ZH-T08NTC 8 通道热电阻测温模块

使用说明书(V6.51)

注：此使用说明通讯举列如无特别说明，模块拨码地址都是以 01 为列，Modbus-TCP 协议前面 4 个字节都填 00。通讯字符串都是以十六进制表示。

一、概述

本模块采用高精度 32 位 AD 芯片+ARM32 位工业级 MCU，精度高，抗干扰好。可测量 0.02Ω至 40MΩ阻值间的各种热电阻阻值，并计算出温度，可广泛用于各种热电阻测温以及热电阻阻值校正场合。

具有以下特点：

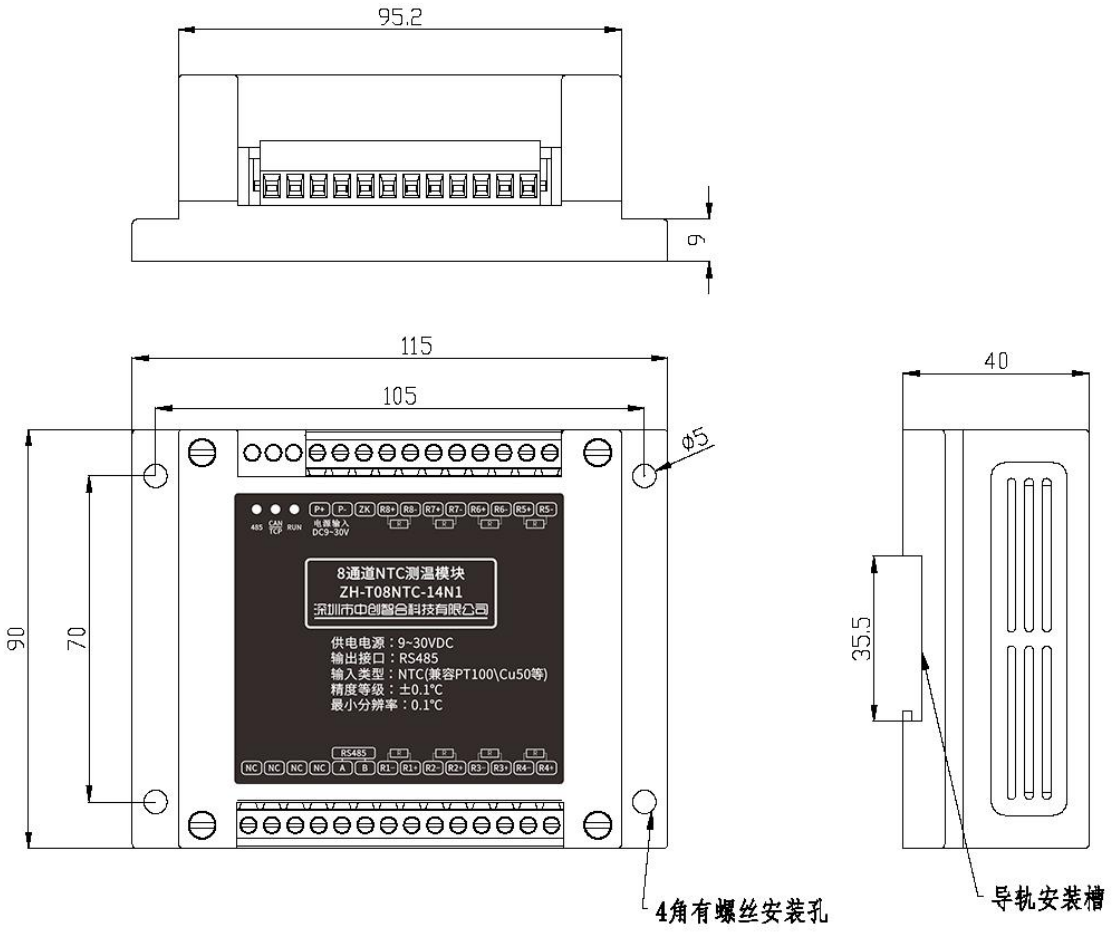
- ✧ 具有宽电源供电 9-30V；
- ✧ 32 位高精度 AD，阻值分辨率达 1mΩ，精度最高达千分之一；温度分辨率 0.1℃，精度达±0.1℃。
- ✧ 每个通道量程独立，且可自行设置，可选 25Ω、1K、5K、20K、100K、1M、10M、40M 八种电阻量程以及 PT100、PT1000、Cu50、Cu100、NTC(10k B=3435)、NTC(10k B=3950)、NTC(100k B=3950)、NTC(MF501 B=4100)等各种热电阻；
- ✧ 具有四种采样速率可调；
- ✧ 可屏蔽不用通道，节省检测时间；
- ✧ 多个被测电阻有一端短路时(须开启共点模式)，不影响测试效果；
- ✧ 可通过寄存器设置，修正因测试接线引起的误差；
- ✧ 低检测激励电压(高阻<2.6V，低阻<2V),低检测激励电流(<1mA)；
- ✧ 检测端口防浪涌防过压保护，端口最大能承受 60V 电压；
- ✧ 采集输入(通道间不隔离)、电源、通讯三者相互隔离，可靠性高；
- ✧ 可以用 RS485、以太网、CAN 做为通信接口，当采用以太网或 CAN 时，可同时使用 RS485 接口，使模块同时拥有两个通讯口，可用于冗余高可靠场合；
- ✧ 可灵活自选 Modbus-RTU 或 Modbus-TCP 工业通信协议或 CAN 自定义协议，与各种组态屏、工控软件以及模组进行可靠通信。

二、产品主型号

ZH-T08NTC-14N1	**8 通道热电阻测温，RS485 接口；**
ZH-T08NTC-34N1	**8 通道热电阻测温，以太网接口+RS485 接口；**
ZH-T08NTC-74N1	**8 通道热电阻测温，CAN 接口+RS485 接口；**

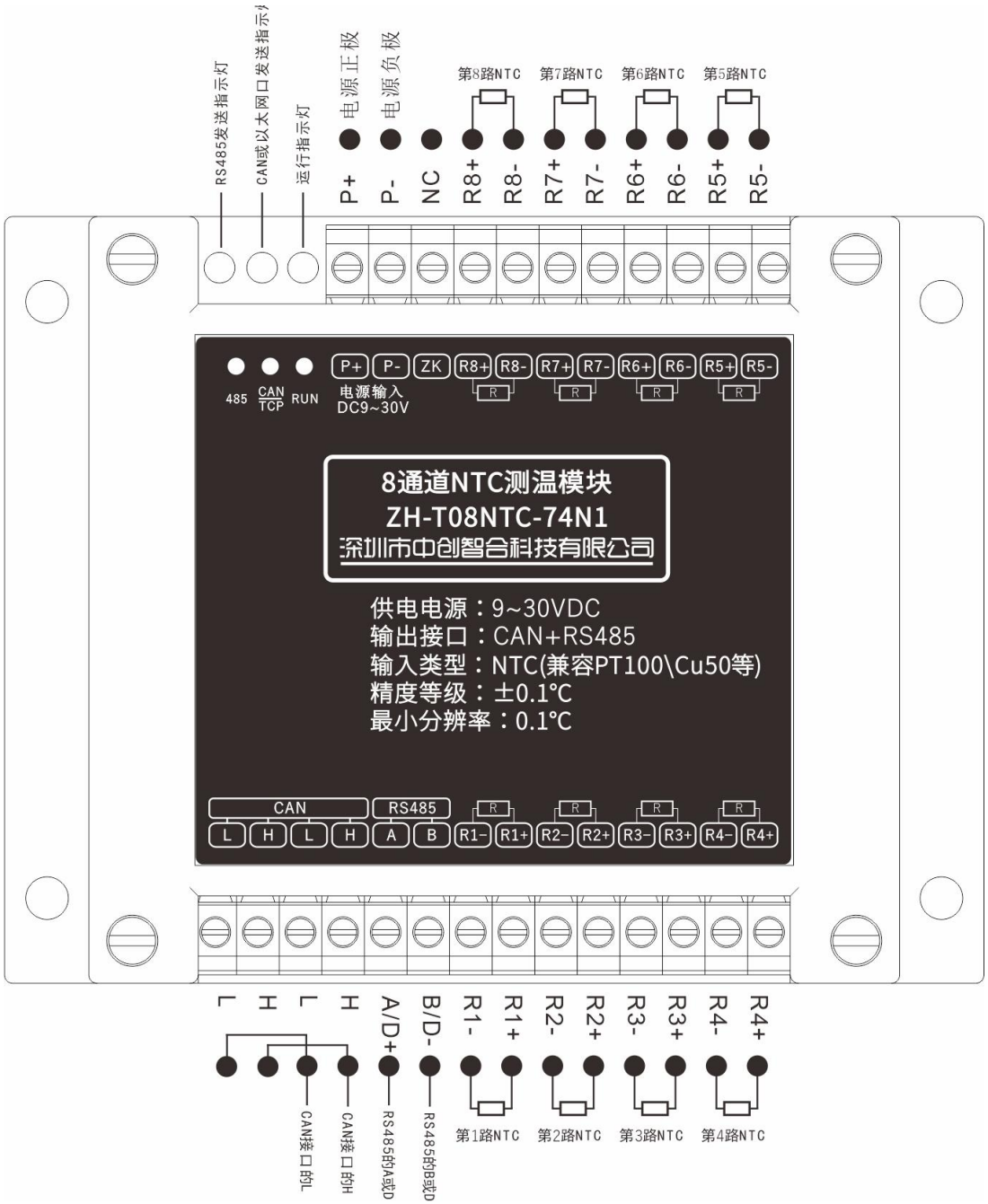
三、外形与尺寸图


8通道电阻测量模块
ZH-T08R-34N1
深圳市中创智合科技有限公司
供电电源：9~30VDC
输出接口：RS485+以太网口
测量范围：0.02Ω~40MΩ
精度等级：±0.1%
最小分辨率：1mΩ
以太网
RS485


95.2
9
115
105
Φ5
90
70
8通道NTC测温模块
ZH-T08NTC-14N1
深圳市中创智合科技有限公司
供电电源：9~30VDC
输出接口：RS485
输入类型：NTC(兼容PT100\Cu50等)
精度等级：±0.1℃
最小分辨率：0.1℃
RS485
40
35.5
4角有螺丝安装孔
导轨安装槽

四、端子接线定义图

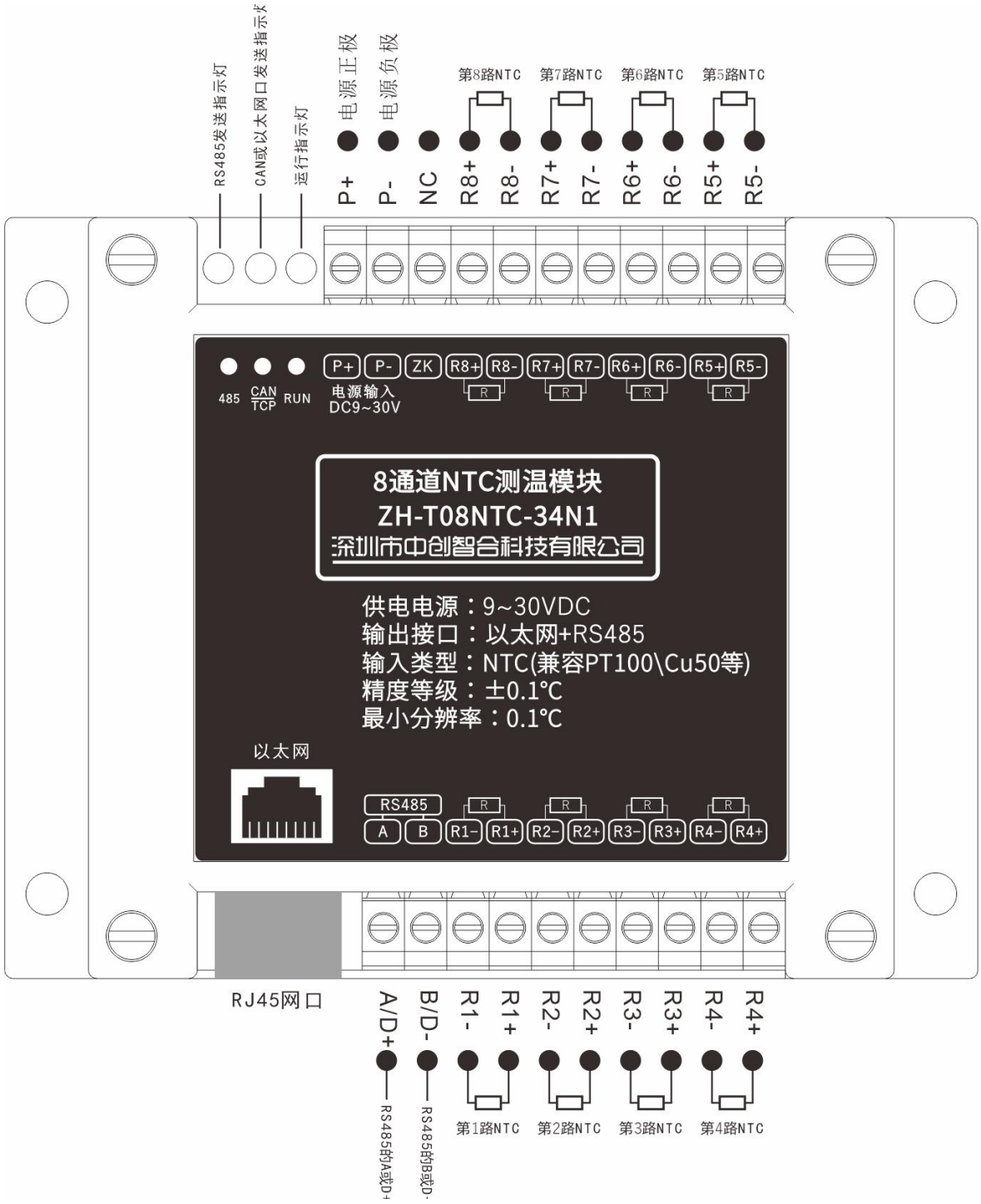
A. CAN 或 RS485 版本端子与接线图:



CAN 接口版本接线图

RS485 版本仅有 485 接口，CAN 接口无效

B. 以太网口版本端子与接线图：



以太网口版本接线图

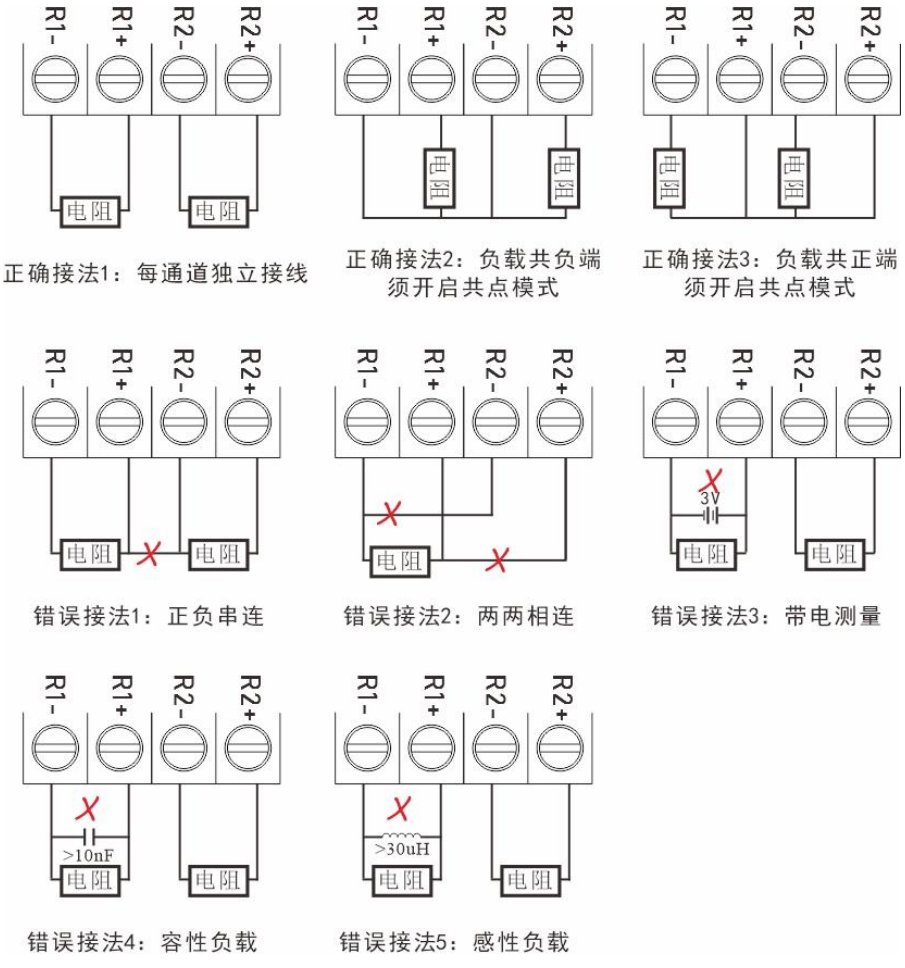
C. 端子定义说明表

表 1：端子定义说明表

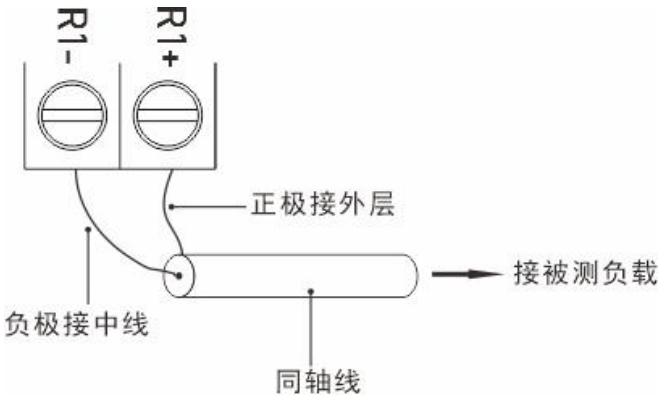
端子名称	端子用途
P+	电源正极
P-	电源负极
D+、A	RS485 端口数据线 A
D-、B	RS485 端口数据线 B
H	CAN 接口 H 端口
L	CAN 接口 L 端口
R1+、R2+.....R8+	电阻正端输入
R1-、R2-.....R8-	电阻负端输入
ZK	调零按键，与 P-短路进行调零
运行灯状态说明	RUN 灯：产品上电运行正常时会闪动；闪动频率与采样转换速率一样。 RS485 灯：本模块的 RS485 有数据发送时会闪烁 CAN/TCP 灯：本模块的 CAN 或以太网口有数据发送时会闪烁

D. NTC 电阻接线方式

请按以下图示正确接线与设置：



同轴线连接方法：



五、性能指标

- **检测激励电流：**

  检测时，检测电路会加载一定的激励电压与电流在被测负载上；

  激励电压为 2.6V~1.4V，阻值越大，激励电压越大；

  激励电流见表：

表 2：被测阻值与检测电压电流

阻值	激励电流	激励电压
<1KΩ	<1mA	<1.5V
>1KΩ	<0.8mA	<1.7V
>10KΩ	<0.2mA	<2V
>100KΩ	<30uA	<2.6V
>1MΩ	<3uA	<2.6V
>10MΩ	<0.3uA	<2.6V

- **温度测量范围与精度：**

  测试条件：8 通道同时接 2 米长 75-2 同轴线连接测试电阻；

  环境温度约 25℃；环境相对湿度约 50%。

表 3：精度与更新速率、量程关系表

0x0200 ~ 0x0207 寄存器值	热电阻类型	测温范围	测量误差		
			第 1 档 更新速率	第 2 档 更新速率	第 3 档与第 4 档 速率
9	PT100	-200℃~800℃	±0.1℃	±0.3℃	±2℃
10	PT1000	-200℃~800℃	±0.1℃	±0.3℃	±2℃
11	NTC10k B=3435	-30℃~100℃	±0.1℃	±0.2℃	±1℃①
14	Cu50	-50℃~150℃	±0.1℃	±0.3℃	±2℃
15	Cu100	-50℃~150℃	±0.1℃	±0.3℃	±2℃
16	NTC100k B=3950	-30℃~300℃	±0.1℃	±0.2℃	±1℃②
17	NTC10k B=3950	-40℃~300℃	±0.1℃	±0.2℃	±1℃①
18	NTC MF501 B=4100 5k	-40℃~80℃	±0.1℃	±0.2℃	±1℃①
30	用户自行填阻值表	自定	阻值精度 0.01%③	阻值精度 0.2%③	阻值精度 0.5%③

注①：当温度大于-20℃时才能使用此档速率;

注②：当温度大于 25℃时才能使用此档速率;

注③：需要自行换算温度精度;

此测量误差用标准电阻对照热电阻标称阻值表校对，实际热电阻阻值与标称值可能有误差，有可能会超过此表误差;

用户可以通过修改 0x2C0~0x02C7 来修正温度误差，这样精度可以达到更高；此值单位为 0.1℃，数值范围为-128 至+127。即可修正-12.8℃至+12.7℃。修改此寄存器，需先向 0x8000 寄存器写 0x000A 解锁。

算方法为：实际温度=测量温度+此寄存器补偿值。

修改举列如下：

假设拨码地址为 1 的模块测量到第 2 通道的温度值大于实际值 2.8℃，则需要向 0x2C1 寄存器写入 0xFFE4，即-28。

协议为 Modbus-RTU 时：

第一步：发送 01 06 80 00 00 0A 20 0D，此步只要发送一次，后面修改同类寄存器时，不再需要发送。

成功解锁时，模块会发回 01 06 80 00 00 0A 20 0D。

第二步：发送 01 06 02 C1 FF E4 98 35，修改成功，会发回 01 06 02 C1 FF E4 98 35。

协议为 Modbus-TCP 时：

第一步：发送 00 00 00 00 00 06 01 06 80 00 00 0A ，此步只要发送一次，后面修改同类寄存器时，不再需要发送。

成功解锁时，模块会发回 00 00 00 00 00 06 01 06 80 00 00 0A。

第二步：发送 00 00 00 00 00 06 01 06 02 C1 FF E4；

修改成功，会发回 00 00 00 00 00 06 01 06 02 C1 FF E4。

- **采集时间**

采集速率可通过 0x0081 寄存器设置，共 4 档（出厂默认为第 1 档）；采集时间与采集速率、阻值大小有关。

下表列出单通道在不同采集速率与 NTC 类型时的更新时间，8 个通道的采集时间需要在此基础上乘以 8。可以通过设置 0x0200~0x0207 寄存器，关闭通道，而减少整体的采集时间。

表 4：单通道采集时间表(0x0081 寄存器可更改采集速度)

测量方式	输入类型	每通道采样时间			
		第 1 档	第 2 档	第 3 档	第 4 档
自动分段精测	PT100	140ms	80ms	50ms	35ms
	PT1000	140ms	80ms	50ms	35ms
	NTC10k B=3435	140~420ms	80~240ms	--	--
	Cu50	140ms	80ms	50ms	35ms
	Cu100	140ms	80ms	50ms	35ms
	NTC100k B=3950	140~420ms	80~240ms	--	--
	NTC10k B=3950	140~420ms	80~240ms	--	--

	NTC MF501 B=4100 5k	140~420ms	80~240ms	--	--

◆ 如要检测速度快可以采取以下方法：

1. 通过设置 0x0200~0x0207 寄存器关闭不使用的通道；
2. 通过 0x0081 寄存器设置合适的采集速率；

◆ 0x0081 寄存器设置采集速率，其数值表示如下：

00--第 1 档，最慢档；01--第 2 档；02--第 3 档；03--第 4 档，最快档。

◆ 修改转换速率的方法举列

协议为 Modbus-RTU 时：

发送 01 06 00 81 00 01 18 22 转换速率调至第 2 档

发送 01 06 00 81 00 00 D9 E2 转换速率调至第 1 档

协议为 Modbus-TCP 时：

发送 00 00 00 00 00 06 01 06 00 81 00 01 转换速率调至第 2 档

发送 00 00 00 00 00 06 01 06 00 81 00 00 转换速率调至第 1 档

➢ **阻值与温度分辨率：**

此模块除了可以读取温度外，也可以读取采集到的探头阻值。温度分辨率为 0.1℃，阻值最小可至 1mΩ，多个种类的寄存器存储不同分辨率数据，用户可以按需读取，列表如下：

表 5：寄存器与分辨率关系表

寄存器地址（16 进制）	分辨率	数据位数	能指示的阻值范围	说明
0x0000~0x0008 或 0x2000~0x2008	0.1℃	16	-3276.7℃~ +3276.6℃	此寄存器存 16 位有符号的摄氏温度值； 如为 0x8000 时，表示传感器短路 如为 0x7FFF 时，表示传感器断开
0x2100~0x2108	0.1℉	16	-3276.7℉~ +3276.6℉	此寄存器存 16 位有符号的华氏温度值； 如为 0x8000 时，表示传感器短路 如为 0x7FFF 时，表示传感器断开
0x1200:0x1201~ 0x120E:0x120F	0.001Ω	32	0.00Ω~ 4.294967295MΩ	两个 16 位寄存器组成一个 32 位寄存器， 高 16 位在前，低 16 位在后； 显示 0xFFFFFFFF 时，表示超量程或断线
0x1240:0x1241~ 0x124E:0x124F	0.01Ω	32	0.00Ω~40MΩ	两个 16 位寄存器组成一个 32 位寄存器， 高 16 位在前，低 16 位在后； 显示 0xFFFFFFFF 时，表示超量程或断线
0x1280:0x1281~ 0x128E:0x128F	0.1Ω	32	0.00Ω~40MΩ	两个 16 位寄存器组成一个 32 位寄存器， 高 16 位在前，低 16 位在后； 显示 0xFFFFFFFF 时，表示超量程或断线
0x12C0:0x12C1~ 0x12CE:0x12CF	1Ω	32	0Ω~40MΩ	两个 16 位寄存器组成一个 32 位寄存器， 高 16 位在前，低 16 位在后； 显示 0xFFFFFFFF 时，表示超量程或断线
0x1000~0x1007	0.001Ω	16	0.000Ω~65.534Ω	显示 0xFFFF 时表示超显示**范围**
0x1040~0x1047	0.01Ω	16	0.00Ω~655.34Ω	
0x1080~0x1087	1Ω	16	0Ω~65534Ω	
0x10C0~0x10C7	0.1kΩ	16	0.0kΩ~6553.4kΩ	
0x1100~0x1107	1kΩ	16	0kΩ~40000kΩ	显示 0xFFFF 时表示超量程或断线

- **通讯反应时间：**

此时间是指收到指令，再开始返回数据的时间间隔，不包括指令传送过程的时间以及 TCP 网络延迟的时间。

表 6：通讯反应时间

波特率	反应时间间隔	备注
2400	25.0~35ms	上位机发送指令后，必须在最小反应时间内开始接收模块返回的数据，否则会出现丢包。比如：用 RS485 总线通讯时，在 115200 波特率下，主机发送指令后，必须在 1.0ms 内开启接收模式。
4800	12.0~15ms	
9600	4.5~10.0ms	
19200	3~5.0ms	
38400	2.5~3.5ms	
57600	1.5~2.5ms	
115200	1.0~2.0ms	

