APRUS Lua-MitsuMC 配置说明

第一章 概述

1.1 文档说明

文档概述 APRUS Lua-MitsuMC 协议相关配置说明，APRUS 的 Lua 包含 APRUS.lua 和 config.lua 两个文件。APRUS 通过 MitsuMC 协议与三菱 MC 系列 PLC 通信，然后配置 config.lua 即可采集需要的数据。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。

此文档目的是为了让相关人员清楚如何通过对脚本的配置实现对应协议的数据采集，详细介绍 config.lua 的相关内容，如遇到新的需求或者处理不当的地方联系相关负责人。

1.2 适用对象

本文档的目的读者是所有通过适配器对三菱 MC 系列 PLC(MitsuMC 协议)进行数据采集的人员。

1.3 术语和缩略词

序号	术语名称	其它名称	术语说明
1	APRUS	Advanced Programmable Remote Utility Server	高级可编程远程数据适配终端
2			

1.4 参考资料

[1].《HslCommunicationDemo》
[2].《昆仑通态 HMI 屏编程软件》

第二章 背景介绍

1.APRUS 在数据获取时，分为两部分，数据采集（CollrReg）和数据上报（CollpReg），为什么要分开呢，主要是由于提升采集效率而设定，数据采集过程中尽可能的把能连续采集的一次采集回来。

2.数据上报是将已经采集的数据进行必要的数据处理，然后上报到云端。

第三章 配置说明

3.1 config.lua 配置

config.Lua 由 5 大部分组成，ipmode、inet_addr、netmask、Luaver、devinfo；分别来介绍这 5 大部分的功能：

示例：

```
APRUSX={
    ipmode="Manual",
    inet_addr="192.168.1.234",
    netmask="255.255.255.0",
    Luaver="MAX.Lua.V030300.R",
    devinfo="MixsuMCDev",
},
```

说明：

序号	模块	说明	备注
1	ipmode	IP 获取方式	默认 Manual，不更改
2	Inet_addr	APRUS IP	与对接的设备保持相同网段，且同网段内 IP 不能重复
3	netmask	子网掩码	默认 255.255.255.0，一般不修改，除非相同网段内没有足够的 IP
4	Luaver	脚本版本号	每次变更脚本都需更新版本，一边区分不同版本的区别
5	devinfo	设备类型	对接的设备类型

