VS-xxxxDN-EM控制板JSON功能解决方案

编制：邹媛审核批准：日期： 20180506

# 前言

## 目的

VS-xxxxDN-EM矩阵系列，包括VS-1616DN-EM矩阵、VS-3232DN-EM矩阵、VS-6464DN-EM矩阵，控制板能够快速获取矩阵的所有状态参数，并生成json格式的文件，用户可以通过命令或网络服务器获取对应的文件，并可以下载json文件到矩阵中，控制板按照下载的json文件配置矩阵。

此文档描述了JSON功能的相关需求和实现方式，为软硬件开发人员提供指导。

## 范围

适用于VS-xxxxDN-EM矩阵的软件设计人员，审核人员，评审人员，测试人员和项目负责人

## 编制依据

《混合矩阵内部协议》

“Configuration\_LiguoFormat\_Demo.JSON”文件

## 术语表

**表1:术语表**

|  |  |
| --- | --- |
| 名称 | 描述 |
| Linux\_ctrl\_board | VS-xxxxDN-EM\_CTRL智能控制板,主控制板，即替代原有设备中的控制板，使用MPC5125为主CPU，搭载Linux系统，用户操作设备的主要通道接口，拥有UART接口、并口、GPIO接口、网络接口等； |
| lig\_core\_demo | 控制板的主APP程序，负责所有功能的实现； |
| input\_board | 输入板，音视频信号输入板，板上对外有2/4/8个输入接口； |
| output\_board | 输出板,音视频信号输出板,板上对外有2/4/8个输出接口； |
| Configuration.JSON | 描述矩阵状态的文件，格式为json，固定名称； |
| metadata.json | 子板参数属性描述文件,格式为json，固定名称； |
| config\_log.json | 配置矩阵过程中的日志文件，格式为json，固定名称； |

# 项目概述

VS-xxxxDN-EM\_CTRL智能控制板（以下简称Linux\_ctrl\_board）,板上搭载Linux操作系统，实现多功能控制。

# 需求描述

获取矩阵所有状态信息，生成configuration.json文件，用户可以通过命令或网络服务器获取此文件。

用户可以通过KUpload把configuration.json文件下载到Linux\_ctrl\_board中，Linux\_ctrl\_board对比文件和当前矩阵的状态，配置矩阵，在配置过程中，需要记录配置过程中的错误信息到config\_log.json文件中。

用户可以通过命令或网络服务器获取config\_log.json文件。

# 方案设计

## 过程说明

### configuration.json初始化

* lig\_core\_demo启动；等待子板正常工作；
* 在“/nandflash/thttpd/www”文件夹中建立“configuration.json”文件（空洞文件），大小为1MB，并映射到config\_file\_mmap；
* lig\_core\_demo进入正常工作流程；JSON文件初始化标记设置为0；
* 在主轮询过程中添加建立JSON文件的过程调用，需要在没有用户命令时再处理获取过程；
* 向“configuration.json”文件写入文件头信息；数据格式参考“[4.2.1.1文件头描述](#_文件头描述)”；
* 向“configuration.json”文件写入矩阵信息；数据格式参考“[4.2.1.2文件数据-矩阵信息](#_文件数据-矩阵信息)”；
* 向“configuration.json”文件写入通讯信息；数据格式参考“[4.2.1.3文件数据-通讯信息](#_文件数据-通讯信息)”；
* 向“configuration.json”文件写入切换信息；数据格式参考“[4.2.1.4文件数据-切换信息](#_文件数据-切换信息)”；
* 向“configuration.json”文件写入子板信息；数据格式参考“[4.2.1.5文件数据-子板信息](#_文件数据-子板信息)”；
* 向“configuration.json”文件写入端口信息；数据格式参考“[4.2.1.6文件数据-端口信息](#_文件数据-端口信息)”；
* JSON文件初始化标记设置为1；

提供一条K3命令查询JSON文件是否准备好（在deviceobj中添加json\_file\_flag），如果返回NO，表示json文件正在初始化；

### configuration.json更新

* lig\_core\_demo按正常流程设置完成后，数据更新写入config\_file\_mmap对象；
* 矩阵基本信息更新，更新内容参考“[4.2.1.2文件数据-矩阵信息](#_文件数据-矩阵信息)”；
* 通讯接口信息更新，包括网络信息更新和串口参数信息更新；更新内容，参考“[4.2.1.3文件数据-通讯信息](#_文件数据-通讯信息)”；
* 切换状态和宏状态更新，更新内容，参考“[4.2.1.4文件数据-切换信息](#_文件数据-切换信息)”；
* 子板信息更新，当前，只有矩阵的默认EDID更新了，对应所有子板的默认EDID需要更新，参考“[4.2.1.5文件数据-子板信息](#_文件数据-子板信息)”；
* 端口信息更新，更新内容，参考“[4.2.1.6文件数据-端口信息](#_文件数据-端口信息)”；**需要注意关联参数，以及端口关联的情况；**

### configuration.json获取

* 查询配置文件是否初始化完成，K3000命令，“#EXT-CFGJSON-STAT?”，返回“~01@EXT-CFGJSON-STAT READY”表示在初始化过程中，此时获取JSON文件，不保证文件的正确性；返回“~01@EXT-CFGJSON-STAT OK”表示初始化完成，可以获取JSON文件，并且JSON文件正确；
* 获取配置文件，K3000命令,“#GET configuration.json”；
* 网页，在IE浏览器的地址栏中输入“<matrix\_IP>/configuration.json”；

