

# 快思聪 SIMPL Windows 编程语言

## 初级教程



# 目 录

<b>第一章 快思聪 SIMPL WINDOWS .....</b>	<b>3</b>
概览 .....	3
关于初级教程 .....	3
快思聪开发软件 .....	3
SIMPL Windows .....	3
快思聪 VisionTool Pro-e .....	3
DEAL™ for Windows .....	3
Media Manager™ System Builder .....	3
D3 Pro™ .....	3
数据库 (Database) .....	3
产品目录光盘 .....	3
快思聪控制系统 .....	3
为什么对控制系统编程? .....	3
控制系统组件 .....	3
主机 .....	3
网络控制模块 .....	3
Plug-in 控制卡 .....	3
用户界面 .....	3
触摸屏 .....	3
键盘 (按键面板) .....	3
无线摇控界面 .....	3
用户设备 .....	3
控制方式 .....	3
继电器控制 .....	3
串口通信控制 .....	3
红外 .....	3
自定义串量 .....	3
RS232, RS422, RS485 .....	3
MIDI (数字音乐设备接口) .....	3
模拟电压 .....	3
自定义快思聪接口界面 .....	3
Cresnet .....	3
<b>第二章 SIMPL WINDOWS 编程 .....</b>	<b>3</b>
SIMPL 介绍 .....	3
函数库 .....	3
设备函数 .....	3
逻辑函数 .....	3
函数属性 .....	3
输入 .....	3
输出 .....	3

参数.....	3
信号类型.....	3
数字量.....	3
模拟信号量.....	3
串量.....	3
特殊信号 0 和 1.....	3
逻辑波跟逻辑解决方案.....	3
用户界面编程.....	3
按钮动作.....	3
按钮反馈.....	3
子页（仅用于触摸屏）.....	3
模拟显示（仅用于触摸屏）.....	3
间接文本（仅用于触摸屏）.....	3
用 SIMPL WINDOWS 来创建一个程序.....	3
编程步骤.....	3
基本编程规则.....	3
建立一个系统.....	3
网络硬件.....	3
控制插卡.....	3
串口设备.....	3
用户设备.....	3
网络 ID.....	3
配置设备.....	3
快思聪网络设备.....	3
以太网设备.....	3
串口设备.....	3
触摸屏.....	3
连接信号.....	3
定义用户界面信号.....	3
使用逻辑函数.....	3
<b>第三章 逻辑函数编程 .....</b>	<b>3</b>
概述 .....	3
逻辑函数的类型.....	3
基本逻辑.....	3
NOT 函数.....	3
OR 函数.....	3
AND 函数.....	3
Buffer 函数.....	3
状态逻辑.....	3
Set/Reset Latch 函数.....	3
Toggle 函数.....	3
Interlock 函数.....	3
基于时间的逻辑.....	3
One Shot 系列.....	3

---

One Shot.....	3
Multiple One Shot.....	3
Retriggerable One Shot.....	3
Delay Symbol.....	3
Oscillator Symbol.....	3
模拟逻辑.....	3
Analog Ramp 函数.....	3
Analog Initialize.....	3
Analog Preset 函数.....	3
Serial/Analog One-Shot.....	3
Modules 模块.....	3
Communication Settings.....	3
Compiling and Uploading Programs.....	3

# 第一章 快思聪 SIMPL Windows

## 概览

### 关于初级教程

编写本教程的目的在于向程序设计人员介绍 **SIMPL windows** 编程技术以及如何应用快思聪控制系统，这包括对控制系统如何利用触摸屏和按钮作为用户界面的理解，通过这些界面，使用者可以发送一个信号（主要是逻辑函数）给控制系统处理后输出用以最终控制一个设备。

简单的控制系统



当然，实际的控制过程比以上的更加复杂多变，然而这只是快思聪程序控制系统编程的基本概念，该教程要求使用者需对以下内容有基本了解：

#### 微软视窗

- ◆ 了解windows基本操作
- ◆ 熟悉windows特性及功能

#### 音/视频

- ◆ 了解不同控制方式（串口，红外，继电器）
- ◆ 熟悉A/V设备
- ◆ 阅读理解控制系统连线图的能力

**SIMPL windows** 提供了大量各种各样的函数，可以用来实现所有可能的实际应用，当对 **SIMPL windows** 变得精通后您就能知道对同一控制问题我们有多种方式来解决，这使得程序更具灵活性及扩展性。

## 快思聪开发软件

### SIMPL Windows

快思聪 SIMPL windows 提供了配置，编程，测试及调试一个集成控制系统需要的所有工具。综合，快思聪 SIMPL windows 结合 Windows 风格的强大拖曳功能和强大的编程能力，建立了快思聪硬件，用户界面跟受控设备之间的联系。

SIMPL windows 配置方面的功能允许您选择安装所需的控制系统，用户设备，网络设备和控制设备。您可以针对这些组件来安排端口地址，网络 ID 和 IP 地址，设定通信参数以及指定哪个设备跟哪个卡连接或者网络控制模式。您也可以指定系统要求所需的 Visiontool Pro-e 触摸屏程序。

编程时允许您选择系统要求的逻辑函数，为函数分配信号且根据逻辑要求，建立信号跟其他函数或设备之间的联系。SIMPL windows 提供了大量各种各样的函数，可以用来实现所有可能的实际应用。随着您对 SIMPL windows 的熟悉，就会明显发现可以用很多方式来解决同一个控制方面的问题，这样就让程序更具灵活性及扩展性

您可用强大的诊断工具来测试调试 SIMPL windows，这些工具包括 Test Manager，Network Analyze 和 Viewport。您可在 SIMPL windows 中调用这些工具，也可以独立运行。

为了实现更多灵活性，SIMPL windows 安装包中包含了 SIMPL+，这个开发工具可以让高级编程人员用类似 C 语言的程序语言创建和编译客户控制程序模块。您可以像添加一个逻辑函数一样将 SIMPL+ 模块添加到 SIMPL Windows 程序或者定制的用户模块中以用于功能扩展或者解决特定的控制问题。

SIMPL windows 完全集成的快思聪软件工具包包括：

### 快思聪 VisionTool Pro-e

VisionTool Pro-e 是快思聪触摸屏界面设计软件，应用 VisionTool Pro-e 软件，程序人员可以创建强大的触摸屏控制界面，包含用于特定设备传输控制的 pop-up 子页面，多种风格的按键以及 3D 效果的滚动条，高分辨率图片，动态文本，视频窗口，声音等等。VisionTool Pro-e 使用 Join Number 来定义按钮按下，反馈，以及其他数字，模拟和串口信号。这些 Join Number 与 SIMPL Windows 中触摸屏函数的输入和输出相对应。

### DEAL™ for Windows

快思聪 DEAL™（Device Editor and Learner）for Windows 软件能够让程序人员可以学习其它厂商的红外信号。结合快思聪红外学习器（CNXLIR）使用，DEAL 允许您创建、修改和测试红外驱动文件，并且将红外驱动文件加入用户数据库，通过该数据库您可以将红外驱动文件加入到 SIMPL windows 程序中。

### Media Manager™ System Builder

媒体管理系统建立工具提供对于例如音频分配，家庭影院和视频会议等家庭及商用应用的自动化程序，该工具提供一个向导式的界面，简单的根据提示选择控制系统，用户界面，设备及功能，创建工具便会自动的生成程序、编译、上传系统，包括 VisionTools Pro-e 触摸屏程序 and 控制系统逻辑程序。

## D3 Pro™

快思聪 D3 Pro™ 软件为家居灯光系统（包括安防系统、动作感应和幕帘等附属设备）提供了设计、建立和存档功能，与 SystemBuilder 一样，D3 Pro 也提供向导式用户界面，编程可通过一系列简单但功能强大的系统设置界面来完成。设计完成后，D3 Pro 自动创建、编译和上载控制系统程序以及触摸屏程序。

这只是快思聪提供的帮助您更加方便快捷完成编程任务的部分软件，您可以在快思聪网站的软件升级区来免费下载所有快思聪软件（要求注册）。

## 数据库（Database）

快思聪数据库（Crestron Database）是一个大型的信息集合，快思聪的各种软件包括 SIMPL windows，D3 Pro 和 System Builer 均访问该数据库，数据库中包含用于控制用户设备（如 CD，DVD，会议设备和其他第三方红外设备）的红外驱动文件。

除了红外驱动文件，快思聪数据库还包含有几百个用于控制第三方设备的快思聪逻辑模块，模块包含由快思聪预先编写、检测、调试过的逻辑程序。这些模块可以添加到程序中用以自动生成控制设备的代码。

用户数据库（User Database）用于存储快思聪数据库中没有包含的红外驱动文件，编程人员通常用快思聪 CNXLIR 软件结合基于 Windows 的 DEAL 软件来生成红外文件。您也可以通过从快思聪设计中心或 FTP 网站下载用户红外文件。

此外，用户模块（User Module）目录用来保存快思聪数据库（Crestron Database）中没有的由用户自行创建的逻辑模块。

## 产品目录光盘

快思聪为您提供了多种获得快思聪硬件信息的方法，最全面的资源就是快思聪网站：[www.crestron.com](http://www.crestron.com) 在这里，您可以下载最新的用户手册，参考指南和所有控制系统、网络模块和触摸屏的 CAD 图。您还可以进入快思聪设计中心（Crestron Design Center）那里提供相关硬件合作制造商那获得的控制设备用的用户模块的扩展信息，包括帮助文档，简单逻辑程序，触摸屏程序，CAD 图以及其他相关资料。

您可以通过 SIMPL windows 在线支持进入快思聪网站，点击 Crestron online 进入快思聪主页，或者点击快思聪设计中心打开分/技术支持主页。

快思聪产品目录和技术参考光盘是另一个有价值的工具，您可以结合快思聪网站来使用，或者当您无法上网时候，该光盘是一个有竞争力的快思聪目录，产品列表，CAD 图和用户手册的图书馆。您可以不通过任何快思聪产品来浏览光盘，或者您可以直接通过 SIMPL windows 来显示您选择的各种信息

### 从SIMPL windows进入用户手册

- 1.将光盘放入CD-Rom（如果有自动读取，可以关闭窗口）
- 2.在SIMPL windows选择您需要查看的信息的whichever条目，快思聪控制系统，网络设备，触摸屏或控制卡，并按F1
- 3.第一次尝试通过SIMPL windows进入产品目录光盘的时候，将会提示浏览光盘目录，选定驱动式文件夹点击open
- 4.如果选定设备文档存在，SIMPL windows会找到并用Adobe Reader打开PDF文件，如果没有相关设备的PDF文档，那SIMPL windows帮助文档会显示设备的帮助程序
- 5.无论何时您想打开文档，CAD图或者附件，您都可以在SIMPL windows帮助菜单上点击**产品目录光盘**

- 
- 6.如果您没插入CD就在设备文档库中按了F1，SIMPL windows会提示您插入光盘，您可以选择插入光盘或者Cancel来在线浏览帮助文档。



## 快思聪控制系统

### 为什么对控制系统编程？

程序读取存储在控制主机中的指令代码，使主机按照程序进行运作。比如，要控制 DVD，您必须事先编程告诉控制系统，DVD 连接到哪个端口，发送什么样的 IR 代码，触摸屏端哪个按钮驱动这些功能。一般来说，一个程序可以包含几百条相似的指令去控制整个机架上的音视频设备。所有程序都用 SIMPL 语言编写，快思聪开发了 SIMPL Windows 开发环境以方便快捷的编制程序。

### 控制系统组件

#### 主机

快思聪控制系统主机是整个遥控系统的核心，它集成其它厂商的设备和进行相互通讯。控制系统的内存（RAM）必须通过特殊的指令或程序进行编程才能与其它需要控制的设备进行通讯。

另外，控制系统包括一个操作系统（OPS）。和计算机操作系统相似，OPS 是一套指令，使控制系统可以执行程序去控制连接在系统中的各种输入输出（I/O）设备，如红外设备。

如果要用到最新的程序功能、最新的快思聪设备或者纠正以前版本的错误都需要升级 OPS，您可以从快思聪网站下载升级软件。OPS 文件名根据不同的主机有不同的扩展名，如 c2.V3080.cuz。下载前请注意确认升级文件是否是适合您的操作系统版本，文件名是否匹配操作系统版本号并且扩展名和主机型号应一致。

**2 系列主机**通过 CUZ 文件为控制系统装载操作系统。2 系列主机提供 32MB DRAM，通过 CF 插槽可将 DRAM 扩展至最大 4G。程序的大小、模拟、数字及字符串信号的数量只受可用 RAM 空间的限制。另外，主机有 256KB 的非擦除内存用于存储 SIMPL+变量以及 SIMPL 中的一些存储函数的变量。这些函数包括 Analog RAM、Digital RAM 和 Analog Non-Volatile Ramp，通常用于灯光和音量的预设。当关机时，非擦除内存也会保留里面的数据。256K 的非擦除内存也可以分配出 64K 或 128K 当作非擦除盘使用。

**X 系列主机**另外有一个基本的监视器来操作系统，也有一些独立的 TCP/IP 堆栈，包含在一个 UPZ 文件中。这些独立的堆栈将用于 CNXENET 或者 CNXENET+以太网卡通信系统中。

X 系列主机支持 16373 用户自定义数据信号和 2048 用户自定义模拟/连续信号。主机也有 256K 非擦写内存，根据所使用的以太网卡内型的不同按不同的方式划分。

ST-CP 和 CN 系列的 Legacy 控制主机支持 4085 用户自定义数字信号和 512 模拟/连续信号。

主机	最大信号数量
2 系列	取决于可用 RAM
X 系列	16373 数字信号 2048 模拟/连续信号
ST-CP 和 CN 系列	4085 数定信号 512 模拟/连续信号

## 网络控制模块

网络控制模块是连接快思聪网络或以太网用以扩展控制系统功能和支持第三方设备。快思聪提供各种不同的网络控制模块，包括音频接收器、混音器、分配切换器、立体声处理器、视频处理器、摄像机控制模块和房间盒模块。

任何 2 系列主机可以作为从属设备被其它 2 系列主机控制，以作为一个功能更强大的网络控制模块使用。

在 SIMPL Windows 设备库里有各种网络控制模块代表快思聪的控制模块和以太网控制模块，灯光控制模块在 Lighting 文件夹中。

## Plug-in 控制卡

快思聪 Plug-in 控制卡可以安装在主机扩展槽里，支持与设备通信。Plug-in 控制卡包括用于连接 2 系列或 X 系列控制系统到以太网的接口卡，在 SIMPL Windows 设备库里有 Plug-in Control Cards 代表控制卡。实际上可以通过安装并配置控制卡支持系统控制各种各样任意数量的设备。

许多控制模块比如串口设备，既可通过 Plug-in Control Cards 也可以通过 Network control Module 找到。

卡通常成本较低，因为不需要机架安装和供电，但控制卡因为主机扩展槽的数量有限也受到限制。

## 用户界面

用户通过用户界面发出各种需求和动作，快思聪生产了各种各样的控制界面，包括简单的低成本手持式遥控器、键盘和高端的触摸屏。

## 触摸屏

快思聪触摸屏是用得最多的控制系统用户界面，触摸屏可以选用黑白显示和彩色显示，可用于快思聪网络、以太网和无线网络。

编程任意通过 Vision Tool Pro-e 软件开发触摸屏的用户使用界面，各种按钮可以定义相应的数字与一些在 SIMPL Windows 程序中定义的特殊的操作相关联。这些关联叫 join number，后面我们将会具体讲到。

## 键盘（按键面板）

键盘只能简单的去操作快思聪网络。按键提供古典样式，可以选配不同的按键和经过抛光的面板。

## 无线摇控界面

快思聪无线触摸屏和摇控器需要配制快思聪无线网关设备（如 CNRFGWA, CNIRGWA 和 CNRFGWX），无线网关通过快思聪网线（Cresnet）连接到控制系统。无线红外（IR）/射频（RF）发送器只单向发送 IR/RF 信号，不能接收。同样，快思聪 CNIRGW 只能单向接收 IR 信号，CNRFGWA 只能单向接收 RF 信号。

## 用户设备

用户设备包括音/视频设备，如 CD, TV, VCR 等由快思聪控制系统进行控制。在 User Device 文件夹中有几百种按生产厂商和设备类型分类的设备驱动文件。

## 控制方式

在编程运行快思聪控制系统时，最重要的是要了解如何去控制设备。任何具备电器接口的设备均可以通过快思聪控制系统进行控制。最常见的控制方式有：

- ． 继电器开关控制（机械方式或固态方式）
- ． 串口通信控制
- ． 模拟电压控制
- ． 定制快思聪专用接口控制

### 继电器控制

许多设备只需通过简单的电气开关就可以去触发相应的功能。在控制领域，这些是通过继电器进行控制的。屏幕、窗帘和第三方灯光控制系统都更偏向于采用这种类型的控制方式。除此之外，不需要调光的灯光系统常用继电器进行开关控制。快思聪生产多种类型的继电器，在不需承载大电流或高电压时可选用低压继电器，而高压继电器则用在需要控制电机和灯光回路方面。另外，继电器可以选用机械和固态两种类型。如果在应用时不确定选用何种继电器，请致电快思聪公司以获得技术支持。

### 串口通信控制

当今许多设备用各种类型的串口通讯方式进行控制。一般来讲，通过串口控制设备常用的有红外，RS-232，RS-422，RS-485，MIDI 方式，或自定义串口方式进行控制。在接下来的章节我们将讨论几种方式的不同之处。

#### 何为串口通讯？

串口通讯指的是采用逐条发送和接收的一种通信方式。打个比方，设想一下我们在打电话的时候，对方讲的每一句话都是一个字接一个字讲出来的。它不同于并行通讯，并行通讯是几条信息同时进行发送接收。

串口通信包括很多种常用的格式快思聪控制系统都兼容。接下来的章节我们将具体讲解最常用的格式。

### 红外

很多年来，红外遥控都非常普遍，直至今今天它也是串口控制中最常用的一种方式。正如它的名字一样，红外控制通过红外进行传输。IR 信号通常通过载波信号进行调制，虽然有些可以高达 1 MHz，但是通常载波信号都在 40MHz 左右。

对于快思聪，红外控制有两种应用。快思聪无线用户界面可以用红外方式与控制系统通信。这时，红外通过一种合适的格式被快思聪设备发送和接收。

#### 快思聪 IR 无线接口



其它的红外控制应用是系统产生红外信号去控制其它厂商设备（像索尼和松下的设备）。系统可以对设备进行远程功能控制。

因为 IR 是单向通信，所以从受控设备端不会收到任何反馈。就是说数据传送到受控设备端，但没有数据从受控设备端返回到控制系统。意思就是当使用红外控制时，受控设备不会发送反馈信号来告诉您发送的指令已收到。这是采用这种控制方式本身所决定的缺点。IR 的另一个缺点就是控制系统和受控设备之间不能有障碍物阻隔。针对这个问题，快思聪提供红外发射棒，红外发射棒通过线缆连接到受控设备的红外接收器上。需要注意的是要确保红外发射棒上的探头（发射器）安装在紧靠发射棒的位置。



生产商一般不会公开发布红外远程控制的数据传输协议，因此，为了使红外发射卡发出正确的控制信号，必须使用专用设备——红外学习器学习红外代码。在电脑上安装快思聪红外学习软件—DEAL FOR WINDOWS，将红外学习器连接到电脑上，对红外遥控进行红外代码的学习，将学习生成的红外文件保存到数据库后，此后就可以在您的程序中插入这个红外模块了。当程序完成并上传到控制系统后，控制系统的红外卡就可以将驱动文件的信息转换成适当的电信号。

### 快思聪红外设备

#### 红外发射器探头（CNXIRP 和 STIRP）

红外发射器探头是一种非常小、有线传输的红外发射器，是快思聪用来外连到红外受控设备的红外接收窗上的工具。它连接于控制系统和设备红外窗口之间，并向受控设备直接发送红外信号。因为红外发射器探头可以外接于设备上，所以无需启用第三方设备来重写控制代码或调整红外窗口。另外，通过直接将红外发射器探头安装到受控设备上，来自日光和灯光的干扰信号可以忽略不计。

