ZHC492S_JSON_应用指导

目 录

Content

1.参数配置

1.1.系统参数

1.1.1.读取基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getSysBasic
devID	是	字符	AAAAAAAAAAAAAAAA，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getSysBasicAck
devID	是	字符	AAAAAAAAAAAAAAAA，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
id	否	字符	设备唯一识别码，16 位
pw	否	字符	纵横云密码
cpuID	否	字符	保留
ver	否	字符	版本号

示例

```
{
    "msgType": "getSysBasic",
    "devID": "AAAAAAAAAAAAAAAA",
    "data": {}
}
```

响应：

```
{
  "devID": "AAAAAAAAAAAAAAAA",
  "msgType": "getSysBasicAck",
  "data": {
```

```
        "id": "492C210803024166",
        "pw": "123456",
        "cpuID": "Mf/TBUJHNzQVgAVX",
        "ver": "1030"
    },
    "timestamp": "1631350007"
}
```


1.1.2.写入基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSysBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式

字段	必须	类型	描述
pw	是	字符	纵横云密码

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSysBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	设参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

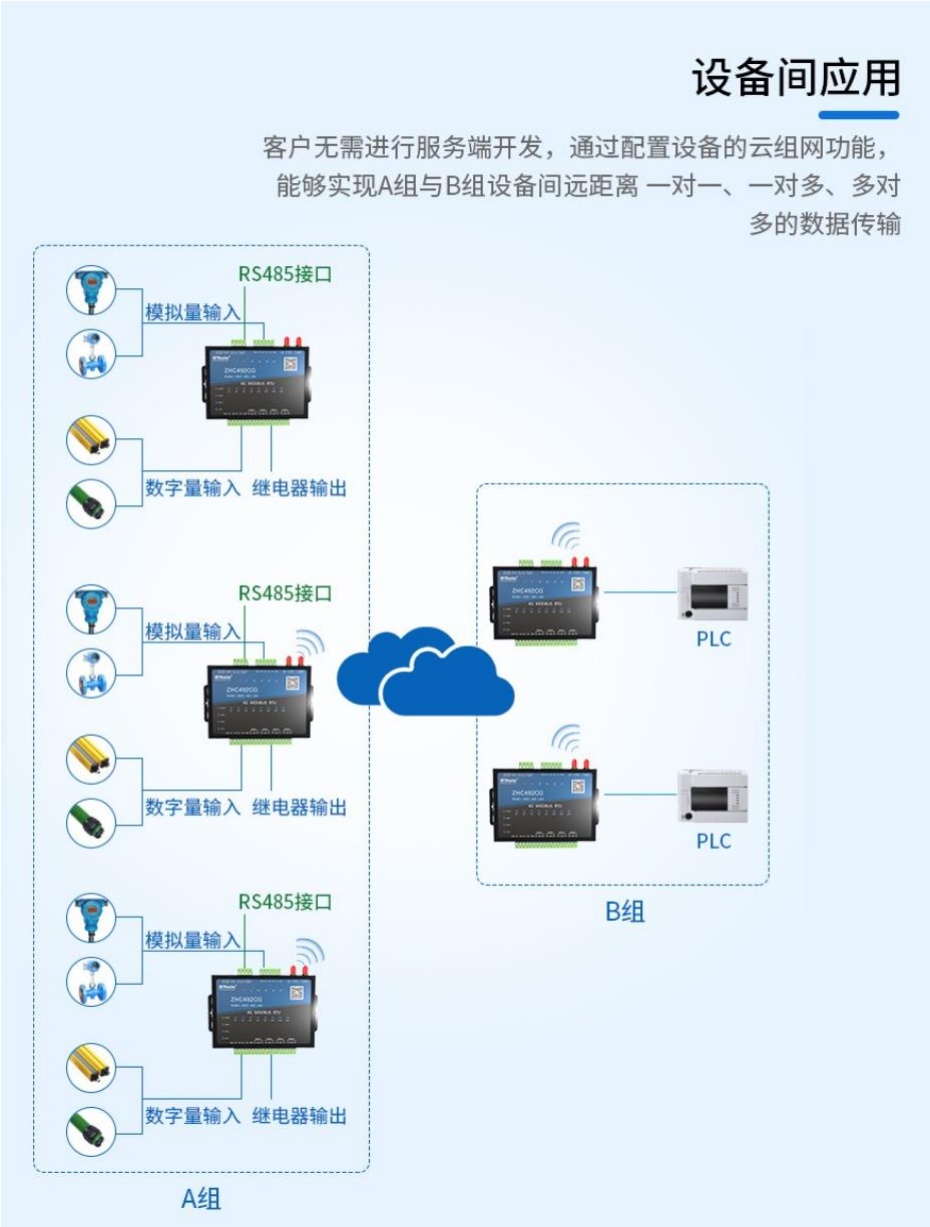

```
{
    "msgType": "setSysBasic",
    "devID": "492C210803024166",
    "data": {
        "pw": "123456"
    }
}
```

响应：


```
{
    "devID": "492C210803024166",
    "msgType": "setSysBasicAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631351423"
}
```


1.1.3.读取纵横云参数

说明：“纵横云参数”能够实现多台设备互传数据，如下图



请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getZHCGroup
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getZHCGroupAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
groupEN	是	整型	使能标志。0:禁用 1:启用
groupType	是	整型	组类型。0:A 1:B 不同类型方可通信
groupID	是	字符	组编号。相同方可通信
groupPW	是	字符	组密码。相同方可通信

