Document No. XE-00007
1st Feb 2020

Solaro Universal Controller User Guide

Contents

1. Overview

This document describes the new Solaro Universal Controller feature that is bundled inside each Solaro DSP device. This makes the deployment of control solution very cost effective, as all you need is a Solaro DSP device, and it provide both the DSP signal processing as well as controller function to control all equipment with your site.

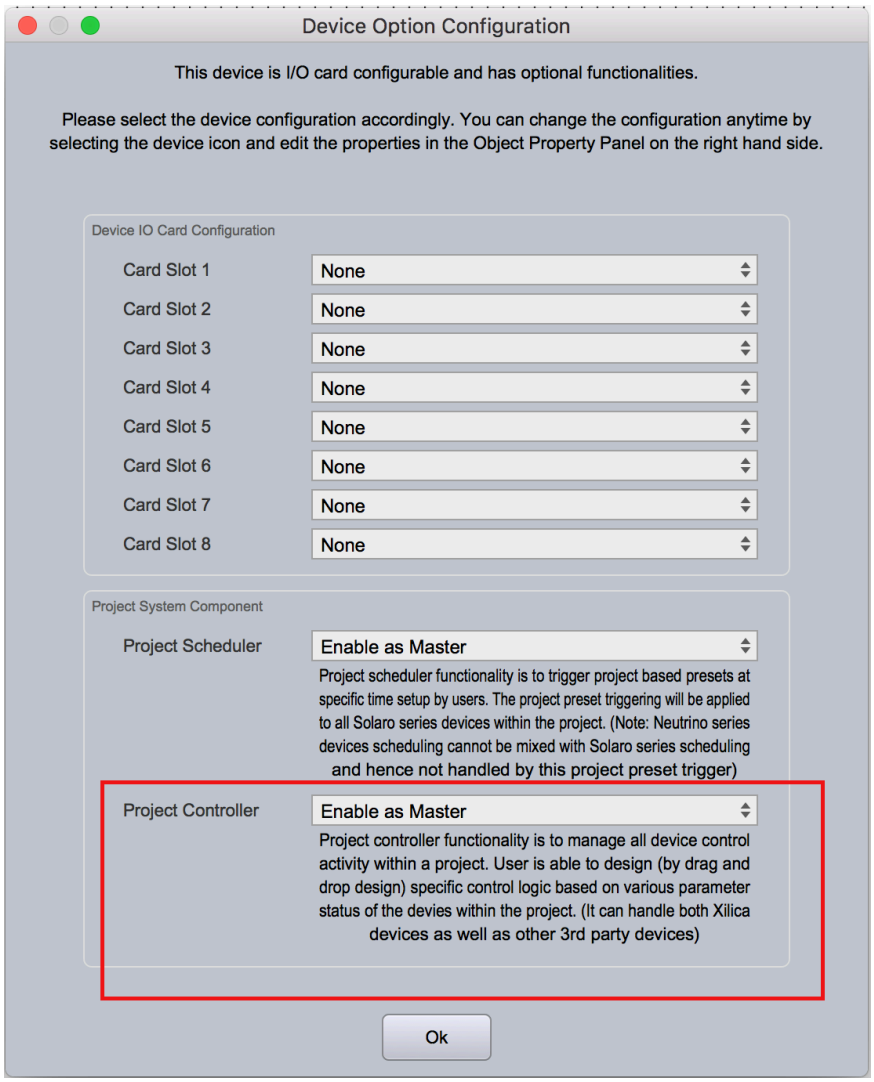
Besides no special H/W cost, another aim of Solaro Universal Controller, is to provide an extremely easy to use configuration interface to program all the site-specific logic using a drag and drop interface. The interface and design concept is very similar to DSP programming. For sound engineer who is familiar with DSP system design, they will find the same design concept for programming the controller functionality. No programming experience is require to get the controller up and running for your site.

For simple site deployment, it can be as simple as drag in the control elements from different devices (from different manufacturers) that you want to control. And then connect up the specific control flow by wiring different control elements accordingly. For more complicated protocol control case, we provide numeric and binary data manipulation modules, that you can create your own control string and send to different devices through ethernet.

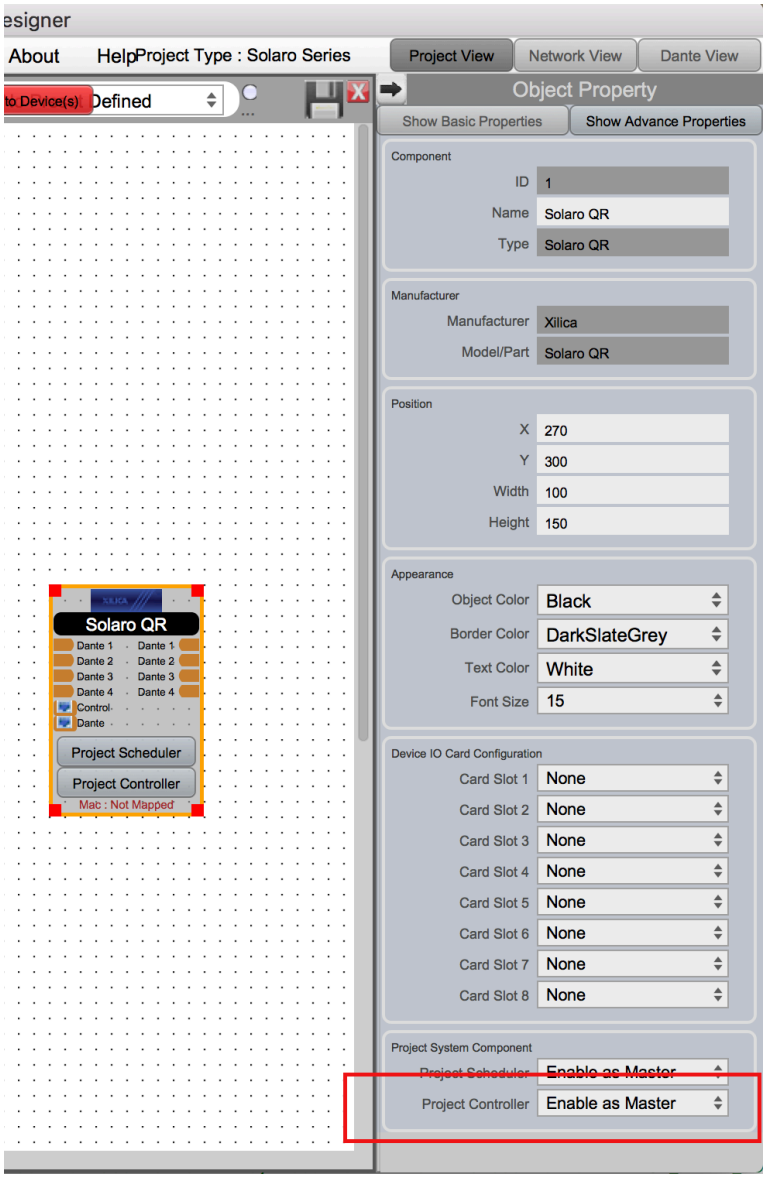
Since Version 4.0.0, we have introduced a new scripting language support to make the controller more powerful. Basically you can write your own processing based on Lua scripting language.

2. Setting up Controller functionality for a Solaro DSP device.

Each Solaro DSP device will come bundle with controller functionality in the software. To use the controller functionality, all you need to do is to enable the project controller in your Solaro DSP device as master for your project. When you drag in a Solaro device, it will have a dialog for you to setup your card slot configuration. At the bottom there is an option to set the Solaro DSP device as project controller master. Then your Solaro DSP device icon will have a Project controller button displayed.

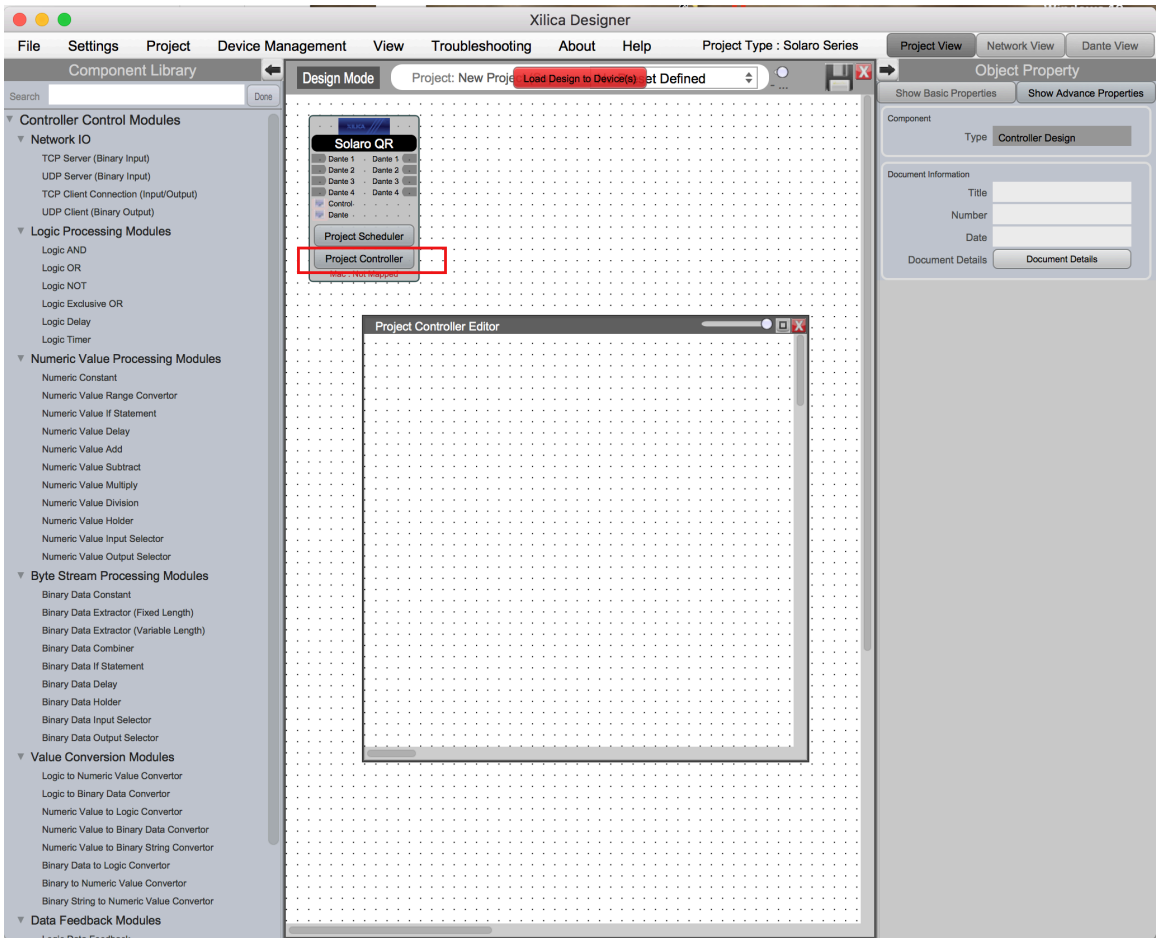


Or you can pick one of the Solaro DSP device in your project as Project Controller Master. You can switch between device at any time during design state. Just click on the device icon and select the Project Controller option at Object Property area.



To edit the controller design, you can click on the “Project Controller” button in the device icon. Then the Project Controller editor will be displayed. And then you can drag and drop in the control element and controller processing modules into your design. Once a design is made, it is being stored at the project level. If you switch the Controller master to another DSP device the project design will remain the same.

Only one Solaro DSP device can be controller master for a project. Xilica Designer software will provide protection to ensure only one device is controller master.


Xilica Designer
File
Settings
Project
Device Management
View
Troubleshooting
About
Help
Project Type : Solaro Series
Project View
Network View
Dante View
Component Library
Search
Done
Controller Control Modules
Network IO
TCP Server (Binary Input)
UDP Server (Binary Input)
TCP Client Connection (Input/Output)
UDP Client (Binary Output)
Logic Processing Modules
Logic AND
Logic OR
Logic NOT
Logic Exclusive OR
Logic Delay
Logic Timer
Numeric Value Processing Modules
Numeric Constant
Numeric Value Range Convertor
Numeric Value If Statement
Numeric Value Delay
Numeric Value Add
Numeric Value Subtract
Numeric Value Multiply
Numeric Value Division
Numeric Value Holder
Numeric Value Input Selector
Numeric Value Output Selector
Byte Stream Processing Modules
Binary Data Constant
Binary Data Extractor (Fixed Length)
Binary Data Extractor (Variable Length)
Binary Data Combiner
Binary Data If Statement
Binary Data Delay
Binary Data Holder
Binary Data Input Selector
Binary Data Output Selector
Value Conversion Modules
Logic to Numeric Value Convertor
Logic to Binary Data Convertor
Numeric Value to Logic Convertor
Numeric Value to Binary Data Convertor
Numeric Value to Binary String Convertor
Binary Data to Logic Convertor
Binary to Numeric Value Convertor
Binary String to Numeric Value Convertor
Data Feedback Modules
Design Mode
Load Design to Device(s)
Solaro QR
Project Scheduler
Project Controller
Project Controller Editor
Object Property
Show Basic Properties
Show Advance Properties
Component
Type
Controller Design
Document Information
Title
Number
Date
Document Details
Document Details

3. Basic data types

The basic operation of the controller is based on processing data retrieved from devices on the network (i.e parameter values such as mute state of a channel), then pass this data to different processing modules for processing. After processing data can then be send to remote devise on network to control the device.

The data retrieved or send to remote devices can be classified into 3 different types.

- Logic Data – Can only be either 0 or 1 (false or true)
    - Example of this data type will be mute status of a channel from a device.
- Numeric Data – Can either be integer value or number with decimal places.
    - Example of this data type will be the volume fader value from a device.
- Binary Data – Series of byte data. String data is also of Binary Data type.
    - Example of this data type can be the control string captured from a device on the network.

Each processing module will have different input and output pins associated with it. And each pin will be of one of the 3 basic data types. To convert from one data type to another data type, a special data conversion module is required to properly convert data to different types.

4. Different types of modules

Processing modules can be grouped into the following types.

a. IO modules
Handle Input and Output of device parameters (Xilica Devices or any universal control devices defined in Xilica Designer).

b. Network IO modules
Handle raw network-based IO binary message processing. These modules is required if you need to handle control for devices that are not defined in Xilica universal control device list in Xilica Designer.

c. Logic processing modules
Handle logic data processing such as logic and etc..

d. Numeric value processing modules
Handle numeric value processing such as value from a volume fader.

e. Binary data processing modules
Handle binary data processing such as extracting specific byte from a byte stream.

f. Data conversion modules
Handle data conversion between different data types.

g. Data Feedback modules
In general processing modules output value cannot be feedback to a module which process before this module. To support such looping operation, these data feedback modules are required.

h. UI modules
Provide UI objects so that user can interact with these UI object to trigger different processing. (Note: In the current Beta version the UI object are very primitive, we will further enhance these modules in production release).

i. Lua Scripting module
Provide Lua scripting capability for user to design their own processing. You can pass in Logic, Numeric, Binary data into the module for processing. And result in Logic, Numeric or Binary form can be send from script module to next processing module.

5. Processing flow and data Validity

Processing Flow

Controller processing flow is very similar to the DSP processing flow. You drag in different processing modules and connecting up the data flow wire between modules to perform your control tasks.

Once a design is done, controller will perform execution every 100ms to retrieve input data (such as device data from remote 3rd party devices). And starting from input module, it will work its way along the processing flow and process the data accordingly. When the processing data gets to a remote-control point (such as a IO modules or Network IO modules), it will send out the data through ethernet to control remote devices.

In general, the processing flow is one way flow, but we have introduced some special processing module called Data Feedback module which support functionality to store the result data in previous execution cycle and pass as input data for next processing cycle. This will provide a feedback mechanism to perform response data to last processing cycle. This is extremely useful to handle network traffic. If you received certain network binary data, you want to process it and then pass it back to the caller. This callback mechanism is essential to do that.

Data Validity and Data Update Mechanism

During processing flow, if a data value has not been acquired yet. (i.e. a remote device parameter has not been read back, due to device is OFFLINE). Then the data point will be marked as invalid. For module to process the flow, data must be valid on the input side. If input in invalid, its corresponding output will also be marked as invalid. (Note: this invalid state will only happen during initial startup. Once it has been validated, it will remain as valid).

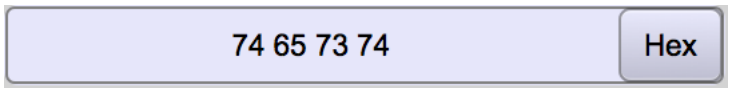
Some module has independent channel processing (i.e. a Logic NOT module). Invalid input data in Channel 1 will only affect the output of Channel 1. While other channels that has valid input state will be processed accordingly.

Besides data value validity, each data point will also has an updated flag associated with it. If an input value has not been updated, the processing module will also skip its corresponding output value processing. This mechanism will avoid duplicate execution of processing flow (Especially in the case of send network control message to remote device.).

6. Binary Data entry and display

For binary data field, we allow user to enter the data in Hex format or in ASCII format. For the data field, there is a button at the right hand side to indicate whether the data is being interpret or displayed as “Hex” or “ASCII”.

For Hex data, you just enter the Hex byte value directly (0-9 and a-f). It will interpret the value as Hex.



You can switch between Hex and ASCII display as long as the data is within the ASCII range. If not, a warning message will be displayed. You might need to removed the non-ASCII data before you can switch to ASCII mode.

7. Working with a Touch Panel

After you make up a design, you may want to have a touch panel to display certain status or you want use the touch panel to change certain button status within the controller design. To do that, you need XTouch50, XTouch80, or any device with XTouchApp installed.

To add a control to the panel, you Ctl-click on the control element in your design and drag it over to the XTouch design. (This is exactly the same mechanism to do your XTouch design on the DSP side.)

Once you have made up a XTouch design, you can directly map the design to a physical XTouch device. Or you can associate the XTouch design to the Solaro DSP device. This way, the design is being stored in the Solaro DSP device. Later on, if you have an XTouch device in the network you can query the Solaro DSP for control panel design. And obtain the design from the DSP device directly (without using Xilica Designer software). This is a very convenient way to setup control panel for Ad-Hoc control device such as a mobile phone.

8. Debugging a design

When we design this controller, we know one very critical usability issue is how to debug your design. Our aim is to have the debugging capability available to user conveniently during runtime.

In each of the module control page, we display the real time data value. With these real time display, you can trace your data flow and processing result to ensure it is working the way you expected.

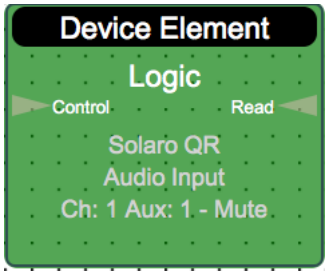
And for the parameters in each module, you can change the value on the fly and the processing module will re-process according to the new parameter value (without requiring to reload the design from Xilica Designer).

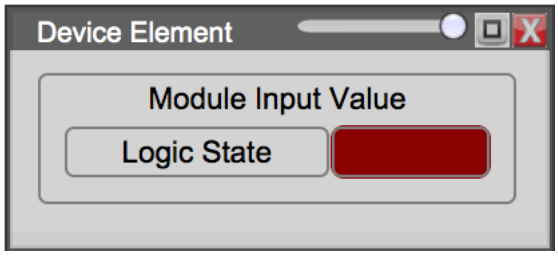
9. IO modules

These IO modules are not being listed on the Controller Control Modules List on the left-hand side. As these modules are the representation of some physical parameters on device. To create these I/O modules, simple drag and drop the control elements from the device control panel onto controller editor. (i.e. a Mute button from Solaro DSP Input module. Or any Universal Control Device control panel)

There are 2 types of IO modules. Logic type or Numeric type. Once the control element has been dropped to the controller editor, the following module icon will be displayed.

