家居智能照明编译器

输入文件格式

文件审批：

	部门/职位	签名	日期
编制			
编制			
会签			
会签			
审核			
批准			

文档修改记录：

版本	修改内容概要	修改人	批准人	生效日期
V0.1	添加整体目录	李宇哲		2021/8/24
V0.2	取消常量表，区域表，新增负载标识字段，新增值描述字段	李宇哲		2021/8/30
V0.3	变量表 名称调整区域名称，负载类型	李宇哲		2021/9/1
V0.4	文件结构调整	李宇哲		2021/12/8
V0.5	增加了内置变量	王春华		2021/12/13
V0.6	1、 变量表、函数表、产品表增加了对应家居 3.0 的数据结构 2、 增加了使用到的公共函数和家居函数	王春华		2021/12/15
V0.7	1、修改函数码值，与 3.0 保持	王春华		2021/12/16

	一致			
V0.8	1、增加 2.6 和 2.7	王春华		2021/12/16
V0.9	1、增加 UTF8 编码要求的描述	李宇哲		2021/12/17
V0.10	1、增加 ulc 生成多少的字段	李宇哲		2021/12/20
V0.11	1、 新增家居工程文件 2、 编译配置信息移到 家居工程文件 3、 多 ulc 每个 ulc 对应的语句列表配置 放到家居工程文件 4、 在家居用户文件里增加 区域表 5、 变量表的区域名称改为 区域编号	王春华		2021/12/21
V0.12	1、输入文件的语句表增加语句编号 2、术语定义表新增术语类型	李宇哲		2021/12/27
V0.13	1、 区域表的字段调整 2、 工程文件字段改成 UlcListInfo	李宇哲		2021/12/29
V0.14	1、 家居用户文件 语句表 的编号可以为空 2、 家居工程文件 多 ulc 每个 ulc 对应的语句列表配置 可为空 3、 家居工程文件 家居编译配置去除 语句生成 ulc 上限 4、 语句表可为空	王春华		2021/12/29
V0.15	1、更新术语字段定义	王春华		2021/12/31
V0.16	1、 插入更新后的输入 3 个文件 2、 文档的字段描述和 3 个文件字段一一对齐	李宇哲		2022/1/6
V0.17	1、用户文件的表字段描述调整，有些不能是数字或者操作符为名称.例如产品名称，产品属性名称，场景名称，时间段名称等	李宇哲		2022/1/18

V0.18	1、语句表格式描述更正	李宇哲		2022/1/20
V0.19	字段更正	李宇哲		2022/3/30
V0.20	增加值描述功能对应字段	李宇哲		2022/04/21
V0.21	增加设备切换表和自适应场景表	李宇哲		2022/04/24
V0.22	调整家居业务数据的字段描述	李宇哲		2022/04/26
V0.23	删除场景表，自适应场景表进行修改，设备表字段调整	李宇哲		2022/04/26
V0.24	字段名称调整	李宇哲		2022/04/26
V0.25	函数的参数值描述字段调整	李宇哲		2022/04/29
V0.26	增加表和术语的关系描述	李宇哲		2022/05/07
V0.27	批处理函数英文名由 BatchFunction 改为 HomeRemoteControl	王春华		2022/07/12
V0.28	1、 修改切换函数入参 2、 新增 检查自适应场景行函数 3、 修改 场景函数 第一个入参 int8 为 int16 4、 修改 2.4 内置变量 与 智能家居管理系统 V4.01.00_Dev0215 5、 修改了 2.1 的数据类型 6、 修改了 2.3 的转换关系	王春华		2022/07/15
V0.29	1、 2.5.1 公共函数的函数 Key 修改为从 0 到 5 2、 2.5.2 应用函数的函数码值改为函数 Key 3、 2.5.2 应用函数的函数 Key 修改为从 6 到 21 4、 编译配置完善，增加默认字段	王春华		2022/08/30
V0.30	1、 增加单个和多个的 ulc 编译配置限制 SH_CONFIG_MAX_ALL_ULCSEGBYTES、SH_CONFIG_MAX_ULCSEGBYTES	李宇哲		2022/09/01
V0.31	1、标准编译配置 备注更新	李宇哲		2022/09/14

一、 智能家居编译输入文件

注意：输入文件必须都是 UTF8 编码。

1.1 家居用户文件

家居用户文件以 txt 格式进行数据构造，格式如下文件：

```
// 双斜杠在行首表示此行是行注释
//如果要用到|字符的，用加转义字符的方式\|来表示，

// 表名由双尖括号（注意不是书名号）包裹
<<表名 A>>
// 表名下方第一行非空行表示表字段列表，用竖划线分隔
表字段 1 | 表字段 2  | …
// 字段列表下方每个非空行是一条记录，多个属性值用竖划线分隔
属性值 1 | 属性值 2  | …
…
// 遇到下一个表名，或遇到文件结束表示上一张表的结束
```

