双向直流电源使用安全注意事项

在安装、维护我司直流电源产品时，为避免出现安全隐患， 请注意以下安全条例

1. 在使用本产品前， 请仔细阅读产品使用说明书。

2. 负责安装、维护我司设备的人员， 必须先经过培训，了解各种安全注意事项。

3. 本产品应放置、安装在远离液体的区域，以防止液体进入设备内部造成短路 。

4. 本产品放置、使用过程中要确保设备内无凝露。

5. 产品搬运过程中，注意防止划伤或跌落砸伤。

6. 产品上电调试过程中，请穿戴好电工绝缘防护设备，避免电击危险。

7. 禁止带电安装、拆除带电模块或线缆。

8. 严禁在雷电、雨、雪、大风等恶劣天气下安装、使用设备。禁止在易燃易爆区域内使用。产品运行过程中请勿触摸产品外壳。 建议产品运行结束 10 分钟内不要触摸产品外壳，避免烫伤或者电击危险。

9. 设备贴有铭牌，其中包含产品相关的重要参数信息，严禁人为涂改和损坏。

10. 非专业人士请勿打开产品内部，避免内部高压电击危险。

直流电源DC30KP-1000-100介绍

产 品特点

1. 超宽输入电压范围交流 180 – 460 Vac 正常工作，适应各种不同的电网环境。

2. 超宽输出输入电压范围直流 50Vdc-1000Vdc。

3. 高环境可靠性： 大面积灌胶防护，防盐雾、防潮湿、防霉变等显著提高。

4. 内置放电电路： 自动泄放残留电荷，简化系统设计，提高系统安全性。

5. 输入限功率控制，输出功率与输入电压的关系。当输入电压在 300Vac～460Vac 之间时可以输出最大功率 30 kW。

6. 输出恒功率控制，额定输入电压时， 电源允许输出功率为 30KW。

7. 温度限功率，-40℃-55℃环境温度以下，电源满功率输出 30kW，55°C ~75°C 线性降载至 50%。过温保护环境过温保护点为 80℃。当内部温度过高，电源过温度保护关机； 温度恢复后才能开机工作。

8. 具有无级限流调压功能；限流点在 0～100A 范围内可调，作为负载使用时电流范围 0～-100A 。输出电压调节调整最小调节步距为 0.1Vdc。

9. 保护功能，输入过/欠压保护，输出过压保护，三相交流输入电压小于 180Vac 或者大于 460Vac 时电源将停止工作。过压保护点可设置 ，设置范围为 51Vdc～1000Vdc，出厂默认值为 1000Vdc，过压保护后需要重启可开机。电源短路时保护，风扇故障时保护。

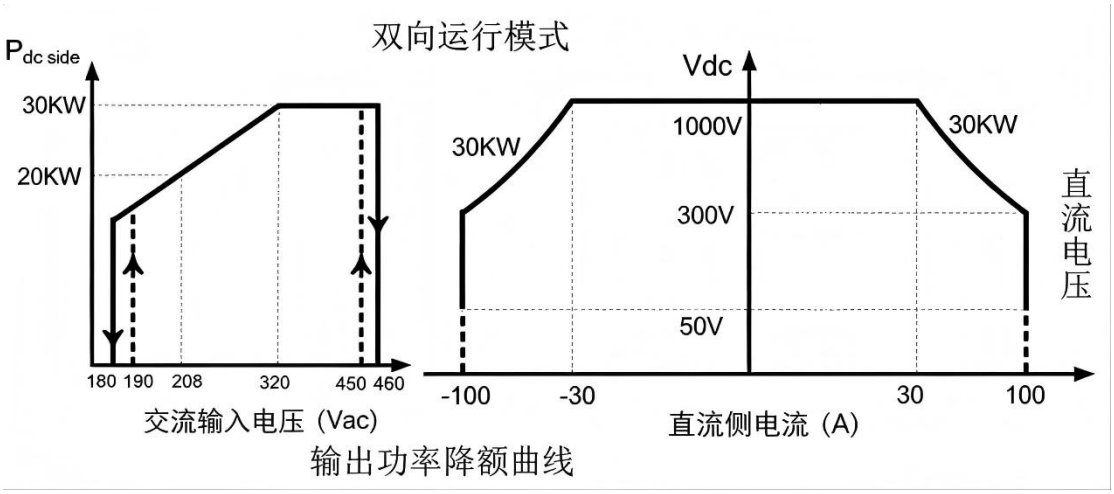
10. 通讯接口采用 RS485 接口 Modbus 协议通信和（选配）网口 Modbus-tcp 协议，选配 can 通讯。

技术规格表

类别	名称	参数
环境条件	工作温度	-40℃~+75℃， 55℃以上需降额使用
	储存温度	-40℃~+70℃
	相对湿度	≤95%RH，无冷凝
	冷却方式	风冷
	大气压力	79kPa~106kPa
	IP 防护等级	IP20
交流输入回馈电网	输入制式	三相+N+PE
	电压范围	180Vac~460Vac
	额定电压	380Vac
	最大电流	±60A
	电网频率	45Hz～65Hz
	额定频率	50Hz/60Hz
直流输出输入	有效电压范围	50Vdc~1000Vdc
	电流范围	-100A～100A（限流点可以设）
	稳压精度	<±0.2%+0.2%FS(50~1000V， 0~20MHz)
	稳流精度	≤±0.2%+0.5%FS （输出负载 20%～100%额定范围）
	纹波系数	满载工况：≤1%FS（典型值<0.5%）(50~1000V, 0~20MHz)
	电网调整率	≤±0.1%（输入交流电压380V）
安规参数	接地电阻电压纹波峰值因数THD	≤5% @50%～100%满载输出功率
		接地电阻≤0.1Ω ， 应能承受电流≥80A
	漏电流	漏电流≤3.5mA

	绝缘电阻	直流部分、交流部分对外壳之间以及交流部分对直流部分之 间的绝缘电阻≥10MΩ
机械参数	尺寸	22cm（高）×48cm （宽）×60cm（深不含端子和把手）
	重量	≤35kg

