上海协堡电子有限公司 ShangHai ShinePoin Electronics Co.,Ltd.
上海市闵行区新骏环路115号寰启商务大厦1幢906室 Tel：021-50933055 Fax：021-51686291

上海协堡电子系列传感器 ModBus 通讯协议

1. 技术参数：

1） 总线接口：RS485/RS422/RS232/TTL；
2） 默认波特率：19200；1 个起始位，8 位数据位，2 个停止位，奇偶校验：无；
3） 从机地址范围：1~247，地址 0 为广播；
4） 功能码：1～127 之间；
5） 数据格式：RTU；
6） 校验方式：循环冗余 CRC 校验；
7） 数据帧结束时间：>3.5 个字符；帧异常：数据流字符间传输的时间在 1.5~3.5 个字符之间；

2. 读保持寄存器（读数据，功能码 0x03）：

主机发送：

从机地址	0x03	寄存器起始地址（2 个字节）	读取寄存器个数（2 个字节）	CR 校验低字节	CRC 校验高字节

如果正确从机响应：

从机地址	0x03	返回字节个数	寄存器数据	CRC 校验低字节	CRC 校验高字节

如果错误从机响应：

从机地址	0x83	错误码	CRC 校验低字节	CRC 校验高字节

3. 写单个寄存器（写参数、启动或者关闭测量，功能码 0x06，一次只能写一个寄存器）：

主机发送：

从机地址	0x06	寄存器起始地址（2 个字节）	写入的数据(2 个字节)	CR 校验低字节	CRC 校验高字节

正确从机响应（原样返回）：

从机地址	0x06	寄存器起始地址（2 个字节）	已写入的数据（2 个字节）	CR 校验低字节	CRC 校验高字节

错误从机响应：

从机地址	0x86	写入的字节数（固定为 2）	CR 校验低字节	CRC 校验高字节

4. 写多个寄存器（从起始地址开始，一次写入多个参数，功能码 0x10）：

主机发送：

从机地址	0x10	寄存器起始地址（高低位 2 个字节）	寄存器的数量（高低位 2 个字节，写多少个寄存器）	字节数（一个字节，寄存器的数量乘以 2）	待写入的数据高低位 n（2 个字节）	CR 校验低字节	CRC 校验高字节

正确从机响应：

从机地址	0x10	寄存器起始地址（高低位 2 个字节）	寄存器的数量高低位	CR 校验低字节	CRC 校验高字节

5. 异常返回：

从机地址	功能码|0x80	异常码	CRC 校验低字节	CRC 校验高字节

6. 异常码：

a. 0x01：功能码非法；
b. 0x02：本机地址非法（保留选项，本传感器中如果首地址和本机地址不一致，传感器不返回异常码）；
c. 0x03：寄存器数量非法；
d. 0x04：数据非法；
e. 0x06：寄存器地址非法；
f. 0x07：CRC 校验错误
g. 0x08：保留；

7. 读寄存器对应地址（功能码 0x03，地址从 100 开始，即 x064）：

寄存器地址	说明		数据
0x0064	读传感器地址		2Byte
0x0066	读模拟量下限高 16 位	获取 4～20mA 或者 0～5V 模拟量对应量程	2Byte
0x0068	读模拟量下限低 16 位		2Byte
0x006A	读模拟量上限高 16 位		2Byte
0x006C	读模拟量上限低 16 位		2Byte
0x006E	读模拟量输出方式		2Byte
0x0070	距离值自动测量时间高 16 位		2Byte
0x0072	距离值自动测量时间低 16 位		2Byte
0x0074	读取参考点信息		2Byte
0x0076	读取开关量输出 1 触发值高 16 位		2Byte
0x0078	读取开关量输出 1 触发值低 16 位		2Byte
0x007A	读取开关量输出 2 触发值高 16 位		2Byte
0x007C	读取开关量输出 2 触发值低 16 位		2Byte
0x007E	读取开关量输出 3 触发值高 16 位		2Byte
0x0080	读取开关量输出 3 触发值低 16 位		2Byte
0x0082	读取开关量输出 4 触发值高 16 位		2Byte
0x0084	读取开关量输出 4 触发值低 16 位		2Byte
0x0086	读取修正值符号		2Byte 符号位
0x0088	读取修正值高 16 位		2Byte
0x008A	读取修正值低 16 位		2Byte
0x008C	读取上电测量模式		2Byte
0x008E	读取内部风扇工作模式		2Byte
0x0090	读传感器的状态		2Byte
0x0092	读通讯波特率		2Byte
0x0094	读取距离值高 16 位		2Byte
0x0096	读取距离值低 16 位		2Byte
0x0098	读取产品序列号，共 16Byte		2Byte
0x009A	读取产品序列号，共 16Byte		2Byte
0x009C	读取产品序列号，共 16Byte		2Byte
0x009E	读取产品序列号，共 16Byte		2Byte
0x00A0	读取产品序列号，共 16Byte		2Byte
0x00A2	读取产品序列号，共 16Byte		2Byte
0x00A4	读取产品序列号，共 16Byte		2Byte
0x00A6	读取产品序列号，共 16Byte		2Byte