示例

```
{
    "msgType": "getZHCGroup",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getZHCGroupAck",
    "data": {
        "groupEN": 0,
        "groupType": 0,
        "groupID": "",
        "groupPW": ""
    },
    "timestamp": "1631352255"
}
```

1.1.4.写入纵横云参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setZHCGroup
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
groupEN	否	整型	使能标志。0:禁用 1:启用
groupType	否	整型	组类型。0:A 1:B 不同类型方可通信
groupID	是	字符	组编号。相同方可通信
groupPW	是	字符	组密码。相同方可通信

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setZHCGroupAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setZHCGroup",
    "devID": "492C210803024166",
    "data": {
        "groupEN": 0,
        "groupType": 0,
        "groupID": "123",
        "groupPW": "123"
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setZHCGroupAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631496488"
}
```


1.1.5.读取定位信息

说明：ZHC492SG 硬件支持定位，ZHC492S 不支持硬件定位

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getLocation
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getLocationAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
cyc	是	字符	定位信息上报周期。0:关闭主动上报
location	是	字符	定位信息。硬件若支持定位，则无法自定义定位信息，同时，定位数据包符合《NMEA1803》协议规范。最大 100 字符

示例


```
{
    "msgType": "getLocation",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getLocationAck",
    "data": {
        "cyc": "0",
        "location": "$GNGGA,001927.103,,,,,0,0,,,M,,M,,*59\r\n"
    },
    "timestamp": "1631496944"
}
```

1.1.6.写入定位信息

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setLocation
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
cyc	否	字符	定位信息上报周期。0:关闭主动上报
location	否	字符	定位信息。硬件若支持定位，则无法自定义定位信息。最大 100 字符

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setLocationAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	设置结果根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例


```
{
    "msgType": "setLocation",
    "devID": "492C210803024166",
    "data": {
        "cyc": "0",
        "location": "11586426.191872796,3551679.881762213"
    }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "setLocationAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631497351"
}
```


1.2.本地逻辑

说明：ZHC492S 支持本地接口的联动。

- DO 正向跟随 DI
  DI 检测到**闭合**，则 DO **常闭**；DI 检测到**断开**，则 DO **常**开
- DO 反向跟随 DI
  与“正向跟随”相反
- 大于等于阈值触发 DO
  AI 检测值**大于**等于设定的阈值，则触发设定的 DO 动作
- 大于等于阈值触发 DO
  AI 检测值**小于**等于设定的阈值，则触发设定的 DO 动作

1.2.1.读取本地逻辑

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getLogic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getLogicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
logicx	是	字符	逻辑根节点。x = 1~8
cdt	是	整型	逻辑模式。0:禁用 1:整向跟随 2:反向跟随 3:保留 4:大于等于 5:小于等于
inReg	是	整型	输入检测通道。取值:0~3。
outReg	是	整型	输出通道。取值:0~3。
DO	是	整型	DO 输出状态。0:断开 1:闭合 2:翻转
AI	是	字符	AI 检测阈值。

示例

```
{
    "msgType": "getLogic",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getLogicAck",
    "data": {
        "logic1": {
            "cdt": 0,
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
            "AI": "0"
        },
        "logic2": {
            "cdt": 0,
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
```

```
        "AI": "0"
    },
    "logic3": {
        "cdt": 0,
        "addr": "00",
        "inReg": 0,
        "outReg": 0,
        "DO": 0,
        "AI": "0"
    },
    "logic4": {
        "cdt": 0,
        "addr": "00",
        "inReg": 0,
        "outReg": 0,
        "DO": 0,
        "AI": "0"
    },
    "logic5": {
        "cdt": 0,
        "addr": "00",
        "inReg": 0,
        "outReg": 0,
        "DO": 0,
        "AI": "0"
    },
    "logic6": {
        "cdt": 0,
        "addr": "00",
        "inReg": 0,
        "outReg": 0,
        "DO": 0,
        "AI": "0"
    },
    "logic7": {
        "cdt": 0,
        "addr": "00",
        "inReg": 0,
        "outReg": 0,
        "DO": 0,
        "AI": "0"
    },
    "logic8": {
        "cdt": 0,
```

```
                "addr": "00",
                "inReg": 0,
                "outReg": 0,
                "DO": 0,
                "AI": "0"
            }
        },
        "timestamp": "1631497918"
}
```


1.2.2.写入本地逻辑

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setLogic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
logicx	否	字符	逻辑根节点。x = 1~8
cdt	否	整型	逻辑模式。0:禁用 1:整向跟随 2:反向跟随 3:保留 4:大于等于 5:小于等于
inReg	否	整型	输入检测通道。取值:0~3。
outReg	否	整型	输出通道。取值:0~3。
DO	否	整型	DO 输出状态。0:断开 1:闭合 2:翻转
AI	否	字符	AI 检测阈值。

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setLogicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例


```
{
    "msgType": "setLogic",
    "devID": "492C210803024166",
    "data": {
        "logic1": {
            "cdt": "0",
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
            "AI": "0"
        },
        "logic2": {
            "cdt": "0",
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
            "AI": "0"
        },
        "logic3": {
            "cdt": "0",
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
            "AI": "0"
        },
        "logic4": {
            "cdt": "0",
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
            "AI": "0"
        },
        "logic5": {
            "cdt": "0",
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
```

```
            "AI": "0"
        },
        "logic6": {
            "cdt": "0",
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
            "AI": "0"
        },
        "logic7": {
            "cdt": "0",
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
            "AI": "0"
        },
        "logic8": {
            "cdt": "0",
            "addr": "00",
            "inReg": 0,
            "outReg": 0,
            "DO": 0,
            "AI": "0"
        }
    }
}
```


响应