设备典型应用如下（30KW 双向电源）：

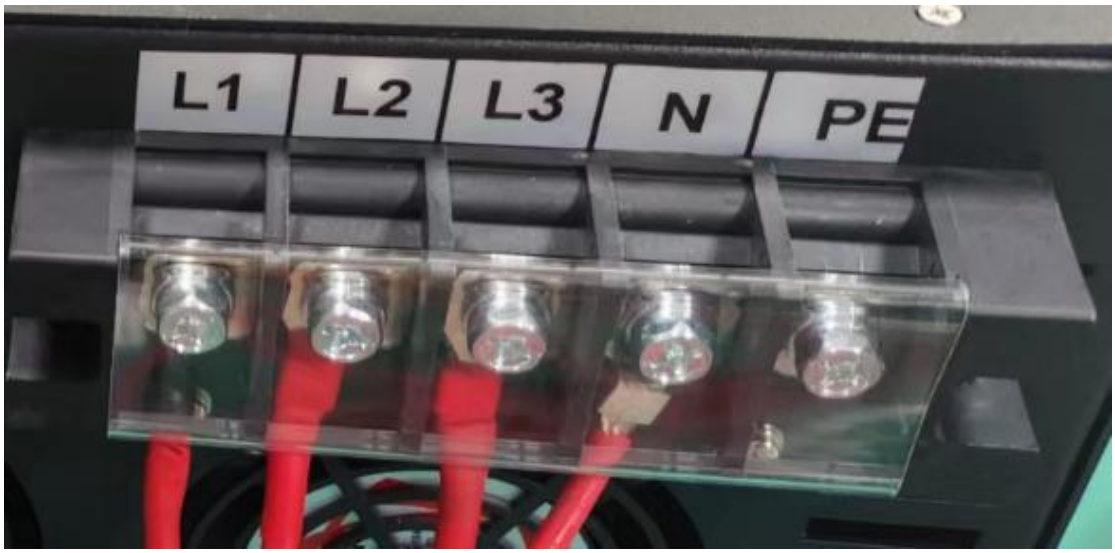


设备图片参考：（触摸屏）

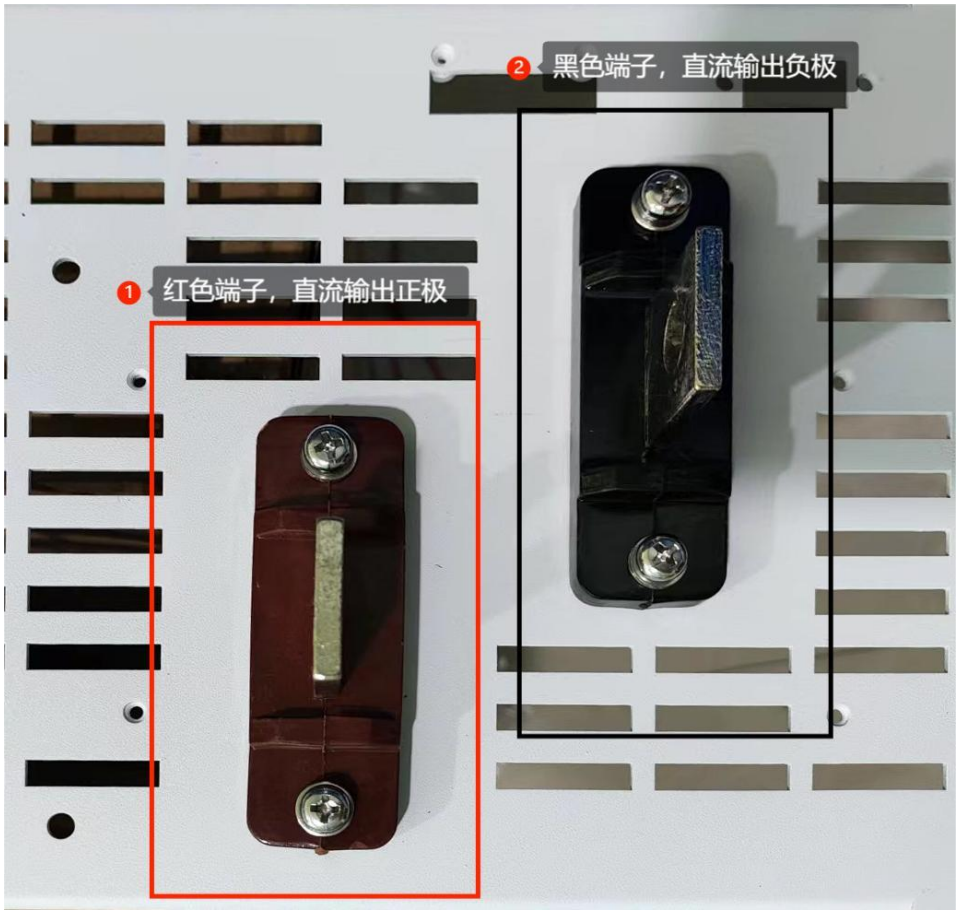
设备安装（以实际设备标志为准）

1.1 从包装箱子中取出直流电源

1.2 直流电源通电，交流输入接线在直流电源尾部，红框 1 所示，用十字螺丝刀打开保护盖，依照 2 图指示的接入三相 380V 的交流电。具体接线方法如下图



1.2.1 直流输出接线



2.设备上电开机

2.1 接好线路之后，在直流电源前面板，开机。

2.2 开机自检，约 10 秒

2.2.1 自检完成后，到主界面。主界面显示设备当前状态，最下面显示交流输入电压，环境温度，电源状态 对外输出定义为正电流称为电源模式，此模式下回读功率为正，回读电压为正，回读电流为正，有三个工作状态恒功，恒压，恒流。负载状态，直流侧吸收外部电流回馈给电网，定义为负电流称为负载模式，回读功率为负，回读电压为正，回读电流为负 有三个状态恒功，恒压，恒流。设备上位机远程控制会显示远程，屏幕操作显示本地。界面操作清除，可以清除电源故障信息。

2.2.2 输出设置

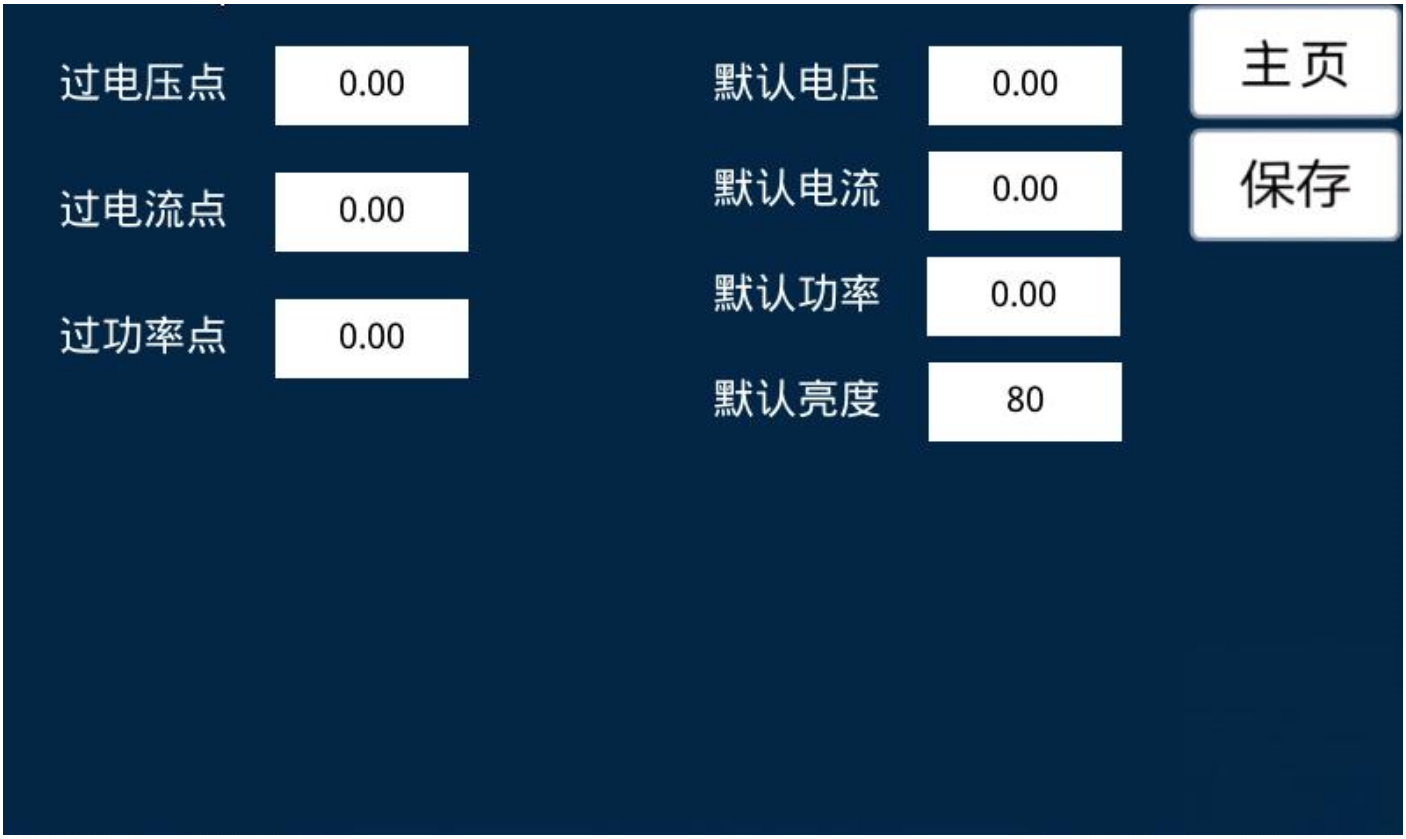
在主界面状态下，按下电压设定框，会出现小键盘，在小键盘上输入需要的电压，在小键盘按一下确认键，电压设置成功，按下开 /关 按键即可输出，输出状态下开按键底色会从灰色变成绿色。

