# Solaro Universal Controller

## ユーザーガイド（V4.0.0）

## 目次


1 概要 3
2 Solaro DSP デバイスでのコントローラー機能の設定 4
3 基本データ型 6
4 モジュールの種類 7
5 処理フローとデータの有効性 8
5.1 処理フロー 8
5.2 データの有効性とデータ更新の仕組み 8
6 バイナリーデータの入力と表示 9
7 タッチパネルの利用 9
8 デザインのデバッグ 10
9 IO モジュール 11
9.1 Device Element – Logic 11
9.2 Device Element – Numeric 12
10 ネットワーク IO モジュール 13
10.1 TCP Server 13
10.2 UDP Server 13
10.3 TCP Client 14
10.4 UDP Client 15
11 ロジック処理モジュール 16
11.1 Logic AND 16
11.2 Logic OR 16
11.3 Logic NOT 17
11.4 Logic Exclusive OR 17
11.5 Logic Delay 18
11.6 Logic Timer 19
11.7 Logic Trigger Queue 20
11.8 Logic Flickering Filter 21
11.9 Logic SR-FlipFlop (Persistent) 21
12 数値処理モジュール 22
12.1 Numeric Constant 22
12.2 Numeric Value Range Convertor 22
12.3 Numeric Value IF Statement 23
12.4 Numeric Value Delay 24
12.5 Numeric Value Add 24
12.6 Numeric Value Subtract 25
12.7 Numeric Value Multiply 25
12.8 Numeric Value Division 26
12.9 Numeric Value Increment/Decrement 27
12.10 Numeric Value Holder 27
12.11 Numeric Value Queue 28
12.12 Numeric Value Input Selector 29
12.13 Numeric Value Output Selector 30
13 バイナリーデータ処理モジュール 30
13.1 Binary Data Constant 30
13.2 Binary Data Extractor (Fixed length) 31
13.3 Binary Data Extractor (Variable length) 31
13.4 Binary Data Combiner 32
13.5 Binary Data IF Statement 33
13.6 Binary Data Delay 33
13.7 Binary Data Holder 34
13.8 Binary Data Queue 35
13.9 Binary Data Input Selector 35
13.10 Binary Data Output Selector 36

14　データ変換モジュール ..... 37
14.1　Logic to Numeric Value Convertor ..... 37
14.2　Logic to Binary Data Convertor ..... 37
14.3　Numeric Value to Logic Convertor ..... 38
14.4　Numeric Value to Binary Data Convertor ..... 39
14.5　Numeric Value to Binary String Convertor ..... 39
14.6　Binary Data to Logic Convertor ..... 40
14.7　Binary Data to Numeric Value Convertor ..... 40
14.8　Binary String to Numeric Value Convertor ..... 41
15　データフィードバックモジュール ..... 42
15.1　Logic Data Feedback ..... 42
15.2　Numeric Data Feedback ..... 42
15.3　Binary Data Feedback ..... 43
16　UI モジュール ..... 43
16.1　Momentary ON/OFF Button User Interface ..... 43
16.2　Toggle ON/OFF Button User Interface ..... 44
16.3　Radio Button User Interface ..... 45
16.4　Slider User Interface ..... 46
17　Lua スクリプトモジュール ..... 47
17.1　Lua Script ..... 47
18　設計例 ..... 50
18.1　制御タイミングシーケンスの例 ..... 50
18.2　ロジックデータフィードバックの例 ..... 51
18.3　数値条件判定の例 ..... 52

# 1　概要

本書は、各 Solaro DSP デバイスに標準搭載されている新機能「Solaro Universal Controller」について説明するものです。サイト内のすべての機器を制御するために必要なのは Solaro DSP デバイスだけで、DSP による信号処理とコントローラー機能の両方を 1 台で提供できるため、制御ソリューションを非常に低コストで導入できます。

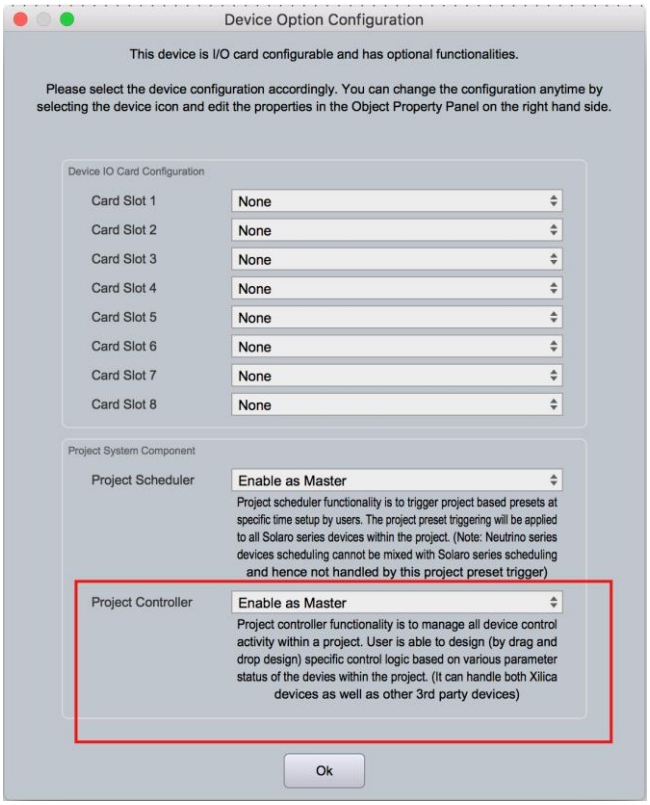
特別なハードウェアコストが不要であることに加え、Solaro Universal Controller のもう一つの狙いは、ドラッグ＆ドロップ操作だけで、サイト固有のロジックをすべてプログラムできる、きわめて使いやすい設定インターフェイスを提供することです。このインターフェイスと設計思想は DSP のプログラミングと非常によく似ています。DSP システム設計に慣れたサウンドエンジニアであれば、コントローラー機能のプログラミングにも同じ設計の考え方がそのまま通用します。お客様のサイトでコントローラーを稼働させるうえで、プログラミング経験は必要ありません。

シンプルなサイトであれば、制御したい各種機器（異なるメーカーの機器でも可）のコントロール要素をドラッグして配置し、それぞれのコントロール要素を配線して制御フローをつなぐだけで構成できます。より複雑なプロトコル制御が必要な場合には、数値およびバイナリーデータの操作モジュールを用意しており、独自の制御文字列を作成して、イーサネット経由で各機器へ送信できます。

バージョン 4.0.0 からは、コントローラーをさらに強力にするため、新たにスクリプト言語のサポートを導入しました。Lua スクリプト言語をベースに、独自の処理を記述できます。

## 2　Solaro DSP デバイスでのコントローラー機能の設定

各 Solaro DSP デバイスのソフトウェアには、コントローラー機能が標準で組み込まれています。コントローラー機能を使用するには、プロジェクトのマスターとなる Solaro DSP デバイスで「プロジェクトコントローラー（Project Controller）」を有効にするだけです。Solaro デバイスをドラッグして配置すると、カードスロット構成を設定するダイアログが表示されます。その下部に、この Solaro DSP デバイスをプロジェクトコントローラーのマスターに設定するオプションがあります。設定すると、Solaro DSP デバイスのアイコンに「Project Controller」ボタンが表示されます。




図：Device Option Configuration ダイアログ（Project Controller の設定項目）

または、プロジェクト内の任意の Solaro DSP デバイスをプロジェクトコントローラーのマスターとして選択することもできます。マスターデバイスは、設計段階であればいつでも切り替えられます。デバイスアイコンをクリックし、オブジェクトプロパティー領域で Project Controller オプションを選択してください。

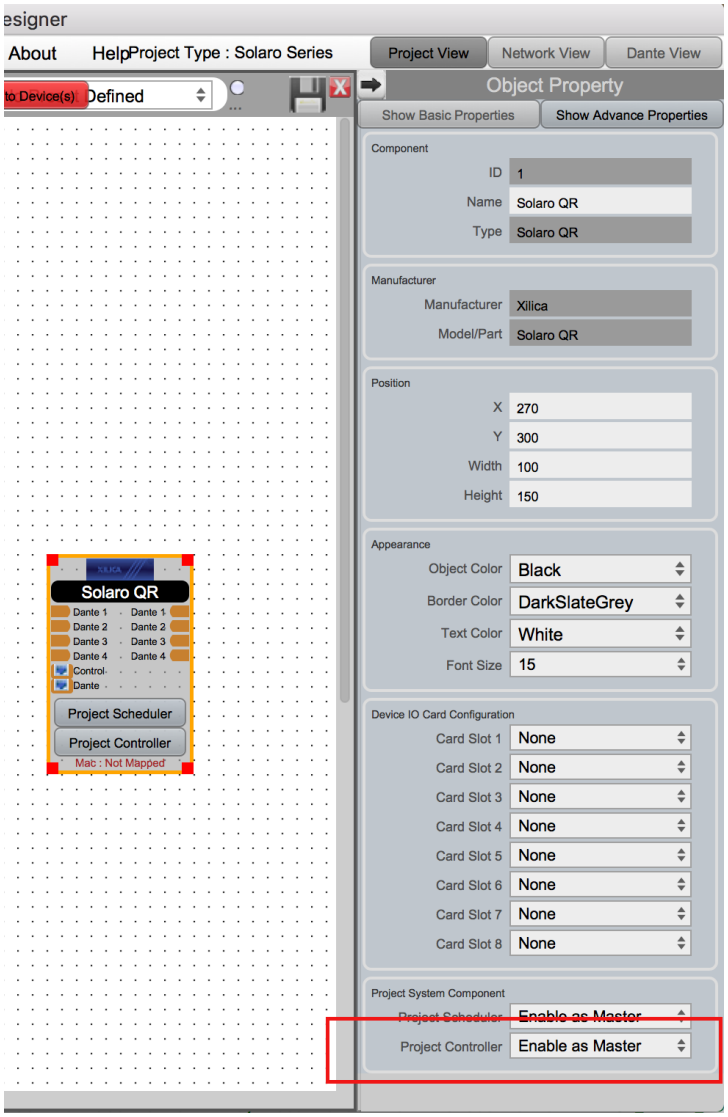
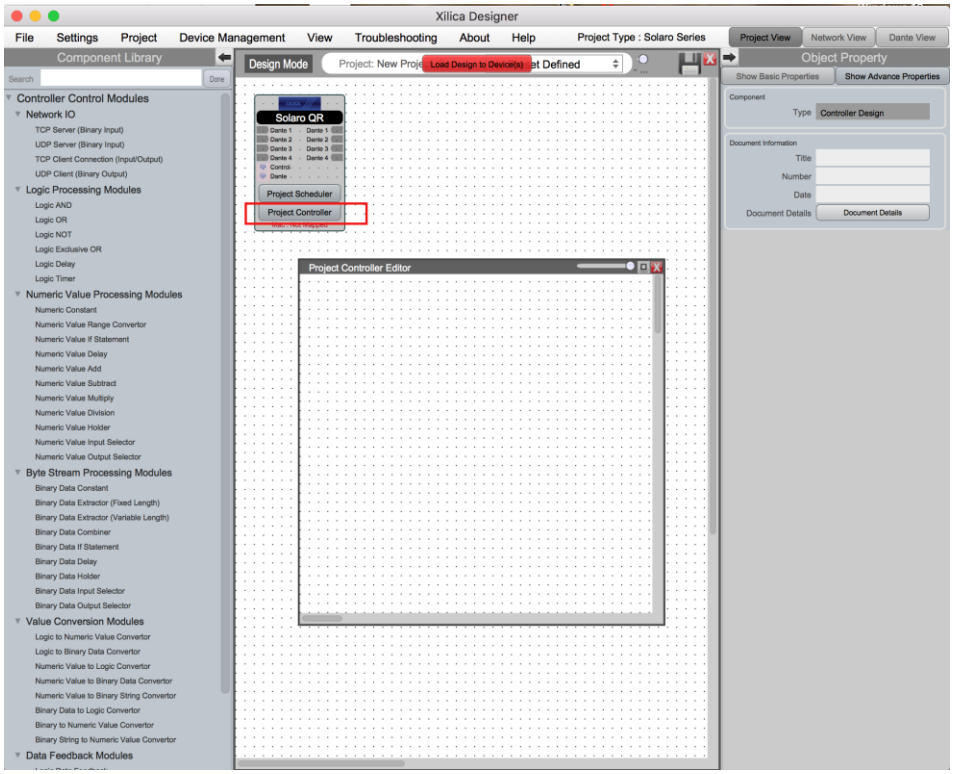



図:オブジェクトプロパティー領域での Project Controller オプション

コントローラーの設計を編集するには、デバイスアイコン上の「Project Controller」ボタンをクリックします。プロジェクトコントローラーエディターが表示され、コントロール要素やコントローラー処理モジュールをドラッグ＆ドロップして設計に追加できます。作成した設計はプロジェクト単位で保存されるため、コントローラーのマスターを別の DSP デバイスに切り替えても、プロジェクトの設計内容はそのまま維持されます。




図：Project Controller エディター

1 つのプロジェクトにおいて、コントローラーのマスターとなれる Solaro DSP デバイスは 1 台のみです。Xilica Designer ソフトウェアは、コントローラーマスターが必ず 1 台だけになるよう保護機能を備えています。

## 3　基本データ型

コントローラーの基本動作は、ネットワーク上のデバイスから取得したデータ（例：チャンネルのミュート状態などのパラメーター値）を各種処理モジュールに渡して処理し、処理後のデータをネットワーク上のリモートデバイスへ送信して制御する、というものです。

リモートデバイスから取得、またはリモートデバイスへ送信するデータは、次の 3 種類に分類されます。

・Logic Data（論理データ）：0 または 1（false または true）のいずれかのみ。

　◇例：デバイスのチャンネルのミュート状態。

・Numeric Data（数値データ）：整数、または小数を含む数値。

　◇例：デバイスのボリュームフェーダーの値。

・Binary Data（バイナリーデータ）：バイト列データ。文字列データもバイナリーデータ型として扱われます。

　◇例：ネットワーク上のデバイスから取得した制御用文字列。

各処理モジュールには、それぞれ異なる入力ピンと出力ピンがあり、各ピンは上記 3 つの基本データ型のいずれかに属します。あるデータ型から別のデータ型へ変換するには、データ型を適切に変換するための専用のデータ変換モジュールが必要です。

## 4　モジュールの種類

処理モジュールは、次のように分類されます。

**a. IO モジュール:**デバイスのパラメーター(Xilica デバイス、または Xilica Designer で定義されたユニバーサルコントロールデバイス)の入出力を扱います。

**b. Network IO モジュール:**ネットワークベースの生のバイナリメッセージの入出力処理を扱います。Xilica Designer のユニバーサルコントロールデバイス一覧に定義されていないデバイスを制御する場合に必要です。

**c. Logic 処理モジュール:**論理演算など、ロジックデータの処理を扱います。

**d. Numeric Value 処理モジュール:**ボリュームフェーダーの値など、数値の処理を扱います。

**e. Binary Data 処理モジュール:**バイトストリームから特定のバイトを抽出するなど、バイナリデータの処理を扱います。

**f. Data 変換モジュール:**異なるデータ型間の変換を扱います。

**g. Data Feedback モジュール:**通常、処理モジュールの出力値を、その前段にあるモジュールへフィードバックすることはできません。このようなループ動作を実現するために、Data Feedback モジュールが必要となります。

**h. UI モジュール:**ユーザーが操作して各種処理のトリガーとなる UI オブジェクトを提供します。(注:現在のベータ版では UI オブジェクトは非常に基本的なものです。正式リリースでさらに機能強化する予定です。)

**i. Lua Scripting モジュール:**ユーザー独自の処理を設計するための Lua スクリプト機能を提供します。Logic・Numeric・Binary のデータをモジュールに入力して処理でき、処理結果を Logic・Numeric・Binary のいずれかの形式でスクリプトモジュールから次の処理モジュールへ渡せます。