```
{
    "devID": "492C210803024166",
    "msgType": "setLogicAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631499165"
}
```

1.3.闹钟

1.3.1.读取闹钟

说明：ZHC492S 支持定时（北京时间）执行某项操作（重启/DO 动作）

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getClock
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getClockAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
clockx	是	字符	闹钟根节点。x=1~8。
en	是	字符	闹钟使能。0:禁用 1:启用
hh	是	字符	时
mm	是	字符	分
ss	是	字符	秒
actType	是	字符	动作类型。0:重启 1:DO 动作
actor	是	字符	动作执行者。取值范围:0~3。
DO	是	字符	DO 动作。0:常开 1:常闭

示例

```
{
    "msgType": "getClock",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getClockAck",
```

```
"data": {
    "clock1": {
        "en": 0,
        "hh": "0",
        "mm": "0",
        "ss": "0",
        "actType": 0,
        "actor": 0,
        "DO": 0
    },
    "clock2": {
        "en": 0,
        "hh": "0",
        "mm": "0",
        "ss": "0",
        "actType": 0,
        "actor": 0,
        "DO": 0
    },
    "clock3": {
        "en": 0,
        "hh": "0",
        "mm": "0",
        "ss": "0",
        "actType": 0,
        "actor": 0,
        "DO": 0
    },
    "clock4": {
        "en": 0,
        "hh": "0",
        "mm": "0",
        "ss": "0",
        "actType": 0,
        "actor": 0,
        "DO": 0
    },
    "clock5": {
        "en": 0,
        "hh": "0",
        "mm": "0",
        "ss": "0",
        "actType": 0,
        "actor": 0,
```

```
                "DO": 0
            },
            "clock6": {
                "en": 0,
                "hh": "0",
                "mm": "0",
                "ss": "0",
                "actType": 0,
                "actor": 0,
                "DO": 0
            },
            "clock7": {
                "en": 0,
                "hh": "0",
                "mm": "0",
                "ss": "0",
                "actType": 0,
                "actor": 0,
                "DO": 0
            },
            "clock8": {
                "en": 0,
                "hh": "0",
                "mm": "0",
                "ss": "0",
                "actType": 0,
                "actor": 0,
                "DO": 0
            }
        },
        "timestamp": "1631499611"
}
```


1.3.2.写入闹钟

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setClock
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
clockx	是	字符	闹钟根节点。x=1~8。
en	否	字符	闹钟使能。0:禁用 1:启用
hh	否	字符	时
mm	否	字符	分
ss	否	字符	秒
actType	否	字符	动作类型。0:重启 1:DO 动作
actor	否	字符	动作执行者。取值范围:0~3。
DO	否	字符	DO 动作。0:常开 1:常闭

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setClockAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setClock",
    "devID": "492C210803024166",
    "data": {
        "clock1": {
            "en": 0,
            "hh": "0",
            "mm": "0",
            "ss": "0",
            "actType": 0,
            "actor": 0,
            "DO": 0
        },
        "clock2": {
            "en": 0,
            "hh": "0",
            "mm": "0",
            "ss": "0",
```

```
            "actType": 0,
            "actor": 0,
            "DO": 0
        },
        "clock3": {
            "en": 0,
            "hh": "0",
            "mm": "0",
            "ss": "0",
            "actType": 0,
            "actor": 0,
            "DO": 0
        },
        "clock4": {
            "en": 0,
            "hh": "0",
            "mm": "0",
            "ss": "0",
            "actType": 0,
            "actor": 0,
            "DO": 0
        },
        "clock5": {
            "en": 0,
            "hh": "0",
            "mm": "0",
            "ss": "0",
            "actType": 0,
            "actor": 0,
            "DO": 0
        },
        "clock6": {
            "en": 0,
            "hh": "0",
            "mm": "0",
            "ss": "0",
            "actType": 0,
            "actor": 0,
            "DO": 0
        },
        "clock7": {
            "en": 0,
            "hh": "0",
            "mm": "0",
```

```
                "ss": "0",
                "actType": 0,
                "actor": 0,
                "DO": 0
            },
            "clock8": {
                "en": 0,
                "hh": "0",
                "mm": "0",
                "ss": "0",
                "actType": 0,
                "actor": 0,
                "DO": 0
            }
        }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "setClockAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631500413"
}
```

1.4.AI

1.4.1 读取 AI 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
enRpt	是	整型	主动上报使能。0:禁用 1:启用
cyc	是	字符	上报周期

示例

```
{
    "msgType": "getAIBasic",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getAIBasicAck",
    "data": {
        "enRpt": 0,
        "cyc": "30"
    },
    "timestamp": "1631501345"
}
```

1.4.2.写入 AI 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAIBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
enRpt	否	整型	主动上报使能。0:禁用 1:启用
cyc	否	字符	上报周期

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAIBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setAIBasic",
    "devID": "492C210803024166",
    "data": {
        "enRpt": 0,
        "cyc": "30"
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setAIBasicAck",
    "data": {
        "status": "0"
    },
```

```
    "timestamp": "1631501681"
}
```

1.4.3 读取 AI 主动上报条件

说明：ZHC492S 支持 AI **进入/退出**阈值区间触发上报。

当定时上报与阈值同时开启时，阈值触发会刷新“定时上报计数器”。

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIRpt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIRptAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
rulex	是	字符	上报条件根节点。x=1~4
mode	是	整型	使能标志。0:禁用 1:进入区间上报 2:退出区间上报
min	是	字符	阈值下限
max	是	字符	阈值上限

示例

