ZH-T08TI 8 路隔离型热电偶温度测量模块

使用说明书(V1.0)

1、概述

本模块采用高精度 32 位 AD 芯片+ARM32 位工业级 MCU，每个测量通道间相互隔离，精度高，速度快，抗干扰好。支持 B、C、E、J、K、N、R、S、T 九种热电偶型号测温，自由配置，可广泛用于各种温度测量场合。

具有以下特点：

- ✧ 具有宽电源供电 DC9-30V；
- ✧ 32 位高精度 AD 高分辨率，误差<±0.1℃（热电偶为 K 型时的采样误差，排除热电偶本身误差后的数值），超小温度漂移；
- ✧ 支持 B、C、E、J、K、N、R、S、T 多种热电偶温度传感器,同一模块可混装不同探头。
- ✧ 采样周期具有 45ms,100ms,220ms,450ms 四种速率可设置；
- ✧ 可选 RS485、CAN、以太网通讯等组合通讯接口，支持 CAN2.0B 通讯协议；可定制 4G、Wifi 等通讯方式。
- ✧ 具有℃(摄氏度)与℉(华氏度)两种温度单位的数据寄存器可自由读取；
- ✧ 每一个通道间相互隔离；测量通道、电源、通讯三种也相互隔离，可靠性高；
- ✧ RS485 或以太网可灵活自选 Modbus-RTU 或 Modbus-TCP 工业通信协议，与各种组态屏、工控软件以及模组进行可靠通信。
- ✧ 可选 CAN 总线通讯，支持 CAN2.0B 通讯协议；

2、产品主型号

ZH-T08TI-14N1　8 路热电偶测温，RS485 接口；
ZH-T08TI-34N1　8 路热电偶测温，以太网接口+RS485；
ZH-T08TI-74N1　8 路热电偶测温，CAN+RS485；

3、性能指标

表 3.1　性能参数表

测量参数		
项目	**参数**	**说明**
热电偶类型	B、C、E、J、K、N、R、S、T	通过修改 0x0200--0x0207 寄存器选择型号，同一模块每个通道可单独设置，可同时用不同的热电偶） 出厂默认:K 型
测温范围	50~1700℃	B 型
	0~2320℃	C 型
	-200~900℃	E 型
	0~750℃	J 型
	-200~1250℃	K 型（出厂默认）
	-270~1300℃	N 型
	0~1450℃	R 型
	0~1450℃	S 型
	-200~350℃	T 型

精度误差（排除了热电偶本身误差与冷端补偿误差之后的精度）	±0.1℃ 最大	450 毫秒更新速率
	±0.1℃ 最大	220 毫秒更新速率
	±0.1℃ 最大	100 毫秒更新速率 （出厂默认）
	±0.2℃ 最大	45 毫秒更新速率
冷端误差	±0.4℃	实际测温误差=精度误差+冷端误差+热电偶本身误差（非简单相加，相互交错）
分 辨 率	0.1℃	
温度漂移	≤±50ppm/℃	
数据更新速率	45ms,100ms,220ms,450ms 共 4 种	速度越慢，精度会高越稳定 修改 0x0081 寄存器调整刷新时间 出厂默认：100ms

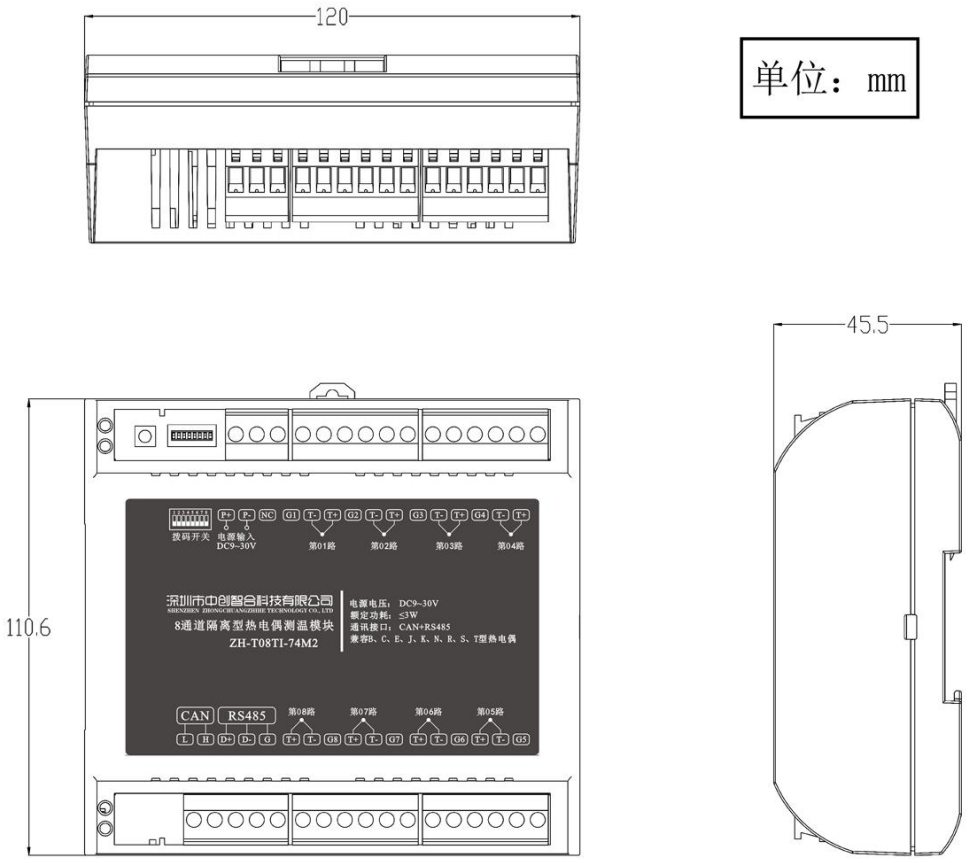
电气参数		
项目	**参数**	**说明**
隔离耐压	>1500V DC	电源、通信、采集电路三者相互隔离 每路采集电路之间相互隔离
供电电源	+9V~+30V DC	
额定功耗	≤ 3W	

