APRUS Lua-zyzn 配置说明

概述

本章主要为 APRUS Lua-ZYZN 协议的相关配置说明，该协议主要针对支持(腾冉电气)ZYZN 协议的 EMS 设备。APRUS 适配器可通过 ZYZN 协议与对接设备进行通信。而其中 APRUS 的 Lua 包含 apru.lua 和 config.lua 两个文件，客户只需配置 config.lua 就可以对支持 ZYZN 协议的设备进行数据采集。如需修改 APRUS.lua 文件的内容时，请咨询相关的技术人员，随意修改会导致适配器不能正常工作，所以此文档主要介绍 config.lua 内容。

对于 ZYZN 协议，APRUS 适配器支持单召遥信（0x02）和单召遥测（0x03）指令；支持单召遥控（0x06）和单召遥调（0x10）控制指令，云平台可以下发整数/浮点数值，因 zyzn 协议限制，遥调只支持有符号整型控制值，所以小数部分会被舍弃；对于遥控指令，只区分 0 值和非 0 值，非 0 则为 1，0 或者 0.0 均为 0；支持心跳包，长时间不通信也无需发送心跳包维持通信；支持总召（0x20）指令，但是未使用。

另外，腾冉电气 EMS，对与过快的通信请求会导致 EMS 端回复错误。所以 config.lua 中需要配置采集间隔以保证不丢失通信连接并且可以获取正确的数据。

1 APRUS.lua 配置说明

demo 示例

```
package.cpath = "./?.so"
package.path = "./?.lua"
cjson = require "cjson"
config = require "config"
require "zyzn"

function act_control(m, json)
    for k, v in pairs(json) do
        if k ~= "Act" then
```

```
                zyzn.write(zyznObj, cjson.encode({ ["name"] = k, ["val"] = v }))
            end
        end
end

function mqttdata_handle(m, topic, data)
        local json = cjson.decode(data)
        if json.Act == "Control" then
            act_control(m, json)
        end
end

function mqttsys_handle(m, code)
        if code == 0 then
            zyzn.stop(zyznObj)
        elseif code == 1 then
            zyzn.run(zyznObj)
        end
end

function zyzn_handle(obj, name, code, data)
        mqtt.publish(mqtt3Obj, name, "r", data)
end

function zyzn_load_collectnodes(obj, nodes)
        for k, v in pairs(nodes)
        do
            zyzn.addcnode(obj, cjson.encode(v))
        end
end

function zyzn_load_varnodes(obj, nodes)
        for k, v in pairs(nodes)
        do
            zyzn.addvnode(obj, cjson.encode(v))
        end
end

function init()
        mqtt3Obj = mqtt.new()
        mqtt.config(mqtt3Obj, nil, "123.249.100.4", "1883", "tr", "")
          --mqtt.config(mqtt3Obj, nil, "123.249.100.4", "1883", "user", "mix123")                    --连接腾冉电气服务器
        user.setluaver(config.AprusX.luaver)
        user.setdevinfo(config.AprusX.devinfo)
        user.ipconfig(config.AprusX.ipmode, config.AprusX.inet_addr, config.AprusX.netmask)
        --      user.reset_eth0()  --if you want to set eth0   use this interface
```

```
    --    user.ipconfig_eth0("manual", "10.100.1.123", "255.255.255.0", "10.100.1.1", "114.114.114.114")
end

function start()
    init()
    zyznObj = zyzn.new("zyzn")
    zyzn.config(zyznObj, cjson.encode(config.zyzn.Device))
    zyzn_load_collectnodes(zyznObj, config.zyzn.ColNode)
    zyzn_load_varnodes(zyznObj, config.zyzn.VarNode)

    ----------RS485 Forward Config----------
    -- fwCfg = config.Forward485
    -- fw485Obj = forward485.new()
    -- forward485.config(fw485Obj, fwCfg.device.rate, fwCfg.device.databit, fwCfg.device.stopbit, fwCfg.device.parity)
    -- forward485.autofilter(fw485Obj, 1, 60)
    -- forward485.timeout(fw485Obj, 3000)
    -- forward485.run(fw485Obj)
    ------------------------------------
    mqtt.run(mqtt3Obj)

    while true do
        local msg = user.waitmsg()
        if msg.from == "mqtt-sys" then
            mqttsys_handle(msg.session, msg.code)
        elseif msg.from == "mqtt-msg" then
            mqttdata_handle(msg.session, msg.topic, msg.payload)
        elseif msg.from == "zyzn" then
            zyzn_handle(msg.obj, msg.name, msg.code, msg.data)
        end
    end
end

start()
```