```
{
    "msgType": "getAIRpt",
    "devID": "492C210803024166",
    "data": {}
}
响应：
{
    "devID": "492C210803024166",
    "msgType": "getAIRptAck",
    "data": {
        "rule1": {
            "mode": 0,
```

```
                "min": "0",
                "max": "0"
            },
            "rule2": {
                "mode": 0,
                "min": "0",
                "max": "0"
            },
            "rule3": {
                "mode": 0,
                "min": "0",
                "max": "0"
            },
            "rule4": {
                "mode": 0,
                "min": "0",
                "max": "0"
            }
        },
        "timestamp": "1631502000"
}
```


1.4.4.写入 AI 主动上报条件

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAIRpt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
rulex	否	字符	上报条件根节点。x=1~4
mode	否	整型	使能标志。0:禁用 1:进入区间上报 2:退出区间上报
min	否	字符	阈值下限
max	否	字符	阈值上限

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAIBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setAIRpt",
    "devID": "492C210803024166",
    "data": {
        "rule1": {
            "mode": 1,
            "min": "4000",
            "max": "2000"
        },
        "rule2": {
            "mode": 1,
            "min": "0",
            "max": "0"
        },
        "rule3": {
            "mode": 0,
            "min": "0",
            "max": "0"
        },
        "rule4": {
            "mode": 0,
            "min": "0",
            "max": "0"
        }
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setAIRptAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631501681"
}
```

1.5.DI

1.5.1 读取 DI 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getDIBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getDIBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
enRpt	是	整型	主动上报使能。0:禁用 1:启用
cyc	是	字符	上报周期

示例

```
{
    "msgType": "getDIBasic",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getDIBasicAck",
    "data": {
        "enRpt": 0,
        "cyc": "30"
    },
    "timestamp": "1631501345"
}
```

1.5.2.写入 DI 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setDIBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
enRpt	否	整型	主动上报使能。0:禁用 1:启用
cyc	否	字符	上报周期

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setDIBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setDIBasic",
    "devID": "492C210803024166",
    "data": {
        "enRpt": 0,
        "cyc": "30"
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setDIBasicAck",
    "data": {
        "status": "0"
    },
```

```
    "timestamp": "1631501681"
}
```

1.6.DO

1.6.1.读取 DO 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getDOBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getDOBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
enRpt	是	整型	主动上报使能。0:禁用 1:启用
rstStatus	是	整型	DO 重启状态。0:不保持 1:保持上一次执行的状态

示例

```
{
    "msgType": "getDOBasic",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getDOBasicAck",
    "data": {
        "enRpt": 0,
        "rstStatus": 0
    },
```

```
    "timestamp": "1631504335"
}
```

1.6.2.写入 DO 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setDOBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
enRpt	是	整型	主动上报使能。0:禁用 1:启用
rstStatus	是	整型	DO 重启状态。0:不保持 1:保持上一次执行的状态

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setDOBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setDOBasic",
    "devID": "492C210803024166",
    "data": {
        "enRpt": 0,
        "rstStatus": 0
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setDOBasicAck",
```

```
    "data": {
        "status": "0"
    },
    "timestamp": "1631501681"
}
```


1.6.3.读取 DO 输出保持时间

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getDOHold
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getDOHoldAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
holdx	是	字符	DO 输出状态维持时间，单位 s/秒，DO 状态改变后，计数器值为“holdx”，按秒递减，归零后 DO 回到上次执行的状态。x=1~4

示例


```
{
    "msgType": "getDOHold",
    "devID": "492C210803024166",
    "data": {}
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "getDOHoldAck",
    "data": {
        "hold1": "0",
```

```
            "hold2": "0",
            "hold3": "0",
            "hold4": "0"
        },
        "timestamp": "1631504632"
}
```


1.6.4.写入 DO 输出保持时间

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setDOHold
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
holdx	是	字符	DO 输出状态维持时间，单位 s/秒，DO 状态改变后，计数器值为“holdx”，按秒递减，归零后 DO 回到上次执行的状态。x=1~4

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setDOHoldAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例


```
{
    "msgType": "setDOHold",
    "devID": "492C210803024166",
    "data": {
        "hold1": "1",
        "hold2": "5",
        "hold3": "9",
```

```
            "hold4": "13"
        }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "setDOHoldAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631504812"
}
```


1.6.5.读取 DO 定时翻转时间

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getDOTurn
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getDOTurnAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
turnx	是	字符	DO 定时翻转时间，单位 s/秒，DO 执行状态按“turnx”往复。x=1~4

示例


```
{
    "msgType": "getDOTurn",
    "devID": "492C210803024166",
    "data": {}
}
```


响应：


```
{
```

```
    "devID": "492C210803024166",
    "msgType": "getDOTurnAck",
    "data": {
        "turn1": "0",
        "turn2": "0",
        "turn3": "0",
        "turn4": "0"
    },
    "timestamp": "1631505191"
}
```

1.6.6.写入 DO 输出保持时间

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setDOTurn
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
turnx	是	字符	DO 定时翻转时间，单位 s/秒。x=1~4

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setDOTurnAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setDOTurn",
    "devID": "492C210803024166",
    "data": {
```

```
            "turn1": "2",
            "turn2": "3",
            "turn3": "4",
            "turn4": "5"
        }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setDOTurnAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631505288"
}
```

1.7.串口

1.7.1.读取串口基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getUartBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getUartBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
br	是	字符	波特率。1200~115200
wl	是	整型	0:8 位 1:9 位
sb	是	整型	0:1 位 1:1.5 位 2:2 位
pa	是	整型	0:无校验 1:奇校验 2:偶校验

示例