Device Element – Logic



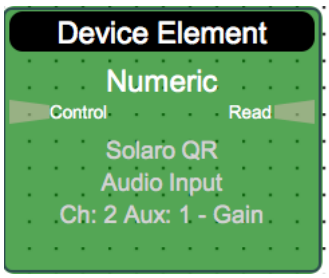


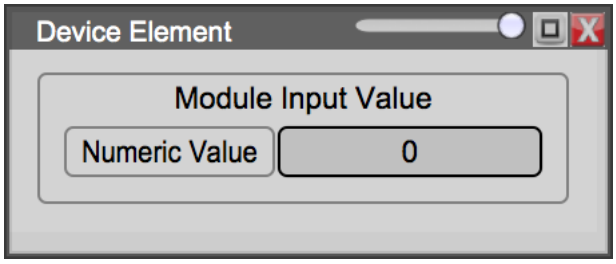
In this module, it has one logic input pin and one logic output pin. Within this icon, it will show which device and what source this control element is from.

If you want to control this control element of the remote device, you can simply feed a logic input value to this module. If you feed in a ON/OFF value, this module will perform all the necessary network protocol handling and control the remote data point to the specific value in input pin.

On the other hand, if the remote value has been changed from remote device. this output pin of this module will reflect the new data value read back from the network.

Device Element – Numeric





In this module, it has one numeric input pin and one numeric output pin. Within this icon, it will show which device and what source this control element is from.

If you want to control this control element of the remote device, you can simply feed a numeric value to this module. This module will perform all the necessary network protocol handling and control the remote data point to the specific value in input pin.

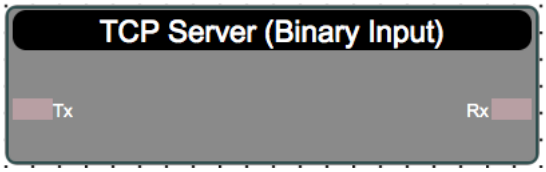
On the other hand, if the remote value has been changed from remote device. this numeric output pin of this module will reflect the new data value read back from the network.

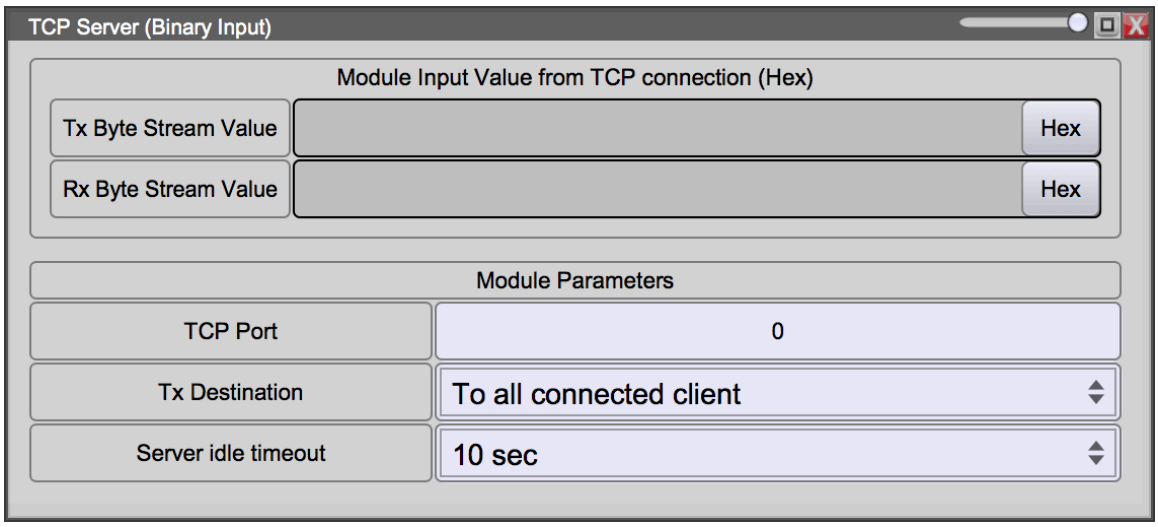
Note:

1) If you are only interested on controlling the remote device parameter, you only need to connect the input pin of these module. If the output pin is not connected, the read back data will be ignored by the system. On the other hand, if you are only interested on the current status of remote parameter, you only need to connect the output pin and pass the read back value for your processing. You can just leave the input pin unconnected. This way, the system will never send control commands to the remote data point.
2) You can drag in the same control element more than once into the controller editor. Multiple modules will same remote device parameter will work independently. But at the end, they will all controlling to or reading data from the same remote device parameter.

10. Network IO modules

TCP Server





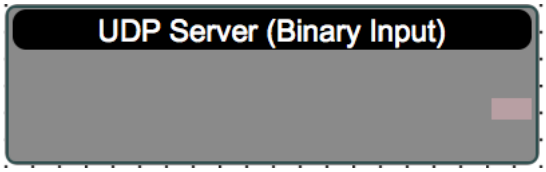
TCP service module provide data send and receive capability. It has a binary input pin for data that need to be send out to connected client. It also has a binary output pin for all binary data received on this server.

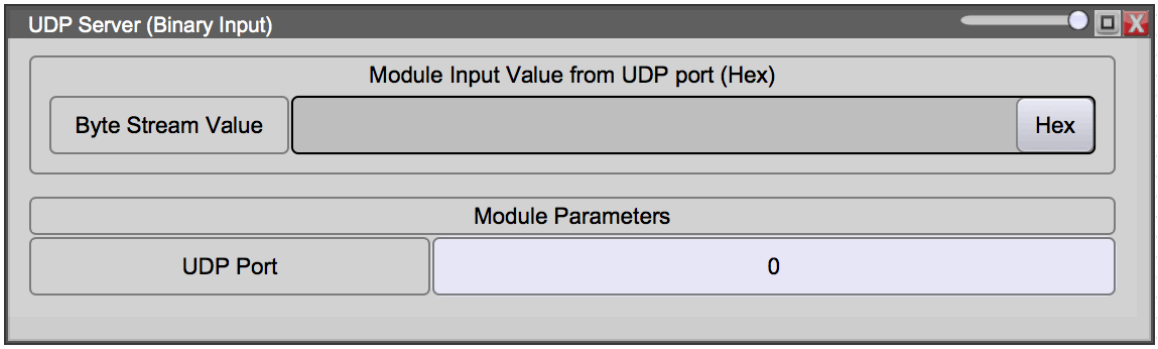
To setup the server, you need to enter the port (And the IP address for this server will be the IP address of the Solaro DSP device). You can setup the server to terminal client connection if the connection is idle. Or you can keep the connection once it is connected.

When sending Tx data back to client, there are 2 options, you can either send data to the last client that has just send data to this server. Or you can send the Tx data to all connected clients.

Data received will be based on the TCP package framing. Each TCP framing data will be treated as a single binary stream and will trigger the output of this server module once.

UDP Server

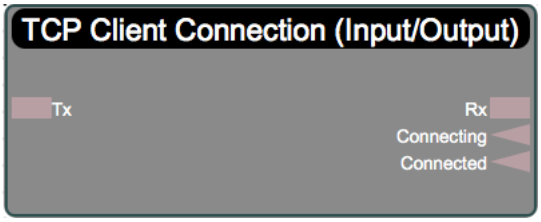


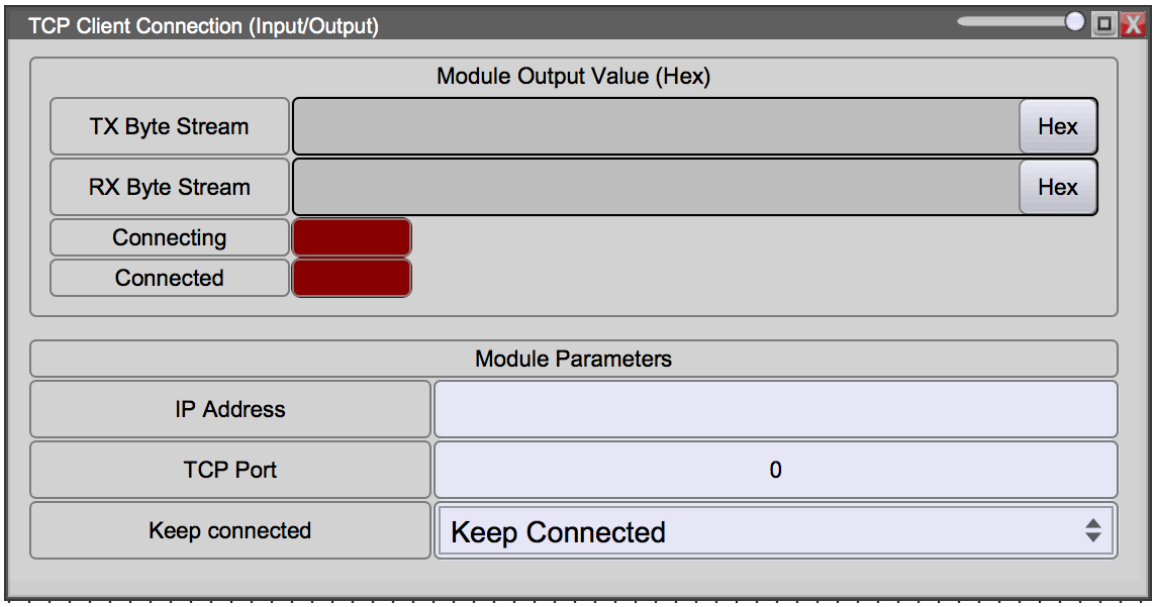


UDP server module only has a Rx output pin. Once a UDP packet has been received for this server port, it will trigger the output of this server module.

To setup this UDP server module, you need to enter the UDP port for this server. The IP address of this server will be the same as the IP address of Solaro DSP device.

TCP Client





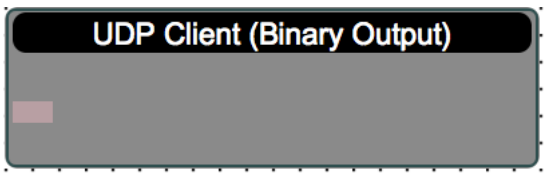
TCP client module has one Tx input pin which receives binary data that will send out to remote TCP server. This module also has a Tx output pin, which reflect the data that is send back from the server end to this client.

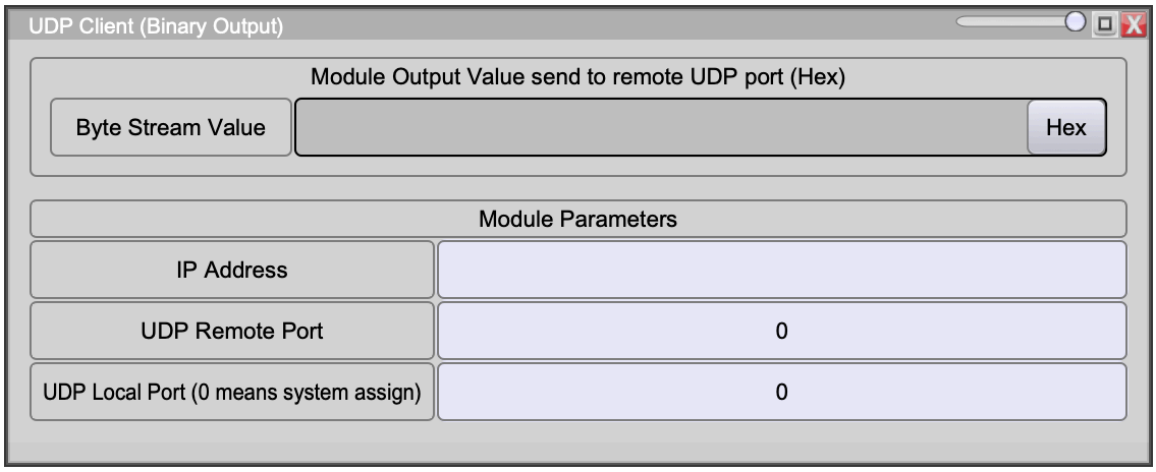
There are 2 more logic output pin which indicate whether the TCP client has estabilished a TCP connection. It also indicate whether the sending binary data has created a new TCP connection (If this creates a new TCP connection, the connecting pin will be true).

To setup this module, you need to enter the IP address of the remote TCP server that this module will connects to. You also need to setup the TCP port that this client need to conenct to.

You also have option to have the TCP client always connected or it can disconnect when idle.

UDP Client

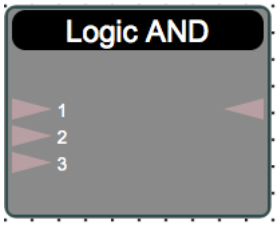


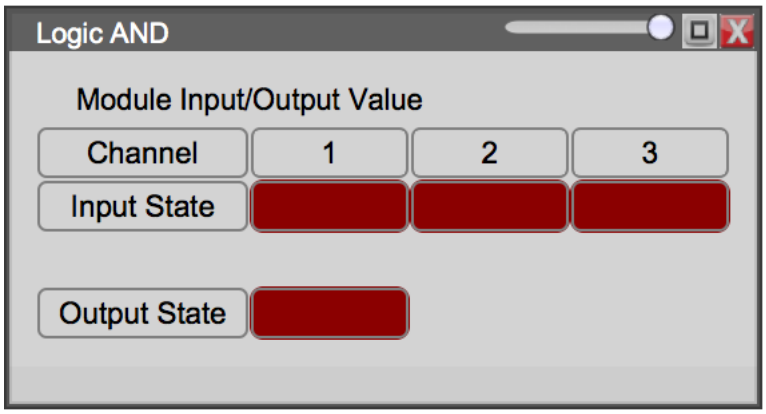


For UDP client module, it has one Tx input pin which takes the binary data and send to remote UDP server. To setup the module, you need to enter the remote UDP server IP address and port number. Besides the remote port that you can want send message to, you can also setup the local port number that you use in the outgoing UDP message. (As some device will send UDP message reply based on the UDP message local port).

11. Logic processing modules

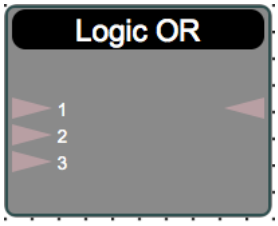
Logic AND

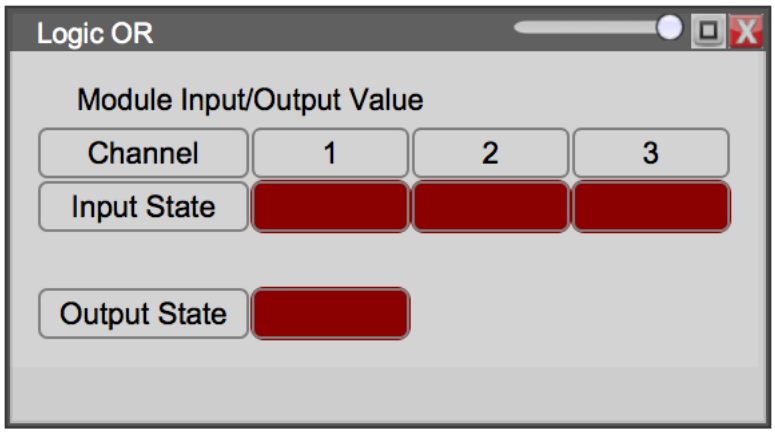




Logic AND module perform logical AND operation for all its input pin. And send the result to the output pin. The output pin will only be valid when all its input pin has valid value. There is NO parameters setting associated with this module.

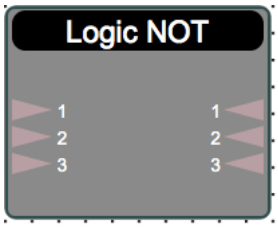
Logic OR

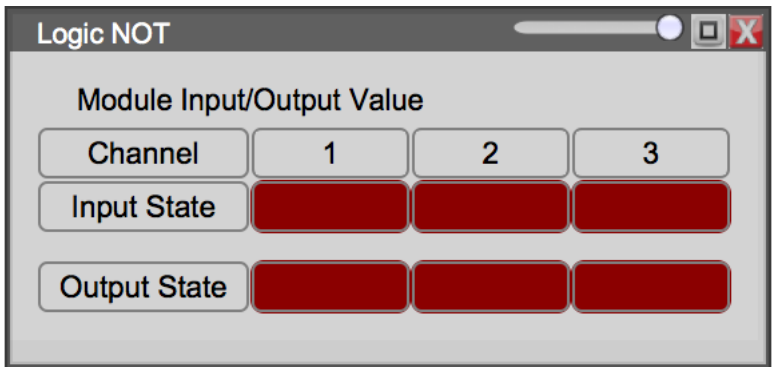




Logic OR module perform logical OR operation for all its input pin. And send the result to the output pin. The output pin will only be valid when all its input pin has valid value. There is NO parameters setting associated with this module.

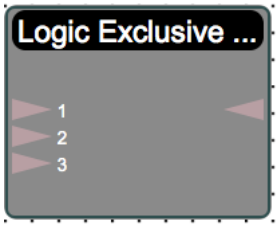
Logic NOT

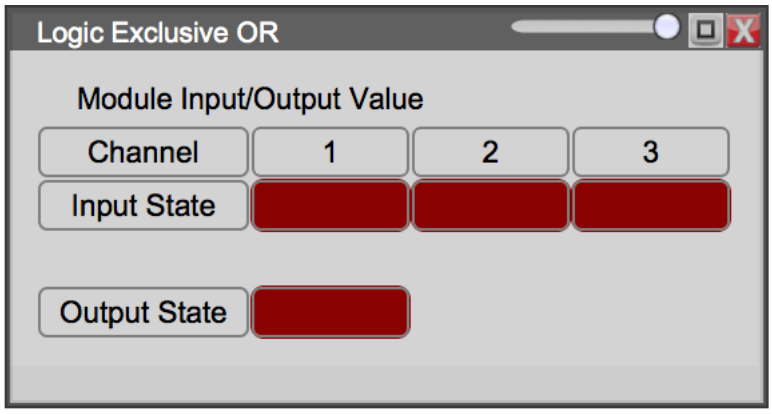




Logic NOT module perform logical NOT operation to each individual channel independently. Channel n output will be valid when the channel n input state become valid.

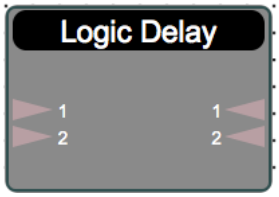
Logic Exclusive OR

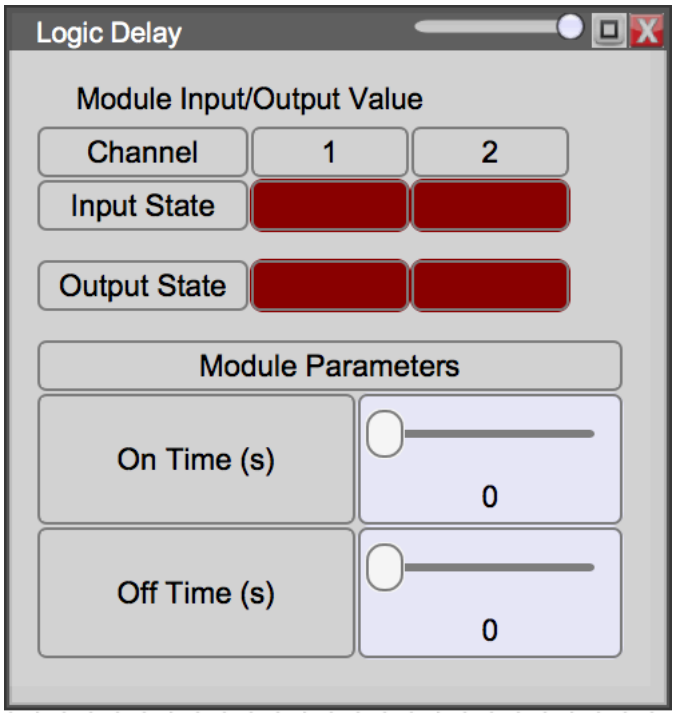




Logic Exclusive OR module is similar to Logic OR module, except it performs exclusive OR operation.

