水中油传感器
（485型）

SN-3001-OIL-N01

Ver 2.0

目录

第 1 章 产品简介

1.1 产品概述

常用的水中油检测有悬浮法（D/λ<=1）、红外分光光度法（不适合低量程）、紫外分光光度法（不适合高量程）等。在线水中油变送器，采用荧光法原理，相比常用的几种方法，荧光法更高效快捷重复性较好，并可在线实时监测。变送器具有更出色的重复性和稳定性。带有自动清洁刷，可消除气泡、减少沾污对测量的影响，使维护周期更长，长期在线使用也能保持极佳的稳定性。可对水中油的污染起到预警作用。

1.2 功能特点

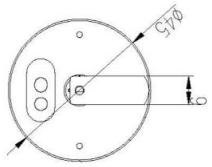
- 数字传感器，RS-485 输出，支持 ModBus；
- 带自动清洁刷，消除油污对测量的影响；
- 采用独特的光学和电子滤光技术，消除环境光对测量的影响；
- 不受水中悬浮物颗粒物的影响；

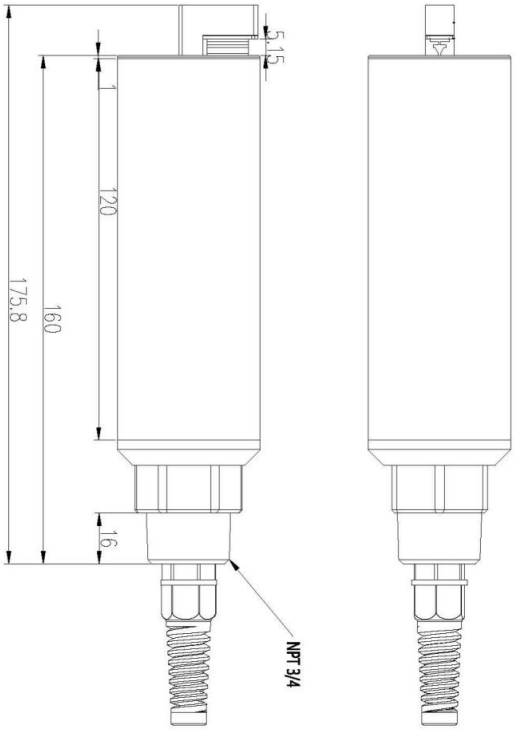
1.3 主要参数

原理	紫外荧光法
量程范围	0-50ppm 或者 0-0.40FLU
分辨率	0.01ppm
精度	±3%F.S.
检出限	根据实际油样决定
线性度	R^2>0.999
防护等级	IP68
最深深度	水下 10m
温度范围	0 ~ 50℃
传感器接口	支持 RS-485， ModBus 协议
装配	投入式
尺寸	Φ45*175.8 mm
电源信息	DC 5~12V，电流<50mA（非清洗时）
探头线缆长度	10m（默认），可定制
外壳材料	316L（可定制钛合金）

自清洁系统	有

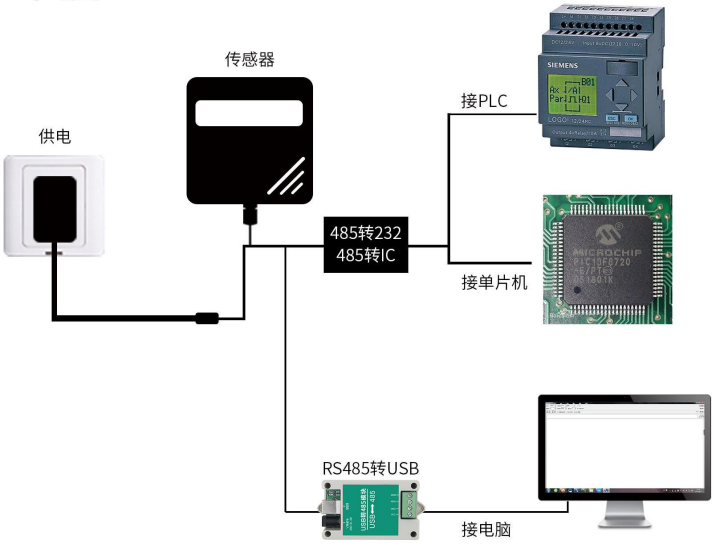
外形尺寸：50*190.8mm（Ø*L）





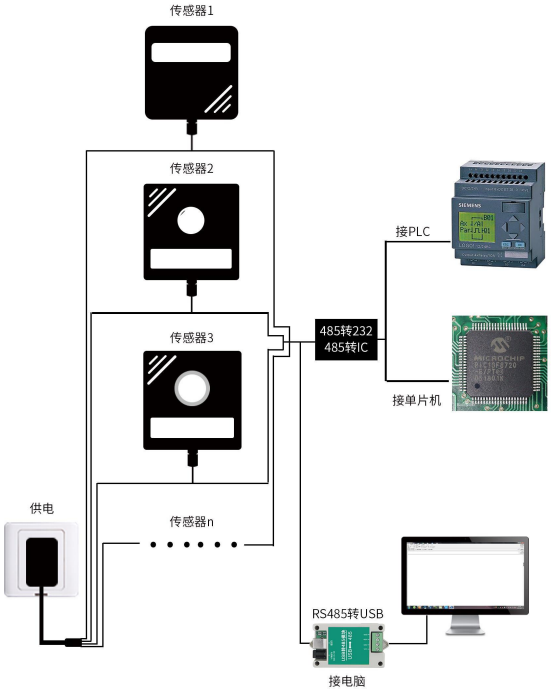
1.4 系统框架图

单接



本产品也可以多个传感器组合在一条 485 总线使用，理论上一条总线可以 254 个 485 传感器，另一端接入带有 485 接口的 PLC、通过 485 接口芯片连接单片机，或者使用 USB 转 485 即可与电脑连接，使用我公司提供的传感器配置工具进行配置和测试（在使用该配置软件时只能接一台设备）。