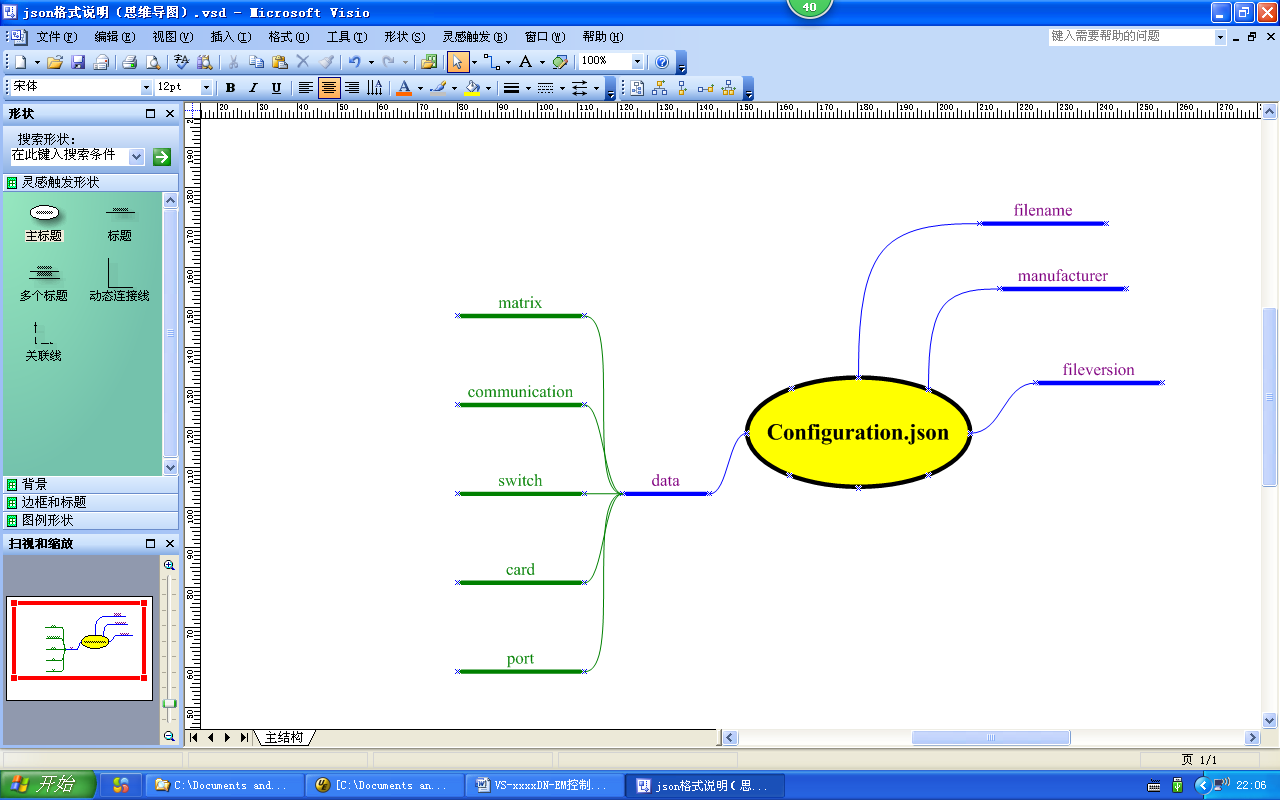
### configuration.json配置

* 用户通过KUpload加载“configuration.json”文件到Linux\_ctrl\_board；
* lig\_core\_demo按用户加载的配置文件配置矩阵；需要先获取用户加载配置文件的内容，与当前矩阵配置值进行对比，如果值不一样，则按用户加载的配置文件配置；
* 矩阵基本信息设置，写入数据库，更新内容参考“[4.2.1.2文件数据-矩阵信息](#_文件数据-矩阵信息)”；
* 通讯接口信息设置，写入数据库，包括网络信息更新和串口参数信息更新；更新内容，参考“[4.2.1.3文件数据-通讯信息](#_文件数据-通讯信息)”；
* 端口信息设置，直接通过LiguoA格式内部命令设置子板端口，参考“[4.2.1.6文件数据-端口信息](#_文件数据-端口信息)”；**需要注意关联参数，以及端口关联的情况;**依次读配置文件的端口对象，匹配端口所在子板是否一致，如果一致再配置端口；
* 完成配置后，重启lig\_core\_demo软件；

## JSON文件说明

JSON格式，是以“key”:“value”为一个基本单位，value可以是数值，字符串，也可以是JSON的对象，或是数组；

### configuration.json



configuration.json文件主要分为文件信息描述，即文件头描述，和数据，即data对象。

data对象由5个对象组成，分别是matrix对象，描述矩阵的基本信息；communication对象，描述矩阵通讯端口信息；switch对象，描述切换和切换预置信息；card对象，描述子板的相关信息；port对象，描述子板端口的相关信息；

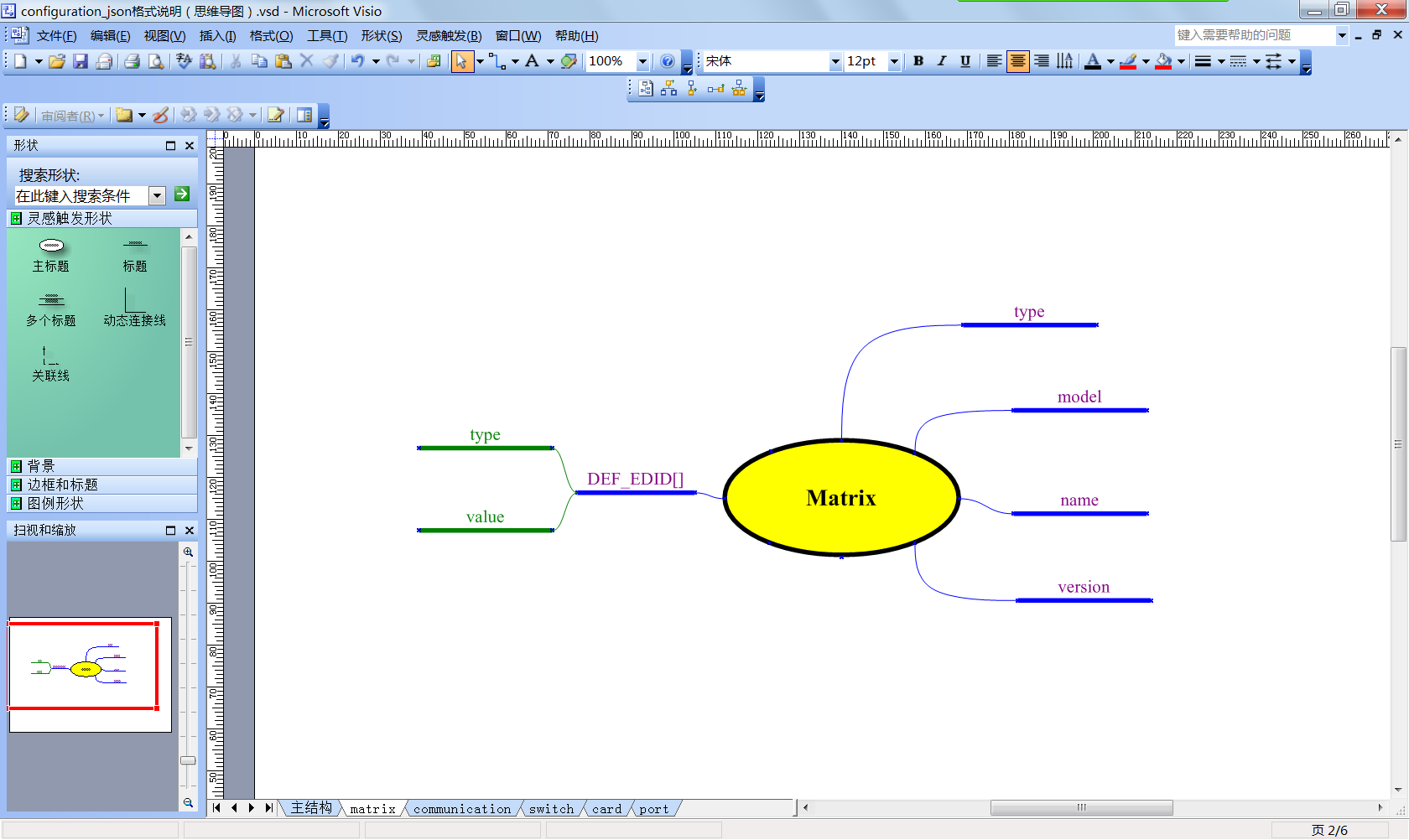
当configuration.json文件的格式错误时，比如缺少大括号等，提示错误“JSON\_ERROR\_100: -100, file format error, bad json;”；如果缺少data对象，提示错误“JSON\_ERROR\_106: -106, file format error, no data;”

#### 文件头描述

文件头描述由3个对象组成，分别是：

* "filename": "matrix configuration"，当前JSON文件功能名称，固定值，不能修改；如果修改，与lig\_core\_demo不匹配，则不执行配置操作，返回错误“JSON\_ERROR\_101: -101, filename mismatch; ”，退出配置过程；如果没有此项，则不执行配置操作，返回错误“JSON\_ERROR\_102: -102, file format error, no filename;”，退出配置过程；
* "fileversion": "1.0.0"，当前JSON文件版本，由lig\_core\_demo软件设置，固定值；lig\_core\_demo支持向下兼容，第一版本为1.0.0，如果当前lig\_core\_demo支持的JSON版本低于文件版本，则不执行配置操作，退出配置过程，返回错误“JSON\_ERROR\_103: -103, file version mismatch;”；如果没有此项，则不执行配置操作，返回错误“JSON\_ERROR\_104: -104, file format error, no fileversion;”，退出配置过程；
* "manufacturer": "Kramer Electronics Ltd."，当前JSON文件的用户或生产商名；lig\_core\_demo软件设置，可修改，配置操作忽略此对象值，但必须有此项，如果没有此项，则不执行配置操作，返回错误“JSON\_ERROR\_105: -105, file format error, no manufacturer;”，退出配置过程；