8. 写寄存器对应地址（功能码 0x06 或者 0x10，地址从 356 开始，即 0x0164）：

寄存器地址	说明		数据
0x0164	设置传感器地址		2Byte
0x0166	设置模拟量下限高 16 位	设置 4～20mA 或者 0～5V 模拟量对应量程	2Byte
0x0168	设置模拟量下限低 16 位		2Byte
0x016A	设置模拟量上限高 16 位		2Byte
0x016C	设置模拟量上限低 16 位		2Byte
0x016E	设置模拟量输出方式		2Byte
0x0170	设置距离值自动测量时间高 16 位		2Byte
0x0172	设置距离值自动测量时间低 16 位		2Byte
0x0174	设置取参考点信息		2Byte
0x0176	设置开关量输出 1 触发值高 16 位		2Byte
0x0178	设置开关量输出 1 触发值低 16 位		2Byte
0x017A	设置开关量输出 2 触发值高 16 位		2Byte
0x017C	设置开关量输出 2 触发值低 16 位		2Byte
0x017E	设置开关量输出 3 触发值高 16 位		2Byte
0x0180	设置开关量输出 3 触发值低 16 位		2Byte
0x0182	设置开关量输出 4 触发值高 16 位		2Byte
0x0184	设置开关量输出 4 触发值低 16 位		2Byte
0x0186	设置修正值符号		2Byte
0x0188	设置修正值高 16 位		2Byte
0x018A	设置修正值低 16 位		2Byte
0x018C	设置上电测量模式		2Byte
0x018E	设置内部风扇工作模式		2Byte
0x0190	测量控制		2Byte
0x**0192**	设置波特率		2Byte

9. 应用例程：

1）启动一次单次测量（传感器地址为 1）：

主机发送（16 进制）：

01 06 01 90 00 01 49 DB

其中： 0x01 为传感器地址；
0x06 为功能码，表示写单个寄存器；
0x01 和 0x90 为本次写入的寄存器地址（测量控制寄存器）0x0190；
0x00 和 0x01 为写入的数据；值为 1 表示单次测量，每次写 1 后将自动清零；
值为 2 表示连续测量；值为 0 表示关机，其它值无效。
0x49 和 0xDB 为 2 个字节的 CRC 校验码；

