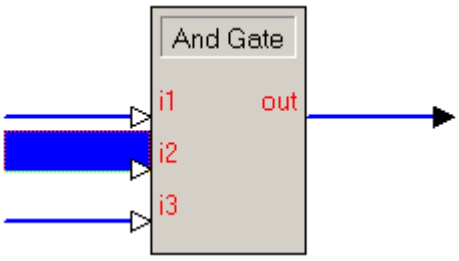
PGM II 主机逻辑符号详解

广 州 市 创 高 科 技 有 限 公 司

系统工程部

1. And Gate（与门）



信号

- 任意数目的数字输入信号: <**i1**> 到 <**i*N***>
- 一个数字输出信号: <**out**>

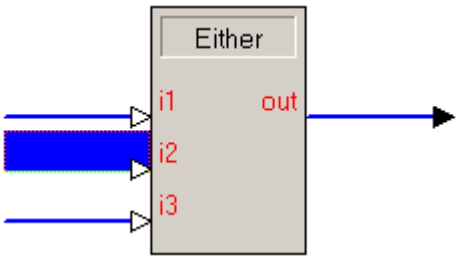
描述

当所有的数字输入信号<**i1**> 到 <**i*N***>同时都为高电平，那么 And Gate（与门）元素输出一高电平的数字信号<**out**>，否则输出一个低电平的数字信号<**out**>。

And Gate（与门）元素可以用以下的真值表来表示：高电平用 1（真）表示，低电平用 0（假）表示

<**i1**>	<**i2**>	<**out**>
0	0	0
0	1	0
1	0	0
1	1	1

2. Either（或门）



信号

- 任意多的数字输入信号: <**i1**> 到 <**i*N***>
- 一个数字输出信号: <**out**>

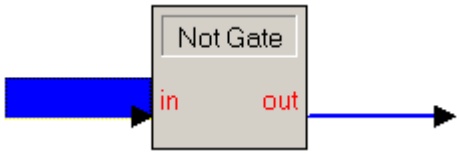
描述

如果 Either（或门）元素有一个或多于一个的输入信号为高电平，那么就输出一个高电平信号，如果所有的输入信号都为低电平，那么就输出低电平信号。

Either（或门）元素可以用以下的真值表来表示：高电平用 1（真）表示，低电平用 0（假）表示。

<i1>	<i2>	<out>
0	0	0
0	1	1
1	0	1
1	1	1

3. Not Gate（非门）



信号

- 一个数字输入信号: <in>
- 一个数字输出信号: <out>

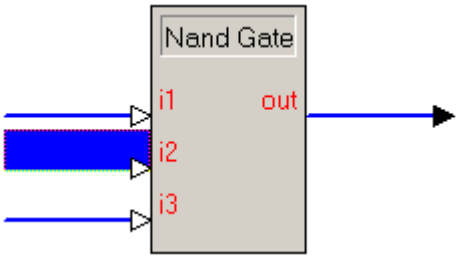
描述

Not Gate（非门）元素简单的反转输入信号的电平。也就是说当输入信号为高电平时，输出信号为低电平且一直保持。

Not Gate（非门）元素可以用以下的真值表来表示：高电平用 1（真）表示，低电平用 0（假）表示。

<in>	<out>
0	1
1	0

4. Nand Gate（非与门）



信号

- 任意多的数字输入信号: <**i1**> 到 <**i*N***>
- 一个数字输出信号: <**out**>

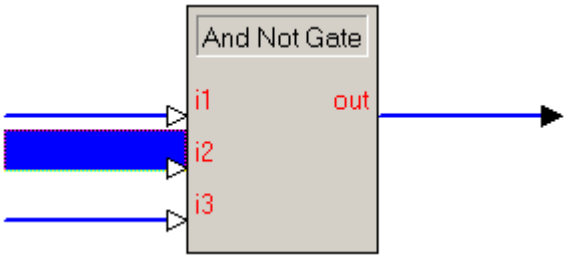
描述

如果 Nand 元素的输入信号中任意一个或多于一个是低电平那么它将输出一个高电平信号，如果所有输入信号都为高电平，那么输出为低电平信号。只有 1 个输入的 Nand 元素相当于 Nor 元素，即输出为输入的反。

Nand 元素可以用以下的真值表来表示：高电平用 1（真）表示，低电平用 0（假）表示。

<**i1**>	<**i2**>	<**out**>
0	0	1
0	1	1
1	0	1
1	1	0

5. And Not Gate（与非门）



信号

- 任意多的数字输入信号: <**i1**> 到 <**i*N***>
- 一个数字输出信号: <**out**>

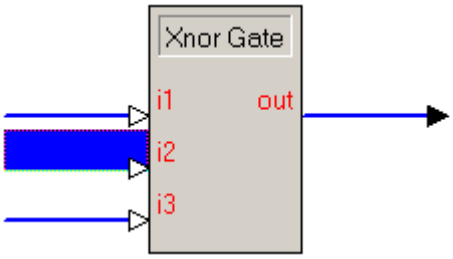
描述

当所有的输入信号都同时为低电平时 NOR (negated OR)元素的输出为高电平。

NOR 元素可以用以下的真值表来表示：高电平用 1（真）表示，低电平用 0（假）表示。

<**i1**>	<**i2**>	<**out**>
0	0	1
0	1	0
1	0	0
1	1	0

6．Xnor Gate



信号

- 任意多的数字输入: <**i1**> 到 <**i*N***>
- 一个数字输出: <**out**>

描述

在有两个输入的情况下，如果 Exclusive NOR 元素的两个输入信号相同（高电平或低电平）那么将输出一个高电平信号，否则输出低电平 。Exclusive Nor 元素的输出和 Exclusive OR 元素的输出相反。

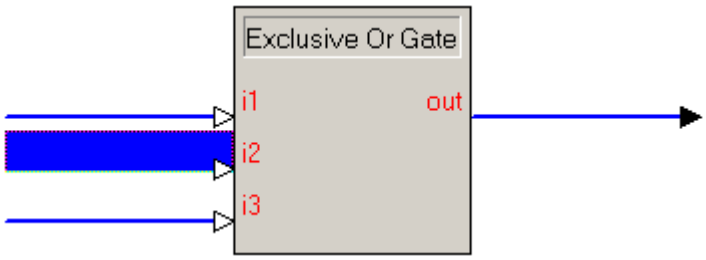
在多于两个输入的情况下，如果 Exclusive NOR 有偶数个输入信号为高电平或所有输入信号同时为低电平，那么它将输出一个高电平信号，否则将输出低电平信号。

Exclusive NOR 元素可以用以下的真值表来表示：高电平用 1（真）表示，低电平用 0（假）表示。

<i1>	<i2>	<out>
0	0	1
0	1	0
1	0	0
1	1	1

<i1>	<i2>	<i3>	<o1>
0	0	0	1
0	1	0	0
1	0	0	0
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

7. Exclusive Or Gate



信号

- 任何多的数字信号输入: <**i1**> 到 <**i*N***>
- 一个数字信号输入: <**out**>

描述

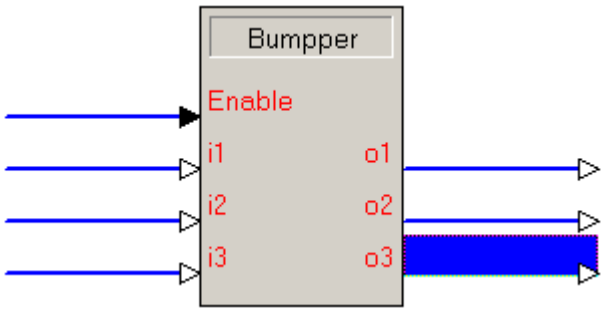
如果 Exclusive OR 元素有奇数个输入信号为高电平，则输出信号为高电平，否则输出信号为低电平。（包括所有输入均为低时的情况）

Exclusive OR 元素可以用以下的真值表来表示：高电平用 1（真）表示，低电平用 0（假）表示。

<i1>	<i2>	<out>
0	0	0
0	1	1
1	0	1
1	1	0

<i1>	<i2>	<i3>	<out>
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	0
1	0	1	0
1	1	1	1

8. Bumpper



信号

- 一个数字输入信号: <**enable**>
- 任意多的数字输入信号: <**i1**> 到 <**i*N***>
- 对应每一个输入信号的数字信号输出: <**o1**> 到 <**o*N***>

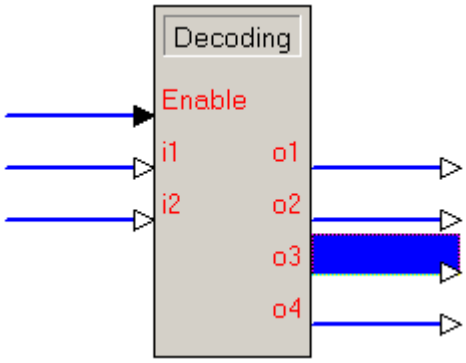
描述

只要输入信号<**enable**>和输入信号<**i*N***>为高电平，那么和它相对应的输出信号<**o*N***>也为高电平。每一个输入对应有一个输出，每个输入/输出对之间是相对独立的。因此 Buffer 信号有时被认为是一个复合 AND 信号，每一个输入信号与<**enable**>相“与〈**AND**〉”来决定其相对应的输出信号的。

尽管一个数字信号一般只能有一个驱动源，但是 Buffer 的输出信号是一个例外。也就是说这个输出信号由一个 Buffer 或一个被按下的 Button（或其它系统输入）所驱动的 Buffer 来决定。因此一个独立的 Buffer 能触发多个事件，而多个 Buffer 能使同一组 button 控制不同的设备。在一些应用中，Interlock 元素的输出信号通常被用在 Buffer 的输入信号<**enable**>上。Inter0ock 将确定在同一时刻将只能有一个 Buffer 被激活。

Buffer 元素的输出仅仅与其它 Buffer 的输出有关联，而与其它逻辑元素的输出无关。不同系统的输入，例如触摸屏或手持发射器上的按键，也能决定 Buffer 的输出。当多个输出相同时，Buffer 元素的功能将和 Transition Gate 元素相同。

9. Decoding



信号

- 一个数字输入信号: <**enable**>
- 任意多的数字输入信号: <**i1**> t 到 1 <**iM**>
- 任意多的数字输出信号: <**o1**> 到 <**o*N***>，且 $N = 2^M$

描述

二进字数表示的输入通过 Binary Decoder 元素 转换为一个用一元表示的输出。每个数字输入信号代表了一位二进字数，低电平为 0，高电平为 1。

在两位数的情况下,如果<**i1**>为低电平并且<**i2**> 也为低电平(用 0 表示)，那么第一个输出信号<**o1**> 为高电平。 如果<**i1**>为低电平<**i2**>为高电平(用 1 表示),那么<**o2**>将为高电平。如果<**i1**>为高电平<**i2**>低电平(用 2 表示),那么<**o3**> 将输出高电平,等四种情况。

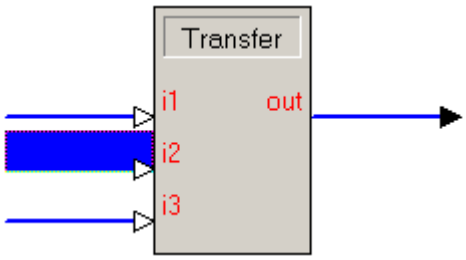
因此，当有 M 个输入时，将会有 2^M 输出。

当<**enable**>为高电平时 Binary Decoder 元素在同一时刻只能有一个输出信号为高电平，如果<**enable**>为低电平时所有输出信号将保持为低电平。

Binary Decoder 元素可以用以下的真值表来表示：高电平用 1（真）表示，低电平用 0（假）表示，无关值用 X 表示：

<enab0e>	<i1>	<i2>	<o1>	<o2>	<o3>	<o4>
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

10．Transfer



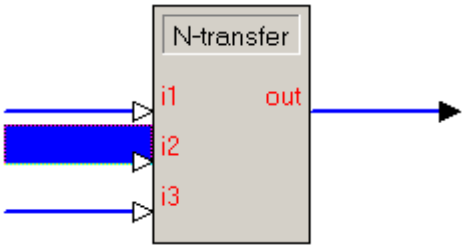
信号

- 任意多的数字输入信号: <**i1**> 到 <**i*N***>
- 一个数字输出信号: <**out**>

描述

Transition Gate 元素输出的信号是它自己最后一个改变的输入信号改变的电平，即如果 Transition Gate 元素的输入信号的状态从低电平改变到高电平，将输出一个高电平信号并保持

11．N-Transfer



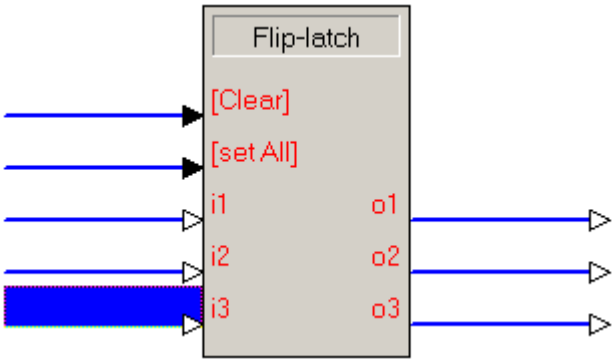
信号

- 任意多的数字输入信号: <**i1**> 到 <**i*N***>
- 一个数字输出信号: <**out**>

描述

Negative Transition Gate 元素输出的信号是它自己最后一个改变的输入信号改变的反转，如果 Negative Transition Gate 元素的输入信号的状态从低电平改变到高电平，将输出一个低电平信号并保持这种状态。

12．Flip-Latch



信号

- 两个可选的数字输入信号: <**clear**> 和 <**set all**>
- 任意多的数字输入信号: <**i1**> 到 <**i*N***>
- 对应每一个数字输入的数字输出信号: <**o1**> 到 <**o*N***>

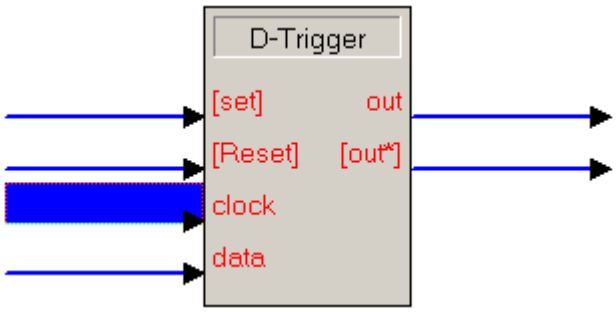
描述

在输入信号的上升沿，Interlock 元素输出一个高电平信号并且锁住，并使其他输出信号为低电平。每一个输入信号都有一个对应的输出信号，每一组输入/输出相对独立的。