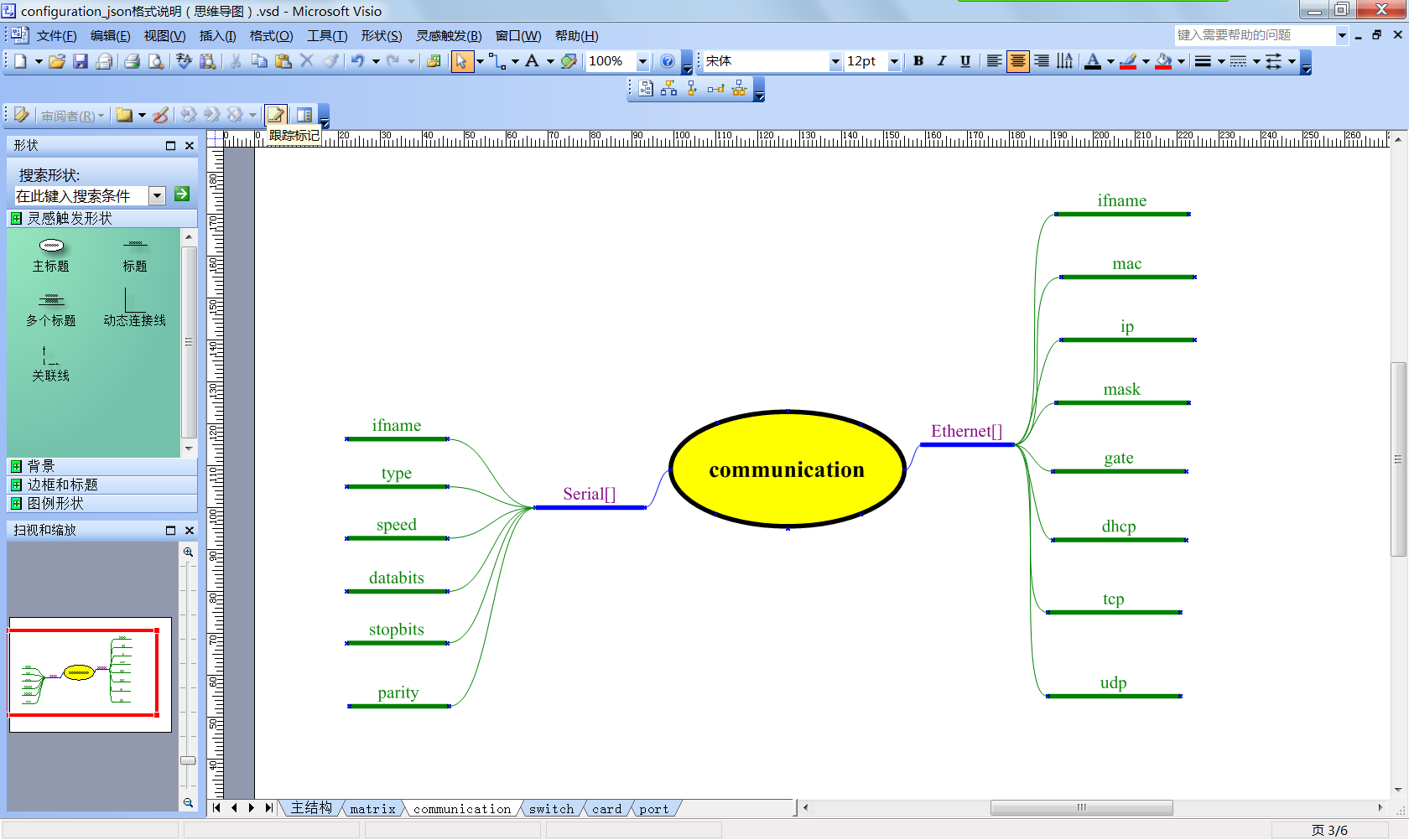
#### 文件数据-矩阵信息



matrix对象，包含当前矩阵的一些基本信息，分别为：

* "type": 16, 只读信息，描述当前矩阵是什么类型的，值为16/32/64；配置过程中，如果当前矩阵的值与此值不一致，则不执行配置操作，返回错误“JSON\_ERROR\_107: -107, Matrix Type mismatch; ”，退出配置过程；如果没有此项，则不执行配置操作，返回错误“JSON\_ERROR\_108: -108, file format error, no matrix type;”，退出配置过程；
* "model": "VS-1616DN-EM", 只读信息，描述矩阵model名，即“#MODEL?”命令查询的值；配置过程中，如果当前矩阵的值与此值不一致，则不执行配置操作，返回错误“JSON\_ERROR\_109: -109, Matrix Model mismatch; ”，退出配置过程；如果没有此项，则不执行配置操作，返回错误“JSON\_ERROR\_110: -110, file format error, no matrix model;”，退出配置过程；
* "name": "KRAMER\_0001", 读写信息，描述矩阵name名，即“#NAME?”命令查询的值；配置过程中，如果当前矩阵的值与此值不一致，将此值写入数据库，需要对此值进行格式检测，按“#NAME”命令设置检测，如果配置文件中值格式不正确，返回错误“JSON\_ERROR\_111: -111, Matrix Name format Error; ”，继续文件的配置过程；如果没有此项，继续文件的配置过程；
* "version": "05.02.0001",只读信息，描述产生这个文件矩阵系统的版本信息，可选属性，如果与当前矩阵版本不一致，返回错误“JSON\_WARN: Matrix Version mismatch; ”，继续文件的配置过程；如果没有此项，继续文件的配置过程；
* "DEF\_EDID":JSON对象数组，每个单元有两个JSON对象组成；由“[]”标记对象数组；此项用于控制板级别的默认EDID描述；详细用法参考“VS-xxxxDN-EM控制板默认EDID功能解决方案-V1.0”；如果没有此项或数组为空，继续文件的配置过程；
* "type":是EDF\_EDID对象的组成单元，用于描述当前单元的默认EDID是用于哪种信号源，值类型为字符串，现有"dvi"和"hdmi"两种；如果配置文件中值格式不正确，返回错误“JSON\_ERROR\_112: -112, Matrix DEF\_EDID\_TYPE format Error; ”，继续文件的配置过程,不需要再读对应"value"；如果没有此项，继续文件的配置过程，不需要再读对应"value"；
* "value":是EDF\_EDID对象的组成单元，用于描述当前单元的默认EDID值，值类型为字符串，由256个十六进制字符组成，即一共512个字符，小于0x10的值需要补“0”；只需要将值写入指定目录的bin文件，详细设置过程参考“VS-xxxxDN-EM控制板默认EDID功能解决方案-V1.0”；如果配置文件中值格式不正确，返回错误“JSON\_ERROR\_113: -113, Matrix DEF\_EDID\_TYPE\_DVI/HDMI EDID VALUE format Error; ”，继续文件的配置过程；如果没有此项，继续文件的配置过程；

