YOSEMITECH

光学溶氧传感器

MODBUS RTU

操作员手册

目录

[1 MODBUS RTU基本信息 3](#_Toc416967999)

[1.1 概述 3](#_Toc416968000)

[1.2 MODBUS命令结构 3](#_Toc416968001)

[1.3 MODBUS RTU 在YOSEMITECH 光学溶氧传感器中的实施 5](#_Toc416968002)

[1.4 YOSEMITECH光学溶氧传感器的MODBUS RTU功能码 5](#_Toc416968003)

[1.5 光学溶氧传感器中的数据格式 7](#_Toc416968004)

[2 MODBUS RTU中的光学溶氧传感器命令 9](#_Toc416968005)

[2.1 概述 9](#_Toc416968006)

[2.2 各命令详述 10](#_Toc416968007)

[3 获取溶氧值流程 14](#_Toc416968008)

# 1 MODBUS RTU基本信息

## 1.1 概述

此文档适合的光学溶氧传感器的硬件版本号为V2.0**；**软件版本号为V5.9及以上。

本文档详细介绍了光学溶氧传感器的MODBUS RTU接口，目标对象是软件程序员。

## 1.2 MODBUS命令结构

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:” YL0114010022”

* + 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:

**数据**

**功能码**

**MODBUS PDU**

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.

**MODBUS SERIAL LINE PDU**

**CRC**

**数据**

**功能码**

**地址域**

**MODBUS 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:**

光学溶氧传感器的从设备地址范围是1…247

在主机发送的请求帧的地址域中设置从机的设备地址。从机回馈响应时，将自己的设备地址放置在响应帧的地址域中，使得主站知道哪个从机回馈响应的。

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. **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。

**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

2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 | 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

**1.2.3 MODBUS RTU信息帧**

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.



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时间间隔超过1.5个字符时，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

**1.2.4 MODBUS RTU CRC校验**

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。

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。

## 1.3 MODBUS RTU 在YOSEMITECH 光学溶氧传感器中的实施

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 | 功能码 | 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

## 1.4 YOSEMITECH光学溶氧传感器的MODBUS RTU功能码

YOSEMITECH光学溶氧传感器仅使用两个MODBUS功能码:

0x03: 读保持寄存器 0x10： 写多重寄存器

**1.4.1 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….0xfffff |
| 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 |  | 字节数 | 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

**1.4.2 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

## 1.5 光学溶氧传感器中的数据格式

**1.5.1 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

= 16 + 1 = 1× + 0× + 0× + 0× + 1×

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

= 0.5 + 0.125 = 1× + 0× + 1×

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× 24 。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:

指数位E：10000100B =1×+0×+0×+0×+1×+0×+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 =××

= ××

= 1×1.964062452316284×32

= 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溶氧值命令，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);

**1.5.2 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

# 2 MODBUS RTU中的光学溶氧传感器命令

## 2.1 概述

为了与光学溶氧传感器电极进行MODBUS RTU通信，需要MODBUS主站终端应用软件。MODBUS RTU是开放式标准，提供多个免费商业应用工具套件。在本手册中，MODBUS寄存器的地址从1开始。但是，MODBUS主站协议从寄存器地址0开始工作。通常，MODBUS主站软件编译地址。因此，寄存器地址2090将被 MODBUS主站软件编译为2089。

## 2.2 各命令详述

**2.2.1 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

**2.2.2 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“YL0114010022”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 | “YL0114010022” | 0x00 | 0x19 | 0x66 |

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 | 0x30 | 0x31 | 0x31 | 0x34 | 0x30 | 0x31 | 0x30 | 0x30 | 0x32 | 0x32 |

图21: 探头的SN

**2.2.3 开始测量**

作用：让探头连续发光。开始溶氧值测量。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

**2.2.4**  **获取温度和溶氧值**

作用：获取探头的温度和溶氧值：温度的单位为摄氏度，溶氧值为经过用户校准后的DO值，单位为%。

可以从起始地址为0x2600的连续4个MODBUS寄存器中读取探头的温度和溶氧值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 起始地址 | 寄存器数量 | 寄存器1、2 | 寄存器3、4 | MODBUS功能码 |
| 0x2600 | 0x04 | 温度值 | 溶氧值 | 0x03 |

图25: 获取温度和溶氧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，溶氧值为17.625为例说明获取温度和溶氧值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 | 0x04 | 0x4F | 0x41 |

图26: 获取温度和溶氧值命令的请求帧

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 内容 | 0x01 | 0x03 | 0x08 | 17.625 | 17.625 | 0x12 | 0x65 |