- **工作温度：**-40℃～+70℃；
- **温度漂移：**≤±50ppm/℃；
- **辅助电源：**+9V~+30VDC；
- **额定功耗：**<0.6W（无以太网口时）；
  <3W（有以太网口时）
- **安装方式：**标准 35mm 导轨或螺丝安装，螺丝安装尺寸 105*70mm，Φ5mm；
- **产品重量：**约 150g
- **输出接口：**RS485 接口或以太网+RS485 接口或 CAN+RS485 接口；
- **通讯协议：**内含 Modbus-RTU、Modbus-TCP 通讯协议、CAN 自定义协议，可自行配置；
- **通讯波特率：**4800、9600、19200、38400、57600、115200bps；
- **数据格式：** 8 个数据位，可自由配置无校验/奇校验/偶校验、1 位停止位/2 位停止位；
- **隔离耐压：**>1500V DC（电源与通讯电路、电源与检测电路、通讯与检测电路相互隔离）；

RS485 口出厂参数: 地址为 1 号,波特率 9600,无校验,8 个数据位，1 个停止位;默认 Modbus-RTU 协议

RJ45 网口出厂参数: TCP server 模式，IP:192.168.0.7,端口号:20108;默认 Modbus-TCP 协议
输入默认 IP 网页登录用户名:admin,登录密码:admin，可修改参数；
也可以用专用工具软件修改参数。

六、电阻量程、温度测量设置

可通过修改测量类型寄存器 0x0200~0x0207（分别对应 1 至 8 路测量通道的属性）来设置电阻测量量程或温度采集功能。

每一个通道都可以独立设置电阻量程或温度采集。

测量温度时，会自动调至 10M 量程并分段检测电阻，可通过读取电阻值寄存器正常读取电阻；通过读取温度寄存器 0x2000~0x2007 来取得测量温度。

目前此模块兼容的温度探头与电阻量程设置如下表：

表 7：量程与输入类型设置

寄存器地址（16 进制）	权限	数据位数	数值代表所接探头类型或量程（十进制）	说明
0x0200:0x0207	读/写	16	9：PT100 10：PT1000 11：NTC 10k　B=3435 12：PTC 14：Cu50 15：Cu100 16：NTC 100k　B3950 17：NTC 10k B3950 18：NTC MF501 B4100 5k 30：填 NTC 阻值表测温 200：电阻 0~25Ω量程 201：电阻 0~1k 量程 202：电阻 0~5k 量程 203：电阻 0~20k 量程 204：电阻 0~100k 量程 205：电阻 0~1M 量程 206：电阻 0~10M 量程 207：电阻 0~40M 量程 255：通道关闭	当设为温度时，阻值寄存器正常显示传感器阻值，通过读取温度寄存器 0x2000~0x2007 得到温度； 当设成 PTC 时，温度寄存器不是指示的温度，而是基数为 0.1Ω的电阻值 。 当设为 200 以上时，温度寄存器的数值无意义。 类型为 30 时，用户可自行填 NTC 阻值表测温，见第九节.

◆ 设置量程寄存器举列如下：

协议为 Modbus-RTU 时：

发送 01 06 02 00 00 C8 89 E4

把 0x0200 寄存器值设成 200(0x00C8)，即把第 1 通道设成 0~25Ω电阻量程

发送 01 06 02 01 00 C8 D8 24

把 0x0201 寄存器值设成 200(0x00C8)，即把第 2 通道设成 0~25Ω电阻量程

发送 01 06 02 02 00 09 E9 B4

把 0x0202 寄存器值设成 9(0x0009)，即把第 3 通道设成 PT100 温度采集

协议为 Modbus-TCP 时：

发送 00 00 00 00 00 06 01 06 02 00 00 C8

把 0x0200 寄存器值设成 200(0x00C8)，即把第 1 通道设成 0~25Ω电阻量程

发送 00 00 00 00 00 06 01 06 02 01 00 C8

把 0x0201 寄存器值设成 200(0x00C8)，即把第 2 通道设成 0~25Ω电阻量程

发送 00 00 00 00 00 06 01 06 02 02 00 09

把 0x0202 寄存器值设成 9(0x0009)，即把第 3 通道设成 PT100 温度采集

七、用户自行填 NTC 阻值表测量温度

用户可以自行填写 NTC 阻值表存入 FLASH，程序通过查表再计算出温度。相关寄存器列表如下：

表 8： 用户自行填 NTC 阻值表测温相关寄存器

寄存器地址（16 进制）	寄存器名称	数据位数	寄存器数量	权限	说明
0x300A	最小温度	16	1	读/写	NTC 测温最小温度，单位为 0.1℃ 如最小温度为-50℃时，则须填值为：0xFE0C,转成十进制为-500。 修改此值，须先向 0x8000 寄存器写 0x0A 解锁
0x300B	NTC 阻值表校验	16	1	读	用于校验用户填入的阻值表是否按从大至小排列，如正确，则读出的值为 0xA5A5，其它值则表示填入的表是错误的。 此值是动态的，用户如果分段填入阻值表，也会检测阻值表是否正确，如果用户没有填完表，则不用理会此值，只须在阻值表完全填完后查看就行。
0x300C:0x300D ~ 0x31FE:0x31FF	阻值表	32	250	读/写	每 2 个 16 位寄存器组成一个 32 位阻值寄存器，高 16 在前，低 16 位在后，共 250 组寄存器； 此值单位为 0.1Ω。 举例： 如需在第 1 个寄存器填入 100kΩ的阻值，则须填入 1000000，转成 16 进制为 0x000F4240；此时 0x300C 存入的值为 0x000F，0x300D 存入的值为 0x4240 修改此值，须先向 0x8000 寄存器写 0x0A 解锁

八、寄存器说明

A. 温度寄存器（支持用 03 和 04 功能码读，不能改写）

表 9：温度寄存器表

寄存器内容		寄存器地址（十六进制）	对应 PLC 或组态软件配置地址（十进制）	寄存器数据说明
16 位摄氏温度寄存器	第 1 通道	0x2000	48193	1. 此寄存器存 16 位有符号的摄氏温度值； 2. 此寄存器数值分辨率为 0.1℃； 3. 温度值=寄存器的数据 ÷ 10，比如：寄存器数据为 0x001F,则温度值为+3.1℃； 4. 如值为 0xFF1F,则温度值为-22.5℃。
	第 2 通道	0x2001	48194	
	第 3 通道	0x2002	48195	
	第 4 通道	0x2003	48196	
	第 5 通道	0x2004	48197	
	第 6 通道	0x2005	48198	
	第 7 通道	0x2006	48199	

	第 8 通道	0x2007	48200	
16 位 华氏温度 寄存器	第 1 通道	0x2100	48449	1. 此寄存器存 16 位有符号的华氏温度值（由摄氏温度寄存器的值换算而来）； 2. 此寄存器数值分辨率为 0.1℉； 3. 温度值=寄存器的数据 ÷10，比如：寄存器数据为 0x001F,则温度值为+3.1℉； 4. 如值为 0xFF1F,则温度值为-22.5℉。
	第 2 通道	0x2101	48450	
	第 3 通道	0x2102	48451	
	第 4 通道	0x2103	48452	
	第 5 通道	0x2104	48453	
	第 6 通道	0x2105	48454	
	第 7 通道	0x2106	48455	
	第 8 通道	0x2107	48456	

B. 电阻寄存器（支持用 03 和 04 功能码读，不能改写）

表 10： 阻值寄存器表

寄存器内容		寄存器地址（十六进制）	对应 PLC 或组态软件配置地址（十进制）	寄存器数据说明
×0.01Ω 32 位 阻值 寄存器	第 1 通道	0x0000:0x0001	40001:40002	1. 此寄存器由两个 16 位寄存器组合成一个 32 位寄存器，高字节在前，低字节在后。 2. 电阻数据分辨率为 0.01Ω，无符号数； 3. 电阻值=寄存器的数据 ÷100，比如：寄存器数据为 0x00010100,转成十进制为 65792，则阻值为 657.92Ω。 4. 当阻值超过 40MΩ时，则数据为 0xFFFFFFFF。
	第 2 通道	0x0002:0x0003	40003:40004	
	第 3 通道	0x0004:0x0005	40005:40006	
	第 4 通道	0x0006:0x0007	40007:40008	
	第 5 通道	0x0008:0x0009	40009:40010	
	第 6 通道	0x000A:0x000B	40011:40012	
	第 7 通道	0x000C:0x000D	40013:40014	
	第 8 通道	0x000E:0x000F	40015:40016	
×0.001Ω 32 位 阻值 寄存器	第 1 通道	0x1200:0x1201	44609:44610	1. 此寄存器由两个 16 位寄存器组合成一个 32 位寄存器，高字节在前，低字节在后。 2. 电阻数据分辨率为 0.001Ω，无符号数； 3. 电阻值=寄存器的数据 ÷1000，比如：寄存器数据为 0x00010100,转成十进制为 65792，则阻值为 65.792Ω。 4. 最大可指示 4.294967294MΩ，当数据为 0xFFFFFFFF，表示阻值超数据范围。
	第 2 通道	0x1202:0x1203	44611:44612	
	第 3 通道	0x1204:0x1205	44613:44614	
	第 4 通道	0x1206:0x1207	44615:44616	
	第 5 通道	0x1208:0x1209	44617:44618	
	第 6 通道	0x120A:0x120B	44619:44620	
	第 7 通道	0x120C:0x120D	44621:44622	
	第 8 通道	0x120E:0x120F	44623:44624	
×0.01Ω 32 位 阻值 寄存器	第 1 通道	0x1240:0x1241	44673:44674	此组寄存器同 0x0000:0x0001~0x000E:0X000F 寄存器。
	第 2 通道	0x1242:0x1243	44675:44676	
	第 3 通道	0x1244:0x1245	44677:44678	
	第 4 通道	0x1246:0x1247	44679:44680	
	第 5 通道	0x1248:0x1249	44681:44682	
	第 6 通道	0x124A:0x124B	44683:44684	
	第 7 通道	0x124C:0x124D	44685:44686	
	第 8 通道	0x124E:0x124F	44687:44688	

×0.1Ω 32 位 阻值 寄存器	第 1 通道	0x1280:0x1281	44737:44738	1. 此寄存器由两个 16 位寄存器组合成一个 32 位寄存器，高字节在前，低字节在后。 2. 电阻数据分辨率为 0.1Ω，无符号数； 3. 电阻值=寄存器的数据 ÷10，比如：寄存器数据为 0x00010100,转成十进制为 65792，则阻值为 6579.2Ω。 4. 当阻值超过 40MΩ时，则数据为 0xFFFFFFFF
	第 2 通道	0x1282:0x1283	44739:44740	
	第 3 通道	0x1284:0x1285	44741:44742	
	第 4 通道	0x1286:0x1287	44743:44744	
	第 5 通道	0x1288:0x1289	44745:44746	
	第 6 通道	0x128A:0x128B	44747:44748	
	第 7 通道	0x128C:0x128D	44749:44750	
	第 8 通道	0x128E:0x128F	44751:44752	
×1Ω 32 位 阻值 寄存器	第 1 通道	0x12C0:0x12C1	44801:44802	1. 此寄存器由两个 16 位寄存器组合成一个 32 位寄存器，高字节在前，低字节在后。 2. 电阻数据分辨率为 1Ω，无符号数； 3. 电阻值=寄存器的数据，比如：寄存器数据为 0x00010100,转成十进制为 65792，则阻值为 65792Ω。 4. 当阻值超过 40MΩ时，则数据为 0xFFFFFFFF
	第 2 通道	0x12C2:0x12C3	44803:44804	
	第 3 通道	0x12C4:0x12C5	44805:44806	
	第 4 通道	0x12C6:0x12C7	44807:44808	
	第 5 通道	0x12C8:0x12C9	44809:44810	
	第 6 通道	0x12CA:0x12CB	44811:44812	
	第 7 通道	0x12CC:0x12CD	44813:44814	
	第 8 通道	0x12CE:0x12CF	44815:44816	
×0.001Ω 阻值 寄存器	第 1 通道	0x1000	44097	1. 此寄存器存 16 位 0.001Ω级电阻值； 2. 电阻数据分辨率为 0.001Ω 3. 电阻值=寄存器的数据 ÷ 1000，比如：寄存器数据为 0x0100,则阻值为 0.256Ω。 4. 当数值为 0xFFFF 时(十进制 65535)，表示电阻超过了 65.534Ω。
	第 2 通道	0x1001	44098	
	第 3 通道	0x1002	44099	
	第 4 通道	0x1003	44100	
	第 5 通道	0x1004	44101	
	第 6 通道	0x1005	44102	
	第 7 通道	0x1006	44103	
	第 8 通道	0x1007	44104	
×0.01Ω 阻值 寄存器	第 1 通道	0x1040	44161	1. 此寄存器存 16 位 0.01Ω级电阻值； 2. 电阻数据分辨率为 0.01Ω 3. 电阻值=寄存器的数据 ÷ 100，比如：寄存器数据为 0x0100, 则阻值为 2. 56Ω。 4. 当数值为 0xFFFF 时(十进制 65535)，表示电阻超过了 655.34Ω。
	第 2 通道	0x1041	44162	
	第 3 通道	0x1042	44163	
	第 4 通道	0x1043	44164	
	第 5 通道	0x1044	44165	
	第 6 通道	0x1045	44166	
	第 7 通道	0x1046	44167	
	第 8 通道	0x1047	44168	
×1Ω 阻值 寄存器	第 1 通道	0x1080	44225	1. 此寄存器存 16 位 1Ω级电阻值； 2. 电阻数据分辨率为 1Ω 3. 电阻值=寄存器的数据 ，比如：寄存器数据为 0x0100, 则阻值为 256Ω。 4. 当数值为 0xFFFF 时(十进制 65535)，表示电阻超过了 65534Ω。
	第 2 通道	0x1081	44226	
	第 3 通道	0x1082	44227	
	第 4 通道	0x1083	44228	
	第 5 通道	0x1084	44229	
	第 6 通道	0x1085	44230	
	第 7 通道	0x1086	44231	
	第 8 通道	0x1087	44232	