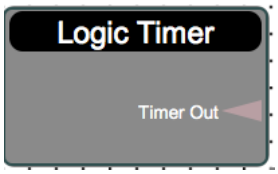
Logic Delay

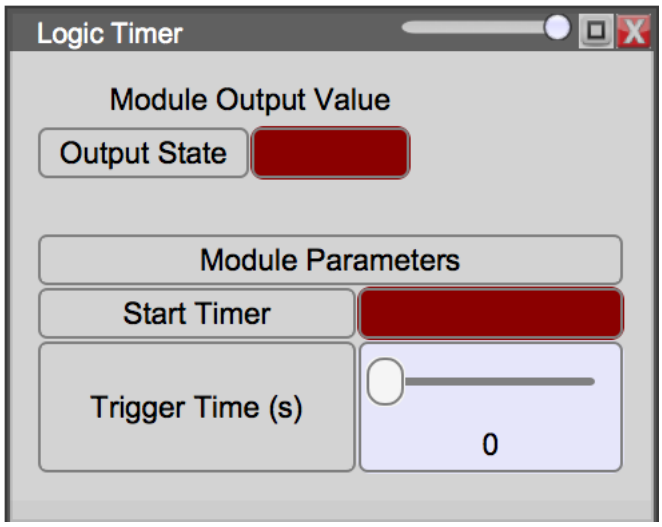




Logic Delay module will pass the input pin state to the output pin state after certain seconds of delay. The delay setting can be different for changing to ON or changing to OFF state. Each channel operation is independent to the other channels.

Logic Timer

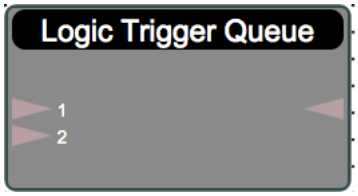


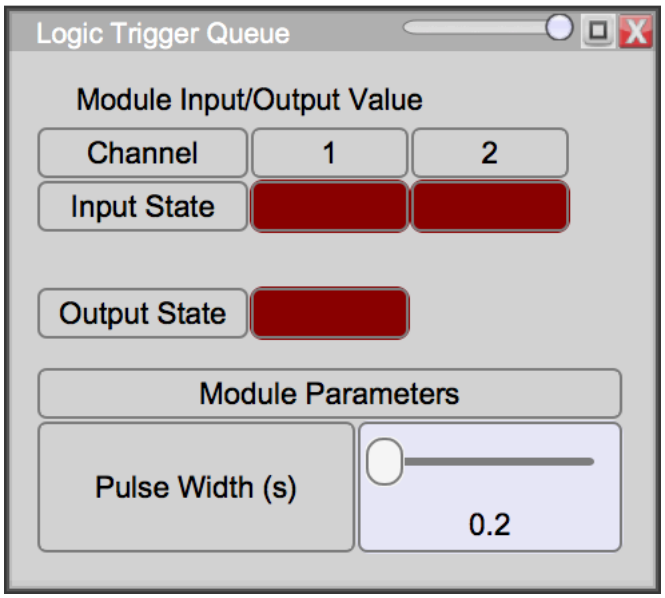


Logic Timer module will provide a periodic status change pulse (Go to logic ON momentary and then back to OFF state). based on the periodic trigger time. If the Start Time state is OFF, the periodic trigger pulse will be stopped.

This module is useful when you have a periodic operation that you need to perform. You can use this timer to trigger periodic actions.

Logic Trigger Queue

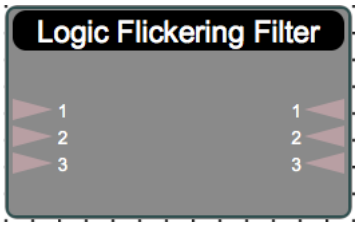


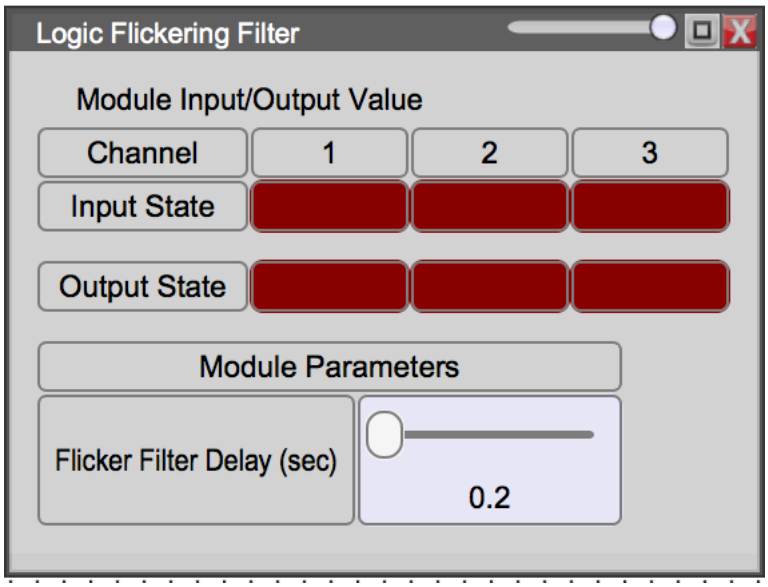


Logic Trigger Queue module provide a mechanism to detect rising edge from input signal and count the number of rising edge events from all input channels. It will then queue all events into a sequential edge trigger pulse signal to the output side. The result pulse width can be adjusted accordingly.

This is useful if you want multiple parts of a signal flow to merge and trigger processing actions in following path.

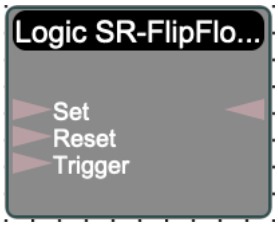
Logic Flickering Filter

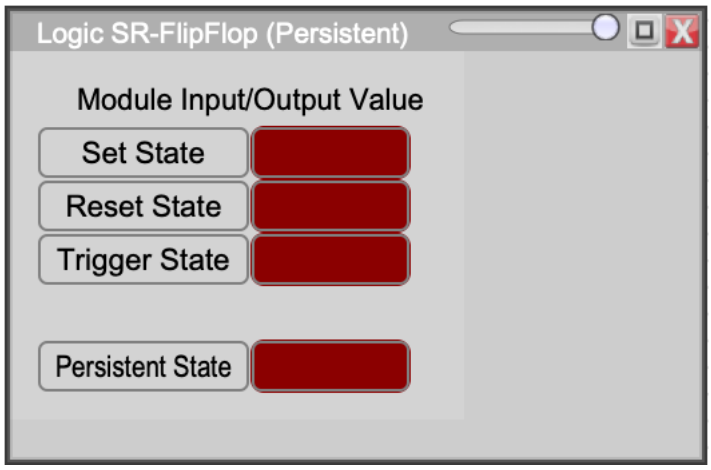




This module can be used to filter out unnecessary state change triggering due to logic bouncing (or logic contact bouncing). If a signal change from OFF state to ON state, we will ensure that the signal stay ON for the filter delay before we consider this as a real state change. If the state change did not last for the filter delay time, the change will be ignored. Similarly, if state change from ON to OFF, same logic applies.

Logic SR-FlipFlop (Persistent)

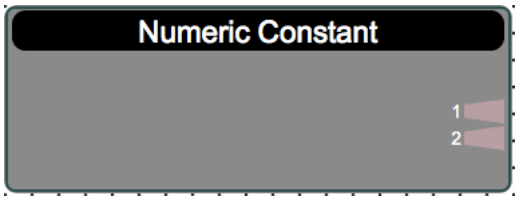


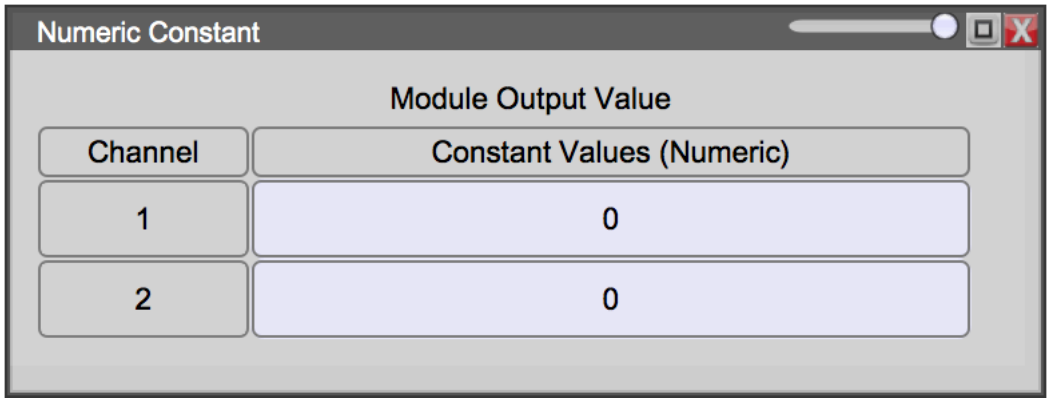


This Logic FlipFlop module provide a mechanism to toggle the logic state by providing "Trigger" signal to the flipflop. This module has "Set", "Reset" pin to directly modify the state of the flipflop.

12. Numeric Value processing modules

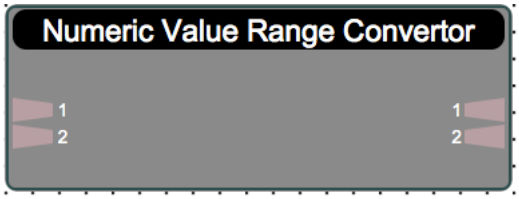
Numeric Constant

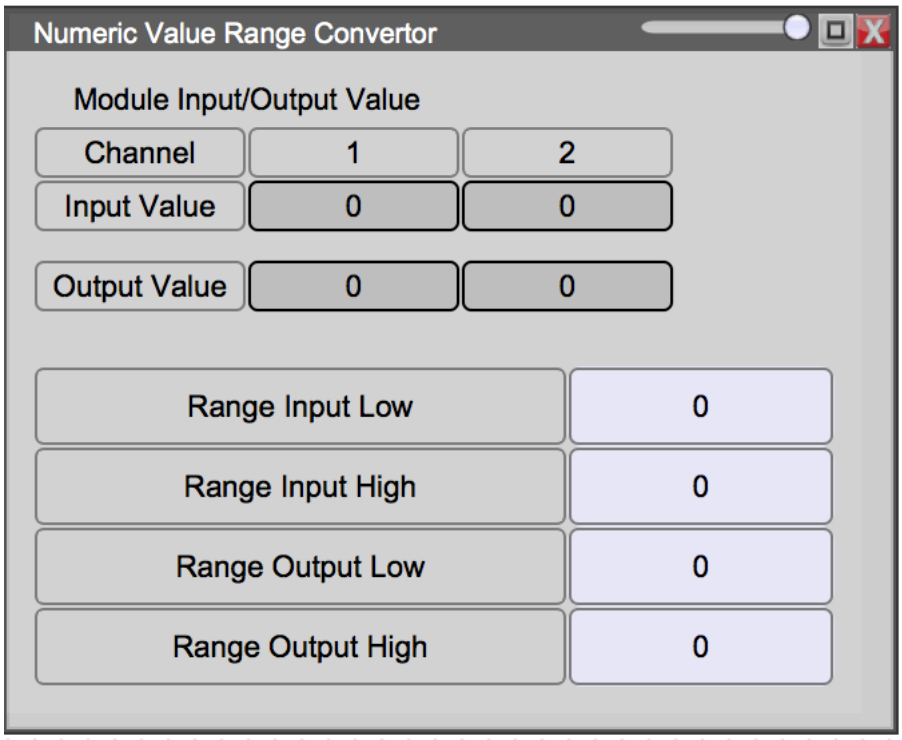




Numeric Constant module provide a place to supply a constant value to your processing design. These values can either be integer or decimal numbers.

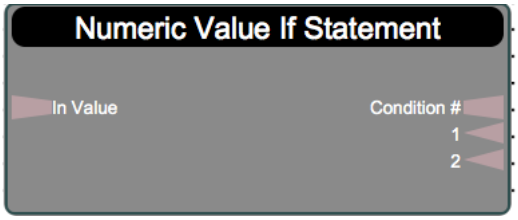
Numeric Value Range Convertor

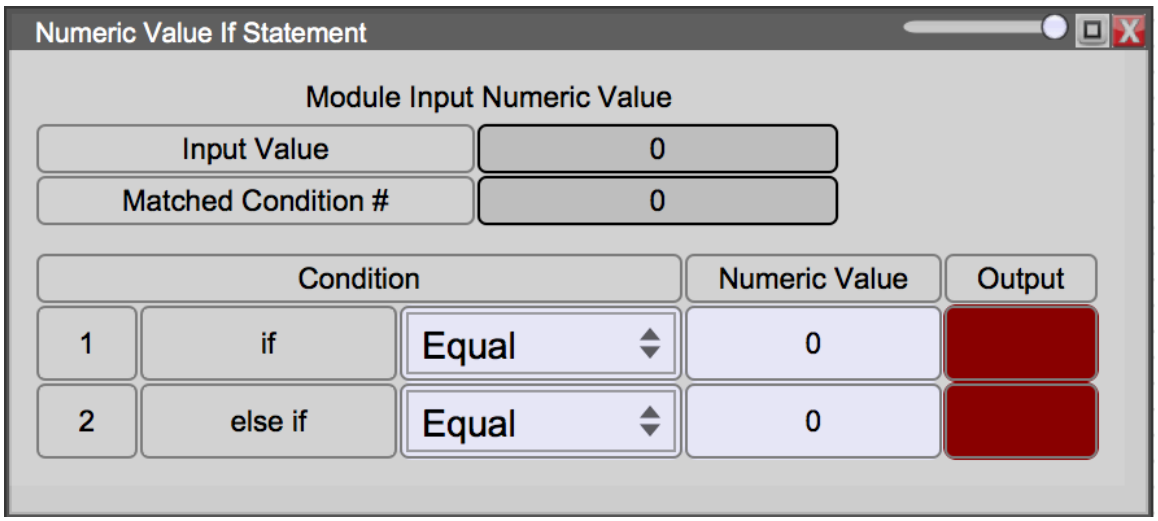




This module will perform value range re-map operation. It will take the input value of a channel and covert it to a percentage based on the input range. Then expend this percentage value based on the output range setting. Each channel of this module is independently to each other.

Numeric Value IF Statement



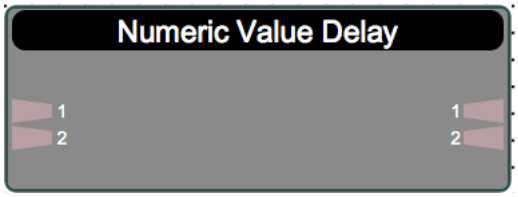


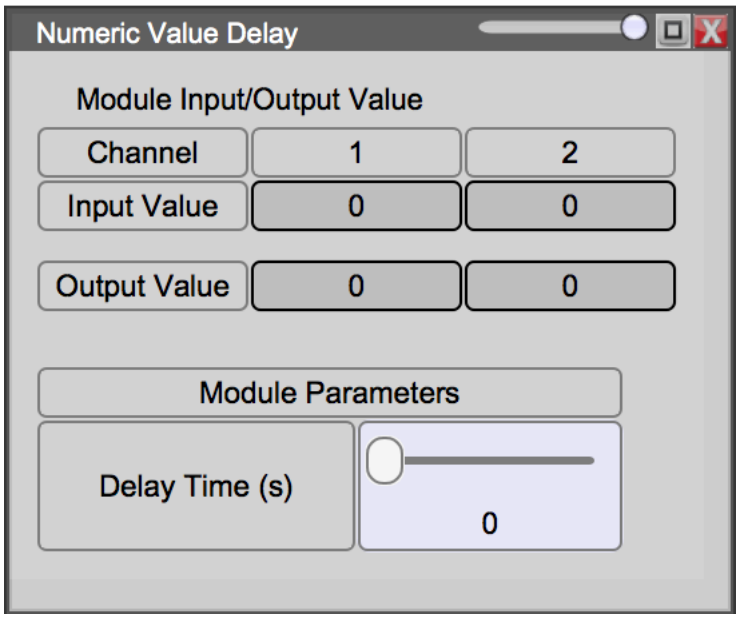
Numeric Value If Statement module provide a test condition on the input value. This module will perform the condition testing following the channel order. As soon as it found a matching condition it will set the logic output of that channel to be ON. It will also update the matched condition # to the output pin.

For the condition test, we support Equal, NOT Equal, Greater Than, Greater Than or Equal, Less Than, Less Than or Equal etc.

If all condition does not match the input value, all output state will become OFF and the Matched Condition # will become “0”.

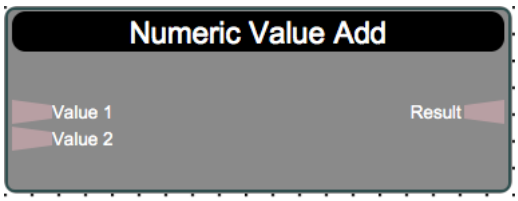
Numeric Value Delay

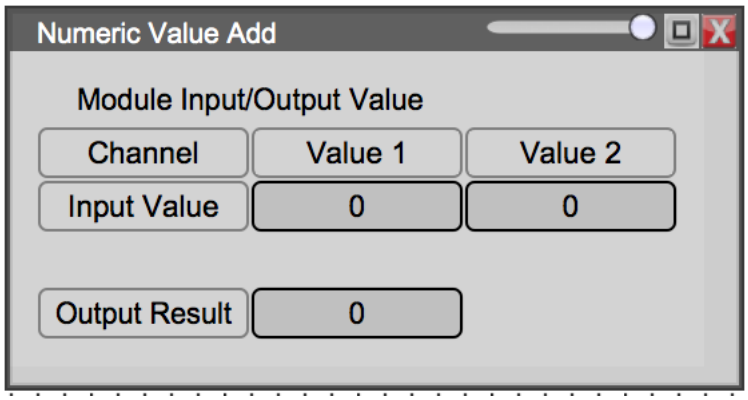




Numeric Value Delay module is similar to Logic Delay module, except it works on numeric values. It will pass the input numeric value to the output pin after certain seconds of delay. Each channel operation is independent to the other channels.

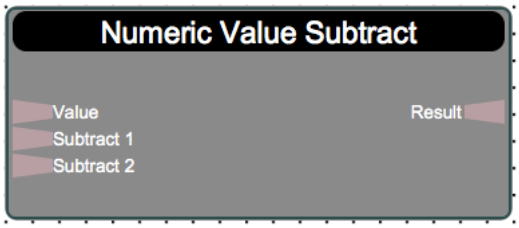
Numeric Value Add

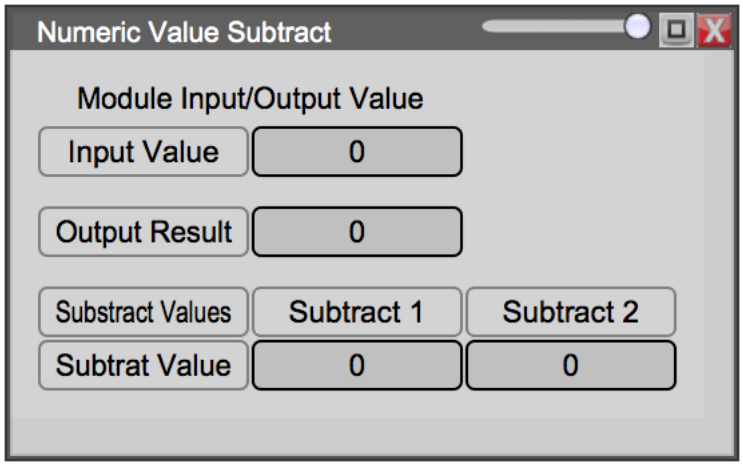




Numeric Value Add module will perform an add operation to all its input pin and send the added result to the output pin This module requires all its input pin to have valid values before it will process the output value.