传感器收到正确的指令（同时传感器启动一次测量）将返回（16 进制）：

01 06 01 90 00 01 49 DB

2）启动一次连续测量（传感器地址为 1）

主机发送（16 进制）：

01 06 01 90 00 02 09 DA

其中： 0x01 为传感器地址；

0x06 为功能码，表示写单个寄存器；

0x01 和 0x90 为本次写入的寄存器地址（测量控制寄存器）0x0190；

0x00 和 0x02 为写入的数据；值为 1 表示单次测量，每次写 1 后将自动清零；值为 2 表示连续测量；值为 0 表示关机，其它值无效。

0x09 和 0xDA 为 2 个字节的 CRC 校验码；

传感器收到正确的指令（同时传感器启动连续测量）将返回（16 进制）：

01 06 01 90 00 02 09 DA

3）关机（传感器地址为 1）

主机发送（16 进制）：

01 06 01 90 00 00 88 1B

其中： 0x01 为传感器地址；

0x06 为功能码，表示写单个寄存器；

0x01 和 0x90 为本次写入的寄存器地址（测量控制寄存器）0x0190；

0x00 和 0x00 为写入的数据；值为 1 表示单次测量，每次写 1 后将自动清零；值为 2 表示连续测量；值为 0 表示关机，其它值无效。

0x88 和 0x1B 为 2 个字节的 CRC 校验码；

传感器收到正确的指令将返回（16 进制）：

01 06 01 90 00 00 88 1B

4）读测量到的距离值例程（传感器地址为 1）：

主机发送（16 进制）：

01 03 00 94 00 02 85 E7

其中： 0x01 为传感器地址；

0x03 为功能码，表示读保持寄存器；

0x00 和 0x94 为要读取寄存器首地址（距离值寄存器）0x0094；

0x00 和 0x02 为本次读寄存器的个数，为 2 个寄存器；modbus 规定一个寄存器的长度是 16bit；

0x85 和 0XE7 为 2 个字节的 CRC 校验码；

传感器收到正确的指令将返回（16 进制）：

01 03 04 00 00 0B 10 FC CF

其中： 0x01 为传感器地址；

0x03 为返回的功能码；

0x04 表示返回 4 个 Byte；

0x00、0x00、0x0B、0x10 为返回的保持寄存器距离值，共 4 个字节，0x00000B10，转换成十进制就是 2832mm，即 2.832 米；

0xFC 和 0xCF 为 2 个字节的 CRC 校验码；

备注：如果传感器的分辨率是 0.1mm，从寄存器中读出的距离值需要除以 10 后距离值的单位才是 mm。

5）设置上电测量模式（传感器地址为 1）：

主机发送（16 进制）：

01 06 01 8C 00 01 88 1D

其中： 0x01 为传感器地址；

0x06 为功能码，表示写单个寄存器；

0x01 和 0x8C 为本次写入的寄存器地址（上电自动测量控制寄存器）0x018C；

0x00 和 0x01 为写入的数据；值为 1 表示上电进入自动测量；值为 0 表示不上电自动测量；数据写入后，掉电自动保存到传感器内部存储器中；

0x88 和 0x1D 为 2 个字节的 CRC 校验码；

传感器收到正确的指令（同时传感器启动一次测量）将返回（16 进制）：

01 06 01 8C 00 01 88 1D

备注：设置上电自动测量后，传感器进入自动测量模式，自动刷新保持寄存器中的值，可用 9.4 中的例程读取最新测量的结果。

6）设置修正值（传感器地址为 1）：

如：传感器测量到的值是 1008mm，实际值是 1000mm，需要将每次测量到的结果减去 8mm；

主机发送（16 进制）：

01 10 01 86 00 03 06 00 2D 00 00 00 08 A8 4B

其中： 0x01 为传感器地址；

0x10 为功能码，表示写多个寄存器；

0x01 和 0x86 为设置修正值寄存器的首地址；

0x00 和 0x03 为本次写入 3 个寄存器；

0x06 表示 6 个字节（3 个寄存器就是 6 个字节）；

0x00 和 0x2D 为本次写入修正值的符号，0x2D 表示减，0x2B 表示加；

最后 0x00、0x00、0x00、0x08 分别为修正值的高 16 位和低 16 位，0x00000008 表示 8mm；

0xA8 和 0x4B 为 2 个字节的 CRC 校验码；

传感器收到正确的指令（同时将修正值保存在传感器内部）将返回（16 进制）：

01 10 01 86 00 03 60 1D

7）设置 4~20mA 对应量程（传感器地址为 1）：

如：需要设置 4~20mA 对应 0~10000mm；

主机发送（16 进制）：

01 10 01 66 00 04 08 00 00 00 00 00 00 27 10 26 4F

其中： 0x01 为传感器地址；

0x10 为功能码，表示写多个寄存器；

0x01 和 0x66 为设置量程寄存器的首地址；

0x00 和 0x04 为本次写入 4 个寄存器；

0x08 表示 8 个字节（4 个寄存器就是 8 个字节）；

0x00、0x00、0x00、0x00、为模拟量输出对应的下限 0；

最后 0x00、0x00、0x27、0x10 为模拟量输出对应的上限 0x00002710，表示 10000mm 即 10 米；

0x26 和 0x4F 为 2 个字节的 CRC 校验码；

传感器收到正确的指令（同时将修正值保存在传感器内部）将返回（16 进制）：

01 10 01 66 00 04 20 29

8）修改传感器地址（默认地址为 1）：

传感器默认地址是 1，需要修改成 3；

主机发送（16 进制）：

01 06 01 64 00 03 89 E8

其中： 0x01 为传感器原地址；
0x06 为功能码，表示写单个寄存器；
0x01 和 0x64 为设置传感器寄存器地址；
0x00 和 0x03 为本次写入寄存器的数据，将地址修改成 3；
0x89 和 0xE8 为 2 个字节的 CRC 校验码；