×0.1kΩ 阻值 寄存器	第 1 通道	0x10C0	44289	1. 此寄存器存 16 位 0.1kΩ级电阻值； 2. 电阻数据分辨率为 0.1kΩ 3. 电阻值=寄存器的数据÷ 10 ，比如：寄存器数据为 0x0100, 则阻值为 25. 6kΩ。 4. 当数值为 0xFFFF 时(十进制 65535)，表示电阻超过了 6553.4kΩ。
	第 2 通道	0x10C1	44290	
	第 3 通道	0x10C2	44291	
	第 4 通道	0x10C3	44292	
	第 5 通道	0x10C4	44293	
	第 6 通道	0x10C5	44294	
	第 7 通道	0x10C6	44295	
	第 8 通道	0x10C7	44296	
×1kΩ 阻值 寄存器	第 1 通道	0x1100	44353	1. 此寄存器存 16 位 kΩ级电阻值； 2. 电阻数据分辨率为 1kΩ 3. 电阻值=寄存器的数据，比如：寄存器数据为 0x0100, 则阻值为 256kΩ。 4. 当数值为 0xFFFF 时(十进制 65535)，表示电阻超过了 40000kΩ。
	第 2 通道	0x1101	44354	
	第 3 通道	0x1102	44355	
	第 4 通道	0x1103	44356	
	第 5 通道	0x1104	44357	
	第 6 通道	0x1105	44358	
	第 7 通道	0x1106	44359	
	第 8 通道	0x1107	44360	

C. 配置字寄存器

此类寄存器只能用 03 功能码读或 06 与 16 功能码写，见表如下：

表 11：设置寄存器表

寄存器地址 (Hex)	保持寄存器内容	寄存器位数	寄存器个数	寄存器状态	数据范围
0x0050	地址	16	1	读/写	地址(此值可填 1-253，254 与 255 为广播地址)(默认 01)
0x0051	RS485 口 波特率	16	1	读/写	0000 设置波特率-115200bps 0001 设置波特率-9600bps(默认) 0002 设置波特率-19200bps 0003 设置波特率-38000bps 0004 设置波特率-2400bps 0005 设置波特率-4800bps 0006 设置波特率-9600bps 0007 设置波特率-19200bps 0008 设置波特率-38400bps 0009 设置波特率-57600bps 000A 设置波特率-115200bps

0x0052	RS485 口 寄偶校验	16	1	读/写	0000 无校验，1 个停止位(默认) 0001 寄校验，1 个停止位 0002 偶校验，1 个停止位 0003 无校验，2 个停止位 0004 寄校验，2 个停止位 0005 偶校验，2 个停止位
0x0055	模块名称--高	16	1	读/写	默认:5A54H（ZT 的 ASCII 码）
0x0056	模块名称--中	16	1	读/写	默认:3038H（08 的 ASCII 码）
0x0057	模块名称--低	16	1	读/写	默认:5250H（R0 的 ASCII 码）
0x0058	软件版本	16	1	读	0616：6.16 版本
0x0059	软件子版本	16	1	读	2405：程序版本日期 24 年 5 月
0x005A	网口与 MCU 通讯 速率	16	1	读/写	同 0x0051
0x005B	网口与 MCU 通讯 寄偶校验	16	1	读/写	同 0x0052
0x0081	采样速率	16	1	读/写	0-第一档，最慢速率（出厂默认） 1-第二档 2-第三档 3-第四档 刷新时间越快，精度越低。
0x0082	电网配置	16	1	读/写	其值为： 50-适用频率为 50HZ 的电网（出厂默认） 60--适用频率为 60HZ 的电网
0x0083	校正标志	16	1	读/写	其值为 0x5AF0 时，表示出厂已校正
0x0085	自动分段精测使能	16	1	读/写	自动分段精测使能： 0-自动分段精测（出厂默认） 1-关闭自动分段，精度降低，检测速度加快。
0x0089	共点测量使能	16	1	读/写	多个电阻一端接同一点时，必须使能此寄存器： 0-关闭共点功能，加强抗干扰（出厂默认值） 1-开启共点功能。
0x01F9	主动上传时间间隔 设定	16	1	读/写	主动上传时间间隔=寄存器值 x0.5ms 出厂默认值=200，即 100ms
0x01FA	通讯协议定义	16	1	读/写	详见附件 《如何切换 Modbus-RTU 与 Modbus-TCP 协议》
0x01FB	主动上传控制	16	1	读/写	Bit4: 控制 RS485 口主动上传功能开启或关闭 Bibt5:控制以太网口主动上传功能开启或关闭 当相应位为 1 时，主动上传开启，当相应位为 0 时，主动上传关闭。 出厂默认为关闭主动上传 主动上传格式请参照第十四节

0x01FC	RS485 接口主动上传类型选择	16	1	读/写	Bit5:温度上传使能，此位为 1 时，会主动上传 1 至 8 路温度； Bit6:二极管电压上传使能，此位为 1 时，会主动上传二极管电压； Bit11~Bit8：阻值上传使能（对应寄存器请参见第十四节）
0x01FD	以太网接口主动上传类型选择	16	1	读/写	定义同 0x01FC 寄存器
0x0200~0x0207	测量类型与量程	16	8	读/写	其值表示测量类型或量程： 9：PT100 10：PT1000 11：NTC 10k B=3435 12：PTC 14：Cu50 15：Cu100 16：NTC 100k B3950 17：NTC 10k B3950（出厂默认值） 18：NTC MF501 B4100 5k 30：用户填 NTC 阻值表测温 200：电阻 0~25Ω量程 201：电阻 0~1k 量程 202：电阻 0~5k 量程 203：电阻 0~20k 量程 204：电阻 0~100k 量程 205：电阻 0~1M 量程 206：电阻 0~10M 量程 207：电阻 0~40M 量程 255：通道关闭
0x02C0~0x02C7	温度误差补偿	16	8	读/写	对应 1 至 8 通道的测温误差补偿，用户可填入正负数值修正温度误差。 此值单位为 0.1℃，数值范围为-128 至+127。即可修正-12.8℃至+12.7℃。 实际温度=测量温度+此补偿值 出厂默认值=0； 如要改写此值，必须向 8000H 寄存器写 0x0A，改写完后，向 8000H 寄存器写 0x05 或重新上电加锁保护。

0x02E0~ 0x02E7	线阻补偿	16	8	读/写	对应 1 至 8 通道电阻补偿，用户可填入正负数值修正电阻误差。 此值单位为 0.001Ω.范围为-32768~+32767（负数为补码）,即可以修正最大误差为 -32.768Ω~+32.767Ω 实际电阻=测量电阻+此补偿值 出厂默认值=0； 如要改写此值，必须向 8000H 寄存器写 0x0A，改写完后，向 8000H 寄存器写 0x05 或重新上电加锁保护。
0x300A	最小温度	16	1	读/写	NTC 测温最小温度，单位为 0.1℃ 如最小温度为-50℃时，则须填值为：0xFE0C,转成十进制为-500。 修改此值，须先向 0x8000 寄存器写 0x0A 解锁
0x300B	NTC 阻值表校验	16	1	读	用于校验用户填入的阻值表是否按从大至小排列，如正确，则读出的值为 0xA5A5，其它值则表示填入的表是错误的。 此值是动态的，用户如果分段填入阻值表，也会检测阻值表是否正确，如果用户没有填完表，则不用理会此值，只须在阻值表完全填完后查看就行。
0x300C: 0x300D ~ 0x31FE: 0x31FF	阻值表	32	250	读/写	每 2 个 16 位寄存器组成一个 32 位阻值寄存器，高 16 在前，低 16 位在后，共 250 组寄存器；此值单位为 0.1Ω。 举例： 如需在第 1 个寄存器填入 100kΩ的阻值，则须填入 1000000，转成 16 进制为 0x000F4240；此时 0x300C 存入的值为 0x000F，0x300D 存入的值为 0x4240 修改此值，须先向 0x8000 寄存器写 0x0A 解锁
0x04E0~ 0x04E7	可控制本模块的 CAN 上位机 ID	11	8	读/写	可控制本模块的 CAN 上位机 ID 前 11 位。出厂默认为：0x7FF 本模块只会响应此 8 个寄存器设置的上位机 ID，其它直接忽略。
0x04E8	CAN 上传数据的本模块 ID 前 3 位	3	1	读/写	CAN 主动上传或响应上位机指令时，由此寄存器值与模块的 8 位拨码地址组成 CAN 的前 11 位 ID。 此值放高 3 位，拨码地址放低 8 位。 出厂默认为 0x07，修改此值可调整主动上传权限
0x04EA	CAN 波特率	16	1	读/写	定义了 CAN 的波特率，其数值含义为： 0：5kbps 1：10kpbs 2：20kpbs 3：40kpbs 4：50kpbs

					5：80kpbs 6：100kpbs 7：125kpbs 8：200kpbs 9：250kpbs 10：400kpbs 11：500kpbs（出厂默认值） 12：800kpbs 13：1000kpbs
0x04EB	CAN 采样率	16	1	保留	目前暂不支持修改 CAN 采样率，各波特率下固定采样率如下： 5kbps：85% 10kpbs：85% 20kpbs：85% 40kpbs：85% 50kpbs：85% 80kpbs：85% 100kpbs：85% 125kpbs：81.25% 200kpbs：85% 250kpbs：81.25% 400kpbs：85% 500kpbs：87.5% 800kpbs：86.66% 1000kpbs：83.33%
0x04F1	CAN 主动上传间隔时间	16	1	读/写	上传间隔时间=寄存器值×0.5ms 出厂默认值=200，即 100ms
0x04F2	各采集类型 CAN 主动上传使能	16	1	读/写	温度、电阻、电压等采集量的 CAN 主动上传使能控制。 Bit4：为 1 时开启 8 通道温度上传 为 0 关闭温度上传。 Bit5：为 1 时开启 8 通道电阻上传； 为 0 关闭电阻上传。 Bit6：为 1 时开启 8 通道二极管电压上传； 为 0 关闭上传。 此寄存器出厂默认值为：0x0010
0x04F3	CAN 主动上传电阻类型定义	16	1	读/写	其值定义了上传何种阻值类型，按 16 进制列表： 0x0004--主动上传 8 通道“16 位 x0.001Ω阻值寄存器”值（0x1000~0x1007 寄存器值） 0x0005--主动上传 8 通道“16 位 x0.01Ω阻值寄存器”值（0x1040~0x1047 寄存器值） 0x0007--主动上传 8 通道“16 位 x1Ω阻值寄存器”值（0x1080~0x1087 寄存器值） 0x0008--主动上传 8 通道“16 位 x0.1KΩ阻值寄存器”值（0x10C0~0x10C7 寄存器值） 0x0009 主动上传 8 通道“16 位 x1KΩ阻值寄存

					器”值（0x1100~0x1107 寄存器值） 0x0013--主动上传 8 通道“32 位 x0.001Ω阻值寄存器”值（0x1200:1201~0x120E:120F 寄存器值） 0x0014--主动上传 8 通道“32 位 x0.01Ω阻值寄存器”值（0x1240:1241~0x124E:124F 寄存器值），出厂默认值 0x0015--主动上传 8 通道“32 位 x0.1Ω阻值寄存器”值（0x1280:1281~0x128E:128F 寄存器值） 0x0016--主动上传 8 通道“32 位 x1Ω阻值寄存器”值（0x12C0:12C1~0x12CE:12CF 寄存器值）
0x7240	转换控制	16	1	读/写	向此寄存器写 0x5A，模块会停止转换电阻值，并把所有阻值寄存器复位； 向此寄存器写除 0x5A 以外的任何值，模块重新转换。
0x8000	寄存器写保护	16	1	写	向此寄存器写 0x0A 才能改写误差补偿寄存器；改写完后，写 0x05 锁存，并生效误差补偿

九、一键复位功能

一键复位功能能在设置出错时，一键复位至出厂状态，步骤如下：

a. 打开外壳，按下 PCB 上的 SET 键不松开；重新上电或按一下复位键；
b. 此时保持 SET 键不松开，RUN 指示会先亮 1 秒，然后熄灭 2 秒；
c. 当出现 RUN 指示灯慢闪时，如果松开 SET 键，则复位通讯设置；
d. 如果想复位其它设置，则在出现 RUN 灯慢闪后，一直按住 SET 键不松开（约 30 秒），直到 RUN 熄灭，此时会复位所有设置，包括：通讯、采样速率、量程、电网设置等等，但不会复位校正参数。

十、一键调零

当测试连接线比较长时，可以通过一键调零功能补偿连接线电阻。调零方法如下：

a. 把测试线接入模块测试端子，并把测试线连接负载的末端短路；
b. 把模块上电，此时 RUN 灯为闪烁，表示正在进行正常测试模式；
c. 把“ZK”端与“P-”短接，等待 3 秒，此时 RUN 灯常亮，表示调零完成；
d. 只有采集到接入的阻值小于 25Ω的通道才会调零，大于 25Ω的通道不会自动调零；所以可以把无需调零的通道断开，这样此通道不受调零影响。
e. 被调零的通道测出的线阻取反后会自动填入 0x2E0~0x2E7 线阻补偿寄存器，用户可以手动修改此寄存器进行后期修正。
f. 0x2E0~0x2E7 线阻补偿寄存器的数值范围为-32768mΩ至+32767mΩ，模块测出的阻值与

线阻补偿寄存器阻值叠加后再存入阻值寄存器，所以要注意线阻补偿数值的正负。

十一、共点测试功能

当多个被测负载一端相连时，需要开启共点测试功能才能正确测试；默认情况下此功能关闭，以加强抗干扰。如有条件采用共正极方式，则尽量采用共正极方式，这样可把通道间相互干扰减少到最低。