多接



1.5 产品选型

SN-				公司代号
	3001-			一代壳体
		OIL-		水中油变送器(默认带温度补偿)
			N01	485输出

1.6 产品外观

第 2 章 硬件连接

2.1 设备安装前检查

设备清单：

■ 主设备 1 台

■ 合格证等

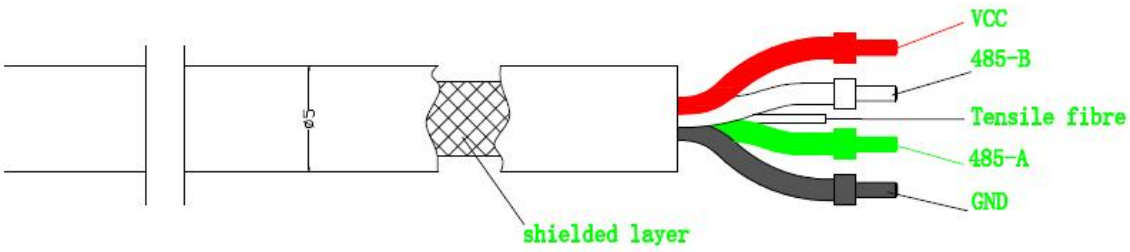
2.2 接口说明

供电电源必须是 DC 5-12V +/-5%，电流<50mA。485 信号线接线时注意 A\B 两条线不能接反，总线上多台设备间地址不能冲突。

2.2.1 传感器接线

线缆信息： 4 wire AWG-24 或 AWG-26 shielding wire. OD=5mm

1、红色线—电源 (VCC)
2、白色线—485 数据_B (485_B)
3、绿色线---485 数据_A (485_A)
4、黑色线 ---地线 (GND)
5、裸露线---- 屏蔽层



2.3 安装说明

1. 传感器校准前需用去离子水清洁传感器测量端面，用无尘纸巾擦拭干净测量窗口。
2. 测试时将校准液放入棕色的广口瓶，将将传感器用铁架台固定。
3. 放入时倾斜着缓慢放入，避免产生气泡。
4. 放入后，传感器前端与容器底部保持距离应>10cm，与侧壁距离应>3cm。
5. 配好的溶液最好 24 小时内使用。

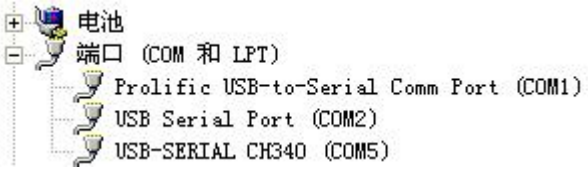
第 3 章 配置软件安装及使用

我司提供配套的“485 参数配置软件”，可以方便的使用电脑读取传感器的参数，同时灵活的修改传感器的设备 ID 和地址。

注意，使用软件自动获取时需要保证 485 总线上只有一个传感器。

3.1 传感器接入电脑

将传感器通过 USB 转 485 正确的连接电脑并提供供电后，可以在电脑中看到正确的 COM 口(“ 我的电脑— 属性—设备管理器—端口”里面查看 COM 端口）。



