AUTO GROUP 自动编组

总览

当一个 AUTO GROUP 设备被放到一个分级块里头时，这个分级块内所有的设备里头有着相同控制名称（ruid）和相同设备类型的读写控制部件将会被自动地编组到一起。使用这个设备来做各个通道需要共享同样的设置的立体声或多通道设备。

注意：这个对图形均衡器不起作用。

设备属性

允许
如果勾选，设备起作用。如果不勾选，设备无效。不勾选相当于删除掉了这个设备，但是如果是在做设计实验的时候，这么做比较容易恢复。

编组名称
如果这个属性是空白的，那么此设备的有效范围是它所在的页面（通常是这个分级块的示意图页面）。但是如果将两个或多个不同区域的这种设备赋予相同的编组名称，那么它们就会联合其控制范围内的所有控制部件。

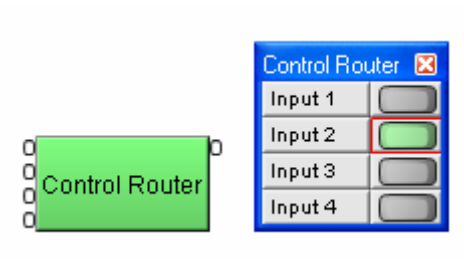
隔离控制部件
输入一个控制部件列表，用逗号分开，来将它们从编组中隔离。例如，如果你创建了一个四进四出的分级块，包含四个四段参量均衡器，你想单独控制每个均衡器第 4 通道的增益的话，那就在这里输入“ band_gain_4 ”。

端口
无。

控制部件
无。

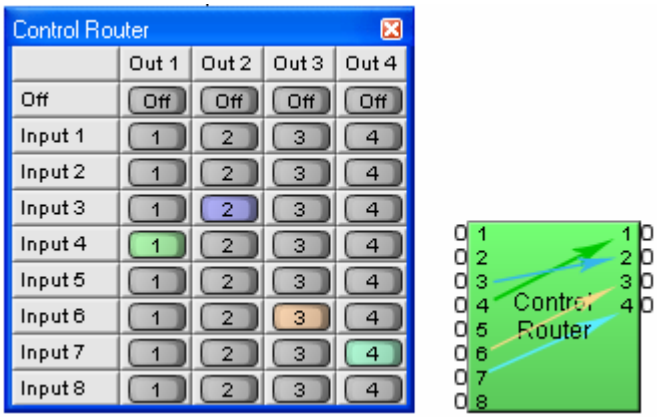
发布于 1.1.0 版。

控制路由器 Control Router



控制路由器和音频路由器很相似，只不过它路由的是控制信号而不是音频信号。它又和虚拟分配器比较相似，只不过它不是象双向（平级）群一样运作，而更象是一个有向群(主控控制隶属,但是隶属不能控制主控,甚至不能控制它们自己)。

这个设备是用来动态地重分配哪个控制部件和/或控制别名在当前控制的是哪个控制部件。注意，你可以使用多进一出的配置方式。如果你有很多通用控制部件和一些固定控制部件，你可能需要重新排列。但是有了这个设备，就不需要重组也可以实现。这里有一个例子：



路由器内的设置来决定哪个输入链接到哪个输出。

设备属性

控制输入数量
指定本设备的输入端口数量。

控制输出数量
指定本设备的输出端口数量。

允许“关闭”状态
指定一个输出端口可以不连接任何输入端口，还是必须连接到输入端口上。

端口

注意：在连线模式下，把线拉到端口上还没连接时，会出现一个工具条，显示该端口的 ID。

input_1 到 input_<控制输入端口数量>
输入端到控制路由矩阵。

output_1 到 output_<控制输出端口数量>
来自控制路由矩阵的输出端。

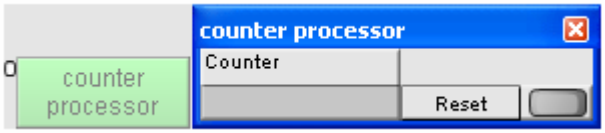
控制部件：

selector_1 到 selector_<控制输出端口数量>

控制部件输入输出是相关的。如果在允许“ 关闭 ”状态下，这个控制部件的有效范围是从 0 到<控制输入端口数量>；如果允许“ 关闭 ”状态取消，有效范围是从 1 到<控制输入端口数量>。这个控制部件可能会以很多按钮的方式来显示，也可能以下拉框的方式来显示，这主要取决于路由器的大小。

