APRUS Lua-PPI 配置说明

概述

本章主要为 APRUS Lua-PPI 协议的相关配置说明，该协议主要针对支持(西门子)PPI 协议的设备。APRUS 适配器可通过 PPI 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 PPI 协议的设备进行数据采集。

目前版本支持 V 寄存器的按位或者字节为单位进行数据采集；对于位数据通过 PPI 协议采集时仅支持一位一位进行采集，不支持同时采集多个位数据。

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.path = "./?.lua"
cjson = require "cjson"
config = require "config"
require "ppi"



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

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

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

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

function ppi_load_varnodes(obj, nodes)
	for k, v in pairs(nodes)
	do
		ppi.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("<--------------------ppi_init()------------------------>")
```

```
    ppiObj = ppi.new("ppi")
    ppi.config(ppiObj, cjson.encode(config.PPI.Device))
    ppi_load_collectnodes(ppiObj, config.PPI.ColNode)
    ppi_load_varnodes(ppiObj, config.PPI.VarNode)
    --ppi.debug(ppiObj, 0x01010101)
    print("<--------------------ppi.run------------------------>")
    ppi.run(ppiObj)
    mqtt.run(mqtt3Obj)

    counter = 0
    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 == "ppi" then
            ppi_handle(msg.obj, msg.name, msg.code, msg.data)
        end
    end