```
{
    "msgType": "getUartBasic",
    "devID": "492C210803024166",
    "data": {}
}
响应：
{
  "devID": "492C210803024166",
  "msgType": "getUartBasicAck",
  "data": {
    "br": "115200",
    "wl": 0,
    "sb": 0,
    "pa": 0
  },
  "timestamp": "1631515814"
}
```

1.7.2.写入串口基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setUartBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
br	是	字符	波特率。1200~115200
wl	是	整型	0:8 位 1:9 位
sb	是	整型	0:1 位 1:1.5 位 2:2 位
pa	是	整型	0:无校验 1:奇校验 2:偶校验

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setUartBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setUartBasic",
    "devID": "492C210803024166",
    "data": {
        "br": "115200",
        "pa": 0,
        "wl": 0,
        "sb": 0
    }
}
响应：
{
    "devID": "492C210803024166",
    "msgType": "setUartBasicAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631501681"
}
```

1.8.网络

1.8.1.读取网络基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getLTEBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getLTEBasicAck
devID	是	字符	设备唯一识别码，16 位

data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
sim	是	字符	SIM 卡号
signal	是	整型	信号强度。 解析参考：\"LTE\",rssi,**rsrp**,sinr,rsrq 极好：rsrp > -85 好：rsrp -85 ~ -95 中：rsrp -95 ~ -105 差：rsrp -105~ -115 极差：rsrp < -115

示例

```
{
    "msgType": "getLTEBasic",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
  "devID": "492C210803024166",
  "msgType": "getLTEBasicAck",
  "data": {
    "sim": "89860474282090004420",
    "signal": "\"LTE\",56,-89,95,-12"
  },
  "timestamp": "1631516352"
}
```

1.8.2.读取模组直接通信

说明：“配置模式”下使能“模组直接通信”，可实现通过串口直接与 LTE 模组通信

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getModule
devID	是	字符	设备唯一识别码，16 位

data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getModuleAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
super	是	整型	0:禁用 1:启用

示例

```
{
    "msgType": "getModule",
    "devID": "492C210803024166",
    "data": {}
}
响应：
{
  "devID": "492C210803024166",
  "msgType": "getModuleAck",
  "data": {
    "super": 0
  },
  "timestamp": "1631516638"
}
```

1.8.3.写入模组直接通信

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setModule
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
super	是	整型	0:禁用 1:启用

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setModuleAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setModule",
    "devID": "492C210803024166",
    "data": {
        "super": 0
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setModuleAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631516873"
}
```

1.8.4.读取 APN 参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAPN
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述

msgType	是	字符	getAPNAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
addr	是	字符	APN 地址。 最大 22 字符
user	是	字符	用户名。 最大 22 字符
pass	是	字符	密码。最大 22 字符

示例

```
{
    "msgType": "getAPN",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
  "devID": "492C210803024166",
  "msgType": "getAPNAck",
  "data": {
    "addr": "",
    "user": "",
    "pass": ""
  },
  "timestamp": "1631517003"
}
```

1.8.5.写入 APN 参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAPN
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
addr	是	字符	APN 地址。 最大 22 字符
user	是	字符	用户名。 最大 22 字符

pass	是	字符	密码。最大 22 字符

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAPNAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setAPN",
    "devID": "492C210803024166",
    "data": {
        "addr": "CNNET",
        "user": "CNNET",
        "pass": "123456"
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setAPNAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631516873"
}
```

1.8.6.读取 Socket

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocket
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式：

字段	必须	类型	描述
index	是	字符	Socket 索引。index = 1~2

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	Socket 索引。index = 1~2
enable	是	整型	使能标志。0:禁用 1:启用
mode	是	整型	0:TCP 1:MQTT
desIP	是	字符	cloud.iotrouter.cn
desPort	是	字符	56000
regMode	是	整型	注册包模式。0:禁用 1:默认 2:自定义 3:ID 4:SIM 卡号
regCdt	是	整型	触发模式。0:连接发送 1:数据携带 2:以上两种
regPkg	是	字符	注册包内容。<=200 字符。16 进制字符串格式，例如：想要写入“123”，则 regPkg=“313233”
hbtMode	是	整型	心跳包模式。0:禁用 1:启用
hbtCyc	是	字符	心跳周期。单位：s/秒
hbtPkg	是	字符	心跳包内容。<=20 字符。6 进制字符串格式，例如：想要写入“123”，则 regPkg=“313233”

示例

```
{
    "msgType": "getSocket",
    "devID": "492C210803024166",
    "data": {
        "index": "1"
    }
}
```

响应：

```
{
  "devID": "492C210803024166",
  "msgType": "getSocketAck",
  "data": {
    "index": "2",
    "enable": 0,
```

```
        "mode": 0,
        "desIP": "cloud.iotrouter.cn",
        "desPort": "56000",
        "regMode": 1,
        "regCdt": 0,
        "regPkg":
"415F315F312E343932433231303830333032343136362E4D54497A4E4455322E393235623566
313865656237396134353363303930343665356563363531343 12E41424344453132333435",
        "hbtMode": 1,
        "hbtCyc": "30",
        "hbtPkg": "50494E472150494E472150494E4721"
    },
    "timestamp": "6"
}
```