通过数字小键盘按键输入需要的功率 电压 电流，比如设定功率 30KW，电压 600V，电流 20A，点击功率设定空白处，出现小键盘，输入 30，点击小键盘回车键（符号 ⏎），点击电压设定空白处，出现小键盘，输入 600，点击小键盘回车键（符号 ⏎），点击电流设定空白处，出现小键盘，输入 20，点击小键盘回车键（符号 ⏎）。最后点击确认

按下屏幕 开 按键有直流输出时，按键会变为绿色，电源输出工作。如下图

2.2.3 在输出状态下可以微调按键调整输出电压。下图红色框框中的-10 -1 +1 +10，可以进行当前电流加 10A 加 1A 减 10A 减 1A 的操作，按下-10 后设定电流会调整，从 20A 变化为 10A，负电流设置，负电流-20A 按下-10 之后，电流会变化成-30A。

2.3 自定义电压和电流，及时调用。在主界面按下过压设置进入过压设置界面再按下自定义默认电压/电流/功率



在自定义中设定常用的电压 电流信息可以在开机状态下直接调用比如

自定义 功率 30 电压 50 电流 30

按屏幕上保存键，保存后重启上电

开机默认功率 30kW 电压 50V 电流 30A

2.4 直流电源本地编程设置，主界面按一下编程按键，双向电源需要设置功率信息。

编程设定框 编辑需要的电压电流跳变，可以实现 100 步编辑，

比如实现 3 步跳变 100 次 设定如下

设定 步 1 电压 100 电流 30 功率 30 时间 30 按保存按键

设定 步 2 电压 390 电流 100 功率 30 时间 60 按保存按键

设定 步 3 电压 220 电流 100 功率 30 时间 180 按保存按键

完成编程后在步循环处输入 100 步数处输入 3 再按保存/确认按键 运行按键就可以输出编程。

2.5 直流电源通讯设置，支持 RS 485 MODBUS RTU 和 LAN MODBUS TCP 协议，按通讯界面，选择需要设置的通讯。设定好后按下保存。

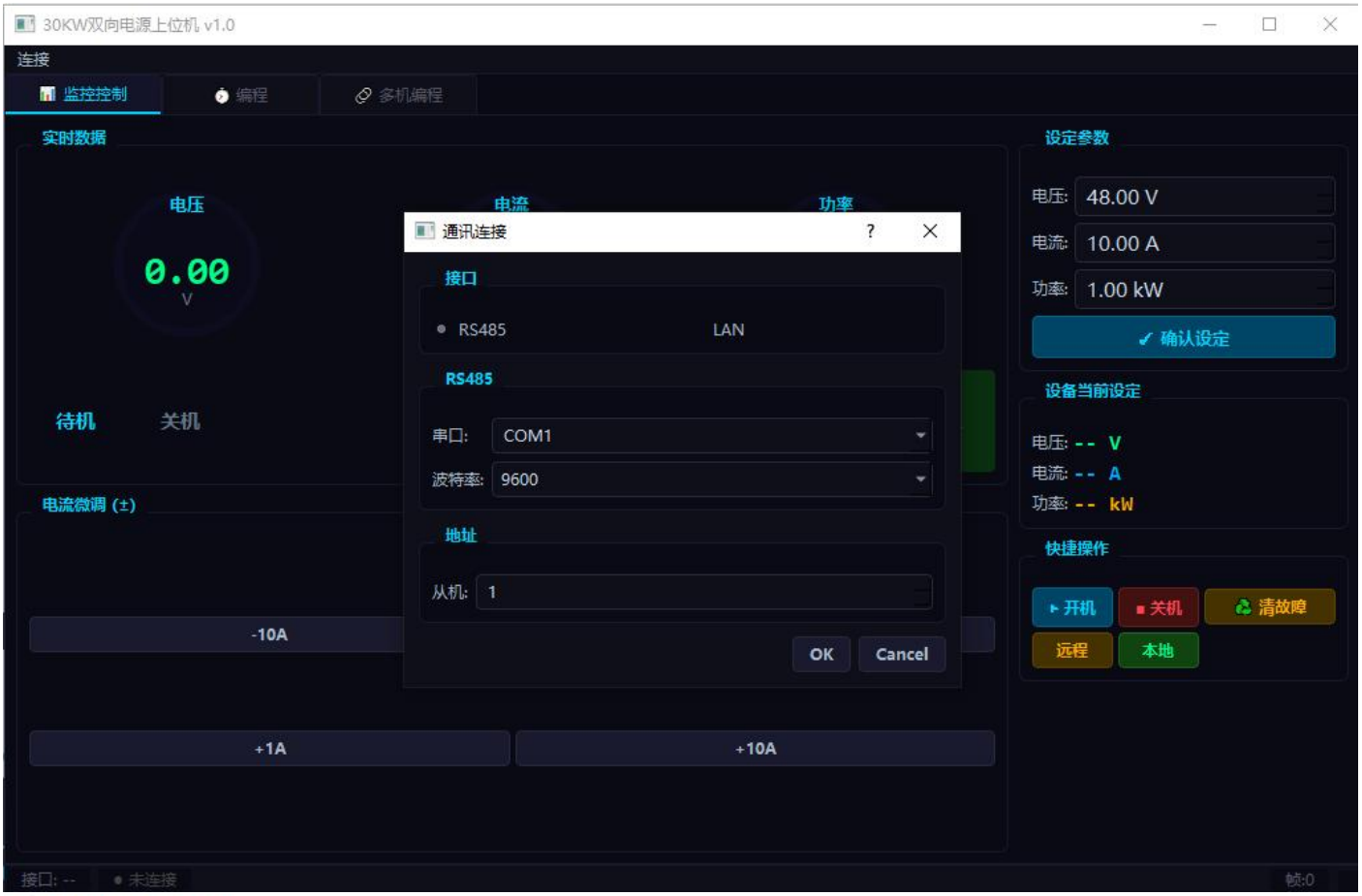
485 设置　网口设置

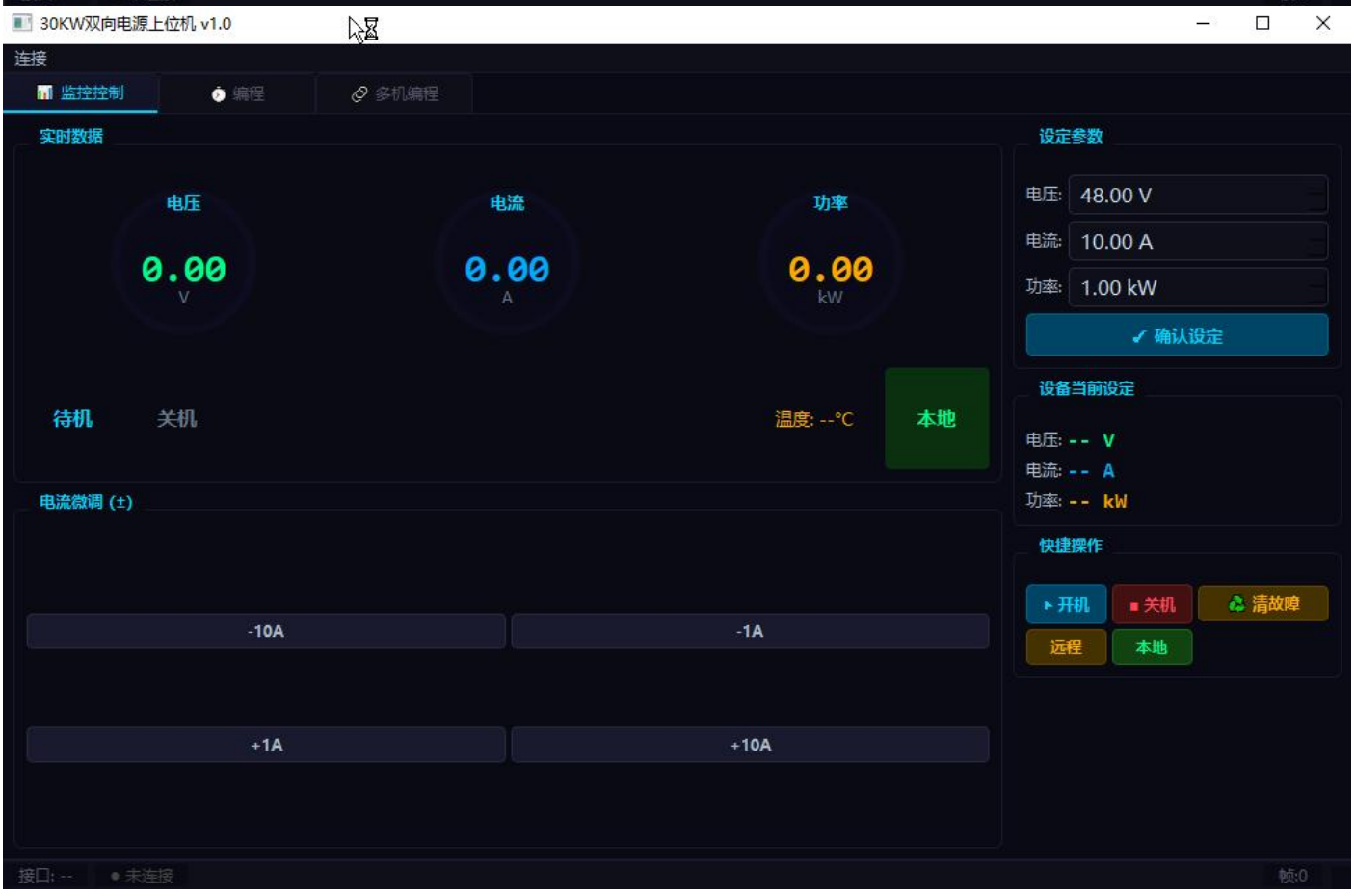
2.6 设备信息查看，按键系统可以进入系统界面查看设备相关信息以及帮助信息。

操作手册提供操作思路参考流程，实际设备在不断升级中，会有和操作手册不匹配的地方，以厂家指导的实际操作为准，请及时联系厂家更新。操作显示界面对应不同的产品有细微区别，以实际为准。