首次发布于 1.1.0 版。

计数处理器 Counter Processor

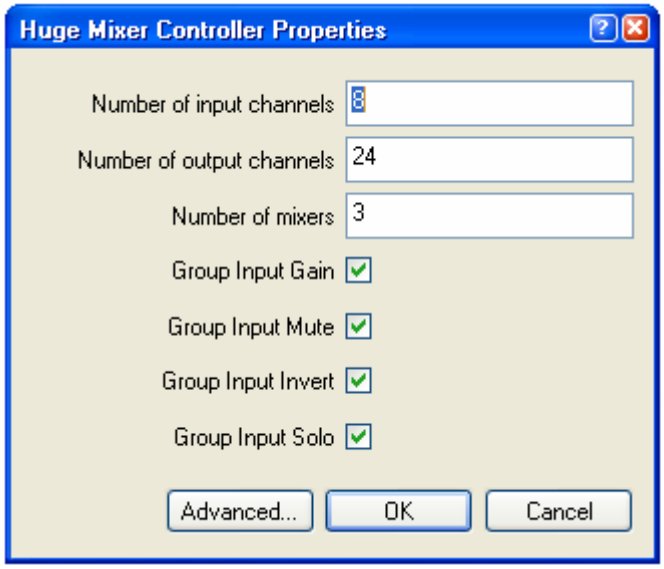


这个计数处理器把每个输入的值相加，然后减去偏移量后输出。这可以使不能被复位的外部计数器（例如 SNMP 计数器）在软件内被“复位”。当按下复位键时，此设备存储当前值（或和）作为一个偏移量，然后从它的输入和中减去这个偏移量。

大型混音台控制器 Huge Mixer Controller

总览
本设备提供映射到发送控制部件的单独设置，通过按一个选择钮，来对应一个由多个同类型混音台（每个对应总输出的一部分）组成的大型混音台中的一个输出的发送控制部件阵列。

设备属性



设置一个由 3 个 8*8 混音台组成的 8*24 混音台属性

输入通道数量
大型混音台的输入数量。（前例中是 48？？？）如果大型混音台是由不止一个小台子组成的，那么每个输入信号必须被扇形输出到所有小台子上同样的输入端口。

输出通道数量
来自大型混音台的输出数量。（前例中是 36？？？）

混音台数量
组成大型混音台的小混音台数量。（前例中是 3）

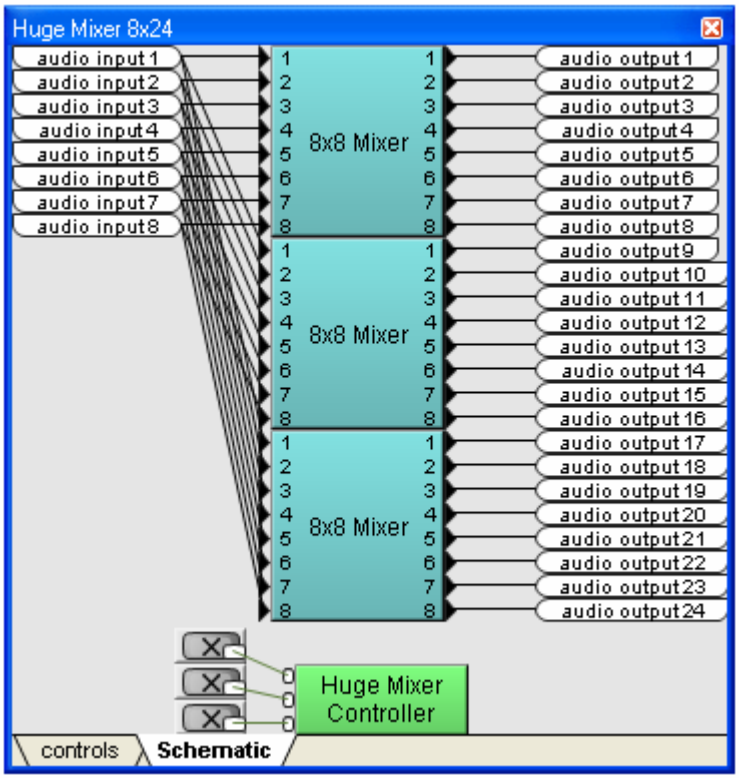
输入增益编组
定义将单个小混音台上的相关输入增益进行编组

输入静音编组
定义将单个小混音台上的相关输入静音进行编组。

输入反相编组
定义将单个小混音台上的输入反相进行编组。

输入独奏编组
定义将单个小混音台上的输入独奏进行编组。

端口

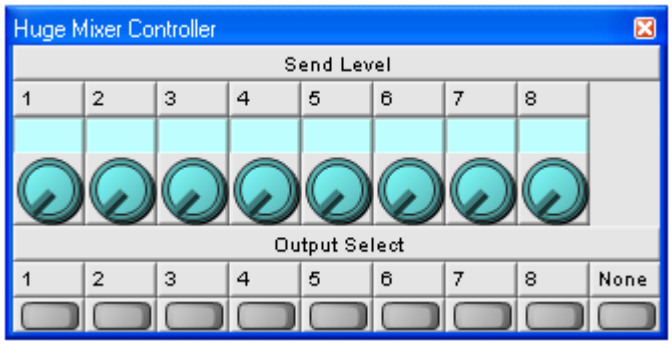


继续这个 8*24 的例子，这里有三个 8*8 的混音台连线成为一个 8*24 的。在最下面，你可以看到三个堆叠的静音按钮。它们分别来自上面三个混音台。它们可以是混音台中的任何一个控制部件，因为它们的作用只是告诉 HMC 知道去控制哪个混音台而已。这些静音按钮也不是为了连线受控于这设备的，而是混音台必须有某一个控制部件要连线到 HMC 上。

