EN: This Datasheet/Document is presented by the manufacturer.

Please visit our website for pricing and availability at www.hestore.hu.

NTA8A01 NTC Temperature Sensor Setting Protocol

Master sends

Address code(1)	Function code(1)	Register Address(2)	Read quantity(2)	CRC16 checksum (2)
	03 Read			
	06 Write			

Read-only register read function code 03				
Register Address	Register contents	Number of bytes	unit	Remark
0x0000	Temperature value	2	0.1 ℃	is not connected or is broken: The data is 0XF555 (-2731)
Read and write registers Read function code 03 Write function code 06				
0x0002	485 Address	2		Read address 0XFF Write address 1-247
0x000 3	Baud rate	2		0~4 0:1200 1:2400 2:4800 3:9600 (default) 4:19200
0x0004	Temperature correction value	2	0.1 ℃	2 bytes, input a positive number to increase the temperature; input a negative number to decrease the temperature

Serial port baud rate: 9600 (can be set), N, 8, 1

Modbus RTU communication protocol:

1. **Read the current temperature**

Send Frame

Address code(1)	Function code(1)	Register Address(2)	Read quantity(2)	CRC16 checksum (2)

R

eturn Frame

Address code(1)	Function code(1)	Length(1)	Data(n)	CRC16 checksum (2)

F

unction code 0x03
Register address: 0x0000
Read quantity: 0x0001

The returned temperature data is two bytes long, with the high bit first and the low bit last. Convert these two bytes to a decimal number and divide by 10 to get the current temperature value. When the highest bit is 1, it indicates a negative value. In this case, you need to add 1 to the value, or directly subtract 65536 from the value to get the current temperature value. The following example illustrates:
Send frame (address is 1): 01 03 00 00 00 01 84 0A
Return frame: 01 03 02 00 DB F8 1F
01 address code, 03 function code, 02 length, F8 1F crc16 checksum
00DB is the temperature value, the highest bit is 0, so the temperature is positive, convert it to decimal = 219 , and then divide it by 10: 21.9 is the current temperature value;
Return frame: 01 03 02 FF 90 F2 3F
FF 90 is the temperature value, the highest bit is 1, so the temperature is negative. Convert it to decimal = 65424 , then subtract 65536 = -11.2 to get the current temperature value.

2. **Read 485 address code** :
Send Frame

Address code(1)	Function code(1)	Register Address(2)	Read quantity(2)	CRC16 checksum (2)

R

eturn Frame

Address code(1)	Function code(1)	Length(1)	Data(n)	CRC16 checksum (2)

A

ddress code 0xff
Function code 0x03
Register address: 0x0002
Read quantity: 0x0001
For example:
Send frame: FF 03 00 02 00 01 30 14
Return frame: FF 03 02 00 01 50 50
FF address code, 03 function code, 02 length, 01 current module address, 50 50 crc16 checksum
Note: When using this command, only one temperature module can be

connected to the 485 bus. If there is more than one, an error will occur!

3. **Set 485 address:**

Send Frame

Address code(1)	Function code(1)	Register Address(2)	Setting contents(2)	CRC16 checksum (2)

Return Frame

Address code(1)	Function code(1)	Register Address(2)	Register value (2)	CRC16 checksum (2)

Function code: 0x06
Register address: 0x0002
Setting content: 2 bytes (value 1-247)
For example, the current 485 address is 1, and you want to change the 485 address to 3:
Send frame (address is 1) 01 06 00 02 00 03 68 0B
Return frame: 01 06 00 02 00 03 68 0B

4. **Read the serial port baud rate** :

Send Frame

Address code(1)	Function code(1)	Register Address(2)	Read quantity(2)	CRC16 checksum (2)

Return Frame

Address code(1)	Function code(1)	Length(1)	Data(n)	CRC16 checksum (2)

Function code 0x03
Register address: 0x0003
Read quantity: 0x0001
For example:
Send frame (address is 1): 01 03 00 03 00 01 74 0A
Return frame: 01 03 02 00 03 F8 45
01 address code, 03 function code, 02 length, 03 means the current baud rate is 9600, F8 45 crc16 checksum
Baud rate corresponding numbers: 0:1200 1:2400 2:4800 3:9600 4:19200

5. **Set the serial port baud rate:**

Send Frame

Address	Function	Register	Setting	CRC16 checksum

code(1)	code(1)	Address(2)	contents(2)	(2)

Return Frame

Address code(1)	Function code(1)	Register Address(2)	Register value (2)	CRC16 checksum (2)

Function code: 0x06
Register address: 0x0003
Setting content: 2 bytes (value 0-4)
For example, to change the baud rate to 4800:
Send frame (address is 1) 01 06 00 03 00 02 F8 0B
Return frame: 01 06 00 03 00 02 F8 0B
Baud rate corresponding numbers: 0:1200 1:2400 2:4800 3:9600 4:19200 5:Restore factory settings
Note: 1. When using this command, the module must be powered on again before the baud rate is updated!
2 When the baud rate is 5, the factory settings can be restored.
For example: 01 06 00 03 00 05 B9 C9

6. **Read the temperature correction value** :

The module may have an error with the actual temperature, and this correction value can correct the error. The unit is 0.1℃. If the correction value is a positive number, add this value to the current temperature, if it is a negative number, subtract this value.