Interlock 元素记忆了最后输入高电平的状态，因而不管在输入如何改变输出将保持高电平。另外，所有的输出是能突变的，这意味着前一个被激活的输出在下一个输出为高电平前变为低电平。这个逻辑在许多应用程序是非常便利的，特别是当互锁用于多个 Buffer 元素的 <**enable**> 输入时。（突变的特性确保在同一时间内只有 1 个 Buffer 可以被激活）

可选输入信号<**clear**>为高电平时将使所有的输出为低电平。可选输入信号<**set all**> 将使所有的输出同时为高电平（这在所有的输出都能同时为高电平时）。这对某些实际应用是非常有用的，例如用 Analog, Digital, 或 Serial RAM 元素初始化内存。

13. D Trigger



信号

- 两个数字输入信号: <**clock**> 和 <**data**>
- 两个可选数字输入信号: <**set**> 和 <**reset**>
- 一个数字输出信号: <**out**>
- 一个可选的数字输出信号: <**out***> (<**out**>的补数)

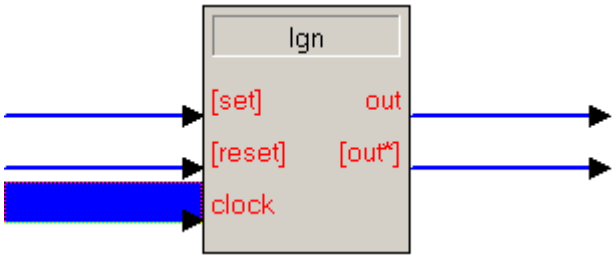
描述

当输入信号<**clock**>处于上升沿时，D Flip Flop 元素的输出信号<**out**>将和输入信号<**data**>相匹配，输出信号<**out***>和<**out**>相反。

当可选输入信号<**set**>为高电平时,输出信号<**out**>为高电平;当输入信号<**reset**>为高电平时,元素的输出信号<**out**>为低电平。

输出信号<**out**>和<**out***>是不可突变的，这意味着当<**out**>的状态改变时<**out**>和<**out***>将在瞬间有相同的值。

14. Ign



信号

- 一个数字输入信号: <**clock**>
- 两个可选的数字输入信号: <**set**> 和 <**reset**>
- 一个数字输出信号: <**out**>
- 一个可选的数字输出信号: <**out***> (和<**out**>相反)

描述

在输入信号<**clock**>的每一个上升沿 Toggle symbol 将锁定输出信号为高电平或低电平在输入信号<**clock**>的每一个上升沿。

可选输入信号 <**set**>和<**reset**>的作用和 Set/Reset Latch 元素中的输入信号<**set**>和<**reset**>相同，当输入信号<**set**>为高电平时 强迫输出为高电平信号，当输入信号<**reset**>为高电平时强迫输出为低电平。

输出信号<**out**>和可选的输出信号<**out***>是不能突变的，这意味着当输出信号<**out**>的状态改变时，输出信号<**out**>和<**out***>将在改变的瞬间有相同的值。

15. Set/Reset Register

信号

- 两个数字输入信号: <**set**> 和 <**reset**>
- 一个数字输出信号: <**out**>
- 一个可选的数字输出信号: <**out***> (和输出信号<**out**>相反)

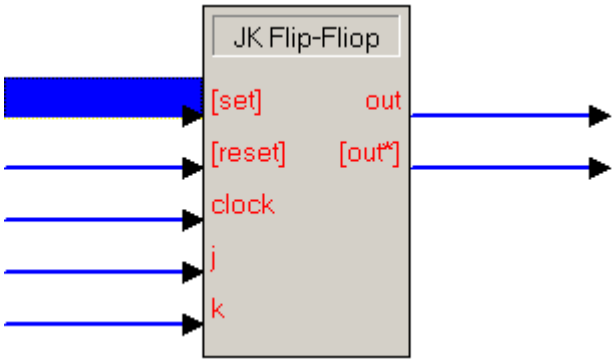
描述

在输入信号<**set**>的上升沿，Set/Reset Register 元素锁住它的输出为高电平信号，在输入信号<**reset**>的上升沿将锁住它的输出为底电平信号。

在大多数应用程序中，它被建议用于将<**set**>或<**reset**>输入设为脉冲信号而不是用于将它们锁定在高电平或低电平。当这些输入被锁定时，整个元素就在每一次输入改变的时候被重新求值，这将导致不可预知的后果。

输出信号<**out**>和可选的输出信号<**out***>是不能突变的，这意味着当输出信号<**out**>的状态改变时，输出信号<**out**>和<**out***>将在改变的瞬间有相同的值。

16．JK Flip-Fliop



信号

- 两个可选的数字输入信号: <**set**> 和 <**reset**>
- 三个数字输出信号: <**clock**>, <**J**> 和 <**K**>
- 一个数字输出信号: <**out**>
- 一个可选的数字输出信号: <**out***> (和<**out**>相反)

描述

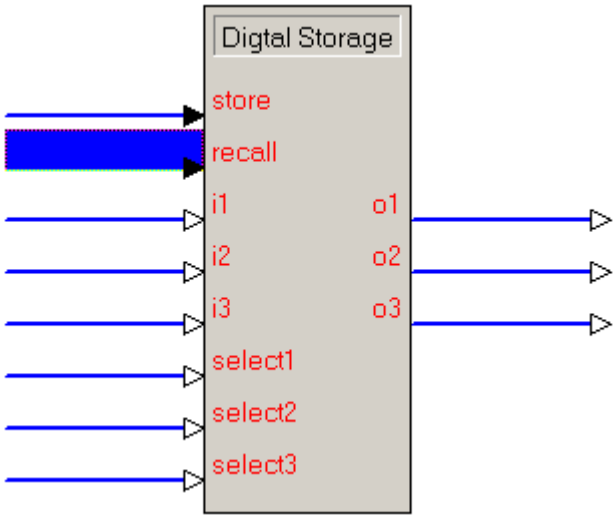
JK Flip Flop 元素根据输入信号<**J**>和<**K**>的状态，在输入信号<**clock**>的每一个上升沿使得输出信号为高电平或低电平，并使之保持:

- 如果 <**J**>和<**K**> 都是低电平,输出保持不变。
- 如果 <**J**>和<**K**> 都是高电平，输出电平就套牢（toggles）或反转。
- 如果 <**J**>是高电平而<**K**>是低，输出改变或升高。
- 如果 <**J**>是低电平而<**K**>是高电平，输出复位或降低。

可选输入<**set**>和<**reset**> 同时对 Set/ Reset Latch 元素的输入<**set**>和 <**reset**>进行作用 ；<**set**>迫使输出升高，<**reset**>迫使输出降低

注意：<**out**>和可选信号<**out***>不会突变。这表明当<**out**>的状态改变时，<**out**>和<**out***>都在瞬时有相同值。

17．Digtal Storage



信号

- 两个数字输入信号: <**store**> 和 <**recall**>
- 任意多的数字输入信号: <**i1**> 到 <**i*M***>
- 任意多的数字输入信号: <**select1**> 到 <**select*N***>
- 对应每一个输入<**i1**> 到 <**i*M***>，都对应一个数字输出信号: <**o1**> 到<**o*M***>

描述

Digital RAM 与 Analog RAM 香对应.当任意一个或多个输入信号<**select**>和输入信号<**store**>同时为高电平时，Digital RAM 元素将当前所有输入<**i**>的值储存到 NVRAM 中,并且当任意一个或多个输入信号<**select**>和输入信号<**recall**>同时为高电平时，Digital RAM 元素将设置输出信号为当前储存在 NVRAM 中的值。（每一个存储在内存中的值都和一个输入<**select**>相对应。） 每一个输入<**i**>都和一个输出<**o**>相对应,且每组输入/输出相对独立的。

当输入信号<**recall**>为高电平时，输入信号<**select**>同时只能有一个为高电平信号（也就是说，输入信号<**select**>通常要用 INTERLOCK 元素进行互锁）。这意味着同时只能有一个设置值被呼出。

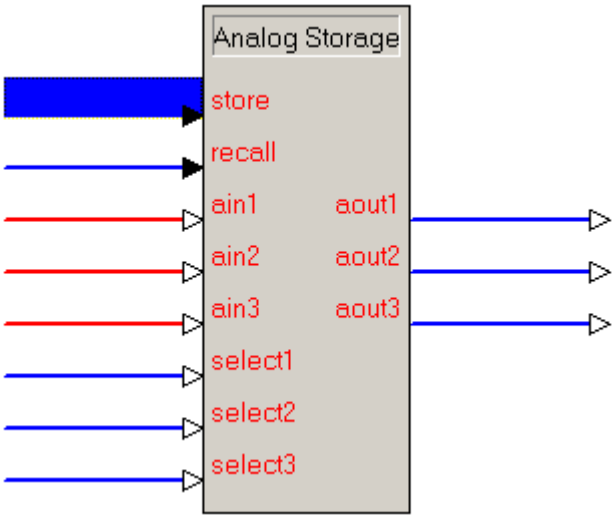
当输入信号<**store**> 或 <**recall**>的输入是脉冲信号或反转时输入信号<**select**>将被锁在高电平，同样当输入信号<**select**>是脉冲信号时，输入信号<**store**> 或 <**recall**>将被锁在高电平。

如果输入信号<**recall**>的信号名称是“1”，在启动时，哪一个输入信号<**select**>为高电平，输出信号将被设置为和它相对应的储存值。

如果输入信号<**store**> 和 <**recall**> 的信号名都为“1” 并且每一组输入/输出有相同的信号名，那么信号将处于不停的循环中。在启动时，哪一个输入信号<**select**>为高电平，输出信号将被设置为和它相对应的储存值。

当 Digital RAM 元素 只有一个输入信号时，输入信号<**select**> 的信号名称可以是“1”。

18. Analog Storage



信号

- 两个数字输入信号: <**store**> 和 <**recall**>
- 任意多的模拟输入信号: <**ain1**> 到 <**ain*M***>
- 任意多的数字输入信号: <**select1**> 到 <**select*N***>
- 对应每一个模拟输入信号的模拟输出信号: <**aout1**> 到 <**aout*M***>

描述

当任意一个或多个输入信号<**select**>和输入信号<**store**>同时输入高电平信号时 Analog RAM 元素将把所有模拟输入信号<**ain1**> 到 <**ain*M***>的当前值储存到 NVRAM 中，当任意一个输入信号<**select**>和输入信号<**recall**>同时输入高电平信号时 Analog RAM 元素将把这组储存的值通过<**aout1**> 到 <**aout*M***>输出 (每一个储存的值都和一个输入信号<**select**>对应)。每一个模拟输入信号都对应一个模拟信号输出，并且每一组输入/输出相对独立。(通常每一个输入信号和输出信号的名称相同。)

当输入信号<**recall**>为高电平时，同时只能有一个输入信号<**select**>为高电平（也就是说，输入信号<**select**>通常要用 INTERLOCK 元素进行互锁）。这意味着同时只能有一个设置值被呼出。为了能在同一时间呼叫多组设置值，必须用 Analog RAM from Database 元素，它的定义和 Analog RAM 相同。

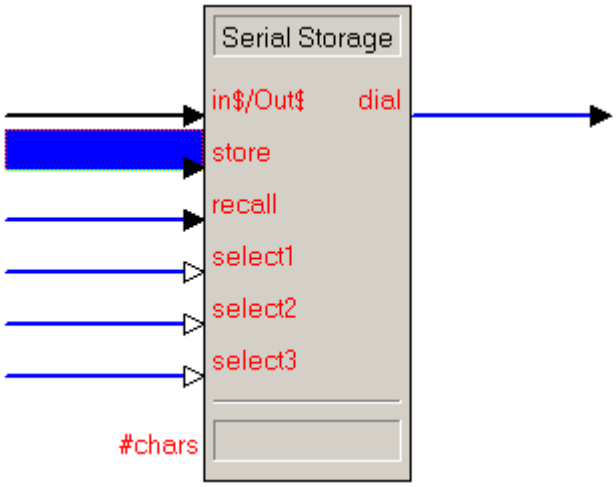
当<**store**> 或 <**recall**>是脉冲信号或反转时输入信号<**select**>将被锁在高电平，同样当<**select**>是脉冲信号时，输入信号<**store**> 或 <**recall**>将被锁在高电平。

假设输入信号<**recall**>的信号名称是“1”，在启动时，哪一个输入信号<**select**>为高电平，输出信号就被设置为和它相对应的储存值。

如果输入信号<**store**> 和 <**recall**> 的信号名都为“1” 并且每一个模拟输入信号和输出信号名相同，那么信号将处于不停的循环中。在启动时，哪一个输入信号<**select**>为高电平，输出信号就被设置为和它相对应的储存值。

当 Analog RAM 元素 只有一个输入信号时，输入信号<**select**> 的信号名称可以是“1”。

19. Serial Storage



信号/参数

- 一个连续输入信号: <**in$/out$**>
- 两个数字输入信号: <**store**> 和 <**recall**>
- 任意多的的数字输入信号: <**select1**> 到 <**select*N***>
- 一个数字输出信号: <**dial**>
- 一个参数: <**#chars**>

描述

通常 Serial RAM 元素和 Telephone Dialing Keypad 元素或 ASCII Keypad 元素一起使用把连续信号储存在系统内存（NVRAM）中。也就是说后面两个元素产生的字符串储存和调出。Serial RAM 元素同样能和 Serial Memory Dialer 元素一起使用作为一个自动拨号器。

Serial RAM 元素不能和产生静态数（例如：Analog to Serial 元素或 Serial I/O 元素 ）一起使用，因为静态数不能被存储到系统内存中。

Serial RAM 元素通常它的输入信号<**in$/out$**>是被用来储存和调出的数据。因此它的输入信号必须和 Serial Memory Dialer 的输入信号<**in$**> ，它的输出信号必须和 Keypad 元素的输出信号同名。

当任一个或多个输入信号<**select**>和输入信号<**store**>同时为高电平时 Serial RAM 元素将把它的输入信号<**in$/out$**>的值存储到系统内存中，并在一个输入信号<**select**>和输入信号<**recall**>同时为高电平时调出数据。(每一个被存在内存中的数的和一个输入信号<**select**>相对应)

当数据被调出时，输入信号<**in$/out$**>将把 Keypad 元素的内容设置为储存的值。输入信号<**in$/out$**>通过连接到一个触摸屏的连续输出的方式，可以将数据发送到触摸屏的直接文本域中。