mixer_1_reference 到 mixer_<混音台数量>_reference

这个控制端口可以被连接到对应混音台上的任意一个控制部件。具体连接哪个控制部件是无所谓的，编译器仅仅是利用这个控制部件来获取该混音台的设备 ID。

控制部件

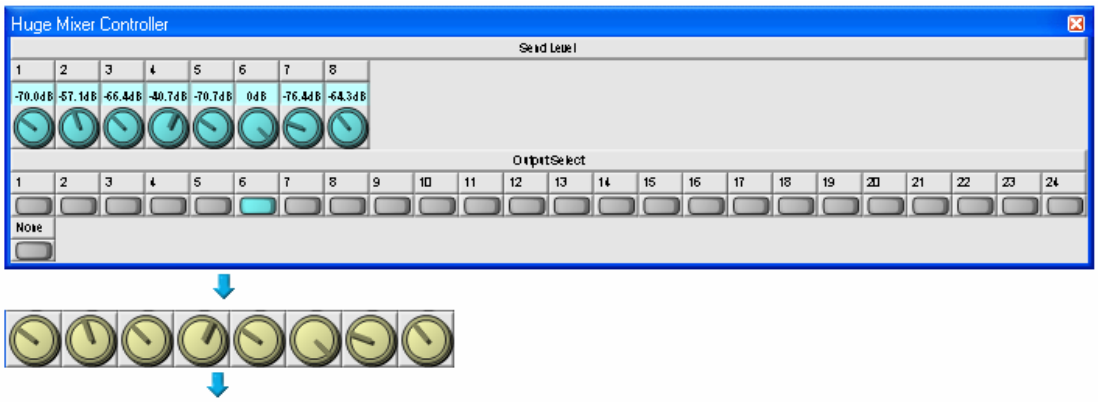


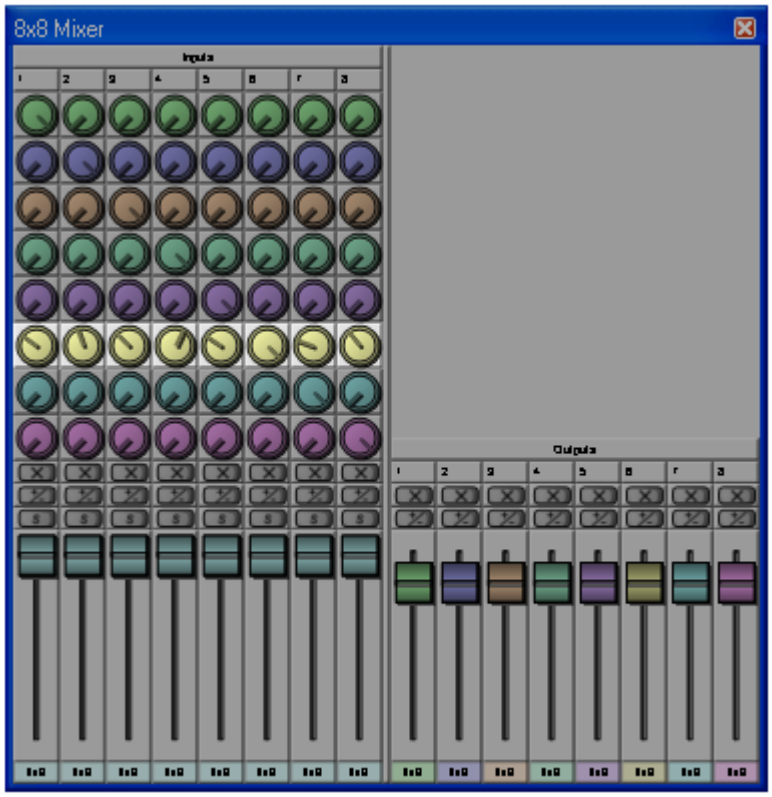
Send level 1 到 Send level <输入通道总数>
当按下一个输出选择钮时，这些虚拟旋钮就映射对应混音台上的某一排发送控制部件。

Output Select 1 到 Output Select <输出通道总数>
这些其实是一个控制部件的多次复制产生的。通过选择 RUID，配置合适的字符串来使它看起来象一组独立的按钮。它是用来选择作用于哪个输出通道的发送电平控制部件的。

None
这是选择控制部件的另一个复制。它选择的是输出 0，即是断开了发送控制部件和所有输出的连接。

用法
这个设备在配合导入房间平面图时非常有用。输出选择按钮可以放在图片上对应的扬声器位置上。发送电平旋钮可以放在图片中对应话筒和其他输入的位置。当选择了一个扬声器后，所有的旋钮都显示的是对应这个扬声器的每个音源的增益。这个界面使系统设置更方便，因为它直观显示相邻扬声器和话筒，而且比传统的大混音台上完整的笨拙的矩阵紧凑得多。


Huge Mixer Controller
Send Level
1 2 3 4 5 6 7 8
-70.0dB -57.1dB -66.4dB -40.7dB -70.7dB 0dB -76.4dB -64.3dB
Output Select
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
None