1.1.1 变量表格式（表不可为空）

```
<<变量表>>
变量名称  |设备名称  |产品名称 |产品内属性名称 | 唯一标识 | 区域编号 | 负载标识
```

表字段名称	是否必填	限制	对应家居 3.0 的表字段
唯一标识	是	必须填写，唯一标识允许使用任意非空字符组成标识，是全局唯一	家居工程配置表（ProjectConfig）：设备配置表（DeviceConfig）的序号（Ordinal）
变量名称	是	任意非空字符，不能为纯数字，不能包含空白字符，不能包含运算符 。是全局唯一	家居工程配置表（ProjectConfig）：设备配置表（DeviceConfig）的设备名称（name）
区域编号	是	变量的区域编号，数字。如果设备没有	家居工程配置表（ProjectConfig）：根

		区域，则填 -1	据设备配置表（DeviceConfig）的区域编号（RegionNo）
负载标识	是	负载类型的标识，非空字符，如果没有负载类型，则填-1	家居工程配置表（ProjectConfig）：根据设备配置表（DeviceConfig）的负载设备类型（LoadDeviceType），查找终端设备类型定义表，获取相应的名称
设备名称	是	非空字符，如果没有设备名称，则填-1	家居工程配置表（ProjectConfig）：产品配置表（ProductConfig）的产品名称（ProductName）
产品名称	是	非空字符，如果没有产品名称，则填-1	家居产品定义表（ProjectDefine）：产品定义表（ProductDefine）的产品名称（ProductName）
产品内属性名称	是	非空字符，如果没有产品内属性名称，则填-1	家居产品定义表（ProjectDefine）：设备定义表（DeviceDefine）的设备名称（DeviceName）

注：运算符：@|☺;|,|-|+|*|\|==|!=|>=|<=|<|>|=|(|)|\!|

内置的变量：请参考附录 2.4

1、表的作用：变量名称和相应信息的关联关系。
2、目前关联术语：在术语描述中，有显示【变量名称】的术语。详见《《家居智能照明编译器 V4.01.01》术语转换规则》文档

1.1.2 时间段表（表可为空）-

```
<<时间段表>>
名称    | 时间段
早晨    | 8:00:00-12:00:00
```

表字段名称	是否必填	限制
名称	是	必须填写，任意非空字符，不能为纯数字，不能包含空白字符，不能包含运算符
时间段	是	非空字符，格式 00:00:00-00:00:00

1、表的作用：对需要将术语和时间段进行管理的术语，需要该表进行定义。
2、目前关联术语：早晨。详见《《家居智能照明编译器 V4.01.01》术语转换规则》文档

1.1.3 语句表（表可为空）

```
<<语句表>>
语句编号 | 条件 | 结果
```

表字段名称	是否必填	限制
语句编号	否	非必填，数字，如果不填写该字段按照默认值，从 0 开始
条件	是	必须填写，非空字符
结果	是	必须填写，非空字符

注：编译阶段该表为空时，给出警告信息。拆分语句阶段该表可以为空。

1、表的作用：编写标准语句或术语语句

1.1.4 区域表（表可为空）

```
<<区域表>>
区域编号 | 区域名称
1 | 客厅
```

表字段名称	是否必填	限制
区域编号	是	必须填写，数字
区域名称	是	必须填写，任意非空字符，不能为纯数字，不能包含空白字符，不能包含运算符

1、 表的作用：将该表的区域编号和变量表中的区域编号进行关联,达到一个区域名称可以关联多个变量名的作用。

2、目前关联术语：灯亮，灯灭。详见《《家居智能照明编译器 V4.01.01》术语转换规则》文档

1.1.5 自适应场景表（表可为空）

```
<<自适应场景表>>
表名称 | 表编号 | 行名称 | 行编号
客厅照明 | 1 |  |
客厅照明 | 1 | 照明场景 1 | 2
```

表字段名称	是否必填	限制
表名称	是	必须填写，任意非空字符，不能为纯数字，不能包含运算符
表编号	是	必须填写，数字

行名称	否	可以为空字符，不能为纯数字，不能包含运算符
行编号	否	可以为空字符，数字

1、 表的作用：通过该表的表名称和行名称来获取表编号，行号信息。
2、目前关联术语：上切换，下切换，执行自适应场景，自适应场景行成立。详见《《家居智能照明编译器 V4.01.01》术语转换规则》文档

1.2 家居工程文件

ConfigInfo	编译配置信息	
	BasicConfig	标准编译配置
	ShConfig	家居编译配置
UlcFileInfo	每个 ulc 对应的语句列表配置	
	ulcId	Ulc 编号
	logicNoList	依赖的语句编号列表
DeviceTable	设备表	
	deviceName	设备名称
	nodeNo	节点编号
	nodeInnerProductNo	节点内产品编号
	channelNo	通道编号
	hostNo	主机号

