SRNE 硕日 光伏储能管控专家	**Specification code:**	
	Version: V1.00	**Secret level: Confidential**
	Effective date:	**Pages:**

SRNE PV Inverter RS485
MODBUS Communication Protocol

ShenZhen ShuoRi New Energy Technology Co.,Ltd.

Date: 2017-07-06

Version	**Revision reason**	**Revision description**	**Revised by**	**Revision time**
V1.00	Create	/		2017-07-06

This document specifies the requirements of the external 485 communication protocols of the SRNE off-grid, grid-connected and energy storage inverters.

The protocol framework is referenced from the Modbus protocol, which actually limits the number of registers that can be read and written once to no more than 20.

The underlying format is fixed at 9600,n,8,1, i.e. baud rate 9600, 8 data bits, no checksum.

Connection method: One master, multiple slaves, star connection, with each slave address set using keyboard in advance. At any time, the inverter supports a universal address, so a new address for the inverter can be set via the universal address (at which point it must be connected one-to-one), and the physical layer limits the maximum number of slaves to 32.

Data format:

Slave address	Functional domain	Data length or data content	CRC check
1 byte	1 byte	N bytes, related to commands	2 bytes, **the check range is all data from the slave address until the CRC check**
0~F7H, 0 is the broadcast address, **F7H is the universal address** (universal address is used to communicate without knowing the address of the inverter; after receiving the command of universal address, the inverter can return data without comparing with the local address. The universal address cannot be used in multi machine communication situations)	03H: Read data 10H: Write data Other: invalid		

For the sake of uniformity, all data is organized in bytes.

Special note: The result calculated by CRC is 16-bit data. In the actual transmission, the low bytes should be transmitted first and then the high bytes. This is a different order of transmission from the rest of the protocol and should be noted in particular.

Attachment: CRC calculation methods:

The CRC domain detects the entire content of the frame, i.e. all data from the slave address until the CRC check. The slave recalculates the CRC check data and compares it with the check values in the received data stream to determine the validity of the received data. The CRC domain is two-byte 16-bit binary data.

There are three methods to carry out CRC calibration, the results of which are the same and can be freely chosen according to the actual situation.

Method 1: CRC software calculation, using the following program for CRC calculation.

```
Unsigned int crc_cal_value(unsigned char*data_value,unsigned char data_length)
{
int i;
unsigned int crc_value=0xffff;
while(data_length--)
{
```

```
        crc_value^=*data_value++;
        for(i=0;i<8;i++)
        {
            if(crc_value&0x0001)
                crc_value=(crc_value>>1)^0xa001;
            else
                crc_value=crc_value>>1;
         }
    }
    return(crc_value);
    }
```

Method 2: CRC lookup table: byte lookup table, using the following table and program for CRC calculation.

```
/* CRC value of high bytes*/
static unsigned int 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,
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,
};

/* CRC value of low bytes*/
static unsigned int 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,
};


/* Function returns CRC with unsigned short type */
//unsigned char *puchMsg ; //Message used to calculate CRC
```

```
//unsigned short usDataLen ; // Number of bytes in message
unsigned int CRC16(unsigned int * puchMsg,unsigned int usDataLen)
{
    unsigned int uchCRCHi = 0xFF; /* High-byte initialization of CRC */
    unsigned int uchCRCLo = 0xFF; /* Low-byte initialization of CRC */
    unsigned int uIndex ;                /* CRC lookup table index */

    while (usDataLen--)                  /* Complete the entire message buffer*/
    {
      uIndex = uchCRCLo ^ *puchMsg++ ; /* 计算 CRC */
      uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
      uchCRCHi = auchCRCLo[uIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

Method 3: CRC lookup table: word lookup table, using the following table and program for CRC calculation.