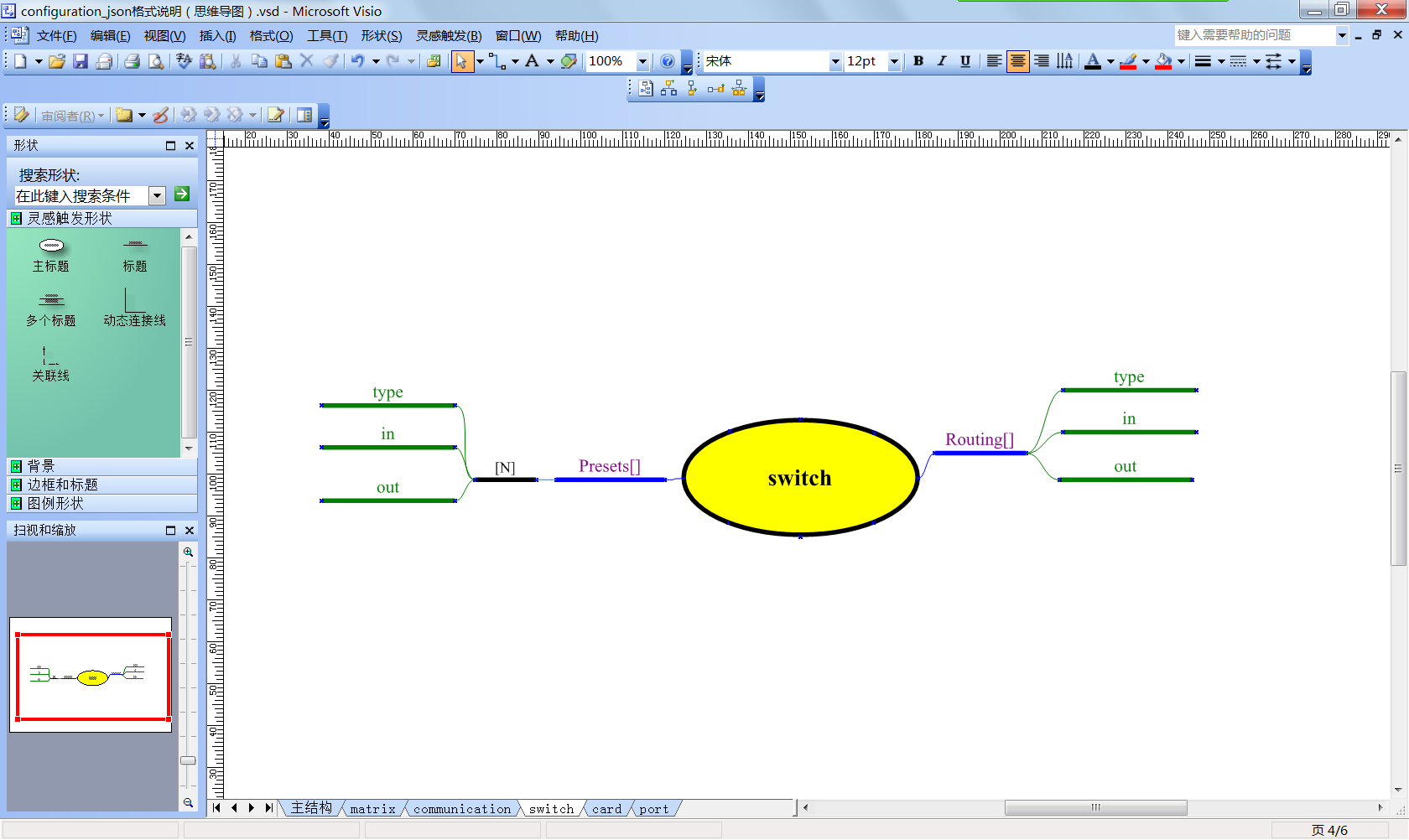
#### 文件数据-通讯信息



Communication对象，包括两个主要通讯对象，为Ethnet端口数组对象和Serial端口数组对象，详细信息为：

* "Ethernet":JSON数组对象，用于描述矩阵网络端口的相关信息；如果没有此项或数组为空，继续文件的配置过程；
* "ifname":"eth0",网络端口名，目前只支持设置eth0，固定名称，如果配置文件中值不正确，返回错误“JSON\_ERROR\_114: -114, Communication Ethernet name Error; ”，继续文件的配置过程,不需要再读其它配置项(mac、tcp等)；如果没有此项，返回错误“JSON\_ERROR\_115: -115, Communication Ethernet name unknown; ”，继续文件的配置过程，不需要再读其它配置项(mac、tcp等)；
* "mac":"40-30-6C-A2-45-59",只读信息，可选属性，描述网络端口的MAC地址；如果没有此项，继续文件的配置过程；
* "ip":"192.168.117.016",只读信息，可选属性，描述网络端口的IP地址；如果没有此项，继续文件的配置过程；
* "mask":"255.255.000.000",只读信息，可选属性，描述网络端口的MASK地址；如果没有此项，继续文件的配置过程；
* "gate":"192.168.000.001",只读信息，可选属性，描述网络端口的GATE地址；如果没有此项，继续文件的配置过程；
* "dhcp":0,数值型，读写属性，可选属性，取值范围[0,1]，值写入数据库；如果没有此项，继续文件的配置过程；如果值不对，返回错误“JSON\_ERROR\_116: -116, Communication Ethernet dhcp Error; ”，继续文件的配置过程；
* "tcp":5000,数值型，读写属性，可选属性，取值范围[1024,65535]，值写入数据库；如果没有此项，继续文件的配置过程；如果值不对，返回错误“JSON\_ERROR\_117: -117, Communication Ethernet tcp port Error; ”，继续文件的配置过程；
* "udp":50000,数值型，读写属性，可选属性，取值范围[1024,65535]，值写入数据库；如果没有此项，继续文件的配置过程；如果值不对，返回错误“JSON\_ERROR\_118: -118, Communication Ethernet udp port Error; ”，继续文件的配置过程；
* "serial":JSON数组对象，用于描述矩阵串口端口的相关信息；如果没有此项或数组为空，继续文件的配置过程；
* "ifname":"serial0",串口端口名，目前只支持设置serial 0（用户串口），固定名称，如果配置文件中值不正确，返回错误“JSON\_ERROR\_119: -119, Communication Serial name Error; ”，继续文件的配置过程,不需要再读其它配置项(speed、type等)；如果没有此项，返回错误“JSON\_ERROR\_120: -120, Communication Serial name unknown; ”，继续文件的配置过程，不需要再读其它配置项(speed、type等)；
* "type":"RS232",串口端口类型，目前只支持设置RS232，固定名称，如果配置文件中值不正确，返回错误“JSON\_ERROR\_121: -121, Communication Serial Type Error; ”，继续文件的配置过程,不需要再读其它配置项(speed等)；如果没有此项，返回错误“JSON\_ERROR\_122: -122, Communication Serial type unknown; ”，继续文件的配置过程，不需要再读其它配置项(speed等)；
* "speed":115200,数值型，读写属性，可选属性，取值范围{9600,19200,14400,38400,115200,576000}，值写入数据库；如果没有此项，继续文件的配置过程；如果值不对，返回错误“JSON\_ERROR\_123: -123, Communication Serial speed Error; ”，继续文件的配置过程；
* "databits":8,数值型，只读信息，可选属性，描述串口属性数据位长度；如果没有此项，继续文件的配置过程；
* "stopbits":1,数值型，只读信息，可选属性，描述串口属性数据停止位；如果没有此项，继续文件的配置过程；
* "parity":0,数值型，只读信息，可选属性，描述串口属性数据校验；如果没有此项，继续文件的配置过程；