1.2.1 每个 ulc 对应的语句列表配置（可为空）

```
" UlcFileInfo":[
        {
            "ulcId": "0",
            "logicNoList": [],
        }
]
```

Ulc 编号	依赖的语句编号列表
0	1,5,9
1	2,3,4,6,7,8
…	…

注：为空时，默认生成一个 ulc 文件

1.2.2 编译配置信息

1.2.2.1 标准编译配置（可为空）

```
{
    “BasicConfig”: {
        “U_CONFIG_VM_FILE_TYPE” : 0,
        “U_CONFIG_VM_FILE_VERSION” : “4.1”,
        “U_CONFIG_VM_FILE_VAR” : “0”,
        “U_CONFIG_VM_FILE_FUN” : “0”,
        “U_CONFIG_LOGIC_NO” : “0”

    }
}
```

<table>
<tr><th colspan="4">BasicConfig 字段数据</th></tr>
<tr><td>配置项</td><td>取值范围</td><td>默认值</td><td>说明</td></tr>
<tr><td>U_CONFIG_VM_FILE_TYPE</td><td>虚拟机文件类型</td><td>默认填 0</td><td>编译生成的可执行文件的类型。暂时只支持生成 ULC 字节码。</td></tr>
<tr><td>U_CONFIG_VM_FILE_VERSION</td><td>虚拟机文件格式版本号</td><td>默认填“4.1”</td><td>一般填默认值</td></tr>
<tr><td>U_CONFIG_VM_FILE_VAR</td><td>变量打包配置</td><td>默认填 0</td><td>Bit0 表示是否打包变量表。
Bit1 表示是否打包变量配置表。
1：打包；0：不打包。
Bit0=1 时，虚拟机会根据变量表对执行过程中的变量数据进行检查；否则不检查。
Bit1=1 时，虚拟机 SDK 可以通过变量 id 获取变量唯一标识，通过变量唯一标识获取变量 id。
<table>
<tr><td>值</td><td>Bit1</td><td>Bit0</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>2</td><td>1</td><td>0</td></tr>
<tr><td>3</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>U_CONFIG_VM_FILE_FUN</td><td>函数打包配置</td><td>默认填 0</td><td>Bit0 表示是否打包函数表、函数索引表。
Bit1 表示是否打包函数配置表。
1：打包；0：不打包。
Bit0=1 时，虚拟机会根据函数表对执行过程中的函数数据进行检查；否则不检查。
Bit1=1 时，虚拟机 SDK 可以通过函数 id 获取函数唯一标识，通过函数唯一标识获取函数 id。<table><tr><td>值</td><td>Bit1</td><td>Bit0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>2</td><td>1</td><td>0</td></tr><tr><td>3</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>U_CONFIG_LOGIC_NO</td><td>语句行号打包配置</td><td>默认填 0</td><td>1：打包；0：不打包。</td></tr>
</table>

注：为空时，取默认值

1.2.2.2 家居编译配置（可为空）

```
{
    “ShConfig”：{
            "SH_CONFIG_LOGOUT_LEVEL": 0,
            "SH_CONFIG_MAX_LOGIC_SEG_BYTES":0,
            "SH_CONFIG_MAX_LOGIC_TRIGGERED":0,
            "SH_CONFIG_MAX_ULCSEGBYTES":0,
            "SH_CONFIG_MAX_ALL_ULCSEGBYTES":0,
            "SH_CONFIG_MAX_LOGICCOUNT":0,
            "SH_CONFIG_MODEL":"Debug"
        }
}
```

BasicConfig 字段数据			
配置项	取值范围	默认值	说明
SH_CONFIG_LOGOUT_LEVEL	日志信息等级开关，0-3	0	编译日志等级开关。 Debug = 0; Info = 1; Warning = 2; Error = 3 编译日志会根据此处设置的等级值进行输出。只会输出等级值大于等于设置的值的日志。
SH_CONFIG_MAX_LOGIC_SEG_BYTES	语句长度限制	0-3072	单条语句分段后（条件、结果）的

			字节码长度限制。0 表示不做限制。该限制与低性能环境运行时内存有关。
SH_CONFIG_MAX_LOGIC_TRIGGERED	同时触发语句总数限制（单个事件同时触发的语句数）	500	变量变化导致语句同时触发的总数的限制。0 表示不做限制。该限制与低性能环境运行时内存有关
SH_CONFIG_MAX_ULCSEGBYTES	单个 Ulc 文件大小上限	262144	单个 Ulc 文件字节大小。
SH_CONFIG_MAX_ALL_ULCSEGBYTES	全部 Ulc 文件大小上限	262144	全部 Ulc 文件字节大小。
SH_CONFIG_MAX_LOGICCOUNT	语句总数上限	500	语句表内语句的上限。
SH_CONFIG_MODEL	Debug/Release	Debug	字符串类型，编译模式

