修改目录：

版本号	日期	修改内容
V0.1	2009-2-16	1. 初始版本
V0.2	2010-7-13	1. 文档规范化。

目录

第一章 通讯功能概要
第二章 通讯指令
附录 1 仪表寄存器地址
附录 2 CRC 循环冗余校验算法
附录 3 IEEE4 字节浮点数传输与编码
附录 4 ASCII 码表

第一章　通讯功能概要

1.1 通信功能一览

本仪表将串行接口（RS232 或 RS485）作为附加规格。

每种接口能够使用的功能及使用这些功能（硬件及软件）的设备如下：

串行接口（RS232 或 RS485）

功能	协议	连接设备
Modbus 从机	Modbus RTU	Modbus 主机设备（测量仪器、PC、PLC 等）
仪表	专用协议	PC（数据管理软件）

1.2 使用串口通讯

本仪表的两种串行接口（RS232 和 RS485）的规格如下：

RS232 接口规格

插座类型	D-Sub 9 芯插座或者 3 点端子板[*1]
连接方式	点对点
通信方式	半双工
同步方式	起止式同步
波特率	1200，2400，4800，9600，19200，38400，57600[bps]
起始位	1 位（固定）
数据位	8 位（固定）
校验位	可选择奇校验，偶校验或 None（无校验）
停止位	1 位（固定）
接收缓冲器大小	128 字节

RS485 接口规格

插座类型	2 点端子板
连接方式	多点，总线式拓扑网络
通信方式	半双工
同步方式	起止式同步
波特率	1200，2400，4800，9600，19200，38400，57600[bps]
起始位	1 位（固定）
数据位	8 位（固定）
校验位	可选择奇校验，偶校验或 None（无校验）
停止位	1 位（固定）
接收缓冲器大小	128 字节
通信距离	最多 1.2km
终端阻抗[*2]	外部：推荐 120Ω,1/2W 电阻

注：

*1　具体插座请参考仪表说明书。

*2　使用多点连接（包括点对点连接）时，仅在链路最末端的仪表上连接一个终端电阻。不要对链路中间的仪表连接终端电阻。如果使用了转换器，打开它的终端阻抗。推荐的转换器上必须附加外部终端阻抗，也有内置终端阻抗的转换器。

第二章　通讯指令

命令代码是 16 进制的。

03H　　读取保持寄存器

描述

读取仪表保持寄存器，包括工程量和累积量。

不支持广播命令。

附录中包含了命令可以访问的寄存器列表。

发送

命令信息中包含了读取寄存器的起始地址和读取长度。

下面是一个从地址为 8 的设备读取地址 00~01 的寄存器的例子。

发送格式

名称	**数据（HEX）**
从设备地址	08H
功能码	03H
起始地址高	00H
起始地址低	00H
寄存器数量高	00H
寄存器数量低	02H
CRC 校验低	C4H
CRC 校验高	92H

返回

在返回的信息中每个寄存器包含两个字节的数据。高字节在前，低字节在后。
下面是上页发送命令的正常返回。

返回格式

名称	数据（HEX）
从设备地址	08H
功能码	03H
字节数量	04H
高字节（寄存器 00）	00H
低字节（寄存器 00）	00H
高字节（寄存器 01）	10H
低字节（寄存器 01）	52H
CRC 校验低	EFH
CRC 校验高	0EH

附录 1　仪表寄存器地址

1. 寄存器地址

工程量(浮点数)：

通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7	通道 8	通道 9
00	02	04	06	08	10	12	14	16
通道 10	通道 11	通道 12	通道 13	通道 14	通道 15	通道 16	通道 17	通道 18
18	20	22	24	26	28	30	32	34
通道 19	通道 20	通道 21	通道 22	通道 23	通道 24	通道 25	通道 26	通道 27
36	38	40	42	44	46	48	50	52
通道 28	通道 29	通道 30	通道 31	通道 32	通道 33	通道 34	通道 35	通道 36
54	56	58	60	62	64	66	68	70
通道 37	通道 38	通道 39	通道 40					
72	74	76	78					