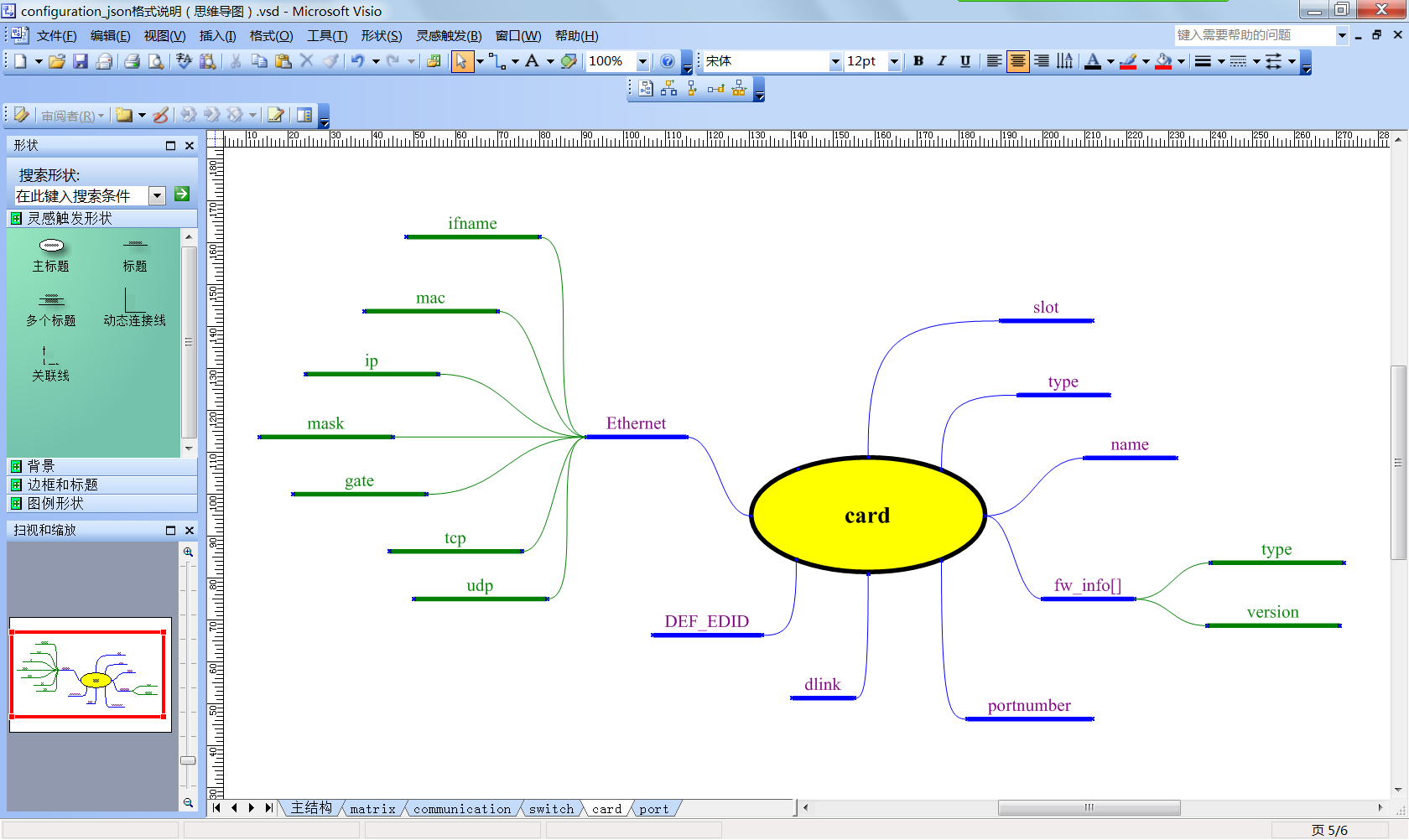
#### 文件数据-切换信息



Switch对象，包括两个主要对象，为Routing当前切换信息对象和Presets切换宏数组对象，详细信息为：

* "routing":JSON数组对象，用于描述矩阵当前切换信息；如果没有此项或数组为空，继续文件的配置过程；数组中单个JSON对象，必须由"in","out","type"组成，缺一不可；
* "in":1, 数值型，读写属性，切换的输入端口号，取值范围[0, input\_max\_munber]，值写入数据库；如果值不对，返回错误“JSON\_ERROR\_124: -124, Switch Routing input port id Error; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(out等)；如果没有此项，返回错误“JSON\_ERROR\_125: -125, Switch Routing input port idunknown; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(out等)；
* "out":1,数值型，读写属性，切换的输出端口号，取值范围[1, output\_max\_munber]，值写入数据库；如果值不对，返回错误“JSON\_ERROR\_126: -126, Switch Routing output port id Error; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；如果没有此项，返回错误“JSON\_ERROR\_127: -127, Switch Routing output port idunknown; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；
* "type":"v",读写属性，切换的操作类型，当前只有“v”，即视频；如果值不对，返回错误“JSON\_ERROR\_128: -128, Switch Routing type Error; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；如果没有此项，返回错误“JSON\_ERROR\_129: -129, Switch Routing typeunknown; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；
* "presets":JSON数组对象，用于描述矩阵当前预置切换宏信息；如果没有此项或数组为空，继续文件的配置过程；数组中单个JSON对象，必须由"in","out","type"组成，缺一不可；
* "N"：预置宏ID号，1～60；如果Key值小于1或是大于60，返回错误“JSON\_ERROR\_130: -130, Switch Presets ID Error; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；如果没有此项，继续文件的配置过程；
* "in":1, 数值型，读写属性，切换的输入端口号，取值范围[0, input\_max\_munber]，值写入数据库的宏记录表；如果值不对，返回错误“JSON\_ERROR\_131: -131, Switch Presets input port id Error; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(out等)；如果没有此项，返回错误“JSON\_ERROR\_132: -132, Switch Presets input port idunknown; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(out等)；
* "out":1,数值型，读写属性，切换的输出端口号，取值范围[1, output\_max\_munber]，值写入数据库的宏记录表；如果值不对，返回错误“JSON\_ERROR\_133: -133, Switch Presets output port id Error; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；如果没有此项，返回错误“JSON\_ERROR\_134: -134, Switch Presets output port idunknown; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；
* "type":"v",读写属性，切换的操作类型，当前只有“v”，即视频；如果值不对，返回错误“JSON\_ERROR\_135: -135, Switch Presets type Error; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；如果没有此项，返回错误“JSON\_ERROR\_136: -136, Switch Routing typeunknown; ”，继续文件的配置过程,不需要再读当前对象的其它配置项(in等)；