# 5　処理フローとデータの有効性

## 5.1 処理フロー

コントローラーの処理フローは DSP の処理フローと非常によく似ています。さまざまな処理モジュールをドラッグして配置し、モジュール間をデータフローの配線でつなぐことで、制御タスクを実現します。
設計が完成すると、コントローラーは 100 ミリ秒ごとに実行され、入力データ（リモートのサードパーティー製デバイスのデバイスデータなど）を取得します。処理は入力モジュールから始まり、処理フローに沿って順次データを処理していきます。処理されたデータがリモート制御ポイント（IO モジュールや Network IO モジュールなど）に到達すると、イーサネット経由でデータを送信し、リモートデバイスを制御します。
処理フローは一般に一方向ですが、Data Feedback モジュールと呼ばれる特殊な処理モジュールを用意しています。これは前回の実行サイクルの結果データを保持し、次の処理サイクルの入力データとして渡す機能を持ちます。これにより、前回の処理サイクルに対して応答するデータを返すためのフィードバック機構を実現できます。ネットワークトラフィックを扱う際に非常に有用です。たとえば、あるネットワーク上のバイナリーデータを受信して処理し、その結果を呼び出し元へ返したい場合、このコールバック機構が不可欠です。

## 5.2 データの有効性とデータ更新の仕組み

処理フローの実行中、データ値がまだ取得できていない場合（例：デバイスがオフラインのため、リモートデバイスのパラメーターを読み戻せていない場合）、そのデータポイントは「無効（invalid）」として扱われます。モジュールが処理を行うには、入力側のデータが有効でなければなりません。入力が無効な場合、それに対応する出力も無効となります。（注：この無効状態は初回起動時にのみ発生します。一度有効になれば、その後は有効のまま維持されます。）
Logic NOT モジュールのように、チャンネルごとに独立して処理を行うモジュールもあります。この場合、チャンネル 1 の入力データが無効であっても、影響を受けるのはチャンネル 1 の出力のみで、入力が有効な他のチャンネルは通常どおり処理されます。
データ値の有効性に加え、各データポイントには「更新済み（updated）」フラグも付与されています。入力値が更新されていない場合、処理モジュールは対応する出力値の処理もスキップします。この仕組みにより、処理フローの重複実行（特にリモートデバイスへネットワーク制御メッセージを送信する場合）を防ぎます。

## 6　バイナリーデータの入力と表示

バイナリーデータのフィールドでは、データを 16 進数(Hex)形式または ASCII 形式のいずれかで入力できます。データフィールドの右側には、データを「Hex」と「ASCII」のどちらで解釈・表示するかを示す(切り替える)ボタンがあります。




図:Hex 表示のバイナリーデータフィールド(例:01 02 ff)

Hex データの場合は、16 進のバイト値(0～9 および a～f)を直接入力します。入力値は 16 進数として解釈されます。




図:ASCII 表示(例:test)

データが ASCII の範囲内であれば、Hex 表示と ASCII 表示を切り替えることができます。範囲外のデータが含まれる場合は警告メッセージが表示されます。ASCII モードに切り替えるには、ASCII 範囲外のデータを削除する必要があります。

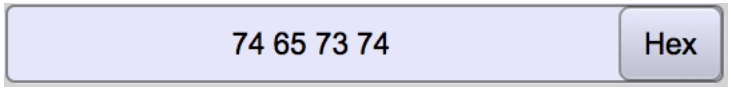



図:同じデータを Hex 表示に切り替えた例(74 65 73 74)

## 7　タッチパネルの利用

設計が完成したら、特定のステータスを表示したり、タッチパネルからコントローラー設計内の特定のボタン状態を変更したりしたい場合があります。そのためには、XTouch50、XTouch80、または XTouchApp をインストールした任意のデバイスが必要です。

パネルにコントロールを追加するには、設計内のコントロール要素を選択して Ctrl キーを押しながらクリックし、XTouch の設計画面へドラッグします。(これは DSP 側で XTouch の設計を行う際の操作とまったく同じです。)

XTouch の設計を作成したら、その設計を物理的な XTouch デバイスに直接マッピングできます。あるいは、XTouch の設計を Solaro DSP デバイスに関連付けることもできます。この場合、設計は Solaro DSP デバイス内に保存されます。後からネットワーク上の XTouch デバイスが Solaro DSP にコントロールパネルの設計を問い合わせ、(Xilica Designer ソフトウェアを使わずに)DSP デバイスから直接設計を取得できます。これは、スマートフォンなどのアドホックな制御デバイスにコントロールパネルを設定する際に非常に便利な方法です。

## 8　デザインのデバッグ

本コントローラーの設計にあたり、最も重要な使い勝手の課題の一つが「設計をどのようにデバッグするか」であると考えました。私たちの目標は、実行時にユーザーが手軽にデバッグ機能を利用できるようにすることです。

各モジュールのコントロールページには、リアルタイムのデータ値が表示されます。このリアルタイム表示により、データの流れと処理結果を追跡し、期待どおりに動作しているかを確認できます。

また、各モジュールのパラメーターは動作中に変更でき、処理モジュールは新しいパラメーター値に基づいて再処理を行います（Xilica Designer から設計を再ロードする必要はありません）。

## 9　IO モジュール

これらの IO モジュールは、左側の「Controller Control Modules List(コントローラーコントロールモジュール一覧)」には掲載されていません。これらのモジュールは、デバイス上の物理的なパラメーターを表すものだからです。IO モジュールを作成するには、デバイスのコントロールパネルからコントロール要素をコントローラーエディターへドラッグ＆ドロップするだけです。(例:Solaro DSP の Input モジュールのミュートボタン、または任意のユニバーサルコントロールデバイスのコントロールパネル上の要素。)

IO モジュールには、Logic タイプと Numeric タイプの 2 種類があります。コントロール要素をコントローラーエディターにドロップすると、次のようなモジュールアイコンが表示されます。

### 9.1 Device Element – Logic

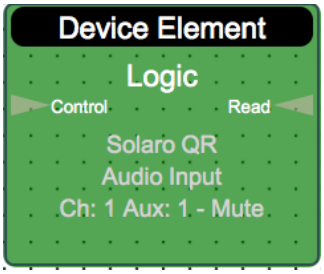



図:Device Element – Logic モジュール(例:Solaro QR / Audio Input / Ch: 1 Aux: 1 - Mute)

このモジュールには、Logic 入力ピンと Logic 出力ピンが 1 つずつあります。アイコン内には、このコントロール要素がどのデバイスのどのソースに由来するかが表示されます。

リモートデバイスのこのコントロール要素を制御したい場合は、このモジュールに Logic 入力値を与えるだけです。ON/OFF の値を入力すると、このモジュールが必要なネットワークプロトコル処理をすべて行い、入力ピンの値に応じてリモートのデータポイントを制御します。

逆に、リモートデバイス側で値が変更された場合は、このモジュールの出力ピンにネットワークから読み戻した新しい値が反映されます。

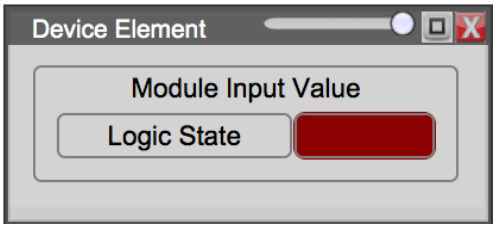



図:Device Element – Logic のモジュールコントロールページ

## 9.2 Device Element – Numeric

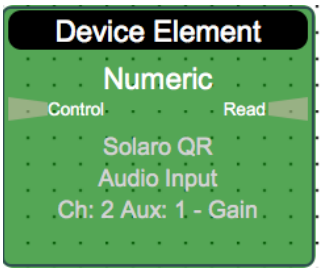



図:Device Element – Numeric モジュール(例:Solaro QR / Audio Input / Ch: 2 Aux: 1 - Gain)

このモジュールには、Numeric 入力ピンと Numeric 出力ピンが 1 つずつあります。アイコン内には、このコントロール要素がどのデバイスのどのソースに由来するかが表示されます。

リモートデバイスのこのコントロール要素を制御したい場合は、このモジュールに数値を与えるだけです。このモジュールが必要なネットワークプロトコル処理をすべて行い、入力ピンの値に応じてリモートのデータポイントを制御します。

逆に、リモートデバイス側で値が変更された場合は、このモジュールの Numeric 出力ピンにネットワークから読み戻した新しい値が反映されます。

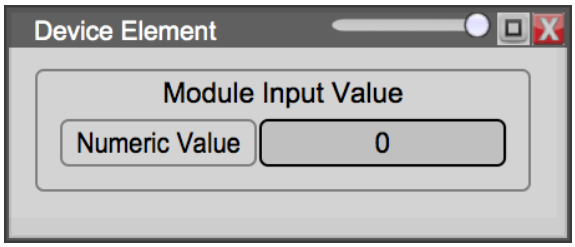



図:Device Element – Numeric のモジュールコントロールページ

**注:**

- リモートデバイスのパラメーターを制御することだけが目的の場合は、これらのモジュールの入力ピンのみを接続してください。出力ピンが未接続の場合、読み戻されたデータはシステムに無視されます。逆に、リモートパラメーターの現在の状態を知りたいだけの場合は、出力ピンのみを接続し、読み戻した値を処理に利用します。入力ピンは未接続のままで構いません。この場合、システムがリモートのデータポイントへ制御コマンドを送信することはありません。
- 同じコントロール要素を、コントローラーエディターに複数回ドラッグして配置することもできます。同じリモートデバイスのパラメーターを指す複数のモジュールは、それぞれ独立して動作しますが、最終的にはすべて同じリモートデバイスのパラメーターに対して制御、またはデータの読み取りを行います。

# 10 ネットワーク IO モジュール

## 10.1 TCP Server

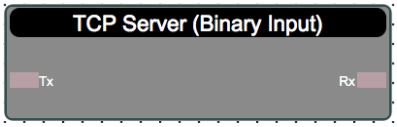



図:TCP Server (Binary Input)モジュール(ピン:Tx／Rx)

TCP Server モジュールは、データの送信と受信の機能を提供します。接続中のクライアントへ送信するデータ用の Binary 入力ピン(Tx)と、このサーバーが受信したすべてのバイナリーデータを出力する Binary 出力ピン(Rx)を備えています。

サーバーを設定するには、ポート番号を入力します(このサーバーの IP アドレスは Solaro DSP デバイスの IP アドレスになります)。接続がアイドル状態になったときにクライアント接続を切断するように設定することも、一度接続されたら接続を維持し続けるように設定することもできます。

クライアントへ Tx データを返送する際には、2 つのオプションがあります。直前にこのサーバーへデータを送信してきた最後のクライアントにのみ送信するか、接続中のすべてのクライアントに Tx データを送信するかを選べます。

受信データは TCP パケットのフレーミングに基づきます。各 TCP フレーミングデータは 1 つのバイナリーストリームとして扱われ、このサーバーモジュールの出力を 1 回トリガーします。

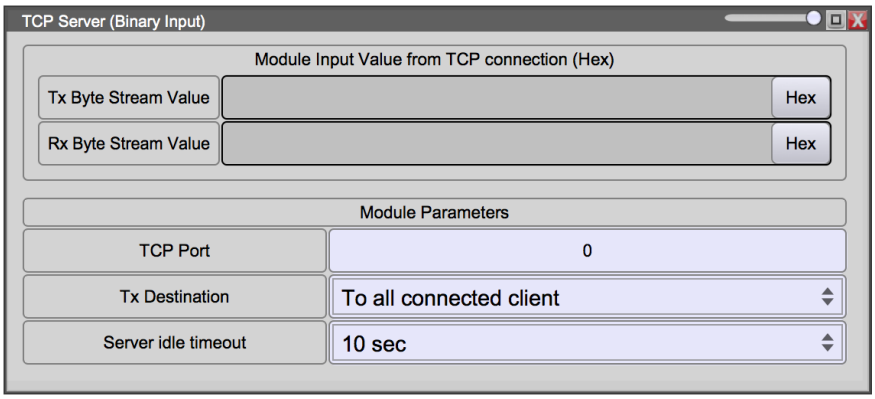



図:TCP Server のモジュールコントロールページ(パラメーター:TCP Port／Tx Destination／Server idle timeout)

## 10.2 UDP Server

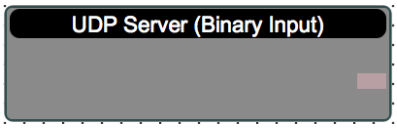



図:UDP Server (Binary Input)モジュール

UDP Server モジュールには Rx 出力ピンしかありません。このサーバーポートで UDP パケットを受信すると、このサーバーモジュールの出力がトリガーされます。

この UDP Server モジュールを設定するには、このサーバーの UDP ポート番号を入力します。サーバーの IP アドレスは Solaro DSP デバイスの IP アドレスと同じになります。

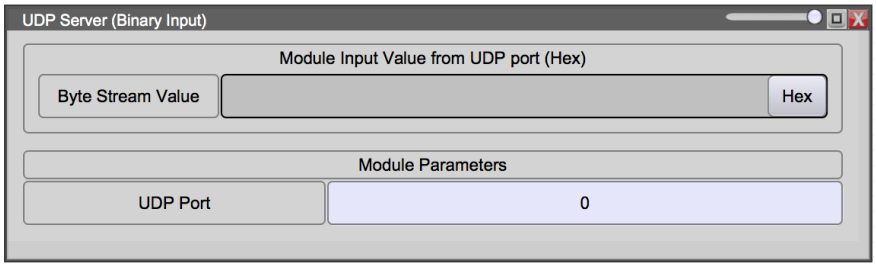



図:UDP Server のモジュールコントロールページ

## 10.3 TCP Client

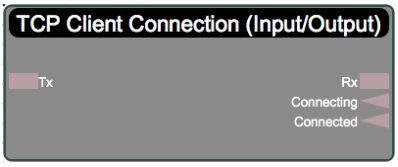



図:TCP Client Connection (Input/Output)モジュール(ピン:Tx／Rx／Connecting／Connected)

TCP Client モジュールには、リモートの TCP サーバーへ送信するバイナリーデータを受け取る Tx 入力ピンが 1 つあります。また、サーバー側から返送されたデータをこのクライアントへ反映する Rx 出力ピンも備えています。

さらに、Logic 出力ピンが 2 つあります。1 つは TCP クライアントが TCP 接続を確立しているかどうかを示します。もう 1 つは、バイナリーデータの送信によって新しい TCP 接続が作成されたかどうかを示します(新しい TCP 接続が作成された場合、Connecting ピンが true になります)。

このモジュールを設定するには、接続先となるリモート TCP サーバーの IP アドレスを入力します。また、このクライアントが接続する TCP ポート番号も設定します。

TCP クライアントを常時接続にするか、アイドル時に切断するかを選択するオプションもあります。

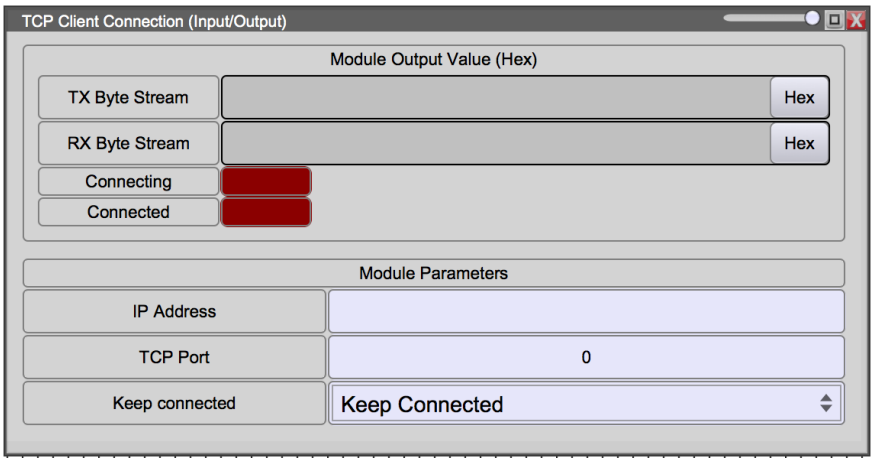



図:TCP Client のモジュールコントロールページ(パラメーター:IP Address／TCP Port／Keep connected)

## 10.4 UDP Client

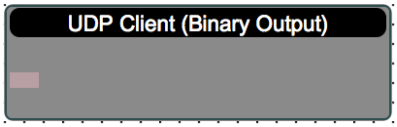



図:UDP Client (Binary Output)モジュール

UDP Client モジュールには、バイナリーデータを受け取ってリモートの UDP サーバーへ送信する Tx 入力ピンが 1 つあります。このモジュールを設定するには、リモート UDP サーバーの IP アドレスとポート番号を入力します。メッセージの送信先となるリモートポートに加え、送信する UDP メッセージで使用するローカルポート番号も設定できます。(デバイスによっては、UDP メッセージのローカルポートを基準に UDP の返信を送ってくるものがあるためです。)