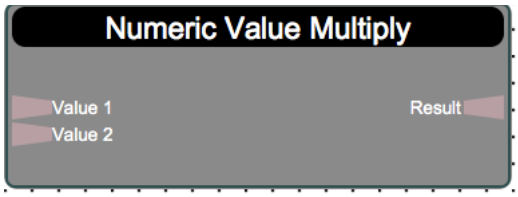
Numeric Value Subtract

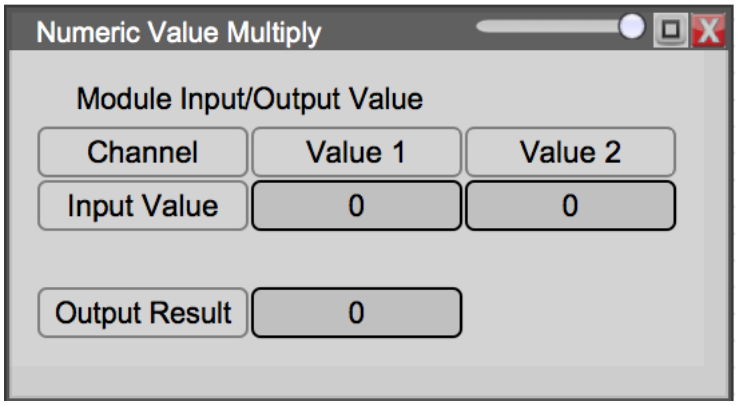




Numeric Value Subtract module will take the input value and subtract out the value form the other input pins. The subtract result will be send to the output pin. This module requires all its input pin to have valid values before it will process the output value.

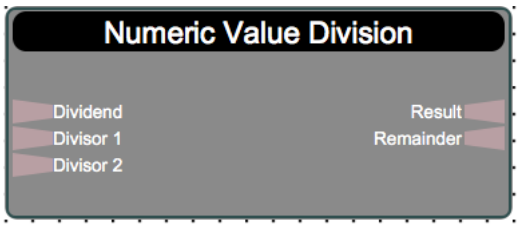
Numeric Value Multiply

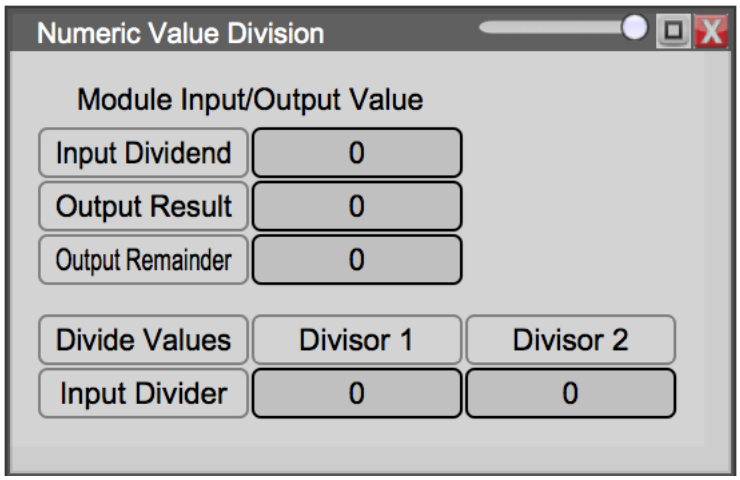




Numeric Value Multiply module will takes all the values from its input pins and multiply them together. The result will be placed to the output pin. This module requires all its input pin to have valid values before it will process the output value.

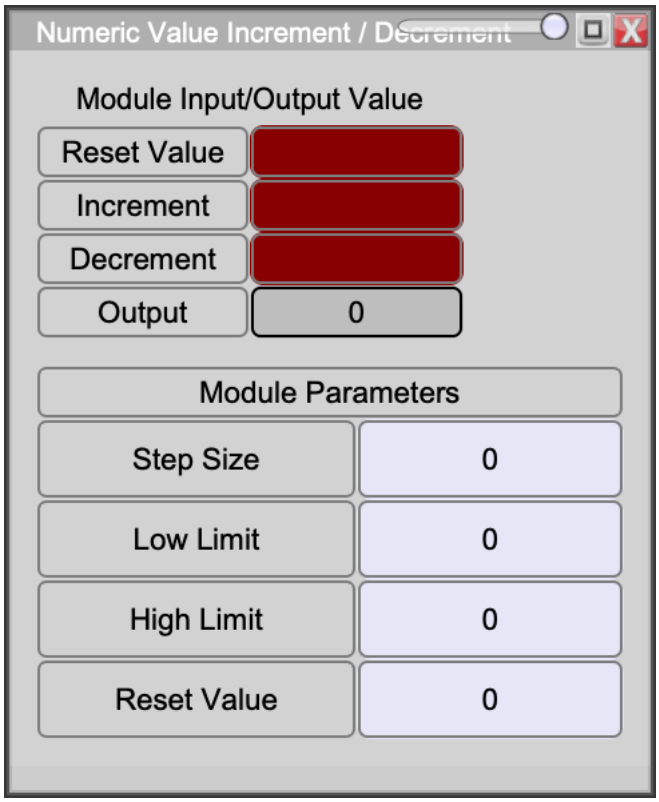
Numeric Value Division





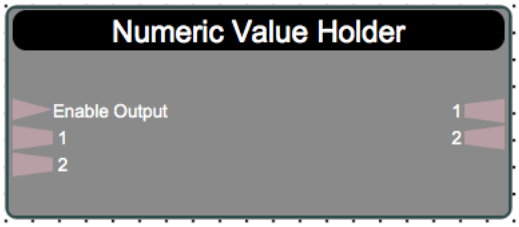
Numeric Value Division module will take the Dividend value and divide it will all its divisor values. The divide result and remainder will be placed to the corresponding output pin. This module requires all its input pin to have valid values before it will process the output value.

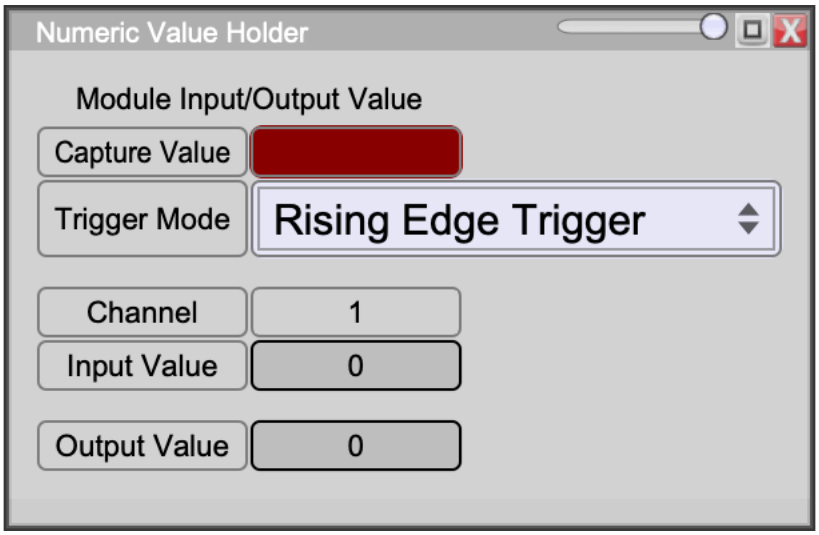
Numeric Value Increment/Decrement



This module has 3 logic input pins. “Reset” pin will reset the value to the specified initial value. “Inc” pin will increment the value by the step size. “Dec” pin will decrement the value by the step size. You can specify the Low/High limit so the module result will not get lower or higher than such limits.

Numeric Value Holder



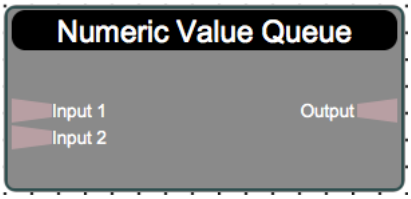


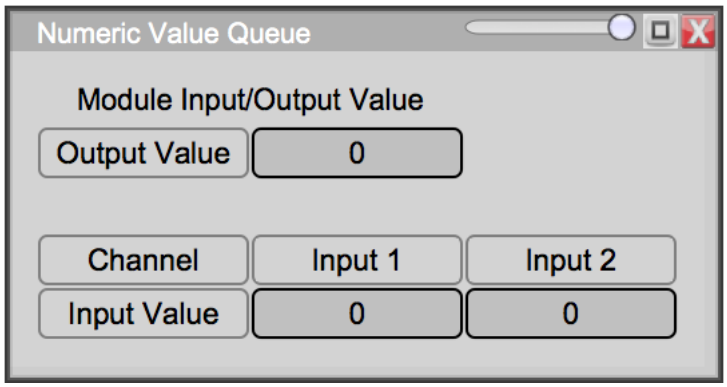
Numeric Value Holder module provide a buffer to capture a value snapshot. (It will hold a value data) When the capture input pin changes, it will read the input pin value for all channels and send the input value to the corresponding output pin. Each channel is independent to each other. Only valid input pin will be processed during and all other invalid pin will remain invalid.

We provide 2 triggering mode. “Rising edge trigger” will only capture the input value when the “Enable Output” pin changes from OFF to ON. When the pin change from ON to OFF, nothing will happen, but it will enable it to perform the next capture operation later.

Second mode is “Level trigger” mode. When this “Enable Output” is ON, it will continuous capture input value and pass it to output side.

Numeric Value Queue

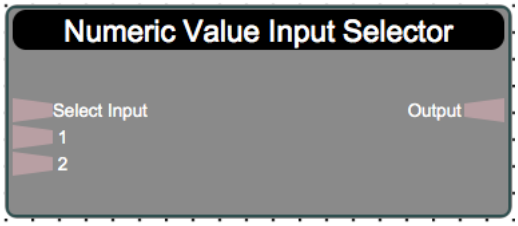


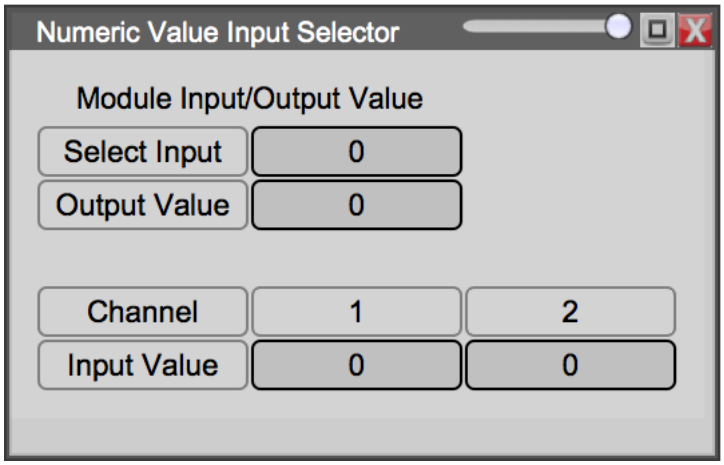


Numeric Value Queue module will take all input value that has been modified (updated by previous processing) and put it in a sequential queue. These values will then be sent to the subsequent module processing one-by-one. Each processing cycle will send out the next value in the queue for processing until all elements in the queue has been processed.

This module is particular useful if you want to have multiple processing path and after each individual processing path performed its action, you want to allow the result in each path to be processed one by one. (i.e. being send through same TCP connection to remote device).

Numeric Value Input Selector

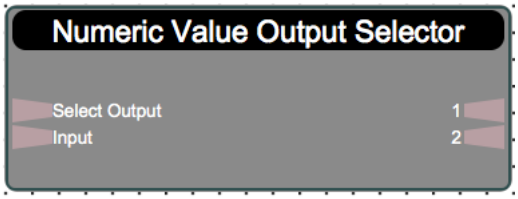


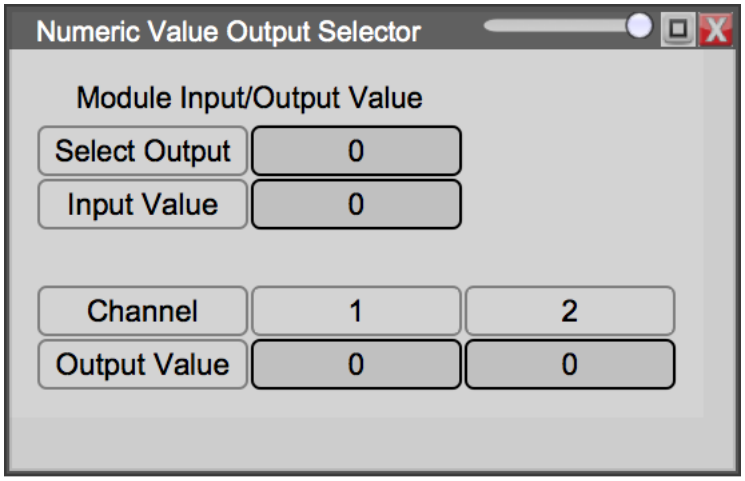


Numeric Value Input Selector module will connect the output pin to a specific input pin. Once connected, all data value change on the specific input pin will be pass to the output pin.

Select Input value should be ranging from 1-n. If the Select Input value is “0”, output will not be connected to anything and no data update will be performed. If the Select Input value is larger than the available input channels, the last input channel will be selected.

Numeric Value Output Selector

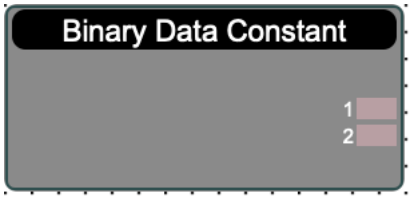


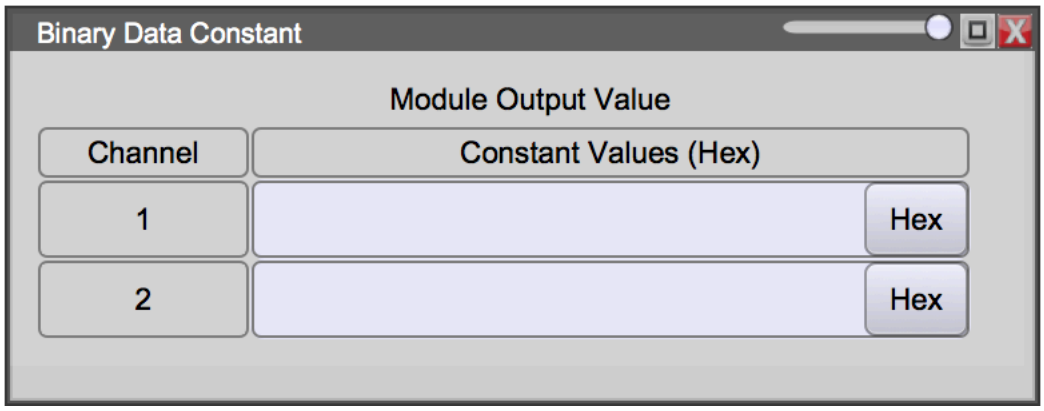


Numeric Value Output Selector is similar to the Input selector, except it work on the output end selection.

13. Binary data processing modules

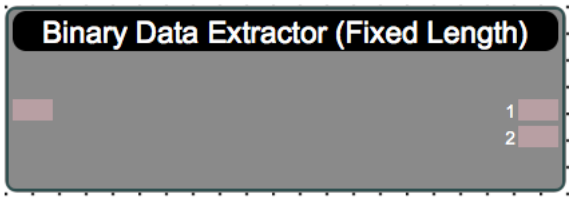
Binary Data Constant

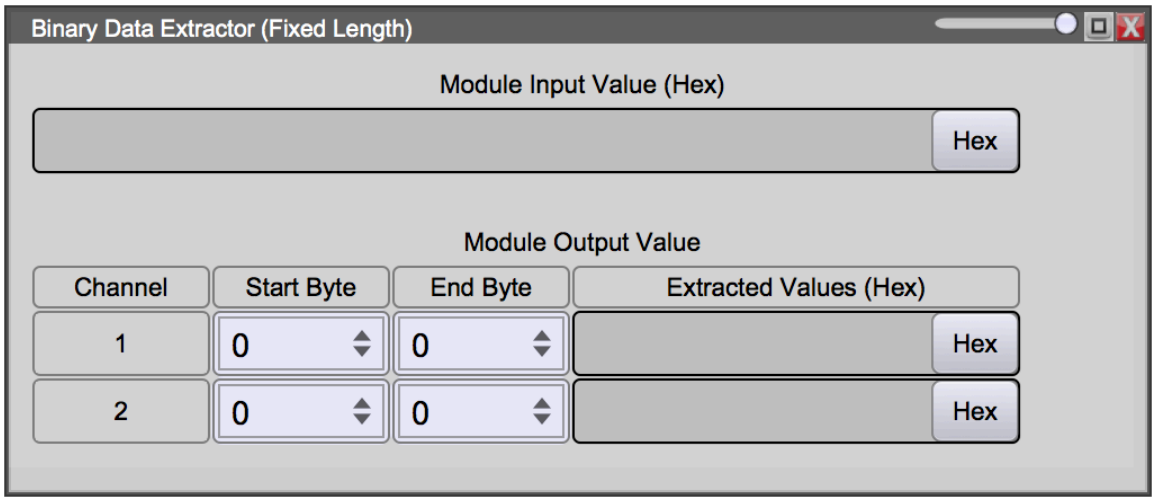




Binary Data Constant module provide a place to supply a constant binary value to your processing design. The entered binary data will be feed to the corresponding output pin.

Binary Data Extractor (Fixed length)

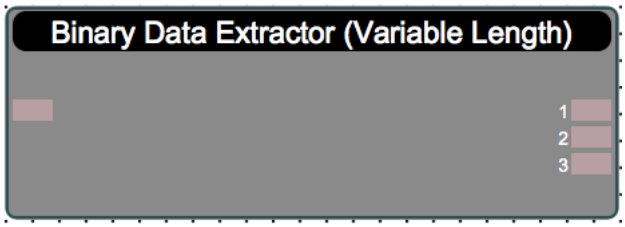


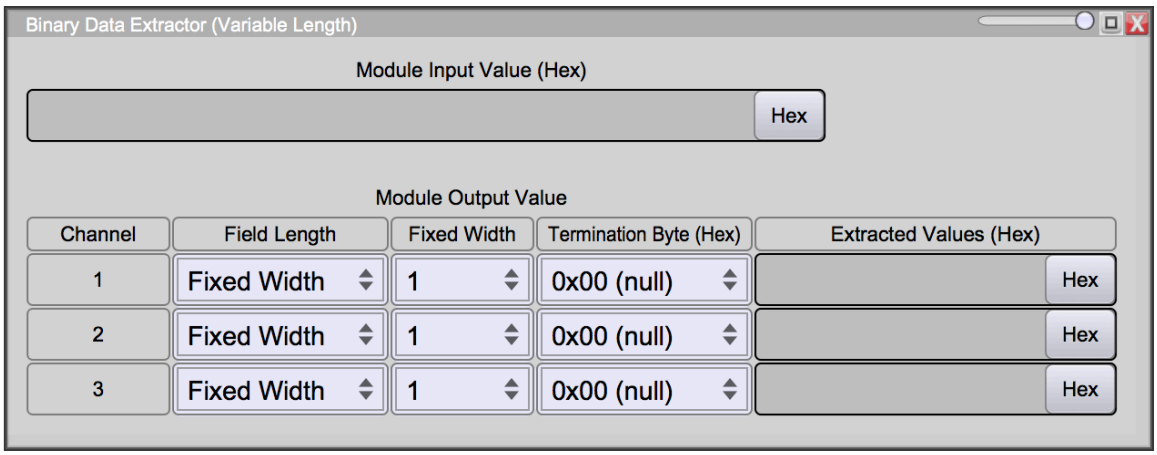


Binary Data Extractor module provide a mechanism to extract out certain part of the binary data to the output. This module works on fixed length operation. For each of the channel, you can specify the starting byte and ending byte to extract from the data array. If the start or end byte is out of range, the data will not be extracted.

