

原理图文档格式

- 1 格式版本
- 2 基础图元格式
 - 2.1 样式
 - 2.1.1 FONTSTYLE 字体样式
 - 2.1.2 LINSTYLE 线型样式
 - 2.2 GROUP 分组控制
 - 2.3 ATTR 属性
 - 2.4 WIRE 导线
 - 2.5 BUS 总线
 - 2.6 BUSENTRY 总线接入标识
 - 2.7 TEXT 文本
 - 2.8 RECT 矩形
 - 2.9 POLY 多边形
 - 2.10 CIRCLE 圆
 - 2.11 ARC 圆弧
 - 2.12 PIN 标号
 - 2.13 OBJ 二进制内嵌对象
- 3 SCH 原理图类型文档
 - 3.1 文件头
 - 3.2 原理图主体
 - 3.3 COMPONENT
 - 3.3.1 Symbol 类型
 - 3.3.2 Component 图元
- 4 INSTANCE 实例属性类型文档
 - 4.1 文件头
 - 4.2 OVERRIDE 实例属性覆盖
- 5 SYMBOL 符号类型文档
 - 5.1 文件头
 - 5.2 PART 子库 1
 - 5.3 PART 子库 2
- 6 修订日志
 - 6.1 2022042501
 - 6.2 2022030701
 - 6.3 2022030201
 - 6.4 2021111801
 - 6.5 2020102301
 - 6.6 2020101401
 - 6.7 2020092801

1 格式版本

```
["DOCTYPE", "SCH", "1.1"]
["DOCTYPE", "SYMBOL", "1.1"]
```

2 基础图元格式

2.1 样式

- 样式机制，仅用于格式内容尺寸压缩的抽象，不具有任何在 XTools 内的实体映射
- 样式只要能出现在被引用前就行
- 所有样式的应用范围都是文件级（例如：放在 PART 里的样式不属于 PART）
- 所有样式内容（除了 样式标识 和 编号）都可以置 null 表示采用默认样式

2.1.1 FONTSTYLE 字体样式

```
["FONTSTYLE", "st001", "#880000", "#880000", null, 7, null, 0, 0, 0, 1, 0]
```

1. 样式标识：FONTSTYLE
2. 字体样式编号：全文件唯一
3. 颜色
4. 背景色
5. 字体名称
6. 字体大小，与坐标等单位相同
7. 是否斜体
8. 是否加粗
9. 是否加下划线
10. 是否加删除线
11. 垂直对齐模式：0 顶部对齐 1 中间对齐 2 底部对齐
12. 水平对齐模式：0 左对齐 1 居中 2 右对齐

```
["FONTSTYLE", "st002", "#880000", "", "Consolas", 7, 1, 0, 0, 1, 1, 2]
```

```
["FONTSTYLE", "st003", "#880000", "", "Consolas", 7, 1, 0, 0, 0, 0, 2]
```

2.1.2 LINSTYLE 线型样式

```
["LINSTYLE", "st004", "#880000", 0, "#664400", 1]
```

1. 样式标识: **LINSTYLE**
2. 字体样式编号: 全文件唯一
3. 颜色
4. 样式: **0** 实线 **1** 短划线 **2** 点线 **3** 点划线
5. 填充颜色: 不填充, 填充自动闭合起始点和结束点
6. 宽度

```
["LINSTYLE", "st005", "#880000", 1, "", 1]
```

```
["LINSTYLE", "st006", "#880000", 0, "#664400", 5]
```

2.2 GROUP 分组控制

```
["GROUP", 1, 0, "Logo", ["e1", "e2"]]
```

1. 分组控制
2. 分组编号: 不能为 **0**
3. 父级分组编号: 为 **0** 则表示无父级
4. 分组名称: 无名称为空字符串
5. 图元编号: 隶属于这个组的所有图元编号

2.3 ATTR 属性

ATTR 是一个较为通用的图元, 其含义为

1. 表达一个由 键(**KEY**)—值(**VALUE**) 组成的的多个属性中的一个
2. 可以在画布上显示, 并控制显示哪些内容, 以及样式位置等

当隶属编号没有指定时, 则默认隶属于当前块级图元上

```
["ATTR", "e177", "", "UUID", "432143214321", 1, 1, 300, 200, 0, "st002", 1]
```