#### 红外 Sprayer

红外 Sprayer 红外发射器中的红外 Sprayer 可以 Sprayer 90 度的红外信号，它无需红外探头，并且可以被摆放于中心位置，与所以设备相连。它被设计用来同时处理多个红外代码，所以仅需一个 Sprayer 就可以处理许多红外设备。

#### 红外设备模块/控制卡

红外设备模块，如：C2IR-8 或控制系统内置的红外口，提供红外或某些串量受控设备的控制。C2IR-8 和内置红外口需要红外探头进行红外通信。

使用红外口进行串量通信时，请参看串量红外通信的有关介绍。

#### 红外学习器（CNXLIR）

快思聪电器设备 CNXLIR 用来学习红外遥控的控制代码。通过学习这些代码，程序员可以创建客户的红外设备驱动。DEAL FOR WINDOWS 软件允许程序员创建、修改和测试驱动文件。程序员可以对它进行存储和学习。用户数据库中的红外文件在 SIMPL Windows 中使用。

### 快思聪数据库

正如前面所述，快思聪数据库包含成千上百个预编码的红外驱动文件供程序员使用。这个数据库涵盖了当前市场上绝大部分的红外受控设备的红外驱动文件。实际上，此数据库支持目前所有控制格式，包括继电器控制、模拟电压和 TCP/IP。程序员可以通过厂商或设备类型来查找数据库。

## 自定义串量

此处自定义串量用来描述通信协议（类似于红外通信协议），但它是有线的，而不是光脉冲，并且没有载波频率。称为“自定义”是因为当前大部分生产商采用这种方法，但没有一个统一的标准。索尼 Control-S 和 Marantz 的 RC—5 标准是当前使用自定义串量格式的例子。

在使用方面，串量通信与红外通信唯一不同的是用专门制作的有线线缆代替红外探头，连接控制系统和受控设备。因为特定类别的红外遥控的数据格式是统一的，所以生成串量驱动文件首先要通过对红外遥控的学习产生红外驱动文件，再将此文件传输到专门的筛选器筛除掉载波。正如红外控制一样，自定义串量信号通过红外卡产生，如：C2IR-8。

### 快思聪自定义串量设备

**CNSP-109:** 是一种家电/Vidikron 线缆，和 C2IR-8 或串量输出端口一起使用。

**CNSP-110:** 是一类索尼 VO5000, 7000, 9000 串量 Vmatic 线缆，和 C2IR-8 一起使用。

**CNSP-112:** 是一类索尼 Control-S 线缆，和 C2IR-8 一起使用。

## RS232, RS422, RS485

RS232, RS422, RS485 都是 EIA 开发的串量通信标准。标准指定了设备间的电气接口。这些标准的发布使得各种不同设备间可以通讯而不用考虑硬件的特殊性。相同标准的任何设备间应该能够进行通信。在这三种格式中，RS232 是目前控制系统中使用最普遍的。在以后的章节中，如无标示，都用 RS232 来描述三种协议。

与红外控制和自定义串量格式不同，RS232 控制不使用预编码的驱动文件，它的数据格式或协议，即受控设备所需要接收的内容将在单元手册中详细描述。协议包含传输和接收数据、通信速率、奇偶校验、数据位个数和停止位个数。另外，给定的设备需要硬件（RTS/CTS）或软件（XON/XOFF）握手，它控制了两个设备间传输的数据流。控制程序对所有这些元素进行调整，以匹配生产商的具体要求。

由于没有驱动文件，人们一般都认为在编程方面，对 RS-232 的控制比对红外和自定义串量的控制要困难。这是因为每当一个 RS-232 设备需要被编程控制时，程序员必须查找相关的协议，并在程序中编写必要逻辑来发送数据。在这个方面，许多设备都有了相应的模块，这些模块可以插入到程序中，并自动生成所有的代码。

RS232, RS422 和 RS485 在物理特性上的区别对程序员没有影响，除了他必须确认使用的快思聪产品支持的格式和设置正确。RS232 使用一线传输数据，一线接收，最远可达 50 英尺。但这也要受许多因素的影响，如：线缆质量、波特率和信噪比的影响。RS422 格式使用一对平衡线传输，另一对用于接收。平衡线对信噪比要求不高，它的信号最远可达 2000 英尺。最后一个标准 RS485 与 RS422 类似，除了单对线导体用来同时传送和接收数据。这一点使得 RS485 在网络应用上很富吸引力，因为网络上的数据需要被共享。一个 HVAC（中央空调系统）应该是一个典型的案例，该系统是通过一个 RS485 局域网与各种不同的温控器和一个控制系统进行通信。快思聪 C2IR-8 扩展控制卡仅能用于 RS232 的单向传输。C2COM-2 扩展控制卡能够产生 RS232、RS422 和 RS485 双向的信号。ST-COM 网络设备可以产生 RS232、RS422 或 RS485 双向通信数据。

### 快思聪 RS232、RS422 和 RS485 设备

#### C2IR-8:

8 个串口，双向 RS232 通信协议

#### C2COM:



扩展控制卡，2 个双向 RS232/RS422/RS485 (DB9) 端口和硬件握手信号

**提示：**C2COM-2 上的 DB9 pin-outs 输出口不是标准的 RS232 端口，如过用直通的串量通信线缆连接，可能会损坏设备。请参照快思聪线缆数据库或联系快思聪获得 pin-outs 串量线缆的详细资料。

#### ST-COM:

网络设备，满足 RS232、RS422 和 RS485 协议标准

**限制：**RS232 传输距离最大可达 50 英尺 (15 米)，至少需要 3 线传输 (RXD、TXD、GND)，每个设备需要用特定格式来接收数据。程序员需要熟悉二进制、十六进制和/或 ASCII 码来编写合适的字符串变量。

### MIDI (数字音乐设备接口)

MIDI 支持数字音乐设备接口，也是另一种串量通信标准。顾名思义，MIDI 通常用来支持音乐设备间的通信。但是，某些时候，在控制系统应用中，一些音频混音器使用 MIDI 控制寻址。从程序员的角度来看，MIDI 与 RS232、RS422 和 RS485 没什么不同。从硬件角度看，CNX-MIDI 卡用来生成恰当的控制信号。

#### 快思聪 MIDI 设备

CNX-MIDI (卡)：控制卡，1 个 MIN IN 端口，1 个 MIN OUT 端口和 1 个 THRU 端口，和混音器、灯光设备一起使用。

### 模拟电压

某些设备，特别如相机 pan tilt 头、灯光控制系统或电压受控 attenuators 单元，可以用同一个模拟电压控制。可编程模拟电压能通过 CNXAO-8 卡或 C2I-IO8 生成。后者含有 8 个 Versiports，可对数字输入/输出或模拟输出信号进行编程。

### 自定义快思聪接口界面

一些设备的控制接口不能简单地归到一种方法或目录类别。在这些情况下，快思聪开发了自定义模块（或者是扩展控制卡或者是网络模块）来提供控制。

例子有：

线缆音频 attenuation (音量控制)

pan/tilt and zoom/focus control(镜头控制)

滑块投影机控制

键盘/鼠标接口

灯光 (可调和不可调) 和马达控制模块

上述的详细信息，请参看快思聪目录查找各模块和参考模块产品的用户手册。

### Cresnet

快思聪网络或 Cresnet。涉及到快思聪使用的网络 topology，RS485 总线用来连接控制系统和快思聪网络设备，如：CNECI-4A (电子控制接口用于交流电设备) 或 CNSC-IA (滑块投影机接口)。当红外和 RS232 不能满足设计需求时，RS485 总线可以用来远程定位设备。如：红外必须用红外发射其探头外连到设备上，RS232 的最大传输距离只有 50 英尺，而 Cresnet RS232 的最远传输距离为 5000 英尺。

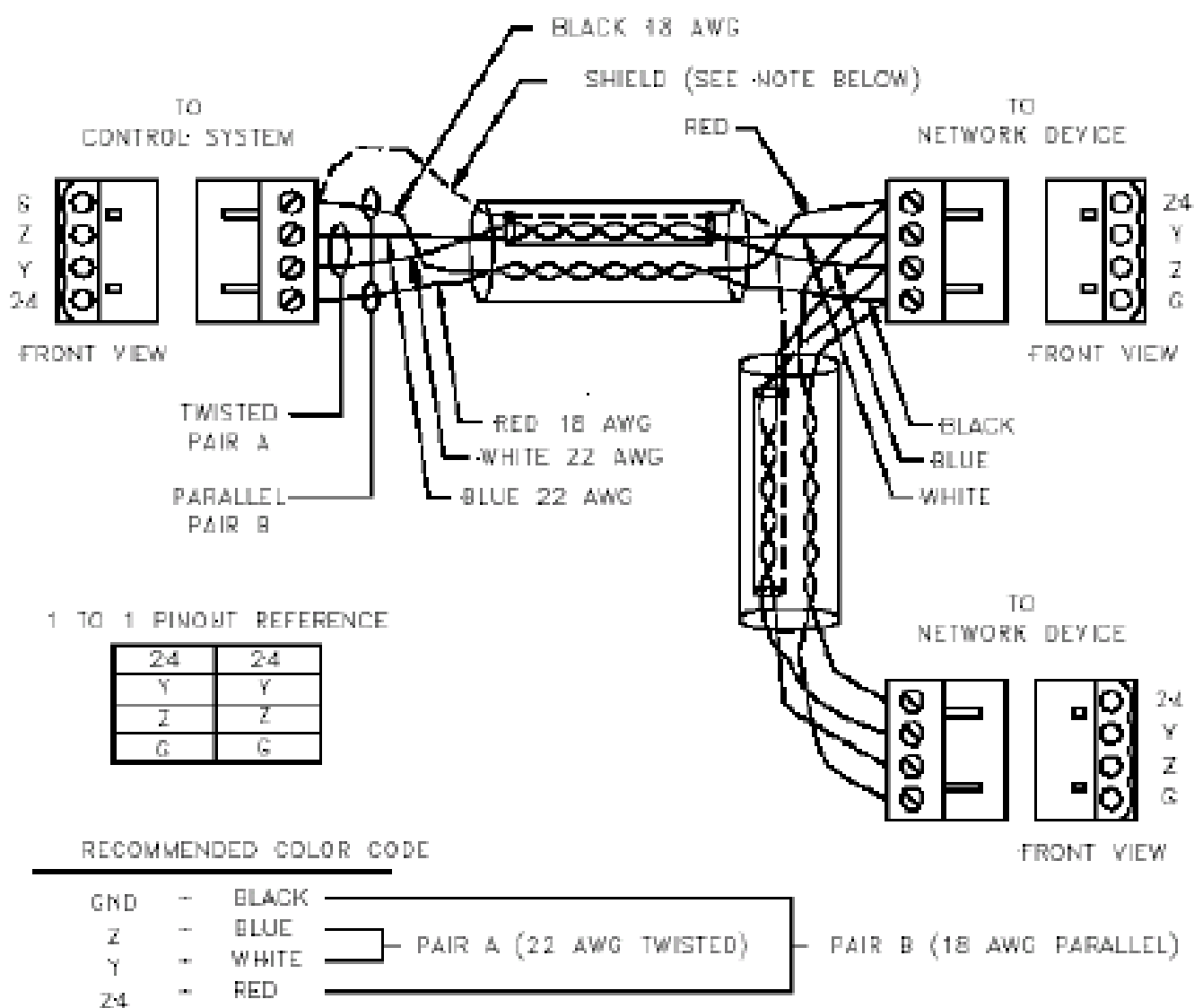
## Cresnet 线缆组成

A 对 #22 AWG, 双绞线对+数据线屏蔽

B 对 #18 AWG, 双绞线对, 用语电源和地

PVC jacket(组)

快思聪网络联接图



## 第二章 SIMPL Windows 编程

### SIMPL 介绍

快思聪工程师们致力于产品及与其他制造商之间界面的开发。然而，为了能够更加客户化的实现每个安装，控制系统需要独立的编程。

快思聪控制系统是利用 SIMPL Windows 编程（函数增强管理编程语言）

SIMPL 是一种面向对象得编程语言，能够方便的实现控制系统要求，这些用于 SIMPL 的对象叫函数，每个函数都能实现一系列特定应用。

依靠信号量来实现函数之间的链接。将函数及他们之间得相互关联组合起来就是一个程序。因此，程序实际上就是由对象（函数）跟链接线路

（信号量）组成的图。这种图也可以理解成其他应用中得块图或流程图。当设计一个 AV 系统安装时，表示系统设备如何链接的块图对安装人员

而言是非常重要的。SIMPL 允许编程人员用相似的风格来开发一个控制系统。应用到的函数以及连接它们的信号量的链接组合起来就类似一个块图。SIMPL 程序的开发与 AV 系统安装的流程图紧密关联。

### 函数库

在 SIMPL 中写程序类似连一个电路图：您需要选择正确的组件并将它们正确的连接起来。正如描述的一样，在 SIMPL 中，组件叫做函数，线叫做信号量，与真实世界中的电子一样，要从大量的函数中做选择来完成目标。随着您对越来越多的系统进行编程，您会找到一个大多数环境下都适用的函数组合。

SIMPL 中的函数可分为两个大类：设备函数和逻辑函数。

### 设备函数

设备函数代表程序中用到的快思聪网络设备，它们只能用 SIMPL 软件中的配置管理器来加入或删除。程序管理器允许设备函数之间互连，但不能加入或删除。设备函数位于配置管理器的设备库中。

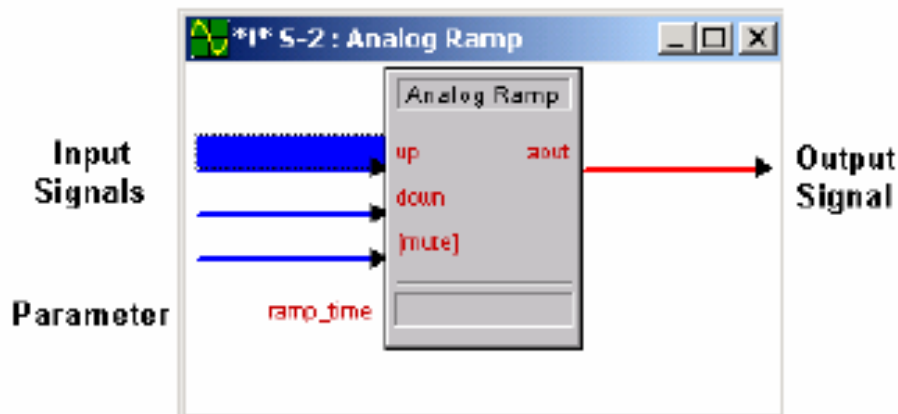
### 逻辑函数

设备函数允许您与外部设备互相通讯，逻辑函数允许您的程序按照您的思路来动作。逻辑函数由最基本的如与，或，非函数到用于特殊应用的函数都有。在函数编程章节中有更深入的讨论。

### 函数属性

虽然每个函数都有特殊的应用，但所有函数都有一些基本属性，他们是输入，输出和参数。





## 输入

函数输入允许函数与程序其他部分相连，根据函数类型，输入信号的当前状态会影响一个或多个输出信号。一些函数有固定数量的输入，然而另外一些有可变的输入数量，由编程人员根据需要来决定。

## 输出

除一些特殊案例，大多数函数的目的是用于更改输出状态，这些输出状态基于函数类型，当前或过去的一些输入信号状态以及参数值。因为函数决定了输出信号，因此函数被认为是输出信号的驱动源，根据信号属性，一些输出会有多种驱动源。

类似函数输入，一些函数也固定了输出数量，另外一些函数有可变的输出。

## 参数

一些函数有参数，这些参数是一些常数值，有助于决定函数是如何动作。举个例子，有一个用于对一个动作延迟一段时间触发的函数，就有一个参数决定延迟多长时间，参数的功能只取决于函数自身类型。

为了方便，参数有多种格式的表示方式（它们之间所有都是相互直接关联的）虽然一个参数可以根据函数类型用一个格式来定义，您也可以在函数值的最后通过改变格式特性来改变格式。

以下列了有效的格式，括号中的字母表示格式定义的标志

- . (d) decimal
- . (h) hexadecimal
- . (%) percentage of 65535
- . (s) second
- . (t) ticks-1 tick = 1/100 秒 (2-series); 或者 17112.5 秒 (X-series)
- . (') character(')(单字节)

为了给一个参数定一个特定格式，在数值后加入定义符号，如 25%，如果参数是单字节，在 ASCII 特性前或后设置信号定额

参数也能特定每日时间，在此，每日时间表示为后面带"秒"格式标志的军用时间，如下：

.HH.MM.SS

.MM.SS.HSs

.SS.HSs

.SSs

.HSs

这里 HH=小时，MM=分钟，SS=秒，HS=百分之一秒

举个例子，参数 20.03.05S 表示一个时间值为 20 分钟，3 秒和 5/100 秒的参数。如果您不使用，当应用这个符号时候就会忽略那些大的时间单位。所以，“3.00.00S”表示 3 分钟，0 秒，百分之 0 秒（不表示 3 小时）

根据函数功能，一个参数分为有符号数和无符号数，有符号数介于-32768~+32767 之间，无符号数的参数介于 0~65535。百分数也可以用负数来表示，例如-25%=65535 的 25%，即 16384（-16384=49152d）因此参数的 25%等同于 49152d。参考 SIMPL 的帮助文档可获得正确参数值的更多信息。

注意参数就是在程序编译过程内必须被调用的常数值，在程序运行过程中无法改变参数值（信号不可安排给参数），要改变参数，必须改变并重新编译程序。

## 信号类型

信号的概念已经提过了，信号就是在程序中用于链接组成程序的各种设备函数跟逻辑函数的元素。然而，对信号的讨论并未就此结束。对初学者而言，信号量有三种类型：数字量，模拟量跟串量。对任何给定信号而言，信号类型取决于驱动源，如果输出模拟量的函数，那么链接那的信号会自定义为一个模拟信号，三个类型的信号更详细的定义如下：

### 数字量

数字量是 SIMPL 语言中最常见的，一个典型的程序有 95%~100%的信号量都是数字信号量。这类型的信号量有两种状态，通常认为是开/关，其他常见描述是高/低，活动/非活动，或 1/0。值的跳变称为上升沿或正态沿。总的来说，SIMPL 程序中的动作是由数字信号从低到高的状态变化来触发的。虽然大多数信号是边沿触发，但另外一些是水平触发（基于当前状态而不是最后的转换）举个例子，一个 Toggle 函数是边沿触发的，它随着每一个输入的上升沿来驱动数字量输出的高低，作为比较，Buffer 函数是水平触发，为了让信号流动，它的 Enable 输入数字量必须保持高电平。

但浏览函数库参考时，记录哪些函数是边沿触发，哪些是水平触发，您可以通过选择函数并按 F1 来找到这些信息。这么做能打开对应函数的文本帮助窗口

如前所说，信号类型由驱动源决定，在许多应用中，有一些是具有多个驱动源的信号量。这些信号量被称为拥塞。作为通用规则，数字量不能拥塞，这意味着它们只能有唯一的驱动源，但有两个重要且常见的函数例外，系统输入（例如按键）和 buffer 函数的输出。这些例外在必须通过多个用户界面共享的应用中十分有用。举个例子，DVD 播放器可以通过触摸屏来控制，也可以通过远程控制器来控制。

在 SIMPL 中，数字量可由蓝色箭头表示。

### 模拟信号量

模拟信号量用 16 字节数字表示，所有值介于 0~65535 ( $2^{16}-1$ ) 之间，这意味着不同于数字信号，模拟信

号可以在数值间连续变化，作为音量或温度之间的应用。该属性可以使模拟量在控制那些没有开关判断的设备时十分有用。例如音量控制器，镜头摇杆控制器和灯光控制器。

与数字量不同，模拟量由于它们的特性，可以有多个驱动源，因此模拟量是可拥塞的。实际上，无论何时超过一个信号量驱动同一模拟信号，持续改变信号值的函数就会动作，模拟信号的典型应用是音量，灯光和温度一集高级串量操作。

