智能控制系统说明书 V2.7

目录

智能控制系统说明书 V1.6 1
一． 系统功能简介。 1
二． 系统架构。 2
三． 软件界面操作说明。 2
四． 数据格式。 5
五． 网络唤醒设置步骤。 6
六． 场景定时触发执行。 7
七． 变量指令。 9
八． 界面组件状态反馈更新。 10
九． 根据接收数据触发发送其他指令。 14
十． Lua 脚本详解。 16
.....错误!未定义书签。
十一． 可视化媒体控制模组。 17
十二． 媒体控制模组。 18
十三． 视频播放组件。 19
十四． 可视化流列表切换组件。 20
十五． 时间显示框.....22

一． 系统功能简介。

1. 本软件系统可作为智能控制系统/智能中控系统/物联网控制系统/智能控制中枢使用，实现对多种外围设备的集成控制。

2. 纯软件系统，可自由搭配各种硬件设备实现定制化集成控制系统。

3. 实现对多种外围设备的集中集成控制，可应用于展厅，多功能厅，会议室，指挥中心，多媒体系统，智能家居等需要智能化自动化集成控制的领域。

4. 搭配智能界面设计软件，零代码自定义编辑 UI 界面和控制逻辑；支持一键预览；编辑完成后，支持一键压缩打包上传界面数据和资源素材（包括界面数据，图片素材，按钮音效素材，字体文件，视频文件）到跨平台触控面板程序，智能触控面板程序支持安卓 Android、Windows 端、苹果 IOS。

5. 支持多种 UI 控制组件，除常见的按钮、图片按钮、切换按钮、文本框、滑动条、时间显示框等外，还支持视频播放组件、可视化媒体控制模组、可视化网络流切换模组。

6. 支持分辨率缩放，编辑好一套界面程序可适配到移动端、平板端、桌面端不同分辨率设备。

7. 支持界面组件状态反馈更新。

8. 搭配智能数据编辑软件，可添加任意数量终端设备，每台设备可添加任意数量控制协议指令数据，支持 16 进制或 ASCII/utf8 字符格式，支持变量指令。

9. 支持添加场景，实现指令序列的功能。

10. 支持场景定时自动触发，实现如定时开关设备，定时开关灯光等功能。可设置按星期循环。

11. 支持场景间隔自动重复触发，实现如数据自动采集等功能。

12. 可自定义编辑 Lua 脚本代码对设备数据进行处理解析，实现控制界面组件反馈更新、指令触发、数据转发的功能；每个设备都可编辑单独运行的脚本程序。

13. 支持多种通信方式的终端设备连接：UDP、TCP 客户端、TCP 服务端、HTTP 客户端、串口通信、MQTT 发布、MQTT 订阅。

14. 搭配性能强大的智能控制服务端软件，可实现对所有终端控制设备同时进行数据收发，数据解析；无网络

设备连接上限；设备之间相互独立，互不干扰。

15. 支持多个智能触控程序同时连接控制。

16. 搭配多媒体互动控制软件，可直接集成媒体播控功能到控制系统，实现播放控制，媒体切换，远程上传管理媒体资源，控制界面媒体资源列表动态更新，可视化预览画面，可视化 KVM 控制等功能。

