

UDP6900系列可编程数控电源

Modbus编程手册

REV 00

2023. 9

UNI-T®

版权

2023 优利德中国科技有限公司

商标信息

UNI-T 是优利德中国科技有限公司的注册商标。

文档编号

1.1.0

软件版本

1.00.0905

软件升级可能更改或增加产品功能，请关注 **UNI-T** 网站获取最新版本手册或联系 **UNI-T** 升级软件。

声明

- 本产品受中国及其它国家和地区的专利（包括已取得的和正在申请的专利）保护。
- 本公司保留改变规格及价格的权利。
- 本手册提供的信息取代以往出版的所有资料。
- 本手册提供的信息如有变更，恕不另行通知。
- 对于本手册可能包含的错误，或因手册所提供的信息及演绎的功能以及因使用本手册而导致的任何偶然或继发的损失，**UNI-T** 概不负责。
- 未经 **UNI-T** 事先书面许可，不得影印、复制或改编本手册的任何部分。

产品认证

UNI-T 认证本产品符合中国国家产品标准和行业产品标准及 ISO9001: 2008 标准和 ISO14001: 2004 标准，并进一步认证本产品符合其它国际标准组织成员的相关标准。

第一章 Modbus 编程

1.1 Modbus 介绍

Modbus 是一种使用十分广泛的现场总线协议。多台从站可通过 Modbus 十分方便地与主站组网，主站可为 PC 或 PLC。Modbus 存在两个变种，分别为 Modbus-RTU 和 Modbus-ASC。

UDP6900 系列电源仅支持 Modbus-RTU 方式。

1.2 通信接口及通信设置

详细说明请参考《UDP6900 系列数控电源说明书》的“第三章 3.15.4 RS232 与 RS485 功能”

1.3 通讯数据格式

通信时数据以字（word--两字节）的形式回送，回送每个字中，高字节在前、低字节在后；如果两个字连续回传（如：浮点数或长整形），则高字在前，低字在后；即为大端格式 Big-Endian 举例：

int16_t / uint16_t: 0x1234 →→ 12 64

int32_t / uint32_t: 0x12345678 →→ 12 34 56 78

float: 5.000f (0x40A00000) →→ 40 A0 00 00

double: 5.00 (0x4014000000000000) →→ 40 14 00 00 00 00 00 00

在线数据转换平台：<https://www.binaryconvert.com/>

1.4 字（Word）与浮点数（float）相互转换

Modbus 协议中一个寄存器为 16 位，即一个字（word）。上节提到浮点数占用两个寄存器，即两个字(word)。用户在接收到字节数据后，需要将字(word)转换为浮点数，或将浮点数转换为字(word)。

以下代码是不错的转换示例：

/* 将一个浮点数转换为两个字的 C 程序 */

```
void FloatToWord(float Data,u16 *Word){
    union{
        float Data;
        unsigned char Byte[4];
    }FloatData;
    FloatData.Data=Data;
    Word[0]=(u16)FloatData.Byte[3]<<8|FloatData.Byte[2];
    Word[1]=(u16)FloatData.Byte[1]<<8|FloatData.Byte[0];
}
/* 将两个字转换为浮点数的 C 程序 */
float WordToFloat(const u16 *Word){
    union{
        float Data;
        unsigned char Byte[4];
    }
```

```

}FloatData;
FloatData.Byte[3]=(Word[0]>>8)&0xFF;
FloatData.Byte[2]=(Word[0])&0xFF;
FloatData.Byte[1]=(Word[1]>>8)&0xFF;
FloatData.Byte[0]=(Word[1])&0xFF;
return FloatData.Data;
}

```

1.5 Modbus-RTU 帧格式

1.5.1 功能码 03H 读寄存器

此命令最少可以读取 1 个字（Word）。以下为 03H 读寄存器主站与从站通讯帧格式

主站:

1 Byte	1 Byte	n Byte(s)		2 Bytes	
从站地址	03H	地址	数量	CRC16_Lo	CRC16_Hi

从站:

1 Byte	1 Byte	n Byte(s)		2 Bytes	
从站地址	03H	地址	数据	CRC16_Lo	CRC16_Hi

1.5.2 功能码 10H 写多个寄存器

此命令最少可以写入 1 个字(word)。以下为 10H 写多个寄存器的主站与从站通讯帧格式。

主站:

1 Byte	1 Byte	n Byte(s)				2 Bytes	
从站地址	10H	寄存器	寄存器数	字节数	数据	CRC16_Lo	CRC16_Hi

从站:

1 Byte	1 Byte	n Byte(s)		2 Bytes	
从站地址	10H	地址	寄存器数	CRC16_Lo	CRC16_Hi

1.5.3 错误代码说明

从站响应消息（异常）中错误代码解析如下表所示。

错误代码	名称	说明
01	非法功能码	从站不支持此功能码
02	非法数据地址	从站接收到的起始数据位置, 或起始数据位置与传输资料数目的组合是不可允许的
03	非法数据值	从站接收到数据不被允许