通讯参数		
项目	**参数**	**说明**
输出接口	RS485	ZH-T08TI-14N1
	以太网口+RS485	ZH-T08TI-34N1
	CAN+RS485	ZH-T08TI-74N1 CAN 与 RS485 共用电源，两者不隔离，但与模块输入电源隔离
通讯协议	RS485 与以太网： Modbus-RTU 或 Modbus-TCP	可通过修改 0x01FA 寄存器配置 RS485 接口：出厂默认 Modbus-RTU 以太网接口：出厂默认 Modbus-TCP
	CAN: CAN2.0B 扩展通讯格式 私有协议	
通讯波特率	RS485 与以太网： 4800、9600、19200、38400、57600、115200bps 供选择	RS485 口通过修改 0x0051 寄存器调整，出厂默认为 9600 网口与主控芯片间波特率通过修改 0x005A 寄存器调整，出厂默认为 115200，一般用户不需调整这个；
	CAN: 5kpbs、10kpbs、20kpbs、40kpbs、50kpbs、80kpbs、100kpbs、125kpbs、200kpbs、250kpbs、400kpbs、500kpbs、800kpbs、1000kpbs 供选择	通过修改 0x0061 寄存器修改波特率，出厂默认为 500kpbs

RS485 口出厂参数: 设备地址为 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 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。
CAN 通讯出厂参数: 设备地址为 1 号（ID 前 11 位的低 8 位，与 RS485 共用同一个设备号），ID 前 11 位的高 3 位为 0b111，波特率 500kpbs，采用 CAN2.0B 扩展格式。

其它参数		
项目	参数	说明
工作温度	-40℃～+70℃	
工作湿度	<90% 相对湿度	相对湿度
外观尺寸	120*110.6*45.5mm	
安装方式	导轨安装	
模块重量	约 250g	

4、外形尺寸图



5、拨码开关与端子接线定义图

5.1 拨码开关配置说明

PCB 上的拨码开关可以定义硬件地址，如下：

表 1

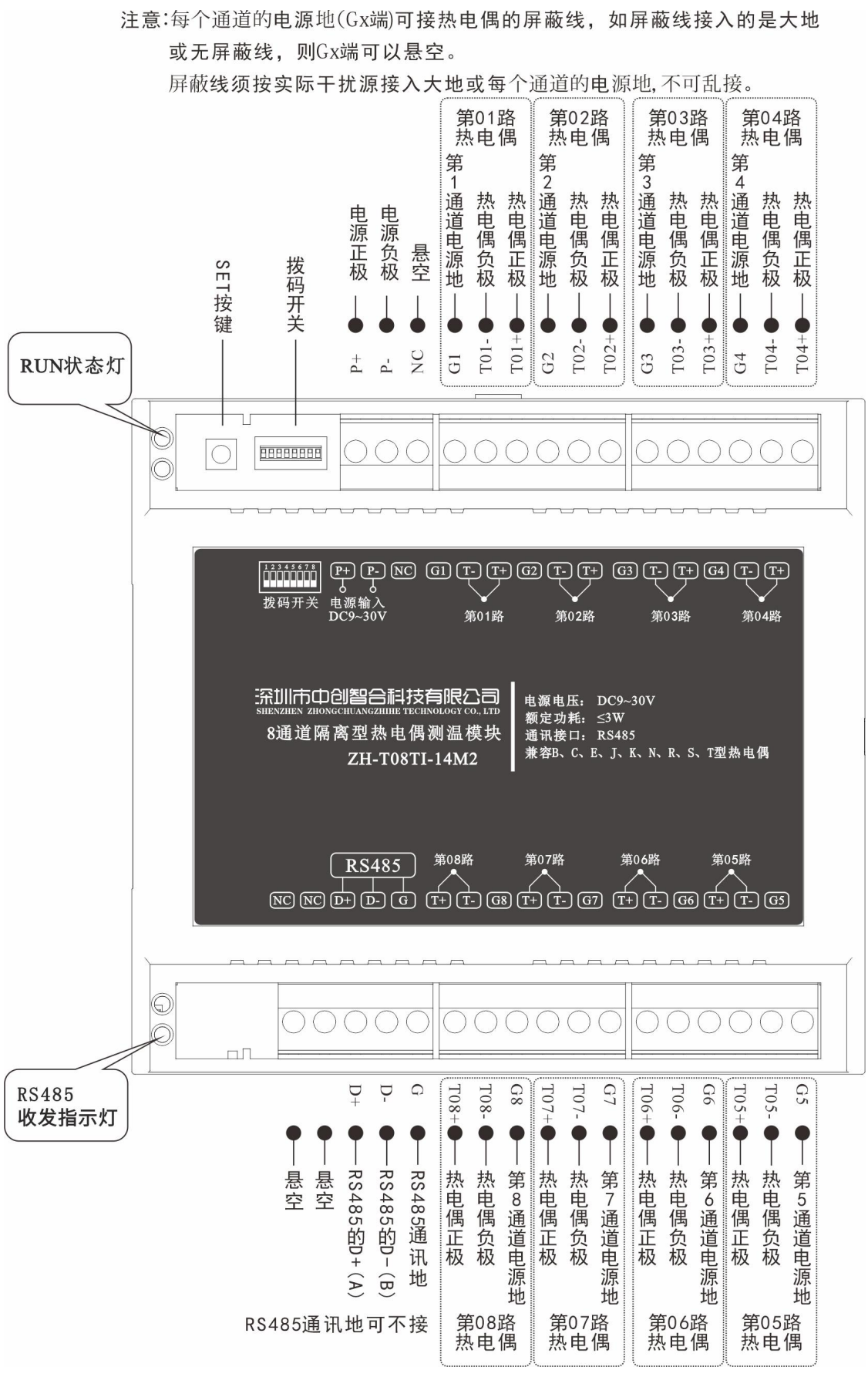
拨码开关位	功能	详情
第 8 位	保留	无效
第 7 位	保留	无效
第 6 位	保留	无效
第 5 至 1 位	设备地址 Bit4 至 Bit0 位	当第 5 至 1 位有拨码时，对应设备址址 Bit4--Bit0，高 2 位恒为 0，举例如下： Bit4=OFF，Bit3=OFF，Bit2=OFF，Bit1=OFF，Bit0=ON，地址为 1 Bit4=OFF，Bit3=OFF，Bit2=OFF，Bit1=ON，Bit0=OFF，地址为 2 。。。 Bit4=ON，Bit3=ON，Bit2=ON，Bit1=ON，Bit0=OFF，地址为 30 Bit4=ON，Bit3=ON，Bit2=ON，Bit1=ON，Bit0=ON，地址为 31 当第 5 至 1 位无拨码时，设备地址为寄存器 0x0050 的值，可通过通讯协议修改此值。
状态灯	RUN：运行指示灯，采集更新速率越快，闪烁频率越高； CAN 通讯灯：　通讯数据收发指示灯，闪红灯发送数据，闪绿接收到数据； RS485 通讯灯：　通讯数据收发指示灯，闪红灯发送数据，闪绿接收到数据；	