1. 属性名称: **ATTR**
2. 编号: 文件内唯一
3. 隶属编号: 隶属于哪个图元, 表示属于当前块 默认块是文件
4. 属性 **Key**
5. 属性 **Value**
6. 是否显示 **Key**
7. 是否显示 **Value**
8. 位置 **X**: 未显示过的属性位置固定为
9. 位置 **Y**: 未显示过的属性位置固定为
10. 旋转角度: 绕 旋转
11. 字体样式编号
12. 是否锁定

```
["ATTR", "e198", "", "ABCD", "kk~AB~Bb~233~dd", 0, 1, 300, 200, 0, "st002", 1]
```

当属性 **Value** 中含有 字符时, **XTools** 需要实现从开始到结束, 遇到第奇数个 开始文字带上划线, 遇到第偶数个 结束文字带上划线

```
["ATTR", "e199", "", "_LTSPICE_PROGRAM_", ".tran 10m", 0, 1, 300, 200, 0, "st002", 1]
```

2.4 WIRE 导线

```
["WIRE", "e170", [[310, 550, 400, 550, 400, 460], [480, 460, 400, 460], [400, 330, 400, 460]], "st004", 0]
```

1. 图元名称: **WIRE**
2. 编号: 文件内唯一
3. 坐标: 分成多段线, 每段都是连续的一组 **X1 Y1 X2 Y2 X3 Y3 ...** 描述的线
4. 线型样式
5. 是否锁定

```
["ATTR", "e199", "e170", "NET", "GND", 1, 1, 300, 200, 0, "st002", 1]
```

导线必须带上 **NET** 属性标识网络名称

2.5 BUS 总线

```
["BUS", "e271", [[310, 550, 400, 550, 400, 460], [480, 460, 400, 460], [400, 330, 400, 460]], "st005", 0]
```

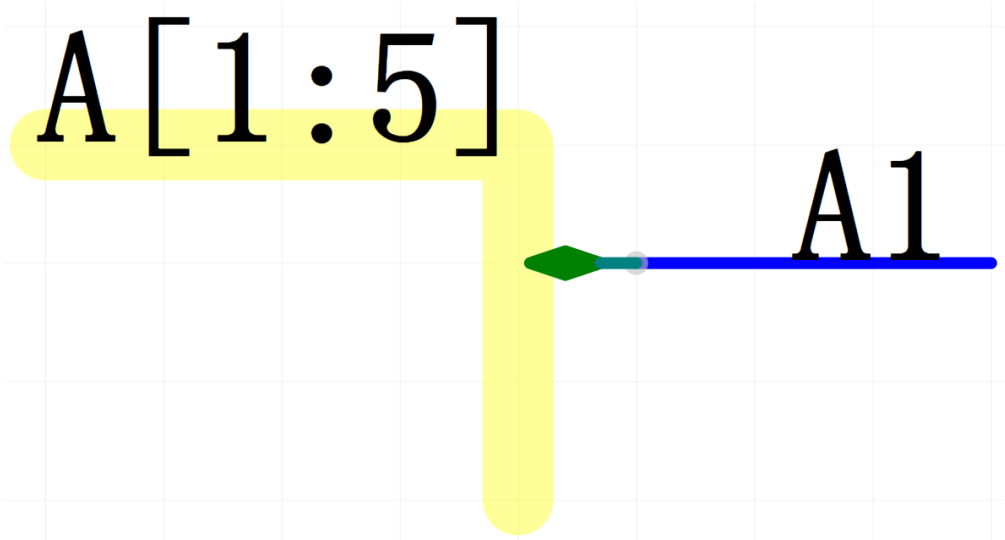
1. 图元名称: **BUS**
2. 编号: 文件内唯一
3. 坐标: 分成多段线, 每段都是连续的一组 **X1 Y1 X2 Y2 X3 Y3 ...** 描述的线
4. 线型样式编号

5. 是否锁定

```
["ATTR", "e200", "e271", "NET", "A[1:5]", 1, 1, 300, 200, 0, "st002", 1]
```

总线必须带上 NET 属性标识网络名称

2.6 BUSENTRY 总线接入标识



image

- 如图所示
 - 浅黄色为 BUS
 - 绿色圆角菱形，以及其向右延伸的一个类似 PIN 的图形，为 BUSENTRY
 - 蓝色为 Wire
- 端点为 WIRE 和 BUSENTRY 类似 PIN 的最右侧端点接触的那个端点的坐标
- 因为 WIRE 和 BUS 可以是任意角度接入的，所以需要指定其旋转方向以和 WIRE 的接入方向一致，例如图内是 180 度
- BUSENTRY 固定一格长
- BUSENTRY 具体的图形，由 XTools 最终解释，不在格式内限定

```
["BUSENTRY", "e380", "e271", 4, 500, 600, 90]
```

