DZ9 单相多功能表通讯协议

DZ9 表使用 Modbus RTU 通信协议，进行 RS485 半双工通信，读功能号 0x03，写功能号 0x10，采用 16 位 CRC 校验，仪表对校验错误不返回，数据包间隔时间为 30ms，如接收数据包间隔时间超过 30ms 则需重新发送命令。

数据帧格式：

起始位	数据位	停止位	校验位
1	8	1	无

通信异常处理：

异常应答时，将功能号的最高位置 1。例如：主机请求功能号是 0x04,则从机返回的功能号对应项为 0x84。

错误类型码：

0x01---功能码非法：仪表不支持接收到的功能号。

0x02---数据位置非法：主机指定的数据位置超出仪表的范围。

0x03---数据值非法：主机发送的数据值超出超出仪表对应的数据范围。

一、读多寄存器

例：主机读取第 1 路报警值

第 1 路报警值 AL1 的地址编码是 0x0001，报警值采用 32 浮点数（占用 4 个字节），占用 2 个数据寄存器。十进制浮点数 200.0 的 IEEE-754 标准 16 进制内存码为 0x00004843

主机请求（读多寄存器）							
1	2	3	4	5	6	7	8
表地址	功能号	起始地址高位	起始地址低位	数据字长高位	数据字长低位	CRC 码的低位	CRC 码的高位
0x01	0x03	0x00	0x01	0x00	0x02	0x95	0xCB

从机正常应答（读多寄存器）								
1	2	3	4	5	6	7	8	9
表地址	功能号	数据字节数	数据 1 高位	数据 1 低位	数据 2 高位	数据 2 低位	CRC 码的低位	CRC 码的高位
0x01	0x03	0x04	0x00	0x00	0x48	0x43	0x8D	0xC2

功能号异常应答：（例如主机请求功能号为 0x04）

从机异常应答（读多寄存器）				
1	2	3	8	9
表地址	功能号	错误码	CRC 码的低位	CRC 码的高位
0x01	0x84	0x01	0x82	0xC0

二、写多路寄存器

例：主机写第 1 路回差 HY1（0.1）

第 1 路 HY1 的地址编码是 0x0002，回差值采用 32 位浮点数（4 字节），占用 2 个数据寄存器。十进制浮点数 0.1 的 IEEE-754 标准 16 进制内存码为 0x00CCCC3D

主机请求（写多寄存器）												
1	2	3	4	5	6	7	8	9	10	11	12	13
表地址	功能号	起始地址高位	起始地址低位	数据字长高位	数据字长低位	数据字节长度	数据1高位	数据1低位	数据2高位	数据2低位	CRC低位	CRC高位
0x01	0x10	0x00	0x02	0x00	0x02	0x04	0x00	0xCC	0xCC	0x3D	0x26	0x98

从机正常应答（写多寄存器）							
1	2	3	4	5	6	7	8
表地址	功能号	起始地址高8位	起始地址低8位	数据字长高位	数据字长低位	CRC码的低位	CRC码的高位
0x01	0x10	0x00	0x02	0x00	0x02	0xE0	0x08

DZ9 相关参数地址映射表

序号	地址映射	变量名称	字长	取值范围	读写允许	备注
0	0x0000	密码 LCK	**2**	-1999~9999	R/W	
1	0x0001	第 1 路报警值 AL1	2	-1999~9999	R/W	
2	0x0002	第 1 路报警回差 HY1	2	-1999~9999	R/W	
3	0x0003	第 2 路报警值 AL2	2	-1999~9999	R/W	
4	0x0004	第 2 路报警回差 HY2	2	-1999~9999	R/W	
5	0x0005	第 3 路报警值 AL3	2	-1999~9999	R/W	
6	0x0006	第 3 路报警回差 HY3	2	-1999~9999	R/W	
7	0x0007	第 4 路报警值 AL4	2	-1999~9999	R/W	
8	0x0008	第 4 路报警回差 HY4	2	-1999~9999	R/W	
9	0x0009	电流系数 CP	2	-1999~9999	R/W	
10	0x000A	变送上限值 BrH	2	-1999~9999	R/W	
11	0x000B	变送下限值 BrL	2	-1999~9999	R/W	
12	0x000C	电压修正值 VPS	2	-1999~9999	R/W	
13	0x000D	电流修正值 APS	2	-1999~9999	R/W	
14	0x000E	电压满量程 FSV	2	0.000~9999	R/W	
15	0x000F	电流满量程 FSA	2	0.000~9999	R/W	
16	0x0010	第 1 路报警值死区设置值	2	0.000~9999	R/W	
17	0x0011	第 2 路报警值死区设置值	2	0.000~9999	R/W	
18	0x0012	第 3 路报警值死区设置值	2	0.000~9999	R/W	
19	0x0013	第 4 路报警值死区设置值	2	0.000~9999	R/W	
保留						
20	0x0020	电压值	2	0.000~9999	R	
21	0x0021	电流值	2	0.000~9999	R	
22	0x0022	功率因数值	2	0.000~9999	R	
23	0x0023	有功功率值	2	-1.000~1.000	R	
24	0x0024	无功功率值	2	0.000~9999	R	
25	0x0025	视在功率值	2	0.000~9999	R	