打开资料包，选择“调试软件”---“485 参数配置软件”，找到 

打开即可。

如果在设备管理器中没有发现 COM 口，则意味您没有安装 USB 转 485 驱动（资料包中有）或者没有正确安装驱动，请联系技术人员取得帮助。

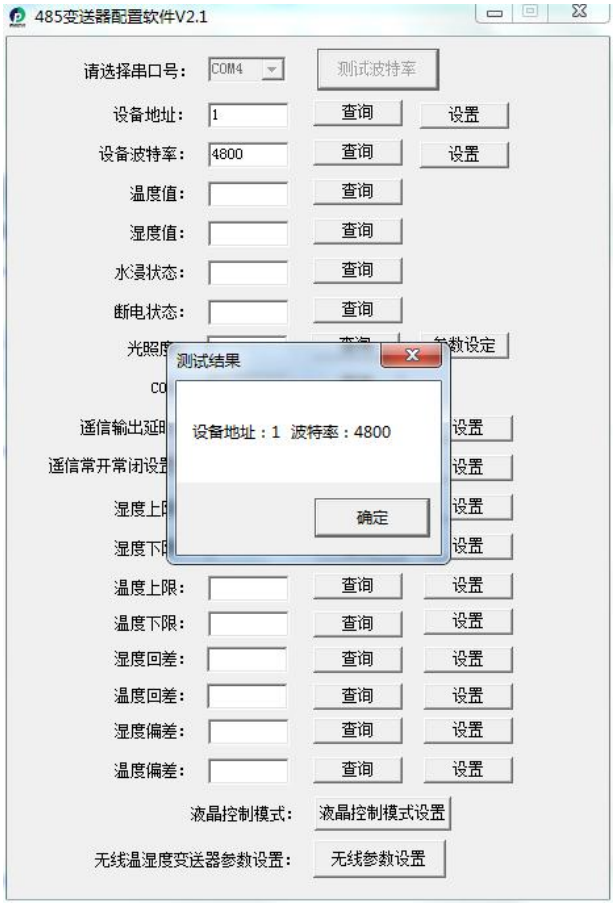
3.2 传感器监控软件的使用

①、配置界面如图所示，首先根据 3.1 章节的方法获取到串口号并选择正确的串口。

②、点击软件的测试波特率，软件会测试出当前设备的波特率以及地址，默认波特率为 4800bit/s,默认地址为 0x01。

③、根据使用需要修改地址以及波特率，同时可查询设备的当前功能状态。

④、如果测试不成功，请重新检查设备接线及 485 驱动安装情况。


485变送器配置软件V2.1
请选择串口号：
COM4
测试波特率
设备地址：
1
查询
设置
设备波特率：
4800
查询
设置
温度值：
查询
湿度值：
查询
水浸状态：
查询
断电状态：
查询
参数设定
测试结果
设备地址：1 波特率：4800
确定
遥信输出延
遥信常开常闭设
湿度上
湿度下
温度上限：
查询
设置
温度下限：
查询
设置
湿度回差：
查询
设置
温度回差：
查询
设置
湿度偏差：
查询
设置
温度偏差：
查询
设置
液晶控制模式：
液晶控制模式设置
无线温湿度变送器参数设置：
无线参数设置

第 4 章 维护和保养

4.1 维护方法

(1）变送器外表面：用自来水清洗变送器的外表面，如果仍有碎屑残留，用湿润的软布进行擦拭，对于一些顽固的污垢，可以在自来水中加入一些家用洗涤液来清洗。

(2）检查变送器的线缆：正常工作时线缆不应绷紧，否则容易使线缆内部断裂，引起变送器不能正常工作。

(3）检查变送器的测量窗口有否有脏污，清洁刷是否正常。

(4）检查变送器的清洁刷是否有所破损。

(5）连续使用 18 个月，需返厂更换动密封装置。

4.2 维护日程

1、在线藻密度变送器带有自动清洁刷，可以自动进行清洗，不需要频繁地进行清洗。

4.3 注意事项

探头中含有敏感的光学和电子部件，确保探头不要受到剧烈的机械撞击。探头内部没有需要维护的部件。

4.4 其他

错误	可能的原因	解决方法
操作界面无法连接 或不显示测量结果	控制器与线缆连接出错	重新连接控制器和线缆
	线缆故障	请联系我们
测量值过高、过低或数值持续不稳定	变送器视窗被外物附着	清洗变送器视窗表面
	变送器海绵破损	更换变送器海绵

第 5 章 变送器的校准

5.1 水中油校准说明

1、水中油传感器的校准：水中油传感器软件(参见 ModBus 说明书)支持 2 点校准。较常见的环境研究硫酸奎宁，可用于传感器校准中。

2、准备

（1）先取 0.1g 硫酸奎宁放入 1000ml 的试剂瓶 A，然后加入 0.5mol/L 的硫酸至 1000ml 充分摇匀此为 100ppm。

（2） 再取 1ml 上述溶液放入 1000ml 的试剂瓶 B，然后加入 0.5mol/L 的硫酸至 1000ml 充分摇匀。则试剂瓶 B 的溶液浓度为 100ppb。

（3）配好的溶液最好 24 小时内使用。

警告：操作过程中必须配带手套。

3、校准(2 点校准)

（1）恢复用户校准数据为默认, K=1, B=0(详见 ModBus 文档)。

（2）将传感器放入超纯水中或去离子水，然后读取水中油值，记录为 X。

（3）将传感器放入标液中(上述的 100ppb 中，标液中水中油的理论值为 Z)，将数值记录为 Y。

（4）按如下算式记录 K 和 B 值:

K=(Z-0)/(Y-X), B= - KX

（5）将 K, B 值写入传感器。（数据帧格式参照 ModBus 文档）

5.2 所需器具及原料

硫酸奎宁溶液

电子天平

蒸馏水或去离子水（屈臣氏蒸馏水）

棕色广口瓶*2（1000mL）

移液枪

手套

附录 数据通信

1. 数据格式

本文档中的数据格式说明;

----二进制显示，后缀用 B，例如:10001B

----十进制显示，无任何前后缀，例如:256

----十六进制显示，前缀用 0x，例如:0x2A

----ASCII 字符或 ASCII 字符串显示，例如:" YL1014010022"

1.1. 命令结构

ModBus 应用协议定义了简单协议数据单元(PDU)，与基础通信层无关:

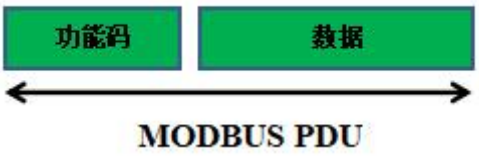


图 1: ModBus 协议数据单元

特定总线或网络上的 ModBus 协议映射介绍了协议数据单元的附加字段。启动 ModBus 交换的客户端创建 ModBus PDU;随后添加域，建立正确的通信 PDU.

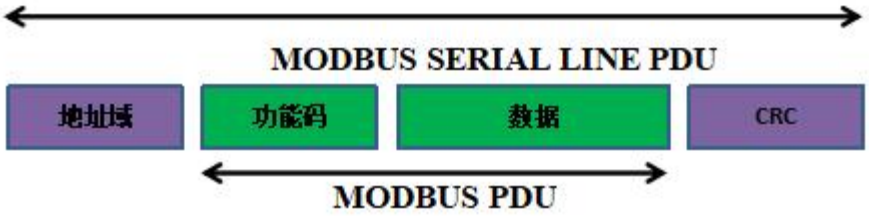


图 2: 串行通信的 ModBus 结构

在 ModBus 串行线上，地址域仅包含从设备地址。

提示:

