Advanced Simulation Technology inc.
500 A Huntmar Park Drive
Herndon, Virginia 20170 U.S.A.
Tel. (703)471-2104 • Fax. (703)471-2108
www.asti-usa.com

ASTi
Model Builder Visual
Component Reference Guide

Document: DOC-01-MBV-CRG-1

Rev. B.1 (June, 2008)

ASTi ASTi Model Builder Visual Components Reference Guide

ASTi

500 A Huntmar Park Drive

Herndon, VA 20170

Table of Contents

1.0. Introduction

Model Builder Visual (MBV) provides the user with a sound and communications simulation model development environment. The toolset uses a visual approach to building and testing sound and communications models.

This manual provides detailed information on the Model Builder Visual components structure and the operation of each component instance.

1.1. Component Viewer

In MBV, the user has the ability to double-click on each component to open the component viewer. The component viewer provides specific component information. The component viewer is where the user sets the component values.

Schematic- The component schematic is displayed when the viewer is first opened. The schematic view shows the processing logic within the component. In this view, the component building blocks known as primitives are shown in red or blue. Red indicates that the primitive handles audio while blue indicates that the primitive handles control logic only. Moving the mouse over a primitive in the schematic view, displays the current values for that primitive. Double-click on a primitive to open it and set the values.

Data Viewer- The data viewer tab lists the primitives in a tree view in alphabetical order. The user can click on the plus sign to expand the view to include the variables within a primitive. In some cases, a variable within a primitive contains its own set of variables and can also be expanded.

Host I/O-The host I/O shows input and output variables to the components and their types, for example, basic/float 32.

AStats- The AStats display the real time statistics in a graphical view with minimum through maximum.

KStats- The KStats display the real time statistics in a graphical view with minimum through maximum.

Links Inspector- The link inspector tab displays in and out links and details including Destination, Destination Variable, Source Variable, Link Description, Type, and Location.

Info.- This may contain information on the component.

Description- This allows the user to add a description for primitives, double-click to view the values.

1.2. Components

By modifying the components, the user can construct anything from a small simulation element to a complete sound and communications audio modeling system for their application. In other words, the components are flexible and sufficiently configurable by the user to construct basic intercom systems to models that closely match the functionality of a commercial or military platform communications system.

The components are organized in the following order.

- Audio Components
- Control Components
- Engine Components
- Intercom Components
- Radio Components

Note: Not all of the features and menu items that appear on your system will be described in this manual. This manual is limited to describe the frequently used components.

For the remainder of this document each component section is organized into the following table sections:

- Inputs
  - Audio Inputs
  - Control Inputs
- Outputs
  - Audio Outputs
  - Control Outputs
- Internal parameters- These are any values that must be set as part of a component configuration that DO NOT have an external connection port.

Note: Not every component will have all the tables listed above, tables may vary depending on the complexity of the component.

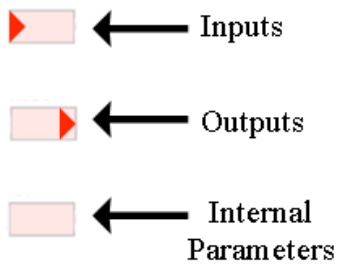


Figure 1: Components Schematic Details

While these components allow the user to construct much of the audio simulation and infrastructure for a given application, it is still necessary for the user to develop a good portion of additional simulation code to drive the constructed model in sufficient fashion to fully realize the sound and communications operations for their application.

1.3. How to Use this Reference Manual

Each component is listed with a general description and the remainder of the section is divided into table formats for the inputs, outputs, and internal parameters. The inputs and outputs are divided as Audio Inputs / Control Inputs and Audio Outputs / Control Outputs tables. Within each table the parameters are organized alphabetically for search ability.

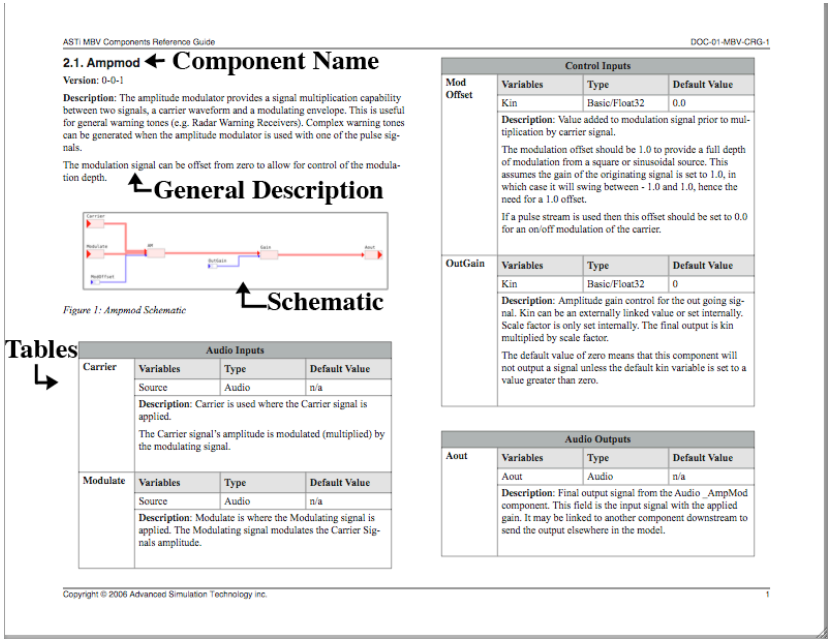


Figure 2: Reference Manual Layout

1.3.1 Creating Links in MBV

When creating links inside your model the user should be aware of a few MBV linking rules. First, you need a three-button mouse when creating models. (You can use a two button mouse by clicking both buttons simultaneously). You must middle-click on a component to link it to another component. When middle-clicking to link components, the options that appear are not the component's parameters but the settable values within the component's parameters. After you select the first object that you want to link you cannot hit the middle mouse button or you will lose the link "focus."

For example, when routing audio from the Sine Wave to the Iris you will middle-click on the Sine Wave object and select

```
from signal -> all of
```

Important! Between this step and the next step, once you select the 'all of' with the middle mouse button, you can only navigate the folders with the left mouse button before finishing the link with the middle mouse button. If you do hit another button you will lose the link "focus" you have from the sine wave.

Then middle-click on the Iris and select

```
to stereoOperator -> AudioOutA
```


2.0. Audio Components

These components can be mixed, filtered, or added into any combination of high-way channels via a feeder connection.

The following section details the audio components and the objects within them.

The audio components include:

- Alevel
- AutoDred
- Ampmod
- Filter
- Gain
- Iris
- Mixer4
- Mixer8
- MixerN
- Noise
- NoiseFilter
- Playlist
- PSound
- Psound255
- Record
- RecordGroup
- Replay
- ReplayGroup
- Resequencer
- Runway Bump
- Sequencer
- Vox
- Vox_AB
- Wave
- WavePulseMod
- WaveModulated

2.1. Alevel

Version: 0-0-1

Description: The Audio Level component takes an input audio signal and applies an input gain and calculates the resulting signal level. This component is most useful when the signal level is needed for calculation such as implementing an Automatic Level Control.

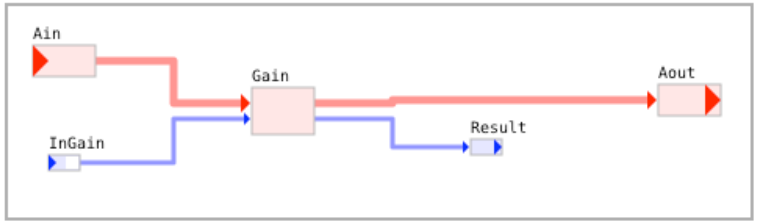


Figure 3: Alevel Schematic

Audio Inputs			
Ain	**Variables**	**Type**	**Default Value**
	Source	audio	n/a
	Description: A connection for an external audio signal whose amplitude is scaled by the value in InGain. ***Warning: Linking a 'signal->all of...' to the ain will result in an error, only link 'signal->audio' to this component.***		

Control Inputs			
InGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Amplitude gain control for the incoming signal. Kin can be an externally linked value or set internally.		

Audio Outputs			
Aout	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: Final output signal of the Alevel component. This field is the input signal with the applied gain. Alevel may be linked to another component downstream to send it elsewhere in the model.		

Control Outputs			
Result	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: The kin value of this field is set by the component. It represents the average signal level over time by averaging the absolute value of the amplitude. The formula used is specially tuned to approximate the level of voice signals rather than basic wave signals. The output is intended to provide feedback to other components or sent out to a host.		

Internal Parameters			
Gain	**Variables**	**Type**	**Default Value**
	lag	basic/float32	0.3
	Description: Provides a lag to the signal level calculation to adjust for large, sudden changes in the input signal, ranges from 0.0 to 1.0.		

2.2. AmpMod

Version: 0-0-1

Description: The Amplitude Modulator provides a signal multiplication capability between two signals, a carrier waveform and a modulating envelope. This is useful for general warning tones (e.g. Radar Warning Receivers). Complex warning tones can be generated when the Amplitude Modulator is used with one of the pulse signals.

The modulation signal can be offset from zero to allow for control of the modulation depth.

Example: Creates a 250 Hz beep every second by linking a 250 Hz sine wave to the Carrier, a 1Hz square wave to the Modulate, and setting the Mod Offset to 1.

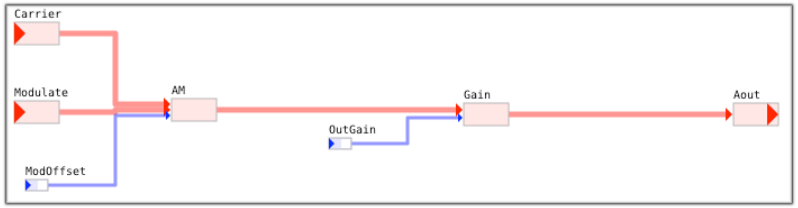


Figure 4: AmpMod Schematic

Audio Inputs			
Carrier	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: Carrier is used where the Carrier signal is applied. The Carrier signal's amplitude is modulated (multiplied) by the modulating signal.		

Audio Inputs			
Modulate	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: Modulate is where the Modulating signal is applied. The Modulating signal modulates the Carrier Signals amplitude.		

Control Inputs			
Mod Offset	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Value added to modulation signal prior to multiplication by carrier signal. The modulation offset should be 1.0 to provide a full depth of modulation from a square or sinusoidal source. This assumes the gain of the originating signal is set to 1.0, in which case it will swing between - 1.0 and 1.0, hence the need for a 1.0 offset. If a pulse stream is used then this offset should be set to 0.0 for an on/off modulation of the carrier.		
OutGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Amplitude gain control for the out going signal. kin can be an externally linked value or set internally. Scale factor is only set internally. The final output is kin multiplied by scale factor. The default value of zero means that this component will not output a signal unless the default kin variable is set to a value greater than zero.		

Audio Outputs			
Aout	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: Final output signal from the Audio _AmpMod component. This field is the input signal with the applied gain. It may be linked to another component downstream to send the output elsewhere in the model.		

Internal Parameters			
AM	**Variables**	**Type**	**Default Value**
	Decay	basic/float32	1.0
	Description: Used to soften edges of the output signal which occur when a square wave modulates a sine wave.		

2.3. AutoDred

Version: 0-0-1

Description: The Auto Daily Readiness (Auto_Dred) component implements a self-test capability for a Telestra system comprised of one or more loudspeakers and a microphone. The intent of this object is to run a number of tests that validate that the sound system is performing within a tolerance based on generating a number of noise and tone signals. The component supports the ability to test up to 12 loudspeaker channels. The microphone is not required to be of measurement quality, but should have adequate frequency response to cover the range of 100 Hz to 10 kHz. The AutoDRED component generates a series of signals, and the return input signal (from the microphone) is compared to a reference level that is set-up as part of the model installation and integration. A +/-2dB tolerance is applied to each measured signal. An output value is passed to the host to indicate whether the test passed or failed, and if the latter, what test failed.

The AutoDred component performs validation of the following system performance:

1. Overall level
2. Individual speaker levels
3. Individual speaker frequency response

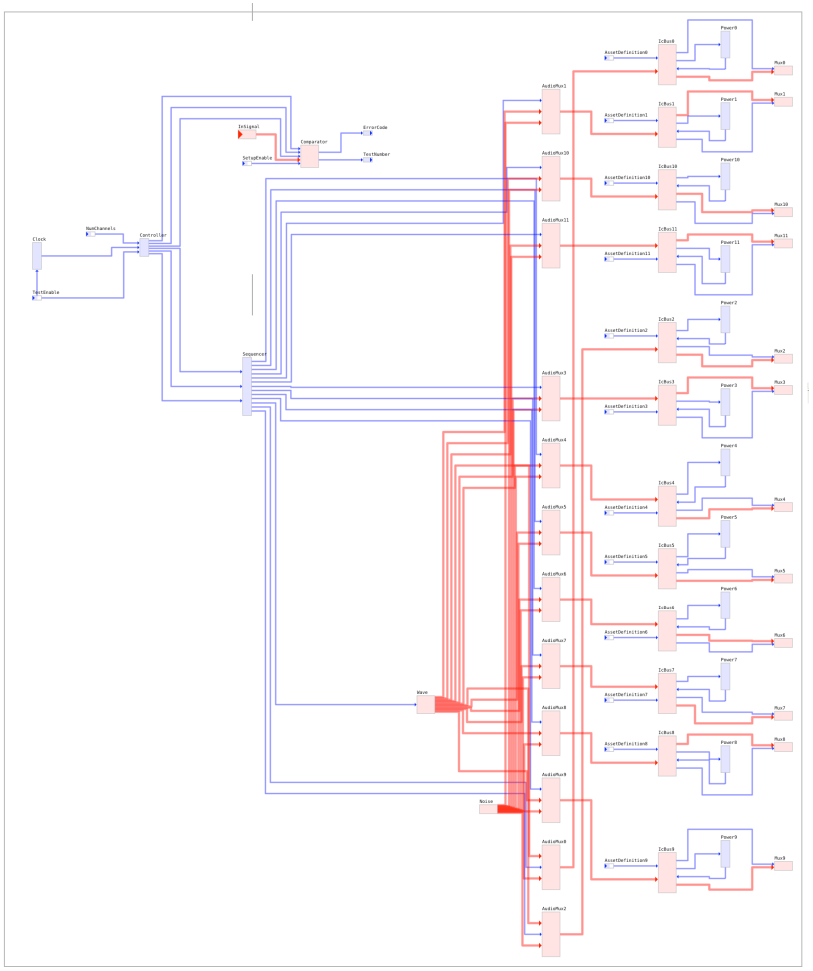


Figure 5: AutoDRED Schematic

Audio Inputs			
InSignal	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: Use InSignal to link the live audio produced by the speakers back to the component for comparison.		

Control Inputs			
Num Channels	**Variables**	**Type**	**Default Value**
	kin	basic/int32	0
	offset	basic/int32	0
	test	control/testmode	Test_Off
	testvalue	basic/int32	0
	Description: Selects the number of audio channels to be tested. This can be driven by the host or set internally. The control allows for a 'test mode' where the number of channels is read from the test value rather than the kin.		
Asset Definition 0-11	**Variables**	**Type**	**Default Value**
	kin->asset->channel	intercom/ ic_channel	n/a
	Description: These are used to assign a particular speaker channel to the component. Use the Channel Handle Editor in order to create audio channels. Auto_Dred can accept up to 12 different channels and test them one at a time.		

Control Inputs			
Setup Enable	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	invert	basic/boolean	False
	Description: Set kin to True in order to run the initial test. The setup test is used to record the speaker outputs for future tests. This control can be driven by the host or set internally.		
Test Enable	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	invert	basic/boolean	False
	Description: Set kin to True in order to run the Auto_Dred test. This must remain True throughout or the test will turn off.		

Control Outputs			
Error Code	**Variables**	**Type**	**Default Value**
	kin	basic/int32	0
	Description: Reports the status of the test process, where no fault is indicated by a value of 0, and a failure is indicated by the corresponding output channel. If an error is detected, the number will be set and held until the TestEnable flag is cleared. If more than one channel fails the test the latest failed channel will be set. Note that the channel is referred to based on which AssetDefinition is used, rather than the Channel Handle.		
Test Number	**Variables**	**Type**	**Default Value**
	kin	basic/int32	4
	Description: Reports the test that is currently running. '1' is the Overall Level Test, '2' is the Individual Speaker Level, '3' is the Individual Speaker Frequency Response, and '4' indicates that all tests are complete.		

Internal Controls			
Compara-tor	**Variables**	**Type**	**Default Value**
	level_difference_threshold	basic/float32	2.0
	Description: The level in dB used to compare a test to the reference levels. The threshold can be altered for systems that require a higher or lower tolerance. The value can only be set internally.		

2.4. Filter

Version: 0-0-1

Description: The filter component applies low-pass, band-pass, or high-pass filtering to an audio signal. The filter quality factor, roll-off frequency, and gain can be controlled by input variables from elsewhere in the model, or from the host interface. There are two filters the two-zero and the two-pole IIR filter. Possible use includes conjunction with voice signals to eliminate frequency ranges for hardware consideration.

Example: Filtering out the lower frequencies from a table microphone to eliminate unwanted vibrations.

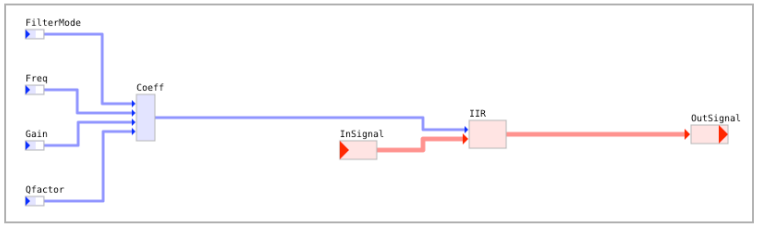


Figure 6: Filter Schematic

Audio Inputs			
InSignal	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: Audio Input Signal to be filtered.		

Control Inputs			
Filter Mode	**Variables**	**Type**	**Default Value**
	kin	audio/filter_type2	Off
	Description: Sets the type of filter for the component. Filter types are LowPass, BandPass, HighPass, and off.		
Freq	**Variables**	**Type**	**Default Value**
	kin	basic/int32	0.0
	Description: Provides the characteristic frequency for the selected filter type. The frequency is in Hertz and can be set via the host or internally. The scale factor, which multiplies the frequency in kin, can only be set internally.		
Qfactor	**Variables**	**Type**	**Default Value**
	kin	basic/int32	1.0
	Description: Amplitude gain control for the incoming signal, which can be set via the host or internally. The scale factor, which multiplies with the kin gain, can only be set internally.		

Audio Outputs			
OutSignal	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: The filtered audio signal. If the filter type is set to off, it will not produce audio.		

Internal Parameters			
Coeff	**Variables**	**Type**	**Default Value**
	filterset->freq	basic/float32	1.0
	filterset->gain	basic/float32	1.0
	filterset->gfactor	basic/float32	1.0
	Description: Apply scale factors to the respective inputs. This should not be changed in this component.		
IIR	**Variables**	**Type**	**Default Value**
	bypass	basic/boolean	False
	Description: Setting bypass to True will cause the audio signal to ignore the filter.		
	damping	basic/float32	0.999
	Description: A damping coefficient is applied to prevent instability due to rounding errors. This should not be changed in this component.		

2.5. Gain

Version: 0-0-2

Description: Gain applies amplitude gain control to an input signal. This component is useful when the same audio is output on multiple Iris channels, but requires independent gain control for each channel.

Figure 7: Gain Schematic

Audio Inputs			
Ain	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: A connection for an external audio signal whose amplitude is scaled by the value in InGain.		

Control Inputs			
InGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Amplitude gain control for the incoming signal. kin can be an externally linked value or set internally. Scale factor is only set internally. The final output is kin multiplied by scale_factor.		

Audio Outputs			
InGain	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: Final output signal from the Audio Gain component. This field is the input signal with the applied gain. It may be linked to another component downstream to send the output elsewhere in the model.		

2.6. Iris

Version: 0-0-1

Description: This component is used internally by MBV and should not be added to model canvas.

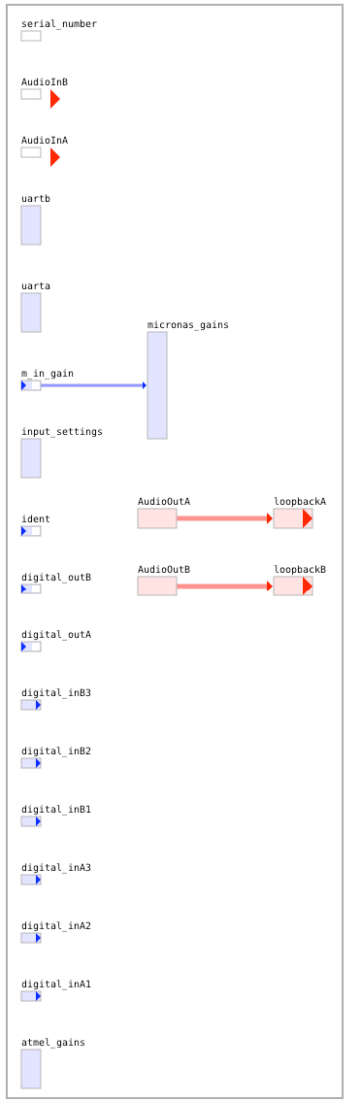


Figure 8: Iris Schematic

2.7. MarkerTone

Version:0-0-1

Description: The Marker Tone component provides the ability to add the Outer Marker, Middle Marker, Inner Marker, and Fan Marker navigational beacons to a model. The default values are set to FAA standards for marker beacons. The component provides four separate beacons but only one of the beacons can be considered at a time. This is a lower fidelity version of the Marker component, which fully simulates the position and transmission of the beacon signal. The Marker Tone will only produce the associated Morse beeps and tones.

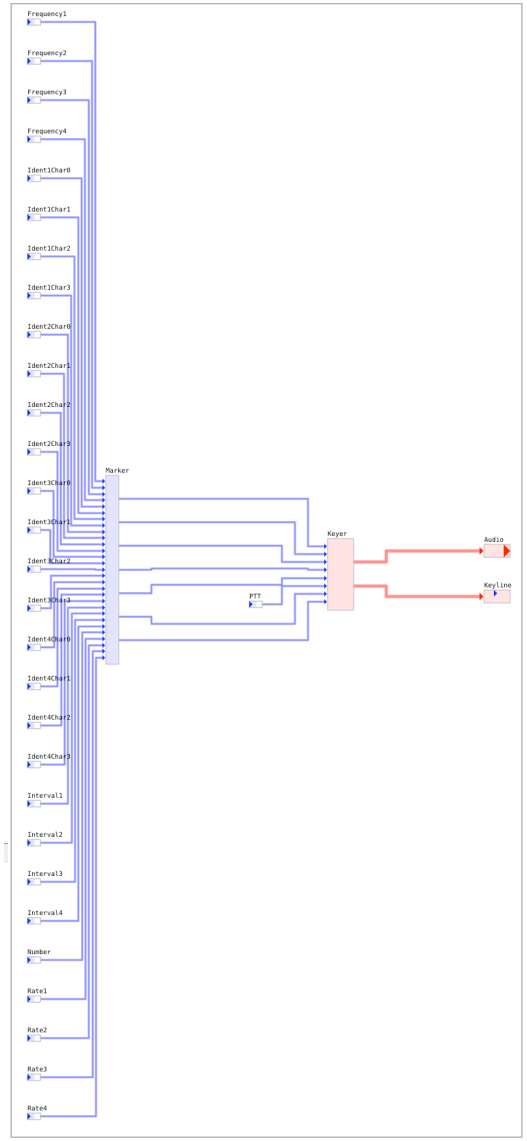


Figure 9: MarkerTone Schematic

Control Inputs			
Frequency 1	**Variables**	**Type**	**Default Value**
	kin	basic/float32	400
	Description: Provides the frequency of the Morse dash or dot for the first word (outer marker). Units are in Hertz.		
Frequency 2	**Variables**	**Type**	**Default Value**
	kin	basic/float32	13000
	Description: Provides the frequency of the Morse dash or dot for the second word (middle marker). Units are in Hertz.		
Frequency 3	**Variables**	**Type**	**Default Value**
	kin	basic/float32	3000
	Description: Provides the frequency of the Morse dash or dot for the third word (inner marker). Units are in Hertz.		
Frequency 4	**Variables**	**Type**	**Default Value**
	kin	basic/float32	3000
	Description: Provides the frequency of the Morse dash or dot for the fourth word (fan marker). Units are in Hertz.		
Ident1 Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The first letter for the outer marker. Relies on the ASCII integer equivalent for the letter, i.e. A = 65. Values for upper or lowercase can be used.		

Control Inputs			
Ident1 Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The second letter for the outer marker. Relies on the ASCII integer equivalent for the letter, i.e. S = 83. Values for upper or lowercase can be used.		
Ident1 Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The third letter for the outer marker. Relies on the ASCII integer equivalent for the letter, i.e. T = 84. Values for upper or lowercase can be used.		
Ident1 Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The fourth letter for the outer marker. Relies on the ASCII integer equivalent for the letter, i.e. i = 105. Values for upper or lowercase can be used. Note that in most situations marker beacons rely on three-character messages and leave the fourth letter as zero.		
Ident2 Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The first letter for the middle marker. Relies on the ASCII integer equivalent for the letter, i.e. A = 65. Values for upper or lowercase can be used.		

Control Inputs			
Ident2 Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The second letter for the middle marker. Relies on the ASCII integer equivalent for the letter, i.e. S = 83. Values for upper or lowercase can be used.		
Ident2 Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The third letter for the middle marker. Relies on the ASCII integer equivalent for the letter, i.e. T = 84. Values for upper or lowercase can be used.		
Ident2 Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The fourth letter for the middle marker. Relies on the ASCII integer equivalent for the letter, i.e. i = 105. Values for upper or lowercase can be used. Note that in most situations marker beacons rely on three-character messages and leave the fourth letter as zero.		
Ident3 Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The first letter for the inner marker. Relies on the ASCII integer equivalent for the letter, i.e. A = 65. Values for upper or lowercase can be used.		

Control Inputs			
Ident3 Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The second letter for the inner marker. Relies on the ASCII integer equivalent for the letter, i.e. S = 83. Values for upper or lowercase can be used.		
Ident3 Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The third letter for the inner marker. Relies on the ASCII integer equivalent for the letter, i.e. T = 84. Values for upper or lowercase can be used.		
Ident3 Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The fourth letter for the inner marker. Relies on the ASCII integer equivalent for the letter, i.e. i = 105. Values for upper or lowercase can be used. Note that in most situations marker beacons rely on three-character messages and leave the fourth letter as zero.		
Ident4 Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The first letter for the fan marker. Relies on the ASCII integer equivalent for the letter, i.e. A = 65. Values for upper or lowercase can be used.		

Control Inputs			
Ident4 Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The second letter for the fan marker. Relies on the ASCII integer equivalent for the letter, i.e. S = 83. Values for upper or lowercase can be used.		
Ident4 Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The third letter for the fan marker. Relies on the ASCII integer equivalent for the letter, i.e. T = 84. Values for upper or lowercase can be used.		
Ident4 Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The fourth letter for the fan marker. Relies on the ASCII integer equivalent for the letter, i.e. i = 105. Values for upper or lowercase can be used. Note that in most situations marker beacons rely on three-character messages and leave the fourth letter as zero.		
Interval1	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: Indicates how often the first word (outer marker) will repeat. Time is in seconds. An interval of 5 equates to the Morse string being played every 5 seconds.		

Control Inputs			
Interval2	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Indicates how often the second word (middle marker) will repeat. Time is in seconds. An interval of 5 equates to the Morse string being played every 5 seconds.		
Interval3	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: Indicates how often the third word (inner marker) will repeat. Time is in seconds. An interval of 5 equates to the Morse string being played every 5 seconds.		
Interval4	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Indicates how often the fourth word (fan marker) will repeat. Time is in seconds. An interval of 5 equates to the Morse string being played every 5 seconds.		
Number	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Selects which marker beacon (IdentChar or word) is to be transmitted. A value of 0 selects none. A value of 1 selects Ident1Char[0-3] (the Outer Marker). A value of 2 selects the second word (Middle marker). A value of 3 selects the third word (Inner Marker). A value of 4 selects the fourth word (Fan Marker).		

Control Inputs			
PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: This is used to trigger the active transmission of the Marker Beacon. When the PTT value remains True, the beacon will continuously play.		
Rate1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	8
	Description: The word rate of the Morse keyer for the first word (outer marker). Determines how fast the individual dashes and dots are played. Units are in dots per second.		
Rate2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	8
	Description: The word rate of the Morse keyer for the second word (middle marker). Determines how fast the individual dashes and dots are played. Units are in dots per second.		
Rate3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	12
	Description: The word rate of the Morse keyer for the third word (inner marker). Determines how fast the individual dashes and dots are played. Units are in dots per second.		
Rate4	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	8
	Description: The word rate of the Morse keyer for the fourth word (fan marker). Determines how fast the individual dashes and dots are played. Units are in dots per second.		

Audio Outputs			
Audio	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: The output audio signal produced by the Marker Tone. It will play the Morse signal that is generated based on the IdentChars.		
Keyline	**Variables**	**Type**	**Default Value**
	aout	basic/boolean	False
	Description: Keyline is True when a Morse signal is produced.This is primarily used to prevent re-keying by a radio or receiver after every dash or dot and instead allows open reception as long as the Morse signal transmits.		

Internal Parameters			
Keyer	**Variables**	**Type**	**Default Value**
	tone_level	basic/float32	0.95
	Description: Sets the volume of the tones produced by the component. This can only be set internally.		

2.8. Mixer4

Version: 0-0-1

Description: The Mixer4 component combines up to four different audio signals and adds them together. There are individual gain (volume) controls and also a master gain for the entire mixer. This component is a simplified version of the Mixer8 component.

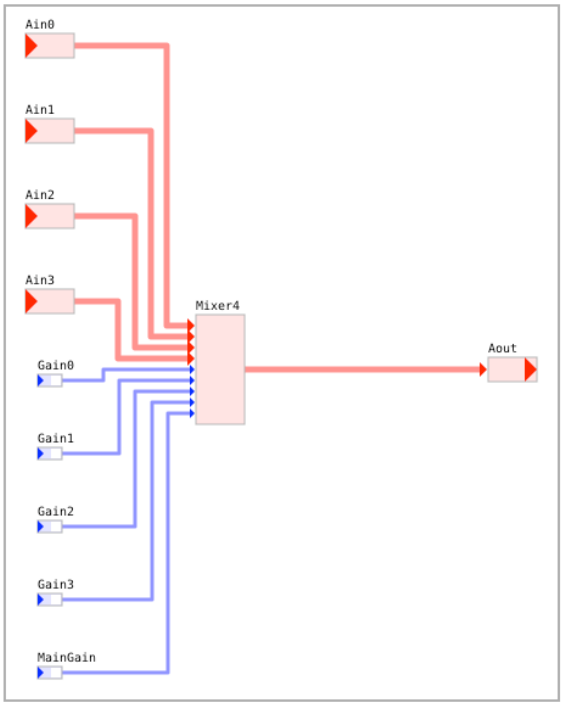


Figure 10: Mixer4

Audio Inputs			
Ain0	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: The input position for one of the audio signals that will be mixed. This is controlled by Gain0.		
Ain1	**Variables**	**Type**	**Default Value**
	audio	n/a	source
	Description: The input position for one of the audio signals that will be mixed. This is controlled by Gain1.		
Ain2	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: The input position for one of the audio signals that will be mixed. This is controlled by Gain2.		
Ain3	**Variables**	**Type**	**Default Value**
	audio	n/a	source
	Description: The input position for one of the audio signals that will be mixed. This is controlled by Gain3.		

Control Inputs			
Gain0	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Amplitude gain control for the incoming signal Ain0, which can be set via the host or internally. The scale factor, which multiplies with the kin gain, can only be set internally.		
Gain1	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Amplitude gain control for the incoming signal Ain1, which can be set via the host or internally. The scale factor, which multiplies with the kin gain, can only be set internally.		
Gain2	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Amplitude gain control for the incoming signal Ain2, which can be set via the host or internally. The scale factor, which multiplies with the kin gain, can only be set internally.		
Gain3	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Amplitude gain control for the incoming signal Ain3, which can be set via the host or internally. The scale factor, which multiplies with the kin gain, can only be set internally.		

Control Inputs			
MainGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Master amplitude gain control for all incoming signals, which can be set via the host or internally. The scale factor, which multiplies with the kin gain, can only be set internally.		

Audio Outputs			
Aout	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: The resulting mixed signal. It is a combination of the four input signals.		

Internal Parameters			
Mixer4	**Variables**	**Type**	**Default Value**
	localgain	basic/float32	1.0
	Description: This applies another gain to the output signal. This control can only be set within the object.		

2.9. Mixer8

Version: 0-0-1

Description: The audio mixer provides a controlled mixing of up to eight audio signals into a single signal. Control is provided over each of the eight signals to be mixed, with both individual signal and overall gain control. If more than 8 signals need to be mixed together simply cascade the mixers by making the “InSignal” another mixer. Additionally, the InSignal can also be used to force a particular signal to be mixed into the audio stream regardless of the control mask settings.

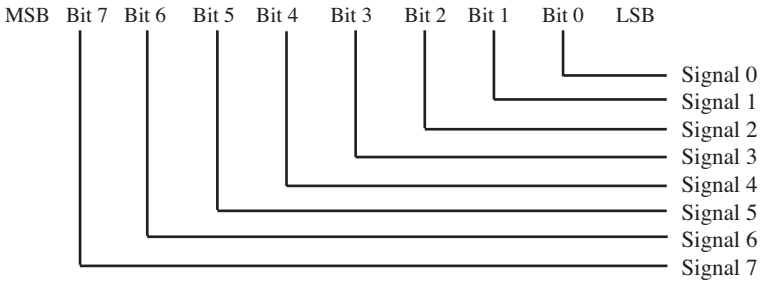


Figure 11: Mixer Signals

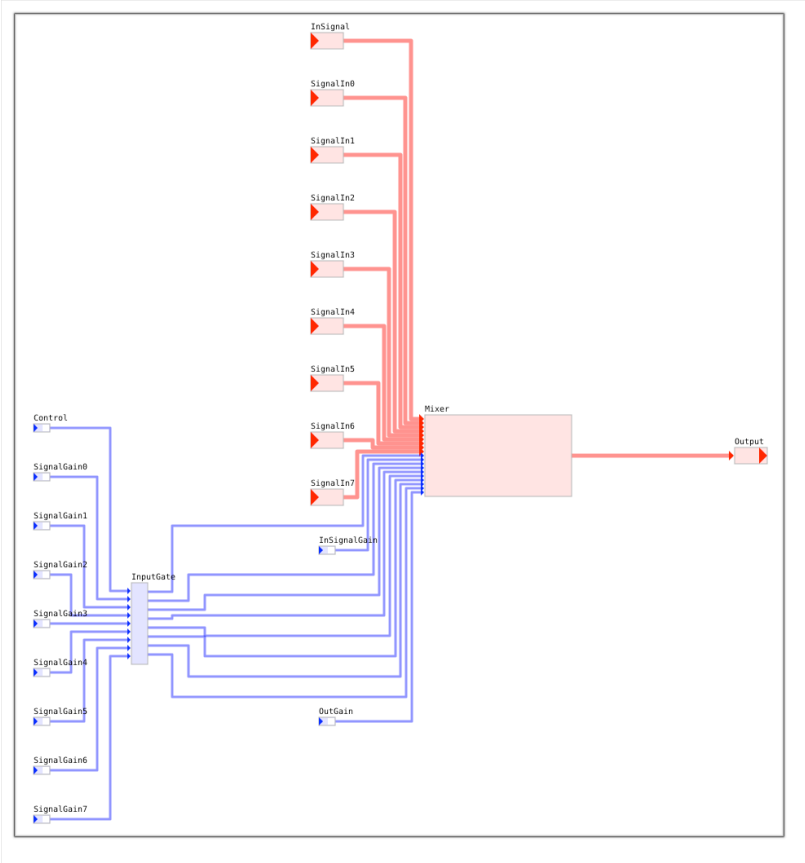


Figure 12: Mixer8 Schematic

Audio Inputs			
InSignal	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: One signal connection (a 9th signal) which is mixed into the output independent of the control mask.		
SignalIn0 through SignalIn7	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: Signal connection for the eight signals. The eight signals can be mixed together in any combination based on the control field bit mask. Only one Audio signal (a mix of all the signals selected via the Control variable) is outputted.		

Control Inputs			
Control	Variables	Type	Default Value
	kin	basic/uint8	255
	Description: Connection to a control bitmask for switching signals on or off. All 8 bits are used to provide a bit mask for each input signal, with the least significant bit controlling signal0 and the most significant bit signal7.		
InSignal Gain	Variables	Type	Default Value
	kin	basic/float32	1.0
	scale_factor	basic/float32	1.0
	Description: Input Gain for the InSignal. kin can be an externally linked value or set internally. Scale factor is only set internally. The final output is kin multiplied by scale_factor.		