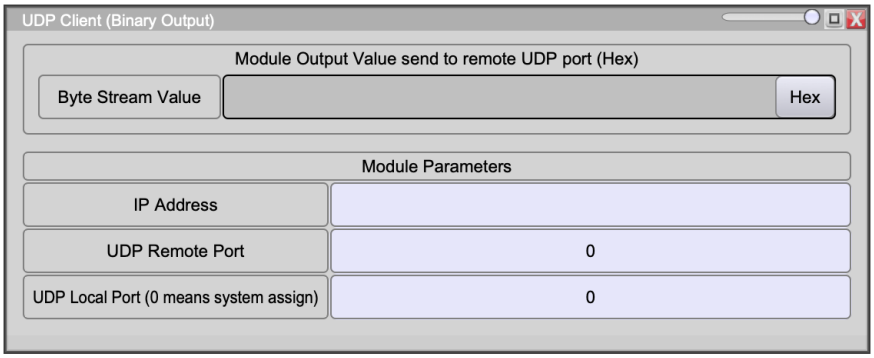



図:UDP Client のモジュールコントロールページ(パラメーター:IP Address／UDP Remote Port／UDP Local Port(0 はシステムによる自動割り当て))

# 11 ロジック処理モジュール

## 11.1 Logic AND

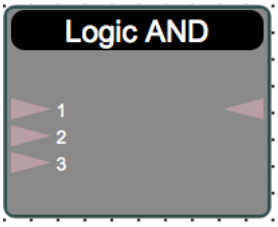



図:Logic AND モジュール

Logic AND モジュールは、すべての入力ピンに対して論理積(AND)演算を行い、その結果を出力ピンに送ります。すべての入力が ON(true)の場合のみ出力は ON(true)になり、いずれかの入力が OFF(false)であれば出力は OFF(false)になります。なお、出力データが有効(valid)になるのは、すべての入力ピンのデータが有効な場合のみです。このモジュールにはパラメーター設定はありません。

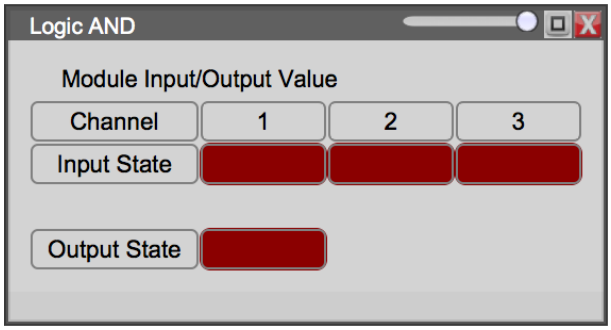



図:Logic AND のモジュールコントロールページ(Input State／Output State)

## 11.2 Logic OR

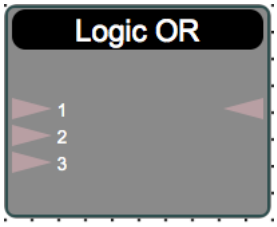



図:Logic OR モジュール

Logic OR モジュールは、すべての入力ピンに対して論理和(OR)演算を行い、その結果を出力ピンに送ります。いずれか 1 つ以上の入力が ON(true)であれば出力は ON(true)になり、すべての入力が OFF(false)の場合のみ出力は OFF(false)になります。なお、出力データが有効(valid)になるのは、すべての入力ピンのデータが有効な場合のみです。このモジュールにはパラメーター設定はありません。

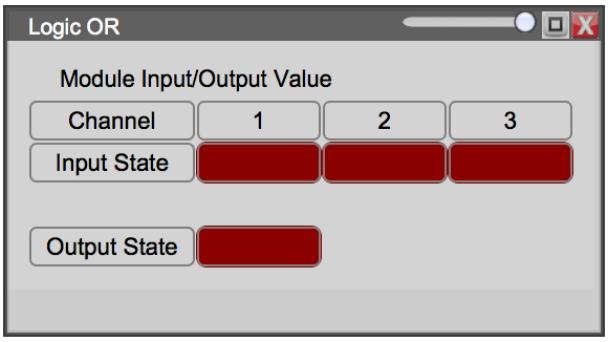



図:Logic OR のモジュールコントロールページ

## 11.3 Logic NOT

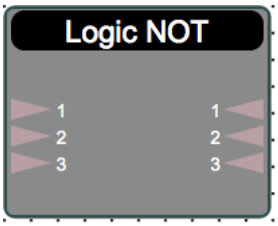



図:Logic NOT モジュール

Logic NOT モジュールは、各チャンネルに対して独立に論理否定(NOT)演算を行います。入力が ON(true)の場合、出力は OFF(false)になり、入力が OFF(false)の場合、出力は ON(true)になります。なお、チャンネル n の出力データは、チャンネル n の入力データが有効(valid)になった時点で有効になります。

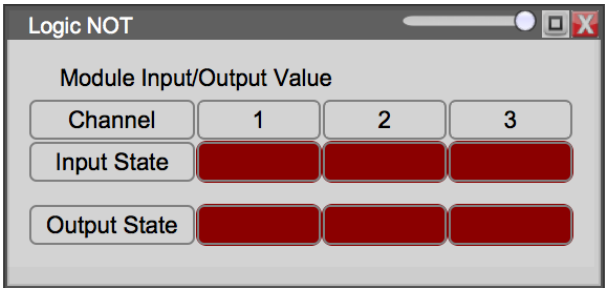



図:Logic NOT のモジュールコントロールページ

## 11.4 Logic Exclusive OR




図:Logic Exclusive OR モジュール

Logic Exclusive OR モジュールは Logic OR モジュールに似ていますが、排他的論理和(XOR)演算を行う点が異なります。OR が「いずれか 1 つ以上の入力が ON なら出力が ON」であるのに対し、XOR は「入力の状態が一致しないときに出力が ON になる」演算です。たとえば 2 つの入力で考えると、どちらか一方だけが ON のときに出力が ON、両方が ON または両方が OFF のときに出力が OFF になります。このモジュールにはパラメーター設定はありません。

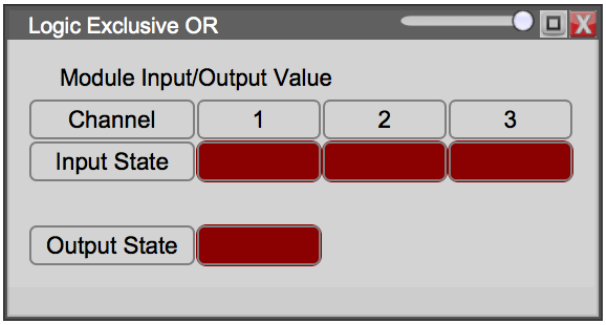



図:Logic Exclusive OR のモジュールコントロールページ

## 11.5 Logic Delay

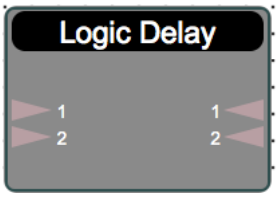
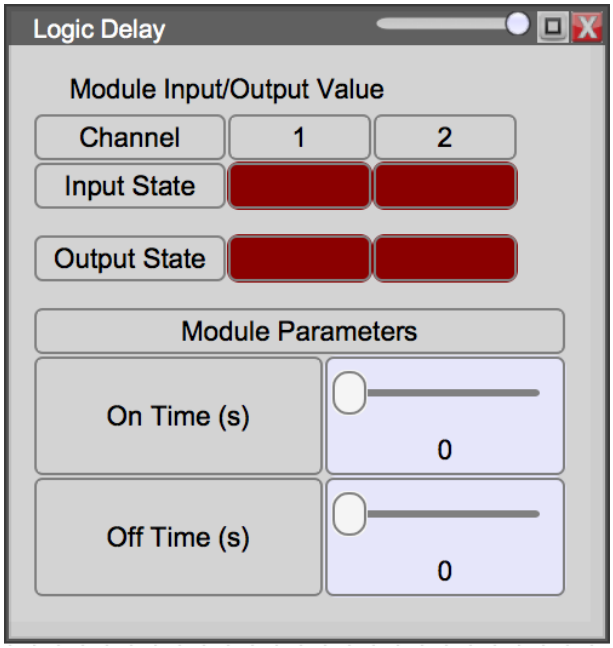


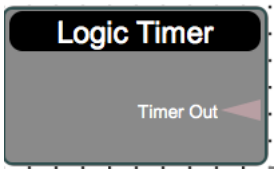

図:Logic Delay モジュール

Logic Delay モジュールは、入力ピンの状態を、一定の秒数だけ遅らせて出力ピンの状態として渡します。遅延の設定は、ON に変化するときと OFF に変化するときで別々に指定できます。各チャンネルは他のチャンネルとは独立して動作します。




図：Logic Delay のモジュールコントロールページ（パラメーター：On Time／Off Time）

## 11.6 Logic Timer




図：Logic Timer モジュール

Logic Timer モジュールは、周期トリガー時間に基づいて、周期的な状態変化パルス（瞬間的に論理 ON になり、その後 OFF に戻る）を出力します。Start Time の状態が OFF の場合、周期トリガーパルスは停止します。

このモジュールは、定期的に実行したい処理がある場合に便利です。このタイマーを使って、周期的な動作をトリガーできます。

使用例：外部機器の状態を定期的に確認したい場合、このタイマーの出力をきっかけとして、TCP Client または UDP Client モジュールから状態問い合わせのコマンドを一定間隔で送信できます。Start Time に会議システムの ON/OFF 状態を接続すれば、システムの使用中だけ問い合わせを行う設計にできます。

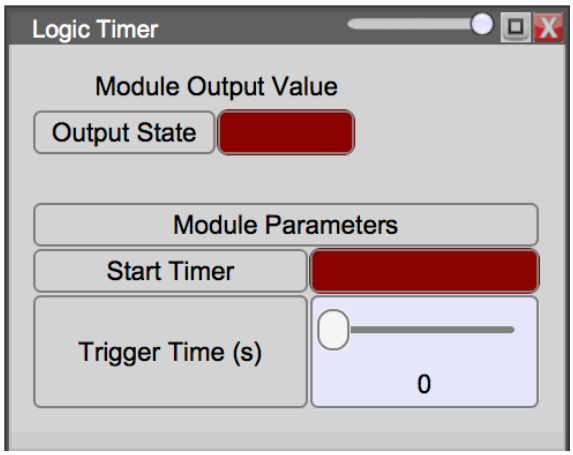



図:Logic Timer のモジュールコントロールページ(パラメーター:Trigger Time)

## 11.7 Logic Trigger Queue

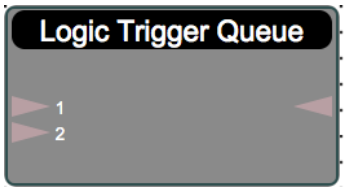



図:Logic Trigger Queue モジュール

Logic Trigger Queue モジュールは、入力信号の立ち上がりエッジを検出し、すべての入力チャンネルにおける立ち上がりエッジイベントの数をカウントします。その後、すべてのイベントをキューに入れ、出力側へ連続したエッジトリガーパルス信号として出力します。出力されるパルスの幅は調整できます。
信号フローの複数の箇所を合流させ、後続の経路で処理をトリガーしたい場合に便利です。

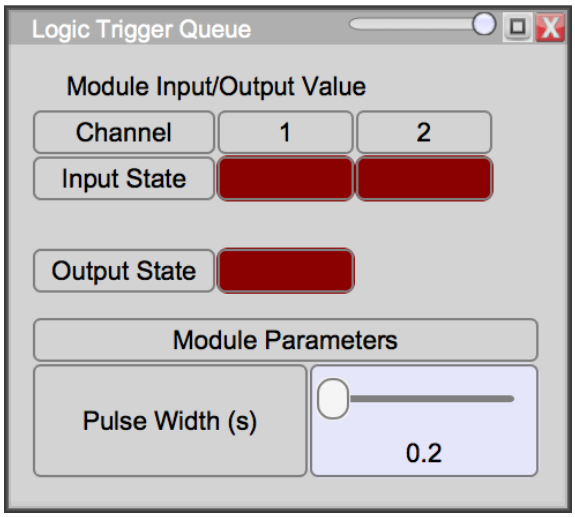



図:Logic Trigger Queue のモジュールコントロールページ(パラメーター:Pulse Width)

## 11.8 Logic Flickering Filter

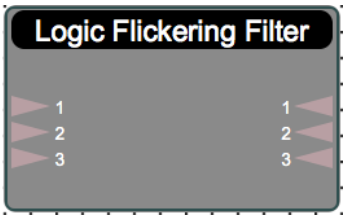



図:Logic Flickering Filter モジュール

このモジュールは、論理のバウンシング(接点のチャタリング)による不要な状態変化のトリガーを除去するために使用します。信号が OFF から ON に変化した場合、フィルター遅延時間のあいだ信号が ON のまま維持されることを確認してから、実際の状態変化とみなします。状態変化がフィルター遅延時間の間続かなかった場合、その変化は無視されます。ON から OFF への変化の場合も同様です。

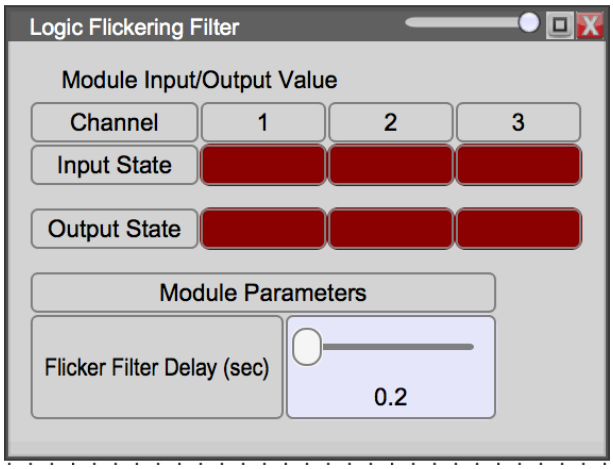



図:Logic Flickering Filter のモジュールコントロールページ(パラメーター:Flicker Filter Delay)

## 11.9 Logic SR-FlipFlop (Persistent)

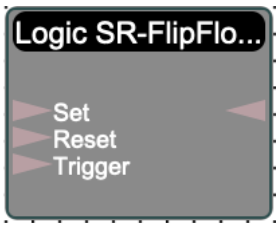



図:Logic SR-FlipFlop (Persistent)モジュール(ピン:Set/Reset/Trigger)

この Logic FlipFlop モジュールは、フリップフロップに「Trigger」信号を与えることで論理状態をトグル(反転)させる仕組みを提供します。また、「Set」ピンと「Reset」ピンを使って、フリップフロップの状態を直接変更することもできます。

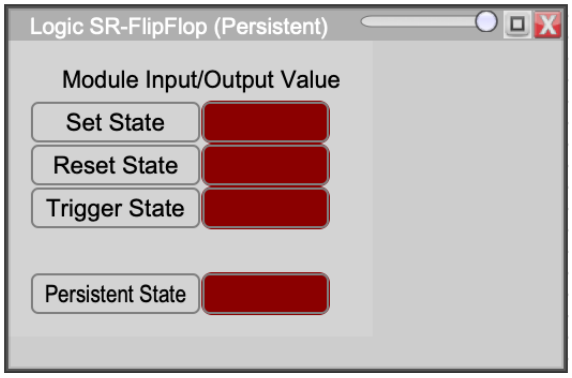



図:Logic SR-FlipFlop のモジュールコントロールページ

# 12 数値処理モジュール

## 12.1 Numeric Constant

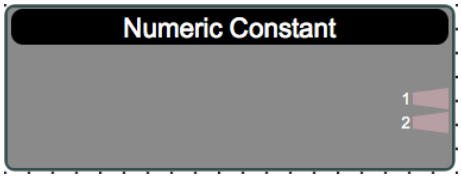



図:Numeric Constant モジュール

Numeric Constant モジュールは、処理設計に定数値を供給します。この値には、整数または小数のどちらも指定できます。

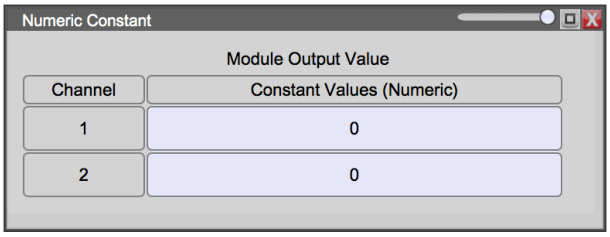



図:Numeric Constant のモジュールコントロールページ

## 12.2 Numeric Value Range Convertor

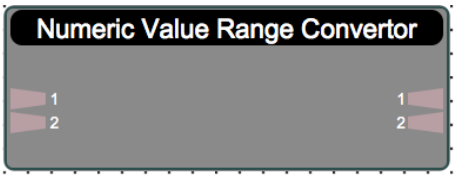



図:Numeric Value Range Convertor モジュール

このモジュールは、値の範囲を再マッピングします。チャンネルごとに入力値を取得し、入力範囲に基づく割合(パーセンテージ)に変換します。続いて、その割合を出力範囲の設定に従って展開します。このモジュールの各チャンネルは互いに独立しています。