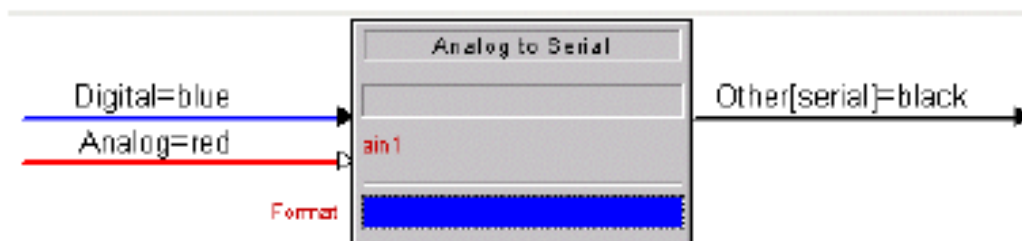
在 SIMPL 中，模拟量用红线表示。

## 串量

串量促进了字符串的传输，这些信号由 com 口或通过一个有串量输出的函数生成。与模拟量类似，串量也是可拥塞的，因此在单个信号上可用多个函数生成字符串。默认的，串量是短暂的，这意味着串量只对生成它的逻辑波有效（逻辑波在下面介绍）例如 Making String Perment 函数允许字符串保存在内存中

在 SIMPL 中，串量用黑线表示。

### 举例：信号颜色



有些信号比较模糊，这意味着信号类型取决于输入源。举个例子，Serial buffer 函数是串函数也是模拟函数，当信号源与一个模拟量或者串量相连时，就可决定模糊信号类型。在程序完成前应该确定模糊信号的类型，否则就会出现编译错误。

在 SIMPL 中，模糊信号用绿线表示，一旦确定，绿色会变成蓝色（数字量）红色（模拟量）或黑色（串量）

## 特殊信号 0 和 1

特殊信号量 0 和 1 用于给一个信号量赋值。1 表示一个值恒为高的纯数字量。数字量 0 总是保持低电平，在模拟量中“0”是强制给信号赋常数值 0，在串量中，输入 0 结果是没有字符串输出

## 逻辑波跟逻辑解决方案

一个逻辑波是一个衡量信号状态变化时刻与所有与之相连的函数的求值时刻之间的这段时间的处理单元。对于当描述数字硬件时应用的“传送延时”过程来说是模拟的。虽然由于实际的时间与编译时间互相决定这一事实让逻辑波不能表示为真实的时间单元（例如百万分之一秒），但 SIMPL 保证所有的函数都有一个确实的逻辑波传送延时。

注意有些函数并不一直符合这个规则，一个基于时间的函数例如 Delay 与 one shot 函数，Delay 函数的延时只由参数值来决定，一个 one shot 函数的输入触发信号升高也有一个信号逻辑波的延时，然而，输入的下降沿无效，且函数输出只有在这个特定时间之后才会下降。

一个或多个逻辑波组成**逻辑解决方案**

逻辑解决方案由它们所在的时间来定义。始于一个外部脉冲，用于 SIMPL 逻辑处理器对所有函数求值，至

程序中所有信号达到稳定状态的那个点为止，即所有信号保持稳定不变状态的那个点。逻辑解决方案的长度可在运行时间内变化且可用逻辑波表示，例如当按键按下，并发逻辑解决方案应发出 6 个逻辑波。

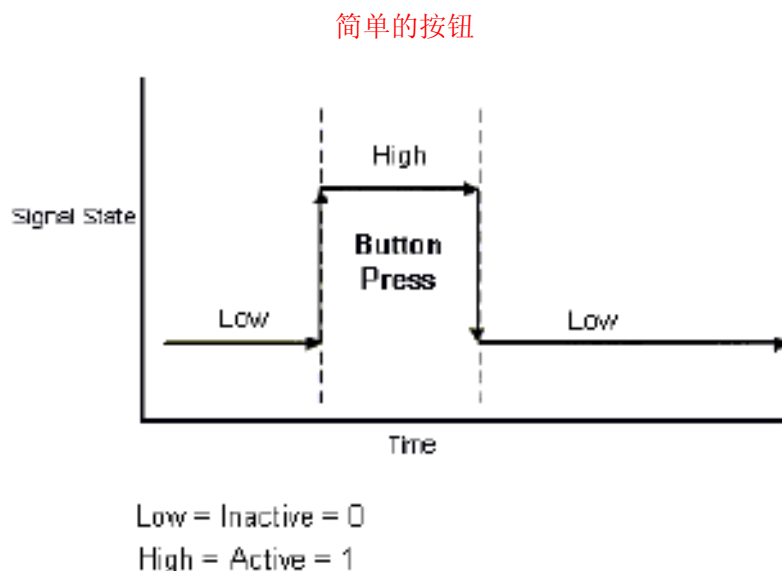
实现当一个逻辑解决方案运行时，基于日程事件不可触发时很重要的，除了作为脉冲来触发。因此，用一个 **Oscillator** 函数不能导致一个无结尾的逻辑解决方案。相反的，每次 **Oscillator** 的输出信号改变时，将触发一个新的逻辑解决方案运行。直到所有影响信号达到最终状态后才停止。另一方面，错误链接逻辑刻产生无穷的逻辑解决方案（例如将 **Nor** 函数的输出引入到输入）这种情况时可以避免的。

## 用户界面编程

用户界面是任何一个设计优良的控制系统核心，它是用户与控制系统之间的桥梁。无论系统的程序的智能程度和技术程度有多高，如果缺乏一个良好的用户界面，系统就不可能得到用户的肯定，也不可能完全发挥其潜能。从高端的 **Isys TPS** 系列触摸屏到简单、高性价比的有线按键面板，快思聪控制系统提供了一系列出色的用户界面。无论您在系统中应用什么类型的界面，本章将解释在程序中如何应用这些界面。

### 按钮动作

在程序中，触摸屏、有线无线按键面板或其他界面按钮的动作都与信号相关联。设备中并非所有的按钮都有用，未使用的按钮不需为其指定信号名称。当某个按钮按下时，程序中对应的信号变成高电平；当按钮松开时，对应信号变成低电平。下图为信号状态变化示意图：



图中按钮决定信号的状态。即当按钮按下信号变为高；当按钮未按下时信号为低——因为按钮是信号的驱动源。按钮只有高低两个状态，因此它驱动的信号也只有两个状态，所以该信号为数字信号。

大多数情况下数字信号只能有一个驱动源，两个不同函数驱动同一个信号是非法的。不过有两个例外：一个是后续会讲到的 **Buffer** 函数；另一个是按钮信号，可以有多个按钮驱动同一个信号，这方便在多个用户界面上实现同一功能。例如：通过触摸屏和无线遥控器实现对同一音量高低的控制。在以上两种情况下，使用相同的信号名称是合法的，而不必为每个信号命不同的名称，然后用 **OR** 函数将不同名称的信号连接起来。

### 按钮反馈

除单向无线界面之外的所有双向通信用户界面都支持按钮反馈。我们用反馈来描述按钮的激活状态。按键面板的反馈通常由按键下方的 **LED** 灯来表示，而触摸屏的反馈有多种表示方法，一般通过改变边框和文本颜色以及模拟 **3D** 外观的变化来表示。

### 可视化的按钮反馈

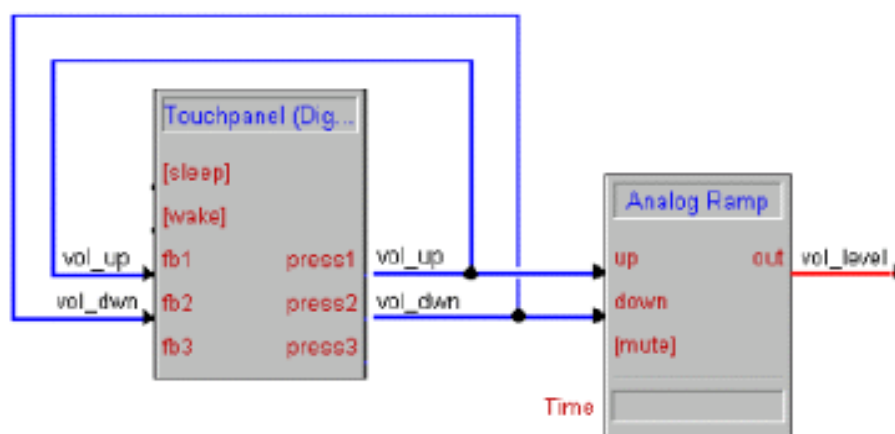


对于一个好的界面设计而言，反馈十分重要。一方面，反馈让使用者了解按钮动作已经传递到系统。这对触摸屏尤其重要，因为通过触摸屏用户不能够感觉到是否按到了正确的位置。另一方面，让用户了解系统当前的状态信息（如 VCR 当前在播放状态）。因此反馈要尽可能准确，因为不准确的或模糊的反馈会让用户感到疑惑。

按钮是否按下取决于用户（确切说是取决于用户的手指），但按钮是否显示反馈状态是由信号决定的。因此，信号是按钮反馈的驱动源。信号的来源取决于需要何种类型的反馈。最基本的反馈类型叫瞬态反馈。只有按钮本身被按下时，瞬态反馈的按钮才显示它的反馈状态。这类反馈适用于只有当按钮按下时才执行的功能。例如：音量高低按钮通常接收瞬态的反馈，因为音量大小只有在相应的按钮按下时才改变（增大或减小）。

瞬态反馈只需将按钮发出的信号名称赋给该按钮的反馈信号就可以实现。如下图所示

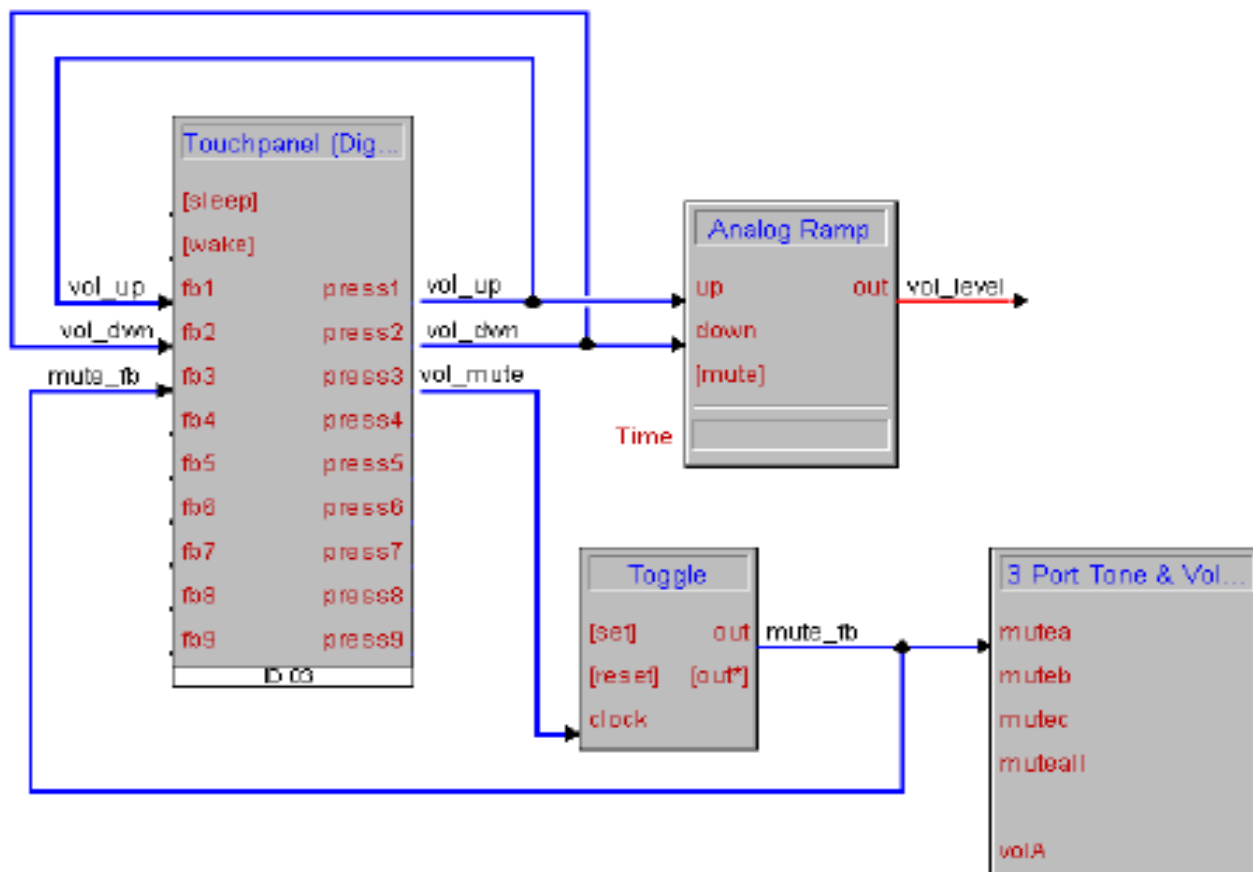
音量即时反馈



某些按钮可能需要更加复杂的反馈。例如：静音按钮交替打开或关闭静音功能。为了表示这一反馈，当静音功能打开时，按钮应该显示反馈状态；而当静音功能关闭时则需取消反馈显示。很明显，瞬态反馈不能实现该功能，所以我们必须用逻辑函数来产生期望的效果。此例说明如何用 **Toggle** 函数来实现这一功能。

静音——Toggle 函数反馈





当控制系统和受控设备之间有双向通讯时，有可能存在更精确的反馈形式。有时可能需要基于从受控设备收到的信息来显示反馈（注意：该信息也叫做反馈，不要与按键反馈混淆）。例如：假设您通过控制一台矩阵来选择四路视频源中的某一路，如果矩阵自身返回信息给控制系统指明当前选择的是哪一路视频源，您的程序可以利用这一信息突出显示表示该路被选视频源的按钮。这样，即使用户不通过控制系统而直接在矩阵上切换，也能保证反馈的正确性。

### 子页（仅用于触摸屏）

子页是触摸屏才有的功能强大的对象。子页与标准触摸屏页面有许多相似之处：都包含有按钮、文本框、图片等等。然而，子页通常不会满屏显示。相反，子页通常定义一小块包含许多按钮的区域来完成特定的功能，比如：VCR 的控制按钮。因此，一个特定的子页可以象 Windows 或 Macintosh 计算机上的对话框一样，任何时候均可以在标准页面上显示或消失。

为了让子页尽可能灵活，我们通过触摸屏反馈信号来控制它们的显示或隐藏。即当需要子页时，在子页将出现的地方创建一个子页引用。子页引用只是到之前建立的子页对象的一个链接。然后为每个子页引用指定一个 Join Number，该 Join Number 的反馈信号将决定子页是否显示。只要信号为高，子页就显示；一旦信号变为低就消失。下面例子说明如何通过一个单独的按钮控制音量子页的逻辑，允许子页根据需要用 Toggle 打开或关闭。

### 模拟显示（仅用于触摸屏）

VisionTools Pro-e 提供一系列针对不同模拟显示的对象。例如：您想创建一个条形图，您可以进入 Pro-e 工具栏，选择 Gauge（条形图）对象并调整到需要的尺寸，然后为表示模拟通道值的这个条形图指定一个模拟反馈 Join Number。

VisionTools Pro-e 提供了几种模拟对象：

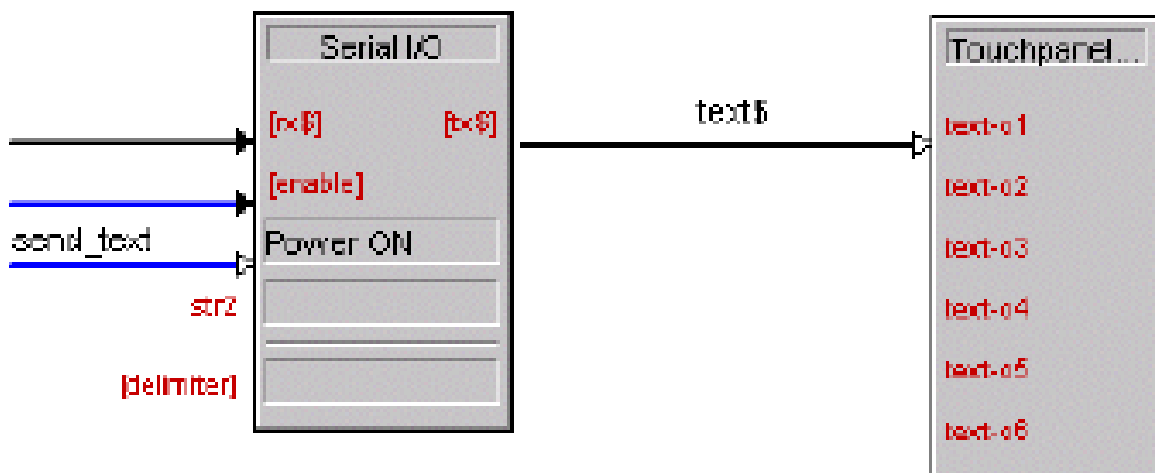
- 1、Gauge（条形图）
- 2、滑块（通过触摸屏可活动的条形图）
- 3、数字标尺（用数字格式显示一个模拟信号）
- 4、百分比（用%显示模拟量）
- 5、时间（用时间格式显示模拟量 HH: MM: SS）

### 间接文本（仅用于触摸屏）

间接文本是在触摸屏按钮上显示的特定的可以根据信号状态改变的文本字符串。例如：当用户按下按钮时它可能显示“电源开”，当再次按下时可能显示“电源关”。

在 SIMPL Windows 中，触摸屏已经定义了用于间接文本区域显示的串量反馈信号。这些文本区域直接接受串量数据信号。

发送到触摸屏文本框的串量





---

## 用 SIMPL Windows 来创建一个程序

### 编程步骤

#### 确定受控设备

编程人员应准备一个列明所有受控设备的清单文档。

#### 确定设备的受控方式

了解各种设备如何受控非常重要：这会让编程人员知道哪种控制设备（网络，模块，控制卡或其他）将用于控制一个设备。

比如：红外受控设备需求一个 C2-IR8 插卡。

#### 在 SIMPL Windows 中配置系统

通过配置管理器来配置系统。在设备库中选择控制系统，将控制系统拖曳到系统浏览窗中。通过添加界面，网络模块、控制卡和其他设备来完成系统配置。全部所需的快思聪硬件都应包含在配置中。

#### 在 SIMPL Windows 中编写系统

添加完所需的快思聪硬件完成系统后，便可在编程管理器中开始系统编程。为从触摸屏和其他用户界面中的每个按钮编写功能。首先为从用户界面上给输出的信号命名，然后从函数库（Symbol Library）中选择编程需要的函数，并将其拖进程序浏览窗（Program View），最后在详细浏览窗（Detail View）中给函数输入输出命名。

### 基本编程规则

- 1、函数可以是设备函数也可以是逻辑函数
- 2、逻辑函数执行一个操作信号的动作、
- 3、逻辑函数只改变输出信号状态。
- 4、信号链接函数
- 5、数字信号应只有一个信号源，程序中只有一个函数将信号作为输出列出