Control Inputs			
OutGain	Variables	Type	Default Value
	kin	basic/float32	255
	scale_factor	basic/float32	1.0
	Description: Output gain for all signals after they are mixed together. Includes Input Signal and Signals 0-7. Kin can be an externally linked value or set internally. Scale factor is only set internally. The final output is kin multiplied by scale_factor.		
Signal Gain0 through Signal Gain7	Variables	Type	Default Value
	kin	basic/float32	1.0
	scale_factor	basic/float32	1.0
	Description: Individual Input Gains for each signal 0-7. Kin can be an externally linked value or set internally. Scale factor is only set internally. The final output is kin multiplied by scale_factor.		

Audio Outputs			
Output	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: Single Audio Output is based on the control mask. The output is a combination of up to 8 input signals. Additionally the InSignal is mixed into the output if connected.		

Internal Parameters			
InputGate	**Variables**	**Type**	**Default Value**
	control_mask	basic/uint8	255
	Description: **control_mask** - This field is logically 'AND' with the Control Field.		

2.10. MixerN

Version: 0-0-1

Description: The MixerN component is used to mix as many audio signals as needed and applies a single master gain to all of them. MixerN is similar to the Mixer4 and Mixer8 objects but does not have a limit to the number of input audio streams. The object is most useful when a large number of signals need to be combined together and individual gain settings are not a concern.

Figure 13: Mixer Schematic

Audio Inputs			
MixerN	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: The input position for all audio. Multiple audio streams can all be linked to the same spot.		

Control Inputs			
Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Master amplitude gain control for all incoming signals, which can be set via the host or internally. The scale factor which multiplies with the kin gain can only be set internally.		

Audio Outputs			
Aout	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: The combination of all input signals mixed together and then adjusted by Gain.		

2.11. Noise

Version: 0-0-1

Description: A white noise generator. The component creates pseudo random noise across the spectrum. Most uses for white noise also require an additional filter stage, thus the Noise component is rarely implemented and instead replaced by the NoiseFilter component.

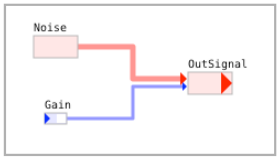


Figure 14: Noise Schematic

Control Inputs			
Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Master amplitude gain control for the noise, which can be set via the host or internally.		

Audio Outputs			
OutSignal	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: The generated white noise signal.		

Internal Parameters			
Noise	**Variables**	**Type**	**Default Value**
	gain	basic/float32	1.0
	enable	basic/boolean	True
	Description: Local controls for the noise source that must be set internally. Gain applies an amplitude control and enable to act as a master on/off switch.		

2.12. NoiseFilter

Version: 0-0-1

Description: The NoiseFilter component produces an internal pseudo random noise source with the ability to apply filtering to the generated noise signal. The type of filtering can be selected from low-pass, band-pass or high-pass. The filter quality factor, roll-off frequency, and gain can be controlled by input variables from elsewhere in the model, or from the host interface. Generally used in conjunction with other objects to create simulated environmental cue sounds such as engine noises, air flow, missile fly out, etc. Also sometimes used to provide radio noise simulation.

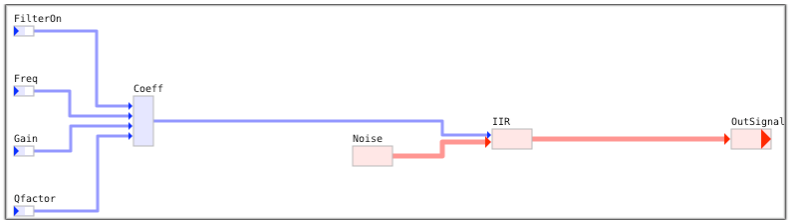


Figure 15: NoiseFilter Schematic

Control Inputs			
FilterOn	**Variables**	**Type**	**Default Value**
	kin	audio/filter_type2	Off
	Description: Allows the selection of the type of filtering to be applied. Choices are off, low-pass, band-pass or high-pass.		

Control Inputs			
Freq	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Roll-off frequency (in Hertz) of filter. The frequency, kin, can be an externally linked value or set internally. Scale factor is only set internally. The final output (kout) is kin multiplied by scale_factor.		
Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Amplitude gain factor of the noise signal. kin can be an externally linked value or set internally. Scale factor is only set internally. The final output is kin multiplied by scale_factor.		
Qfactor	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Provides the quality factor for the filter. A typical value is 0.707 provides a steep roll-off and minimal pass band ripple.		

Audio Outputs			
Outsignal	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: Final output signal from the NoiseFilter component. This field is internally generated noise with the selected filtering applied. It may be linked to another component downstream to send the output elsewhere in the model.		

Internal Parameters			
Noise	**Variables**	**Type**	**Default Value**
	enable	basic/boolean	TRUE
	Description: Enables/Disables the noise signal. TRUE = Enabled, FALSE = Disabled		
Coeff	**Variables**	**Type**	**Default Value**
	filterset-> freq	basic/float32	1.0
	filterset->gain	basic/float32	1.0
	filterset->qfactor	basic/float32	1.0
	Description: Apply scale factors to the respective inputs. This should not be changed in this component.		
IIR	**Variables**	**Type**	**Default Value**
	bypass	basic/boolean	False
	Description: Setting bypass to True will cause the audio signal to ignore the filter.		
	damping	basic/float32	0.999
	Description: A damping coefficient is applied to prevent instability due to rounding errors. This should not be altered.		

2.13. Psound

Version: 0-0-1

Description: The Psound component is an audio object that allows for the playing of one or multiple audio files. In MBV, these files are called Playsound Files. The audio files must be in the following format: wave 16-bit PCM (*.wav), 48 kHz sample rate, and mono.

Playsound files are best handled as fixed off-line recorded sound files. Before using the Psound component, the files must be added to a model via the Sound Library tool in MBV. Each instance of the Psound object can then reference a set of playfiles from the Sound Library and assign a unique index to each playfile. The component subsequently relies on the indices when playing the audio files.

For more information and examples on using playsound files and the Psound component see tutorial 3 of the MBV Basic Model Tutorial Guide. This document can be found on the ASTi web site at www.asti-usa.com. Also see the Psound255 component for more information.

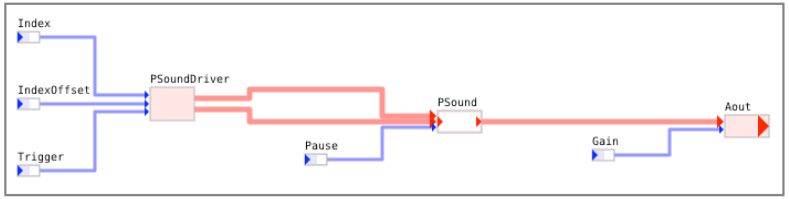


Figure 16: Psound Schematic

Control Inputs			
Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	scale_factor	basic/float32	1.0
	Description: Applies a gain to the overall output signal, typical values range from 0-1. kin can be an externally linked value or set internally. Scale factor is only set internally. The final output (kout) is kin multiplied by scale_factor.		

Control Inputs			
Index	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The current value of the file index. This index is used to select one of a group of files from the playfiles list. An Index value of 0 selects no files. An Index value of 1 selects the first file in the playfiles list to play when triggered. A value of 2 selects the second play sound file, and so on.		
Index Offset	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Adds the value set in this field to the value set in the Index field. For example, if the kin variable of the Index field was set to 1, and the kin IndexOffset was set to 2, the kout or net index value would be 3 and the third playfiles would play when triggered.		
Pause	**Variables**	**Type**	**Default Value**
	invert	basic/boolean	FALSE
	kin	basic/boolean	FALSE
	Description: Pauses the audio. A value of On freezes the soundfile playing. A value of Off allows the play to continue from the current file position. The invert flag can only be set internally and causes the Pause state to be opposite of the kin.		

Control Inputs			
Trigger	**Variables**	**Type**	**Default Value**
	invert	basic/boolean	FALSE
	kin	basic/boolean	FALSE
	Description: The current trigger state. A value of On starts playing the currently indexed soundfile. If in continuous mode the file replays while this trigger is On, if in one-shot mode the file plays once for each Off to On transition of the trigger. If invert is set to TRUE, the trigger condition is inverted. FALSE = Triggered, TRUE = Non triggered. One-shot mode plays for each On to Off transition of the trigger.		

Audio Outputs			
Aout	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: Final output signal from the Audio_PSound component. This field is the audio signal with the applied gain. It may be linked to another component downstream to send the output elsewhere in the model.		

Control Outputs			
Psound	**Variables**	**Type**	**Default Value**
	playingindex	basic/uint8	0
	Description: Returns the index of the playfile that is currently being played. A value of zero is given when no files are being played. This value can be linked back to the host or downstream elsewhere in the model.		

Internal Parameters			
Psound	**Variables**	**Type**	**Default Value**
	Playfiles->playfiles [0-31]	audio/playcmds	n/a
	Description: The PSound field contains the playfiles list. Playfiles created with the Sound Library Editor are loaded into the indexed positions under the *playfiles* variable.		

2.14. Psound255

Version: 0-0-1

Description: The Psound255 component is a variation of the Psound object. Psound255 has the ability to store and index up to 256 different playfiles instead of the 32 provided by the Psound component. All other variables and controls act in the same way.

Please see Psound component for more information.

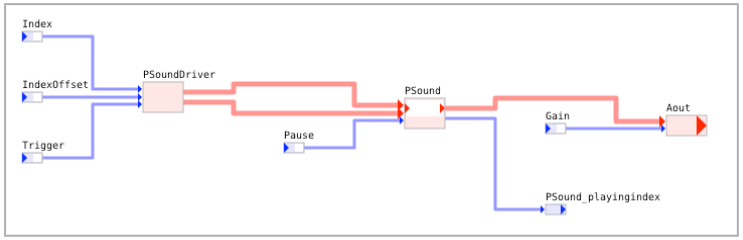


Figure 17: Psound255 Schematic

2.15. Playlist

Version: 0-0-1

Description: The Playlist component is used to store the order of a large group of playfiles. The object is most commonly implemented when small sets from a bank of playfiles need to be played in a specific order, and that order may change throughout an exercise.

Playlist must always be used in conjunction with the Sequencer or Resequencer components. When applied together, the components provide seamless replay of multiple recordings and remove any possible gaps or pauses between playfiles.

For more information see Psound, Sequencer, and Resequencer.

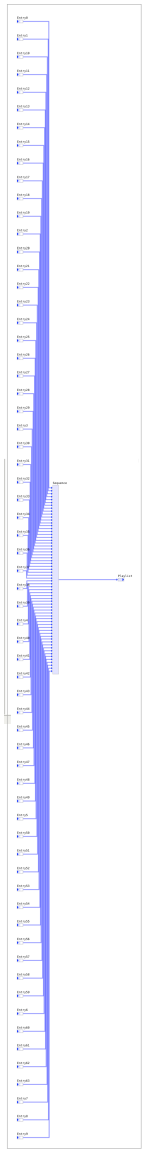


Figure 18: Playlist Schematic

Control Inputs			
Entry 0-63	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Each Entry stores a single index for a playfile. The order applied to the playlist is numerical such that Entry0 will be the first index, Entry1 will be the second index, et. al. Entries with a value of zero indicate the end of a playlist and subsequent entries will be ignored.		

Control Outputs			
Playlist	**Variables**	**Type**	**Default Value**
	kin	audio/ playsound_sequence	n/a
	Description: The resulting array of indices produced by the component. A special vartype is used that can only be linked to a Sequencer or Resequencer. When linking downstream, be sure to select ‘Playlist ->all of....’.		

2.16. Record and Playback in MBV in Telestra software version 3.31 or greater

Record and Playback rely on several components and tools for use in MBV. The remainder of this section describes how Record and Playback is implemented and how to use the ‘Edit Record Files’ and ‘Edit Replay Files’ tools within MBV.

MBV can record audio streams for play back during an after action review or even in the middle of an exercise. Recorded sound is stored in a raw format audio file on the hard disk. Management of these files in done through the 'Edit Record Files' tool in MBV. Inside the model, the Record and RecordGroup components are responsible for writing to the record files during runtime.

After creating a recording the files can be played back via the Replay and ReplayGroup components. A similar tool, 'Edit Replay Files,' is used to interface with the files on the hard disk. Although a single file is created, MBV treats the audio with two separate interfaces, a recordfile and a replayfile. This schism allows the Telestra to handle recording and replaying at the same time without reloading the model.

The 'Edit Record Files' tool can be found under the ‘Tools’ menu in MBV. Recordfiles can be stored as indexed groups or as a single file. Select 'Add File' and choose the group of NONE to name a single file. Once a recordfile has been named, it can be linked to an actual file on the hard disk.

Click 'Browse' to select a file and location for recording. The default location is /usr/local/asti/record/, but files can be saved anywhere on the hard drive. Type in a file name in the 'Choose a record file' pop-up window and select save. The path and file name should now appear in the Edit Record Files box next to 'Filename.'

Recordfiles also have grouping capability. Storing files in groups allows MBV to change what file it is recording to during runtime, without reloading the model. Use the 'Add Group' button to create a group name. A recordfile can be added to the group by selecting it from the 'Select a Group' pull down menu. Grouped files are given a default index that can be changed by the user.

The 'Edit Replay Files' tool has the same user interface as 'Edit Record Files.' Adding files and groups are implemented in the same way as recordfiles. In order to playback a recording, select the audio file by using the 'Browse' button. Note that the same file on the hard disk can be used as both a recordfile and a replayfile.

When adding or editing the Record/Replay files, the model must be reloaded in order for the changes to take effect.

For more information see Record, RecordGroup, Replay, and ReplayGroup.

2.17. Record

Version: 0-0-3

Description: The Record component provides the ability to record audio streams in the model to a selected file on the hard disk. The object takes an input audio along with triggers for starting, resetting and looping. A special record file must be created and named via the 'Edit Record Files' Tool in MBV before the component can be utilized.

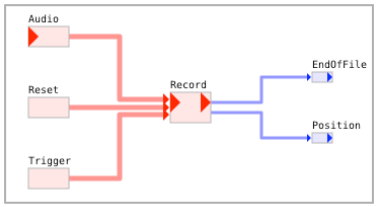


Figure 19: Record Schematic

Audio Inputs			
Audio	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: Audio linked to this position will be written to the record file.		
Record	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Do not link audio to any of the inputs under the 'to Record' section. These are used by the Record Service internally and are not meant for linking.		

Control Inputs			
Loop Mode	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Sets the component to loop during recording. If False, the component will stop recording when the end of file (EOF) is reached. Note that EOF will only exist if the duration of the record file is non-zero. Duration can be set in the 'Edit Record Files' tool within MBV. If LoopMode is set to True, when the record object is at the end of file, it will continue to record starting at position 0 (the beginning of the file). This will overwrite audio that was already recorded. In order for looping to occur, the record file must have a non-zero duration. If the state of LoopMode is changed while recording, the component must be re-triggered in order for the change to take effect.		
Reset	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: When set to True, the Record component will set the position to 0, and begin recording based on the Time control. The recording will not start until Reset reverts to False.		
Time	**Variables**	**Type**	**Default Value**
	kin	basic/unit32	0
	Description: The time the component should start recording the file. Units are in milliseconds. If a value of zero is used, the object will ignore the Time field and continue recording based on the previous position. The Time field will also be ignored if it is greater than the duration of the selected file.		

Control Inputs			
Trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Set the trigger to True to begin recording. Setting to False will cause the recording to stop.		

Control Outputs			
End of File	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Indicates if the component is at the end of the record file. If the LoopMode is not active, the object will stop recording after reaching the end of the file. If End of File is True and recording is retriggered, the recording will start back at 0. This control can be sent downstream to other components or given back to the host.		
Position	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: Returns the position of the record file in milliseconds. This control is updated as the component records audio. If the component stops recording, the position will hold at its current value until recording starts or the reset flag is set.		

Internal Parameters			
Record	**Variables**	**Type**	**Default Value**
	kin->entry	audio/recordfile	n/a
	Description: Entry stores the record file on the hard disk. The record file must first be created in the ‘Edit Record Files’ tool. Right-click ‘entry’ and then select ‘Choose Record File’ and pick the desired file from the list. Once a file has been selected, the model must be reloaded in order for the selection to take effect.		

2.18. RecordGroup

Version: 0-0-2

Description: The RecordGroup component provides the ability to record audio streams in the model to selected files on the hard disk. The object takes an input audio along with triggers for starting, resetting, and index controls. A special record group must be created and named via the 'Edit Record Files' Tool in MBV before the component can be utilized.

The RecordGroup differs from the Record object in two ways; the grouped files with indexing allow for a user to change the record file without reloading the model but do not have the same flexibility and controls that are associated with the Record component.

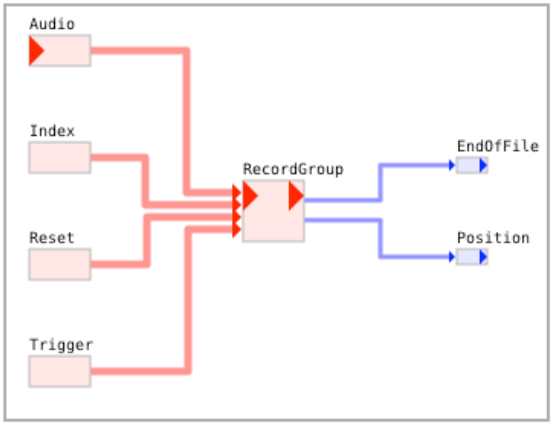


Figure 20: RecordGroup

Audio Inputs			
Audio	**Variables**	**Type**	**Default Value**
	source	audio	n/a
	Description: Audio linked to this position is written to a record file. The file is selected based on the Index control.		

Audio Inputs			
Record Group	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Do note link audio to any of the inputs under the 'to RecordGroup' section. These are used by the Record Service internally and are not mean for linking.		

Control Inputs			
Index	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: Selects the recordfile of a group based on the index that is assigned using the 'Edit Record Files' tool. The index can be changed during runtime.		
Reset	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: When set to True, the RecordGroup component will set the position to 0, and begin recording. The recording will not start until Reset reverts to False.		
Trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Set the trigger to True to begin recording. Set the trigger to False to stop recording. Recording will continue at the same position that is was stopped at unless a reset is implemented. If the End of File has been reached, retriggering will cause the recording to start back at 0.		

Control Outputs			
EndofFile	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Indicates if the component is at the end of the record file. This control can be sent downstream to other components or given back to the host. If the EOF is True, triggering the object will cause the recording to start back at the beginning of the file.		
Position	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: Returns the position of the record file in milli-seconds. This control is updated as the component records audio. If the component stops recording, the position will hold at its current value until recording starts or the reset flag is set.		

Internal Parameters			
Record-Group	**Variables**	**Type**	**Default Value**
	file_list	audio/recordgroup	n/a
	Description: File_list stores the record group on the hard disk. The record group must first be created in the 'Edit Record Files' tool. Right-click 'file_list' then select 'Choose Record Group' and pick the desired group from the list. Once a group has been selected, the model must be reloaded in order for the selection to take effect.		

2.19. Replay

Version: 0-0-4

Description: The Replay component provides the ability to play audio streams that are recorded via the Record or RecordGroup components. Controls are included for gain, looping, pausing, and selecting a start position. A special replay file must be created and named using the 'Edit Replay Files' Tool in MBV before the component can be utilized.

For more information see Record & Replay in MBV, Record, RecordGroup, and ReplayGroup.

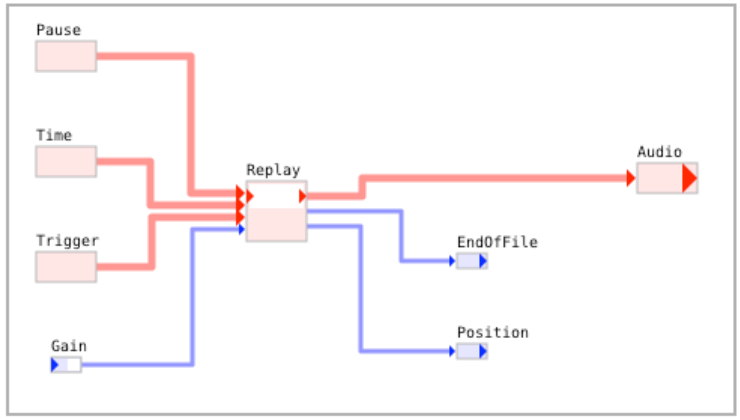


Figure 21: Replay Schematic

Audio Inputs			
Replay	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Do not link audio to any of the inputs under the 'to Replay' section. These are used by the Replay Service internally and are not mean for linking.		

Control Inputs			
Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the audio files, which can be set via the host or internally.		
Loop Mode	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Sets the component to continuous play in loop. If False, the component will stop playing after reaching the end of file (EOF). Note that EOF is based on the Duration set in the 'Edit Replay Files' tool within MBV. It is possible to set the Duration to be shorter than the total file length. If LoopMode is set to True, when the replay object is at the end of file, it will continue to replay starting at position 0 (the beginning of the file). If the Duration is set to zero, the file will loop based on the size of the file. If the state of LoopMode is changed while recording, the component must be re-triggered in order for the change to take effect.		
Pause	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Pauses the playing audio. Audio streams will stop and hold their position while pause is set to True. This component can be controlled by the host.		

Control Inputs			
Time	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: Time instructs the component where to start replaying in the file. Units are in milliseconds. If a value of zero is used, the object will ignore the Time field and continue playing based on the previous position. If Time is greater than the duration of the selected file the object will not play any audio.		
Trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Set the trigger to True to start replaying. Setting to False will cause the replay to stop.		

Audio Outputs			
Audio	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: Final output signal of the replay component. Audio is controlled by the Gain and read from the raw audio record file.		

Control Outputs			
EndofFile	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Indicates if the component is at the end of the replay file. If LoopMode is not active, the object will stop playing after reaching the end of the file. End of file occurs based on the Duration set in the 'Edit Replay Files' tool, a Duration of 0 will use the true end of file. This control can be sent downstream to other components or given back to the host.		
Position	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: Returns the position of the replay file in milliseconds. This control is updated as the component plays audio. If the component stops playing, the position will hold at its current value.		

Internal Controls			
Replay	**Variables**	**Type**	**Default Value**
	control_init	audio/replayfile	n/a
	Description: The control_init field contains the replayfile name. The replay file is first created in the 'Edit Replay Files' tool. Right-click 'control_init' then select 'Choose Replay File' and pick the desired file from the list. Once the file is selected the model must be reloaded in order for the selection to take effect.		

2.20. ReplayGroup

Version: 0-0-2

Description: The ReplayGroup component provides the ability to play audio streams that have been recorded via the Record or RecordGroup components. Controls are included for gain, pausing, and selecting a start position. A special replay group must be created and named using the 'Edit Replay Files' Tool in MBV before the component can be utilized.

For more related information see Record and Replay in MBV, Record, RecordGroup and Replay.

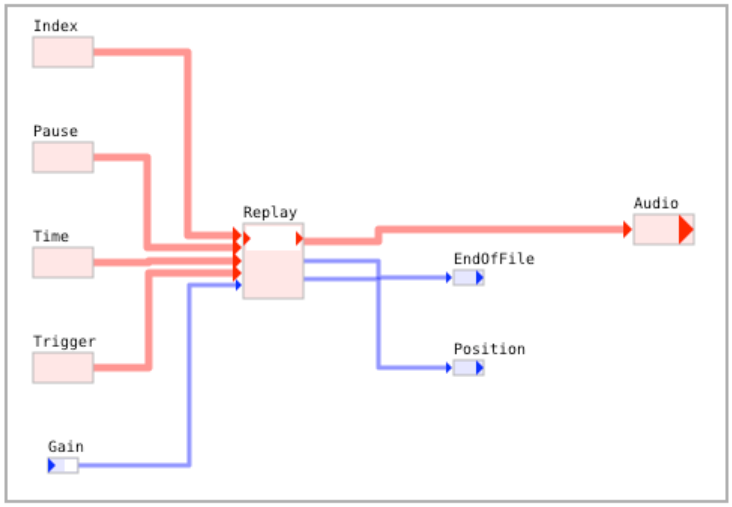


Figure 22: ReplayGroup Schematic

Audio Inputs			
Replay	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Do not link audio to any of the inputs under the 'to Replay' section. These are used by the Replay Service internally and are not mean for linking.		

Control Inputs			
Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the audio files, which can be set via the host or internally.		
Index	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: Selects the replayfile based on the index assigned within the 'Edit Replay Files' tool. The index can be changed during runtime.		
Pause	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Pauses the playing audio. Audio streams will stop and hold their position while pause is set to True. The Pause component can be controlled by the host.		

Control Inputs			
Time	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: Tells the component where to start replaying in the file. Units are in milliseconds. If a value of zero is used, the object will ignore the Time field and continue playing based on the previous position. If Time is greater than the duration of the selected file, not audio is played.		
Trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	false
	Description: Set the trigger to True to start replaying. Setting to False will cause the playing to stop.		

Audio Outputs			
Audio	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: Final output signal of the replay component. Audio is controlled by the Gain and read from the raw audio record file.		

Control Outputs			
EndofFile	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Indicates if the component is at the end of the replay file. If LoopMode is not active, the object will stop playing after reaching the End of File. End of File occurs based on the Duration set in the 'Edit Replay Files' tool, a Duration of 0 will use the true end of file. This control can be sent downstream to other components or given back to the host.		
Position	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: Returns the position of the replay file in milli-seconds. This control is updated as the component plays audio. If the component stops playing, the position will hold at its current value.		

Internal Parameters			
Replay	**Variables**	**Type**	**Default Value**
	file_list	audio/replaygroup	n/a
	Description: The file_list field contains the replaygroup name. The replay group must first be created in the 'Edit Replay Files' tool. Right-click 'file_list' then select 'Choose Replay Group' and pick the desired group from the list. Once a file group is selected, the model must be reloaded in order for the selection to take effect.		

2.21. ReSequencer

Version: 0-0-1

Description: The ReSequencer component uses 4 playlist inputs that each contain 64 index values. The 64 index values correspond to the indices in the PSound255 component to form a possible sequence of 255 indices. When the ReSequencer is triggered, the indices in the playlist will play their corresponding playsounds in a PSound255 component in sequence. The sequence will continue until the entire list is exhausted or it reaches a value of zero.

In order for the ReSequencer to work properly, it must be used with a PSound255 component (unlike the Sequencer component which has the internal Psound). The output index control of the ReSequencer must be connected to the input index control of the PSound255. The TriggerOut must be connected to the Trigger input. The PSound255 Aout must be connected to the Active Audio input of the ReSequencer.

For additional information see Playlist, Psound255, and Sequencer.

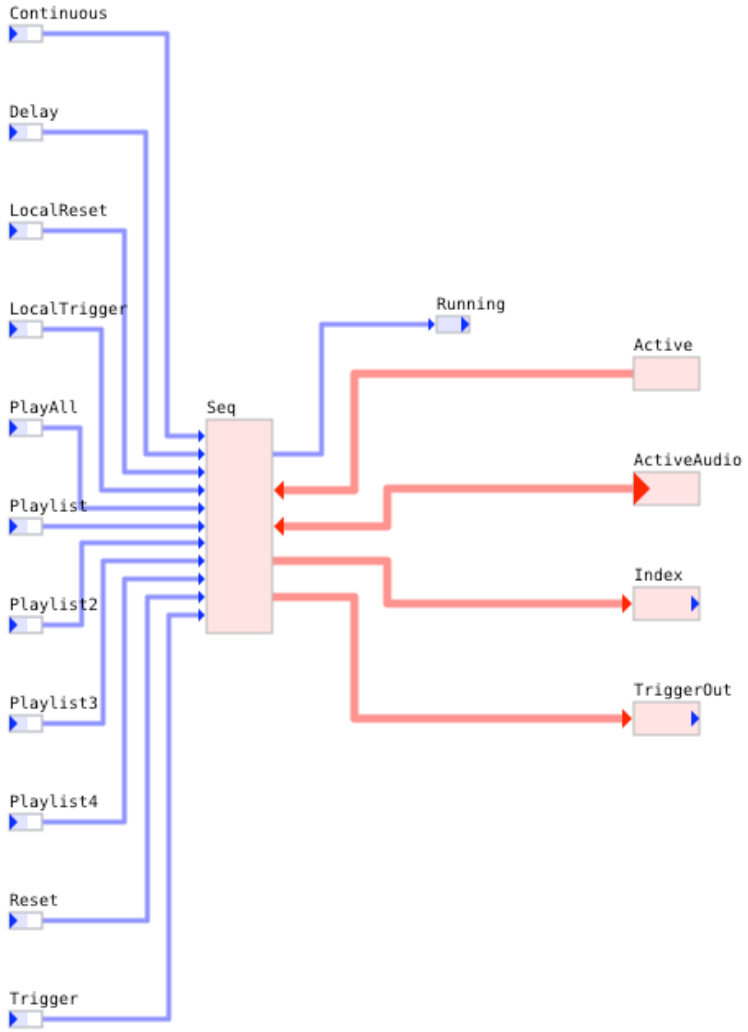


Figure 23: ReSequencer Schematic

Audio Inputs			
Active Audio	**Variables**	**Type**	**Default Value**
	Ain	audio	n/a
	Description: **IMPORTANT: The Aout audio from the PSound255 must be connected to this control.**		

Control Inputs			
Continuous	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: When the trigger value is true the sequence repeats continuously.		
Delay	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: The delay in a repeating sequence when the continuous flag is true (delays in msec.).		
Local Reset	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: LocalReset functionality is identical to the Reset control. The final Reset output is the exclusive oring of LocalReset and Reset.		
Local Trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: LocalTrigger functionality is identical to the Trigger control. The final Trigger output is the exclusive oring of LocalTrigger and Trigger.		

Control Inputs			
PlayAll	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: When the sequencer is not in continuous mode, if PlayAll is set to true, the sequence will complete in its entirety once the sequencer is triggered. If PlayAll is set to false, the sequence will stop playing as soon as the trigger is set to false.		
Playlist	**Variables**	**Type**	**Default Value**
	kin>>[0-63]	playsound_sequence	0
	Description: Control should be connected to the output of a Playlist component. The sequence of the index connected in the Playlist is used in the first 64 positions (Sequencer positions 0 through 63) of the ReSequencers 255 wide.		
Playlist2	**Variables**	**Type**	**Default Value**
	kin>>[0-63]	playsound_sequence	0
	Description: Control should be connected to the output of a Playlist component. The sequence of the index connected in the Playlist is used in the first 64 positions (Sequencer positions 64 through 127) of the ReSequencers 255 wide.		
Playlist3	**Variables**	**Type**	**Default Value**
	kin>>[0-63]	playsound_sequence	0
	Description: Control should be connected to the output of a Playlist component. The sequence of the index connected in the Playlist is used in the first 64 positions (Sequencer positions 128 through 191) of the ReSequencers 255 wide.		

Control Inputs			
Playlist4	**Variables**	**Type**	**Default Value**
	kin>>[0-63]	playsound_sequence	0
	Description: Control should be connected to the output of a Playlist component. The sequence of the index connected in the Playlist is used in the first 64 positions (Sequencer positions 192 through 254) of the ReSequencers 255 wide.		
Reset	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: When True the sequencer stops and reloads the Playlist.		
Trigger	**Variables**	**Type**	**Default Value**
	invert	basic/boolean	False
	kin	basic/boolean	False
	Description: The current trigger state. A value of On starts playing the currently indexed soundfile. If in continuous mode the file replays while this trigger is On, if in one-shot mode the file plays once for each Off to On transition of the trigger. If invert is set to TRUE, the trigger condition is inverted. FALSE = Triggered, TRUE = Non triggered. One-shot mode plays for each On to Off transition of the trigger.		

Control Outputs			
Index	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Outputs the index number from the playlist sequence that is currently being played in the sequencer. **IMPORTANT: This must be connected to the Index input control of a PSound255.**		
Running	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	False
	Description: A value of True indicates the sequencer is currently playing a soundfile.		
Trigger Out	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: **IMPORTANT: This must be linked to the Trigger of a PSound255.**		

2.22. RunwayBump

Version: 0-0-1

Description: The Runway Bump component generates the sound from plane tires hitting the gaps between the concrete runway blocks on the tarmac. The object can be set to auto-generate the sounds or it can be triggered by the host in order to sync with external visuals. Two audio signals are generated, one for the main wheels and one for the front wheels.

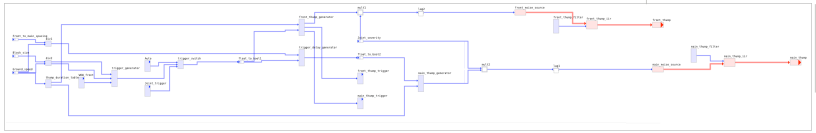


Figure 24: RunwayBump Schematic

Control Inputs			
Auto	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	kin_bool	basic/boolean	False
	kin_int	basic/int32	0
	kin_uint	basic/uint32	0
	type	basic/vartype	Boolean
	Description: One of the trigger types for the runway bump. Properly setting auto will cause the component to determine when the bumps should occur based on ground speed. Auto is a multi-vartype input, allowing the use of more than one type of variable. Type must be set within the Auto primitive to the vartype that is being driven, i.e. when using kin, type must be set to Float.		

Control Inputs			
Block_size	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The physical length of the concrete runway blocks. The block_size is considered as a length, this variable indicates the distance between the gaps. This component is only needed in Auto mode. A scale_factor is included as a scalar modifier to the kin input. Units for block_size are dependant upon the ground_speed.		
Front_to_main_spacing	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: The distance between the front wheels and the main wheels. This is only needed when in Auto mode. A scale_factor is included as a scalar modifier to the kin input. Units for front_to_main_spacing are dependant upon ground_speed.		
Ground_speed	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: The speed of the object moving the runway. This is only needed when in Auto mode. A scale_factor is included as a scalar modifier to the kin input.		

Control Inputs			
Joint_ severity	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: An indication of the depth of the gaps between runway blocks or how much they protrude. The effect is a multiplier where the average runway joint has a severity of 1.		
Joint_ trigger	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	kin_bool	basic/boolean	False
	kin_int	basic/int32	0
	kin_uint	basic/uint32	0
	type	basic/vartype	boolean
	Description: Per bump trigger for the component. If used controls such as ground_speed are ignored. The component will simply bump when the trigger is high. Joint_trigger is a multi vartype input, allowing for the use of more than one type of variable. Type must be set within the joint_trigger primitive to the vartype that is being driven, i.e. when using kin, type must be set Float.		

Control Inputs			
WOW_ front	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	kin_bool	basic/boolean	False
	kin_int	basic/int32	0
	kin_uint	basic/uint32	0
	type	basic/vartype	boolean
	Description: Parameter indicating the weight on the front wheels. This input effects the character and amplitude level of the bumps. WOW_front is a multi vartype input, allowing for the use of more than one type of variable. Type must be set within the WOW_front primitive to the vartype that is being driven, i.e. when using kin, type must be set to Float.		
The following control inputs are for internal use only and should not be linked. • div1_Y_scale • div1_Gain • div1_X_scale • div2_Y_scale • div2_Gain • div2_X_scale • trigger_switch_gain			

Audio Outputs			
Front_ thump	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: The audio signal of the front wheels hitting a runway bump.		
Main_ thump	**Variables**	**Type**	**Default Value**
	aout	audio	n/a
	Description: The audio signal of the main wheels hitting a runway bump.		

Control Outputs			
Front_ thump_ trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Indicates when the component triggers a runway bump for the front wheels. A value of True indicates a bump is playing.		
Main_ thump_ trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: Indicates when the component triggers a runway bump for the main wheels. A value of True indicates a bump is playing.		

Internal Parameters			
Thump_ duration_ table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	False
	table	control/tablexy	0.0 for all points
	Description: Control table that takes in the ground_speed as and input and determines how long the bump should last. See TableXY16 for more information on using tables in MBV.		

2.23. Sequencer

Version: 0-0-1

Description: The Sequencer component uses a playlist input that contains a sequence of 64 index values that correspond to their indices in the Psound primitive. When the Sequencer is triggered, the indices in the playlist will play the corresponding playsounds in the Psound in sequence continuing until the entire list is exhausted or it reaches a value of zero.

For additional information see the Playlist, Psound and Resequencer.