Note: the end byte position will also be included into the extracted data.

Binary Data Extractor (Variable length)



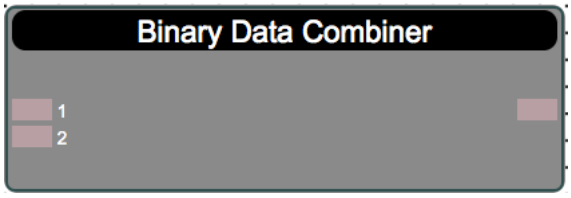


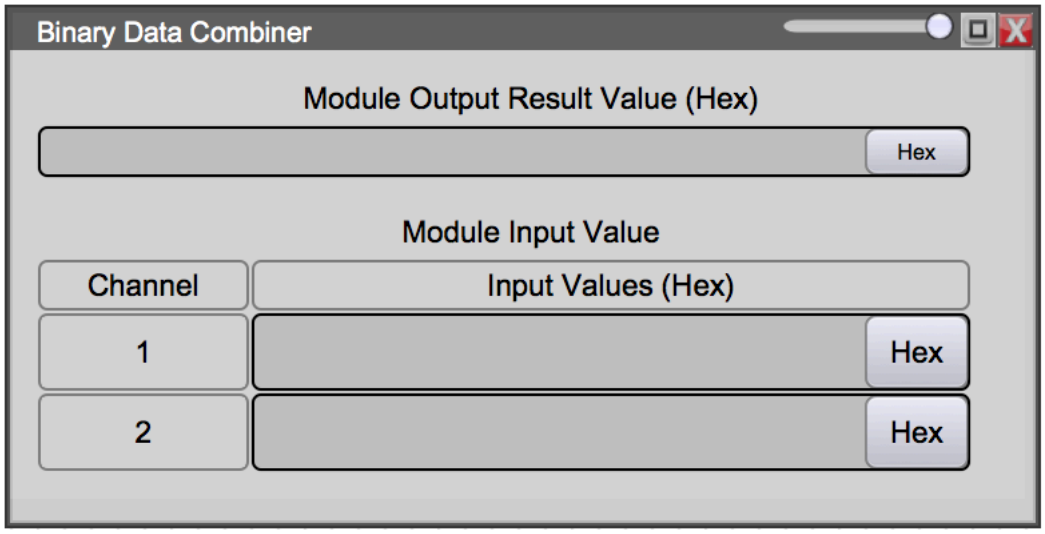
Binary Data Extractor (Variable length) is similar to the fixed length Binary data extractor. Except it allows user to select whether the specific part of a binary steam can be interpreted as variable length. This module will start from the beginning of the binary data and try to interpret each individual channel (starting from Channel 1). Once the previous channel is interpreted, it will use the remaining data for next channel interpretation.

Each Channel can be interpreted as fixed length or variable length. In the case of variable length, it will look for a termination byte to determine end of a channel data.

If termination byte is not found in the binary data, the complete input data will be passed to the channel result. And there is NO more data for next channel processing.

Binary Data Combiner

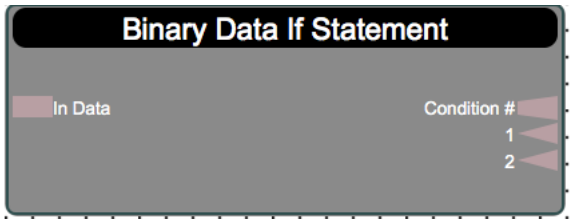


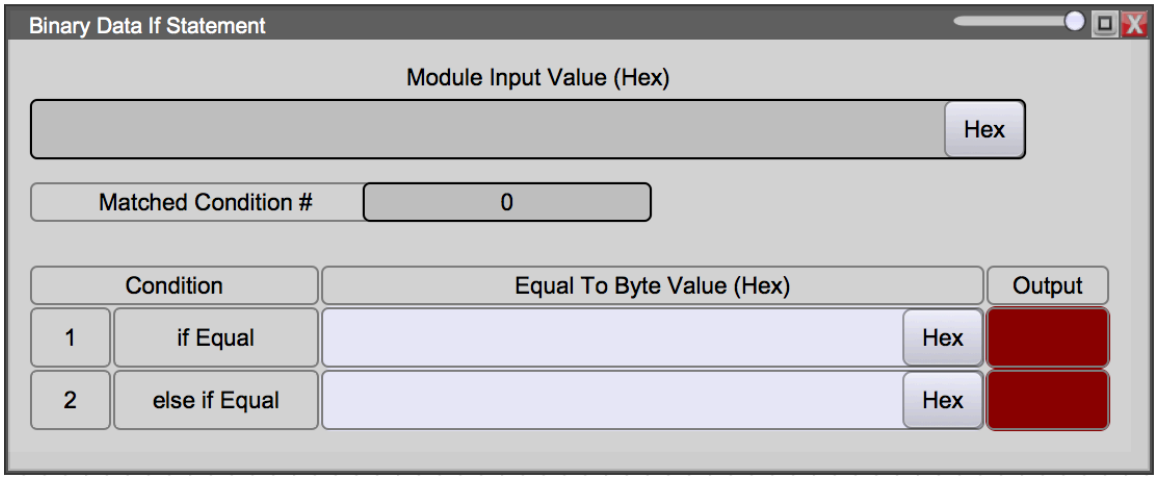


Binary Data Combiner module will concatenate all its input binary data into a single large binary data. The order of combining the data will follow the pin order.

This module is useful if you want to form a control string which consists of different values obtained from control flow.

Binary Data IF Statement



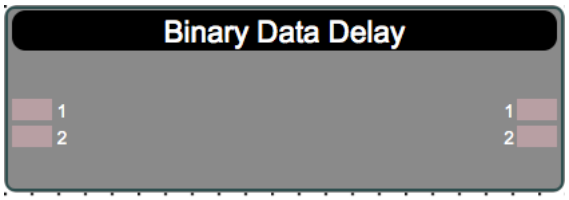


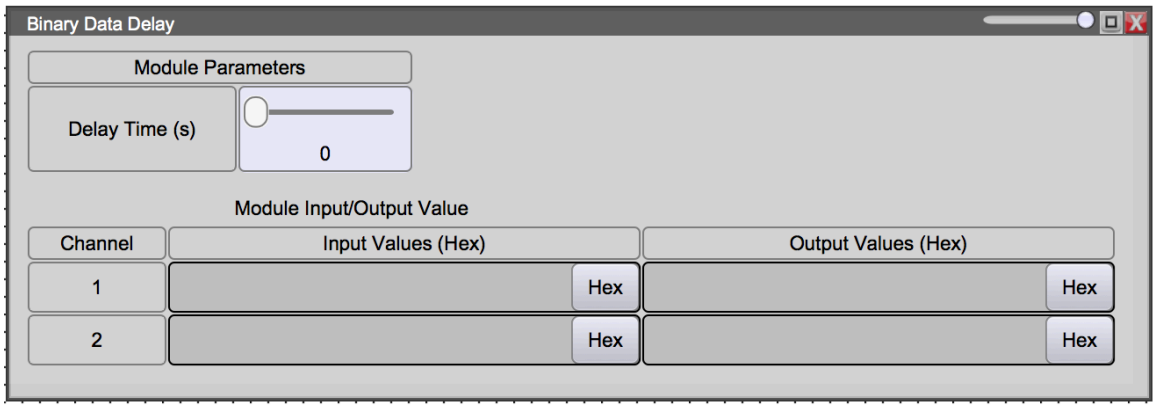
Binary Data If Statement module provide a test condition on the input binary data. This module will perform the condition testing following the channel order. As soon as it found a matching condition it will set the logic output of that channel to be ON. It will also update the matched condition # to the output pin.

For the condition test, we only support exact match of binary data.

If all condition does not match the input binary data, all output state will become OFF and the Matched Condition # will become “0”.

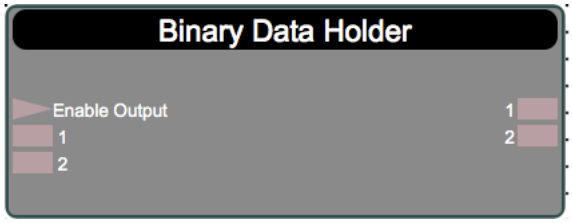
Binary Data Delay

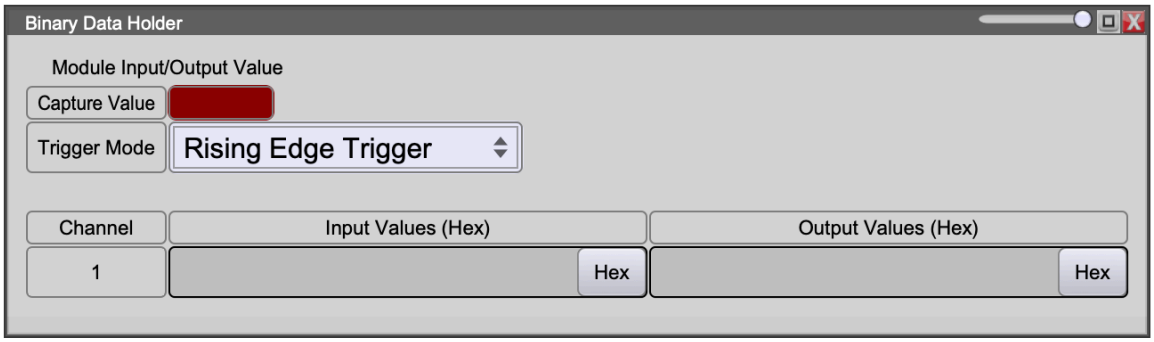




Binary Data Delay module is similar to Logic Delay module, except it works on binary data values. It will pass the input binary data to the output pin after certain seconds of delay. Each channel operation is independent to the other channels.

Binary Data Holder



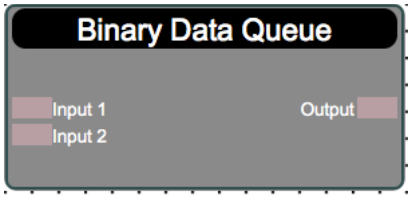


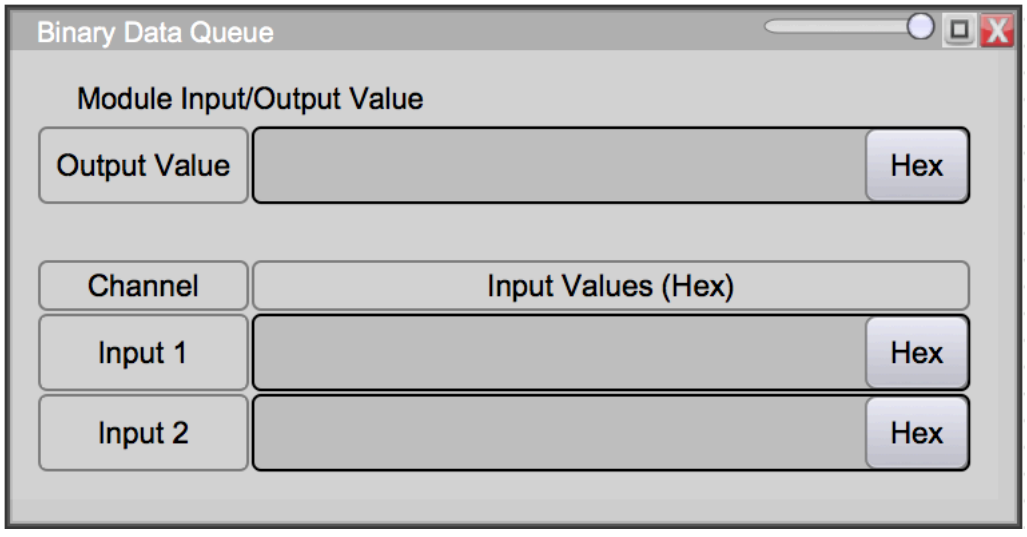
Binary Data Holder module provide a buffer to capture a value snapshot. (It will hold a binary data) When the capture input pin changes, it will read the input pin binary data for all channels and send the input binary data to the corresponding output pin. Each channel is independent to each other. Only valid input pin will be processed during and all other invalid pin will remain invalid.

We provide 2 triggering mode. “Rising edge trigger” will only capture the input value when the “Enable Output” pin changes from OFF to ON. When the pin change from ON to OFF, nothing will happen, but it will enable it to perform the next capture operation later.

Second mode is “Level trigger” mode. When this “Enable Output” is ON, it will continuous capture input value and pass it to output side.

Binary Data Queue



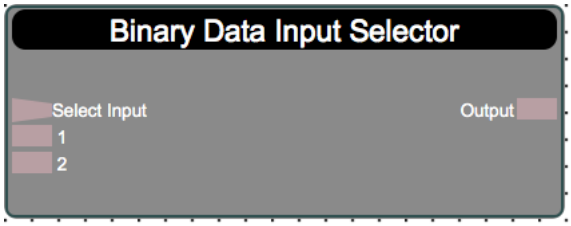


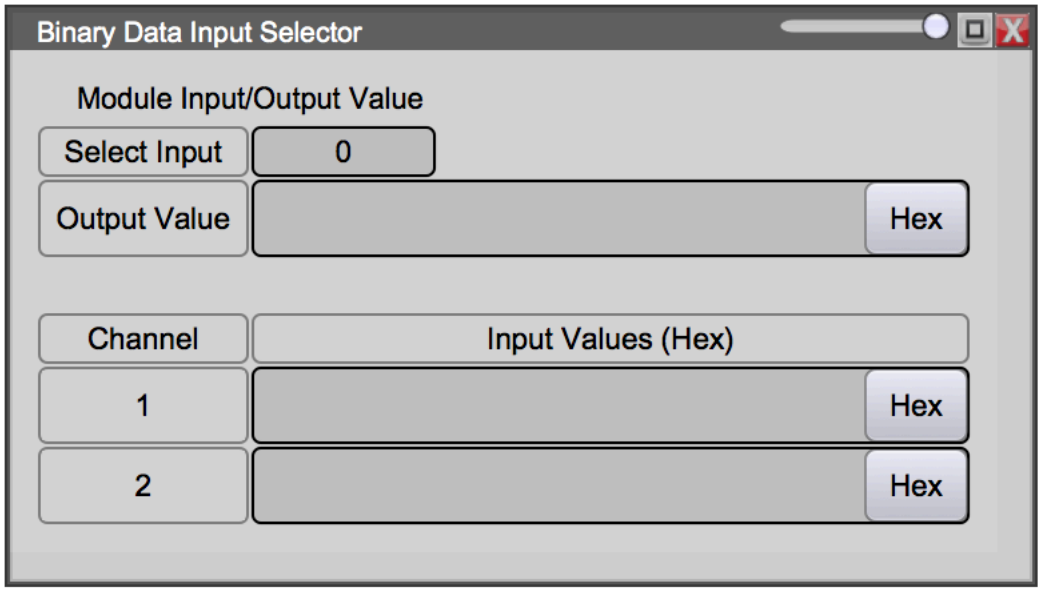
Binary Data Queue module will take all input binary data that has been modified (updated by previous processing) and put it in a sequential queue. These data will then be

sent to the subsequent module processing one-by-one. Each processing cycle will send out the next value in the queue for processing until all elements in the queue has been processed.

This module is particular useful if you want to have multiple processing path and after each individual processing path performed its action, you want to allow the result in each path to be processed one by one. (i.e. being send through same TCP connection to remote device).

Binary Data Input Selector

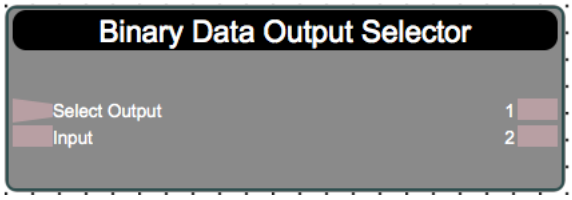


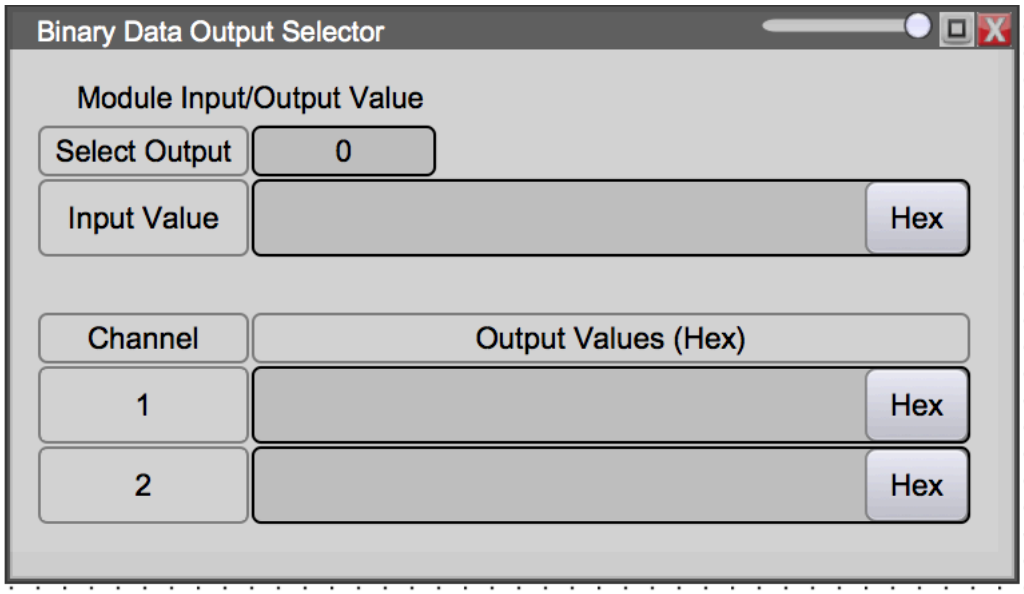


Binary Data Input Selector module will connect the output pin to a specific input pin. Once connected, all binary data value change on the specific input pin will be pass to the output pin.

Select Input value should be ranging from 1-n. If the Select Input value is “0”, output will not be connected to anything and no data update will be performed. If the Select Input value is larger than the available input channels, the last input channel will be selected.