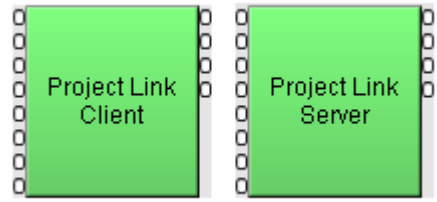

8x8 Mixer
Inputs
Outputs

平级状态 Peer Status

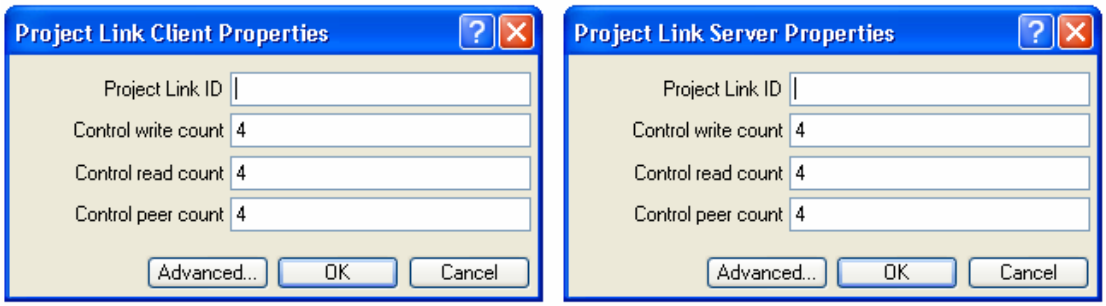
平级状态设备只用在多 NION 系统中。它显示设计中的一个特定的 NIONODE 是否可以在控制级水平上与其他角色进行通信。角色名称会被自动填写，但是设计者要做的是确保有足够的槽来适配其它角色。

工程链接 Project Linking

工程链接设备允许一个工程中的控制部件链接到另一个工程中的控制部件。工程链接服务器设备放置在被控制和监视的工程中；工程链接客户端放在施行控制和监视的工程中。工程链接服务器是被动式的：它在网络中自我广告，等待工程链接客户端发现它并开始读写控制值。



工程链接设备属性
这两个设备都有四个属性。读和写这两个属于一直是从客户端设备的角度来说的，即使在讨论服务器设备的时候。



工程链接 ID
这个属性标识服务器设备如何在网络中广告自己，以及客户端如何识别服务器。如果多个服务器共享同一个工程链接 ID，就不能确定客户端该与哪个服务器相连。如果这个属性保持空白，那么在这个设备里的运行时间控制部件将用来决定服务器如何被广告和发现。

控制写计数
控制写计数是客户端写给服务器的控制部件数量。每个控制部件由工程链接客户端设备左边的或者工程链接服务器设备右边的一个控制节点来代表。典型的应用是在客户端工程中把一个控制部件连线到一个客户端设备，并在服务器工程中把服务器设备连线到需要被控制的控制部件。这个例子里服务器和客户端都设置成写计数=1 而其他属性为 0。（注意典型的情况下，服务器和客户端不会在同一个工程中！）使用写控制部件有两个危险。第一，如果多个客户端连线到同一个服务器，它们会以“末位优先”为原则争夺设定值。第二，在服务器工程中的控制部件将不能再被调整，因为当它连接到一个工程链接服务器设备上时，它就变成了一个只读的部件了。

控制读计数
控制读计数是由服务器写的，由客户端读的控制部件数量。每个控制部件由客户

端设备右边和服务器设备左边的控制节点来表示。它的典型应用是在下列图片中做表头监视。两个设备都把控制读计数设为 1，其它属性设为 0。

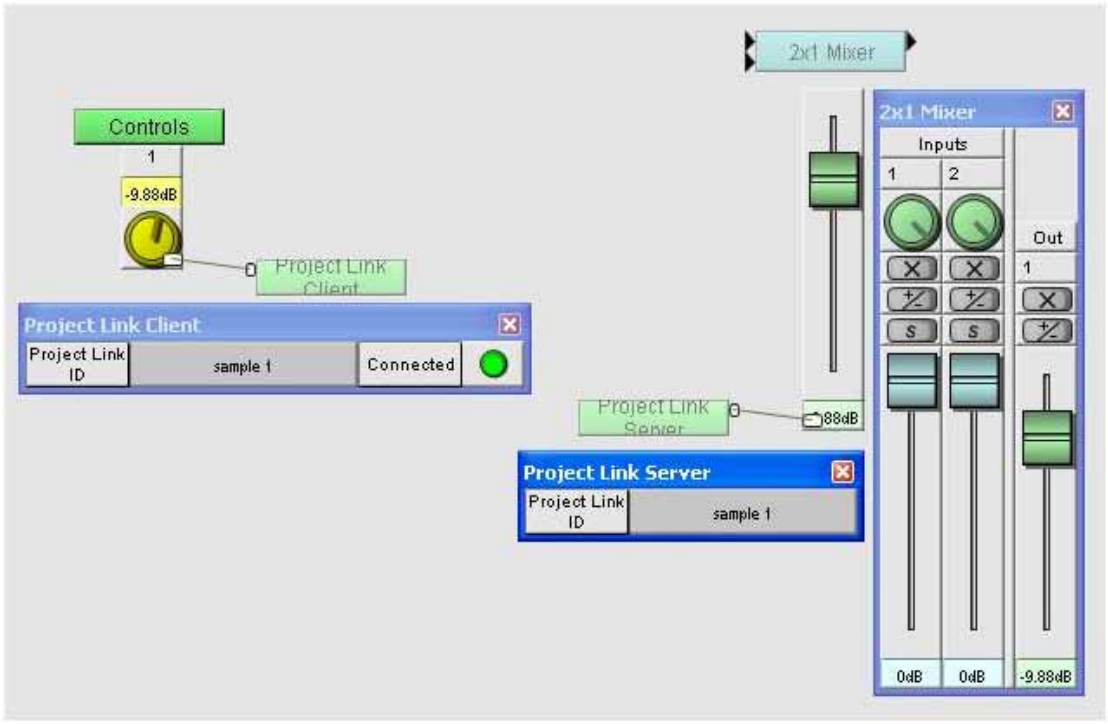
控制平级计数
控制平级计数是可以由服务器或客户端来写的控制部件的数量。每个控制部件由设备左边的一个控制节点表示。这是最有用的配置方法，因为它本质上允许服务器和任意数量的客户端平级成组。注意，尽管这表现得象是平级编组，但是控制连线必须接到控制部件的主控节点上，而不是平级节点。

