

位移传感器MODBUS

通讯协议说明V1.2.1

V1.2.0 版本说明:

序号	升版描述
1	第 1 节, 修改描述: “主机发送”及“从机应答”帧 CRC 高低字节顺序。
2	第 2 节, 开放寄存器地址 0010 (传感器地址) 和 0011 (传感器波特率)。
3	第 2 节, int32 型寄存器值更改为实际值的 1000 倍, 精度提高。
4	第 2 节, 增加关于传感器波特率索引的描述。
5	第 3 节, 增加通讯示例: 修改传感器地址和修改传感器波特率。

V1.2.1 版本说明:

序号	升版描述
1	勘误: 在本文档中将“传感器值”及“传感器温度”整型数据描述由 uint32 更正为 int32 (有符号型)。将“传感器地址”及“传感器波特率”整型数据描述由 uint32 更正为 uint16。

1. MODBUS-RTU 方式通讯协议

1.1 硬件采用 RS-485，主从式半双工通讯，主机呼叫从机地址，从机应答方式通讯。

1.2 数据帧：1 个起始位，8 个数据位，1 个停止位，无校验。

1.3 功能码 03H：读寄存器值

主机发送：

1	2	3	4	5	6	7	8
ADR	03H	起始寄存器高字节	起始寄存器低字节	寄存器数高字节	寄存器数低字节	CRC 低字节	CRC 高字节

第 1 字节 ADR：从机地址码（001~254），默认地址为 2

第 2 字节 03H：读寄存器值功能码

第 3、4 字节：要读的寄存器起始地址

第 5、6 字节：要读的寄存器数量

第 7、8 字节：从字节 1 到 6 的 CRC16 校验和

从机应答：

1	2	3	4、5	6、7		M-1、M	M+1	M+2
ADR	03H	字节总数	寄存器数据 1	寄存器数据 2	。 。 。	寄存器数据 M	CRC 低字节	CRC 高字节

第 1 字节 ADR：从机地址码（001~254）

第 2 字节 03H：返回读功能码

第 3 字节：从 4 到 M（包括 4 及 M）的字节总数

第 4 到 M 字节：寄存器数据

第 M+1、M+2 字节：从字节 1 到 M 的 CRC16 校验和

1.4 功能码 10H：写寄存器值

主机发送：

1	2	3	4	5	6	7	8	9	10	11
ADR	10H	起始寄存器高字节	起始寄存器低字节	寄存器个数高字节	寄存器个数低字节	写入的数据字节数	写入的数据高字节	写入的数据低字节	CRC 低字节	CRC 高字节

第 1 字节 ADR: 从机地址码 (001~254)

- 第 2 字节 10H: 写寄存器值功能码
- 第 3、4 字节: 要写入的寄存器起始地址
- 第 5、6 字节: 要写入的寄存器个数
- 第 7 字节: 要写入的字节数
- 第 8、9 字节: 要写入的数据
- 第 10、11 字节: 从字节 1 至 9 的 CRC16 校验和

从机应答:

1	2	3	4	5	6	7	8
ADR	10H	起始寄存器高字节	起始寄存器低字节	寄存器个数高字节	寄存器个数低字节	CRC 低字节	CRC 高字节

- 第 1 字节 ADR: 从机地址码 (001~254)
- 第 2 字节 10H: 返回写功能码
- 第 3、4 字节: 起始寄存器
- 第 5、6 字节: 寄存器个数
- 第 7、8 字节: 从字节 1 到 6 的 CRC16 校验和

2. 寄存器地址说明

寄存器地址	内容说明	只读	读写	寄存器地址	内容说明	只读	读写
0000	传感器值 (int32 高 2 字节)	√		0001	传感器值 (int32 低 2 字节)	√	
0002	传感器温度 (int32 高 2 字节)	√		0003	传感器温度 (int32 低 2 字节)	√	
0004	传感器值 (int32 低 2 字节)	√		0005	传感器值 (int32 高 2 字节)	√	
0006	传感器温度 (int32 低 2 字节)	√		0007	传感器温度 (int32 高 2 字节)	√	
0008	传感器值 (float 高 2 字节)	√		0009	传感器值 (float 低 2 字节)	√	

000A	传感器温度 (float 高 2 字节)	√		000B	传感器温度 (float 低 2 字节)	√	
000C	传感器值 (float 低 2 字节)	√		000D	传感器值 (float 高 2 字节)	√	
000E	传感器温度 (float 低 2 字节)	√		000F	传感器温度 (float 高 2 字节)	√	
0010	传感器地址 (uint16 2 字节)		√	0011	传感器波特率 (uint16 2 字节)		√

说明：

(1) 寄存器 0、1 和寄存器 4、5 存放的都是整型传感器值，只是字节顺序不一样，以适用于不同的主机格式；

(2) 寄存器 2、3 和寄存器 6、7 存放的都是整型传感器温度值，只是字节顺序不一样，以适用于不同的主机格式；

(3) 寄存器 8、9 和寄存器 C、D 存放的都是浮点型传感器值，只是字节顺序不一样，以适用于不同的主机格式；

(4) 寄存器 A、B 和寄存器 E、F 存放的都是浮点型传感器温度值，只是字节顺序不一样，以适用于不同的主机格式；

(5) 为保证精度，传感器 int32 型寄存器值（0000~0007）为实际值的 1000 倍；

(6) 为简便起见，传感器波特率按照以下索引进行定义：

0：	代表 1200	1：	代表 2400
2：	代表 4800	3：	代表 9600
4：	代表 19200	5：	代表 38400
6：	代表 57600	7：	代表 115200

