



NTA8A01_232_5V

5V TTL232 No sensor



NTA8A01_485_5V

5V RS485 No sensor



NTA8A01_485_12V

12V RS485 No sensor



A Version

B3950 10K 1% NTC Thermal sensor



B Version

B3950 10K 1% NTC Thermal sensor

NTA8A01 Module Description:

Working voltage: 5V version: DC 3.7-5.5V; 12V version: 6-25V

Working current: 7-13MA

MODBUS RTU protocol, 03 read command, 06 write command.

Serial port baud rate: 9600 (default), N, 8, 1

By modifying the 485 address, up to 247 modules can be cascaded (more than 16 please use R485 repeater)

Adapted sensor: B3950 10K 1% NTC thermistor

Temperature measurement range (using our NTC sensor): -20 ° C to +125 ° C.

Temperature measurement accuracy (using our NTC sensor): 1%.

Size: 30 X 15 X 6MM

Weight: 5V version 2g, 12V version 2.5g

MODBUS RTU protocol please refer to : " NTA8A01 modbus rtu protocol"

NTC sensor specifications:

Product Name: 5*25mmNTC Waterproof Probe

Total length : 50CM

Resistance value: 10K

Resistance accuracy: $\pm 1\%$

Resistance B value: $3950 \pm 1\%$

Temperature range: -20°C - 125°C

Wire size: 2651 26# parallel line temperature 105°C

Plug model: XH2.54-2P

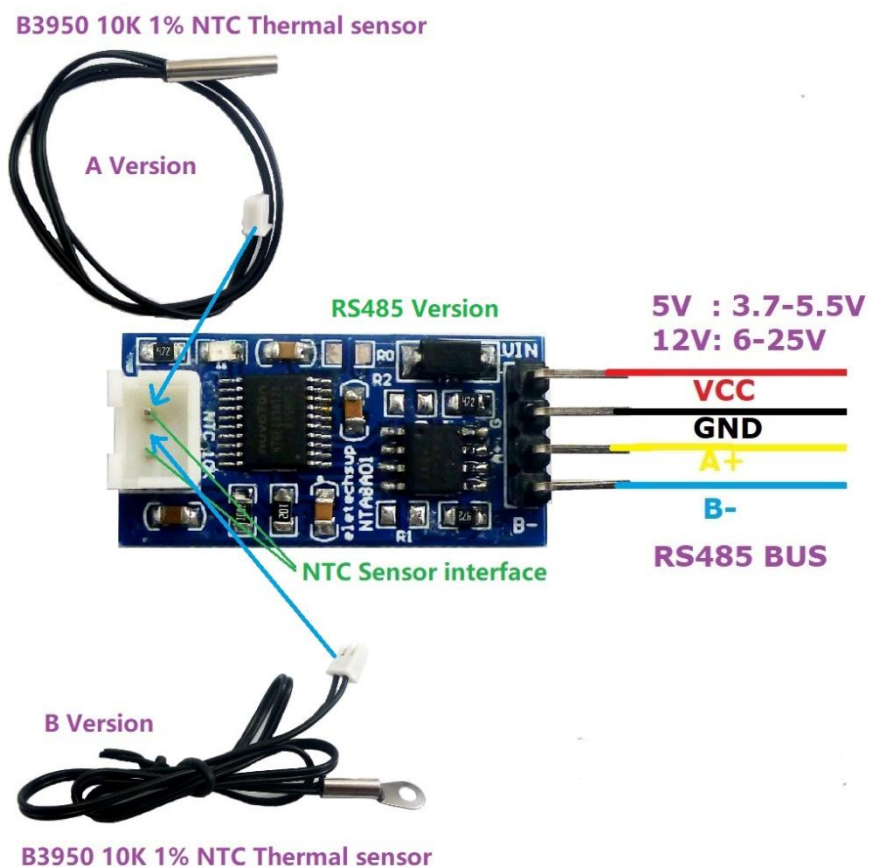
Probe: A version without mounting holes, B version with mounting holes (4MM/M4)

Note:

1 This module does not have an NTC sensor. The NTC sensor needs to be purchased separately (B3950 10K 1%); The module's measured resistance value is 1%, and the theoretical temperature range is -273 - 200°C . The final temperature measurement accuracy and temperature range are determined by NTC thermistor.