1. 图元名称: BUSENTRY
2. 编号: 文件内唯一
3. 隶属编号: 隶属于哪个 BUS
4. 顺序编号: 在隶属的 BUS 里的顺序编号, 可重复
比如 BUS 网络为 A[2:3]B[7:6] 可以具有 0 1 2 3 0 1 2 3 ... 一系列顺序编号的 BUSENTRY, 其中
0 一定代表分支 A2B7
1 一定代表分支 A2B6
2 一定代表分支 A3B7
3 一定代表分支 A3B6
5. 端点 X
6. 端点 Y
7. 旋转角度: 绕 端点 旋转

2.7 TEXT 文本

```
["TEXT", "e171", 108, 804.5, 0, "任意字符doukeyi!@#$", "st002", 1]
```

1. 图元名称: TEXT
2. 编号: 文件内唯一
3. 文本坐标 X
4. 文本坐标 Y
5. 旋转角度, 绕 文本坐标 旋转
6. 文本内容: 任意字符
7. 字体样式编号
8. 是否锁定: 可选

2.8 RECT 矩形

矩形由其对角的两个点定义, 其旋转是绕点1进行的

```
["RECT", "e172", 340, 210, 100, 200, 40, 30, 90, "st006", 0]
```

1. 图元名称: RECT

- 2. 编号：文件内唯一
- 3. 点 1 X
- 4. 点 1 Y
- 5. 点 2 X
- 6. 点 2 Y
- 7. 圆角半径 X: 0 表示非圆角
- 8. 圆角半径 Y: 0 表示非圆角
- 9. 旋转角度：绕 点1 旋转
- 10. 线型样式编号
- 11. 是否锁定

2.9 POLY 多边形

```
[ "POLY", "e173", [390, 260, 450, 300, 560, 280, 540, 320], 0, "st005", 0]
```

- 1. 图元名称：POLY
- 2. 编号：文件内唯一
- 3. 点集坐标：X Y X Y X Y ...
- 4. 是否自动闭合：如果自动闭合，则结束点会自动连上起始点
- 5. 线型样式编号
- 6. 是否锁定

2.10 CIRCLE 圆

```
[ "CIRCLE", "e174", 430, 200, 21, "st006", 0]
```

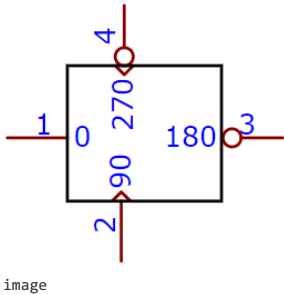
- 1. 图元名称：CIRCLE
- 2. 编号：文件内唯一
- 3. 圆心 X
- 4. 圆心 Y
- 5. 半径 r
- 6. 线型样式编号
- 7. 是否锁定

2.11 ARC 圆弧

```
[ "ARC", "e174", -10, 0, 0, 10, 10, 0, "st005", 0]
```

- 1. 图元名称：ARC
- 2. 编号：文件内唯一
- 3. 起始 X
- 4. 起始 Y
- 5. 参考 X
- 6. 参考 Y
- 7. 结束 X
- 8. 结束 Y
- 9. 线型样式编号
- 10. 是否锁定

2.12 PIN 标号



如图所示

- 所有 PIN 的 位置 X/Y 都是离黑色矩形最远的那个端点
- PIN 1 为 0 度旋转方向，引脚样式 0
- PIN 2 为 90 度旋转方向，引脚样式 1
- PIN 3 为 180 度旋转方向，引脚样式 2
- PIN 4 为 270 度旋转方向，引脚样式 3

```
[ "PIN", "e102", 1, 0, 350, 170, 20, 0, "#880000", 3, 1]
```