```
Static unsigned int tblCRC[] =
{
    0x0000,0xC1C0,0x81C1,0x4001,0x01C3,0xC003,0x8002,0x41C2,
    0x01C6,0xC006,0x8007,0x41C7,0x0005,0xC1C5,0x81C4,0x4004,
    0x01CC,0xC00C,0x800D,0x41CD,0x000F,0xC1CF,0x81CE,0x400E,
    0x000A,0xC1CA,0x81CB,0x400B,0x01C9,0xC009,0x8008,0x41C8,
    0x01D8,0xC018,0x8019,0x41D9,0x001B,0xC1DB,0x81DA,0x401A,
    0x001E,0xC1DE,0x81DF,0x401F,0x01DD,0xC01D,0x801C,0x41DC,
    0x0014,0xC1D4,0x81D5,0x4015,0x01D7,0xC017,0x8016,0x41D6,
    0x01D2,0xC012,0x8013,0x41D3,0x0011,0xC1D1,0x81D0,0x4010,
    0x01F0,0xC030,0x8031,0x41F1,0x0033,0xC1F3,0x81F2,0x4032,
    0x0036,0xC1F6,0x81F7,0x4037,0x01F5,0xC035,0x8034,0x41F4,
    0x003C,0xC1FC,0x81FD,0x403D,0x01FF,0xC03F,0x803E,0x41FE,
    0x01FA,0xC03A,0x803B,0x41FB,0x0039,0xC1F9,0x81F8,0x4038,
    0x0028,0xC1E8,0x81E9,0x4029,0x01EB,0xC02B,0x802A,0x41EA,
    0x01EE,0xC02E,0x802F,0x41EF,0x002D,0xC1ED,0x81EC,0x402C,
    0x01E4,0xC024,0x8025,0x41E5,0x0027,0xC1E7,0x81E6,0x4026,
    0x0022,0xC1E2,0x81E3,0x4023,0x01E1,0xC021,0x8020,0x41E0,
    0x01A0,0xC060,0x8061,0x41A1,0x0063,0xC1A3,0x81A2,0x4062,
    0x0066,0xC1A6,0x81A7,0x4067,0x01A5,0xC065,0x8064,0x41A4,
    0x006C,0xC1AC,0x81AD,0x406D,0x01AF,0xC06F,0x806E,0x41AE,
    0x01AA,0xC06A,0x806B,0x41AB,0x0069,0xC1A9,0x81A8,0x4068,
    0x0078,0xC1B8,0x81B9,0x4079,0x01BB,0xC07B,0x807A,0x41BA,
    0x01BE,0xC07E,0x807F,0x41BF,0x007D,0xC1BD,0x81BC,0x407C,
    0x01B4,0xC074,0x8075,0x41B5,0x0077,0xC1B7,0x81B6,0x4076,
    0x0072,0xC1B2,0x81B3,0x4073,0x01B1,0xC071,0x8070,0x41B0,
    0x0050,0xC190,0x8191,0x4051,0x0193,0xC053,0x8052,0x4192,
    0x0196,0xC056,0x8057,0x4197,0x0055,0xC195,0x8194,0x4054,
    0x019C,0xC05C,0x805D,0x419D,0x005F,0xC19F,0x819E,0x405E,
    0x005A,0xC19A,0x819B,0x405B,0x0199,0xC059,0x8058,0x4198,
    0x0188,0xC048,0x8049,0x4189,0x004B,0xC18B,0x818A,0x404A,
    0x004E,0xC18E,0x818F,0x404F,0x018D,0xC04D,0x804C,0x418C,
    0x0044,0xC184,0x8185,0x4045,0x0187,0xC047,0x8046,0x4186,
    0x0182,0xC042,0x8043,0x4183,0x0041,0xC181,0x8180,0x4040,
};
```