1.1 导入 zyzn 协议支持库

语句	require "zyzn"
说明	导入 zyzn 协议支持库

1.2 加载 config.lua 配置文件

语句	config = require "config"
说明	导入 config.lua 中的配置信息(下一节介绍 config.lua)

1.3 创建 zyzn 对象

语句	zyznObj= zyzn.new("zyzn")
说明	创建 zyzn 对象，返回 zyznObj 供全局使用

1.4 配置 zyzn 对象接口参数

语句	zyzn.config(zyznObj, cjson.encode(config.zyzn.Device))
说明	配置 zyzn 通信接口 RS485 参数

1.5 添加 zyzn 采集节点

语句	zyzn_load_collectnodes(zyznObj ,config.zyzn.ColNode)
说明	添加 config.lua 中所配置的采集节点

1.6 添加 zyzn 上报变量节点

语句	zyzn_load_varnodes(zyznObj,config.zyzn.VarNode)
说明	添加 config.lua 中所配置的变量节点

1.7 运行 zyzn 实例

语句	zyzn.run(zyznObj)
说明	在 mqtt 建立连接后 运行 zyzn 实例 即开始采集+运算

1.8 暂停 zyzn 实例

语句	zyzn.stop(zyznObj)
说明	在 mqtt 连接断开时 暂停 zyzn 实例

1.9 设置 zyzn 实例的调试打印级别

语句	zyzn.debug(zyznObj, debugNumber)
说明	启用详细的 zyzn 调试信息。第一个参数为 zyzn.new 实例的返回值，第二个参数的值为 32 位十六进制数字，这个数值分成 4 个字节，单独每个字节为 1 时开启对应功能，为 0 时关闭对应功能。从低位到高 4 个字节的功能分别是：错误信息、连接信息和一般信息、接收内容详情、发送内容详情。此设置可以帮助您了解设备的运行情况，排查错误，优化性能。请根据需要选择要开启的调试信息级别。

1.10 等待 zyzn 事件

语句	local msg = user.waitmsg()

说明	启动事件等待，该接口会返回全局各种事件，包括 zyzn 对象事件

1.11 zyzn 事件处理

语句	elseif msg.from == "zyzn" then zyzn_handle(msg.obj, msg.name, msg.code, msg.data)
说明	当接收到 zyzn 对象事件时，调用处理函数 zyzn_handle msg.obj 即 new 时所返回的对象 msg.name 即 new 时所配置的对象名称 msg.code 事件码 区分改变上报事件 周期上报事件等等 msg.data 事件数据

1.12 mqtt 数据上报

语句	function zyzn_handle(obj, name, code, data) mqtt.publish(mqtt3Obj, name, "r", data) end
说明	若无需特殊处理，则直接将事件数据通过 mqtt 发送 如需要二次处理可将 data 展开做分析

1.13 反向控制

语句	zyzn.write(zyznObj, cjson.encode({["name"] = k, ["val"] = v}))

