APRUS Lua-Modbus3 配置说明

概述

本章主要为 APRUS Lua-Modbus3 协议的相关配置说明，该协议主要针对支持(捷豹)3 字节浮点数 modbus 协议（汉字温压液晶）（以下简称 modbus3）的设备。APRUS 适配器可通过 modbus3 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 modbus3 协议的设备进行数据采集。

目前版本支持采集结点起始地址为 0 开始采集数据，根据相关协议无法自主选择采集点位个数及点位类型；上报类型仅支持 float 上报；一次采集到的数据包中含有不同的上报值类型，可以通过上报结点的 vartype 字段进行区分。

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。

1 APRUS.lua 配置说明

demo 示例

```
package.cpath = "./?.so"
package.cpath = "./?.so"
package.path = "./?.lua"
cjson = require "cjson"
config = require "config"
require "modbus3"



function modbus3Server_load_station(svr, cfg)
	for k, v in pairs(cfg)
	do
		print("modbus3Server load Station" .. cjson.encode(v))
		modbus3Server.setStation(svr, cjson.encode(v))
	end
end

function act_control(m, json)
	for k, v in pairs(json) do
		if k ~= "Act" then
			modbus3.write(modbus3Obj, 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
        modbus3.stop(modbus3Obj)
    elseif code == 1 then
        --modbus3.run(modbus3Obj)
    end
end

function modbus3_handle(obj, name, code, data)
    mqtt.publish(mqtt3Obj, name, "r", data)
end

function modbus3_load_collectnodes(obj, nodes)
    for k, v in pairs(nodes)
    do
        modbus3.addcnode(obj, cjson.encode(v))
    end
end

function modbus3_load_varnodes(obj, nodes)
    for k, v in pairs(nodes)
    do
        modbus3.addvnode(obj, cjson.encode(v))
    end
end

function init()
    mqtt3Obj = mqtt.new()
    mqtt.config(mqtt3Obj, "", "gards.mixyun.com", "31883", "aprusdev@aprus", "mixlinker123") --V8
    user.setluaver(config.AprusX.luaver)
    user.setdevinfo(config.AprusX.devinfo)
    user.ipconfig(config.AprusX.ipmode, config.AprusX.inet_addr, config.AprusX.netmask)
end
```

```
function start()
    print("<--------------------user lua start------------------------>")
    init()
    print("<--------------------modbus3_init()------------------------>")
    modbus3Obj = modbus3.new("modbus3")
    modbus3.config(modbus3Obj, cjson.encode(config.MODBUS3.Device))
    --modbus3.debug(modbus3Obj, 0x01010001)
    modbus3_load_collectnodes(modbus3Obj, config.MODBUS3.ColNode)
    modbus3_load_varnodes(modbus3Obj, config.MODBUS3.VarNode)
    print("<--------------------modbus3.run------------------------>")
    modbus3.run(modbus3Obj)
    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 == "modbus3" then
            modbus3_handle(msg.obj, msg.name, msg.code, msg.data)
        end
    end

    print("<--------------------user lua over------------------------>")
end

start()
```

1.1 导入 modbus3 协议支持库

语句	require "modbus3"
说明	导入 modbus3 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 modbus3 对象

语句	modbus3obj = modbus3.new("modbus3")

说明	创建 modbus3 对象，返回 modbus3obj 供全局使用

1.4 配置 modbus3 对象接口参数

语句	modbus3.config(modbus3obj, cjson.encode(config.modbus3.Device))
说明	配置 modbus3 通信接口 RS485 参数

1.5 添加 modbus3 采集节点

语句	modbus3_load_collectnodes(modbus3obj,config.modbus3.ColNode)
说明	添加 config.lua 中所配置的采集节点

1.6 添加 modbus3 上报变量节点

语句	modbus3_load_varnodes(modbus3obj,config.modbus3.VarNode)
说明	添加 config.lua 中所配置的变量节点

1.7 运行 modbus3 实例

语句	modbus3.run(modbus3obj)
说明	在 mqtt 建立连接后 运行 modbus3 实例 即开始采集+运算

1.8 暂停 modbus3 实例

语句	modbus3.stop(modbus3obj)
说明	在 mqtt 连接断开时 暂停 modbus3 实例

1.9 设置 modbus3 实例的调试打印级别

语句	modbus3.debug(modbus3Obj, debugNumber)
说明	启用详细的 modbus3 调试信息。第一个参数为 modbus3.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 modbus3 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 modbus3 对象事件