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

start()
```

1.1 导入 ppi 协议支持库

语句	require "ppi"
说明	导入 ppi 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 ppi 对象

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

1.4 配置 ppi 对象接口参数

语句	ppi.config(ppiobj, cjson.encode(config.PPI.Device))
说明	配置 ppi 通信接口 RS485 参数

1.5 添加 ppi 采集节点

语句	ppi_load_collectnodes(ppiobj,config.PPI.ColNode)
说明	添加 config.lua 中所配置的采集节点

1.6 添加 ppi 上报变量节点

语句	ppi_load_varnodes(ppiobj,config.PPI.VarNode)
说明	添加 config.lua 中所配置的变量节点

1.7 运行 ppi 实例

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

1.8 暂停 ppi 实例

语句	ppi.stop(ppiobj)
说明	在 mqtt 连接断开时 暂停 ppi 实例

1.9 设置 ppi 实例的调试打印级别

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

1.11 msg 事件处理

语句	elseif msg.from == "msg " 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 ppi_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(暂不支持)

语句	ppi.write(ppiobj, cjson.encode({["name"] = k, ["val"] = v}))
说明	反向控制时调用的 ppi 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",
	},
	Forward485 = {
		device = {
			rate = 9600,
			databit = 8,
			stopbit = 1,
			parity = "Even", -- None/Odd/Even
			timeout = 3000,
		},
	},
	PPI = {
		Device = {
			portindex = 1,
			rate = 9600,
			databit = 8,
			stopbit = 1,
			parity = "Even", -- None/Odd/Even
			queryInterval = 100
		},
		ColNode = {
			--采集只有 bit 和 byte 采集两种模式，默认是 byte 采集

			--bit   ctype = "bit" 采集的时候只能单采，cnt 只能==1    bitn 默认是 0
			{ ID = 2, reg = "V",   addr = 100, cnt = 1, ctype = "bit", bitn = 0, dbName = "PPI_2V100L0" },
			{ ID = 2, reg = "V",   addr = 100, cnt = 1, ctype = "bit", bitn = 1, dbName = "PPI_2V100L1" },
			{ ID = 2, reg = "V",   addr = 100, cnt = 1, ctype = "bit", bitn = 2, dbName = "PPI_2V100L2" },
			{ ID = 2, reg = "V",   addr = 100, cnt = 1, ctype = "bit", bitn = 3, dbName = "PPI_2V100L3" },
			{ ID = 2, reg = "V",   addr = 100, cnt = 1, ctype = "bit", bitn = 4, dbName = "PPI_2V100L4" },
			{ ID = 2, reg = "V",   addr = 100, cnt = 1, ctype = "bit", bitn = 5, dbName = "PPI_2V100L5" },
			{ ID = 2, reg = "V",   addr = 100, cnt = 1, ctype = "bit", bitn = 6, dbName = "PPI_2V100L6" },
			{ ID = 2, reg = "V",   addr = 100, cnt = 1, ctype = "bit", bitn = 7, dbName = "PPI_2V100L7" },

			{ ID = 2, reg = "V",   addr = 101, cnt = 1, ctype = "bit", bitn = 0, dbName = "PPI_2V101L0" },
			{ ID = 2, reg = "V",   addr = 101, cnt = 1, ctype = "bit", bitn = 1, dbName = "PPI_2V101L1" },
```

```
        { ID = 2, reg = "V",   addr = 101, cnt = 1, ctype = "bit", bitn = 2, dbName = "PPI_2V101L2" },
        { ID = 2, reg = "V",   addr = 101, cnt = 1, ctype = "bit", bitn = 3, dbName = "PPI_2V101L3" },
        { ID = 2, reg = "V",   addr = 101, cnt = 1, ctype = "bit", bitn = 4, dbName = "PPI_2V101L4" },
        { ID = 2, reg = "V",   addr = 101, cnt = 1, ctype = "bit", bitn = 5, dbName = "PPI_2V101L5" },
        { ID = 2, reg = "V",   addr = 101, cnt = 1, ctype = "bit", bitn = 6, dbName = "PPI_2V101L6" },
        { ID = 2, reg = "V",   addr = 101, cnt = 1, ctype = "bit", bitn = 7, dbName = "PPI_2V101L7" },


        { ID = 2, reg = "V",   addr = 506, cnt = 1, ctype = "bit", bitn = 0, dbName = "PPI_2V506L0" },
        { ID = 2, reg = "V",   addr = 506, cnt = 1, ctype = "bit", bitn = 1, dbName = "PPI_2V506L1" },
        { ID = 2, reg = "V",   addr = 506, cnt = 1, ctype = "bit", bitn = 2, dbName = "PPI_2V506L2" },
        { ID = 2, reg = "V",   addr = 506, cnt = 1, ctype = "bit", bitn = 3, dbName = "PPI_2V506L3" },
        { ID = 2, reg = "V",   addr = 506, cnt = 1, ctype = "bit", bitn = 4, dbName = "PPI_2V506L4" },
        { ID = 2, reg = "V",   addr = 506, cnt = 1, ctype = "bit", bitn = 5, dbName = "PPI_2V506L5" },
        { ID = 2, reg = "V",   addr = 506, cnt = 1, ctype = "bit", bitn = 6, dbName = "PPI_2V506L6" },
        { ID = 2, reg = "V",   addr = 506, cnt = 1, ctype = "bit", bitn = 7, dbName = "PPI_2V506L7" },
        --bit

        --byte
        { ID = 2, reg = "V",   addr = 102, cnt = 10, ctype = "byte", dbName = "PPI_2V102L10"},
        { ID = 2, reg = "V",   addr = 507, cnt = 2,   ctype = "byte", dbName = "PPI_2V507L1"},
        --byte

        --otherReg
        --{ ID = 2, reg = "S",   addr = 0, cnt = 30,   ctype = "byte",   dbName = "PPI_2S0L30"},
        --{ ID = 2, reg = "SM",   addr = 0, cnt = 50,   ctype = "byte",   dbName = "PPI_2SM0L50"},
    },