注：为空时，取默认值

1.2.3 设备表(可为空)

```
{
        "DeviceTable": [
         {
             "deviceName": "客厅西开关",
             "nodeNo":0,
             "nodeInnerProductNo":1,
             "channelNo":2,
             "hostNo":3
         },
         {
             "deviceName": "客厅东开关",
             "nodeNo":1,
             "nodeInnerProductNo":1,
             "channelNo":2,
             "hostNo":3
         }
    ]
}
```

DeviceTable 字段数据			
配置项	取值范围	数据类型	说明
deviceName	设备名称	字符串	任意非空字符，不能为纯数字，不能包含空白字符，不能包含运算符
nodeNo	节点编号	整型	
nodeInnerProductNo	节点内产品编号	整型	

channelNo	通道编号	整型	
hostNo	主机号	整型	

1、 表的作用：通过该表的设备名称获取设备信息。
2、目前关联术语：左切换，右切换。详见《《家居智能照明编译器 V4.01.01》术语转换规则》文档

1.2 家居系统文件

家居术语定义文件以 json 格式进行数据构造，格式如下文件：

GlossaryInfo	术语信息	
	GlossaryDefine	术语规则表
	LuaData	Lua 脚本内容
ShInfo	家居业务数据	
	FunctionDefine	家居函数表
	ProductAttributeDefine	家居产品设备定义表

1.3.1 术语信息

1.3.1.1 术语规则表

1、glossaryDefine 格式字段定义如下：

```
    “ glossaryDefine”: [
            {
                    “level”: 0,
                   ″category″: 0,
                   ″type″: 12,
                    “name”: “或”,
                    “logicRegularMatch”: “或”,
                    “logicConversionRules”: “\″%s\″,\″||\″”,
                        “tokenRegularMatch”: “(或)”,
                    “tokenConversionRules”:
    “\″%s\″,getTokenList(sz[1],\″LogicGlossary\″)”
            },
              {
                 “level”:”1”,
                ″ category ″: 1,
                ″type″: 12,
```

```
                “name”:”等于”,
                “logicRegularMatch”:”(.+)等于(.+)”
                “logicConversionRules”:”
    \"%s\",coverOperator(sz[1],sz[2],\"==\")”,
                    “tokenRegularMatch”: “(.+)(等于)(.+)”,
        “tokenConversionRules”:“\"%s\",getTokenList(sz[1],\"Variable\",sz[2],\"
RelayGlossary\",sz[3],\"Identifer\")”

            }
    ]
```

字段名	字段含义	数据结构
glossaryDefine	术语定义组	vector<glossary>
glossary 字段数据		
name	术语名称	String
level	术语优先级	int 全局唯一
category	术语类别	0 切词类术语 1 匹配类术语
logicRegularMatch	术语转标准语句正则匹配规则	String
logicConversionRules	术语转标准语句转换规则	string
tokenRegularMatch	术语拆分的正则匹配规则	String
tokenConversionRules	术语拆分转换规则	string
type	术语类型	string

字符串名称	含义
Identifer	未知类型
Variable	变量
Const	常量
DelayGlossary	延时术语
CancelDelayGlossary	取消延时术语
TimeDotGlossary	时间点术语
TimeRangeGlossary	时间段术语
WeekGlossary	星期术语
HolidayGlossary	节假日术语
LightOnGlossary	灯亮术语
LightOffGlossary	灯灭术语
ConstGlossary	常量值术语
LogicGlossary	逻辑运算符术语
RelayGlossary	关系运算符术语

			ActionGlossary	动作分隔符术语
			ScenceGlossary	场景术语
			ReverseGlossary	置反术语
			Expression	表达式

1.3.1.2 lua 脚本内容

LuaData 格式字段定义如下：

```
{
     “LuaData”: {
          “type”:0
          “data”:”路径字符串/脚本内容”
     }
}
```

字段名	字段含义		
LuaData	type	0 路径类型　1 脚本类型	
	data	路径字符串/脚本内容	

1.3.2 家居业务数据

1.3.2.1 家居领域函数以及应用函数定义

FunctionDefine 格式字段定义如下：

```
“ FunctionDefine”: [
     {
          “funKey “: “5”,
          “funName”: “AdaptSceneTable”,
          “returnDataType”: “int8”,
          “parasList”: [
            {
               “dataType”: “int8”,
              “parasDescription”: “[0,255]”
            },
            {
               “dataType”: “int8”,
              “parasDescription”: {1,4}
            }
          ],
          “description”: “自适应场景表函数”,
     }
]
```