向 0x0089 寄存器写 1，就可以开启此功能。修改方法举如下：

协议为 Modbus-RTU 时：

发送 01 06 00 89 00 01 99 E0　开启共点测试

发送 01 06 00 89 00 00 58 20　关闭共点测试

协议为 Modbus-TCP 时：

发送 00 00 00 00 00 06 01 06 00 89 00 01　开启共点测试

发送 00 00 00 00 00 06 01 06 00 89 00 00　关闭共点测试

十二、主动上传

1. 主动上传格式

此模块可以开启主动上传功能，开启后，通信口就会按主动上传时间间隔寄存器（0x01F9）设定的时间间隔不断上传 8 个通道电阻值或温度。

主动上传出厂默认设置为上传关闭，上传类型为“16 位摄式温度寄存器”数据，上传间隔时间为 100ms。上传格式按 03 功能返回码格式，Modbus-RTU 协议时，如下表：

表 12：Modbus-RTU 协议主动上传

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
2	03	功能码	1
3	20 或 10	返回的数据字节个数，32 位数据时为 32 个字节，16 位数据时为 16 个字节。	1
4	01 37 03 05 00 00 ...	依次上传 8 个通道的数据，第 1 个寄存器数据在前，第 8 个寄存器数据在最后；当数据为 16 位时，每 2 个字节表示一个寄存器数据，高 8 位在前，低 8 位在后；当数据为 32 位时，每 4 个字节表示一个寄存器数据，最高 8 位在最前，最低 8 位在最后；	32 或 16
5	CRC	CRC 校验码（低位在前，高位在后）	2

Modbus-TCP 协议时，如下表：

表 13：Modbus-TCP 协议主动上传

排列顺序	数据字符（16 进制）	数据说明	字节数
1	00 00	通信事务处理标识符，每次上传会滚动加 1，	2
2	00 00	表示协议标识符，固定为 00 00	2
3	00 23 或 00 13	后面数据包的长度，当为 32 位数据时，为 0x0023；当为 16 位数据时，为 0x0013	2
4	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1

5	03	功能码	1
6	20 或 10	返回的数据字节个数，32 位数据时为 32 个字节，16 位数据时为 16 个字节。	1
7	01 37 03 05 00 00 ...	依次上传 8 个通道的数据，第 1 个寄存器数据在前，第 8 个寄存器数据在最后；当数据为 16 位时，每 2 个字节表示一个寄存器数据，高 8 位在前，低 8 位在后；当数据为 32 位时，每 4 个字节表示一个寄存器数据，最高 8 位在最前，最低 8 位在最后；	32 或 16

2. 主动上传相关寄存器设置。

寄存器名称：主动上传间隔时间寄存器
寄存器地址：0x01F9
寄存器位长：16(WORD)
出厂默认值：200
寄存器功能：主动上传间隔时间设置。

位序	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
功能	UpTime															
权限	R/W															
UpTime 取值范围为：8~65535； 主动上传间隔时间=UpTime×0.5ms																

寄存器名称：主动上传使能寄存器
寄存器地址：0x01FB
寄存器位长：16(WORD)
出厂默认值：0x0000
寄存器功能：使能或除能主动上传功能。

位序	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
功能	保留		保留		保留		保留		保留		U2up	U1up	保留	保留	保留	保留
权限	R/W		R/W		R/W		R/W		R/W		R/W	R/W	R/W	R/W	R/W	R/W
Bit4-- U1up 通讯口 1 主动上传使能 U1up=0 通讯口 1(RS485)主动上传关闭 U1up=1 通讯口 1(RS485)主动上传开启 Bit5-- U2up 通讯口 2 主动上传使能 U2up=0 通讯口 2(以太网)主动上传关闭 U2up=1 通讯口 2(以太网)主动上传开启																

寄存器名称：RS485 接口主动上传类型选择
寄存器地址：0x01FC
寄存器位长：16(WORD)
出厂默认值：0x0020
寄存器功能：主动上传类型选择。

位序	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
功能	保留		保留		UpR				保留	UpVd	UpTp	保留	保留	保留	保留	保留
权限	R/W		R/W		R/W				R/W		R/W	R/W	R/W	R/W	R/W	R/W

Bit5-- UpTp 主动上传温度 ，出厂默认值

UpTp=0 关闭温度主动上传

UpTp=1 主动上传 8 通道摄氏温度值（0x2000~0x2008 寄存器值）

Bit6-- UpVd 主动上传二极管电压

UpVd=0 关闭二极管电压上传

UpVd=1 主动上传采集到的 8 通道二极管电压（0x2300~0x2308 寄存器值）

Bit11:Bit08-- UpR 主动上传电阻值

UpR=0 关闭电阻值主动上传

UpR=1 主动上传 8 通道“32 位 x0.001Ω阻值寄存器”值（0x1200:1201~0x120E:120F 寄存器值）

UpR=2 主动上传 8 通道“32 位 x0.01Ω阻值寄存器”值（0x1240:1241~0x124E:124F 寄存器值）

UpR=3 主动上传 8 通道“32 位 x0.1Ω阻值寄存器”值（0x1280:1281~0x128E:128F 寄存器值）

UpR=4 主动上传 8 通道“32 位 x1Ω阻值寄存器”值（0x12C0:12C1~0x12CE:12CF 寄存器值）

UpR=5 主动上传 8 通道“16 位 x0.001Ω阻值寄存器”值（0x1000~0x1007 寄存器值）

UpR=6 主动上传 8 通道“16 位 x0.01Ω阻值寄存器”值（0x1040~0x1047 寄存器值）

UpR=7 主动上传 8 通道“16 位 x1Ω阻值寄存器”值（0x1080~0x1087 寄存器值）

UpR=8 主动上传 8 通道“16 位 x0.1KΩ阻值寄存器”值（0x10C0~0x10C7 寄存器值）

UpR=9 主动上传 8 通道“16 位 x1KΩ阻值寄存器”值（0x1100~0x1107 寄存器值）

主动上传可以同时开启不同种类的数据上传，比如，同时开启温度与电阻上传；但不能同时开启同一种类的不同数据格式上传，比如电阻只能选择一种寄存器上传。

寄存器名称：以太网口主动上传类型选择

寄存器地址：0x01FD

寄存器位长：16(WORD)

出厂默认值：0x0000

寄存器功能：主动上传类型选择。

位序	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
功能	保留		保留		UpR				保留	UpVd	UpTp	保留	保留	保留	保留	保留
权限	R/W		R/W		R/W				R/W		R/W	R/W	R/W	R/W	R/W	R/W

Bit5-- UpTp 主动上传温度,出厂默认值

UpTp=0 关闭温度主动上传

UpTp=1 主动上传 8 通道摄氏温度值（0x2000~0x2008 寄存器值）

Bit6-- UpVd 主动上传二极管电压

UpVd=0 关闭二极管电压上传

UpVd=1 主动上传采集到的 8 通道二极管电压（0x2300~0x2308 寄存器值）

Bit11:Bit08-- UpR 主动上传电阻值

UpR=0 关闭电阻值主动上传

UpR=1 主动上传 8 通道“32 位 x0.001Ω阻值寄存器”值（0x1200:1201~0x120E:120F 寄存器值）

UpR=2 主动上传 8 通道“32 位 x0.01Ω阻值寄存器”值（0x1240:1241~0x124E:124F 寄存器值）

UpR=3 主动上传 8 通道“32 位 x0.1Ω阻值寄存器”值（0x1280:1281~0x128E:128F 寄存器值）

UpR=4 主动上传 8 通道“32 位 x1Ω阻值寄存器”值（0x12C0:12C1~0x12CE:12CF 寄存器值）

UpR=5 主动上传 8 通道“16 位 x0.001Ω阻值寄存器”值（0x1000~0x1007 寄存器值）

UpR=6 主动上传 8 通道“16 位 x0.01Ω阻值寄存器”值（0x1040~0x1047 寄存器值）

UpR=7 主动上传 8 通道“16 位 x1Ω阻值寄存器”值（0x1080~0x1087 寄存器值）

UpR=8 主动上传 8 通道“16 位 x0.1KΩ阻值寄存器”值（0x10C0~0x10C7 寄存器值）

UpR=9　主动上传 8 通道“16 位 x1KΩ阻值寄存器”值（0x1100~0x1107 寄存器值）

主动上传可以同时开启不同种类的数据上传，比如，同时开启温度与电阻上传；但不能同时开启同一种类的不同数据格式上传，比如电阻只能选择一种寄存器上传。

3. 开启与关闭主动上传举列：

协议为 Modbus-RTU 时：

发送 01 06 01 FB 00 10 F8 0B　开启 RS485 接口主动上传，关闭网口主动上传

发送 01 06 01 FB 00 20 F8 1F　开启网口主动上传 ，关闭 RS485 接口主动上传

发送 01 06 01 FB 00 00 F9 C7　关闭 RS485 接口与网口主动上传

协议为 Modbus-TCP 时：

发送 00 00 00 00 00 06 01 06 01 FB 00 10　开启 RS485 接口主动上传，关闭网口主动上传

发送 00 00 00 00 00 06 01 06 01 FB 00 20　开启网口主动上传 ，关闭 RS485 接口主动上传

发送 00 00 00 00 00 06 01 06 01 FB 00 00　关闭 RS485 接口与网口主动上传

版本： V6.0　2023.10.26 进行重大升级，加强了端口保护，改进了高阻检测性能。

V6.5　2024.03.13 更新主动上传功能与 CAN

附件 1

Modbus-RTU 通讯协议

Modbus-RTU 通讯协议举例

如下所有命令都是以硬件地址为 01 来举例说明；

1. 读模块配置字寄存器命令（03 功能码）

主设备发送报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
2	03	功能码	1
3	00 55	数据起始寄存器地址，高 8 位在前，低 8 位在后；参照“配置字寄存器表”	2
4	00 02	读取寄存器个数，高 8 位在前，低 8 位在后（此列读取 2 个寄存器数据）	2
5	D4 1B	CRC 校验码（低位在前，高位在后）	2

从设备返回正确报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
2	03	功能码	1
3	04	返回的数据字节个数，2 个寄存器*2	1
4	35 39 30 39	读取的寄存器数据，每 2 个字节表示一个寄存器数据，高位在前，低位在后；第 1 个寄存器数据在前	可变
5	F1 E0	CRC 校验码（低位在前，高位在后）	2

2. 读 8 路电阻值或温度、电压（以温度寄存器 0x2000~0x2007 为列）（支持 04 与 03 功能码,字节读）

主设备发送报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
2	03	功能码	1
3	20 00	起始通道序号，高 8 位在前，低 8 位在后；	2
4	00 08	读取 8 个通道温度值，高 8 位在前，低 8 位在后	2
5	4F CC	CRC 校验码（低位在前，高位在后）	2

从设备返回正确报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1

2	03	功能码	1
3	10	返回的数据字节个数，8 个寄存器*2	1
4	01 37 03 05 00 00 ...	读取的寄存器数据，每 2 个字节表示一个寄存器数据，高位在前，低位在后；第 1 个寄存器数据在前，数据还原参照 6.1	可变
5	CRC	CRC 校验码（低位在前，高位在后）	2

3. 配置寄存器修改命令：

3.1 单个寄存器修改命令（06 功能码，每次只能修改一个寄存器，举例修改通讯地址）

主设备发送报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
2	06	功能码	1
3	00 50	寄存器地址，高 8 位在前，低 8 位在后，参照“配置字寄存器表”	2
4	00 02	寄存器数据，参照“配置字寄存器表”	2
5	08 1A	CRC 校验码（低位在前，高位在后）	2

从设备返回正确报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
2	06	功能码	1
3	00 50	寄存器地址，返回相同	2
4	00 02	寄存器数据，返回相同	2
5	08 1A	CRC 校验码，返回相同	2

3.2 连续修改多个寄存器命令（16 功能码，举例修改各通道补偿值）

主设备发送报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
2	10	功能码	1
3	04 40	起始寄存器，高 8 位在前，低 8 位在后 参照“配置字寄存器表”	2
4	00 04	写入寄存器长度，高 8 位在前，低 8 位在后 （此列写入 4 个寄存器）	2
5	08	写入字节长度（写入寄存器长度 x2）	1
6	00 00 00 01 00 03 00 06	写入的数据,每 2 个字节表示一个寄存器数据，高位在前，低位在后；此列表示把 0440H 寄存器写入数据 0000H,0441H 寄存器写入数据 0001H，0442H 寄存器写入数据 0003H，0443H 寄存器写入数据 0006H	按序列 8 表示的字节数
7	F4 03	CRC 校验码（低位在前，高位在后）	2

从设备返回正确报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	01	从设备地址，与主设备发送报文保持一致	1
2	10	功能码	1
3	04 40	起始寄存器，高 8 位在前，低 8 位在后 与主设备发送的报文相同	2
4	00 04	写入寄存器长度，高 8 位在前，低 8 位在后 与主设备发送的报文相同	2
5	C1 2E	CRC 校验码（低位在前，高位在后）	2

附件 2：

Modbus-TCP 通讯协议

如下所有命令都是以硬件地址为 01 来举例说明；

1　读模块配置字寄存器命令（03 功能码）

主设备发送报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	3D 46	为此次通信事务处理标识符，一般每次通信之后将被要求加 1 以区别不同的通信数据报文（此处以 3D 46 为列）	2
2	00 01	表示协议标识符（此处以 00 01 为列）	2
3	00 06	为数据长度，用来指示接下来数据的长度，高 8 位在前，低 8 位在后（此列表示后面跟随有 6 个字节的数据）	2
4	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
5	03	功能码	1
6	00 55	数据起始寄存器地址，高 8 位在前，低 8 位在后； 参照“配置字寄存器表”	2
7	00 02	读取寄存器个数，高 8 位在前，低 8 位在后 （此列读取 2 个寄存器数据）	2