3.1.1 Ipmode:ip 获取方式

ipmode="Manual"。

ipmode 为 APRUS 的 ip 获取方式，默认为 Manual，手动模式，人为填写 APRUS 的 ip。

3.1.2 Inet_addr:APRUS Ip

Inet_addr="192.168.1.234"。

Inet_addr 为 APRUS 的 ip，手动填写 ip，需要同对接设备保持在相同的网段内，否则无法与对接设备通过网口通信。

3.1.3 netmask:子网掩码

netmask="255.255.255.0",

netmask 为子网掩码，默认为 255.255.255.0，与客户保持一直，一般情况下均为 255.255.255.0 不变。

3.1.4 devinfo:设备类型

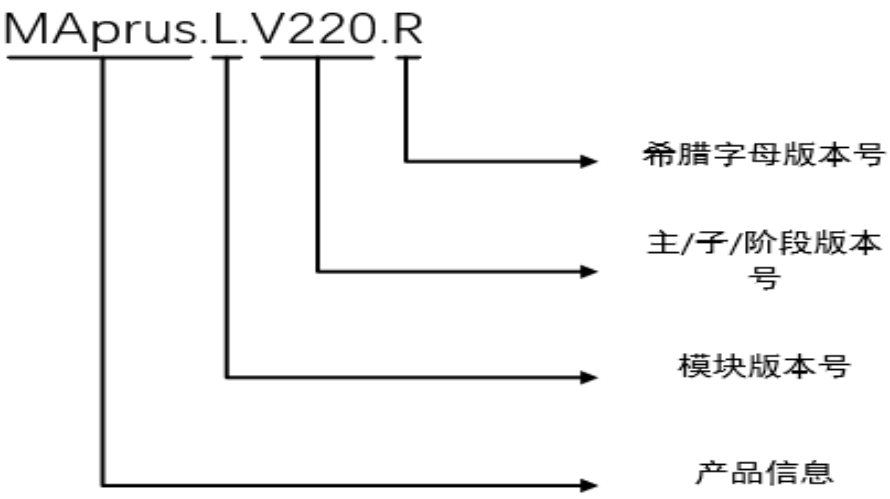
devinfo= "MixsuMCDev"。

devinfo 设置设备类型，由于对接的设备类型不同，因此为了方便使用，知道对接的是哪类设备，我们设定了设备类型。

3.1.5 Luaver:脚本版本

Luaver = "MAPRUS.L.V030005.R"。

Luaver 设置当前脚本的版本号，每次更新都需要更改并向上增加版本号；版本号在增加过程中，只需要修改主/子/阶段版本号即可，其他无需修改，主/子/阶段版本号不能为 0，一般情况下如果只更新 config.Lua 的配置无需更新主/子版本号，只要修改阶段版本号就行，版本号一定是要向上叠加，这样才能保证之前的版本有备份，才能自动升级。



3.1.6 MitsuMC:Device: MitsuMC 设备信息

```
Device={
    Type="3E-Binary"
    ip="192.168.1.200",
    port=502
},
```

Mitsufx 设备信息部分:

序号	模块	说明
1	Type="3E-Binary"	MC 协议子类 目前仅支持 3E-Binary
2	ip="192.168.1.200"	PLC IP 地址
3	Port=502	PLC 端口号

3.1.7 Node 节点信息

```
node={
    collect={
            {reg="X", addr=0, cnt=50},
            {reg="D", addr=0, cnt=50},
        },
    variable={
        {reg="X",addr=0,dtype="bit",dBit=0,pMode={1,8},dStyle={"L1_X_0"}},
```

```
        {reg="D",addr=0,dtype="bit",dBit=5,pMode={1,8},dStyle={"L1_D_0_5"}},
    },
```

1 collect:采集配置

collect 设置采集配置，采集配置主要有几个参数构成

采集配置	参数	说明	备注
CollrReg	reg	寄存器类型	X/Y/M/L/F/V/B D
	addr	采集地址起始	以双字节为单位的起始地址 X/Y/M/L/F/V/B 寄存器: 以位为单位的起始地址
	cnt	采集长度	D 寄存器: 一个采集长度为 2 字节 X/Y/M/L/F/V/B 寄存器: 一个采集长度为 1 位

2 variable:上报配置

variable 设置上报配置，上报配置时需要把采集到的数据处理成最终实际使用的数据，包括数据处理逻辑，上报逻辑，数据分类等等；由于采集时是按照双字（short）或位(bit)采集，所以在数据处理的时候要需要注意。

注意：上报配置设置的地址，一定是在采集配置里面有的，否则数据不准确。

采集配置	参数	说明		备注
CollpReg	reg	寄存器	寄存器类型	X/Y/M/L/F/V/B 寄存器: 仅支持 bit 数据类型 D 寄存器: 支持 bit、byte、short、int float 数据类型
	addr	变量地址	addr=0	需要上报的数据地址
	dtype	变量数据类型	dtype="short"	bit
				byte
				short/ushort（Word 类型即为 short/ushort）

				Int/uint（Double Word 类型即为 int/uint）
				float
	pMode	采集上报方式	pMode={1,5}	详见附件一
	dOffset	数据偏移	dOffset={{"+",10},{"*",5}}	数据偏移在关于 bit 处理的时候是不需要的，详见附件二
	dBit	位处理	dBit=5	位标识，详见附件三
	dExt	附件报文处理	dExt={0,{0,0},{0,0}}	(选填)改变上报条件限制，当 pMode 为{2,0}时有效，{">",100} 表示 当原始数据发生改变 并且改变值大于 100 时上报，{"<", 100} 表示 当原始数据发生改变 并且改变值小于 100 时上报，{"=", 100} 表示 当原始数据发改变 并且改变值等于 100 时上报
	dStyle	KEY	dStyle={"L1_3_0_0"}	自行设置
	len	字符串长度	len=5	当 dtype 为 bytes 时有效，表示字符串长度

附件一

模块		举例	参数 1	参数 2
			上报模式	上报周期
pMode	上报类型	pMode={1,5}	1：周期上报	自行设置上报周期
			2：改变上报	0

附件二

模块	参数 1	参数 2	备注
	偏移参数 1	偏移参数 2	

dOffset	dOffset={{"+",10},{"*",5}}	"+"	N	"+"	N	
		"-"	N	"-"	N	
		"*"	N	"*"	N	
		"/"	N	"/"	N	
		"."	N	"."	N	保留小数位数