1.8.7.写入 Socket

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocket
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	Socket 索引。index = 1~2
enable	是	整型	使能标志。0:禁用 1:启用
mode	是	整型	0:TCP 1:MQTT
desIP	是	字符	cloud.iotrouter.cn
desPort	是	字符	56000
regMode	是	整型	注册包模式。0:禁用 1:默认 2:自定义 3:ID 4:SIM 卡号
regCdt	是	整型	触发模式。0:连接发送 1:数据携带 2:以上两种
regPkg	是	字符	注册包内容。<=200 字符。16 进制字符串格式，例如：想要写入“123”，则 regPkg=“313233”
hbtMode	是	整型	心跳包模式。0:禁用 1:启用
hbtCyc	是	字符	心跳周期。单位：s/秒
hbtPkg	是	字符	心跳包内容。<=20 字符。6 进制字符串格式，例如：想要写入“123”，则 regPkg=“313233”

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketAck
devID	是	字符	设备唯一识别码，16 位

data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例


```
{
    "msgType": "setSocket",
    "devID": "492C210803024166",
    "data": {
        "index": "1",
        "enable": 1,
        "mode": 0,
        "desIP": "cloud.iotrouter.cn",
        "desPort": "56000",
        "regMode": 2,
        "regCdt": 0,
        "regPkg": "313233",
        "hbtMode": 1,
        "hbtCyc": "30",
        "hbtPkg": "313233"
    }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "setSocketAck",
    "data": {
        "index": "1",
        "status": "0"
    },
    "timestamp": "1631516873"
}
```


1.8.8.读取 MQTT

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getMqtt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式：

字段	必须	类型	描述
index	是	字符	Socket 索引。index = 1

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getMqttAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	Socket 索引。index = 1
id	是	字符	客户端 ID。最大 60 字符
name	是	字符	客户端名称。最大 60 字符
pw	是	字符	密码。最大 60 字符
sub	是	字符	订阅主题。最大 60 字符
pub	是	字符	发布主题。最大 60 字符
alive	是	字符	保活时间。最大 60 字符

示例

```
{
    "msgType": "getMqtt",
    "devID": "492C210803024166",
    "data": {
        "index": "1"
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getMqttAck",
    "data": {
        "index": "1",
        "id": "123",
```

```
            "name": "www",
            "pw": "yyy",
            "sub": "/public/zhcpub",
            "pub": "/public/zhcsub",
            "alive": "30"
        },
        "timestamp": "6"
}
```

1.8.9.写入 MQTT

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setMqtt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	Socket 索引。index = 1
id	是	字符	客户端 ID。最大 60 字符
name	是	字符	客户端名称。最大 60 字符
pw	是	字符	密码。最大 60 字符
sub	是	字符	订阅主题。最大 60 字符
pub	是	字符	发布主题。最大 60 字符
alive	是	字符	保活时间。最大 60 字符

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setMqttAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
```

```
    "msgType": "setMqtt",
    "devID": "492C210803024166",
    "data": {
        "index": "1",
        "id": "zhc492C",
        "name": "111_A",
        "pw": "123456",
        "sub": "/public/zhcpub",
        "pub": "/public/zhcsub",
        "alive": "30"
    }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "setMqttAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631518608"
}
```


2.硬件接口

2.1.DO

2.1.1.读取 DO

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getDOValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getDOValueAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
DOx	是	整型	DO 输出值。0:常开 1:常闭。x = 1~4

示例

```
{
    "msgType": "getDOValue",
    "devID": "492C210803024166",
    "data": {}
}
响应：
{
    "devID": "492C210803024166",
    "msgType": "getDOValueAck",
    "data": {
        "DO1": 1,
        "DO2": 0,
        "DO3": 0,
        "DO4": 0
    },
    "timestamp": "1631519671"
}
```

2.1.2.写入 DO

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setDOValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
DOx	是	整型	DO 输出值。0:常开 1:常闭。x = 1~4

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setDOValueAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例

```
{
    "msgType": "setDOValue",
    "devID": "492C210803024166",
    "data": {
        "DO2": 1
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "setDOValueAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631518608"
}
```

2.2.DI

2.2.1.读取 DI

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getDIValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getDIValueAck

devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
DIx	是	整型	DI 检测值。0:断开 1:闭合。x = 1~4

示例

```
{
    "msgType": "getDIValue",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getDIValueAck",
    "data": {
        "DI1": 1,
        "DI2": 1,
        "DI3": 0,
        "DI4": 0
    },
    "timestamp": "1631520428"
}
```

2.3.AI

2.3.1.读取 AI

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述

msgType	是	字符	getAIValueAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
AIx	是	整型	AI 检测值。单位 uA/微安

示例

```
{
    "msgType": "getAIValue",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
  "devID": "492C210803024166",
  "msgType": "getAIValueAck",
  "data": {
    "AI1": "1000",
    "AI2": "4000",
    "AI3": "0",
    "AI4": "0"
  },
  "timestamp": "1631520844"
}
```

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setDIBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
enRpt	否	字符	状态改变使能。65535/启用 0/禁用
cyc	是	字符	循环上报周期。单位：s/秒

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setDIBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

3.数据点

3.1.读取数据点数量

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getMonitorCnt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getMonitorCntAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
count	是	字符	节点数量。<=10

示例：