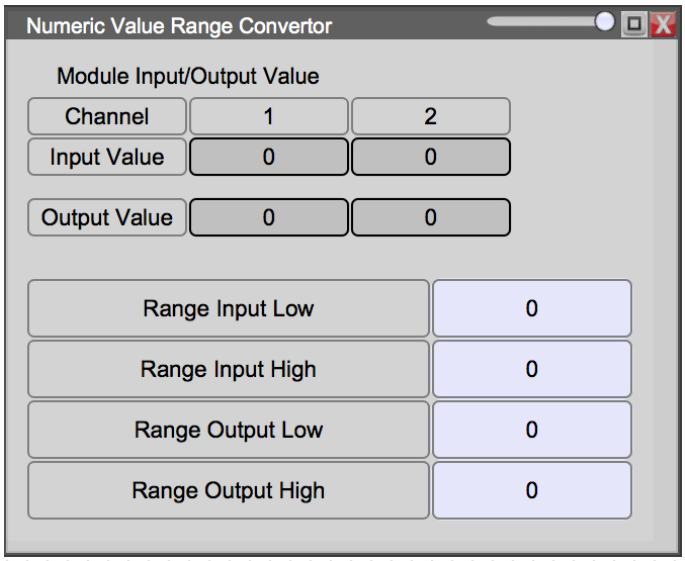



図:Numeric Value Range Convertor のモジュールコントロールページ(パラメーター:Range Input Low／High、Range Output Low／High)

## 12.3 Numeric Value IF Statement

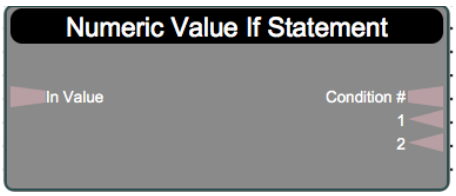



図:Numeric Value IF Statement モジュール(ピン:In Value／Condition #)

Numeric Value IF Statement モジュールは、入力値に対する判定条件を提供します。このモジュールは、チャンネルの順に条件判定を行います。一致する条件が見つかった時点で、そのチャンネルの Logic 出力を ON にし、一致した条件の番号(Matched Condition #)も出力ピンに出力します。

条件判定には、Equal(等しい)、NOT Equal(等しくない)、Greater Than(より大きい)、Greater Than or Equal(以上)、Less Than(より小さい)、Less Than or Equal(以下)などを使用できます。

入力値がどの条件にも一致しない場合、すべての出力状態は OFF となり、Matched Condition # は「0」になります。

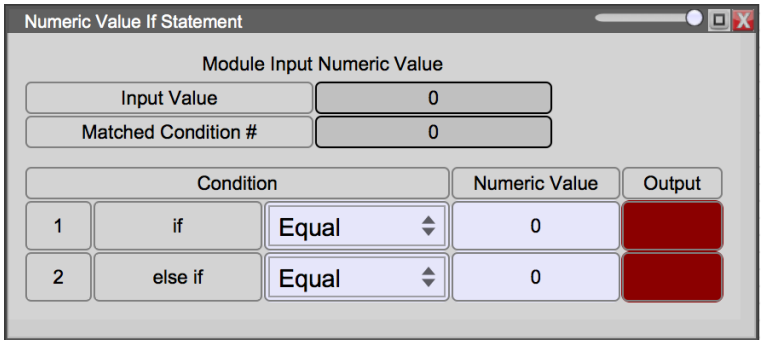



図:Numeric Value IF Statement のモジュールコントロールページ

## 12.4 Numeric Value Delay

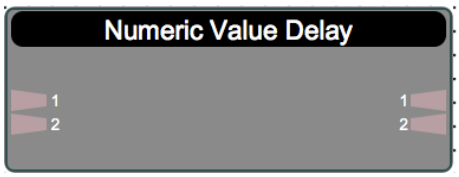



図:Numeric Value Delay モジュール

Numeric Value Delay モジュールは Logic Delay モジュールに似ていますが、数値を扱う点が異なります。入力された数値を、一定の秒数だけ遅らせて出力ピンに渡します。各チャンネルは他のチャンネルとは独立して動作します。

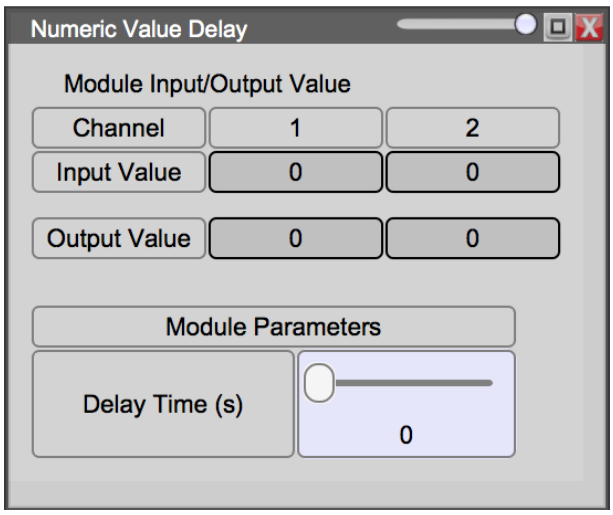



図:Numeric Value Delay のモジュールコントロールページ(パラメーター:Delay Time)

## 12.5 Numeric Value Add

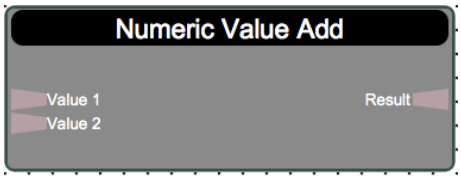



図:Numeric Value Add モジュール

Numeric Value Add モジュールは、すべての入力ピンの値を加算し、その結果を出力ピンに送ります。このモジュールは、すべての入力ピンが有効な値を持っていなければ、出力値を処理しません。

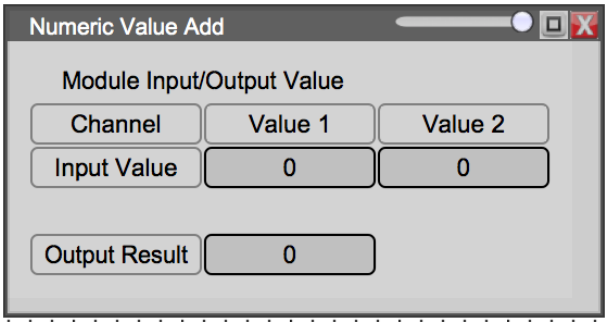



図:Numeric Value Add のモジュールコントロールページ

## 12.6 Numeric Value Subtract

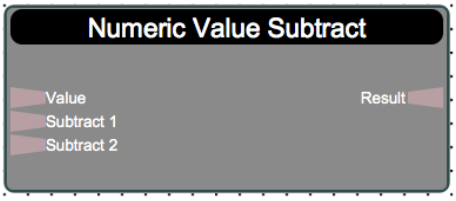



図:Numeric Value Subtract モジュール

Numeric Value Subtract モジュールは、入力値(Value)から、他の入力ピン(Subtract 1、Subtract 2…)の値を引きます。引き算の結果は出力ピンに送られます。このモジュールは、すべての入力ピンが有効な値を持っていなければ、出力値を処理しません。

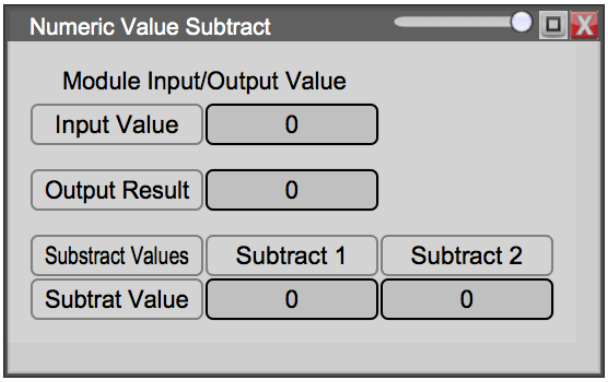



図:Numeric Value Subtract のモジュールコントロールページ

## 12.7 Numeric Value Multiply

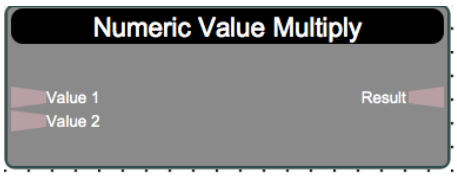



図:Numeric Value Multiply モジュール

Numeric Value Multiply モジュールは、すべての入力ピンの値を掛け合わせ、その結果を出力ピンに送ります。このモジュールは、すべての入力ピンが有効な値を持っていなければ、出力値を処理しません。

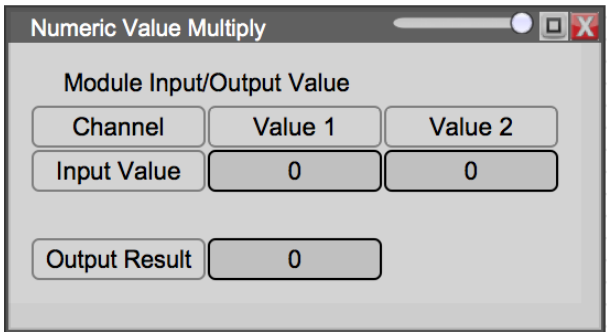



図:Numeric Value Multiply のモジュールコントロールページ

## 12.8 Numeric Value Division

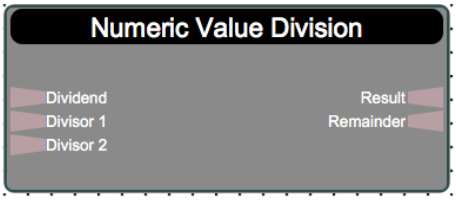



図:Numeric Value Division モジュール(ピン:Dividend／Divisor 1／Divisor 2、出力:Result／Remainder)

Numeric Value Division モジュールは、被除数(Dividend)の値を、すべての除数(Divisor)で割ります。割り算の結果(商)と余りは、それぞれ対応する出力ピンに送られます。このモジュールは、すべての入力ピンが有効な値を持っていなければ、出力値を処理しません。

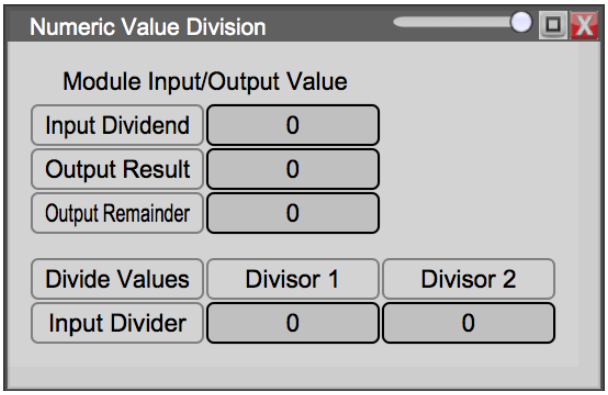



図:Numeric Value Division のモジュールコントロールページ

## 12.9 Numeric Value Increment/Decrement




図:Numeric Value Increment / Decrement モジュール

このモジュールには、Logic 入力ピンが 3 つあります。「Reset」ピンは、値を指定した初期値にリセットします。「Inc」ピンは、値をステップ幅の分だけ増加させます。「Dec」ピンは、値をステップ幅の分だけ減少させます。下限(Low)と上限(High)を指定すると、モジュールの結果がそれらの限度を下回ったり上回ったりしないようにできます。

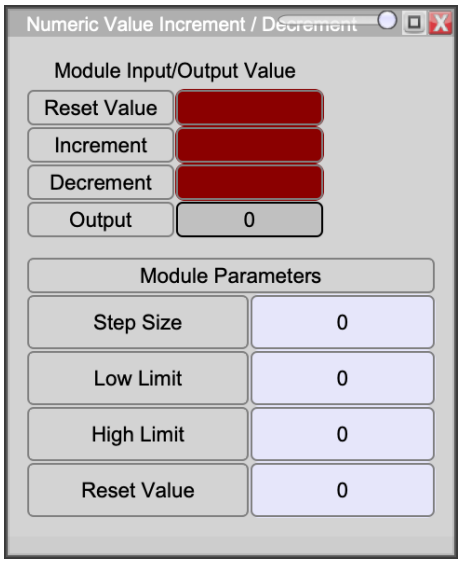



図:Numeric Value Increment/Decrement のモジュールコントロールページ(パラメーター:Step Size／Low Limit／High Limit／Reset Value)

## 12.10 Numeric Value Holder

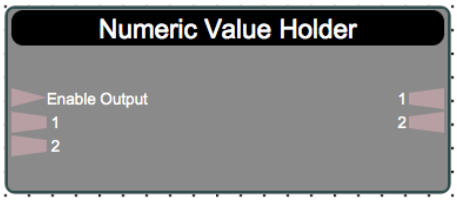



図:Numeric Value Holder モジュール(ピン:Enable Output)

Numeric Value Holder モジュールは、値のスナップショットを取り込むためのバッファーを提供します(値のデータを保持します)。キャプチャー入力ピンが変化すると、全チャンネルの入力ピンの値を読み取り、その

入力値を対応する出力ピンに送ります。各チャンネルは互いに独立しています。処理されるのは有効な入力ピンのみで、その他の無効なピンは無効のままとなります。

トリガーモードは 2 種類あります。「Rising edge trigger(立ち上がりエッジトリガー)」では、「Enable Output」ピンが OFF から ON に変化したときにのみ入力値を取り込みます。ピンが ON から OFF に変化したときは何も起こりませんが、これにより、次回の取り込み動作が可能な状態になります。

2 つ目は「Level trigger(レベルトリガー)」モードです。「Enable Output」が ON の間、入力値を継続的に取り込み、出力側へ渡し続けます。

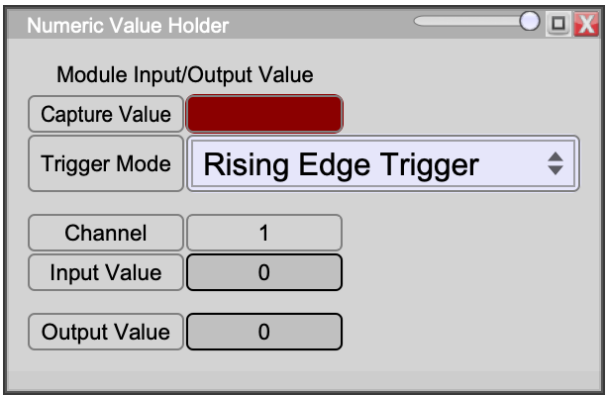



図:Numeric Value Holder のモジュールコントロールページ(パラメーター:Trigger Mode)

## 12.11 Numeric Value Queue

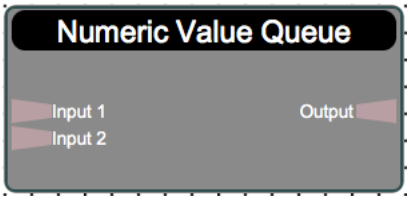



図:Numeric Value Queue モジュール

Numeric Value Queue モジュールは、変更された(前段の処理で更新された)すべての入力値を取得し、順番待ちのキューに入れます。これらの値は、後続のモジュールへ 1 つずつ送られます。処理サイクルごとにキュー内の次の値を 1 つずつ送出し、キュー内のすべての要素が処理されるまで続けます。

このモジュールは、複数の処理経路があり、各経路で処理が行われた後、それぞれの経路の結果を 1 つずつ処理させたい場合に特に便利です(例:同じ TCP 接続を通じてリモートデバイスへ送信する場合)。