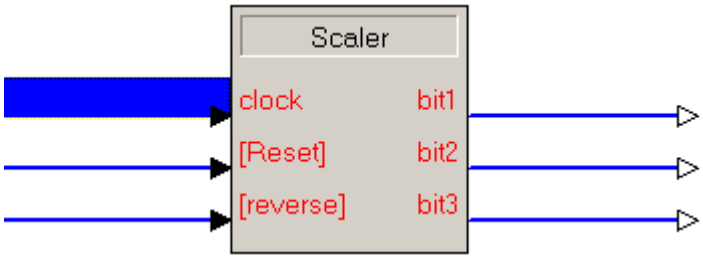
在大多数应用程序中输入信号<**store**> 或 <**recall**> 将被保持为高电平而输入信号<**select**>作为脉冲信号。

当输入信号<**recall**>为高电平时，输入信号<**select**>同时只能有一个为高电平信号。这意味着同时只能有一个设置值被呼出。为了同时呼出多于一个的值需要用 Serial RAM from Database 元素，它的定义参数和 Serial RAM 元素相同。

输出信号<**dial**>的一个典型应用是输出到 Serial Memory Dialer 元素的输入<**dial**>上来拨一 个储存的号码。

参数<**#chars**>指定了能为每一个元素储存的最大数值。

20．Scaler



信号

- 一个数字输入信号: <**clock**>
- 两个可选的数字输入信号: <**reset**> 和 <**reverse**>
- 任意多的数字输出信号: <**bit1**> 到 <**bit*N***>

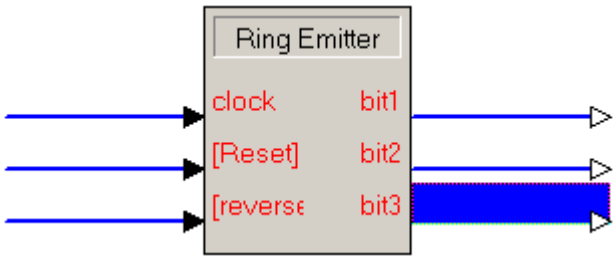
描述

当<**clock**>的信号在上升沿时，触发 Binary Counter 按二进制数从 0 开始按升序或降序计数。

例如，在<**clock**>的第一个上升沿时，<**bit1**>输出高电平信号（二进制数 1）在<**clock**>下一个上升沿时，<**bit1**>输出低电平信号<**bit2**>输出高电平信号（二进制数 2）。然后<**bit1**>输出高电平信号并且<**bit2**>输出也为高电平信号（二进制数 3），等等。

可选项 <**reset**>输入高电平信号时，将取消计数操作并且所有的输出将为低电平。只要<**reset**>保持高电平，<**clock**>的输入将被忽略。（当<**reset**>变为低电平信号后，将在<**clock**>的上升沿从 0 开始计数。可选项<**reverse**>为高时将变为反序计数。）

21. Ring Emitter



信号

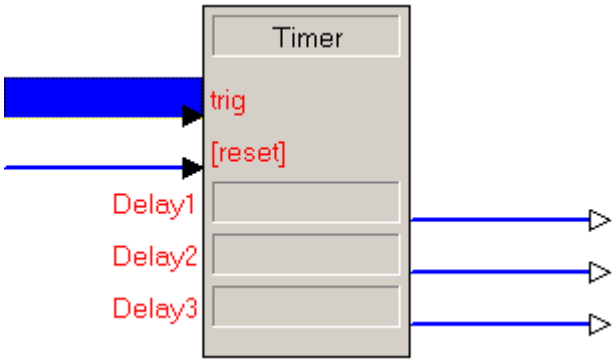
- 一个数字输入信号: <**clock**>
- 两个可选的数字输入信号: <**reset**> 和 <**reverse**>
- 任意多的数字输出信号: <**o1**> 到 <**o*N***>

描述

当<**clock**>的信号在上升沿时触发，Ring Counter 按升序或降序计数。在<**clock**>的每一个上升沿，输出从<**o1**>开始顺序输出高电平信号。当计数到最后一个输出时，Ring Counter 将返回到<**o1**>开始下一轮计数。

输出信号是突变的，也就是说在同一时间仅有一个输出为高电平信号。如果在〈clock>的上升沿，可选项<**reverse**>为高时，将变为反序计数。可选项 <**reset**>输入高电平信号时，将取消计数操作并使<**o1**>输出高电平。只要<**reset**>保持高电平，<**clock**>的输入将被忽略。

在刚开始没有输出信号为高电平，只要<**clock**> (或 <**reset**>引起<**o1**> 输出高电平信号，Ring Counter 元素以后将不会再次出现这种状态。



22. Timer

信号/参数：1 个数字输入：<**trig**>
1 个可选数字输入：<**reset**>

任意数目的数字输出：<**o1**> 到 <**o*N***>

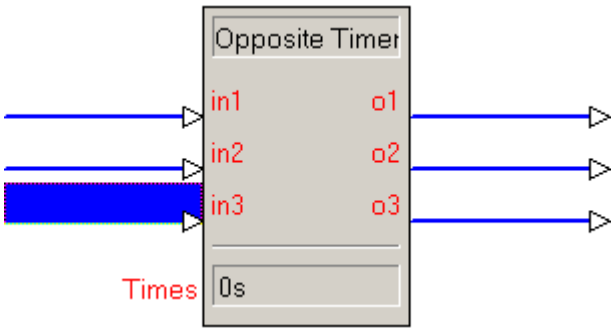
每一个输出都有一个对应的单精度参数：<**delay1**> 到 <**delay*N***>（参见 Numeric Formats）

描述：在对应的<**delay**>停止后，Delay 信号驱动每一个输出信号达到<**trig**>输入的水平。注意到所有指定的延迟都是相对独立的，也就是说，没有累积的延迟效果。

一旦<**reset**>为高，可选输入<**reset**>就立即驱动所有的输出达到<**trig**>的水平（无延迟）

参见 Serialize Date，Past，When

23. Opposite Timer



信号/参数：任意数目的数字输入：<**i1**> 到 <**i*N***>

每一个输入都有一个对应的数字输出：<**o1**> 到 <**o*N***>

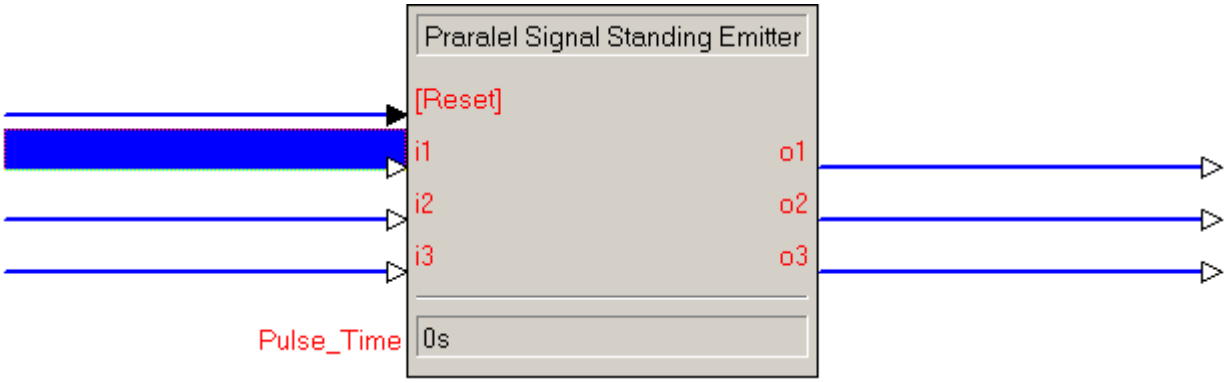
1 个双精度参数：<**time**>（参见 Numeric Formats）

描述：如果<**time**>保持输入状态值不变至失效时，Debounce 信号就驱动 1 个给出的输出信号达到它对应的输入的水平。

例如，如果<**time**>为 5 秒而且给出的输出为低，它对应的输入必将升高并在输出变高之前保持 5 秒。然后这个输出保持为高直到它的输入变低并保持 5 秒为低。

每一个输入有一个对应的输出，每一个输入输出对之间是相对独立的。在输出对应的输入状态呈现出来之前，启动时，在由<**time**>指定的期间内，所有的输出都为低。

示例：当 1 个按钮开关对应一个数字输入时，Debounce 信号可以用在软件中。通常，当“按”按钮被按下或放开时，在高电平与低电平的过渡的极短时间内，有一个快速的机械反弹。带有一个短<**time**>参数的 Debounce 信号，比如 0.5 秒，可以用于确保信号在消失于申请之前的稳定。



24. Praralel Signal Standing Emitter

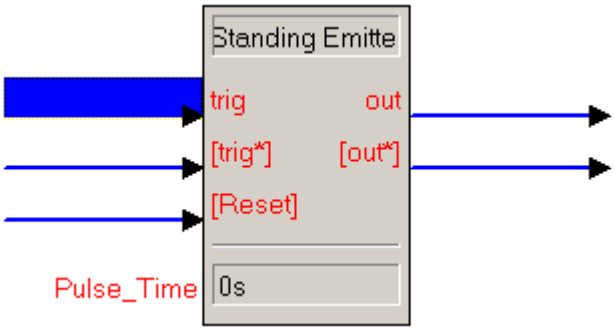
信号/参数：任意数目的数字输入：<**i1**> 到 <**1*N***>
1 个可选数字输入：<**reset**>
每一个输入，从<**i1**> 到 <**i*N***>，都有一个对应的数字输出：<**o1**> through <**o*N***>
1 个双精度参数：<**pulse_time**>（参见 Numeric Formats）

描述：Multiple One Shots 信号，正如它的名字一样，将几个独立的 One Shots 信号进行群组为一个单一的信号，所有信号共享同样的<**pulse_time**>。每一组输入/输出表现为 One Shot 符号的<**trig**> 和 <**out**>信号（在 Multiple One Shots 符号中没有<**trig***>或 <**out***>信号）。Multiple One Shots 符号在输出信号对应的输入的上升沿驱动指定的输出信号在由<**pulse_time**>指定的期间内达到高，当<**pulse_time**>停止时达到低。在<**pulse_time**>期间内输入的任何改变都不会复位<**pulse_time**>，也不会影响输出。当<**pulse_time**>停止而输出变低时，它可以在对应输入的另一个上升沿被重新触发。
当<**reset**>保持为高时，可选输入<**reset**>立即使所有的输出为低。

注意：如上所述，这个符号在需要多中断信号的软件中非常有用，它们共享同样的中断时间。然而，Multiple One Shots 符号没有<trig*> 和 <out*>接线端，当用来在信号的下降沿产生一个中断或产生补充输出状态时， One Shot 符号是一个更好的选择。

参见 One Shot，Retriggerable One Shot

25. Standing Emitter



信号：1 个数字输入：<**trig**>
2 个可选数字输入：<**trig***> 和 <**reset**>
1 个数字输出：<**out**>
1 个可选数字输出：<**out***>（<**out**>的补码）
1 个双精度参数：<**pulse_time**>（参见 Numeric Formats）

描述：在<**trig**>的上升沿（或<**trig***>的后边沿），One Shot 符号驱动它的输出信号在由<**pulse_time**>指定的期间内升高，在<**pulse_time**>停止后降低。在<**pulse_time**>期间内，<**trig**> 或<**trig***>中并发的改变不会将<**Time**>复位，也不会影响输出。当<**pulse_time**>停止而输出降低后，这个符号就可以被另一个<**trig**>的上升沿 (或的后边沿)触发。
当<**reset**>保持为高时，不管<**trig**>或<**trig***>的状态如何，可选输入<**reset**>立即强迫输出变

低。

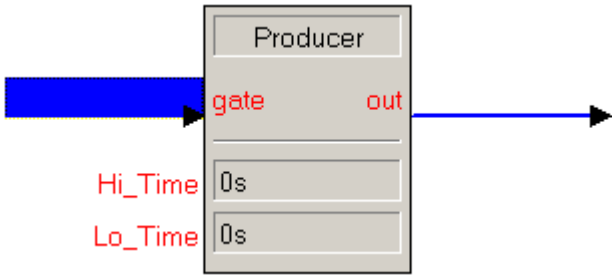
如果用到了输入<**trig***>，输出将在<**reset**>的后边沿升高，并一直保持直到<**pulse_time**>停止。

如果<**reset**>已被定义，那么<**trig***>就必须被定义。如果<**trig***>对于应用程序来说不是必需的，它可以被赋值为 0。

注意：<**out**> 和可选信号<**out***>不能突变。这意味着当<**out**>的状态改变时，<**out**>和<**out***>都将在瞬时拥有同一个值。

参见 Multiple One Shots，Retriggerable One Shot

26. Producer



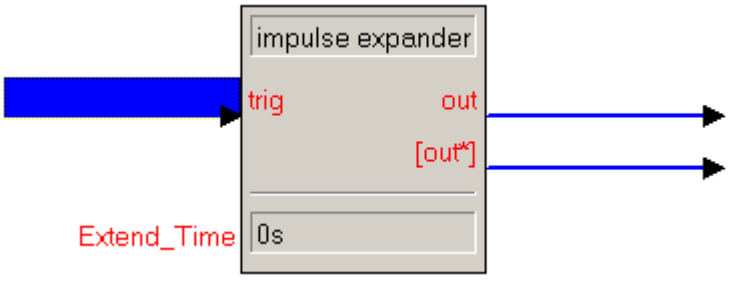
信号：1 个数字输入：<**gate**>

1 个数字输出：<**out**>

2 个双精度参数：<**hi_time**> 和<**lo_time**>（参见 Numeric Formats）

描述：Oscillator 符号驱动它的输出在由<**hi_time**>指定的期间内升高，在由<**lo_time**>指定的期间内降低，在<**gate**>为高时，持续在这两个状态之间振荡。这种振荡开始于<**gate**>的上升沿。当<**gate**>变低时，输出立即变低。

27. Impulse Expander



信号/参数：1 个数字输入：<**trig**>

1 个数字输出：<**out**>

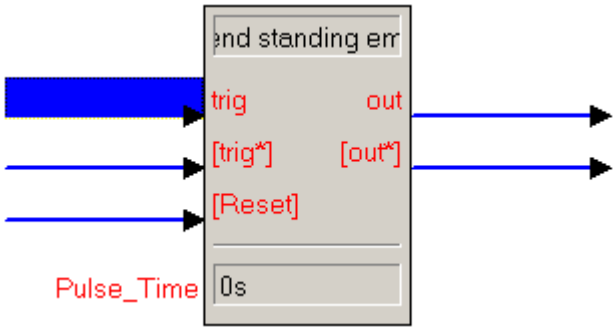
1 个可选数字输出：<**out***>（<**out**>的补码）

1 个双精度参数：<**delay_time**>（参见 Numeric Formats）

描述：一旦<**trig**>输入为高，Pulse Stretcher 符号驱动它的输出信号达到高。在<**trig**>降低后，输出在由<**Time**>指定的额外期间内仍然保持为高（除非是<**trig**>升高，否则<**Time**>参数不会

