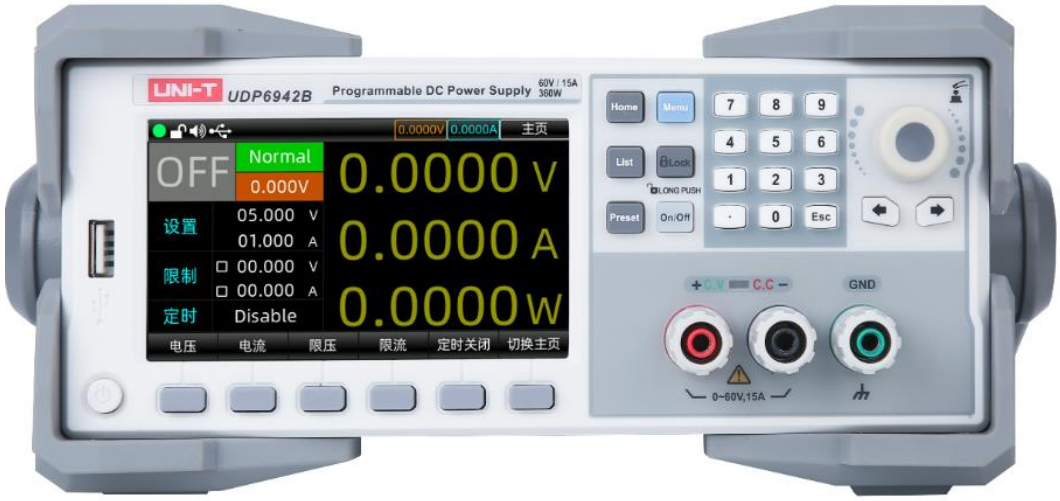
UNI-T®

Instruments.uni-trend.com



Modbus Programming Manual

UDP6900 Series Programmable Digital Power Control

Copyright

Trademark Information

UNI-T is the registered trademark of Uni-Trend Technology (China) Co., Ltd.

File Number

1.1.0

Software Version

1.00.0905

Software upgrade may have some change and add more Function, please subscribe UNI-T website to get the new version or contact UNI-T.

Statement

- UNI-T products are protected by patents (including obtained and pending) in China and other countries and regions.
- UNI-T reserves the right to change specifications and prices.
- The information provided in this manual supersedes all previous publications.
- The information provided in this manual is subject to change without notice.
- UNI-T shall not be liable for any errors that may be contained in this manual. For any incidental or consequential damages arising out of the use or the information and deductive Functions provided in this manual.
- No part of this manual shall be photocopied, reproduced or adapted without the prior written permission of UNI-T.

Product Certification

UNI-T has certified that the product conforms to China national product standard and industry product standard as well as ISO9001:2008 standard and ISO14001:2004 standard. UNI-T will go further to certificate product to meet the standard of other member of the international standards organization.

Chapter 1 Modbus

1.1 Modbus Introduction

Modbus is a widely used field bus protocol. Multiple slave machines can easily network with the host through Modbus, the host computer can be PC or PLC. Modbus has two varieties, which is Modbus-RTU and Modbus-ASC. UDP6900 only supports Modbus-RTU.

1.2 Communication Interface and Setting

The detailed explanation can refer to "Chapter 3 RS232 and RS485 Function" of UDP6900 User's Manual.

1.3 Communication Data Format

Communication data is return in the form of words (word - two bytes). In each returned word, the high byte in the first, low byte in the back;
If the two words are transmitted consecutively (e.g., floating-point or long integer), the high byte in the first and the low byte in the back; i.e., it is the Big-Endian format.

For example

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 00

Online data transformation: https://www.binaryconvert.com/

1.4 Interconversion of Word and Float

A register in Modbus protocol is 16 bits, which is a word. The previous section mentioned that floating-point occupies two registers, i.e., two words. After receiving byte data, the user needs to convert a word to a floating-point or a floating-point number to a word.
The following code is a good example for interconversion of word and float point.

/* C program for converting a floating point number to two words */

```
void FloatToWord(float Data,u16 *Word){
    union{
        float Data;
        unsigned char Byte[4];
    }Floating data ;
    Floating data .Data=Data;
    Word[0]=(u16)Floating data .Byte[3]<<8|Floating data .Byte[2];
    Word[1]=(u16)Floating data .Byte[1]<<8|Floating data .Byte[0];
```

```
}
/* C program for converting two words to a floating point number */
float WordToFloat(const u16 *Word){
    union{
        float Data;
        unsigned char Byte[4];
    }Floating data ;
    Floating data .Byte[3]=(Word[0]>>8)&0xFF;
    Floating data .Byte[2]=(Word[0])&0xFF;
    Floating data .Byte[1]=(Word[1]>>8)&0xFF;
    Floating data .Byte[0]=(Word[1])&0xFF;
    return Floating data .Data;
}
```

1.5 Modbus-RTU

1.5 1 Function Code 03H Read register

This command can read at least one word. The following example issues the communication frame format of 03H read register master-slave.