Binary Data Output Selector

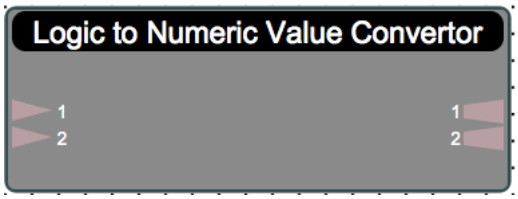


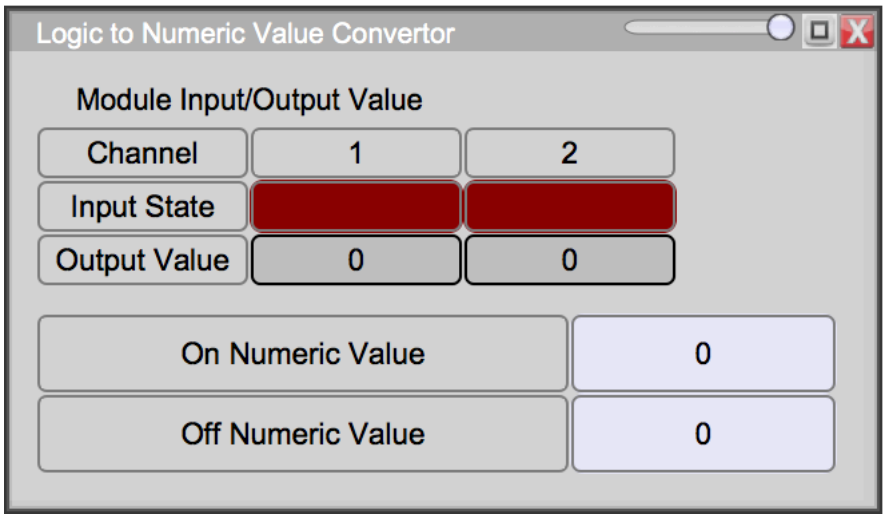


Binary Data Output Selector is similar to the Input selector, except it work on the output end selection.

14. Data conversion modules

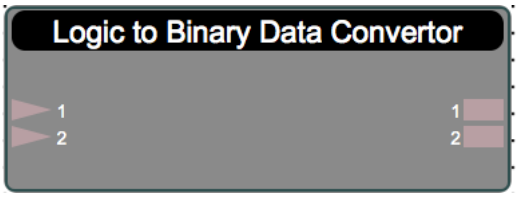
Logic to Numeric Value Convertor

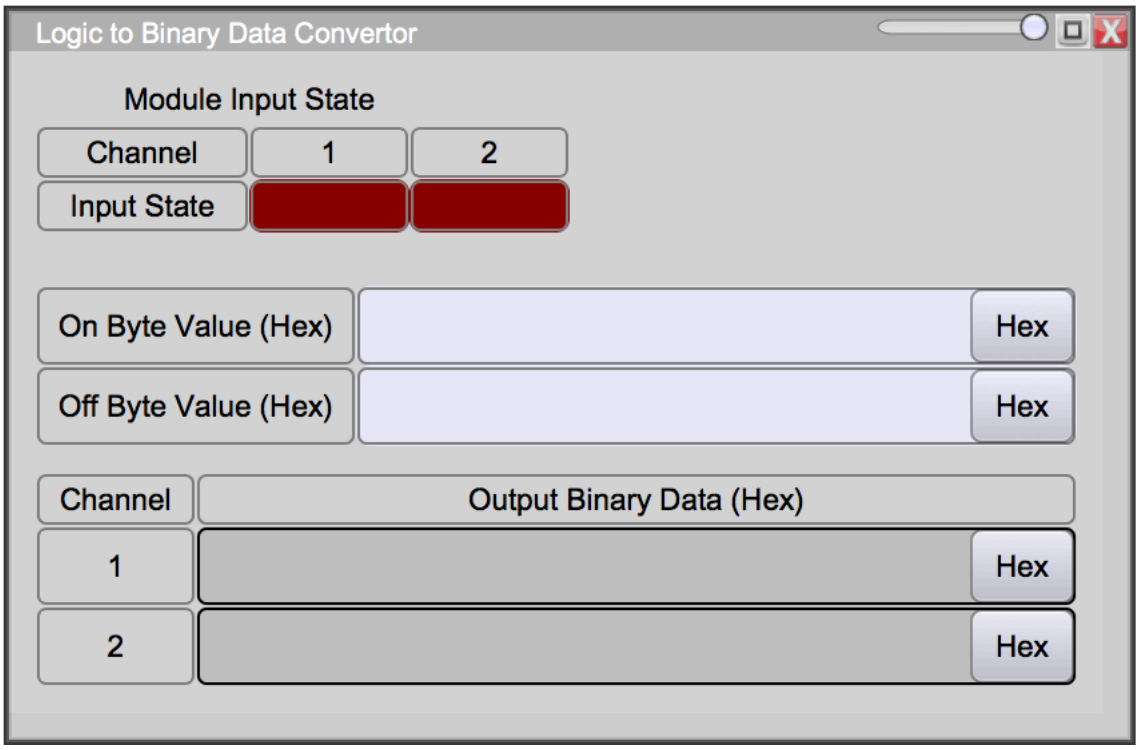




Logic to Numeric Value Convertor module will convert logic ON/OFF signal into a numeric value. You can enter the numeric values for ON and OFF state. Each channel output value will be set as the entered numeric value based on the input state. If the input pin state is invalid, the output will remain invalid. Each channel is independent to each other.

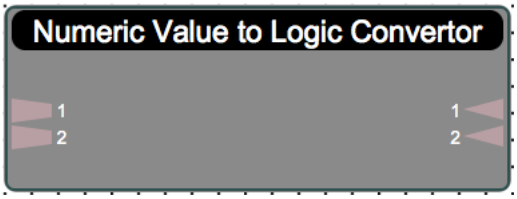
Logic to Binary Data Convertor

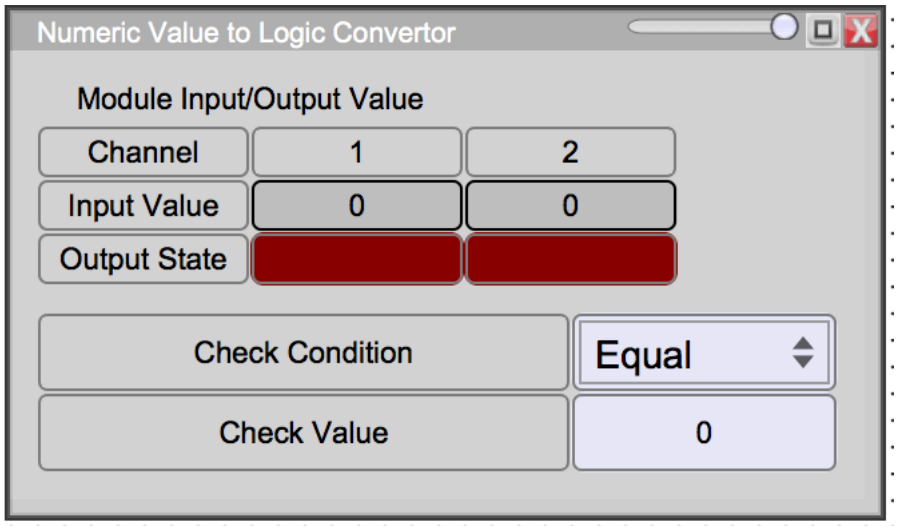




Logic to Binary Data Convertor module will convert logic ON/OFF signal into a binary data. You can enter the binary data for ON and OFF state. Each channel output value will be set as the entered binary data value based on the input state. If the input pin state is invalid, the output will remain invalid. Each channel is independent to each other.

Numeric Value to Logic Convertor



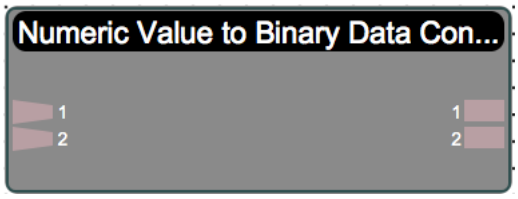


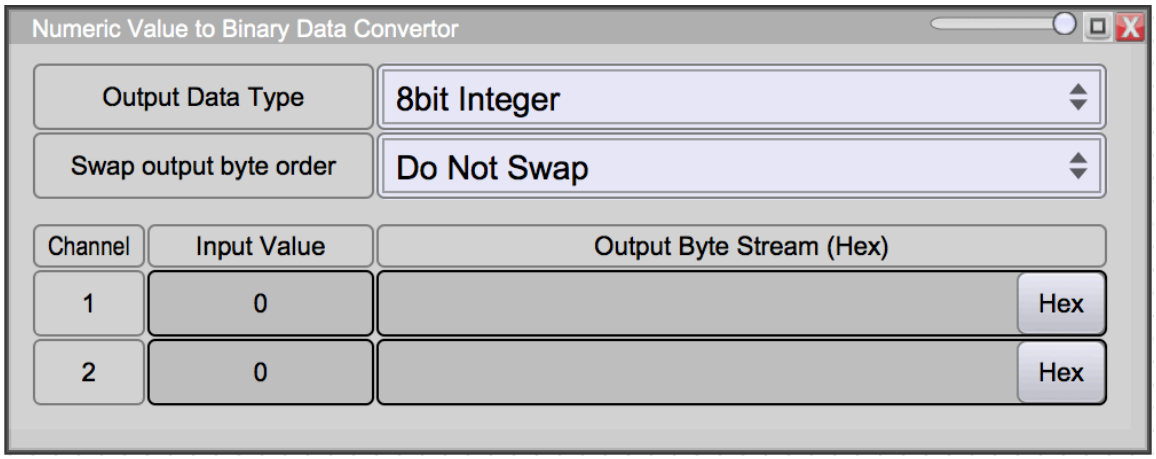
Numeric Value to Logic Convertor module will set the channel output pin logic state based on the input numeric value. If the input value falls into the check condition, the corresponding output state will be set accordingly.

For the condition test, we support Equal, NOT Equal, Greater Than, Greater Than or Equal, Less Than, Less Than or Equal etc.

Each channel is independent to each other. And if the input is invalid the corresponding pin will remain invalid.

Numeric Value to Binary Data Convertor



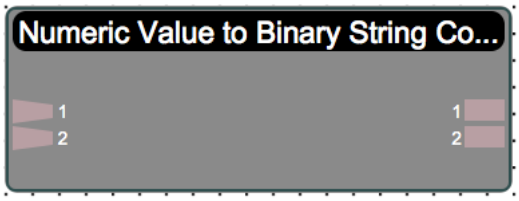


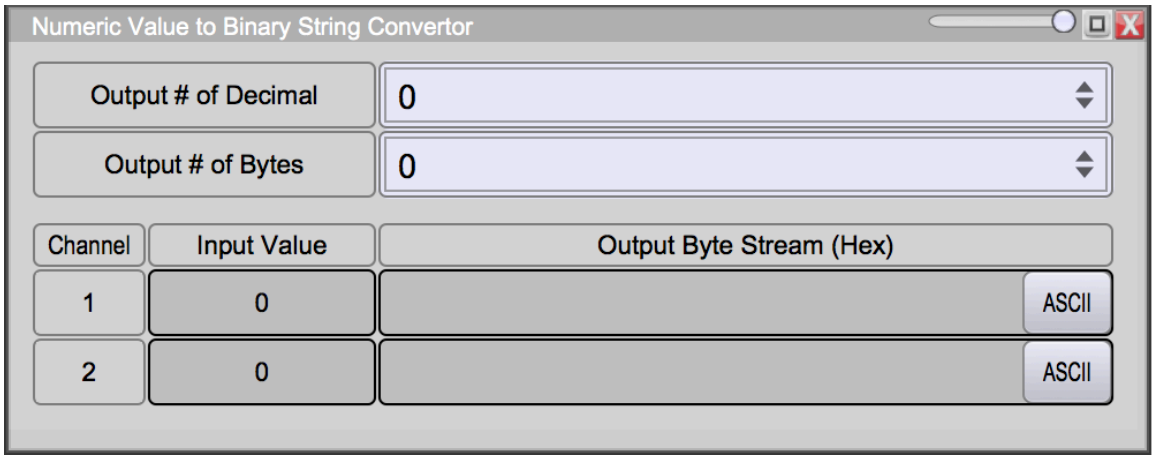
Numeric Value to Binary Data Convertor module will convert the numeric data value into the binary representation of the value. We support the following binary data format for the input value:

8 bit integer - convert to 1 byte representation of a decimal value
16 bit integer - convert to 2 bytes representation of a decimal value
32 bit integer - convert to 4 bytes representation of a decimal value
32 bit float - convert to 4 bytes floating point representation of the value
64 bit double - convert to 8 bytes floating point representation of the value

For output format that has more than 1 byte, there is an option to swap the byte ordering for (little endian or big endian representation).

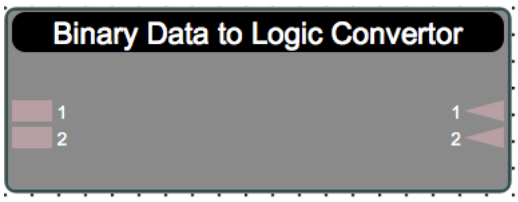
Numeric Value to Binary String Convertor

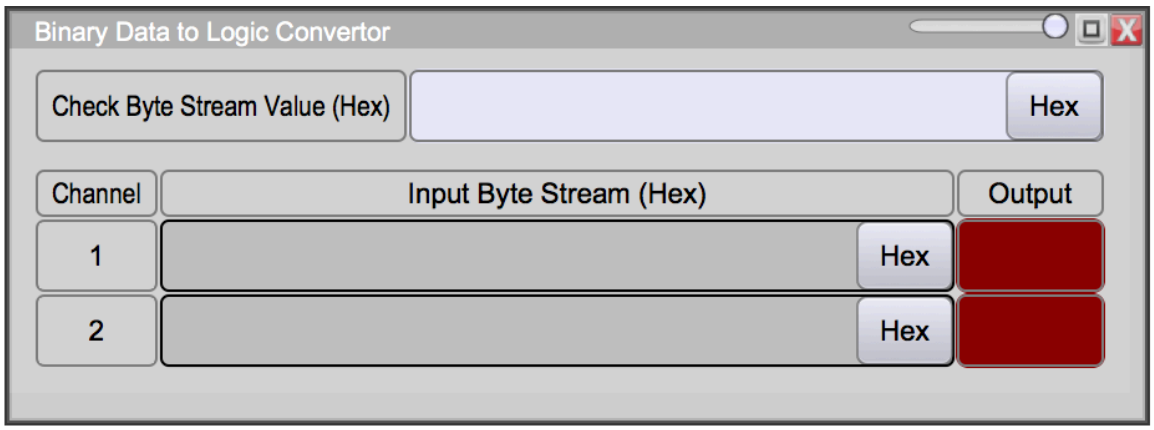




Numeric Value to Binary String convertor module will convert the binary data into string representation of the value. You can specify the total number of bytes after the conversion and the number of decimal point need to maintain in the result string.

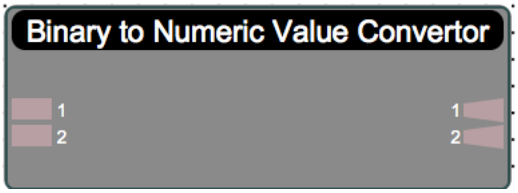
Binary Data to Logic Convertor

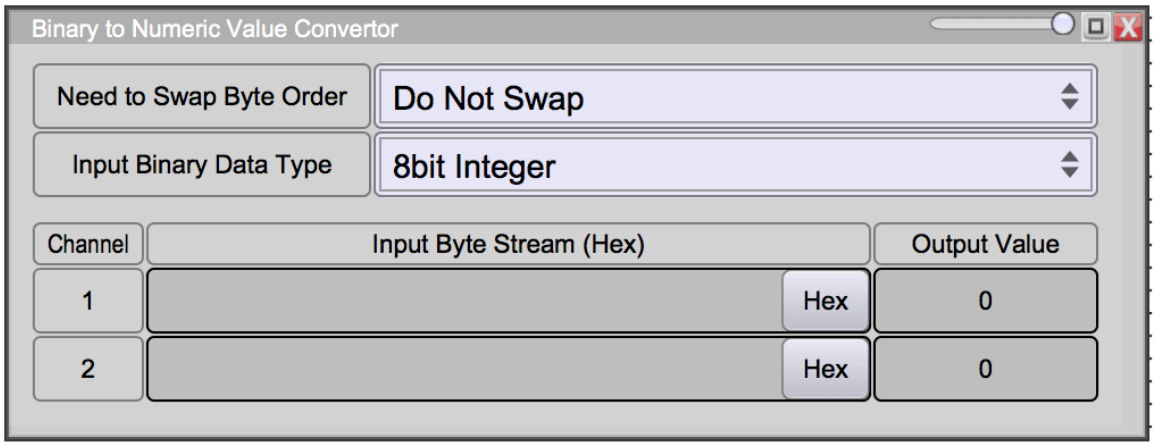




Binary Data to Logic Convertor module will check the input byte data against the “Check Byte Stream Value”. If the binary data values match, the output state will be ON. Otherwise the output state will be OFF.

Binary Data to Numeric Value Convertor



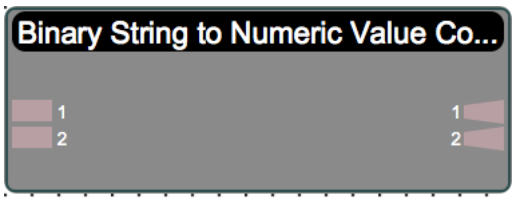


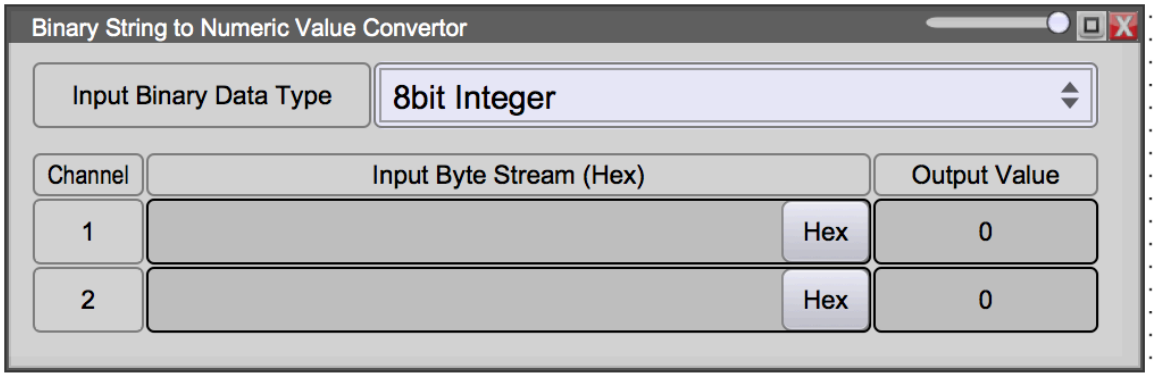
Binary Data to Numeric Value Convertor module will interpret the input binary data as a numeric value representation in binary format. We support the following formats:

8 bit integer - convert as1 byte representation of a decimal value
16 bit integer - convert as 2 bytes representation of a decimal value
32 bit integer - convert as 4 bytes representation of a decimal value
32 bit float - convert as 4 bytes floating point representation of the value
64 bit double - convert as 8 bytes floating point representation of the value

If the input data has more bytes that required, the remaining bytes will be ignored.

Binary String to Numeric Value Convertor





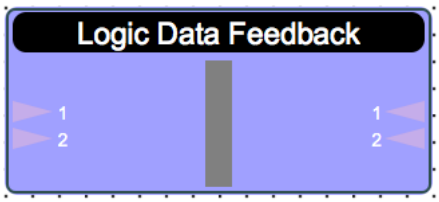
Binary String to Numeric Value Convertor module will interpret the incoming binary data as string value. And it will convert the string value into corresponding number value.