OIW 探头的从设备地址范围为 1…247

通过在信息的地址域中设置从设备地址主站分配从设备。从设备回馈响应后，将原地址放置在响应地址域中，使得主站知道进行响应的从设备。

功能码指示服务器执行的操作类型。

CRC 域是"冗余校验"计算结果，按照信息内容执行。

1.2 ModBus RTU 传输模式

设备使用 RTU(远程终端单元)模式进行 ModBus 串行通信时，每条信息的 8 位字节包含两个 4 位十六进制字符。此模式的主要优点是具有更大的字符密度，比相同波特率的 ASCII 模式具有更好的数据吞吐量。每条信息必须以连续的字符串传输。

1.3 在 RTU 模式中的每个字节的格式(11 位)

编码系统：　　8 位二进制

　　报文中每个 8 位字节含有两个 4 位十六进制字符(0-9、A-F)

每个字节中的位:　　1 个起始位

　　8 个数据位，先发最低有效位

　　无奇偶校验位

　　1 位停止位

波特率:　　9600bps

字符是如何串行传送的:

每个字符或字节均由此顺序发送(从左到右)

最低有效位(LSB)……最高有效位(MSB)

起始位	1	2	3	4	5	6	7	8	停止位

图 3: RTU 模式位序列

检查域结构:

循环冗余校验(CRC16)

结构说明:

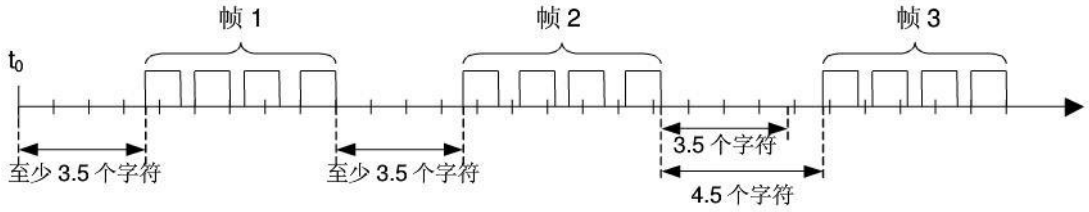
从设备地址	功能码	数据	CRC	
1 个字节	1 个字节	0…252 字节	2 个字节	
			CRC 低字节	CRC 高字节

图 4: RTU 信息结构

ModBus 帧最大为 256 字节

2. 信息帧格式

在 RTU 模式，报文帧由时长至少为 3.5 个字符时间的空闲间隔区分，在后续部分，这个时间区间被称作 t3.5.



起始	地址	功能代码	数据	CRC 校验	结束
>=3.5 个字符	8 位	8 位	N×8 位	16 位	>=3.5 个字符

图 5: RTU 报文帧

整个报文帧必须以连续的字符流发送。

两个字符之间的停顿间隔超过 1.5 个字符，信息帧认为不完整，接收方不接收此信息帧。

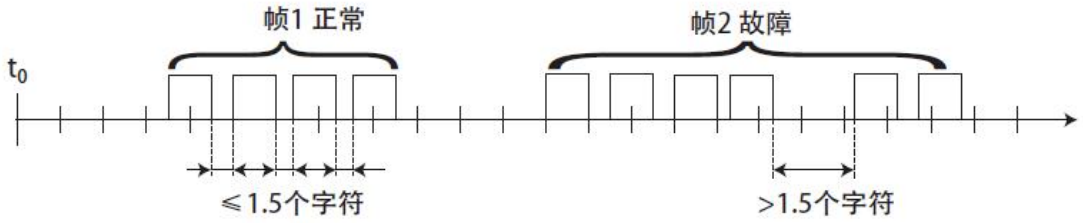


图 6: 帧的数据传输

2.1 ModBus-RTU CRC 校验

1、在 RTU 模式包含一个对全部报文内容执行的，基于循环冗余校验(CRC)算法的错误检测域。

2、CRC 域检查整个报文的内容，不管报文有无奇偶校验，均执行此校验。CRC 域包含由两个 8 位字节组成的一个 16 位值。采用 CRC16 校验。低字节在前，高字节在后。

2.2 ModBus-RTU 在 YOSEMITECH BGA 探头中的实施

根据官方 ModBus 定义，由 3.5 个字符间隔触发命令开始，同样，命令结束也通过 3.5 个字符间隔表示。设备地址和 ModBus 功能代码有 8 位。数据字符串包含 n×8 位，数据字符串包含寄存器的起始地址和读/写寄存器的数量。CRC 校验为 16 位。

	开始	设备地址	功能	数据	总和校验		结束
数值	在 3.5 个字符期间无信号	1-247	符合 ModBus 规范的功能码	符合 ModBus 规范的数据	CRCL	CRCH	在 3.5 个字符期间无信号
字节	3.5	1	1	n	1	1	3.5

图 7: 数据传输的 ModBus 定义

2.3 OIW 探头的 ModBus-RTU 功能码

OIW 探头仅使用两个 ModBus 功能码:

0x03: 读保持寄存器 0x10： 写多重寄存器

2.3.1 ModBus 功能码 0x03：读保持寄存器

此功能码用于读取远程设备的保持寄存器的连续块内容。请求 PDU 指定开始寄存器地址和寄存器数量。从零开始寻址寄存器。因此，寻址寄存器 1-16 为 0-15。响应信息中的寄存器数据按照每个寄存器两个字节打包。对于每个寄存器，第一个字节包含高位比特，第二个字节包含低位比特。