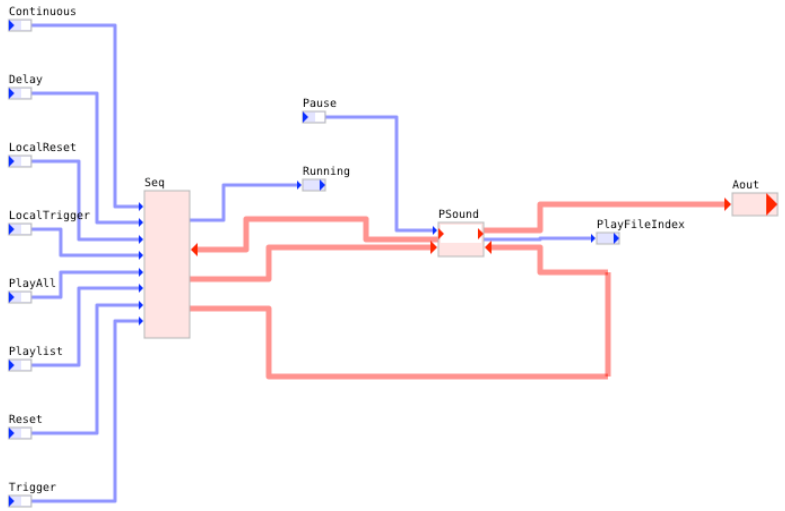


Figure 25: Sequencer Schematic

Control Inputs			
Continuous	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: When the trigger value is True the sequence from the playlist repeats continuously.		

Control Inputs			
Delay	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	0
	Description: The delay in a repeating sequence when the continuous flag is true (delays in msec.).		
Local Reset	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: LocalReset functionality is identical to the Reset control. The final Reset output is the exclusive oring of LocalReset and Reset.		
Local Trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: LocalTrigger functionality is identical to the Trigger control. The final Trigger output is the exclusive oring of LocalTrigger and Trigger.		
Pause	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: A value of true will pause the currently playing sound.		
PlayAll	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: When the sequencer is not in continuous mode, if PlayAll is set to true, the sequence will complete in its entirety once the sequencer is triggered. If PlayAll is set to false, the sequence will stop playing as soon as the trigger is set to false.		

Control Inputs			
Playlist	**Variables**	**Type**	**Default Value**
	kin>>[0-63]	playsound_sequence	0
	Description: The sequence of indices mapped to the play-sound in the PSound primitive. This input can only be generated from a Playlist component.		
Reset	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: When True the sequencer stops and reloads the Playlist.		
Trigger	**Variables**	**Type**	**Default Value**
	invert	basic/boolean	False
	kin	basic/boolean	False
	Description: The current trigger state. A value of On starts playing the currently indexed soundfile. If in continuous mode the file replays while this trigger is On, if in one-shot mode the file plays once for each Off to On transition of the trigger. If invert is set to TRUE, the trigger condition is inverted. FALSE = Triggered, TRUE = Non triggered. One-shot mode plays for each On to Off transition of the trigger.		

Audio Outputs			
Aout	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: Final output signal from the Audio_PSound component. This field is the audio signal with the applied gain. It may be linked to another component downstream to send the output elsewhere in the model.		

Control Outputs			
Playfile Index	**Variables**	**Type**	**Default Value**
	kout	basic/uint8	0
	Description: Produces the index number from the playlist control that is currently being played in the sequencer.		
Running	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	False
	Description: A value of True indicates the sequencer is currently playing a soundfile.		

2.24. Vox

Version: 0-0-1

Description: The VOX object allows VOX or press-to-talk (PTT) control over an input channel or a signal. If the filtered input sound level exceeds the VOX level, the VOX comparator will turn on. Once the input signal level has dropped below the VOX threshold, the VOX object will remain ON for the period of time specified by the VOX delay.

There are two Audio out streams. They are both filtered (if chosen to be) and originate from the one VOX input signal. The difference is that each Audio Out stream's amplitude can be individually set (OutGain and Out2Gain). This allows for two different volume levels of the same VOX signal.

For additional information also see Filter and Vox_AB.

Example: Trainee ICS microphone input, Trainee Instructor ICS microphone input, IOS ICS microphone input

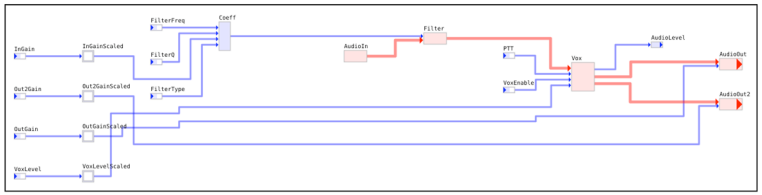


Figure 26: Vox Schematic Overview

Audio Inputs			
AudioIn	**Variables**	**Type**	**Default Value**
	source	audio	N/A
	Description: Audio Input Signal. Generally, this is the audio picked up from a microphone and fed into the model. The level of the inputted audio must surpass the VoxLevel, if Vox is enabled, in order to be passed to the AudioOut and AudioOut2 fields. If Vox is not enabled, the PTT field must be TRUE in order for the input audio to be passed on.		

Control Inputs			
Filter Freq	**Variables**	**Type**	**Default Value**
	kin	basic/float32	4000
	Description: Provides the characteristic frequency for the selected filter type.		
FilterType	**Variables**	**Type**	**Default Value**
	kin	audio/filter_type2	Off
	Description: Type of filter the input signal will be filtered. The filtering occurs before the VOX compares the signal level to the VOX Level. Filter types are off, BandPass, HighPass and LowPass.		
FilterQ	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.707
	Description: Provides the quality factor for the filter. A typical value is 0.707, which provides a steep rolloff and minimal pass band ripple.		

Control Inputs			
InGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the incoming signal. Kin can be an externally linked value or set internally.		
OutGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Output amplitude gain control for the first Audio Out signal.		
Out2Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Output amplitude gain control for the second Audio Out signal. Kin can be an externally linked value or set internally.		
PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: If the Press To Talk is ON(TRUE), then the input signal will be passed to the output.		
Vox Enable	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: If the kin variable is set to TRUE, VOX is enabled. The audio signal from AudioIn must surpass the audio level set in the VoxLevel field. If kin is set to FALSE, Vox is disabled and PTT must be set to TRUE in order for the AudioIn Signal to be passed on.		

Control Inputs			
VoxLevel	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.002
	Description: If the VOX enable is on, then the Signal Level is compared to the VOX Level. If the signal level is higher, it enables the input to be fed through for a period of time equal to the VOX Delay. If the signal level exceeds the VOX level while the timer is on, it will reset the timer. (i.e. If the input signal exceeds the VOX level at least once every delay time, the input signal will be continually fed through.)		

Audio Outputs			
AudioOut	**Variables**	**Type**	**Default Value**
	source	audio	N/A
	Description: One of two audio output signals. The output signal is the filtered input signal with its amplitude scaled by the value set in the OutGain field.		
AudioOut 2	**Variables**	**Type**	**Default Value**
	source	audio	N/A
	Description: One of two audio output signals. The output signal is the filtered input signal with its amplitude scaled by the value set in the OutGain field.		

Control Outputs			
Audio Level	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: The kin value of this field is set by the Vox component. It represents the current AudioIn level of the input signal. This output is intended to provide feedback to other components or send out to a host.		

Internal Parameters			
InGain Scaled	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain scale for the incoming audio signal. The gain _scale value is multiplied by the value set in the InGain field. The output is applied to the pre-filtered audio input signal.		
OutGain Scaled	**Variables**	**Type**	**Default Value**
	gain_scale	basic/float32	1.0
	Description: Amplitude gain scale for the Output audio signal. The gain _scale value is multiplied by the value set in the OutGain field. The output is applied to the filtered audio input signal which is routed to the AudioOut field.		
Out2Gain Scaled	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain scale for the Output2 audio signal. The gain _scale value is multiplied by the value set in the Out2Gain field. The output is applied to the filtered audio input signal which is routed to the AudioOut2 field.		
VoxLevel Scaled	**Variables**	**Type**	**Default Value**
	gain_scale	basic/float32	1.0
	Description: Provides an offset value for the VOX level Connection. If the VOX Level Connection Field is empty, then this becomes the VOX Level.		

2.25. Vox_AB

Version: 0-0-1

Description: Vox_AB is a modified form of the Vox component. Instead of a single audio stream, the object takes two audio inputs and mixes them together before applying the same controls that exist in the Vox object.

For additional information see Vox and Filter.

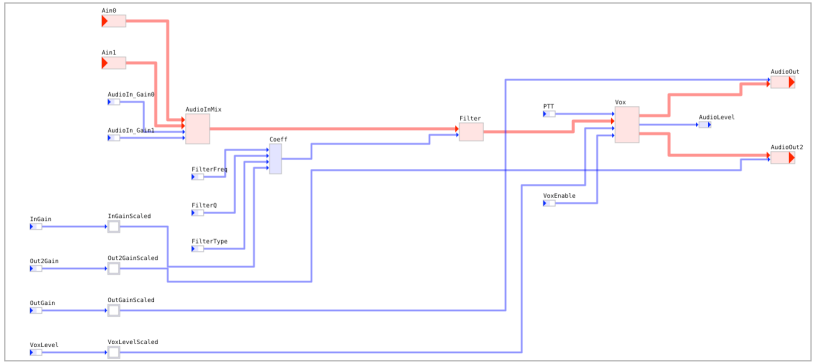


Figure 27: Vox_AB

Audio Inputs			
AudioIn	**Variables**	**Type**	**Default Value**
	source	audio	N/A
	Description: Audio Input Signal. Generally, this is the audio picked up from a microphone and fed into the model. The level of the inputted audio must surpass the VoxLevel, if Vox is enabled, in order to be passed to the AudioOut and AudioOut2 fields. If Vox is not enabled, the PTT field must be TRUE in order for the input audio to be passed on.		

Control Inputs			
AudioIN_ Gain0	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Amplitude gain control for Ain0. Gain is applied before the two signals are mixed together.		
AudioIN_ Gain1	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Amplitude gain control for Ain1. Gain is applied before the two signals are mixed together.		
Filter Freq	**Variables**	**Type**	**Default Value**
	kin	basic/float32	4000
	Description: Provides the characteristic frequency for the selected filter type.		
FilterType	**Variables**	**Type**	**Default Value**
	kin	audio/filter_type2	Off
	Description: Type of filter the input signal will be filtered. The filtering occurs before the VOX compares the signal level to the VOX Level. Filter types are off, BandPass, HighPass and LowPass.		
FilterQ	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.707
	Description: Provides the quality factor for the filter. A typical value is 0.707, which provides a steep rolloff and minimal pass band ripple.		

Control Inputs			
InGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the incoming signal. Kin can be an externally linked value or set internally.		
OutGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Output amplitude gain control for the first Audio Out signal.		
Out2Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Output amplitude gain control for the second Audio Out signal. Kin can be an externally linked value or set internally.		
PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: If the Press To Talk is ON(TRUE), then the input signal will be passed to the output.		
Vox Enable	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: If the kin variable is set to TRUE, VOX is enabled. The audio signal from AudioIn must surpass the audio level set in the VoxLevel field. If kin is set to FALSE, Vox is disabled and PTT must be set to TRUE in order for the AudioIn Signal to be passed on.		

Control Inputs			
VoxLevel	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.002
	Description: If the VOX enable is on, then the Signal Level is compared to the VOX Level. If the signal level is higher, it enables the input to be fed through for a period of time equal to the VOX Delay. If the signal level exceeds the VOX level while the timer is on, it will reset the timer. (i.e. If the input signal exceeds the VOX level at least once every delay time, the input signal will be continually fed through.)		

Audio Outputs			
AudioOut	**Variables**	**Type**	**Default Value**
	source	audio	N/A
	Description: One of two audio output signals. The output signal is the filtered input signal with its amplitude scaled by the value set in the OutGain field.		
AudioOut 2	**Variables**	**Type**	**Default Value**
	source	audio	N/A
	Description: One of two audio output signals. The output signal is the filtered input signal with its amplitude scaled by the value set in the OutGain field.		

Control Outputs			
Audio Level	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: The kin value of this field is set by the Vox component. It represents the current AudioIn level of the input signal. This output is intended to provide feedback to other components or send out to a host.		

Internal Parameters			
InGain Scaled	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain scale for the incoming audio signal. The gain _scale value is multiplied by the value set in the InGain field. The output is applied to the pre-filtered audio input signal.		
OutGain Scaled	**Variables**	**Type**	**Default Value**
	gain_scale	basic/float32	1.0
	Description: Amplitude gain scale for the Output audio signal. The gain _scale value is multiplied by the value set in the OutGain field. The output is applied to the filtered audio input signal which is routed to the AudioOut field.		
Out2Gain Scaled	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain scale for the Output2 audio signal. The gain _scale value is multiplied by the value set in the Out2Gain field. The output is applied to the filtered audio input signal which is routed to the AudioOut2 field.		
VoxLevel Scaled	**Variables**	**Type**	**Default Value**
	gain_scale	basic/float32	1.0
	Description: Provides an offset value for the VOX level Connection. If the VOX Level Connection Field is empty, then this becomes the VOX Level.		

2.26. Wave

Version: 0-0-1

Description: The Wave component provides audio waveform generation of various types including:

- Pulse
- Triangle
- Sawtooth
- Pulse_inverse
- Off - (no signal)
- Sine
- Square

Square wave selection generates a positive and negative-going square wave pattern with constant 50 percent duty cycle. Pulse and Inverse Pulse both produce a positive-going pulse type wave pattern. Pulse width can be adjusted by the pulse width control input

Both the amplitude and frequency can be controlled for all wave types and the width of the wave types can be controlled when applicable. If not applicable, the Width field is ignored. When the waveform frequency and/or the amplitude is less than or equal to 0, the wave component does not generate a signal.

Audio Inputs			
Wavetype	**Variables**	**Type**	**Default Value**
	kin	audio/waveshape	Off
	Description: Selects one of the available wave types (sine, triangle, sawtooth, square, pulse, inverted pulse). Valid inputs are 0 to 6 and correspond to the following: 0 - Off (no waveform) 1 - Sawtooth 2 - Triangle 3 - Sine 4 - Square 5 - Pulse 6 - Inverted Pulse Within the component, viewer values display as an enumerated type. If there is no connection to the input, the waveform type can be hard coded by setting the kin variable in the Wavetype block to the desired default value.		

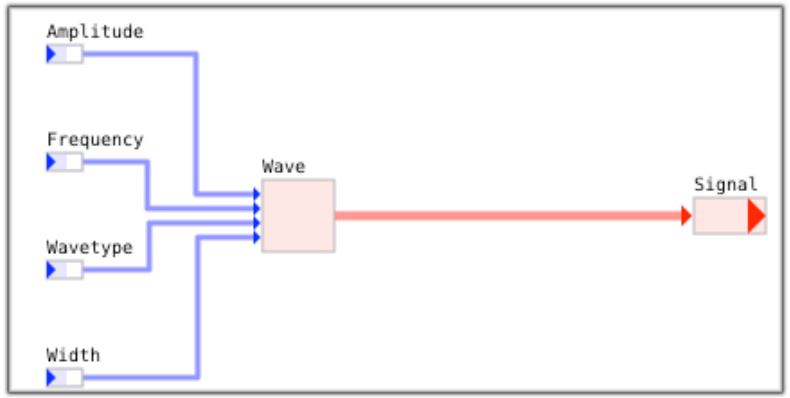


Figure 28: Wave Schematic

Control Inputs			
Amplitude	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Sets the amplitude of the selected waveform with a typical range of 0 - 1.0. If there is not a connection to the input, the amplitude can be hard coded by setting the kin variable in the Amplitude block to the desired default value. A scaling factor variable scale_factor is also available which can be set to scale the amplitude value. The final output (kout) is kin multiplied by Scale Factor.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Sets the frequency of the selected waveform in Hertz. The frequency allowable range is 0 - 1/2 the system audio rate (i.e. 0-24000). If there is not a connection to the input, the frequency can be hard coded by setting the kin variable in the frequency block to the desired default value. A scaling factor variable scale_factor is also available which can be set to scale the frequency value. The final output (kout) is kin multiplied by Scale Factor.		

Control Inputs			
Width	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Applicable only when Wavetype is set to either Pulse, Inverted Pulse or Square. When a pulse waveform is selected, this variable sets the duration time of the pulse in seconds. When an inverted pulse waveform is selected, this variable sets the duration time in seconds for which the waveform is at 0. Allowable range is 0.0 - n where n < 1/frequency of the current waveform. If 'n' is not within this range the resulting waveform output is 0 (off). When a square wave is selected, the variable sets the duty cycle ratio. Valid range is -1 to 1. A value of zero is a 50 percent duty cycle.		

Audio Outputs			
Signal	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: Output for the generated wave component signal. The signal may be linked to another component downstream to send the output elsewhere in the model.		

Internal Parameters			
Wave	**Variables**	**Type**	**Default Value**
	Gaindecay	basic/float32	0.010
	Width_units	Control/ Width_units	width_is_radio
	Description: **Gaindecay** - A decay factor used to smooth abrupt changes in amplitude. **Width_units** - Used to set the units of a signals width. Choices are Width_is_Ratio or Width_is_seconds.		

2.27. WavePulseMod

Version: 0-0-1

Description: The WavePulseMod is a component designed to synthesize a variety of complex tones such as pulsed beeps and chirps. For example, aircraft Radar Warning devices commonly produce these complex tones. The WavePulseMod component provides the functionality of the old WaveModulated component, but adds a PopFilter primitive. The AM input is usually a pulse signal to turn on and off the output signal. These abrupt transitions generate pops or clicks in the Signal output. The PopFilter smooths these transitions to eliminate these unintended pops and clicks, but it only works for a 0.0->1.0 amplitude pulse signal connected to the AM input.

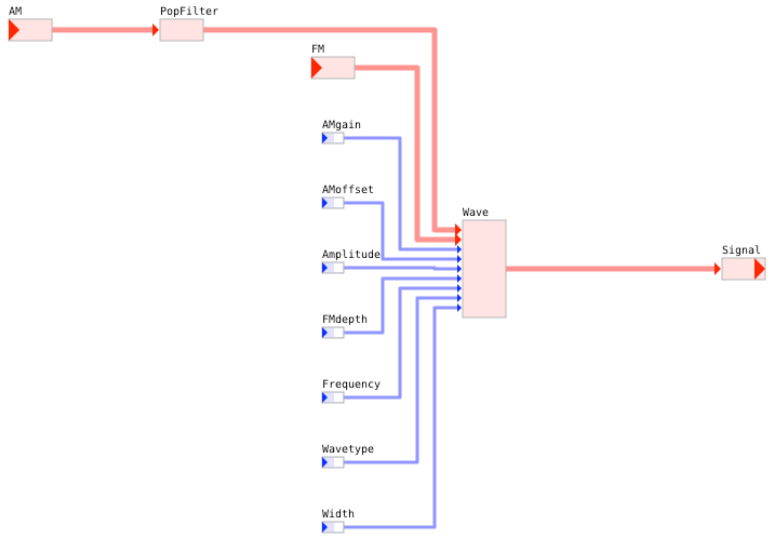
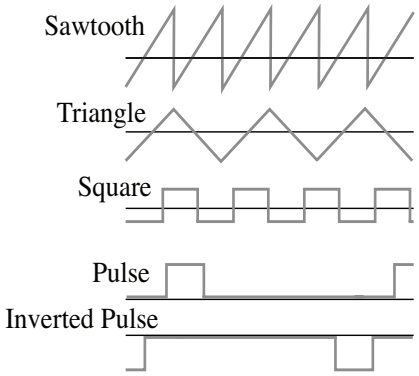


Figure 29: WavePulseMod

Audio Inputs			
AM	**Variables**	**Type**	**Default Value**
	"all of..."	audio	N/A
	Description: Connecting another waveform or other audio source to this field modulates the amplitude of the generated waveform in accordance with the following equation: Actual Amplitude = Amplitude * (AM + AM offset) Generally, the input source is another waveform but any audio source including sound file sources can be used. Although not necessary, audio sources which fall in the range of 0.0 to 1.0 work best.		
FM	**Variables**	**Type**	**Default Value**
	"all of..."	audio	N/A
	Description: Connecting another waveform or other audio source to this field modulates the frequency of the generated waveform in accordance with the following equation: Actual Frequency = Frequency * (1 + (FM Depth * FM Audio Input)) Generally, the input source is another waveform but any audio source including sound file sources can be used. Although no necessary, audio sources which fall in the range of -f to 24k/f, where f = base frequency, work best.		

Control Inputs			
AMgain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Gain factor applied to AM input signal. A scaling factor variable scale_factor is also available which can be set to scale the AMgain value.		
AMoffset	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: DC offset to add to AM input signal. A scaling factor variable scale_factor is also available which can be set to scale the amplitude value.		
Amplitude	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Sets the amplitude of the selected waveform. The typical range is 0 - 1.0. If there is no connection to the input, the amplitude can be hard coded by setting the kin variable in the Amplitude block to the desired default value. A scaling factor variable scale_factor is also available which can be set to scale the amplitude value.		

Control Inputs			
FMdepth	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Sets the frequency modulation depth value when the FM (Frequency Modulation) connection is used in accordance with the following equation Actual Frequency = Frequency * (1 + (FM Depth * FM Audio Input)) Values greater than or equal to 0.0 are acceptable. Usually, values fall in the range of 0 to 1.0, when used with a unity gain waveform source or other audio source whose sample values fall within the same range. **Note**: The product of modulation depth and the audio sample values from the audio source should not span a range greater than -1.0 to +1.0. Values beyond that range may cause unpredictable behavior. A scaling factor variable scale_factor is also available which can be set to scale the modulation depth value.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Sets the frequency of the selected waveform in Hertz. Allowable range is 0 - 1/2 the system audio rate (i.e. 0-24000). If there is no connection to the input, the frequency can be hard coded by setting the kin variable in the Frequency block to the desired default value. A scaling factor variable scale_factor is also available which can be set to scale the frequency value.		

Control Inputs			
Wavetype	**Variables**	**Type**	**Default Value**
	kin	basic/int8	0
	Description: Selects one of the available wave types (sine, triangle, sawtooth, square, pulse, inverted pulse). Valid inputs are from 0 - 6 and correspond to the following wave types: 0 - Off (no waveform) 1 - Sawtooth 2 - Triangle 3 - Sine 4 - Square 5 - Pulse 6 - Inverted Pulse  In the component viewer the values are displayed as enumerated types. If there is no connection to the input, the waveform type can be hard coded by setting the kin variable in the Wavetype block to the desired default value.		

Control Inputs			
Width	**Variables**	**Type**	**Default Value**
	kin	basic/float32	if pulse, pulse_inverse, square; default = 0.5
	Description: Width is applicable only when Wavetype is set to either Square, Pulse or Inverted Pulse. When a pulse waveform is selected, this variable sets the duration time of the pulse in seconds. When an inverted pulse waveform is selected, this variable sets the duration time in seconds for which the waveform is at 0. Allowable range is 0.0 - n, where n < 1/freq of the current waveform. If 'n' is not within this range, the resulting waveform output is 0 (off). If width_units = WIDTH_IS_RATIO (default), then width = duty cycle (i.e. ratio of asserted/de-asserted, e.g. 0.5 means half the time on, half the time off).		

Control Output			
Signal	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: Output for the generated wave component signal. The signal may be linked to another component downstream to send the output elsewhere in the model.		

Internal Parameters			
PopFilter.bypass	**Variables**	**Type**	**Default Value**
	n/a	boolean	FALSE
	Description: If TRUE, AM input signal is used unmodified. Use TRUE if you are not using a 0.0 -> 1.0 pulse to drive the AM input.		
Wave.width_units	**Variables**	**Type**	**Default Value**
	n/a	control/ width_units	n/a
	Description: WIDTH_IS_SECONDS(absolute), WIDTH_IS_RATIO(relative)		

2.28. WaveModulated

Version: 0-0-1

Description: This component is superseded by the WavePulseMod component. This component has been left in for backwards compatibility. The functionality and characteristics of this component have been captured by the WavePulseMod. Please see WavePulseMod for WaveModulated details.

3.0. Control Components

Control components provide the logic for driving the functions of the objects in the model.

This section provides the control components which include the following:

- ARC232_Freq
- BitOr
- BitToByte
- BitToByteControl
- BitToBytePrioritized
- BoolToFloat
- ByteToBit
- Comparator
- Control
- Counter
- DynamicTable8
- F15K_Ios_RadioParams
- FloatMultiplier
- FourChPTTDecoder
- FourChPTTRXSelect8
- InputBool
- InputInt16
- InputInt32
- InputInt64
- InputUInt8
- Krandf32
- LagFilter
- LogicTable4
- MathFunction2
- MathFunction3
- RangeCheck
- StereoPan3D
- Switch
- TableXY16
- TableXY32
- Uint8ToFloat

3.1. ARC232_Freq

Version: 0-0-1

Description: This is an outdated component that was previously used to detect Air Traffic Control frequencies. The component is no longer valid and should not be used.

3.2. BitOR

Version: 0-0-1

Description: Performs OR logic on two bits.

Truth Table		
Bit1	**Bit2**	**Output**
0	0	ResultBit
0	1	ResultBit
1	0	ResultBit
1	1	ResultBit

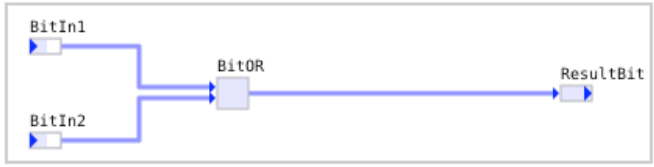


Figure 30: BitOR Schematic

Control Inputs			
BitIn1	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: First of two input bits to be OR'd together.		
BitIn2	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Second of two input bits to be OR'd together.		

Control Outputs			
ResultBit	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	n/a
	Description: Outputs result of OR functions, see the truth table in BitOR description.		

3.3. BitToByte

Version: 0-0-1

Description: The BitToByte component combines eight bits, expressed as a boolean type, into a single byte wide value. Each individual bit can either be hard coded to TRUE, FALSE, or dynamically set by either host controls or another linked component.

This component is typically used to create the bit mask input for input/control selectors on Mixers and Intercom Receive/Transmit Selectors such as those found on the Communications Panel (ComSing/ComQuad component). Each bit is also passed through a gate primitive that can be set. Also see BitToByteControl and BitToBytePrioritized.

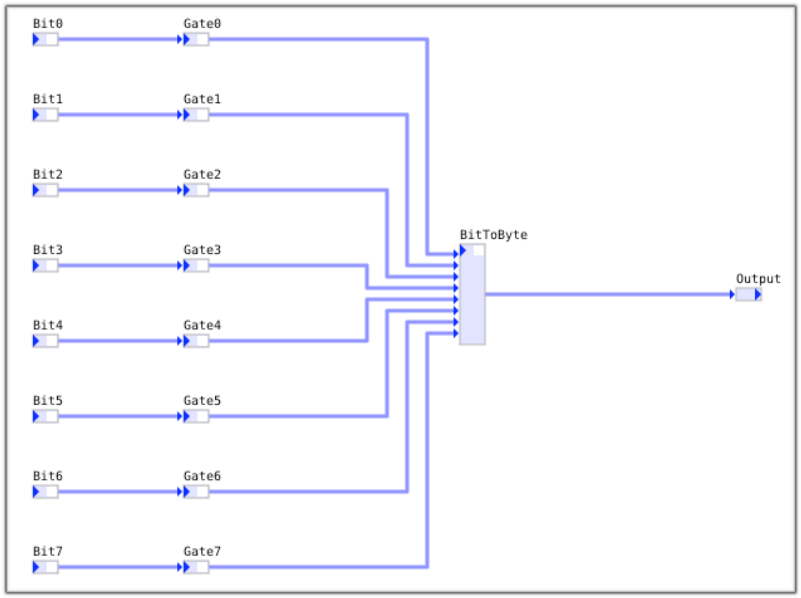


Figure 31: Bit To Byte Schematic

Control Inputs			
Bit0 through Bit7	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: These are the individual bits of the single byte word. Bit0 is the least significant bit and Bit7 is the most significant bit. Each control bit can be individually set to either TRUE or FALSE from a connected source. If a connection does not exist, the kin variable can be hard coded to a default value. These 8 bit values will dictate the single byte wide value of the output field.		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	Multiple Output Types		
	kout _bool	Type 1- basic/uint8	0
	kout_float	Type 2- basic/float32	0
	kout_int16	Type 3- basic/int16	0
		Type 4- basic/int32 Type 5- basic/int64 Type 6- basic/int8 Type 7- basic/uint16 Type 8- basic/uint32 Type 9- basic/uint64 Type 10- basic/uint8	
	Description: Single byte output, which is a result of the Bit0 through Bit7 inputs as well as the Gate0 through Gate7 values. The kout of this field can be different variable types. The type needed will vary upon the required input type to whatever object this output will be linked to. For example, if this component was driving the Control Input of the Mixer component, the Mixer's Control Input kin variable requires it to be an uint8, the corresponding output here should also be an uint8.		

Internal Parameters			
Gate 0 through Gate 7	**Variables**	**Type**	**Default Value**
	Value	basic/boolean	FALSE
	Description: The Gate0 through Gate7 primitives act as either Buffers or Inverters to each of their respective Bit Inputs depending on the state of the value variable. If the value variable in the Gate primitive is set to TRUE, then the Bit Input which is passed to the Gate primitive is inverted. If the value variable is set to FALSE, then the Bit Input which is passed to the Gate primitive is not inverted and is passed through unchanged.		

3.4. BitToByteControl

Version: 0-0-1

Description: A different version of the BitToByte component, this object allows the user to include an offset to the output as well as a control flag, which will force the output to zero. BitToByteControl still retains the 8 bit inputs that combine to form a single byte with multiple output vartypes.

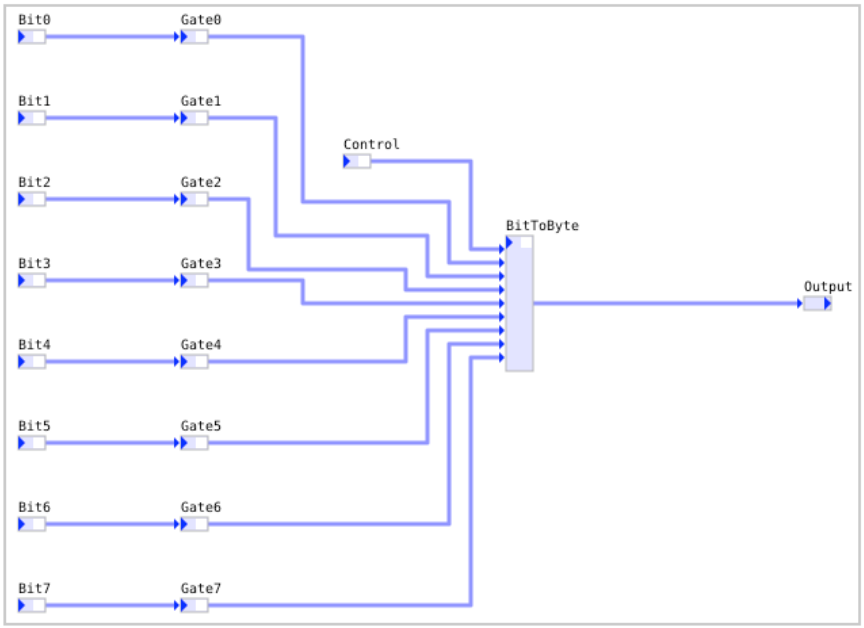


Figure 32: Bit To Byte Control Schematic

Control Inputs			
Bit0 through Bit7	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: These are the individual bits of the single byte word. Bit0 is the least significant bit and Bit7 is the most significant bit. Each control bit can be individually set to either TRUE or FALSE from a connected source. If a connection does not exist, the kin variable can be hard coded to a default value. These 8 bit values will dictate the single byte wide value of the output field.		
Control	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Setting this to false will force the output of the component to be zero, regardless of the input bits.		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	Multiple Output Types		
	kout _bool	Type 1- basic/uint8	0
	kout_float	Type 2- basic/float32	0
	kout_int16	Type 3- basic/int16	0
		Type 4- basic/int32 Type 5- basic/int64 Type 6- basic/int8 Type 7- basic/uint16 Type 8- basic/uint32 Type 9- basic/uint64 Type 10- basic/uint8	
	Description: Single byte output, which is a result of the Bit0 through Bit7 inputs as well as the Gate0 through Gate7 values. Output will ignore all of the bits, gates, and offset if the control flag is set to false and instead it will give a result of zero. The kout of this field can be different variable types. The type needed will vary upon the required input type to whatever object this output will be linked to. For example, if this component was driving the Control Input of the Mixer component, the Mixer's Control Input kin variable requires it to be an uint8, the corresponding output here should also be an uint8.		

Internal Parameters			
Gate 0 through Gate 7	**Variables**	**Type**	**Default Value**
	Value	basic/boolean	FALSE
	Description: The Gate0 through Gate7 primitives act as either Buffers or Inverters to each of their respective Bit Inputs depending on the state of the value variable. If the value variable in the Gate primitive is set to TRUE, then the Bit Input which is passed to the Gate primitive is inverted. If the value variable is set to FALSE, then the Bit Input which is passed to the Gate primitive is not inverted and is passed through unchanged.		
BitToByte ->Offset	**Variables**	**Type**	**Default Value**
	Offset	basic/int32	0
	Description: The BitToByte Offset changes the final output of the component by the number indicated in Offset. A positive number will add into the final value, a negative number will subtract from it. The Control flag must be TRUE for the Offset to take effect.		

3.5. BitToBytePrioritized

Version: 0-0-1

Description: This is a modified version of the BitToByte component, a Prioritized BitToByte will ignore all lower bits once it detects the most significant high bit. As a result, the object will only ever produce numbers that are a power of two.

For example, if Bit5, Bit6, and Bit7 are all set to True, the final output will be 128. The component will ignore bits 5 and 6 and create a byte with only Bit7 set high. Also see BitToByte and BitToByteControl.

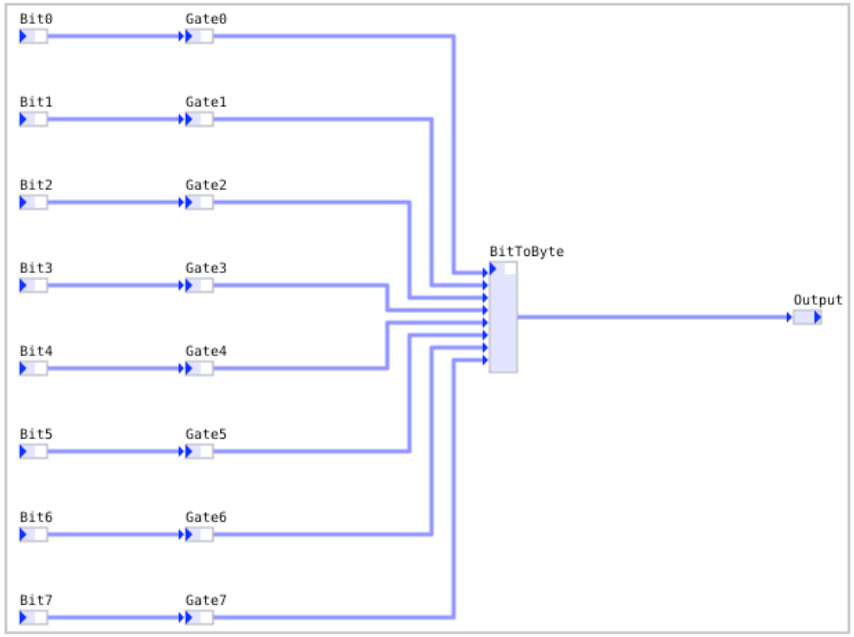


Figure 33: Bit To Byte Prioritized Schematic

Control Inputs			
Bit0 through Bit7	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: These are the individual bits of the single byte word. Bit0 is the least significant bit and Bit7 is the most significant bit. Each control bit can be individually set to either TRUE or FALSE from a connected source. If a connection does not exist, the kin variable can be hard coded to a default value. These 8 bit values will dictate the single byte wide value of the output field. Note that once a bit is set to TRUE all bits with less significance will be treated as FALSE, regardless of their actual setting.		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	Multiple Output Types		
	kout _bool	Type 1- basic/uint8	0
	kout_float	Type 2- basic/float32	0
	kout_int16	Type 3- basic/int16	0
		Type 4- basic/int32 Type 5- basic/int64 Type 6- basic/int8 Type 7- basic/uint16 Type 8- basic/uint32 Type 9- basic/uint64 Type 10- basic/uint8	
	Description: Single byte output, which is a result of the Bit0 through Bit7 inputs as well as the Gate0 through Gate7 values. The kout of this field can be different variable types. The type needed will vary upon the required input type to whatever object this output will be linked to. For example, if this component was driving the Control Input of the Mixer component, the Mixer's Control Input kin variable requires it to be an uint8, the corresponding output here should also be an uint8.		

Internal Parameters			
Gate 0 through Gate 7	**Variables**	**Type**	**Default Value**
	Value	basic/boolean	FALSE
	Description: The Gate0 through Gate7 primitives act as either Buffers or Inverters to each of their respective Bit Inputs depending on the state of the value variable. If the value variable in the Gate primitive is set to TRUE, then the Bit Input which is passed to the Gate primitive is inverted. If the value variable is set to FALSE, then the Bit Input which is passed to the Gate primitive is not inverted and is passed through unchanged.		

3.6. BoolToFloat

Version: 0-0-1

Description: This component takes a boolean input variable and does the type conversion to float, giving an output of 1.0 if the input is TRUE and a value of 0.0 if the input is FALSE. Offsets can be included from within the object. Also see InputBool.

