**Ovation串口通讯说明书（U3-1021，U3-1050）（修订版）**

**New Ref: CON\_015,CON\_044**

ZW Rev02.1 (Mar, 2018)

目录

[**一, 硬件部分** 1](#_Toc508372517)

[**二，软件部分** 1](#_Toc508372518)

[**三，应用详解** 4](#_Toc508372519)

[**四，其他问题** 8](#_Toc508372520)

[**五，配置文件范例** 14](#_Toc508372521)

# **一, 硬件部分**

1，两线485时

A15与B15短接、A16与B16短接、

信号线： B15---- 接正端 B16---- 接负端

电缆屏蔽线如果在Ovation侧接地，接在C17(GND)，如果在远端接地，接在C16(SH)。由于485总线是差分电压，通常不需要接地（GND）。

2，四线485/422时

A14与A15、B14与B15短接

信号线： A15---- 接接收正端RX+ A16---- 接接收负端RX-

B15---- 接发送正端TX+ B16---- 接发送负端TX-

短接B14（RES）和B15(TX+)，启用发送端终端电阻。

短接A14（RES）和A15(RX+)，启用接收端终端电阻。通常不需要终端电阻，除非设备在网络终点。

3，注意，目前的LC卡做从站时，必须使用4线制接线方式。即使你用2线制通讯做成功了，也无法保证稳定运行。如果对方的主站只支持2线制，可以用使用485-422转换器，将2线制转换为4线制，比如MOXA的TCC120或TCC120I或者研华也有相应的转换器。

4，当波特率为9600时，将B7和B8短接，当波特率为19200时不需要短接。这个通讯频率其实是指通讯卡在写程序时的频率，即笔记本电脑和通讯卡之间的通讯频率。而通讯卡和外接数据采集模块之间的数据通讯频率则是在配置文件中来设置。其速度并不限制在以上的两个频率。可以是1200，2400，4800，9600，19200等等。所以说B7,B8短接这个功能并没有什么用。

5，使用RS232接口通讯时（G01的P卡），要求多股带屏蔽电缆（通常我们不作这种，RS232一般的通讯距离只有10米~15米）；使用RS485接口通讯时（G02的P卡），电缆方面要求是多股双绞线，推荐使用紫色DP总线电缆，单个信号的两根线绞合，不同信号的线不绞合。屏蔽线都是系统侧（DCS）接地，在C17端子。外部接地时，DCS不必再接。

6，所需要的设备

带串口的电脑一台或USB转232串口转换器。如需要调试，应该再备一个232转485的转换器。

# **二，软件部分**

1、所需软件(2018年1月)

使用LC Loader。目前2.0的版本可以在Win7 32位上使用，不可以在64位使用。64位有一个测试版本，但无法用在Win 10。其他办法，可以考虑把软件安装在虚拟机，再将串口映射到虚拟机内。或者使用Total Command之类的软件连接虚拟盘。

2、首先生成配置文件（最终该文件被存到LC通讯卡中，其实编写配置文件很简单）

之前有使用Access的一个工具程序生成配置文件，不过基本上没人使用，自行手写一个也是非常方便的。具体可以参考第五部分的配置文件范例。

注意：LC卡寄存器地址为0-2047，每个寄存器均为16bit

1. 浮点数时（一般为32bit），为F0000、F0002、F0004……
2. 整数时（16bit或32bit），为I0000、I0001、I0002、I0003……
3. 开关量时，为D0000、D0001、D0002、D0003

|  |
| --- |
| LCAddr |

但无论是F0000还是D0000，其后四位数都不能重复，如果在一个LC卡中，既有AI量，又有DI量，假设AI量在前面DI量在后面，则其地址可以是：

I0000—I0099，而接下来的DI量的地址需是：D0100—D0199，依此类推。

这里需要强调一个一直以来容易被搞错的问题，Modbus的协议地址都应该是从0开始的（参考标准协议文档PI\_MBUS\_300.pdf中”Data Addresses in Modbus Messages”）。 如果说COIL 1，那么它的地址就是线圈地址0；如果说HR40011，那么它的地址就是保持寄存器的10。也就是说通常双方约定通讯地址时，要告诉对方使用哪种功能码(RCS/RHR/FMC/PMR等)，然后配合相应的点地址；如果一方给出的地址列表是40001~40100，那么实际就是告诉你HR的地址0~99；如果给出的地址是30010~30110，那么实际就是告诉你IR的地址9~109。Ovation的LC卡配置使用功能码结合0开始的地址表示方式。