1.11 msg 事件处理

语句	elseif msg.from == "modbus3" then msg _handle(msg.obj, msg.name, msg.code, msg.data)
说明	当接收到 msg 对象事件时，调用处理函数 modbus3_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 modbus3_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(暂不支持)

语句	modbus3.write(modbus3obj, cjson.encode({["name"] = k, ["val"] = v}))
说明	反向控制时调用的 modbus3 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",
    },
    MODBUS3 = {
        Device = {
            portindex = 1,
            rate = 9600,
            databit = 8,
            stopbit = 1,
            parity = "None", -- None/Odd/Even
            queryInterval = 100
        },
        ColNode = {
            { ID = 1, reg = "3", addr = 0, cnt = 20, dbName = "MOBUS_ID1A00B6" },
            -- { ID = 2, reg = "3", addr = 0, cnt = 20, dbName = "MOBUS_ID2A00B6" },
        },
        VarNode = {
            { ID = 1, reg = "3", addr = 0, vartype = "flow",             dtype = "float", cycle = 6, name = "L1_3_0_flow" },
             { ID = 1, reg = "3", addr = 0, vartype = "temperature",     dtype = "float", cycle = 6, name =
"L1_3_0_temperature" },
             { ID = 1, reg = "3", addr = 0, vartype = "pressure",        dtype = "float", cycle = 6, name =
"L1_3_0_pressure" },
             { ID = 1, reg = "3", addr = 0, vartype = "density",         dtype = "float", cycle = 6, name =
"L1_3_0_density" },
             { ID = 1, reg = "3", addr = 0, vartype = "frequency",       dtype = "float", cycle = 6, name =
"L1_3_0_frequency " },
             { ID = 1, reg = "3", addr = 0, vartype = "cumulative_flow", dtype = "float", cycle = 6, offset =
"*100;+1;~1",          name = "L1_3_0_cumulative_flow" },

            { ID = 2, reg = "3", addr = 0, vartype = "flow",             dtype = "float", cycle = 6, name = "L1_3_0_flow" },
             { ID = 2, reg = "3", addr = 0, vartype = "temperature",     dtype = "float", cycle = 6, name =
"L2_3_0_temperature" },
             { ID = 2, reg = "3", addr = 0, vartype = "pressure",        dtype = "float", cycle = 6, name =
"L2_3_0_pressure" },
             { ID = 2, reg = "3", addr = 0, vartype = "density",         dtype = "float", cycle = 6, name =
"L2_3_0_density" },
```

```
                { ID = 2, reg = "3", addr = 0, vartype = "frequency",        dtype = "float", cycle = 6, name = "L2_3_0_frequency " },
                { ID = 2, reg = "3", addr = 0, vartype = "cumulative_flow", dtype = "float", cycle = 6, name = "L2_3_0_cumulative_flow" },

            }
        },
    }
```

2.1 APRUS：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	"modbus3Dev"	与 APRUS 对接设备，可根据需求自行填写

2.2 modbus3-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

2.3 modbus3-ColNode ：采集节点属性

参数	值	说明
ID	整型：1~n	设备 ID
reg	字符串：3	寄存器类型
addr	整型：0	采集起始地址
cnt	整型：20	采集单位数
dbname	字符串： MOBUS_ID1A00B6	数据中心的数据库名字（详见数据中心）

2.4 modbus3-VarNode:上报节点属性

参数	值	说明
ID	整型：1~n	设备 ID
reg	字符串：3	寄存器类型
addr	整型：0	数据起始地址
vartype	字符串：flow /temperature/pressure/density/frequency/ cumulative_flow	上报数据类型：流量值/温度值/压力值 /密度值/频率值/累计流量
dtype	字符串：float	变量类型
cycle	整型：0~n	单位：秒 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
offset （选填）	+ - * / ~ 数字	例如：offset="*10; +5.123; ~1" 表示 乘以 10 再加 5.123 结果保留 1 位小数
name	字符串：上报名称	