**N4DIG08 Modbus RTU Protocol(4 Input 4 Output)**

**MODBUS command (function code 06 is Control command,03 is Read status command)**

Note :

1 MODBUS command must be HEX

2 Slave ID (device address) must be correct, the default slave address is 01, and the Slave ID is set to see the bottom.

3 If you don't remember the Slave ID, use the command Read Slave ID : FF 03 00 FE 00 01 F0 24

As a 4-channel input 4-channel output function, the jumper must be shorted as follows: 

9600 Band ,8 Data bits,None Parity,1 Stop Bit。

**Function code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RS485 address  (Station address)  (1) | Function (1) | Register address (2) | Read number (2) | CRC16 (2) |
|  | 03 Read |  |  |  |
|  | 06 Write |  |  |  |
|  | 16(0x10)  Write multiple registers |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Function code | Register address | Register contents | Number of bytes | Register value | Remarks |
| 03 06  16(0X10) | 0x0005-0x0008 | Output port status | 2 | 0X0000  0X0001 | 0X0000 Relay Close  0X0001 Relay Open |
| 03 | 0x0081-0x0084  (129-132) | Input port status | 2 | 0X0000  0X0001 | 0X0000 No input  0X0001 Has input |
| 03 06 | 0x00FA  (250) | Onput level | 2 | 0X0000  0X0001 | 0X0000 PNP Low level output (default)  0X0001 NPN High level onput |
| 03 06 | 0x00FB  (251) | Input and output relationship | 2 | 0X0000-  0X0003 | 0x0000 Unrelated  0x0001 Self-locking relationship (default)  0x0002 Interlocking relationship  0x0003 Momentary relationship  Other values are the same as 0 |
| 03 06 | 0x00FC  (252) | Input level | 2 | 0X0000  0X0001 | 0X0000 Low level input (default)  0X0001 High level input |
| 03 06 | 0x00FD  (253) | Input port status automatic reporting function | 2 | 0-255 | 0: Query function (default) 1-255: Automatically report, the unit is second.  1: Report every 1 second  2: Report every 2 seconds  10: Report every 10 seconds Maximum interval of 255 seconds |
| 03 06 | 0x00FE | RS485 address  (Station address) | 2 |  | Read Address 0XFF  Write Address 1-247 |
| 03 06 | 0x00FF | Baud rate | 2 |  | 0~4 0:1200  1:2400 2:4800  3:9600（default）  4:19200  5: Factory reset |

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

**Modbus RTU Communication protocol：**

1. **Output port status**

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) |

RS485 address:0x01-0xFE

Function code : 0x03 read, 0x06/0x10 (set)

Register address：0x0005-0x0008 5-8 channel Onput Port Status

Read number：0x0001-0x0004

For details, please refer to " N4DIG08 Modbus RTU Protocol(8 Output)"

1. **Read input port status**

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) |

RS485 address:0x01-0xFE

Function code 0x03

Register address：0x0081-0x0084 1-4 channel input port status

Read number：0x0001-0x0004

0X000 No input, when “low level input” is selected, the port is high level or floating; when “high level input” is selected, the port is low level.

0X0001 has input, when “low level input” is selected, the port is low level; when “high level input” is selected, the port is high level or floating.

For example: Read channel 1 port value:

Send data(RS485 address is 1)：01 03 00 81 00 01 D4 22

Returns data：01 03 02 00 01 79 84

01 RS485 address，03 Function，02 length，79 84 crc16

00 01 means there is input. If low level input is selected (default), the port level at this time is low level. If high level input is selected, the port level at this time is high level (or floating).

For example: Read channel 2 port value:

Send data(RS485 address is 1)：01 03 00 82 00 01 24 22

Returns data：01 03 02 00 00 B8 44

01 RS485 address，03 Function，02 length，B8 44 crc16

00 00 means no input. If low level input is selected (default), the port level at this time is high level (or floating). If high level input is selected, the port level at this time is low level.