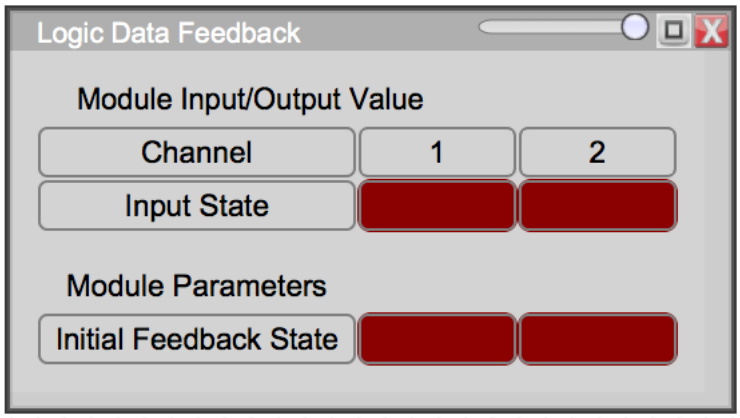
15. Data Feedback modules

In general controller processing flow will start from the input modules and process according to the signal flow. Looping of normal modules is not allowed. But in some situation feeding back the data value back to previous module is required. This can be achieved by the Data Feedback modules, these module will obtain the data value from previous execution cycle and that value will be use as input for next execution cycle.

Each data type will have its own feedback module to handle the corresponding data types. Each module will also provide an initial value so that the first execution cycle will have some data to work from.

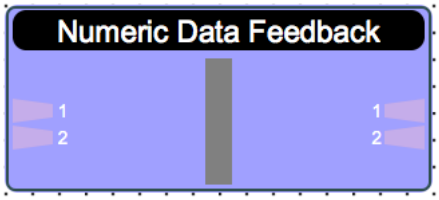
Logic Data Feedback

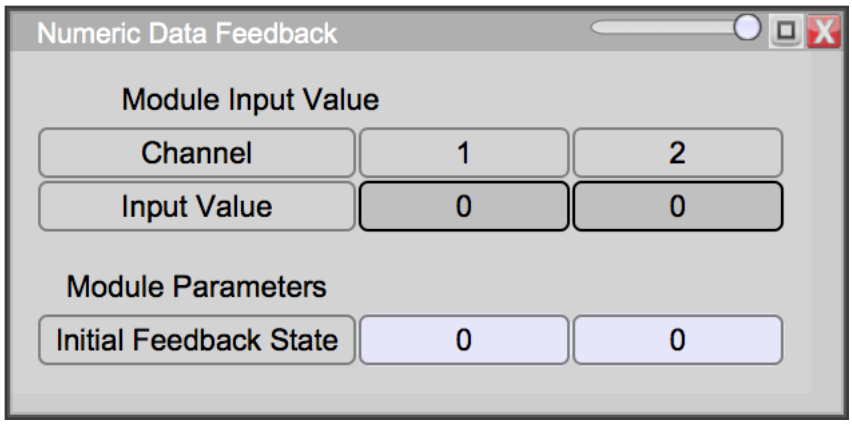




Logic Data Feedback module handles logic data feedback. User can select the initial value to be feed into the processing flow in the first execution cycle.

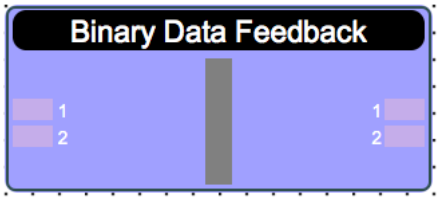
Numeric Data Feedback





Numeric Data Feedback module handles numeric data feedback. User can select the initial value to be feed into the processing flow in the first execution cycle.

Binary Data Feedback

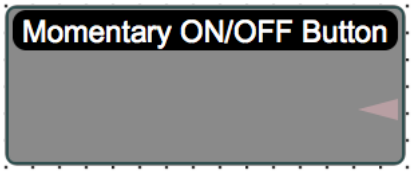


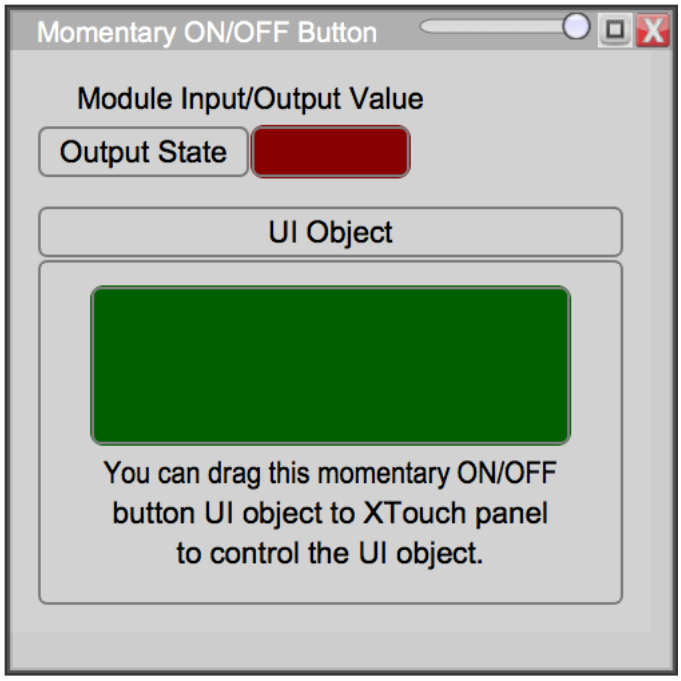
Binary Data Feedback module handles binary data feedback. User can select the initial value to be feed into the processing flow in the first execution cycle.

16. UI modules

Besides internal processing the design flow, Solaro Universal Controller also supports User Interface object manipulation by user.

Momentary ON/OFF Button User Interface



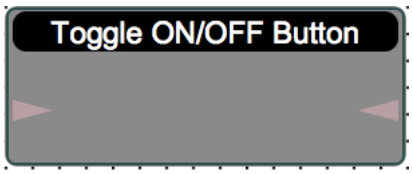


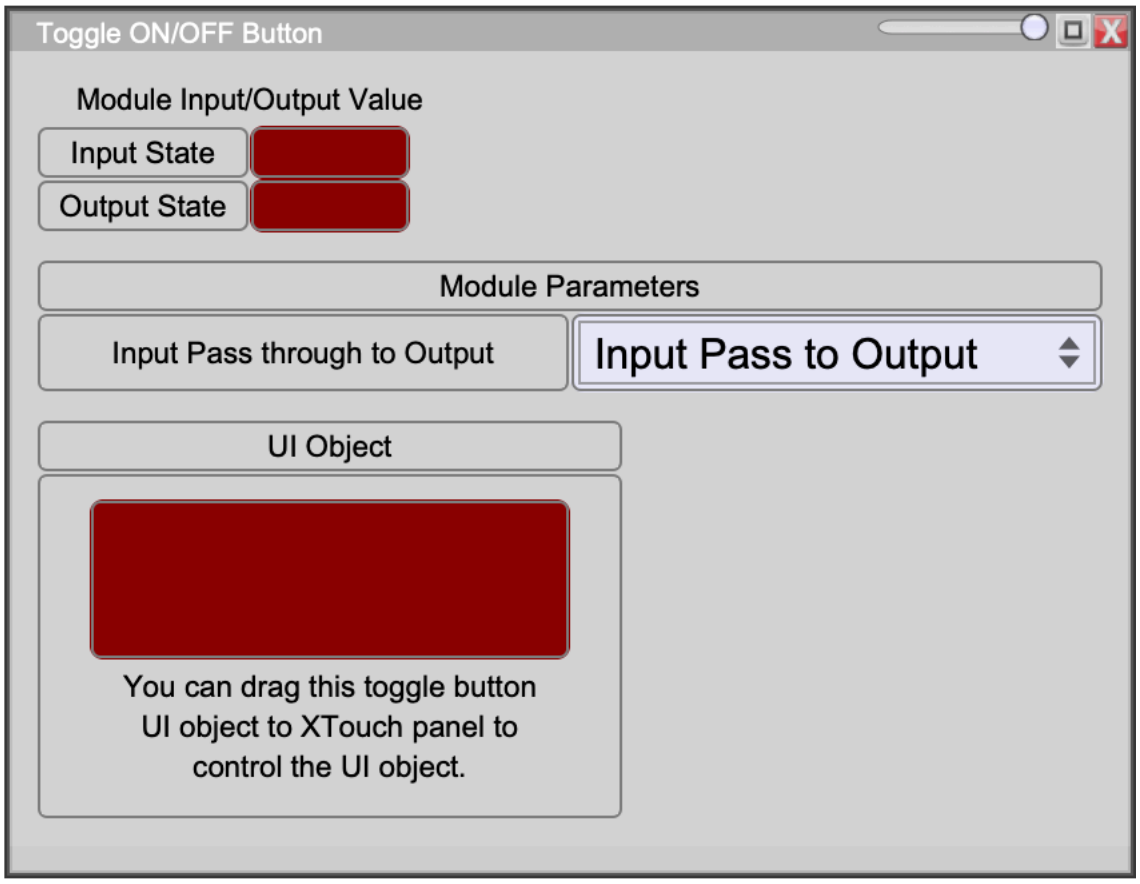
Momentary ON/OFF Button User Interface module provide a push button object for user to control ON/OFF state. When you press the UI button, it will generate an ON state to the output of the module. The ON state will remain until you release the button, then the output state will become OFF.

This module will only have output pin, as this UI object state cannot be controlled by other modules result.

This UI object can be placed onto the XTouch panel (or XTouchApp running on Android and iOS). This will provide user control for the controller functionality.

Toggle ON/OFF Button User Interface



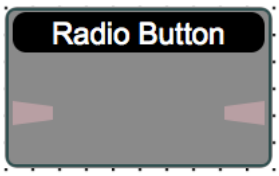


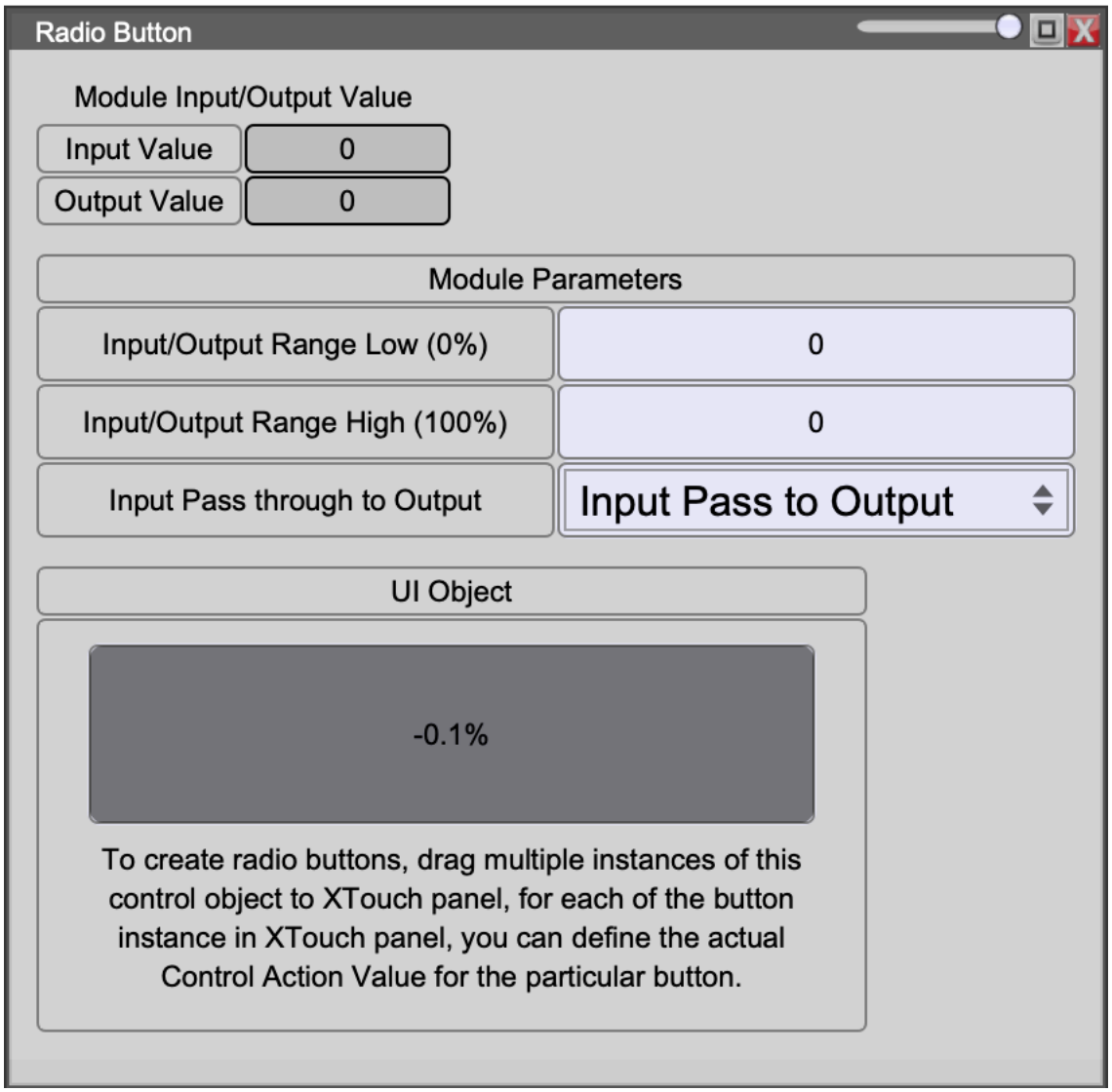
Toggle ON/OFF Button User Interface module provide a toggle button object for user to control. If you change the state on this UI object, the output pin of this module will follow the UI state. On the other hand, if you feed a logic data to this module, the UI object state will also follow the input value.

We provided 2 modes of operation. You can pass the input pin state to the output pin or you can only allow the input pin to update UI state only. Under the non-pass through case, output pin will only be changed when UI object is being modified by user action.

This UI object can be placed onto the XTouch panel (or XTouchApp running on Android and iOS). This will provide user control for the controller functionality.

Radio Button User Interface





Radio Button User Interface module provide a mechanism for user to define its own radio button. This radio button is only useful when you define it in an XTouch panel, as in this UI panel only 1 button is available, you need to drag out multiple instances of this button to XTouch panel to form a Radio button group.

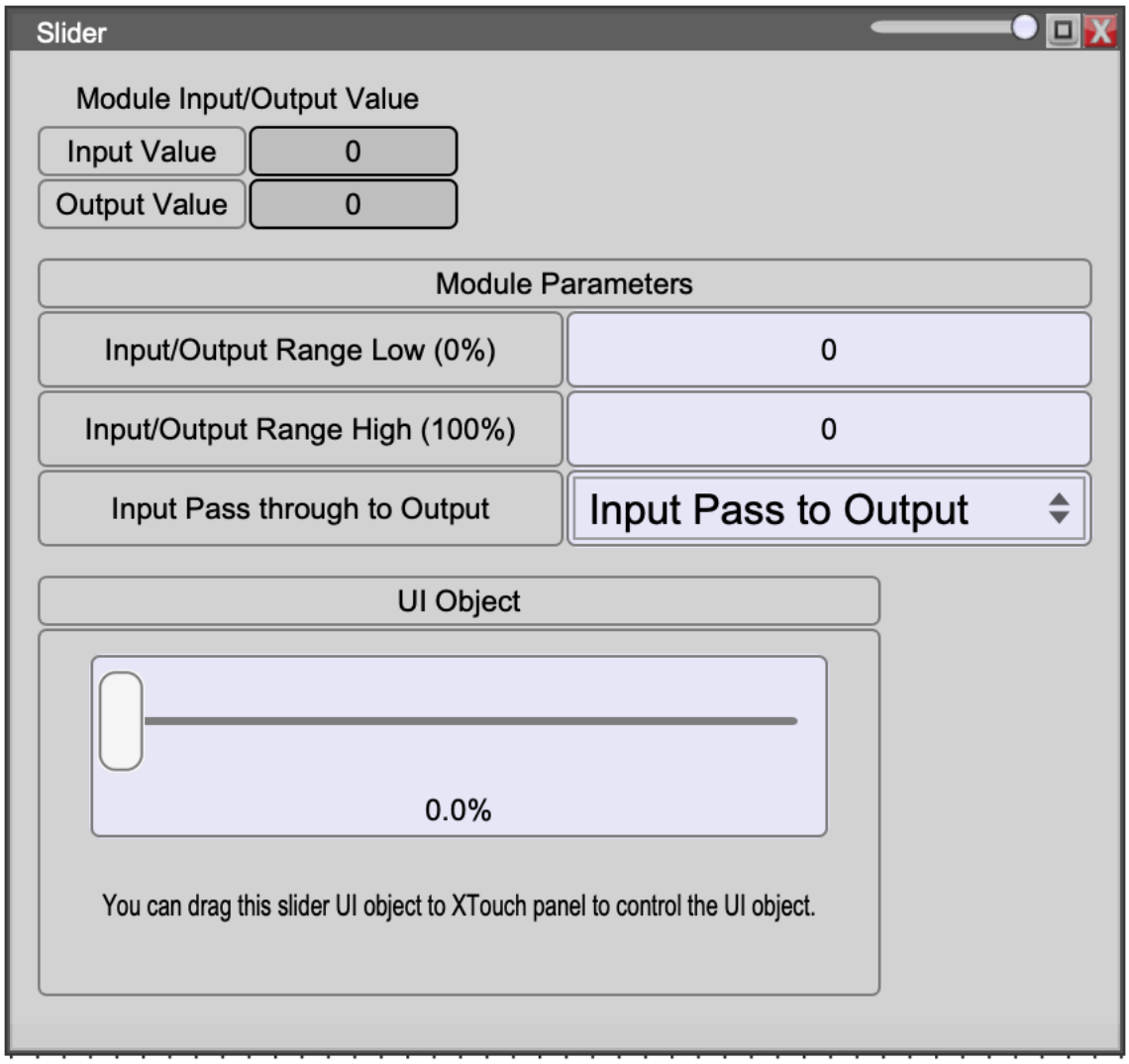
When you drag the multiple buttons to the XTouch panel, these buttons will automatically be grouped together by the control parameter behind it. Meaning the actual parameter state are sharing the same parameter value. You can define the Control Action Value for each individual button within the Radio button group. When the button is pressed, the Control Action Value will be sent to set the parameter value. Other buttons in the group will received the new parameter value change, and each button will highlight

itself if the parameter value matches with its Control Action Value. This will create Radio button effect.

To set this radio button to have a different scale, you CANNOT do it in this control page. You need to drag this radio button to the XTouch panel, in there you can further customize this UI object. You can specify the max, min, number of decimal and display units for the button. Usually you will use the same range as the radio button UI object setup in controller side. In XTouch application, the proper value range and unit can be displayed.

We provided 2 modes of operation. You can pass the input pin state to the output pin or you can only allow the input pin to update UI state only. Under the non-pass through case, output pin will only be changed when UI object is being modified by user action.

Slider User Interface



Slider User Interface module provide a slider object for user to control. The slider object scale is based on a 0% to 100% scale. And when you change the slider object, it will calculate the actual value according to the range provided. And the calculated value will be sent to the output numeric pin.

On the other hand, when a numeric value is pass into this module, it will be converted to a % value based on the ranges. The % value will be updated to the slide 0-100% range. In this module control panel, the unit for the slider is 0-100%.

To set this slider to have a different scale, you CANNOT do it in this control page. You need to drag this slider to the XTouch panel, in there you can further customize this UI object. You can specify the max, min, number of decimal and display units for the slider.

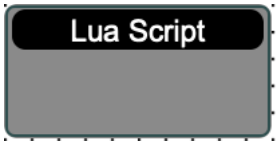
Usually you will use the same range as the slider UI object setup in controller side. In XTouch application, the proper value range and unit can be displayed.