    VarNode = {
        --ctype = "bit", dtype = "bit"
        { ID = 2, reg = "V",   addr = 100, ctype = "bit",     dtype = "bit",   bitn = 0, cycle = 1,    name =
"L2_V_100_0" },
        { ID = 2, reg = "V",   addr = 100, ctype = "bit",     dtype = "bit",   bitn = 1, cycle = 1,    name =
"L2_V_100_1" },
        { ID = 2, reg = "V",   addr = 100, ctype = "bit",     dtype = "bit",   bitn = 2, cycle = 1,    name =
"L2_V_100_2" },
        { ID = 2, reg = "V",   addr = 100, ctype = "bit",     dtype = "bit",   bitn = 3, cycle = 1,    name =
"L2_V_100_3" },
        { ID = 2, reg = "V",   addr = 100, ctype = "bit",     dtype = "bit",   bitn = 4, cycle = 1,    name =
"L2_V_100_4" },
        { ID = 2, reg = "V",   addr = 100, ctype = "bit",     dtype = "bit",   bitn = 5, cycle = 1,    name =
"L2_V_100_5" },
        { ID = 2, reg = "V",   addr = 100, ctype = "bit",     dtype = "bit",   bitn = 6, cycle = 1,    name =
"L2_V_100_6" },
        { ID = 2, reg = "V",   addr = 100, ctype = "bit",     dtype = "bit",   bitn = 7, cycle = 1,    name =
"L2_V_100_7" },

        { ID = 2, reg = "V",   addr = 101, ctype = "bit",     dtype = "bit",   bitn = 0, cycle = 1,    name =
"L2_V_101_0" },
        { ID = 2, reg = "V",   addr = 101, ctype = "bit",     dtype = "bit",   bitn = 1, cycle = 1,    name =
"L2_V_101_1" },
        { ID = 2, reg = "V",   addr = 101, ctype = "bit",     dtype = "bit",   bitn = 2, cycle = 1,    name =
"L2_V_101_2" },
        { ID = 2, reg = "V",   addr = 101, ctype = "bit",     dtype = "bit",   bitn = 3, cycle = 1,    name =
"L2_V_101_3" },
        { ID = 2, reg = "V",   addr = 101, ctype = "bit",     dtype = "bit",   bitn = 4, cycle = 1,    name =
"L2_V_101_4" },
        { ID = 2, reg = "V",   addr = 101, ctype = "bit",     dtype = "bit",   bitn = 5, cycle = 1,    name =
"L2_V_101_5" },
        { ID = 2, reg = "V",   addr = 101, ctype = "bit",     dtype = "bit",   bitn = 6, cycle = 1,    name =
"L2_V_101_6" },
        { ID = 2, reg = "V",   addr = 101, ctype = "bit",     dtype = "bit",   bitn = 7, cycle = 1,    name =
"L2_V_101_7" },


        { ID = 2, reg = "V",   addr = 506, ctype = "bit",     dtype = "bit",   bitn = 0, cycle = 1,    name =
"L2_V_506_0" },
        { ID = 2, reg = "V",   addr = 506, ctype = "bit",     dtype = "bit",   bitn = 1, cycle = 1,    name =
"L2_V_506_1" },
        { ID = 2, reg = "V",   addr = 506, ctype = "bit",     dtype = "bit",   bitn = 2, cycle = 1,    name =
"L2_V_506_2" },
        { ID = 2, reg = "V",   addr = 506, ctype = "bit",     dtype = "bit",   bitn = 3, cycle = 1,    name =
"L2_V_506_3" },
```

```
                { ID = 2, reg = "V",   addr = 506, ctype = "bit",     dtype = "bit",   bitn = 4, cycle = 1,    name =
"L2_V_506_4" },
                { ID = 2, reg = "V",   addr = 506, ctype = "bit",     dtype = "bit",   bitn = 5, cycle = 1,    name =
"L2_V_506_5" },
                { ID = 2, reg = "V",   addr = 506, ctype = "bit",     dtype = "bit",   bitn = 6, cycle = 1,    name =
"L2_V_506_6" },
                { ID = 2, reg = "V",   addr = 506, ctype = "bit",     dtype = "bit",   bitn = 7, cycle = 1,    name =
"L2_V_506_7" },
                --ctype = "bit"


                --ctype = "byte",   dtype = "byte"
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",   dtype = "byte", cycle = 1, offset="*100; ~1",
name = "L2_V_102" },
                { ID = 2, reg = "V",   addr = 103, ctype = "byte",   dtype = "byte", cycle = 1,   name = "L2_V_103" },
                { ID = 2, reg = "V",   addr = 104, ctype = "byte",   dtype = "byte", cycle = 1,   name = "L2_V_104" },
                --ctype = "byte",   dtype = "byte"
                { ID = 2, reg = "V",   addr = 105, ctype = "byte",   dtype = "ubyte", cycle = 1,   name = "L2_V_105" },
                { ID = 2, reg = "V",   addr = 106, ctype = "byte",   dtype = "ubyte", cycle = 1,   name = "L2_V_106" },
                { ID = 2, reg = "V",   addr = 107, ctype = "byte",   dtype = "ubyte", cycle = 1,   name = "L2_V_107" },
                { ID = 2, reg = "V",   addr = 108, ctype = "byte",   dtype = "ubyte", cycle = 1,   name = "L2_V_108" },
                { ID = 2, reg = "V",   addr = 109, ctype = "byte",   dtype = "ubyte", cycle = 1,   name = "L2_V_109" },
                { ID = 2, reg = "V",   addr = 110, ctype = "byte",   dtype = "ubyte", cycle = 1,   name = "L2_V_110" },