注意：

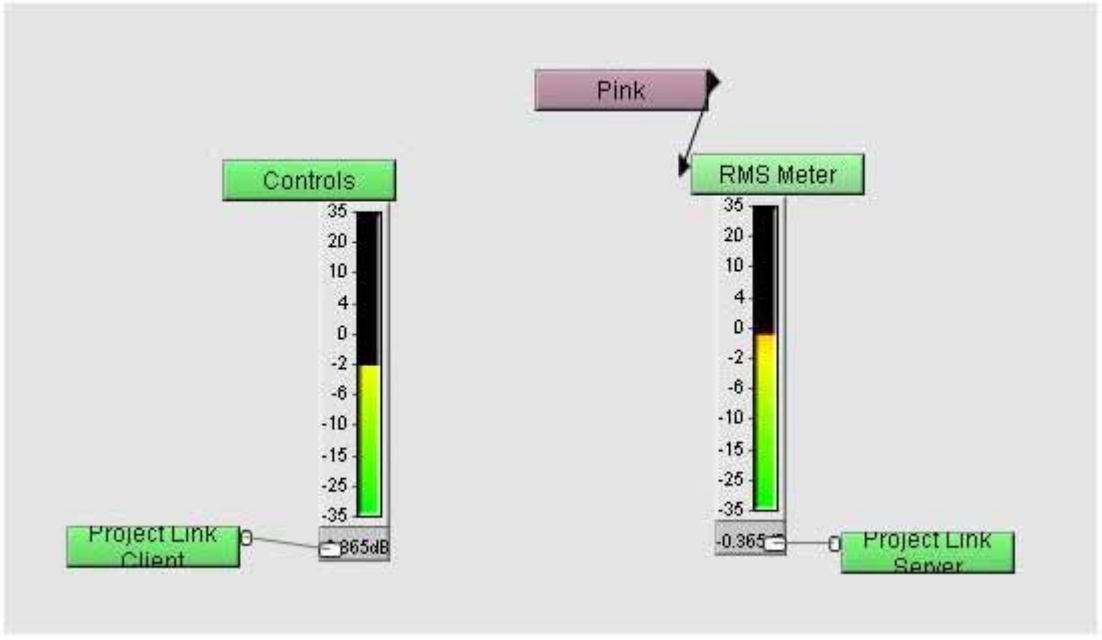
*这三种模式完全可以同时使用。
*在服务器/客户端工程中，当平级控制部件和读/写控制部件同时使用时，你必须在连线模式下拉着线在节点上晃悠，才能从标签上看出来它是写控制节点（read_1,read_2.../write_1,write_2...）还是一个平级控制节点（peer_1,peer_2...）。