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 温度值（3~6） | | | | 溶氧值（7-10） | | | |
| 0x00 | 0x00 | 0x8D | 0x41 | 0x00 | 0x00 | 0x8D | 0x41 |

图28: 温度值和溶氧值字节分布

**2.2.5 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2.0，软件版本5.7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 | 0xc5 | 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 | 0x02 | 0x00 | 0x05 | 0x07 | 0xb9 | 0x19 |

图31: 获取软件和硬件版本号命令的应答帧

**2.2.6 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

**2.2.7 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溶氧值偏差，校准公式DOfinal=K\*DO +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

下面以从设备地址0x01, 返回的K=1.0，B=0.0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 | 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

**2.2.8 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

**2.2.9 设置荧光帽参数**

作用：设置探头荧光帽参数（K0~K7），更换荧光帽时使用。

可以通过起始地址为0x2700的连续16个MODBUS寄存器设置探头荧光帽参数。

|  |  |  |  |
| --- | --- | --- | --- |
| 起始地址 | 寄存器数量 | 寄存器1~16 | MODBUS功能码 |
| 0x2700 | 0x10 | K0~ K7 | 0x10 |

图43: 设置荧光帽参数命令的寄存器定义

下面以从设备地址0x01为例说明设置荧光帽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 | 7~38 | 39 | 40 |
| 内容 | 0x01 | 0x10 | 0x27 | 0x00 | 0x00 | 0x10 | 0x20 | K0~ K7 | XX | XX |

图44: 设置荧光帽参数命令的请求帧 备注: Ki(i=0~7)荧光帽参数：小端存储模式，浮点数

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 0x27 | 0x00 | 0x00 | 0x10 | 0xcb | 0x71 |

图45: 设置荧光帽参数命令的应答帧

**2.2.10**  **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46: 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47: 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48: 获取从机ID命令的应答帧

# 3 获取溶氧值流程

**开始测量**

探头开始测量发光

获取温度和溶氧

计算DO（mg/L）

延时>=1s

Y

**获取温度和溶氧值**

处理流程 对应帧命令

N

延时1s

Y

获取温度和溶氧值是否大于10次

图47：测试溶氧值的流程图

* **开始测量，**让探头连续激发LED发光，求出补偿和校准后的DO值。
* **获取温度和溶氧值，**获取探头的温度（摄氏度），和DO（%），在开始测量命令后延时1s执行此命令。

**注意:流程图中红色部分代表的含义为建议用户获取10次温度和溶氧值后计算其平均值，然后再进行一次DO(%)到DO(mg/L)的转换。**

* **计算DO（mg/L），**即把DO（%）转化成DO（mg/L）

根据公式DO(mg/L)=DO(%)\*X1\*X2\*1.4276;

其中1 ml/L = 1.4276 mg/L；

lnX1= Al + A2 100/T + A3 ln T/1OO + A4 T/1OO + S\* [B1 + B2 T/100 + B3 (T/100)^2]；

其中Al = -173.4292， B1 = - 0.033096，

A2 = 249.6339， B2 = 0.014259，

A3 = 143.3483， B3 = - 0.001700；

A4 = -21.8492；

T =273.15 + t ，T为绝对温度，t为摄氏温度；

S为盐度，纯净水中，S=0；

X2= (Phmg - u) / (760 - u);

其中Phmg= pressure \* 760 / 101.325，pressure为气压值，单位kpa；

Logu= 8.10765 - (1750.286/ (235+t)), t为摄氏温度。

以下是参考代码

#include <math.h>

float pressure = 0.0;

float Phmg= 0.0;

float t = 0.0;

float T = 0.0;

float S = 0.0;

T = 273.15 + t; //t为当前温度

X1**’** = -173.4292 // X1**’** = ln X1根据以上的经典公式

+ 249.6339\*(100/ T)

+ 143.3483\*log(T /100) //log()函数即ln(x)

+ -21.8492\*( T /100)

+S\*(-0.033096 + (0.014259\* T)/100

-0.001700\*( T /100)\*( T /100));

**X1** = exp(X1**’**); //自然对数

// log u = 8.10765 - (1750.286/ (235+t))

u’ = 8.10765 - (1750.286/ (235 + t));// u’ = log u

u = pow(10, u’); //u=10^u’

Phmg = pressure\*760/101.325;

**X2** = ((Phmg - u)/(760 - u));

DO(mg/L)= DO(%)\***X1**\***X2**\*1.4276;