注：字段都为必填，不可以缺字段名,内容可以为空

字段名	字段含义	数据结构	对应家居 3.0 的表字段	字段说明
FunctionDefine	函数定义	vector<funcDefine>	系统数据表（SystemData）的系统函数表（SystemFunction）和家居产品定义表（ProjectDefine）的家居函数定义表（SHFunctionDefine）	
funcDefine 字段数据				

funKey	函数唯一标识	string	码值（Code）	
funName	函数名	string	名称（Name）	
returnDataType	函数返回类型	string	返回值类型（ReturnDataType）	
parasList	入参列表	vector<string>	形参列表（ParasList）	
description	函数名称	string	描述（Description）	
funcParasDescription	函数参数值描述	string		

parasList 字段数据				
dataType	数据类型	string		请参考附录 2.1，2.2,2.3
parasDescription	参数值描述	string		“参数值描述”是“值描述”的列表，由多个“值描述”组成，每个“值描述”对应函数的每个参数。请参考 1.3.2.2 家居字段的 valueDescription

备注：具体函数 请看附录 2.5

1.3.2.2 家居产品属性定义

ProductAttributeDefine 格式字段定义如下：

```
“ ProductAttributeDefine”: [
    {
        “productName”: “三开智能控制面板”,
        “ productAttributeInfo “: [
            {
                “attributeName”: “上左键”,
                “attributeType”: 0,
                “dataType”: “int8”,
                “valueDescription”:”{单击=1,双击=2,长按=3}”
            }
        ]
    }
]
```

注：字段都为必填，不可以缺字段名,内容可以为空

字段名	字段含义	数据结构	对应家居 3.0 的表字段	字段说明

ProductAttributeDefine	产品定义	vector<productAttribute>		
productDefine 字段数据				
productName	产品名称	string	家居产品定义表（ProjectDefine）：产品定义表（ProductDefine）的产品名称（ProductName）	
productAttributeInfo	产品属性信息	vector< productInfo >		
productInfo 字段数据				
attributeName	属性名称	string	家居产品定义表（ProjectDefine）：设备定义表（DeviceDefine）的设备名称（DeviceName）	
attributeType	属性类型	int8 1:输入类 2:输出类 3:输入输出类	家居产品定义表（ProjectDefine）：设备定义表（DeviceDefine）的设备类型（DeviceType） 1:输入类 **对应** 1：事件类设备（实设备：输入设备） 2:输出类 **对应** 2：状态类设备（实设备：输出设备） 3：输入输出类 **对应** 3：事件状态类设备（实设备：在条件中是输入设备，在结果中是输出设备）和 5:虚拟设备（虚设备）；	
dataType	数据类型	String，请参考附录 2.1，2.2,2.3	家居产品定义表（ProjectDefine）：设备功能定义表（DeviceFunctionDefine）的数据类型（DataType）	
valueDescription	值描述	字符串		**值 描** **描述** **举例**

	范围			**述类型**		
				枚举型	1、描述数值的取值集合，表示取值范围仅允许使用集合中已定义的数值。 取值集合使用数学上的集合表示法“{}”。 2、集合内各数值用逗号“,”分隔。 在数学集合表示法的基础上，可以为每个数值设定别名.同一值描述内，别名（枚举）不允许重复	{1,3,5,6,7}； {红色=1,黄色=3,绿色=5,紫色=6,蓝色=7}
				范围型	1、描述数值的取值区间，表示取值范围仅允许使用不超出区间范围的数值。 使用数学上的区间表示法，包括开区间“()”与闭区间“[]”。 区间起始值与结束值之间用逗号“,”分隔. 2、起始值不能大于结束值。 3、区间使用的符号均为半角符号，起始值与结束值的取值范围暂只支	[1,10]：大于等于1，小于等于10； (1,10)：大于1，小于10； [1,10)：大于等于1，小于10； (1,10]：大于1，小于等于10。

					持数值（int8~int64，float，double）	

二、附录

2.1 家居领域语言 V3.0 数据类型

类型	描述
State	1. 整型，取值范围：[0, 56]
ShortInt（短整型）	1. 整型，取值范围：[-2047, +2047] 2. Data.No.1byte 和 Data.No.2byte 为有效字节
LongInt（长整型）	1. 整型，取值范围：[-524287,+524287] 2. Data.No.1byte、Data.No.2byte 和 Data.No.3byte 为有效字节。
Double（非整形）	1. Data.No.1byte、No.2byte 和 No.3byte 为有效字节。取值范围：[-4095.99,+4095.99] 2. Data.No.3byte 的第 0-6 位表示小数部分，精度为 2 位小数。 3. Data.No.1byte 的第 0-2 位、No.2byte 的 0-7 位和 No.3byte 的第 0-7 位表示整数部分。
String（字符串类型）	1. 字符串的最大长度不大于 4095 个字节，以“/0”结束。