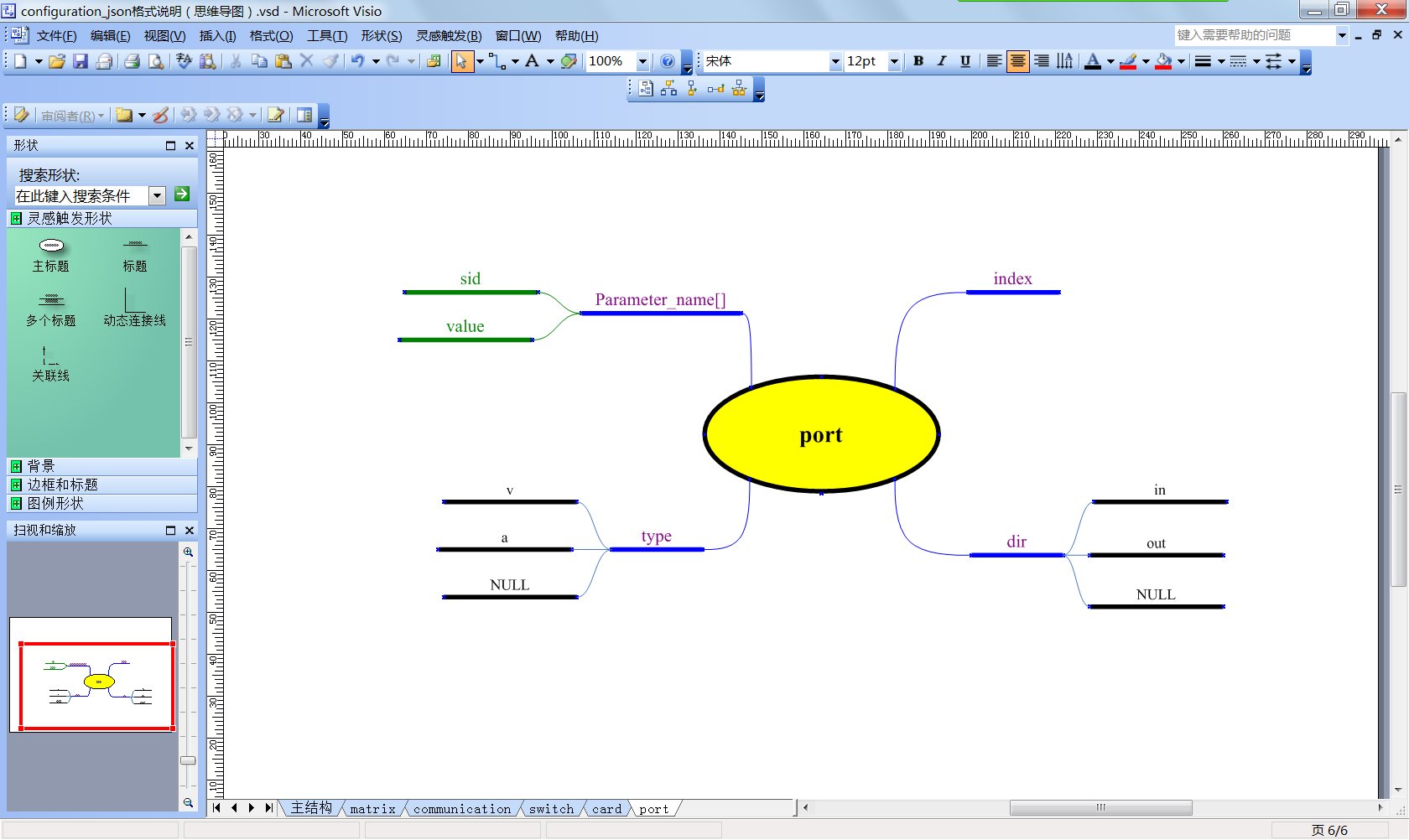
#### 文件数据-子板信息



Card对象，为JSON数组对象，独立一个JSON对象描述当前插槽子板信息，独立JSON对象由8个元素组成，卡对象，主要用于端口配置时的判断过程，无任何配置过程，详细信息为：

* "slot": 1, 数值型，只读属性，矩阵插槽号，正常取值范围[1,16]；如果没有此项，返回错误“JSON\_ERROR\_137: -137, Card slot ID unknown; ”；
* "type": 34, 数值型，只读属性，矩阵插槽号，正常取值范围[0,255]；如果此插槽上有板卡，但板卡出现未知异常，此值为“-1”；如果此插槽上有板卡，但板卡出现与控制板通讯异常错误，此值为“-2”；如果此插槽上没有板卡，此值为“-3”；如果没有此项，返回错误“JSON\_ERROR\_138: -138, Card TYPE ID unknown; ”；
* "name": "DT",只读属性，可选属性，子板英文名称，如果插槽上没有板卡（"type":-1），此属性为空字符；
* "fw\_info":JSON数组对象，只读属性，可选属性，用于描述子板固件版本号，如果插槽上没有板卡（"type":-1），此属性为空数组；
* "type":"cpu",固件类型，“cpu”指示主控制芯片固件版本；
* "version":"1.0.3"，对应芯片的版本号；
* "portnumber": 1/2/4/8,数值型，只读属性，描述子板上有几个端口，如果插槽上没有板卡（"type":-1），此属性为“-1”；如果有值错误，返回错误“JSON\_ERROR\_139: -139, Card port number unknown; ”；如果没有此项，返回错误“JSON\_ERROR\_140: -140, Card port number unknown; ”；
* "dlink": 0,只读属性，可选属性，描述子板是否是dlink；如果插槽上没有板卡（"type":-1），此属性为“-1”；
* "DEF\_EDID":子板默认EDID，只读属性，可选属性，描述子板的默认EDID；如果插槽上没有板卡（"type":-1），此属性为空；
* "Ethernet":子板的网络属性参数，为6G子板预留，可选属性，如果插槽上没有板卡（"type":-1），此属性为空；
* "ifname": "eth0",网络端口名；
* "mac": "40-30-6C-A2-6B-5A",网卡MAC地址；
* "ip": "192.168.2.101",网口IP地址；
* "mask": "255.255.0.0",网口MASK地址；
* "gate": "192.168.2.1",网口GATE地址；
* "tcp": 5000,TCP端口号；
* "udp": 50000，UDP端口号；

#### 文件数据-端口信息



Port对象，为JSON数组对象，独立一个JSON对象描述当前端口信息，在配置过程中，如果端口所在的子板信息与当前矩阵信息不能匹配（子板类型type编号不匹配）则抛出错误，返回错误“JSON\_ERROR\_141: -141, the card of the port mismatch; ”；

端口详细信息为：

* "index":1, 端口的物理编号，只读属性，取值范围为[1，32/64/128]；如果端口编号值错误，超出当前矩阵类型最大端口个数，返回错误“JSON\_ERROR\_142: -142, The index of the port unknown; ”,继续文件的配置过程,不需要再读当前对象的其它配置项；
* "dir":"in"/"out"，只读属性，指示当前端口输入输出方向；如果此端口不存在，即当前插槽上没有子板，此参数值为空；
* "type":"v"，只读属性，指示当前端口类型，当前只支持视频端口；如果此端口不存在，即当前插槽上没有子板，此参数值为空；
* "typeid":1，端口类型编号，只读属性，取值范围为[1，16/32/64]；表示当前视频输入或输出端口的编号，逻辑编号；如果此端口不存在，即当前插槽上没有子板，此参数值为“-1”；
* Parameter\_name对象：端口属性对象，即端口子参数，依据当前端口属性实际个数设置；key名为属性名称；value为JSON对象，由两个元素组成“sid”和“value”；
* "sid":81,子参数编号，依据LiguoA内部协议描述设置；EDID的sid为300；在配置时，如过子板类型匹配，实际矩阵端口不支持文件中设置的子参数，返回错误“JSON\_ERROR\_143: -143, Matrix port parameterdon’t support; ”,继续文件的配置过程；
* "value":50，对应子参数的值，除特殊几个子参数为字符型，如EDID，温度、电压等；其它均为数值型；配置时，如果当前矩阵的端口参数与文件不一致，则使用文件中的值设置端口，设置过程中，命令返回失败，返回错误“JSON\_ERROR\_144: -144, Matrix port parameterFailed; ”,继续文件的配置过程；如果没有此项，返回错误“JSON\_ERROR\_145: -145, Matrix port parameter no find value; ”；