Modbus地址表参考如下：

|  |  |  |  |
| --- | --- | --- | --- |
| Modbus地址类型 | 传统地址 | 协议地址 | Modbus功能码 |
| Coils线圈 | C0001 to C9999  C00001 to C65536 | 0 to 65535 | RCS, FSC, FMC |
| Inputs输入状态 | I10001 to I19999  I100001 to I165536 | 0 to 65535 | RIS |
| Input Registers输入寄存器 | IR30001 to IR39999  IR300001 to IR365536 | 0 to 65535 | RIR |
| Holding Registers保持寄存器 | HR40001 to HR49999  HR400001 to HR465536 | 0 to 65535 | RHR, PSR, PMR |

注意：MODBUS协议地址是16bit，所以地址范围是0-65535，而单个MODBUS的Response报文所包含的字节数以8bit表示，所以一个Response报文最多包含255个字节(8bit)。

Ovation LC卡支持的Modbus功能码

|  |  |  |
| --- | --- | --- |
| Modbus功能码 | LC卡主站 | LC卡从站 |
| Preset Single Register  (PSR 06) | 支持 | 支持 |
| Preset Multiple Registers  (PMR 16) | 支持 | 支持 |
| Force Single Coil  (FSC 05) | 支持 | 支持 |
| Force Multiple Coils  (FMC 15) | 支持 | 支持 |
| Read Coil Status  (RCS 01) | 支持 | 支持 |
| Read Holding Registers  (RHR 03) | 支持 | 支持 |
| Read Input Registers  (RIR 04) | 支持 | 支持 |
| Read Input Status  (RIS 02) | 支持 | 支持 |
| Read Exception Status  (RES 07) | 支持 | 不支持 |

将配置文件导入到LC卡中的步骤

最早以前有使用DOS程序的方法，这种方法已经淘汰，不再阐述。目前都应该使用LC Loader软件，具体步骤见本文第四部分。

在通讯正常时，1/2灯交替闪烁。

8灯亮时表示冗余LC卡时当前卡工作在主卡状态。

|  |
| --- |
| LEDIntep |

**在CB组态中的要求：**

1. CB组态的范例见实际组态图。
2. 当浮点数时(FRMT=1)，寄存器地址里填写0、32、64、96；当整数时(FRMT=1)，寄存器地址号分别填写0、16、32、48。
3. LC卡口地址通过 POINT INFORMATION 来确认，打开点信息查询，输入LC卡的点名（由I/O BUILDER 里查询），检查其硬件地址，如果其地址是00000003DH，则在SLCAIN的口地址中填写48，如果是00000030DH，则在SLCAIN的口地址中填写768，00000003DH表示地址是16进制的30H，而00000030DH表示是16进制的300H。而SLCAIN中的LC口地址是十进制的。

数字量时，不是用SLCAIN算法块，而是用数字类型的输入块。

|  |
| --- |
| HD |
| Card Hardware Address |

注意：请再三确认在SLC算法块中卡件地址(PHW/SHW)，卡件类型(QLC/RLC)，PCRL卡(CARD)的序号正确无误后再做Drop Loader，从OCR400开始，PCRL卡被IOIC取代，PCRL序号都应该是1。否则可能产生不可预期的后果!

1. 当数据为数字量时，用不同的算法块。
2. 数据类型通常选REAL(32bit float)。
3. 通讯选RLC。
4. 通讯来的数据必需为其事先建立对应的点（用POINT BUILDER）。
5. 通讯来的数可以进行变换（在通讯过程中或在点中）。
6. 一般AI量需要在PB中建立LA类型的内部点，而DI点需要在PB中建立LC型的内部数字量点。

