Aprus Lua-MitsuCNC 配置说明

概述

本章主要为 Aprus Lua-mitsucnc 协议的相关配置说明，该协议主要针对支持 mitsucnc 协议的设备。Aprus 适配器可通过 mitsucnc 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 mitsucnc 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。

1 aprus.lua 配置说明

demo 示例

```
package.cpath="/home/zdw/test/lib/x86/output/lua-cjson/lib/?.so;./?.so"
package.path="./?.lua"
cjson = require "cjson"
config = require "config"
require "mitsucnc"

function act_control(m, json)

    for k,v in pairs(json) do
        if k ~= "Act" then
            mitsucnc.write(mitsucncobj, 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
        mitsucnc.stop(mitsucncobj)
    elseif code == 1 then
        mitsucnc.run(mitsucncobj)
```

```
    end
end

function mitsucnc_handle(obj, name, code, data)
    mqtt.publish(m1, name, "r", data)
end

function mitsucnc_load_collectnodes(obj, nodes)

    for k,v in pairs(nodes)
    do
        mitsucnc.addcnode(obj,cjson.encode(v))

    end
end

function mitsucnc_load_varnodes(obj, nodes)

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

function init()
    m1 = mqtt.new()

    user.setluaver(config.Aprus.luaver)
    user.setdevinfo(config.Aprus.devinfo)
    user.ipconfig(config.Aprus.ipmode, config.Aprus.inet_addr, config.Aprus.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()
    mitsucncobj = mitsucnc.new("mitsucnc")
    mitsucnc.config(mitsucncobj, cjson.encode(config.mitsucnc.Device))
    mitsucnc_load_collectnodes(mitsucncobj,config.mitsucnc.Node)
    mitsucnc_load_varnodes(mitsucncobj,config.mitsucnc.Node)

    mqtt.run(m1)
    while true do
        local r = user.waitmsg()
```

```
            if r.from == "mqtt-sys" then
                mqttsys_handle(r.session, r.code)
            elseif r.from == "mqtt-msg" then
                mqttdata_handle(r.session, r.topic, r.payload)
            elseif r.from == "mitsucnc" then
                mitsucnc_handle(r.obj, r.name, r.code, r.data)
            end
        end
    end
start()
```

内容较多，不逐行描述，仅对关键部分(已标红)进行解释

1.1 导入 mitsucnc 协议支持库

语句	require "mitsucnc"
说明	导入 mitsucnc 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 mitsucnc 对象

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

1.4 配置 mitsucnc 对象接口参数

语句	mitsucnc.config(mitsucncobj, cjson.encode(config.mitsucnc.Device))
说明	配置 mitsucnc 通信网卡参数

1.5 添加 mitsucnc 采集节点

语句	mitsucnc_load_collectnodes(mitsucncobj,config.mitsucnc.Node)
说明	添加 config.lua 中所配置的采集节点

1.6 添加 mitsucnc 上报变量节点

语句	mitsucnc_load_varnodes(mitsucncobj,config.mitsucnc.Node)
说明	添加 config.lua 中所配置的变量节点

1.7 运行 mitsucnc 实例

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

1.8 暂停 mitsucnc 实例

语句	mitsucnc.stop(mitsucncobj)
说明	在 mqtt 连接断开时 暂停 mitsucnc 实例

1.9 等待 mitsucnc 事件

语句	local r = user.waitmsg()
说明	启动事件等待，该接口会返回全局各种事件，包括 mitsucnc 对象事件

1.10 mitsucnc 事件处理

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

1.11 mqtt 数据上报

语句	function mitsucnc_handle(obj, name, code, data) mqtt.publish(m1, 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

