S3600

4 路 3 线制 RTD 采集模块

使用说明书

上海世杰电子有限公司

销售：**michael@shjelectronic.com**

技术支持：**support@shjelectronic.com**

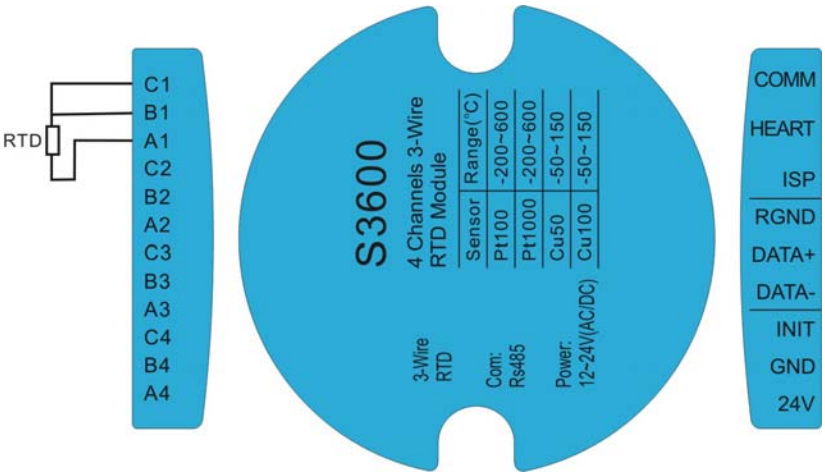
一、概述

S3600 是一款中高档的模拟量采集模块，具有 4 通道三线制热电阻模拟量输入，采用 100k 采样速率的 12 位 AD 转换器，每输入通道有防雷、静电保护，默认输入时 PT100，可根据客户传感器调整，测温范围为-200 到 600 摄氏度，有 17 个校准点，可以根据客户测温范围任意调整，达到最佳校准效果，比如客户需要 50 到 200 度，那么 17 个校准点都分配给 50 到 200 度。输出总线为 RS485，，输出有防雷、静电保护。设计上还通过使用外部看门狗，表面贴装工艺和单点共地技术提高系统稳定性。输出协议为 Modbus RTU,可以和组态软件和 PLC 通讯。

二、技术参数

分辨率----------16 位
输入通道----------4
输入信号---------- Cu50,Cu100,PT100,PT1000
输入保护----------防雷，静电
准确度----------±0.1%
零漂移---------- ±3uV/℃
采集速率----------95 次/秒（16 通道），710 次/秒（1 通道）。【通道数可配置】
输出----------RS485
输出保护----------防雷，静电
电源---------- 12~24V(AC/DC),标准 24VAC
功耗---------- <0.6W
工作温度---------- -20~85℃(-13~185℉)
存储温度---------- -40~100℃(-40~212℉)
相对湿度---------- 5%~95%RH (不凝露)
尺寸----------100*69*25mm

三、接线说明



接线 1

1、 输入

C1~C4: 见接线 1

B1~B4: 见接线 1

A1~A4: 见接线 1

2、 电源

直流：24V 接正极

GND 接负极

注： 有反接保护

交流：不分正负极

3、 RS485 输出

DATA+接 485 总线 A 端

DATA-接 485 总线 B 端

RGND:接大地或悬空

4、 参数复位

跳线跳在 INIT 和 GND 端，下面这些参数恢复为出厂值。

- 地址：254
- 波特率：19200
- 通道：使能所有通道
- 单位：AD 采集原始数据值
- 滤波系数：10

5、 人机界面

Heart： 系统工作时这个 LED 闪烁，代表活着。

Comm：通讯时这个 LED 闪烁

四、寄存器列表

注：带*号的数值为出厂值。

地址	字节数	数值范围		描述	属性
		最小值	最大值		
0-3	4	1	4294967295	产品序列号，每个产品唯一。	只读
4-5	2	100	65535	固件版本号	只读
6	1	1	254	MODBUS 通讯地址，254*为出厂值。	读写
7	2	3600	3600	产品型号	只读
8	1	1	255	硬件版本号	只读
9	2	12	1152	波特率设置寄存器.	读写