# **三，应用详解**

1. LC卡做主站
2. 程序应该使用3.0版本的modbus.exe，这个版本支持G01/G02卡。
3. 主站支持的模拟量数据类型参考下表

|  |  |  |
| --- | --- | --- |
| 配置文件data\_type | SLC算法的配置 | 说明 |
| int16/short (I型占1个地址) | INTEGER |  |
| uint16/word (F型占2个地址) | REAL |  |
| int32/long (F型占2个地址) | REAL |  |
| uint32/ulong (F型占2个地址) | REAL |  |
| bcd4/ bcd8 (F型占2个地址) | REAL | 0x1234表示十进制1234 |
| bcd8 (F型占2个地址) | REAL | 0x12345678表示十进制12345678 |
| int32rev/uint32rev (F型占2个地址) | REAL |  |
| float/floatrev (F型占2个地址) | REAL |  |
| int32\_bs | REAL | 字节交换的int32 |
| int32\_ws\_bs/int32\_enron | REAL | 字节交换的int32 |
| uint32\_rev | REAL |  |
| uint32\_bs | REAL | 字节交换的uint32 |
| uint32\_ws\_b2/uint32\_enron | REAL | 字节交换的uint32 |
| float\_bs | REAL | 字节交换的float |
| float\_ws\_bs/float\_enron | REAL | 字节交换的float |

1. 主站全局配置参数的说明

*MessageTimeout(timeout)*

主站等待从站响应的时间，单位毫秒，默认值为5000(真的吗?)，最大值65535。比如设置3000ms，那么在发出一个tx保文后，如果从站立即有应答，LC卡会立即发下一个tx报文，否则会等待3000ms再发出。

*InterMsgDelay(inter\_msg\_delay)*

单位毫秒，默认值为0，最大值65535。比如设置3000ms，在发出一个tx报文后，不管从站有没有响应，LC卡都会等待3000ms后再发出下一个tx。在多从站网络中，通常不要将这个参数设得太大，以免单个从站故障过于影响整个网络的通讯。

*link\_stat\_reg(linkstatreg, qlcstatusreg)*

默认为不使用。可以设置为0~2044。

用于设置故障信息接收寄存器，如设置为2000，那么2000~2002这3个寄存器用于接收故障信息。使用SLCSTATUS算法读取状态值。

*status\_hold\_time(loop\_time)*

SLCTSTAUS算法中状态值保持的时间。默认为1000ms。范围为0~2147483647。

*watchdog\_time(watchdogtime, watch\_dog\_time)*

默认为5000ms，可以设置范围为0~2147483647。

*control\_reg(controlreg)*

用于冗余。没有默认值，可以设置为0~2047。

*executedoncereg(executed\_once\_reg)*

用于冗余。没有默认值，可以设置为0~2047。

*backupmodeaction(backup\_mode\_action)*

用于冗余。可以设置为read\_only, mute或never，默认值为never。

*diagreg(diag\_reg)*

诊断计数器。没有默认值，可以设置为0~2047。

*list*

可以设置为on或者off，默认为off。

*daniel*

可以设置为on或者off，默认为off。用于设置Daniel浮点数。

*first\_dk\_addr*

默认为7000，可以设置范围0~65534。与daniel配合使用。

1. 主站GROUP参数的说明

*operation*

periodic/gated/triggered 三种类型的通讯组periodic为周期性根据interval时间发送报文；gated为根据gatereg寄存器发送报文，gatereg相当于使能寄存器，为1时才发送报文；triggered为根据triggerreg寄存器发送报文，triggerreg在0变1的情况下，发送一次报文（逻辑中如果用keyboard来触发，注意keyboard如果在100ms这样快速任务区时，要将其Length设长一些，否则按几次才会出一次）。

*interval*

对于设置为periodic的组，可以单独设置每个组发送报文的时间间隔，默认值为0，最大值3600000。比如两个组，一个为30000，一个为60000，那么一个每隔30秒发一次tx，另一个每隔60秒发一次tx。

*groupstat*

用于设置监视group通讯状态的寄存器，可以用SLCDIN读取。1表示通讯正常，0表示通讯中断。不过这个功能似乎有些小问题，0变1反应很快，1变0需要很长时间。

*slave*

与该组报文通讯的modbus从站地址

*function*

与该组报文通讯的modbus功能码

1. 关于软重启还是硬重启

一般来说使用CTRL+SHIFT+DEL可以软重启LC卡，有时候不成功就要拔插LC卡硬重启。

1. 关于转换系数

如果我们需要对点的数值作转换，可以有3种方法，罗列如下：

1. 在使用SLCAIN读取后，使用Ovation的GainBias之类的算法做转换，优点是随时可以调整转换系数，缺点是每一个通讯点都要再额外多建至少两个点，浪费系统点资源；
2. 在LC卡通讯配置文件中每一个点可以设置单独的conv\_type，通常不建议，特别是对于通讯点多的情况，会造成配置文件过大，LC卡启动时间太长。另外如果是int类型，在卡里面转换时也会截断小数部分。由于LC卡内置的CPU相当于12MHz的Intel 80196，没有浮点协处理器，在LC卡内做浮点数的数值转换会造成很大的性能负担，因此不推荐这种方式。
3. 像普通的Ovation I/O点那样设置点本身的转换系数，这是推荐的，可以节约系统资源。特别是在通讯点数量非常大的时候。
4. 中断通讯程序的运行