2 Be sure to use the correct sensor (B3950 10K 1% NTC), otherwise the temperature will not be accurate

Wiring diagram:

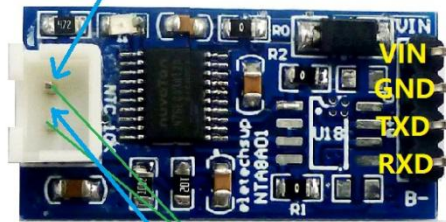


B3950 10K 1% NTC Thermal sensor



A Version

TTL232 Version



NTC Sensor interface

5V : 3.7-5.5V

VCC

GND

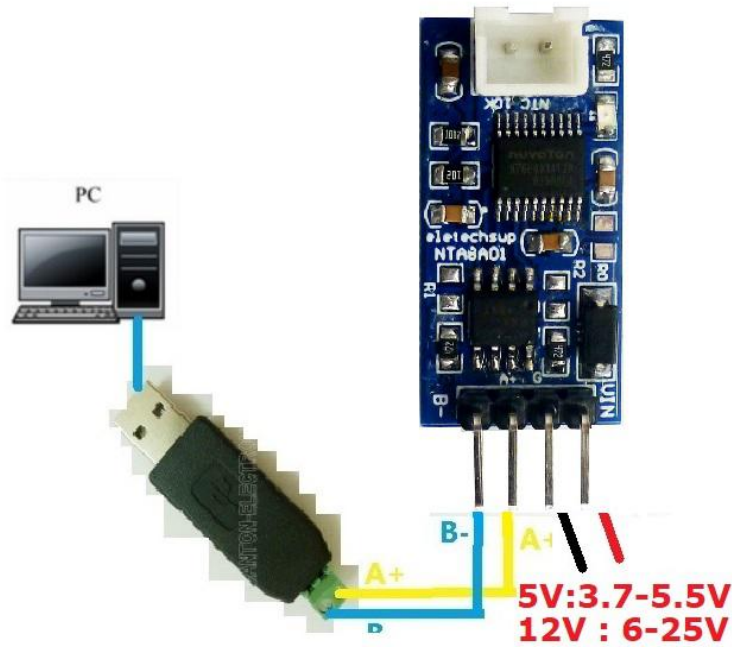
RXD

TXD

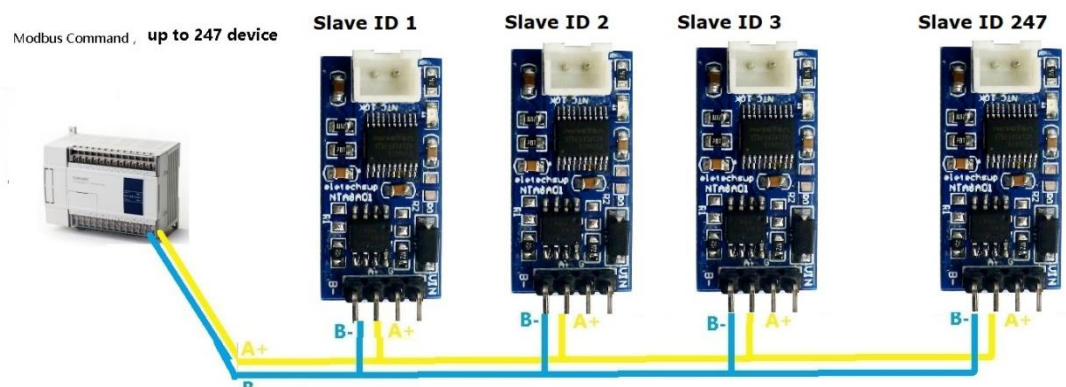
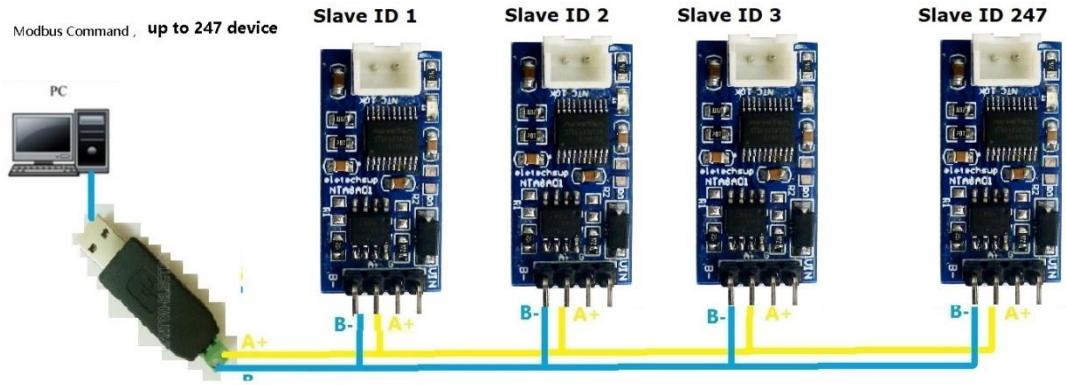
B Version