例：

"Volume":{"sid":81,"value":50},

"EDID":{"sid":300,"value},

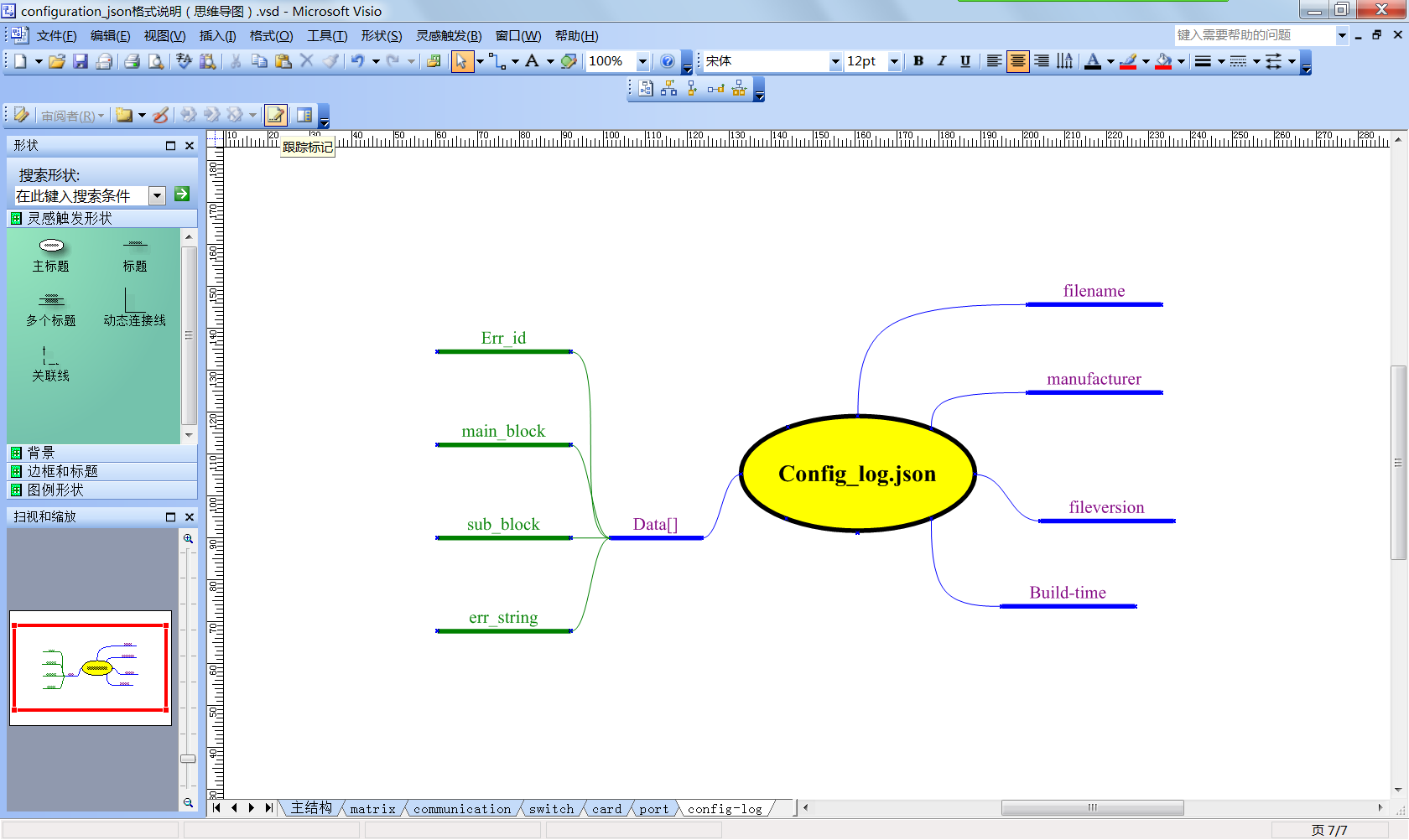
### metadate.json

子板参数属性描述文件,格式为json；存放在“/nandflash/thttpd/www”文件夹中，由外部加载更新，lig\_core\_demo不负责主动更新。

### config\_log.json

配置矩阵过程中的日志文件，格式为json。在配置过程中产生，记录错误信息，仅记录本次配置过程，在配置前，删除原有文件，再新建一个config\_log.json文件。存放在“/nandflash/thttpd/www”文件夹中。

格式为：



* "filename": "matrix configuration log"，当前JSON文件功能名称，固定值；
* "fileversion": "1.0.0"，当前JSON文件版本；
* "manufacturer": "Kramer Electronics Ltd."，当前JSON文件的用户或生产商名；
* "build-time":"2018-05-09 17.30.00",当前文件构建时间；
* "data":错误信息对象数组；
* "err\_id":-100,错误编号；
* "main\_block":"communication",错误产生于哪个块；
* "sub\_block":"Ethernet",错误产生于主块下的哪个子块；
* "err\_string":"xxxxx"，错误描述信息；

## 子板内部参数说明

子板内部参数，通过内部0x80­\_0x00命令获取当前的值，但是回送的值，除了当前参数值，还包括参数的一些其它信息，因此，需要通过移位和MASK过程才能获取真正的值。命令中参数子的bit7为0，拆分高低位；

通过内部0x80\_0x01命令设置子参数。

lig\_core\_demo.lig\_portobj.c文件中定义了一个端口子参数的解析结构体对象数组，记录格式化子参数的相关信息：

**#define** LIG\_POBJ\_SUBFUN\_ACTIVE\_NUMBER 95

LIG\_STRUCT\_SUBFUN\_INFO**const**

m\_lig\_pobj\_subfun\_info\_tag[LIG\_POBJ\_SUBFUN\_ACTIVE\_NUMBER]