二. 系统架构。

本系统由 4 套软件构成。

1. 智能控制服务端 

：作为控制中心服务软件，实现对外围设备的数据通信连接和数据收发；本系统的其他软件也需要连接到服务端软件进行数据上传更新和连接控制。

2. 智能界面设计器 

：设计编辑智能触控面板程序的 UI 界面和控制逻辑。编辑完成后需要连接到服务端，并上传界面数据到智能触控面板程序。

3. 控制数据编辑器 

：添加外围通信设备，编辑设备的通信参数、控制指令、Lua 解析脚本。编辑完成后需要连接到服务端，并上传数据到服务端。

4. 智能触控面板 

：控制端软件，连接到服务端进行控制或接收界面数据更新。支持跨平台多端同时运行。

三. 软件界面操作说明。

1. 智能控制服务端 

。

（1）界面功能。

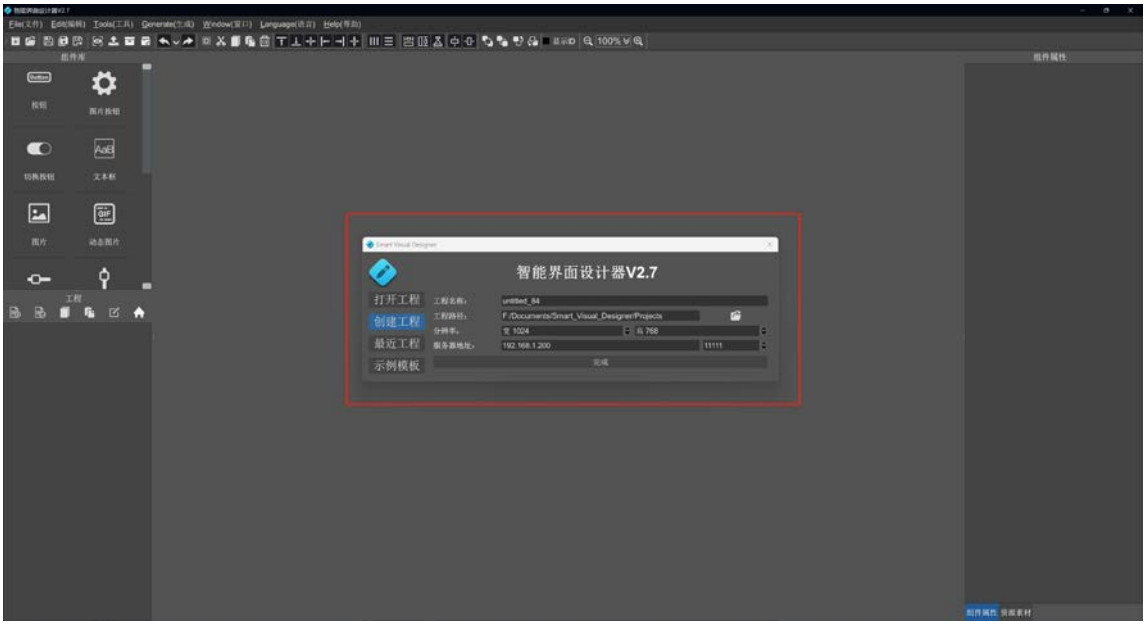
软件打开后界面如下图：

（a）可设置使用的网卡，服务器 IP 地址和端口，设置完成后重新开启服务器生效。

2. 智能界面设计器。

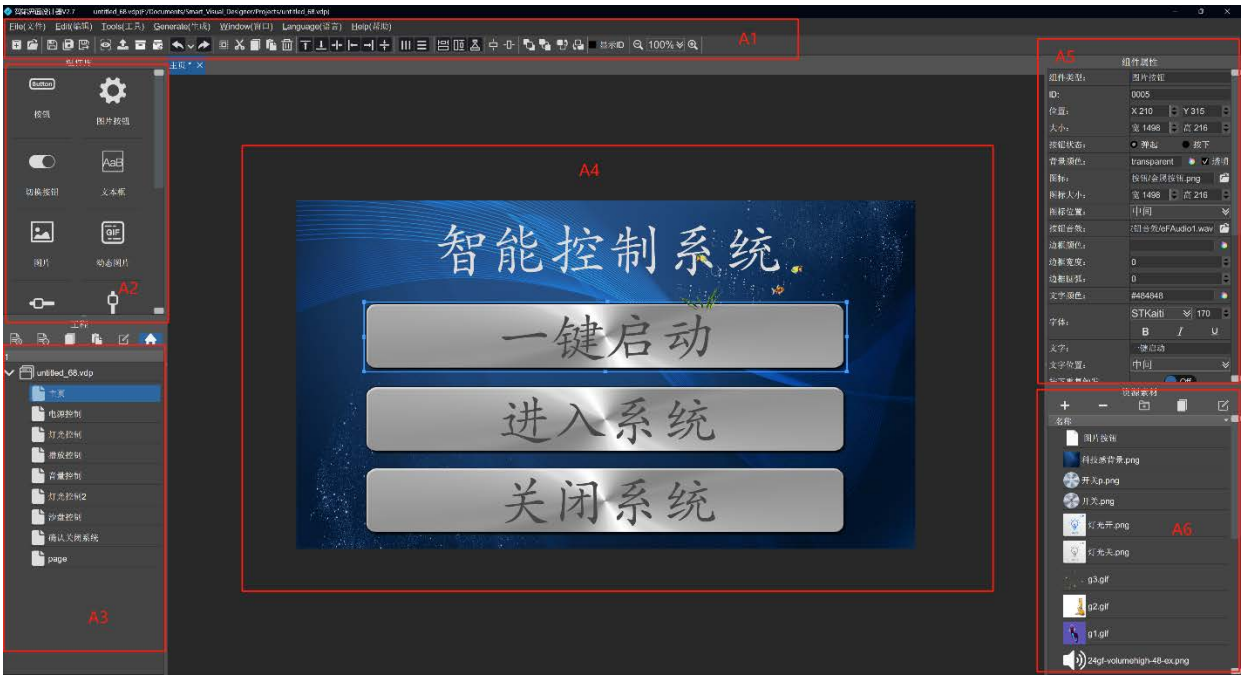
（1）界面功能。

软件打开后界面如下图：



软件打开后会弹出开始对话框（如图 A1），你可以在这里打开工程或者创建工程。

工程创建或者打开后界面如下图：

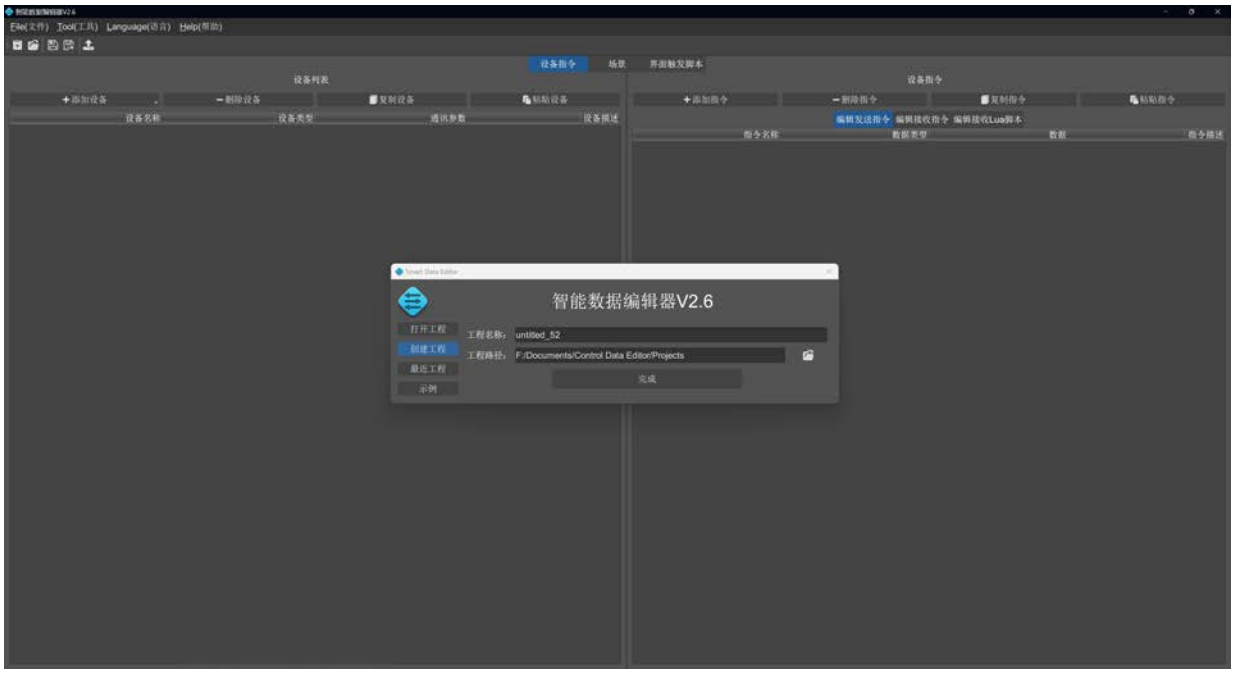


（a） A1 区为工具栏和菜单栏，将鼠标悬停在工具图标上可以看到相应的功能。
（b） A2 区为组件库，拖拽组件到画布区即可创建添加需要的组件。
（c） A3 区为工程目录区，可以添加、删除、复制、粘贴、重命名页面以及设置主页。
（d） A4 区为画布区，可以拖动改变组件的大小位置、拖动单选或者多选组件。
（e） A5 区为组件属性区，可以编辑选中组件的属性。
（f） A6 区为资源素材区，可以添加管理素材、复制资源路径等操作。