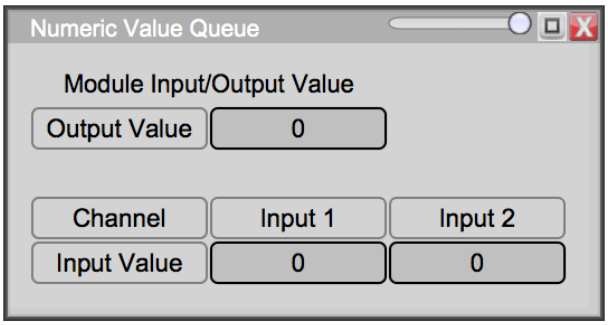



図:Numeric Value Queue のモジュールコントロールページ

## 12.12 Numeric Value Input Selector

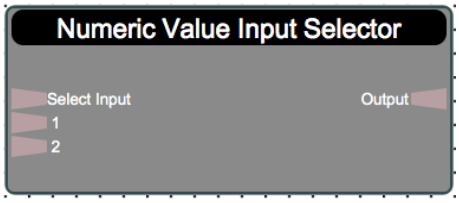



図:Numeric Value Input Selector モジュール(ピン:Select Input)

Numeric Value Input Selector モジュールは、出力ピンを特定の入力ピンに接続します。接続されると、その入力ピンでのデータ値の変化はすべて出力ピンに渡されます。

Select Input の値は、1～n の範囲で指定します。Select Input の値が「0」の場合、出力はどこにも接続されず、データの更新は行われません。Select Input の値が利用可能な入力チャンネル数より大きい場合は、最後の入力チャンネルが選択されます。

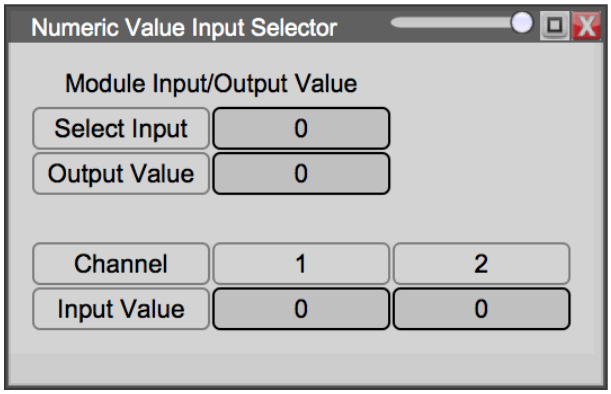



図:Numeric Value Input Selector のモジュールコントロールページ

## 12.13 Numeric Value Output Selector

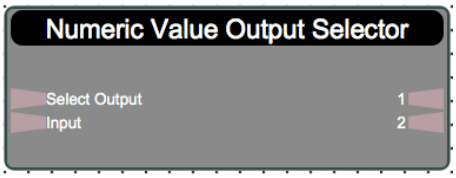



図:Numeric Value Output Selector モジュール(ピン:Select Output／Input)

Numeric Value Output Selector モジュールは Input Selector に似ていますが、選択を出力側で行う点が異なります。

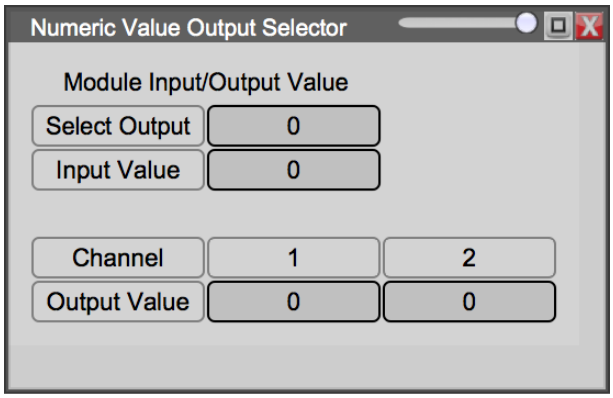



図:Numeric Value Output Selector のモジュールコントロールページ

# 13 バイナリーデータ処理モジュール

## 13.1 Binary Data Constant

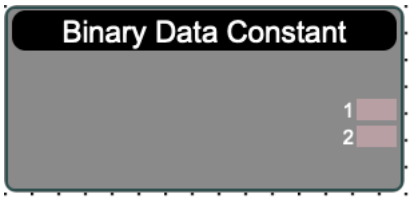



図:Binary Data Constant モジュール

Binary Data Constant モジュールは、処理設計に定数のバイナリー値を供給します。入力されたバイナリーデータは、対応する出力ピンに送られます。

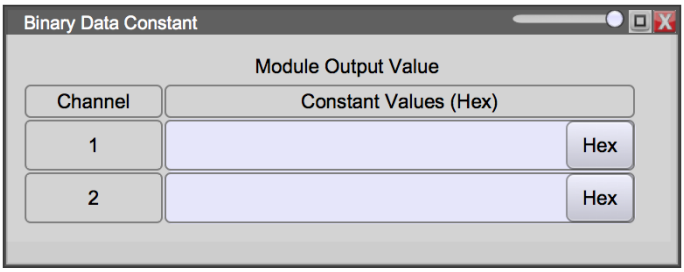



図:Binary Data Constant のモジュールコントロールページ(Hex 表示)

## 13.2 Binary Data Extractor (Fixed length)

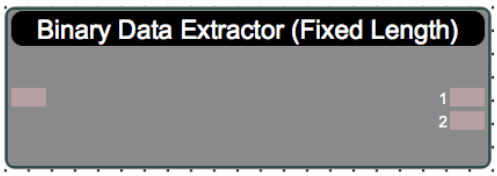



図:Binary Data Extractor (Fixed Length)モジュール

Binary Data Extractor モジュールは、バイナリーデータの特定の部分を抽出して出力するための仕組みを提供します。このモジュールは固定長で動作します。チャンネルごとに、データ配列から抽出する開始バイトと終了バイトを指定できます。開始バイトまたは終了バイトが範囲外の場合、データは抽出されません。

注:終了バイトの位置も、抽出されるデータに含まれます。

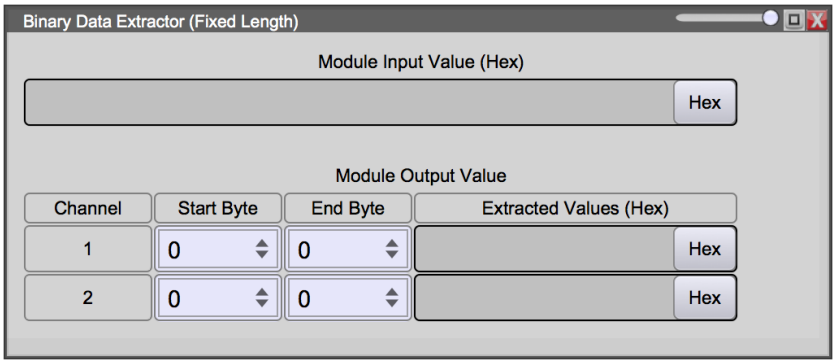



図:Binary Data Extractor (Fixed Length)のモジュールコントロールページ(Start Byte／End Byte)

## 13.3 Binary Data Extractor (Variable length)

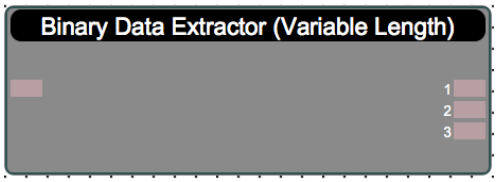



図:Binary Data Extractor (Variable Length)モジュール

Binary Data Extractor (Variable length)は、固定長の Binary Data Extractor に似ていますが、バイナリーストリームの特定の部分を可変長として解釈するかどうかを選択できる点が異なります。このモジュールは、バイナリーデータの先頭から、チャンネル 1 から順に各チャンネルを解釈していきます。前のチャンネルの解釈が終わると、残りのデータを次のチャンネルの解釈に使用します。

各チャンネルは、固定長または可変長として解釈できます。可変長の場合は、終端バイト(Termination Byte)を探して、そのチャンネルのデータの終わりを判断します。

バイナリーデータの中に終端バイトが見つからない場合は、入力データ全体がそのチャンネルの結果として渡されます。この場合、次のチャンネルで処理するデータは残りません。

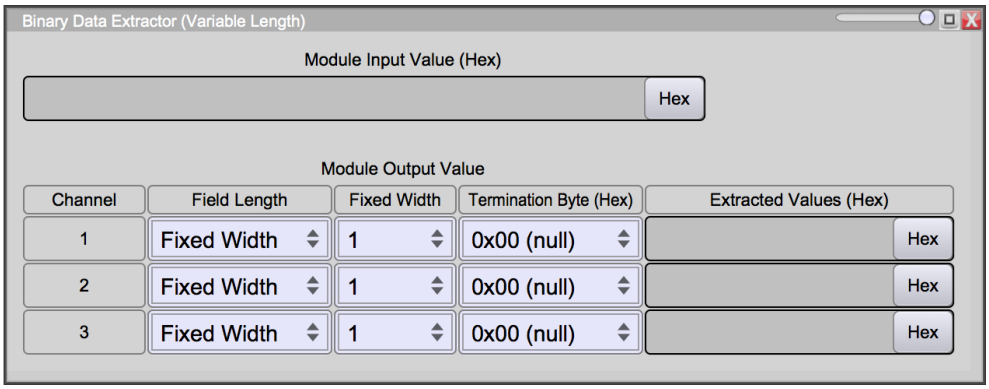



図:Binary Data Extractor (Variable Length)のモジュールコントロールページ(Field Length／Fixed Width ／Termination Byte)

## 13.4 Binary Data Combiner

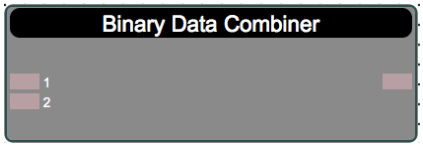



図:Binary Data Combiner モジュール

Binary Data Combiner モジュールは、すべての入力バイナリーデータを連結して、1 つの大きなバイナリーデータにします。データを連結する順序は、ピンの並び順に従います。

このモジュールは、制御フローから得られたさまざまな値を組み合わせて、1 つの制御文字列を作成したい場合に便利です。

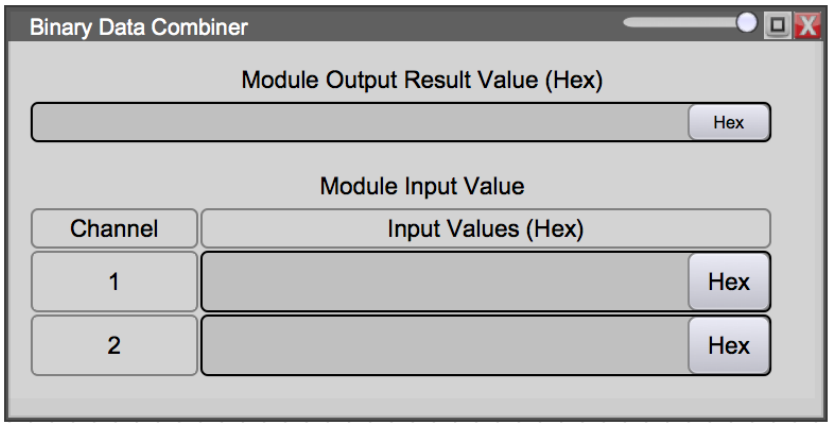



図:Binary Data Combiner のモジュールコントロールページ

## 13.5 Binary Data IF Statement

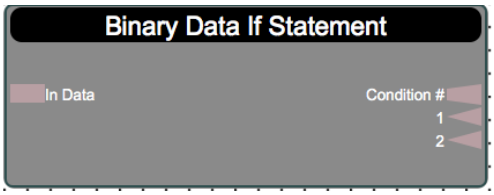



図:Binary Data IF Statement モジュール(ピン:In Data／Condition #)

Binary Data IF Statement モジュールは、入力バイナリーデータに対する判定条件を提供します。このモジュールは、チャンネルの順に条件判定を行います。一致する条件が見つかった時点で、そのチャンネルの Logic 出力を ON にし、一致した条件の番号(Matched Condition #)も出力ピンに出力します。

条件判定では、バイナリーデータの完全一致のみをサポートしています。

入力バイナリーデータがどの条件にも一致しない場合、すべての出力状態は OFF となり、Matched Condition # は「0」になります。

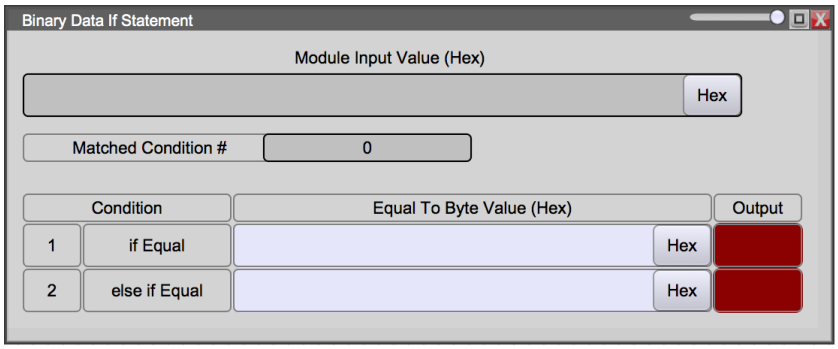



図:Binary Data IF Statement のモジュールコントロールページ(Equal To Byte Value)

## 13.6 Binary Data Delay

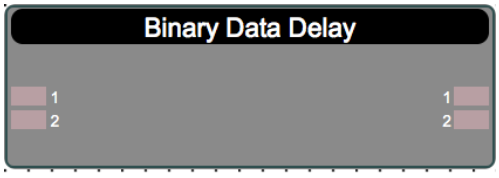



図:Binary Data Delay モジュール

Binary Data Delay モジュールは Logic Delay モジュールに似ていますが、バイナリーデータ値を扱う点が異なります。入力されたバイナリーデータを、一定の秒数だけ遅らせて出力ピンに渡します。各チャンネルは他のチャンネルとは独立して動作します。

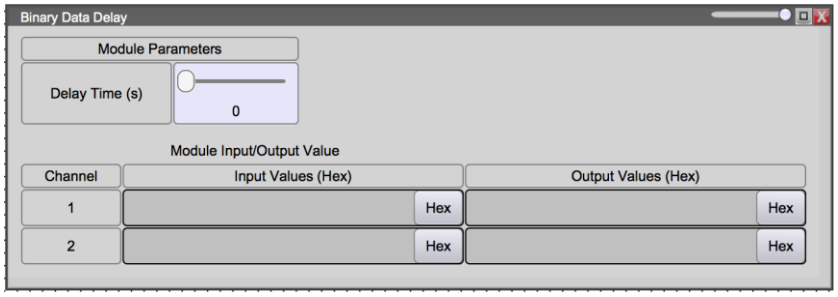



図:Binary Data Delay のモジュールコントロールページ(パラメーター:Delay Time)

## 13.7 Binary Data Holder

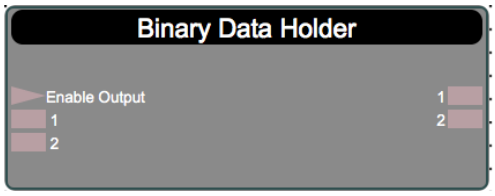



図:Binary Data Holder モジュール(ピン:Enable Output)

Binary Data Holder モジュールは、値のスナップショットを取り込むためのバッファーを提供します(バイナリーデータを保持します)。キャプチャー入力ピンが変化すると、全チャンネルの入力ピンのバイナリーデータを読み取り、その入力バイナリーデータを対応する出力ピンに送ります。各チャンネルは互いに独立しています。処理されるのは有効な入力ピンのみで、その他の無効なピンは無効のままとなります。

トリガーモードは 2 種類あります。「Rising edge trigger(立ち上がりエッジトリガー)」では、「Enable Output」ピンが OFF から ON に変化したときにのみ入力値を取り込みます。ピンが ON から OFF に変化したときは何も起こりませんが、これにより、次回の取り込み動作が可能な状態になります。

2 つ目は「Level trigger(レベルトリガー)」モードです。「Enable Output」が ON の間、入力値を継続的に取り込み、出力側へ渡し続けます。

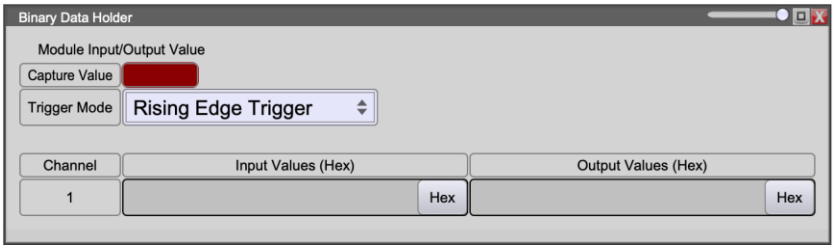



図:Binary Data Holder のモジュールコントロールページ(パラメーター:Trigger Mode)