2.2 家居领域语言 V4.0 数据类型

数据类型分类	类型	存储大小	值范围	精度
整数类型	int8	1 字节	-128 到 127	
	int16	2 字节	-32,768 到 32,767	
	int32	4 字节	-2,147,483,648 到 2,147,483,647	
	int64	8 字节	-9,223,372,036,854,775,808	

			到 9,223,372,036,854,775,807	
浮点数类型	float	4 字节	1.2E-38 到 3.4E+38	符合 IEEE 754 标准的 6 位有效数字
	double	8 字节	2.3E-308 到 1.7E+308	符合 IEEE 754 标准的 15 位有效数字
布尔类型	bool	1 字节	false（不限大小写）； true（不限大小写）	
字符串类型	string			

2.3 家居领域语言 V4.0 与 V3.0 之间的转换关系

家居领域语言 V3.0	转换为家居领域语言 V4.0
State(状态型)	int8
ShortInt（短整型）	int16
LongInt（长整型）	int32
Double（非整形）	double
String（字符串类型）	string

2.4 内置变量

2.4.1 IDE(参考 3.0)：

序号	名称	负载设备类型	子设备配置	功能码	子设备类型	数据类型	设备码值	通道编号	拨码地址
1	TimeValue	32512	9	240	39	LongInt	32512	0	255
2	YKSB	32512	9	241	39	String	32513	0	255
3	CJSB	32512	9	243	39	ShortInt	32514	0	255

4	SQSB	32512	9	244	39	ShortInt	32515	15	255
5	DelayVar	32512	9	246	39	LongInt	32516	0	255
6	vSceneManual	32512	9	243	39	ShortInt	32514	15	255
7	vSceneStart	32512	9	243	39	ShortInt	32514	15	255
8	vAdaptScene	32512	9	243	39	ShortInt	32514	15	255

注：

1） DelayVar、vSceneManual、vSceneStart、vAdaptScene 是新增变量

2） TimeValue 和 DelayVar　这两个变量在执行时 在　优特云公共函数库维护

3） 序号这一列 家居领域语言编译器不再经过转换

2.4.2 对应 Txt：

变量名称	数据类型	设备名称	产品名称	产品内属性名称	唯一标识	区域名称	负载标识
TimeValue	int32	TimeValue	时间	时间		-1	-1
YKSB	string	YKSB	遥控	遥控		-1	-1
CJSB	int16	CJSB	场景	场景		-1	-1
SQSB	int16	SQSB	授权	授权		-1	-1
DelayVar	int32	DelayVar	延时	延时		-1	-1
vSceneManual	int16	vSceneManual	手动操作场景	场景		-1	-1
vSceneStart	int16	vSceneStart	场景启动	场景		-1	-1
vAdaptScene	int16	vAdaptScene	自适应场景变量	场景		-1	-1

2.5　函数

2.5.1 公共函数

2.5.1.1 时间类函数

2.5.1.1.1 时间段函数

函数原型	bool TimeRange(int16 startYear, int16 startMonth, int16 startDay, int32 startTimeHMS, int16 endYear, int16 endMonth, int16 endDay, int32 endTimeHMS)
所属类型	公共函数

函数 KEY	0
函数功能	判断当前时间是否在指定的时间段内
输入参数	startYear：开始年份，以 2000 年为起始年，【2000-2100】 startMonth：开始月份，【1-12】 startDay：开始日，【1-31】 startTimeHMS：开始时间一天的秒延时信息，相对于 00:00:00 过了多少秒 endYear：结束年份，以 2000 年为起始年，【2000-2100】 endMonth：结束月份，【1-12】 endDay：结束日，【1-31】 endTimeHMS：结束时间一天的秒延时信息，相对于 00:00:00 过了多少秒
返回值	true：判断成立；false：判断不成立
是否同步	是
举例说明	时间段 07:00:00-09:00:00：TimeRange（-1,-1,-1，25200，-1，-1，-1，32400）
其他说明	只能参与在条件中，当不需要某个参数（年、月、日）时，传入-1 即可

2.5.1.1.2 阴历函数

函数原型	bool LunarTime(Int8 lunarMonth, Int8 lunarDay)
所属类型	**公共函数**
函数 KEY	1
函数功能	判断当前时间是否是指定的阴历时间
输入参数	lunarMonth：阴历月份 lunarDay：阴历日
输出参数	无
返回值	true：判断成立；false：判断不成立
举例说明	如阴历 2 月 3 日，LunarTime(2, 3)
其他说明	只能参与在条件中

2.5.1.1.3 阳历函数

函数原型	bool Calendar(int16 year, int8 month, int8 day)
所属类型	**公共函数**
函数 KEY	2
函数功能	判断当前时间是否是指定的阳历时间
输入参数	year：年份，以 2000 年为起始年，【2000-2100】 month：月份，【1-12】 day：日，【1-31】
输出参数	无
返回值	true：判断成立；false：判断不成立
举例说明	如阳历 2017 年 5 月 1 日， Calendar(17, 5, 1);
其他说明	只能参与在条件中