影响输出）。

28. Extend Standing Emitter



信号/参数：1 个数字输入：<**trig**>

2 个可选数字输入：<**trig***> 和 <**reset**>

1 个数字输出：<**out**>

1 个可选数字输出：<**out***>（<**out**>的补码）

1 个双精度参数：<**pulse_time**>（参见 Numeric Formats）

描述：在<**trig**>的上升沿（或<**trig***>的后边沿），Retriggerable One Shot 符号驱动它的输出信号在由<**pulse_time**>指定的期间内达到高。

与 One Shot 符号不同 Retriggerable One Shot 符号认可任何并发的<**trig**>的上升沿（或<**trig***>的后边沿），甚至当<**out**>为高而引起它重新计算时也是如此。输出将不会变低直到全部的<**pulse_time**>持续时间无打断的全部过去为止。

一旦<**reset**>保持为高，不论<**trig**> 或 <**trig***>的状态如何，可选输入<**reset**>都立即使输出变低。

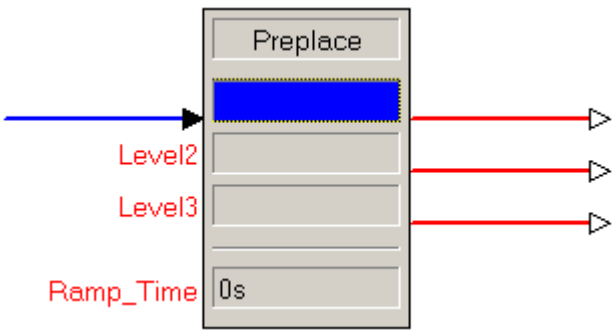
如果 <**trig***>输入被用到，输出在<**reset**>的后边沿将升高并保持为高直到<**pulse_time**>停止。

如果<**reset**>已被定义，那么<**trig***>就必须被定义。如果应用软件中不需要<**trig***>，它可以被赋值为 0。

注意：<**out**>和可选<**out***>信号不会突变。这表示当<**out**>的状态改变时，<**out**>和<**out***>都将在一瞬时拥有同样的值。

参见 One Shot，Multiple One Shot

29. Preplace



信号/参数：1 个数字输入：<**trig**>

任意数量的模拟输出：<**aout1**>到<**aout*N***>

每一个输出都对应有一个目的单元格值：<**level1**>到<**level*N***>（参见 Numeric Formats）

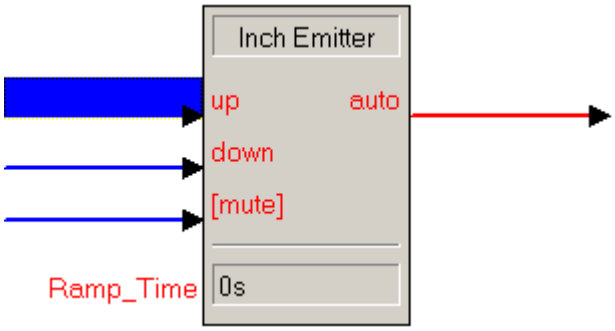
1 个双精度参数：<**ramp_time**>（参见 Numeric Formats）

描述：Analog Preset 信号在被<**ramp_time**>指定期间，通过从它现有的值到新值之间进行平滑的浮动来驱动每一个模拟输出信号到它相应的<**level**>目的单元格值。浮动从<**trig**>的上升沿开始。所有的输出将同时到达它们最终的水平。

注意：在转化期间，输入的第二个上升沿将驱动输出值立即达到它们的最终值。这叫做“CUT”。既然模拟信号能够持有多重合法来源，而且通常是合适的，那么它就可以联合模拟预设的输出与模拟（Analog Ramp）信号的输出并对两者进行比较。当完成后，哪个信号（预设或 Ramp）最后驱动模拟输出值，这个模拟输出值就由此信号决定。在灯光和音量的例子中，由手动控制和预设控制结合起来进行控制。

参见 Analog Variable Preset（模拟预设变量），Analog Ramp

30. Inch Emitter



信号/参数：2 个数字输入：<**up**>和<**down**>

1 个附加数字输入：<**mute**>

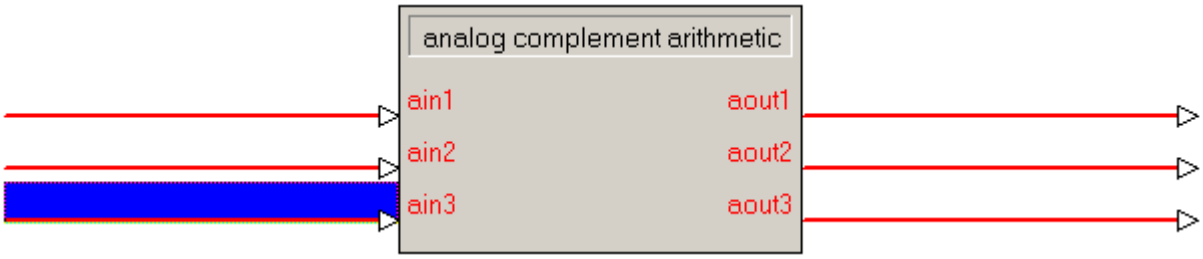
1 个模拟输入：<**aout**>

1 个双精度参数：<**ramp_time**>（参见 Numeric Formats）

描述：无论输入<**up**> 或 <**down**>为高时，Analog Ramp 信号产生出 1 个线性变化模拟输出信号。<**ramp_time**>参数指定了开始使输出在 0%到 100%（或 vice-versa）之间浮动的时间。附加<**mute**>输入使输出在<**mute**>的每一个上升沿变成 0%，并在下降沿返回原值。如果<**mute**>信号要保持输出为 0%，它就应由一个锁定信号（Toggle symbol）来驱动。

注意：既然<**up**>和<**down**>的输入会忽略<**mute**>并开始第二个浮动操作，这表明当<**mute**>为高时要用一个缓冲器来禁止<**up**>和<**down**>。

参见 Analog Non-Volatile Ramp



31. Analog Complement Arithmetic

信号：任意数的模拟信号输入 <**ain1**> 到 <**ain*N***>
对应每个输入有对应的模拟信号输出 <**aout1**> 到 <**aout*N***>

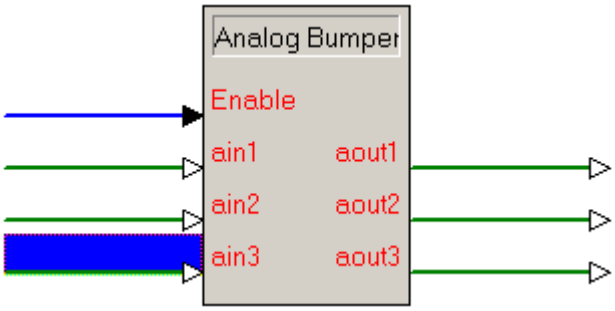
描述：Analog 2's Offset Converter 用 16 位数作为输入。用这种方法，它将模拟输入从补码转换成有正负之分的码。

每一个输入都有一个相对应的输出，并且每一组输入输出之间都相对独立。因此，每一个独立的模块都可以进行转换成和输入相对应多的输出。转换是对称的，即转换后的码再进行转换就变成源码。

转换示例（用 16 进制数表示）：

<**ain1**>	<**aout1**>
0000	8000
8000	0000
FFFF	7FFF
7FFF	FFFF
5432	D432
D432	5432

32. Analog Bumper



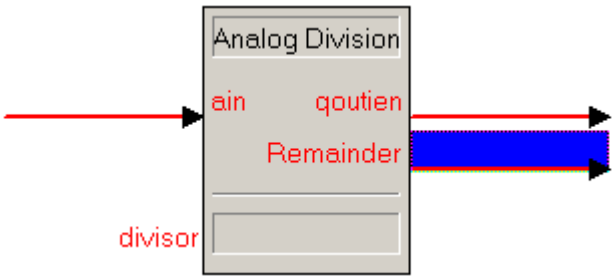
信号：1 个数字输入（Enable）
任意数目的模拟信号输入或连续数据输入: <**ain1**> 到 <**ain*N***>
对应每个输入的输出：<**aout1**> 到 <**aout*N***>

描述： Analog Buffer 元素在上升沿<**enable**>驱动一个给出的输出对应于输入的水平。只要<**enable**>是高电平，在输入中任何一个并发的改变将传递到输出。当<**enable**> 是低电平时，所有的输出将保持不变。每一个输入都有一个相对应的输出，并且每一组输入输出之间都相对独立。

注意： 虽然 Analog Buffer 元素能够传递连续的数据，在大多数情况下，建议使用 Serial Buffer 元素。模拟信号和数字信号的值会一直保持直到它们被赋予新的值，与它们不同，大多数连续信号是瞬时的，这意味着它们的数据只能临时保持。Serial Buffer 元素更适合处理这种特性。

33. Analog Divsion

信号/参数：1 个模拟输入: <**ain**>

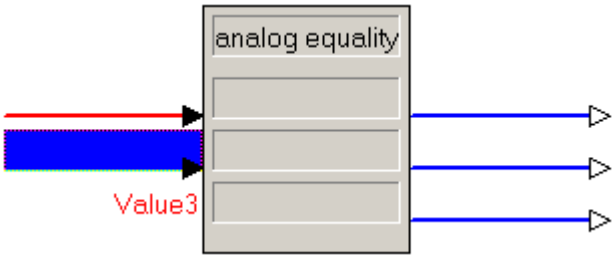


2 个模拟输入: 商<**quotient**> 和 余数<**remainder**>

1 个参数: 除数<**divisor**> (*参见* Numeric Formats)

描述： Analog DivMod 元素对它的输入执行取整和取模操作来产生两个输出。因此，如果 <**ain**> 的值是 5 而 <**divisor**> 是 2,5 的整数部分值(truncated)被 2 除的结果是 2 (<**quotient**> = 2) 而 5 mod 2 is 1 (<**remainder**> = 1)。所有取整操作是无正负之分的，也就是说所有的值都是正的。设置<**divisor**>到 256d 将返回高位数为<**quotient**>，返回低位数为<**remainder**>.

34. Analog Equality



信号/参数：1 个模拟输入: <**ain**>

1 个可选的数字输入: <**enable**>

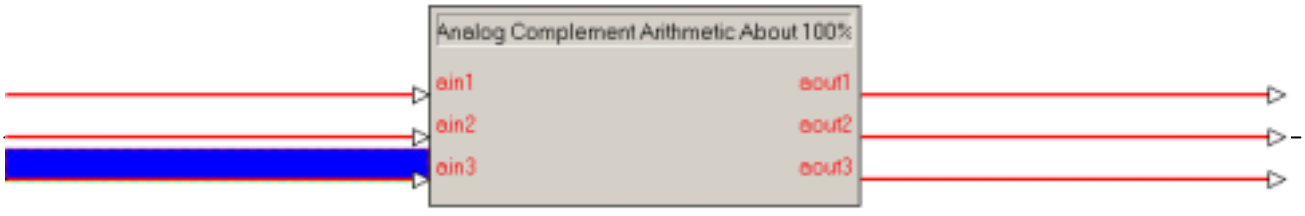
任意数目的数字输出: <**o1**> 到 <**o*N***>

每一个输出都有一个对应的参数: <**value1**> 到 <**value*N***> (参见 Numeric Formats)

描述：若 <value> 参数对应的值与输入的模拟信号值相匹配，那么 Analog Equate 元素将使输出信号值为高。输出将保持为高直到另外一个匹配值被发现。

注意：1 个 Analog Equate 元素的输出在产生之前是不会变化的。这表明两个有不同<**value**> 参数的输出有可能同一瞬时同时达到高。

当可选输入 <**enable**> 为高时，激活此元素;当可选输入 <**enable**> 为低时，所有输出为 0;每次 <**enable**> 为高时，此元素就将输入重新求值并赋予相应的输出为高。



35. Analog Complement Arithmetic About 100%

信号：任意数目的模拟输入: <**ain1**> 到 <**ain*N***>
每一个输入都有一个对应的模拟输出: <**aout1**> 到 <**aout*N***>

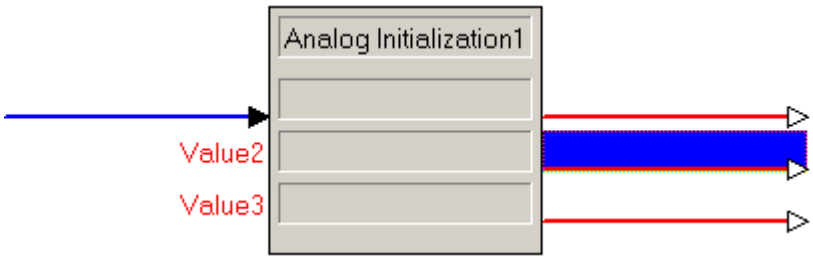
描述：Analog Flip 信号在输出的基础上产生了输入的 2 补码。因此它将从 0%到 100%的范围转化成了 100%到 0%。（1 个输入为 50%的余不变）每一个输入都有一个对应的输出，每一个输入/输出对都是相对独立的。

示例： Analog Flip 信号可用于反转触摸屏的感应信号，或反转用于控制 CPC-CAMI 的感应信号。当 PAN-TILT 单位装配反了，这时，右变成了左，上变成了下。

Sample inversions:

<ain1>	<aout1>
50%	50%
0%	100%
100%	0%
75%	25%
25%	75%
80%	20%
20%	80%

36. Analog Initialization1



信号/参数：

信号输入的形式：1 个数字输入：<**trig1**>
任意数量的模拟输出：<**aout1**>到<**aout*N***>
对应于每个输出，都有一个对应的参数：<**value1**>到<**value*N***>（参见 Mumeric Formats）

信号输出的形式：任意数量的数字输入：<**trig1**>到<**trig*N***>
1 个模拟输出：<**aout1**>
对应于每个输入，都有一个对应的参数：<**value1**>到<**value*N***>（参见 Mumeric Formats）