实例：

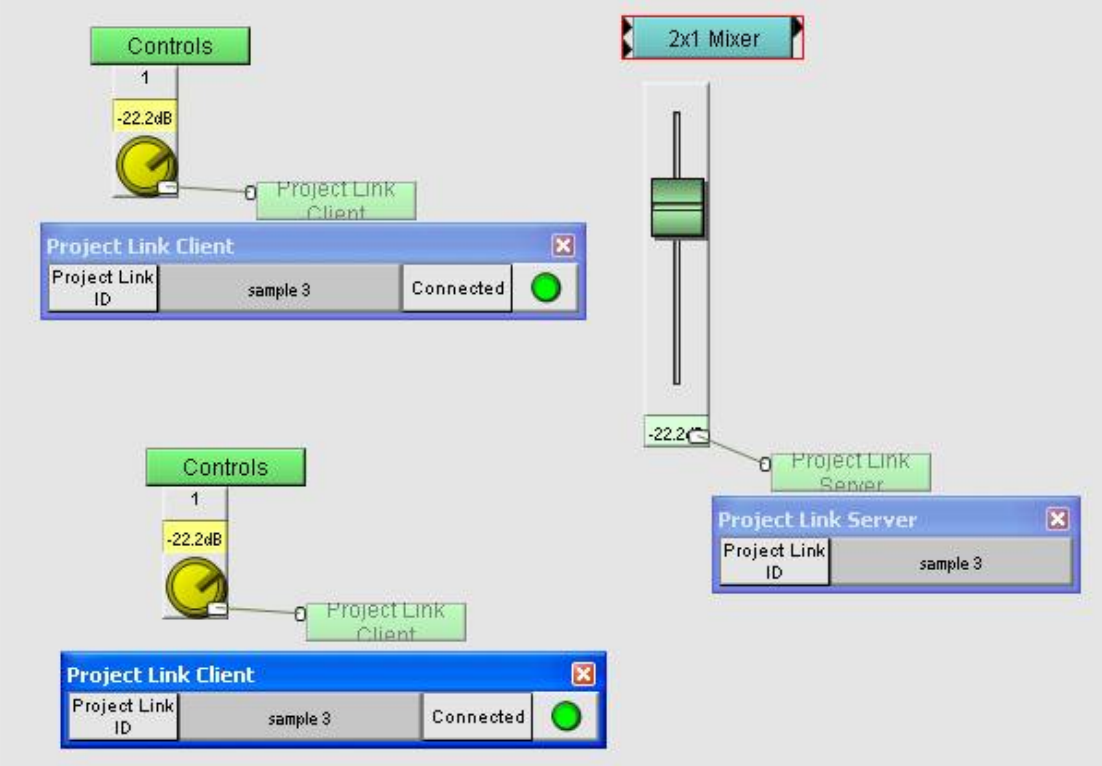
这里是一些实例的抓屏，以供参考。尽管所有的实例，如你所见，是合法的也是可以正常工作的，但是你应该使你的服务器和客户端设备处在同一个网络中不同的工程文件中运行。这些图片仅仅是图例。本功能仅仅用于不同工程中的 NION 之间的控制部件的链接（任何其它的用法都是对资源的浪费）。这两个设备最初是为了使用一个控制管理 PC 来管理数个不同的 NION 工程而创建的。作为一个副产品，它也可以用于在 NION 之间用于管理最多 8 个其他的 NION 工程。控制管理受到更大的限制。



例 1：“ sample 1 ”上的客户端设备控制着同一个 NION 上的一个推子。一般地，相关的服务器设备应该是在同一个网络内的另外一个工程中。注意每个工程都能在不逆向影响另外一个工程的情况下进行单独的重编译。节点是通过设置控制写计数属性为 1 来获得的（记住“ 写 ”是从客户端的角度来说的）。

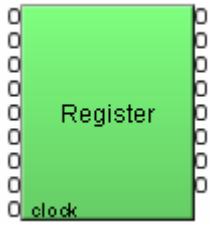


例 2：这是一个利用工程链接来做监视的实例。在这种情况下，右边的 RMS 表头是真正的表头，将它的值通过工程链接发送到左边的表头（其实是一个通用控制部件）。事实上我们在这里没有去控制什么，只是发送供监视的值到另外一个工程。通过将控制读计数属性为 1 来获得节点（记住“ 读 ”是从客户端的角度来说的）。



例 3：这个实例和例 1 很相似只不过这里有两个客户端。这里的节点也是通过控制平级计数来创建的，它允许所有设备平等运行，只要不是以主/从方式连线。改动任何控制部件都会将值发送到其他控制部件。“ 真正的 ” 控制部件还是需要连线到服务器设备。

寄存器 register



总览

寄存器设备在时钟（clock）输入信号的上升沿时，存储连线到它的输入端的控制部件的值，并且在时钟的下降沿将这些值发送给连线到输出端的控制部件。它能创建一些同步状态装置和其它一些在异步环境下无法正确运行的逻辑电路。
注意：这个设备和控制操作器（control operator）的自动链接（auto link）很类似，它们都是判断最适合输入控制部件的值的类型来从输入端获得值（数值，位置或者字符串）。