26	0x0026	频率	2	0.000~400	R	
27	0x0027	电能值	2	0.000~999999	R	
保留						
28	0x0050	第 1 路报警模式 Ad1	1	0~13	R/W	注①
29	0x0051	第 1 路报警单位	1	0~1	R/W	注②
30	0x0052	第 2 路报警模式 Ad2	1	0~13	R/W	注①
31	0x0053	第 2 路报警单位	1	0~1	R/W	注②
32	0x0054	第 3 路报警模式 Ad3	1	0~13	R/W	注①
33	0x0055	第 3 路报警单位	1	0~1	R/W	注②
34	0x0056	第 4 路报警模式 Ad4	1	0~13	R/W	注①
35	0x0057	第 4 路报警单位	1	0~1	R/W	注②
36	0x0058	变送模式 brM	1	0~6	R/W	注③
37	0x0059	变送单位	1	0~1	R/W	注②
38	0x005A	波特率 bAd	1	0~1	R/W	注④
39	0x005B	表地址 Add	1	0~255	R/W	
40	0x005C	第 1 路报警死区开放	1	0~1	R/W	
41	0x005D	第 2 路报警死区开放	1	0~1	R/W	
42	0x005E	第 3 路报警死区开放	1	0~1	R/W	
43	0x005F	第 4 路报警死区开放	1	0~1	R/W	
44	0x0060	报警状态指示	1	0~F	R	注⑤
45	0x0061	表名称	1	0x02	R	
保留						

R:只读; R/W:可读写.

注①：报警模式

上限报警	通信数值	下限报警	通信数值	报警内容
VH	0	VL	1	电压
AH	2	AL	3	电流
PFH	4	PFL	5	功率因数
WH	6	WL	7	有功功率
VARH	8	VARL	9	无功功率
VAH	10	VAL	11	视在功率
HZH	12	HZL	13	频率

注②：单位

通讯数值	0	1
内容	表示为标准单位	表示为标准单位的 1000 倍

注③：变送模式

通信数值	0	1	2	3	4	5	6
菜单显示	V	A	PF	W	VAR	VA	HZ
变送内容	电压	电流	功率因数	有功功率	无功功率	视在功率	频率

注④:波特率

通信数值	0	1
菜单显示	4.8	9.6

注 5 测量状态指示

D7	D6	D5	D4	D3	D2	D1	D0
保留	保留	保留	保留	AL4	AL3	AL2	AL1

16 位 CRC 校验码获取程序

```
unsigned int G_CRC(unsigned char *p, uchar num)
{
    unsigned i,j;
    unsigned int Crc = 0xFFFF;
    for(i=0; i<num; i++)
    {
        Crc ^= (unsigned int)(p[i]);
        for(j=0; j<8; j++)
        {
            if(Crc & 1){Crc >>= 1; Crc ^= 0xA001; }
            else
                Crc >>= 1;
        }
    }
      return Crc;
}
```

4 字节字符内码转化成十进制浮点数程序

```
float   Ch_To_Float(unsigned char *ch)
{
    float result;
    unsigned char *p;
    p=(unsigned char *)&result;
    *p=*ch; *(p+1)=*(ch+1); *(p+2)=*(ch+2); *(p+3)=*(ch+3);
    return result;
}
```

十进制浮点数按转化成 4 字节字符内码程序

```
void Float_To_Char(float value, unsigned char *ch)
{
   unsigned char *p;
   p=(unsigned char*)& value;
   *ch=*p;   *(ch+1)=*(p+1);   *(ch+2)=*(p+2);   *(ch+3)=*(p+3);
}
```