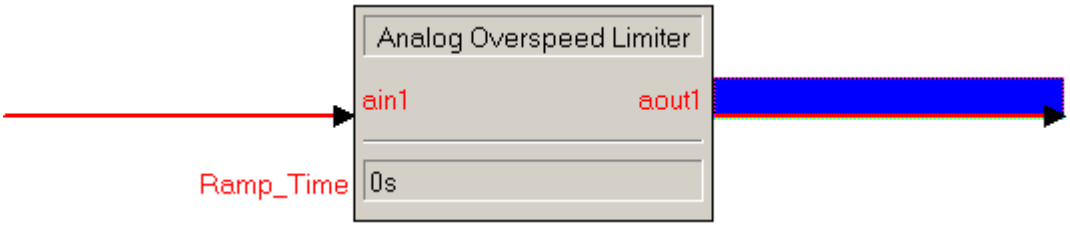
描述：在单一的输入形式中，Analog Initialize 信号通过它对应于在输入信号的每一个上升沿的<**value**>参数来驱动每一个输出到一个特定的值。

在单一输出形式中，信号将对在任何一个输入的上升沿对输出值进行初始化。输出将被设置成对应于最近升高的输入的<**value**>参数。

在启动时，所有的输出都为 0 值，但当输入仅有一个且被赋予信号名 1 时除外。在这种情况下，输出将通过它们对应的<**value**>参数而获得特定的值。

37. Analog Initialization2

38. Analog Overspeed Limiter



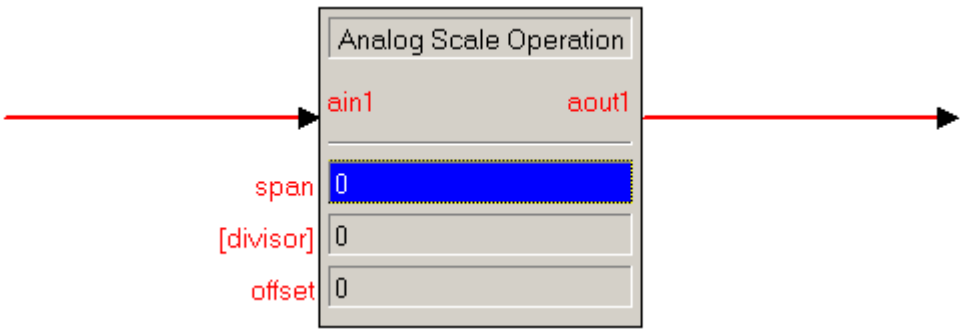
信号/参数：1 个模拟输入：<**ain**>

1 个模拟输出：<**aout**>

1 个双精度参数：<**ramp_time**>（参见 Numeric Formats）

描述：，Analog Rate Limiter 信号通过一个步进输入或任何与瞬变对应的输入而产生 1 个平滑变化的输出，例如从触摸屏上的滑动条。<**ramp_time**>参数指定了 *slew-rate（回转速率）*，或指定了输出从其现值浮动到新值的时间。例如，如果输出现在为 75%，而<**ramp_time**>等于 5 秒，<**ain**>变成 25%时输出将在 5 秒内从 75%转化成 25%。事实上，指定的时间代表了它开始全范围（0-65535）浮动的时间，并由任何局部浮动来进行衡量。例如，如果时间被指定为 10 秒，从 25%到 75%的转化将需要 5 秒的时间，由一半范围起算。

40. Analog Scale Operation



信号/参数：1 个模拟输入：<**ain**>

1 个模拟输出：<**aout**>
2 个参数：<**span**>和<**offset**>（参见 Numeric Formats）
1 个可选参数：<**divisor**>

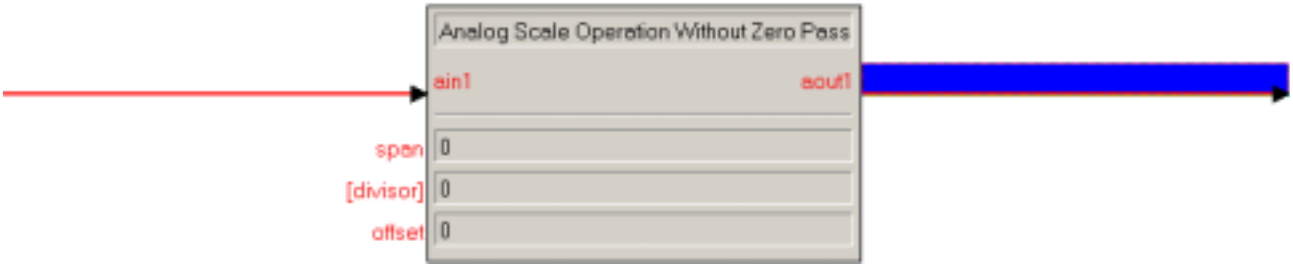
描述：当<**span**>表示比例因子，并且<**offset**>为最小值时，Analog Scaler 信号用以下公式定义其模拟输入信号的值：

<**aout**> = (<**ain**> * <**span**> / <**divisor**>) + <**offset**>

可选参数<**divisor**>用来限定大的范围并且默认值为 100%。（既然<**span**>达不到 100%，当<**divisor**>未定时，输出就可以被定义的更小而不是更大）。

如果输入达到 0%，不论<**offset**>的值是多少，输出将立即被设为 0。Analog Scaler without Zero Pass 信号则失去了这种“零传递”特性。

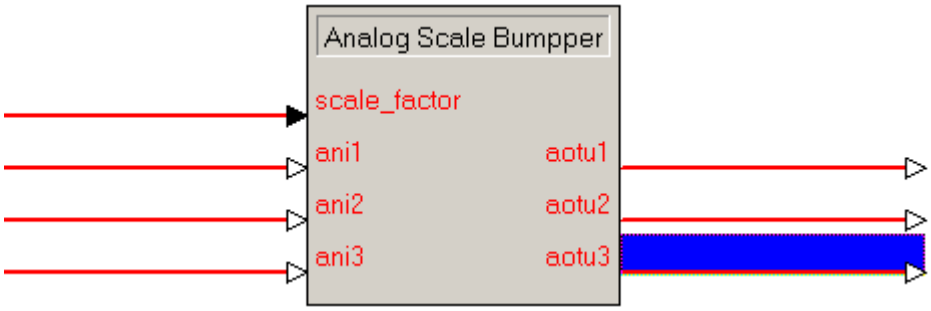
41. Analog Scale Operation Without Zero Pass



信号/参数：1 个模拟输入：<**ain**>
1 个模拟输出：<**aout**>
2 个参数：<**span**> 和 <**offset**>（参见 Numeric Formats）
1 个可选参数：<**divisor**>

描述：Analog Scaler without Zero Pass 信号除了没有“零传递”特性外，和 Analog Scaler 信号的作用一样。也就是说，当输入为 0%时，输出等于<**offset**>而不会减弱。

42. Analog Scale Bumpper



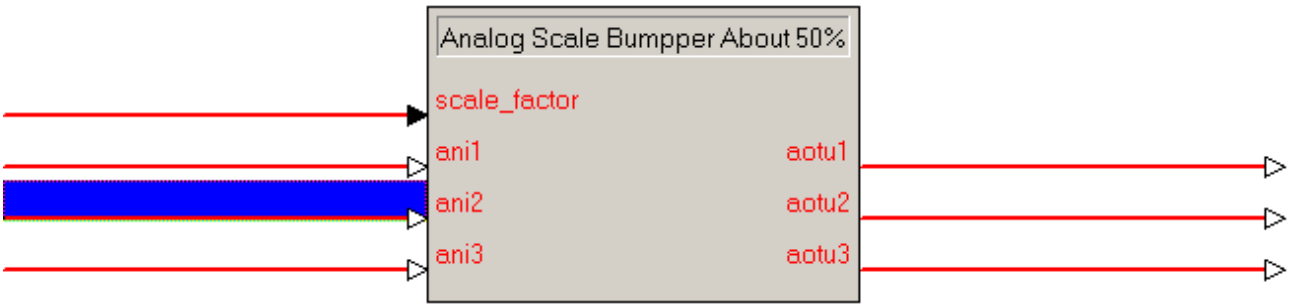
信号/参数：1 个模拟输入：<**scale_factor**>
任意数量的模拟输入：<**ain1**> 到 <**ain*N***>

对应每个输入的输出：<**aout1**> 到 <**aout*N***>

描述：Analog Scaling Buffer 信号按比例限定它的输出为由<**scale_factor**>定义的范围。也就是说，如果<**scale_factor**> = 50%，输出将被设为对应输入的50%。（输入为60%，输出就为30%。）每一个输入信号都有一个对应的输出，并且每一个输入/输出对之间是相对独立的。

注意：因为<**scale_factor**>是一个范围从 0%到 100%的百分数，模拟输出信号的值就不可能超过它的输入。

43. Analog Scale Bumpper About 50%



信号/参数：1 个模拟输入：<**scale_factor**>

任意数量的模拟输入：<**ain1**> 到 <**ain*N***>

对应每一个输入的输出：<**aout1**>到<**aout*N***>

描述：Analog Scaling Buffer about 50%信号通过<**gain**>限定它的输入范围在 50%上下。此过程按照以下公式进行：

minimum value of <**aout1**> = 50% - (<**scale_factor**> / 2)

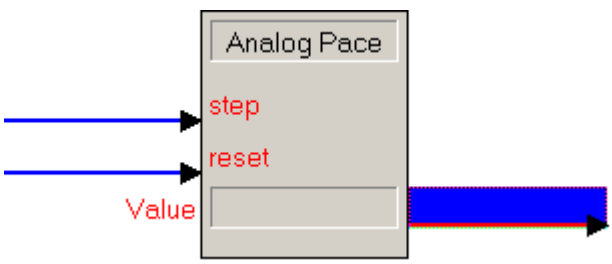
maximum value of <**aout1**> = 50% + (<**scale_factor**> / 2)

因此，如果<**scale_factor**>被设为 50%，一个通常在 0%到 100%之间变化的输入信号将被限定在 25%--75%之间。这个特性也许可以用于限制一个镜头在高度放大时面板的灵敏度。一个滑动条将显示从 0%到 100%的等级，但是镜头放大的输出将只会在 25%到 75%之间。

一个联系<**aout1**> 到 <**ain1**>通式为：

<**aout1**> = (<**scale_factor**>*<**ain1**>) + <**aout1**>$_{min}$

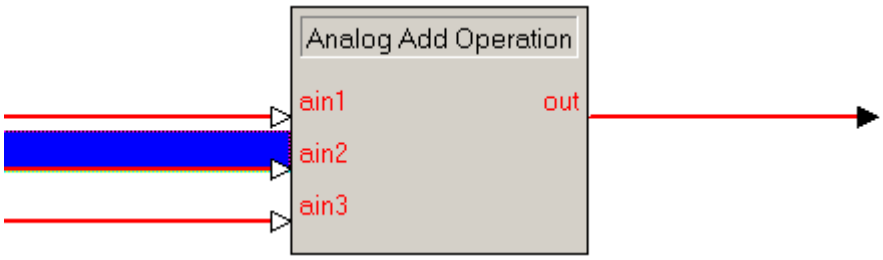
44. Analog Pace



信号/参数：2 个数字输入：<**step**> 和 <**reset**>
1 个模拟输出：<**aout**>
任意数量的参数：<**value**>（参见 umeric Formats）

描述：Analog Step 信号在<**step**>的每一个上升沿设定它的输出为下一个指定的<**value**>。当输出接近最后的<**value**>时，它就复位到第一个<**value**>。当<**reset**>升高，输出复位到第一个<**value**>。

45. Analog Add Operation

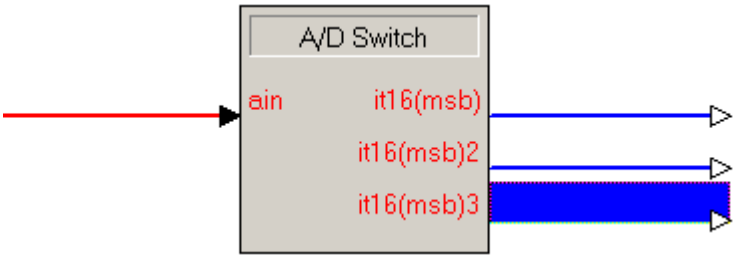


信号/参数：任意数量的模拟输入：<**ain1**>到<**ain*N***>
1 个模拟输出：<**aout**>

描述：Analog Sum 信号以 16 位整数的形式产生它的输出为它的输入之和。而且无论任何输入改变时，都更新输出值。对于一个或更多加数来说，减少可能是由于用二补码符号来表示负值所致。

与 Analog Flip 对照

46. A/D Switch

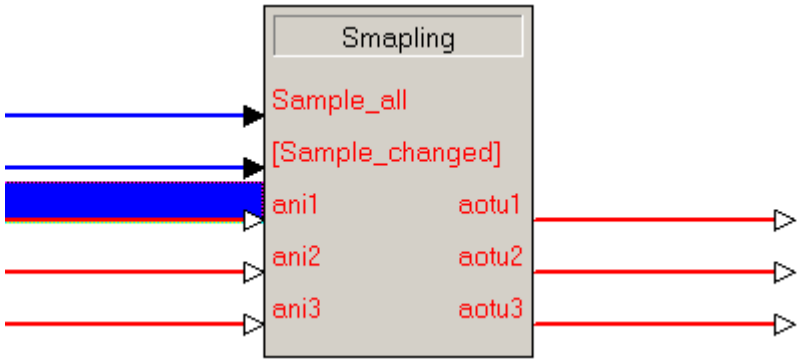


信号/参数：1 个模拟输入：<**ain**>
最多 16 个数字输出：：<**bit16 (msb)**>, <**bit15**> 到<**bit2**>, 以及 <**bit1 (lsb)**>

描述：Analog to Digital 信号是一个转换器，它把模拟信号转换成数字信号。当低对应于 0 而高对应于 1 时，16 位的模拟输入的值由最多 16 个数字输出来表示（开始于最重要的位，或 MSB），0 表示低电平，1 表示高电平。
如果定义少于 16 个输出，则只输出高位。

参见 Digital to Analog

47. Sampling



信号/参数：1 个数字输入：<**sample_all**>

1 个可选数字输入：<**sample_changed**>

任意数量的模拟输入：<**ain1**>到<**ain*N***>

对应每一个输入，都有一个模拟输出：<**aout1**> 到 <**aout*N***>

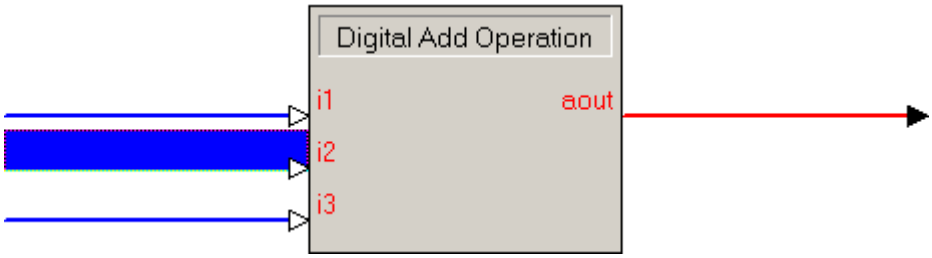
描述：Analog Value Sample 信号仅只在<**sample_all**>的上升沿赋予输出以对应于输入的值，其它情况下无论<**sample_all**>的水平如何，输入怎样变化都不对输出产生影响。这基本上形成了一个样例。每一个输入都有一个对应的输出，并且每一个输入/输出对之间是相对独立的。