注意：

1.如果无偏移计算，则无须填写此字段，或者填写为 dOffset={}。

2.如果只有单层偏移，那么只需要设置参数 1 即可，假如一个参数需要*10，即 dOffset={{"*",10}}。

3.如果有双层偏移，那么需要设置参数 1，参数 2，加入一个参数需要*10，然后在+10，即 dOffset={{"*",10},{"+",10}}。

4.如果上报数据包含小数且需要保留指定的小数位数时，即可对参数 1 进行设置，假如需要保留 3 位小数，即 dOffset={{".",3}}。

附件三

标识	说明
dBit	当 dtype 为 bit 时有效，范围 0~15 标识第几位
	当 dtype 为 byte/ubyte 时有效 范围 0~1 标识 低/高字节

3.2 apurs.lua 配置

说明：function start() 为主流程入口函数

3.2.1 user 全局通用/配置类

1)本机网络配置方法

```
user.ipconfig(config.APRUSX.ipmode,config.APRUSX.inet_addr,config.APRUSX.netmask)
```

2)消息捕获方法

```
msg = user.waitmsg()
```

msg.from：消息来源

mqtt-sys 来自 mqtt 系统消息

mqtt-msg 来自 mqtt 数据消息

mitsumc 来自 mitsumc 管理器的消息

3)以下为 msg.from 为 modbus opcua mitsufx mitsumc dlt645 等协议时通用

msg.session: 消息会话对象

msg.code：消息号

msg.style_L：变量名称

msg.val_L：变量值

msg.style_E：事件变量名称

msg.val_E：事件变量值

msg.z：条件改变标志位

4)以下为 msg.from 为 mqtt 时使用

msg.topic：mqtt 话题

msg.payload：mqtt 消息数据

5)以下为 msg.from 为 mqtt-sys 时使用

msg.code：mqtt 事件

0： mqtt 断开

1： mqtt 连接

6)设置 Lua 版本信息

user.setLuaver(Luaver)

7)设置设备信息

user.setdevinfo(devinfo)

3.2.2 mqtt 对象方法

(1)创建 mqtt 对象实例

mqttobj = mqtt.new()

(2)配置 mqtt 连接信息 (选填)

mqtt.config(mqttobj, "mqtt_1", "192.168.1.159", "1883")

param 1 mqtt 对象实例

param 2 实例 id

param 3 mqtt 服务器 ip

param 4 mqtt 服务器端口

注：此配置选填,如果不填,则通过 gards 服务器自动指定

(3)话题订阅

mqtt.subscribe(mqttobj, "p2p")

(4)发布消息

mqtt. publish (mqttobj, reserve, topic, payload)

reserve: 预留参数,必须有,可填 nil

(5)mqtt 运行

mqtt.run(mqttobj)

备注：支持多实例，最大可创建 3 个 mqtt 对象

3.2.3 MitsuMc 对象方法

(1)创建 mitsumc 对象实例

mitsumcobj = mitsumc.new()

(2)Mitsumc 对象配置

mitsumc.config(mitsumcobj, ip, port, type)

param 1 mitsumc 对象实例

param 2 设备 ip

param 3 设备端口

param 4 协议子类型

(3)Mitsumc 添加采集节点

mitsumc.add_collectnode(mitsumcobj, reg, addr, cnt)

param 1 mitsumc 对象实例

param 2 寄存器名称

param 3 寄存器地址

param 4 采集长度

(4)Mitsumc 添加变量节点

mitsumc.add_collectnode (mitsumcobj, reg, addr, dtype, dBit, len, pMode, dStyle, dOffset, dExt)

param 1 mitsumc 对象实例

param 2 寄存器名称

param 3 寄存器地址

param 4 变量类型

param 5 位偏移 (只有当数据类型为 bit 时有效 但此字段必须有)

param 6 变量长度 (只有当数据类型为 bytes 时有效 但此字段必须有)

param 7 采集上报模式

param 8 采集上报名称

param 9 数据补偿计算

param 10 改变上报条件限制

(5)Mitsumc 向被指定节点写入数据

mitsufx.write(mitsumcobj, style, val)

param 1 mitsumc 对象实例

param 2 变量名称

param 3 变量值

(6)Mitsumc 实例启动

mitsumc.run(mitsumcobj)

(7)Mitsumc 实例停止

mitsumc.stop(mitsumcobj)

备注 ： mitsumc 可支持多实例模式，即一对多(PLC)采集