从设备返回正确报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	3D 46	为此次通信事务处理标识符，应答报文要求与先前对应的主设备发送报文保持一致	2
2	00 01	表示协议标识符，与主设备发送报文保持一致	2
3	00 07	为数据长度，用来指示接下来数据的长度，高 8 位在前，低 8 位在后（此列表示后面跟随有 7 个字节的数据）	2
4	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
5	03	功能码	1
6	04	返回的数据字节个数，2 个寄存器*2	1
7	35 39 30 39	读取的寄存器数据，每 2 个字节表示一个寄存器数据，高位在前，低位在后；第 1 个寄存器数据在前	可变

2　读 8 路电阻值或温度、电压(以 1Ω分辩率寄存器为列）命令（支持 04 与 03 功能码,字节读）

主设备发送报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	3D 46	为此次通信事务处理标识符，一般每次通信之后将被要求加 1 以区别不同的通信数据报文（此处以 3D 46 为列）	2
2	00 01	表示协议标识符（此处以 00 01 为列）	2
3	00 06	为数据长度，用来指示接下来数据的长度，高 8 位在前，低 8 位	2

		在后（此列表示后面跟随有 6 个字节的数据）	
4	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
5	03	功能码	1
6	10 80	起始通道序号，高 8 位在前，低 8 位在后；参照”阻值寄存器表”	2
7	00 08	读取 8 个通道电阻值，高 8 位在前，低 8 位在后	2

从设备返回正确报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	3D 46	为此次通信事务处理标识符，应答报文要求与先前对应的主设备发送报文保持一致	2
2	00 01	表示协议标识符，与主设备发送报文保持一致	2
3	00 13	为数据长度，用来指示接下来数据的长度，高 8 位在前，低 8 位在后（此列表示后面跟随有 19 个字节的数据）	2
4	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
5	03	功能码 03 或 04	1
6	10	返回的数据字节个数，8 个寄存器*2	1
7	01 37 03 05 00 00 ...	读取的寄存器数据，每 2 个字节表示一个寄存器数据，高位在前，低位在后；第 1 个寄存器数据在前，数据还原参照”阻值寄存器表”	可变

3 配置寄存器修改命令：

3.1 单个寄存器修改命令（06 功能码，每次只能修改一个寄存器，举例修改通讯地址）

主设备发送报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	3D 46	为此次通信事务处理标识符，一般每次通信之后将被要求加 1 以区别不同的通信数据报文（此处以 3D 46 为列）	2
2	00 01	表示协议标识符（此处以 00 01 为列）	2
3	00 06	为数据长度，用来指示接下来数据的长度，高 8 位在前，低 8 位在后（此列表示后面跟随有 6 个字节的数据）	2
4	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
5	06	功能码	1
6	00 50	寄存器地址，高 8 位在前，低 8 位在后，参照“配置字寄存器表”	2
7	00 02	寄存器数据，参照“配置字寄存器表”	2

从设备返回正确报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	3D 46	为此次通信事务处理标识符，应答报文要求与先前对应的主设备发送报文保持一致	2
2	00 01	表示协议标识符，与主设备发送报文保持一致	2
3	00 06	为数据长度，用来指示接下来数据的长度，高 8 位在前，低 8 位在后（此列表示后面跟随有 6 个字节的数据）	2

4	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
5	06	功能码	1
6	00 50	寄存器地址，返回相同	2
7	00 02	寄存器数据，返回相同	2

3.2 连续修改多个寄存器命令（16 功能码，举例修改各通道补偿值）

主设备发送报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	3D 46	为此次通信事务处理标识符，一般每次通信之后将被要求加 1 以区别不同的通信数据报文（此处以 3D 46 为列）	2
2	00 01	表示协议标识符（此处以 00 01 为列）	2
3	00 0F	为数据长度，用来指示接下来数据的长度，高 8 位在前，低 8 位在后（此列表示后面跟随有 6 个字节的数据）	2
4	01	从设备地址，可变（1-253,254 与 255 为广播地址）（此列为 01 设备地址）	1
5	10	功能码	1
6	04 40	起始寄存器，高 8 位在前，低 8 位在后 参照“配置字寄存器表”	2
7	00 04	写入寄存器长度，高 8 位在前，低 8 位在后 （此列写入 4 个寄存器）	2
8	08	写入字节长度（写入寄存器长度 x2）	1
9	00 00 00 01 00 03 00 06	写入的数据,每 2 个字节表示一个寄存器数据，高位在前，低位在后；此列表示把 0440H 寄存器写入数据 0000H,0441H 寄存器写入数据 0001H，0442H 寄存器写入数据 0003H，0443H 寄存器写入数据 0006H	按序列 8 表示的字节数

从设备返回正确报文

排列顺序	数据字符（16 进制）	数据说明	字节数
1	3D 46	为此次通信事务处理标识符，应答报文要求与先前对应的主设备发送报文保持一致	2
2	00 01	表示协议标识符，与主设备发送报文保持一致	2
3	00 06	为数据长度，用来指示接下来数据的长度，高 8 位在前，低 8 位在后（此列表示后面跟随有 6 个字节的数据）	2
4	01	从设备地址，与主设备发送报文保持一致	1
5	10	功能码	1
6	04 40	起始寄存器，高 8 位在前，低 8 位在后 与主设备发送的报文相同	2
7	00 04	写入寄存器长度，高 8 位在前，低 8 位在后 与主设备发送的报文相同	2

附件 3：

如何切换 Modbus-RTU 与 Modbus-TCP 协议

（本说明适用 ZH-T16xx 与 ZH-T08xx 全系列产品）

如何在产品中切换 Modbus-TCP、Modbus-RTU、自定义协议以及用 Modbus-RTU 扩展下联模块？

A. 只需要用 06 功能码修改 0x1FA 寄存器就可改变串口的通信协议和工作方式。

B. 0x1FA 寄存器为 16 位寄存器，每 4 位对应一个通讯口设置，列表如下:

表（1）

0x1FA 寄存器位	对应产品通讯接口序号	对应产品通信接口	数据含义代码（16 进制）
Bit3:Bit0	第一通讯口	RS485 口	0x0--从机 Modbus-RTU 协议（RS485 出厂默认协议） 0x1--从机 Modbus-TCP 协议(以太网口出厂默认协议） 0x4--从机自定义协议 1 0x6--从机自定义协议 2
Bit7:Bit4	第二通讯口	以太网口	

C. 注意：因为所有通讯口的协议格式存储在同一寄存器（0x1FA)的不同位上(16 位 2 个字节)，而我们用 06 或 16 功能码修改时，是按字节修改的，所以在修改一个通讯口的协议时，要把其它通讯口的原协议代码保留填入，否则会同步修改。

D. 举例，当 RS485 口为 Modbus-RTU 协议时，通过 RS485 口更改通讯协议：

➢ RS485 口保持 Modbus-RTU 协议不变，以太网口协议修改为 Modbus-TCP，则需发送命令如下：

命令：01 06 01 FA 00 10 A9 CB(返回相同代码即修改成功)，解析如下表：

设备地址	功能码	改写的寄存器		改写的数据		CRC校验码	
		高8位	低8位	高8位(Bit15:Bit8)	低8位 (Bit7:Bit0)	高8位	低8位
01	06	01	FA	00 第4通讯口格式 第3通讯口格式	10 第2通讯口格式 第1通讯口格式	A9	CB

注：表中第 4 通讯口与第 3 通讯口未用到，填 0 就可以了。

➢ 当需要把 RS485 由当前通讯协议 Modbus-RTU 更改为 Modbus-TCP 协议，以太网口通讯协议改为 Modbus-RTU 时，，则需发送命令如下：

命令：01 06 01 FA 00 01 69 C7(返回相同指令即修改成功)；解析如下表：

设备地址	功能码	改写的寄存器		改写的数据		CRC校验码	
		高8位	低8位	高8位	低8位	高8位	低8位
01	06	01	FA	00 第4通讯口格式 第3通讯口格式	01 第2通讯口格式 第1通讯口格式	69	C7

E. 举例，由 Modbus-TCP 协议更改为 Modbus-RTU：

➢ RS485 口与以太网口当前通讯协议为 Modbus-TCP，如要全改成 Modbus-RTU 协议，则需要发命令：

命令：00 00 00 00 00 06 01 06 01 FA 00 00(返回相同代码即修改成功)；解析如下表：

事务标示符		协议标示符		数据长度		设备地址	功能码	改写的寄存器		改写的数据	
高8位	低8位	高8位	低8位	高8位	低8位			高8位	低8位	高8位	低8位
00	00	00	00	00	06	01	06	01	FA	00 第4通讯口格式 第3通讯口格式	00 第2通讯口格式 第1通讯口格式

附 4：

网络接口模块测试与设置方法

1、**网口功能特点**:

- 10/100Mbps 自适应以太网接口，支持 AUTO-MDIX 网线交叉直连自动切换;
- 工作模式可选择 TCP Serve、TCP Client、UDP Client、UDP Server、Httpd Client;
- 自定义心跳包机制，保证连接真实可靠，可用来检测死连接;
- 自定义注册包机制，可检测连接状态，识别模块，也可做自定义包头;
- TCP Server 模式下，连接 Client 的数量可在 1 到 16 个之间任意设置，默认 4 个，已连接 Client 的 IP 可在内置网页状态界面显示，按连接计算发送/接收数据;
- TCP Server 模式下，当连接数量达到最大值时，新连接是否踢掉旧连接可设置;
- 支持 TCP Client 短连接功能，短连接断开时间自定义;
- 支持超时重启（无数据重启）功能，重启时间自定义;
- TCP 连接建立前，数据缓存是否清理可设置;
- DHCP 功能，能够自动获取 IP;
- MAC 地址可修改，出厂烧写全球唯一 MAC，支持自定义 MAC 功能;
- DNS 功能，域名解析; DNS 服务器地址可自定义;
- 支持虚拟串口，可提供配套的虚拟串口软件;
- 可以跨越网关，交换机，路由器运行; 可以工作在局域网，也可访问外网;

网口出厂默认参数：

工作模式：TCP Serve;
IP：192.168.0.7;
端口号：20108;
用户名：admin;
密码：admin

与主芯片通信：波特率 115200pbs,数据位 8 位，1 位停止位，无奇偶校验。

2、模块工作方式设置（可网页登录设置或用专用的设置软件方式）：

2.1 自带内置的网页服务器，与常规的网页服务器相同，用户可以通过网页登录设置参数也可以通过网页查看模块的相关状态。网页服务器的端口号可设置，默认为 80。

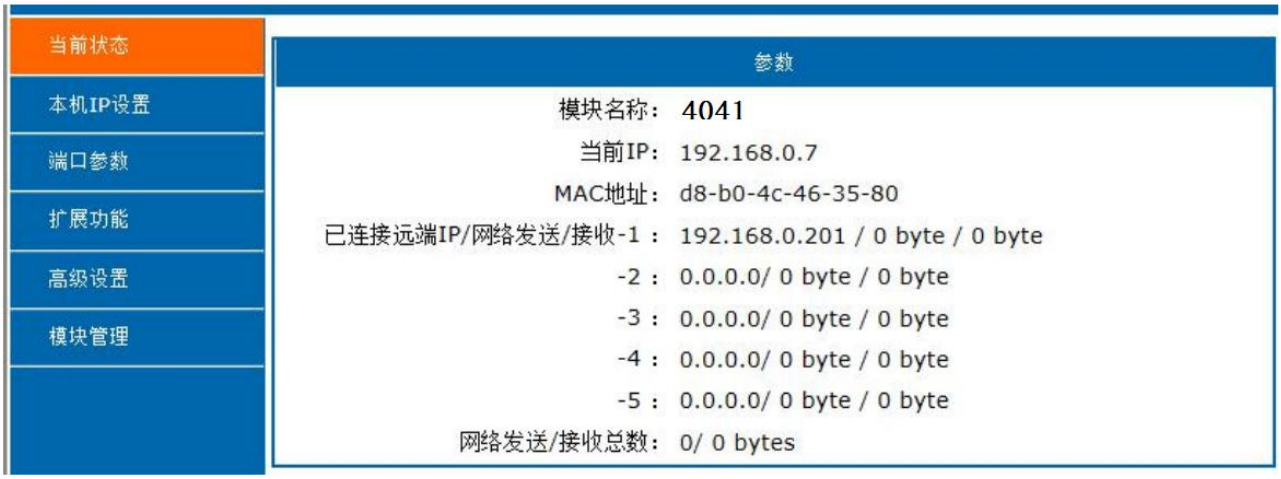
默认首页为当前状态界面，每隔 10s 刷新一次，显示模块工作状态：

网络发送总数：通过网络发送数据可以判断 模块发送多少数据到外网;

网络接收总数：通过接收计数可以判断有多少数据从网络发向模块;