可以按CTRL+C或者ESC，但是推荐ESC。CTRL+C中断用几次后会造成LC软重启起不来，停在COM那一步，ESC中断就不会。

1. 留白。
2. LC卡主站冗余

主站冗余主要看LC是仅读取数据还是同时需要发送指令(有PSR/PMR/FSC/FMC等报文)。

如果是仅读取数据，通常只需要按照普通单卡配置也可以正常使用了。

如果还有发送指令，就至少要用controlreg寄存器。在controlreg为1时，LC卡发送报文；为0时，LC卡不发送保文。

controlreg是全局配置参数，使用SLCDOUT写入。比如设置controlreg为1000，在做冗余时使用SLCSTATUS读取主站通讯状态，通讯正常的卡设置为主卡并给controlreg置1，注意主从卡一定要互斥，主卡controlreg为1时，从卡必须为0。主要的原因是SLC算法虽然里面可以填两个冗余LC卡的地址，但是控制器实际上通过该算法，同时只能读写一个LC卡的值。如果两块卡同时发送指令报文，那么有一个卡发出的指令数值实际上都是0。

主站冗余主要有以下4个配置需要考虑：

1. controlreg

冗余控制寄存器

1. executedoncereg

冗余反馈寄存器

1. backupmodeaction

冗余时，备卡工作模式

1. watchdog\_time

看门狗时间

1. LC卡做从站(手册参考CON\_044)
2. 程序应该使用3.0版本的mbs.exe，这个版本支持G01/G02卡。
3. 从站支持的模拟量数据类型参考下表
4. 从站全局配置参数的说明

*link\_status\_reg*

默认为-1,表示不使用。可以设置为0~2045。

用于设置故障信息接收寄存器，如设置为2000，那么2000~2002这3个寄存器用于接收故障信息。

*link\_ok\_reg*

默认为-1,表示不使用。可以设置为0~2047。

用于表示链路通讯是否正常。0表示链路故障；1表示链路正常。

*hold\_time(loop\_time)*

上述故障信息的保持时间，如设置为3000，那么在一条故障产生后会保持3秒，再更新。

*slave\_address*

本LC卡的Modbus从站地址，范围是1~247。

*timeout*

link\_status\_reg清零的等待时间，如设置为5000，那么过5秒，又没有新的通讯故障，则将link\_status\_reg设置为0。注意，使用SLCSTAT算法必须要设置link\_status\_reg和hold\_time才会有效；同时SLCSTAT算法所在任务区的执行时间必须两倍快于hold\_time。

*rts\_on\_tx*

可以设置为on或off，默认为off。这里我们建议设成on，特别是如果遇到RLC卡收到主站请求报文且回送了响应报文而主站却接收不到时，很可能是这个原因（当然接线不正确也能造成这个问题）。对于单主站多从站的系统，应该设置rts\_on\_rtx，即启用流量控制。

1. 点列表参数的说明

*modbus\_type(m\_type)*

modbus寄存器类型，hr/holding\_reg表示保持寄存器，c/coil表示线圈，ip/input表示输入状态（只读），ir表示输入寄存器（只读）。

*out/in/point*

表示LC卡寄存器类型

*modbus\_addr(m\_addr)*

点的modbus地址

*modbus\_data(m\_data)*

点的数据类型。需要注意的是，从站只支持int16/float/floatrev这三种数据类型，填写错误的数据类型会缺省设置为INT16。

*timeout\_action(tmo\_action, action)*

可以设置为quality,digital,drop\_fault(fault)。默认为none。quality只适用于开关量点，即当该开关量在tmo时间超过后没有通讯时，质量变坏点。digital用于设置一个寄存器tmo\_reg，当该点在tmo时间超过后没有通讯时，将tmo\_reg对应的寄存器设置为1。drop\_fault在通讯超时后发出控制器报警。