---

注意：除了一些特例，数字信号都只有一个驱动源，这个特例是按钮及 Buffer 输出。

---

### 建立一个系统

SIMPL Windows 配置管理器允许您：

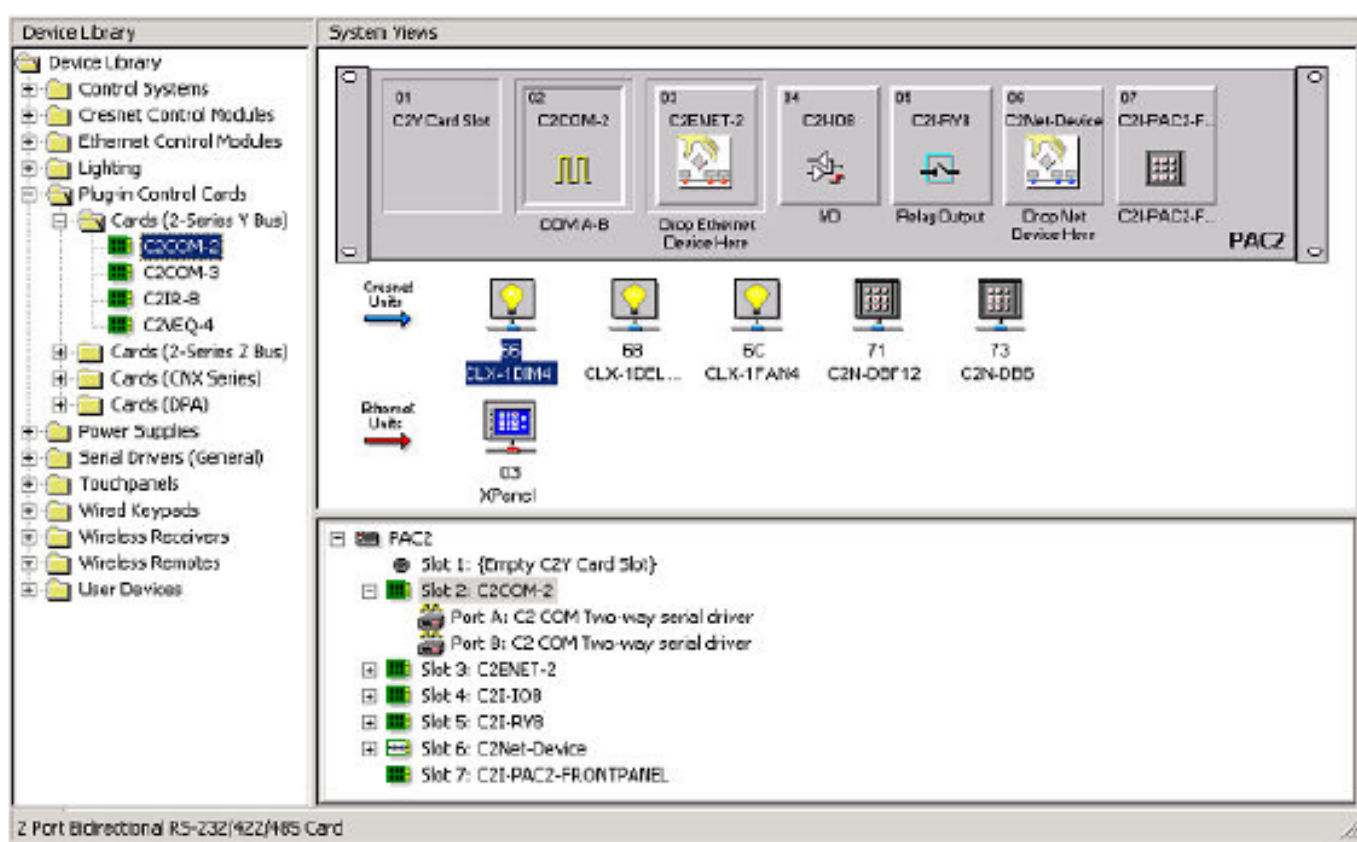
- ◆ 选择控制系统
- ◆ 选择安装过程中要加入的额外快思聪硬件设备，包括控制插卡，网络模块，触摸屏和无线遥控器
- ◆ 选择第三方用户设备。通常是被控设备，包括 CD 播放器，DVD 播放器，有线电视和任何用户通过控制系统界面来控制的设备。

- ◆ 配置受控设备各自连接到哪个控制卡或网络模块上。指定端口、网络地址、IP 信息、配置通信设置、存档系统连接/安装文档。
- ◆ 确定系统所需的 VisionTools Pro-e 项目程序。

配置管理器包含三个窗口：一个是**设备库(Device Library)**，另外两个是**系统浏览框(System Views)**。

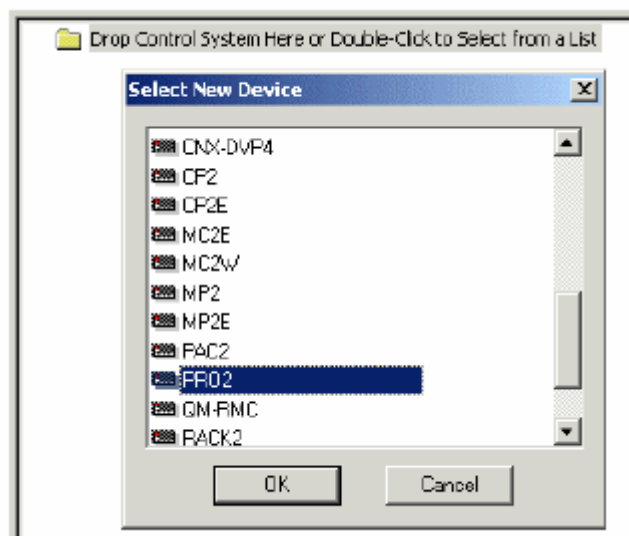
设备库是所有快思聪及第三方硬件设备的树形结构文件夹目录，包括快思聪控制系统，快思聪网络及以太网控制模块，控制插卡，触摸屏和用户设备。可通过打开关闭树形目录中的文件夹来浏览硬件。当选中某个设备，SIMPL Windows 状态栏会显示该设备的描述。

上方的系统浏览框是控制系统和任何您添加到程序中的快思聪网络或以太网设备的图片浏览框。下方的系统浏览窗树形显示控制系统中带有标号的卡槽，您可以打开和关闭它来浏览网络地址和配置每个选项。

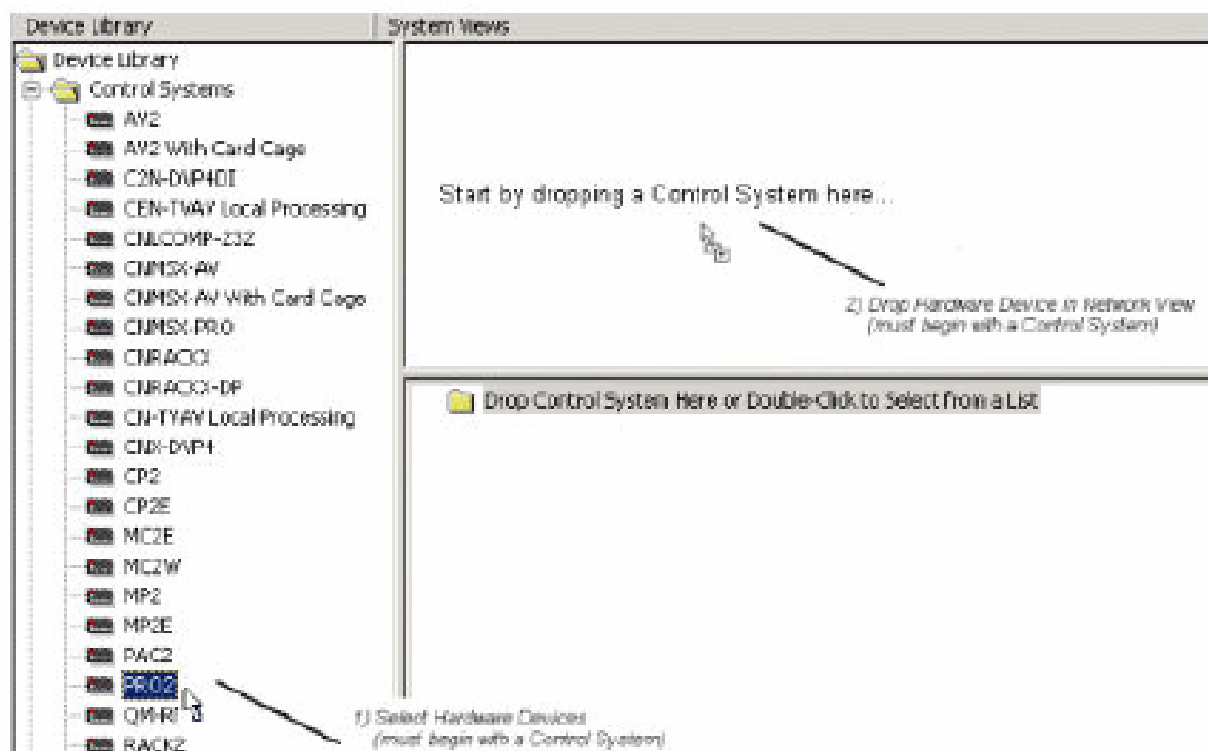


## 控制系统

用 SIMPL Windows 建立系统的第一步是选择控制主机。设备库包含了所有当前快思聪的网络和以太网控制系统，包括 2 系列、QM 系列和 X 系列。您可以通过双击下方系统浏览框口(System View)中的空文件夹，从列表中选择控制系统来快速添加控制系统。



您也可以拖动控制系统到任何一个系统浏览框。



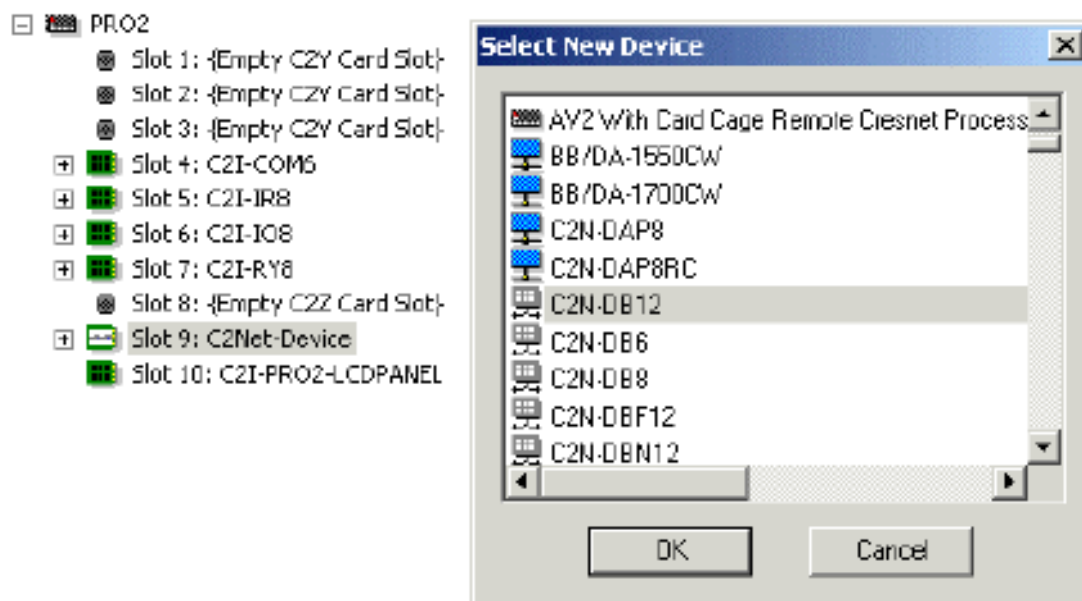
另一个方法是在设备库中右键点击控制系统并从子菜单中选择添加设备选项(Add device to system)。



## 网络硬件

添加好控制系统后，下一步是添加组成系统的其他设备，包括网络控制模块、触摸屏和其他界面、扩展插卡和快思聪数据库中的用户设备。

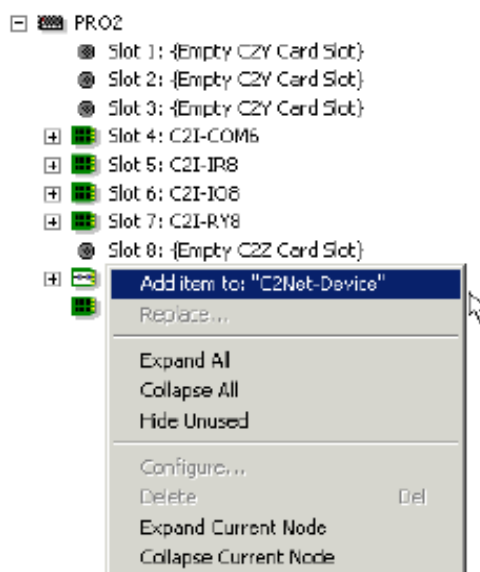
添加快思聪硬件最快捷的方法是从树形图中双击卡槽或网络 ID，打开与对应卡槽匹配的快思聪所有可选设备列表。比如：如果您想添加一个快思聪设备，您可以双击快思聪网络设备槽（Net Device），然后滚动列表添加设备。



快速添加快思聪设备的多个拷贝，可以通过右击设备库中的设备，选择子菜单中的添加多个拷贝（Add multiple copies）选项，输入要添加的设备数量后点击 OK。



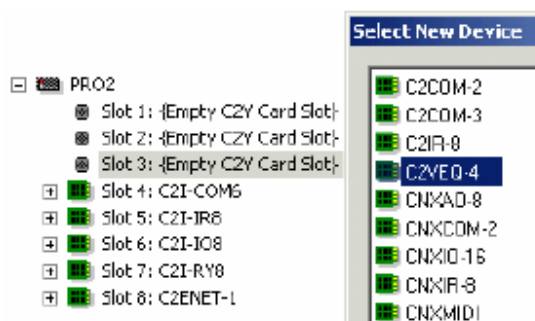
如果您将一个快思聪设备加入控制卡槽，SIMPL Windows 会自动分配网路地址，优先分配出厂默认 ID，之后顺序分配。要在指定的网络地址处添加快思聪设备，只需双击网络地址，打开设备选择清单，从中选择设备加入选定的 ID。另外一个打开设备选择清单的方法是右击卡槽或网络地址，从子菜单中选择加入设备（Add item）选项。



## 控制插卡

快思聪控制插卡是一种可方便安装在主机扩展槽中的电路板。通过扩展卡可以为您提供额外的红外或串行控制端口、增加主机的音频混音功能、将控制系统连入以太网。您可以从设备库的控制插卡文件夹（**Plug-in Control Cards**）中选择扩展卡。

双击主机系统目录树的空卡槽将显示适合这一插槽的所有扩展卡列表，以帮助您快速确定哪个卡适合哪个槽。

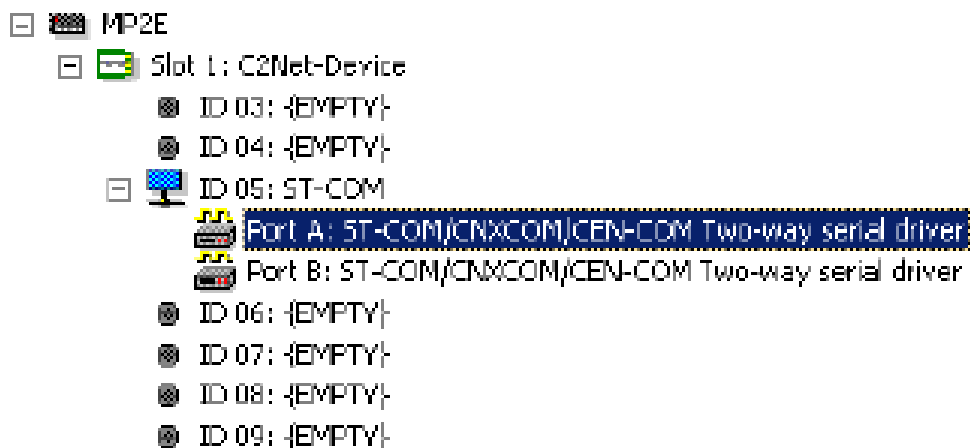


## 串口设备

快思聪主机通过 RS232、422、485 标准接口来控制串口设备。与红外不同，RS-232 控制不使用现成的驱动文件，而设备所需要的数据格式或协议由制造商在设备的说明文档中提供。

大多数快思聪控制系统提供内置的串口卡和红外卡，以支持单向和双向串行控制。另外，快思聪还提供各种用于串口控制的串口扩展卡，红外扩展插卡和独立快思聪网络设备。比如：**C2COM-2** 扩展插卡支持 RS-232/422/485 双向通信，**ST-COM** 是一个同样支持双向通信的独立快思聪网络设备。**C2-IR8** 卡提供了支持单路 RS-232 信号的 8 个红外/串行端口。

通过配置快思聪串口卡内置的串口驱动实现双向通讯。如果知道串口设备的协议，您可以通过配置串口驱动的通讯设置项来“添加”设备。您可以通过在控制系统目录树中展开串口卡或者串口设备来浏览串口。



双击串口驱动将显示设备设置。点击串口设置（Serial Settings）标签，根据制造商说明配置设备。这将在配置设备章节中详细讲述。

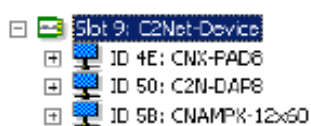
## 用户设备

您可以根据不同的制造商或者设备类型在设备库中添加第三方设备。要添加用户设备，右击选中的设备并选择添加设备到系统（Add device to system）。如下所示，也可以选择加入多个设备拷贝。



您也可以用标准 Windows 拖曳的方法将设备拖入卡槽或者在树形浏览图中加入端口地址。

要从树形图中隐藏不用的插槽，右击树形图的任何一个项并从子菜单中选择隐藏未用设备（Hide Unused），这样只显示有连接的快思聪网络设备，使目录树变得简洁。



## 网络 ID

网络 ID 是一个两位十六进制数值，用以标识控制系统网络中的每台快思聪设备。每台可编程的快思聪设备，包括网络控制模块，远端主机，触摸屏，远端和以太网设备，都必须设置唯一网络地址才能与控制系统通讯。设备的 ID 类型取决于通讯格式。

快思聪设备获得一个快思聪 ID（Cresnet ID），也叫网络 ID（NET ID）。SIMPL Windows 中控制系统的网络设备槽（Net-Device Slot）提供了 252 个网络 ID 以连接快思聪设备。当物理网络上连接超过 30 个快思聪网络设备时，要求提供网络扩展设备。网络 ID 的分布从 03~FE。

以太网设备除了需要唯一的 IP 地址或者主机名外，还需要有一个 IPID。SIMPL Windows 中的以太网卡了

提供 252 个 IP ID 用于连接以太网设备。IPID 范围是从 03 到 FE。

无线屏和遥控器必须与快思聪无线网关接收器通讯并且指定一个 IR ID 或 RF ID。可用 ID 数量取决于不同的无线网关。象 CNRFGWX 这样的快思聪双向无线网关提供 15 个 RFID (01~0F) 与无线屏连接，其他网关例如 CNIRFW 提供多达 254 个 IR ID (01~FF)

在 SIMPL Windows 为设备设置的网络 ID 必须与设备硬件出厂内置的 ID 相同。通常需要修改快思聪设备硬件 ID 以使其与软件 ID 相同。硬件 ID 的设置步骤不同设备会有所不同，在每台设备的说明文档中有描述。

当您将快思聪设备添加入程序时，可以设置或者更改其网络 ID。您可以直接将设备拖到网络地址上，也可以将设备从一个地址拖到另外一个地址。

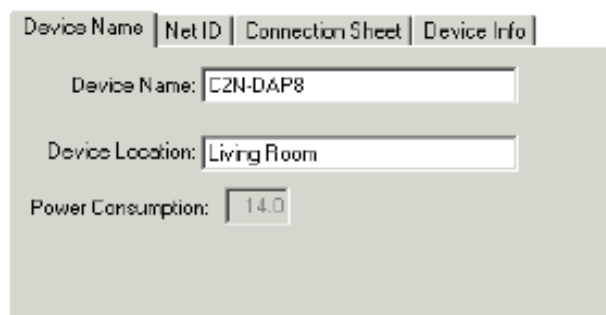
最后，可以通过双击设备打开设备设置菜单，以编辑网络 ID，这将在下一节叙述

## 配置设备

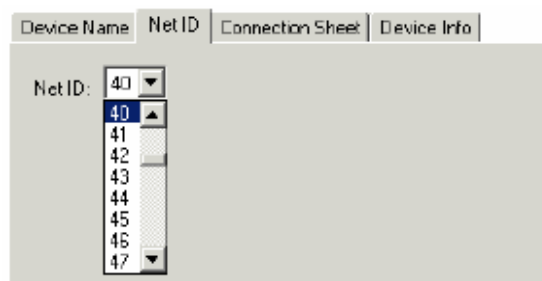
将所有快思聪设备和第三方硬件设备添加入系统后，您可以在系统浏览栏（System Views）中双击设备来配置这些设备。您也可以右击设备选择配置（Configure）选项，打开设备设置对话框。配置选项随设备不同而不同。

### 快思聪网络设备

**设备名称（Device Name）：**您可以用这个标签来改变设备默认名称并输入地址。新名称会在系统浏览框和程序浏览框中显示，同样也会显示在设备函数的标题栏上。设备名称及地址将包含到地址报告中。为所有网络设备提供描述性的文字和指定地址有助于调试。通过快思聪网络供电的设备会显示消耗的瓦数。

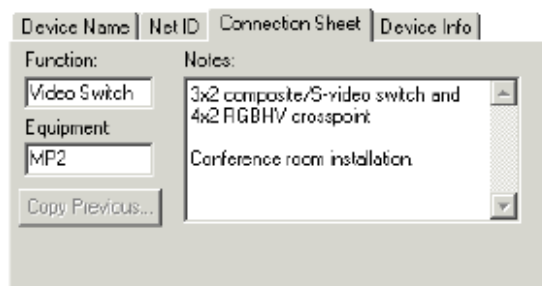


**网络 ID（Net ID）：**您可以通过列表选择来改变设备的网络 ID。



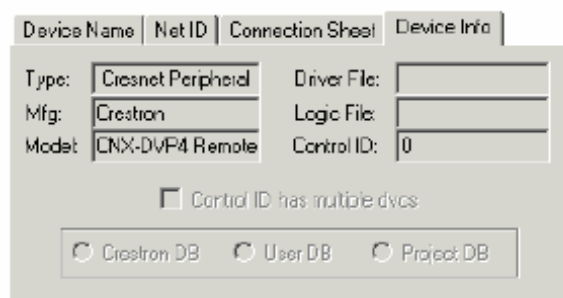
**连接表（Connection Sheet）：**您可以用这些自由文本区来输入关于已连接设备、已安装设备、功能，和其他想相关信息。该信息会包含进 SIMPL Windows 报告中：如连接表，设备地址或者添加的其他信息。





The screenshot shows the 'Device Info' tab of a software window. It has four tabs: 'Device Name', 'Net ID', 'Connection Sheet', and 'Device Info'. The 'Device Info' tab is active. It contains two main sections: 'Function:' and 'Notes:'. The 'Function:' section has a text box with 'Video Switch' and a 'Copy Previous...' button. The 'Notes:' section has a text box with '3x2 composite/S-video switch and 4x2 RGBHV crosspoint' and 'Conference room installation'. There is also an 'Equipment' section with a text box containing 'MP2'.