2.5.1.1.4 星期函数

函数原型	bool WeekDay(Int8 weekInfo)
所属类型	**公共函数**
函数 KEY	3

函数功能	判断当前时间是否为指定星期的组合
输入参数	weekInfo：星期组合信息。 用低 7 位表示，每一位代表一个具体的星期，1 表示有效，0 表示无效； 星期一：0x01 星期二：0x02 星期三：0x04 星期四：0x08 星期五：0x10 星期六：0x20 星期日：0x40
输出参数	无
返回值	true：判断成立；false：判断不成立
举例说明	如周末（包括周六和周日），WeekDay (0x60)
其他说明	只能参与在条件中

2.5.1.1.5 延时函数

函数原型	英文：void Delay (int16 id, int32 seconds)
所属类型	**公共函数**
函数 KEY	4
函数功能	延时
输入参数	id：延时 id seconds：时间（单位秒）
返回值	无
是否同步	是

2.5.1.1.6 取消延时函数

函数原型	英文：void CancelDelay (int16 id)
所属类型	**公共函数**
函数 KEY	5
函数功能	取消延时
输入参数	id：延时 id
返回值	无
是否同步	是

2.5.2 应用函数

2.5.2.1 HomeSceneTable 场景函数

函数原型	bool HomeSceneTable(int16 SceneNo)
函数 KEY	6
函数功能	实现场景控制

输入参数	SceneNo：场景编号
输出参数	无
返回值	bool=1 控制成功；bool=0 控制失败
举例说明	HomeSceneTable(0x01)
其他说明	只能参与在结果中

2.5.2.2 HomeRemoteControl 批处理函数

函数原型	bool HomeRemoteControl (string varKey)
函数 KEY	7
函数功能	手机端直接操作设备
输入参数	varKey：变量 Key
输出参数	无
返回值	bool=1 控制成功；bool=0 控制失败
举例说明	BatchFunction (1)
其他说明	**批处理对应的是家居 3.0 的遥控函数（HomeRemoteControl）**

2.5.2.3 HomeHoliday 节日函数

函数原型	bool HomeHoliday (int16 HolidaynNo)
函数 KEY	8
函数功能	实现节日判断功能
输入参数	HolidaynNo：节日编号
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	HomeHoliday (0x01) 判断当前是否为春节
其他说明	只能参与在条件中

2.5.2.4 LockAuthTable 门锁授权表函数

函数原型	bool LockAuthTable (int16 AuthNum)
函数 KEY	9
函数功能	实现门锁授权功能
输入参数	AuthNum：授权编号
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	LockAuthTable (0x0001) 判断是否授权
其他说明	只能参与在结果中

2.5.2.5 AdaptSceneTable 自适应场景表函数

函数原型	bool AdaptSceneTable (int8 TableID, int8 Drection)
函数 KEY	10
函数功能	实现自适应场景
输入参数	TableID：场景表编号 ； Drection:切换方向 0-无动作 1-减 2-增
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	AdaptSceneTable (0x01, 0) 编号 01 表执行自适应 AdaptSceneTable (0x01, 1) 编号 01 表向下减一个场景 AdaptSceneTable (0x01, 2) 编号 01 表向上增一个场景
其他说明	只能参与在结果中

2.5.2.6 DirectControl 直控函数

函数原型	bool DirectControl (String DirectCtlDev, int8 NodeNO, int8 NodeProductNO, int8 ChNO, int8 HostID)
函数 KEY	11
函数功能	实现设备直接控制
输入参数	DirectCtlDev :直控设备，NodeNO：节点编号 NodeProductNO：节点内产品编号 ChNO：通道编号 HostID:主机号
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	DirectControl （“设备+值”,0x01, 0x02, 0x00, 0x01) 直控设备值，节点编号 1，节点内产品编号 2 通道 0 主机号 1
其他说明	只能参与在结果中

2.5.2.7 设备直控函数（DirectControlDev）

函数原型	bool DirectControlDev (String DirectCtlDev, int8 NodeDevId, int8 NodeNO, int8 NodeProductNO, int8 ChNO, int8 HostID)
函数 KEY	12
函数功能	实现设备直接控制
输入参数	DirectCtlDev :直控设备名称； NodeDevId：直控对象（设备）节点内设备编号； NodeNO：直控对象（设备）节点编号； NodeProductNO：直控对象（设备）节点内产品编号； ChNO：直控对象（设备）通道编号； HostID ： 直控对象（设备）主机号；

输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	DirectControlDev（“直控 1”,0x03，0x01，0x02，0x00，0x01）直控设备名称，节点内设备编号 3， 节点编号 1，节点内产品编号 2 ，通道 0， 主机号 1）
其他说明	只能参与在结果中