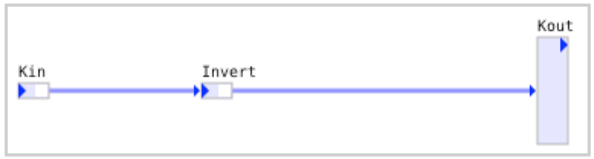


Figure 34: Bool To Float Schematic

Control Inputs			
Kin	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	False
	Description: This is where the input Boolean variable should be linked for type conversion. A local invert flag can be set from within the component.		
Invert	**Variables**	**Type**	**Default Value**
	value	basic/boolean	FALSE
	Description: Flips the kin value. This flag can be linked to and driven via the host controls.		

Control Outputs			
Kout	**Variables**	**Type**	**Default Value**
	kout	basic/float32	0.0
	Description: The input value converted to a float. A value of TRUE will set kout to 1.0 and a value of FALSE will set kout to 0.0.		

Internal Parameters			
Kout	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	kin_int	basic/int32	0
	kin_uint16	basic/unit16	0
	kin_uint32	basic/uint32	0
	kin_uint64	basic/uint64	0
	kin_uint8	basic/uint8	0
	Description: The kin parameters will add an offset to the kout value. Note that all of the vartypes are summed together to produce a single offset. For example, setting kin_int to 5 and kin_uint16 to 3 will cause kout to have an offset of 8.0.		

3.7. ByteToBit

Version: 0-0-1

Description: The ByteToBit component converts a single input byte into its eight individual bit values. The BytetoBit component also has a Control Input which acts as an on/off switch turning the decoding On when TRUE and Off when FALSE.

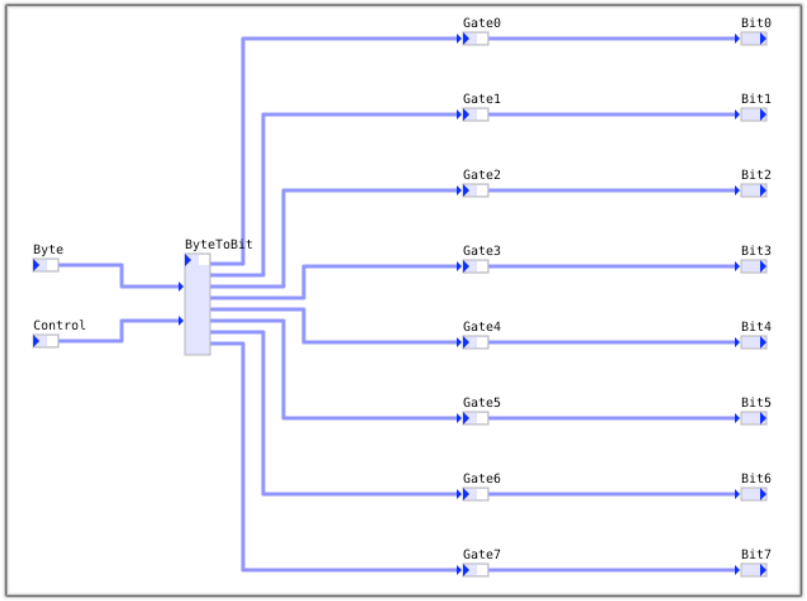


Figure 35: Byte To Bit Schematic

Control Inputs			
Byte	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: A Single byte Input that can either be hard coded or dynamically set by a host control or another linked component. Valid range is from 0 to 255. If the Control input is set to TRUE, then the input byte value will be decoded and the Output Bits will be appropriately switched to TRUE or FALSE.		
Control	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: The Control Input acts as an on/off switch for the BytetoBit component. A kin value of TRUE enables the decoding of a Byte into its bits while a kin value of FALSE disables the decoding and the output bits will default to their initialized state. This Control Input is useful in conjunction with the internal Gate0 through Gate7 primitives. The Gate primitives can be used to change the FALSE default state of the Output Bits to default to TRUE. In this situation, one can create any combination of initial bit states. Turning the Control Input to FALSE effectively returns this component to its default state and consequently the component inputs linked to the Output Bits can be changed back to their defaults.		
Gate0 through Gate7	**Variables**	**Type**	**Default Value**
	Value	basic/boolean	FALSE
	Description: The Gate0 through Gate7 control inputs act as either buffers or inverters to each of their respective Bit Outputs depending on the state of the value variable. If the value variable in the Gate input is set to TRUE, then the corresponding Output Bit is inverted. If the value variable is set to FALSE, then the corresponding Output Bit will match its state based upon the Input Byte value.		

Control Outputs			
Bit0 through Bit7	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	n/a
	Description: These are the 8 individual bit outputs that are the result of the input byte word and the Gate0 through Gate7 primitives. Bit0 is the least significant bit and Bit7 is the most significant bit.		

3.8. Comparator

Version: 0-0-1

Description: The comparator function consists of 2 inputs, X and Y. X is compared against the value of Y. X can either be equal to, less than, or greater than Y. Based upon the result of the comparison between X and Y, the output will be one of the corresponding user defined values for each of the three possible results.

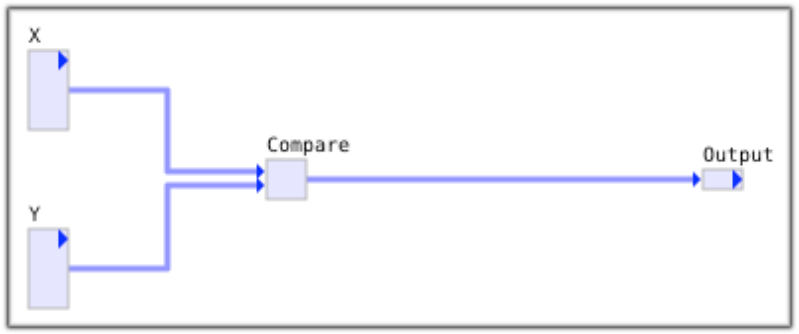


Figure 36: Comparator Schematic

Control Inputs			
X	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kin_uint	basic/uint32	0
	Type	basic/vartype	Float
	Description: The X Input is the value that will be compared against the value of Y. There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The 'Type' variable is used to select which one of the 4 input kin's will be used in the comparison.**		

Control Inputs			
Y	**Variables**	**Type**	**Default Value**
	kint	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kin_uint	basic/uint32	0
	Type	basic/vartype	Float
	Description: Y is the threshold value that X is compared against. There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The 'Type' variable is used to select which one of the 4 input kin's will be used in the comparison.**		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_unit	basic/uint8	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_unit16	basic/uint16	n/a
	kout_unit32	basic/uint32	n/a
	kout_unit64	basic/uint64	n/a
	kout_unit8	basic/uint8	n/a
	Description: The output is a result of the comparison of X against a given threshold (Y) and the values that are set in the Internal Parameter Compare (see Internal Parameters - Compare). The values specified in the compare primitive will dictate the output value based on the less than, equal to, or greater than result of the X comparison against Y.		

Internal Parameters			
Compare	**Variables**	**Type**	**Default Value**
	EqualTo	basic/float32	0
	Gain	basic/float32	1
	GreaterThan	basic/float32	0
	LessThan	basic/float32	0
	Scale->Scale (0)	basic/float32	1
	Scale->Scale (1)	basic/float32	1
	Description: The compare primitive contains the 2 inputs X and Y. These inputs appear as kin[0] and kin[1], respectively, within the primitive. Each input can be scaled (multiplied) by the Scale [0] and Scale[1] variables. (X, or kin[0] scaled by Scale[0] and Y, or kin[1], scaled by Scale[1]). Values are defined for the EqualTo, GreaterThan, and LessThan variables. Based on the value of X compared to Y one of these three values will get multiplied by the value set in the Gain variable and passed on to the Output.		

3.9. Control

Version: 0-0-1

Description: The Control component does type conversion for the float variable type. A float input is accepted and nine other variable types are given as an output.

Figure 37: Control Schematic

Control Inputs			
Kin	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: The kin control links the float value to undergo type conversion to kin. The value may be set inside the component or linked from elsewhere in the model.		

Control Outputs			
Kout	**Variables**	**Type**	**Default Value**
	kout_bool	basic/boolean	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_uint16	basic/uint16	n/a
	kout_uint32	basic/uint32	n/a
	kout_uint64	basic/uint64	n/a
	kout_uint8	basic/uint8	n/a
	Description: This returns the value of kin after proper type conversion. The value can be linked downstream to other components.		

3.10. Counter

Version: 0-0-2

Description: The Counter provides a time-based general purpose event or continuous ramping function. In single shot (non-continuous) mode, the Counter provides an externally triggered function lookup suitable for amplitude or frequency control of signal sources for explosions, touch-down thumps, etc., where a pre-recorded sound file is not available or applicable. When set to continuous it provides a table driven modulation of waveforms, where the modulation rate is slower than the overall model execution rate (e.g. 0 to 50 hertz). The counter can also be used for generic time-based control of other components.

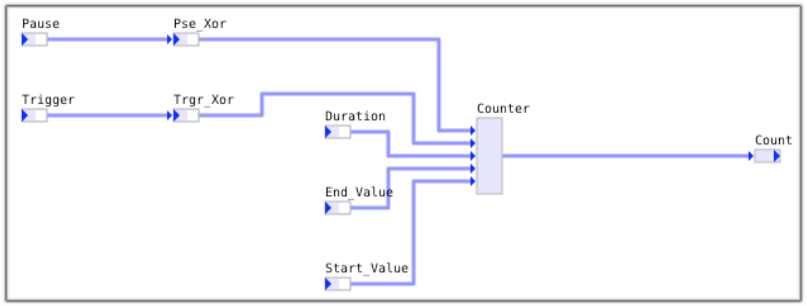


Figure 38: Counter Schematic

Control Inputs			
Duration	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Length of time in seconds for the counter to go from Start_Value to End_Value (if UpDown is FALSE) or from Start_Value to End_Value and back to Start_Value (if UpDown is TRUE). Updown is described in the Counter's internal parameters table. Units are in seconds.		

Control Inputs			
End_value	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Ending value for counter cycle.		
Pause	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: The counter will pause at its current value when pause is TRUE, until Pause is set back to False.		
PSE_Xor	**Variables**	**Type**	**Default Value**
	Value	basic/boolean	FALSE
	Description: Inverts the polarity of the Pause input if set to TRUE.		
Start_ value	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Starting value for counter cycle.		
Trgr_Xor	**Variables**	**Type**	**Default Value**
	Value	basic/boolean	FALSE
	Description: Inverts the polarity of the Pause input if set to TRUE.		

Control Inputs			
Trigger	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: When the trigger is set to a value of TRUE, counting is initiated. If the counter is in continuous mode, counting will continue so long as the trigger remains TRUE. If the counter is in one-shot mode, the count will go from start_value to end_value once on every FALSE to TRUE transition of the trigger.		

Control Outputs			
Count	**Variables**	**Type**	**Default Value**
	kout_bool	basic/uint8	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_unit16	basic/uint16	n/a
	kout_unit32	basic/uint32	n/a
	kout_unit64	basic/uint64	n/a
	kout_unit8	basic/ubasic/ int32int8	n/a
	Description: The output value of the counter as it runs from the start to end value. It can be used to drive other controls and linked downstream.		

Internal Parameters			
Counter	**Variables**	**Type**	**Default Value**
	Continuous	basic/boolean	FALSE
	CountAll	basic/boolean	FALSE
	DefaultVal	basic/float32	0.0
	Updown	basic/boolean	FALSE
	Delay	basic/float32	0.0
	Description: **Continuous** - If TRUE, counter will start a new cycle after completing a cycle as long as Trigger remains TRUE. **CountAll** - If TRUE, counter will always complete cycle even if trigger is removed. **DefaultVal** - Default output value when counter is inactive. (A paused counter is still considered active as long as pause was set to TRUE while the counter was running.) **UpDown** - If TRUE, counter will ramp up to EndVal then ramp down to StartVal within the cycle. **Delay** - Adds a time between cycles if the counter is in continuous mode. The counter will hold at the StartValue during the delay. Units are in seconds.		

3.11. DynamicTable8

Version: 0-0-1

Description: Dynamic Table 8 takes an input byte and runs a bit-wise table lookup. The bit with the least significance will cause the output to be based on the corresponding value for that bit. For example, an input value of 128 (1000 0000 in binary) will return the Value8 in the lookup table, whereas an input of 129 (1000 0001 in binary) will return Value1 in the lookup table. This component differs from other table components because the values in the lookup table can be driven dynamically from the host. Also see LogicTable4, TableXY16, and TableXY32.

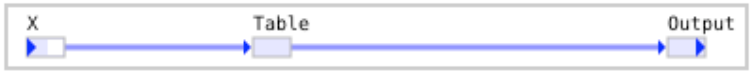


Figure 39: Dynamic Table 8 Schematic

Control Inputs			
X	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The basic input into the table. Values driven into X are transformed via the table parameters. The table can be thought of as a function with X as an input.		
Table	**Variables**	**Type**	**Default Value**
	gain	basic/float32	1.0
	Description: The output of the table is multiplied by gain before leaving the component. Gain is set internally or driven from elsewhere in the model.		
	value1-value7	basic/float32	0.0
	Description: The bit-wise lookup values for the table. Value1 corresponds to the least significant bit of the input, and Value7 corresponds to the most significant bit. The output of the table is based on the Value number for the least significant active bit.		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_bool	basic/boolean	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_uint16	basic/uint16	n/a
	kout_uint32	basic/uint32	n/a
	kout_uint64	basic/uint64	n/a
	kout_uint8	basic/uint8	n/a
	Description: Returns the final value of the table in multiple variable types. The final value is equal to the Value number corresponding to the least significant active bit of the input X, and then multiplied by gain.		

3.12. F15K_Ios_RadioParams

Version: 0-0-1

Description: The Radio Parameter component is a specialized lookup table suited for driving radios. An input index chooses between seven different sets, each of which contains a frequency, mode, and a secure flag. A host computer can drive all frequencies, modes, and secure flags, or the values can be set internally. The sets of parameters are selected based upon the table below.

TuneTo	Parameter Set
0	IOS_Radio
1	WST_Rad1
2	WST_Rad2
3	CPT1_Rad1
4	CPT1_Rad2
5	CPT2_Rad1
6	CPT2_Rad2

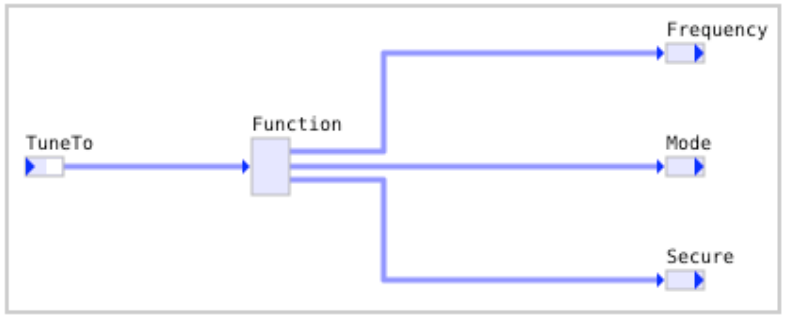


Figure 40: F15K_Ios_RadioParams Schematic

Control Inputs			
TuneTo	**Variables**	**Type**	**Default Value**
	kin	basic/int8	0
	Description: This input chooses which set of radio parameters that will be returned based on the table (shown in the Description). TuneTo ranges from 0-6.		
Function	**Variables**	**Type**	**Default Value**
	Ios_Radio_Freq	basic/int32	0
	Ios_Radio_Mode	basic/int8	0
	Ios_Radio_Secure	basic/int8	0
	Description: Setting the TuneTo index to 0 will cause Ios_Radio_Freq, Mode and Secure to be returned by the component. Each parameter corresponds to settings that can be used for Radio components. These values can be set internally or driven by the host.		
	WST_Rad1_Freq	basic/int32	0
	WST_Rad1_ Mode	basic/int8	0
	WST_Rad1_ Secure	basic/int8	0
	Description: Setting the TuneTo index to 1 will cause the WST_Rad1_Freq, Mode, and Secure to be returned by the component.		

Control Inputs			
Function continued	WST_Rad2_Freq	basic/int32	0
	WST_Rad2_ Mode	basic/int8	0
	WST_Rad2_ Secure	basic/int8	0
	Description: Setting the TuneTo index to 2 will cause the WST_Rad2_Freq, Mode, and Secure to be returned by the component.		
	CPT1_Rad1_Freq	basic/int32	0
	CPT1_Rad1_ Mode	basic/int8	0
	CPT1_Rad1_ Secure	basic/int8	0
	Description: Setting the TuneTo index to 3 will cause CPT1_Rad1_Freq, Mode, and Secure to be returned by the component.		
	CPT1_Rad2_Freq	basic/int32	0
	CPT1_Rad2_ Mode	basic/int8	0
	CPT1_Rad2_ Secure	basic/int8	0
	Description: Setting the TuneTo index to 4 will cause CPT1_Rad2_Freq, Mode, and Secure to be returned by the component.		
	CPT2_Rad1_Freq	basic/int32	0
	CPT2_Rad1_ Mode	basic/int8	0
	CPT2_Rad1_ Secure	basic/int8	0
	Description: Setting the TuneTo index to 5 will cause CPT2_Rad1_Freq, Mode, and Secure to be returned by the component.		

Control Inputs			
Function continued	CPT2_Rad2_Freq	basic/int32	0
	CPT2_Rad2_ Mode	basic/int8	0
	CPT2_Rad2_ Secure	basic/int8	0
	Description: Setting the TuneTo index to 6 will cause CPT2_Rad2_Freq, Mode, and Secure to be returned by the component.		

Control Outputs			
Frequency	**Variables**	**Type**	**Default Value**
	kout	basic/int32	0
	Description: Returns the appropriate frequency based on the TuneTo index. This value can be linked directly to the MainFrequency input of a radio component.		
Mode	**Variables**	**Type**	**Default Value**
	kout	basic/int8	0
	Description: Returns the appropriate mode based on the TuneTo index. This value can be linked directly to the Main-Mode input of a radio component.		
Secure	**Variables**	**Type**	**Default Value**
	kout	basic/int8	0
	Description: Returns the appropriate secure mode flag based on the TuneTo index. This value can be linked directly to the SecureMode input of a radio component.		

3.13. FloatMultiplier

Version: 0-0-1

Description: The FloatMultiplier component accepts two input float values and returns their product.

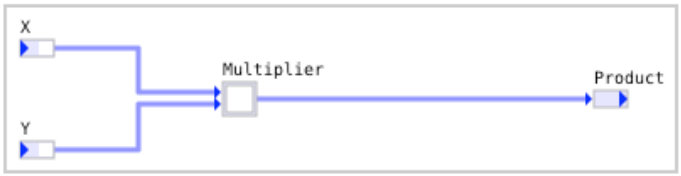


Figure 41: Float Multiplier Schematic

Control Inputs			
X	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/flaot32	0.0
	Description: X is one of the two input values to be multiplied. A scale factor that can only be set internally multiplies with the kin value		
Y	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Y is one of the two input values to be multiplied. A scale factor that can only be set internally multiplies with the kin value.		

Control Outputs			
Product	**Variables**	**Type**	**Default Value**
	kout	basic/float32	0.0
	Description: The final output of the component. Returns the product of X and Y once they have been multiplied together.		

3.14. FourChPTTDecoder

Version: 0-0-1

Description: The Four Channel PTT Decoder component is used in conjunction with the 4 Channel PTT switch. The 4 Channel PTT switch outputs an analog resistive value for each of its four positions. The analog value can then be linked into this decoder component to be used with a lookup table to output a one byte value in either byte or individual bit format. The lookup table for the FourChPTTDecoder is defaulted for use with the resistive values of an ASTi 4 channel PTT switch. The outputs are 1, 2, 4 and 8 for switch positions 1, 2, 3, and 4 respectively. The default outputs can be changed by editing the LevelMap primitive.

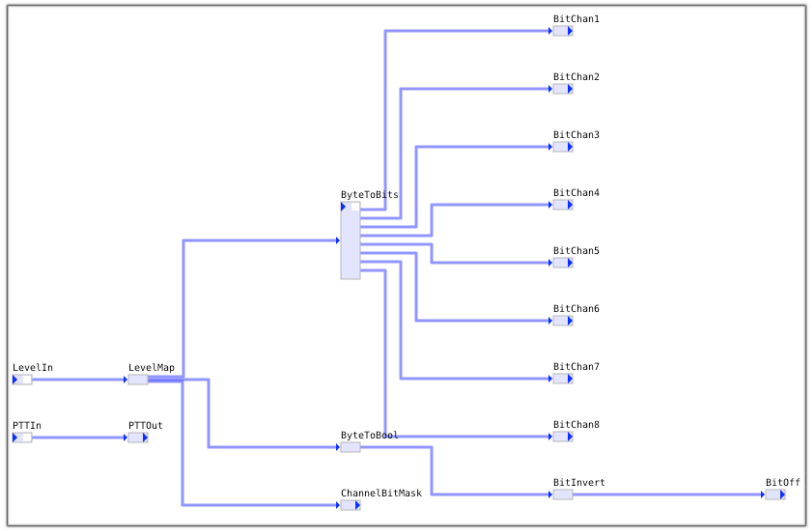


Figure 42: Four Channel PTT Decoder Schematic

Control Inputs			
LevelIn	**Variables**	**Type**	**Default Value**
	kout	basic/float32	0
	Description: Where the 4 channel PTT switch input is connected to the component. This input requires a float32 value. In order for the 4 position switch to output an analog float32 value the digital input of the Iris object must be changed to analog.		
PTTIn (Press to Talk Input)	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Invert	basic/boolean	FALSE
	Description: The kin variable takes a boolean input and directly maps it to the PTTout output. The invert variable inverts the polarity of the input if set to TRUE.		

Control Outputs			
BitChan1 through BitChan8	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	kin
	Description: Outputs the individual bits of the byte word produced from the LevelMap. BitChan1 is the least significant bit and BitChan7 is the most significant bit. See the BytetoBit component for a more detailed description of the BytetoBit operation.		
BitOff	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	kin
	Description: Outputs a FALSE when the LevelMap output value is greater than 1.		

Control Outputs			
Channel Bitmask	**Variables**	**Type**	**Default Value**
	kout	Control/ptt_channel	kin
	Description: Outputs the Byte value of the LevelMap output. The output is displayed as a name corresponding to each of the channels. These names have no effect on the model operation and are solely for the convenience of the user. The value behind the name is an uint8 value.		
PTTOut	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	kin
	Description: Output the boolean value from the PTTIn input.		

Internal Parameters			
LevelMap	**Variables**	**Type**	**Default Value**
	Table->Data[0]->X	basic/float32	0
	Table->Data[0]->Y	basic/float32	0
	Table->Data[1]->X	basic/float32	70.0
	Table->Data[1]->Y	basic/float32	1
	Table->Data[2]->X	basic/float32	101.0
	Table->Data[2]->Y	basic/float32	2
	Table->Data[3]->X	basic/float32	146.0
	Table->Data[3]->Y	basic/float32	4
	Table->Data[4]->X	basic/float32	191.0
	Table->Data[4]->Y	basic/float32	8
	Table->Data[5]->X	basic/float32	221.0
	Table->Data[5]->Y	basic/float32	0
	Table->Data[6-15]->X	basic/float32	0
	Table->Data[6-15]->Y	basic/float32	0
	Description: The LevelMap is a standard TableXY16 lookup table that has been pre configured for typical 4 channel PTT switch values. The LevelIn Input value is compared against the values set in the Data[1-16]->X variables. The Y value corresponding to the X value that is closest to the LevelIn value is the output. The X and Y values are rounded down to the nearest integer value in order for the output to be in Byte/Bit format.		

3.15. FourChPTTRXSelect8

Version: 0-0-2

Description: The Four Channel PTT Rx Select 8 component is intended to enhance the functionality of the 4 channel PTT switch. In conjunction with the FourChPTTDecoder and most commonly a communications panel, this component can dynamically change the bitmask selections driven into a communications panel by the 4 switch positions of the PTT switch.

Normally, the outputs of the 4 channel PTT are static byte words (if using the FourchPTTDecoder component) and if used as a bit mask only 4 masks are possible. With the FourChPTTRXSelect8, the 4 channel PTT switches output is used to drive a multiplexer whose inputs can be driven dynamically from a host.

The FourChPTTRXSelect8 also has an override input which can be used to override the multiplexer (and thus the 4 channel PTT switch) output at anytime. There is an override for both the Input (Tx) and Output (Rx) selectors used to drive the communication panel selections.

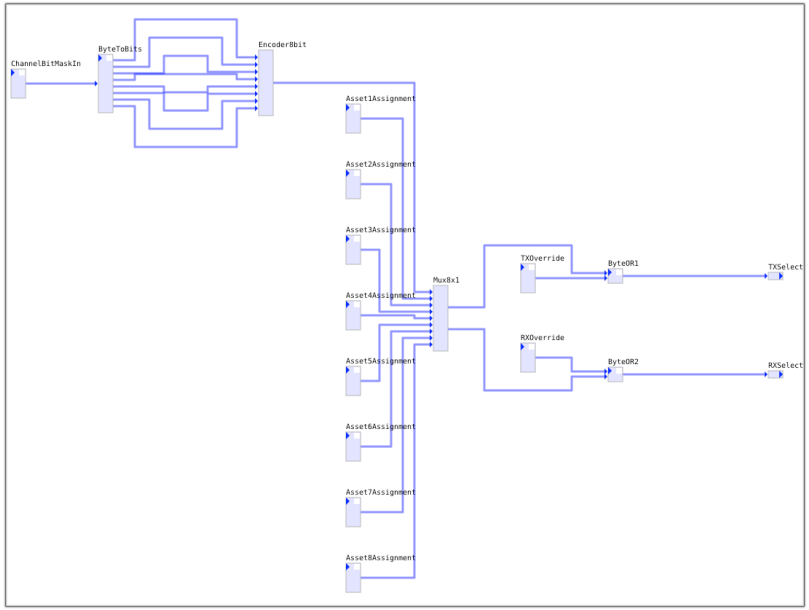


Figure 43: Four Channel PTT Rx Select 8 Schematic

Control Inputs			
Asset1 Assignment through Asset8 Assignment	**Variables**	**Type**	**Default Value**
	kin	Basic/Unint8	0
	Description: The asset assignments are the inputs to the multiplexer. They are single Byte words that are selected via the value of the ChannelBitMaskin input. The asset assignments are routed to both the TxSelect and RxSelect outputs. The byte words of the asset assignments are commonly used as bit masks which drive Tx and Rx selections in communication panels. The values of each of the asset assignments can be hardcoded or dynamically set from external components or driven from host controls.		
Channel Bit Maskin	**Variables**	**Type**	**Default Value**
	kin	Basic/Unint8	0
	Description: Single Byte word used to drive the input of the multiplexer. This input is commonly linked from the ChannelBitMask output of the FourChPTTDecoder.		
RX Override	**Variables**	**Type**	**Default Value**
	kin	Basic/Unint8	0
	Description: Contrary to the name of the input, the value input here is BitOR'd with the output of the multiplexer. Instead of overriding the output, this input can only add additional selections to the Rx output mask.		
TX Override	**Variables**	**Type**	**Default Value**
	kin	Basic/Unint8	0
	Description: Contrary to the name of the input, the value input here is BitOR'd with the output of the multiplexer. Instead of overriding the output, this input can only add additional selections to the Tx output mask.		

Control Outputs			
RX Select	**Variables**	**Type**	**Default Value**
	kout	Basic/Unint8	kin
	Description: Outputs the selected Asset Assignment (selected via the ChannelBitMaskin input). This output is commonly used as the bitmask for the output selector for the ComSing component.		
TX Select	**Variables**	**Type**	**Default Value**
	kout	Basic/Unint8	kin
	Description: Outputs the selected Asset Assignment (selected via the ChannelBitMaskin input). This output is commonly used as the bitmask for the input selector for the ComSing component.		

3.16. InputBool

Version: 0-0-1

Description: The InputBool component does type conversion for the boolean variable type. A boolean input is accepted and nine other variable types are given as an output.

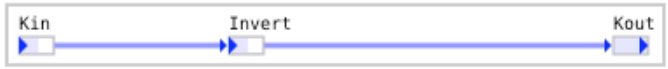


Figure 44: Input Bool Schematic

Control Inputs			
Kin	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	invert	basic/boolean	FALSE
	Description: Link the boolean value to undergo type conversion to kin. The value may be set inside the component or linked from elsewhere in the model. Setting the invert flag to TRUE will flip the value of the input.		
Invert	**Variables**	**Type**	**Default Value**
	value	basic/boolean	FALSE
	Description: If value is set to TRUE, the final output of the component will be inverted, i.e. an input of TRUE will be FALSE and an input of FALSE will be TRUE. Invert>value is functionally the same as the invert flag under kin, except value can be driven from outside the component.		

Control Outputs			
Kout	**Variables**	**Type**	**Default Value**
	kout_bool	basic/boolean	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_uint16	basic/uint16	n/a
	kout_uint32	basic/uint32	n/a
	kout_uint64	basic/uint64	n/a
	kout_uint8	basic/uint8	n/a
	Description: Returns the value of kin after proper type conversion. Value can be linked downstream to other components.		

3.17. InputInt16

Version: 0-0-1

Description: The InputInt16 component does type conversion for a 16-bit signed integer variable type. An int16 input is accepted and nine other variable types are given as an output.

Figure 45: InputInt 16 Schematic

Control Inputs			
Kin	**Variables**	**Type**	**Default Value**
	kin	basic/int16	FALSE
	offset	basic/int16	FALSE
	test	control/testmode	TEST_OFF
	testvalue	basic/int16	0
	Description: Link the int16 value to undergo type conversion to kin. The value may be set inside the component or linked from elsewhere in the model. The offset is added into the kin value before type conversion. Setting test to TEST_ON will cause kin to be ignored and the testvalue will be used in its place.		

Control Outputs			
Kout	**Variables**	**Type**	**Default Value**
	kout_bool	basic/boolean	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_uint16	basic/uint16	n/a
	kout_uint32	basic/uint32	n/a
	kout_uint64	basic/uint64	n/a
	kout_uint8	basic/uint8	n/a
	Description: Returns the value of kin after proper type conversion. Value can be linked downstream to other components.		

3.18. InputInt32

Version: 0-0-1

Description: The InputInt32 component works exactly as the InputInt16 component, except that it accepts a 32-bit signed integer, instead of a 16-bit input.

For more details see InputInt16.

Figure 46: InputInt 32 Schematic

3.19. InputUint64

Version: 0-0-1

Description: The InputUint64 component works exactly as the InputInt16 component, except that it accepts a 64-bit unsigned integer, instead of a 16-bit signed input.

For more details see InputInt16.

Figure 47: InputInt 64 Schematic

3.20. InputUint8

Version: 0-0-1

Description: The InputUint8 component works exactly as the InputUint8 component, except that accepts an 8-bit unsigned integer, instead of a 16-bit signed input.

For more details see InputInt16.

Figure 48: Input Uint 8 Schematic

3.21. Krandf32

Version: 0-0-1

Description: The Random Float32 component is a random number generator. A new float32 value will be produced by the component across a specified range of numbers for use elsewhere in the model. For example, a Krandf32 can be attached to the gain of a noise filter, which can add a random static that will cut in and out of an audio signal. Another possible use would be linking the random generator to a component that is simulating lightning strikes. Krandf32 can drive a trigger that will make lightning strikes occur randomly during an exercise.

Controls are included for the range of random numbers and also how often a random number is produced. The output of Krandf32 follows the formula:

Output= [(random number)*Scale_Factor + Range_Offset]*Out_Gain

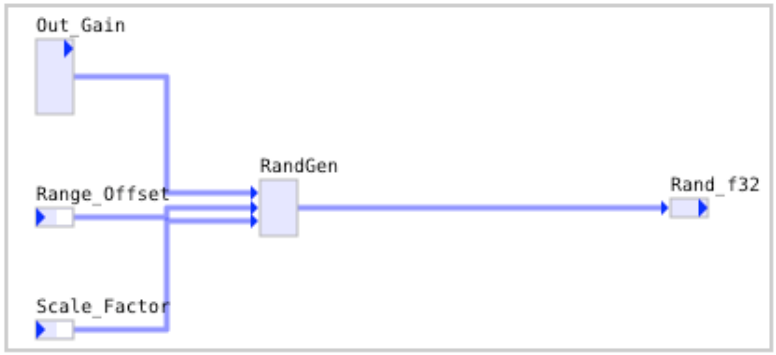


Figure 49: Krandf32 Schematic

Control Inputs			
Out_Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kin_uint	basic/uint32	0
	type	basic/vartype	Float
	Description: Out_Gain multiplies with the output of the random number generator to produce the final output of the component. Setting Out_Gain to zero will effectively turn off the component. Multiple vartypes are accepted, but the type parameter must be set to select which one of the four inputs will be used by the component.		
Range_ Offset	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Range_Offset is added to the random number before it is multiplied by Out_Gain.		
Scale_ Factor	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Scale_Factor is multiplied with the random number before Range_Offset is added. Setting it to zero will effectively turn off the component.		

Control Outputs			
Rand_f32	**Variables**	**Type**	**Default Value**
	kout_bool	basic/boolean	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_uint16	basic/uint16	n/a
	kout_uint32	basic/uint32	n/a
	kout_uint64	basic/uint64	n/a
	kout_uint8	basic/uint8	n/a
	Description: Rand_f32 is the final output of the component. Note that Rand_f32 = [(random number) *Scale_factor + Range_Offset]*Out_Gain. Though the object produces a random float number, there is local type conversion to give multiple types of outputs.		

Internal Parameters			
RandGen	**Variables**	**Type**	**Default Value**
	high	basic/float32	1.0
	low	basic/float32	0.0
	period	basic/uint32	0
	seed	basic/uint32	0
	Description: **High** provides the top range for the random number generator. Setting it to 10 ensures that the highest possible random number to be produced will 10. Note that this is before the scaling and offset is applied. **Low** provides the bottom of the range of random numbers. Setting it to zero will keep any negative numbers from being produced. As with high, the range is set before scaling or offsets. **Period** sets how often it generates a random number. Keeping the value at zero will produce a new number every 10msec. Setting Period to 100 will produce a new number every second. **Seed** effects how the random numbers are generated. Since the numbers produced are actually pseudo-random, patterns can develop when multiple Krandf32 components interact with each other. Setting the seed to different values will help to remove and noticeable patterns.		

3.22. LagFilter

Version: 0-0-1

Description: The lag filter provides a simple slew-rate limiting filter, which is useful for fade-in and fade-out effects. The filter can be used to smooth harsh transitions within a model and cause a quick numerical jump to change slower over time.

The filter function is defined as:

Y N=Y N-1+ K (X N- Y N-1)

Where:

X N = new input value

Y N = new output value

Y N-1 = last frame's output value

K = Attack const If X N > Y N-1

K = Decay const If X N < Y N-1

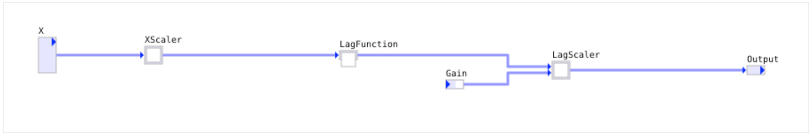


Figure 50: Lagfilter Schematic

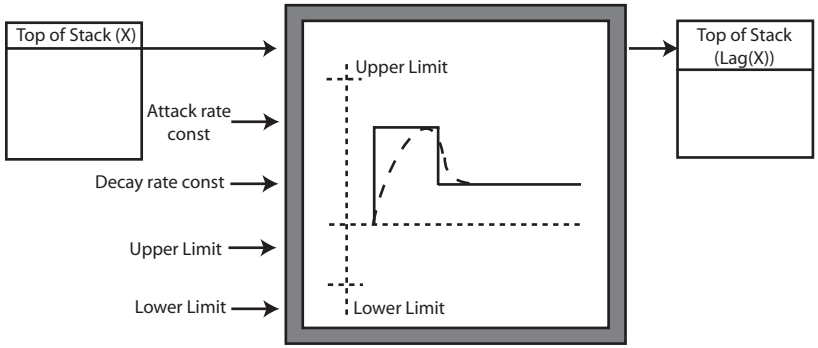


Figure 51: Lagfilter Function

Control Inputs			
Gain	**Variables**	**Type**	**Default Value**
	Scale	basic/float32	1
	Description: The value of the kin input is multiplied by the scale value to obtain a gain which is then fed to the output.		
X	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	0
	kin_int	basic/int32	0
	kin_uint	basic/uint32	0
	Type	basic/vartype	Float
	Description: The input value is read into the Lag Function at the control rate of the model, which is 1,000Hz. The input can be a float, boolean, integer or unsigned integer. The type is used to select which one of the four inputs will be used in the comparison.		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_unit	basic/uint8	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_unit16	basic/uint16	n/a
	kout_unit32	basic/uint32	n/a
	kout_unit64	basic/uint64	n/a
	kout_unit8	basic/uint8	n/a
	Description: Output of the Lag Filter component.		

