing data in both directions with these programs. This enables the VEE user to utilize the full power of modern scripting languages. Scripting was developed by a [University of Calgary](#) team.

VEEOS Scripting enables access to world-class numerical and symbolic computation, as well as statistics, general programming, system analysis and much more.



The Script Server

Written in Python, the script server is the heart of VEEOS scripting capability. This small light code communicates with VEE and in turn has control and communications with the various language engines. The script server comprises three separate pieces of Python code along with an embedded minimal Python installation. This work is open source and available from the Github project site.

- server.py - handles all communications between VEE and script languages. Must always be present for scripting. By default shuts itself down when not being used.
- VEEParser.py - translates between VEE container format and Python variables. Is imported as needed by the server.
- bridgescripts.py - communicates between the python server and python (embedded

or installed) used to run user scripts. Is started by the server as needed.

Supported Languages

With the exception of the embedded Python interpreter, the user is expected to install and control their own selected version of the various languages. The server is written to be compatible with almost any version of the languages. For details related to each language see the below documentation.

- [Python](#) *NOTE: Python is case sensitive*
 - Embedded - provided in the VEEOS library. This copy of python is the default and includes the standard libraries so is capable of executing most "vanilla" python code. If specialized libraries for image processing or anything else are needed, then it is appropriate to install Python and use the installed version instead of this embedded version.
 - [Installed](#) - more or less any distribution of Python newer than 2.5 will work. Use the RC file to configure this version of python.
- [Octave](#)
- [R](#) *NOTE: All variables must be arrays (no scalars)*
- [Maxima](#)
- [Fiji \(ImageJ\)](#)
- SPICE

Scripting Configuration

After a scripting language is installed it is necessary to configure VEEOS such that the script server can automatically exercise the installed scripting language. In order to use the [Script](#) and [ScriptPro](#) capabilities of VEEOS, the configuration is done via a plain text (RC) file. This file may be manually edited or it may be edited using the [ScriptConfig](#) tool. In either case VEEOS reads this file automatically.

The relevant section of the default RC file looks like this:

```
# Scripting - modify to use server other than default
# gvar scriptHost Text <remote server IP or name>
gvar scriptPort Int32 65433
gvar scriptHost Text localhost

# Script options - uncomment as needed
#gvar scriptTimeout Int32 2
#gvar scriptExecutionMode Text DEBUG
#gvar scriptLoopDelay Real64 100m
#gvar scriptServerShutdownTimeout Int32 120

# Script engine install details - uncomment as needed
```



```
#gvar scriptPythonPath Text "VEER00T/veeos/python/python.exe"
#gvar scriptInstalledPythonPath Text "full path"
#gvar scriptOctavePath Text "full path"
#gvar scriptRPath Text "full path"
#gvar scriptMaximaPath Text "full path"
#gvar scriptFijiPath Text "full path"
#gvar scriptSpicePath Text "full path"
```

Note that the system-wide RC file is <vee install dir>/veeos/etc/veeos.rc although other hierarchical locations work also. See the [Bootstrap](#) documentation. The ScriptConfig tool specifically edits the user's RC file <home directory>/veeos.rc.

Embedded Python

As part of the VEEOS library, a copy of Python is included. This is embeddable since it is compact and fast and is version 3.8 since that way Windows 7 can be supported. If Windows XP needs to be supported then the user can install Python 2.7 or 3.4 and configure scripting to use that install rather than the embedded python.

Remote Servers

Credits

Scripting in VEE was developed as a [student project](#) at the [University of Calgary](#) in summer 2021. Led by Faculty Advisor [Dr. Richard Wilson](#) with the University's Cumming School of Medicine, students Faisal Hossain and Khaled Elmalawany from Computer Sciences spent many hours designing and developing the scripting interface as an extension of VEE. As a neuroscientist, Dr Wilson has long been a classical VEE user: a non-programmer who nevertheless needs to automate various tasks. The most recent work has been image processing using ImageJ as part of his basic research. By sponsoring the scripting effort, Dr. Wilson gained access to many powerful tools including ImageJ (via the Fiji interface) from within his chosen VEE environment. [Stan Bischof](#) provided technical support as well as integration into VEEOS. Special thanks to Keysight CEO Ron Nersesian who very graciously granted VEE licenses for use on this project.

Script Engine: Python

[Python](#) is a programming language that lets you work more quickly and integrate your systems more effectively. Python is very widely used and well supported by the user community.



Since it is likely that the embedded Python engine does not have the extensions and utilities needed for many Python tasks, the user is free to install most any Python distribution, such as Jupyter, SciPy, SymPy. Once configured, the script server will automatically use the designated Python.

- Recommended Versions
- Installation
- Configuration

Recommended Versions

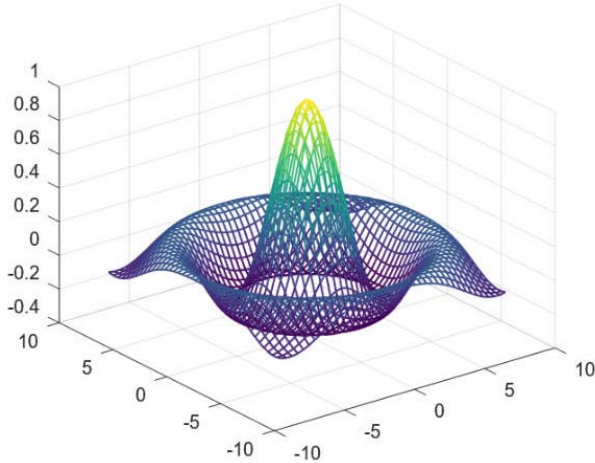
Installation

Configuration

© 2015-2021. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 10 August 2021 09:17.