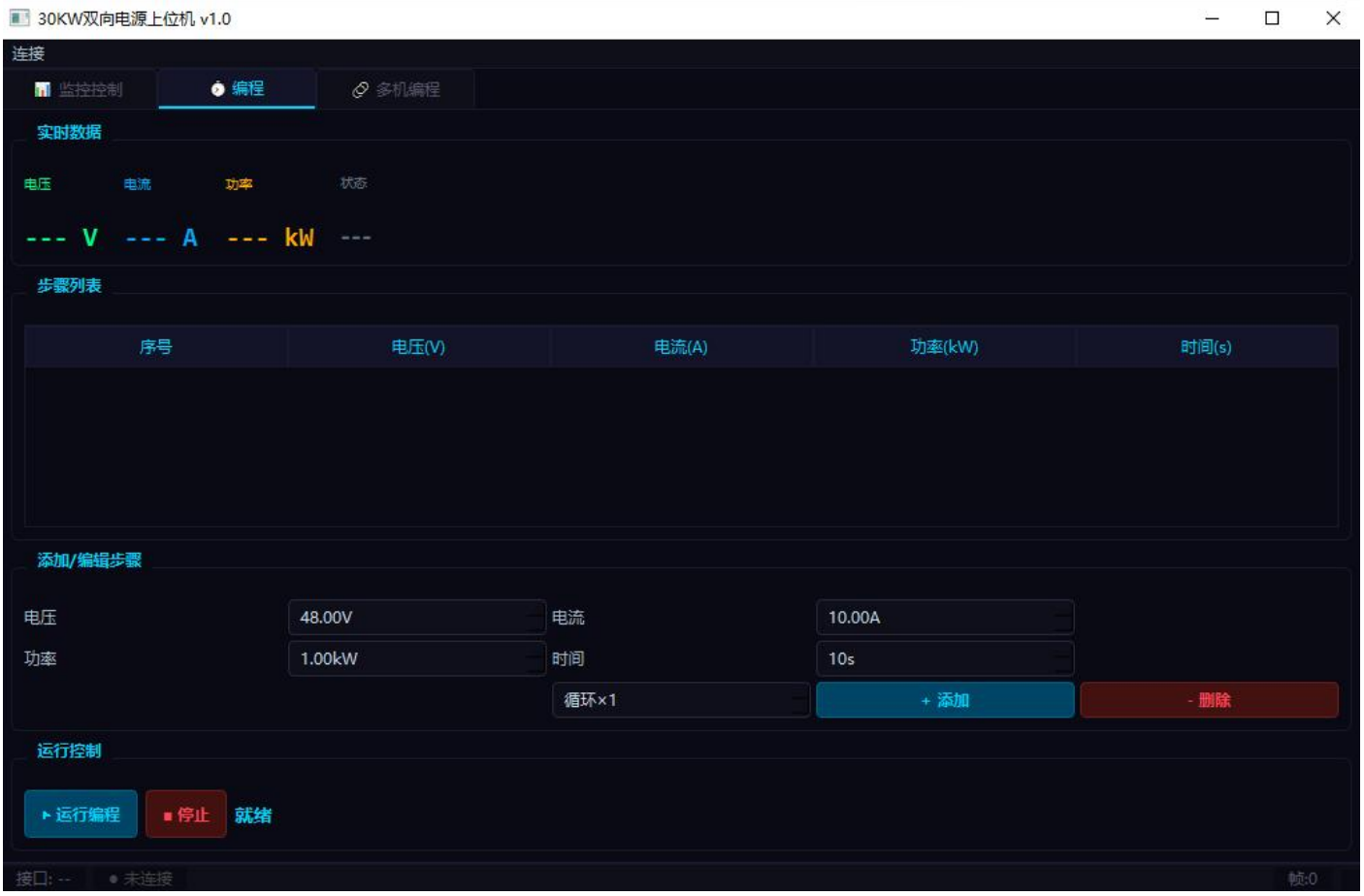
选配：通讯 RS485 和网口

2.7 默认自带上位机控制。主界面如下



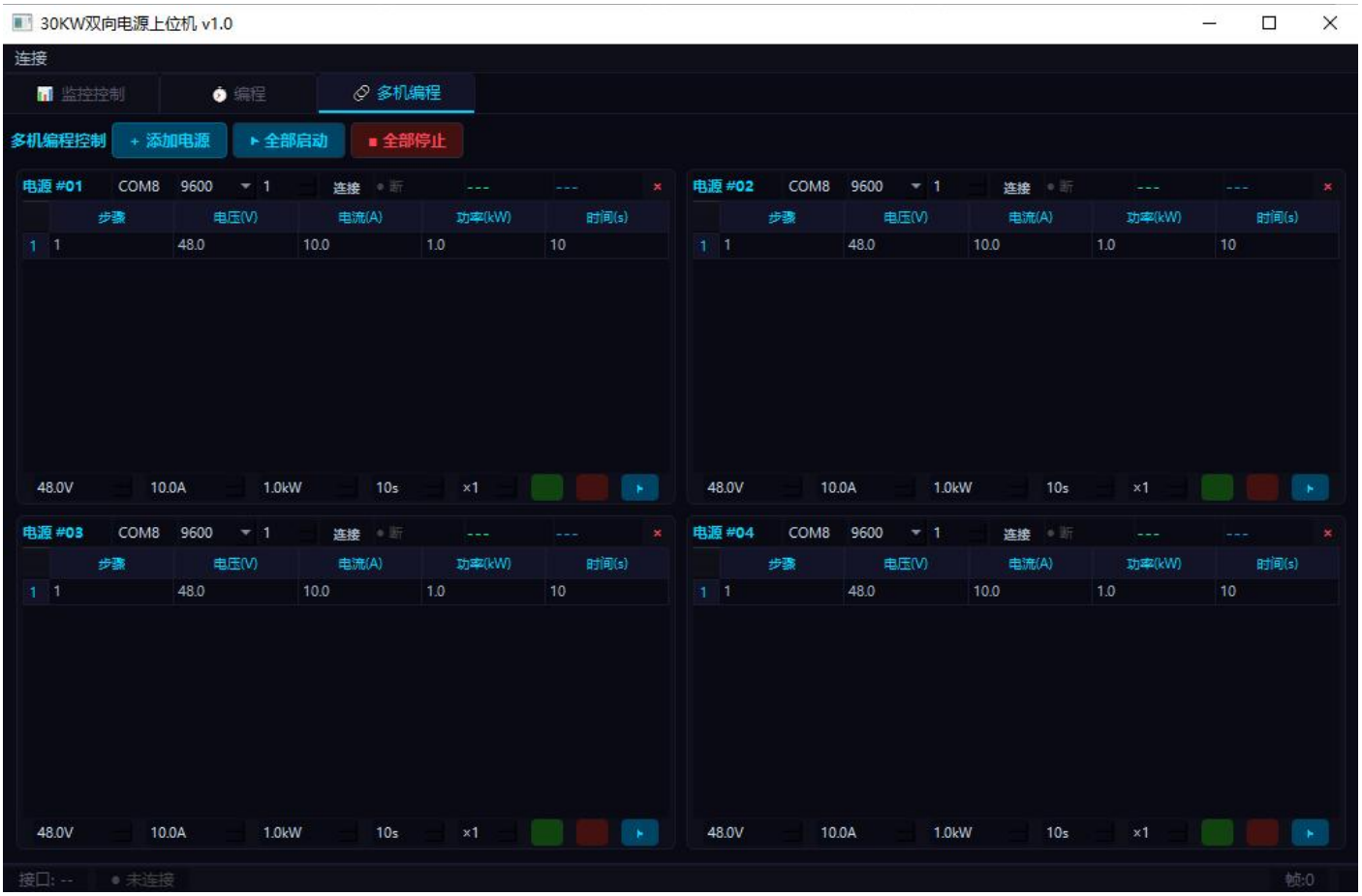


2.8 单机编程界面


30KW双向电源上位机 v1.0
连接
监控控制
编程
多机编程
实时数据
电压
电流
功率
状态
--- V
--- A
--- kW

步骤列表
序号
电压(V)
电流(A)
功率(kW)
时间(s)
添加/编辑步骤
电压
48.00V
电流
10.00A
功率
1.00kW
时间
10s
循环×1
+ 添加
- 删除
运行控制
▸ 运行编程
■ 停止
就绪
接口: --
● 未连接
帧:0

2.9 多机编程界面


30KW双向电源上位机 v1.0
连接
监控控制
编程
多机编程
多机编程控制
+ 添加电源
▸ 全部启动
■ 全部停止
电源 #01
电源 #02
电源 #03
电源 #04
COM8
9600
连接
步骤
电压(V)
电流(A)
功率(kW)
时间(s)
48.0V
10.0A
1.0kW
10s
×1
接口: --
● 未连接
帧:0

30KW双向直流电源 Modbus RTU/TCP 通讯协议手册

版本: V1.0 |**日期**: 2026-06-09 |**产品**: 30KW双向直流电源
协议: Modbus RTU (RS485) / Modbus TCP (LAN)

目录

1.产品概述

1.1 产品简介

本手册适用于30KW双向直流电源的485/LAN通讯控制。电源内置TFT触摸屏作为Modbus从机，通过RS485或LAN接口与上位机通讯，实现远程监控、参数设定和编程控制。

1.2 通讯架构

上位机(PC/Python/Qt5) ── RS485 / LAN ──> TFT触摸屏(Modbus从机) ── CAN ──> 电源模块

- 上位机通过Modbus RTU/TCP协议与TFT触摸屏通讯
- 触摸屏作为Modbus从机，将Modbus指令转换为电源模块的CAN控制指令
- 支持双接口：RS485(Modbus RTU) 和 LAN(Modbus TCP)

1.3 技术参数

参数	范围	精度	说明
设定电压	50 ~ 1000 V	0.1V	寄存器值×10
设定电流	-100 ~ +100 A	0.1A	正=源模式，负=负载模式
设定功率	0 ~ 30000 W	1W	
输出电压	50 ~ 1000 V	1V	只读
输出电流	-100 ~ +100 A	1A	只读，有符号
输出功率	0 ~ 30000 W	1W	只读
从机地址	1 ~ 247	-	默认1