可选输入<**sample_changed**>只反映改变了的信号。

示例：一个 Analog Value Sample 信号占用了一个或多个模拟水平的一个“快照”。在更新的模拟信号通过 Intersystem Communications 信号发到远程系统时，它经常与 Oscillator 信号（震荡器信号）一起用于控制速率。具有代表性的是，当较慢的 Oscillator 驱动<**sample_all**>去补偿传送错误时，较快的 Oscillator 就驱动可选输入<**sample_changed**>去检测改变、远程启动系统等。

参见 Oscillator，Intersvstem Communications。

48. Digital Add Operation

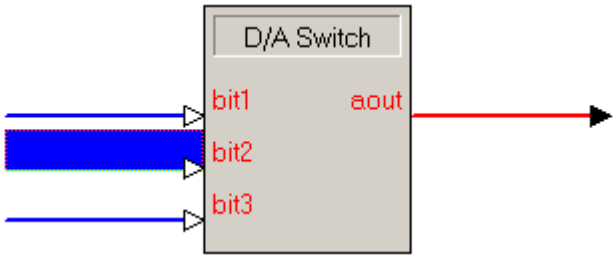


信号/参数：任意数量的数字输入：<**i1**> 到<**i*N***>

1 个模拟输出：<**aout**>

描述：Digital Sum 信号根据它的输出产生一定数量的高电平输出信号，并更新输出值。

49．D/A Switch



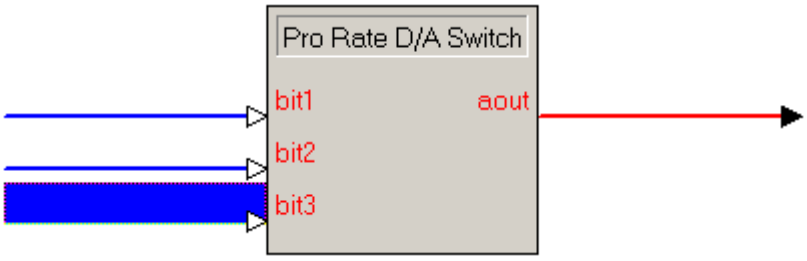
信号/参数：最高 16 个数字输入：<**bit16 (msb)**>, <**bit15**> 到 <**bit2**>， 和<**bit1 (lsb)**>
1 个模拟输出：<**aout**>

描述：Digital to Analog 信号是一个数字-模拟转换器。也就是说，它通过它的数字输入驱动它的输出达到预期的值。每一个输入对应一位（开始于最高位，或 MSB），低电平对应 0，高电平对应 1。任何不明确的输入被假定为 0。

例如，如果用到了 4 个输入，<**bit16**> 到 <**bit13**>，输出范围就为从 0%到 93.751%。

对照 Digital to Scaled Analog，Analog to Digital。

50．Pro Rate D/A Switch



信号/参数：最高 16 个数字输入：<**bit16 (msb)**>, <**bit15**> 到 <**bit2**>，和 <**bit1 (lsb)**>
1 个模拟输出：<**aout**>

描述：Digital to Scaled Analog 信号将它的数字输入转换成一个有范围的模拟输出信号，如下：每一个输入表示一位（从最高位开始，或 MSB），低电平对应 0，高电平对应 1。因而得到的输出值范围就由输入定义的二进制值限定，这样当所有定义的输入为高时，输出值为 100%，而当所有定义的输入为低时，输出值为 0%。

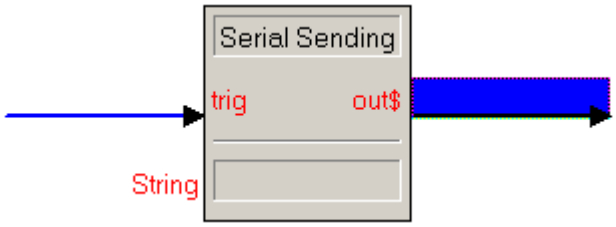
例如，如果数字 7 由 4 个输入表示，<**bit16**>=0（低）而<**bit15**>到<**bit13**>=1（高）。因为一个 4 位数可以有从 0 到 15 之间的值，而 7 是中间值，模拟输出将为 50%。

注意：如果所有 16 个输入都用到的话，Digital to Scaled Analog 信号将立即把数字信号转换为模拟信号。

参见 Digital to Analog ，Analog to Digital。

51. Serial Sending

信号/参数：1 个数字输入：<**trig**>



1 个连续输出：<**out$**>

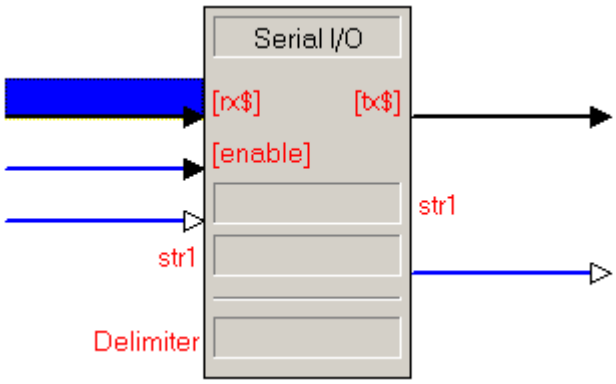
1 个参数：<**string**>

描述：Serial Send 元素在<**trig**>的每一个上升沿传送由<**string**>参数指定的文本。

在应用程序中，当多重字符串被传送时，推荐使用 Serial I/O 元素的输出形式，作为一种有效的可选择项来使用多重 Serial Send 元素。

注意：如果多重 Serial Send 元素的输出积压在一起而它们相应的<**trig**>输入同时升高，元素就维持串联（不需要 Serial Concatenation 元素）。

52. Serial I/O



信号：

连续输入信号：

1 个连续输入：<**rx$**>

1 个可选数字输入：<**enable**>

任意数量的数字输出：<**o1**> 到 <**o*M***>

每一个输出都有一个对应的参数：<**str1**> 到 <**str*M***>

连续输出信号：

任意数量的数字输入：<**i1**> 到 <**i*N***>

1 个可选数字输入：<**enable**>

每一个输入都有一个对应的参数：<**str1**> 到 <**str*N***>

1 个连续输出：<**tx$**>

参数：

连续输入/输出参数：1 个可选参数：<**delimiter**>

描述：

连续输入信号：

在连续输入信号形式中，如果一个给出的输出信号的对应<**str**>参数与<**rx$**>输入相匹配，Serial I/O 元素就驱动这个输出信号升高。

在启动时，元素放置一个内部缓冲器，它的大小与最长的<**str**>参数相同。缓冲器是“后进先出”，因而用新的数据替换出旧的数据。当新的数据进入到 Serial I/O 元素中，它就被加入到缓冲器中并且所有的输出信号被清空（有效地使输出突变（break before make））。然后元素重新核对字符串的所有<**str**>参数，如果发现一个匹配的，就驱动对应的输出升高。

有时，一行中同样的字符串数据可能进入元素两次，在这种情况下，对应的输出将暂时降低然后再升高。这样就提供了一个可以被逻辑认可的上升沿，此逻辑是由元素驱动的。另外，如果输入包含字符串“HelloWorld”，而两个<**str**>参数被定义为 "Hello" 和 "World" ，对应于 "Hello" 的输出将先被驱动为高电平，然后对应于 "World" 的输出再被驱动为高电平。

可选输入<**enable**>为高电平时激活元素。当<**enable**>为低电平时，输入被忽略，输出为低电平。

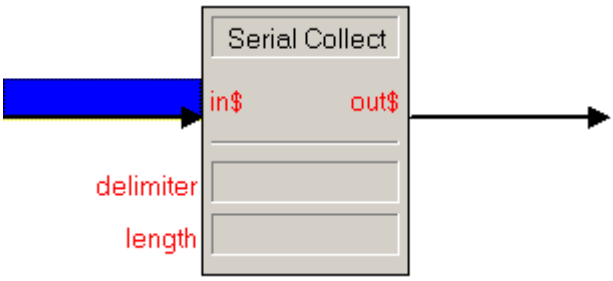
连续输出信号：

在连续输出信号形式中，元素在信号的上升沿传送那些对应输入信号升高的字符串。

可选输入<**enable**>为高电平时激活元素。当<**enable**>为低电平时，所有输入被忽略。如果<**enable**>变为低电平时有一个输入为高电平，那么这个输入在<**enable**>重新升高时将被忽略，直到它的下一个上升沿为止。

在输入和输出形式中，可选参数<**delimiter**>添加指定的特征到所有<**str**>参数上。例如，如果协议要求：从一个设备处来的和到这个设备的所有字符串要有一个回车/ linefeed，<**delimiter**>应为 "\n" 。

53. Serial Collect



信号/参数：1 个连续输入：<**in$**>

1 个连续输出：<**out$**>

2 个参数：<**delimiter**> 和 <**length**>

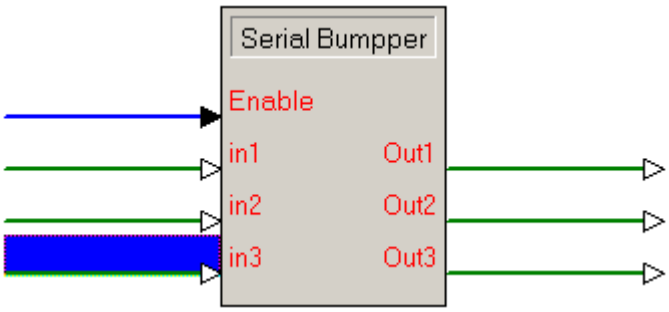
描述：Serial Collect 元素在它找到由<**delimiter**>参数指定的字符后才求出它的连续输入的值，然后传送字符串已界定的部分（包括<**delimiter**>）。剩下的字符串片断储存在内部缓冲器中，大小与<**length**>参数相同。如果输入能够包含多重界定的字符串的话，每一个界定了的部分将按顺序传送直到元素再也找不到有效的<**delimiter**>为止。

缓冲器<**length**>至少应该与最长的字符串一样长，但不应超过字符串长度的 254 个字节。如果元素收到超过<**length**>特征的字符串，而又没有发现<**delimiter**>的话，就会出现缓冲器溢出的现象。错误信息 "Memory Overflow in Gather" 将被传送到控制台，缓冲器将被清零。

Serial Collect 元素在获取数据碎片方面很有用，例如，接收从 COM 口传送来的数据，再将数据整合后重新发出。传送数据到某些元素时是很危险的，如传送到 Serial Substring 元素上。

参见 Serial Substring

54. Serial Bumpper



信号/参数：1 个数字输入：<**enable**>

任意数量的连续输出：<**in1$**> 到 <**in*N*$**>

每一个连续输入都有一个对应的连续输出：<**out1$**>到<**out*N*$**>

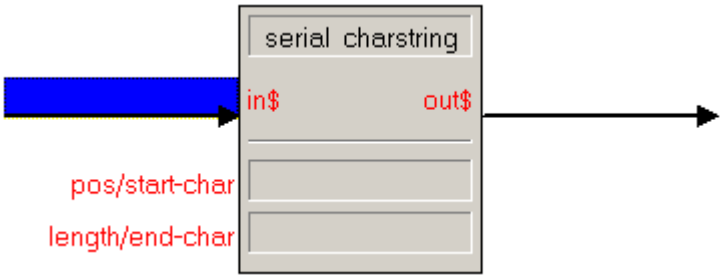
描述：只要<**enable**>为高电平，Serial Buffer 元素就传送它的连续输入到它们相应的输出上。每一个输入都有一对相应的输出，每一个输入/输出对之间是相互独立的。

同其他的 Buffer 元素一样，Serial Buffer 元素的输出可以被“压”在一起。不过，如果多重输入同时传播到“压”在一起的输出时，只有 1 个字符串可以被传送——其他的将丢失。因此，如果应用程序需要发出所有的字符串时，通过 Serial Concatenation 元素来路由输出信号就是非常必要的。

注意：与 Analog Buffer 元素不同，Serial Buffer 元素在<**enable**>的上升沿或下降沿不传播信号，而是允许信号在<**enable**>为高电平时不停的流动。这是十分必要的，因为大多数连续信号是短暂的。

比较 Analog Buffer, Serial Concatenation 元素

55. Serial Charstring



信号/参数：1 个连续输入：<**in$**>

1 个连续输出：<**out$**>

2 个参数：<**pos**> 和 <**length**>

描述：Serial Substring 元素求出它的连续输入信号的值，然后取出并传送字符串中由<**pos**> 和 <**length**>参数定义的的部分。这些参数可以用以下两种形式表现：

第一种形式，<**pos**>指定第一个被传送字符的位置被传送，而<**length**>给出要传送的所有字符的数量。第一个字符的位置由左到右计算，即如果<**pos**>=1，被取出的子字符串由左边的第一个字符开始。如果<**pos**>=2，被取出的子字符串由左边的第二个字符开始，依此类推。唯一的例外是当<**pos**>=0 时，子字符串由最右边的字符开始。

第二种形式，<**pos**> 和 <**length**>用十六进制格式 *xxyy*h 指定特殊情况来界定字符。这里的 *xx* 表

示出现的字符而 yy 表示字符的 ASCII 码。例如，要寻找第三个出现的字母“P”（ASCII 码为十六进制数 50），参数<**pos**>应为 0350h。要传送字符从这一点到第二个出现的相连字符（ASCII 码为十六进制数 2D）上，<**length**>应为 022Dh。

这两种形式不能被混合。也就是说，当<**length**>指定一个字符的出现时（第二种形式），<**pos**>不能被指定为一个绝对的字符位置（第一种形式），相反亦是如此。

注意：由第二种形式产生的子字符串不包括界定的字符。同样，<**length**>参数是由<**pos**>参数计算得出，而不是输入字符串的开始。因此，就如例子中描述的，子字符串将以第二个由<**pos**>指定的字符的连字号结束。

应用举例：

形式一：

一个安全的系统以如下格式传送数据：

ZONE：MSTR BED　STATUS：FAULT<CR>

要取出并传送区名 "MSTR BED" ，<**pos**> = 6 而 <**length**> = 12（宽度固定）。要取出并传送区的状态，<**pos**> = 18 而 <**length**> = 5。（在这种情况下，区名后将跟有空格。）