*tmo\_reg(reg)*

设置一个寄存器，如果点发生timeout时，就将该寄存器设置为1。范围为0~2047。

*tmo\_time(tmo)*

点的timeout时间，范围为0~4294967295。

1. 关于软重启还是硬重启

一般来说使用CTRL+SHIFT+DEL可以软重启LC卡，有时候不成功就要拔插LC卡硬重启。LC卡从站当连接主站且主站发送通讯报文时，往往会无法启动到通讯开始，而此时需要拔插LC卡重启。建议在重启从站LC卡时，先断开一根线，可以保证LC卡一次启动成功。

1. 留白。
2. LC卡从站冗余

# **四，其他问题**

1. 配置文件的最后多输入一行空白。AUTOEXEC.BAT的第一行要写SET NO\_87=project，这里project也可以是其它任何文本，如果不写，可能造成RLC启动后无法成功运行程序。
2. 使用RLCFLASH的0C版本刷卡，大小为16097字节，时间戳为2011年8月15日。另外建议RLCFLASH在C盘使用。这个版本可以支持新旧卡。
3. 如果LC卡刷成砖头，可以使用短接C7-C8的方法从PC机引导。短接后重启LC卡到出现C:提示符需要2-5分钟。输入C:\DOS\FORMAT A: /S格式化LC卡。如果是G02的卡，先CD DOS切换目录，然后输入LOADFIX FORMAT A: /S格式化LC卡。格式化完成后，要把RLCFLASH.EXE一同拷贝到A盘并刷一遍卡。G01-28F400CVT
4. 在进行通讯前，要同串口通讯的厂家相互协调，定好协议、数据长度、数据类型、是否带小数点及几位小数等。
5. 尤其重要的是，传输的数据清单一定要案准确，并根据此清单进行配置文件的生成。
6. 在正式进行通讯之前，先用Modbus软件进行一次测试。
7. 仿真运行后，将对方地址设定为约定的值，数据类型代码根据前面的规定（AI：RHR，DI：RIS）设定数据类型，再定好采样个数。按F1即可运行并读出对方的数据。
8. 尽可能要求通讯厂家提供的通讯点是成组连续的MODBUS地址，因为MODBUS的请求报文(Query)是以起始地址+连续点数的形式发出的。例如：需要通讯的MODBUS地址为0-100和200-300时，分两个报文通讯，大小均为101个字(16bit)；如果MODBUS地址以0,5,10,…,95,100和200,220,240,…,300时，分两个报文通讯，大小仍旧均为101个字(16bit)。连续的MODBUS地址有助于提高通讯效率。
9. 配置文件典型示范
10. 传输整型数（INT16）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TINDX | PNAME | DESC | HW\_IN\_OUT | SLAVE\_ADDR | RLCADDR | MODADDR | MODBCMD | MDATYPE |
| 75 |  | #1机#1空冷变测量Ia | AI | 1 | I0200 | 0 | RHR | INT16 |
| 76 |  | #1机#1空冷变有功P | AI | 1 | I0201 | 1 | RHR | INT16 |
| 77 |  | #1机#2空冷变测量Ia | AI | 1 | I0202 | 2 | RHR | INT16 |
| 78 |  | #1机#2空冷变有功P | AI | 1 | I0203 | 3 | RHR | INT16 |
| 79 |  | #1机#1脱硫变测量Ia | AI | 1 | I0204 | 4 | RHR | INT16 |

CB中用SLCAIN算法块，其数据类型用INT。

1. 无符号整型数（UINT16）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TINDX | PNAME | DESC | HW\_IN\_OUT | SLAVE\_ADDR | RLCADDR | MODADDR | MODBCMD | MDATYPE |
| 1 |  | #1机#1空冷变PE | AI | 1 | F0000 | 2048 | RHR | UINT16 |
| 2 |  |  | AI | 1 | F0002 | 2049 | RHR | UINT16 |
| 3 |  | #1机#2空冷变PE | AI | 1 | F0004 | 2050 | RHR | UINT16 |
| 4 |  |  | AI | 1 | F0006 | 2051 | RHR | UINT16 |