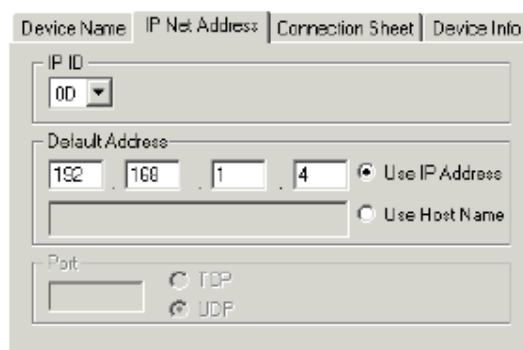
**设备信息（Device Info）：**这个标签提供每个设备的只读程序信息。



The screenshot shows the 'Device Info' tab with more detailed information. It includes fields for 'Type:' (Cresnet Peripheral), 'Mfg:' (Crestron), and 'Model:' (CNX-DVP4 Remote). There are also fields for 'Driver File:', 'Logic File:', and 'Control ID:' (0). A checkbox labeled 'Control ID has multiple dyes' is present. At the bottom, there are three radio buttons: 'Crestron DB', 'User DB', and 'Project DB'.

## 以太网设备

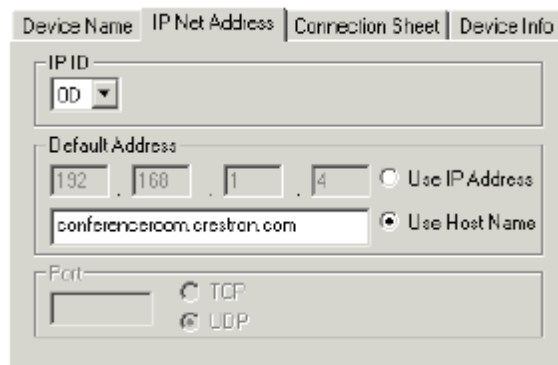
**IP 地址（IP Net Address）：**您可以通过列表选择来改变以太网设备或者以太网远程主机的 IP 地址。除了 IP ID，您还必须指定唯一的 IP 地址。如果设备或者远程主机有一个静态 IP 地址，点击使用 IP 地址（Use IP Address）且在文本框输入。



The screenshot shows the 'IP Net Address' tab. It features an 'IP ID' dropdown menu set to '00'. Below it is a 'Default Address' section with four input boxes containing '192', '168', '1', and '4'. There are two radio buttons: 'Use IP Address' (selected) and 'Use Host Name'. At the bottom, there is a 'Port' section with a text box and two radio buttons: 'TCP' and 'UDP'.

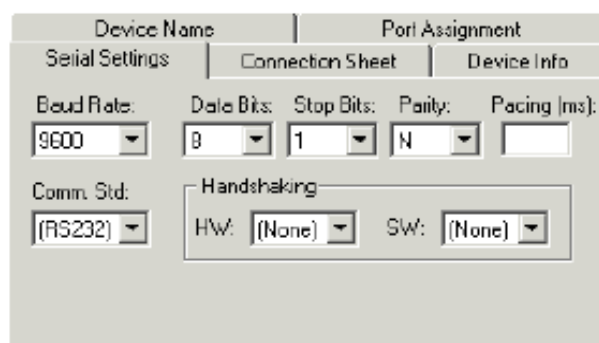
如果设备或远程主机由于设置为支持 DHCP 或系统管理员指定了主机名，那么点击应用主机名（Use Host Name）并输入域名。





## 串口设备

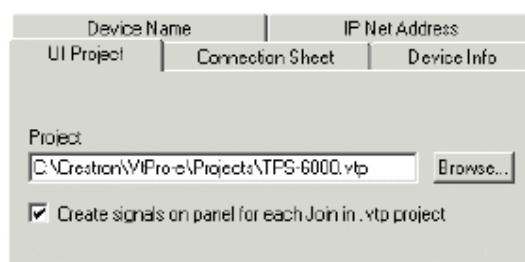
**串口设置 (Serial Settings)：**根据厂商提供的说明设置设备串口协议。设置的协议包含传输标准（如 RS-232）、数据传输速率（波特率）、错误检查（奇偶校验）、数据位、停止位。另外，一些设备可能要求硬件（RTS/CTS）或软件（XON/XOFF）握手协议，以控制两个设备间的数据流。最后，一些设备传输时要求字符间有时序间隔或暂停。



## 触摸屏

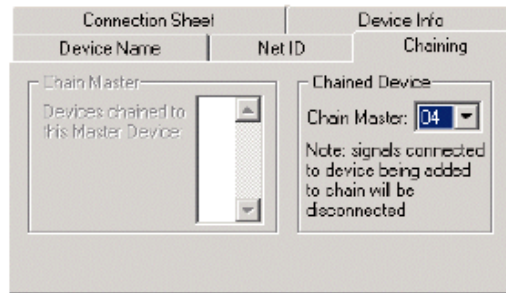
**用户界面 (UI Project)：**在 SIMPL Windows 程序中，每个触摸屏都有对应 VisionTools Pro-e 项目程序，其中为每个触摸屏发送到控制系统或者反馈接收的数字、模拟、串量信号定义了 Join Number。您需要将在 VT Pro-e 中定义的触摸屏 Join Number 映射到 SIMPL Windows 中触摸屏函数的输入输出信号端。您可将触摸屏程序导入 SIMPL 程序中来促进该动作。

要导入触摸屏程序，点击浏览 (Browse) 选定文件。选中创建信号 (Create signals) 复选框，自动根据触摸屏定义数字和模拟信号。串量、语音、网页控制和模拟状态的 Join Number 只能手动定义不能自动导入。信号名称格式为：页名\_按钮文本 (Page name\_button text)，注意此操作不会覆盖、修改触摸屏函数上已有的信号。

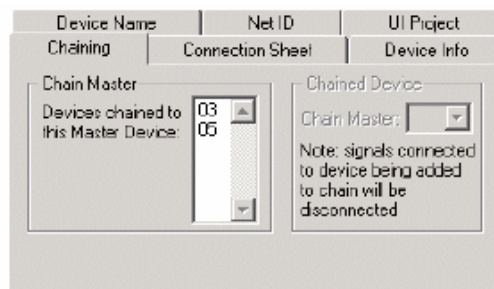


**关联（Chaining）（只用于快思聪网络）：**如果有相同型号的多个触摸屏且功能也相同，您可以将它们关联起来。这样对它们进行统一编程。关联主设备（Chain Master）是决定其它屏的所有参数的触摸屏。要将某触摸屏关联到一个关联主设备上，从下拉菜单上选择关联主设备的网络 ID，在下面例子中，触摸屏被关联到网络 ID 为 04 的关联主设备上，关联设备共享主设备的定义，在上方系统浏览框中该设备将显示被关联（Chained）

图标 。



关联主设备界面会自动显示与之相关联的所有触摸屏的网络 ID。下例中，关联主设备关联有两个 ID 为 03 和 05 的触摸屏。如下所示，被关联设备将丢失之前定义的所有信号。



## 连接信号

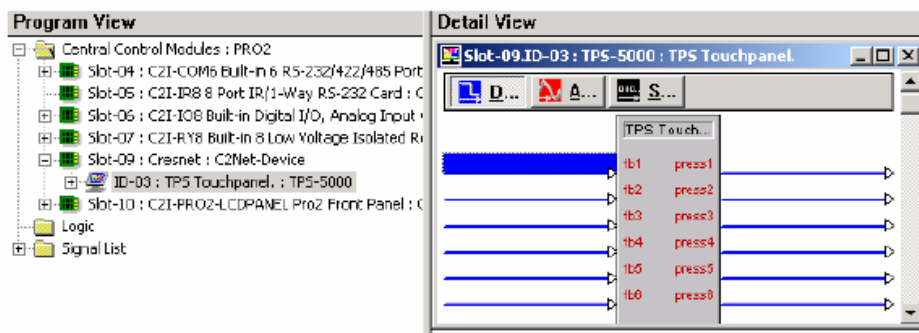
通过配置管理器（Configuration Manager）中加入所有必须的快思聪硬件建立好系统后，将在编程管理器（Programming Manager）中对系统进行编程。为系统中触摸屏或其他用户界面的每个按钮的功能编程。首先为用户界面输出的信号命名（在配置管理器中对触摸屏配置时，如果导入了 VT Pro—e 项目程序，那相应的输出信号会有一个默认名称）

## 定义用户界面信号

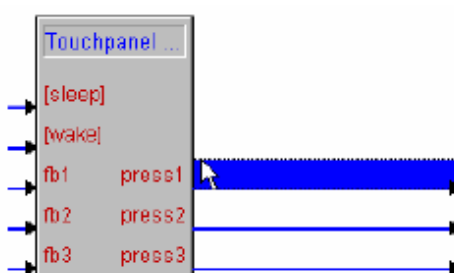
从用户界面开始编程通常是最方便的。在程序中，按钮按下（无论从触摸屏，有线或无线按钮面板，键盘或其他界面）就关联有信号量。当一个按钮被按下，在控制系统程序中对应的信号有效，当按钮松开，对应信号失效。

按按钮是用户为一个特定需求发出的动作。因此，必须为用户界面的输出信号命名或“定义”。这样每个信号都能被正确识别和路由。

对用户界面编程，在程序视图（Program View）中打开相关的槽（Slot）或文件夹，将界面拖入详细视图（Detail View）或双击打开，在详细视图中就会显示触摸屏的定义。



点击要命名的每一个信号。应该为每个信号量赋予一个描述性的命名，例如 Power\_On, Screen-up, 或 DVD\_play。



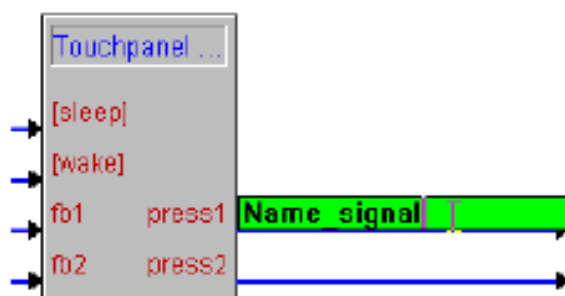
只有选中或突出显示信号时，才可以输入文本。默认的，当信号第一次显示时，顶部的输入信号（左上）会自动选中，可以通过鼠标或键盘按键选择函数的信号。

---

**注意：**函数标有[ ]的信号量是可选信号。它们不影响函数的正常工作，可以不定义。

---

当信号高亮显示时，您可以输入信号名称。

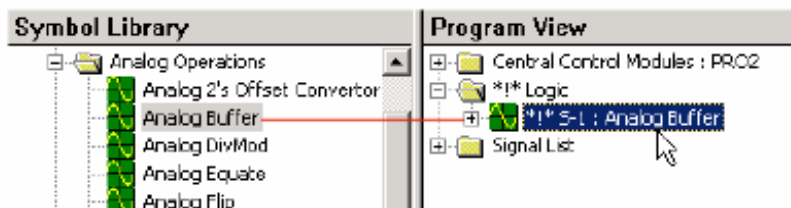


## 使用逻辑函数

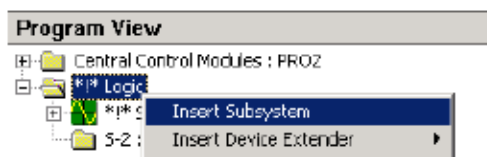
进行 SIMPL Windows 编程时，编程人员通常会使用很多函数。对于同一个控制问题，逻辑函数提供多种不同的解决方法，为编程人员提供无限的灵活性。逻辑函数包括从最基本 AND, OR, NOT 函数到特定功能应用的函数。

您可以通过在函数库中选择函数、拖入程序视图，将函数添加入程序。注意，如果您将函数直接拉入详细

视图，该函数将不放在 Logic 文件夹中。



SIMPL Windows 允许您创建子系统——在逻辑 Logic 文件夹下的子文件夹。您可以根据逻辑功能来组织函数。要创建子系统，选中逻辑 Logic 文件夹、从右键中选择插入子系统（Insert Subsystem）项。



当创建一个新文件夹，您可以右键为其指定一个有含义的名称。可以将函数从函数库直接拖入新建文件夹。

## 第三章 逻辑函数编程

### 概述

上一章我们学习了如何运用设备函数和通过信号连接的简单编程。这类程序用的是“直接功能”，也就是说，按下按钮（或其它任何的系统输入）直接关联控制功能。如通过 IR 驱动的“PLAY”指令，当按下按钮，“PLAY”指令就通过红外端口发出，当释放按钮命令就停止。这是非常简单的编程，但当控制系统变得复杂时，您会发现您需要远比程序自身更多的控制，这就需要用到逻辑函数来进行处理。

一般来讲，每个逻辑函数都可以看作是一个特定的处理器。逻辑函数会检测连接到其输入端的信号状态，产生适当的信号值输出到其输出端。输出信号值由逻辑函数本身的特性以及函数参数值决定。

### 逻辑函数的类型

目前有 100 多个逻辑函数用于 SIMPL 编程，这些函数包括从简单的数字逻辑门到复杂的串量生成器。为了方便查找和某些应用的需要，整个逻辑函数集归类如下：

**Analog Operations**——用于如灯光的调节和音量预设的模拟功能函数。

**Conditional**——根据条件的成立与否确定其输出的逻辑门函数。

**Counters**——二进制和十进制计数器函数。

**Debugging**——用于运行时追踪信号的函数或使信号生成模拟量、数字量或串量的函数。

**Device Interface**——鼠标和键盘模拟器函数。

**e-Control Software**——用于 CRESTRON E-mail 强大应用功能函数。

**Memory**——读写 NVRAM，用于断电存储数据的函数。

**Program Formatting**——有助于分类和排列程序信息的函数。

**Sequencing Operations**——按照顺序决定输出的函数。

**Serial**——生成、分析和处理串量的函数。

**System Control**——虚拟逻辑主机与远程系统通信并且向控制台直接发送数据的函数。

**Time/Date**——处理关于时间的数据，包括天文时钟，日历和主机时钟的函数。

**Timers**——在设定时间或一定延迟时间让预设状态触发相应功能的函数。

**Touch-panel Interface**——广播串量到网络触摸屏的函数。

本章后续部分将举例说明常用函数的用法。读完本章后，您可以参考 SIMPL Windows 的帮助文件，在帮助文件中有 SIMPL 语言每个逻辑函数的单独帮助说明。

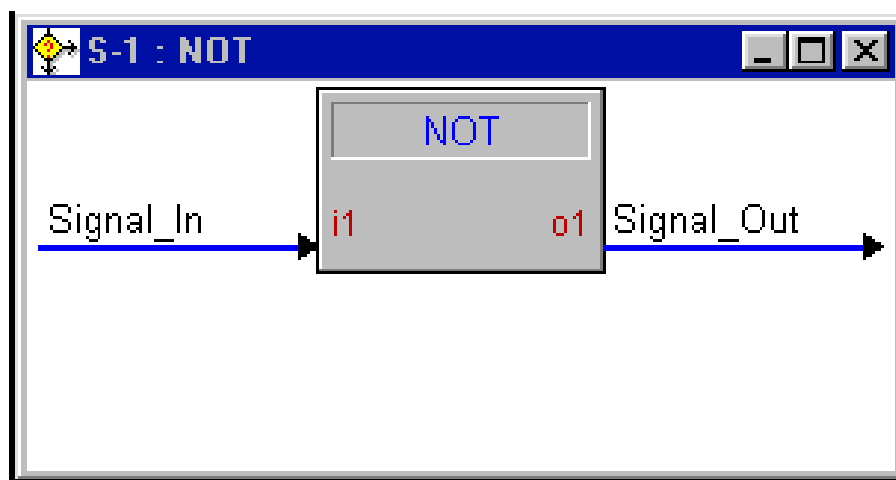
## 基本逻辑

基本逻辑在这里是指 SIMPL 语言中最基础的逻辑，如果您有关于数字逻辑的工作经验，那么您会觉得与这些函数非常的相似。即使您没有数字逻辑的知识，本节可以教会您这些编写 SIMPL 程序的逻辑。

### NOT 函数

NOT 函数也称反转器，反转输入信号到输出端，如果输入是高电平，输出就是低电平，反之亦然。

#### NOT 函数



我们用一个真值表来说明其输入和输出的逻辑关系，表示如下：

Signal In	Signal Out
Low	High
High	Low

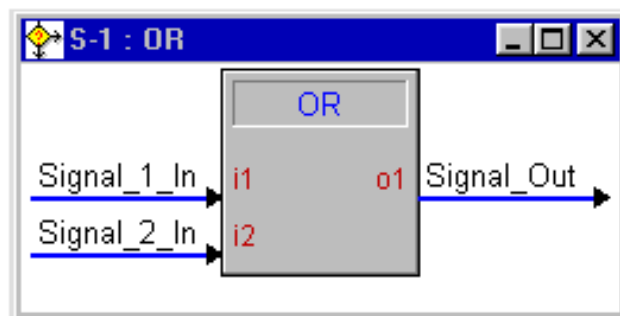
#### NOT 函数应用举例：自动摄像控制

有时设备提供的信号与您想要的恰好相反，比如就您的控制系统而言，麦克风混音器通过设置触点闭合可以知道哪一路麦克风正在传入声音。在一个视频会议应用中，这些闭合也可以用来控制摄像头使其直接对准发言人。但是，假设当麦克风有人发言时不是提供的闭合信号，而继电器平时是闭合的，但麦克风有人发言时提供一个断开的信号触发。您可以把每个输入信号接入 NOT 函数，首先使电平反转输出，用输出信号在您逻辑程序中其它地方去触发摄像机的预设状态。

### OR 函数

当任何一个输入为高电平时，OR 函数将输出高电平，看下面的图表，当“Signal\_1 in”或“Signal\_2 in”为高电平时，“Signal out”为高电平。

#### OR 函数



两个输入的 OR 函数真值表如下：

Signal In 1	Signal In 2	Signal Out
Low	Low	Low
High	Low	High
Low	High	High
High	High	High