已连接远端 IP/ 网络发送/ 接收：通过此项，可以看到 模块 与哪一个设备进行连接，该连接发送和接收的数据量有多少，目前只支持 5 个连接状态显示。

UDP Server 模式下，只显示发送/接收数据，不显示连接 IP。



图一、网页工作状态显示页面

图 2、模块参数网页设置页面

2.2 可至我司网站下载专用的设置工具软件，设置更直观快捷。

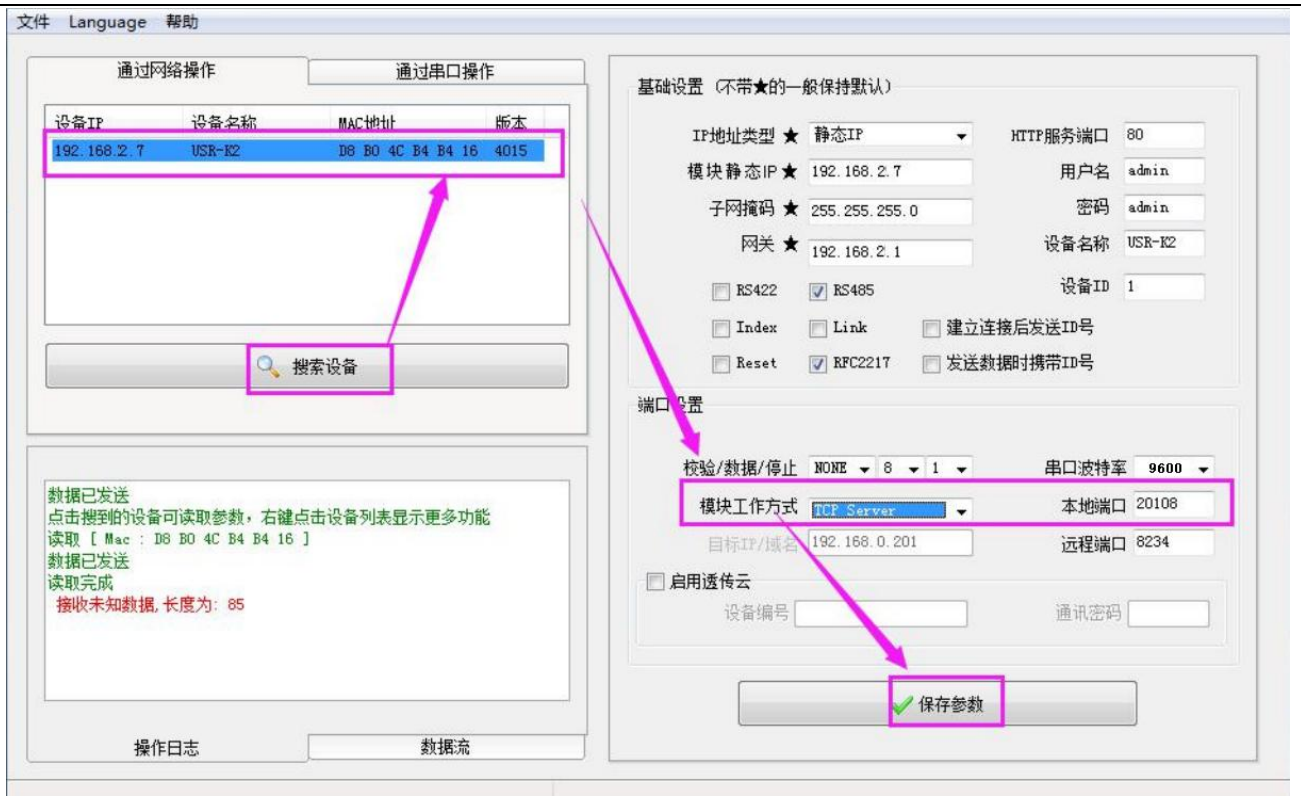


图 3、模块参数软件设置页面（可到本公司官网下载“网络设置软件”）

3、TCP Serve 模式通讯实例

模块设置为 TCP Serve 模式，IP 为 192.168.2.7，端口为 20108 的情况下，打开调试助手软件（本软件可以在本公司网站下载“串口调试助手”）按以下页面设置,本地 IP 需选择正错的本机电脑 IP;

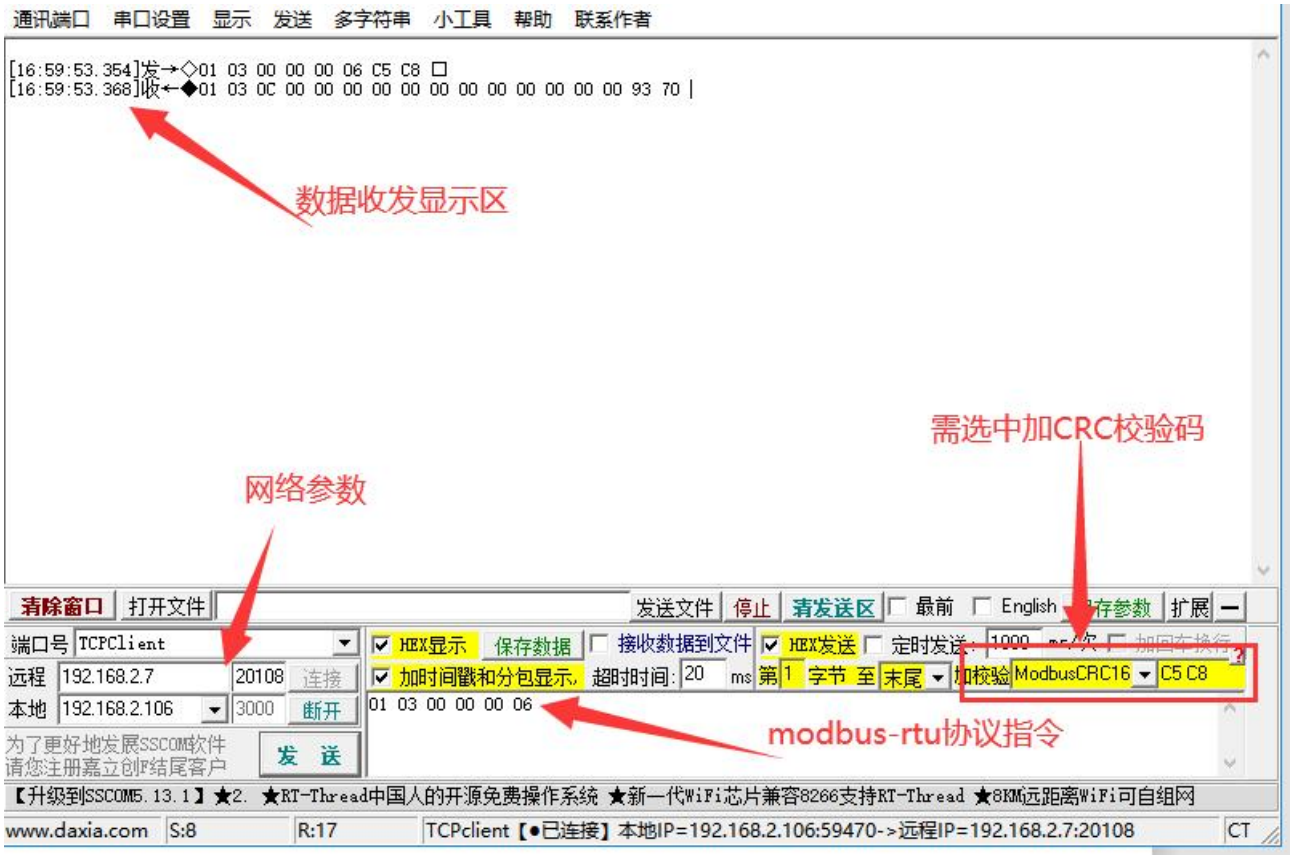


图 4、modbus-rtu 协议指令测试页面

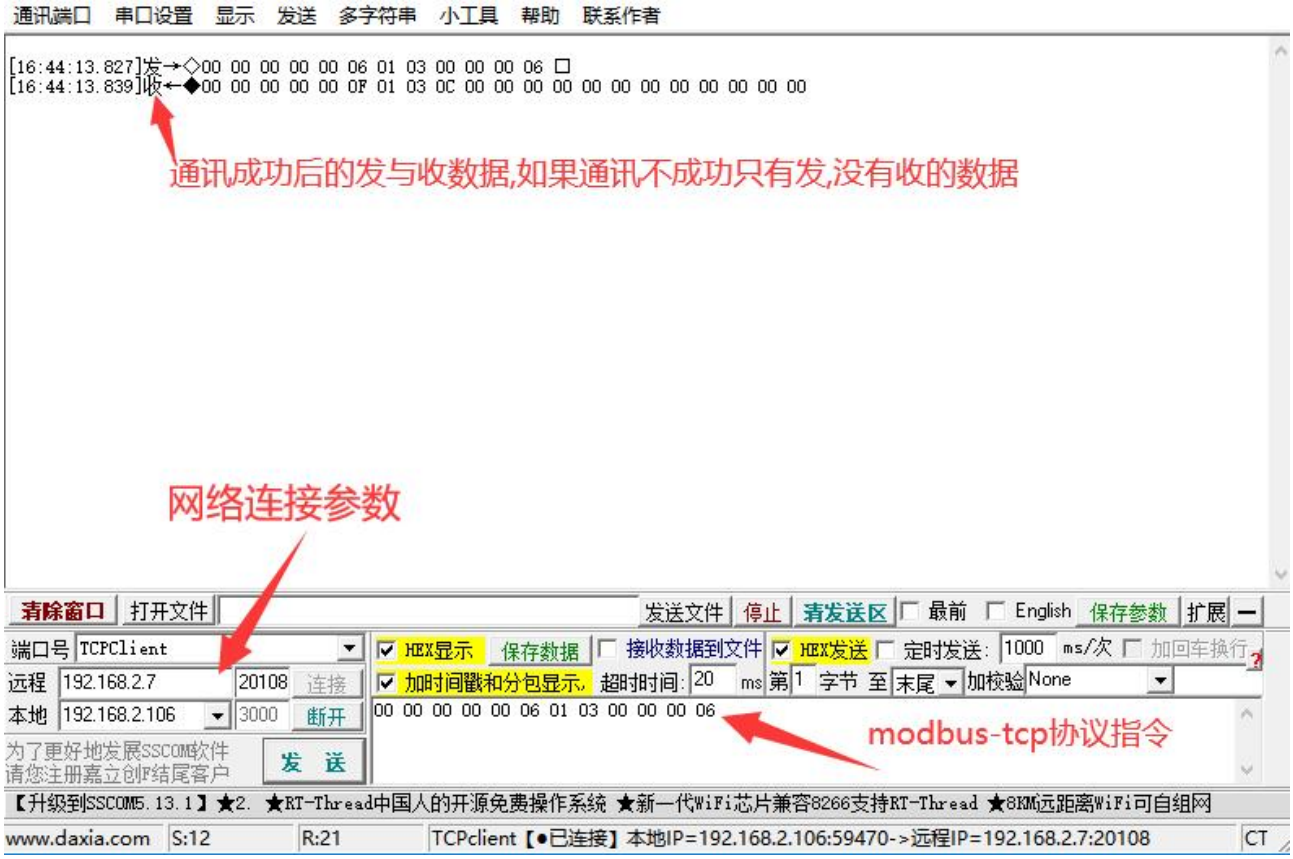


图 5、modbus-tcp 协议指令测试页面

附件 5：

8 路电阻与温度模块 CAN 通讯协议与设置

（适用于所有 8 路电阻或温度采集模块）

本系列模块全部采用 CAN2.0 B 扩展协议。分两种数据包情况，第 1 种为模块主动上传采样数据给上位机；第 2 种为上位机下发指令给模块。

1. 采集模块主动上传采样数据

模块主动上传数据时,把 ID 分成两部分，属于标准帧的前 11 位 ID，定义了 ID 优先级与模块的设备编址；

属于扩展帧里的 18 位 ID 定义了数据含义与数据包的帧数、帧序。

数据段则按顺序放置了主动上传的报警信号、开关量、温度、阻值、电压、电流等数据。列表格如下：

(1) 当数据为 16 位时：

CAN ID 仲裁段（硬件按位从左至右发送，表格省略了 RTR IDE 等位）					
前 11 位 ID			后 18 位 ID		
bit10:bit8	bit7:bit0		bit17:bit14	bit13:bit7	bit6:bit0
高 3 位默认为 0x7，即 3 位都是 1,可通过指令修改 0x04E8 寄存器改变此值，用于与其它不同类型设备优先级仲裁	低 8 位为设备编码地址，通过拨码开关或寄存器设置， 数据范围为 1~254，不能为 255。		用于指定上传类型,其值定义如下： 11--8 路温度值（16 位数据） 12--8 路电阻值（数据宽度由 0x04F3 寄存器定义） 13--8 路二极管电压（16 位数据）	数据包的帧数 如果是 16 位数据，则值为 2	此帧在数据包中的序号，第 1 帧为 0x01，第 2 帧为 0x02.....

每个数据包，仲裁段前 11 位 ID 与后 18 位 ID 的前 11 位每帧都是相同的，只有后 18 位 ID 的后 7 位不同

16 位数据 CAN 上传数据段（硬件按字节从左至右发送）									
每帧数据上传 4 路数据，每路占用两个字节，需要 2 帧传完 8 个通道									
帧序号	DLC	数据字节							
		1	2	3	4	5	6	7	8
1	8	第 1 路 高字节	第 1 路 低字节	第 2 路 高字节	第 2 路 低字节	第 3 路 高字节	第 3 路 低字节	第 4 路 高字节	第 4 路 低字节
2	8	第 5 路 高字节	第 5 路 低字节	第 6 路 高字节	第 6 路 低字节	第 7 路 高字节	第 7 路 低字节	第 8 路 高字节	第 8 路 低字节

表中为方便对照说明每帧数据，省略了每帧的 ID 仲裁段。

(2) 当数据为 32 位时：

CAN ID 仲裁段（硬件按位从左至右发送，表格省略了 RTR IDE 等位）					
前 11 位 ID			后 18 位 ID		
bit10:bit8	bit7:bit0		bit17:bit14	bit13:bit7	bit6:bit0
高 3 位默认为 0x7，即 3 位都是 1,可通过指令修改 0x04E8 寄存器改变此值，用于与其它不同类型设备优先级仲裁	低 8 位为设备编码地址，通过拨码开关或寄存器设置， 数据范围为 1~254，不能为 255。		用于指定上传类型,其值定义如下： 11--8 路温度值（16 位数据） 12--8 路电阻值（数据宽度由 0x04F3 寄存器定义） 13--8 路二极管电压（16 位数据）	数据包的帧数 如果上传的是 32 位数据，则值为 4	此帧在数据包中的序号，第 1 帧为 0x01，第 2 帧为 0x02.....