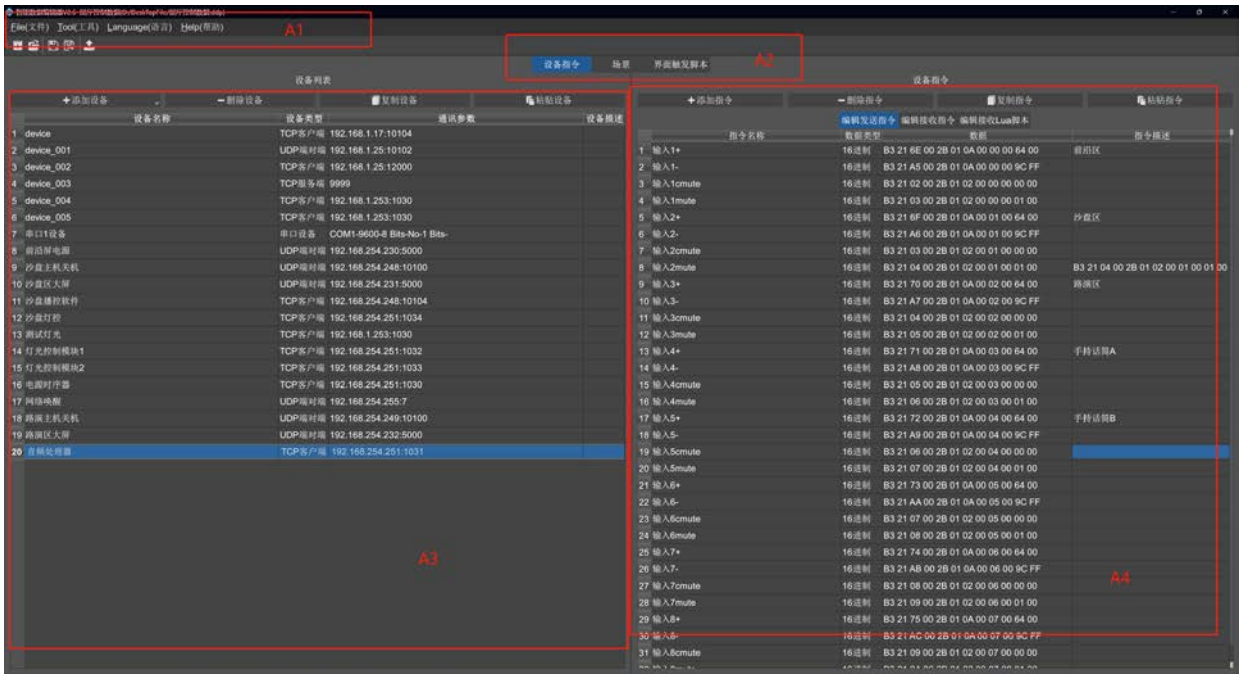
3. 智能数据编辑器。

（1） 界面功能。

打开界面如下图，可以在开始对话框创建、打开工程文件。



打开工程后，如下图：



（a） A1 区为工具栏、菜单栏区，可以打开、创建、保持工程文件。

（b） A2 为功能选择区，可以选择设备指令编辑区、场景编辑区、界面触发脚本编辑区。

（c） A3 为设备编辑区，可以在这里添加、删除、编辑设备。

（d） A4 为对应指令编辑区，选中相应设备后，即可编辑对应设备的指令。

4. 智能触控面板。

软件打开后界面如下图：

如果未登录到服务器，会跳出登录框，输入正确的 IP 地址和端口，登录即可。登录成功后，即可接收智能界面设计器的界面文件，接收到界面文件后自动更新。

四． 数据格式。

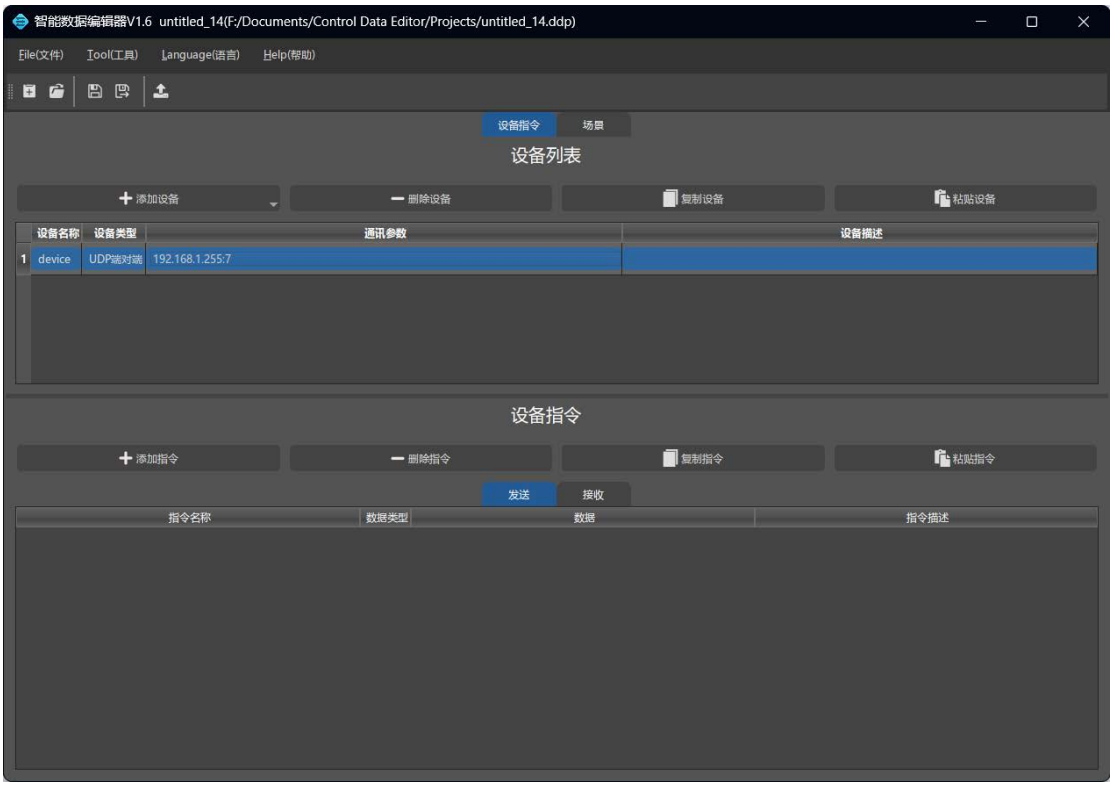
字符格式支持 ASCII 或者 utf8 格式。

16 进制格式，使用空格分割每个字节（byte），比如“FF 00 01 02 FF”，注意前面和后面没有空格。

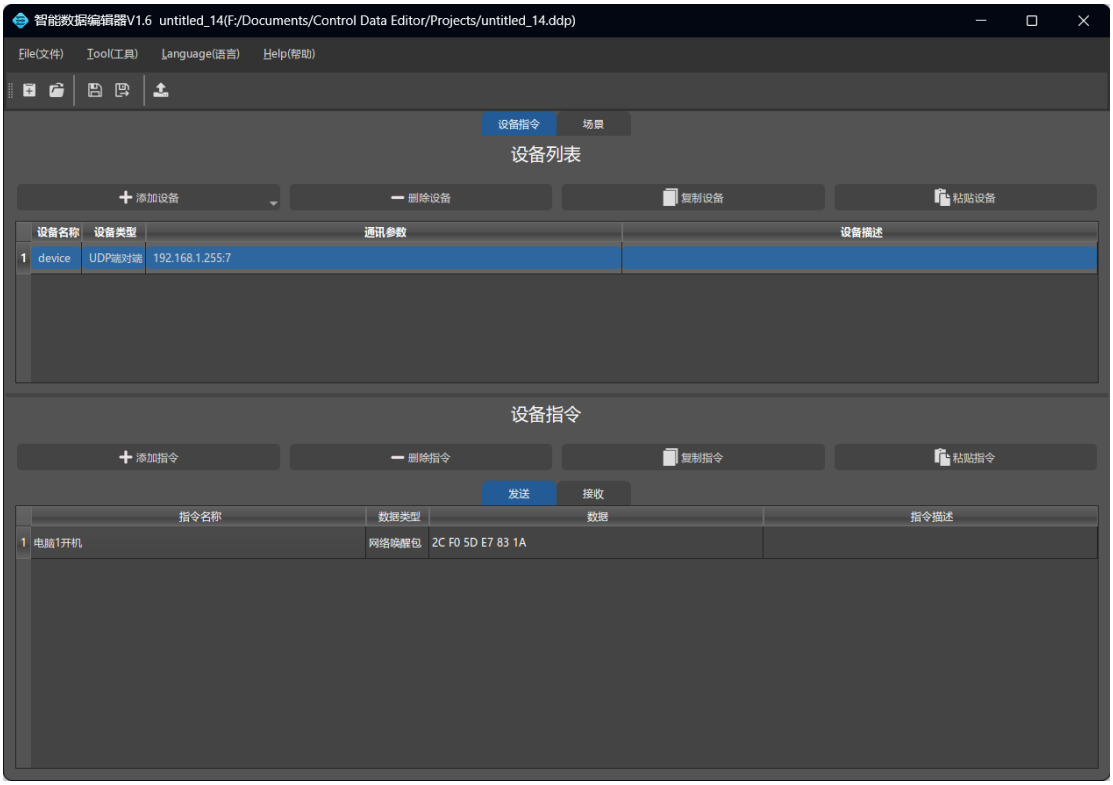
五． 网络唤醒设置步骤。

1. 被网络唤醒的电脑主板必须支持网络唤醒（wake on lan），然后需要打开主板的网络唤醒功能；操作系统的网络唤醒功能也需要打开：在控制面板->网络连接->选中实际工作网卡->右键属性->点击配置->电源管理->勾选运行此设备唤醒计算机->勾选只允许幻数据包唤醒计算机。

2. 在指令数据编辑器添加一个 UDP 设备，通信参数设置为局域网网段的广播地址，比如需要网络唤醒主机的 IP 地址为“192.168.1.22”，广播地址为“192.168.1.255”，以此类推。



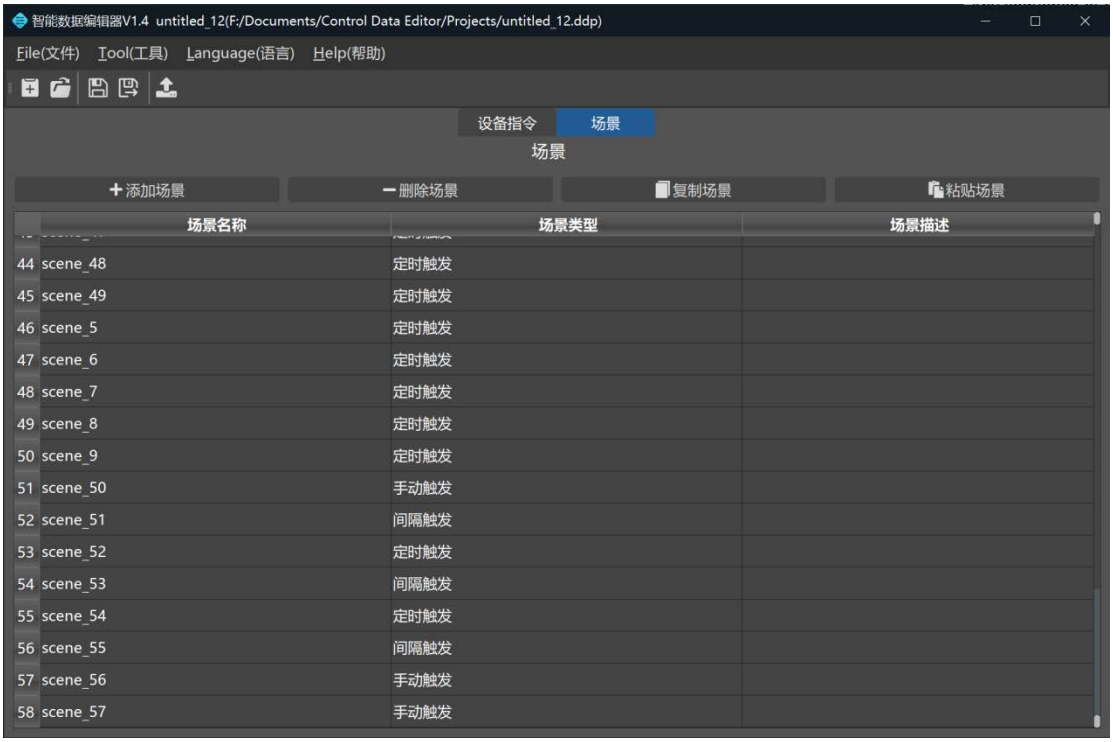
3. 在添加的 UDP 设备下，添加指令数据，数据类型选择“网络唤醒包”，数据添加需要网络唤醒电脑的 MAC 地址。**注意 MAC 地址数据的格式是 16 进制，空格分割，前后无空格。**MAC 地址可以通过命令行工具，输入“ipconfig/all”查看。



4. 把添加的开机命令绑定到界面按钮触发即可。

六． 场景定时触发执行。

场景是由多个指令按顺序执行的指令序列。
可以在场景栏添加新的场景。



点击 

后，会弹出配置框，如下图：

你可以在配置框添加指令、设置指令间执行的间隔、设置场景的类型。
手动触发指令用于绑定到界面组件，通过组件触发，比如按钮的点击。

定时触发用设定一个时间，每日触发，可以设置按星期循环。

间隔触发用设置一个间隔时间，指令按间隔时间重复触发执行。

七． 变量指令。

变量指令用于一些控制指令需要动态变化的场景，比如音量调节、播放　进度、矩阵切换等。
变量指令用“[X]”来标识变量位置，X 表示变量 ID，用十进制数字表示。
假设某矩阵的切换指令为“01*02!”表示输入源 1 路切换到输出 2 路，则用变量指令表示为“[1]*[2]!”，输入源的变量 ID 为“1”，输出源的变量 ID 为“2”。
假设指令添加后如下图：