2.5.2.8 比较函数（Compare）

函数原型	bool Compare (int32 srcNodeNO, int32 srcNodeProductNO,int8 srcChNO, int8 srcHostID, int32 dstNodeNO, int32 dstNodeProductNO,int8 dstChNO, int8 dstHostID)
函数 KEY	13
函数功能	比较函数
输入参数	srcNodeNO：源直控对象（设备）节点编号； srcNodeProductNO：源直控对象（设备）节点内产品编号； srcChNO：源直控对象（设备）通道编号； srcHostID ：源直控对象（设备）主机号； dstNodeNO：源直控对象（设备）节点编号； dstNodeProductNO：源直控对象（设备）节点内产品编号； dstChNO：源直控对象（设备）通道编号； dstHostID ：源直控对象（设备）主机号；
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	
其他说明	

2.5.2.9 切换函数（Switch）

函数原型	bool Switch (int32 srcNodeNO, int32 srcNodeProductNO,int8 srcChNO, int8 srcHostID, int32 direction)
函数 KEY	14
函数功能	切换函数
输入参数	srcNodeNO：源直控对象（设备）节点编号； srcNodeProductNO：源直控对象（设备）节点内产品编号； srcChNO：源直控对象（设备）通道编号； srcHostID ：源直控对象（设备）主机号； direction：切换方向 1-左切换 2-右切换
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败

举例说明	
其他说明	

2.5.2.10 置反函数（Reverse）

函数原型	bool Reverse (int32 varId)
函数 KEY	15
函数功能	变量值取反
输入参数	varId:变量 ID,编译器内部生成，业务方通过 ID 获取 key。
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	
其他说明	只能参与在结果中

2.5.2.11 获取场景状态（GetSceneStatus）

函数原型	Int32 GetSceneStatus (int32 sceneNo)
函数 KEY	16
函数功能	获取场景状态
输入参数	sceneNo:场景编号
输出参数	无
返回值	场景状态
举例说明	
其他说明	

2.5.2.12 通过场景类型和场景编号获取场景优先级（GetScencePriority）

函数原型	Int32 GetScencePriority (int32 sceneSchemeType, int32 sceneNo)
函数 KEY	17
函数功能	通过场景类型和场景编号获取场景优先级
输入参数	sceneSchemeType:场景方案类型 sceneNo：场景编号
输出参数	
返回值	
举例说明	
其他说明	

2.5.2.13 通过场景类型获取场景优先级（GetScencePriority）

函数原型	Int32 GetScencePriority (int32 sceneSchemeType)
函数 KEY	18
函数功能	通过场景类型获取场景优先级
输入参数	sceneSchemeType:场景方案类型
输出参数	
返回值	
举例说明	
其他说明	

2.5.2.14 设置场景优先级（SetScencePriority）

函数原型	bool SetScencePriority (int32 sceneSchemeType, int32 sceneNo,int32 sceneValue)
函数 KEY	19
函数功能	设置场景优先级
输入参数	sceneSchemeType:场景方案类型 sceneNo：场景编号 sceneValue：场景值
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	
其他说明	

2.5.2.15 重置场景优先级（ResetScencePriority）

函数原型	bool ResetScencePriority ()
函数 KEY	20
函数功能	重置场景优先级
输入参数	
输出参数	
返回值	
举例说明	
其他说明	

2.5.2.16 检查自适应场景行函数（CheckAdaptScene）

函数原型	bool CheckAdaptScene (int8 TableID, int8 RowId)
函数 KEY	21
函数功能	检查自适应场景行是否满足条件
输入参数	TableID：场景表 ID； RowId:场景行 ID;
输出参数	无
返回值	bool=1 执行成功；bool=0 执行失败
举例说明	
其他说明	

2.6 用户语言 3.0 和 4.0 语法区别

用户语言3.0语法

```
(触发条件) , (参与条件)   @ 真动作    : 假动作    !
(Var1==1)  , (var2==1)    @ FUNC1()  : FUNC2()  !
```

用户语言4.0语法

```
触发条件    ? 参与条件  @ 真动作    : 假动作    !
Var1==1     ? var2==1   @ FUNC1()  : FUNC2()  !
```

2.7 用户语言运算符 3.0 和 4.0 区别

运算符分类	运算符名称	用户语言 4.0 标识
赋值运算符	赋值	=
关系运算符	相等	==
	不等于	！=
	不小于	>=
	不大于	<=
	小于	<

	大于	>
逻辑运算符	与	&&
	或	||
数学运算符	加法	+
	减法	-
	乘法	*
	除法	/
分隔符	条件动作分隔符	@
	事件条件分隔符	?
	真假动作分隔符	:
	动作分隔符	;
	调用参数分隔符	,
结束符	结束符	!
双引号	双引号	”
	转义双引号	/”
括号	左括号	(
	右括号	)