B3950 10K 1% NTC Thermal sensor



5V:3.7-5.5V
12V : 6-25V



NTA8A01 modbus rtu protocol

Function code

RS485 address (Station address) (1)	Function (1)	Register address (2)	Read number (2)	CRC16 (2)
	03 Read			
	06 Write			

Read-only register, Read Function code is 03

Register address	Register contents	Number of bytes	Units	Remarks
0x0000	Temperature value	2	0.1℃	When the data is 0XF555(-2731), it indicates no sensor or error

Read / write register; Read function code is 03 ,Write function code is 06

0x0002	RS485 address (Station address)	2		Read Address 0xFF Write Address 1-247
0x0003	Baud rate	2		0~4 0:1200 1:2400 2:4800 3:9600 (default) 4:19200 5: Factory reset
0x0004	Temperature correction value	2	0.1℃	>0 Temperature increase <0 temperature decrease Default : 0

Serial baud rate: 9600 (default), N, 8, 1

Modbus RTU Communication protocol:

1. Read temperature

Send data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Read number (2)	CRC16 (2)
---	-----------------	-------------------------	-----------------	-----------

Returns data

RS485 address (Station address)	Function (1)	Number of bytes (1)	data (n)	CRC16 (2)
------------------------------------	-----------------	------------------------	----------	-----------

(1)				
-----	--	--	--	--

Function code 0x03

Register address: 0x0000

Read number: 0x0001

The return of the temperature data is two bytes, High-bit in the former and low-bit in the post, convert it to decimal and divided by 10, is the current temperature value; The highest bit 1 indicates a negative value, this value directly subtracting 65536, is the current temperature value.

For example:

send data(RS485 address is 1): 01 03 00 00 00 01 84 0A

Returns data: 01 03 02 00 DB F8 1F

01 RS485 address, 03 Function, 02 length, F8 1F crc16

00DB is the temperature value, the highest bit is 0, so the temperature is positive, it is converted to decimal = 219, $219/10=21.9$ is the current temperature value;

Returns data: 01 03 02 FF 90 F2 3F

FF90 is the temperature value, the highest bit is 1, so the temperature is negative, it is converted to decimal = 65424, $(65424-65536)/10=-11.2$ is the current temperature value

2. Read RS485 address

Send data

RS485 address (Broadcast address) (1)	Function (1)	Register address (2)	Read number (2)	CRC16(2)
--	-----------------	-------------------------	-----------------	--------------

Returns data

RS485 address (Broadcast address) (1)	Function (1)	Number of bytes (1)	data (n)	CRC16(2)
--	-----------------	------------------------	----------	--------------

Broadcast address 0xff

Function code 0x03

Register address: 0x0002

Read number: 0x0001

For example:

send data: FF 03 00 02 00 01 30 14

Returns data: FF 03 02 00 01 50 50

FF Broadcast address, 03 Function, 02 length, 01 is the current module RS485 address, 50 50 crc16

Note: When using this command, only one temperature module can be connected to the RS485 bus, more than one will be wrong!

3. Write RS485 address

Send data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Setting Content (2)	CRC16(2)
---	-----------------	-------------------------	------------------------	--------------

Returns data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Register value (2)	CRC16(2)
---	-----------------	----------------------------	-----------------------	--------------

Function code 0x06

Register address: 0x0002

Setting Content: 2Bytes(1-247)

For example, The current RS485 address is 1, We need to change the RS485 address to 3:

send data(RS485 address is 1): 01 06 00 02 00 03 68 0B

Returns data: 01 06 00 02 00 03 68 0B

4. Read baud rate

Send data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Read number (2)	CRC16(2)
---	-----------------	-------------------------	-----------------	--------------