Script Engine: Octave

[GNU Octave](#) is a high-level language, primarily intended for numerical computations. Octave closely resembles Matlab.



To use Octave scripting the end user needs to first install Octave and configure VEEOS.

- Recommended Versions
- Installation
- Configuration

Recommended Versions

Installation

Configuration

© 2015-2021. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 10 August 2021 09:17.

Script Engine: R

R is a language and environment for statistical computing and graphics. R closely resembles SPlus.



To use R scripting the end user needs to first install R and configure VEEOS.

- Recommended Versions
 - Installation
 - Configuration
-

Recommended Versions

Installation

Configuration

© 2015-2021. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](mailto:stan@worldbadminton.com). Last updated 10 August 2021 09:17.

Script Engine: Maxima

[Maxima](#) is a premier algebraic engine, allowing a very broad range of symbolic and numerical math.

```
(%i3) integrate ( 1 / (1 + x^4), x);  
  
              2              2  
      log(x  + sqrt(2) x + 1)  log(x  - sqrt(2) x + 1)  
(%o3) -----  
              4 sqrt(2)              4 sqrt(2)  
              2 x + sqrt(2)          2 x - sqrt(2)  
      atan(-----)  atan(-----)  
              sqrt(2)              sqrt(2)  
      + ----- + -----  
              2 sqrt(2)          2 sqrt(2)
```

To use Maxima scripting the end user needs to first install Maxima and configure VEEOS.

- Recommended Versions
- Installation
- Configuration

Recommended Versions

Installation

Download directly from the [Maxima website](#)

Configuration

© 2015-2021. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](#). Last updated 10 August 2021 09:17.

Script Engine: Fiji

[Fiji](#) is an image processing package—a “batteries-included” distribution of [ImageJ2](#).



To use Fiji scripting the end user needs to first install Fiji and configure VEEOS.

- Recommended Versions
- Installation
- Configuration

Recommended Versions

Installation

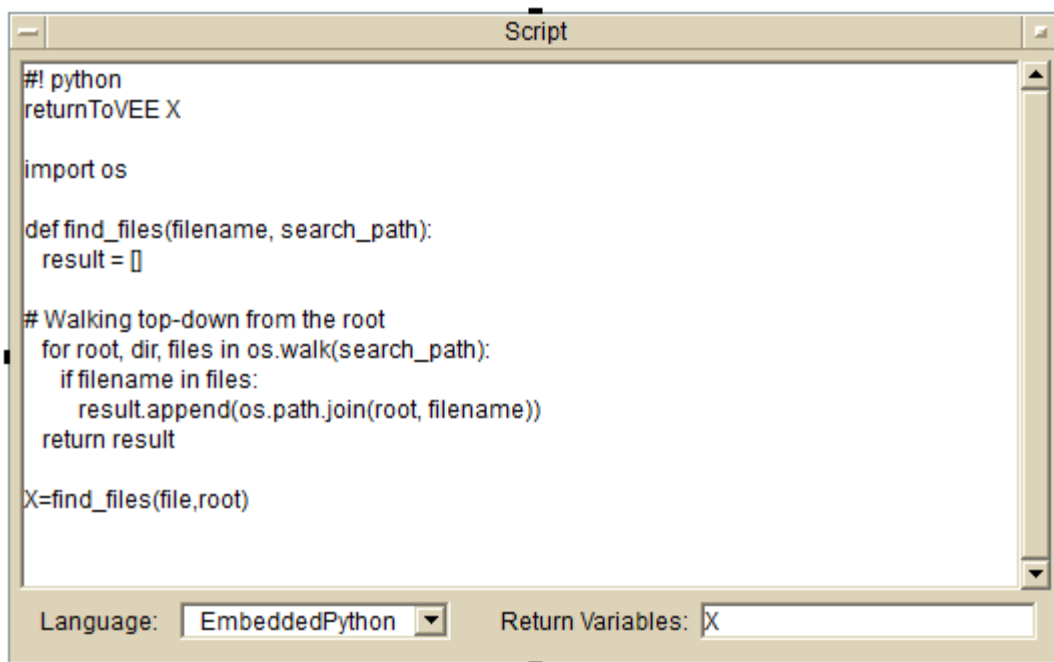
Configuration

© 2015-2021. All Rights Reserved. [Stan Bischof \(stan@worldbadminton.com\)](#). Last updated 10 August 2021 09:17.

Script Interface Project

The [University of Calgary's VEE Scripting Interface Project](#) is an undergraduate Programming Project designed to create a scripting extension to [Keysight VEE](#). Using Python and other tools as needed, the end goal is to provide a set of VEE objects that allow the end user to execute arbitrary scripts from inside VEE.

Since this is a student project, the ongoing details contain personal information and hence are available only to the project team. As results are realized they will be open-sourced and posted here for all to see.



20210228 - project initiated. It is expected that there will be a working general purpose script interface summer 2021.

202108?? - First release