每个数据包，仲裁段前 11 位 ID 与后 18 位 ID 的前 11 位每帧都是相同的，只有后 18 位 ID 的后 7 位不同

32 位数据 CAN 上传数据段（硬件按字节从左至右发送）									
每帧数据上传 2 路数据，每路占用 4 个字节，需要 4 帧传完 8 个通道									
帧序号	DLC	数据字节							
		1	2	3	4	5	6	7	8
1	8	第 1 路 bit31:bit24	第 1 路 bit23:bit16	第 1 路 bit15:bit8	第 1 路 bit7:bit0	第 2 路 bit31:bit24	第 2 路 bit23:bit16	第 2 路 bit15:bit8	第 2 路 bit7:bit0
2	8	第 3 路 bit31:bit24	第 3 路 bit23:bit16	第 3 路 bit15:bit8	第 3 路 bit7:bit0	第 4 路 bit31:bit24	第 4 路 bit23:bit16	第 4 路 bit15:bit8	第 4 路 bit7:bit0
3	8	第 5 路 bit31:bit24	第 5 路 bit23:bit16	第 5 路 bit15:bit8	第 5 路 bit7:bit0	第 6 路 bit31:bit24	第 6 路 bit23:bit16	第 6 路 bit15:bit8	第 6 路 bit7:bit0
4	8	第 7 路 bit31:bit24	第 7 路 bit23:bit16	第 7 路 bit15:bit8	第 7 路 bit7:bit0	第 8 路 bit31:bit24	第 8 路 bit23:bit16	第 8 路 bit15:bit8	第 8 路 bit7:bit0

表中为方便对照说明每帧数据，省略了每帧的 ID 仲裁段。

2. 上位机读取本模块配置寄存器命令(03 指令)

(1) 上位机下发 03 指令：

上位机下发的指令，ID 仲裁段格式与模块主动上传数据时差不多，只是前 11 位 ID 换成了上位机设备编址或指令专用 ID.

CAN ID 仲裁段（硬件按位从左至右发送，表格省略了 RTR IDE 等位）				
前 11 位 ID		后 18 位 ID		
bit10:bit0		bit17:bit14	bit13:bit7	bit6:bit0
上位机设备编址或指令专用 ID. 下位机模块只会接收由 0x04E0 至 0x04E7 这 8 个寄存器设定好的专用 ID 指令，其它 ID 会忽略。 出厂时默认为：0x7FF 可通过指令修改此寄存器值		指令代码：03	数据包的帧数， 03 下发指令只有一帧，所以这里为 01	帧序号：01

数据段则分 3 部分：

第 1 字节为数据包编号，上位机每发一个命令，此编号需加 1，直到 255 时，又从 0 开始；

第 2 字节为模块的设备地址，只有此字节与模块设备地址相匹配时，模块才会执行上位机指令；

第 3 至第 8 字节为要读取的起始寄存器地址与要读取寄存器数量。

03 指令　CAN 数据段（硬件按字节从左至右发送）						
03 指令用于读取单个或多个配置寄存器值 寄存器地址按 16 位编码，03 指令只有 1 帧数据。						
帧顺序	数据长度 DLC	数据字节				
		1	2	3	4	5
第 1 帧	5	数据包编号，上位机每发一次命令就加 1，从 0 至 255 循环。同一次指令，此字节每帧都必须相同。	受控模块设备地址，只有与此字节匹配的模块设备才会响应指令，同一次指令，此字节每帧都必须相同。	需读取的起始配置寄存器地址高 8 位	需读取的起始配置寄存器地址低 8 位	需读取的寄存器数量，最大 255

(2) 模块返回的数据格式：

只有当上位机指令前 11 位 ID 与模块 0x04E0 至 0x04E7 寄存器值相匹配，且上位机指令数据段第 2 字节的值与模块的设备拨码地址相同时，模块才会响应上位机指令，并返回读取的寄存器数据报文。

CAN ID 仲裁段（硬件按位从左至右发送，表格省略了 RTR IDE 等位）				
前 11 位 ID		后 18 位 ID		
bit10:bit8	bit7:bit0	bit17:bit14	bit13:bit7	bit6:bit0
与模块主动上传采集数据时的内容相同	受控模块的设备地址 与模块主动上传采集数据时的内容相同	指令代码：03	数据包的帧数。 每帧可以上传 3 个寄存器的数据，所以此值与读取的寄存器数量相关。比如要读取 6 个寄存器，则此处为 02，如果要读取 7 个寄存器，则此处为 03.	帧序号

数据段则分 3 部分：

第 1 字节为数据包编号，与上位机指令中的编号对应；

第 2 字节为上位机指令前 11 位 ID 的低 8 位，注意此处不是 11 位，因为只有一个字节；

数据段第 1 字节与第 2 字节每帧都相同。

第 3 至第 8 字节依次排放的寄存器数值，可能会分多帧排放寄存器数据。

03 指令返回数据　CAN 数据段（硬件按字节从左至右发送）		
03 指令用于读取单个或多个配置寄存器值； 寄存器地址按 16 位编码，寄存器数据也是按 16 位存放，如果数据为 32 位的寄存器，则拆分成两上 16 位寄存器，高 16 位在前，低 16 位在后； 03 指令会返回至少 1 帧以上数据。		
帧顺序	数据	数据字节

	长度 DLC	1	2	3	4	5	6	7	8
第 1 帧	数据排满 时为 8，如不够 3 个寄存器值，则为实际字节数量	数据包编号，与上位机发送的指令中的编号相同。	上位机设备编码或指令专用 ID，与上位机指令中前 11 位 ID 的低 8 位相同。	读取的第 1 个寄存器的值高 8 位	读取的第 1 个寄存器的值低 8 位	每帧放置 3 个被读取的寄存器数据，依次按寄存器地址排放，末尾如不够 3 个寄存器，则按实际所剩寄存器数量发送字节.....			
第 2 帧	数据排满 时为 8，如不够 3 个寄存器值，则为实际字节数量			.....依次排放寄存器数据，直到所有寄存器数据发完					
.....	.....			.....依次排放寄存器数据，直到所有寄存器数据发完					

(3) 指令举例：

假设模块认可的上位机 ID（0x0063 寄存器值）为：0x7FF；模块本身的设备地址为 01；如果要读取模块的当前采样更新速率，即 0x0081 寄存器的值，则上位机发送：

ID 仲裁段（按二进制排列）： 111 11111111 0011 0000001 0000001 注意二进制位数

数据段（按字节排列,第 1 字节数据包编码随意填）：0x20 0x01 0x00 0x81 0x01 DLC 为 05

如果当前采样速率为第 2 档，则模块会返回数据：

ID 仲裁段（按二进制排列）： 111 00000001 0011 0000001 0000001 注意二进制位数

数据段（按字节排列）：0x20 0xFF 0x00 0x02 DLC 为 04

3. 上位机修改本模块配置寄存器命令(06 指令)

(1) 上位机下发 06 指令：

CAN ID 仲裁段（硬件按位从左至右发送，表格省略了 RTR IDE 等位）			
前 11 位 ID	后 18 位 ID		
bit10:bit0	bit17:bit14	bit13:bit7	bit6:bit0
上位机设备编址或指令专用 ID. 下位机模块只会接收由 0x0061 配置寄存器设定好的专用 ID 指令，其它 ID 会忽略。 出厂时默认为：0x7FF 可通过指令修改此寄存器值	指令代码：06	数据包的帧数，06 功能码第 1 帧为命令帧，第 2 帧开始为寄存器数据，每 3 个寄存器一帧。所以此处的值=1+需修改的寄存器数/3，如寄存器数除以 3 还有余数，则此值还要加 1	帧序号

数据段则分 3 部分：

第 1 字节为数据包编号，上位机每发一个命令，此编号需加 1，直到 255 时，又从 0 开始；

第 2 字节为模块的设备地址，只有此字节与模块设备地址相匹配时，模块才会执行上位机指令；

数据段第 1 字节与第 2 字节每帧都相同。

第 1 帧的第 3 至 5 字节为起始寄存器地址与数量；

第 2 帧开始，第 3 至第 8 字节依次排放要修改寄存器数据。

<table>
<tr><th colspan="10">06 指令　CAN 数据段（硬件按字节从左至右发送）</th></tr>
<tr><td colspan="10">06 指令用于修改单个或多个配置寄存器值
寄存器地址按 16 位编码，寄存器数据也是按 16 位存放，如为 32 位寄存器，则拆分成两个 16 位寄存器，高 16 位在前，低 16 位在后，06 指令可能有 2 至 127 帧数据</td></tr>
<tr><td rowspan="2">帧顺序</td><td rowspan="2">数据
长度
DLC</td><td colspan="8">数据字节</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>第 1 帧</td><td>5</td><td rowspan="4">数据包编号，上位机每发一次命令就加 1，从 0 至 255 循环。同一次指令，每帧都必须相同。</td><td rowspan="4">受控模块设备地址，同一次指令，每帧都必须相同。</td><td>需修改的起始寄存器地址高 8 位</td><td>需修改的起始寄存器地址低 8 位</td><td>需修改的寄存器数量，最大 255</td><td>空</td><td>空</td><td>空</td></tr>
<tr><td>第 2 帧</td><td>数据排满 时为 8，如不够 3 个寄存器值，则为实际字节数量</td><td>被修改的第 1 个寄存器的值高 8 位</td><td>被修改的第 1 个寄存器的值低 8 位</td><td colspan="4">从第 2 帧开始，每帧放置 3 个寄存器的数据，依次排放被修改的寄存器值，末尾如不够 3 个寄存器，则按实际所剩寄存器数量发送字节.....</td></tr>
<tr><td>第 3 帧</td><td>数据排满 时为 8，如不够 3 个寄存器值，则为实际字节数量</td><td colspan="6">.....依次排放寄存器数据，直到所有寄存器数值发完</td></tr>
<tr><td>.....</td><td>.....</td><td colspan="6">.....依次排放寄存器数据，直到所有寄存器数值发完</td></tr>
</table>

(2) 模块返回的数据格式：

当模块收到的指令 ID 与设备地址都相匹配时，会立即用收到的数据修改寄存器，然后再返回寄存器修改的值（包括没有修改成功的寄存器值）。

<table>
<tr><th colspan="5">CAN ID 仲裁段（硬件按位从左至右发送，表格省略了 RTR IDE 等位）</th></tr>
<tr><th colspan="2">前 11 位 ID</th><th></th><th colspan="2">后 18 位 ID</th></tr>
<tr><td>bit10:bit8</td><td>bit7:bit0</td><td></td><td>bit17:bit14</td><td>bit13:bit7</td><td>bit6:bit0</td></tr>
<tr><td>与模块主动上传采集数据时的内容相同</td><td>受控模块的设备地址
与模块主动上传采集数据时的内容相同</td><td></td><td>指令代码：06</td><td>数据包的帧数。
每帧可以上传 3 个寄存器的数据，所以此值与修改的寄存器数量相关。比如要修改 6 个寄存器，则此处为 02，如果要读取 7 个寄存器，则此处为 03；</td><td>帧序号</td></tr>
</table>

数据段则分 3 部分：

第 1 字节为数据包编号，与上位机指令中的编号对应；

第 2 字节为上位机指令前 11 位 ID 的低 8 位，注意此处不是 11 位，因为只有一个字节；

数据段第 1 字节与第 2 字节每帧都相同。

第 3 至第 8 字节依次排放的寄存器数值，可能会分多帧排放寄存器数据。

<table>
<tr><th colspan="10">06 指令返回数据　CAN 数据段（硬件按字节从左至右发送）</th></tr>
<tr><td colspan="10">寄存器地址按 16 位编码，寄存器数据也是按 16 位存放；
06 指令会返回至少 1 帧以上数据。</td></tr>
<tr><td rowspan="2">帧顺序</td><td rowspan="2">数据
长度
DLC</td><td colspan="8">数据字节</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>第 1 帧</td><td>数据排满 时为 8，如不够 3 个寄存器值，则为实际字节数量</td><td rowspan="2">数据包编号，与上位机发送的指令中的编号相同。</td><td rowspan="2">上位机设备编码或指令专用 ID，与上位机指令中前 11 位 ID 的低 8 位相同。</td><td>被修改的
第 1 个
寄存器值
高 8 位</td><td>被修改的
第 1 个
寄存器值
低 8 位</td><td colspan="4">每帧放置 3 个被修改后的寄存器数据，依次按寄存器地址排放，末尾如不够 3 个寄存器，则按实际所剩寄存器数量发送字节.....</td></tr>
<tr><td>第 2 帧</td><td>数据排满 时为 8，如不够 3 个寄存器值，则为实际字节数量</td><td colspan="6">.....依次排放寄存器数据，直到所有寄存器数据发完</td></tr>
<tr><td>.....</td><td>.....</td><td colspan="8">.....依次排放寄存器数据，直到所有寄存器数据发完</td></tr>
</table>

(3) 指令举例：

假设模块认可的上位机 ID（0x0063 寄存器值）为：0x7FF；模块本身的设备地址为 01；如果要把模块的第 1 通道改成 E 型热电偶采集温度，即 0x0200 寄存器的值要改为 02。

需要分两帧发送：

第一帧：

ID 仲裁段（按二进制排列）： 111 11111111 0110 0000010 0000001　　注意二进制位数

数据段（按字节排列，第 1 字节数据包编码随意填）：0x21 0x01 0x02 0x00 0x01　DLC 为 05

第二帧：

ID 仲裁段（按二进制排列）： 111 11111111 0110 0000010 0000010　　注意二进制位数

数据段（按字节排列，第 1 字节数据包编码随意填）：0x21 0x01 0x00 0x02　　DLC 为 04

如果修改成功，则模块会返回数据：

ID 仲裁段（按二进制排列）： 111 00000001 0110 0000001 0000001　　注意二进制位数

数据段（按字节排列）：0x21 0xFF 0x00 0x02　　DLC 为 04