2.物理接口与接线

2.1 RS485 接口参数

参数	默认值	可选值
接口类型	RS485 半双工	A+/B-差分信号
波特率	9600 bps	4800 / 9600 / 19200 / 38400 / 115200
数据位	8	固定
校验位	无 (None)	None / Odd / Even
停止位	1	-
从机地址	1	1 ~ 247 (屏幕通讯界面可调)

2.2 LAN 接口参数

参数	默认值	说明
接口类型	RJ45	以太网
协议	Modbus TCP	-
IP 地址	192.168.1.168	屏幕通讯界面可修改
端口	502	标准Modbus TCP端口

2.3 RS485 接线

电脑USB ── USB转RS485模块 ── A+(A) ──── 屏幕RS485 A+ B-(B) ──── 屏幕RS485 B-

⚠ A+接A+, B-接B-，不可接反。长距离(>50m)建议加120Ω终端电阻。
USB转RS485模块推荐：CH340、CP2102、FT232芯片方案。

2.4 LAN 接线

电脑网口 ── 网线 ── 屏幕RJ45口（或通过交换机/路由器连接）

3.Modbus 协议基础

3.1 Modbus RTU 帧格式

字段	长度	说明	示例
地址码	1字节	从机地址 1~247	0x01
功能码	1字节	03H/04H/06H	0x06
数据区	N字节	寄存器地址+数据	00 00 00 01
CRC16	2字节	低字节在前	89 CA

帧与帧之间用**3.5个字符时间**的空闲间隔分隔。

3.2 支持的功能码

功能码	名称	用途	典型场景

功能码	名称	用途	典型场景
03H(3)	读保持寄存器	读1~9个16位寄存器	读全部设定值和状态
04H(4)	读输入寄存器	读只读16位寄存器	读实时输出电压/电流
06H(6)	写单寄存器	写1个16位寄存器	切换远程/本地、下发命令、写设定值

3.3 数据格式与转换规则

所有寄存器为**16位无符号整数**：

寄存器	存储范围	物理值转换	示例
SetVoltage	0~10000	val ÷ 10= V	480 → 48.0V
SetCurrent	0~65535	s16(val) ÷ 10= A	100 → +10.0A, 65436 → -10.0A
SetPower	0~30000	val= W	1000 → 1000W
OutVoltage	0~1000	val= V	48 → 48V
OutCurrent	0~65535	s16(val)= A	10 → 10A, 65526 → -10A
OutPower	0~30000	val= W	2000 → 2000W
RemoteMode	0~65535	0=本地, ≠0=远程	-
Command	1~4	见5.2节	-
PowerState	0/1	0=关机, 1=开机	-

有符号数转换:s16(v) = v < 32768 ? v : v - 65536

4.寄存器映射表

基地址从**0x0000**开始，共**9**个寄存器：

地址	名称	功能码	读写	类型	说明
0000	**RemoteMode**	03H/06H	RW	U16	0=本地, ≠0=远程
0001	**Command**	06H	WO	U16	1=开机, 2=关机, 3=清故障, 4=确认参数
0002	**SetVoltage**	03H/06H	RW	U16	设定电压×10
0003	**SetCurrent**	03H/06H	RW	S16	设定电流×10(有符号)
0004	**SetPower**	03H/06H	RW	U16	设定功率(W)
0005	**OutVoltage**	04H	RO	U16	输出电压(V)
0006	**OutCurrent**	04H	RO	S16	输出电流(A,有符号)
0007	**OutPower**	04H	RO	U16	输出功率(W)
0008	**PowerState**	04H	RO	U16	0=关机, 1=开机

4.1 读写权限分类

权限	含义	寄存器
RW	可读可写 (03H/06H)	RemoteMode, SetVoltage, SetCurrent, SetPower
WO	只写 (06H)，读回0	Command
RO	只读 (04H)	OutVoltage, OutCurrent, OutPower, PowerState

5.寄存器详细说明

5.1 RemoteMode (0x0000) — 远程/本地模式

属性	值
功能码	03H(读) / 06H(写)
有效值	0 = 本地, 1~65535 = 远程

行为说明：

- **本地模式**: 屏幕可操作，上位机**只可读不可写**
- **远程模式**: 屏幕锁定设定值输入，上位机可远程控制

5.2 Command (0x0001) — 控制命令

属性	值
功能码	06H(只写)
写入后自动清零	是

命令值	功能	说明	前置条件
1	PowerOn 开机	启动电源输出	需先确认参数(Command=4)
2	PowerOff 关机	关闭电源输出	无
3	ClearFault 清故障	清除保护故障状态	远程模式
4	ConfirmEnter 确认参数	保存设定值→Flash→下发CAN命令	先写 SetVoltage/Current/Power

⚠**关键**: 修改设定值(SetVoltage/SetCurrent/SetPower)后，**必须发送Command=4**才会生效！

5.3 SetVoltage (0x0002) — 设定电压

属性	值
范围	0~~10000 (0.0V~~1000.0V)
转换	实际电压(V) = 寄存器值 / 10

5.4 SetCurrent (0x0003) — 设定电流

属性	值
范围	有符号: 0~1000(正), ≥32768(负)
转换	实际电流(A) = s16(寄存器值) / 10

寄存器值	有符号值	物理值
100	+100	+10.0A (源模式)
500	+500	+50.0A
65436 (0xFF9C)	-100	-10.0A (负载模式)
65036 (0xFE0C)	-500	-50.0A

5.5 SetPower (0x0004) — 设定功率

属性	值
范围	0 ~ 30000 W
转换	实际功率(W) = 寄存器值

5.6 只读寄存器 (0x0005~0x0008)

地址	名称	功能码	说明
0005	OutVoltage	04H	实时输出电压(V), 范围0~1000

地址	名称	功能码	说明
0006	OutCurrent	04H	实时输出电流(A), 正=源 负=负载
0007	OutPower	04H	实时输出功率(W), 范围0~30000
0008	PowerState	04H	0=电源关机, 1=电源开机

6.通讯调试实例

6.1 调试工具推荐

工具	用途	平台
30KW双向电源上位机.exe	本产品配套上位机	Windows
Modbus Poll	标准Modbus调试	Windows
SSCOM	串口HEX收发调试	Windows
Python + pymodbus	编程调试	全平台

6.2 实例1: 切换到远程模式

目的: 通过Modbus将屏幕从本地控制切换到远程控制。

功能码: 06H (写单寄存器)寄存器: 0x0000 (RemoteMode)写入值: 1 (远程模式)

发送: 01 06 00 00 00 01 89 CA│ │ └──────┘ └──────┘ └──────┘ │ │ 地址=0 数据=1 CRC16│ └── 功能码(写单寄存器) └── 从机地址=1

响应: 01 06 00 00 00 01 89 CA (原样返回)

预期结果: 屏幕显示"远程"，设定值输入框锁住。

6.3 实例2: 设定48V电压 → 确认参数 → 开机

目的: 完整演示"设参→确认→开机"三步标准流程。

步骤A - 写设定电压 = 48.0V

寄存器值 = 48.0 × 10 = 480 = 0x01E0

发送: 01 06 00 02 01 E0 28 2F地址=2(SetVoltage), 值=0x01E0(480)响应: 01 06 00 02 01 E0 28 2F