说明	反向控制时调用的 zyzn 写入接口

2 config.lua 配置说明

Demo 示例：

```
return
{
    AprusX = {
        ipmode = "none", --auto/manual/none
        inet_addr = "192.168.1.234",
        netmask = "255.255.255.0",
        luaver = "V00.R",
        devinfo = "ModbusRtuDev",
    },
    zyzn = {
        Device = {
            portindex = 1,
            rate = 9600,
            databit = 8,
            stopbit = 1,
            parity = "None",          -- None/Odd/Even
            QueryTimeout = 3000,   --查询超时，单位 ms
            queryInterval = 5000, --采集间隔，单位 ms
        },
        ColNode = {
            --{ ID = 子站地址（数值）, reg = 功能码（字符串）, addr = 数据地址（数值）, ctype = 采集数据类型（字符串）, cnt = 采集长度（数值） },
            { ID = 1, reg = "2", addr = 1,       ctype = "bit",     cnt = 95 },   --遥信点位采集
            { ID = 1, reg = "3", addr = 16385, ctype = "dword", cnt = 473 }, --遥测点位采集；采集结点数量 cnt 不设限制
            { ID = 1, reg = "3", addr = 25089, ctype = "dword", cnt = 10 },   --遥调点位采集

            { ID = 6, reg = "2", addr = 24577, ctype = "bit",     cnt = 1 },    --遥控点位采集
            { ID = 6, reg = "3", addr = 25089, ctype = "dword", cnt = 10 },   --遥调点位采集
            { ID = 6, reg = "3", addr = 16385, ctype = "dword", cnt = 20 },   --遥调点位采集
        },
        VarNode = {
            --{ ID = 子站地址（数值）, reg = 功能码（字符串）, addr = 数据地址（数值）, ctype = 采集数据类型（字符串）, dtype = 上报数据类型（字符串）, cycle = 上报周期（数值，单位：秒），    name = 上报标签（字符串） },
            { ID = 1, reg = "3",   addr = 25089, ctype = "dword", dtype = "float", cycle = 120, name = "L6_3_11111" }, --遥调（上报）
            { ID = 1, reg = "2",   addr = 95,     ctype = "bit",    dtype = "bit",    cycle = 120, name = "L1_3_95" }, --遥信：通信状态（上报）
            { ID = 1, reg = "3",   addr = 16385, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16385" }, --遥测：预充总压（上报）
            { ID = 1, reg = "3",   addr = 16386, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16386" }, --遥测：电池总电压（上报）
            { ID = 1, reg = "3",   addr = 16387, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16387" }, --遥测：电池总电流（上报）
            { ID = 1, reg = "3",   addr = 16388, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16388" }, --遥测：SOC（上报）
            { ID = 1, reg = "3",   addr = 16389, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16389" }, --遥测：SOH（上报）
            { ID = 1, reg = "3",   addr = 16390, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16390" }, --遥测：绝缘值（上报）
            { ID = 1, reg = "3",   addr = 16391, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16391" }, --遥测：正极绝缘值（上报）
            { ID = 1, reg = "3",   addr = 16392, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16392" }, --遥测：负极绝缘值（上报）
```

```
                    { ID = 1, reg = "3",   addr = 16393, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16393" },
    --遥测：最大充电电流（上报）
                    { ID = 1, reg = "3",   addr = 16394, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16394" },
    --遥测：最大放电电流（上报）
                    { ID = 1, reg = "3",   addr = 16395, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16395" },
    --遥测：单体电压最高节序号（上报）
                    { ID = 1, reg = "3",   addr = 16396, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16396" },
    --遥测：单体最高电压值（上报）
                    { ID = 1, reg = "3",   addr = 16397, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16397" },
    --遥测：单体电压最低节序号（上报）
                    { ID = 1, reg = "3",   addr = 16398, ctype = "dword", dtype = "float", cycle = 120, name = "L1_3_16398" },
    --遥测：单体最低电压值（上报）

                    { ID = 6, reg = "3",   addr = 16385, ctype = "dword", dtype = "float", cycle = 120, name = "L6_3_16385" },
    --遥调（上报）
                    { ID = 6, reg = "3",   addr = 16387, ctype = "dword", dtype = "float", cycle = 120, name = "L6_3_16387" },
    --遥调（上报）
                    { ID = 6, reg = "3",   addr = 16388, ctype = "dword", dtype = "float", cycle = 120, name = "L6_3_16388" },
    --遥调（上报）
                    { ID = 6, reg = "3",   addr = 25089, ctype = "dword", dtype = "float", cycle = 120, name = "L6_3_25089" },
    --遥调：制冷点设定（上报）
                    { ID = 6, reg = "3",   addr = 25090, ctype = "dword", dtype = "float", cycle = 120, name = "L6_3_25090" },
    --遥调：冷回差设定（上报）
                    { ID = 6, reg = "3",   addr = 25091, ctype = "dword", dtype = "float", cycle = 120, name = "L6_3_25091" },
    --遥调：加热点设定（上报）
                    { ID = 6, reg = "2",   addr = 24577, ctype = "bit",    dtype = "bit",    bitn = 0,    cycle = 120,
    name = "L6_2_24577" },   --遥控：开关机（上报）

                    { ID = 1, reg = "16", addr = 25089, ctype = "float", dtype = "float", bitn = 0,    cycle = 120,
    name = "CL1_3_25089" }, --遥调（控制）

                    { ID = 6, reg = "16", addr = 25089, ctype = "float", dtype = "float", bitn = 0,    cycle = 120,
    name = "CL6_3_25089" }, --遥调：制冷点设定（控制）
                    { ID = 6, reg = "16", addr = 25090, ctype = "float", dtype = "float", bitn = 0,    cycle = 120,
    name = "CL6_3_25090" }, --遥调：冷回差设定（控制）
                    { ID = 6, reg = "16", addr = 25091, ctype = "float", dtype = "float", bitn = 0,    cycle = 120,
    name = "CL6_3_25091" }, --遥调：加热点设定（控制）
                    { ID = 6, reg = "06", addr = 24577, ctype = "bit",    dtype = "bit",    bitn = 0,    cycle = 120,
    name = "CL6_2_24577" }, --遥控：开关机（控制）
                }
            },
}
```