注意 1：从真值表中可以看出，当两路输入信号都为高电平时，输出也为高电平。如果您需要在只有一路输入为高电平时，输出才为高电平，您可以用一个异或（XOR）函数。

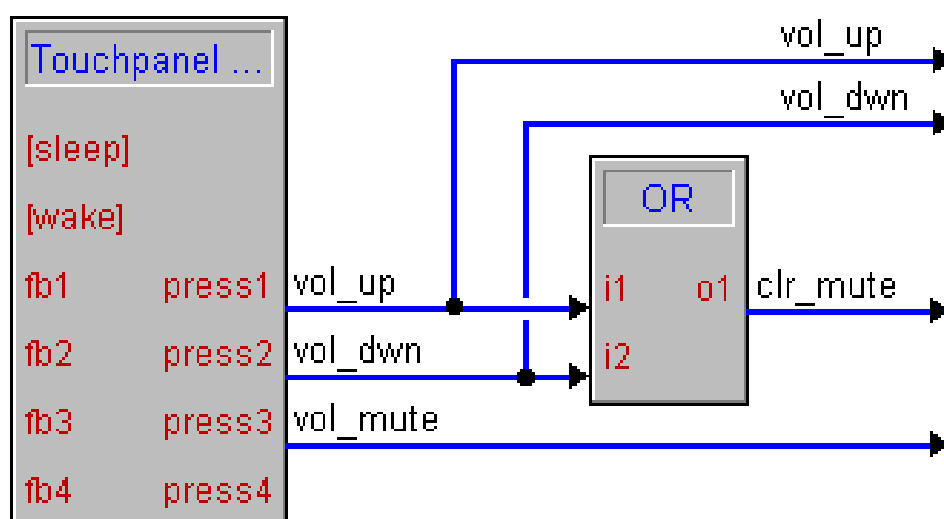
注意 2：OR 函数的输入端口数是可以随意增加的（取决于特定的应用），并且只有一路信号输出。就是说，一个 20-Input 的 OR 函数的任何一个输入端为高电平，输出都为高电平。

### OR 函数举例：解除静音控制

比如，典型的音量控制包括调高、调低和静音按钮。如果您想在静音时通过按调高或调低按钮就自动解除静音，您可以用一个 OR 函数完成此项功能。

注意：要完成该例的功能还需要另外的逻辑，这在后面的章节中作介绍。

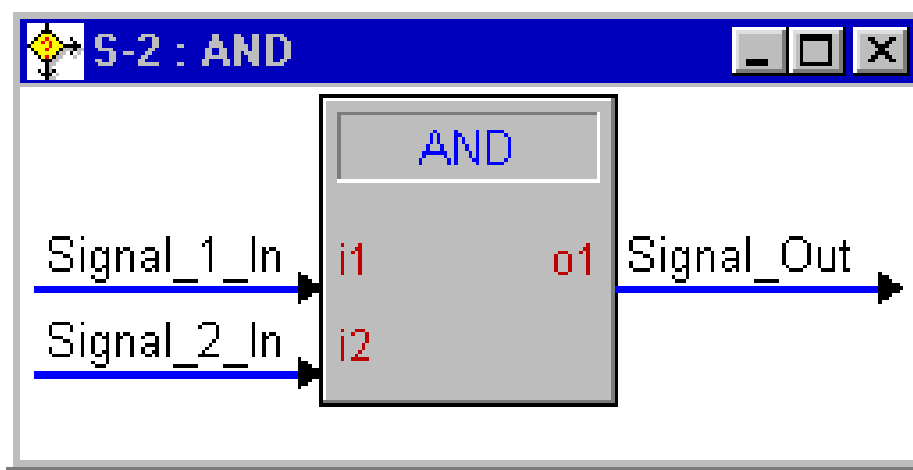
### OR 函数例子



## AND 函数

AND 函数在所有的输入都为高电平时输出为高电平。如下面的图表所示，如果“Signal\_1\_in”和“Signal\_2\_in”都为高电平时，“Signal\_out”为高电平。

### AND 函数



AND 函数的真值表如下：

SignalIn 1	Signal In 2	Signal Out
Low	Low	Low
High	Low	Low
Low	High	Low
High	High	High

注意：同 OR 函数相似，AND 函数有一个可以任意增加的输入端，但只有一个输出端。

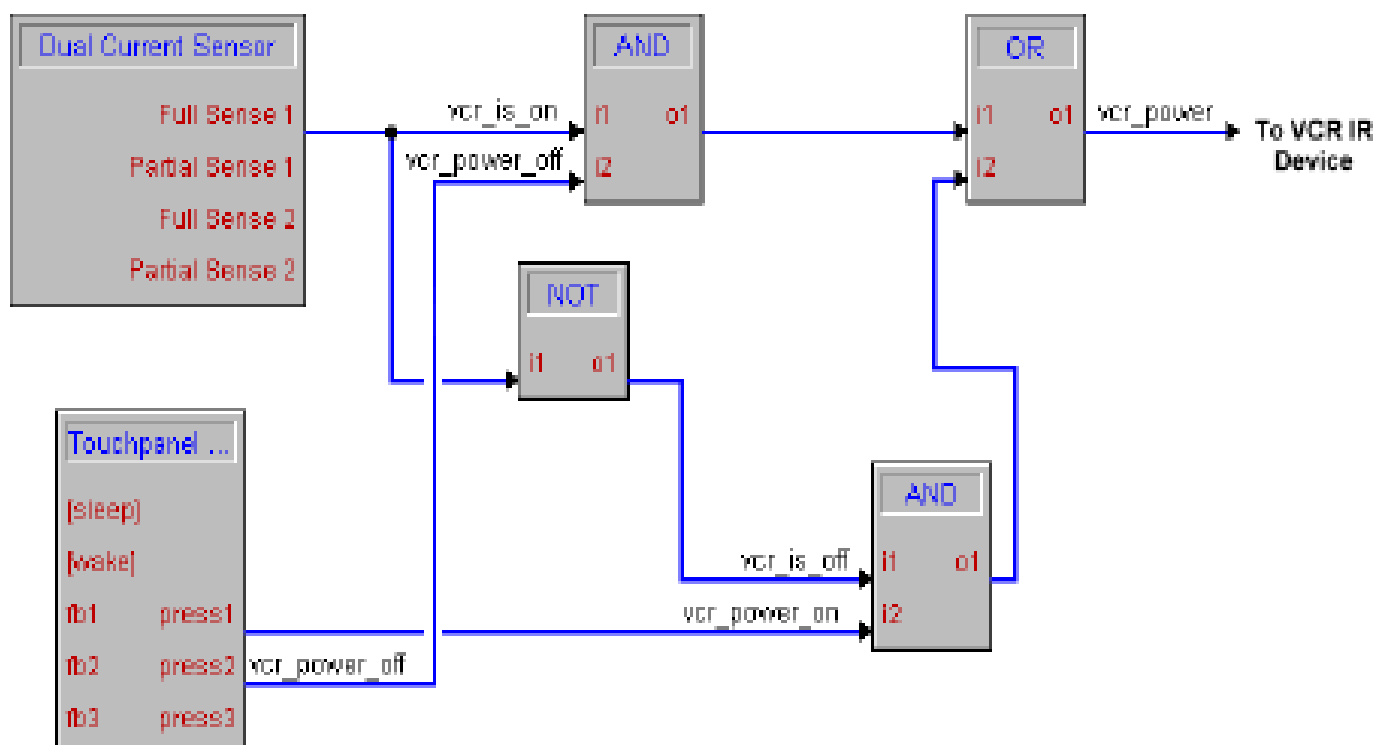
### AND 函数举例：分离电源开关

因为只有所有的输入端为高电平的时候，输出端才为高电平，一般不会将多个按钮的触发直接关联到 AND 函数的输入端，这意味着两个按钮要同时被按下才行。接到 AND 函数的一个或多个输入通常是锁定的电平信号，经常用来确定系统当前的某种状态。

比如，由红外控制的 VCR 只有一个电源功能键（on/off），假设我们想将电源开和电源关的功能分开。通过电流感应设备检测 VCR 电源开关的状态，产生一个数字信号代表 VCR 当前是开或关。当信号“vcr\_is\_on”为高电平时，表示 VCR 开。用 AND 函数我们可以确保当 VCR 为开时，按下“vcr\_is\_off”按钮发送电源（关机）指令到 VCR；当 VCR 是关时，“power\_off”按钮按下时，我们不希望发送电源指令，因为这样可能使 VCR 又打开。我们用同样的逻辑去处理“vcr\_power\_on”按钮按下时的情况。见下面 AND 函数应用于完整电流感应程序的例子。

### AND 函数举例



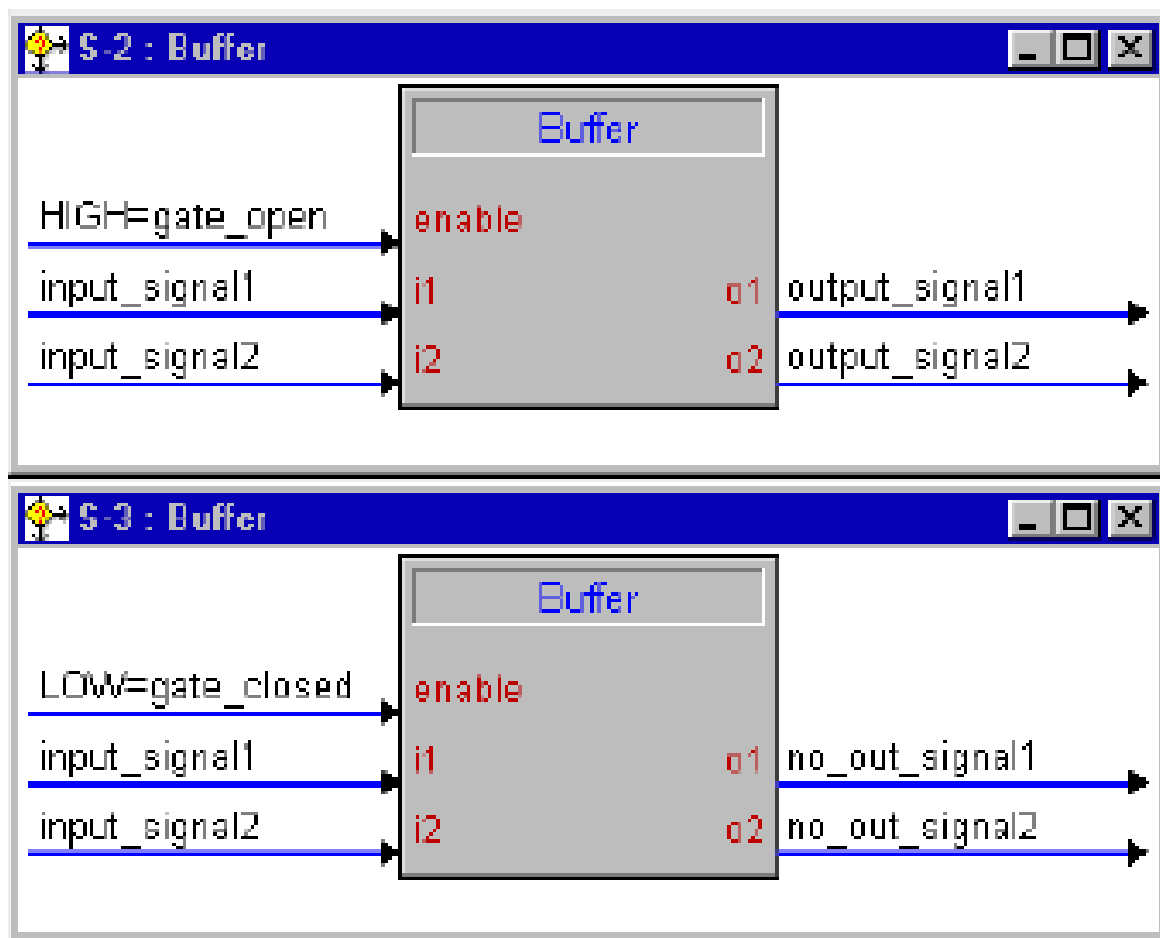


## Buffer 函数

Buffer 函数可以理解为可开关的门，控制数据流通。当门开时，数据信号不做任何改变从输入端流向对应的输出端；当门关闭时，所有的输出被置低电平，并且与输入的信号无关。

“门”的开关是由 **enable** 输入端所控制的，当 **enable** 为高电平时，Buffer 使能（即门开），当信号为低电平时，函数不可用（即门关）。

### Buffer 函数—High/Low



**Buffer** 与之前讨论的函数不同，它的输入端数量不仅可以扩展（**enable** 除外），而且每个输入有一个对应的输出，这与 **NOT**、**OR** 和 **AND** 函数只有一个输出端不同。而且 **Buffer** 每一对的输入输出与其它输入输出相互独立。也就是说，一个输入端的信号只与对应的输出端有关系（当函数被使能时），不会影响其它输出端。因为 **Buffer** 有时相当于复合的 **AND** 函数，每个输入端与 **enable** 端 **AND** 后决定对应输出端的状态。

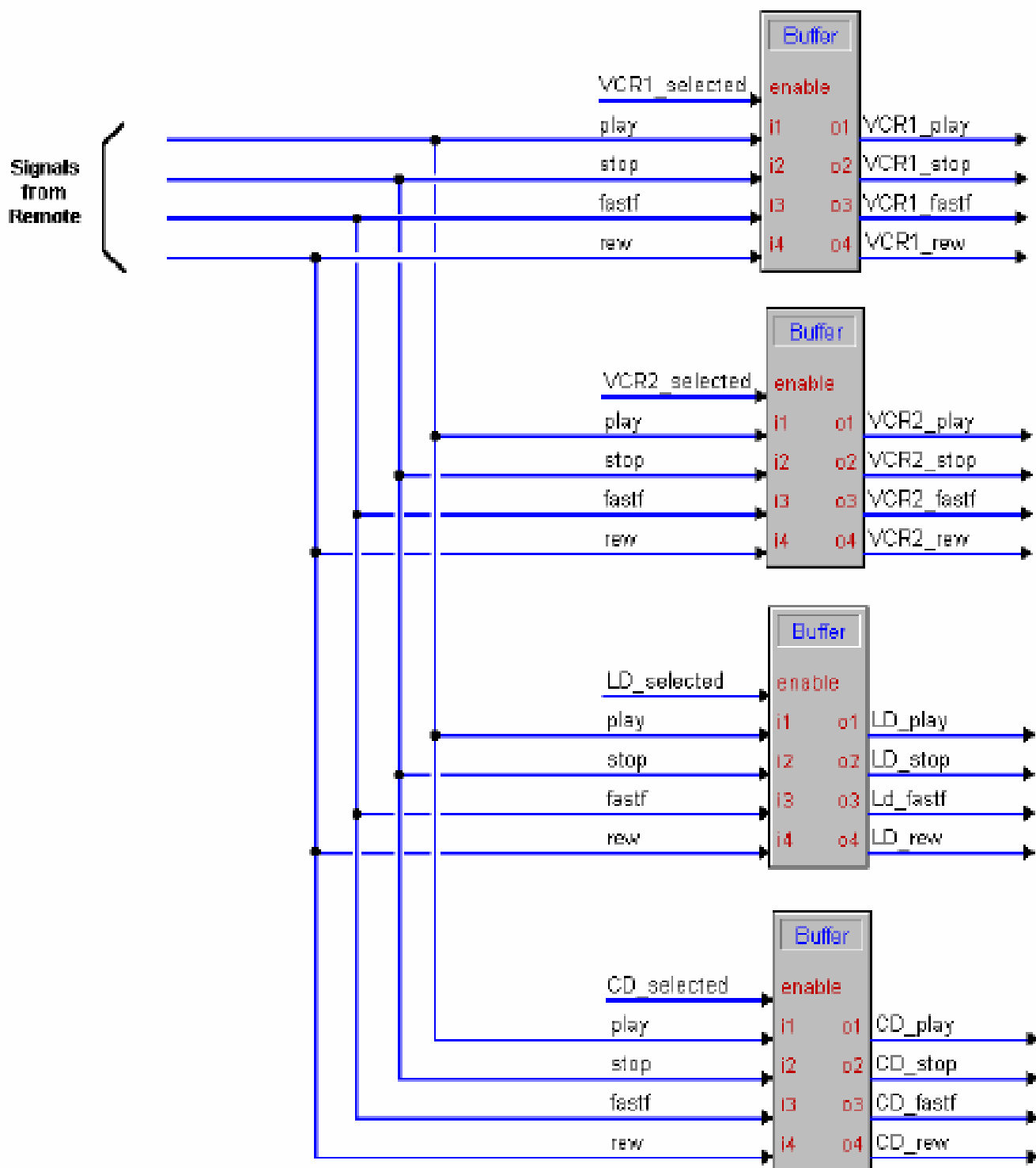
**Buffer** 函数的输出端信号有一个有趣且非常实用的特点。前面我们讲到了数字信号时提到每个数字端只可有一个驱动源，但是这里有一些例外。我们知道象按钮输入这样的系统输入是一个例外，**Buffer** 函数的输出是另一个例外。即一个信号不仅可以由一个 **Buffer** 驱动同时也可以被其它 **Buffer** 驱动或者由一个按钮（或其它系统输入）驱动。这个特点有一个深层次的逻辑关系，这将在手册后面详细讲解。

### Buffer 举例：多设备控制

对于 **Buffer** 函数一个典型的应用就是可以用一组共用的按钮实现对多个设备的控制。当对按钮有限的手持遥控设备编程时，这一用法非常有用。针对这样的设备，通常的界面是定义一组共用按钮和不同的设备源按钮，通过选择源按钮决定共用按钮控制某台设备。

因为 **Buffer** 函数相当于复合的 **AND** 函数，我们经常需要产生“状态”信号。就是定义系统中某一事件状态的信号。在本例中，我们需要为每个源设备设定信号来确定某台设备是否被选中。在接下来的章节里，我们将会了解到如何产生这些信号，但现在我们简单的假设我们已有这样的信号，请看下面例程。

### Buffer 举例



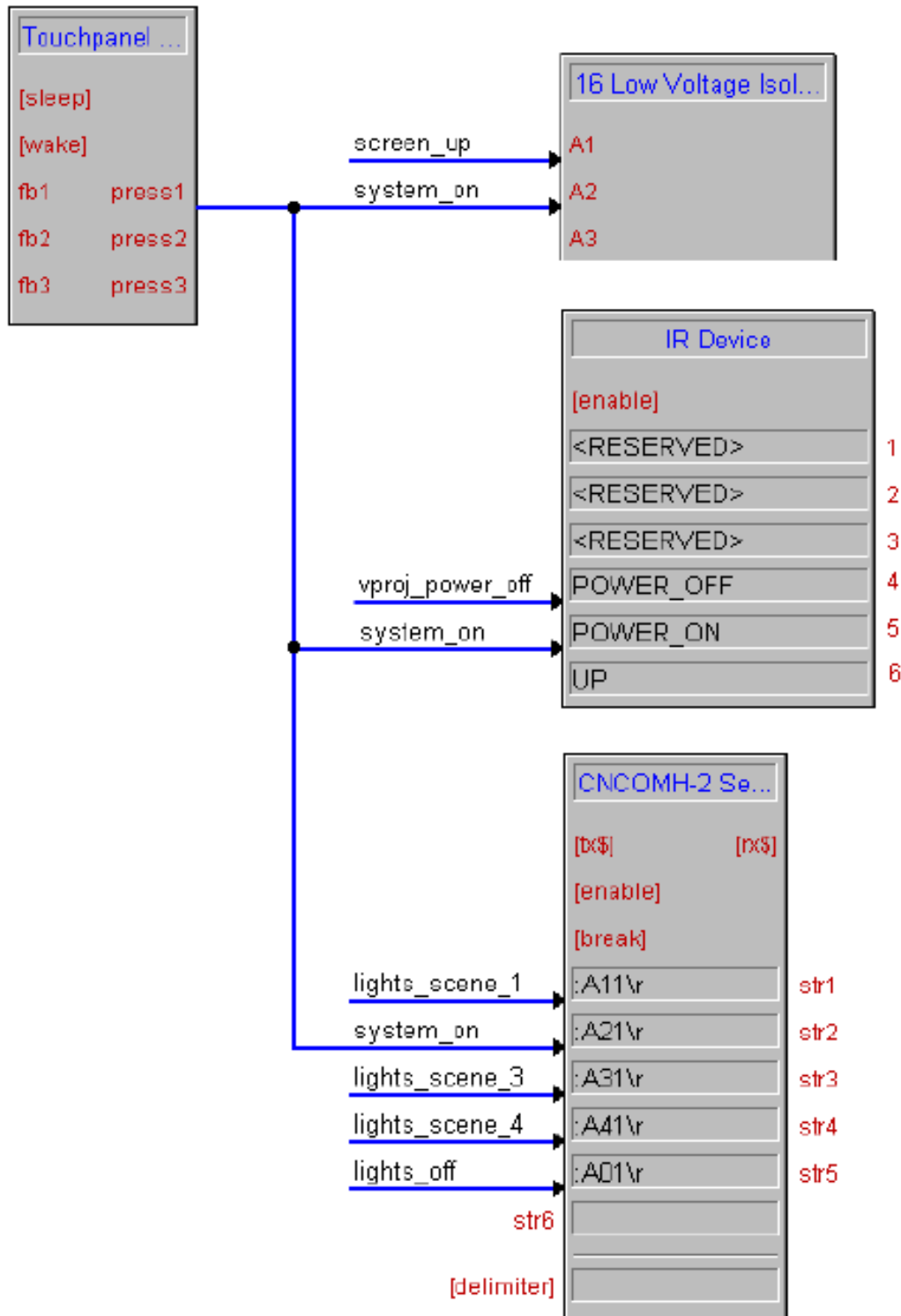
注意，在这个例子中，任一时间只能有一个 **Buffer** 是有效的。如果两个 **Buffer** 同时有效，按 **PLAY** 按钮就会发出多个命令，这是我们在例子中不需要的。为保证不会发生这种情况，我们强制在同一时刻只有一个状态信号为高电平。这在后面的章节将作介绍。