5.2 一键复位

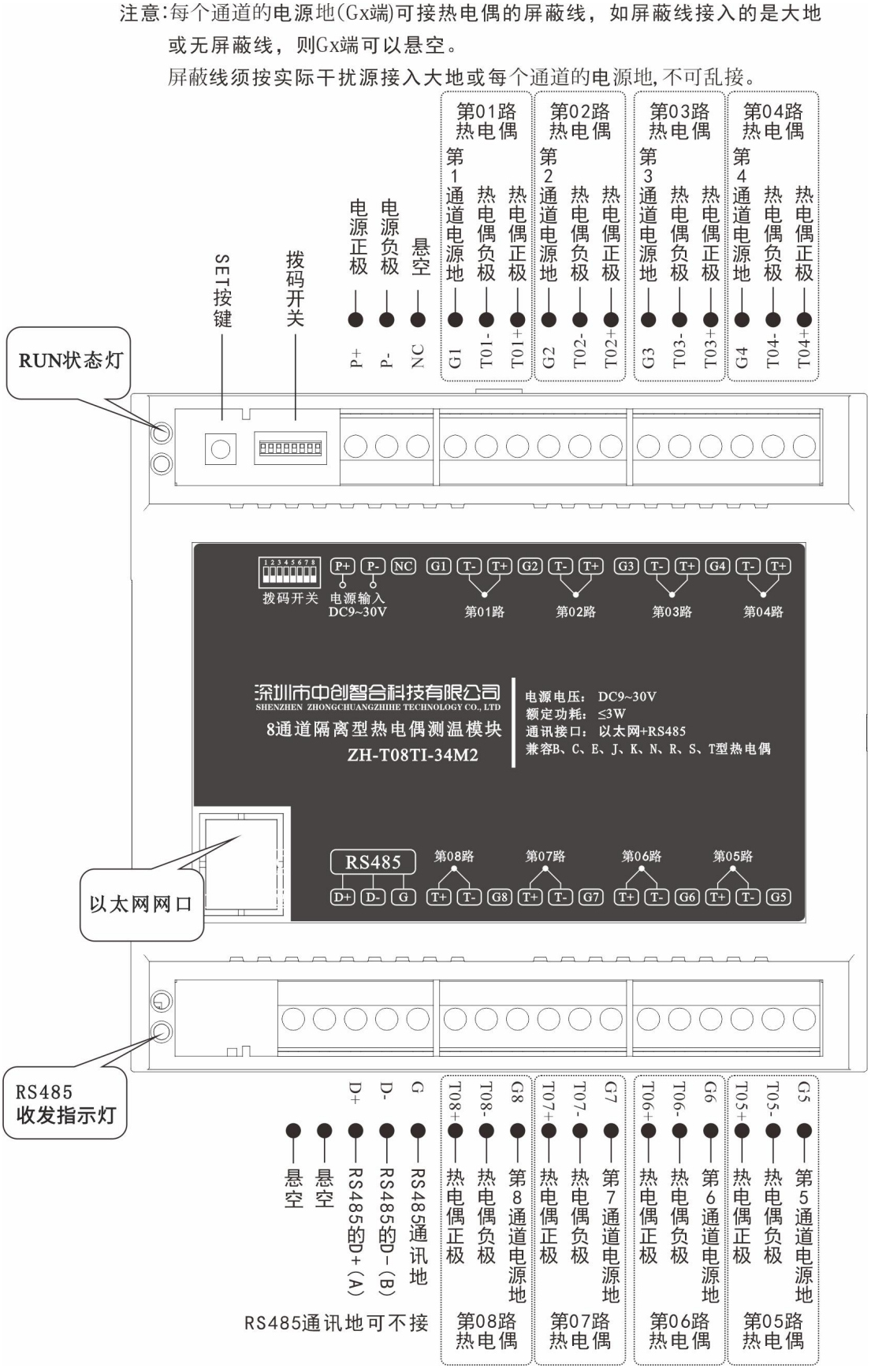
一键复位功能能在设置出错时，一键复位至出厂状态，步骤如下：

a. 产品左上角 SET 键不松开；
b. 重新上电或按一下复位键；
c. 此时保持 SET 键不松开，RUN 指示会先亮 1 秒，然后熄灭 2 秒；
d. 当出现 RUN 指示灯慢闪时，如果松开 SET 键，则复位通讯设置；
e. 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。

5.3 端子定义:



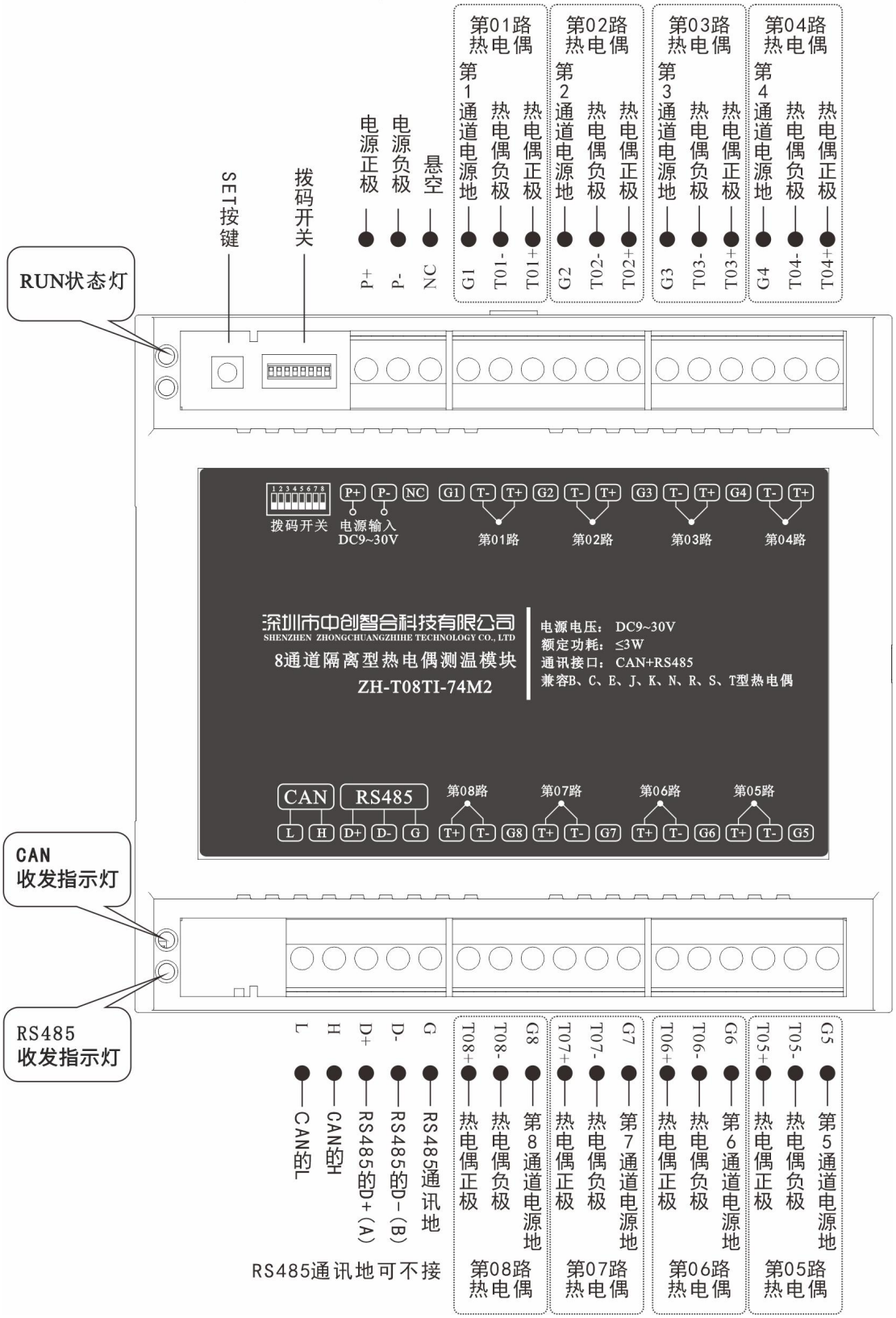
图：RS485 接口版本接线图



图：以太网接口版本接线图

注意:每个通道的电源地(Gx端)可接热电偶的屏蔽线，如屏蔽线接入的是大地或无屏蔽线，则Gx端可以悬空。

屏蔽线须按实际干扰源接入大地或每个通道的电源地,不可乱接。



图：CAN 接口版本接线图

5.4 各端子符号说明：

表 5.4.1　端子符号说明

符号	定义	说明
P+	电源正极	
P-	电源负极	
D+	RS485 接口 D+	也称 A 端
D-	RS485 接口 D-	也称 B 端
G	RS485 接口的电源地	实际应用一般不接线
H	CAN 接口的 H 端	
L	CAN 接口的 L 端	
T+、T01+、T02+...T08+	热电偶正端输入	
T-、T01-、T02-...T08-	热电偶负端输入	
G1、G2...G8	每个采样通道的电源地	每个通道的电源相互隔离，不可以接在一起，否则隔离不起作用 可视现场情况，接入热电偶屏蔽线