1. **Read Input level selection register**

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) |

RS485 address:0x01-0xFE

Function code 0x03

Register address：0x00FC

Read number：0x0001

For example:

send data(RS485 address is 1)：01 03 00 FC 00 01 44 3A

Returns data 1 ：01 03 02 00 00 B8 44 Low level input (default)

Returns data 2 ：01 03 02 00 01 79 84 High level input

1. **Set Input level selection register (4 channels set at the same time)**

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) |

RS485 address:0x01-0xFE

Function code 0x06

Register address：0x00FC

Setting Content：1Bytes

0X0000 Low level input (default): Low level is valid input, high level or floating invalid input 0X0001 High level input: high level or floating valid input, low level invalid input

Set low level input:

Send data (RS485 address is 1) 01 06 00 FC 00 00 49 FA

Return data: 01 06 00 FC 00 00 49 FA

Set high level input:

Send data (RS485 address is 1) 01 06 00 FC 00 01 88 3A

Return frame: 01 06 00 FC 00 01 88 3A

1. **Read Input port status reporting function**

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) |

RS485 address:0x01-0xFE

Function code 0x03

Register address：0x00FD

Read number：0x0001

For example:

send data(RS485 address is 1)：01 03 00 FD 00 01 15 FA

Returns data：01 03 02 00 00 B8 44

01 RS485 address，03 Function，02 length，00 means query function ,

B8 44 crc16

1. **Set Input port status reporting function(8 channels set at the same time)**

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) |

RS485 address:0x01-0xFE

Function code 0x06

Register address：0x00FD

Setting Content：1Bytes

For example : For example, the current query function should be changed to automatic reporting:

Automatically report in 1 second, send frame (address is 1) 01 06 00 FD 00 01 D9 FA

Automatically report in 2 second, send frame (address is 1) 01 06 00 FD 00 02 99 FB

Automatically report in 3 second, send frame (address is 1) 01 06 00 FD 00 03 58 3B

Automatically report in 4 second, send frame (address is 1) 01 06 00 FD 00 04 19 F9

Automatically report in 5 second, send frame (address is 1) 01 06 00 FD 00 05 D8 39

Automatically report in 10 second, send frame (address is 1) 01 06 00 FD 00 0A 98 3D

Disable reporting function: send frame (address is 1) 01 06 00 FD 00 00 18 3A

1. **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：0x00FE

Read number：0x0001

For example:

send data：FF 03 00 FE 00 01 F0 24

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!

1. **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) |

RS485 address(Slave ID) : 0x01~0xFE

Function code 0x06

Register address：0x00FE

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 FE 00 03 A8 3B

Returns data：01 06 00 FE 00 03 A8 3B

1. **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) |

RS485 address(Slave ID) : 0x01~0xFE

Function code 0x03

Register address：0x000FF

Read number：0x0001

For example:

send data(RS485 address is 1)：01 03 00 FF 00 01 B4 3A

Returns data：01 03 02 00 03 F8 45

01 RS485 address，03 Function，02 length，F8 45 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

1. **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) |

RS485 address(Slave ID) : 0x01~0xFE

Function code 0x06

Register address：0x00FF

Setting Content：2Bytes(0-4)

For example, Change the baud rate to 4800bps:

send data(RS485 address is 1)：01 06 00 FF 00 02 38 3B

Returns data：01 06 00 FF 00 02 38 3B

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 FF 00 05 79 F9

1. **Read input and output relation register**

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：0x00FB

Read number：0x0001

For example:

send data(RS485 address is 1)：01 03 00 FB 00 01 F5 FB

Returns data：01 03 02 00 01 79 84

01 RS485 address，03 Function，02 length，15 FA crc16

Register corresponding value:

0x0000 Unrelated

0x0001 Self-locking relationship (default)

0x0002 Interlocking relationship

0x0003 Momentary relationship

Other values are the same as 0

1. **Write input and output relation register**

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：0x00FB

Setting Content：2Bytes(0-3)

For example, Set the input and output to be unrelated, and change the register value to 0X0000:

Send data(RS485 address is 1)：01 06 00 FB 00 00 F8 3B

Returns data：01 06 00 FB 00 00 F8 3B

Register corresponding value:

0x0000 Unrelated

0x0001 Self-locking relationship (default)

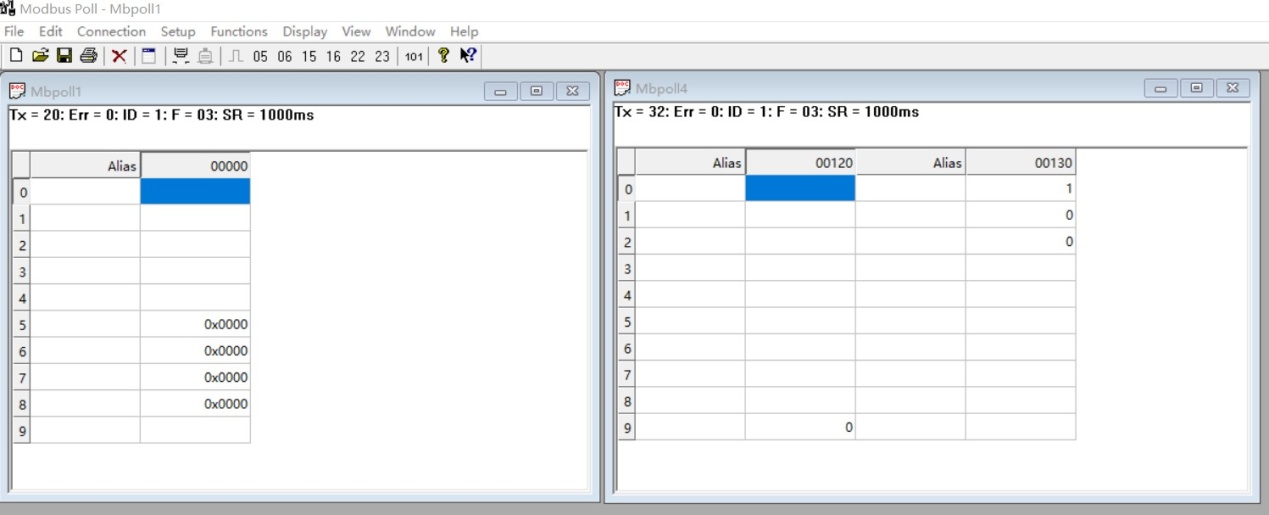
0x0002 Interlocking relationship

0x0003 Momentary relationship

Other values are the same as 0

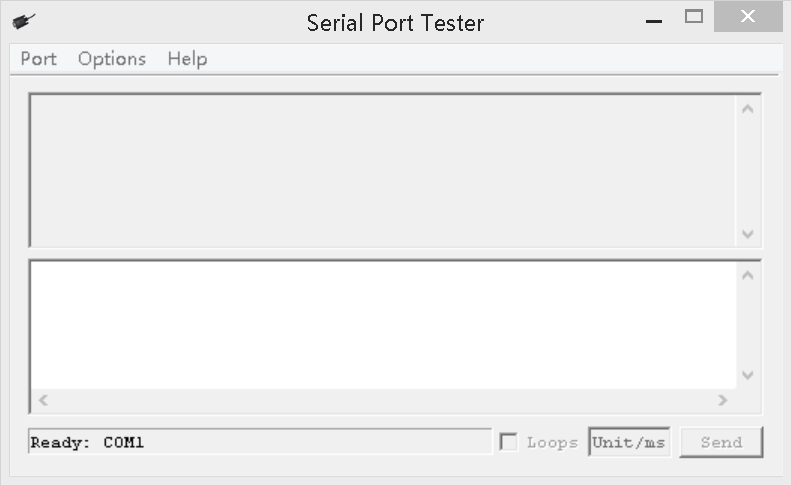
MODBUS commands you can use "Modbus Poll" input, as shown below

（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) ;

}