## 13.8 Binary Data Queue

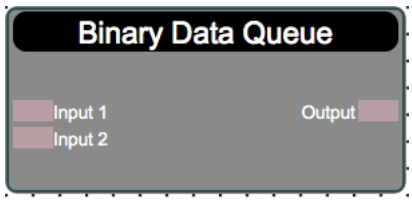



図:Binary Data Queue モジュール

Binary Data Queue モジュールは、変更された(前段の処理で更新された)すべての入力バイナリーデータを取得し、順番待ちのキューに入れます。これらのデータは、後続のモジュールへ 1 つずつ送られます。処理サイクルごとにキュー内の次の値を 1 つずつ送出し、キュー内のすべての要素が処理されるまで続けます。

このモジュールは、複数の処理経路があり、各経路で処理が行われた後、それぞれの経路の結果を 1 つずつ処理させたい場合に特に便利です(例:同じ TCP 接続を通じてリモートデバイスへ送信する場合)。

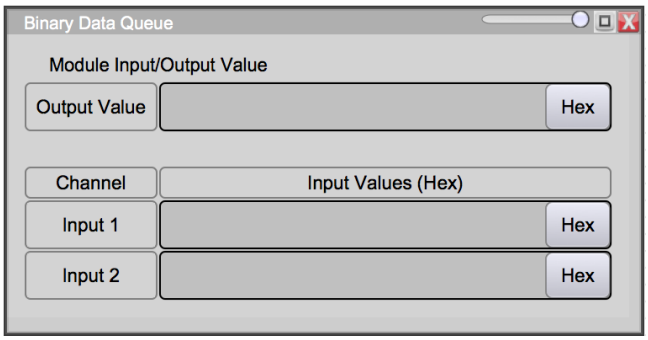



図:Binary Data Queue のモジュールコントロールページ

## 13.9 Binary Data Input Selector

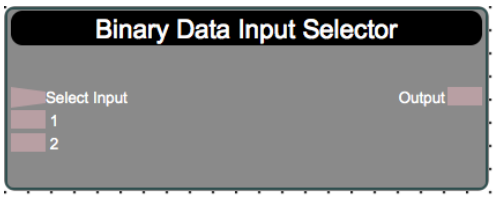



図:Binary Data Input Selector モジュール(ピン:Select Input)

Binary Data Input Selector モジュールは、出力ピンを特定の入力ピンに接続します。接続されると、その入力ピンでのバイナリーデータ値の変化はすべて出力ピンに渡されます。

Select Input の値は、1～n の範囲で指定します。Select Input の値が「0」の場合、出力はどこにも接続されず、データの更新は行われません。Select Input の値が利用可能な入力チャンネル数より大きい場合は、最後の入力チャンネルが選択されます。

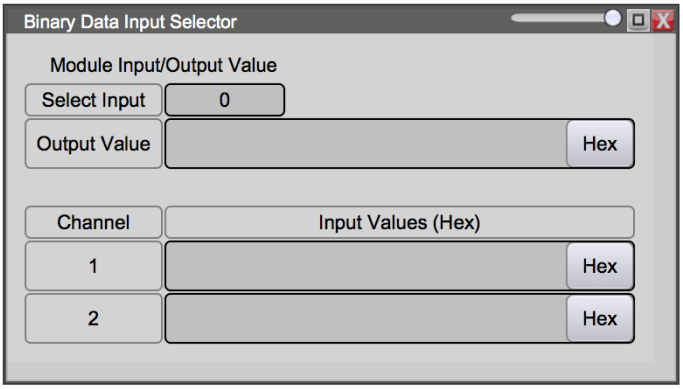



図:Binary Data Input Selector のモジュールコントロールページ

## 13.10 Binary Data Output Selector

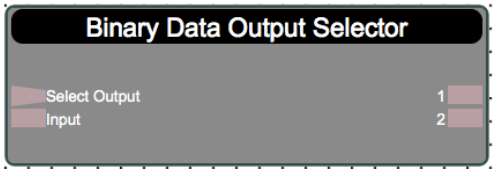



図:Binary Data Output Selector モジュール(ピン:Select Output／Input)

Binary Data Output Selector モジュールは Input Selector に似ていますが、選択を出力側で行う点が異なります。

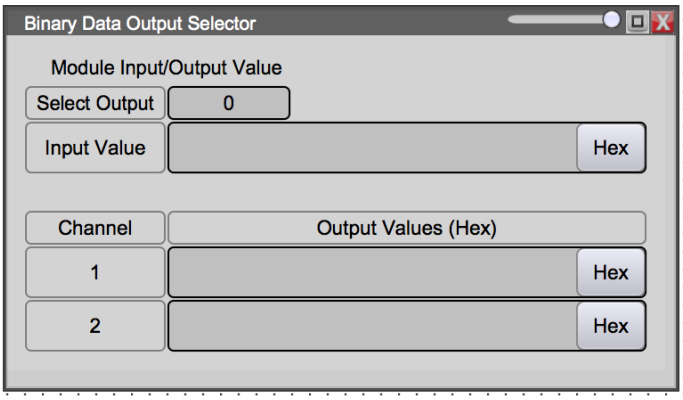



図:Binary Data Output Selector のモジュールコントロールページ

# 14 データ変換モジュール

## 14.1 Logic to Numeric Value Convertor

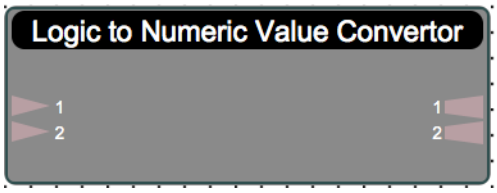



図:Logic to Numeric Value Convertor モジュール

Logic to Numeric Value Convertor モジュールは、Logic の ON/OFF 信号を数値に変換します。ON 状態と OFF 状態のそれぞれに対応する数値を入力できます。各チャンネルの出力値は、入力状態に基づいて、入力された数値に設定されます。入力ピンの状態が無効な場合、出力は無効のままです。各チャンネルは互いに独立しています。

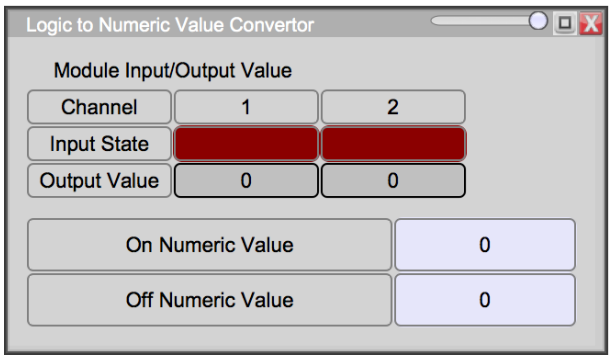



図:Logic to Numeric Value Convertor のモジュールコントロールページ(On Numeric Value／Off Numeric Value)

## 14.2 Logic to Binary Data Convertor

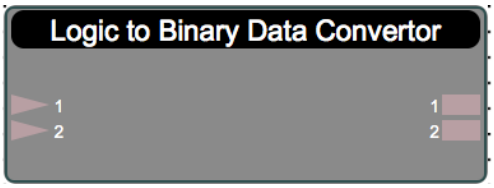



図:Logic to Binary Data Convertor モジュール

Logic to Binary Data Convertor モジュールは、Logic の ON/OFF 信号をバイナリーデータに変換します。ON 状態と OFF 状態のそれぞれに対応するバイナリーデータを入力できます。各チャンネルの出力値は、入力状態に基づいて、入力されたバイナリーデータ値に設定されます。入力ピンの状態が無効な場合、出力は無効のままです。各チャンネルは互いに独立しています。

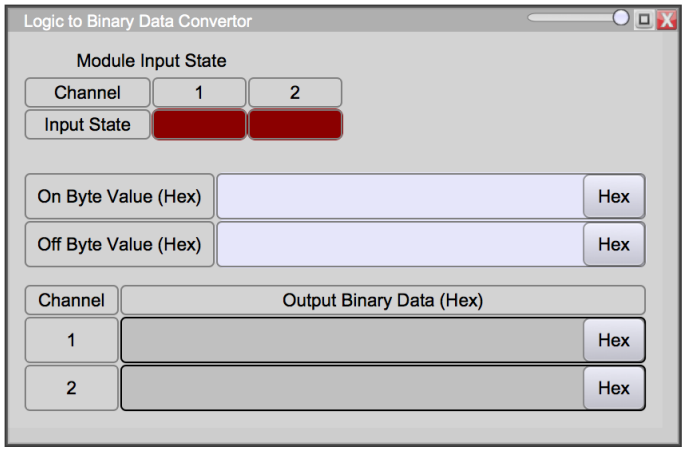



図:Logic to Binary Data Convertor のモジュールコントロールページ(On Byte Value／Off Byte Value)

## 14.3 Numeric Value to Logic Convertor

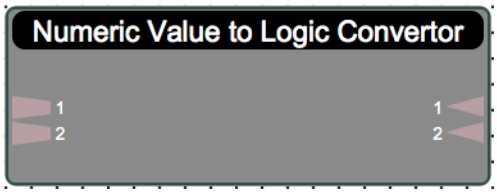



図:Numeric Value to Logic Convertor モジュール

Numeric Value to Logic Convertor モジュールは、入力された数値に基づいて、チャンネルの出力ピンの論理状態を設定します。入力値が判定条件に該当する場合、対応する出力状態がそれに応じて設定されます。

条件判定には、Equal(等しい)、NOT Equal(等しくない)、Greater Than(より大きい)、Greater Than or Equal(以上)、Less Than(より小さい)、Less Than or Equal(以下)などを使用できます。

各チャンネルは互いに独立しています。入力が無効な場合、対応するピンは無効のままです。

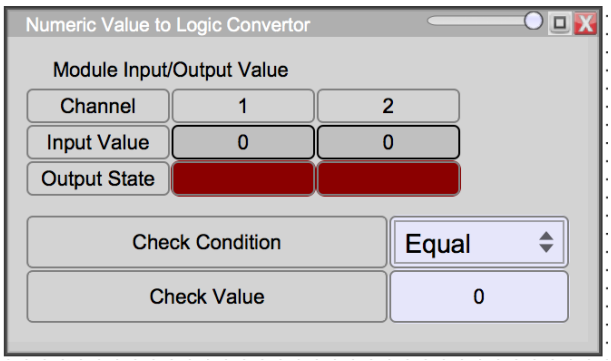



図:Numeric Value to Logic Convertor のモジュールコントロールページ(Check Condition／Check Value)

## 14.4 Numeric Value to Binary Data Convertor

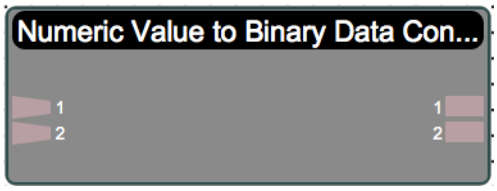



図:Numeric Value to Binary Data Convertor モジュール

Numeric Value to Binary Data Convertor モジュールは、数値データを、その値のバイナリー表現に変換します。入力値に対して、次のバイナリーデータ形式をサポートしています。

・8 bit integer:10 進数値を 1 バイトの表現に変換します。
・16 bit integer:10 進数値を 2 バイトの表現に変換します。
・32 bit integer:10 進数値を 4 バイトの表現に変換します。
・32 bit float:値を 4 バイトの浮動小数点表現に変換します。
・64 bit double:値を 8 バイトの浮動小数点表現に変換します。

出力形式が 1 バイトを超える場合は、バイト順を入れ替えるオプション(リトルエンディアンまたはビッグエンディアン表現)があります。

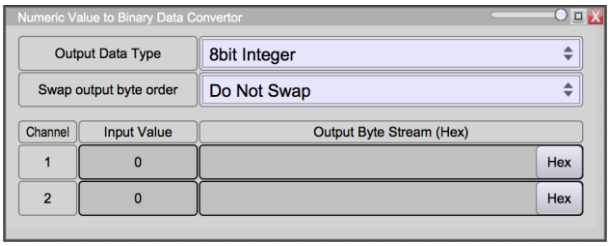



図:Numeric Value to Binary Data Convertor のモジュールコントロールページ(Output Data Type／Swap output byte order)

## 14.5 Numeric Value to Binary String Convertor

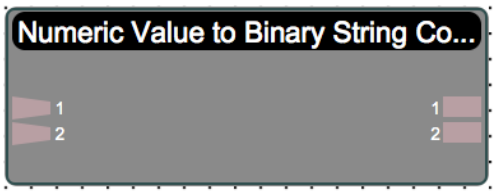



図:Numeric Value to Binary String Convertor モジュール

Numeric Value to Binary String Convertor モジュールは、数値を文字列表現に変換します。変換後の総バイト数と、結果の文字列に保持する小数点以下の桁数を指定できます。

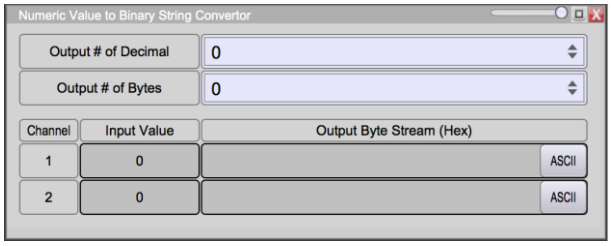



図:Numeric Value to Binary String Convertor のモジュールコントロールページ(Output # of Decimal／Output # of Bytes)

## 14.6 Binary Data to Logic Convertor

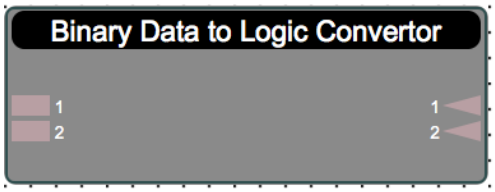



図:Binary Data to Logic Convertor モジュール

Binary Data to Logic Convertor モジュールは、入力されたバイトデータを「Check Byte Stream Value」と照合します。バイナリーデータの値が一致する場合、出力状態は ON になります。一致しない場合、出力状態は OFF になります。

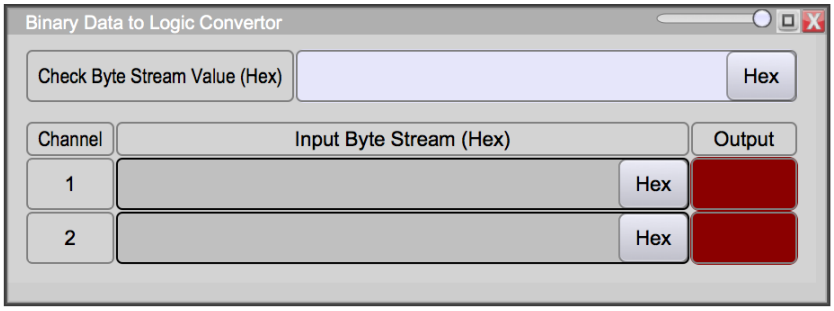



図:Binary Data to Logic Convertor のモジュールコントロールページ(Check Byte Stream Value)

## 14.7 Binary Data to Numeric Value Convertor

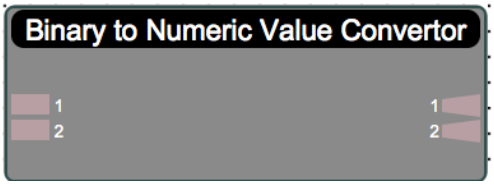



図:Binary Data to Numeric Value Convertor モジュール

Binary Data to Numeric Value Convertor モジュールは、入力されたバイナリーデータを、バイナリー形式の数値表現として解釈します。次の形式をサポートしています。

・8 bit integer:1 バイトの表現を 10 進数値として解釈します。

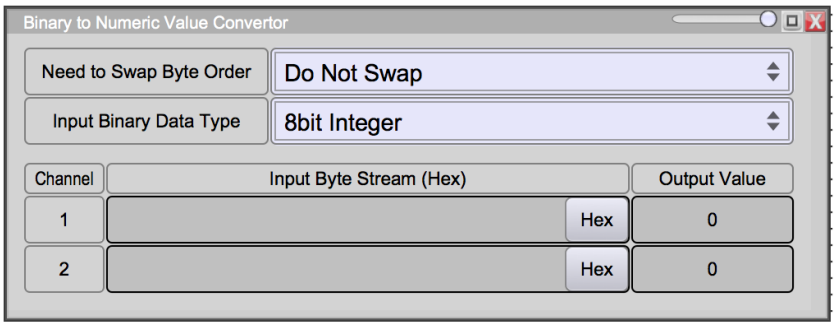
・16 bit integer:2 バイトの表現を 10 進数値として解釈します。