Master station

1 Byte	1 Byte	n Byte(s)		2 Bytes	
Slave station	03H	Address	Quantity	CRC16_Lo	CRC16_Hi

Slave station

1 Byte	1 Byte	n Byte(s)		2 Bytes	
Slave station	03H	Address	Data	CRC16_Lo	CRC16_Hi

1.5.2 Function Code 10H Write multiple registers

This command can write at least one word. The following example issues the communication frame format of 10H read register master-slave.

Master station

1 Byte	1 Byte	n Byte(s)				2 Bytes	
Slave station	10H	Register	Register number	Byte number	Data	CRC16_Lo	CRC16_Hi

Slave station

1 Byte	1 Byte	n Byte(s)		2 Bytes	
Slave station	10H	Address	Register number	CRC16_Lo	CRC16_Hi

1.5.3 Description of Error Code

Error code parsing for respond message of slave station (abnormal) as shown in the following table.

Error Code	Name	Description
01	Illegal function code	The slave station does not support this function code.
02	Illegal data address	The starting data position received by the slave station, or the combination of the starting data position and the number of transmitted data is not allowable.
03	Illegal data value	Data received from the slave station is not allowed.

1.6 Register List of Meter

Data Name	Data Format	Unit	Start Address	Register number	Read/Write	Remarks
Power Output Setting						
Switch of power output	U16		512	1	R/W	0: turn off output 1: turn on output
Voltage output setting	Float	V	513	2	R/W	Output the voltage
Current output setting	Float	A	515	2	R/W	Output the current
OVP setting	Float	V	517	2	R/W	Limits of voltage output
OCP setting	Float	A	519	2	R/W	Limits of current output
Switch setting of OVP	U16		521	1	R/W	0: turn off OVP 1: turn on OVP
Switch setting of OCP	U16		522	1	R/W	0: turn off OCP 1: turn on OCP
Measurement Data						
Currently output voltage	Float	V	523	2	R	Output voltage read-back
Currently current voltage	Float	A	525	2	R	Output current read-back
Currently power voltage	Float	W	527	2	R	Output power read-back
CC/CV mode	U16		529	1	R	0: CV mode 1: CC mode 0xFF: turn off the power

1.7 Communication Example

Unless otherwise specified, the data are all hexadecimal in the following example.

1. Power Output (Read / Write)

Write Register

1	2	3	4	5	6	7	8	9	10	11
01	10	02	00	00	01	02	00	01	44	50
Slave station	Write multiple registers Register	Register		Register number		Byte number	0x0000:turn off the power 0x0001:turn on the power		CRC	

Response

1	2	3	4	5	6	7	8
01	10	02	00	00	01	00	71
Slave station	Write multiple registers (response)	Register		Register number		CRC	

Read Register

1	2	3	4	5	6	7	8
01	03	02	00	00	01	85	B2
Slave station	Read register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7
01	03	02	00	01	79	84
Slave station	Read register (response)	Data byte number	0: turn off the power; 1: turn on the power		CRC	

2. Voltage Setting (Read / Write)

Write Register

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
Slave station	Write multiple registers Register	Register		Register number		Byte numbe r	Floating data (5V) 0x40A00000				CRC	

Response

1	2	3	4	5	6	7	8
01	10	02	01	00	02	11	B0
Slave station	Write multiple registers (response)	Register		Register number		CRC	

Read Register

1	2	3	4	5	6	7	8
01	03	02	01	00	02	94	73
Slave station	Read register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7	8	9
01	03	04	40	A0	00	00	EF	D1
Slave station	Read register (response)	Data byte number	Floating data (5V) 0x40A00000				CRC	

3. Current Setting (Read / Write)

Write Register

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
Slave station	Write multiple registers Register	Register		Register number		Byte number	Floating data (1A) 0x3F800000				CRC	

Response

1	2	3	4	5	6	7	8
01	10	02	03	00	02	B0	70

Slave station	Write multiple registers (response)	Register	Register number	CRC

Read Register

1	2	3	4	5	6	7	8
01	03	02	03	00	02	35	B3
Slave station	Read register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7	8	9
01	03	04	3F	80	00	00	F7	CF
Slave station	Read register (response)	Data byte number	Floating data (1A) 0x3F800000				CRC	