请求

功能码	1 个字节	0x03
开始地址	2 个字节	0x0000….0xffff
读取寄存器数量	2 个字节	1…125

图 8: 读取保持寄存器请求帧

响应

功能码	1 个字节	0x03
字节数	1 个字节	N×2
寄存器值	N×2 个字节	

N = 寄存器数量

图 9: 读取保持寄存器响应帧

下面以读取保持寄存器 108-110 为例说明请求帧和响应帧。(寄存器 108 的内容只读，为两个字节数值 0X022B，寄存器 109-110 内容为 0X0000 和 0X0064)

请求帧		响应帧	
数制	(十六进制)	数制	(十六进制)
功能码	0x03	功能码	0x03
开始地址(高字节)	0x00	计算字节	0x06
开始地址(低字节)	0x6B	寄存器值(高字节) (108)	0x02
读取寄存器数量(高字节)	0x00	寄存器值(低字节) (108)	0x2B
读取寄存器数量(低字节)	0x03	寄存器值(高字节) (109)	0x00
		寄存器值(低字节) (109)	0x00
		寄存器值(高字节) (110)	0x00
		寄存器值(低字节) (110)	0x64

图 10: 读取保持寄存器请求帧和响应帧实例

2.3.2 ModBus 功能码 0x10：写多重寄存器

此功能码用于向远程设备中写入连续寄存器(1…123 个寄存器)块，在请求数据帧中指定写入的寄存器值。数据以每个寄存器两个字节打包。响应帧返回功能码，开始地址和写入的寄存器的数量。

请求

功能码	1 个字节	0x10
开始地址	2 个字节	0x0000….0xffff
输入寄存器数量	2 个字节	0x0001….0x0078
字节数	1 个字节	N×2

寄存器值	N×2 个字节	值

N = 寄存器数量

图 11: 写多重寄存器请求帧

响应

功能码	1 个字节	0x10
开始地址	2 个字节	0x0000….0xffff
寄存器数量	2 个字节	1…123(0x7B)

N = 寄存器数量

图 12: 写多重寄存器响应帧

下面以写入数值 0x000A 和 0x0102 至开始地址为 2 的两个寄存器中为例说明请求帧和响应帧。

请求帧		响应帧	
数制	(十六进制)	数制	(十六进制)
功能码	0x10	功能码	0x10
开始地址(高字节)	0x00	开始地址(高字节)	0x00
开始地址(低字节)	0x01	开始地址(低字节)	0x01
输入寄存器数量(高字节)	0x00	输入寄存器数量(高字节)	0x00
输入寄存器数量(低字节)	0x02	输入寄存器数量(低字节)	0x02
字节数	0x04		
寄存器值(高字节)	0x00		
寄存器值(低字节)	0x0A		
寄存器值(高字节)	0x01		
寄存器值(低字节)	0x02		

图 13: 写多重寄存器请求帧和响应帧实例

3. 数据结构类型

3.1 浮点数

定义: 浮点数，符合 IEEE 754(单精度)

说明	符号	指数	尾数	总和
位	31	30…23	22…0	32
指数偏差	127			

图 14: 浮点数单精度定义(4 个字节，2 个 ModBus 寄存器)

实例: 将十进制数 17.625 编译成二进制数

步骤 1：将十进制形式表示的 17.625 转换成二进制形式的浮点数

先求整数部分的二进制表示

$17 = 16 + 1 = 1\times 2^4 + 0\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0$

所以整数部分 17 的二进制表示为 10001B

再求小数部分的二进制表示

$0.625 = 0.5 + 0.125 = 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3}$

所以小数部分 0.625 的二进制表示为 0.101B

所以十进制形式表示的 17.625 的二进制形式的浮点数为 10001.101B

步骤 2：移位求指数。

将 10001.101B 向左移，直到小数点前只剩下一位，得到 1.0001101B，而 $10001.101B = 1.0001101\ B\times 2^4$ 。所以指数部分为 4，加上 127，变为 131，其二进制表示为 10000011B，

步骤 3：计算尾数

去除 1.0001101B 的小数点前的 1 得到尾数为 0001101B（因为小数点前必定为 1，所以 IEEE 规定只记录小数点后面的就可以），针对 23 位尾数的重要说明:第一位(即隐藏位)不编译。隐藏位是分隔符左侧的位，此位通常被设置为 1 并抑制。

步骤 4：符号位定义

正数的符号位为 0，负数的符号位为 1,所以 17.625 的符号位为 0.

步骤 5：转化为浮点数

1 位符号 + 8 位指数 + 23 位尾数

0 10000011 00011010000000000000000B(对应十六进制表示为 0x418D0000)

参考代码:

1、如果用户使用的编译器有实现此功能的库函数则可以直接调用此库函数，例如使用的是 C 语言，那么可以直接调用 C 库函数 memcpy 获取一个浮点数在内存中存储格式的整数表示；

```
例如:   float   floatdata;//被转化的浮点数
        void*   outdata;
        memcpy(outdata,&floatdata,4);
```

假如 floatdata=17.625

若为小端存储模式则执行完上面的语句后则

地址单元 outdata 存储的数据为 0x00

地址单元(outdata+1) 存储的数据为 0x00

地址单元(outdata+2) 存储的数据为 0x8D

地址单元(outdata+3) 存储的数据为 0x41

若为大端存储模式则执行完上面的语句后

地址单元 outdata 存储的数据为 0x41

地址单元(outdata+1) 存储的数据为 0x8D

地址单元(outdata+2) 存储的数据为 0x00

地址单元(outdata+3) 存储的数据为 0x00

2、如果用户使用的编译器没有实现此功能的库函数则可以用如下的函数实现此功能:

```
void memcpy(void *dest,void *src,int n)
{
    char *pd = (char *)dest;
    char *ps = (char *)src;
    for(int i=0;i<n;i++)    *pd++ = *ps++;
}
```