Setting it to 0 disables this feature.

Send Frame

Address code(1)	Function code(1)	Register Address(2)	Read quantity(2)	CRC16 checksum (2)

R

eturn Frame

Address code(1)	Function code(1)	Length(1)	Data(n)	CRC16 checksum (2)

A

ddress code 0x01~0xFE
Function code 0x03
Register address: 0x0004
Read quantity: 0x0001
Return data: Celsius, this value needs to be divided by 10.

Example 1:
Send frame: 01 03 00 04 00 01 C5 CB
Return frame: 01 03 02 00 64 B9 AF 0064 is the correction value , which is expressed in decimal as 100, and then divided by 10 = 10.0℃;

Example 2:
Send frame: 01 03 00 04 00 01 C5 CB
Return frame: 01 03 02 FF F1 38 30 FF F1 is the correction value, which is -15 in decimal, and then divided by 10 = -1.5℃;

7. **To set the temperature correction value** :
   If the temperature of the module deviates from the actual temperature, this value can be used to correct it. A positive number is addition and a negative number is subtraction.
   Send Frame

Address code(1)	Function code(1)	Register Address(2)	Setting value(2)	CRC16 checksum (2)

   Return Frame

Address code(1)	Function code(1)	Register Address(2)	Setting value(2)	CRC16 checksum (2)

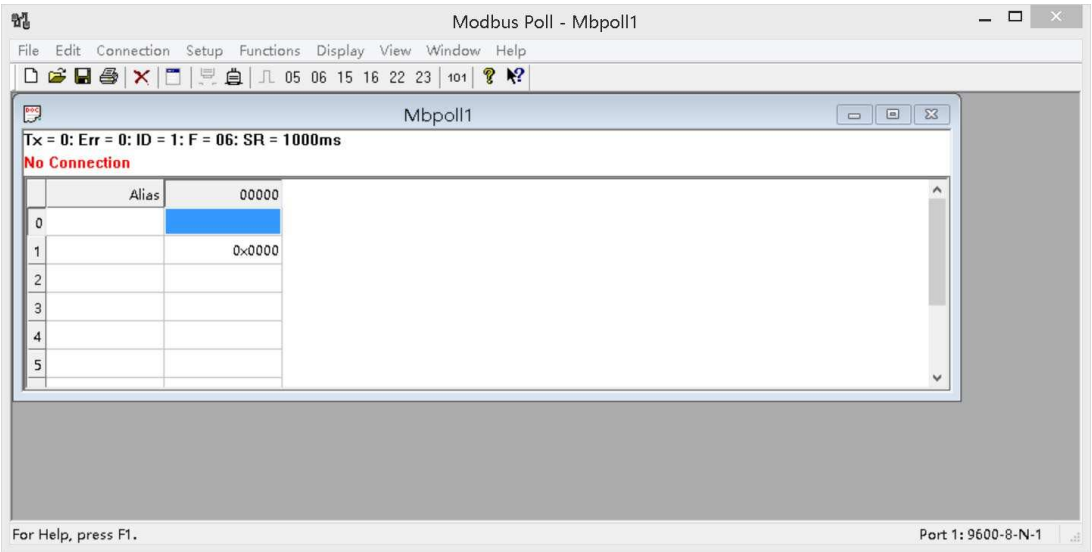
   Address code 0x01~0xFE
   Function code 0x06
   Register address: 0x0004
Setting value: 2 bytes, the highest bit indicates the sign of positive and negative values, 0 indicates positive, 1 indicates negative, unit is 0.1 ℃. When the highest bit is 1, it indicates a negative value. In this case, you need to add 1 to this value, or you can directly subtract 65536 from this value to get the current temperature value. Disable correction value and set the register to "0X0000"

Example 1: The offset value is set to 2.0 ° C
Send frame: 01 06 00 04 00 14 C 8 04
Return frame: 01 06 00 04 00 14 C 8 04 The return frame is the same as the send frame.

Example 2: The offset value is set to -3.0 ° C, 65536-30=65506 =0XFFE2
Send frame: 01 06 00 04 FF E2 09 B2
Return frame: 01 06 00 04 FF E2 09 B2 The return frame is the same as the sending frame.

Example 3: Disable the correction value to set the register to "0X0000"
Send frame: 01 06 00 04 00 00 C8 0B
Return frame: 01 06 00 04 00 00 C8 0B The return frame is the same as the sending frame.

MODBUS commands can be input using " Modbus Poll" , as shown below



You can also use the serial port hyperterminal input, as shown below

CRC16 verification procedure:

const unsigned char code auchCRCHi[256] = {
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 code auchCRCLo[256] = {
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 CRC_16(unsigned char *str,unsigned int usDataLen)
{
 unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
 unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
       unsigned uIndex ; /* will index into CRC lookup table */
       while (usDataLen--)/* pass through message buffer */
       {
          uIndex = uchCRCHi ^ *str++ ; /* calculate the CRC */
          uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex];
          uchCRCLo = auchCRCLo[uIndex] ;
       }
 return (uchCRCHi << 8 | uchCRCLo) ;
}
```