工程量(百分比)：

通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7	通道 8	通道 9
80	81	82	83	84	85	86	87	88
通道 10	通道 11	通道 12	通道 13	通道 14	通道 15	通道 16	通道 17	通道 18
89	90	91	92	93	94	95	96	97
通道 19	通道 20	通道 21	通道 22	通道 23	通道 24	通道 25	通道 26	通道 27
98	99	100	101	102	103	104	105	106
通道 28	通道 29	通道 30	通道 31	通道 32	通道 33	通道 34	通道 35	通道 36
107	108	109	110	111	112	113	114	115
通道 37	通道 38	通道 39	通道 40					
116	117	118	119					

累积量(ulong)：

通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7	通道 8	通道 9
280	282	284	286	288	290	292	294	296
通道 10	通道 11	通道 12	通道 13	通道 14	通道 15	通道 16	通道 17	通道 18
298	300	302	304	306	308	310	312	314
通道 19	通道 20	通道 21	通道 22	通道 23	通道 24	通道 25	通道 26	通道 27
316	318	320	322	324	326	328	330	332
通道 28	通道 29	通道 30	通道 31	通道 32	通道 33	通道 34	通道 35	通道 36
334	336	338	340	342	344	346	348	350
通道 37	通道 38	通道 39	通道 40					
352	354	356	358					

注: 实际累积量 = 读取累积量(ulong)

附录 2　CRC 循环冗余校验算法

1. CRC 校验概述

CRC 校验码的基本思想是利用线性编码理论，在发送端根据要传送的 k 位二进制码序列，以一定的规则产生一个校验用的监督码（既 CRC 码）r 位，并附在信息后边，构成一个新的二进制码序列数共(k+ r)位，最后发送出去。在接收端，则根据信息码和 CRC 码之间所遵循的规则进行检验，以确定传送中是否出错。

2. CRC 校验算法

```
//CalCrc====================
//功能        计算      CRC 校验
//参数        buf       校验缓冲
//            length    检验长度
//返回        CRC 校验结果,短整形表示  HL
const uchar ucCRCHi[] =
{
    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
};
```

```
const uchar ucCRCLo[] =
{
    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
} ;
//CRC 计算
ushort CalCrc(uchar *pucData , ushort usDataLen)
{
    uchar ucCrcLo = 0xFF ;
    uchar ucCrcHi = 0xFF ;
    uchar ucIndex ;
    while(usDataLen--)
    {
        ucIndex = ucCrcLo ^ *pucData++ ;
        ucCrcLo = ucCrcHi ^ ucCRCHi[ucIndex] ;
        ucCrcHi = ucCRCLo[ucIndex] ;
    };
    return (ucCrcHi * 0x100 + ucCrcLo) ;
}
```

附录 3　IEEE4 字节浮点数传输与编码

1. IEEE4 字节浮点数编码简介

4 字节共 32 位

00-22 位　　尾数（1.尾数）

23-30 位　　阶码（2^（ 阶码 - 127））

31 位　　符号(0 正;1 负)

值=（符号）[（1.尾数）*（2^（阶码 - 127））]

2. 举例说明

1、读取数据：08 03 00 C0 00 02 C4 AE[偏移 192，长度 2 字]

2、收取数据：08 03 04 00 00 40 88 52 95[数据 00004088，长度 04]

3、分析数据：00 00 40 88[FF1 FF2 FF3 FF4]

A、前后字交换顺序 40 88 00 00[FF3 FF4 FF1 FF2]

二进制：0100 0000 1000 1000 0000 0000 0000 0000

B、尾数 =　000 1000 0000 0000 0000 0000 B = 0.0625;

阶码 =　10000001B - 127 = 129 - 127 = 2 ;

符号 =　(+) ;

C、计算数值:+[1.0625 * 2^2] = 4.25

附录 4　ASCII 码表

		高 4 位															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
低 4 位	0			SP	0	@	P		p								
	1				1	A	Q	a	q								
	2				2	B	R	b	r								
	3			#	3	C	S	c	s								
	4				4	D	T	d	t								
	5			%	5	E	U	e	u								
	6			&	6	F	V	f	v								
	7				7	G	W	g	w								
	8			(	8	H	X	h	x								
	9			)	9	I	Y	i	y								
	A	LF		*	:	J	Z	j	z								
	B		ESC	+		K		k									
	C					L		l									
	D	CR		-		M		m									
	E			.		N		n									
	F			/		O		o									