CB中用SLCAIN算法块，其数据类型用REAL。

1. 数字量

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TINDX | PNAME | DESC | HW\_IN\_OUT | SLAVE\_ADDR | RLCADDR | MODADDR | MODBCMD |
| 149 |  | #1机#1空冷变外接开入1（超温告警） | DI | 1 | D0500 | 0 | RIS |
| 150 |  | #1机#1空冷变外接开入2（超温跳闸） | DI | 1 | D0501 | 1 | RIS |
| 151 |  | #1机#1空冷变外接开入3（弹簧未储能） | DI | 1 | D0502 | 2 | RIS |
| 152 |  | #1机#1空冷变事故总信号 | DI | 1 | D0503 | 3 | RIS |

CB中用SLCDIN算法块

**LC Loader For Windows**

目前，已经有了Windows版本的LC配置软件，完全可以取代DOS版本，不再需要软驱，方便实用。如果是Windows XP/2003，安装LC Loader 1.0就可以了，从Windows 7版本开始，需要安装LC Loader 2.0.

1，安装完LC Loader后可以在”我的电脑”中看到LCNameSpace，需要将导入LC卡的文件先Copy/Paste到这个LCNameSpace的虚拟驱动器中去。

|  |
| --- |
| LCNameSpace |

2，复制需要的文件至该虚拟驱动器中

|  |
| --- |
| DiskSpace |

3，使用串口电缆连接PC的COM1口和LC卡的RS-485口后，开启”Terminal”程序。(注意：Terminal启动后就不能再向LCNameSpace中复制文件了)。

|  |
| --- |
| Terminal |

4，基本上使用默认的设置就可以了，如果用9600波特率连接LC卡的话，注意要将LC卡的B7,B8短接。确定后就连接到了LC卡件。使用CTL+SHIFT+DEL重启LC卡，在启动完成后，如果是从未使用过的LC需要设置时间；已经装载过程序的LC在重启完成后，按CTRL+C中断退出到A:状态下。此时A:为LC卡件，C:为PC机的虚拟驱动器。使用copy将需要的程序和配置文件复制到A:下。

|  |
| --- |
| rebootlc |
| 新的LC需要设置时间 |

|  |
| --- |
| ctrlc |
| CTRL+C中断LC卡件中程序运行，退回到A盘 |

|  |
| --- |
| copyfiles |
| 复制相应程序和配置文件至LC卡，其中A盘为LC卡，C盘为LCNamespace中对应的驱动器 |

5，复制完成后，使用RLCFLASH.EXE命令，更新LC的Flash Memory后，LC卡就装载完成了。更新完成后，CTRL+SHIFT+DEL重启LC卡，完成后，LC卡配置完成。如果遇到无法用CTRL+SHIFT+DEL重启，将卡件插拔一下，同样可以重启LC卡。对于以后再次修改配置文件的刷卡，建议在C盘运行刷卡程序，即用C:\RLCFLASH。如果一直用A盘的RLCFLASH，容易把LC卡刷成砖头。