                { ID = 2, reg = "V",   addr = 111, ctype = "byte",   dtype = "ubyte", cycle = 1,   name = "L2_V_111" },
                { ID = 2, reg = "V",   addr = 507, ctype = "byte",   dtype = "ubyte", cycle = 1,   name = "L2_V_507" },
                --ctype = "byte",   dtype = "short"
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_102_S" },
                { ID = 2, reg = "V",   addr = 103, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_103_S" },
                { ID = 2, reg = "V",   addr = 104, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_104_S" },
                { ID = 2, reg = "V",   addr = 105, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_105_S" },
                { ID = 2, reg = "V",   addr = 106, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_106_S" },
                { ID = 2, reg = "V",   addr = 107, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_107_S" },
                { ID = 2, reg = "V",   addr = 108, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_108_S" },
                { ID = 2, reg = "V",   addr = 109, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_109_S" },
                { ID = 2, reg = "V",   addr = 507, ctype = "byte",   dtype = "short", cycle = 1,   name =
"L2_V_507_S" },
                --ctype = "byte"

                --ctype = "byte", dtype = "bit"
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",     dtype = "bit",   bitn = 0, cycle = 1,    name =
"L2_V_102_0" },
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",     dtype = "bit",   bitn = 1, cycle = 1,    name =
"L2_V_102_1" },
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",     dtype = "bit",   bitn = 2, cycle = 1,    name =
"L2_V_102_2" },
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",     dtype = "bit",   bitn = 3, cycle = 1,    name =
"L2_V_102_3" },
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",     dtype = "bit",   bitn = 4, cycle = 1,    name =
"L2_V_102_4" },
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",     dtype = "bit",   bitn = 5, cycle = 1,    name =
"L2_V_102_5" },
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",     dtype = "bit",   bitn = 6, cycle = 1,    name =
"L2_V_102_6" },
                { ID = 2, reg = "V",   addr = 102, ctype = "byte",     dtype = "bit",   bitn = 7, cycle = 1,    name =
"L2_V_102_7" },

                { ID = 2, reg = "V",   addr = 507, ctype = "byte",     dtype = "bit",   bitn = 0, cycle = 1,    name =
"L2_V_507_0" },
                { ID = 2, reg = "V",   addr = 507, ctype = "byte",     dtype = "bit",   bitn = 1, cycle = 1,    name =
"L2_V_507_1" },
                { ID = 2, reg = "V",   addr = 507, ctype = "byte",     dtype = "bit",   bitn = 2, cycle = 1,    name =
"L2_V_507_2" },
                { ID = 2, reg = "V",   addr = 507, ctype = "byte",     dtype = "bit",   bitn = 3, cycle = 1,    name =
"L2_V_507_3" },
```

```
                    { ID = 2, reg = "V",   addr = 507, ctype = "byte",    dtype = "bit",   bitn = 4, cycle = 1,    name =
"L2_V_507_4" },
                    { ID = 2, reg = "V",   addr = 507, ctype = "byte",    dtype = "bit",   bitn = 5, cycle = 1,    name =
"L2_V_507_5" },
                    { ID = 2, reg = "V",   addr = 507, ctype = "byte",    dtype = "bit",   bitn = 6, cycle = 1,    name =
"L2_V_507_6" },
                    { ID = 2, reg = "V",   addr = 507, ctype = "byte",    dtype = "bit",   bitn = 7, cycle = 1,    name =
"L2_V_507_7" },
                    --ctype = "byte"
                }
            },
        }
```

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	"ppiDev"	与 APRUS 对接设备，可根据需求自行填写

2.2 PPI-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 PPI-ColNode ：采集节点属性

参数	值	说明
ID	整型：1~n	设备 ID
reg	字符串： V	寄存器类型
addr	整型：0~n	采集起始地址
cnt	整型：1~n	采集单位数

ctype(选填)	字符串：byte(默认)/bit	bit:以位为单位采集 Byte:以字节为单位采集
bitn	整型：0~7	位偏移量，在 addr 的基础上做偏移 即采集起始地址=addr*8+bitn
dbname	字符串： PPI_2V100L0/PPI_2V101L0	数据中心的数据库名字（详见数据中心）

2.4 PPI-VarNode:上报节点属性

参数	值	说明
ID	整型：1~n	设备 ID
reg	字符串： V	寄存器类型
addr	整型：0~n	数据起始地址
ctype(选填)	字符串：byte(默认)/bit	用于关连对应的采集节点
dtype	字符串： bit/byte/ubyte/short/ushort/	变量类型（若上报类型是 short 则需要采集类型为 byte，cnt=2）
bitn(选填)	整型：0~n	位偏移量，当数据类型为 bit 时有效
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
name	字符串：上报名称	
offset	+ - * / ~ 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