传感器收到正确的指令（同时将新地址保存在传感器内部）将返回（16 进制）：

01 06 01 64 00 03 89 E8

9）设置传感器距离值自动测量时间（默认地址为 1）：

如：要将传感器距离值自动返回时间改成半个小时，传感器返回时间寄存器是以 mS 为单位，半小时即 1800000mS，十六进制是 0x001B7740；

主机发送（16 进制）：

01 10 01 70 00 02 04 00 1B 77 40 AF 1C

其中： 0x01 为传感器地址；
0x10 为功能码，表示写多个寄存器；
0x01 和 0x70 为设置传感器距离值自动返回时间寄存器首地址；
0x00 和 0x02 为本次写入 2 个寄存器；
0x04 表示 4 个字节（2 个寄存器就是 4 个字节）；
0x00、0x1B、0x77、0x40 表示距离值自动返回时间的高 16 位和低 16 位，0x001B7740 换算成 10 进制就是 1800000，单位为 mS 即半小时；
0xAF 和 0x1C 为 2 个字节的 CRC 校验码；

传感器收到正确的指令（同时将距离值自动返回时间保存在传感器内部）将返回（16 进制）：

01 10 01 70 00 02 41 EF

备注：如果设置距离值自动返回时间小于传感器自身数据刷新数据的时间，数据将按传感器自身最高的速率返回；例如：传感器自身每秒刷新 10 个数据，即 100mS 刷新一次，设置的距离值自动返回时间大于 100mS 是有效的，如果将返回时间设置成 50mS，传感器仍按自身最高速率 100mS 速率刷新数据。

✧ **为了降功耗，如果设置传感器距离值自动返回时间大于等于 5 秒且传感器工作在连续测量模式或者上电自动测量模式下，传感器将自动每间隔一段时间（根据设置值）打开激光测量一次后自动关机，同时刷新距离值寄存器；**

10）修改波特率（默认地址为 1）：

传感器默认波特率是 19200，需要修改成 9600；

主机发送（16 进制）：

01 06 01 92 00 03 69 DA

其中： 0x01 为传感器原地址；
0x06 为功能码，表示写单个寄存器；
0x01 和 0x92 为设置传感器波特率寄存器地址；
0x00 和 0x03 为本次写入寄存器的数据，将波特率修改成 3，1 为 2400，2 为 4800，3 为 9600，4 为 19200，5 为 115200；

0x69 和 0xDA 为 2 个字节的 CRC 校验码；

传感器收到正确的指令后将按原波特率返回，再次通讯需要按新波特率（16 进制）：

01 06 01 92 00 03 69 DA

10. CRC 计算示例（查表）

```
static const uint8_t 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};
```

```
static const uint8_t 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};

// Start_Byte：数组指针，Num_Bytes：参与 CRC 计算的字节数
uint16_t CRC16(uint8_t *Start_Byte,uint16_t Num_Bytes)
    {
        uint8_t   uchCRCHi =0xff;      //高字节初始化
        uint8_t   uchCRCLo =0xff;      //低字节初始化
        uint16_t ulndex;               //指针
        uint16_t crc_vaule;            //返回值
        while(Num_Bytes--)             //参与 CRC 计算的数据长度
            {
              ulndex = uchCRCLo^*Start_Byte++;
              uchCRCLo = uchCRCHi^auchCRCHi[ulndex];
              uchCRCHi = auchCRCLo[ulndex];
            }
       crc_vaule = uchCRCLo;
       crc_vaule<<=8;
       crc_vaule |=uchCRCHi ;
       return(crc_vaule);              //返回 CRC 值
    }
```

慎重声明：本协议由上海协堡电子有限公司研发部制定发布，配套的上位机软件属于本公司软件著作权；未经许可，非法拷贝、使用、套用必究。

版本号：V1.5；

首次发布时间：2016 年 8 月；

历史更新说明：

- **2017 年 01 月 05 日：增加应用例程说明；**
- **2017 年 04 月 21 日：增加波特率设置功能，修正部分可能造成歧义的说明；**
- **2017 年 10 月 26 日，增加设置修正值、设置模拟量对应的量程例子，即写多个寄存器的方式；**