|  |
| --- |
| flash |
| 更新LC卡的Flash Memory,重启后完成配置 |

6，LC卡的调试。（如果连接后无法进入调试模式，尝试重启LC卡）

* 1. 分析模式，连接到LC卡后，按’A’键进入分析模式（再按一下退出分析模式），察看Modbus报文。

|  |
| --- |
| anlysis |
| 分析模式，请求(Query)报文 |

|  |
| --- |
| a |
| 收到(Received)回应(Response)报文，显然这个回应报文是错误的(☺) |

* 1. 寄存器模式，连接到LC卡后，按’T’键进入寄存器查询模式。可以察看Ovation的0~2047号寄存器的值。

|  |
| --- |
| tmode |
| 寄存器察看模式 |

**Modbus通讯协议**

Modbus通讯协议的标准文档可以在[www.modbus.org](http://www.modbus.org)下载

# **五，配置文件范例**

1. 主站配置范例1

|  |
| --- |
| \* RLC Link Test, Modbus Master  platform RLC  baud 19200  data\_bits 8  parity Even  stop\_bits 1  duplex full  flow\_ctl rts\_on\_tx  link\_stat\_reg 2000  status\_hold\_time 2000  MessageTimeout 30000  InterMsgDelay 0  syslog 3  watchdog\_time 5000  group "AI-1"  operation periodic  interval 1000  slave 1 func RHR  point I0100 address 0000 Data\_type int16  point I0101 address 0001 Data\_type short \*same as int16  point F0110 address 0010 Data\_type uint16  point F0112 address 0012 Data\_type word \*same as uint16  point F0120 address 0020 Data\_type int32  point F0122 address 0022 Data\_type long \*same as int32  point F0130 address 0030 Data\_type uint32  point F0132 address 0032 Data\_type ulong \*same as uint32  point F0140 address 0040 Data\_type int32rev  point F0142 address 0042 Data\_type uint32rev  point F0150 address 0050 Data\_type bcd4 \*0x1234 -> 1234  point F0152 address 0052 Data\_type bcd8 \*0x12345678 ->12345678  point I0156 address 0056 Data\_type bcd4 \*0x1234 -> 1234  point I0157 address 0057 Data\_type bcd4 \*0x1234 -> 1234  point I0158 address 0058 Data\_type bcd4 \*0x1234 -> 1234  point F0160 address 0060 Data\_type float  point F0162 address 0062 Data\_type floatrev  group "AI-1-2"  operation gated  gatereg 1500  groupstat 1000  slave 1 func RHR  point I0100 address 0066 Data\_type int16  point I0101 address 0067 Data\_type short \*same as int16    group "AO-2 gated group"  operation gated  gatereg 1501  groupstat 1001  slave 1 func PMR  point F0170 address 0070 Data\_type float  point F0172 address 0072 Data\_type float  group "AO-3 trigger group"  operation triggered  triggerreg 1502  slave 1 func PMR  point F0180 address 0080 Data\_type float  point F0182 address 0082 Data\_type float |

1. 从站配置范例1

|  |
| --- |
| platform=RLC  baud=19200  data\_bits=8  parity=even  stop\_bits=1  slave\_address=1  rts\_on\_tx=on  link\_status\_reg = 2000  link\_ok\_reg = 2010  hold\_time=3000  \* FGD as slave station  \* Input Digital, DCS write FGD  m\_type = coil in= D0100 m\_addr = 0000 tmo\_action=quality tmo=5000  m\_type = coil in= D0101 m\_addr = 0001 tmo\_action=digital tmo=5000 reg=360  m\_type = coil in= D0102 m\_addr = 0002 tmo\_action=fault tmo=5000  m\_type = coil in= D0103 m\_addr = 0003  m\_type = coil in= D0104 m\_addr = 0004  \* Output Digital, FGD write DCS  m\_type = coil out= D0200 m\_addr = 0100  m\_type = coil out= D0201 m\_addr = 0101  m\_type = coil out= D0202 m\_addr = 0102  m\_type = coil out= D0203 m\_addr = 0103  m\_type = coil out= D0204 m\_addr = 0104  \* Input Analog DCS write FGD  m\_type = hr in= F0300 m\_addr = 0000 m\_data = FLOAT tmo\_action=quality tmo=5000  m\_type = hr in= F0302 m\_addr = 0002 m\_data = FLOAT tmo\_action=digital tmo=5000 reg=350  m\_type = hr in= F0304 m\_addr = 0004 m\_data = FLOAT tmo\_action=fault tmo=5000  m\_type = hr in= F0306 m\_addr = 0006 m\_data = FLOAT  m\_type = hr in= F0308 m\_addr = 0008 m\_data = FLOAT  \* Output Analog FGD write DCS  m\_type = hr out= F0500 m\_addr = 0300 m\_data = FLOAT  m\_type = hr out= F0502 m\_addr = 0302 m\_data = FLOAT  m\_type = hr out= F0504 m\_addr = 0304 m\_data = FLOAT  m\_type = hr out= F0506 m\_addr = 0306 m\_data = FLOAT  m\_type = hr out= F0508 m\_addr = 0308 m\_data = FLOAT  \* Input Packet DCS write FGD  m\_type = hr in= I0600 m\_addr = 0500 m\_data = INT16  m\_type = hr in= I0601 m\_addr = 0501 m\_data = INT16  m\_type = hr in= I0602 m\_addr = 0502 m\_data = INT16  m\_type = hr in= I0603 m\_addr = 0503 m\_data = INT16  m\_type = hr in= I0604 m\_addr = 0504 m\_data = INT16  \* Output Packet FGD write DCS  m\_type = hr out= I0700 m\_addr = 0600 m\_data = INT16  m\_type = hr out= I0701 m\_addr = 0601 m\_data = INT16  m\_type = hr out= I0702 m\_addr = 0602 m\_data = INT16  m\_type = hr out= I0703 m\_addr = 0603 m\_data = INT16  m\_type = hr out= I0704 m\_addr = 0604 m\_data = INT16 |

1. 留白