Internal Parameters			
Lag Function	**Variables**	**Type**	**Default Value**
	Attack_rate	basic/float32	1
	Decay_rate	basic/float32	1
	Max_limit	basic/float32	1
	Min_limit	basic/float32	0
	Description: Settable lag function parameters: **Attack rate** - Determines how quickly the function goes from the old value to the new value when the new value is greater than the old value. The smaller the attack rate the slower the function (or the longer the lag). **Decay rate** - Determines how quickly the function goes from the old value to the new value when the new value is less than the old value. The smaller the attack rate the slower the function (or the longer the lag). **Max Limit** - Maximum limit value for filter accumulation to prevent saturation of integrators. **Min Limit** - Minimum limit value for filter accumulation to prevent saturation of integrators.		
XScaler	**Variables**	**Type**	**Default Value**
	kin->kin[1]	Basic/Fkoat32	1
	Description: The multiply function that takes the X input value and multiplies it by the value defined in kin[1].		

3.23. LogicTable4

Version: 0-0-1

Description: The LogicTable4 object provides a mechanism for combining up to four boolean controls into a single function. The four inputs are combined to form a 4 bit number which is used as an index into a 16 value array. This array contains floating point values, so that a combination of control functions can be achieved in a simple fashion.

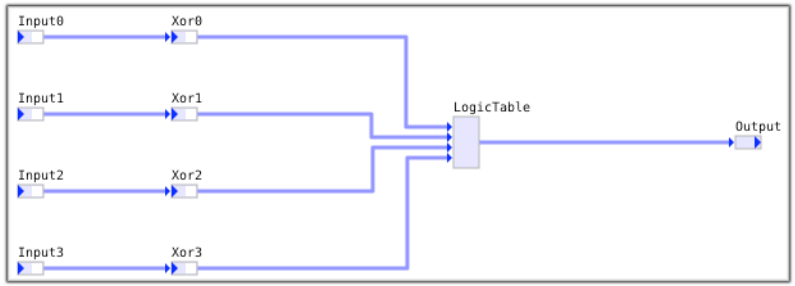


Figure 52: LogicTable4 Schematic

Control Inputs			
Input0 through Input3	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: The four inputs that make up the four bit number used to index the LogicTable array. Input0 is the least significant bit and Input3 is the most significant bit.		
Xor0 through Xor3	**Variables**	**Type**	**Default Value**
	Value	basic/boolean	FALSE
	Description: Used to invert the polarity of each of the inputs. Value set to TRUE inverts where FALSE does not.		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_unit	basic/uint8	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_unit16	basic/uint16	n/a
	kout_unit32	basic/uint32	n/a
	kout_unit64	basic/uint64	n/a
	kout_unit8	basic/uint8	n/a
	Description: Output of the LogicTable as determined by the 4 bit index word and the associated value set in the LogicTable.		

Internal Parameters			
Table[0] through Table[15]	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Each table value is set by the user and is a position in a 16 index array. The 4-bit word created by Input0 through Input3 is used to index and output the array's values. For example, setting Input0 and Input2 to TRUE will produce the binary number 0101, which is equal to 5 in decimal. The final output of the component will be the float value that is set under table[5].		

3.24. MathFunction2

Version: 0-0-1

Description: The MathFunction2 component performs basic one and two variable functions. The following math functions can be performed by this component:

- Exponent X^Y
- Antilog 10^X
- Divide $X \div Y$
- LogicalOR $X \text{ OR } Y, X \,||\, Y$
- Multiply $X \times Y$
- LogicalXOR $X \text{ XOR } Y, X \oplus Y$
- Subtract $X - Y$
- LogicalAND $X \text{ AND } Y, X \,\&\&\, Y$
- Add $X + Y$
- Log10 $\log_{10} X$

In addition to performing basic math functions between two variables, this component can also be used to type convert any variable. By fixing one of the inputs as 1 and setting the math function to multiply, the type of variable output can be selected without altering the value of the input variable.

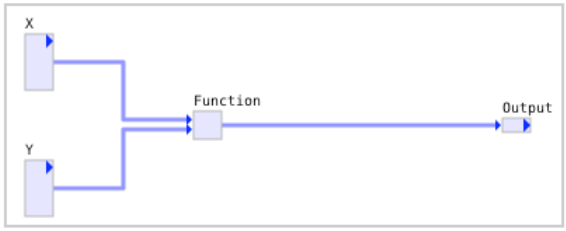


Figure 53: MathFuncion2 Schematic

Control Inputs			
X	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	Type	basic/vartype	Float
	Description: There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The type variable is used to select which one of the 4 input kins will be used in the comparison.**		
Y	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	Type	basic/vartype	Float
	Description: There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The type variable is used to select which one of the 4 input kins will be used in the comparison.**		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_unit	basic/uint8	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_unit16	basic/uint16	n/a
	kout_unit32	basic/uint32	n/a
	kout_unit64	basic/uint64	n/a
	kout_unit8	basic/uint8	n/a
	Description: The output is a result of the function F(X,Y), where the function is determined by the Type variable in the function primitive.		

Internal Parameters			
Output	**Variables**	**Type**	**Default Value**
	Gain	basic/float32	1.0
	Type	Control/ Math_function	none
	X_scale	basic/float32	1.0
	Y_scale	basic/float32	1.0
	Description: **Gain** - A multiplier used to scale the output of the function F(X,Y). **X_Scale** - A multiplier used to scale the X variable input before being used in the function. **Y_Scale** - A multiplier used to scale the Y variable input before being used in the function. **Type** - Used to select the type of function to be applied to X and Y. See description for valid function choices.		

3.25. MathFunction3

Version: 0-0-1

Description: The Mathfunction3 uses two embedded MathFunciton2s in order to simulate a three variable math function. The X and Y inputs are computed first and their result is computed with Z. The final three variable function can be written like:

F(X,Y,Z) = F (F(X,Y),Z)

The following math functions can be performed:

- Exponent X^Y
- Antilog 10^X
- Divide $X \div Y$
- LogicalOR $X \text{ OR } Y, X \| Y$
- Multiply $X \times Y$
- LogicalXOR $X \text{ XOR } Y, X \oplus Y$
- Subtract $X - Y$
- LogicalAND $X \text{ AND } Y, X \&\& Y$
- Add $X + Y$
- Log10 $\log_{10} X$

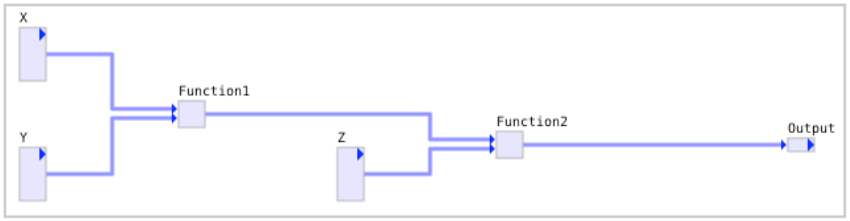


Figure 54: MathFunction3

Control Inputs			
X	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	Type	basic/vartype	float
	Description: There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The type variable is used to select which one of the 4 input kins will be used in the function.**		
Y	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	Type	basic/vartype	float
	Description: There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The type variable is used to select which one of the 4 input kins will be used in the function.**		

Control Inputs			
Z	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	Type	basic/vartype	Float
	Description: There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The type variable is used to select which one of the 4 input kin's will be used in the function.**		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_unit	basic/uint8	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_unit16	basic/uint16	n/a
	kout_unit32	basic/uint32	n/a
	kout_unit64	basic/uint64	n/a
	kout_unit8	basic/uint8	n/a
	Description: The output is a result of the function F(P(X,Y)Z) where the function is determined by the Type variable in the function2 primitive.		

Internal Parameters			
Function1	**Variables**	**Type**	**Default Value**
	Gain	basic/float32	1.0
	Type	Control/ Math_function	none
	X_scale	basic/float32	1.0
	Y_scale	basic/float32	1.0
	Description: **Gain** - A multiplier used to scale the output of the function F(X,Y). **X_Scale** - A multiplier used to scale the X variable input before being used in the function. **Y_Scale** - A multiplier used to scale the Y variable input before being used in the function. **Type** - Used to select the type of function to be applied to X and Y. See description for valid function choices.		
Function2	**Variables**	**Type**	**Default Value**
	Gain	basic/float32	1.0
	Type	Control/ Math_function	none
	X_scale	basic/float32	1.0
	Y_scale	basic/float32	1.0
	Description: **Gain** - A multiplier used to scale the output of the function F(X,Y). **X_Scale** - A multiplier used to scale the X variable input before being used in the function. **Y_Scale** - A multiplier used to scale the Y variable input before being used in the function. **Type** - Used to select the type of function to be applied to X and Y. See description for valid function choices.		

3.26. RangeCheck

Version: 0-0-1

Description: The Range Check component returns a 1 if the input is between specified values. The input can only be a float32 value.

Figure 55: Range Check Schematic

Control Inputs			
X	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: X is the input number whose range is checked by the component.		
Compare	**Variables**	**Type**	**Default Value**
	gain	basic/float32	1.0
	Description: The gain is multiplied with the output to produce a final output. If X is in range, the final output of the component will be equal to gain.		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_bool	basic/boolean	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_uint16	basic/uint16	n/a
	kout_uint32	basic/uint32	n/a
	kout_uint64	basic/uint64	n/a
	kout_uint8	basic/uint8	n/a
	Description: Output is equal to 1 if the input X value is within range (assuming a gain of 1.0). Multiple variable types are returned for use downstream in the model.		

Internal Parameters			
Compare	**Variables**	**Type**	**Default Value**
	Lower	basic/float32	0.0
	Lower_Inclusive	basic/boolean	FALSE
	Scale	basic/float32	1.0
	Upper	basic/float32	0.0
	Upper_Inclusive	basic/boolean	FALSE
	Description: **Lower** is the bottom number of the range that X is checked against. Setting **Lower_Inclusive** to TRUE will cause the component to return a 1 if the input is exactly the same as Lower. **Scale** multiplies with the input X value before it is compared against the given range. **Upper** is the top number of the range that X is checked against. Setting **Upper_Inclusive** to TRUE will cause the component to return a 1 if the input is exactly the same as Upper.		

3.27. StereoPan3D

Version: 0-0-1

Description: The 3D Stereo Panner component can drive the volume for eight different speaker positions based upon the position of an input sound. Since an engine on the left side of an airplane should be heard louder on the speakers at the left side of a simulator than on the right side, the panner object will automatically determine gains based on the position of the object creating sound and the position of the speakers in the cockpit. In addition to stationary sounds like an engine, the Stereo Panner can be used to control volumes to simulate moving objects like a rocket flying by the airplane.

StereoPan 3D uses an input position for the object creating sound, and also coordinates for the position of each output speaker. The default values assume that the pilot is at 0,0,0 and the speaker positions are relative to the pilot positions, i.e. Speaker1 is directly right of the pilot at position 1, 0, 0. The final output of the component is eight different volume levels, which correspond to the eight different speaker positions.

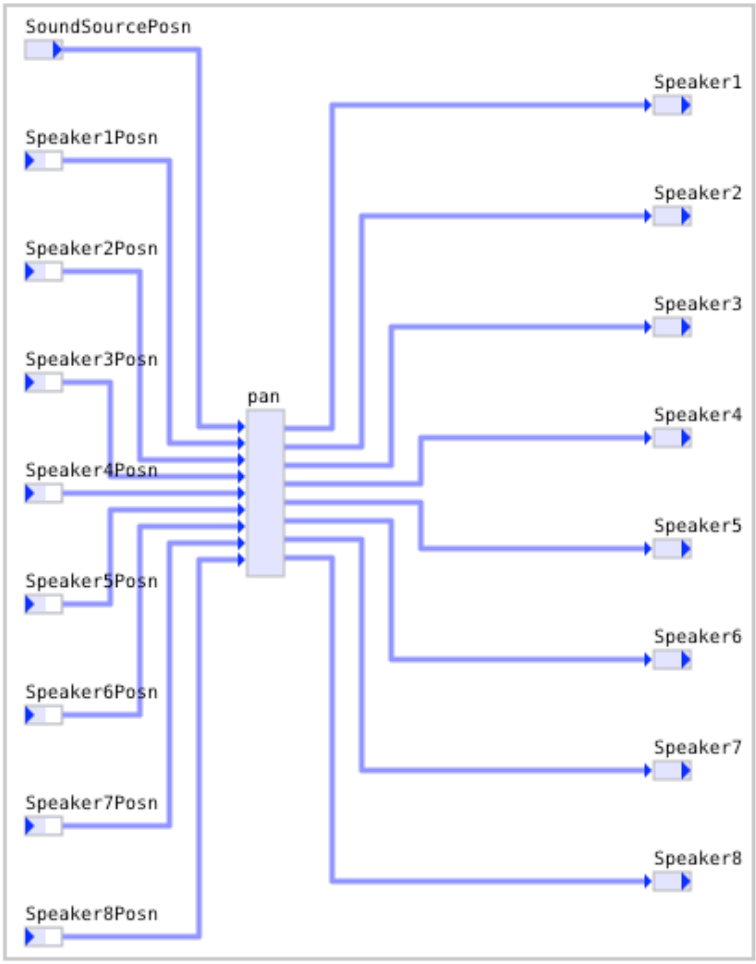


Figure 56: Stereo Pan 3D Schematic

Control Inputs			
Sound-Source-Posn	**Variables**	**Type**	**Default Value**
	kin		
	x	basic/float32	0.0
	y	basic/float32	0.0
	z	basic/float32	0.0
	kin_geocentric		
	x	basic/float64	0.0
	y	basic/float64	0.0
	z	basic/float64	0.0
	type	radio/coord_type	COORD_FLOAT
	Description: The position of the input sound source. Setting type to COORD_GEOCENT will use the kin_geocentric values, which are float64 instead of float32. The default is to use the x, y, and z values under kin.		
Speaker1 Posn through Speaker8 Posn	**Variables**	**Type**	**Default Value**
	kin		
	x	basic/float64	varies
	y	basic/float64	varies
	z	basic/float64	varies
	Description: The location of a particular speaker relative to the pilot located at 0,0,0. See the table to the right for the default values for each speaker position.		

Speaker	x, y, z
1	1, 0, 0
2	-1, 0, 0
3	0, 1, 0
4	0, -1, 0
5	0, 0, 1
6	0, 0, -1
7	-1, -1, 0
8	1, -1, 0

3.28. Switch

Version: 0-0-2

Description: The switch component provides the ability to switch between two values dependent upon a conditional threshold. There are three input values, X, Y and Z. Y and Z are the output values and X is the value upon which the condition is tested. If the condition is TRUE, then Y is outputted otherwise Z is outputted. The functionality of the component can be read like so:

If 'X> =,' or If 'X < = ' threshold then

Output Y

Else

Output Z

endif

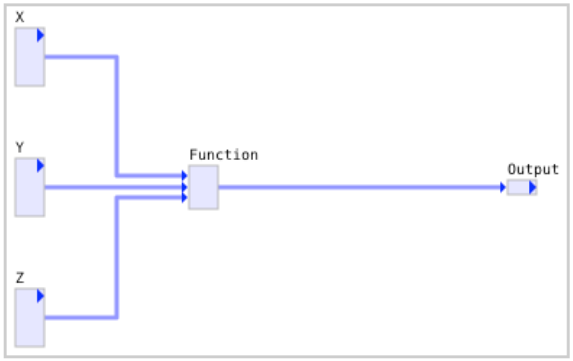


Figure 57: Switch Schematic

Control Inputs			
X	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	Type	basic/vartype	Float
	Description: The X input is used to test against the threshold value set in the function primitive. There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The type variable is used to select which one of the 4 input kins will be used in the conditional test.**		
Y	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	Type	basic/vartype	float
	Description: The Y variable is outputted when then conditional test performed in the function primitive is TRUE. There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The type variable is used to select which one of the 4 input kins will be used.**		

Control Inputs			
Z	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	Type	basic/vartype	float
	Description: The Z variable is outputted when the conditional test performed in the function primitive is TRUE. There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. **The type variable is used to select which one of the 4 input kin's will be used.**		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_unit	basic/uint8	n/a
	kout_float	basic/float32	n/a
	kout_int16	basic/int16	n/a
	kout_int32	basic/int32	n/a
	kout_int64	basic/int64	n/a
	kout_int8	basic/int8	n/a
	kout_unit16	basic/uint16	n/a
	kout_unit32	basic/uint32	n/a
	kout_unit64	basic/uint64	n/a
	kout_unit8	basic/uint8	n/a
	Description: The output is a result of the comparison between X and the threshold value. When the condition is TRUE, Y is outputted. When the condition is FALSE Z is outputted.		

Internal Parameters			
Function	**Variables**	**Type**	**Default Value**
	Gain	basic/float32	1.0
	Threshold	basic/float32	0.0
	Type	control/ compare	Greater-ThanEqualTo
	Val1_Scale	basic/float32	1.0
	Val2_Scale	basic/float32	1.0
	X_scale	basic/float32	1.0
	Description: **Gain** - A multiplier used to scale the final output of the function. **Threshold** - The value used to test against X. **Type** - Used to select the type of comparison to perform between X and the Threshold. Valid choices are Greater-ThanEqualTo and LessThanEqualTo **Val1_Scale** - A multiplier used to scale the Y variable input. **Val2_Scale** - A multiplier used to scale the Z variable input. **X_Scale** - A multiplier used to scale the X variable input.		

3.29. TableXY16

Version: 0-0-1

Description: The TableXY16 component provides a single function look-up table with up to 16 user definable values.

An Interpolation parameter allows for either a non-interpolated stepwise output, based upon the closest value to the X input, or a linear interpolation of data between adjacent table inputs.

Figure 58: TableXY16 Schematic

Control Inputs			
X	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	kin_bool	basic/boolean	FALSE
	kin_int	basic/int32	0
	kint_uint	basic/uint32	0
	type	basic/vartype	Float
	Description: The X input is used as the lookup value in the table primitive. There are 4 different types of input kin variables: float, boolean, integer and unsigned integer. The type variable is used to select which one of the 4 input kins will be used in the table lookup.		

Control Outputs			
Output	**Variables**	**Type**	**Default Value**
	kout_unit	basic/uint8	kin
	kout_float	basic/float32	kin
	kout_int16	basic/int16	kin
	kout_int32	basic/int32	kin
	kout_int64	basic/int64	kin
	kout_int8	basic/int8	kin
	kout_unit16	basic/uint16	kin
	kout_unit32	basic/uint32	kin
	kout_unit64	basic/uint64	kin
	kout_unit8	basic/uint8	kin
	Description: The output value is either the Y value that corresponds to the closest match between the X Input and the Table X value if interpolation is off, or is an interpolated value corresponding the X inputs position between two table X values.		

Internal Parameters			
Table	**Variables**	**Type**	**Default Value**
	Interpolate	basic/boolean	FALSE
	Interpolate Flag	basic/boolean	FALSE
	Scale	basic/float32	1.0
	Table->Data[16]->Data[0 through 15]->X	basic/float32	0.0
	Table->Data[16]->Data[0 through 15]->Y	basic/float32	0.0
	Description: **Interpolate** - Enables / disables interpolation between adjoining data positions in the table. Interpolate flag must be set to TRUE for interpolation to occur. **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier which scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up with the X Input value. **Table->data[16]->Data[0 through 15]->Y** - The value that is output from a table look-up.		

3.30. TableXY32

Version: 0-0-1

Description: The TableXY32 component is functionally the same as the TableXY16 object, except that 32 different points can be defined in the table, as opposed to the 16 available in TableXY16.

For more information see TableXY16.

Figure 59: Table XY 32 Schematic

3.31. Uint8ToFloat

Version: 0-0-1

Description: This component takes an unsigned byte input variable and does the type conversion to float. Offsets can be included from within the Object.

For more information also see BoolToFloat and InputUint8.

Figure 60: Uint8 To Float

Control Inputs			
kin	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	offset	basic/uint8	0
	Description: This is where the input byte variable should be linked for type conversion. The offset can be driven to add into the kin value before type conversion takes place.		

Control Outputs			
kout	**Variables**	**Type**	**Default Value**
	kout	basic/float32	0.0
	Description: The input value converted to a float after all offsets have been included.		

Internal Parameters			
kout	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	kin_int	basic/int32	0
	kin_uint16	basic/uint16	0
	kin_uint32	basic/uint32	0
	kin_uint64	basic/uint64	0
	kin_uint8	basic/uint8	0
	Description: The kin parameters will add an offset to the kout value. Note that all of the vartypes are summed together to produce a single offset. For example, setting kin_int to 5 and kin_uint16 to 3 will cause kout to have an offset of 8.0.		

4.0. DRED Components

The DRED components are special reserved objects and are not intended for general use. They are used by RMS to create the readiness test and must be in a separate folder in order to accommodate the auto generation process. The available DRED components can be found elsewhere in the component tree, these include AmpMod, BitToByte, Gain, InputInt32, MathFunction3, Mixer4, and Wave.

5.0. Engine Components

Real engines produce different sounds based on how fast they are running; therefore, it is nearly impossible to produce a recording of an engine for use in a simulator. The audio must be synthesized in such a way that it can be controlled by the RPM of the device. The Engine components are able to model the variable sound and are used to simulate jet engines, helicopter propellers, APUs, and other speed driven devices.

The Engine, Engine2, and EngineLevelD components are based on the idea that the sound of an engine consists of a number of fundamental whines. Each whine increases in frequency as the speed of the engine increases, but the amplitude of the whine will vary with speed. With the addition of a few noise sources to simulate sounds like the engine lighting, a realistic portrayal of an engine can be achieved.

The Rotor and SimpleRotor components are used to simulate propellers and are based on the physics of the actual propellers by using inputs such as blade radius, blade angle, or mach limit.

5.1. Engine

Version: 0-0-3

Description: The Engine component is outdated and not recommended for use. The Engine component remains in the component tree for backwards compatibility, use the Engine2 component in its place.

5.2. Engine2

Version: 0-0-1

Description: The Engine2 component provides a composite source suitable for simulating a single jet engine. It uses three triangle waves and two band-limited white noise sources.

Control is provided over triangle wave frequencies and amplitudes, noise bandwidth and gain, and overall gain for the composite engine sound.

The noise and whine gain inputs are typically driven by the engine's RPM. This is usually a value passed from the host computer. The noise and whine tables are user-defined interpretive values to control how the noise and whine's frequency and volume change based on engine RPM. Typically, as engine RPM increases so do the associated whine and noise frequencies and volumes. Tables allow for specific modeling of engine characteristics.

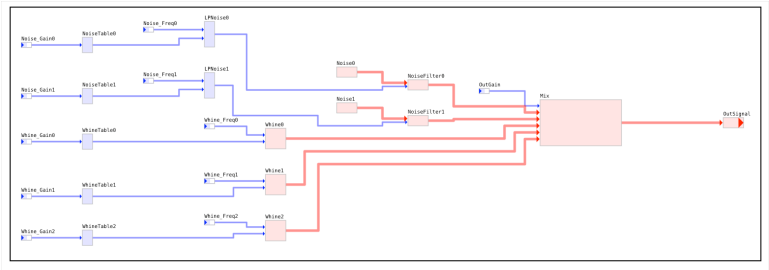


Figure 61: Engine 2 Schematic Overview

Control Inputs			
Noise_ Freq0	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Noise_Freq0 input is the roll-off frequency (in Hertz) of the bandwidth limited noise0 source used to produce the engine hiss/roar. The scale_factor is a multiplier to the input value. The outputted frequency is kin multiplied by scale_factor.		
Noise_ Freq1	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Noise_Freq1 input is the roll-off frequency (in Hertz) of the bandwidth limited noise1 source used to produce the engine hiss/roar. The scale_factor is a multiplier to the input value. The outputted frequency is kin multiplied by scale_factor.		
Noise Gain0	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Noise_Gain0 input is used as the input into the NoiseTable0 primitive. The NoiseTable0 is a tableXY16 look up table. Inside the table are the actual gain values to be applied to the final Noise0 signal. The scale_factor is a multiplier to the input value. The final output driven into the table is kin multiplied by scale_factor.		

Control Inputs			
Noise Gain1	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Noise_Gain1 input is used as the input into the NoiseTable1 primitive. The NoiseTable1 is a tableXY16 look up table. Inside the table are the actual gain values to be applied to the final Noise1 signal. The scale_factor is a multiplier to the input value. The final output driven into the table is kin multiplied by scale_factor.		
OutGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The OutGain Input is a gain multiplier applied to the Engine signal amplitude after all signals have been mixed together. The actual gain applied to the Engine signal amplitude is kin multiplied by the scale_factor.		
Whine_ Freq0	**Variables**	**Type**	**Default Value**
	kine	basic/float32	0
	scale_factor	Basic/Float 32	1.0
	Description: The Whine_Freq0 is the frequency (in Hertz) of the Whine0 triangle wave used to produce the engine whines. The scale_factor is a multiplier to the input value. The outputted frequency is kin multiplied by scale_factor.		

Control Inputs			
Whine_ Freq1	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Whine_Freq1 is the frequency (in Hertz) of the Whine1 triangle wave used to produce the engine whines. The scale_factor is a multiplier to the input value. The outputted frequency is kin multiplied by scale_factor.		
Whine_ Freq2	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Whine_Freq2 is the frequency (in Hertz) of the Whine2 triangle wave used to produce the engine whines. The scale_factor is a multiplier to the input value. The outputted frequency is kin multiplied by scale_factor.		
Whine_ Gain0	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Whine_Gain0 input is used as the input into the WhineTable0 primitive. The WhineTable0 is a tableXY16 look up table. Inside the table are the actual gain values to be applied to the final Whine0 signal. The scale_factor is a multiplier to the input value. The final output driven into the table is kin multiplied by scale_factor.		

Control Inputs			
Whine_ Gain1	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Whine_Gain1 input is used as the input into the WhineTable1 primitive. The WhineTable1 is a tableXY16 look up table. Inside the table are the actual gain values to be applied to the final Whine1 signal. The scale_factor is a multiplier to the input value. The final output driven into the table is kin multiplied by scale_factor.		
Whine_ Gain2	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Whine_Gain2 input is used as the input into the WhineTable2 primitive. The WhineTable2 is a tableXY16 look up table. Inside the table are the actual gain values to be applied to the final Whine2 signal. The scale_factor is a multiplier to the input value. The final output driven into the table is kin multiplied by scale_factor.		

Audio Outputs			
OutSignal	**Variables**	**Type**	**Default Value**
	Aout	audio/audio	n/a
	Description: The final Engine Sound signal after all inputs and gains have been applied.		

Internal Parameters			
Noise Table0 and Noise Table1	**Variables**	**Type**	**Default Value**
	Interpolate	basic/boolean	TRUE
	Interpolate	basic/boolean	FALSE
	Scale	basic/float32	1.0
	Table->data[16] ->data[0 through 16] ->X	basic/float32	0.0
	Table->data[16]->data[0 through 16] ->Y	basic/float32	0.0
	Description: **Interpolate** - Enables / Disables interpolation between adjoining data positions in the table. Interpolate flag must be set to TRUE for interpolation to occur. **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up with the Noise_Gain input values. **Table->data[16]->Data[0 through 15]->Y** - The value to be outputted from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

Internal Parameters			
Whine Tabl0 through Whine Table2	**Variables**	**Type**	**Default Value**
	Gaindecay	basic/float32	0.010
	Wavetype	audio/waveshape	triangle
	Width	basic/float32	0.5
	Width_units	control/width/units	width_is_ratio
	Description: Signal wave generators used to create the whine noise associated with the engine. **Gaindecay**- The exponential decay factor used to prevent clipping noises due to sudden changes in the Whine_Gain value. **WaveType**- Used to select the generated wave type. Valid choices are: pulse, triangle, sawtooth, pulse_inverted, sine and square waves. **Width** - Sets the width of wave form. Applicable for triangles, sawtooths, and pulses. **Width_units**- Determines the units for the width variable. Units can either be a ratio or in seconds.		

5.3. EngineLevelD

Version: 0-0-1

Description: The Level D Engine component provides a composite source suitable for simulating a single, multi-stage jet engine. It uses six triangle waves, two pulse waves, and two band-limited white noise sources.

Control is provided over triangle wave frequencies and amplitudes, noise bandwidth and gain, and overall gain for the composite engine sound.

The noise and whine gain inputs are typically driven by the engine's RPM. This is usually a value passed from the host computer. The noise and whine tables are user-defined interpretive values to control how the noise and whine's frequency and volume change based on engine RPM. Typically, as engine RPM increases so do the associated whine and noise frequencies and volumes. Tables allow for specific modeling of engine characteristics.

The Engine Level D is specifically designed for use on a simulator trying to meet the FAA's Level D Standard. This component provides more controls and input signals to aide in matching the spectral response of the real engine.

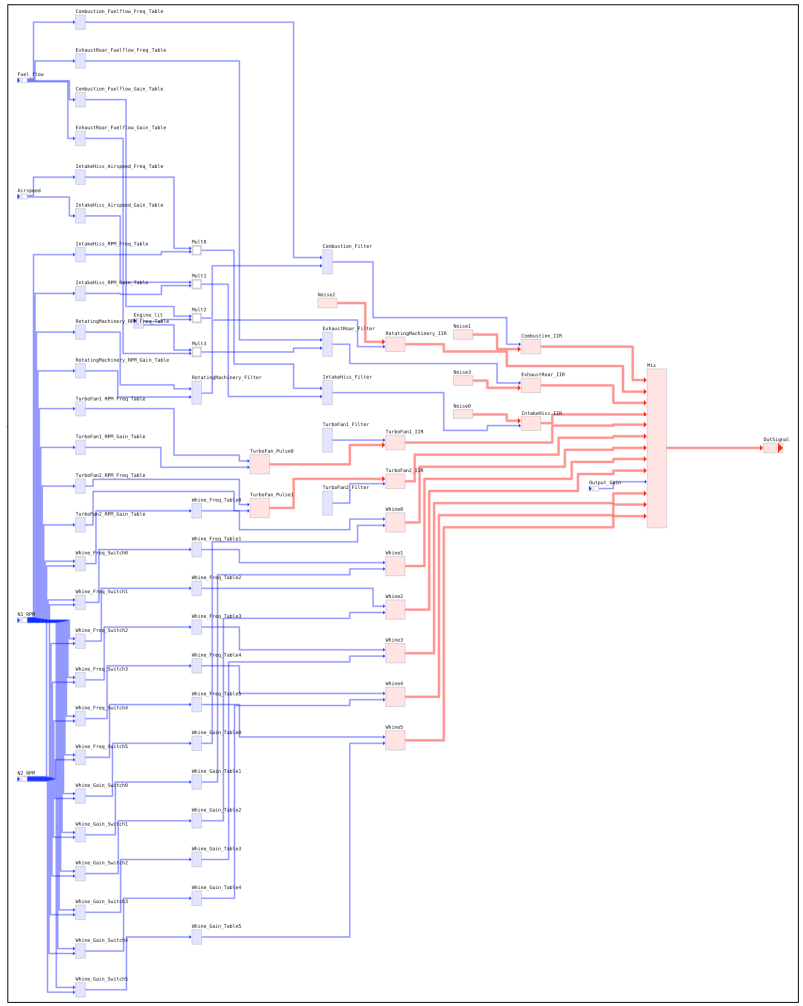


Figure 62: Engine Level D Schematic

Control Inputs			
Airspeed	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Airspeed measured in knots or miles per hour (mi/hr.).		
FuelFlow	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Fuel flow rate measured in pounds per hour (lbs./hr.).		
N1_RPM	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: Low-pressure fan rotor RPM percentage.		
N2_RPM	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: High-pressure compressor rotor RPM percentage.		

Control Inputs			
Output_ Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: The Output_Gain input is a gain multiplier applied to the amplitude after all signals have been mixed together. The actual gain applied to the Engine signal amplitude is kin multiplied by the scale_factor.		

Audio Outputs			
OutSignal	**Variables**	**Type**	**Default Value**
	Aout	audio/audio	n/a
	Description: The final Engine Sound signal after all inputs and gains have been applied.		

Internal Parameters			
Combustion_Filter	**Variables**	**Type**	**Default Value**
	Coeff	audio/coeff2	n/a
	Filter	audio/filter_type2	LowPass
	Filterset	audio/filterset	n/a
	Freq	basic/float32	0.0
	Gain	basic/float32	0.0
	QFactor	basic/float32	1.0
	Description: Combustion noise filter. Frequency is determined by **Combustion_Fuelflow_Freq_Table** and gain is determined by **Engine_Lit** multiplied by **Combustion_Fuelflow_Gain_Table**.		
Combustion_Fuel Flow_ FreqTable	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15->X** - The X value used in the table look-up (**FuelFlow**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

Internal Parameters			
Combustion_Fuel Flow_ Gain_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**FuelFlow**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		
Combustion_IIR	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Applies the filter coefficients determined by Combustion_Filter to a noise signal.		

Internal Parameters			
Exhaust Roar_ Filter	**Variables**	**Type**	**Default Value**
	Coeff	audio/coeff2	n/a
	Filter	audio/filter_type2	LowPass
	Filterset	audio/filterset	n/a
	Freq	basic/float32	0.0
	Gain	basic/float32	0.0
	QFactor	basic/float32	1.0
	Description: Exhaust roar noise filter. Frequency is determined by **ExhaustRoar_Fuelflow_Freq_Table** and gain is determined by **Engine_Lit** multiplied by **ExhaustRoar_Fuelflow_Gain_Table**.		
Exhaust Roar_Fuel Flow_ FreqTable	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data{0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**FuelFlow**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

Internal Parameters			
Exhaust Roar_Fuel Flow_ Gain_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**FuelFlow**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		
Exhaust Roar_IIR	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Applies the filter coefficients determined by **ExhaustRoar_Filter** to a noise signal.		

Internal Parameters			
Intake Hiss_ Airspeed_ Freq_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]p->Data[0 through 15]->X** - The X value used in the table look-up (**Airspeed**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

Internal Parameters			
Intake Hiss_ Airspeed_ Gain_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16] ->data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**Airspeed**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		
Intake Hiss_IIR	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Applies the filter coefficients determined by **IntakeHiss_Filter** to a noise signal.		

Internal Parameters			
Intake Hiss_ RPM_ Freq_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16] ->data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**N1 RPM**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

Internal Parameters			
Intake Hiss_ RPM_ Gain_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**N1 RPM**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		
Intake-HissFilter	**Variables**	**Type**	**Default Value**
	Coeff	audio/coeff2	n/a
	Filter	audio/filter_type2	LowPass
	Filterset	audio/filterset	n/a
	Freq	basic/float32	0.0
	Gain	basic/float32	1.0
	QFactor	basic/float32	1.0
	Description: Intake hiss noise filter. Frequency is determined by **IntakeHiss_Airspeed_Freq_Table** multiplied by **IntakeHiss_RPM_Freq_Table**. Gain is determined by **IntakeHiss_Airspeed_Gain_Table** multiplied by **IntakeHiss_RPM_Gain_Table**.		

Internal Parameters			
Mix	**Variables**	**Type**	**Default Value**
	Gain [0 through 16]	basic/float32	1.0
	LocalGain	basic/float32	1.0
	OutGain	basic/float32	0.0
	Description: Mixer that combines all signals generated by the LevelD_Engine object into a single audio stream for output.		
Noise (0-3)	**Variables**	**Type**	**Default Value**
	Enable	basic/boolean	TRUE
	Gain	basic/float32	1.0
	Description: Generic full spectrum noise generators.		
Rotating Machinery_IIR	**Variables**	**Type**	**Default Value**
	Bypass	basic/boolean	FALSE
	Coeff	audio/coeff2	n/a
	Gain	basic/float32	1.0
	Description: Applies the filter coefficients determined by **Rotating_Machinery_Filter** to a noise signal.		
TurboFan_Pulse0	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Pulse waveform generators. Frequency is determined by the corresponding **TurboFan1_Freq_Table**. Gain is determined by the corresponding **TurboFan1_Gain_Table**.		

Internal Parameters			
TurboFan_Pulse1	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Pulse waveform generators. Frequency is determined by the corresponding TurboFan2_Freq_Table. Gain is determined by the corresponding TurboFan2_Gain_Table.		
TurboFan 1_Filter	**Variables**	**Type**	**Default Value**
	Coeff	audio/coeff2	n/a
	Filter	audio/filter_type2	LowPass
	Filterset	audio/filterset	n/a
	Freq	basic/float32	0.0
	Gain	basic/float32	0.0
	QFactor	basic/float32	1.0
	Description: Turbo fan 1 noise filter.		
TurboFan 1_IIR	**Variables**	**Type**	**Default Value**
	Bypass	basic/boolean	FALSE
	Coeff	audio/coeff2	n/a
	Gain	basic/float32	1.0
	Description: Applies the filter coefficients determined by **TurboFan1_Filter** to a noise signal.		