然后同上进行调用 memcpy(outdata,&floatdata,4);

实例: 将二进制浮点数 0100 0010 0111 1011 0110 0110 0110 0110B 编译为十进制数

步骤 1: 将二进制浮点数 0100 0010 0111 1011 0110 0110 0110 0110B 分为符号位、指数位和尾数位

0 10000100 11110110110011001100110B

1 位符号 + 8 位指数 + 23 位尾数

符号位 S: 0 表示正数

指数位 E：

10000100B =1× 2^7+0× 2^6+0× 2^5+0× 2^4 +0 × 2^3+1× 2^2+0× 2^1+0× 2^0

=128+0+0+0+0+4+0+0=132

尾数位 M：11110110110011001100110B =8087142

步骤 2: 计算十进制数

$$D=(-1)^S\times(1.0+M/2^{23})\times2^{E-127}$$

$$=(-1)^0\times(1.0+8087142/2^{23})\times2^{132-127}$$

$$=1\times1.964062452316284\times32$$

$$=62.85$$

参考代码:

```
float floatTOdecimal(long int byte0, long int byte1, long int byte2, long int byte3)
{
    long int realbyte0,realbyte1,realbyte2,realbyte3;
char S;
long int    E,M;
```

```
float D;
                realbyte0 = byte3;
realbyte1 = byte2;
realbyte2 = byte1;
                realbyte3 = byte0;

if((realbyte0&0x80)==0)
{
S = 0;//正数
}
else
{
S = 1;//负数
}
E = ((realbyte0<<1)|(realbyte1&0x80)>>7)-127;
M = ((realbyte1&0x7f) << 16) | (realbyte2<< 8)| realbyte3;
D = pow(-1,S)*(1.0 + M/pow(2,23))* pow(2,E);
return D;
}
```

函数说明:参数----byte0、byte1、byte2、byte3 代表二进制浮点数的 4 个字节(

返回值----转换得到的十进制数

例如用户向探头发送获取温度值和 OIW 值命令，收到的应答帧中的代表温度值的 4 个字节为 0x00,0x00,0x8d,0x41,那么用户可以通过下面的调用语句得到对应的温度值的十进制数

即 temperature = 17.625。

```
float temperature = floatTOdecimal( 0x00, 0x00, 0x8d, 0x41);
```

3.1.1 字符

定义: 字符的用 ASCII 码表示.

实例:字符串“YL”在命令传输中表示为对应的 ASCII 码(以下编译参考 ASCII 码表格)

“Y” 在命令传输中表示为 0x59

“L” 在命令传输中表示为 0x4C

4. 寄存器地址

4.1 设置从机 ID

作用：设置电极的 ModBus 从设备地址，地址范围为 1~247。

可以通过地址为 0x3000 的 ModBus 寄存器设置电极的 ModBus 从设备地址。

起始地址	寄存器数量	寄存器 1	ModBus 功能码
0x3000	0x01	新的设备地址	0x10

图 15:　设置从机 ID 命令寄存器定义

下面以电极旧的设备地址=0x01，新的设备地址=0x14 为例说明设置从机 ID 命令的请求帧和应答帧

定义	地址域	功能码	起始地址		寄存器数量		字节数	寄存器值		CRC	
字节	0	1	2	3	4	5	6	7	8	9	10
内容	0x01	0x10	0x30	0x00	0x00	0x01	0x02	0x14	0x00	0x99	0x53

图 16:　设置从机 ID 请求帧实例　　备注: byte8 为保留值，无意义

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x10	0x30	0x00	0x00	0x01	0x0E	0xC9

图 17:　设置从机 ID 应答帧实例

4.2 获取 SN

作用：获取电极的识别号 SN，每个电极都有惟一的 SN。

可以从起始地址为 0x0900 的连续 7 个 ModBus 寄存器中读取探头的 SN。

起始地址	寄存器数量	寄存器 1—7	ModBus 功能码
0x0900	0x07	SN	0x03

图 18:　获取 SN 命令寄存器定义

下面以从设备地址 0x01，返回的 SN“YL1014010022”为例说明获取 SN 命令的请求帧和应答帧

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x03	0x09	0x00	0x00	0x07	0x07	0x94

图 19:　获取 SN 命令请求帧实例

定义	地址域	功能码	字节数	寄存器值			CRC	
字节	0	1	2	3	4~15	16	17	18
内容	0x01	0x03	0x0E	0x00	“YL1014010022”	0x00	0x4c	0x5f

图 20:　获取 SN 命令应答帧实例　　备注:探头 SN 如下，以 ASCII 形式存储

字节	4	5	6	7	8	9	10	11	12	13	14	15
内容	0x59	0x4C	0x31	0x30	0x31	0x34	0x30	0x31	0x30	0x30	0x32	0x32

图 21:　探头的 SN

4.3 开始测量

作用：让探头开始测量（默认上电就开始测量）。使用 ModBus 寄存器 0x2500。

起始地址	寄存器数量	ModBus 功能码
0x2500	0x01	0x03

图 22: 开始测量命令的寄存器定义

下面以从设备地址 0x01 为例说明开始测量命令的请求帧和应答帧

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x03	0x25	0x00	0x00	0x01	0x8F	0x06