### Buffer 函数举例：多事件触发

对于像快思聪这样可定制程序的控制系统，其优势在于可以为用户提供自动功能以满足他们的真正需求。一个设计良好的控制系统会让用户通过尽可能少的操作完成他们想要实现的控制功能。这就需要在许多情况中用一个按钮去触发多个事件。

SIMPL 语言可以很容易的实现一个按钮触发多个事件（或相关的任何事件）。比如：一个标准功能为“System On”的按钮需要设计为同时降下屏幕，打开投影机，并且选择某种预先设置的灯光模式，这可以通过将按钮的输出信号连接到一个继电器去降下屏幕，向投影机的红外接收端口发送“Power On”指令，并且向 RS-232 端口发送一个命令串控制灯光系统。不需要任何的逻辑编程便可完成。下面是程序图：

### Single Button Press Example 单个按钮触发举例

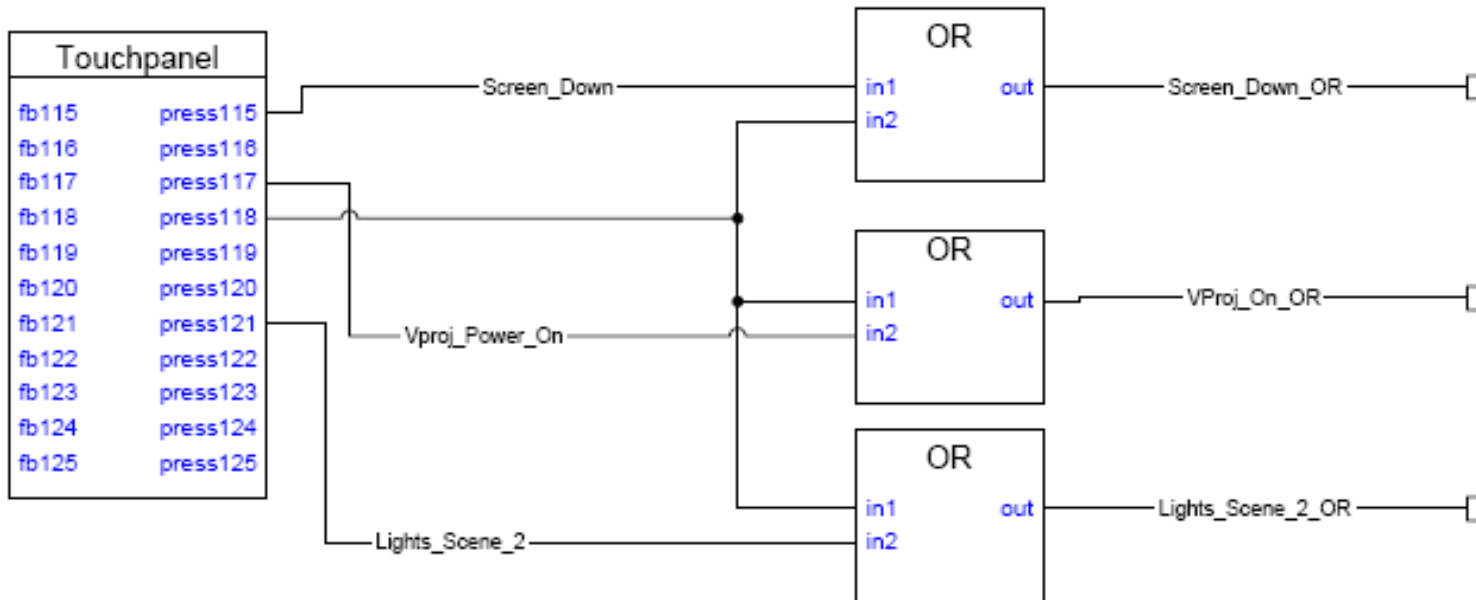


然而，采用这种方法有一些缺点。首先，按这种方式编写的程序很难阅读，您必需全程跟踪信号直到终端才知道它触发了什么事件。采用 SIMPL Windows 的“Show Routings”指令比较容易些。第二个缺点更严重：

如果您想对屏幕、投影机电源和灯光预设做单独控制怎么办呢？就以上的例子中，这三种控制功能被绑在一起而不能独立控制。即使您认为单独控制并不需要，但也许将来会需要，您就需要对之前的程序做大量的修改。

我们可以通过在程序中增加逻辑来避免这种缺陷。第一种解决方法可以用 **OR** 函数将所有的事件联在一起去触发一个事件。比如，我们想通过按“**system on**”按钮或“**screen down**”按钮去降下屏幕，我们可以用一个 **OR** 函数去实现。如下图所示，对于上例中的程序，现在采用 **OR** 函数，除实现连动以外还可实现单独控制。

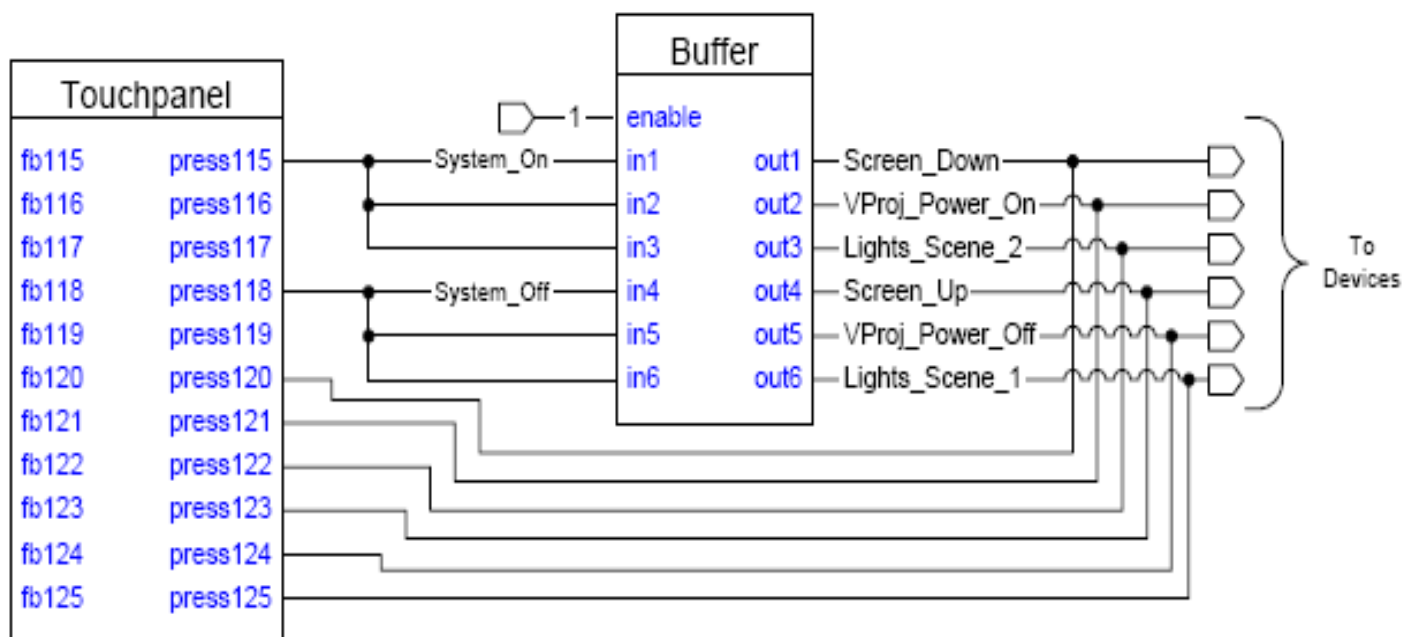
### 多个 OR 函数的分离控制



以上这种方式灵活了许多，但仍有缺点。首先，与之前的例子一样程序流程难以阅读，尤其是程序变得庞大时，对于每一项功能您都需要通过一个 **OR** 函数向后追踪才能知道究竟是什么事件触发了它。第二，程序增大时，可能其它地方也需要去触发某个功能。比如，也许您希望每次选择一个信号源时都降下屏幕，可以简单的增加 **OR** 函数的输入端，但最后会导致程序很乱难以调试。

我们回到 **Buffer** 函数，还记得 **Buffer** 函数的输出信号可以连接到已有的由系统输入或其它 **Buffer** 函数驱动的输出信号。这样我们可以用一个或多个 **Buffer** 函数编写一个优雅的程序去处理多事件触发。下面的程序图只用一个单独的 **Buffer** 函数实现了比上面采用 **OR** 函数例子中更多的功能。

### Buffer



您会发现在上面的例子中有两个有趣的特点。首先，在 **enable** 输入端连接有一个为“1”的信号，之前我们曾讲过这是一个一直为逻辑高电平的数字信号。在这种情况下，**Buffer** 表示永久使能。当您不是用 **Buffer** 去控制信号流向，而是将一个信号“映射”到多个信号时就可以这样使用。第二个独特的特点就是在 **Buffer** 的输入端同一信号被使用了多次，这样便可以通过单独一个输入信号将多个输出信号同时驱动为高电平。最后，注意由于 **Buffer** 对应的输入和输出端是独立的，我们用一个函数可实现系统开和系统关时序。但是，为了清晰起见，我们可以用两个分开的 **Buffer** 函数。

## 状态逻辑

在上一节我们讨论了一些最常用的基本逻辑函数。这节我们将涉及到包含状态信息的常用函数。这些函数在 **SIMPL** 程序中提供了存储的基本形式。需要注意的是这种存储是易失的，存储的信息在程序重启或控制系统断电后会丢失。

本节函数和前一节中所讲函数之间的基本不同是，这些函数输出端信号不是由输入端信号的当前状态来决定的。相反，它们的当前状态表示的是过去发生事件的结果。这也就是存储概念的由来。此外，这些函数的输入端是靠“边沿触发”，也就是说输出信号状态取决于输入信号的跃变（脉冲变化）。大多数情况下，函数是正边沿触发，因为输入端由低到高的跳变会触发输出变化。但某些情况下，输入端也可以负边沿触发，这样输入端由高到低的跳变也同样会影响输出端。而上一节中讨论的函数是“电平触发”，因为总是由函数输入端的当前值决定输出端的状态。

### Set/Reset Latch 函数

该函数实现了最基本的存储原理，同时也叫置位/清除翻转(**Set/Reset Flip-Flop**)。很明显，函数有置位(**Set**)和清除(**Reset**)两种状态。当为置位时，连接到输出端(**Out**)的信号为高电平，当为清除时，输出端(**Out**)为低电平。而输出端 **Out\***与 **Out** 互补。就是当 **Out** 为高，**Out\***就为低，反之亦然。在重启（程序复位）时，

Out 的值为低（Out\*的值为高）。

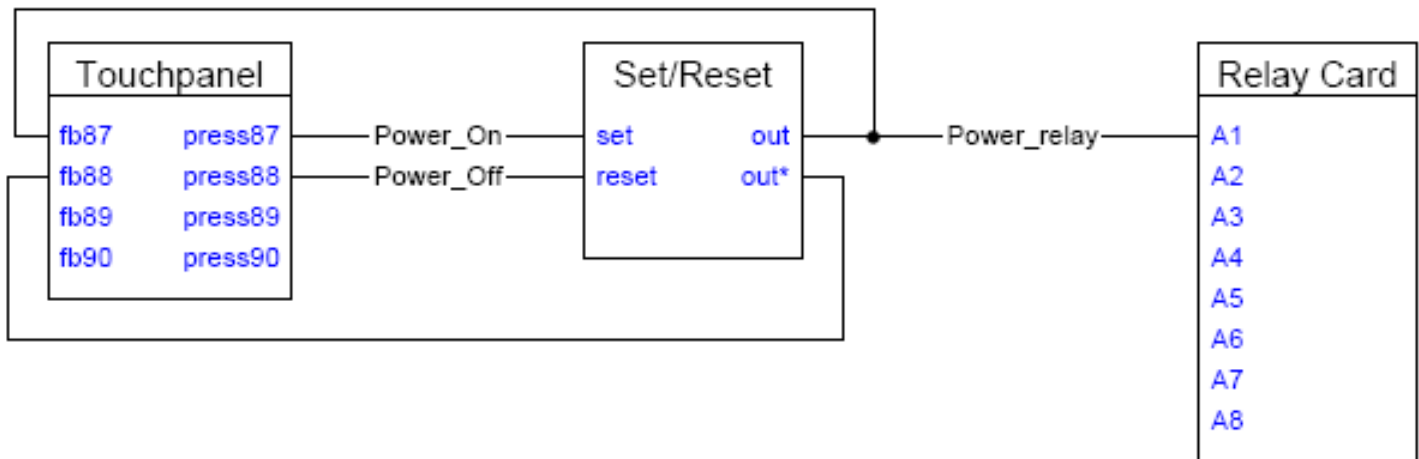
输入端 Set 有上升沿时输出端 Out 变成高电平，一旦在输入端 Set 有上升沿跳变，即使 Set 变为低时输出端也将不再变化。我们称输出端 Out 被锁定。要使输出端 Out 再次变成低电平（Out\*变成高电平），必需在输入端 Reset 有一个上升沿。

### Set/Reset Latch 函数举例：系统电源开关

假设您用交流电源控制器去控制机柜的电源开关。通常这样的电源控制器需要一个低压中间继电器：中间继电器触点闭合电源打开，断开触点电源关闭。这种情况显然需要我们有一个锁定的信号连接到快思聪网络系统中的低压继电器上（如 CNRY-8/16）。

Set/Reset Latch 函数非常适合应用于这种场合。只要简单的将“Power\_On”信号从 Set 输入端，将“Power\_Off”连接到 Reset 输入端，最后将输出端 Out 连接到继电器。当 Set/Reset Latch 函数的 Set 为高，连接到 Out 端的信号也为高，那么继电器将闭合电源开启。记住这一输出信号将一直保持为高直到函数被复位（Reset 端出现上升沿）。

### Set/Reset Latch 举例:系统开关机继电器



## Toggle 函数

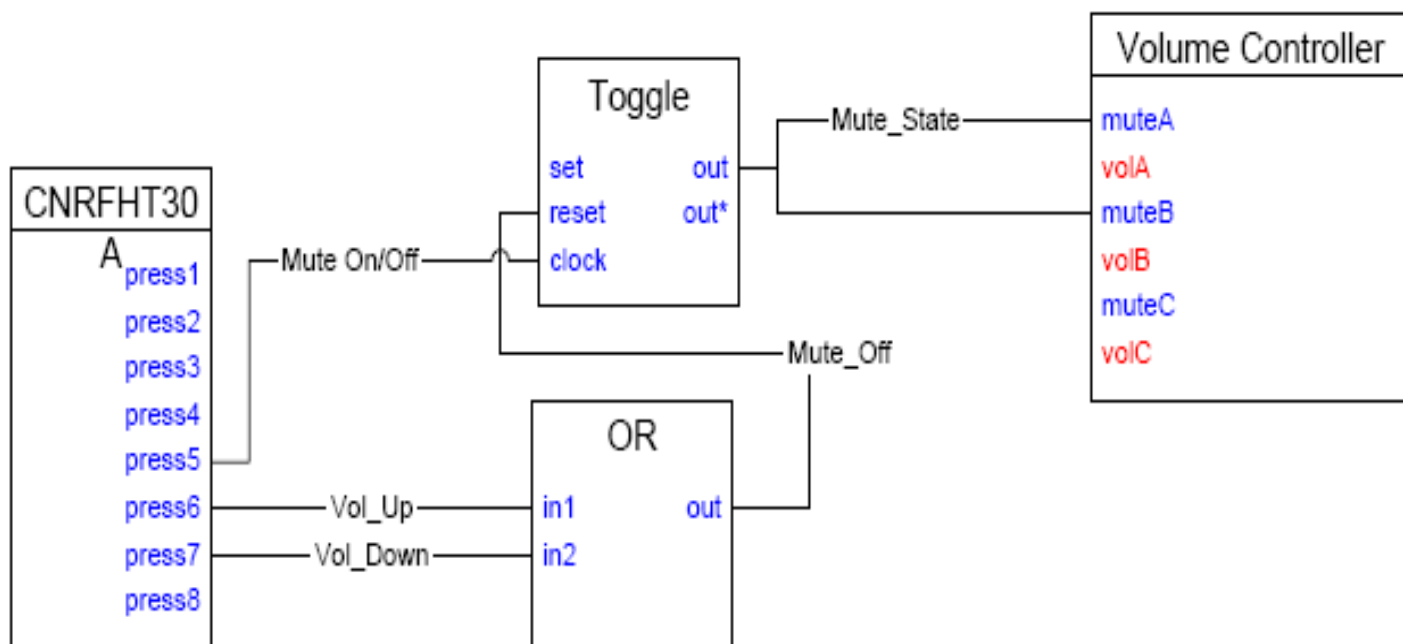
Toggle 函数和 Set/Reset Latch 函数有许多相似之处。实际上，Toggle 函数就是 Set/Reset Latch 函数增加了一个“Clock”输入端。在“Clock”端的每个上升沿都会使函数在 Set 和 Reset 两种状态下翻转。同样设有 Set 和 Reset 输入端用以强制函数输出为指定状态（置 1 或置 0）。

### Toggle 函数举例：音量静音控制

Toggle 函数很容易实现单个按钮对静音的控制。如下图所示，静音按钮连接到“Clock”输入端。每次按按钮都会使信号“Mute\_On”在高电平和低电平之间交替变化，通常该输出信号将连接到像 C2N-VEQ4 音量控制器中的静音继电器上。同时注意“Reset”端的用法，不管按 Volume Up 还是 Volume Down 按钮都可以消除静音。