				数值 / 波特率	
				12 / 1200	
				24 / 2400	
				48 / 4800	
				96 / 9600	
				192* / 19200	
				384 / 38400	
				576 / 57600	
				1152 / 115200	
10-99	-	-	-	保留	-
100-103	2	0	65535	输入 1 到 4 的读数	读写
104~107	2	0	2	输入 1 到 4 的单位设置。0 = 原始 ADC 采样数据,1 = 摄氏度,2 = 华氏度	读写
108~111	-	-	-	Reserved	读写
112~115	-	-	-	Reserved	读写
116	2	0	65535	输入 1 的第 1 个校准点，原始 ADC 数据	读写
117	2	0	65535	输入 1 的第 1 个校准点，电阻值，乘以 100	读写
118	2	0	65535	输入 1 的第 2 个校准点，原始 ADC 数据	读写
119	2	0	65535	输入 1 的第 2 个校准点，电阻值，乘以 100	读写
120	2	0	65535	输入 1 的第 3 个校准点，原始 ADC 数据	读写
121	2	0	65535	输入 1 的第 3 个校准点，电阻值，乘以 100	读写
122	2	0	65535	输入 1 的第 4 个校准点，原始 ADC 数据	读写
123	2	0	65535	输入 1 的第 4 个校准点，电阻值，乘以 100	读写
124	2	0	65535	输入 1 的第 5 个校准点，原始 ADC 数据 ata	读写
125	2	0	65535	输入 1 的第 5 个校准点，电阻值，乘以 100	读写
126	2	0	65535	输入 1 的第 6 个校准点，原始 ADC 数据	读写
127	2	0	65535	输入 1 的第 6 个校准点，电阻值，乘以 100	读写
128	2	0	65535	输入 1 的第 7 个校准点，原始 ADC 数据	读写
129	2	0	65535	输入 1 的第 7 个校准点，电阻值，乘以 100	读写
130	2	0	65535	输入 1 的第 8 个校准点，原始 ADC 数据	读写
131	2	0	65535	输入 1 的第 8 个校准点，电阻值，乘以 100	读写
132	2	0	65535	输入 1 的第 9 个校准点，原始 ADC 数据	读写
133	2	0	65535	输入 1 的第 9 个校准点，电阻值，乘以 100	读写
134	2	0	65535	输入 1 的第 10 个校准点，原始 ADC 数据	读写
135	2	0	65535	输入 1 的第 10 个校准点，电阻值，乘以 100	读写
136	2	0	65535	输入 1 的第 11 个校准点，原始 ADC 数据	读写
137	2	0	65535	输入 1 的第 11 个校准点，电阻值，乘以 100	读写
138	2	0	65535	输入 1 的第 12 个校准点，原始 ADC 数据	读写