图 23: 开始测量命令请求帧实例

定义	地址域	功能码	字节数	寄存器值	CRC	
字节	0	1	2	3~4	5	6
内容	0x01	0x03	0x02	无意义		

图 24: 开始测量命令应答帧实例

4.4 获取温度和 OIW 值

作用：获取探头的温度和 OIW 值：温度的单位为摄氏度，OIW 值为经过用户校准后的值，单位为 ppm。

可以从起始地址为 0x2600 的连续 5 个 ModBus 寄存器中读取探头的温度和 OIW 值(另外还包括一个错误标志用于指示刷子是否出现问题，0—无问题；0XFF—刷子没停到正确的位置，探头将不再测量)。

起始地址	寄存器数量	寄存器 1、2	寄存器 3、4	寄存器 5	ModBus 功能码
0x2600	0x05	温度值	OIW 值	错误标志	0x03

也可以单独获取每个值。

含义	地址	寄存器数量	字节数
温度值	0x2600H	2	4
OIW 值	0x2602H	2	4

图 25: 获取温度和 OIW 值命令的寄存器定义

下面以从设备地址 0x01,返回的温度值为 17.625，OIW 值为 7.625 为例说明获取获取温度和 OIW 命令的请求帧和应答帧。

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x03	0x26	0x00	0x00	0x05	0x8E	0x81
	0x01	0x03	0x26	0x02	0x00	0x03	0xAF	0x43

图 26: 获取温度和 OIW 值命令的请求帧

定义	地址域	功能码	字节数	寄存器值				CRC	
字节	0	1	2	3~6	7~10	11	12	13	14
内容	0x01	0x03	0x0A	17.625	7.625	0x00	0x00	0Xc7	0x33
	0x01	0x03	0x06	7.625					

图 27: 获取温度和 OIW 值命令的应答帧 备注: 温度值、OIW 值：小端存储模式，浮点数

温度值(17.625)				OIW 值(7.625)				错误标志(11)	保留位(12)
0x00	0x00	0x8D	0x41	0x0 0	0x00	0xF4	0x40	0x00	0x00

图 28: 温度值和 OIW 值字节分布

4.5 获取软件和硬件版本号

作用：获取当前使用的硬件版本号和软件版本号。

可以从起始地址为 0x0700 的连续 2 个 ModBus 寄存器中读取探头的软件和硬件版本号。

起始地址	寄存器数量	寄存器 1	寄存器 2	ModBus 功能码
0x0700	0x02	硬件版本号	软件版本号	0x03

图 29: 获取软件和硬件版本号命令的寄存器定义

下面以从设备地址 0x01，返回的硬件版本 1.0，软件版本 1.0 为例说明获取软件和硬件版本号命令的请求帧和应答帧。

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x03	0x07	0x00	0x00	0x02	0x c5	0x7f

图 30: 获取软件和硬件版本号命令的请求帧

定义	地址域	功能码	字节数	寄存器值				CRC	
字节	0	1	2	3~4		5~6		7	8
内容	0x01	0x03	0x04	0x01	0x00	0x01	0x00	0xfa	0x5f

图 31: 获取软件和硬件版本号命令的应答帧

4.6 停止测量

作用：当数据稳定后可以停止测量。

使用 ModBus 寄存器 0x2E00。

起始地址	寄存器数量	ModBus 功能码
0x2E00	0x01	0x03

图 32: 停止测量命令的寄存器定义

下面以从设备地址 0x01 为例说明停止测量命令的请求帧和应答帧。

定义	地址域	功能码	起始地址	寄存器数量	CRC

字节	0	1	2	3	4	5	6	7
内容	0x01	0x03	0x2E	0x00	0x00	0x01	0x8D	0x22

图 33:　停止测量命令的请求帧

定义	地址域	功能码	字节数	寄存器值	CRC	
字节	0	1	2	3~4	5	6
内容	0x01	0x03	0x02	无意义		

图 34:　停止测量命令的应答帧

4.7 获取用户校准参数

作用：获取两个校准参数 K、B。(为防止探头老化等因素造成 OIW 值偏差，校准公式 OIW_{final}=K*OIW +B，一般默认值为：**K=1;B=0**。)

可以从起始地址为 0x1100 的连续 4 个 ModBus 寄存器中读取用户校准参数 K、B

起始地址	寄存器数量	寄存器 1、2	寄存器 3、4	ModBus 功能码
0x1100	0x04	K 值	B 值	0x03

图 35:　获取用户校准参数命令的寄存器定义

下面从设备地址 0x01，返回 K=1.0，B=0.0 为例说明获取用户校准参数命令请求帧和应答帧。

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x03	0x11	0x00	0x00	0x04	0x41	0X35

图 36:　获取用户校准参数命令请求帧

定义	地址域	功能码	字节数	寄存器值		CRC	
字节	0	1	2	3~6	7~10	11	12
内容	0x01	0x03	0x08	1.0	0.0	0x9E	0x12

图 37:　获取用户校准参数命令应答帧　　备注: K、B：小端存储模式，浮点数

K(3~6)				B(7~10)			
0x00	0x00	0x80	0x3F	0x00	0x00	0x00	0x00

图 38:　K、B 值的字节分布

4.8 设置用户校准参数

作用：设置两个校准参数 K、B。

可以通过起始地址为 0x1100 的连续 4 个 ModBus 寄存器设置用户校准参数 K、B。

起始地址	寄存器数量	寄存器 1、2	寄存器 3、4	ModBus 功能码
0x1100	0x04	K 值	B 值	0x10

图 39:　设置用户校准参数命令的寄存器定义

下面以从设备地址 0x01, K=1.0，B=0.0 为例说明设置用户校准参数命令的请求帧和应答帧。

定义	地址域	功能码	起始地址	寄存器数量	字节数	寄存器值	CRC

字节	0	1	2	3	4	5	6	7~10	11~14	15	16
内容	0x01	0x10	0x11	0x00	0x00	0x04	0x08	1.0	0.0	0x81	0xAE