### Toggle 举例：静音控制

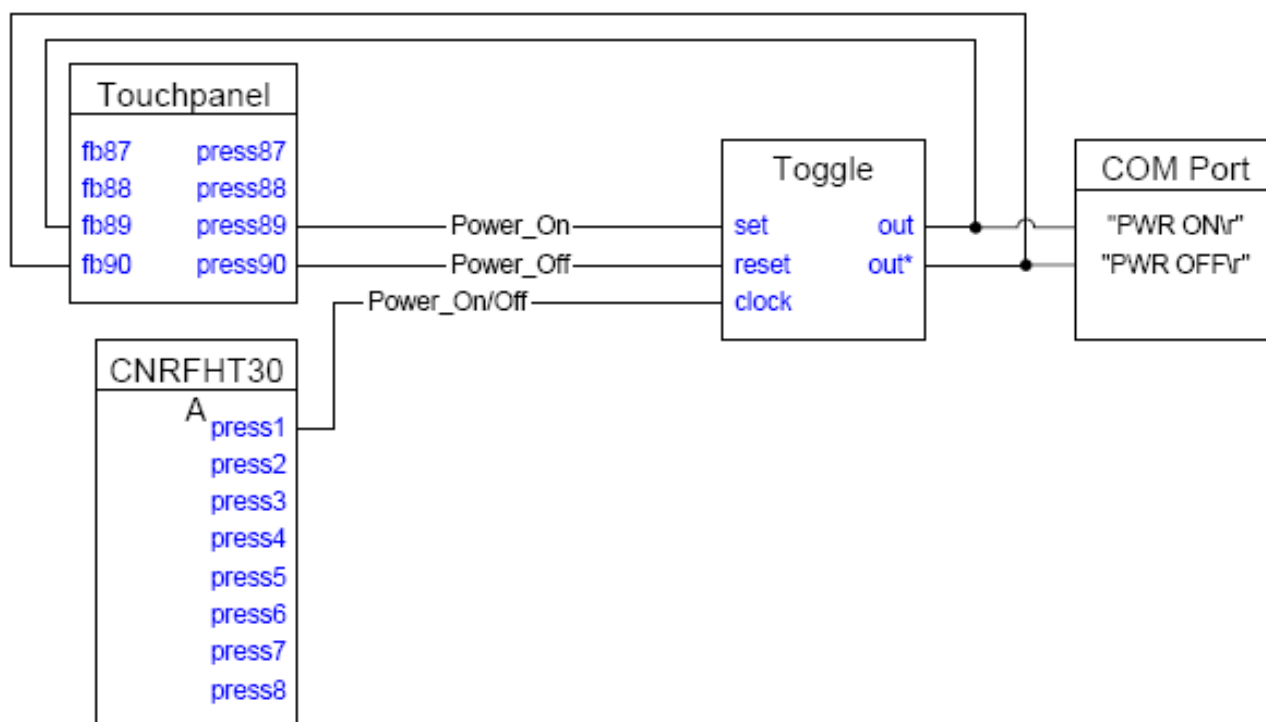


注意：在控制自身带有静音锁定功能的设备时，不建议这样使用 **Toggle** 函数。比如说，如果您通过红外控制 AV 接收器，通过其自身的静音按钮，每发送一次指令，设备会在静音和解除静音之间切换。这种情况下不需要锁定的静音信号，而应该直接将静音按钮发出的信号连接到红外指令上。可能您希望用 **Toggle** 函数为触摸屏提供实际反馈，让用户了解接收器实际的状态。这里要非常注意，通常不建议提供状态反馈，除非您能确定反馈是正确的。如果您认为反馈和接收器实际状态之间可能会出现不同步，最好使用瞬时反馈。

### Toggle 函数举例：设备电源开关

在上一个例子中，仅仅用了 **Toggle** 函数的 **Out** 输出端来控制静音状态。有时需要用到 **Out** 和 **Out\*** 两个输出端，比如用 RS-232 控制的投影机。通常这种设备的开关机指令是分开的。下面的程序说明用 **Toggle** 函数的两个输出端去驱动开关机指令。“vproj\_Power”信号的每个上升沿都会触发一条电源指令。当然要注意的是 **Toggle** 的输出端是锁定的，但 RS-232 指令是在驱动信号的上升沿发出，所以这不会出现问题。但是，如果投影机是由红外控制的，我们就不能这样编程了，因为相应的红外指令会不断的发送到设备。而我们可以增加另外的逻辑通过 **Toggle** 的输出端产生脉冲。这在本书的后面将会涉及到。

### Toggle 举例：设备电源开/关



## Interlock 函数

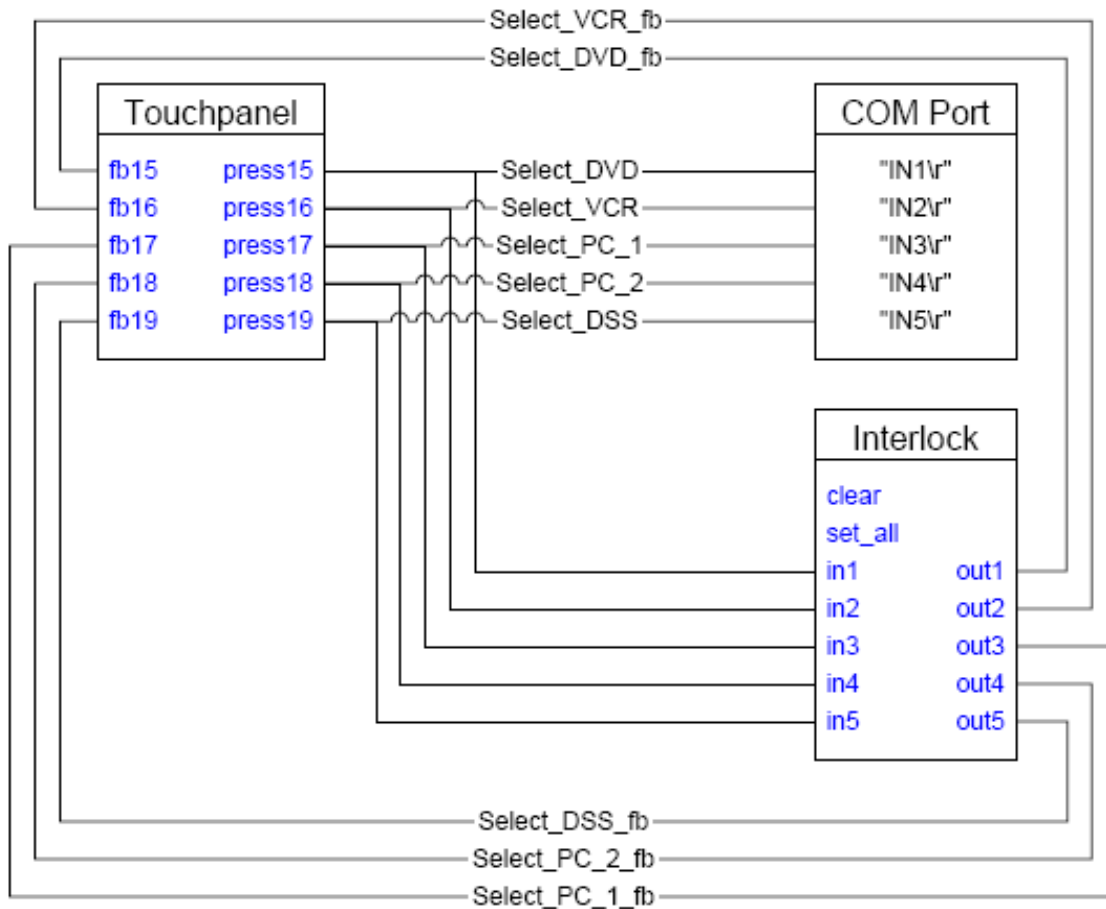
**Interlock** 函数输入端出现上升沿时会使对应的输出端锁定为高电平。另外，其它先前为高电平的输出端解除锁定变为低电平。这样，**Interlock** 函数在任何时刻只有一个输出端为高电平，其余都为低电平（除了“**set all**”输入端，后面讨论）。该属性叫做 **break before make**。事实上 **Toggle** 函数将记住最后变为高电平的输入端。这在当用户想在多个选项中间选择控制时十分方便。

**Interlock** 函数也有两个特殊的输入端“**clear**”和“**set all**”。“**clear**”将使先前为高电平的输出端变为低电平，“**set all**”使所有输出同时变为高电平，这是不只一个输出端为高的唯一场合。这在涉及一些非易失内存的特定应用时有用。

### Interlock 举例：（音 / 视频）源选择反馈

许多音视频系统组成一组源选择。用户可以在一系列的音视频源中选择观看和收听。典型的例子是在会议室中，可能有录像机、影碟机、幻灯机或者计算机，可以通过向矩阵或投影机发送指令选择音视频源。下面所示为使用 **Interlock** 函数实现反馈显示，提示用户当前选定的音视频源。

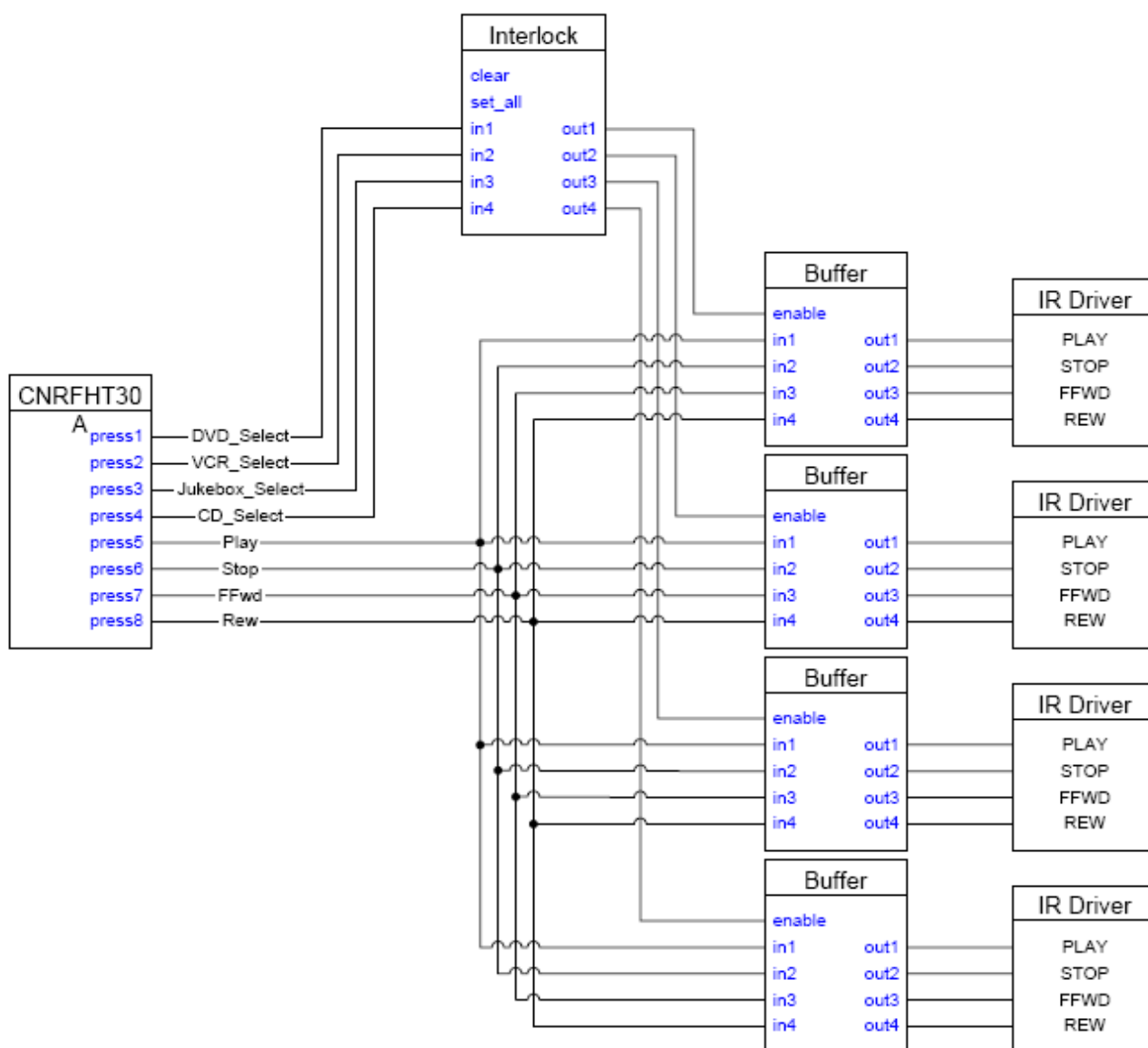
### Interlock 举例：源选择反馈



注意：在这个例子中的输出信号仅仅用做反馈，并没有连接到矩阵去做源选择。这是因为 **Interlock** 的输出端是锁定的，编程时通常很少用锁定的信号去做瞬时的控制功能（像 **RS232** 指令），即使有些情况下这种方式也可以正常工作。因为我们在这个函数的输入端已经有瞬时信号，“clear”使用它们去驱动矩阵更简洁。

用 **Interlock** 的输入信号去驱动矩阵而不用输出信号还有一个原因。如果我们用输出信号去控制，可能因为某种原因需要再次选择同一个信号源（比方说，有人手动切换过矩阵），这时就不能正常工作，除非先选择另一个信号源再选择您想要的源。这是因为 **RS-232** 驱动器是在驱动信号的上升沿发送数据的，一旦 **Interlock** 的一个输出端变高，它不会再次提供上升沿，除非首先关闭（通过选择另外的输入端）然后再次打开。

**Interlock 举例：控制多个设备(part2)- 每个 IR 驱动函数用不同的文件去控制不同的设备**



本章前面我们讨论过用 **Buffer** 函数的一组按钮去控制多台设备。当时我们假设用已产生的适当信号去使能或阻止 **Buffer**，并确定在任意时刻只有一个 **Buffer** 是可用的。现在我们介绍了 **Interlock** 函数，就可以如上图所示完成这个例子了。

## 基于时间的逻辑

到目前为止，我们所看到的逻辑函数都是基于事件驱动的逻辑。当某个事件发生时（如：上升沿跳变），逻辑函数的输出信号变化到对应的状态。但是，仅有事件驱动的逻辑是不够的。某些时候，我们需要控制事件什么时候发生。因此，在这一节里，我们将讨论一些基于时间控制的逻辑函数。

### One Shot 系列

现在我们已经知道如何改变函数输出信号的值（高或低），但是作为程序员，您不能控制这些信号保持同一状态的时间。ONE SHOT 系列的逻辑函数能够实现这种类型的控制。

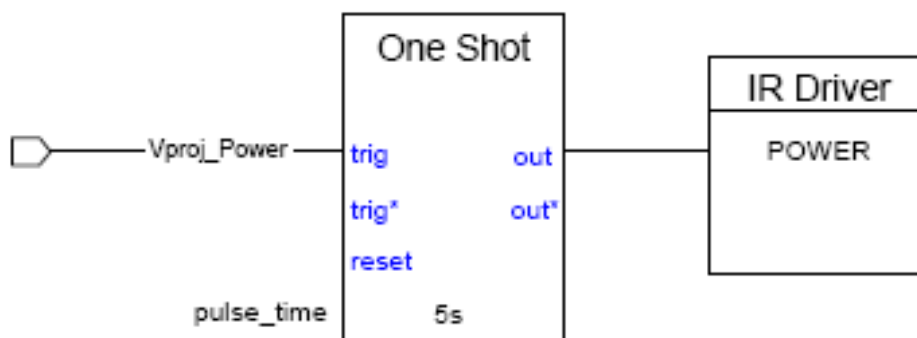
## One Shot

在这个系列中最基本的函数是 One Shot。当输入端“Trigger”出现上升沿时，将触发连接到输出端“Out”上的信号输出一定时间的高电平，时间长度由双精度的“Time”参数决定。这段时间内，不论“Trigger”输入端发生怎样的变化，输出“Out”的值为高。当“Out”的值变低时，函数才可以由另一个上升沿重新触发。同 Set/Reset Latch 及 Toggle 函数的情况一样，输出“Out\*”的值与“Out”互补。

One Shot 还有一个“Trigger\*”输入端，它由下降沿触发。在输入端“Trigger\*”上出现下降沿的效果和“Trigger”上出现上升沿一样。另外还有一个“Reset”输入，它允许您取消正在进行中的 One-Shot 操作。即一旦 Trigger 输入端出现上升沿，连接到“Out”输出端的信号将变高并保持“Pulse\_Time”参数中指定的时长。输出脉冲一旦开始，在脉冲时间未到之前取消它的唯一办法是使“Reset”输入端变高。当 Reset 为高时，“Trig”和“Trig\*”输入无效。

One-Shot 系列中的所有函数（Multiple One-Shots 除外）都有 Reset 输入端，工作方式也都相同。

**One Shot 举例：视频投影机的开机控制图：**



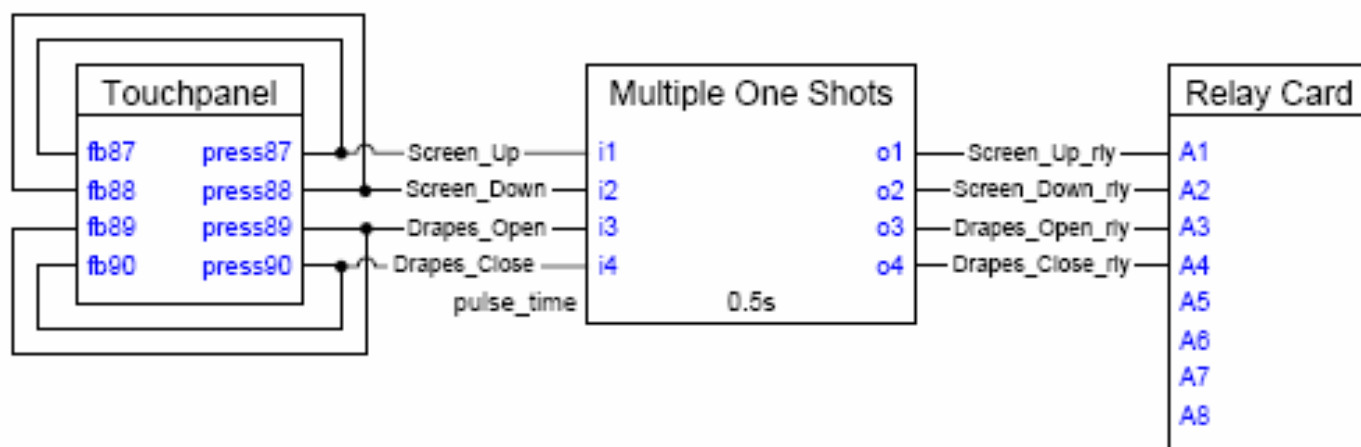
有些品牌的投影机，需要按住红外遥控上的“Power on”一段时间后，才可以正常开机。这是为了防止人们在使用遥控器时意外地开机或关机。在定制的控制系统中，我们通常内部处理这种操作，用户只需要按一下按钮，就可以正常开机了。如上图所示，One Shot 由一个按钮触发，并产生一个 4 秒钟的脉冲驱动红外控制命令。方便的是如果投影机电源开是由 Buffer 编写的电源开启时序中的一个功能，One Shot 将确保同样会产生一个 4 秒钟的脉冲到红外驱动。

## Multiple One Shot

经常您会发现当我们需要用 One Shot 产生一个固定脉冲宽度的信号时，我们需要对其它相关的信号进行同样的处理。这可以通过多个 One Shot 函数来实现，但因为这种情况非常常见，所以有另外一个特别的函数 Multiple One Shot。

Multiple One Shot 函数其实是由多个独立 One Shot 组合而成的。需要注意的是虽然每一对输入/输出分别代表 One Shot 函数的“Trigger”和“Out”，但 Multiple One Shot 没有“trigger\*”和“out\*”。此外，尽管所有的 input/output 对是独立的，但它们具有相同的由双精度参数“Time”确定的脉冲时间。

**Multiple One Shot 举例：电动屏幕和窗帘的继电器控制（图）**



当通过低压继电器控制电动屏幕和窗帘时，必须让继电器闭合足够长的时间，以确保设备接口能够正确识别。多数情况下这些接口需要半秒钟闭合，但是不能确保用户会按足半秒钟。上图中的 Multiple One Shot 编程例子是一种解决方法。

## Retriggerable One Shot

另一种形式的 One Shot 是 Retriggerable One Shot，它的功能基本上和 One Shot 完全一样。唯一不同的是 One Shot 在输出“out”为高时，会忽略“trigger”和“trigger\*”的变化。而 Retriggerable One Shot 无论输出“out”是否为高，都会识别“trigger”的上升沿（或“trigger\*”的下降沿），并再次触发和重新计时。只有当最后一次触发的脉冲时间结束后，“out”输出的值才会变低。

**Retriggerable One Shot：自动关机**