・32 bit integer：4 バイトの表現を 10 進数値として解釈します。

・32 bit float：4 バイトの浮動小数点表現を値として解釈します。

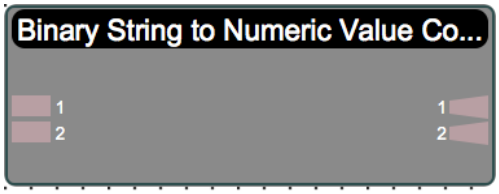
・64 bit double：8 バイトの浮動小数点表現を値として解釈します。

入力データに必要なバイト数より多くのバイトが含まれている場合、残りのバイトは無視されます。



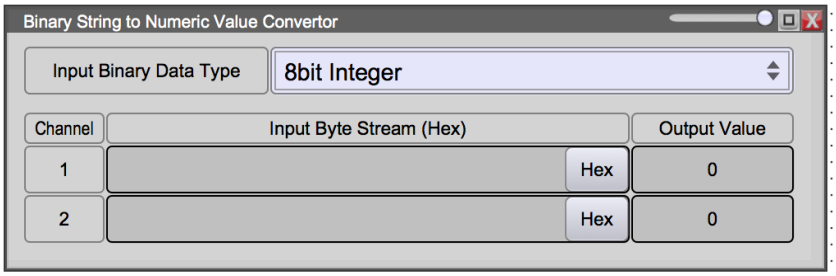

図：Binary Data to Numeric Value Convertor のモジュールコントロールページ（Need to Swap Byte Order ／Input Binary Data Type）

## 14.8 Binary String to Numeric Value Convertor




図：Binary String to Numeric Value Convertor モジュール

Binary String to Numeric Value Convertor モジュールは、入力されたバイナリーデータを文字列値として解釈し、その文字列値を対応する数値に変換します。




図：Binary String to Numeric Value Convertor のモジュールコントロールページ（Input Binary Data Type）

# 15 データフィードバックモジュール

一般に、コントローラーの処理フローは入力モジュールから始まり、信号フローに従って処理が進みます。通常のモジュールをループさせることはできません。しかし、状況によっては、データ値を前段のモジュールへフィードバックする必要があります。これは Data Feedback モジュールで実現できます。これらのモジュールは前回の実行サイクルのデータ値を取得し、その値を次の実行サイクルの入力として使用します。
データ型ごとに専用のフィードバックモジュールがあり、それぞれ対応するデータ型を扱います。各モジュールは初期値も提供するため、最初の実行サイクルでも処理の基になるデータが得られます。

## 15.1 Logic Data Feedback

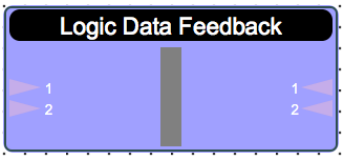



図:Logic Data Feedback モジュール

Logic Data Feedback モジュールは、ロジックデータのフィードバックを扱います。最初の実行サイクルで処理フローに入力される初期値を選択できます。

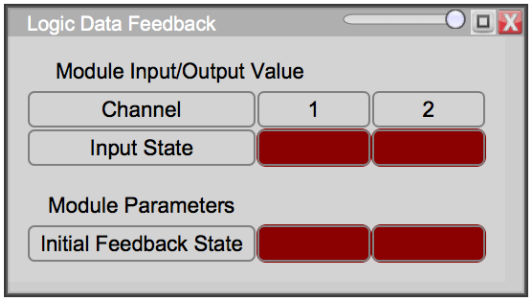



図:Logic Data Feedback のモジュールコントロールページ(Initial Feedback State)

## 15.2 Numeric Data Feedback

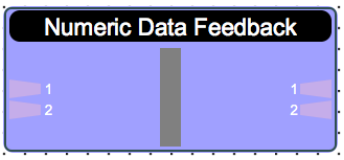



図:Numeric Data Feedback モジュール

Numeric Data Feedback モジュールは、数値データのフィードバックを扱います。最初の実行サイクルで処理フローに入力される初期値を選択できます。

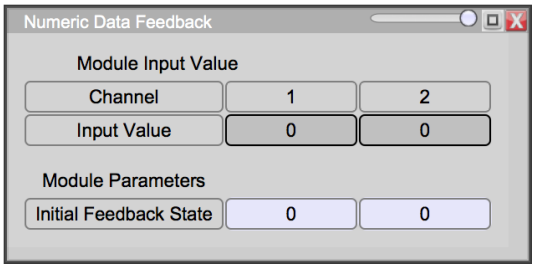



図:Numeric Data Feedback のモジュールコントロールページ(Initial Feedback State)

## 15.3 Binary Data Feedback

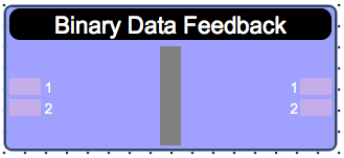



図:Binary Data Feedback モジュール

Binary Data Feedback モジュールは、バイナリーデータのフィードバックを扱います。最初の実行サイクルで処理フローに入力される初期値を選択できます。




図:Binary Data Feedback のモジュールコントロールページ(Initial Feedback Value)

# 16 UI モジュール

ユーザーが操作するユーザーインターフェイス(UI)オブジェクトです。

## 16.1 Momentary ON/OFF Button User Interface

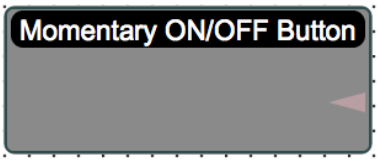



図:Momentary ON/OFF Button モジュール

Momentary ON/OFF Button User Interface モジュールは、ユーザーが ON/OFF 状態を操作するためのプッシュボタンオブジェクトを提供します。UI ボタンを押すと、モジュールの出力に ON 状態が出力されます。ボタンを離すまで ON 状態が続き、離すと出力状態は OFF になります。

このモジュールには出力ピンしかありません。この UI オブジェクトの状態は、他のモジュールの処理結果では制御できないためです。

このUIオブジェクトは、XTouchパネル(またはAndroidやiOSで動作するXTouchApp)に配置できます。これにより、ユーザーがコントローラー機能を操作できるようになります。

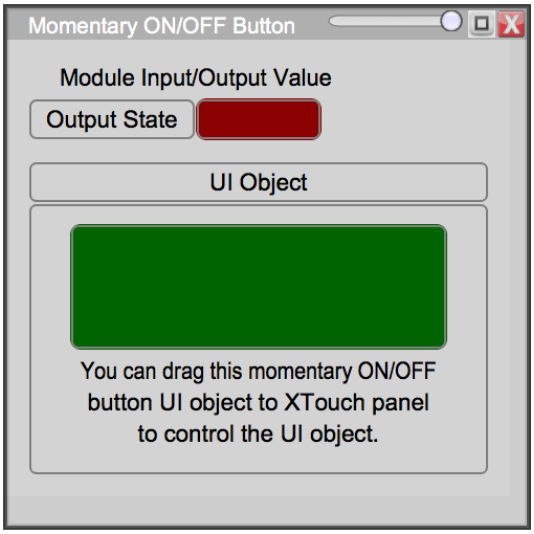



図:Momentary ON/OFF Button のモジュールコントロールページ(UI Object)

## 16.2 Toggle ON/OFF Button User Interface

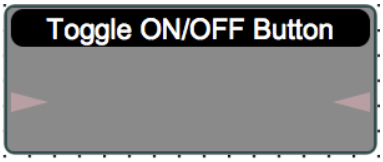



図:Toggle ON/OFF Button モジュール

Toggle ON/OFF Button User Interface モジュールは、ユーザーが操作するトグルボタンオブジェクトを提供します。この UI オブジェクトの状態を変更すると、このモジュールの出力ピンが UI の状態に追従します。逆に、このモジュールに Logic データを入力した場合は、UI オブジェクトの状態も入力値に追従します。

動作モードは 2 種類あります。入力ピンの状態を出力ピンへ渡す(Input Pass through to Output)モードと、入力ピンでは UI の状態の更新のみを許可するモードです。パススルーしない場合、出力ピンは、ユーザーの操作によって UI オブジェクトが変更されたときにのみ変化します。

このUIオブジェクトは、XTouchパネル(またはAndroidやiOSで動作するXTouchApp)に配置できます。これにより、ユーザーがコントローラー機能を操作できるようになります。

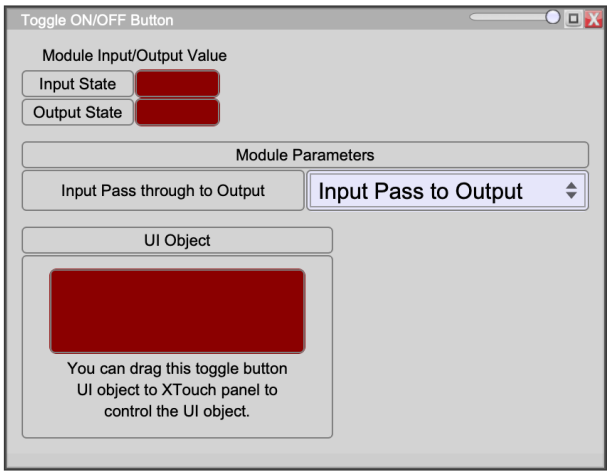



図:Toggle ON/OFF Button のモジュールコントロールページ(パラメーター:Input Pass through to Output)

## 16.3 Radio Button User Interface

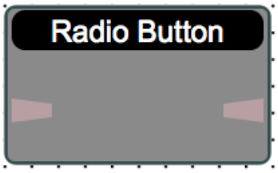



図:Radio Button モジュール

Radio Button User Interface モジュールは、ユーザーが独自のラジオボタンを定義するための仕組みを提供します。このラジオボタンは、XTouch パネルで定義する場合にのみ有用です。この UI パネルでは使用できるボタンが 1 つだけなので、ラジオボタングループを作るには、このボタンを複数個 XTouch パネルへドラッグする必要があります。

複数のボタンを XTouch パネルへドラッグすると、ボタンの背後にあるコントロールパラメーターによって、それらのボタンが自動的にグループ化されます。つまり、実際のパラメーター状態は、同じパラメーター値を共有します。ラジオボタングループ内の各ボタンには、Control Action Value を個別に定義できます。ボタンを押すと、その Control Action Value が送信されてパラメーター値が設定されます。グループ内の他のボタンはパラメーター値の変更を受け取り、パラメーター値が自分の Control Action Value と一致する場合に、そのボタンがハイライト表示されます。これにより、ラジオボタンの効果が得られます。

このラジオボタンのスケールを変更したい場合、このコントロールページでは設定できません。このラジオボタンを XTouch パネルへドラッグし、そこで UI オブジェクトをさらにカスタマイズします。ボタンの最大値、最小値、小数点以下の桁数、表示単位を指定できます。通常は、コントローラー側で設定したラジオボタン UI オブジェクトと同じ範囲を使用します。XTouch アプリケーションでは、適切な値の範囲と単位が表示されます。

動作モードは 2 種類あります。入力ピンの状態を出力ピンへ渡すモードと、入力ピンでは UI の状態の更新のみを許可するモードです。パススルーしない場合、出力ピンは、ユーザーの操作によって UI オブジェクトが変更されたときにのみ変化します。

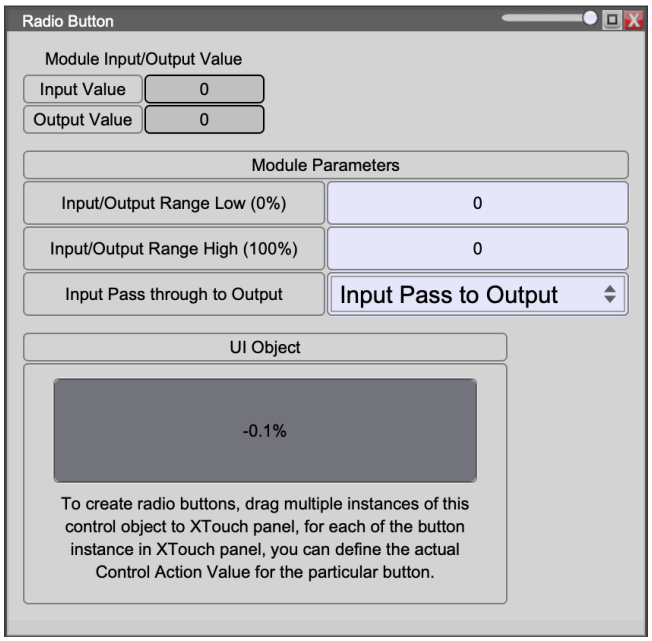



図:Radio Button のモジュールコントロールページ

## 16.4 Slider User Interface

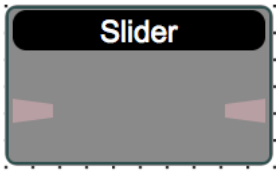



図:Slider モジュール

Slider User Interface モジュールは、ユーザーが操作するスライダーオブジェクトを提供します。スライダーオブジェクトのスケールは 0%～100% が基準です。スライダーを操作すると、設定された範囲に従って実際の値が計算され、その計算値が Numeric 出力ピンに送られます。

逆に、このモジュールに数値が入力された場合は、範囲に基づいて % 値に変換され、その % 値がスライダーの 0～100% の範囲に反映されます。このモジュールのコントロールパネルでは、スライダーの単位は 0～100% です。

このスライダーのスケールを変更したい場合、このコントロールページでは設定できません。このスライダーを XTouch パネルへドラッグし、そこで UI オブジェクトをさらにカスタマイズします。スライダーの最大値、最小値、小数点以下の桁数、表示単位を指定できます。通常は、コントローラー側で設定したスライダーUI オブジェクトと同じ範囲を使用します。XTouch アプリケーションでは、適切な値の範囲と単位が表示されます。

動作モードは 2 種類あります。入力ピンの状態を出力ピンへ渡すモードと、入力ピンでは UI の状態の更新のみを許可するモードです。パススルーしない場合、出力ピンは、ユーザーの操作によって UI オブジェクトが変更されたときにのみ変化します。

このUIオブジェクトは、XTouchパネル(またはAndroidやiOSで動作するXTouchApp)に配置できます。これにより、ユーザーがコントローラー機能を操作できるようになります。

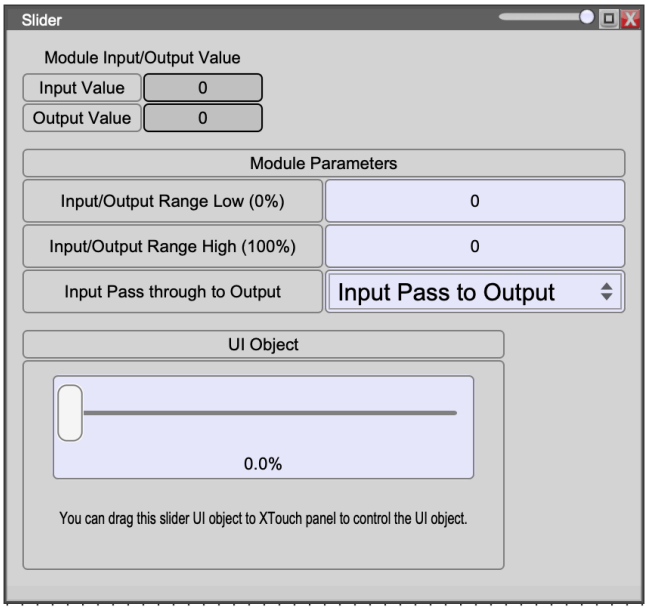



図:Sliderのモジュールコントロールページ(パラメーター:Input/Output Range Low (0%)／High (100%)、Input Pass through to Output)