Returns data

RS485 address (Station address) (1)	Function (1)	Number of bytes (1)	data (n)	CRC16(2)
---	-----------------	------------------------	----------	--------------

Function code 0x03

Register address: 0x0003

Read number: 0x0001

For example:

send data(RS485 address is 1): 01 03 00 03 00 01 74 0A

Returns data: 01 03 02 00 03 F8 45

01 RS485 address, 03 Function, 02 length, 38 F1 crc16

03 means the current baud rate is 9600bps

Baud rate corresponds to the number: 0: 1200 1: 2400 2: 4800 3: 9600 4: 19200

5. Write RS485 address

Send data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Setting Content (2)	CRC16(2)
---	-----------------	-------------------------	------------------------	--------------

Returns data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Register value (2)	CRC16(2)
---	-----------------	----------------------------	-----------------------	--------------

Function code 0x06

Register address: 0x0003

Setting Content: 2Bytes(0-4)

For example, Change the baud rate to 4800bps:

send data(RS485 address is 1): 01 06 00 03 00 02 F8 0B

Returns data: 01 06 00 03 00 02 F8 0B

Baud rate corresponds to the number: 0: 1200 1: 2400 2: 4800 3: 9600 4: 19200

5: Factory reset

Note: 1 The baud rate will be updated when the module is powered up again!

2 The factory setting can be restored when the baud rate corresponding to the number is 5.

For example: 01 06 00 03 00 05 B9 C9

6. Read the temperature correction value:

Send data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Read number (2)	CRC16(2)
---	-----------------	-------------------------	-----------------	--------------

Returns data

RS485 address (Station address)	Function (1)	Number of bytes (1)	data (n)	CRC16(2)
------------------------------------	-----------------	------------------------	----------	--------------

(1)				
-----	--	--	--	--

Function code 0x03

Register address: 0x0004

Read number: 0x0001

Return data: Celsius, you need to divide this value by 10.

The temperature sensor may have an error with the actual temperature. This correction value can correct the error. The unit is 0.1 °C. If the correction value is a positive number, the value is added at the current temperature, and if it is a negative number, the value is subtracted. Setting it to 0 disables this feature.

For example 1:

send data(RS485 address is 1): 01 03 00 04 00 01 C5 CB

Returns data: 01 03 02 00 64 B9 AF

0064 is the correction value, which is expressed as decimal in 100, divided by 10 = 10.0 °C;

For example 2:

send data(RS485 address is 1): 01 03 00 04 00 01 C5 CB

Returns data: 01 03 02 FF F1 38 30

FF F1 is the correction value, which is expressed as decimal in -15, divided by 10 = -1.5° C;

7. Set the temperature correction value

If the temperature of the module deviates from the actual temperature, it can be corrected with this value. >0 temperature increases, <0 temperature decreases.

Send data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Setting Content (2)	CRC16(2)
---	-----------------	-------------------------	------------------------	--------------

Returns data

RS485 address (Station address) (1)	Function (1)	Register address (2)	Register value (2)	CRC16(2)
---	-----------------	----------------------------	-----------------------	--------------

Function code 0x06

Register address: 0x0004

Setting Content: 2Bytes

The highest digit indicates a positive or negative sign, 0 indicates positive, and 1 indicates negative, and the unit is 0.1 °C. When the highest bit is 1, it indicates a

negative value. In this case, you need to add 1 to this value. You can also subtract 65536 from the value, which is the current temperature value. Prohibit the correction value to set the register to "0X0000"

For example 1: offset value is set to 2.0 °C

Send frame: 01 06 00 04 00 14 C8 04

Return frame: 01 06 00 04 00 14 C8 04 The return frame is the same as the send frame.

For 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 send frame.