6、寄存器说明

6.1 温度寄存器（支持用 03 和 04 功能码读，不能改写）

寄存器内容		寄存器地址（十进制）	对应 PLC 或组态软件配置地址	寄存器数据说明
摄氏度格式	第 1 通道温度值	0000	40001	1、数据是 16 位有符号二进制数据（2 字节），高字节在前，低字节在后。 2、采集的温度分辨率为 0.1 摄氏度。 3、温度值 = 寄存器的数据 ÷ 10（请参考下表例子）。 4、当数值为 0x7FFF 时，表示传感器断开或过温；0x8000 时，表示传感器短路或超低温。
	第 2 通道温度值	0001	40002	
	第 3 通道温度值	0002	40003	
	第 4 通道温度值	0003	40004	
	第 5 通道温度值	0004	40005	
	第 6 通道温度值	0005	40006	
	第 7 通道温度值	0006	40007	
	第 8 通道温度值	0007	40008	
	冷端温度值	8480	48481	
华氏度格式	第 1 通道温度值	8448	48449	1、数据是 16 位有符号二进制数据（2 字节），高字节在前，低字节在后。 2、采集的温度分辨率为 0.1 华氏度。 3、温度值 = 寄存器的数据 ÷ 10 4、当数值为 0x7FFF 时，表示传感器断开或过温；0x8000 时，表示传感器短路或超低温。
	第 2 通道温度值	8449	48450	
	第 3 通道温度值	8450	48451	
	第 4 通道温度值	8451	48452	
	第 5 通道温度值	8452	48453	
	第 6 通道温度值	8453	48454	
	第 7 通道温度值	8454	48455	
	第 8 通道温度值	8455	408456	
	冷端温度值	8481	48482	

温度值数据举例：

比如数值 16 进制时为 0x058E，转成 10 进制为 1422，则温度为 142.2℃；

比如数值 16 进制时为 0xFDA3，第 16 位为 1，则为负数，转成 10 进制为-605，则温度为-60.5℃.

6.2 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 3

寄存器地址 (Hex)	保持寄存器内容	寄存器个数	寄存器状态	数据范围
0050H	地址	1	读/写	地址(此值可填 1-253，254 与 255 为广播地址)(默认 01) 如果板端拨码开关第 1 位至 5 位无拨码，则产品用此寄存器地址；如果有拨码，则由拨码开关第 5 至 1 位（对应二进制 bit4 至 bit0 位）决定地址。
0051H	波特率 （RS485 口）	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
0052H	寄偶校验 （RS485 口）	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
0055H	模块名称--高	1	读/写	默认:5430H（T0 的 ASCII 码）
0056H	模块名称--中	1	读/写	默认:3853H（8T 的 ASCII 码）
0057H	模块名称--低	1	读/写	默认:6900H（i 的 ASCII 码）
0058H	软件版本	1	读	3033：03 的 ASCII 码
0059H	软件子版本	1	读	3031：01 的 ASCII 码
005AH	网口与 MCU 通讯速率	1	读/写	同 0051H
005BH	网口与 MCU 通讯寄偶校验	1	读/写	同 0052H
0060H	CAN 标准帧 11 位 ID 高 3 位	1	读/写	出厂默认为：07，即 3 位都是 1

0061H	CAN 的波特率	1	读/写	数值定义： 0--5kpbs 1--10kpbs 2--20kpbs 3--40kpbs 4--50kpbs 5--80kpbs 6--100kpbs 7--125kpbs 8--200kpbs 9--250kpbs 10--400kpbs 11--500kpbs 12--800kpbs 13--1Mpbs 出厂默认为：11--500kpcs
0063H	可改写或控制本模块的 上位机 ID (CAN 的前 11 位 ID)	1	读/写	出厂默认为： 0x07FF
0081H	采样速率	1	读/写	0--刷新时间 450 毫秒 1--刷新时间 220 毫秒 2--刷新时间 100 毫秒（默认） 3--刷新时间 45 毫秒 刷新时间越慢，精度越高越稳定。
0082H	电网配置	1	读/写	其值为： 50--适用频率为 50HZ 的电网 60--适用频率为 60HZ 的电网
0083H	校正标志	1	读/写	其值为 0x5AF0 时，表示出厂已校正
0087H	冷端温度补偿	1	读/写	用户可通过对此寄存器写正数或负数来校正冷端温度。 参与运算的冷端温度=测量出的值+此误差值
01FAH	通讯协议定义	1	读/写	详见附件 《如何切换 Modbus-RTU 与 Modbus-TCP 协议》
01FBH	RS485 接口与 以太网口 主动上传控制	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。 主动上传格式请参照主动上传小节

0200H--0207H	热电偶型号设置	8	读/写	对应 1 至 8 路传感器型号。其值为： 0--B 型热电偶 1--C 型热电偶 2--E 型热电偶 3--J 型热电偶 4--K 型热电偶 5--N 型热电偶 6--R 型热电偶 7--S 型热电偶 8--T 型热电偶 出厂默认为 K 型热电偶
02C0H~02C7H	客户温度补偿	8	读/写	对应 1 至 8 通道温度补偿，用户可填入正负数值修正温度误差。此值单位为:0.1℃ 实际温度=测量温度+此补偿值 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 加锁保护。
8000H	温度补偿与报警值写保护	1	读/写	向此寄存器写 0x0A 才能改写温度补偿寄存器；改写完后，写 0x05 锁存，并生效温度补偿