139	2	0	65535	输入 1 的第 12 个校准点，电阻值，乘以 100	读写
140	2	0	65535	输入 1 的第 13 个校准点，原始 ADC 数据	读写
141	2	0	65535	输入 1 的第 13 个校准点，电阻值，乘以 100	读写
142	2	0	65535	输入 1 的第 14 个校准点，原始 ADC 数据 ta	读写
143	2	0	65535	输入 1 的第 14 个校准点，电阻值，乘以 100	读写
144	2	0	65535	输入 1 的第 15 个校准点，原始 ADC 数据	读写
145	2	0	65535	输入 1 的第 15 个校准点，电阻值，乘以 100	读写
146	2	0	65535	输入 1 的第 16 个校准点，原始 ADC 数据	读写
147	2	0	65535	输入 1 的第 16 个校准点，电阻值，乘以 100	读写
148	2	0	65535	输入 1 的第 17 个校准点，原始 ADC 数据	读写
149	2	0	65535	输入 1 的第 17 个校准点，电阻值，乘以 100	读写
					读写
150	2	0	65535	输入 2 的第 1 个校准点，原始 ADC 数据	读写
151	2	0	65535	输入 2 的第 1 个校准点，电阻值，乘以 100	读写
152	2	0	65535	输入 2 的第 2 个校准点，原始 ADC 数据	读写
153	2	0	65535	输入 2 的第 2 个校准点，电阻值，乘以 100	读写
154	2	0	65535	输入 2 的第 3 个校准点，原始 ADC 数据	读写
155	2	0	65535	输入 2 的第 3 个校准点，电阻值，乘以 100	读写
156	2	0	65535	输入 2 的第 4 个校准点，原始 ADC 数据	读写
157	2	0	65535	输入 2 的第 4 个校准点，电阻值，乘以 100	读写
158	2	0	65535	输入 2 的第 5 个校准点，原始 ADC 数据 ata	读写
159	2	0	65535	输入 2 的第 5 个校准点，电阻值，乘以 100	读写
160	2	0	65535	输入 2 的第 6 个校准点，原始 ADC 数据	读写
161	2	0	65535	输入 2 的第 6 个校准点，电阻值，乘以 100	读写
162	2	0	65535	输入 2 的第 7 个校准点，原始 ADC 数据	读写
163	2	0	65535	输入 2 的第 7 个校准点，电阻值，乘以 100	读写
164	2	0	65535	输入 2 的第 8 个校准点，原始 ADC 数据	读写
165	2	0	65535	输入 2 的第 8 个校准点，电阻值，乘以 100	读写
166	2	0	65535	输入 2 的第 9 个校准点，原始 ADC 数据	读写
167	2	0	65535	输入 2 的第 9 个校准点，电阻值，乘以 100	读写
168	2	0	65535	输入 2 的第 10 个校准点，原始 ADC 数据	读写
169	2	0	65535	输入 2 的第 10 个校准点，电阻值，乘以 100	读写
170	2	0	65535	输入 2 的第 11 个校准点，原始 ADC 数据	读写
171	2	0	65535	输入 2 的第 11 个校准点，电阻值，乘以 100	读写
172	2	0	65535	输入 2 的第 12 个校准点，原始 ADC 数据	读写
173	2	0	65535	输入 2 的第 12 个校准点，电阻值，乘以 100	读写