图 40:　设置用户校准参数命令请求帧　　备注: K，B：小端存储模式，浮点数

K(7~10)				B(11~14)			
0x00	0x00	0x80	0x3F	0x00	0x00	0x00	0x00

图 41:　K、B 值的字节分布

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x10	0x11	0x00	0x00	0x04	0xc4	0xf6

图 42:　设置用户校准参数命令应答帧

4.9 开启刷子

作用：让刷子开始转动，建议刚上电开启一次刷子（默认关闭）。

可以通过起始地址为 0x3100 让刷子开始转动。

起始地址	寄存器数量	ModBus 功能码
0x3100	0x00	0x10

图 43:　开启刷子命令的寄存器定义

下面以从设备地址 0x01 为例说明开启刷子命令的请求帧和应答帧。

定义	地址域	功能码	起始地址		寄存器数量		字节数	CRC	
字节	0	1	2	3	4	5	6	7	8
内容	0x01	0x10	0x31	0x00	0x00	0x00	0x00	0x74	0x94

图 44:　开启刷子命令请求帧

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x10	0x31	0x00	0x00	0x00	0xce	0xf5

图 45:　开启刷子命令应答帧

4.10 设置刷子转动间隔

作用：设置刷子的转动间隔，单位为 min。

可以通过起始地址为 0x3200 的 1 个 ModBus 寄存器设置刷子的转动间隔。

起始地址	寄存器数量	寄存器 1	ModBus 功能码
0x3200	0x01	转动间隔值（min）	0x10

图 46:　设置刷子的转动间隔命令的寄存器定义

下面以从设备地址 0x01,设置时间为 10min 为例说明开启刷子命令的请求帧和应答帧。

定义	地址域	功能码	起始地址	寄存器数量	字节数	寄存器值	CRC

字节	0	1	2	3	4	5	6	7	8	9	10
内容	0x01	0x10	0x32	0x00	0x00	0x01	0x02	0x0A	0x00	0xB3	0x33

图 47:　设置刷子的转动间隔命令请求帧　　注意：小端模式

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x10	0x32	0x00	0x00	0x01	0x0f	0x71

图 48:　设置刷子的转动间隔命令应答帧

4.11 获取刷子转动间隔

作用：获取刷子的转动间隔，默认为 30min 转一次

可以从起始地址为 0x3200 的 1 个 ModBus 寄存器中读取刷子的转动间隔

起始地址	寄存器数量	寄存器 1	ModBus 功能码
0x3200	0x01	转动间隔值（min）	0x03

图 49:　获取刷子的转动间隔命令的寄存器定义

下面以从设备地址 0x01, 返回的时间为 30min 为例说明获取用户校准参数命令的请求帧和应答帧。

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	0x01	0x03	0x32	0x00	0x00	0x01	0x8a	0Xb2

图 50:　获取刷子的转动间隔命令请求帧

定义	地址域	功能码	字节数	寄存器值		CRC	
字节	0	1	2	3	4	5	6
内容	0x01	0x03	0x02	0x1E	0x00	0xb1	0xe4

图 51:　获取刷子的转动间隔命令应答帧　　备注:小端存储模式

4.12 设置刷子上电工作模式

作用：设置刷子上电工作模式，默认为上电刷子自动工作，掉电保存。

可以通过起始地址为 0x1B00 的 1 个 ModBus 寄存器设置刷子上电工作模式。

起始地址	寄存器数量	寄存器 1	ModBus 功能码
0x1b00	0x01	刷子上电工作模式	0x10

图 52:　设置刷子上电工作模式的寄存器定义

下面以从设备地址 0x01,设置上电刷子自动工作为例说明开启刷子命令的请求帧和应答帧。

定义	地址域	功能码	起始地址		寄存器数量		字节数	寄存器值		CRC	
字节	0	1	2	3	4	5	6	7	8	9	10
内容	0x01	0x10	0x1b	0x00	0x00	0x01	0x02	**0x01**	0x00		

图 53:　设置刷子的转动间隔命令请求帧

上电刷子自动转动	上电刷子不会转动（需 2.2.9 开启刷子）
01	00

图 54: 设置刷子模式定义

4.13 获取刷子上电工作模式

作用：获取刷子上电工作模式

可以从起始地址为 0x1b00 的 1 个 ModBus 寄存器中读取刷子的转动间隔

起始地址	寄存器数量	寄存器 1	ModBus 功能码
0x1b00	0x01	刷子上电工作模式	0x03

图 55: 获取刷子上电工作模式命令的寄存器定义

4.14 获取从机 ID

作用：获取当前电极的 ModBus 从设备地址。该命令以 0xFF 作为固定地址域

可以从起始地址为 0x3000 的 ModBus 寄存器中读取当前电极的 ModBus 从设备地址。

起始地址	寄存器数量	寄存器 1	ModBus 功能码
0x3000	0x01	当前设备地址	0x03

图 56: 获取从机 ID 的寄存器定义

下面以返回的地址 0x03 为例说明获取从机 ID 命令的请求帧和应答帧。

定义	地址域	功能码	起始地址		寄存器数量		CRC	
字节	0	1	2	3	4	5	6	7
内容	**0xFF**	0x03	0x30	0x00	0x00	0x01	0x9E	0xD4

图 57: 获取从机 ID 命令的请求帧

定义	地址域	功能码	字节数	寄存器值		CRC	
字节	0	1	2	3	4	5	6
内容	0xFF	0x03	0x02	0x03	0x00(保留)	0x91	0x60

图 58: 获取从机 ID 命令的应答帧