下面讲解如何在界面设计器应用这条指令：
这条切换指令需要用按钮组件来实现。需要配置的按钮属性如下：

所有用于矩阵切换的按钮的按下触发指令设置为刚才添加的指令。

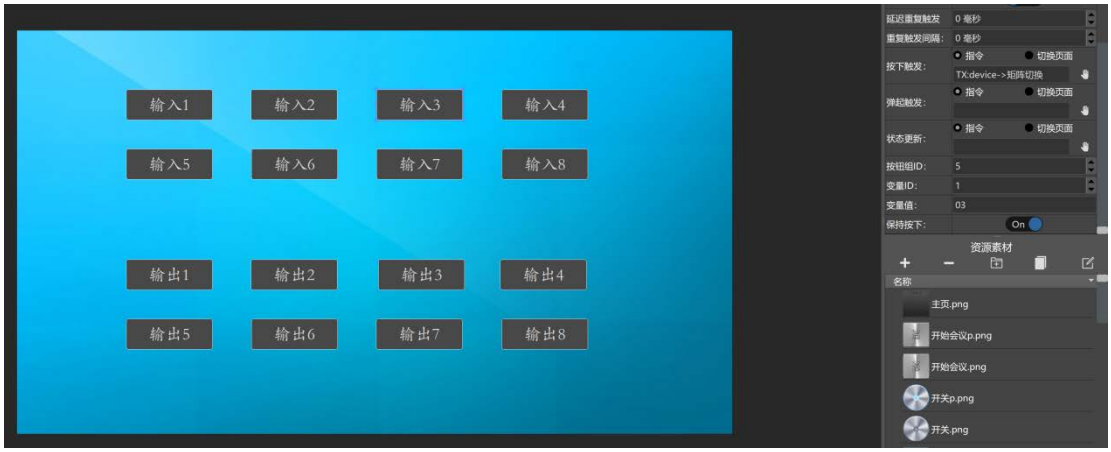
把所有用于矩阵切换的按钮设置到同一个按钮组，这里假设是 5。
注意不要更跟其他按钮组 ID 重复。

设置一个变量 ID，该变量 ID 就是对应上面变量指令的 ID，上面的指令有“1”、“2”两个 ID，分别代表输入输出。

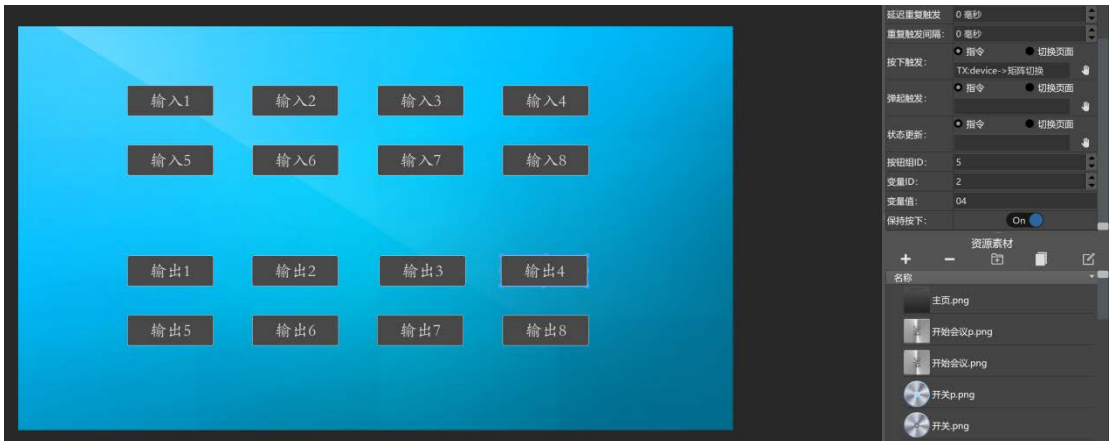
设置该按钮的变量 ID 对应的变量值，这里假设是“01”，指令执行后会把该变量值替换到对应变量 ID 的位置

在切换过程中，如果需要按钮保存按下状态直到指令切换完毕，可以打开“保存按下”开关。

假设是 8*8 的切换矩阵，
按钮“输入 3”的属性如下：



按钮“输出 4”的属性如下：

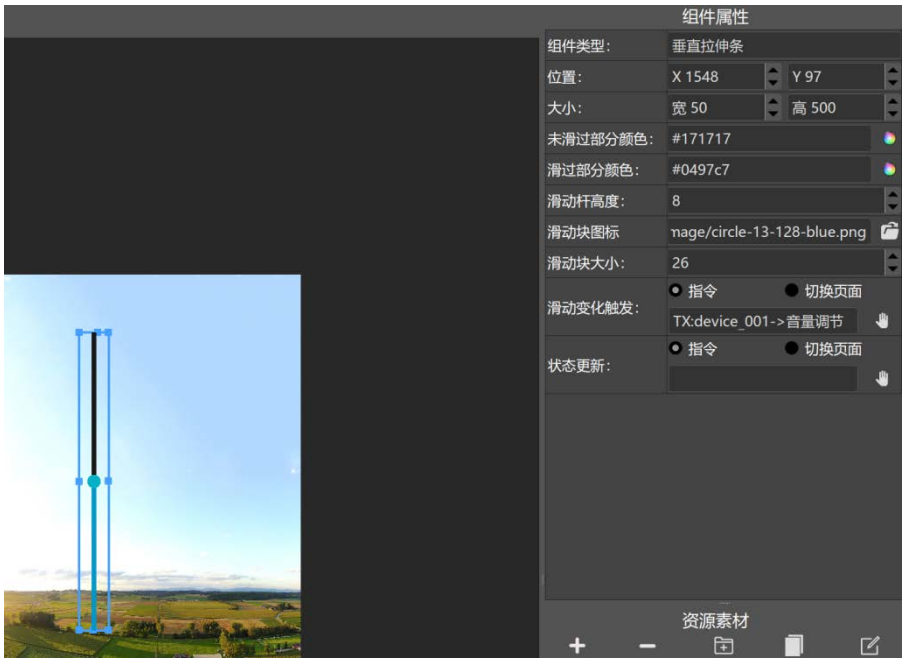


除了按钮组件外，滑动条组件也是通过变量发送指令的。

假设某设备设置音量的指令为“volumeTo=50##”表示调节音量到 50%，则用变量指令表示为“volumeTo=[1]##”。变量 ID“1”表示滑动条滑动后的值，滑动条值范围为 0-100。

假设指令添加后如下图：

界面滑动条组件属性如下图：



当滑动条滑动后会自动把值替换到变量 ID“1”。

八． 界面组件状态反馈更新。

实现界面组件的动态更新。比如当继电器或电脑关闭后，通过反馈回来的数据更新按钮状态是按下还是弹起。反馈更新需要接收设备发送过来的数据，并通过简单的脚本分析后更新到界面组件。

以本公司的 pc 控制精灵和多媒体播控软件的状态反馈举例，根据电脑主机状态反馈数据更新控制界面的按钮、文字框、滑动条变化：

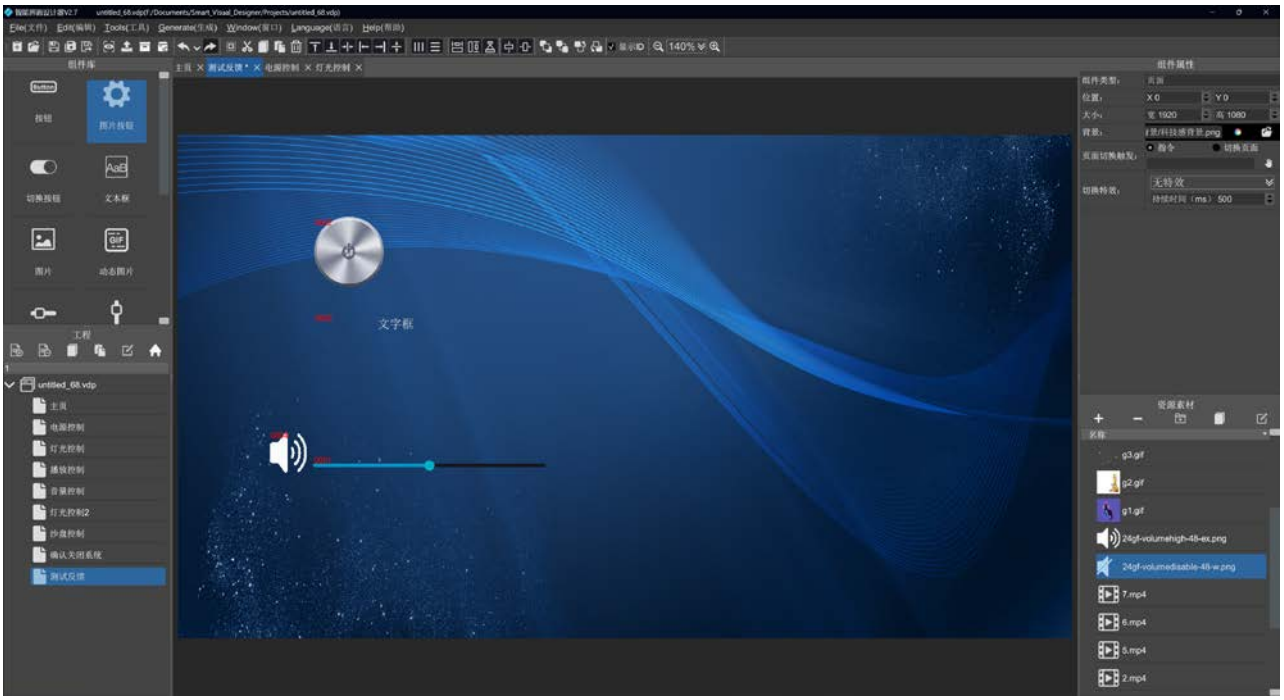
关机后返回字符串“OFFLINE”，开机后返回字符串“ONLINE”。

音量变化返回字符串“volume=25”表示电脑音量在 25%。

静音返回字符串“mute=1”，未静音返回字符串“mute=0”。

首先在界面编辑器添加好需要的组件，这里假设界面如下，在名称为“测试反馈”的页面，添加了一个保持按下

按钮，一个状态文本框，一个音量滑动条，勾选显示 ID 可显示每个组件的 ID：



我们可以看到状态按钮的 ID 为“0002”、文本框的 ID 为“0003”、滑动条的 ID 为“0001”、静音状态按钮 ID 为“0004”，在编辑脚本的时候需要用到这些 ID。