2.1 AprusX：接口属性

参数	值	说明
ipmode	"auto"/"manual"/"none"	ip 获取方式
inet_addr	"192.168.1.234"	APRUS 的 IP 地址
netmask	"255.255.255.0"	子网掩码
luaver	"MAX.LUA.V032700.R"	APRUS 的 Lua 版本信息，根据实际脚本自行填写
devinfo	"zyznDev"	与 APRUS 对接设备，可根据需求自行填写

2.2 zyzn-Device: 接口属性

参数	值	说明
portindex	整型：1/2	RS485-1/ 2 接口
rate	整型：2400/4800/.../15200	波特率
databit	整型：5/6/7/8	数据位
stopbit	整型：1/2	停止位
parity	字符串：Even/Odd/None	校验
QueryTimeout	整形：1~n	查询超时（单位：毫秒）
queryInterval	整形：1~n	采集间隔（单位：毫秒）

2.3 zyzn-ColNode :采集节点属性

参数	值	说明
ID	整型：1~n	设备 ID
reg	字符串："2"/"3"	寄存器类型
addr	整型：0~n	采集起始地址
ctype	字符串：dword/bit	bit:以位为单位采集（对应遥信、遥控） dword:以双字节为单位采集(对应遥测、遥调)
cnt	整型：1~n	采集个数(无上限)

2.4 zyzn-VarNode:上报/控制节点属性

参数	值	说明
ID	整型：1~n	设备 ID
reg	字符串："2"/"3"	寄存器类型
addr	整型：0~n	数据起始地址
ctype	字符串：dword/bit	用于关联对应的采集节点
dtype	字符串： bit/byte/ubyte/short/ushort/int/uint/long/ulong/float/double/bytes	变量类型
len(选填)	整型：1~n	数据长度，仅当数据类型位 bytes 时有效，用于指定字符串长度
cycle	整型：0~n	单位：秒　精度：0.1 等于 0：即指定上报类型为改变上报 大于 0：即指定上报周期
name	字符串：上报/控制名称	