- 1. 图元名称：PIN
- 2. 编号：文件内唯一

3. 是否显示
4. 电气特性: 0 UNKNOWN 1 INPUT 2 OUTPUT 3 BI
5. 位置 X
6. 位置 Y
7. 引脚长度
8. 旋转角度: 0 90 180 270
9. 引脚颜色
10. 引脚样式: 1 Clock 2 DOT, 可支持按位或运算, 比如 3 = 1 | 2
举例说明: 0 无附加 1 Clock 2 DOT 3 Clock & DOT
11. 是否锁定

```
[ "ATTR", "e184", "e102", "NAME", "VCC", 1, 1, 108, 804.5, 0, "st002", 1]
```

PIN 必须具有 NAME 属性

```
[ "ATTR", "e185", "e102", "NUMBER", "1", 1, 1, 108, 804.5, 0, "st002", 1]
```

PIN 必须具有 NUMBER 属性

2.13 OBJ 二进制内嵌对象

内嵌于图顶上的图片和文件等数据, 可作为附件下载, 以及直接显示 (EDA 自行决定, 不在格式内要求)

```
[ "OBJ", "e662", "a.txt", 200, 300, 10, 20, 0, 0, "data:text/plain;base64,MTIzNA==", 1]
```

1. 二进制内嵌对象标识: OBJ
2. 编号: 文件内唯一
3. 文件名
4. 左上角 X
5. 左上角 Y
6. 宽
7. 高
8. 旋转角度: 绕左上角旋转
9. 是否镜像
10. 二进制数据, 有两种模式
 1. 一般格式, 遵循 Data Urls 规范 data:[<mediatype>][;base64],<data>
 - 如 data:image/png;base64,asdfasdfwer
 - 如 data:text/html,<html></html>
 2. BLOB 引用模式, blob:hashid
11. 是否锁定

```
[ "OBJ", "e663", "b.svg", 200, 300, 100, 200, 15, 1, "data:image/svg+xml;base64,PHN2Zz48L3N2Zz4=", 1]
```

3 SCH 原理图类型文档

3.1 文件头

```
[ "DOCTYPE", "SCH", "1.0"]
```

```
[ "HEAD", { "ORIGIN_X": 0, "ORIGIN_Y": 0, "editorVersion": "4.7.8", "importFlag": 0}]
```

1. 文档头标识: HEAD
2. 内部参数: Key-Value
 - 预留 ORIGIN_X ORIGIN_Y 为画布原点偏移, 可选
 - 其它为编辑器附加信息, 用于数据分析等功能, 可选

3.2 原理图主体

```
[ "FONTSTYLE", "st001", "#880000", "Consolas", 7, 1, 0, 0, 0, 0]
```

```
[ "FONTSTYLE", "st002", "#880000", "Consolas", 7, 1, 0, 0, 0, 2]
```

```
[ "FONTSTYLE", "st003", "#880000", "Consolas", 7, 1, 0, 0, 0, 2]
```

```
[ "LINESTYLE", "st004", "#880000", 0, "#664400", 1]
```

```
[ "LINESTYLE", "st005", "#880000", 1, "", 1]
```

```
[ "LINESTYLE", "st006", "#880000", 0, "#664400", 5]
```

```
[ "WIRE", "e112", [455, 265, 455, 485, 720, 485], "st005", 0]
```

```
[ "ATTR", "e111", "e112", "NET", "+5V", 1, 1, 108, 804.5, 0, "st002", 1]
```

```
[ "BUS", "e106", [455, 265, 455, 485, 720, 485], "st005", 0]
```

```
[ "ATTR", "e111", "e106", "NET", "+5V", 1, 1, 108, 804.5, 0, "st002", 1]
```

3.3 COMPONENT

- COMPONENT 引用了 Symbol, Symbol 支持多 PART, 所以带了 子库编号 属性指示具体哪一个, 如果是单 PART 则使用默认值 ""

- COMPONENT 下可绑定许多 ATTR，具体的属性行为将由工具定义

3.3.1 Symbol 类型

| Symbol 类型编号 | Symbol 类型 | 说明 |
|-------------|-----------------|---|
| 2 | Part Symbol | 普通器件 |
| 17 | Block Symbol | 层次图符号 |
| 18 | NetFlag Symbol | 全局网络符号 |
| 19 | NetPort Symbol | 层次图网络导出符号 |
| 20 | Sheet Symbol | 专用于提供原理图图纸的重用机制 |
| 21 | NoneElec Symbol | 无电气特性符号 NoneElec 是一类不具有 PIN 的无电气特性图元 也可以用作特殊的图标，版权声明文字等的重用机制 NoneElec 全称 None Electrical，不具有电气特性的意思 |
| 22 | Short Symbol | 短接符 Short Symbol 是一个特殊 Symbol，必须具有两个 PIN 所有与同一个 Short Symbol 的 PIN 相连的网络，将在电气特性上对其进行短接 比如网络 A 连接到了 PIN1，网络 B 连接到了 PIN2，则表示 A 和 B 是同一个网络 |