然后在指令编辑器添加一个设备，切换到编辑接收 Lua 脚本区，添加如下脚本：

```lua
function process(data)

    --对比是否是离线数据
     if data == "OFFLINE" then
        --更新按钮状态，"测试反馈"为页面名称，"0002"为组件 ID，false 表示弹起状态
        update("测试反馈", "0002", false)
        --更新文本框状态，
        update("测试反馈", "0003", "text:离线;color:#ff0000")
    --对比是否是在线数据
    elseif data == "ONLINE" then
        --更新按钮状态，"测试反馈"为页面名称，"0002"为组件 ID，false 表示按下状态
        update("测试反馈", "0002", true)
        --更新文本框状态，
        update("测试反馈", "0003", "text:在线;color:#00ff00")
    --对比是否是音量变化数据
    elseif string.sub(data, 1, 7) == "volume=" then
        --更新滑动条进度，页面名称和 ID 根据实际情况修改
        update("测试反馈", "0001", tonumber(string.sub(data, 8)))
    --对比是否是静音数据
    elseif data == "mute=1" then
        --更新静音按钮为按下状态，页面名称和 ID 根据实际情况修改
        update("测试反馈", "0004", true)
    elseif data == "mute=0" then
        --更新静音按钮为弹起状态，页面名称和 ID 根据实际情况修改
        update("测试反馈", "0004", false)
    end
end
```

所有数据对比都在函数 process 内，函数参数 data 是该设备接收到的数据。
下面这行代码对比接收到的数据是否等于“OFFLINE”表示主机关机离线

```
if data == "OFFLINE" then
```

如果等于的话就执行界面更新函数 update(“页面名称”, “组件 ID”, “变量值”)

```
update("测试反馈", "0002", false)
```

每次对比可调用多次 update 函数更新多个组件 ，下面更新的是文本框组件的文字和文字颜色。

```
update("测试反馈", "0003", "text:离线;color:#ff0000")
```

如果需要进行下一次对比，使用下面代码继续对比接收到的数据是否等于“ONLINE”，表示主机开机上线：

```
--更新按钮状态，"测试反馈"为页面名称，"0002"为组件 ID，false 表示按下状态
update("测试反馈", "0002", true)
--更新文本框状态，
update("测试反馈", "0003", "text:在线;color:#00ff00")
```

继续对比音量变化反馈，音量的反馈代码为“volume=25”，该字符串从第 1 到第 7 个字符（“volume=”）不会发送变化，从第 8 个字符串开始是音量数值变量，所以需要对比部分字符串，使用下面代码：

```
elseif string.sub(data, 1, 7) == "volume=" then
    --更新滑动条进度，页面名称和 ID 根据实际情况修改
    update("测试反馈", "0001", tonumber(string.sub(data, 8)))
```

string.sub(data, 1, 7)返回数据接收数据 data 的第 1 到第 7 为的数据，然后对比是否等于“volume=”，如果对比成功，执行更新指令“音量变化”，变量为音量数值，音量数值是从第 8 个字符串开始的，使用 tonumber(string.sub(data, 8))将第 8 个字符串之后的数据转换为数值。

最后继续对比静音变化反馈，静音的反馈代码为“mute=1”，未静音代码未“mute=0”

```
elseif data == "mute=1" then
    --更新静音按钮为按下状态，页面名称和 ID 根据实际情况修改
    update("测试反馈", "0004", true)
elseif data == "mute=0" then
    --更新静音按钮为弹起状态，页面名称和 ID 根据实际情况修改
    update("测试反馈", "0004", false)
end
```

编写完指令和脚本后点击保存。
注意：脚本编辑好后必须点保存才能生效，如果提示编写的脚本代码有错误，需要修改正确才能保存。

脚本里除了对比字符串外，也可以对比 16 进制数据，每位 16 进制数值以\x 开头。
下面代码实现了对某款灯光控制器的状态反馈更新：
该灯光控制模块获取回路状态后返回代码：

“03 03 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 93 B1”，第 1 位到第 3 位不变可作为区分代码，第 4 到位到第 27 位共 24 位数据表示 12

路开关的状态，每 2 位表示 1 路状态，00 00 表示关，00 01 表示开。

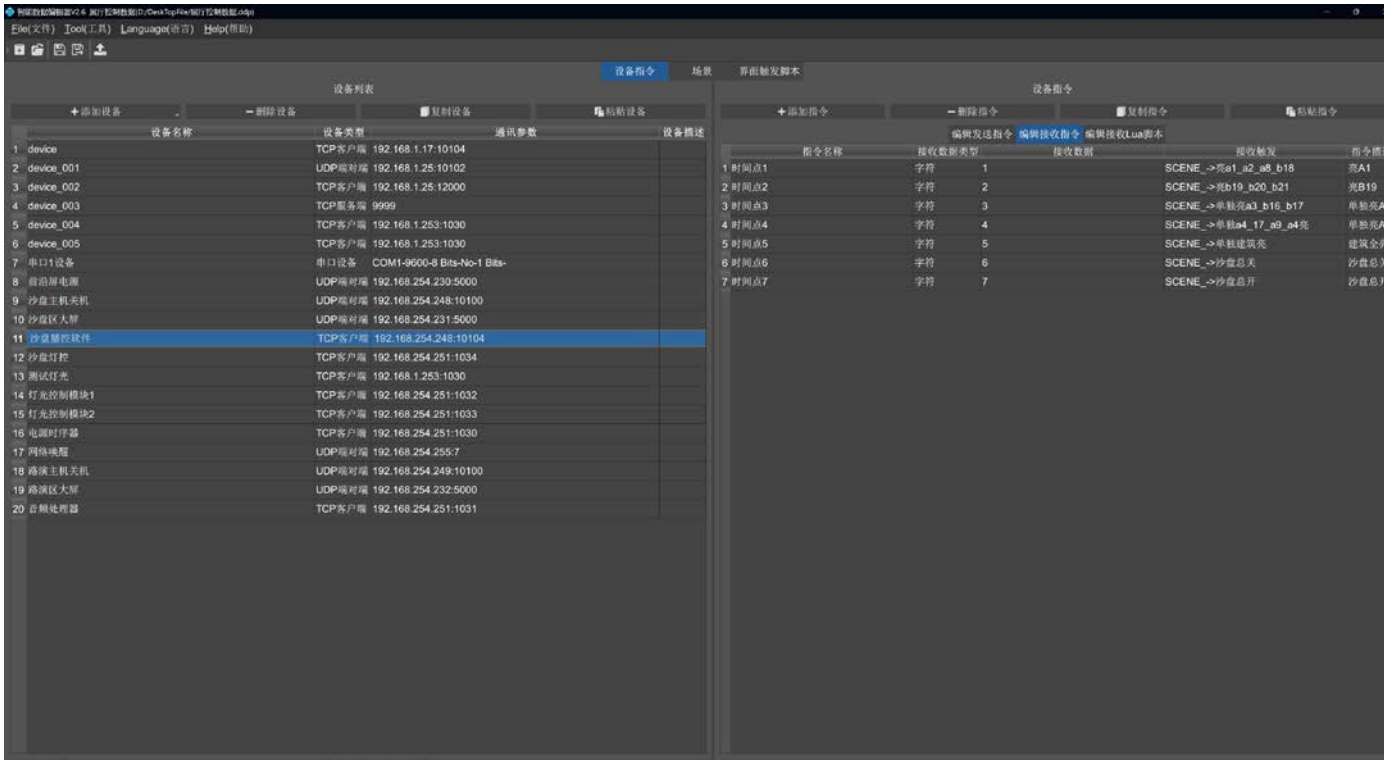
脚本如下，注意页面名称和 ID 根据实际界面情况变化：

```
function process(data)

    if(string.sub(data, 1, 3) == "\x03\x03\x18") then

            if(string.sub(data, 5, 5) == "\x00") then
                update("灯光控制", "0001", false)
            end

            if(string.sub(data, 5, 5) == "\x01") then
                update("灯光控制", "0001", false)
            end

            if(string.sub(data, 7, 7) == "\x00") then
                update("灯光控制", "0002", false)

            end

            if(string.sub(data, 7, 7) == "\x01") then
                update("灯光控制", "0002", false)
            end

            if(string.sub(data, 9, 9) == "\x00") then
                update("灯光控制", "0003", false)
            end

            if(string.sub(data, 9, 9) == "\x01") then
                update("灯光控制", "0003", false)
            end

            if(string.sub(data, 11, 11) == "\x00") then
                update("灯光控制", "0004", false)
            end

            if(string.sub(data, 11, 11) == "\x01") then
                update("灯光控制", "0004", false)
            end

            if(string.sub(data, 13, 13) == "\x00") then
                update("灯光控制", "0005", false)
            end

            if(string.sub(data, 13, 13) == "\x01") then
                update("灯光控制", "0005", false)
            end

            if(string.sub(data, 15, 15) == "\x00") then
                update("灯光控制", "0006", false)
```

```
            end

            if(string.sub(data, 15, 15) == "\x01") then
                update("灯光控制", "0006", false)
            end

            if(string.sub(data, 17, 17) == "\x00") then
                update("灯光控制", "0007", false)
            end

            if(string.sub(data, 17, 17) == "\x01") then
                update("灯光控制", "0007", false)
            end

            if(string.sub(data, 19, 19) == "\x00") then
                update("灯光控制", "0008", false)
            end

            if(string.sub(data, 19, 19) == "\x01") then
                update("测试反馈", "0008", false)
            end

    end
end
```