**typedefstruct**

{

**int**subfun\_id;子参数ID号

**char** \* subfun\_name;子参数名称

**int**subfun\_value\_offset;子参数值偏移bit

**unsignedlong**subfun\_value\_mask;子参数屏蔽位

**int**subfun\_value\_size;子参数值长度，占几个BYTE

}LIG\_STRUCT\_SUBFUN\_INFO;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Parameter** | **sid** | **offset** | **mask** | **value\_size** | **备注** | **Ability** |
| 1 | Resolution Detect | 01 |  | Bit4 | 1byte | Val:  0 – auto, 1 – Force. | Set:  0 – auto, 1 – force, 2 – Do-auto-once. |
| 2 | Phase Mode | 02 |  | Bit4 | 1byte | Val:  0 – auto, 1 – force | Set:  0 – auto, 1 – force, 2 – Do-auto-once. |
| 3 | HDMI | 09 |  | Bit5-4 | 1byte | Val:  0 – auto,  1 – HDMI,  2 – DVI | Set:  Same with “Val” |
| 4 | Color Space | 10 |  | Bit1-0  (2nd byte) | 2 bytes | Val:  0 – auto,  1 – RGB444,  2 – YCbCr422,  3 – YCbCr444 | Set:  Same with “Val” |
| 5 | Deep Color | 11 |  | Bit2-0  (2nd byte) | 2 bytes | Val:  0 – auto,  1 –24bpp,  2 –30bpp,  3 –36bpp,  4 –48bpp | Set:  Same with “Val” |
| 6 | Pattern | 12 |  | Bit6-0 | 1 byte | Val: 1- 127 | Set:  Same with “Val” |
| 7 | Brightness | 20 |  | Bit6-0 | 1 byte | Val: 0- 63 | Set:  Same with “Val” |
| 8 | Contrast | 21 |  | Bit6-0 | 1 byte | Val: 0- 63 | Set:  Same with “Val” |
| 9 | Color | 23 |  | Bit6-0 | 1 byte | Val: 0- 63 | Set:  Same with “Val” |
| 10 | Sharpness | 24 |  | Bit6-0 | 1 byte | Val: 0- 63 | Set:  Same with “Val” |
| 11 | Phase adjustment | 30 |  | Bit6-0 | 1 byte | Val: 0- 127 | Set:  Same with “Val” |
| 12 | Reset | 36 |  | NULL | 1 byte | Val: Not available | Set:  1 – re-power,  2 - factory |
| 13 | Video-Freeze | 40 |  | Bit4 | 1byte | Val:  0 – Non-freeze, 1 – freeze | Set:  Same with “Val” |
| 14 | Picture Aspect Ratio | 43 | Mode1:  Bit7=0  (for 2 bytes) | Bit2-0  (2nd byte) | 2 bytes | Val:  0 – 1:1,  1 – 4:3,  2 – 16:9,  3 – 16:10,  4 – fill,  5 – 5:4,  6 – auto. | Set:  Same with “Val” |
| Mode2:  Bit7=1  (for 2 bytes) | Bit2-0  (2nd byte) | 2 bytes | Val:  0 –Stretch,  1 –Crop,  2 –Envelop, | Set:  Same with “Val” |
| 15 | Out Resolution | 45 |  | Bit6-0  (byte10) | 12 bytes | Val：0 - 127 | Set:  Same with “Val” |
| 16 | Hor-Total Pixels | 64 |  | Bit6-0(byte1)  Bit6-0(byte2) | 2 bytes | Val = byte1\*128 + byte2 | Set:  Same with “Val” |
| 17 | Horizontal-Start | 67 |  | Bit6-0(byte1)  Bit6-0(byte2) | 2 bytes | Val = byte1\*128 + byte2 | Set:  Same with “Val” |
| 18 | Horizontal-Active | 68 |  | Bit6-0(byte1)  Bit6-0(byte2) | 2 bytes | Val = byte1\*128 + byte2 | Set:  Same with “Val” |
| 19 | Vertical-Start | 69 |  | Bit6-0(byte1)  Bit6-0(byte2) | 2 bytes | Val = byte1\*128 + byte2 | Set:  Same with “Val” |
| 20 | Vertical-Active | 70 |  | Bit6-0(byte1)  Bit6-0(byte2) | 2 bytes | Val = byte1\*128 + byte2 | Set:  Same with “Val” |
| 21 | Save Timing Para | 73 |  | Bit6-0 | 1 byte | Val: 0-127 | Set:  Same with “Val” |
| 22 | Recall Timing Para | 74 |  | Bit6-0 | 1 byte | Val: 0-127 | Set:  Same with “Val” |
| 23 | Switch Speed | 75 |  | Bit1-0  (2nd byte) | 2 bytes | Val:  0 – Normal,  1 – Fast,  2 – Ex-Fast,  3 – Seamless, | Set:  Same with “Val” |
| 24 | Volume | 81 |  | Bit6-0 | 1 byte | Val: 0-70 | Set:  Same with “Val” |
| 25 | Audio Balance | 84 |  | Bit6-0 | 1 byte | Val: 0-100 | Set:  Same with “Val” |
| 26 | Audio Bass | 87 |  | Bit6-0 | 1 byte | Val: 0-100 | Set:  Same with “Val” |
| 27 | Audio Treble | 88 |  | Bit6-0 | 1 byte | Val: 0-100 | Set:  Same with “Val” |
| 28 | Audio Mute | 91 |  | Bit0 | 1 byte | Val:  0 – MUTE，  1 – UN-MUTE | Set:  Same with “Val” |
| 29 | Audio Mono | 94 |  | Bit6-0 | 1 byte | Val:  0 – UN-Mono,  1 – Mono. | Set:  Same with “Val” |
| 30 | Audio Select | 98 |  | Bit1-0 | 1 byte | Val:  0 – auto,  1 – AUD-Embedded,  2 – AUD-Ex-Digital,  3 – AUD-Ex-Analog | Set:  Same with “Val” |
| 31 | Temprature | 125 |  | Bit6-0  (2nd byte) | 4 bytes  /group | Val: 0 – 127 | Not available. |
| 32 | Voltage | 126 |  | Bit6-0  (4th byte) | 6 bytes  /group | Val: 0 – 127 | Not available. |
| 33 | R-offset | 152 |  | Bit6-0  (2nd byte) | 2 bytes | Val: 0 – 63 | Set:  Same with “Val” |
| 34 | G-offset | 153 |  | Bit6-0  (2nd byte) | 2 bytes | Val: 0 – 63 | Set:  Same with “Val” |
| 35 | B-offset | 154 |  | Bit6-0  (2nd byte) | 2 bytes | Val: 0 – 63 | Set:  Same with “Val” |
| 36 | R-gain | 155 |  | Bit6-0  (2nd byte) | 2 bytes | Val: 0 – 63 | Set:  Same with “Val” |
| 37 | G-gain | 156 |  | Bit6-0  (2nd byte) | 2 bytes | Val: 0 – 63 | Set:  Same with “Val” |
| 38 | B-gain | 157 |  | Bit6-0  (2nd byte) | 2 bytes | Val: 0 – 63 | Set:  Same with “Val” |
| 39 | OUT\_HSYNC\_INVERT | 152 |  | Bit6-0  (2nd byte) | 2 bytes | Val:  0 – Invert  1 – Not-Invert | Set:  Same with “Val” |
| 40 | OUT\_VSYNC\_INVERT | 153 |  | Bit6-0  (2nd byte) | 2 bytes | Val:  0 – Invert  1 – Not-Invert | Set:  Same with “Val” |
| 41 | HDCP | 155 |  | Bit6-0  (2nd byte) | 2 bytes | Val:  0 – HDCP\_Enable  1 – HDCP\_Disable | Set:  Same with “Val” |
| 42 | XTRA | 156 |  | Bit6-0  (2nd byte) | 2 bytes | Val:  0 – XTRA\_Enable  1 – XTRA\_Disable | Set:  Same with “Val” |
| 43 | ST | 158 |  | Bit6-0  (2nd byte) | 2 bytes | Val:  0 – Analog\_AUD\_In  1 –Analog\_AUD\_Out  2 – Analog\_Out\_From\_ARC | Set:  Same with “Val” |
| 44 | InputHDCP | 157 |  | Bit6-0  (2nd byte) | 2 bytes | Val:  0 – HDCP\_Enable  1 – HDCP\_Disable | Set:  Same with “Val” |
| 45 | BlueScreen | 160 |  | Bit6-0  (2nd byte) | 2 bytes | Val:  0 –MUTE (output blue)  1 –UN-MUTE | Set:  Same with “Val” |
| 46 | PIP-enable(Menu) | 161 |  | Bit6-0  (2nd byte) | 2 bytes | Val:  0 – PIP\_Enable,  1 – PIP\_Disable,  2 – PIP\_En5S\_ThenDis | Set:  Same with “Val” |
| 47 | AUTO SYNC OFF | 未找到 |  |  |  |  |  |
| 48 | SDI\_AUDIO | 未找到 |  |  |  |  |  |

## lig\_core\_demo程序说明

lig\_core\_demo.lig\_deviceobj.c文件中，定义了“g\_lig\_device”结构体，在此结构体中添加json文件初始化标记；此模块还提供所有矩阵信息的内部结构；可通过此模块获取端口和子板的相关信息；

lig\_core\_demo.lig\_portobj.c模块，提供端口子参数分析函数；

lig\_core\_demo.lig\_boardobj.c模块，提供子板信息的分析函数；

# **异常处理**

## 异常No.1

# **约束**

## 响应速度

* #GET和网络服务器，必须在1秒内能够返回数据；
* 当矩阵参数改变时，必须在3秒内更新configuration.json文件；

# **附录**

## 实验 mmap 文件映射内存

如何使用mmap功能？

控制板系统是否支持文件映射内存功能mmap？

thttpd能否访问共享文件？

读写文件同步？

写JSON文件的快速定位和修改方式？文件插入字符是否会覆盖后面的内容？

空洞文件的建立：

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdlib.h>

#define ERR\_EXIT(m) \

do \

{ \

perror(m); \

exit(EXIT\_FAILURE); \

} while(0)

int main(void)

{

int fd;

int ret;

fd = open("hole.txt",O\_WRONLY|O\_CREAT|O\_TRUNC,0644);

if(fd == -1)

ERR\_EXIT("open error");

write(fd,"hello",5);

ret = lseek(fd,1024\*1024\*1024,SEEK\_CUR);

if(ret == -1)

ERR\_EXIT("lseek error");

write(fd,"world",5);

close(fd);

return0;

}

## 实验powerpcjson.so库的使用

是否能直接使用文件流？

在C工程中使用C++库，编译是否有什么限制？

lig\_core\_demo工程，由C工程转为C++工程；

## Error Code描述

0 - 正确，无异常；

-1 ～-99 - 端口类型或子板类型错误标记；

-100～-199 - 文件错误相关；

|  |  |  |  |
| --- | --- | --- | --- |
| id | 错误信息err string | 描述 | 后续的动作 |
| -1 | "indicate card error" | 用户下载的配置文件指示的插槽中的卡为工作异常 | continue |
| -2 | "indicate card communication error" | 用户下载的配置文件指示的插槽中的卡为通讯异常 | continue |
| -3 | "outline " | 用户下载的配置文件指示的插槽中无卡 | continue |
| -4 | "parameter error" | 远程参数值错误/超界等 | continue |
| -5 | "parameter different" | 个别的参数不同需要记录(card) | continue |
| -6 | "don't find the key" | 在本地找不到某个key | continue |
| -7 | "set parameter error" | 设置参数时错误 | continue |

|  |  |  |  |
| --- | --- | --- | --- |
| id | 错误信息err string | 描述 | 后续的动作 |
| -100 | file format error, bad json; | json格式错误 | exit |
| -101 | "file header error;" | json文件头错误 | exit |
| -102 | "remote lack of key;" | 远程缺少关键key | exit |
| -103 | "local lack of key;" | 本地缺少关键key | exit |
| -104 | "remote parameter type error" | 远程参数类型错误 | continue |
| -105 | "local parameter type error" | 本地参数类型错误 | continue |