174	2	0	65535	输入 2 的第 13 个校准点，原始 ADC 数据	读写
175	2	0	65535	输入 2 的第 13 个校准点，电阻值，乘以 100	读写
176	2	0	65535	输入 2 的第 14 个校准点，原始 ADC 数据 ta	读写
177	2	0	65535	输入 2 的第 14 个校准点，电阻值，乘以 100	读写
178	2	0	65535	输入 2 的第 15 个校准点，原始 ADC 数据	读写
179	2	0	65535	输入 2 的第 15 个校准点，电阻值，乘以 100	读写
180	2	0	65535	输入 2 的第 16 个校准点，原始 ADC 数据	读写
181	2	0	65535	输入 2 的第 16 个校准点，电阻值，乘以 100	读写
182	2	0	65535	输入 2 的第 17 个校准点，原始 ADC 数据	读写
183	2	0	65535	输入 2 的第 17 个校准点，电阻值，乘以 100	读写
					读写
184	2	0	65535	输入 3 的第 1 个校准点，原始 ADC 数据	读写
185	2	0	65535	输入 3 的第 1 个校准点，电阻值，乘以 100	读写
186	2	0	65535	输入 3 的第 2 个校准点，原始 ADC 数据	读写
187	2	0	65535	输入 3 的第 2 个校准点，电阻值，乘以 100	读写
188	2	0	65535	输入 3 的第 3 个校准点，原始 ADC 数据	读写
189	2	0	65535	输入 3 的第 3 个校准点，电阻值，乘以 100	读写
190	2	0	65535	输入 3 的第 4 个校准点，原始 ADC 数据	读写
191	2	0	65535	输入 3 的第 4 个校准点，电阻值，乘以 100	读写
192	2	0	65535	输入 3 的第 5 个校准点，原始 ADC 数据 ata	读写
193	2	0	65535	输入 3 的第 5 个校准点，电阻值，乘以 100	读写
194	2	0	65535	输入 3 第 6 个校准点，原始 ADC 数据	读写
195	2	0	65535	输入 3 的第 6 个校准点，电阻值，乘以 100	读写
196	2	0	65535	输入 3 的第 7 个校准点，原始 ADC 数据	读写
197	2	0	65535	输入 3 的第 7 个校准点，电阻值，乘以 100	读写
198	2	0	65535	输入 3 的第 8 个校准点，原始 ADC 数据	读写
199	2	0	65535	输入 3 的第 8 个校准点，电阻值，乘以 100	读写
200	2	0	65535	输入 3 的第 9 个校准点，原始 ADC 数据	读写
201	2	0	65535	输入 3 的第 9 个校准点，电阻值，乘以 100	读写
202	2	0	65535	输入 3 的第 10 个校准点，原始 ADC 数据	读写
203	2	0	65535	输入 3 的第 10 个校准点，电阻值，乘以 100	读写
204	2	0	65535	输入 3 的第 11 个校准点，原始 ADC 数据	读写
205	2	0	65535	输入 3 的第 11 个校准点，电阻值，乘以 100	读写
206	2	0	65535	输入 3 的第 12 个校准点，原始 ADC 数据	读写
207	2	0	65535	输入 3 的第 12 个校准点，电阻值，乘以 100	读写
208	2	0	65535	输入 3 的第 13 个校准点，原始 ADC 数据	读写

209	2	0	65535	输入 3 的第 13 个校准点，电阻值，乘以 100	读写
210	2	0	65535	输入 3 的第 14 个校准点，原始 ADC 数据 ta	读写
211	2	0	65535	输入 3 的第 14 个校准点，电阻值，乘以 100	读写
212	2	0	65535	输入 3 的第 15 个校准点，原始 ADC 数据	读写
213	2	0	65535	输入 3 的第 15 个校准点，电阻值，乘以 100	读写
214	2	0	65535	输入 3 的第 16 个校准点，原始 ADC 数据	读写
215	2	0	65535	输入 3 的第 16 个校准点，电阻值，乘以 100	读写
216	2	0	65535	输入 3 的第 17 个校准点，原始 ADC 数据	读写
217	2	0	65535	输入 3 的第 17 个校准点，电阻值，乘以 100	读写
					读写
218	2	0	65535	输入 4 的第 1 个校准点，原始 ADC 数据	读写
219	2	0	65535	输入 4 的第 1 个校准点，电阻值，乘以 100	读写
220	2	0	65535	输入 4 的第 2 个校准点，原始 ADC 数据	读写
221	2	0	65535	输入 4 的第 2 个校准点，电阻值，乘以 100	读写
222	2	0	65535	输入 4 的第 3 个校准点，原始 ADC 数据	读写
223	2	0	65535	输入 4 的第 3 个校准点，电阻值，乘以 100	读写
224	2	0	65535	输入 4 的第 4 个校准点，原始 ADC 数据	读写
225	2	0	65535	输入 4 的第 4 个校准点，电阻值，乘以 100	读写
226	2	0	65535	输入 4 的第 5 个校准点，原始 ADC 数据 ata	读写
227	2	0	65535	输入 4 的第 5 个校准点，电阻值，乘以 100	读写
228	2	0	65535	输入 4 的第 6 个校准点，原始 ADC 数据	读写
229	2	0	65535	输入 4 的第 6 个校准点，电阻值，乘以 100	读写
230	2	0	65535	输入 4 的第 7 个校准点，原始 ADC 数据	读写
231	2	0	65535	输入 4 的第 7 个校准点，电阻值，乘以 100	读写
232	2	0	65535	输入 4 的第 8 个校准点，原始 ADC 数据	读写
233	2	0	65535	输入 4 第 8 个校准点，电阻值，乘以 100	读写
234	2	0	65535	输入 4 的第 9 个校准点，原始 ADC 数据	读写
235	2	0	65535	输入 4 的第 9 个校准点，电阻值，乘以 100	读写
236	2	0	65535	输入 4 的第 10 个校准点，原始 ADC 数据	读写
237	2	0	65535	输入 4 的第 10 个校准点，电阻值，乘以 100	读写
238	2	0	65535	输入 4 的第 11 个校准点，原始 ADC 数据	读写
239	2	0	65535	输入 4 的第 11 个校准点，电阻值，乘以 100	读写
240	2	0	65535	输入 4 的第 12 个校准点，原始 ADC 数据	读写
241	2	0	65535	输入 4 的第 12 个校准点，电阻值，乘以 100	读写
242	2	0	65535	输入 4 的第 13 个校准点，原始 ADC 数据	读写
243	2	0	65535	输入 4 的第 13 个校准点，电阻值，乘以 100	读写