# 17 Luaスクリプトモジュール

## 17.1 Lua Script

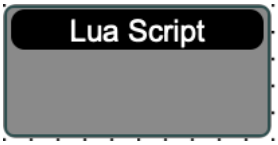



図:Lua Scriptモジュール

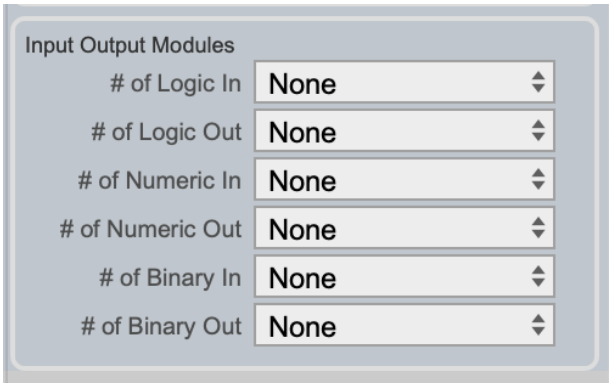



図:入力/出力ピン数の設定(オブジェクトプロパティー領域:# of Logic In／Out、# of Numeric In／Out、# of Binary In／Out)

Lua Script モジュールは、ユーザーが定義したスクリプトに基づいて実行される特別なモジュールです。設計内には、Lua モジュールをいくつでも配置できます。各 Lua モジュールは、処理フローの順に実行されます。これらすべての Lua モジュールは同じ環境で動作します（つまり、グローバル変数を共有できます）。

Lua スクリプトモジュールを設計するには、まず、モジュールが持つ IO ピンの数を定義する必要があります。Logic、Numeric、Binary の各データ型に対応しています。入力ピンと出力ピンの数を指定するには、モジュールを選択し、右側の「オブジェクトプロパティー」領域で、データ型ごとに入力/出力の数を選択します。

スクリプトの実行時、すべての入力ピンのデータは「InTable」というテーブルとして渡されます。各ピンの値は、インデックスを指定してテーブルから値を取得することで参照できます。インデックスの名前は、モジュール上のピンの位置と、対応するデータ型に基づきます（例：1 番目のピンは「1_」の後にデータ型が続きます。Logic なら「1_Logic」、Numeric なら「1_Numeric」、Binary なら「1_Binary」になります）。出力テーブルにも、同様の命名規則が適用されます。

処理後、結果を出力配列「OutTable」に格納します。同様に、出力ピンの位置とデータ型に対応するインデックスを使って、出力テーブルの値を設定します。

データ処理に加えて、デバッグの仕組みも用意しています。「DebugMessage」という変数に対して print 関数を実行すると、その DebugMessage の値が Debug 領域に表示されます。これにより、ある程度のデバッグが可能です。

ONLINE モードでは、スクリプトを修正して「Submit」ボタンを押すと、デバイス上のスクリプトを更新できます。新しいスクリプトは、次回の処理実行時から実行されます。これにより、迅速な開発サイクルが可能になります。

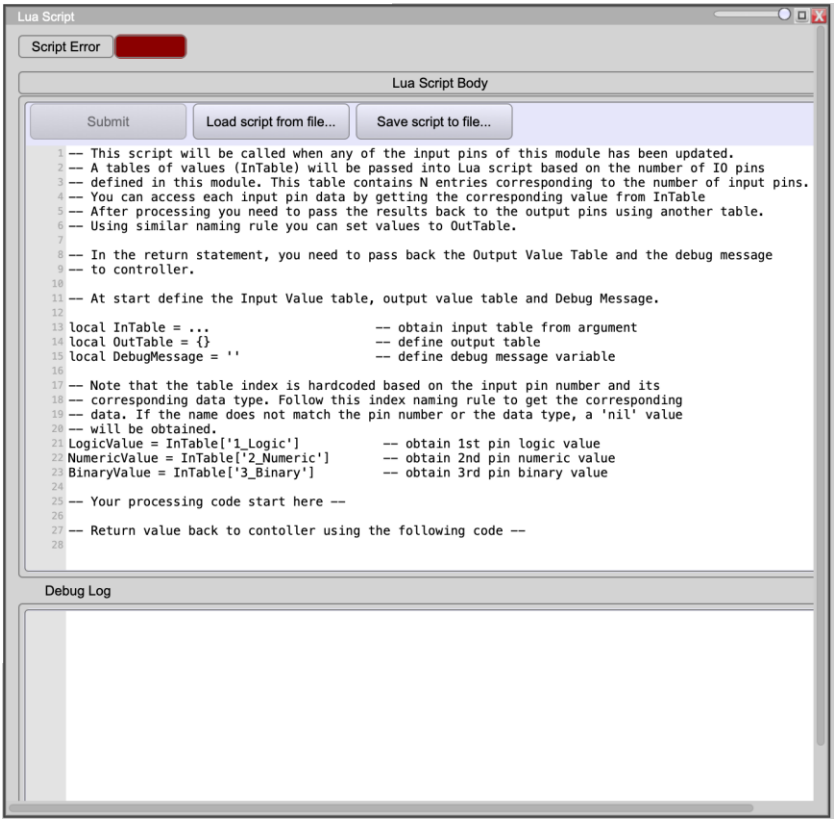



図:Lua Script のモジュールコントロールページ(Script Error／Submit／Load script from file／Save script to file／Lua Script Body／Debug Log)

# 18 設計例

## 18.1 制御タイミングシーケンスの例

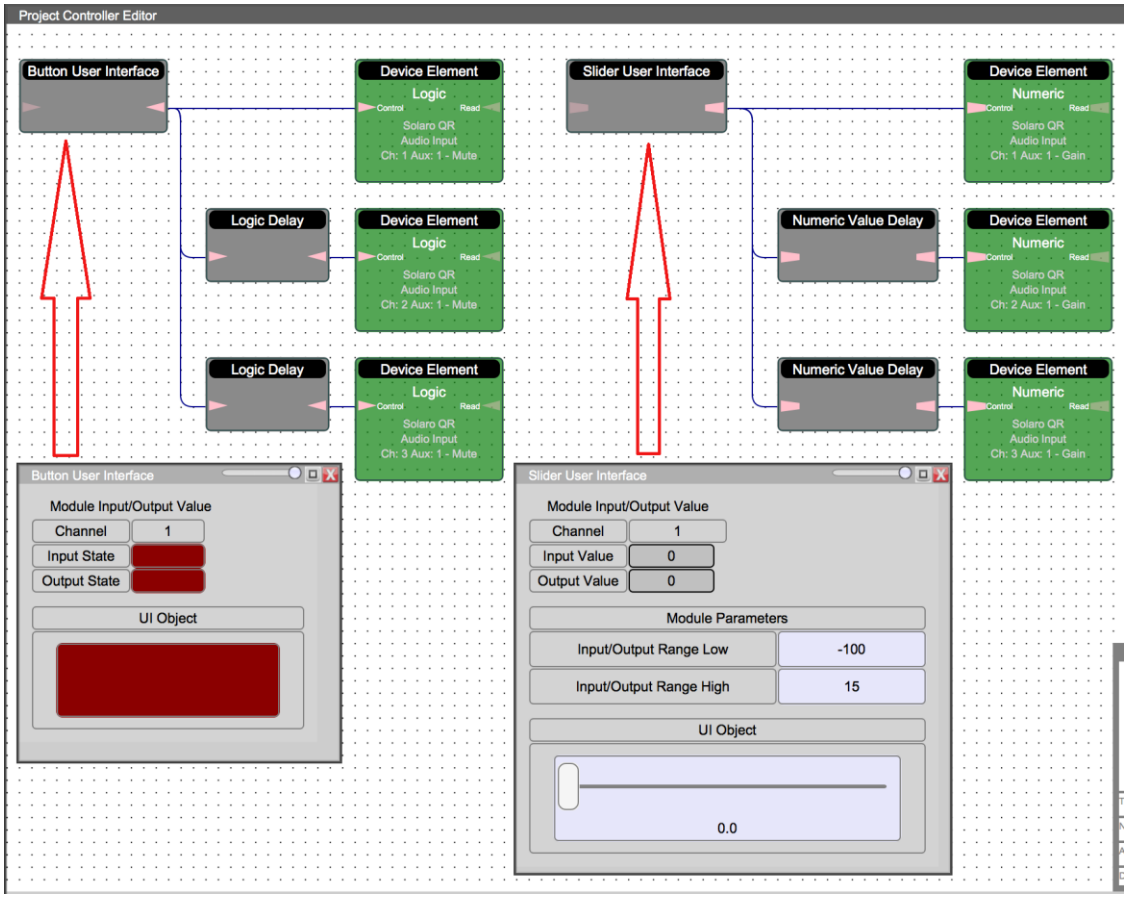



図:制御タイミングシーケンスの設計例(Project Controller エディター)

この例は、ミュート制御部(Logic)とスライダー制御部(Numeric)に分けて考えることができます。

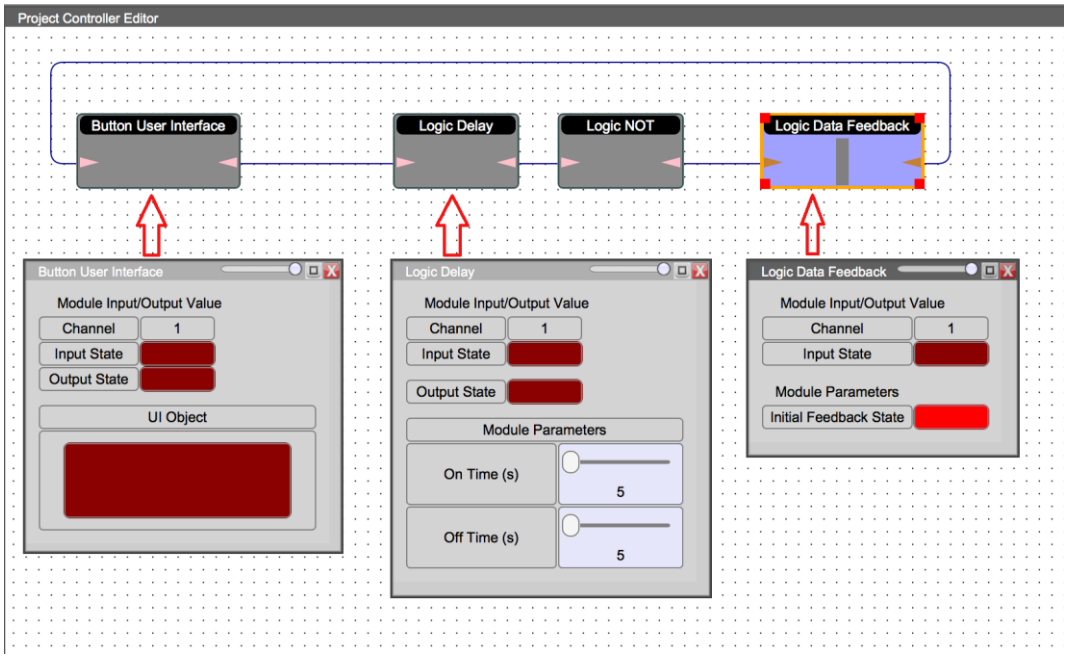
ミュート制御部では、UI ボタンモジュールを使って、3 つの異なるミュートボタンへ制御を送ります。これらのミュートボタンの制御は、Logic Delay モジュールで設定した遅延に基づいて行われます。これは、機器の ON/OFF 状態をタイミングシーケンスで制御したい場合の典型的な使い方です。UI ボタンモジュールの詳細ページでは、パラメーター設定は不要です。UI オブジェクトのボタンをクリックするだけで、処理フローに ON/OFF コマンドを送れます。

UI トグルボタンをタッチパネルで使いたい場合は、この UI オブジェクトをタッチパネルの設計へドラッグ&ドロップするだけです。

ミュートに加えて、スライダーによる制御シーケンスも行えます。これはコントローラー設計の 2 つ目の部分です。スライダーUI オブジェクトでは、UI スライダーの 0~100% が出力範囲をどのように表すかを設定する必要があります。この例では、最大値が 15dB、最小値が -100dB です。これらのスライダー値は、遅延時間に基づいて、異なるチャンネルのゲイン制御へ送られます。

UI トグルボタンと同様に、スライダーUI オブジェクトもタッチパネルへドラッグ&ドロップして制御できます。

## 18.2　ロジックデータフィードバックの例




図：ロジックデータフィードバックの設計例

この例では、データフィードバックモジュールの使い方を示します。この例では、UI オブジェクトの状態が 5 秒ごとに ON と OFF の間で切り替わります。以下で、設計の流れをより詳しく説明します。

最初のモジュールは User Interface ボタンで、その初期状態は Logic Data Feedback の初期フィードバック状態から与えられます。ここでは ON に設定しています。この ON 状態により、UI ボタンが ON になります。次に、その ON 状態を Logic Delay モジュールへ送ります。5 秒後、ON 状態は Logic NOT に渡されます。Logic NOT モジュールは ON 状態を OFF に変え、フィードバックモジュールに保存します。

次のコントローラーの実行サイクルでは、Button User Interface オブジェクトへの入力は OFF になります。この OFF 状態は 5 秒遅延された後、Logic NOT モジュールによって ON に変換され、次のサイクルで使用するために Logic Data Feedback モジュールに保存されます。

これにより、UI オブジェクトの ON/OFF 状態が自動的に繰り返し切り替わります。

## 18.3　数値条件判定の例

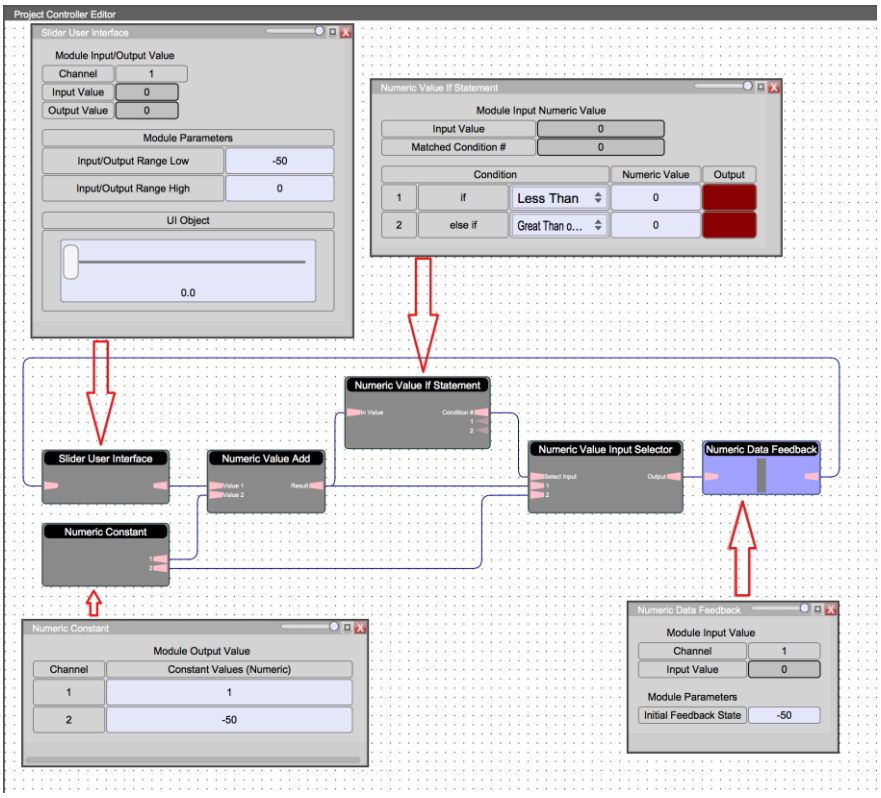



図:数値条件判定の設計例

この例では、Numeric IF Statement モジュールと Numeric Value Input Selector モジュールの使い方を説明します。この例の目的は、-50dB から 0dB まで 1dB ずつ自動的に上昇するスライダーUI オブジェクトを実現することです。0dB に達すると、スライダーは -50dB にリセットされ、再び上昇します。

処理フローの最初は、初期値が -50 の Numeric Data Feedback モジュールです。この数値は Slider User Interface オブジェクトに送られ、スライダーが 0% に設定されます。この -50 は Numeric Value Add モジュールにも渡され、1 が加算されます。その結果である -49 は、さらに処理するため、2 つの別のモジュールに渡されます。

1 つは Numeric IF Statement モジュールで、値を、モジュールに設定した条件と照合します。値が 0 未満(Less Than 0)の場合、Matched Condition ピンは「1」になります。値が 0 以上(Greater Than or Equal to 0)の場合、Matched Condition ピンは「2」になります。この Matched Condition ピンは、Numeric Input Selector モジュールが、どの入力値を選択するかを判断するために使用されます。

値が 0 未満の場合は、Numeric Add モジュールの出力を選択します。値が 0 に達した場合は、Numeric Constant モジュールの定数値(-50)を選択します。

以上