4. OVP Setting (Read / Write)

Write Register

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
Slave station	Write multiple registers Register	Register		Register number		Byte number	Floating data (62V) 0x42780000				CRC	

Response

1	2	3	4	5	6	7	8
01	10	02	05	00	02	50	71
Slave station	Write multiple registers (response)	Register		Register number		CRC	

Read Register

1	2	3	4	5	6	7	8
01	03	02	05	00	02	D5	B2
Slave station	Read-hold register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7	8	9
01	03	04	42	78	00	00	6E	52
Slave station	Read register (response)	Data byte number	Floating data (62V) 0x42780000				CRC	

5. OCP Setting (Read / Write)

Write Register

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
Slave station	Write multiple registers Register	Register		Register number		Byte number	Floating data (15.5A) 0x41780000				CRC	

Response

1	2	3	4	5	6	7	8
01	10	02	03	00	02	B0	70
Slave station	Write multiple registers (response)	Register		Register number		CRC	

Read Register

1	2	3	4	5	6	7	8
01	03	02	07	00	02	74	72
Slave station	Read register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7	8	9
01	03	04	41	78	00	00	6E	16
Slave station	Read register (response)	Data byte number	Floating data (15.5A) 0x41780000				CRC	

6. OVP Switch (Read / Write)

Write Register

1	2	3	4	5	6	7	8	9	10	11
01	10	02	09	00	01	02	00	01	44	C9
Slave station	Write multiple registers Register	Register		Register number		Byte number	0x0000: turn off OVP 0x0001: turn on OVP		CRC	

Response

1	2	3	4	5	6	7	8
01	10	02	09	00	01	D0	73

Slave station	Write multiple registers (response)	Register	Register number	CRC

Read Register

1	2	3	4	5	6	7	8
01	03	02	09	00	01	55	B0
Slave station	Read-hold register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7
01	03	02	00	01	79	84
Slave station	Read register (response)	Data byte number	0x0000: turn off OVP 0x0001: turn on OVP		CRC	

7. OCP Switch (Read / Write)

Write Register

1	2	3	4	5	6	7	8	9	10	11
01	10	02	0A	00	01	02	00	01	44	FA
Slave station	Write multiple registers Register	Register		Register number		Byte number	0x0000: turn off OVP 0x0001: turn on OVP		CRC	

Response

1	2	3	4	5	6	7	8
01	10	02	0A	00	01	20	73
Slave station	Write multiple registers (response)	Register		Register number		CRC	

Read Register

1	2	3	4	5	6	7	8
01	03	02	0A	00	01	A5	B0
Slave station	Read-hold register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7
01	03	02	00	01	79	84
Slave station	Read-hold register (response)	Data byte number	0x0000: turn off OVP 0x0001: turn on OVP		CRC	

8. Voltage Read-back (Read Only)

Read Register

1	2	3	4	5	6	7	8
01	03	02	0B	00	02	B4	71
Slave station	Read register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7	8	9
01	03	04	3F	FF	E9	54	88	78
Slave station	Read register (response)	Data byte number	Floating data (1.9993V) 0x3FFFE954				CRC	

9. Current Read-back (Read Only)

Read Register

1	2	3	4	5	6	7	8
01	03	02	0D	00	02	54	70
Slave station	Read-hold register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7	8	9
01	03	04	00	00	00	00	FA	33
Slave station	Read register (response)	Data byte number	Floating data (0A) 0x00000000				CRC	

10. Power Read-back (Read Only)

Read Register

1	2	3	4	5	6	7	8
01	03	02	0F	00	02	F5	B0
Slave station	Read-hold register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7	8	9
01	03	04	00	00	00	00	FA	33

Slave station	Read register (response)	Data byte number	Floating data (0W) 0x00000000	CRC

11. CC/CV Mode (Read Only)

Read Register

1	2	3	4	5	6	7	8
01	03	02	11	00	01	D5	B7
Slave station	Read-hold register	Register		Register number		CRC	

Response

1	2	3	4	5	6	7
01	03	02	00	01	79	84
Slave station	Read register (response)	Data byte number	0:CV mode 1:CC mode 0xFF: turn off the power		CRC	

Appendix 1 CRC Calculation

```
const unsigned char aucCRCHi[ ] = {
    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 unsigned char aucCRCLo[ ] = {
        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); /* Start to caclulating CRC */
    SendBuf[SendLen++]=CRC&0xFF;        /* CRC LSB */
    SendBuf[SendLen++]=(CRC>>8)&0xFF;    /* CRC MSB */
}
```