244	2	0	65535	输入 4 的第 14 个校准点，原始 ADC 数据 ta	读写
245	2	0	65535	输入 4 的第 14 个校准点，电阻值，乘以 100	读写
246	2	0	65535	输入 4 的第 15 个校准点，原始 ADC 数据	读写
247	2	0	65535	输入 4 的第 15 个校准点，电阻值，乘以 100	读写
248	2	0	65535	输入 4 的第 16 个校准点，原始 ADC 数据	读写
249	2	0	65535	输入 4 的第 16 个校准点，电阻值，乘以 100	读写
250	2	0	65535	输入 4 的第 17 个校准点，原始 ADC 数据	读写
251	2	0	65535	输入 4 的第 17 个校准点，电阻值，乘以 100	读写
252	1	0	2	写入 1，通道 1 的所有校准点回到出厂值	读写
253	1	0	2	写入 1，通道 2 的所有校准点回到出厂值	读写
254	1	0	2	写入 1，通道 3 的所有校准点回到出厂值	读写
255	1	0	2	写入 1，通道 4 的所有校准点回到出厂值	读写
256	1	1	255	Modbus 响应延时，单位是 2.5ms	读写

默认设置:

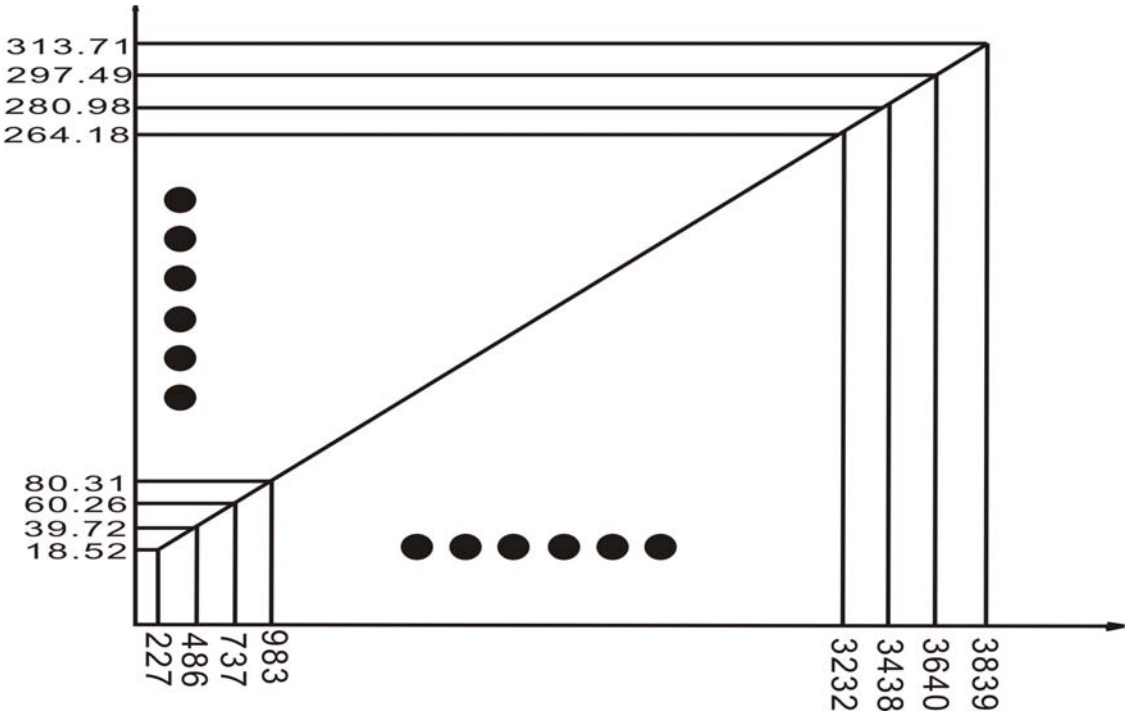
设备 ID: 254, 255 是广播地址

数据格式 1-8-N-1

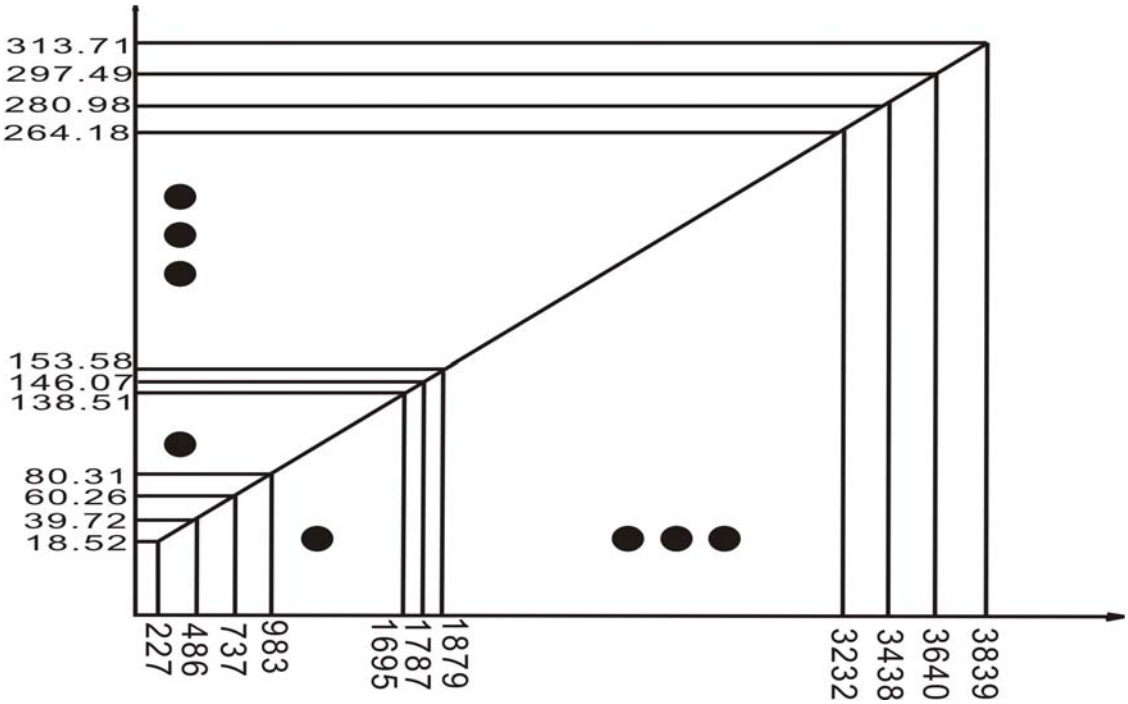
波特率: 19200

五、校准

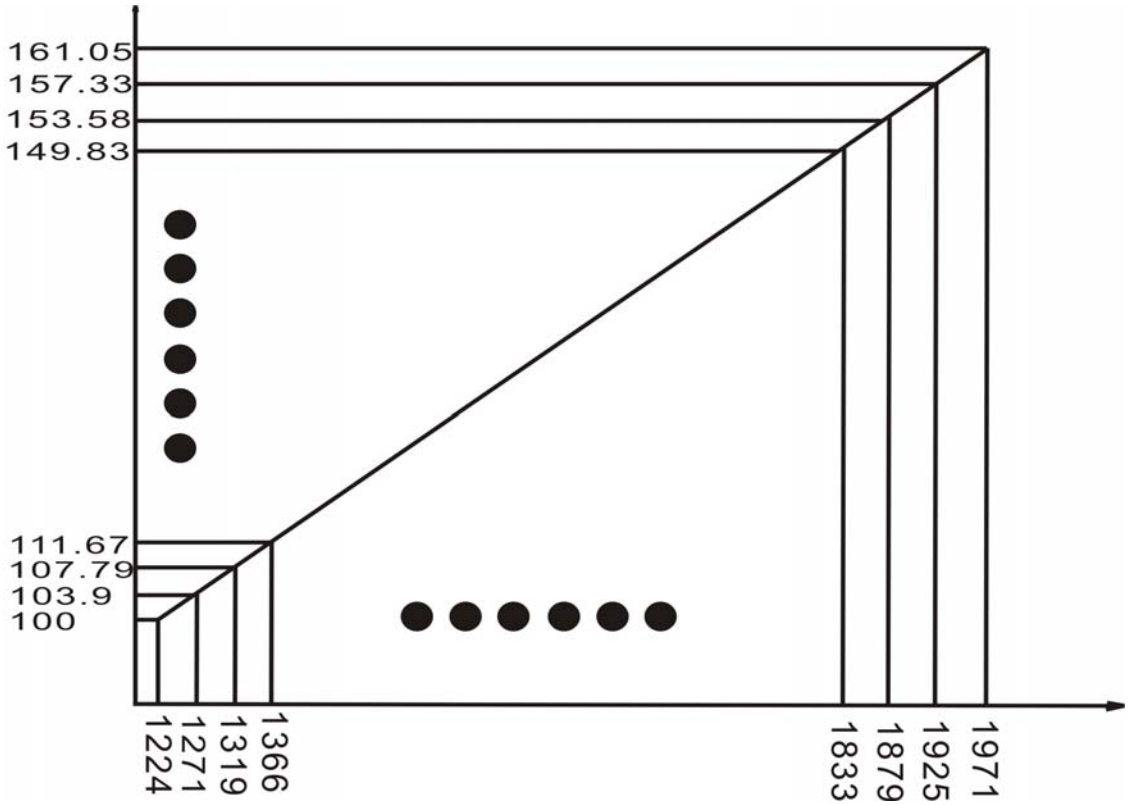
S3600 有 17 个校准点，对应温度范围为-200 到 600 摄氏度，每 50 摄氏度对应一个校准点，对应电阻值为 18.72 到 313.71.每个校准点包含电阻值和对应的原始 ADC 采样值，如下图。



校准点间隔不是固定 50 度，如果需要在 100 到 140 度得到精准值，可以 20 度一格。



或 10 摄氏度一格。



六 MODBUS 通信规约

概述

ModBus 协议是 Modicon 公司于 1978 年发明的一种用于电子控制器进行控制和通讯的通讯协议。通过此协议，控制器相互之间、控制器经由网络（例如以太网）和其它设备之间可以进行通信。它的开放性、可扩充性和标准化使它成为一个通用工业标准。ModBus 有 27 种命令，SHJ-3100 只用了 READ,WRITE 两种 ，物理层为 RS485 或 RS232, 串口数据格式为一个起始位，8 个数据位，1 个停止位。