步骤B - 确认参数 (Command=4)

发送: 01 06 00 01 00 04 59 CA 地址=1(Command), 值=4(ConfirmEnter)响应: 01 06 00 01 00 04 59 CA

屏幕将设定值保存到Flash并通过CAN发送给电源模块。

步骤C - 开机 (Command=1)

发送: 01 06 00 01 00 01 98 0A 地址=1(Command), 值=1(PowerOn)响应: 01 06 00 01 00 01 98 0A

预期结果: 屏幕显示"设备开机"，电源启动输出，表盘开始显示实时数据。

6.4 实例3: 批量读取全部状态 (9个寄存器)

目的: 一次Modbus请求读取所有设定值和状态，减少通讯次数。

功能码: 03H (读保持寄存器)起始地址: 0x0000数量: 9

发送: 01 03 00 00 00 09 84 0C│ │ └──────┘ └──────┘ └──────┘│ │ 起始=0 数量=9 CRC16│ └── 功能码 └── 从机地址

响应: 01 03 12 00 01 00 00 01 E0 00 64 03 E8 00 30 00 0A 07 D0 00 01 XX XX解析:│ 12 = 18字节(9个寄存器×2) ├── 00 01 = RemoteMode = 1(远程) ├── 00 00 = Command = 0(已清零) ├── 01 E0 = SetVoltage = 480 → 48.0V ├── 00 64 = SetCurrent = 100 → +10.0A ├── 03 E8 = SetPower = 1000W ├── 00 30 = OutVoltage = 48V ├── 00 0A = OutCurrent = +10A ├── 07 D0 = OutPower = 2000W └── 00 01 = PowerState = 1(开机)

6.5 实例4: 设定负电流（负载模式）

目的: 设定-50A电流，电源进入负载模式（吸收外部电能）。

50A → 500(×10) → 存为负数: 65536 - 500 = 65036 = 0xFE0C

步骤1: 写设定电流发送: 01 06 00 03 FE 0C A0 5F地址=3(SetCurrent), 值=0xFE0C(65036 → -50.0A)响应: 01 06 00 03 FE 0C A0 5F

步骤2: 确认参数发送: 01 06 00 01 00 04 59 CA

步骤3: 开机发送: 01 06 00 01 00 01 98 0A

6.6 实例5: Python 代码调试 (pymodbus)

import timefrom pymodbus.client import ModbusSerialClient

连接 RS485

client = ModbusSerialClient(port="COM8", baudrate=9600,parity="N", stopbits=1, bytesize=8, timeout=0.2)assert client.connect(), "连接失败!"

---- 1. 远程模式 ----

client.write_regi$ster(0,1,device_id = 1)$# RemoteMode=1time.sleep(0.05)

---- 2. 设定电压48V + 电流10A + 功率1000W ----

client.write_regi$ster(2,480,device_id = 1)$# SetVoltage=480 → 48.0Vtime.sleep(0.05)client.write_regi$ster(3,100,device_id = 1)$# SetCurrent=100 → 10.0Atime.sleep(0.05)client.write_regi$ster(4,1000,device_id = 1)$# SetPower=1000Wtime.sleep(0.05)

---- 3. 确认参数 (必须!) ----

client.write_register(1,4, device_id = 1)# Command=4(ConfirmEnter)time.sleep(0.1)

---- 4. 开机 ----

client.write_register(1,1, device_id = 1)# Command=1(PowerOn)time.sleep(0.5)

---- 5. 读取状态 ----

rr = client.read_holding_registers(0,9, device_id = 1)ifrrandnotrr.isError():print(f"RemoteMode = {rr.registers[0]}")print(f"SetVoltage = {rr.registers[2]/10:.1f} V")print(f"SetCurrent = {rr.registers[3]/10:.1f} A")print(f"PowerState = {'开机' if rr.registers[8] else '关机'}")

---- 6. 读取实时数据 ----

rr2 = client.read_input_registers(5,4, device_id = 1)ifrr2andnotrr2.isError():print(f"OutVoltage = {rr2.registers[0]} V")cur = rr2.registers[1]if cur >= 32768: cur -= 65536print(f"OutCurrent = {cur} A")print(f"OutPower = {rr2.registers[2]} W")print(f"PowerState = {'开机' if rr2.registers[3] else '关机'}")

---- 7. 关机 ----

client.write_register(1,2, device_id = 1)# Command=2(PowerOff)client.close()

6.7 实例6: SSCOM/串口助手 HEX 调试

使用SSCOM等串口工具，选择COM口和波特率，勾选"HEX发送"和"HEX显示"：

【测试连接 - 读PowerState】发送: 01 04 00 08 00 01 11 CB接收: 01 04 02 00 01 78 F0 → PowerState=1(开机) ✓

【测试远程】发送: 01 06 00 00 00 01 89 CA接收: 01 06 00 00 00 01 89 CA → 远程模式成功 ✓

【测试关机】发送: 01 06 00 01 00 02 D9 CB接收: 01 06 00 01 00 02 D9 CB → 关机成功 ✓

【读全部9个寄存器】发送: 01 03 00 00 00 09 84 0C接收: 01 03 12 ... XX XX → 18字节数据

6.8 实例7: 从机地址=1以外的调试

如果屏幕COM界面设置的从机地址不是1，需要修改地址字节：

从机地址=6 时: 远程模式: 06 06 00 00 00 01 29 F4 (首字节06替代01, CRC不同!) 读全部: 06 03 00 00 00 09 24 3A 开机: 06 06 00 01 00 01 98 0A → 需要重新算CRC!

CRC计算工具: https://www.lammertbies.nl/comm/info/crc-calculation

7.典型工作流程

7.1 远程监控流程

1. RS485或LAN连接上位机2. 写 RemoteMode = 13. 循环(间隔≥1秒): ├─ 读 OutVoltage (0x0005) ├─ 读 OutCurrent (0x0006) ├─ 读 OutPower (0x0007) └─ 读 PowerState (0x0008)

优化: 使用FC=03H一次性读9个寄存器（见实例3），减少通讯次数。

7.2 远程控制流程

1. 写 RemoteMode = 1 → 切换到远程2. 写 SetVoltage / SetCurrent / SetPower → 写入参数3. 写 Command = 4 → 确认参数（保存+下发CAN）⭐ ⭐ ⭐ 4. 写 Command = 1 → 开机5. 监控运行数据6. 写 Command = 2 → 关机

⚠**步骤3是最容易被忽略的一步！**如果不发Command=4，参数不会真正下发到电源模块。

7.3 故障恢复流程

1. 检测到保护 → 电源自动关机2. 读 OutVoltage/OutCurrent/OutPower 确认异常3. 排查并排除故障原因4. 写 Command = 3 (ClearFault) → 清除故障状态5. 重新设参 → Command=4 → Command=1 → 恢复运行

8.上位机使用指南

8.1 配套上位机

30KW双向电源上位机.exe(上位机)

标签页	功能
监控控制	实时表盘(V/A/kW) + 设定值输入 + 开关机/电流微调
编程	步骤序列编程，支持循环和定时
多机编程	默认支持4台设备同时编程控制

