MPS-X00XH-1 Series Modbus Protocol Communication Protocol

The list of register addresses and corresponding function codes is as follows:

Register address	Name	Function	Function code 0 X03	Function code 0X06	Function code 0X10	Assignment range
0X0000	Remote Mode	Remote operation mode	Y	Y	Y	0: local touch 1: remote mode
0X0001	CH1 V_Set	CH1 voltage parameter setting	Y	Y	Y	0 ~ Vmax step 1mV
0X0002	CH1 A_Set	CH1 current parameter setting	Y	Y	Y	0 ~ Amax step 1mA
0X0003	CH1 OVP_Set	CH 1 overvoltage protection parameter setting	Y	Y	Y	0 ~ Vmax step 1mV
0X0004	CH1 OCP_Set	CH 1 overcurrent protection parameter settings	Y	Y	Y	0 ~ Amax step 1mA
0X0005	CH1 OVP_State	CH 1 overvoltage protection status settings	Y	Y	Y	0: Disable 1: Enable
0X0006	CH1 OCP_State	CH 1 overcurrent protection status settings	Y	Y	Y	0: Disable 1: Enable
0X0007	CH1 OUT_State	CH 1 output status settings	Y	Y	Y	0: Off 1: Output
0X0008			Y	Y	Y	
0X0009			Y	Y	Y	
0X000A			Y	Y	Y	
0X000B			Y	Y	Y	
0X000C			Y	Y	Y	
0X000D			Y	Y	Y	
0X000E			Y	Y	Y	
0X000F	CH1 V_Out	Actual output value of CH1 voltage	Y	N	N	0 ~ Vmax step 1mV
0X0010	CH1 A_Out	Actual output value of CH1 current	Y	N	N	0 ~ Amax step 1mA
0X0011	CH1 WorkMode	CH 1 working status	Y	N	N	0: CV mode 1: CC mode
0X0012			Y	N	N	
0X0013			Y	N	N	
0X0014			Y	N	N	

Note: All operations are based on the remote operation mode (the value of register 0X0000 is 0X0001)!

* Based on local operating mode after power re-up!

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 a single register (0 X 0000)

PC send: 01 03 00 00 00 01 84 0 A (start address 0x0000, length 1 register, CRC16 check result 0x0A84)

Power return: 01 03 02 00 01 79 84 (0x0001 indicates remote operation mode, and 0X8479 is CRC16 verification result)

Example 2: Read multiple registers (0x0001 ~ 0X0002)

PC send: 01 03 00 01 00 02 95 CB (start address 0x0001, length 2 registers, CRC16 check result 0xCB95)

Power supply return: 01 03 04 01 F4 03 E8 79 84 (0X01F4 represents 0.500V, 0X03E8 represents 1.000A, and 0X8479 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						

Example 1: Write Register Status (0 * 0000)
PC sends: 01 06 00 00 00 01 48 0A (write address is 0x0000, write data is 0x0001, CRC16 check result is 0x0A48)
Power return: 01 06 00 00 00 01 48 0 A (write address is 0x0000, 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: Write multiple register States (0x0001 to 0x0002)
PC send: 01 10 00 01 00 02 04 01 F4 03 E8 72 D3 (start address 0x0001, length 2 registers, write data 0.500V/1.000A,Verification result of CRC16 is 0XD372)
Power return: 01 10 00 01 00 04 90 0A (the start address is 0x0001, the data length is 4 bytes, and the CRC16 check result is 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;
}
```