3. 通讯示例

(1) 通过上位机获取传感器 16 个寄存器的数值，上位机发送帧如下表：

02	03	00	00	00	10	44	35
----	----	----	----	----	----	----	----

02： 从机地址为 02

03: 读寄存器值功能码 03

00 00: 要读的寄存器起始地址为 00

00 10: 要读的寄存器个数为 16

44 35: 从字节 1 到 6 的 CRC16 校验和为 44 35

从机应答帧:

02	03	20	00	00	25	AE	00	00	7F	71	25	AE	00	00	7F	71	00	00
41	1A	56	6F	42	02	80	00	56	6F	41	1A	80	00	42	02	11	87	

02: 从机地址为 02

03: 返回读功能码 03

20: 从 4 到 35 (包含 4 及 35) 的字节总数为 20, 转化为 10 进制为 32

00 00 25 AE: 地址为 0000 与 0001 的两个寄存器 (传感器值) 转换为 10 进制后为 $9646/1000=9.646$

00 00 7F 71: 地址为 0002 与 0003 的两个寄存器 (传感器温度) 转换为 10 进制后为 $32625/1000=32.625$

25 AE 00 00: 地址为 0004 与 0005 的两个寄存器值与地址为 0000 与 0001 的两个寄存器值是相同的, 只是字节顺序不同。字节顺序由“低-高”调整为“高-低”后为 00 00 25 AE, (传感器值) 转换为 10 进制后为 $9646/1000=9.646$

7F 71 00 00: 地址为 0006 与 0007 的两个寄存器值与地址为 0002 与 0003 的两个寄存器值是相同的, 只是字节顺序不同。字节顺序由“低-高”调整为“高-低”后为 00 00 7F 71, (传感器温度) 转换为 10 进制后为 $32625/1000=32.625$

41 1A 56 6F: 地址为 0008 与 0009 的两个寄存器 (传感器值) 转换为浮点型后为 9.646101951599121

42 02 80 00: 地址为 000A 与 000B 的两个寄存器 (传感器温度) 转换为浮点型后为 32.625000000000000

56 6F 41 1A: 地址为 000C 与 000D 的两个寄存器值与地址为 0008 与 0009 的两个寄存器的值是相同的, 只是字节顺序不同。字节顺序由“低-高”调整为“高-低”后为 41 1A 56 6F, (传感器值) 转换为浮点型后为 9.646101951599121

80 00 42 02: 地址为 000E 与 000F 的两个寄存器值与地址为 000A 与 000B 的两个寄存器的值是相同的, 只是字节顺序不同。字节顺序由“低-高”调整为

“高-低”后为 42 02 80 00, (传感器温度)转换为浮点型后为 32.625000000000000

11 87: 从字节 1 到 35 的 CRC16 校验和为 11 87

(2) 通过上位机修改传感器地址, 上位机发送帧如下表:

02	10	00	10	00	01	02	00	03	F0	31
----	----	----	----	----	----	----	----	----	----	----

02: 从机地址为 02

10: 写寄存器值功能码 10

00 10: 要写入的寄存器起始地址为 00 10

00 01: 要写入的寄存器个数为 00 01

02: 要写入的字节个数为 02

00 03: 将传感器地址更改为 03

F0 31: 从字节 1 到 9 的 CRC16 校验和为 F0 31

从机应答帧:

03	10	00	10	00	01	01	EE
----	----	----	----	----	----	----	----

03: 从机地址为 03

10: 返回写功能码 10

00 10: 要写入的寄存器起始地址为 00 10

00 01: 要写入的寄存器个数为 00 01

01 EE: 从字节 1 到 6 的 CRC16 校验和为 01 EE

(3) 通过上位机修改传感器波特率, 上位机发送帧如下表:

02	10	00	11	00	01	02	00	04	B0	22
----	----	----	----	----	----	----	----	----	----	----

02: 从机地址为 02

10: 写寄存器值功能码 10

00 11: 要写入的寄存器起始地址为 00 11

00 01: 要写入的寄存器个数为 00 01

02: 要写入的字节个数为 02

00 04: 将传感器波特率更改为 00 04(00 04 为索引值, 代表波特率 19200)

B0 22: 从字节 1 到 9 的 CRC16 校验和为 B0 22

从机应答帧:

02	10	00	11	00	01	51	FF
----	----	----	----	----	----	----	----

02: 从机地址为 02

10: 返回写功能码 10

00 11: 要写入的寄存器起始地址为 00 11

00 01: 要写入的寄存器个数为 01

51 FF: 字节 1 到 6 的 CRC16 校验和为 51 FF

4. ModBus CRC 校验的 C 语言代码

```
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, 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, 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
```

};

```
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  
};  
  
unsigned short CRC16ModBus(unsigned char * puchMsg, unsigned short  
usDataLen)  
{  
    unsigned char uchCRCHi = 0xFF;
```

unsigned char uchCRCLo = 0xFF;

```
    unsigned char uIndex;

    unsigned short i = 0;
    while (usDataLen-- > 0)
    {
        uIndex = (uint8)(uchCRCHi ^ puchMsg[i++]);
        uchCRCHi = (uint8)(uchCRCLo ^ auchCRCHi[uIndex]);
        uchCRCLo = auchCRCLo[uIndex];
    }

    return (unsigned short)(( unsigned short)uchCRCHi << 8 | uchCRCLo);
}
```