8.2 快速上手

1. 打开 **30KW双向电源上位机.exe**. 菜单 → 连接设置 (Ctrl+C)3. 选择 RS485: COM口 + 波特率 + 从机地址 或选择 LAN: IP + 端口 + 从机地址4. 点击确定 → 状态栏显示"● 已连接"5. 点击"远程" → 输入V/A/kW → "确认设定" → "开机"

9.CRC16 校验算法

9.1 算法参数

参数	值
多项式	0xA001 (Modbus反向)
初始值	0xFFFF
输出顺序	低字节在前

9.2 C语言

uint16_t CRC16_Mo$dbus(uint8_t * data, intlen)uint16_t crc = 0xFFFF$; for $inti = 0; i < len; i ++$ { crc ^= data[i]; for $intj = 0; j < 8; j ++$ { if (crc & 0x0001) crc = (crc >> 1) ^ 0xA001; else crc = crc >> 1; } } return crc;}

9.3 Python

def crc16_modbus(data: bytes) -> int: crc = 0xFFFF for b in data: crc ^= b for _ in range(8): if crc & 1: crc = (crc >> 1) ^ 0xA001 else: crc >>= 1 return crc

使用示例

frame = bytes([0x01, 0x06, 0x00, 0x00, 0x00, 0x01])crc = crc16_modbus(frame)print$f"CRC16 = 0xcrc: 04X"$ # 输出: 0xCA89print(f"帧中写为: {crc & 0xFF:02X} {(crc >> 8) & 0xFF:02X}") # 89 CA

9.4 在线验证

测试数据01 06 00 00 00 01：

- CRC结果:0xCA89
- 帧中顺序:89 CA(低字节在前)

在线工具: https://www.lammertbies.nl/comm/info/crc-calculation

10.故障排查

10.1 常见问题

现象	可能原因	解决方法
上位机连接失败	波特率/串口号不匹配	检查设备管理器COM口, 确认波特率与屏幕一致
连接成功但读写无响应	从机地址不匹配	确认屏幕COM界面显示的从机地址
写寄存器返回异常	不在远程模式	先写RemoteMode=1
设定值一直为0	参数未确认	写完设定值后必须发Command=4
读取数据全为0	电源未开机	先发Command=1开机
CRC校验失败	接线不良/干扰	换短线, 加120Ω终端电阻
偶尔丢帧	波特率过高	降低到9600或19200
屏幕不显示"远程"	Modbus轮询未启动或CAN未就绪	确认固件已更新

10.2 调试检查清单

□ 屏幕COM界面: 波特率和从机地址是否正确？□ RS485接线: A+→A+, B-→B- 是否正确？□ 上位机COM口是否与设备管理器一致？□ 是否先切换到远程模式(RemoteMode=1)？□ 设定值后是否发送了确认参数(Command=4)？□ CAN通讯是否正常(屏幕是否显示实时电压)？□ 120Ω终端电阻是否在长距离时加上？

10.3 从机地址变更后CRC重新计算

从机地址	远程模式指令	读全部指令
1	01 06 00 00 00 01 89 CA	01 03 00 00 00 09 84 0C
2	02 06 00 00 00 01 89 F9	02 03 00 00 00 09 84 3F
6	06 06 00 00 00 01 29 F4	06 03 00 00 00 09 24 3A
68	44 06 00 00 00 01 89 22	44 03 00 00 00 09 CB 8E

11.附录

11.1 HEX 快捷指令表 (从机地址=1)

功能	HEX指令
远程模式	01 06 00 00 00 01 89 CA
本地模式	01 06 00 00 00 00 C9 DB
开机	01 06 00 01 00 01 98 0A
关机	01 06 00 01 00 02 D9 CB
清故障	01 06 00 01 00 03 18 0B
确认参数	01 06 00 01 00 04 59 CA
读全部(9个)	01 03 00 00 00 09 84 0C
读OutVoltage	01 04 00 05 00 01 90 08
读PowerState	01 04 00 08 00 01 11 CB
写Voltage=48V	01 06 00 02 01 E0 28 2F
写Current=10A	01 06 00 03 00 64 79 CB
写Power=1000W	01 06 00 04 03 E8 C8 3D
写Current=-10A	01 06 00 03 FF 9C 59 F9

11.2 寄存器速查卡

地址	名称	FC	范围	转换
0	RemoteMode	03/06	0~65535	-
1	Command	06	1~4	-
2	SetVoltage	03/06	0~10000	÷10 = V
3	SetCurrent	03/06	0~65535	s16÷10 = A
4	SetPower	03/06	0~30000	= W
5	OutVoltage	04	0~1000	= V
6	OutCurrent	04	0~65535	s16 = A
7	OutPower	04	0~30000	= W
8	PowerState	04	0/1	0=关, 1=开

11.3 Python 完整示例工程

"""30KW双向电源 Modbus 调试脚本pip install pymodbus pyserial"""import timefrom pymodbus.client import ModbusSerialClient

class PowerSupply:definit$self, port, baud = 9600, unit = 1$:self.client = ModbusSerialClient(port=port, baudrate=baud, parity='N',stopbits=1, bytesize=8, timeout=0.2)self.unit = unitassert self.client.connect(), f"连接{port} 失败!"

def remote(self): self.client.write_regi$ster(0,1, device_id = self.unit)deflocal$(self): self.client.write_regi$ster(0,0, device_id = self.unit)def power_on(self): self.client.write_regi$ster(1,1, device_id = self.unit)defpower$_off(self): self.client.write_regi$ster(1,2, device_id = self.unit)def clear_fault(self):self.client.write_regi$ster(1,3, device_id = self.unit)

def set_and_confirm(self, voltage, current, power_w): """设定参数并确认""" self.client.write_register(2, int(voltage * 10), device_id=self.unit) time.sleep(0.05) ic = int(current * 10) if ic < 0: ic += 65536 self.client.write_regi$ster(3, ic, device_id = self.unit)time$.sleep(0.05) self.client.write_regi$ster(4, int(power_w), device_id = self.unit)time$.sleep(0.05) self.client.write_regi$ster(1,4, device_id = self.unit)# ConfirmEnter time.sleep(0.1)

def read_all(self): rr = self.client.read_holding_regis$ters(0,9, device_id = self.unit)if rrandnotrr$.isError(): return { 'remote': rr.registers[0] != 0, 'set_v': rr.registers[2] / 10, 'set_c': (rr.registers[3] - 65536 if rr.registers[3] >= 32768 else rr.registers[3]) / 10, 'set_p': rr.registers[4], 'out_v': rr.registers[5], 'out_c': rr.registers[6] - 65536 if rr.registers[6] >= 32768 else rr.registers[6], 'out_p': rr.registers[7], 'power_on': rr.registers[8] == 1, }

def close(self): self.client.close()

---- 使用示例 ----

ifname== 'main':ps = PowerSupply'$COM8', 9600, unit = 1$ps.remote()ps.set_and_confirm(voltage=48.0, current=10.0, power_w=1000)ps.power_on()time.sleep(1)data = ps.read_all()print(f"电压: {$data['out_v']V$, 电流: {$data['out_c']A$, 开机: {$data['power_on']$")ps.power_off()ps.close()

11.4 修订历史

版本	日期	修订内容
V1.0	2026-06-09	初版发布: Modbus RTU/TCP协议, 9寄存器, 7组调试实例