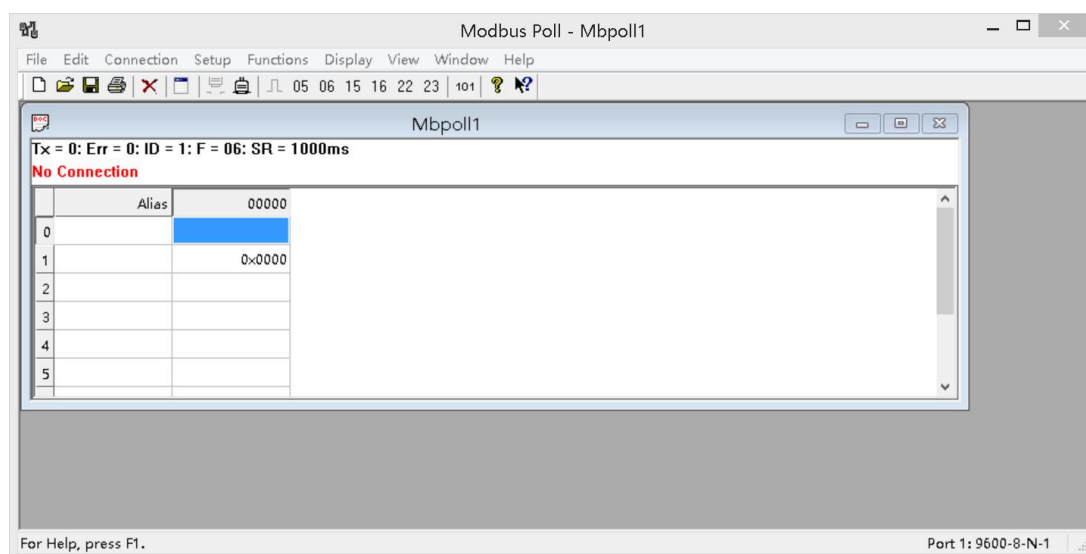
For example 3: Prohibit the correction value and set the register to "0X0000"

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 send frame.

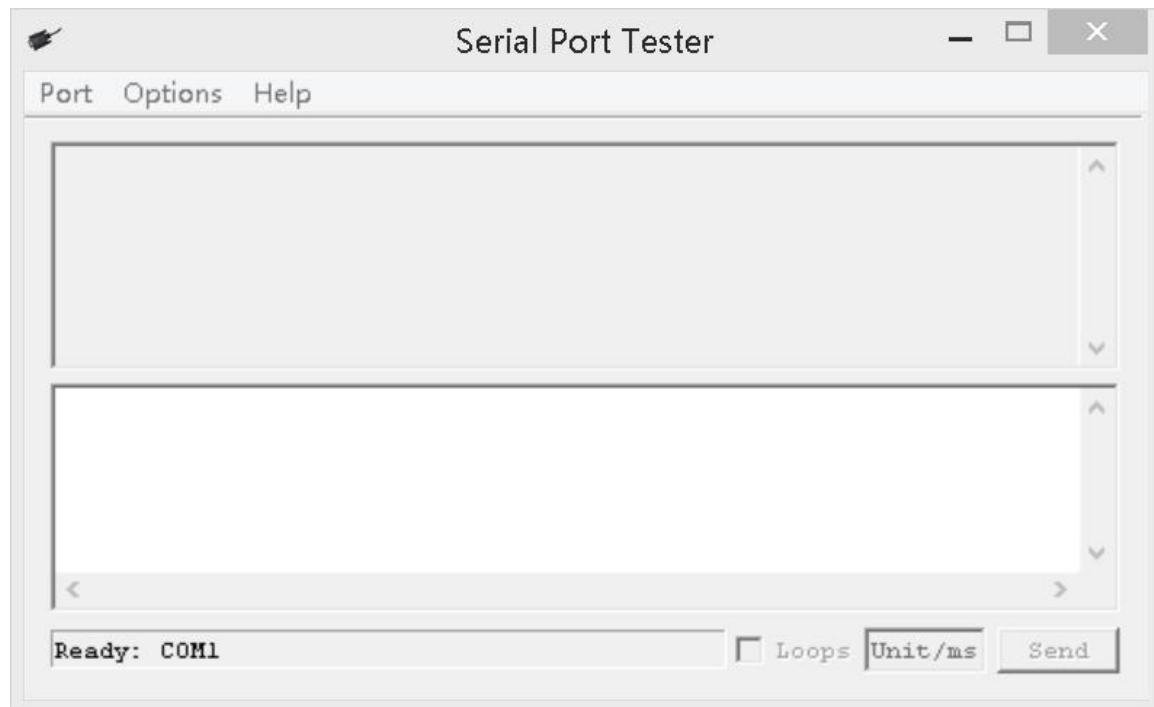
MODBUS commands you can use "Modbus Poll" input, as shown below

(CRC check generated automatically)



You can also use HyperTerminal serial input, as shown below

(Manually add CRC check)



CRC check code(C51 MCU):

```
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) ;  
}
```