```
{
    "msgType": "getMonitorCnt",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getMonitorCntAck",
    "data": {
        "count": "0"
    },
    "timestamp": "1631520988"
}
```

3.2.根据索引读取数据点

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getMonitorIndex
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	数据点索引。index < count

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getMonitorIndexAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	数据点索引
id	是	字符	数据点唯一识别码，最大 12 位
name	是	字符	数据点名称/备注，最大 12 位
staNo	是	字符	数据点对应的 Modbus 地址，16 进制。例如：0x55，则 staNo=55
unit	是	字符	单位，最大 16 位
regType	是	整型	寄存器类型。0:线圈 1:触点 2:输入寄存器 3:保持寄存器
regAddr	是	字符	寄存器地址。16 进制。例如：0x0001，则 regAddr=0001
regNum	是	字符	寄存器数量
dataType	是	整型	数据类型。 0: 16 位无符号整数 AB 1: 16 位无符号整数 BA 2: 16 位有符号整数 AB 3: 16 位有符号整数 BA 4:单精度浮点数 ABCD 5:单精度浮点数 CDAB 6:单精度浮点数 BADC 7:单精度浮点数 DCBA 8: 32 位无符号整数 ABCD 9: 32 位无符号整数 CDAB 10: 32 位无符号整数 BADC 11: 32 位无符号整数 DCBA 12: 32 位有符号整数 ABCD 13: 32 位有符号整数 CDAB 14: 32 位有符号整数 BADC

			15: 32 位有符号整数 DCBA 16:开关量 17:字符
decimalPos	是	整型	小数点位数。0:0 位 1:1 位 2:2 位 3:3 位 4:4 位
logic	是	整型	本地计算。0:禁用 1:启用比例转换 2:启用整除
dataMin	是	字符	数值下限
dataMax	是	字符	数值上限
propMin	是	字符	量程下限
propMax	是	字符	量程上限
divide	是	字符	除数

示例：


```
{
    "msgType": "getMonitorIndex",
    "devID": "492C210803024166",
    "data": {
        "index": "0"
    }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "getMonitorIndexAck",
    "data": {
        "index": "0",
        "id": "123",
        "name": "AI_3",
        "staNo": "01",
        "unit": "mA",
        "regType": "2",
        "regAddr": "0002",
        "regNum": "1",
        "dataType": "0",
        "decimalPos": "2",
        "logic": "2",
        "dataMin": "0",
        "dataMax": "0",
        "propMin": "0",
        "propMax": "0",
        "divide": "10"
    },
    "timestamp": "1631585078"
}
```

3.3.根据 ID 获取数据点

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getMonitor
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
id	是	字符	数据点唯一识别码。最大 12 位

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getMonitorAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
id	是	字符	数据点唯一识别码，最大 12 位
name	是	字符	数据点名称/备注，最大 12 位
staNo	是	字符	数据点对应的 Modbus 地址，16 进制。例如：0x55，则 staNo=55
unit	是	字符	单位，最大 16 位
regType	是	整型	寄存器类型。0:线圈 1:触点 2:输入寄存器 3:保持寄存器
regAddr	是	字符	寄存器地址。16 进制。例如：0x0001，则 regAddr=0001
regNum	是	字符	寄存器数量
dataType	是	整型	数据类型。 0: 16 位无符号整数 AB 1: 16 位无符号整数 BA 2: 16 位有符号整数 AB 3: 16 位有符号整数 BA 4:单精度浮点数 ABCD 5:单精度浮点数 CDAB 6:单精度浮点数 BADC 7:单精度浮点数 DCBA 8: 32 位无符号整数 ABCD 9: 32 位无符号整数 CDAB 10: 32 位无符号整数 BADC 11: 32 位无符号整数 DCBA 12: 32 位有符号整数 ABCD 13: 32 位有符号整数 CDAB 14: 32 位有符号整数 BADC 15: 32 位有符号整数 DCBA

			16:开关量 17:字符
decimalPos	是	整型	小数点位数。0:0 位 1:1 位 2:2 位 3:3 位 4:4 位
logic	是	整型	本地计算。0:禁用 1:启用比例转换 2:启用整除
dataMin	是	字符	数值下限
dataMax	是	字符	数值上限
propMin	是	字符	量程下限
propMax	是	字符	量程上限
divide	是	字符	除数

示例：


```
{
    "msgType": "getMonitor",
    "devID": "492C210803024166",
    "data": {
        "id": "123"
    }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "getMonitorAck",
    "data": {
        "id": "123",
        "name": "AI_3",
        "staNo": "01",
        "unit": "mA",
        "regType": "2",
        "regAddr": "0002",
        "regNum": "1",
        "dataType": "0",
        "decimalPos": "2",
        "logic": "2",
        "dataMin": "0",
        "dataMax": "0",
        "propMin": "0",
        "propMax": "0",
        "divide": "10"
    },
    "timestamp": "1631585338"
}
```

3.4.添加数据点

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	addMonitor
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
id	是	字符	数据点唯一识别码，最大 12 位
name	是	字符	数据点名称/备注，最大 12 位
staNo	是	字符	数据点对应的 Modbus 地址，16 进制。例如：0x55，则 staNo=55
unit	否	字符	单位，最大 16 位
regType	否	整型	寄存器类型。0:线圈 1:触点 2:输入寄存器 3:保持寄存器
regAddr	否	字符	寄存器地址。16 进制。例如：0x0001，则 regAddr=0001
regNum	否	字符	寄存器数量
dataType	否	整型	数据类型。 0: 16 位无符号整数 AB 1: 16 位无符号整数 BA 2: 16 位有符号整数 AB 3: 16 位有符号整数 BA 4:单精度浮点数 ABCD 5:单精度浮点数 CDAB 6:单精度浮点数 BADC 7:单精度浮点数 DCBA 8: 32 位无符号整数 ABCD 9: 32 位无符号整数 CDAB 10: 32 位无符号整数 BADC 11: 32 位无符号整数 DCBA 12: 32 位有符号整数 ABCD 13: 32 位有符号整数 CDAB 14: 32 位有符号整数 BADC 15: 32 位有符号整数 DCBA 16:开关量 17:字符
decimalPos	否	整型	小数点位数。0:0 位 1:1 位 2:2 位 3:3 位 4:4 位
logic	否	整型	本地计算。0:禁用 1:启用比例转换 2:启用整除
dataMin	否	字符	数值下限
dataMax	否	字符	数值上限
propMin	否	字符	量程下限
propMax	否	字符	量程上限
divide	否	字符	除数

响应帧格式：

字段	必须	类型	描述

msgType	是	字符	setMqttAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例：