Internal Parameters			
TurboFan 1_RPM_ Freq_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**N1 RPM**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

Internal Parameters			
TurboFan 1_RPM_ Gain_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**N1 RPM**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		
TurboFan 2_Filter	**Variables**	**Type**	**Default Value**
	Coeff	audio/coeff2	n/a
	Filter	audio/filter_type2	LowPass
	Filterset	audio/filterset	n/a
	Freq	basic/float32	0.0
	Gain	basic/float32	0.0
	QFactor	basic/float32	1.0
	Description: Turbo fan 2 noise filter.		

Internal Parameters			
TurboFan 2_IIR	**Variables**	**Type**	**Default Value**
	Bypass	basic/boolean	FALSE
	Coeff	audio/coeff2	n/a
	Gain	basic/float32	1.0
	Description: Applies the filter coefficients determined by **TurboFan2_Filter** to a noise signal.		
TurboFan 2_RPM_ Freq_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16] ->data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**N1 RPM**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

Internal Parameters			
TurboFan 2_RPM_ Gain_ Table	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**N1 RPM**). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		
Whine (0 through 5)	**Variables**	**Type**	**Default Value**
	n/a	n/a	n/a
	Description: Triangle waveform generators. Frequency is determined by the corresponding **Whine_Freq_Table**. Gain is determined by the corresponding **Whine_Gain_Table**.		

Internal Parameters			
Whine_ Freq_ Switch (0 through 5)	**Variables**	**Type**	**Default Value**
	Gain	basic/float32	0.0
	Threshold	basic/float32	0.5
	Type	Control/Compare	LessThanEqualTo
	Val1_Scale	basic/float32	1.0
	Val2_Scale	basic/float32	1.0
	Value1	basic/float32	0.0
	Value2	Basic/Floatt32	0.0
	X	basic/float32	0.0
	X_Scale	basic/float32	1.0
	Description: Switch for selecting which RPM (**N1** or **N2**) to pass through for whine frequency calculations. Setting **X**=0 will cause the switch to pass **N1 RPM** through. Setting **X**=1 will cause the switch to pass **N2 RPM** through. **Note**: Gain must be set to 1.0.		

Internal Parameters			
Whine_ Freq_ Table (0 through 5)	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16]-> data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**N1 RPM** or **N2 RPM,** determined by the corresponding Switch). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

Internal Parameters			
Whine_ Gain_ Switch (0 through 5)	**Variables**	**Type**	**Default Value**
	Gain	basic/float32	0.0
	Threshold	basic/float32	0.5
	Type	Control/Compare	LessThanEqualTo
	Val1_Scale	basic/float32	1.0
	Val2_Scale	basic/float32	1.0
	Value1	basic/float32	0.0
	Value2	Basic/Floatt32	0.0
	X	basic/float32	0.0
	X_Scale	basic/float32	1.0
	Description: Switch for selecting which RPM (**N1** or **N2**) to pass through for whine frequency calculations. Setting **X**=0 will cause the switch to pass **N1 RPM** through. Setting **X**=1 will cause the switch to pass **N2 RPM** through. **Note**: Gain must be set to 1.0.		

Internal Parameters			
Whine_ Gain_ Table (0 through 5)	**Variables**	**Type**	**Default Value**
	Interpolate_Flag	basic/boolean	FALSE
	kin	basic/float32	0.0
	Scale	basic/float32	1.0
	Table->[16] ->data[0 through 15]->X	basic/float32	0.0
	Description: **Interpolate flag** - Turns on the interpolation feature if set to TRUE. **Scale** - A gain multiplier that scales the output of the selected table Y value. **Table->data[16]->Data[0 through 15]->X** - The X value used in the table look-up (**N1 RPM** or **N2 RPM,** determined by the corresponding Switch). **Table->data[16]->Data[0 through 15]->Y** - The value to be output from a table look-up. **Note**: The Y value of the first data position (Data[0]->Y) will always be used as the output so long as the input X value is less than the set X value in Data[0]->X.		

5.4. Rotor

Version: 0-0-3

Description: The Rotor component is an outdated component and not recommended for use. The Rotor component remains in the component tree for backwards compatibility, use the SimpleRotor in its place.

5.5. SimpleRotor

Version: 0-0-3

Description: The Simple Rotor component provides a composite sound for a rotating helicopter blade. It includes the three principal sources of noise; air noise from the movement of air over the blades, Force noise from the impact of the blade with the air medium, and thickness noise due to the dual edge sound sources on a blade.

The overall sound can be tuned based upon blade parameters such as radius and blade count, with overall gain control based on both RPM and blade angle.

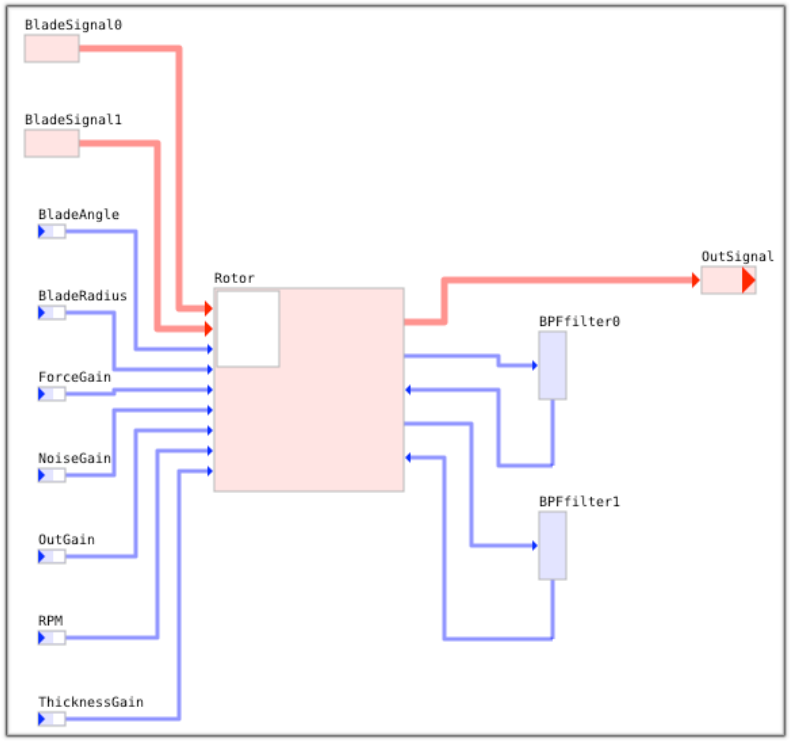


Figure 63: SimpleRotor Schematic

Control Inputs			
Blade Angle	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: Input for setting the angle of the blade. scale_factor is a gain multiplier applied to the input angle value. Final output is the kin value multiplied by the scale_factor value.		
Blade Radius	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: Input for setting the radius of the blade. scale_factor is a gain multiplier applied to the inputted radius value. Final output is the kin value multiplied by the scale_factor value.		
Force Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The gain for the principle component of the blade sound due to the air force on the blade. scale_factor is an additional gain multiplier applied to the ForceGain value. Final output is the kin value multiplied by the scale_factor value.		

Control Inputs			
NoiseGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The gain for the air noise component of the blade sound. scale_factor is an additional gain multiplier applied to the NoiseGain value. Final output is the kin value multiplied by the scale_factor value.		
OutGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Output gain applied to the final rotor signal. The final output signal is the kin value multiplied by the rotor signal produced in the rotor primitive.		
RPM	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: Input for the frequency (in Revolutions per Minute) of the blade shaft. scale_factor is an additional gain multiplier applied to the RPM value. Final output is the kin value multiplied by the scale_factor value.		
Thickness Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The gain for the thickness noise component of the blade sound. The ThicknessGain is the impulsive sound associated with transonic tip speeds characteristic of helicopter rotor blades, and increases significantly in volume as the tip mach speed approaches 1.0.		

Audio Outputs			
OutSignal	**Variables**	**Type**	**Default Value**
	Aout	audio/audio	n/a
	Description: The final Rotor Sound signal after all inputs and gains are applied.		

Internal Parameters			
Blade Signal0 and Blade Signal1	**Variables**	**Type**	**Default Value**
	Gain	basic/float32	1.25
	Enable	basic/boolean	TRUE
	Description: Gain is a multiplier applied to the amplitude of the internally generated BladeSignal. Enable turns the generated BladeSignal on and off. This is used if only one of the BladeSignals is desired. Disabling one of the blade signals will significantly degrade the quality of the rotor sound, this is not recommended.		

Internal Parameters			
Rotor	**Variables**	**Type**	**Default Value**
	Bladeno	basic/uint8	2
	Machdecay	basic/float32	0.01
	Machlimit	basic/float32	0.995
	RPMdecay	basic/float32	0.0
	RPMScale	basic/float32	1.0
	Description: **Bladeno**- Sets the number of blades on the shaft. **Machdecay**- The exponential decay factor used to prevent clipping noises due to sudden changes in mach speed. **Machlimit**- A limit for the calculated tip mach speed given by the equation **TipMach**- (2*pi * EffectiveBladeRadius) / (60 * SpeedOfSound). This limit keeps the maximum mach speed to a predetermined maximum. Normal values are usually between 0.95 and 0.99, depending how dominant the thickness noise is required. The sound model is not accurate above 0.99 since supersonic effects start to dominate the sound spectrum. **RPMDecay**- The exponential decay factor used to prevent clipping noises due to sudden changes RPM. **RPMScale**- An additional gain multiplier to the RPM input value.		

6.0. Intercom Components

The Intercom components provide an audio bus structure to which other components can connect, giving the ability to distribute audio throughout a model. Additionally, the Intercom service and components can simulate the intercom bus structures of real aircraft and other training applications.

This section of components works in conjunction with the Intercom service and the Channel Handle Editor, found under the Tools menu. The Intercom service can be thought of as the Telestra software that routes the audio around the mode, and the components and Channel Handle Editor are interfaces to this service.

In order to use the Intercom service, a channel handle must be created from the Channel Handle Editor. The following components link to these handles via their "AssetDefinition" primitives, using the channel parameter under kin > asset.

For more information about using channel handles and setting up in the Intercom service, see the MBV Basic Tutorials available at the ASTi website:

http://www.asti-usa.com/support/mbv/index.html

This section provides details on the following intercom components:

- Balancer
- ComMH60 and ComMH60_INS
- ComQuad
- ComSing
- IcomAudioSrc
- Icom RX
- Icom Tx
- Shared Radio

6.1. Balancer

Version: 0-0-1

Description: The Balancer component is used to put a single audio signal onto multiple intercom channels with the ability to adjust the individual gains of each of the signals before being put onto their intercom channel. Each balancer can route a single audio signal and up to 8 different intercom buses.

The balancer is commonly used for environmental cueing applications where a sound is to be heard on multiple speakers but with different volumes for each speaker.

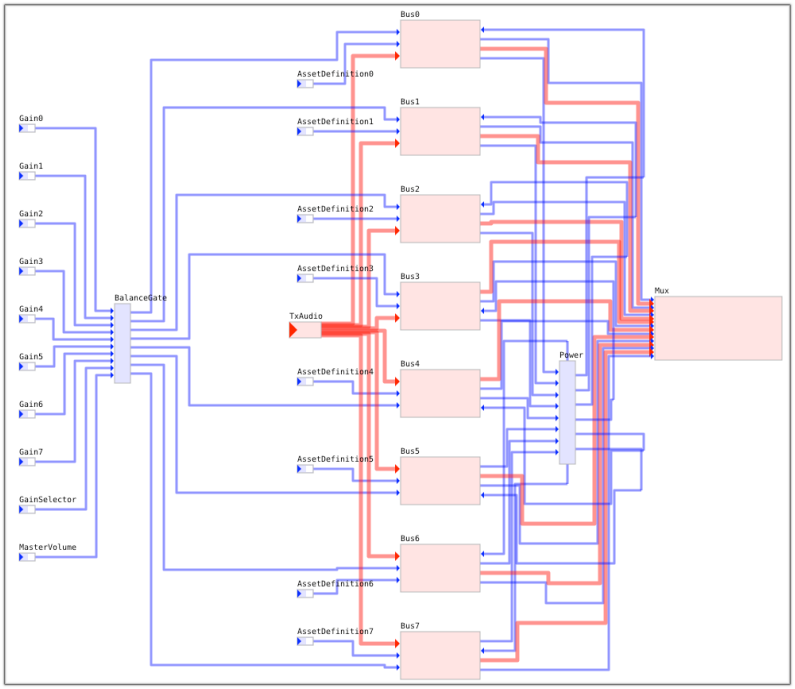


Figure 64: Balancer Schematic

Audio Inputs			
TxAudio	**Variables**	**Type**	**Default Value**
	Ain	audio	n/a
	Description: Provides a connection to an audio signal that is to be distributed onto the specified intercom busses.		

Control Inputs			
Asset Definition 0 through Asset Definition 7	**Variables**	**Type**	**Default Value**
	kin->asset->channel	intercom/IC_channel	0
	Description: Used to specify a set of 8 intercom busses to route to the input audio signal.		
Gain0 through Gain7	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	scale_factor	basic/float32	1.0
	Description: The Gain inputs are multipliers to the assigned Asset-Definition audio bus signals. Each audio buses signal amplitude is multiplied by its corresponding gain value prior to being transmitted onto the audio bus. The actual gain applied is kin multiplied by the scale_factor.		
Gain Selector	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	255
	Description: The GainSelector is a bit mask used to select which of the gain inputs are to be applied.		

Control Inputs			
Main Volume	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	scale_factor	basic/float32	1.0
	Description: The MasterVolume input is a gain multiplier applied uniformly to the amplitude of all audio signals. The actual gain applied is kin multiplied by the scale_factor.		

Internal Parameter			
Balancer	**Variables**	**Type**	**Default Value**
	control_mask	basic/uint8	255
	Description: The bit value in the control mask is AND'd with the bit value of the GainSelector input in order to select which gains are applied. The control_mask is used to prevent the GainSelector from choosing select gains.		

6.2. ComMH60 and ComMH60_INS

Version: 0-0-1

Description: The MH60 and MH60 Comm Panels are specially designed to emulate the real panels found on an MH-60 helicopter. These objects are not intended for general use because of the highly specific logic built into them. Special PTT, priority, and sidetone controls have been implemented within both of the compoents. For more information please contact ASTi.

6.3. ComQuad

Version:0-0-2

Description: ComQuad provides simulation of a radio/communications control panel. It manages the transmit and receive selections for up to 32 separate communication assets (e.g. a radio or an intercom). Signal outputs are mixed into a single output stream. A single signal input is transmitted over one or more of the assets.

The ComQuad component is equivalent to have four ComSing objects which share the same PTT, TxAudio, PowerSwitch, and MasterVolume.

Whereas the Comsing provides eight buses: Bus0 - Bus7 = 1 through 8

The ComQuad provides 32 buses:

BusA0 - BusA7 = 1 through 8

BusB0 - BusB7 = 9 through 16

BusC0 - BusC7 = 17 through24

BusD0 - BusD7 = 25 through 32

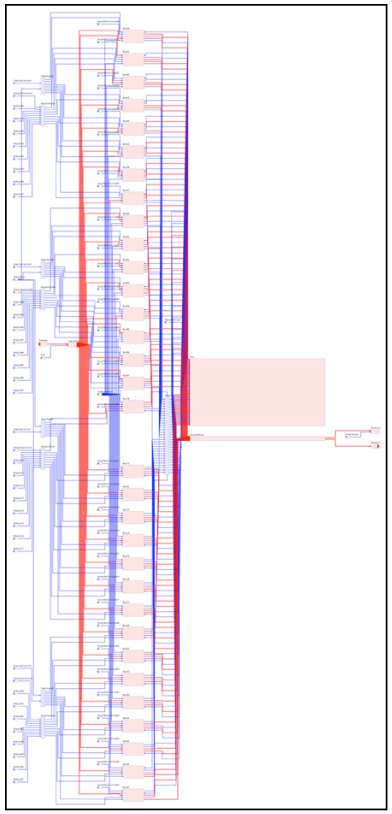


Figure 65: ComQuad Schematic Overview

Audio Inputs			
TxAudio	**Variables**	**Type**	**Default Value**
	Ain	Audio	n/a
	Description: A connection to an external transmit audio source (like a microphone interface from an Iris asset) that is routed to selected Buses for transmission on associated Channels. Routing TxAudio to selected Bus(es) requires all of the following conditions: 1) InputSelector = TRUE bit for selected Bus 2) InGainGate = TRUE bit for selected Bus 3) PTT = TRUE 4) InputGain > 0.0		

Control Inputs			
Asset Definition A0 through Asset Definition D7	**Variables**	**Type**	**Default Value**
	kin->asset->channel	intercom/IC_channel	0
	Description: AssetDefinition links a Commsing Bus to a Channel. The link provides two-way communications between the Channel and the Bus: - Receive audio is routed from the Channel to the ComSing Bus - Transmit audio is routed from the ComSing Bus to the Channel. The linking mechanism is a common Channel, which is connected to the AssetDefinition for a particular ComSing Bus and to the AssetDefinition of another signal component (ComSing, Radio, Intercom, etc.). Use example - connecting a Radio to a ComSing Bus: 1) Create a Channel Handle: Tools / Channel Handle: Channel Handle Editor, select New Handle and enter Handle and Channel values. 2) Connect the Channel Handle to a Radio / Asset Definition: Kin>Asset>Channel [Channel value] 3) Connect the Channel Handle to a Commsing / AssetDefinition: Kin>Asset>Channel [Channel value]		
InputGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the input audio signal (TxAudio). Application hint: set the input audio gain using a two step process involving the CommSing and Iris Asset components. First, set the course gain using a combination of the Iris Asset component PreAmp and ATMEL input gains. Second, set the fine gain using the CommSing Input Gain. Following this process will insure optimization of the signal-to-noise ratio (and hence, the audio quality) of input audio signal.		

Control Inputs			
Input SelectorA through Input SelectorD	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	1.0
	Description: Connects the input audio signal (TxAudio) to selected audio Buses. Each selected Bus routes the input audio to a transmitting Channel. The InputSelector is a control bitmask capable of routing the TxAudio signal to (up to) 32 Buses.		
Master Volume	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: A gain multiplier applied uniformly to the amplitude of all audio signals in the communication panel.		
Output SelectorA through Output SelectorD	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	10
	Description: Connects selected audio Buses to the output audio signal (RxAudio). Bus audio is received from other components via Channel assignments. Multiple Bus outputs are mixed down to a single stream prior to routing to RxAudio. The OutputSelector is a control bitmask capable of routing output audio from (up to) 32 Buses to RxAudio.		
Power_ Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: Input control to turn the ComSing's power on and off.		
PTT (Press-to-Talk)	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Input control connection to an external push-to-transmit control signal.		

Control Inputs			
RxGainA0 through RxGainD7	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Receive audio gain controls for the corresponding Busses. The gain is applied prior to the audio signal being mixed and sent to the RxAudio output. The letter of the RxGain corresponds to the AssetDefintion letter.		
Sidetone Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Provides gain control for sidetone audio.		
Sidetone-SelectorA through Sidetone-SelectorD	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	255
	Description: Connects input audio (TxAudio) to the output audio bus (RxAudio). The SidetoneSelector is a control bitmask capable of routing the signal being placed on the buses back out to the RxAudio stream, effectively allowing a user to hear what they are saying as they speak, i.e. sidetone. The sidetone will only be produced for the properly masked buses. See InputSelector for an example of the bit mask.		

Audio Outputs			
RxAudio	**Variables**	**Type**	**Default Value**
	Aout	audio	n/a
	Description: RxAudio is an audio output, whose source is the mixture of output audio from selected Buses (Bus0 - Bus7). Routing audio from selected Bus(es) to RxAudio requires all of the following conditions: 1) OutputSelector = TRUE bit for each selected Bus 2) RxGain > 0.0 for each selected Bus 3) Master Volume > 0.0		

Internal Parameters			
Input Gain GateA through InGain GateD	**Variables**	**Type**	**Default Value**
	control_mask	basic/uint8	255
	Description: The Bit value in the control mask is AND'd with the Bit value of the InputSelector input in order to select which AssetDefintions are to be selected. The control_mask is used to prevent the InputSelector from choosing select AssetDefintions.		
InputGate	**Variables**	**Type**	**Default Value**
	Ptt_invert	basic/boolean	FALSE
	Description: If ptt_invert is set to TRUE, the PTT input value is inverted.		

Internal Parameters			
Output Gain GateA through InGain GateD	**Variables**	**Type**	**Default Value**
	control_mask	basic/uint8	255
	Description: The Bit value in the control mask is AND'd with the Bit value of the OutputSelector input in order to select which AssetDefintions are to be selected. The control_mask is used to prevent the OutputSelector from choosing select AssetDefintions.		
Sidetone-GateA through Sidetone-GateD	**Variables**	**Type**	**Default Value**
	control_mask	basic/uint8	255
	Description: SidetoneGate is a control bitmask that is used in combination with the SidetoneSelector control to enable routing of sidetone audio from selected CommSing Buses. The bits in the SidetoneSelector value are logically AND'ed with corresponding bits in the SidetoneGate value. The result of this logic operation is used to enable the sidetone associated with each bus, routing the TxAudio to the RxAudio.		

6.4. Comsing

Version: 0-0-1

Description: The Single Comm Panel component provides a simulation of a radio/ communications control panel. It manages the transmit and receive selections for up to eight separate communication assets (e.g., radios nav .aides, or intercoms). The eight signals are mixed together into a single output stream and a single audio input can be transmitted across all eight assets. See also ComQuad.

For more information about Comm Panel implementation, see the Advanced Radio tutorial on the ASTi website at:

http://www.asti-usa.com/support/mbv/index.html

Example Uses: Radio Comm Panels, Intercommunications System (ICS) Panels

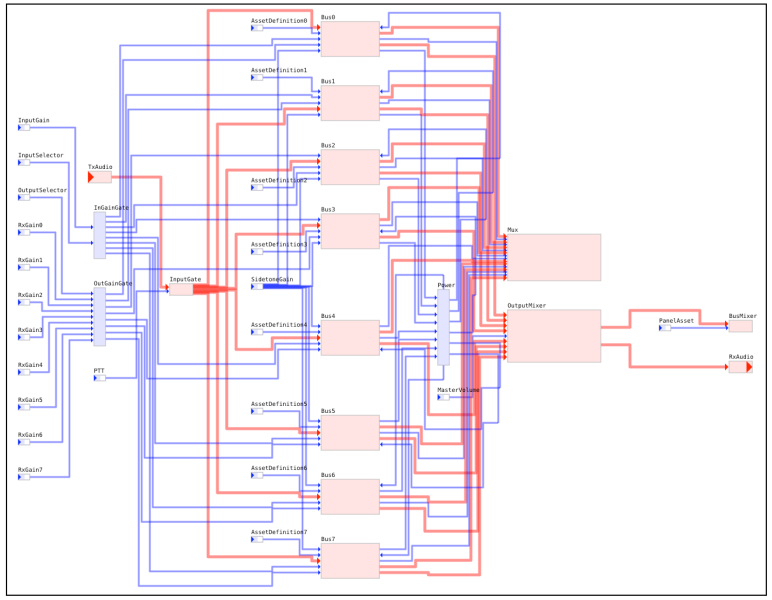


Figure 66: ComSing Schematic Overview

Audio Inputs			
TxAudio	**Variables**	**Type**	**Default Value**
	Ain	audio	n/a
	Description: A connection to an external audio source (like a microphone input from an Iris asset) that is routed to selected Buses for transmission on associated Channels. Routing TxAudio to selected Bus(es) requires all of the following conditions: 1) InputSelector = TRUE bit for selected Bus 2) PTT = TRUE 3) InputGain > 0.0		

Control Inputs			
Asset Defintion0 through Asset Definition 7	**Variables**	**Type**	**Default Value**
	kin->asset->Channel	intercom/IC_channel	0
	Description: AssetDefinition links a Comsing Bus to a Channel. The link provides two-way communications between the Channel and the Bus: - Receive audio is routed from the Channel to the ComSing Bus. - Transmit audio is routed from the ComSing Bus to the Channel. The linking mechanism is a common Channel, which is connected to the AssetDefinition for a particular ComSing Bus and to the AssetDefinition of another signal component (ComSing, Radio, Intercom, etc.). For example - Connecting a Radio to a ComSing Bus: 1) Create a Channel Handle: Tools / Channel Handle: Channel Handle Editor, select New Handle and enter Handle and Channel values. 2) Connect the Channel Handle to a Radio / Asset Definition: Kin>Asset>Channel [Channel value] 3) Connect the Channel Handle to a ComSing / AssetDefinition: Kin>Asset>Channel [Channel value]		
InputGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the input audio signal (TxAudio). Application hint: set the input audio gain using a two step process involving the ComSing and Iris Asset components. First, set the course gain using a combination of the Iris Asset component PreAmp and ATMEL input gains. See the Iris Asset chapter for details. Second, set the fine gain using the ComSing Input Gain. Following this process will insure optimization of the signal-to-noise ratio (and hence, the audio quality) of input audio signal.		

Control Inputs			
Input Selector	**Variables**	**Type**	**Default Value**
	kin	Basic/uint8	0
	Description: Connects the input audio signal (TxAudio) to selected audio Buses. Each selected Bus routes the input audio to a transmitting Channel (internal intercom channel). The InputSelector is a control bitmask capable of routing the TxAudio signal to (up to) eight Buses. Example: A value of 4 (0000 0100 in binary) enables the input to go to Bus 2. A value of 5 (0000 0101 in binary) enables Bus 0 and 2.		
Master Volume	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: A gain multiplier applied uniformly to the amplitude of all receive audio signals in the communication panel.		
Output Selector	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Connects selected audio Buses to the output audio signal (RxAudio). Bus audio is received from other components via Channel assignments. Multiple Bus outputs are mixed down to a single stream prior to routing to RxAudio. The OutputSelector is a control bitmask capable of routing output audio from (up to) eight Buses to RxAudio. For example see Input Selector.		
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: Input control to turn the ComSing power on and off.		

Control Inputs			
PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Input control connection to an external push-to-transmit control signal.		
RxGain0 through RxGain7	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Receive audio gain controls for the corresponding Busses. The gain is applied prior to the audio signal being mixed and sent to the RxAudio Output.		
Sidetone Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	scale_factor	basic/float32	1.0
	Description: Amplitude gain control for the TxAudio that is sent back to the user (i.e. sidetone). Sidetone is effectively hearing your own voice when you transmit. SidetoneGain controls the volume of that voice. The gain is also multiplied by the scale_factor before being applied.		
Sidetone Selector	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	255
	Description: Connects input audio (TxAudio) to the output audio bus (RxAudio). The SidetoneSelector is a control bitmask capable of routing the signal being placed on the buses back out to the RxAudio stream, effectively allowing a user to hear what they are saying as they speak, i.e. sidetone. The sidetone will only be produced for the properly masked buses. See InputSelector for an example of the bit mask.		

Audio Outputs			
RxAudio	**Variables**	**Type**	**Default Value**
	Ain	Audio	n/a
	Description: RxAudio is an audio output, whose source is the mixture of output audio from selected Buses (Bus0 - Bus7). Routing audio from selected Bus(es) to RxAudio requires all of the following conditions: 1) OutputSelector = TRUE bit for each selected Bus 2) OutGainGate = TRUE bit for each selected Bus 3) RxGain > 0.0 for each selected Bus		

Internal Parameters			
InGain Gate	**Variables**	**Type**	**Default Value**
	control_mask	basic/uint8	255
	Description: InGainGate is a control bitmask that is used in combination with the InputSelector control to enable routing of TxAudio to selected ComSing Buses. The bits in the InputSelector value are logic ANDed with corresponding bits in the InGainGate value. The result of this logic operation is used to enable the tx.gain in selected ComSing Buses, enabling the routing of TxAudio to that bus.		
InputGate	**Variables**	**Type**	**Default Value**
	Ptt_invert	basic/boolean	FALSE
	Description: Performs a logic invert on the PTT value. True = Invert		

Internal Parameters			
OutGain Gate	**Variables**	**Type**	**Default Value**
	control_mask	basic/uint8	255
	Description: OutGainGate is a control bitmask that is used in combination with the OutputSelector control to enable routing of output audio from selected CommSing Buses. The bits in the OutputSelector value are logic ANDed with corresponding bits in the OutGainGate value. The result of this logic operation is used to enable the rx.gain in selected CommSing Buses, enabling the routing of output audio from that bus.		
PanelAsset	**Variables**	**Type**	**Default Value**
	kin > asset > channel	intercom/ic_channel	None
	kin > asset > exercise	basic/uint8	0
	Description: The Panel Asset definition will link the output audio to a channel handle. Unlike most other AssetDefintions, the exercise must also be set to 1 in order for the link to properly function. See the AssetDefinition section in the Control Inputs for examples.		
SidetoneGate	**Variables**	**Type**	**Default Value**
	control_mask	basic/uint8	255
	Description: SidetoneGate is a control bitmask that is used in combination with the Sidetone Selector control to enable routing of sidetone audio from selected CommSing Buses. The bits in the SidetoneSelector value are logically AND'ed with corresponding bits in the SidetoneGate value. The result of this logic operation is used to enable the sidetone associated with each bus, routing the TxAudio tot he RxAudio.		

6.5. IcomAudioSrc

Version: 0-0-1

Description: The Intercom Audio Source component is a local intercom object. It allows the user to put audio onto a channel handle and also listen to that channel at the same time. For more information see IcomRx, IcomTx, and Net Intercom.

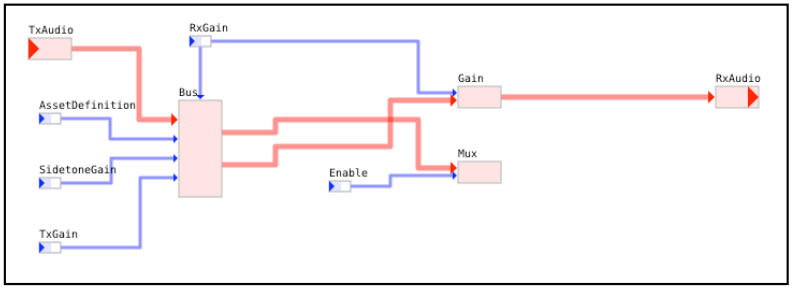


Figure 67: Intercom Audio Source Schematic

Audio Inputs			
TxAudio	**Variables**	**Type**	**Default Value**
	Ain	audio	n/a
	Description: Input audio stream. IcomAudioSrc places all signals from TxAudio onto a particular channel handle defined elsewhere in the component.		

Control Inputs			
AssetDefinition	**Variables**	**Type**	**Default Value**
	kin > asset > channel	intercom/ic_channel	<None>
	Description: AssetDefinition links the IcomAudioSrc to a channel as defined in the Channel Handle Editor Tool. By linking the asset, the component places audio in from TxAudio onto a Channel Handle, and plays all signals on the Channel out the Rx Audio.		
Enable	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: Power switch for the intercom. Setting it to FALSE will not allow the component to route any audio.		
RxGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Amplitude gain control for the output audio stream, RxAudio. The gain is multiplicative, so a value of 1.0 will not affect the RxAudio stream.		
SidetoneGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.0
	Description: Amplitude gain control for the sidetone. Sidetone refers to audio that is being received as it is being transmitted, i.e. hearing yourself while you speak.		
TxGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the TxAudio signal. The gain is applied prior to the signal being transmitted on the audio bus.		

6.6. IcomRx

Version: 0-0-1

Description: The Intercom Receiver component receives audio from an internal intercom channel (Channel Handle) and routes the audio signal to other components in the model.

IcomRx is one of the most basic components for interfacing with the audio bus structure of MBV. In conjunction with IcomTx, the two objects provide for routing audio throughout a model without making the explicit links. In a large model with many subfolders and a complicated structure, the cleanest and most organized way of routing audio is via the IcomRx and IcomTx objects, as opposed to red link arrows that run across multiple folders. For related information also see IcomTx and ComSing components.

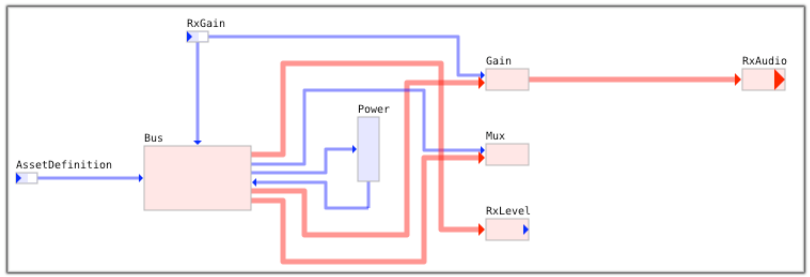


Figure 68: IcomRx Schematic

Control Inputs			
Asset Definition	**Variables**	**Type**	**Default Value**
	kin->asset->channel	intercom/IC_channel	0
	Description: AssetDefinition links an IcomRx Bus to a Channel. The link provides one-way audio routing between the Channel and the IcomRx Bus: receive audio is routed from the Channel to the IcomRx Bus The linking mechanism is a common Channel, which is connected to the IcomRx AssetDefinition and to the AssetDefinition of another signal component (ComSing, Radio, Intercom, etc.).		

Control Inputs			
RxGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the corresponding audio channel selected in the AssetDefinition input.		

Audio Outputs			
RxAudio	**Variables**	**Type**	**Default Value**
	Aout	audio/audio	n/a
	Description: Outputs the selected audio Bus signal after the RxGain input has been applied.		

6.7. IcomTx

Version: 0-0-1

Description: The IcomTx component is used to transmit an audio signal onto an internal intercom channel (channel handles) from within the model.

IcomTx is one of the more basic components for interfacing with the audio bus structure of MBV. In conjunction with IcomRx, the two objects provide for routing audio throughout a model without making the explicit links. In a large model with many subfolders and a complicated structure, the cleanest and most organized way of routing audio is via the IcomRx and IcomTx objects, as opposed to red link arrows that run across multiple folders.

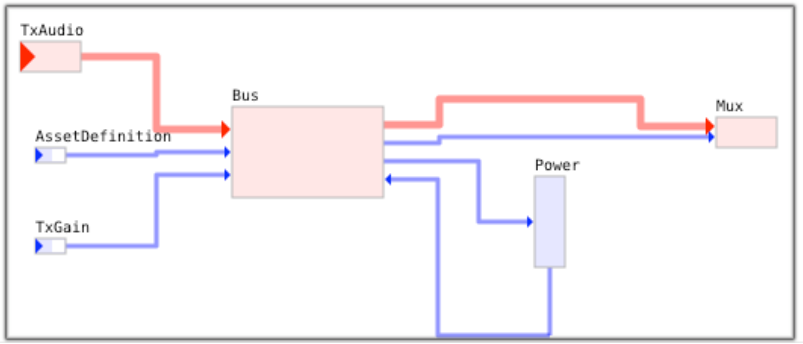


Figure 69: IcomTx Schematic

Audio Inputs			
TxAudio	**Variables**	**Type**	**Default Value**
	Ain	audio	n/a
	Description: PA connection to an external transmit audio source (like a microphone interface from an Iris asset) that is routed to the IcomTx Bus for transmission on the associated Channel.		

Control Inputs			
Asset Definition	**Variables**	**Type**	**Default Value**
	kin->asset->channel	intercom/IC_channel	0
	Description: AssetDefinition links an IcomTx Bus to a Channel. The link provides one-way audio routing between the Channel and the IcomTx Bus: transmit audio from TxAudio is routed to the IcomTx Bus and on to the associated Channel. The linking mechanism is a common Channel, which is connected to the IcomTx AssetDefinition and to the AssetDefinition of another signal component (ComSing, Radio, Intercom, etc.).		
TxGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Amplitude gain control for the TxAudio signal. The gain is applied prior to the signal being transmitted on the audio Bus.		

6.8. SharedRadio

Version: 0-0-2

Description: The Shared Radio component is a special object designed for sharing a radio across multiple Telestras. The component design is tuned for implementation on specific programs, and the component is not recommended for general use. If you are looking to design models that will share radios across a network, contact ASTi for recommendations on how to proceed.

7.0. Radio Components

Radio Signals

The radio object simulates a receiver/transmitter unit, while the receiver objects provide a simulation of radio signal reception only. Each frame, the signal list is scanned to determine which radios or receivers are tuned to the same frequency as the various transmitters, then any signals attached to the matching transmitters (including the TX part of the radio) are passed across to each radio receiver.

The radio object provides a simulation of the radio environment, which includes a discrimination for AM and FM band radios, and full background noise and signal strength effects.

Many radios can be handled at the same time, each with its own position. Positions are specified as world positions in X,Y, and Z or latitude, longitude, and altitude, with all of the range calculations done automatically.

If the terrain interface package is installed, MBV will determine in-tune transmitter-receiver pairs and will generate data packets containing the transmitter-receiver world positions. These packets may be processed by the host computing system, combined with a suitable terrain database to determine highly accurate line-of-sight terrain obscuration checks in addition to the range calculations that are performed. Without the terrain package, ranging is limited by a calculation based upon earth curvature.