1.6 仪表寄存器列表

数据名称	数据格式	单位	起始地址	寄存器数量	读写	备注
电源输出设置						
电源输出开关	U16		512	1	R/W	0: 关闭输出 1: 打开输出
电压输出设置	Float	V	513	2	R/W	输出电压
电流输出设置	Float	A	515	2	R/W	输出电流
OVP 设置	Float	V	517	2	R/W	输出电压限制
OCP 设置	Float	A	519	2	R/W	输出电流限制
OVP 开关设置	U16		521	1	R/W	0:关闭 OVP 1:打开 OVP
OCP 开关设置	U16		522	1	R/W	0:关闭 OCP 1:打开 OCP
测量数据						
当前输出电压	Float	V	523	2	R	输出电压回读
当前输出电流	Float	A	525	2	R	输出电流回读
当前输出功率	Float	W	527	2	R	输出功率回读
CC/CV 状态	U16		529	1	R	0: CV 恒压模式 1: CC 恒流模式 0xFF: 电源关闭

1.7 通讯示例

非特别注明,则示例中数据均为十六进制

1. 电源输出 (Read / Write)

写寄存器:

1	2	3	4	5	6	7	8	9	10	11
01	10	02	00	00	01	02	00	01	44	50
从站地址	写多个寄存器	寄存器		寄存器数		字节数	0x0000: 电源关闭 0x0001: 电源打开		CRC	

响应:

1	2	3	4	5	6	7	8
01	10	02	00	00	01	00	71
从站地址	写多个寄存器 (响应)	寄存器		寄存器数		CRC	

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	00	00	01	85	B2
从站地址	读寄存器	寄存器		寄存器数量		CRC	

响应:

1	2	3	4	5	6	7
01	03	02	00	01	79	84
从站地址	读寄存器(响应)	数据字节数	0:电源关闭 1:电源打开		CRC	

2. 电压设置(Read / Write)

写寄存器:

1	2	3	4	5	6	7	8	9	10	11	12	13
01	10	02	01	00	02	04	40	A0	00	00	3E	E1
从站地址	写多个寄存器	寄存器	寄存器数	字节数	Float 数据(5V) 0x40A00000					CRC		

响应:

1	2	3	4	5	6	7	8
01	10	02	01	00	02	11	B0
从站地址	写多个寄存器(响应)	寄存器	寄存器数	CRC			

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	01	00	02	94	73
从站地址	读寄存器	寄存器	寄存器数量	CRC			

响应:

1	2	3	4	5	6	7	8	9
01	03	04	40	A0	00	00	EF	D1
从站地址	读寄存器(响应)	数据字节数	Float 数据(5V) 0x40A00000				CRC	

3. 电流设置(Read / Write)

写寄存器:

1	2	3	4	5	6	7	8	9	10	11	12	13
01	10	02	03	00	02	04	3F	80	00	00	A7	26
从站地址	写多个寄存器	寄存器	寄存器数	字节数	Float 数据(1A) 0x3F800000					CRC		

响应:

1	2	3	4	5	6	7	8
01	10	02	03	00	02	B0	70
从站地址	写多个寄存器(响应)	寄存器	寄存器数	CRC			

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	03	00	02	35	B3
从站地址	读寄存器	寄存器		寄存器数量		CRC	

响应:

1	2	3	4	5	6	7	8	9
01	03	04	3F	80	00	00	F7	CF
从站地址	读寄存器(响应)	数据字节数	Float 数据(1A) 0x3F800000				CRC	

4. OVP 设置(Read / Write)**写寄存器:**

1	2	3	4	5	6	7	8	9	10	11	12	13
01	10	02	05	00	02	04	42	78	00	00	BE	91
从站地址	写多个寄存器	寄存器	寄存器数	字节数	Float 数据(62V) 0x42780000				CRC			

响应:

1	2	3	4	5	6	7	8
01	10	02	05	00	02	50	71
从站地址	写多个寄存器(响应)	寄存器	寄存器数	CRC			

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	05	00	02	D5	B2
从站地址	读保持寄存器	寄存器	寄存器数量	CRC			

响应:

1	2	3	4	5	6	7	8	9
01	03	04	42	78	00	00	6E	52
从站地址	读寄存器(响应)	数据字节数	Float 数据(62V) 0x42780000				CRC	

5. OCP 设置(Read / Write)**写寄存器:**

1	2	3	4	5	6	7	8	9	10	11	12	13
01	10	02	07	00	02	04	41	78	00	00	3F	0C
从站地址	写多个寄存器	寄存器	寄存器数	字节数	Float 数据(15.5A) 0x41780000				CRC			

响应:

1	2	3	4	5	6	7	8
01	10	02	03	00	02	B0	70
从站地址	写多个寄存器(响应)	寄存器		寄存器数		CRC	

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	07	00	02	74	72
从站地址	读寄存器	寄存器		寄存器数量		CRC	

响应:

1	2	3	4	5	6	7	8	9
01	03	04	41	78	00	00	6E	16
从站地址	读寄存器(响应)	数据字节数	Float 数据(15.5A) 0x41780000				CRC	

6. OVP 开关(Read / Write)

写寄存器:

1	2	3	4	5	6	7	8	9	10	11
01	10	02	09	00	01	02	00	01	44	C9
从站地址	写多个寄存器	寄存器	寄存器数	字节数	0x0000:OVP 关闭 0x0001:OVP 打开		CRC			

响应:

1	2	3	4	5	6	7	8
01	10	02	09	00	01	D0	73
从站地址	写多个寄存器(响应)	寄存器		寄存器数		CRC	

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	09	00	01	55	B0
从站地址	读保持寄存器	寄存器		寄存器数量		CRC	

响应:

1	2	3	4	5	6	7
01	03	02	00	01	79	84
从站地址	读寄存器(响应)	数据字节数	0x0000:OVP 关闭 0x0001:OVP 打开		CRC	

7. OCP 开关 (Read / Write)

写寄存器:

1	2	3	4	5	6	7	8	9	10	11
01	10	02	0A	00	01	02	00	01	44	FA
从站地址	写多个寄存器	寄存器	寄存器数	字节数	0x0000:OVP 关闭 0x0001:OVP 打开			CRC		

响应:

1	2	3	4	5	6	7	8
01	10	02	0A	00	01	20	73
从站地址	写多个寄存器 (响应)	寄存器	寄存器数	CRC			

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	0A	00	01	A5	B0
从站地址	读保持寄存器	寄存器	寄存器数量	CRC			

响应:

1	2	3	4	5	6	7
01	03	02	00	01	79	84
从站地址	读保持寄存器 (响应)	数据字节数	0x0000:OVP 关闭 0x0001:OVP 打开		CRC	

8. 电压回读 (Read Only)

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	0B	00	02	B4	71
从站地址	读寄存器	寄存器	寄存器数量	CRC			

响应:

1	2	3	4	5	6	7	8	9
01	03	04	3F	FF	E9	54	88	78
从站地址	读寄存器 (响应)	数据字节数	Float 数据 (1.9993V) 0x3FFFE954				CRC	

9. 电流回读 (Read Only)

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	0D	00	02	54	70
从站地址	读保持寄存器	寄存器	寄存器数量	CRC			

响应:

1	2	3	4	5	6	7	8	9
01	03	04	00	00	00	00	FA	33
从站地址	读寄存器(响应)	数据字节数	Float 数据(0A) 0x00000000				CRC	

10. 功率回读(Read Only)

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	0F	00	02	F5	B0
从站地址	读保持寄存器	寄存器		寄存器数量		CRC	

响应:

1	2	3	4	5	6	7	8	9
01	03	04	00	00	00	00	FA	33
从站地址	读寄存器(响应)	数据字节数	Float 数据(0W) 0x00000000				CRC	

11. CC/CV 状态(Read Only)

读寄存器:

1	2	3	4	5	6	7	8
01	03	02	11	00	01	D5	B7
从站地址	读保持寄存器	寄存器		寄存器数量		CRC	

响应:

1	2	3	4	5	6	7
01	03	02	00	01	79	84
从站地址	读寄存器 (响应)	数据字节数	0:CV 模式 1:CC 模式 0xFF:电源关闭		CRC	


```
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 usMBCRC16( unsigned char * pucFrame, unsigned short usLen )
```

```
{
```

```
    unsigned char ucCRCHi = 0xFF;
```

```
    unsigned char ucCRCLo = 0xFF;
```

```

int          iIndex;
while( usLen-- )
{
    iIndex = ucCRCLo ^ *( pucFrame++ );
    ucCRCLo = ( UCHAR )( ucCRChi ^ aucCRCHI[iIndex] );
    ucCRChi = aucCRCLo[iIndex];
}
return ( unsigned short )( ucCRChi << 8 | ucCRCLo );
}

unsigned char SendBuf[30];
void main(void)
{
    unsigned short CRC;
    unsigned short SendLen;
    SendLen = 0;

    SendBuf[SendLen++] = 0x01;
    SendBuf[SendLen++] = 0x03;
    SendBuf[SendLen++] = 0x00;
    SendBuf[SendLen++] = 0x96;
    SendBuf[SendLen++] = 0x00;
    SendBuf[SendLen++] = 0x02;

    CRC = usMBCRC16(SendBuf,SendLen); /*开始计算 CRC */
    SendBuf[SendLen++] = CRC & 0xFF; /* CRC 低字节 */
    SendBuf[SendLen++] = (CRC >> 8) & 0xFF; /* CRC 高字节 */
}

```