版本： V1.0 2023.05.24

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

序列	数据举列 (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

序列	数据举列 (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 路温度命令（支持 04 功能码,字节读）

主设备发送报文

序列	数据举列 (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 00	起始通道序号，高 8 位在前，低 8 位在后；参照“温度寄存器表”	2
4	00 08	读取 8 个通道温度，高 8 位在前，低 8 位在后	2
5	44 0C	CRC 校验码（低位在前，高位在后）	2

从设备返回正确报文

序列	数据举列 (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“温度寄存器表”	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

序列	数据举列 (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

序列	数据举列 (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

序列	数据举列 (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	按序列 8 表示的字节数

		数据 0006H	
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

序列	数据举列 (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

序列	数据举列 (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

序列	数据举列 (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 路温度命令（支持 04 与 03 功能码,字节读）

主设备发送报文

序列	数据举列 (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 03 或 04	1
6	00 00	起始通道序号，高 8 位在前，低 8 位在后；参照“温度寄存器表”	2
7	00 08	读取 8 个通道温度，高 8 位在前，低 8 位在后	2

从设备返回正确报文

序列	数据举列 (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“温度寄存器表”	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

序列	数据举列 (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

序列	数据举列 (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

序列	数据举列 (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

序列	数据举列 (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。

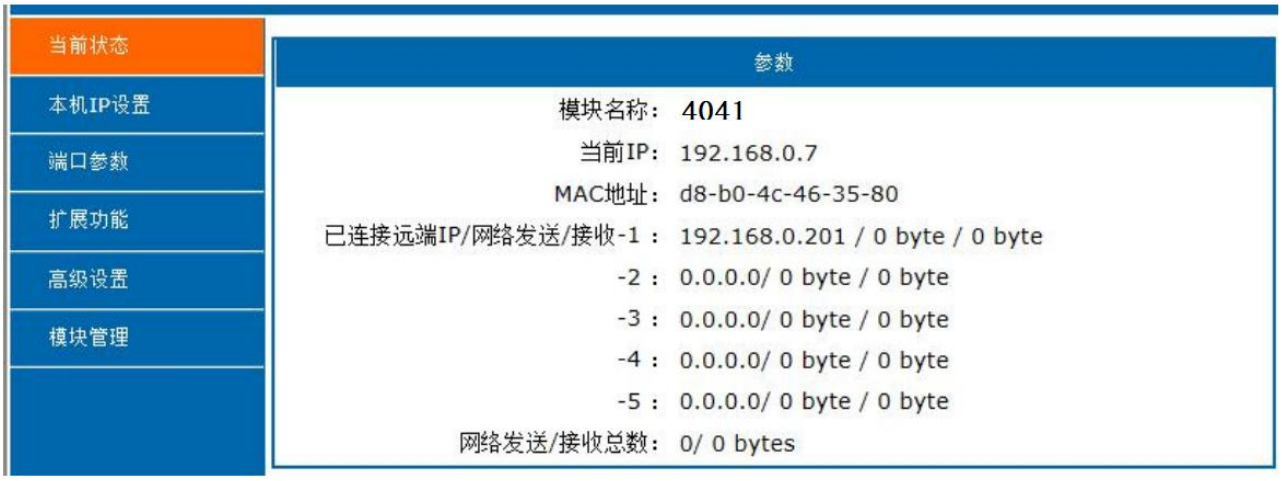
默认首页为当前状态界面，每隔 10s 刷新一次，显示模块工作状态：

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。

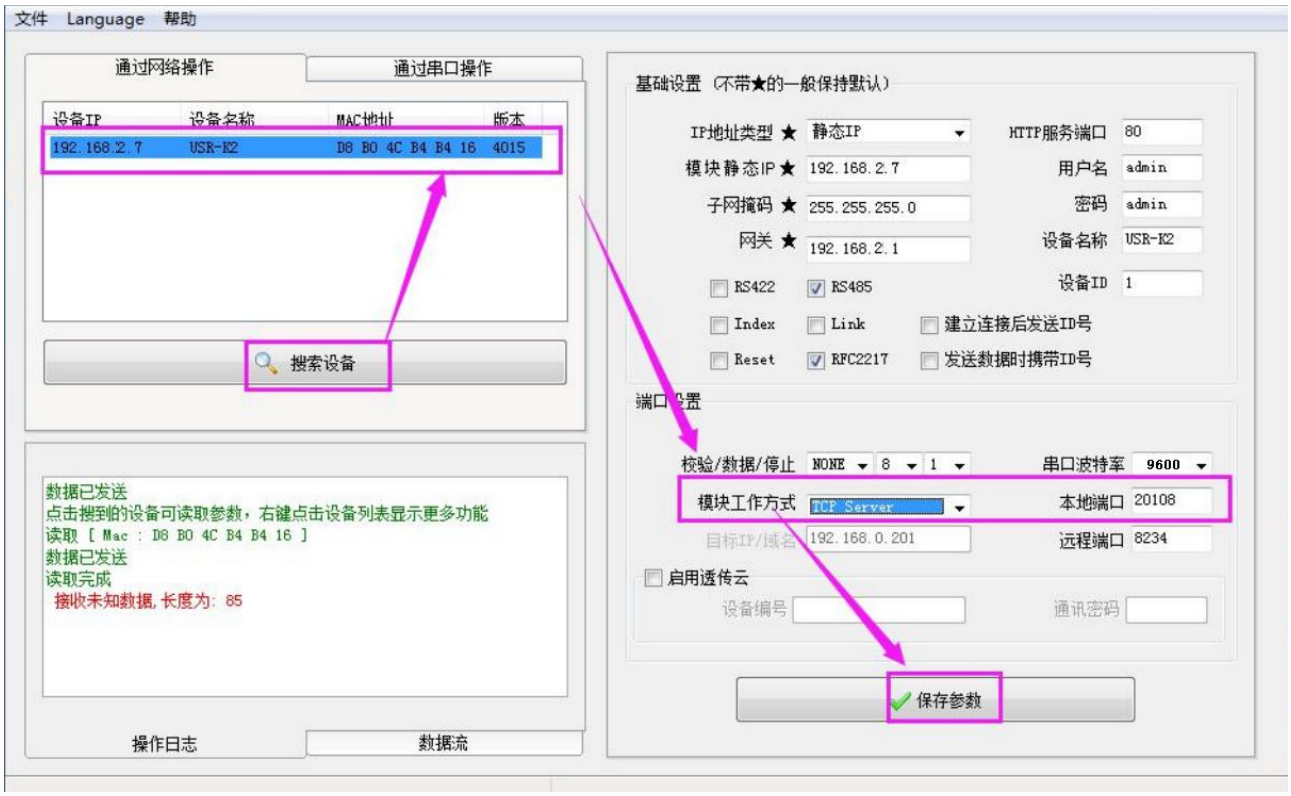


图 3、模块参数软件设置页面（可到本公司官网下载“网络设置软件”）

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;