A radio can be configured to operate in up to 16 modes, each of which can be custom tailored to provide control over parameters such as Automatic Gain Control (AGC), antenna gain, internal radio noise, and other parameters. The default settings for these modes are commonly used radios, including UHF, VHF and SINCGARS. This allows the user to get started quickly, while retaining the flexibility to further fine tune the simulation.

Radios with Tactical Data Link (TDL) are capable of bridging data message streams, transmitted from host computers, to the ASTi radio environment where they are transported across DIS or HLA networks (in the same manner as voice streams) for reception by radios on remote ASTi nodes and bridging to receiving host computers.

Extending the radios for DIS mode simply involves changing a flag in the attached World Position object and assigning appropriate DIS ID numbers. (See the World Position object for more information on this.) DIS radios can also be attached to other entities on the DIS network, through the Entity object.

Radios are configured for use with Telestras to enable an extended set of advanced radio simulation features. Telestra software includes these optional radio features:

- HLA Communications Environment
- Satcom Server
- HF Server
- ALE Server
- Terrain Interface and Database Server

Refer to the Telestra 3.x User Guide for complete information about these advanced radio features.

The Generic Radio Component

The Generic Radio component provides a generic, high level radio simulation, which includes transmit and receive operations, frequency tuning effects, AM or FM modes, signal strength variation due to range, sidetone, background noise, squelch, and AGC.

As with real world radio equipment, the Generic Radio component can both transmit and receive signals. Typically, a Generic Radio is attached to a Comm Panel, which provides the audio for transmission (usually a microphone), and which routes the received radio audio. Transmitted signals form the output of the Generic Radio and are broadcast to all other radios in the model. With DIS, the radio can transmit to other radios on a network.

MBV scans all of the radio transmitters to determine which radios are in tune to the receiver. The received signal strength is computed for all in tune radios based on the power of the transmitter, the antenna gains of the transmitter and receivers, and the relative world positions. If the terrain interface is installed, the gain factor for the in-tune radios is factored in the calculator. MBV features radio objects with variable bandwidth. This allows accurate simulation of the bandwidth characteristics of various radios, including the tuning effects of wide and narrow band selection.

If frequency hopping or encryption is enabled, the parameters of the transmitter and receiver are compared to see if the audio can be received. (In frequency hopping mode, the frequency field is ignored. The frequency is implied in the selected hopset).

If multiple transmitters are broadcasting on the same frequency, one of two things may happen. For AM signals, the received RF power will be combined and the received audio will be a sum of the transmitted signals in proportion to their signal strength. For FM signals, only the strongest received signal will be included.

Once the received power is determined, the RF signal/noise ratio is calculated. The noise level is determined by thermal noise, internal radio noise, and other parameters which can be set in the radio object. (The default values are set to give some common generic radios, and can be adjusted.) The signal/noise ratio is then compared to the squelch level. If the ratio is less than the squelch level, the signal will not be received. Setting the squelch to zero disables the squelch.

After the signal is determined to be received, the signal power and noise power are multiplied by the AGC. This simulates the AGC operation in a real radio, so even with high signal to noise ratio, a signal will be very low if the maximum AGC value is set low. Additionally, when the squelch is off, the maximum AGC will determine the background noise when no signal is being received.

Once the RF signal/noise is computed and the audio is to be received, it computes the audio signal to noise ratio. This is based on the simulated RF signal/noise ratio, the major and minor modulation type, noise bandwidth of the radio, whether the voice is analog or digital, and other parameters.

The received audio is then routed out. If the radio is attached to one or more communications select panels, they receive the audio. Additionally, if the output feeder in the radio is specified, the audio is sent there as well. (This allows monitoring of a particular radio.) If the radio is connected to an intercom bus (by plugging the intercom bus signal into the transmit signal connection of the radio), the received sound is also put onto the intercom bus.

The radio object can transmit as well as receive, just like a real radio. When the radio receives audio to transmit (either through the transmit signal connection or from a communications select panel), reception is cut off and the radio transmits. Transmit overrides receive for a half duplex radio.

There are also receiver and transmitter objects that are subsets of the radio object. The transmitter will only transmit and the receiver will only receive, but otherwise they function exactly the same way as the full radio.

7.1. ADF_Rx

Version: 0-0-1

Description: The ADF_Rx is a receiver with an added Beat Frequency Oscillator. It provides all of the features of the Receiver component described in this section.

The BFO is an additional oscillator whose tone strength is proportional to the received carrier strength. It is generally used for detecting the Morse code keying present on a continuous wave beacon, which has very basic carrier keying, with no tone modulation.

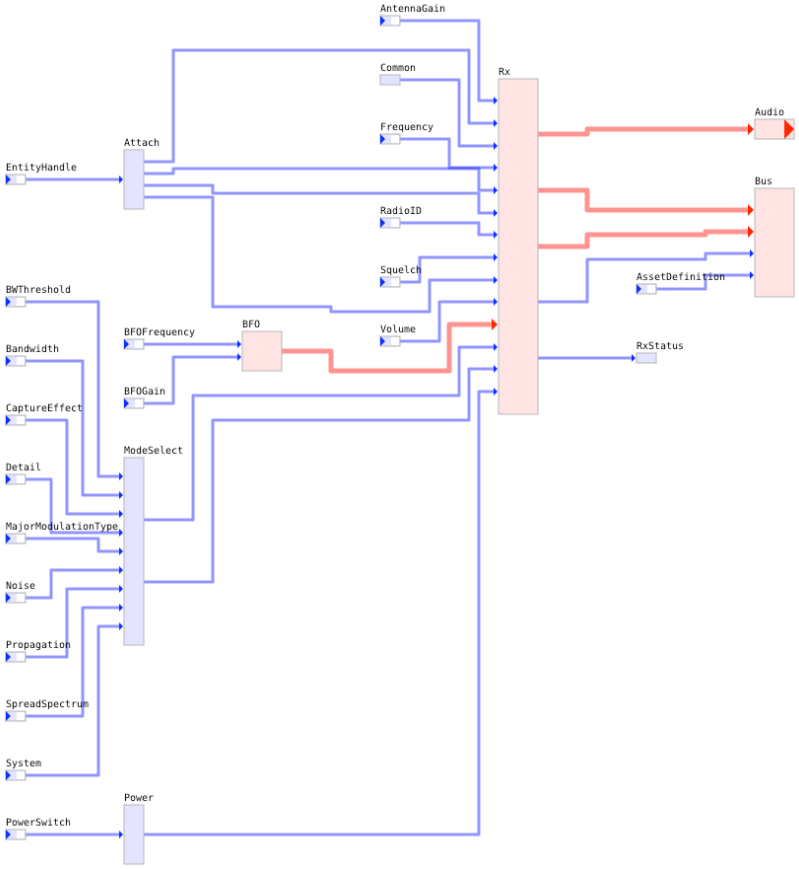


Figure 70: ADF_Rx Schematic

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Value set here will be added to the AntennaGain value set in the MainModeSelect Internal primitive modes. Units are in dBm.		
Asset Definition	**Variables**	**Type**	**Default Value**
	kin->	channel intercom/ ic_channel	0
	Description: Used to assign a channel handle to the ADF_Rx in order to connect it to an intercom component via the local intercom bus.		
Band width	**Variables**	**Type**	**Default Value**
	kin	basic/float32	5000
	Description: Sets the bandwidth of the radio around its center frequency.		
BFO Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/float32	400
	Description: Used to set the frequency of the wave type selected in the BFO internal primitive.		
BFOGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	400
	Description: Sets the amplitude of the wave type selected in the BFO internal primitive.		

Control Inputs			
BW Threshold	**Variables**	**Type**	**Default Value**
	kin	basic/float32	.8
	Description: A percentage amount of the radio's bandwidth that must overlap with another radio's bandwidth in order for the radios to be intune.		
Capture Effect	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: When set to FALSE, all intune signals will be received and mixed together. When set to TRUE, only the intune radio with the strongest signal will be received.		
Detail	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	3
	Description: Sets the detail part of the DIS modulation type code. See the Generic Radio component description for valid options and values. Default is set for Continuous Wave.		
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the ADF_Rx to an entity component. If the value here matches the handle field in an entity component, the ADF_Rx will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/uint64	1000000
	Description: The receive frequency for the ADF Receiver. Units are in Hertz.		

Control Inputs			
Major ModulationType	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set to Amplitude.		
Noise	**Variables**	**Type**	**Default Value**
	kin	basic/float32	-100
	Description: Internal noise figure for the receiver. Units are in dBm.		
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: Turns the radio power on and off. TRUE=on, FALSE=off		
Propagation	**Variables**	**Type**	**Default Value**
	kin->Ionospheric	basic/boolean	FALSE
	kin->Occulting	basic/boolean	TRUE
	kin->Range	basic/boolean	TRUE
	kin->Terrain	basic/boolean	FALSE
	Description: These are propagation flags. A TRUE value will use the respective propagation effect in calculating signal pathloss. A FALSE value will ignore the respective propagation effect in calculating pathloss.		

Control Inputs			
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: ID number of the ADF_Rx that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the EntityHandle control. A value of 0 will default the radio ID to the object number of the component.		
Spread Spectrum	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: Flag used to enable/disable frequency hopping settings. Value must be set to 1 to use frequency hopping settings and 0 disables.		
Squelch	**Variables**	**Type**	**Default Value**
	kin	basic/float32	.2
	Description: When the received RF signal/noise ratio is less than the squelch value given in this field, the AGC gain is set to zero, providing the normal background noise suppression. To disable the squelch, set the squelch level to zero.		
System	**Variables**	**Type**	**Default Value**
	kin	uint16	1
	Description: Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set for Generic.		

Control Inputs			
Volume	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1
	Description: Sets the main volume of the radio.		

Control Outputs			
RxStatus	**Variables**	**Type**	**Default Value**
	kout_radio_status->		
	Distance	basic/float32	n/a
	Frequency	basic/uint64	n/a
	InBand	basic/float32	n/a
	Jammed	basic/boolean	n/a
	Noise	basic/flat32	n/a
	Pathfactor	basic/float32	n/a
	ReceiveState	basic/boolean	n/a
	RxPower	basic/float32	n/a
	SNR	basic/float32	n/a
	Sources	basic/uint16	n/a
	TransmitState	basic/boolean	n/a
	Description: Each respective RxStatus output variable will report its current state or value. The variables can be linked and sent to other components or out to a host computer via UDP packets. These fields are most commonly used to monitor the status of the radio by external means.		

Internal Parameters			
BFO	**Variables**	**Type**	**Default Value**
	Wavetype	audio/waveshape	sine
	Width	basic/float32	0.5
	Width_units	control/width_units	width_is_ratio
	Description: **Wavetype -** Selects one of the available wave types (sine, triangle, sawtooth, square, pulse, inverted pulse). Valid inputs are 0 to 6 and correspond to the following: 0 - Off (no waveform) 1 - Sawtooth 2 - Triangle 3 - Sine 4 - Square 5 - Pulse 6 - Inverted Pulse Within the component, viewer values display as an enumerated type. If there is no connection to the input, the waveform type can be hard coded by setting the kin variable in the Wavetype block to the desired default value. **Width** - Applicable only when Wavetype is set to either Pulse, Inverted Pulse or Square. When a pulse waveform is selected, this variable sets the duration time of the pulse in seconds. When an inverted pulse waveform is selected, this variable sets the duration time in seconds for which the waveform is at 0. Allowable range is 0.0 - n where n < 1/frequency of the current waveform. If 'n' is not within this range the resulting waveform output is 0 (off). When a square wave is selected, the variable sets the duty cycle ratio. Valid range is -1 to 1. A value of zero is a 50 percent duty cycle.		

7.2. Entity

Version: 0-0-1

Description: The entity component in MBV is analogous to the Model Builder world position. The entity is used to set the DIS site, host and entity fields for the DIS ID. The network type DIS or HLA is also set in this field.

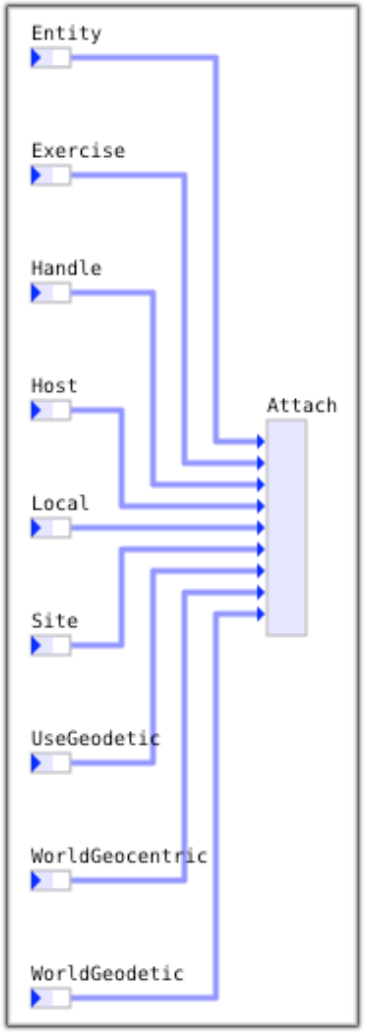


Figure 71: Entity Schematic

Control Inputs			
Entity	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The entity field value for the DIS ID.		
Exercise	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	1
	Description: All entities with the same exercise number will coexist in an exercise.		
Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The handle is used to attach a component to an entity. A component whose handle number matches the entities handle number will have all of the entities values assigned to it. The handle also allows multiple components to attach to an entity.		
Host	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The DIS ID host field value. A default value of 0 will use the fourth byte on the Telestra's IP address.		
Local	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: This determines whether or not the radio is used locally on the Telestra. TRUE means it can be used locally, FALSE means in can be used on the network.		

Control Inputs			
Site	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The DIS ID for the site field value. A default value of 0 will use the third byte of the Telestra's IP address.		
Use Geodetic	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: This determines whether or not the position is geodetic. TRUE means it will use geodetic information, FALSE mean it will use Geocentric information.		
World Geocentric	**Variables**	**Type**	**Default Value**
	kin->X	Basic/Float64	0
	kin->Y	Basic/Float64	0
	kin->Z	Basic/Float64	0
	Description: This is the x, y, and z world position coordinates, the units are in meters.		
World Geodetic	**Variables**	**Type**	**Default Value**
	kin->AboveSeaLevel	basic/float32	0
	kin->Lat	basic/float32	0
	kin->Long	basic/float32	0
	Description: This is the world position in altitude, latitude, and longitude in degrees.		

Internal Parameters			
Attach	**Variables**	**Type**	**Default Value**
	Elevation_scaling_factor	basic/float32	1
	Geocentric_scaling_factor	basic/float32	1
	Description: The elevation scale is a multiplier to convert input elevation units to meters. The geocentric scale is the multiplier to convert input of x, y, and z units to meters. In MBV, the geocentric and geodetic units are natively in meters. These two fields are used if host inputs are not in meters.		

7.3. Generic

Version: 0-0-1

Description: **System** - Used to set the system type. Valid systems and their corresponding value are as follows:

0-Other

1-Generic

2-HQ

3-HQII

4-HQIIA

5-SINCGARS

6-CCTT SINCGARS

MajorModulationType - Used to set the Major Modulation Type for the particular mode. Valid Modulations and their corresponding value are as follows:

0-other

1-Amplitude

2-Amplitude and Angle

3-Angle

4-Combination

5-Pulse

6-Unmodulated

Detail - Used to set the Detailed Modulation type based upon the MajorModulation type selected. The valid detailed modulation types for each of the major modulation types is as follows:

Major Modulation Type	Detailed Modulation
1- Amplitude	0- Other 1- AFSK (Audio Frequency Shift Keying 2- AM (Amplitude Modulation) 3- CW (Continuous Wave Modulation) 4 - DSB (Double Sideband) 5- ISB (Independent Sideband) 6- LSB (Single Band Suppressed Carrier, Lower Sideband Mode) 7- SSB-Fill (Single Sideband Full Carrier 8- SSB-Reduce (Single Band Reduced Carrier) 9- USB (Single Band Suppressed Carrier, Upper Sidband Mode) 10- VSB (Vestigial Sideband)
2- Amplitude and Angle	1- Other 2- Amplitude and Angle
3- Angle	0-Other 1-FM (Frequency Modulation) 2- FSK (Frequency Shift Keying) 3- PM (Phase Modulation)
4- Combination	0- Other 1- Amplitude-Angle-Pulse
5- Pulse	0- Other 1-Pulse
6- Unmodulated	0- Other 1- Continuous Wave emission of an unmodulated carrier.

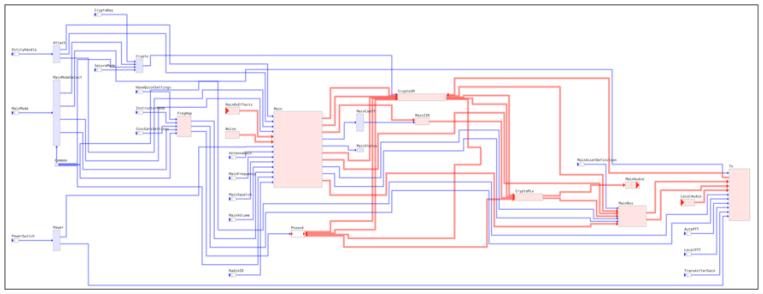


Figure 72: Generic Schematic Overview

Audio Inputs			
Local Audio	**Variables**	**Type**	**Default Value**
	Source	audio/audio	n/a
	Description: A connection for an audio stream that can be transmitted.		
MainRx Effects	**Variables**	**Type**	**Default Value**
	Source	audio/audio	n/a
	Description: The applied input signal is mixed with any received signal and then sent out as output. This allows the user to mix an audio signal with a received signal.		

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Values set here will be added to the AntennaGain value set in the MainModeSelect Internal primitive modes. Units are in dBm.		

Control Inputs			
Auto PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: When this is enabled the radio will go into transmit when an active audio signal is detected in the LocalAudio input.		
Crypto Key	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: If two radios are using encryption then they must have matching crypto keys for the crypto modes. This field must match in order for proper radio tuning.		
CrypLib Override Enable	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Setting this flag to TRUE will cause the radio to use the value located in CryptoLibraryOverride for the Crypto Library instead of the value stored inside the mode. See the Crypto Library section of MainModeSelect for more information.		
Crypto Library Override	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The Generic Radio normally detects the Crypto Library from the MainModeSelect Primitive. This field, in conjunction with CrypLibOverrideEnable, will allow the user to drive the Crypto Library from the host or another component versus using the value stored in the mode table. See the MainModeSelect and Psound primitive for more information.		

Control Inputs			
Crypto System Override	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Setting this input to a non-zero value will cause the radio to override the Crypto System parameter found in the MainModeSelect primitive. This allows the radio's Crypto System to be host driven, or controlled by another component upstream. For more information see the Crypto System field of the MainModeSelect primitive.		
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the Generic Radio to an entity component. If the value here matches the handle field in an entity component, the Generic Radio will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Have Quick Settings	**Variables**	**Type**	**Default Value**
	kin->NetID	basic/uint16	0
	kin->SynceOffset	basic/uint32	0
	kin->TimeofDay	basic/uint16	0
	kin->TransecKey	basic/uint16	0
	kin->WordofDayID	basic/uint16	0
	Description: These are used to set HaveQuick parameters. The HaveQuick parameters must match in order for Radios in HaveQuick mode to be in tune.		

Control Inputs			
Instructor Mode	**Variables**	**Type**	**Default Value**
	kin->bandwidth	basic/boolean	TRUE
	kin->Crypto	basic/boolean	TRUE
	kin->DisableJammers	basic/boolean	FALSE
	kin->DisablePropagation	basic/boolean	FALSE
	kin->Enable	basic/boolean	FALSE
	kin->Mode	basic/boolean	TRUE
	kin->ModeParms	basic/boolean	TRUE
	Range	basic/boolean	TRUE
	Description: **Bandwidth** - When TRUE, disables the bandwidth matching condition between a radio with instructor mode enabled and all receiving radios. **Crypto** - When TRUE, disables all crypto matching conditions between a radio with instructor mode enabled and all receiving radios. **DisableJammers** - When TRUE, enables radio with instructor mode enabled to ignore all jamming effects (Radio is immune to being jammed). **DisablePropagation** - When TRUE, disables all propagation effects on radio with instructor mode enabled. (Effectively puts instructor radio at center of earth - overriding world position.) **Enable** - When TRUE, enables instructor radio settings as described in this Input control. **Mode** - When TRUE, a radio with instructor mode enabled will be able to transmit to any radio without consideration for mode matching. (Instructor Radio is effectively transmitting on all modes). **Modeparms** - When TRUE, a radio with instructor mode enabled will be able to transmit to any radio without consideration for frequency hoping parameters. (Instructor radio will always match a receiving radios frequency hoping parameters.) continues next column		

Control Inputs			
Instructor Mode continued	**Range** - When TRUE, a radio with instructor mode enabled will override any receiving radios ranging effects. (Effectively placing the receiving radio at the center of the earth.)		
Local PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: This input is used to key the transmitter. When set to TRUE the radio goes into transmit state.		
Main Asset Definition	**Variables**	**Type**	**Default Value**
	kin	channel intercom/ ic_channel	0
	Description: Used to assign a channel handle to the Generic Radio in order to connect it to an intercom component via the local intercom bus.		
Main Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/uint64	100000000
	Description: This is the receive frequency for a radio, also used as the transmit frequency if the transmit frequency input control equals 0.		
Main Mode	**Variables**	**Type**	**Default Value**
	kin	basic/int32	1
	Description: This is used to select the radio mode (which are set in the MainModeSelect internal primitive). One corresponds to mode one in the MainModeSelect internal primitive, two corresponds to mode two etc.		

Control Inputs			
Main Squelch	**Variables**	**Type**	**Default Value**
	kin	basic/float32	.2
	Description: When the received RF signal/noise ratio is less than the squelch value given in this field, the AGC gain is set to zero, providing the normal background noise suppression. To disable the squelch, set the squelch level to zero.		
Main Volume	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1
	Description: Sets the main volume of the radio. The volume scales the amplitude of both receiver and transmit signals uniformly.		
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: This turns the radio power on and off. TRUE = On, FALSE = Off		

Control Inputs			
Psound	**Variables**	**Type**	**Default Value**
	Playfiles->play-files[0-59]	audioplaycmds	unknown
	Description: The internal parameter is used in conjunction with the crypto settings. Each playfile is associated with a crypto state play-sound. Of the 60 playsounds, there are 5 sets of 12 which correspond to the crypto library number in the MainModeSelect internal primitive. A crypto library number 0 corresponds to the first 12 positions. A crypto library number of 1 corresponds to the second set of 12 playsound indices, etc. See the list below for the playfile index which corresponds to a crypto state sound. For each set of 12 playsounds, the following index positions correspond to the different types of crypto playsound actions: 1 = rx_pre1 2 = rx_pre2 3 = rx_post 4 = rx_clr 5 = rx_match 6 = rx_mismatch 7 = tx_pre1 8 = tx_pre2 9 = tx_post 10 = tx_clr 11 = tx_crypto 12 = no playsound		

Control Inputs			
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: ID number of the Generic Radio that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the Entity-Handle control. A value of 0 will default the radio ID to the object number of the component.		
Secure Mode	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: A radio in a secure state can only receive and transmit with another radio in a secure state. A radio in secure state can receive an unsecured radio but an unsecured radio cannot receive a secured radio.		

Control Inputs			
SincGars Settings	**Variables**	**Type**	**Default Value**
	kin->ClearChannel	basic/uint16	0
	kin->HopSetID	basic/uint16	0
	kin->LockoutSetID	basic/uint16	0
	kin->NetID	basic/uint16	0
	kin->SyncOffset	basic/uint32	0
	kin->TSecID	basic/uint16	0
	Description: These are used to set SINCGARS parameters. **NetID** - Identifies the frequency hopping network **HopSetID** - Identifies the set of frequencies used in hopping pattern **LockoutSetID** - Identifies the set of frequencies that are excluded from hopping pattern **TSecID** - Identifies the transmission security key used in generating hopping pattern. **SyncOffset** - Specifies whether the radio is starting or continuing a transmission. **ClearChannel** - 0 Specifies Not clear, 1 Specifies Clear Channel. Clear channel value of 1 makes the transmission immune to any type of signal degradation. **When simulating a frequency hopping radio, the Generic radio at no time hops its frequency. The frequency remains fixed. The values in the above fields are simply used for matching purposes. i.e. If another SINCGARS radio has matching values in its fields then the radios will be in-tune. If the fields don't match, they are out of tune. **		

Control Inputs			
Transmitter Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Used to scale the transmit power of the transmitter. Actual transmit power is TransmitterGain multiplied by the transmitter power value in the mode sections of the MainModeSelect internal primitive.		

Control Outputs			
Main Status	**Variables**	**Type**	**Default Value**
	kout_radio_status->		
	Distance	basic/float32	n/a
	Frequency	basic/uint64	n/a
	InBand	basic/float32	n/a
	Jammed	basic/boolean	n/a
	Noise	basic/float32	n/a
	PathFactor	basic/float32	n/a
	ReceiveState	basic/boolean	n/a
	RxPower	basic/float32	n/a
	SNR	basic/float32	n/a
	Sources	basic/uint16	n/a
	TransmitState	basic/boolean	n/a
	Description: Each respective MainStatus output variable will report its current state or value. The variables can be linked and sent to other components or out to a host computer via UDP packets. These fields are most commonly used to monitor the status of the radio by external means.		

Internal Parameters			
Main Mode Select	**Variables**	**Type**	**Default Value**
	Mode 1->		
	AntennaGain	basic/float32	0
	BWOverlapThreshold	basic/float32	.8
	Bandwidth	basic/float32	25000
	CaptureEffect	basic/uint16	0
	CryptoLibrary	basic/uint16	0
	CryptoSystem	basic/uint16	0
	Detail	basic/uint16	2
	DigitalMode	basic/uint16	0
	Encoding	basic/uint16	0
	FullDuplex	basic/uint16	0
	MajorModulation-Type	basic/uint16	1
	Noise	basic/float32	-105
	Propagation->		
	Ionospheric	basic/boolean	FALSE
	Occulting	basic/boolean	TRUE
	Range	basic/boolean	TRUE
	Terrain	basic/boolean	TRUE
	Mode 1->		
	ReceiveOnly	basic/uint16	0
	RxFreqOffset	basic/int32	0
	SpreadSpectrum	basic/uint16	0
	System	basic/uint16	1
	TxPower	basic/float32	10.0
	Mode defaults 2-15 differ from those shown above for Mode 1. See additional table at end of section for Mode 1-15 MajorModulationType, Detail, and System. Continues next column		

Main Mode Select cont.	Variables	Type	Default Value
	Description: AntennaGain - Adds a dB gain to the received signal. Units are in dBm. **BWOverlapThreshold** - As a percentage, the amount of a radio's bandwidth that must overlap with another radio's bandwidth in order for the radios to be intune. **Bandwidth** - Sets the bandwidth of the radio around its center frequency. **Capture Effect** - Used with a value of 0 all intune signals will be mixed together and heard. With a value of 1, only the intune radio with the strongest signal will be heard. **Crypto Library** - The number here corresponds to a set of playfiles in the psound internal component, which will be used for crypto sounds. 0 corresponds to the playfiles 1-12 1 2nd 12 2 3rd 12 3 4th 12 4 5th 12 **Crypto System** - Sets the type of crypto system for the radio. The following are valid values as per the DIS standard. 0- Other 1- KY-28 2 - KY-58 3- Narrow Spectrum Secure Voice (NSVE) 4 - Wide Spectrum Secure Voice (WSVE) 5 - SINCGARS ICOM **Detail** - Sets the detail part of the DIS modulation type code. See the Generic Radio component description for valid options and values. **Digital Mode** - 0 disables dm, 1 enables. Continues next page		

Main Mode Select cont.	**Encoding** - Allows overwriting in the encoding asset in the encoding input, 0 uses the radios. **FullDuplex** - 0 is half duplex, 1 is full duplex. **MajorModulationType** - Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. **Noise** - Internal noise figure for the radio receiver. Units are in dBm. **Propagation** - All four set the effects as part of the pathloss, FALSE not used in calculating the pathloss, TRUE it is used in pathloss. **ReceiveOnly** - 0 means off will tx and rx, 1 will only rx and not tx **ReceiveFrequencyOffset** - An offset in hertz applied to the main frequency **SpreadSpectrum** - Flag used to enable/disable frequency hopping settings. Value must be set to 1 to use frequency hopping settings. 0 disables. **System** - Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. **TxPower** - The power used in transmission.

Mode	MajorModulation Type	Detail	System
Mode 1	Amplitude	Amplitude & Angle	Generic
Mode 2	Angle	Frequency Modulation	Generic
Mode 3	Angle	Frequency Modulation	Generic
Mode 4	SATCOM*	Fixed Delay	Generic
Mode 5	Amplitude & Angle	Other	Generic
Mode 6	Angle	FM	HQ
Mode 7	Amplitude & Angle	Other	SINCGARS
Mode 8	Amplitude & Angle	Other	HQII
Mode 9	Angle	Frequency Modulation	HQII
Mode 10	Amplitude	Amplitude Modulation	Generic
Mode 11	Amplitude	Amplitude Modulation	Generic
Mode 12	Amplitude & Angle	Other	Generic
Mode 13	Other	-	-
Mode 14	Other	-	-
Mode 15	SATCOM*	Fixed Delay	Generic
*ASTi Equipment Only			

7.4. Jammer

Version: 0-0-1

Description: The Jammer component provides a specialized transmitter for broad band signal sources. When the Jammer is provided with certain frequency and an appropriate bandwidth it is capable of jamming radio signals of the same Modulation type that fall within the bandwidth envelope.

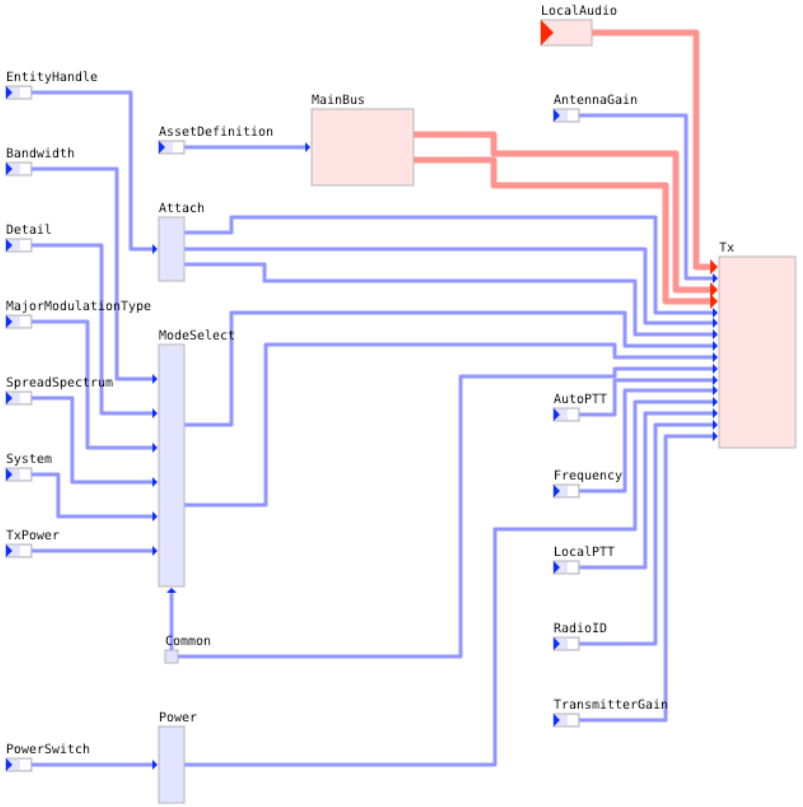


Figure 73: Jammer Schematic

Audio Inputs			
Local Audio	**Variables**	**Type**	**Default Value**
	Source	audio/audio	n/a
	Description: A connection for an audio stream that can be transmitted.		

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Values set here will be added to the AntennaGain value set in the MainModeSelect Internal primitive modes. Units are in dBm.		
Asset Definition	**Variables**	**Type**	**Default Value**
	kin->	Channel Intercom/ ic_channel	0
	Description: Used to assign a channel handle to the Jammer in order to connect it to an intercom component via the local intercom bus.		
AutoPTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: When this is enabled the jammer will go into transmit when an active audio signal is detected in the local audio input.		
Bandwidth	**Variables**	**Type**	**Default Value**
	kin	basic/float32	10000000
	Description: Sets the bandwidth of the jammer around its center frequency. Units are in Hertz.		

Control Inputs			
Detail	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the detail part of the DIS modulation type code. See the Generic Radio component description for valid options and values. Default is set for Pulse.		
Encoding	**Variables**	**Type**	**Default Value**
	kin	radio/encoding	default_encoding
	Description: Determines the audio encoding type valid options are CVSD_mil, MuLaw, PCM_16, Default encoding. DE selection will use the encoding type as set via RMS radios settings page.		
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the Jammer to an entity component. If the value here matches the handle field in an entity component, the Jammer will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/uint64	300000000
	Description: Sets the center frequency for the jammer.		
Local PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Used to key the transmitter. When TRUE the jammer goes into transmit state.		

Control Inputs			
Major Modulation Type	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	5
	Description: Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set for Pulse.		
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: Used to turn the jammer power on and off.		
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: ID number of the Jammer that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the EntityHandle control. A value of 0 will default the radio ID to the object number of the component.		
Sample Rate	**Variables**	**Type**	**Default Value**
	kin	Radio/Samplerate	default_rate
	Description: This determines the audio sampling rate for the radio valid options include: sample_8k, sample_16k, sample_32k, sample_48k, default_rate. DR selection will use the audio sampling type as set via RMS radios settings page.		
Spread Spectrum	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: N/A for this component.		

Control Inputs			
System	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set for Generic.		
Transmitter Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Scales the transmit power value input. Final transmitter power is the Transmit power multiplied by the Transmitter Gain.		
TxPower	**Variables**	**Type**	**Default Value**
	kin	basic/float32	100.0
	Description: Sets the transmit power for the jammer.		

7.5. Keyer

Version: 0-0-1

Description: The Keyer component is used to translate 4 letter words into Morse Code. The component allows control over the wordrate, interval and the frequency of the generated tone.

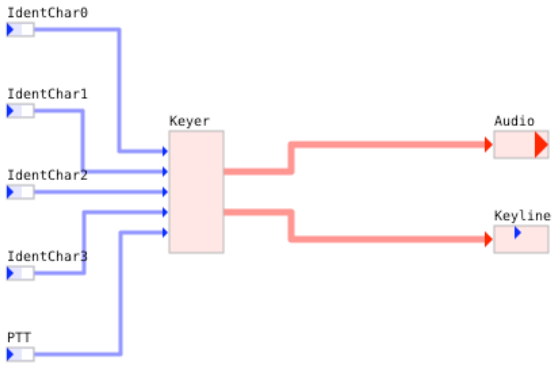


Figure 74: Keyer Schematic

Control Inputs			
IdentChar 0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The first letter of the word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
IdentChar 1	**Variables**	**Type**	**Default Value**
	kin	Uint8	0
	Description: The second letter of the word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		

Control Inputs			
IdentChar 2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The third letter of the word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
IdentChar 3	**Variables**	**Type**	**Default Value**
	kin	Uint8	0
	Description: The fourth letter of the word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Used trigger the Morse code keying. When value remains TRUE, the component will continuously key the Morse Code word.		

Audio Outputs			
Audio	**Variables**	**Type**	**Default Value**
	Audio	audio/audio	n/a
	Description: Outputs the keyed word when the PTT control is enabled.		

Control Outputs			
Keyline	**Variables**	**Type**	**Default Value**
	Aout	basic/boolean	FALSE
	Description: Outputs the keyed state of the Keyer. When keying, an output of TRUE is shown.		

Internal Parameters			
Keyer	**Variables**	**Type**	**Default Value**
	Freq	basic/float32	1020
	Description: Used to set the frequency of the tone.		
	Variables	**Type**	**Default Value**
	Interval	basic/uint16	5
	Description: Sets the repeat rate of the Morse word. Units are in seconds.		
	Variables	**Type**	**Default Value**
	wordrate	basic/uint8	8
	Description: Determines the rate at which the word is played. Units are in dots per second.		

7.6. Marker

Version: 0-0-1

Description: The Marker component provides the ability to add the Outer Marker, Middle Marker, Inner Marker, and Fan Marker navigational beacons to a model. The default values are set to FAA standards. Whilst the component provides four separate beacons, the component can be considered only one of the beacons at a time. An input control allows the selection of which beacon to use. If all four beacons were desired to be used in a simulation, four instances of this component would need to be used, each selected to be one of the marker beacons. An entity attach field is also provided in order to give the beacon a world location.