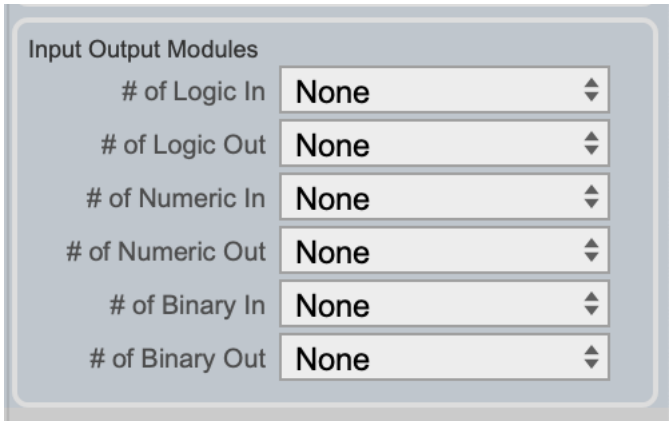
We provided 2 modes of operation. You can pass the input pin state to the output pin or you can only allow the input pin to update UI state only. Under the non-pass through case, output pin will only be changed when UI object is being modified by user action.

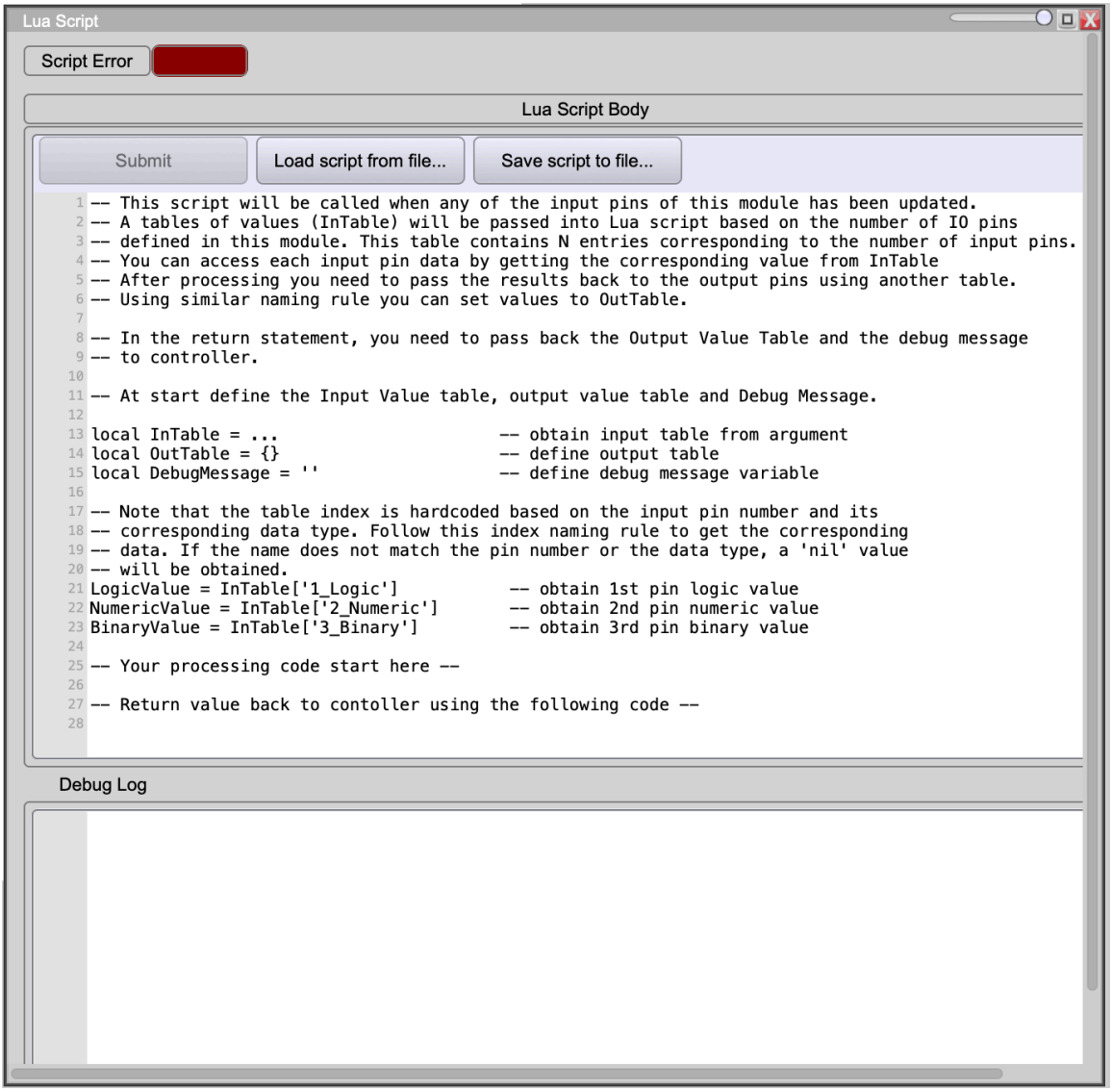
This UI object can be placed onto the XTouch panel (or XTouchApp running on Android and iOS). This will provide user control for the controller functionality.

17. Lua Script module

Lua Script






Lua Script
Script Error
Lua Script Body
Submit
Load script from file...
Save script to file...
-- This script will be called when any of the input pins of this module has been updated.
-- A tables of values (InTable) will be passed into Lua script based on the number of IO pins
-- defined in this module. This table contains N entries corresponding to the number of input pins.
-- You can access each input pin data by getting the corresponding value from InTable
-- After processing you need to pass the results back to the output pins using another table.
-- Using similar naming rule you can set values to OutTable.
-- In the return statement, you need to pass back the Output Value Table and the debug message
-- to controller.
-- At start define the Input Value table, output value table and Debug Message.
local InTable = ... -- obtain input table from argument
local OutTable = {} -- define output table
local DebugMessage = '' -- define debug message variable
-- Note that the table index is hardcoded based on the input pin number and its
-- corresponding data type. Follow this index naming rule to get the corresponding
-- data. If the name does not match the pin number or the data type, a 'nil' value
-- will be obtained.
LogicValue = InTable['1_Logic'] -- obtain 1st pin logic value
NumericValue = InTable['2_Numeric'] -- obtain 2nd pin numeric value
BinaryValue = InTable['3_Binary'] -- obtain 3rd pin binary value
-- Your processing code start here --
-- Return value back to contoller using the following code --
Debug Log

Lua script module is a special module that will execute based on user defined scripts. You can have as many Lua module in your design as you like. Each Lua module will be executed in the order of processing flow. All these Lua modules runs on the same environment (meaning you can share global variables among them).

To design your Lua script module, you need to first define the number of IO pins the module has. It can support Logic, Numeric and Binary data types. To specify the number of Input and Output pin, you need to select the module and then on the right-hand side "object property" area you can select the number of Input/Output for each data types.

When the script is executed, all input pins data will be passed in as a table called "InTable" you can access each pin value by obtaining the table value from its index. The index naming is based on the position of the pin in the module and the corresponding data type. (i.e. first pin will be using "1_" following by its data type. If it is logic it will be "1_Logic" If it is numeric it will be "1_Numeric". If it is binary it will be "1_Binary". Similar naming convention will apply to output table as well.

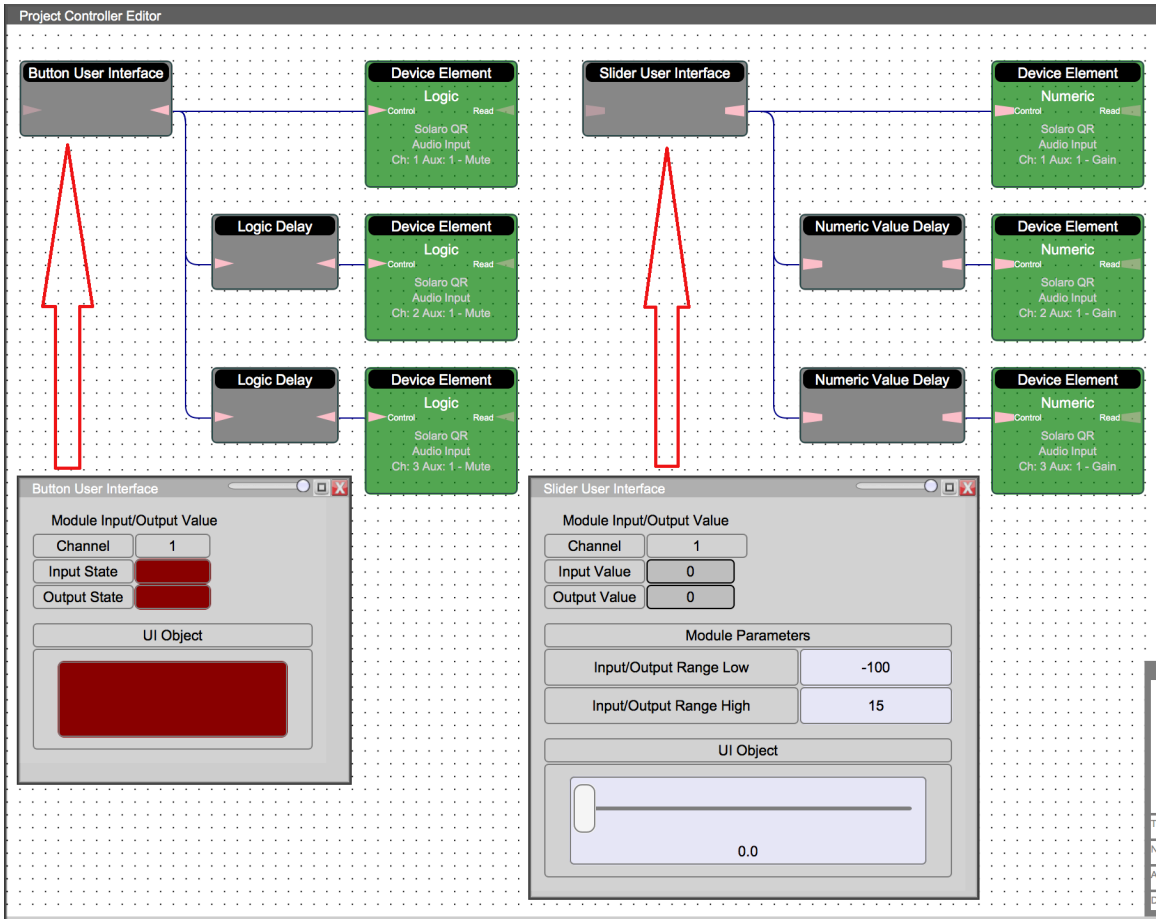
After processing, you will save your result to a output array "OutTable". Similarly you set the output table value using index corresponding to the output pin position and data types.

Besides data processing, we also provide a debug mechanism, you can do print function to a variable called "DebugMessage", this DebugMessage value will be display at the Debug area. This way you can have some debug capability.

During ONLINE mode, you can modify the script and press the "Submit" button to update the script on the device. This new script will be executed in the next processing run. This will enable fast development cycle.

18. Design Examples

Control Timing Sequence Example



In this example, you can separate it into the mute control part (Logic) and slider control part (Numeric).

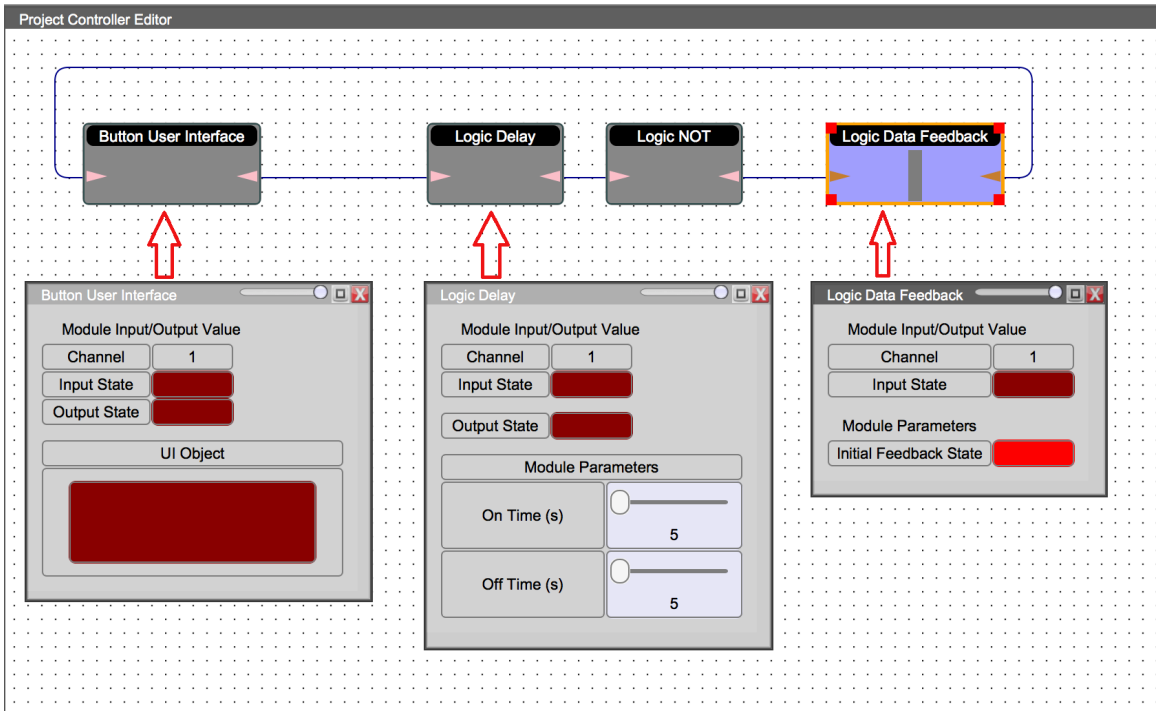
For the mute control part, we use a UI button module to send control to 3 different mute buttons. And these mute buttons control will be based on the delay setup in the Logic Delay module. This is a typical application if you want to control equipment ON/OFF state in a timing sequence. In the UI button module detail page, there is NO parameter setting required. You can just click on the UI object button to send ON/OFF command to the processing flow.

If you want to have the UI toggle button to be used in a touch panel, just drag and drop this UI Object to the touch panel design.

Beside mute, you can also do slider control sequence, which is the second part of the controller design. In the slider UI object, you need to setup how the 0-100% of the UI slider represented the output range. In this case the max is 15dB and min is -100dB. These slider value will be feed to the gain control of different channel based on the delay time.

Similar to the UI toggle button, you can drag and drop the slider UI object to a Touch panel for control.

Logical Data Feedback Example



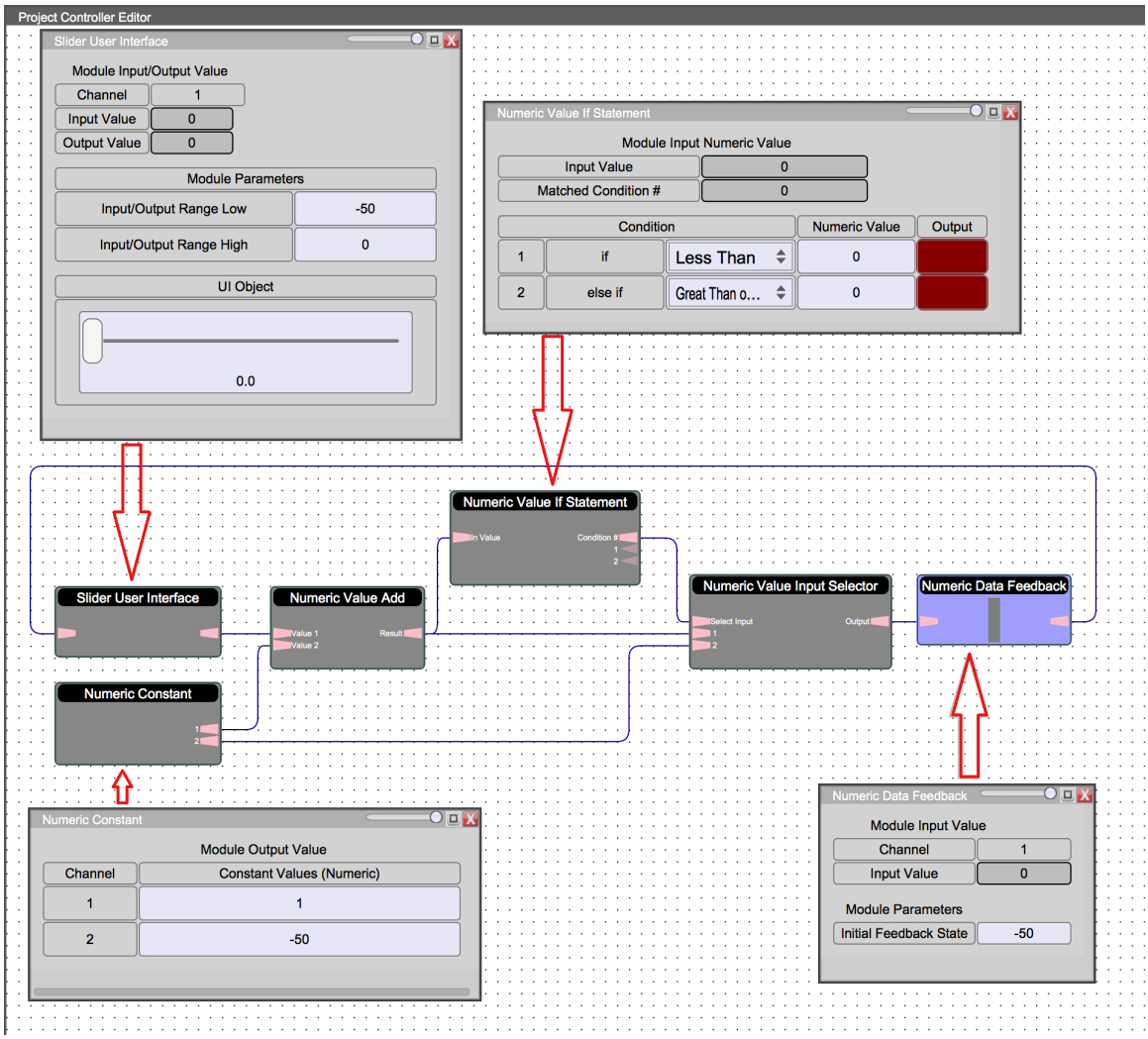
In this example it will demonstrate the use of data feedback module. The result of this example will toggle the UI Object state between ON and OFF every 5 seconds. The following is a more detail walk through on the design.

The first module is a User Interface button, and its initial state is being feed by the Logic Data Feedback initial feedback state. In this case I have set it as ON. This ON state will trigger the UI button to turn ON. It then feed the ON state to a logic delay module. So after 5 seconds, it will pass the ON state to Logic NOT. The Logic NOT module will change the ON state to OFF and store it to the feedback module.

On the next controller execution cycle, the input to Button User Interface object will become OFF. And this OFF state will be delayed for 5 seconds and then be converted by the Logic NOT module to ON. And stored in Logic data feedback module for next cycle use.

This will result in a self-running ON/OFF state change in UI object.

Numeric Condition Checking Example



In this example we will explain the use of Numeric IF Statement module and Numeric Value Input Selector module. The purpose of this example will provide a self-running slider UI object that will goes from -50dB up to 0dB in steps of 1dB. When it gets to 0dB, the slider will be reset back to -50dB and then ramp up again.

To start off with the processing flow. It will start at the Numeric Data Feedback module which has an initial value of -50. This numeric value will be fed to the Slider User Interface object to set it to 0%. This -50 numeric value will be passed to Numeric Value Add module which will add 1 to it. The result of -49 numeric value will then be passed to two different modules for further processing.

One of the module is a Numeric IF statement module which will check on the value to the conditions set in the module. If the value is Less Then 0, the Matched Condition pin will become “1”. If the value is Greater Then or Equal to 0, the Matched Condition pin will become “2”. This Matched Condition pin will be used by Numeric Input Selector

module to determine which input value to select. If the value is less then 0, it will pick the output from the Numeric Add module. If the value reached 0, it will pick the constant value from Numeric Constant module, which is -50.