2 config.lua 配置说明

Demo 示例：

```
Aprus={
        ipmode="manual",                        --auto/manual/none
        inet_addr="192.168.0.234",
        netmask="255.255.255.0",
        luaver="V00.R",
        devinfo="mitsucncDev",
},

mitsucnc={

        Device={

            ip ="192.168.0.10",

            port = "683",

            type = "m80"

        },

        Node={

            {vartype=1,cycle=3,name="L1"},    --加工语句号

            {vartype=2,cycle=3,name="L2"},    --刀补 D 编号

            {vartype=3,cycle=3,name="L3"},    --刀补 H 编号

            {vartype=4,cycle=3,name="L4"},    --快速倍率

            {vartype=5,cycle=3,name="L5"},    --操作模式

            {vartype=6,cycle=3,name="L6"},    --当前刀具号

            {vartype=7,cycle=3,name="L7"},    --运行模式

            {vartype=8,cycle=3,name="L8"},    --累计加工时间

            {vartype=9,cycle=3,name="L9"},    --主轴负载

            {vartype=10,cycle=5,name="L10"}, --主轴转速

            {vartype=11,cycle=3,name="L11"}, --主轴报警

            {vartype=12,cycle=5,name="L12"}, --获取指令进给速度（FA）

            {vartype=13,cycle=3,name="L13"}, --X 进给负载

            {vartype=14,cycle=3,name="L14"}, --Y 进给负载

            {vartype=15,cycle=3,name="L15"}, --Z 进给负载

            {vartype=16,cycle=3,name="L16"}, --A 进给负载

            {vartype=17,cycle=3,name="L17"}, --加工数
```

```
            {vartype=18,cycle=3,name="L18"}, --程序名称
            {vartype=19,cycle=3,name="L19"}, --程序编号
            {vartype=20,cycle=3,name="L20"}, --刀具测量 0:未测量 非0:已测量
            {vartype=21,cycle=3,name="L21"}, --是否自动运行 0:未自动运行 非0:自动运行
            {vartype=22,cycle=3,name="L22"}, --是否开始自动运行 0:未开始自动运行 非0:开始自动运行
        {vartype=23,cycle=3,name="L23"}, --自动运行是否暂停 0:自动运行未暂停 非0:自动运行已经暂停
            {vartype=24,cycle=3,name="L24"}, --系统号码
            {vartype=25,cycle=3,name="L25"}, --NC报警
            {vartype=26,cycle=5,name="L26"}, --进给速率
            {vartype=100,var1=10, var2=12, cycle=5,name="L100",offset="/10;+10.658;~2"}, -- 主轴倍率 = 主轴转速 / 获取指令进给速度 (FA)
            {vartype=101,var1=26, var2=12, cycle=5,name="L101",offset="*10.123;+5;~1"}, --进给倍率 = 进给速率/ 获取指令进给速度 (FA)
        },
    },
```

2.1 Aprus: 接口属性

参数	值	说明
ipmode	"auto"/"manual"/"none"	ip 获取方式
inet_addr	"192.168.0.234"	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	"mitsucncDev"	与 Aprus 对接设备，可根据需求自行填写

2.2 mitsucnc-Device: 接口属性

参数	值	说明
ip	字符串	设备的 ip
port	字符串	设备的端口号
type	"m80"	设备的型号

2.3 mitsucnc-Node:节点属性

参数	值	说明
vartype	1	加工语句号
	2	刀补 D 编号
	3	刀补 H 编号
	4	快速倍率
	5	操作模式
	6	当前刀具号
	7	运行模式
	8	累计加工时间
	9	主轴负载
	10	主轴转速
	11	主轴报警
	12	获取指令进给速度（FA）
	13	X 进给负载
	14	Y 进给负载
	15	Z 进给负载
	16	A 进给负载
	17	加工数
	18	程序名称
	19	程序编号
	20	刀具测量 0:未测量 非 0:已测量
	21	是否自动运行 0:未自动运行 非 0:自动运行
	22	是否开始自动运行 0:未开始自动运行 非 0:开始自动运行
	23	自动运行是否暂停 0:自动运行未暂停 非 0:自动运行已经暂停
	24	系统号码

	25	NC 报警
	26	进给速率
	100	主轴倍率 = 主轴转速 / 获取指令进给速度 (FA)
	101	进给倍率 = 进给速率 / 获取指令进给速度 (FA)
var1	整型：1~26	当 vartype>=100 时才能使用。 本节点值= var1 节点值 / var2 节点值
var2	整型：1~26	当 vartype>=100 时才能使用。 本节点值= var1 节点值 / var2 节点值
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