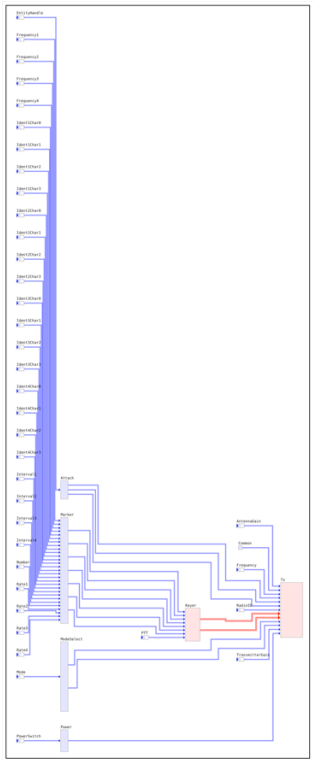


Figure 75: Marker Schematic Overview

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Values set here will be added to the AntennaGain value set in the MainModeSelect Internal primitive modes. Units are in dBm.		
Encoding	**Variables**	**Type**	**Default Value**
	kin	radio/encoding	default_encoding
	Description: Determines the audio encoding type. valid options are CVSD_mil, MuLaw, PCM_16, Default encoding. DE selection will use the encoding type as set via RMS radios settings page.		
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the Marker to an entity component. If the value here matches the handle field in an entity component, the Marker will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	uint64	3000000
	Description: Sets the carrier frequency for the beacon. Units are in Hertz. Default is 75 MHz as per FAA standards.		
Frequency 1	**Variables**	**Type**	**Default Value**
	kin	basic/float32	400
	Description: Sets the frequency for the keyed tone. Default is set to 400Hz for the Outer Marker.		

Control Inputs			
Frequency 2	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1300
	Description: Sets the frequency for the keyed tone. Default is set to 1300Hz for the Middle Marker.		
Frequency 3	**Variables**	**Type**	**Default Value**
	kin	basic/float32	3000
	Description: Sets the frequency for the keyed tone. Default is set to 3000Hz for the Inner Marker.		
Frequency 4	**Variables**	**Type**	**Default Value**
	kin	basic/float32	3000
	Description: Sets the frequency for the keyed tone. Default is set to 3000Hz for the Fan Marker.		
Ident1 Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	77
	Description: Outer Marker Morse code letter. Default value corresponds to dash-dash for Outer Marker.		
Ident1 Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Second letter of first word. Not used for Marker Beacon operations.		

Control Inputs			
Ident1 Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Third letter of first word. Not used for Marker Beacon operations.		
Ident1 Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Fourth letter of first word. Not used for Marker Beacon operations.		
Ident2 Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	65
	Description: Middle Marker Morse code letter. Default value corresponds to dot-dash for Middle Marker.		
Ident2 Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Second letter of second word. Not used for Marker Beacon operations.		
Ident2 Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Third letter of second word. Not used for Marker Beacon operations.		

Control Inputs			
Ident2 Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Fourth letter of second word. Not used for Marker Beacon operations.		
Ident3 Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	69
	Description: Inner Marker Morse code letter. Default value corresponds to dot for Inner Marker.		
Ident3 Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Second letter of third word. Not used for Marker Beacon operations.		
Ident3 Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Third letter of third word. Not used for Marker Beacon operations.		
Ident3 Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Fourth letter of third word. Not used for Marker Beacon operations.		

Control Inputs			
Ident4 Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Can be used to set Morse code key for Fan Marker. No default tone set for the Fan Marker.		
Ident4 Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Second letter of fourth word. Not used for Marker Beacon operations.		
Ident4 Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Third letter of fourth word. Not used for Marker Beacon operations.		
Ident4 Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Fourth letter of fourth word. Not used for Marker Beacon operations.		
Interval1	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: Used to set the delay between repeating the first keyed word. Units are in seconds.		

Control Inputs			
Interval2	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Used to set the delay between repeating the second keyed word. Units are in seconds.		
Interval3	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: Used to set the delay between repeating the third keyed word. Units are in seconds.		
Interval4	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Used to set the delay between repeating the keyed word. Units are in seconds.		
Mode	**Variables**	**Type**	**Default Value**
	kin	basic/int32	1
	Description: This is used to select the mode for the transmitter beacon (which are set in the mainselect internal primitive) 1 corresponds to mode 1 in the mainselect primitive, 2 corresponds to mode 2, etc.		
Number	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Selects which marker beacon (IdentChar or word) is to be transmitted. A value of 0 selects none. A value of 1 selects Ident1Char[0-3] (the Outer Marker). A value of 2 selects the second word (Middle Marker). A value of 3 selects the third word (Inner Marker). A Value of 4 selects the fourth word (Fan Marker).		

Control Inputs			
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: Used to turn power on and off to the radio. TRUE = On, FALSE = Off		
PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Used trigger the active transmission of the Marker Beacon. When the PTT value remains TRUE, the beacon will continuously transmit.		
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: ID number of the Marker that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the EntityHandle control. A value of 0 will default the radio ID to the object number of the component.		
Rate1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	8
	Description: Used to set the keying rate for the first word (Outer Marker). Default is set for two dashes per second.		
Rate2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	12
	Description: Used to set the keying rate for the second word (Middle Marker). Default is set for 95 dot-dashes per minute.		

Control Inputs			
Rate3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	8
	Description: Used to set the keying rate for the third word (Inner Marker). Default is set for six dots per second.		
Rate4	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	8
	Description: Used to set the keying rate for the fourth word (Outer Marker).		
Sample Rate	**Variables**	**Type**	**Default Value**
	kin	Radio/Samplerate	default_rate
	Description: This determines the audio sampling rate for the radio valid options include: sample_8k, sample_16k, sample_32k, sample_48k, default_rate. DR selection will use the audio sampling type as set via RMS radios settings page.		
Transmit-ter Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Scales the transmit power value input. (As set in the ModeSelect internal primitive for each mode) Final transmitter power is the Transmit power multiplied by the Transmitter Gain.		

Internal Parameters			
Main Mode Select	**Variables**	**Type**	**Default Value**
	Mode 1->		
	AntennaGain	basic/float32	0
	BWOverlapThreshold	basic/float32	.8
	Bandwidth	basic/float32	25000
	CaptureEffect	basic/uint16	0
	CryptoLibrary	basic/uint16	0
	CryptoSystem	basic/uint16	0
	Detail	basic/uint16	2
	DigitalMode	basic/uint16	0
	Encoding	basic/uint16	0
	FullDuplex	basic/uint16	0
	MajorModulation-Type	basic/uint16	1
	Noise	basic/float32	-105
	Propagation->		
	Ionospheric	basic/boolean	FALSE
	Occulting	basic/boolean	TRUE
	Range	basic/boolean	TRUE
	Terrain	basic/boolean	TRUE
	Mode 1->		
	ReceiveOnly	basic/uint16	0
	RxFreqOffset	basic/int32	0
	SpreadSpectrum	basic/uint16	0
	System	basic/uint16	1
	TxPower	basic/float32	10.0
	Mode defaults 2-15 differ from those shown above for Mode 1. See additional table at end of section for Mode 1-15 MajorModulationType, Detail, and System. Continues on next page		

Main Mode Select cont.	Variables	Type	Default Value
	Description: AntennaGain - a mode specific scaler to the antenna power **BWOverlapThreshold** - The percentage of a radio's bandwidth that must overlap with another radio's bandwidth in order for the radios to be intune **Bandwidth** - sets the bandwidth of the radio around its center frequency **Capture Effect** - N/A for this component. **Crypto Library** - N/A for this component. **Crypto System** - N/A for this component. **Detail** - Sets the detail part of the DIS modulation type code. See the Generic Radio component description for valid options and values. **Digital Mode** - 0 disables dm, 1 enables **Encoding** - this allows overwrite the encoding asset in the encoding input, 0 uses the components Encoding Input value. **FullDuplex** - N/A for this component. **MajorModulationType** - Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. **Noise** - N/A for this component. **Propagation** - A TRUE value will use the respective propagation effect in calculating signal pathloss. A FALSE value will ignore the respective propagation effect in calculating pathloss. **ReceiveOnly** - 0 means will tx and rx, 1 will only rx and not tx **ReceiveFrequencyOffset** - N/A for this component. **SpreadSpectrum** - N/A for this component. **System** - Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. **TxPower** - The power used in transmission. Continues next page		

Main Mode Select cont.	
	Encoding - Allows overwriting in the encoding asset in the encoding input, 0 uses the radios. **FullDuplex** - 0 is half duplex, 1 is full duplex. **MajorModulationType** - Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. **Noise** - Internal noise figure for the radio receiver. Units are in dBm. **Propagation** - All four set the effects as part of the pathloss, FALSE not used in calculating the pathloss, TRUE it is used in pathloss. **ReceiveOnly** - 0 means off will tx and rx, 1 will only rx and not tx **ReceiveFrequencyOffset** - An offset in hertz applied to the main frequency **SpreadSpectrum** - Flag used to enable/disable frequency hopping settings. Value must be set to 1 to use frequency hopping settings. 0 disables. **System** - Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. **TxPower** - The power used in transmission.

Mode	MajorModulation Type	Detail	System
Mode 1	Amplitude	Amplitude & Angle	Generic
Mode 2	Angle	Frequency Modulation	Generic
Mode 3	Angle	Frequency Modulation	Generic
Mode 4	SATCOM*	Fixed Delay	Generic
Mode 5	Amplitude & Angle	Other	HQ
Mode 6	Angle	FM	HQ
Mode 7	Amplitude & Angle	Other	SINCGARS
Mode 8	Amplitude & Angle	Other	HQII
Mode 9	Angle	Frequency Modulation	HQII
Mode 10	Amplitude	Amplitude Modulation	Generic
Mode 11	Amplitude	Amplitude Modulation	Generic
Mode 12	Amplitude & Angle	Other	Generic
Mode 13	Other	-	-
Mode 14	Other	-	-
Mode 15	SATCOM*	Fixed Delay	Generic
*ASTi Equipment Only			

7.7. NDB_Tx

Version: 0-0-1

Description: The Non Directional Beacon (NDB) component is a transmitter with an embedded identifier tone element. It keys a four letter word on a user defined tone to produce a simulation of an NDB which can be heard over a suitably tuned radio or receiver.

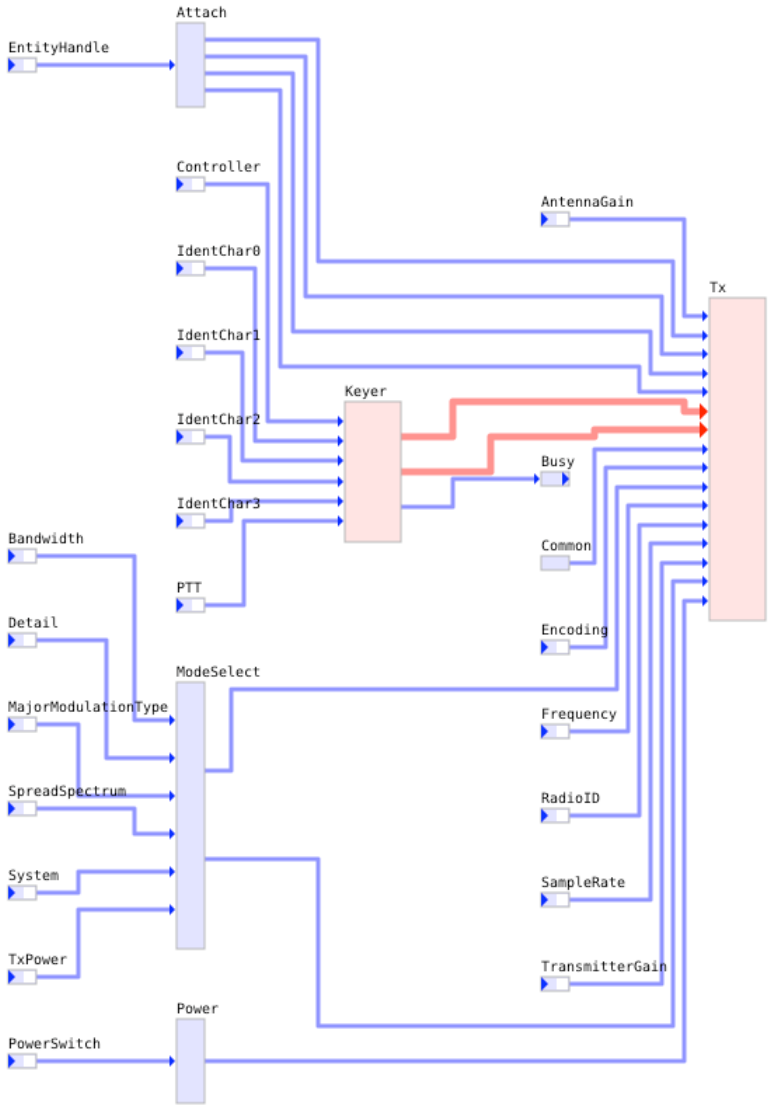


Figure 76: NDB_Tx Schematic

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: N/A for this component.		
Band-width	**Variables**	**Type**	**Default Value**
	kin	basic/float32	5000
	Description: Sets the bandwidth of the jammer around its center frequency. Units are in Hertz.		
Controller	**Variables**	**Type**	**Default Value**
	Description: Controller does not work for this component. To set parameters, use the corresponding input controls or the Keyer Internal Primitive.		
Detail	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	3
	Description: Sets the detail part of the DIS modulation type code. See the Generic Radio component description for valid options and values. Default is set for Continuous Wave.		
Encoding	**Variables**	**Type**	**Default Value**
	kin	radio/encoding	default_encoding
	Description: Determines the audio encoding type valid options are CVSD_mil, MuLaw, PCM_16, Default encoding. DE selection will use the encoding type as set via RMS radios settings page.		

Control Inputs			
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the NDB_Tx to an entity component. If the value here matches the handle field in an entity component, the NDB_Tx will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/uint64	?
	Description: Sets the carrier frequency for the Beacon.		
Ident1Char0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The first letter of the keyed word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
Ident1Char1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The second letter of the keyed word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
Ident1Char2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The third letter of the keyed word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		

Control Inputs			
Ident1Char3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The fourth letter of the keyed word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
Major Modulation Type	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set for Amplitude Modulation.		
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: Used to turn power on and off to the jammer.		
PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: Used trigger the transmitter of the beacon. When value remains TRUE, the NDB beacon will continuously transmit its keyed tone.		
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: ID number of the NDB_Tx that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the EntityHandle control. A value of 0 will default the radio ID to the object number of the component.		

Control Inputs			
Sample Rate	**Variables**	**Type**	**Default Value**
	kin	radio/samplerate	default_rate
	Description: This determines the audio sampling rate for the radio valid options include: sample_8k, sample_16k, sample_32k, sample_48k, default_rate. DR selection will use the audio sampling type as set via RMS radios settings page.		
Spread Spectrum	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: N/A for this component		
System	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set to Generic.		
Transmitter Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Scales the transmit power value input. Final transmitter power is the Transmit power multiplied by the Transmitter Gain.		
TxPower	**Variables**	**Type**	**Default Value**
	kin	basic/float32	15.0
	Description: Sets the transmit power for the beacon.		

Control Outputs			
Busy	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	FALSE
	Description: Ouputs the keyed state of the Beacon. When keying an output of TRUE is shown.		

Internal Parameters			
Keyer	**Variables**	**Type**	**Default Value**
	Freq	basic/float32	400
	Interval	basic/uint16	5
	Wordrate	basic/uint8	10
	Description: **Freq** - This is used to set the frequency of the tone. **Interval** - Sets the repeat rate of the morse world. Units are in seconds. **Wordrate** - Determines the rate at which the word is keyed.		

7.8. NetIntercom

Version: 0-0-1

Description: The network intercom object provides a simple simulation of an intercom bus between communication panels which are located across the DIS network.

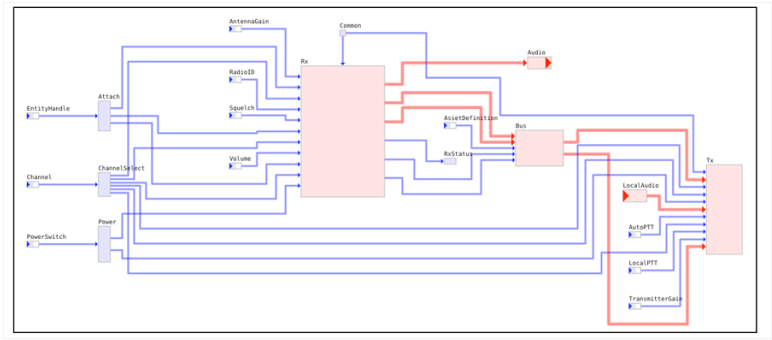


Figure 77: NetIntercom Schematic Schematic

Audio Inputs			
Local Audio	**Variables**	**Type**	**Default Value**
	Source	audio/audio	n/a
	Description: A connection for an audio stream that can be transmitted.		

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Values set here will be added to the AntennaGain value set in the MainModeSelect Internal primitive modes. Units are in dBm.		
Asset Definition	**Variables**	**Type**	**Default Value**
	kin->	channel intercom/ ic_channel	0
	Description: Used to assign a channel handle to the NetIntercom in order to connect it to an intercom component via the local intercom bus.		
AutoPTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: When this is enabled the intercom will go into transmit when an active audio signal is detected in the local audio input.		
Channel	**Variables**	**Type**	**Default Value**
	kin	basic/uint32	1
	Description: Defines the intercom channel number. In order to be received across the network, both channel numbers must be equal. Valid values: 0-10,000.		
Encoding	**Variables**	**Type**	**Default Value**
	kin	basic/encoding	Default_encoding
	Description: Determines the audio encoding type, valid options are CVSD_mil, MuLaw, PCM_16, Default encoding. DE selection will use the encoding type as set via RMS radios settings page.		

Control Inputs			
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the NetIntercom to an entity component. If the value here matches the handle field in an entity component, the NetIntercom will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: This is used to turn power on and off to the intercom.		
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: ID number of the NetIntercom that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the EntityHandle control. A value of 0 will default the radio ID to the object number of the component.		
Sample Rate	**Variables**	**Type**	**Default Value**
	kin	radio/samplerate	default_rate
	Description: This determines the audio sampling rate for the radio valid options include: sample_8k, sample_16k, sample_32k, sample_48k, default_rate. DR selection will use the audio sampling type as set via RMS radios settings page.		

Control Inputs			
Squelch	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.2
	Description: When the received RF signal/noise ratio is less than the squelch value given in this field, the AGC gain is set to zero, providing the normal background noise suppression. To disable the squelch, set the squelch level to zero.		
Transmitter Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Sets the gain of the transmitter. Since intercoms do not normally have pathloss or propagation effects associated with them, this field is typically left at the default and not used.		
Volume	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1
	Description: Sets the main volume of the intercom. The value here scales the amplitude of the signal being received.		

Control Outputs			
RxStatus	**Variables**	**Type**	**Default Value**
	kout_radio_status->		
	Distance	basic/float32	n/a
	Frequency	basic/uint64	n/a
	InBand	basic/float32	n/a
	Jammed	basic/boolean	n/a
	Noise	basic/float32	n/a
	PathFactor	basic/float32	n/a
	ReceiveState	basic/boolean	n/a
	RxPower	basic/float32	n/a
	SNR	basic/float32	n/a
	Sources	basic/uint16	n/a
	TransmitState	basic/boolean	n/a
	Description: Each respective RxStatus output variable will report its current state or value. The variables can be linked and sent to other components or out to a host computer via UDP packets. These fields are most commonly used to monitor the status of the radio by external means.		

7.9. Receiver

Version: 0-0-1

Description: The Receiver Object provides a generic, high level radio receiver simulation, which includes frequency tuning effects, AM or FM bands, signal strength variation due to range, sidetone, background noise, squelch, and AGC.

The Receiver object is the same as the Generic Radio component with all the fields relating to transmission removed. Please see the Generic Radio description for details of the Receiver controls.

7.10. Tacan_Rx

Version: 0-0-1

Description: The TACAN_Rx component is a receiver designed to receive the embedded tones produced from the TACAN_Tx component for TACAN navigational aid.

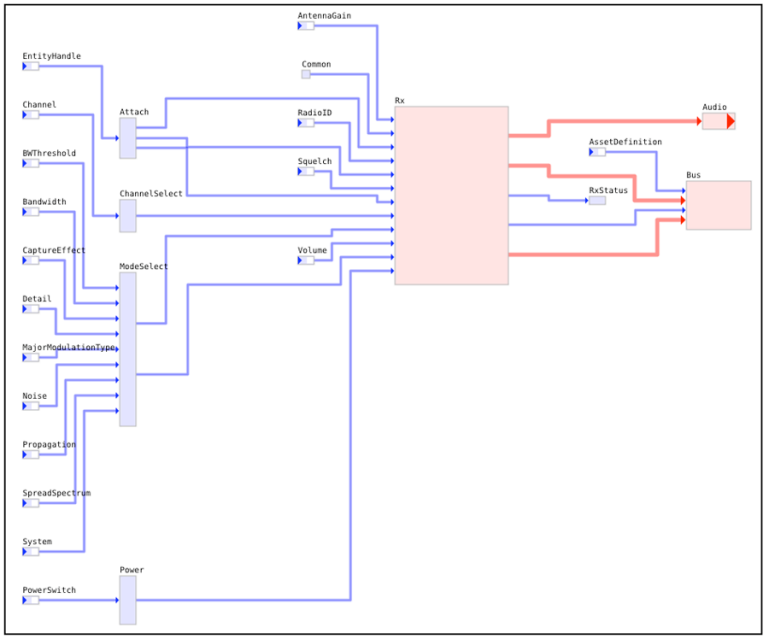


Figure 78: TACAN Rx Schematic

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: Adds a dB gain to the receiver. Units are in dBm.		
Asset Definition	**Variables**	**Type**	**Default Value**
	kin->	channel intercom/ ic_channel	0
	Description: Assigns a channel handle to the receiver. The channel handle allows connection between the receiver and a communications panel.		
Bandwidth	**Variables**	**Type**	**Default Value**
	kin	basic/float32	5000
	Description: Sets the bandwidth of the radio around its center frequency.		
BW Threshold	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.8
	Description: A percentage amount of the radio's bandwidth that must overlap with another radio's bandwidth in order for the radios to be intune.		
Capture Effect	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: When set to FALSE, all intune signals will be received and mixed together. When set to TRUE, only the intune radio with the strongest signal will be received.		

Control Inputs			
Channel	**Variables**	**Type**	**Default Value**
	kin	basic/uint64	0
	Description: This is used to set the Tacan channel number. Tacan channel frequencies are set in the ChannelSelect Internal primitive.		
Detail	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the detail part of the DIS modulation type code. See the Generic Radio component description for valid options and values. Default is set for Pulse Modulation.		
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the TACAN_Rx to an entity component. If the value here matches the handle field in an entity component, the TACAN_Rx will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/uint64	1000000
	Description: This is the receive frequency for the VOR receiver. Units are in Hertz.		
Major Modulation Type	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	5
	Description: Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set for Pulse Modulation.		

Control Inputs			
Noise	**Variables**	**Type**	**Default Value**
	kin	basic/float32	-100
	Description: Internal noise figure for the receiver. Units are in dBm.		
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: This is used to turn power on and off to the radio.		
Propagation	**Variables**	**Type**	**Default Value**
	kin->Ionospheric	basic/boolean	FALSE
	kin->Occulting	basic/boolean	TRUE
	kin->Range	basic/boolean	TRUE
	kin->Terrain	basic/boolean	FALSE
	Description: These are propagation flags. A TRUE value will use the respective propagation effect in calculating signal pathloss. A FALSE value will ignore the respective propagation effect in calculating pathloss.		
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: ID number of the TACAN_Rx that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the EntityHandle control. A value of 0 will default the radio ID to the object number of the component.		

Control Inputs			
Spread Spectrum	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: Flag used to enable/disable frequency hopping settings. Value must be set to 1 to use frequency hopping settings. 0 disables.		
Squelch	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.2
	Description: When the received RF signal/noise ratio is less than the squelch value given in this field, the AGC gain is set to zero, providing the normal background noise suppression. To disable the squelch, set the squelch level to zero.		
System	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set to Generic.		
Volume	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1
	Description: Sets the main volume of the receiver, scales the amplitude being received.		

Control Outputs			
RxStatus	**Variables**	**Type**	**Default Value**
	kout_radio_status->		
	Distance	basic/float32	n/a
	Frequency	basic/uint64	n/a
	InBand	basic/float32	n/a
	Jammed	basic/boolean	n/a
	Noise	basic/float32	n/a
	PathFactor	basic/float32	n/a
	ReceiveState	basic/boolean	n/a
	RxPower	basic/float32	n/a
	SNR	basic/float32	n/a
	Sources	basic/uint16	n/a
	TransmitState	basic/boolean	n/a
	Description: Each respective RxStatus output variable will report its current state or value. The variables can be linked and sent to other components or out to a host computer via UDP packets. These fields are most commonly used to monitor the status of the radio by external means.		

7.11. Tacan_Tx

Version: 0-0-1

Description: The Tacan Transmitter Beacon component is a transmitter with an embedded identifier tone element. It keys a four letter word on a user defined tone to produce a simulation of a Tacan beacon which can be heard over a suitably tuned radio or receiver.

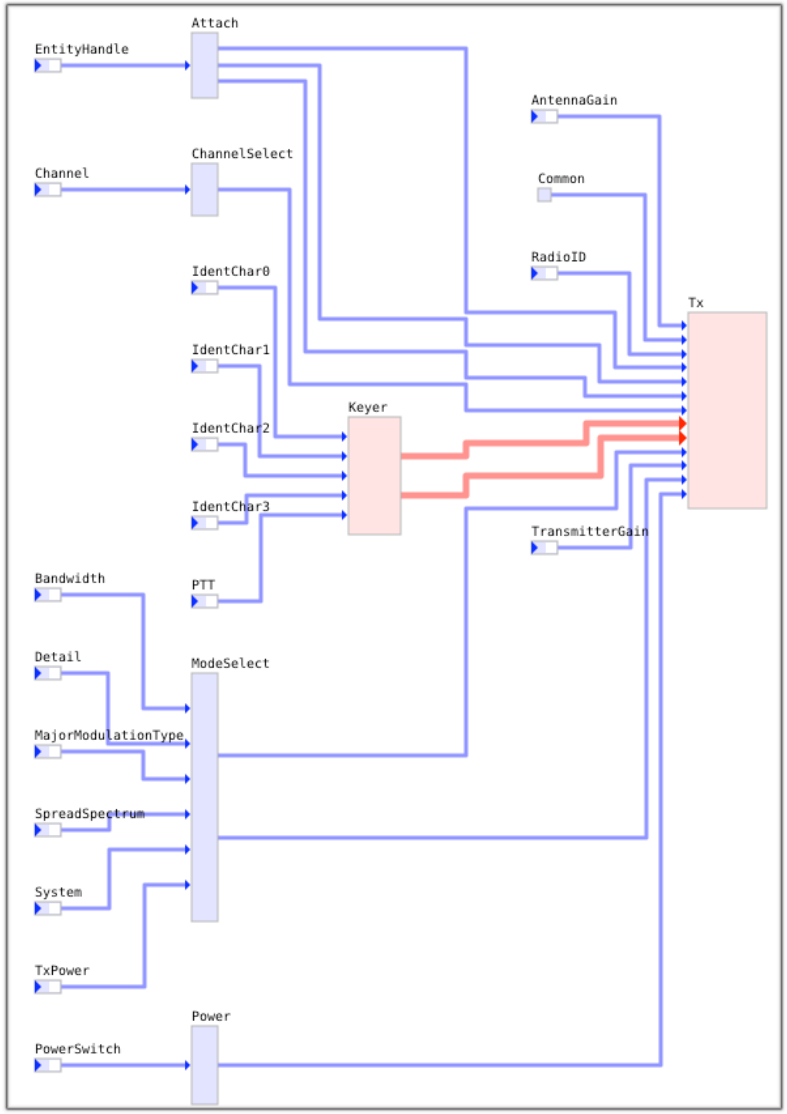


Figure 79: TACAN Tx Schematic

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: N/A for this component.		
Band-width	**Variables**	**Type**	**Default Value**
	kin	basic/float32	5000
	Description: Sets the bandwidth of the Tacan beacon around its center frequency. Units are in Hertz.		
Channel	**Variables**	**Type**	**Default Value**
	kin	basic/uint64	1
	Description: This is used to set the Tacan channel number. Tacan channel frequencies are set in the ChannelSelect Internal primitive.		
Controller	**Variables**	**Type**	**Default Value**
	Description: Does not work for this component. To set parameters, use the corresponding input controls or the Keyer Internal Primitive.		
Detail	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	3
	Description: Sets the detail part of the DIS modulation type code. See the Generic Radio component description for valid options and values. Default is set for Phase Modulation.		

Control Inputs			
Encoding	**Variables**	**Type**	**Default Value**
	kin	radio/encoding	default_encoding
	Description: Determines the audio encoding type valid options are CVSD_mil, MuLaw, PCM_16, Default encoding. DE selection will use the encoding type as set via RMS radios settings page.		
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the TACAN_Tx to an entity component. If the value here matches the handle field in an entity component, the TACAN_Tx will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	basic/uint64	108000000
	Description: Sets the carrier frequency for the Beacon.		
IdentChar 0	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The first letter of the keyed word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
IdentChar 1	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The second letter of the keyed word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		

Control Inputs			
IdentChar 2	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The third letter of the keyed word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
IdentChar 3	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: The fourth letter of the keyed word. Expected input is the ASCII decimal value for the letter. Lower and upper case values are both valid.		
Major Modulation Type	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	3
	Description: Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set for Pulse Modulation.		
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: Used to turn power on and off to the jammer.		
PTT	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: This is used to trigger the transmitter of the beacon. When value remains TRUE, the NDB beacon will continuously transmit its keyed tone.		

Control Inputs			
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: ID number of the TACAN_Tx that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the EntityHandle control. A value of 0 will default the radio ID to the object number of the component.		
Sample Rate	**Variables**	**Type**	**Default Value**
	kin	radio/samplerate	default_rate
	Description: This determines the audio sampling rate for the radio valid options include: sample_8k, sample_16k, sample_32k, sample_48k, default_rate. DR selection will use the audio sampling type as set via RMS radios settings page.		
Spread Spectrum	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: N/A for this component.		
System	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set to Generic.		

Control Inputs			
TransmitterGain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1.0
	Description: Scales the transmit power value input. Final transmitter power is the Transmit power multiplied by the Transmitter Gain.		
TxPower	**Variables**	**Type**	**Default Value**
	kin	basic/float32	15
	Description: Sets the Transmit power for the beacon.		

Control Outputs			
Busy	**Variables**	**Type**	**Default Value**
	kout	basic/boolean	FALSE
	Description: Outputs the keyed state of the Beacon. When keying, an output of TRUE is shown.		

Internal Parameters			
Channel Select	**Variables**	**Type**	**Default Value**
	Base_Frequency	basic/uint64	1025000000
	Channel_spacing	basic/uint32	1000000
	Top_channel	basic/uint32	126
	Description:		
Keyer	**Variables**	**Type**	**Default Value**
	Frequency	basic/float32	1020
	Interval	basic/uint16	5
	Wordrate	basic/uint8	8
	Description: **Frequency** - This is used to set the frequency of the tone. **Interval** - Sets the repeat rate of the Morse word. Units are in seconds. **Wordrate** - Determines the rate at which the word is keyed.		

7.12. VOR_RX

Version: 0-0-1

Description: VOR_Rx component is a receiver designed to receive the embedded tones produced from the VOR_Tx component for VOR navigational aid.

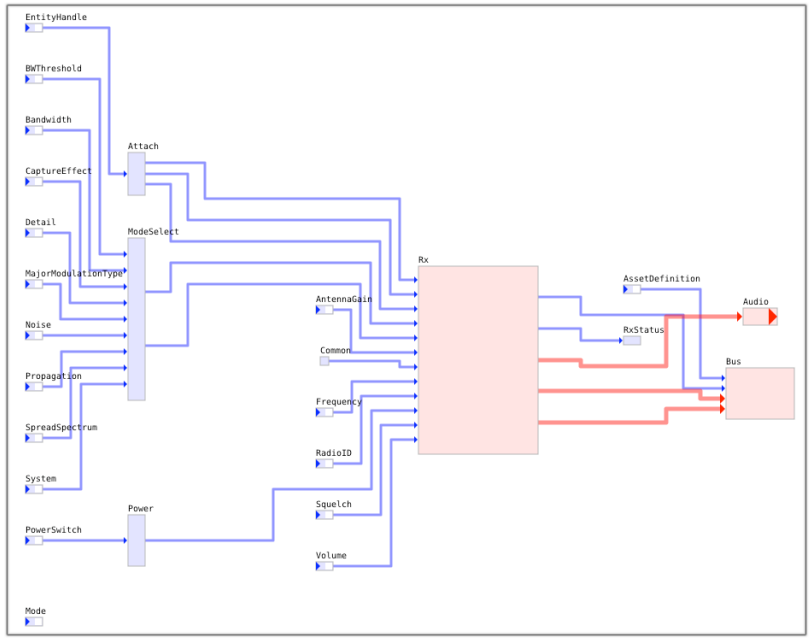


Figure 80: VOR_RX Schematic

Control Inputs			
Antenna Gain	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0
	Description: This adds a dB gain to the receiver. Units are in dBm.		
Asset Definition	**Variables**	**Type**	**Default Value**
	kin->	Channel Intercom/ ic_channel	0
	Description: Used to assign a channel handle to the VOR_Rx in order to connect it to an intercom component via the local intercom bus.		
Bandwidth	**Variables**	**Type**	**Default Value**
	kin	basic/float32	5000
	Description: Sets the bandwidth of the radio around its center frequency.		
BW Threshold	**Variables**	**Type**	**Default Value**
	kin	**basic/float32**	**0.8**
	Description: A percentage amount of the radio's bandwidth that must overlap with another radio's bandwidth in order for the radios to be intune.		
Capture Effect	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	FALSE
	Description: When set to FALSE, all intune signals will be received and mixed together. When set to TRUE, only the intune radio with the strongest signal will be received.		

Control Inputs			
Detail	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	3
	Description: Sets the detail part of the DIS modulation type code. See the Generic Radio component description for valid options and values. Default is set for Continuous Wave Modulation.		
Entity Handle	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: The EntityHandle is used to attach the VOR_Rx to an entity component. If the value here matches the handle field in an entity component, the VOR_Rx will adopt the World Position, DIS Site-Host-Entity ID fields, and the Network type contained in the entity component.		
Frequency	**Variables**	**Type**	**Default Value**
	kin	uint64	1000000
	Description: This is the receive frequency for the VOR receiver. Units are in Hertz.		
Major Modulation Type	**Variables**	**Type**	**Default Value**
	kin	basic/float32	-100
	Description: Sets the modulation type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set for Amplitude Modulation.		
Noise	**Variables**	**Type**	**Default Value**
	kin	basic/uint8	0
	Description: Internal noise figure for the receiver. Units are in dBm.		

Control Inputs			
Power Switch	**Variables**	**Type**	**Default Value**
	kin	basic/boolean	TRUE
	Description: This is used to turn power on and off to the radio.		
Propagation	**Variables**	**Type**	**Default Value**
	kin->Ionospheric	basic/boolean	FALSE
	kin->Occulting	basic/boolean	TRUE
	kin->Range	basic/boolean	TRUE
	kin->Terrain	basic/boolean	FALSE
	Description: These are propagation flags. A TRUE value will use the respective propagation effect in calculating signal pathloss. A FALSE value will ignore the respective propagation effect in calculating pathloss.		
RadioID	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: ID number of the VOR_Rx that must be assigned to use DIS, HLA or Local Net. The host number, site number, entity number and Radio ID of the DIS ID must form a unique set. All numbers but the radio ID are set through the EntityHandle control. A value of 0 will default the radio ID to the object number of the component.		
Spread Spectrum	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	0
	Description: Flag used to enable/disable frequency hopping settings. Value must be set to 1 to use frequency hopping settings. 0 disables.		

Control Inputs			
Squelch	**Variables**	**Type**	**Default Value**
	kin	basic/float32	0.2
	Description: When the received RF signal/noise ratio is less than the squelch value given in this field, the AGC gain is set to zero, providing the normal background noise suppression. To disable the squelch, set the squelch level to zero.		
System	**Variables**	**Type**	**Default Value**
	kin	basic/uint16	1
	Description: Sets the system type as per DIS standards. See the Generic Radio component description for valid options and values. Default is set to Generic.		
Volume	**Variables**	**Type**	**Default Value**
	kin	basic/float32	1
	Description: Sets the main volume of the receiver, scales the amplitude being received.		

Control Outputs			
RxStatus	**Variables**	**Type**	**Default Value**
	kout_radio_status->		
	Distance	basic/float32	n/a
	Frequency	basic/uint64	n/a
	InBand	basic/float32	n/a
	Jammed	basic/boolean	n/a
	Noise	basic/float32	n/a
	PathFactor	basic/float32	n/a
	ReceiveState	basic/boolean	n/a
	RxPower	basic/float32	n/a
	SNR	basic/float32	n/a
	Sources	basic/uint16	n/a
	TransmitState	basic/boolean	n/a
	Description: Each respective RxStatus output variable will report its current state or value. The variables can be linked and sent to other components or out to a host computer via UDP packets. These fields are most commonly used to monitor the status of the radio by external means.		