九． 根据接收数据触发发送其他指令。

对比接收到的数据触发其他指令，比如实现沙盘互动，根据视频播放进度触发沙盘灯光亮灭。
在数据编辑器接收栏添加指令，选中接收数据类型，输入接收的数据值，选择触发的指令或者场景。
除了在这里添加外，也可以在接收数据进行解析后触发。


File(文件) Tool(工具) Language(语言) Help(帮助)
设备指令
场景
界面触发脚本
设备列表
+添加设备
-删除设备
复制设备
粘贴设备
设备名称
设备类型
通讯参数
设备描述
1 device TCP客户端 192.168.1.17:10104
2 device_001 UDP端对端 192.168.1.25:10102
3 device_002 TCP客户端 192.168.1.25:12000
4 device_003 TCP服务端 9999
5 device_004 TCP客户端 192.168.1.253:1030
6 device_005 TCP客户端 192.168.1.253:1030
7 串口1设备 串口设备 COM1-9600-8 Bits-No-1 Bits-
8 前沿屏电源 UDP端对端 192.168.254.230:5000
9 沙盘主机关机 UDP端对端 192.168.254.248:10100
10 沙盘区大屏 UDP端对端 192.168.254.231:5000
11 沙盘基控软件 TCP客户端 192.168.254.248:10104
12 沙盘灯控 TCP客户端 192.168.254.251:1034
13 测试灯光 TCP客户端 192.168.1.253:1030
14 灯光控制模块1 TCP客户端 192.168.254.251:1032
15 灯光控制模块2 TCP客户端 192.168.254.251:1033
16 电源时序器 TCP客户端 192.168.254.251:1030
17 网络唤醒 UDP端对端 192.168.254.255:7
18 路演主机关机 UDP端对端 192.168.254.249:10100
19 路演区大屏 UDP端对端 192.168.254.232:5000
20 音频处理器 TCP客户端 192.168.254.251:1031
设备指令
+添加指令
-删除指令
复制指令
粘贴指令
编辑发送指令
编辑接收指令
编辑接收Lua脚本
指令名称
接收数据类型
接收数据
接收触发
1 时间点1 字符 1 SCENE_->亮a1_a2_a8_b18 亮A1
2 时间点2 字符 2 SCENE_->亮b19_b20_b21 亮B19
3 时间点3 字符 3 SCENE_->单独亮a3_b16_b17 单独亮A
4 时间点4 字符 4 SCENE_->单独a4_17_a9_a4亮 单独亮A
5 时间点5 字符 5 SCENE_->单独建筑亮 建筑全
6 时间点6 字符 6 SCENE_->沙盘总关 沙盘总
7 时间点7 字符 7 SCENE_->沙盘总开 沙盘总

十． Lua 脚本详解。

数据编辑器的脚本，支持所有标准的 Lua 脚本语法和函数方法；另外还提供 3 个专用于和软件内部交互的函数：

send_command(″设备名称″, ″指令名称″)这个函数可以执行一条发送指令

send_data(″设备名称″, ″要发送的数据″)这个函数可以从指定设备发送自定的数据

update(″页面名称″, ″组件 ID″, ″变量″)这个函数可以更新界面组件的状态，如果组件时按钮，变量值可以时 true 或者 false；如果时滑动条，变量值时数值；如果时文本框，变量值时字符。

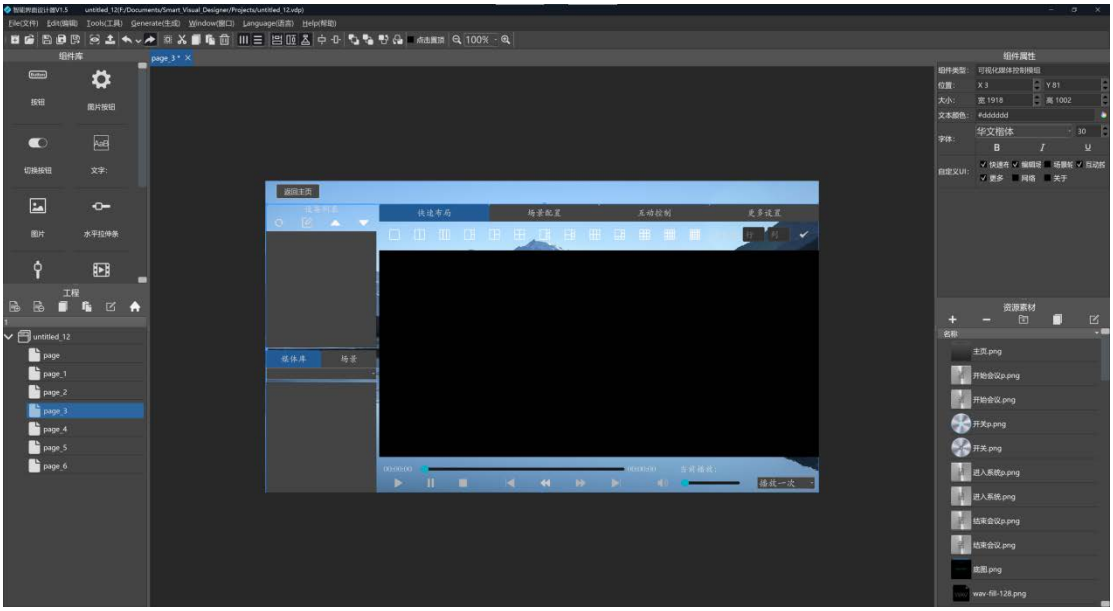
注意！！千万不要用中文的符号，比如逗号、冒号、括弧等等，必须英文的。字符串内例外。

```lua
function process(data)

    --对比接收数据是否等于字符串"123"
    if data == "123" then
            --执行一条发送指令，"device_001"为设备名称，"command"为命令名称。
            send_command("device_001", "command")
    end

    --对比接收数据是否等于字符串"456"
    if data == "456" then
            --直接从指定设备发送字符数据，"device_001"为设备名称，第二个参数为要发送的数据。
            send_data("device_001", "some raw data bytes")
    end

    --对比接收数据是否等于 16 进制 01 01 02
    if data == "\x01\x01\x02" then
        --直接从指定设备发送 16 进制数据，"device_001"为设备名称，第二个参数为要发送的数据。
        send_data("device_001", "\xff\xff\x01")
    end

    if data == "aaa" then
            --执行一个场景指令，如果设备名称为空的，后面的指令名称就是场景名称。
            send_command("", "scene")
    end

    if data == "on" then
            --更新界面组件的状态，"灯光控制"为页面名称，"0013"为组件的 ID，true 是变量参数，
            --如果这个组件为可按下按钮，true 更新按钮为按下状态，false 更新为弹起状态
            update("灯光控制", "0013", true)
    end

    if data == "off" then
            --同上，更新按钮为弹起状态
            update("灯光控制", "0013", false)
    end
end
```

十一． 可视化媒体控制模组。

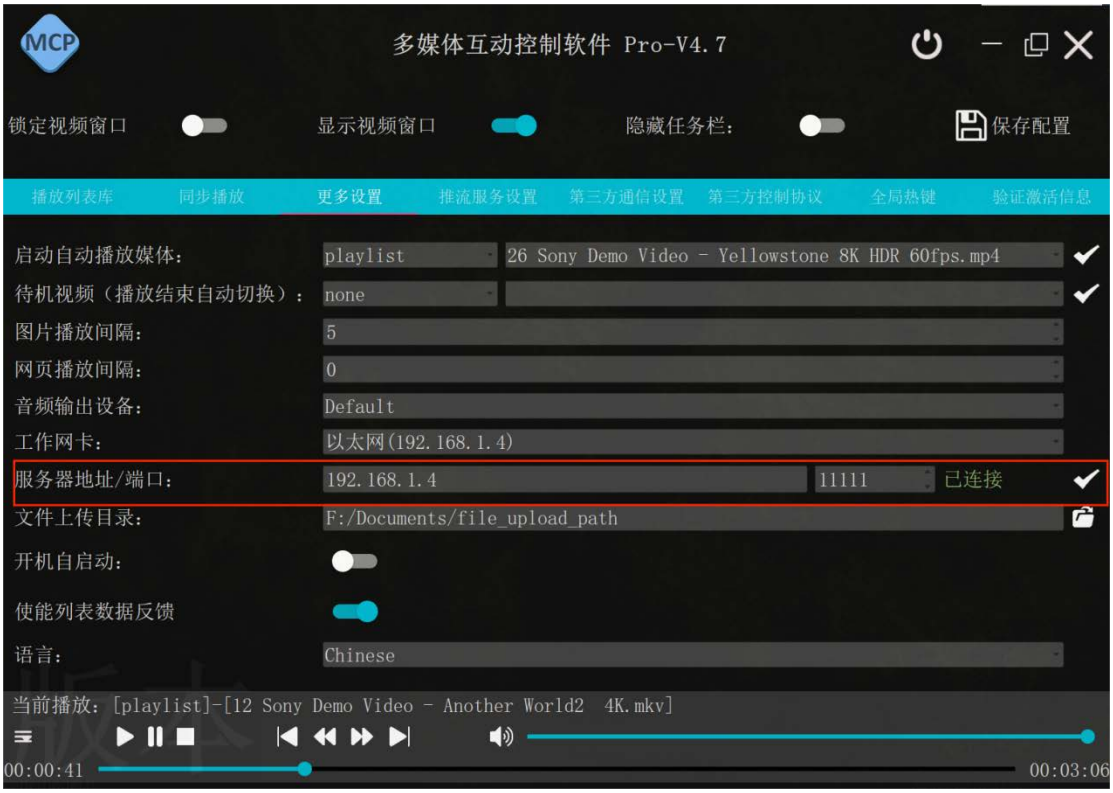
【可视化媒体控制模组】可直接搭配本公司的多媒体互动控制软件，实现集成可视化播放控制到中控系统。实现对播放电脑的媒体切换、播放暂停、音量控制等。如果播放的网页媒体，还可以实现对网页的互动控制，如点击，上下翻页，上下滑动。