```
/* Function returns CRC with unsigned short type */
//unsigned char *puchMsg; // Message used to calculate CRC
//unsigned short usDataLen; // Number of bytes in message
unsigned int CRC16(unsigned int * puchMsg,unsigned int usDataLen)
{
    unsigned int uchCRCHi = 0xFF; /* High-byte initialization of CRC */
    unsigned int uchCRCLo = 0xFF; /* Low-byte initialization of CRC */
    unsigned int uIndex ;                /* CRC lookup table index */
    unsigned int hi,low;

    while (usDataLen--)                  /* Complete the entire message buffer*/
    {
    uIndex = uchCRCLo ^ *puchMsg++ ; /*  计算 CRC */
    hi = tblCRC[uIndex] >> 8;
    low = tblCRC[uIndex] & 0xff;
    uchCRCLo = uchCRCHi ^ hi;
    uchCRCHi = low;
}
return (uchCRCHi << 8 | uchCRCLo) ;
}
```

Read data format

Master sends:

Slave address	Functional domain	Data domain				CRC check
1 byte	1 byte	4 bytes				2 bytes
Actual address	03H	Address of register to be read high byte	Address of register to be read low byte	Number of registers high byte, 00H	Number of registers low byte. N<=20, allowing up to 20 registers of 40 bytes of data to be read at a time	

Slave returns:

Slave address	Functional domain	Data content	CRC check
1 byte	1 byte	2*N+1 bytes	2 bytes
Actual address	03H	Data content is shown in the table below.	/

1 byte data length (bytes)	N registers, 2*N bytes of data			
Number of bytes of data returned	Data high byte	Data low byte	Data high byte	Data low byte

Return in case of error:

Slave address	Functional domain	Error code	CRC check
1 bytes	0x83	See the error code table.	2 bytes

Write data format

Master sends:

Slave address	Functional domain	Data length	CRC check
1 bytes	1 bytes	5+2*N bytes	2 bytes
Actual address	10H	N<=20, Data content is shown in the table below	/

Register address high byte	Register address low byte	Number of registers high byte	Number of registers low byte	Number of bytes	Data content
Address high bit	Address low bit	Number high bit	Number low bit	2 * N	Content of N registers, 2*N bytes in total
1	1	1	1	1	

Slave returns:

Slave address	Functional domain	Data length	CRC check
1 byte	1 byte	4 bytes	2 bytes
Actual address	10H	Data content is shown in the table below	/

Register address high bit	Register data address low bit	Number of write registers high bit	Number of registers low bit
Address high byte	Address low byte	Number high byte	Number low byte
1	1	1	1

Return in case of error:

Slave address	Functional domain	Error code	CRC check
1 bytes	0x90	See the error code table.	2 bytes

Error code table:

Codes	Name	Meaning
01H	Illegal command	When the command code received from the upper computer is an impermissible operation, this may be because the function code is only applicable to new device and is not implemented in this device; in addition, it may also be that the slave processes the request in an error state.
02H	Illegal data address	For an inverter, the request data address of the upper computer is a unauthorized address; in particular, the combination of the register address and the number of transmitted bytes is invalid.
03H	Illegal data value	When the received data domain contains an impermissible value. This value indicates an error in the remaining structure in the combined request. Note: It in no way means that the data items being submitted for storage in the register have a value other than what the application expects.
04H	Operation failed	During parameter write operation, the parameter is set to be invalid; for example, the function input terminal cannot be set repeatedly.
05H	Password error	The password written at the password check address is different from the password set by the user of register 0x1305.
06H	Data frame error	When the length of data frame is incorrect or RTU format CRC bits are different from the check calculation number of lower computer in frame information sent by the upper computer.
07H	Parameter is read only	The parameters changed in the upper computer write operation are read-only parameters.
08H	Parameter cannot be changed during running	The parameters changed in the upper computer write operation are the parameters that cannot be changed during running.
09H	Password protection	When the upper computer reads or writes, if user password is set while password is not unlocked, it will report that system is locked.
0AH	Length error	The number of registers required to read during the read process exceeds 12. The number of register data issued during writing exceeds 12
0BH	Permission denied	No enough permissions to do this operation