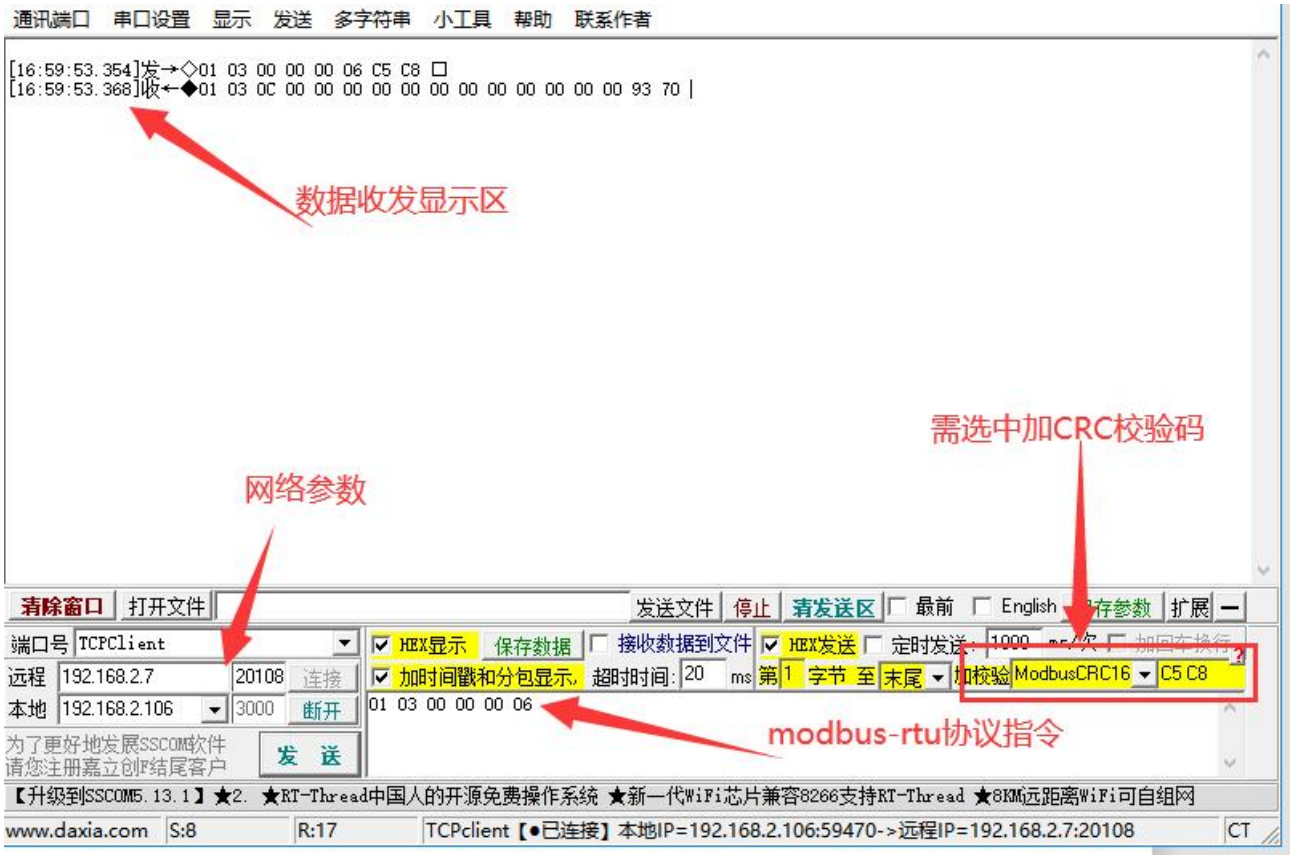


图 4、modbus-rtu 协议指令测试页面

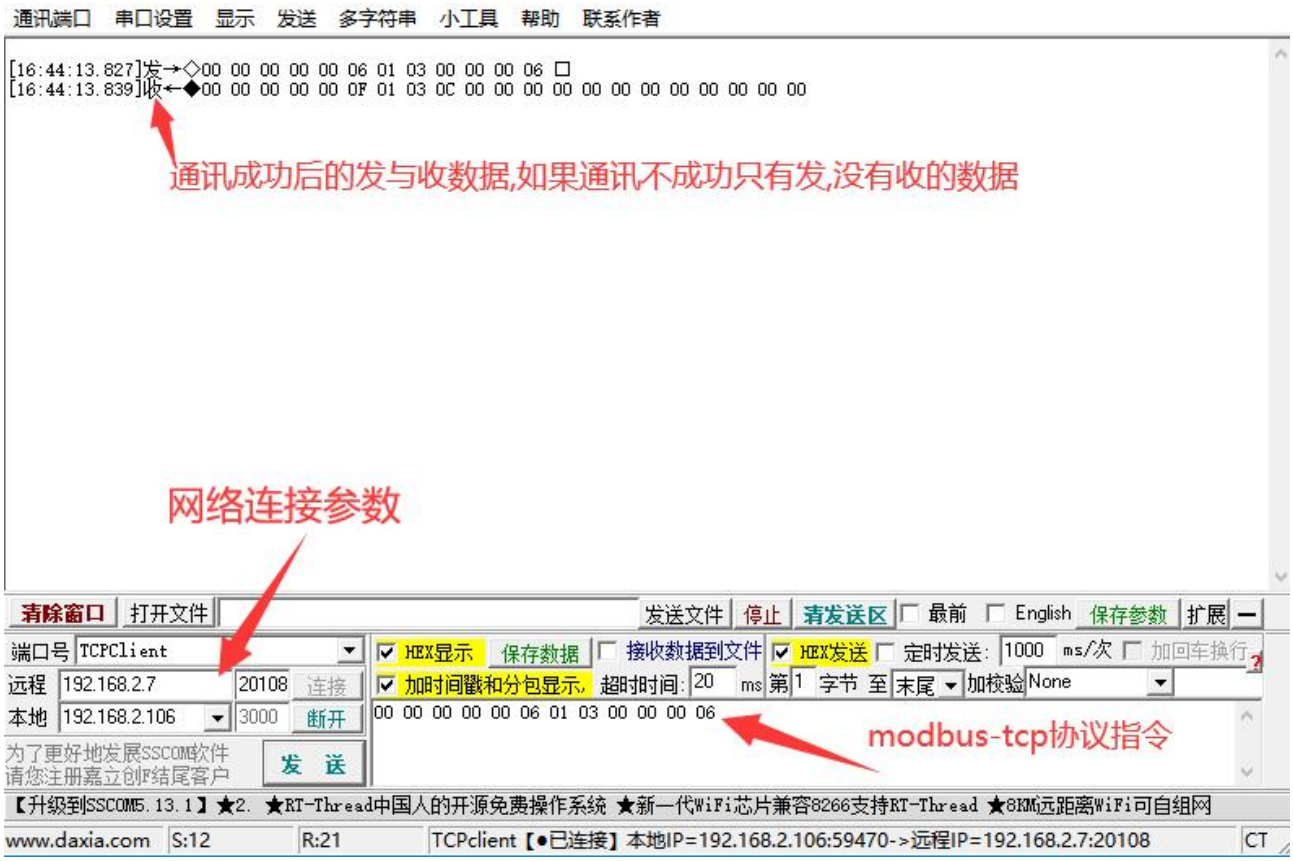


图 5、modbus-tcp 协议指令测试页面

附件 5：

CAN 通讯协议

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。规则如下：

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；

因为 CAN 每帧只能上传 8 个字节，温度为 16 位数据，占用两个字节，所以需要两帧才可以上传完 8 个通道的温度，所以用属于扩展帧里的 18 位 ID 定义每次上传的数据包的帧数与帧序。

数据段则按顺序放置了采集到的温度数据。列表格如下：

第一帧：

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 0x0060 寄存改变此值，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。	固定为 07	数据包的帧数，在这里固定为 0x02	此帧在数据包中的序号，第 1 帧为 0x01

CAN 数据段（硬件按字节从左至右发送）									
每帧数据上传 4 路温度数据，每路温度占用两个字节									
字节序号	DLC	1	2	3	4	5	6	7	8
内容	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

第二帧：

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
同第 1 帧	同第 1 帧	同第 1 帧	同第 1 帧	此帧在数据包中的序号，第 2 帧为 0x02

CAN 数据段（硬件按字节从左至右发送）									
每帧数据上传 4 路温度数据，每路温度占用两个字节									
字节序号	DLC	1	2	3	4	5	6	7	8
内容	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

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 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：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 0x0061 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； 03 指令会返回至少 1 帧以上数据。									
帧顺序	数据长度 DLC	数据字节							
		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。

06 指令　CAN 数据段（硬件按字节从左至右发送）									
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，06 指令可能有 2 至 127 帧数据									
帧顺序	数据长度 DLC	数据字节							
		1	2	3	4	5	6	7	8
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，每帧都必须相同。	受控模块设备地址，同一次指令，每帧都必须相同。	需修改的起始寄存器地址高 8 位	需修改的起始寄存器地址低 8 位	需修改的寄存器数量，最大 255	空	空	空
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			被修改的第 1 个寄存器的值高 8 位	被修改的第 1 个寄存器的值低 8 位	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.....			
第 3 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，直到所有寄存器数值发完					
.....	.....			.....依次排放寄存器数据，直到所有寄存器数值发完					

(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）。

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：06	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；	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。

06 指令返回数据　CAN 数据段（硬件按字节从左至右发送）

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。

帧顺序	数据长度 DLC	数据字节							
		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。	被修改的第 1 个寄存器值高 8 位	被修改的第 1 个寄存器值低 8 位	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.....			
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