只需要将界面编辑器组件库的【可视化媒体控制模组】拖拽到界面页面上，调整好位置大小，上传到控制设备即可，无需填写一行代码。



界面编辑好后，还需要对多媒体互动控制软件进行必要的配置：

1. 在更多设置->服务器地址/端口处填写智能控制服务端的 ip 地址和端口，然后点勾应用。连接成功后，后面会显示【已连接】状态。

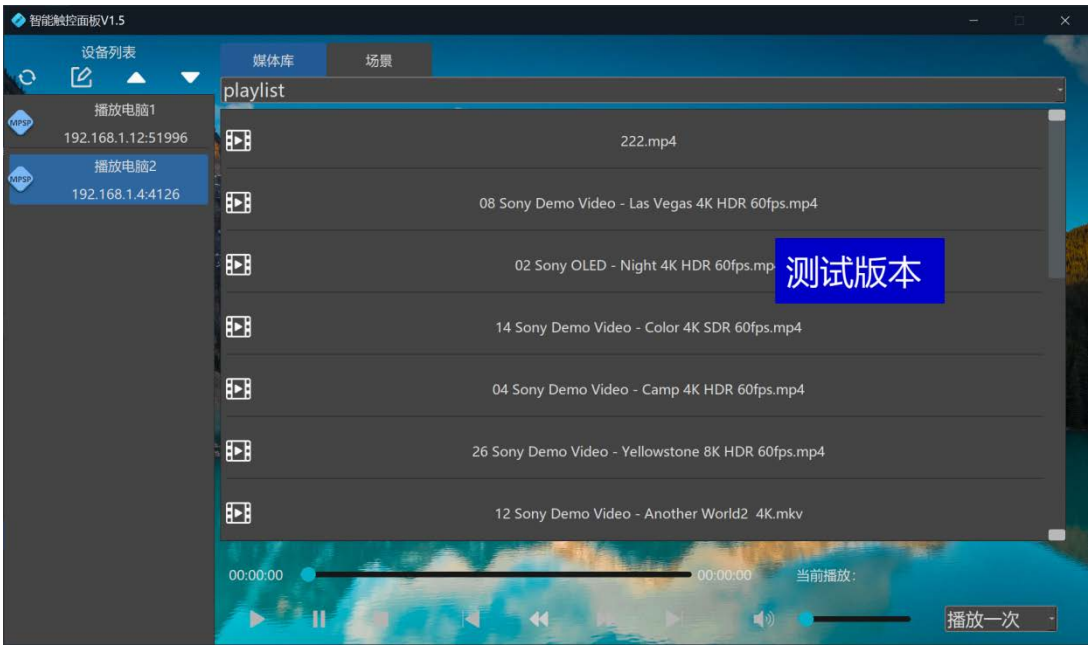


2. 在推流服务设置里，打开【启动推流】开关，如果推流 IP 不是本机 IP，设置成本机 IP，本 IP 不会随电脑 IP 自动变化，设置完毕点【应用】。

3. 设置完毕后必须点保存配置，然后重启软件生效。

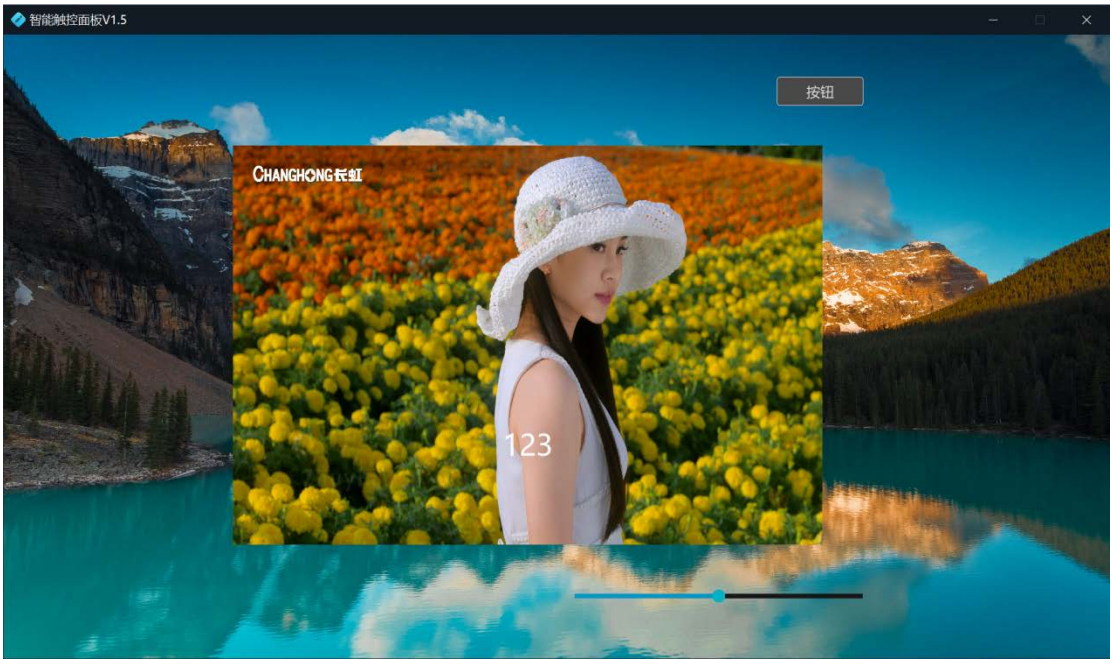
十二． 媒体控制模组。

【媒体控制模组】是去除了可视化预览的控制模组，实现跟【可视化媒体控制模组】一样的功能，配置方法也是一样的，因为去除了可视化预览，【多媒体互动控制软件】不需要打开推流，只需要连接到服务器即可。