形式二：

一个 COM 口以如下格式传送字符串：

TEXT 3-Back in the High Life||Winwood, Steve||Rock\x0D\x0A

应用程序需要把歌曲的名字，艺术家的名字以及流派分成三个独立的字符串进行传送。这就要求用一个 Serial Gather 元素控制三个 Serial Substring 元素。

1、从 COM 口把字符串和作为<**delimiter**>参数的 \x0A (line feed) 路由到一个 Serial Gather 元素。

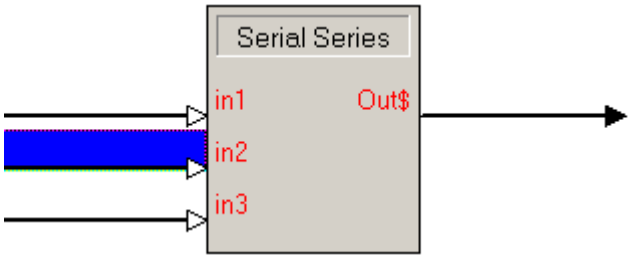
2、传递字符串 "gathered" 到 Serial Substring symbol 元素，并从第一个出现的连字号字符（ASCII 码为十六进制数 2D）开始至第一个出现的（|）字符（ASCII 码为十六进制数 7C）为止提取数据。因此<**pos**> =012Dh 而<**length**> = 017Ch。这样就传送了字符串 "Back in the High Life" 。

3、传递字符串 "gathered" 到第二个 Serial Substring 元素，并从第二个出现的（|）字符至第一个出现的（|）字符间提取数据。因此<**pos**> = 027Ch 而 <**length**> = 017Ch。这样就传送了字符串"Winwood, Steve"。

4、传递字符串 "gathered" 到第三个 Serial Substring 元素，并从第四个出现的（|）字符至第一个回车（ASCII 码为十六进制数 0D）间提取数据（ASCII 码为十六进制数 0D）。因此<**pos**> = 047Ch 而 <**length**> = 010Dh。这样就传送了字符串 "Rock" 。

比较 Serial Gather

56. Serial Series



信号/参数：任意数量的连续输入：<**in1$**> 到 <**in*N*$**>

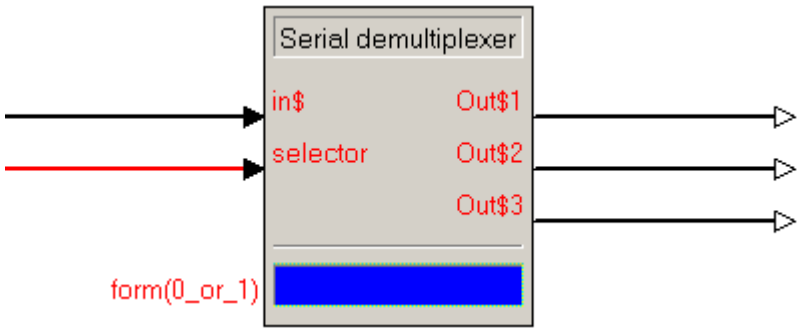
1 个连续输出：<**out$**>

描述：Serial Concatenation 元素可以同时接收多重连续输入信号。然后在连续逻辑波上每次传送一个字符串，从<**in1$**>开始，依次进行。

用这种方式，Serial Concatenation 元素不需逐字的连接字符串，而是交错进行传送来确保不丢失数据。比较典型的是，Serial Buffer 元素的压入的输出由 Serial Concatenation 元素从彼此覆盖状态路由到预设的字符串上。

参见 Serial Buffer

57. Serial demultiplexer



信号/参数：1 个连续输入：<**in$**>

1 个可选模拟输入：<**selector**>

任意数量的连续输出：<**out1$**> 到 <**out*N*$**>

1 个参数：<**form**>

描述：*multiplexor* 使复合信号能够沿着单一的路径被传送，而不丢失任何单独信号的整数部分。原始信号在接收端由 *demultiplexor* 重组。

Serial Demultiplexor 元素以如下两种方式传播它的输入到一个特定的输出上：

当<**form**>参数为 0 时，字符串被路由到与<**selector**>输入值匹配的输出上。例如，若<**selector**>的值为 7，字符串就被路由到<**out7$**>输出。若<**selector**>的值为 0，将不传播任何数据。

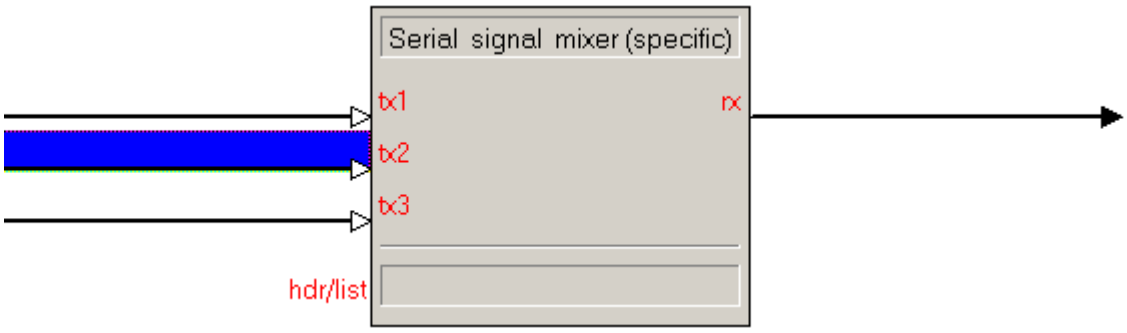
当<**form**>=1 时，输入的字符串必须加上前缀：

\x1B\x*MM*

这里 *MM* 表示字符串将被路由到的输出信号。*MM* 从 21 开始，即指定<**out1$**>输出（<**form**> = \x1B\x21）。这样，要路由字符串到<**out7$**>输出，字符串必须加上前缀\x1B\x27。如果 *MM*=20，则不传播任何数据。要把\x1B 当作输出数据的一部分来传送，就要用\x1B\x1B，这里只有一个\x1B 可以通过输出。

注意：<**form**>参数只有两个有效值：0 或 1。

58. Serial Signal Mixer (Specific)



信号/参数：任意数量的连续输入：<**tx1$**> 到 <**tx*N*$**>
1 个连续输出：<**rx$**>
1 个参数：<**hdr/list**>

描述：*multiplexor* 使复合信号能够沿着单一的路径被传送，而不丢失任何单独信号的整数部分。原始信号在接收端由 *demultiplexor* 重组。

在 SIMPL 里，Serial Multiplexor (Special)元素通过一个单一的输出路径来传送复合字符串，而 Serial Demultiplexor (Special)元素重组原始字符串。

注意：既然这两个元素是相互对称的，那么这两个元素的参数也必然是严密吻合的。

<**hdr/list**>参数可以用两种方法来区别字符串：一种是为每一个字符串指定一个头；另一种是确定每一个字符串的位置（从 1 到 N）。Multiplexor 把这两条信息加到每一个字符串上，而 Demultiplexor 利用这个信息将字符串分开并路由它们到相应的输出上。

"hdr"部分指定了一个 2 字节的头，这个头可以是以下四个字符中的任意一个：4*, --, AB 或 \x45\x33。"list"部分由每一个字符串中的一个字符组成。也就是说，给出五个字符串，（<**tx1$**> 到 <**tx5$**>），"list"部分应该是由任意五个字符组成，如：ABCDE, qwert, 或 \x43\x23\x9A\x21\x33 等等。因此，拥有五个字符串的<**hdr/list**>参数看起来应该如下所示：

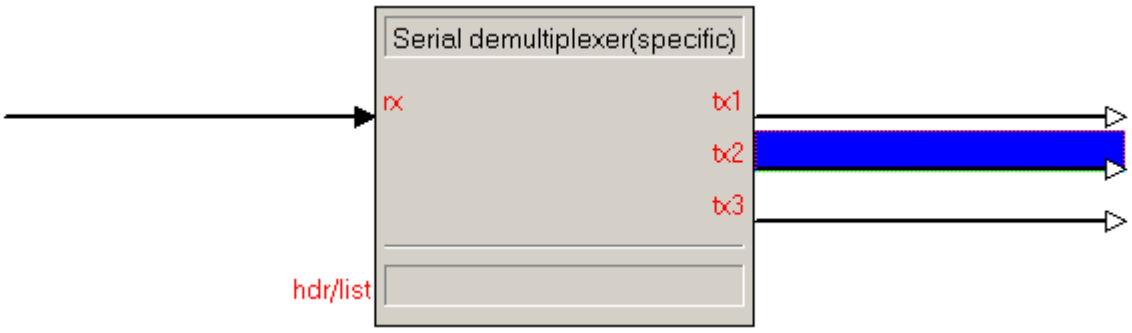
4*ABCDE
--qwert
AB\x43\x23\x9A\x21\x33

以上面的第一个例子来说，如果字符串 "TEST" 得到值为<**tx2$**>，Multiplexor 将在字符串前加头"*4"和字符"B"作为前缀，字符是对应于<**tx2$**>的。Multiplexor 又固定另一个信息，如字符串的长度和校验和，来进一步保证正确的解码。然后 Demultiplexor 校验控制信息并路由字符串到<**tx2$**>输出。

注意：Multiplexor 的输出字符串的实际形式是：{2 字节的头 (<**hdr/list**>的 "hdr" 部分)} + {单字节字符串代码(<**hdr/list**>的 "list" 部分) + { <**tx*N*$**>的字符串的长度，用字节表示} + {<**tx*N*$**>} + {信息包中已存在的所有字节的单字节校验和}

参见 Serial Demultiplexor (Special)

59. Serial Demultiplexer(Specific)



信号/参数：1 个连续输入：<**rx$**>

任意数量的连续输出：<**tx1$**> 到 <**tx*N*$**>

1 个参数：<**hdr/list**>

描述：*multiplexor* 使复合信号能够沿着单一的路径被传送，而不丢失任何单独信号的整数部分。原始信号在接收端由 *demultiplexor* 重组。

在 SIMPL 里，Serial Multiplexor (Special)元素通过一个单一的输出路径来传送复合字符串，而 Serial Demultiplexor (Special)元素重组原始字符串。

注意：既然这两个元素是相互对称的，那么这两个元素的参数也必然是严密吻合的。

<**hdr/list**>参数可以用两种方法来区别字符串：一种是为每一个字符串指定一个头；另一种是确定每一个字符串的位置（从 1 到 N）。Multiplexor 把这两条信息加到每一个字符串上，而 Demultiplexor 利用这个信息将字符串分开并路由它们到相应的输出上。

"hdr"部分指定了一个 2 字节的头，这个头可以是以下四个字符中的任意一个：4*, --, AB 或 \x45\x33。"list"部分由每一个字符串中的一个字符组成。也就是说，给出五个字符串，（<**tx1$**> 到 <**tx5$**>），"list"部分应该是由任意五个字符组成，如：ABCDE, qwert, 或 \x43\x23\x9A\x21\x33 等等。因此，拥有五个字符串的<**hdr/list**>参数看起来应该如下所示：

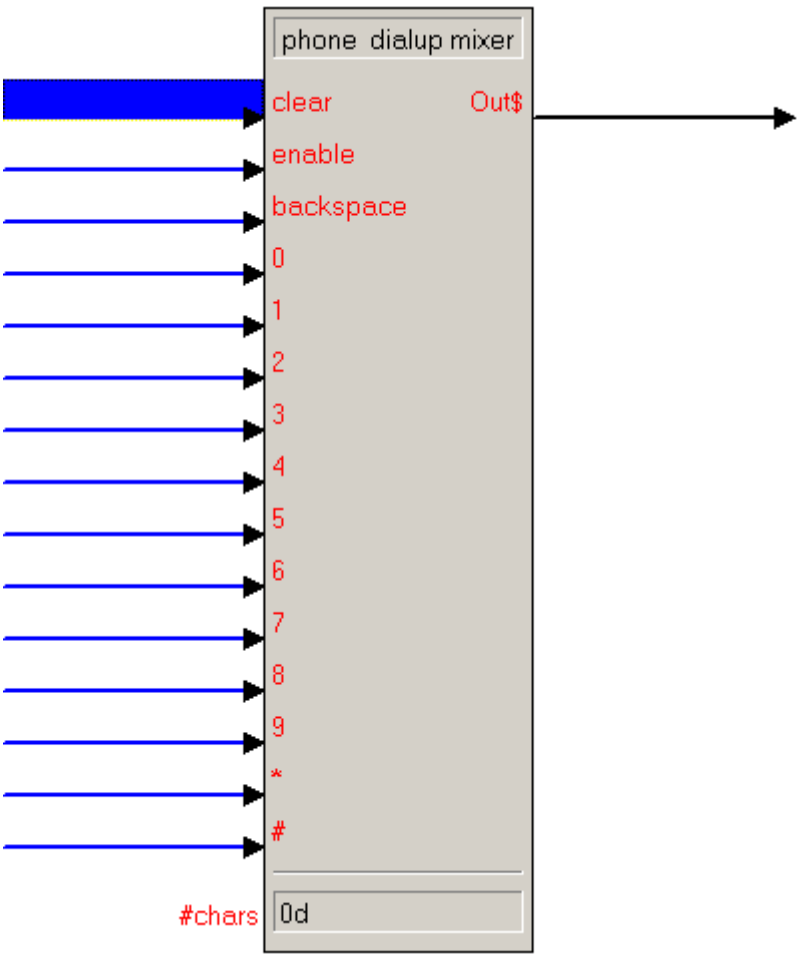
4*ABCDE
--qwert
AB\x43\x23\x9A\x21\x33

以上面的第一个例子来说，如果字符串 "TEST" 得到值为<**tx2$**>，Multiplexor 将在字符串前加头"*4"和字符"B"作为前缀，字符是对应于<**tx2$**>的。Multiplexor 又固定另一个信息，如字符串的长度和校验和，来进一步保证正确的解码。然后 Demultiplexor 校验控制信息并路由字符串到<**tx2$**>输出。

注意：Multiplexor 的输出字符串的实际形式是：{2 字节的头（<**hdr/list**>的 "hdr" 部分)} + {单字节字符串代码(<**hdr/list**>的 "list" 部分) + { <**tx*N*$**>的字符串的长度，用字节表示} + {<**tx*N*$**>} + {信息包中已存在的所有字节的单字节校验和}

参见 Serial Multiplexor (Special)