属性

通道数量
设备可支持的控制部件的通道数量

端口

Input_1 到 input_<通道数量>
输入端连线到需要在时钟上升沿锁存值的控制部件。

output_1 到 output_<通道数量>
输出端连线到需要在时钟下降沿获得锁存中值的控制部件。每个输出控制部件和输入控制部件应当有相同的值类型和范围。

时钟（clock）
控制部件端口连线到一个布尔控制部件，来控制输入数据的锁存和传递。它可以连线到一个通用控制部件中的闪烁指示灯设备。

控制部件：
无

最早发布在 1.1.0 版

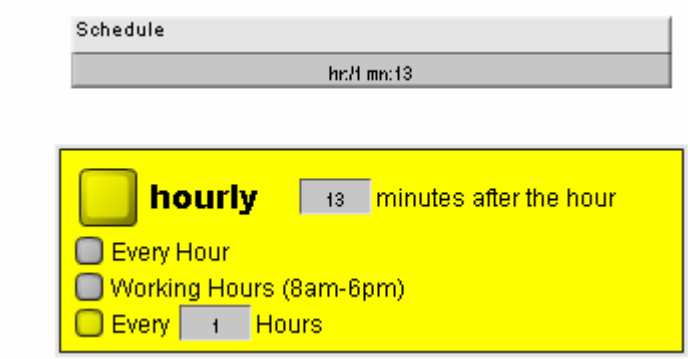
日程发生器 schedule generator

用途
日程发生器有一个易懂的用户界面，能产生日程配置字符串。要使用日程发生器，首先要将它连线到日程控制设备的输入端。如下：

使用
日程发生器的不同部分是互斥的，这就是说你一次只能使用其中的一个部分。这里有大部分默认组合，但是如果你需要一些此处没有提供的东西，你也可以用这个设备为指导，来编配一个自定义匹配字符。

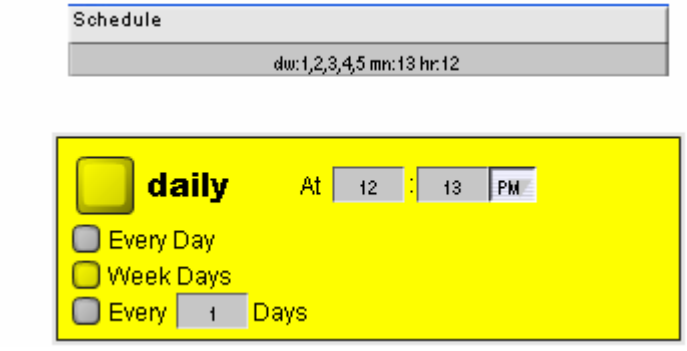
下面我们分别涉及不同的部分：

每小时：



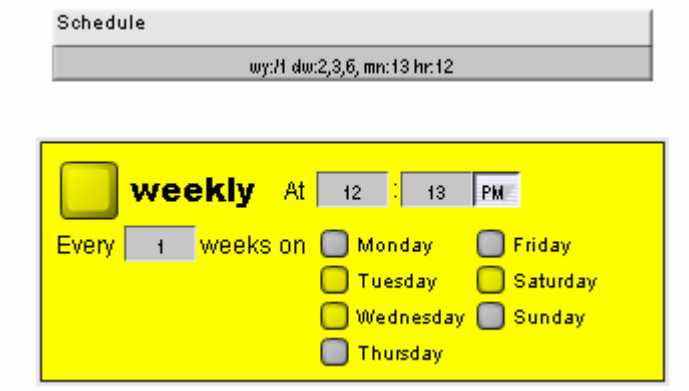
每小时部分可以在小时的基础上开启事件。这里，你的选择可以指定在哪个小时激活以及在该小时的哪分钟执行。注意在工作时间（working hours）选项中，如果不把“minutes after the hour”设为 0，那么只能激活 10 次。这个部分要这么读：
The event occurs hourly <time> minutes after the hour <every hour/working hours/Every x hours>.

每天：



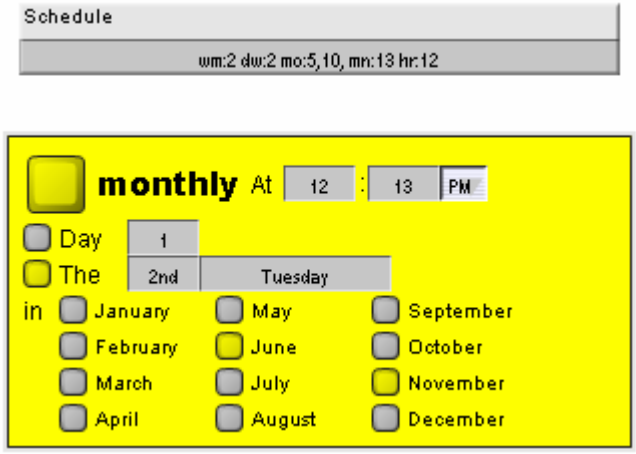
每天意味着在指定的每天的相同时间激活。这个部分这么读：
The event occurs daily at <time> every <day/weekday/number of days>.

每周：



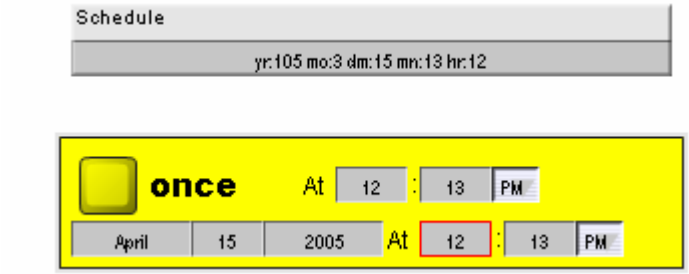
每周是以周为基础地激活。它是这么读：
The event occurs weekly at <time> every <number of weeks> weeks on <day(s) of week>.

每月：



每月地激活。这么读：
The event occurs monthly at <time> on <day of month> in the months of <months>.