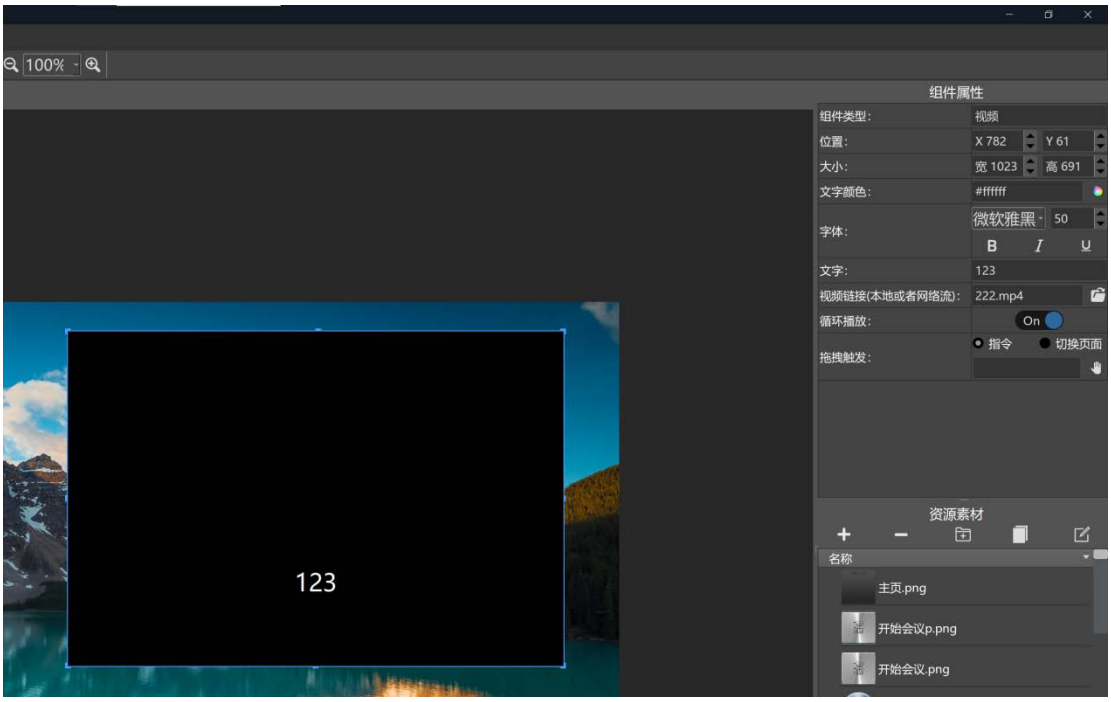


十三． 视频播放组件。

【视频播放组件】可播放网络流或者本地文件媒体。



拖拽视频组件到画布后，点击该组件可编辑属性：



【文字】属性可设置是否显示文字。

【视频链接】属性可以输入网络流链接、从资源素材库复制视频链接、或者添加视频文件到素材库。

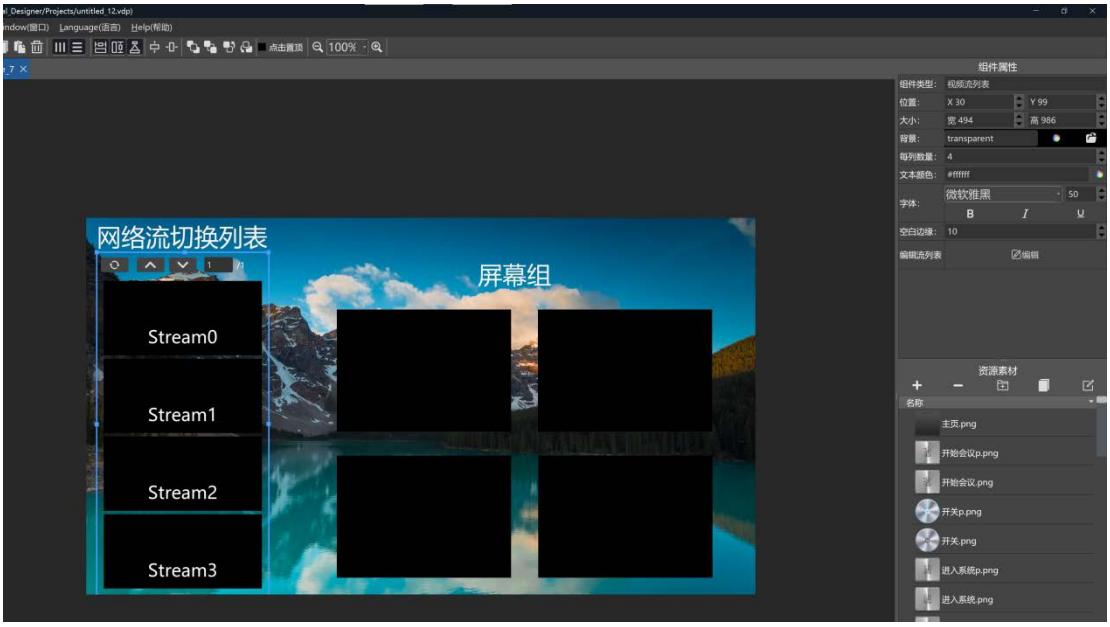
【循环播放】素材可以设置是否循环播放。

【拖拽触发】用于配合【可视化流列表组件】使用。

十四． 可视化流列表切换组件。

【可视化流列表切换组件】可用于搭配分布式实现可视化切换网络流的作用。通过添加一个流切换列表，多个视频播放组件构成屏幕组来实现。当拖动流列表的单元到某一个视频播放组件后，该视频播放组件会播放该流单元的辅流，同时发送变量指令到外部设备，变量指令的参数是主流链接。

拖拽该组件到画布后，点击该组件可编辑属性：



【背景】属性可设置列表的背景图片、颜色，参数设置未 transparent 表示透明背景。

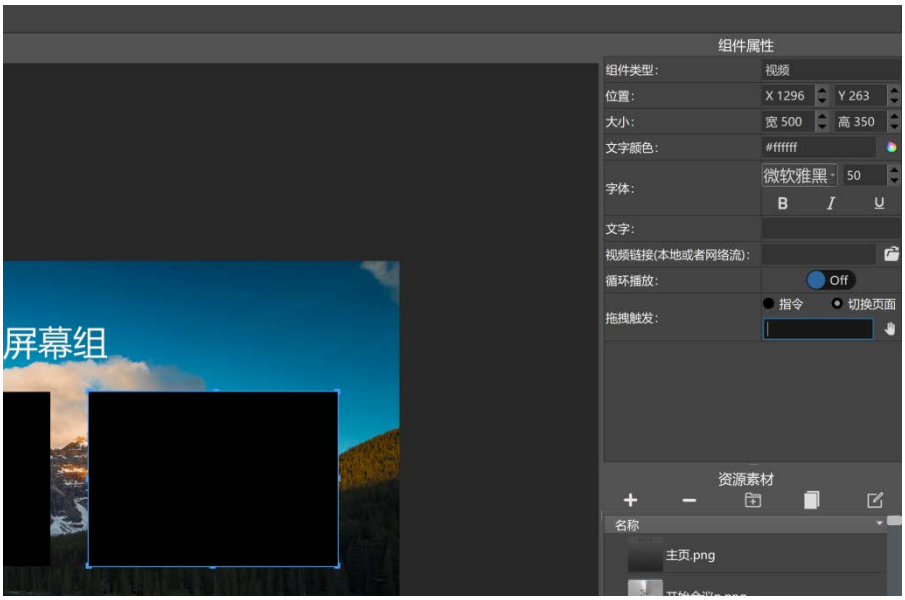
【每列数量】属性可设置每页列表显示的网络流源数量，注意不要设置太多，因为每页的流都是同时显示的，如果设置太多，会很吃性能，可能会导致卡顿。

【空白边缘】属性可设置单元之间的空白间隔。

【编辑流列表】属性可以添加配置流单元。

每个流单元是由名称、主流、辅流组成。预览显示的是辅流，切换成功后发送出去的切换数据是主流。

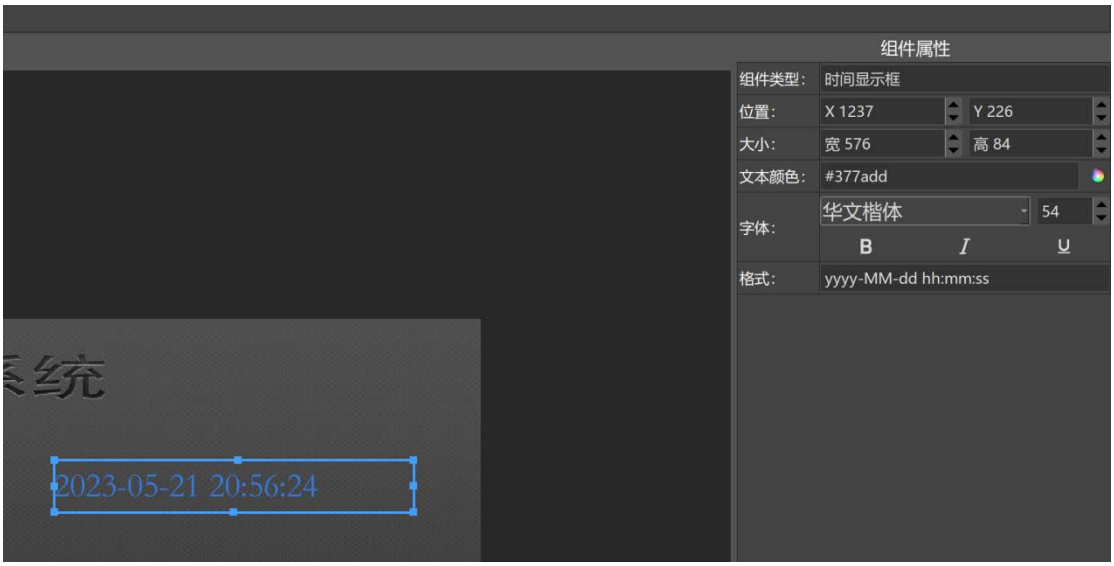
然后再按需求添加多个视频播放组件构成屏幕组。当流列表的单元拖拽到某一个视频播放组件后会触发一个变量指令，该指令在在【拖拽触发】添加绑定，在指令编辑器里编辑添加：



假设某分布式盒子的切换播放流链接的指令是“play=rtsp://192.168.1.12:8888/live”，则变量指令为“play=[1]”。绑定该指令到视频播放组件后，当组件接收到流列表的拖拽，会把该流单元的主流替换到变量指令的变量 ID 为“1”的位置，然后发送。

十五． 时间显示框

时间显示框可作为装饰作用添加到界面。可自定义时间的显示格式。



【格式】属性可定义时间显示的格式，规定如下：

“yyyy”代表年；

“MM”代表月；

“dd”代表日；

“hh”代表小时；

“mm”代表分钟；

“ss”代表秒；

“dddd”代表星期；

“ap”代表上午下午；

“<CR>”代表换行；

比如将格式改为“yyyy 年 MM 月 dd 日<CR>hh 时 mm 分 ss 秒 dddd ap”；效果如下：


组件属性
组件类型:
时间显示框
位置:
X 1236
Y 157
大小:
宽 660
高 144
文本颜色:
#377add
字体:
华文楷体
54
B
I
U
格式:
yyyy年MM月dd日<CR>hh时mm分ss秒 dddd ap
统
2023年05月21日
09时09分34秒 星期日 下午