60. Phone Dialup Mixer

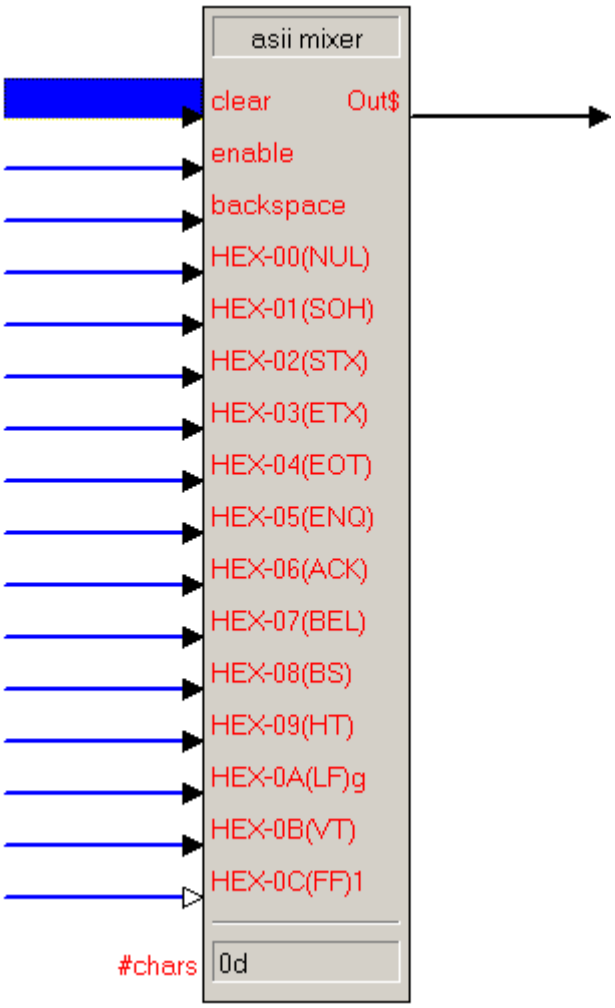


信号/参数：3 个数字输入：<**clear**>, <**enable**> 和 <**backspace**>
12 个数字输入：<**0**> 到 <**9**>, <***>> 和 <**#**>
1 个连续输出：<**out$**>
1 个参数：<**#chars**>

描述：当<**enable**>为高电平时，Telephone Dialing Keypad 元素从它的数字输入中产生一个静态的字符串。就这点来说，它和 ASCII Keypad symbol 元素的操作是相同的，除了 ASCII Keypad symbol 元素的输入可以用任何一个标准 ASCII 码来表示，而 Telephone Dialing Keypad 元素的输入用数字表示。
注意：输出信号的静态特性允许它可以通过 Serial RAM 元素被储存在内存中。
在启动时，Telephone Dialing Keypad 元素放置一个内部缓冲器，此缓冲器的大小和<**#chars**>参数相同。（<**#chars**>参数等价于 ASCII Keypad 元素的<**#chars**>参数。）如果进入更多的数字，超过了<**#chars**>参数指定的范围，额外的数字将被忽略。
因为每一个数字输入升高，对应的特征积聚在缓冲器中，而进入的字符串被重新传送。例如，如果输入字符串 "6532" ，元素将先传送 "6"，然后传送 "65"，接着是 "653"，最后是 "6532"。
<**clear**>输入移走当前积聚的字符串并传送一个空字符串。<**backspace**>输入的操作就像一个退格键，使最终用户不需要清除全部序列也可以进行编辑。
当<**enable**>为低电平时，所有的输入信号被忽略（包括<**clear**> 和 <**backspace**>）。如果在<**enable**>为低电平时仍有积聚的字符串，那么这个字符串将保留在缓冲器中，在<**enable**>为高电平时被重新传送。

参见 ASCII Keypad, Serial RAM, Serial Memory Search, Telephone Dialing Keypad w/o Backspace

61. Asii Mixer



信号/参数：1 个连续输入：<**in$**>

1 个数字输入：<**dial**>

1 个数字输出：<**busy**>

最多 127 个数字输出：<**HEX-00**>到<**HEX-7E**>（表现为标准 ASCII 字符）

1 个双精度参数：<**time_betwn_chars**> (参见 Numeric Formats)

描述：ASCII Serial Decoder 元素的操作和 Serial Memory Dialer 元素相同，除了 Serial Memory Dialer 元素的输出表现为数字而 ASCII Serial Decoder 元素的输出可以表现为任何标准 ASCII 字符。

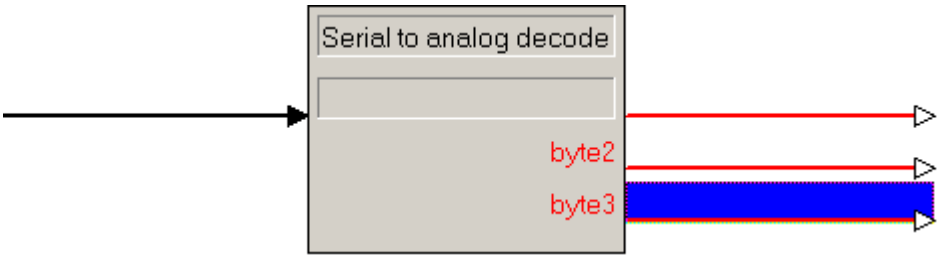
ASCII Serial Decoder 元素通常与 ASCII Keypad 元素和 Serial RAM 元素一起工作，如下：

ASCII Keypad 元素产生 1 个静态字符串。它的输出与 Serial RAM 元素（此元素将字符串存储在内存里）和 ASCII Serial Decoder 元素的<**in$**>输入相关。

通常，Serial RAM 元素的<**dial**>输出驱动 ASCII Serial Decoder 元素的<**dial**>输入。当<**dial**>升高时，<**in$**>从内存中唤醒，相应的数字输出顺序升高。当出现"dialed"字符串时，<**busy**>输出升高。<**time_betwn_chars**>参数控制拨号操作的速度并指定从一个输出降低到下一个输出升高之间的时间。在同一时间内只有一个输出为高电平。

参见 Serial RAM, Serial Memory Dialer, ASCII Keypad, Telephone Dialing Keypad

62. Serial To Analog Decode



信号/参数：1 个连续输入：<**rx$**>

任意数量的模拟输出：<**byte1**> 到 <**byte*M***>

任意数量的参数：<**p1**> 到 <**p*N***>

注意：<**p**>参数列表可以通过 **Alt+** 命令扩展。

描述：当 Serial to Analog 元素找到一个与由<**p**>参数定义的字符串（或字符串片断）完相匹配者时，它就求出它的连续输入信号的值。然后它从字符串中取出选中的字符并以单独模拟信号的形式传送每一个字符，这样每一个输出信号的值和取出字符的 ASCII 值相同。

等待输入的字符串必须有固定的长度和已知的格式，每一个<**p**>参数必须对应于输入字符串的一个字节。也就是说，如果字符串是由 8 个字节组成，就必须有 8 个<**p**>参数（<**p1**> 到 <**p8**>）。这些参数用以下三种方式进行定义：

1. <**p*N***> = 0000h　表示字符是不相关的，可以被忽略。
2. <**p*N***> = 01*xx*h　表示字符必须和 *xx* 匹配。例如，如果第三个字符为字母“A”（ASCII 码为十六进制数 41），那么<**p3**>就为 0141h。
3. <**p*N***> = 0200h　确定一个取出的字符并传送。

应用举例：

假设是一个安全的系统，以如下格式传送数据：

\x2A {单字节区数字} {2 字节区状态} {5 字节区名字} \x35 {单字节区校验和} \x26

在这里，十六进制数为常数而括号内的值为可变值。假设当区数等于 \x01 时，表示区的状态的 2 字节必须被取出并路由到 2 个模拟输出信号上。并假设区的名字和校验和不相关。

既然这里有 12 字节的输入数据，就必须有 12 个<**p**>参数，以如下方式定义：

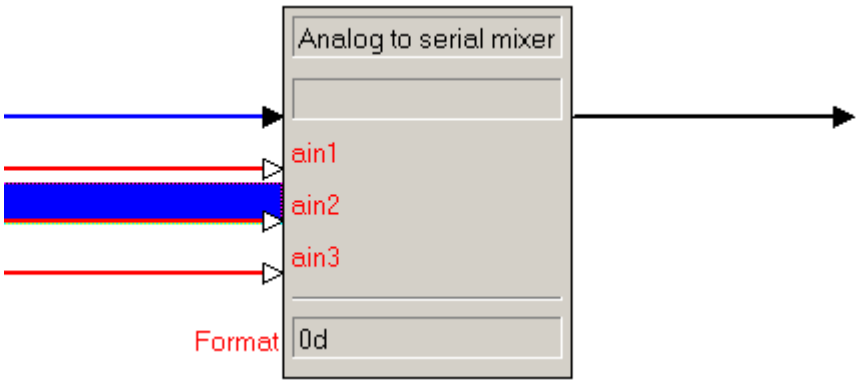
<**p1**> (必须和 \x2A 匹配): 012Ah
<**p2**> (必须和 \x01 匹配): 0101h
<**p3**> (被取出并路由到<**byte1**>): 0200h
<**p4**> (被取出并路由到 <**byte 2**>): 0200h
<**p5**> (忽略): 0000h
<**p6**> (忽略): 0000h
<**p7**> (忽略): 0000h
<**p8**> (忽略): 0000h
<**p9**> (忽略): 0000h
<**p10**> 必须和 \x35 匹配): 0135h
<**p11**> (忽略校验和): 0000h
<**p12**> (必须和 \x26 匹配): 0126h

这里的<**p3**> 和 <**p4**>，表示区状态的 2 字节，将被取出并路由到<**byte 1**>和<**byte 2**>模拟输出上去。

注意：Serial to Analog 元素将数据放置在输出信号的低字节中。

参见 Serial Gather, Serial I/O, Serial Substring

63. Analog To Serial Mixer



信号/参数：1 个数字输入：<**trig**>

任意数量的模拟输入：<**ain1**>到<**ain*M***>

1 个连续输入：<**out$**>

任意数值的参数：<**String1**>到<**String*N***>，N 值从 1 到 M+1

1 个参数：<**Format**>

注意：参数<**String**>的列表可以用 Alt+命令 的方式被扩展。

描述：Analog to Serial 元素从它的模拟输入的低字节（或高字节）或 1 个联合模拟输入和字符串常量中构造了 1 个字符串，然后在<**trig**>的每一个上升沿传送这个字符串。用这种方式，元素能够更新和传送在运行时间内改变了的数据。

<**Format**>参数决定是否要从模拟输入的低字节或高字节里接受数据。它也将那些特殊的格式列入清单，即使这些格式在被传送之前是应用于字符串的。关于<**Format**>将在后面的部分作详细的描述。

输出字符串是这样构造的（假设设置为低字节）：

如果没有用到字符串常量，每一个模拟输入的低字节就连接起来（从<**ain1**>开始顺序进行）然后相应的字符串被传送。比如，如果从<**ain1**>到<**ain5**>的低字节为以下几个十六进制值：

48h, 65h, 6Ch, 6Ch and 6Fh

字符串“Hello”就被传送。在大多数情况下，Analog Initialize 元素将用于给模拟输入赋值。（此值可以用可选择的字符形式来传送：'H', 'e', 'l', 'l', 'o'。）就像前面规定的，字符串被更新并在<**trig**>的每一个上升沿被传送。

有时应用程序要求字符串常量内含在输出数据中。这里，元素用顺序连接<**String1**> + <**ain1**> + <**String2**> + <**ain2**> + <**String3**> + <**ain3**>+---- 的方式构造了输出字符串。任何没有用到的参数都要被定义并被赋值为空字符串（“”）。例如，假设输出字符串将由以下信号和参数构成：

Two analog signals (<**ain1**> and <**ain2**>)

The string "TEST" (<**String3**>)

Two analog signals (<**ain3**> and <**ain4**>)

The string "\x0D\x0E" (<**String5**>)

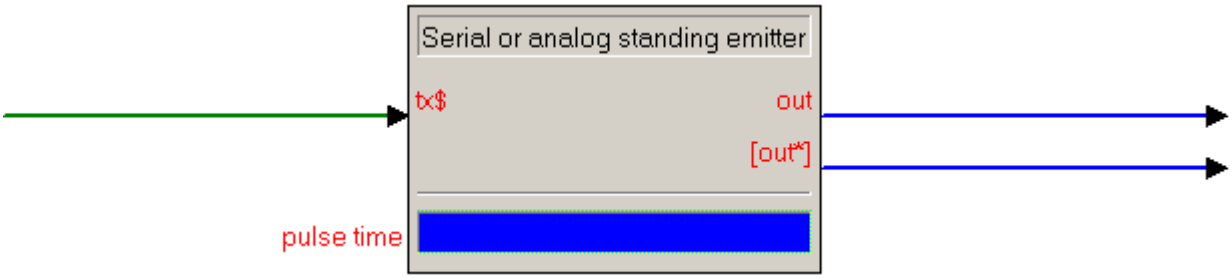
在上面这个例子中，<**String1**>, <**String2**>和<**String4**>必须被定义并赋值为空字符串（“”）。（如果不需要插入字符串，将不需要填写所有<**String**>参数。）

格式： 如上所述，<**Format**>参数适用于以下两种不同的情况：①指定每一个模拟输入要减小的字节；②指定特殊的格式，即使此格式是应用于传送前的字符串的。作为一个 16 位数，<**Format**>可以在逻辑上自己分成一个低字节和高字节，每一个字节用于一个不同的用途，见下面的叙述。

要指定用模拟输入的哪一个低字节或高字节，调整<**Format**>的第 9 位的值。如果此位为 0，就用高字节（不推荐用于大多数应用程序中）。要用低字节（推荐），可通过在<**Format**>的总值上加十进制数 256 的方法来设置第 9 位为 1。

参数的低字节可以用于添加一个求校验和（checksum）的特殊类型到输出字符串上，如下表。

64. Serial Or Analog Standing emitter



信号/参数：1 个连续或模拟输入：<**rx$**>

1 个数字输出：<**out**>

1 个可选数字输出：<**out***>（<**out**>的组件）

1 个双精度参数：<**pulse_time**> (参见 Numeric Formats)

描述：Serial/Analog One-Shot 元素在由<**pulse_time**>指定的期间内驱动它的输出信号在输入信号改变时升高，在<**pulse_time**>失效后驱动它的输出信号降低。

Serial/Analog One-Shot 元素可以被重触发，意思是说它可以辨识输入的任何改变，甚至当<**out**>为高电平而导致它重新开始计算时也是如此。直到全部<**pulse_time**>已不间断流逝，输出信号才降低。

注意：<**out**>和可选信号<**out***>不是突变信号。这意味着当<**out**>的状态改变时，<**out**>和<**out***>都将在同一瞬时拥有相同的值。

65. Serial SendPro