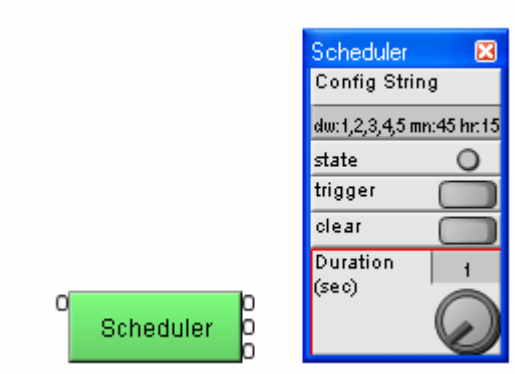
一次：



只激活一次。这么读：
The event occurs once at <time> on <day of the year>.

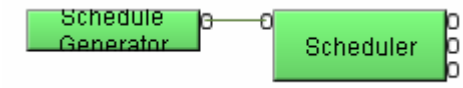
参见：
日程安排设备

日程安排设备 scheduler



日程安排设备用于重复事件的触发。每次事件发生时，start 输出触发，state 输出则在持续控制期间保持高电平。超过持续时间后，state 输出变低，stop 输出触发。日程安排设备可以手动按触发键触发。状态可以按清除键清除（在持续期间）。

你可以自由地使用下面的字符来创建自定义的日程事件，但是这里已经创建了一个专门的日程发生器来简化匹配字符串的创建。日程发生器的输出应该连线到日程安排设备的输入，如下：



使用：
日程发生器通过匹配字符串来配置。匹配字符串是一系列由空格隔开的匹配，每个匹配都是“ 单元：指定 1，指定 2，指定 3 等...的格式。该单元指定时间（年月等）单元来匹配。指定则是定义以什么来匹配这个单元。

有效单元：

yr	year (+1900, so 2004 = 104)
mo	month (January = 0)
wy	week
wm	week of month
dw	day of week (Sunday = 0)
dm	day of month
dy	day of year
hr	hour
mn	minute

使用举例：

mn:32 hr:9	Every day at 9:32 AM
mn:/15	Every 15 minutes (at 0, 15, 30, and 45 minutes after the hour)
mn:/15+4	Every 15 minutes (at 4, 19, 34, and 49 minutes after the hour)
dm:15 mo:8 mn:32 hr:9	Once a year at 9:32 AM on September 16th
dw:1,2,3,4,5 mn:32 hr:9	Every weekday at 9:32 AM

节点：

输入节点：
为日程安排设备接收包含匹配字符串的字符串。它设计用来从日程发生器接收字符串，但是也可以从一个 python 脚本控制设备或者一个通用字符串控制部件获得字符串。脚本设备对日程安排设备的控制功能非常强大。它引入 nionode 内外的事件来作用于日程安排设备。

输出 1-start
Start 是一个触发输出，当输入字符串匹配当前时间时被触发。用于触发控制部件的例子如：预设和快照的加载和保存。

输出 2-stop
Stop 是一个触发输出，当输入字符串不再匹配当前时间时被触发。简单地用于事件完成后的触发。

输出 3-state
State 是一个布尔输出，显示日程事件的当前状态。输出在事件的持续期间为高，在事件结束后为低。

参见：
日程发生器

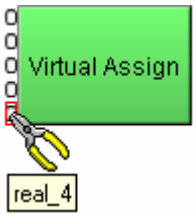
虚拟指派设备 virtual assignment

总览

虚拟指派设备允许一个单独的“ 虚拟 ”控制部件可选地编组到很多“ 真实 ”的控制部件。它可以用来创建紧凑的用户界面，使用两种控制部件（选择器和虚拟控制部件）来代替一堆独立的控制部件。当选择器变动时，虚拟控制部件控制对应的真实控制部件的值。当真实的控制部件改变值时，虚拟控制部件也被更新；当虚拟部件调整时，选中的真实控制部件也会被更新。

属性
虚拟控制部件数量
真实控制部件数量

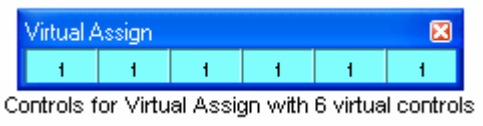
端口
注意：连线模式下，鼠标浮在端口上可以识别它。有一个标题条来显示端口的 ID 号。这个 ID 有如下两种类型：



virtual_1 through virtual_< Number of virtual controls >
输入控制端口将连线到“ 虚拟 ”控制部件。典型地，它们应该是被配置成与真实控制部件具有相同类型和范围的通用控制部件。

real_1 through real_< Number of real controls >
输入控制端口将连线到“ 真实 ”控制部件。它们连线到设计中需要的控制部件。这些部件应该具有相同的类型和范围。它们可以连线到相应的多通道处理设备的控制部件，或者是一个多频段设备的多个频段的控制部件。

控制部件：



Controls for Virtual Assign with 6 virtual controls

Selector_1 through selector_< Number of virtual controls >

这个控制部件选择哪个真实的控制部件与对应的虚拟控制部件的编组。这个控制部件的选择范围为整数的从 1 到<真实控制部件数量>