MPS-200/300S Series Modbus Protocol Communication Protocol

Register definition:

Register address	Name	03 Function code	06 Function code	10 Function code	Explain
0X0000	Remote Mode	Y	Y	Y	Remote control status 0: Local mode 1: Remote mode
0X0001	V_SET	Y	N	Y	Voltage setting register float type
0X0003	A_SET	Y	N	Y	Current setting register float type
0X0005	V_MIN	Y	N	Y	Minimum voltage setting register float type
0X0007	V_MAX	Y	N	Y	Maximum voltage setting register float type
0X0009	A_MIN	Y	N	Y	Minimum current setting register float
0X000B	A_MAX	Y	N	Y	Maximum current setting register float type
0X000D	OVP_SET	Y	N	Y	Overvoltage parameter setting register float
0X000F	OCP_SET	Y	N	Y	Overcurrent parameter setting register float
0X0011	OVP_STATE	Y	Y	Y	Overvoltage status setting register 0: protection off 1: protection on
0X0012	OCP_STATE	Y	Y	Y	Overcurrent status setting register 0: protection off 1: protection on
0X0013	OUTPUT	Y	Y	Y	Output status setting register 0: output off 1: output on
0X0014	STATE	Y	Y	Y	Status read clear register type u16, refer to status register bit definition table
0X0015	V_OUT	Y	N	N	Output actual voltage register float type
0X0017	A_OUT	Y	N	N	Output actual current register float type
0X0019	CV/CC	Y	N	N	Output working status register 0: CV status 1: CC status

Status register (0X0014) bit definition table:

Bit	Name	Attribute	Explain
2	OTP_FLAG	**W/R**	1 is the over-temperature protection flag. Write 1 to clear the over-temperature flag (when the temperature returns to the safe value).
1	OCP_FLAG	**W/R**	1 is the overcurrent protection flag. Write 1 to clear the overcurrent flag.
0	OVP_FLAG	**W/R**	1 is the overvoltage protection flag. Write 1 to clear the overvoltage flag.

Note that all float parameters take two register addr

Function code table and interval time are as follows:

Function Code	Corresponding function	Time between two operations
0X03	Read through one or more registers command	N * 5ms
0X06	Single write one register command	10ms
0X10	Command to concatenate one or more register	N * 5ms

Note: N is the number of registers

The communication protocol format is as follows:

Function code 0 X03

PC Send: 8 Byte

Address	Function Code	Start Address High Byte	Start address low byte	Number of registers high byte	Low byte of register number	Low byte of CRC16 check code	High byte of CRC16 check code
0X01	0X03						

Power return: 5 + N * 2 Byte

Address	Function Code	Data length byte	Return data High byte	Return data Low byte	Return data N + 1 High byte	Return data N + 1 Low byte	Low byte of CRC16 check code	High byte of CRC16 check code
0X01	0X03							

Example 1: Read the output voltage value (0x001D)
PC send: 01 03 00 1D 00 02 54 0D (start address 0x001D, length 2 registers, CRC16 check result 0x0D54)
Power return: 01 03 04 40 A0 00 00 EF D1 (5.000V for 0X40A00000, CRC16 verification result for 0XD1EF)

Example 2: Read the output voltage and current values (0x001D ~ 0x001F)
PC send: 01 03 00 1D 00 04 D4 0F (start address is 0x001D, length is 4 registers, CRC16 check result is 0x0FD4)
Power supply return: 01 03 08 40 A0 00 00 40 00 00 24 2D (0X40A00000 represents 5.00V, 0X40000000 represents 2.0000A, and 0X2D24 represents CRC16 verification result)

Function code 0X06

PC Send: 8 Byte

Address	Function Code	Write Address High Byte	Write address low byte	Write Data High Byte	Write Data Low Byte	Low byte of CRC16 check code	High byte of CRC16 check code
0X01	0X06						

Power Return: 8 Byte

Address	Function Code	Write Address High Byte	Write address low byte	Write Data High Byte	Write Data Low Byte	Low byte of CRC16 check code	High byte of CRC16 check code
0X01	0X06						

Note that all float-type parameter registers do not support code 06, which only supports single register operations

Example 1: Turn on the power output (0X001B)
PC sends: 01 06 00 1B 00 01 38 0D (write address is 0x001B, write data is 0x0001, CRC16 check result is 0x0D38)
Power return: 01 06 00 1B 00 01 38 0D (write address is 0x001B, write data is 0x0001, CRC16 check result is 0x0A48)

Function code 0X10

PC Send: 9 + N * 2 Byte

Address	Function Code	Start Address High Byte	Start address low byte	Number of registers high byte	Low byte of register number	Data length Byte	Write data High byte	Write data Low byte	Write data N + 1 High byte	Write data N + 1 Low byte	Low byte of CRC16 check code	High byte of CRC16 check code
0X01	0X10											

Power Return: 8 Byte

Address	Function Code	Start Address High Byte	Start address low byte	Data length High byte	Data length Low byte	Low byte of CRC16 check code	High byte of CRC16 check code
0X01	0X10						

Example 1: Set the voltage and current values simultaneously (0x0001 ~ 0x0003)
PC send: 01 10 00 01 00 04 08 40 A0 00 40 00 FA 43 (start address is 0x0001, length is 4 registers, 0x40A00000 is 5.00 V/40 00 is 1.0000 A, Verification result of CRC16 is 0x43FA)
Power return: 01 10 00 01 00 04 90 0A (start address 0x0001, 4 registers, CRC16 check result 0x0A90)

CRCR16 Calculation Code:

```
const unsigned char    CRCHTalbe[] =
{
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    CRCLTalbe[] =
{
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 int crc16(unsigned char *DData,unsigned char len)
{
    unsigned char CRCHi = 0XFF;
    unsigned char CRCLo = 0XFF;
    unsigned int wIndex;
    unsigned int CRC_DData;
    while(len--)
     {
     wIndex = CRCLo ^ *DData++;
     CRCLo = CRCHi ^ CRCHTalbe[wIndex];
     CRCHi = CRCLTalbe[wIndex];
     }
CRC_DData = CRCHi;
CRC_DData<<= 8;
CRC_DData |=CRCLo;
    return CRC_DData;
}
```