3.3.2 Component 图元

```
["COMPONENT", "e176", "1", 300, 200, 15, 0, {}, 0]
```

- COMPONENT 标识: COMPONENT
- 编号: 文件内唯一
- 子库编号: 默认 ""
- 位置 X
- 位置 Y
- 旋转角度: 绕 位置 旋转
- 是否镜像
- 纯数据属性: 附加信息，用于编辑器内部的一些逻辑
- 是否锁定

Component 所引用的 Symbol 图元一定是按照如下顺序执行的变换

- 按照 旋转角度 绕原点 (0,0) 逆时针旋转
- 如果 是否镜像 为 1，则绕原点 (0,0) 所在的 Y 轴进行水平镜像
- 根据 位置 进行平移

或者可以理解成如下等价的变换（但是实现更繁琐一些）

- 根据 位置 进行平移
- 按照 旋转角度 绕 位置 逆时针旋转
- 如果 是否镜像 为 1，则绕 位置 所在的 Y 轴进行水平镜像

```
["ATTR", "e187", "e176", "Device", "device-uuid-1", 1, 1, 300, 200, 0, "st002", 1]
```

Device uuid. 与 project.json 里 devices 对应的文件名称一致

```
["ATTR", "e188", "e176", "Symbol", "symbol-uuid-1", 1, 1, 300, 200, 0, "st002", 1]
```

COMPONENT 内的 ATTR 会对模板内同名属性覆盖，覆盖 Symbol 后会影响此器件对符号的绑定

```
["ATTR", "e188", "e176", "Footprint", "footprint-uuid-1", 1, 1, 300, 200, 0, "st002", 1]
```

COMPONENT 内的 ATTR 会对模板内同名属性覆盖，覆盖 Footprint 后会影响此器件对封装的绑定

```
["ATTR", "e178", "e176", "Designator", "U1", 1, 1, 300, 200, 0, "st002", 1]
```

COMPONENT 内的 ATTR 会对模板内同名属性覆盖

```
["ATTR", "e180", "e176e5", "NUMBER", "1", 1, 1, 108, 804.5, 0, "st002", 1]
```

PIN 属性覆盖的方式关键在 ATTR 的 隶属编号 上

编号分两部分，例如 e176e5，其中 e176 为 COMPONENT 的编号，e5 为在模板内的 PIN 编号

4 INSTANCE 实例属性类型文档

4.1 文件头

```
["DOCTYPE", "INSTANCE", "1.0"]
```

4.2 OVERRIDE 实例属性覆盖

实例属性覆盖为块级图元，与 PART 类似

```
['OVERRIDE', ['Schematic ID', '$e100', '$1e55', '$6e15', '$8'], { 'e176': { 'Designator': 'U15', 'ASDF': '1234' }, '': { 'Author': 'abc' }, 'e176e5': { 'NUMBER': 2 } }]
```

- OVERRIDE 实例属性覆盖
- 实例路径
 - Schematic ID 为顶层原理图，与导出格式里 project.json 里的 schemataics 下的名字要对应上
 - 最后一个只到 Sheet 编号

3. 中间所有的都是使用编号组合语法定位的 `Block Symbol`，如 `$1e2`，其中 `1` 为 `sheetid`，`e2` 为 `Block Symbol` 编号
3. 属性覆盖，数据签名名为 `[{ parentId: string]: { [key: string]: string } }`

```
['OVERRIDE', ['Schematic ID', '$5'], { 'e176': { 'Designator': 'U15', 'ASDF': '1234' }, '': { 'Author': 'abc' }, 'e176e5': { 'NUMBER': 2 } } ]]
```

这种写法就是针对非层次图实例的属性覆盖

5 SYMBOL 符号类型文档

5.1 文件头

```
[ "DOCTYPE", "SYMBOL", "1.0" ]
```

```
[ "HEAD", { "symbolType": 2 } ]
```