ModBus 标准数据格式为：

字节 1：从节点地址，地址范围为 1-254，255 为广播地址
字节 2：命令，读或写
字节 3：读或写寄存器起始地址的高字节
字节 4：读或写寄存器起始地址的低字节
字节 5：读或写寄存器数据长度的高字节
字节 6：读或写寄存器数据长度的低字节
字节 7：CRC 高字节
字节 8：CRC 低字节

命令示例：

1、读命令（0x03）

这个命令用来读取多个寄存器的内容，主节点需要指明要操作的从节点的地址，起始寄存器地址和要读取寄存器的个数。如果寄存器内容是整型，则高字节在前，低字节在后。例：读取从节点 18，起始寄存器为 100，读 3 个寄存器，主节点应发送如下数据.

字节 1：从节点地址	0x12
字节 2：读命令字	0x03
字节 3：寄存器起始地址的高字节	0x00
字节 4：寄存器起始地址的低字节	0x64
字节 5：寄存器个数的高字节	0x00
字节 6：寄存器个数的低字节	0x03
字节 7：CRC 校验高字节	0x46
字节 8：CRC 校验低字节	0xb7

从节点在几毫秒内返回如下数据。

字节 1： 从节点地址	0x12
字节 2： 读命令字	0x03
字节 3 ：数据个数（寄存器数*2）	0x06
字节 4： 数据 1 的高字节	0xff
字节 5： 数据 1 的低字节	0xff
字节 6： 数据 2 的高字节	0xff
字节 7： 数据 2 的低字节	0xff
字节 8： 数据 3 的高字节	0xff
字节 9： 数据 3 的低字节	0xff

字节 10：CRC 的高字节 0xXX
字节 11：CRC 的低字节 0xXX

2、写命令（0x06）

这个命令用来向单个寄存器写入数据， 主节点需要指明要操作的从节点的地址，寄存器地址和要写入的数据。 例：写从节点 18，寄存器为 100，数据为 512，主节点应发送如下数据.

字节 1：从节点地址 0x12
字节 2：写命令字 0x06
字节 3：寄存器地址的高字节 0x00
字节 4：寄存器地址的低字节 0x64
字节 5：写入数据的高字节 0x02
字节 6：写入数据的低字节 0x00
字节 7：CRC 校验高字节 0xcb
字节 8：CRC 校验低字节 0xd6

从节点在几毫秒内返回如下数据。

字节 1：从节点地址 0x12
字节 2：写命令字 0x06
字节 3：寄存器地址的高字节 0x00
字节 4：寄存器地址的低字节 0x64
字节 5：写入数据的高字节 0x02
字节 6：写入数据的低字节 0x00
字节 7：CRC 校验高字节 0xcb
字节 8：CRC 校验低字节 0xd6

从节点返回数据和发送数据相同，代表成功收到数据。

CRC 校验

下面表格为 ModBus 的 CRC 校验查找表，为了帮助软件工程师快速完成 CRC 程序编写，我们提供示例程序，有需要请通知我们，我们会把如下代码发给你。

CRC 高字节查找表

```
static unsigned char auchCRCHi[ ] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
```

```
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;
```

CRC 低字节查找表

```
static unsigned char auchCRCLo[ ] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
} ;
```

例：计算存储在*puchMsg 里的 usDataLen 个数据的 CRC.

```
unsigned short CRC16 (unsigned char *puchMsg, unsigned char usDataLen)
{
        unsigned char uchCRCHi = 0xFF ; /* CRC 高字节初始化  */
        unsigned char uchCRCLo = 0xFF ; /* CRC 低字节初始化*/
        unsigned uIndex ;
        while (usDataLen--)
        {
```

```
            uIndex = uchCRCHi ^ *puchMsg++ ; /* calculate the CRC */
            uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
            uchCRCLo = auchCRCLo[uIndex] ;
        }
        return (uchCRCHi << 8 | uchCRCLo) ;
}
```