```
{
    "msgType": "addMonitor",
    "devID": "492C210803024166",
    "data": {
        "id": "124",
        "name": "AI_1",
        "staNo": "01",
        "unit": "mA",
        "regType": 2,
        "regAddr": "0000",
        "regNum": "1",
        "dataType": 0,
        "decimalPos": 1,
        "logic": 0,
        "dataMin": "",
        "dataMax": "",
        "propMin": "",
        "propMax": "",
        "divide": ""
    }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "addMonitorAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631585482"
}
```

3.5.编辑数据点

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	editMonitor
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
id	是	字符	数据点唯一识别码，最大 12 位
name	是	字符	数据点名称/备注，最大 12 位
staNo	是	字符	数据点对应的 Modbus 地址，16 进制。例如：0x55，则 staNo=55
unit	否	字符	单位，最大 16 位
regType	否	整型	寄存器类型。0:线圈 1:触点 2:输入寄存器 3:保持寄存器
regAddr	否	字符	寄存器地址。16 进制。例如：0x0001，则 regAddr=0001
regNum	否	字符	寄存器数量
dataType	否	整型	数据类型。 0: 16 位无符号整数 AB 1: 16 位无符号整数 BA 2: 16 位有符号整数 AB 3: 16 位有符号整数 BA 4:单精度浮点数 ABCD 5:单精度浮点数 CDAB 6:单精度浮点数 BADC 7:单精度浮点数 DCBA 8: 32 位无符号整数 ABCD 9: 32 位无符号整数 CDAB 10: 32 位无符号整数 BADC 11: 32 位无符号整数 DCBA 12: 32 位有符号整数 ABCD 13: 32 位有符号整数 CDAB 14: 32 位有符号整数 BADC 15: 32 位有符号整数 DCBA 16:开关量 17:字符
decimalPos	否	整型	小数点位数。0:0 位 1:1 位 2:2 位 3:3 位 4:4 位
logic	否	整型	本地计算。0:禁用 1:启用比例转换 2:启用整除
dataMin	否	字符	数值下限
dataMax	否	字符	数值上限
propMin	否	字符	量程下限
propMax	否	字符	量程上限
divide	否	字符	除数

响应帧格式：

字段	必须	类型	描述

msgType	是	字符	setMqttAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例：


```
{
    "msgType": "editMonitor",
    "devID": "492C210803024166",
    "data": {
        "id": "124",
        "name": "AI_1",
        "staNo": "01",
        "unit": "mA",
        "regType": 2,
        "regAddr": "0000",
        "regNum": "1",
        "dataType": 0,
        "decimalPos": 2,
        "logic": 2,
        "dataMin": "0",
        "dataMax": "0",
        "propMin": "0",
        "propMax": "0",
        "divide": "1"
    }
}
```


响应：


```
{
    "devID": "492C210803024166",
    "msgType": "editMonitorAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631585688"
}
```

3.6.删除数据点

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	delMonitor
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
id	是	字符	数据点唯一识别码，最大 12 位

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	delMonitorAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
status	是	字符	0:成功 1:失败

示例：

```
{
    "msgType": "delMonitor",
    "devID": "492C210803024166",
    "data": {
        "id": "124"
    }
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": " delMonitorAck",
    "data": {
        "status": "0"
    },
    "timestamp": "1631585688"
}
```

3.7.获取数据点采集值

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getMonitorValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式

字段	必须	类型	描述
id	否	字符	数据点唯一识别码。最大 12 位。当 data 为空时，返回所有点位的采集值

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getMonitorValueAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
“name”	是	字符	数据点根节点。数据点名称
status	是	字符	超时标志。0:超时 1:正常
value	是	字符	采集值
addr	是	字符	寄存器地址。16 进制

示例：

```
{
    "msgType": "getMonitorValue",
    "devID": "492C210803024166",
    "data": {}
}
```

响应：

```
{
    "devID": "492C210803024166",
    "msgType": "getMonitorValueAck",
    "data": {
        "AI_3": {
            "status": "1",
```

```
            "value": "1637.40mA",
            "addr": "0x0002"
        },
        "AI_1": {
            "status": "1",
            "value": "10.00mA",
            "addr": "0x0000"
        }
    },
    "timestamp": "1631586042"
}
```


3.8.数据点主动上报

上报帧格式：

字段	必须	类型	描述
msgType	是	字符	dataPointRpt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
“name”	是	字符	数据点根节点。数据点名称
status	是	字符	超时标志。0:超时 1:正常
value	是	字符	采集值
addr	是	字符	寄存器地址。16 进制

示例：


```
{
    "devID": "492C210803024166",
    "msgType": "dataPointRpt",
    "data": {
        "AI_3": {
            "status": "1",
            "value": "1637.40mA",
            "addr": "0x0002"
        },
        "AI_1": {
            "status": "1",
```

```
                "value": "10.00mA",
                "addr": "0x0000"
            }
        },
        "timestamp": "1631586372"
}
```