1. 文档头标识: `HEAD`
2. 内部参数: `Key-Value`
 - 预留 `symbolType` 为 `Symbol` 类型，给其它工具用于识别符号类型
XTools 核心逻辑不应关注这个属性
 - *其它为编辑器附加信息，用于数据分析等功能，可选

5.2 PART 子库 1

`PART` 是一个块级元素

要求无任何图元在 `PART` 范围外

如果是单 `Part` 器件也必须有一个 `PART`，子库编号留空 `""`

```
[ "PART", "1", { "BBOX": [-10, -20, 10, 20] } ]
```

1. 子库图元: `PART`
2. 子库编号
3. 内部参数: `Key-Value`
 - 预留 `BBOX` 为能刚好框住 `PART` 下所有图元的矩形包围盒任意对角的两个点
XTools 核心逻辑不应关注这个属性
 - 其它为编辑器附加信息，用于数据分析等功能，可选

```
[ "FONTSTYLE", "st002", "#880000", "Consolas", 7, 1, 0, 0, 0, 0, 2 ]
```

```
[ "ATTR", "e180", "", "NAME", "myname.1", 1, 1, 300, 200, 0, "st002", 1 ]
```

```
[ "ATTR", "e181", "", "PREFIX", "prefix.1", 1, 1, 300, 200, 0, "st002", 1 ]
```

```
[ "PIN", "e102", 1, 0, 350, 170, 20, 0, "#880000", 3, 0, 1 ]
```

```
[ "ATTR", "e184", "e102", "NAME", "VCC", 1, 1, 108, 804.5, 0, "st002", 1 ]
```

```
[ "ATTR", "e185", "e102", "NUMBER", "1", 1, 1, 108, 804.5, 0, "st002", 1 ]
```

```
[ "TEXT", "e106", 108, 804.5, 0, "任意字符doukeyi!@#$", "st002", 1 ]
```

```
[ "LINESTYLE", "st006", "#880000", 0, "#664400", 5 ]
```

```
[ "RECT", "e107", 340, 210, 350, 220, 0, 0, 90, "st006", 0 ]
```

```
[ "POLY", "e108", [390, 260, 450, 300, 560, 280, 540, 320], 0, "st006", 0 ]
```

```
[ "LINESTYLE", "st005", "#880000", 1, "", 1 ]
```

```
[ "ARC", "e174", -10, 0, 0, 10, 10, 0, "st005", 0 ]
```

5.3 PART 子库 2

```
[ "PART", "2", { "BBOX": [-10, -20, 10, 20] } ]
```

```
[ "ATTR", "e180", "", "NAME", "myname.1", 1, 1, 300, 200, 0, "st002", 1 ]
```

```
[ "ATTR", "e181", "", "PREFIX", "prefix.1", 1, 1, 300, 200, 0, "st002", 1 ]
```

```
[ "PIN", "e102", 1, 0, 350, 170, 20, 0, "#880000", 3, 0, 0, 1 ]
```

```
[ "ATTR", "e184", "e102", "NAME", "GND", 1, 1, 108, 804.5, 0, "st002", 1 ]
```

```
[ "ATTR", "e185", "e102", "NUMBER", "2", 1, 1, 108, 804.5, 0, "st002", 1 ]
```

```
[ "TEXT", "e106", 108, 804.5, 0, "任意字符doukeyi!@#$", "st002", 1 ]
```

```
[ "RECT", "e107", 340, 210, 350, 220, 0, 0, 90, "st006", 0 ]
```

```
[ "POLY", "e108", [390, 260, 450, 300, 560, 280, 540, 320], 0, "st006", 0 ]
```

```
[ "ARC", "e174", -10, 0, 0, 10, 10, 0, "st005", 0 ]
```

6 修订日志

6.1 2022042501

添加 `GROUP` 分组控制元素

6.2 2022030701

重新设计 `OBJ` 格式

6.3 2022030201

`OBJ` 添加 `BLOB` 引用模式

6.4 2021111801

未显示过的 `ATTR`，通过 `X/Y` 任一为 `null` 标记

6.5 2020102301

重新设计 `INSTANCE` 属性覆盖 文档格式

6.6 2020101401

`COMPONENT` 图元不再将 `device` 放入图元，而是作为其属性存在

6.7 2020092801

`SYMBOL` 图元调整为 `COMPONENT` 图元，引入 `Device` 和 `symbol_uuid` 属性覆盖机制