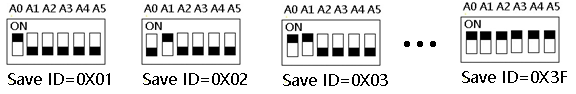
**32 Channels rs485 commamd**

**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 consistent with the DIP switches (A0-A5)



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

**MODBUS** 06 Command (**Control command** ,HEX):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bytes Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| MODBUS Definitions | Slave ID | Function | Address | | Data | | CRC Check | |
| Function | Device Address | Function | Channel number | | Command | Delay time | CRC Check | |
| Open | 0x00-0x2F | 0x06 | 0x0001-  0x0020 | | 0x01 | 0x00 | 2Bytes CRC | |
| Close | 0x00-0x2F | 0x06 | 0x0001-  0x0020 | | 0x02 | 0x00 | 2Bytes CRC | |
| Toggle (Self-locking) | 0x00-0x2F | 0x06 | 0x0001-  0x0020 | | 0x03 | 0x00 | 2Bytes CRC | |
| Latch Inter-locking) | 0x00-0x2F | 0x06 | 0x0001-  0x0020 | | 0x04 | 0x00 | 2Bytes CRC | |
| Momentary (Non-locking) | 0x00-0x2F | 0x06 | 0x0001-  0x0020 | | 0x05 | 0x00 | 2Bytes CRC | |
| Delay | 0x00-0x2F | 0x06 | 0x0001-  0x0020 | | 0x06 | 0x00-0xff | 2Bytes CRC | |
| Open all relays | 0x00-0x2F | 0x06 | 0x0000 | | 0x07 | 0x00 | 16/32 channel version only | |
| Close all relays | 0x00-0x2F | 0x06 | 0x0000 | | 0x08 | 0x00 | 16/32 channel version only | |

Remarks:

1 Momentary mode, delay time is 1 seconds

2 Delay mode, delay time is 1-255 seconds

Return command：

Command is active, return to send commands; instruction is invalid no return.

**MODBUS** 03 Command (**Read status command** ,HEX):

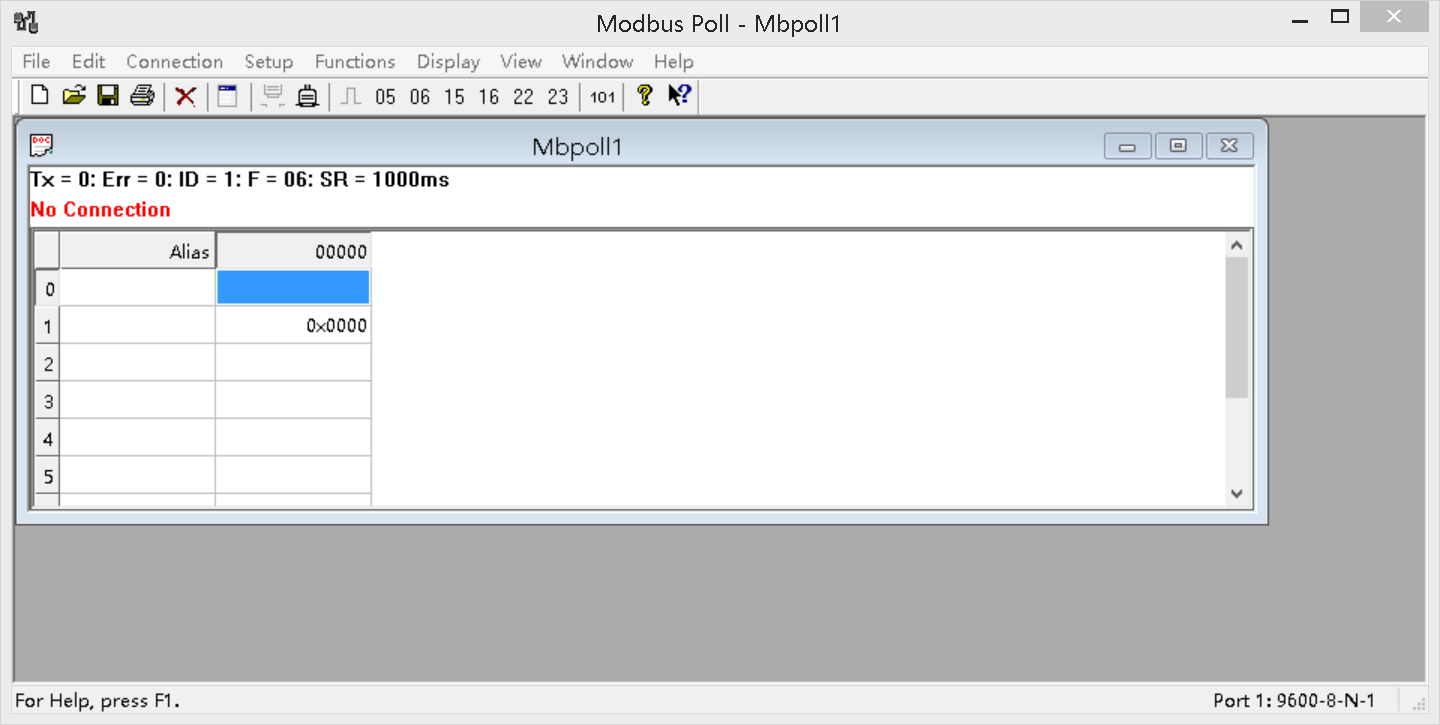
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bytes Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| MODBUS Definitions | Slave ID | Function | Address | | Data | | CRC Check | |
| Function | Device Address | Function | Starting register address | | Register length | | CRC Check | |
| Read Channel 1 State | 0x00-0x2F | 0x03 | 0x0001 | | 0x0001 | |  | |
| Read Channel 2 State | 0x00-0x2F | 0x03 | 0x0002 | | 0x0001 | |  | |
| Read 2 consecutive channels status | 0x00-0x2F | 0x03 | 0x0001-0x0003 | | 0x0002 | |  | |
| Read 3 consecutive channels status | 0x00-0x2F | 0x03 | 0x0001-0x0002 | | 0x0003 | |  | |
| Read all 32 channels status | 0x00-0x2F | 0x03 | 0x0001 | | 0x0020 | |  | |

Read status command returns (function code 03, HEX format):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bytes length | 1 | 1 | 1 |  | 2 |
| MODBUS Definitions | Slave ID | Function | data length | data | CRC16 Check |
| Function | Device Address | Function | data length | Relay state  0x0001 open  0x0000 close | CRC16 Check |
| Channel 1  open | 0x00-0x1F | 0x03 | 0x02 | 0x0001 |  |
| Channel 1  close | 0x00-0x1F | 0x03 | 0x02 | 0x0000 |  |
| Channel 2  open | 0x00-0x1F | 0x03 | 0x02 | 0x0001 |  |
| Channel 2  close | 0x00-0x1F | 0x03 | 0x02 | 0x0000 |  |
| Channel 1 open  Channel 2 open | 0x00-0x1F | 0x03 | 0x04 | 0x0001 0x0001 |  |
| Channel 1 open  Channel 2 close | 0x00-0x1F | 0x03 | 0x04 | 0x0001 0x0000 |  |
| Channel 1 close  Channel 2 open | 0x00-0x1F | 0x03 | 0x04 | 0x0000 0x0001 |  |
| Channel 1 close  Channel 2 close | 0x00-0x1F | 0x03 | 0x04 | 0x0000 0x0000 |  |

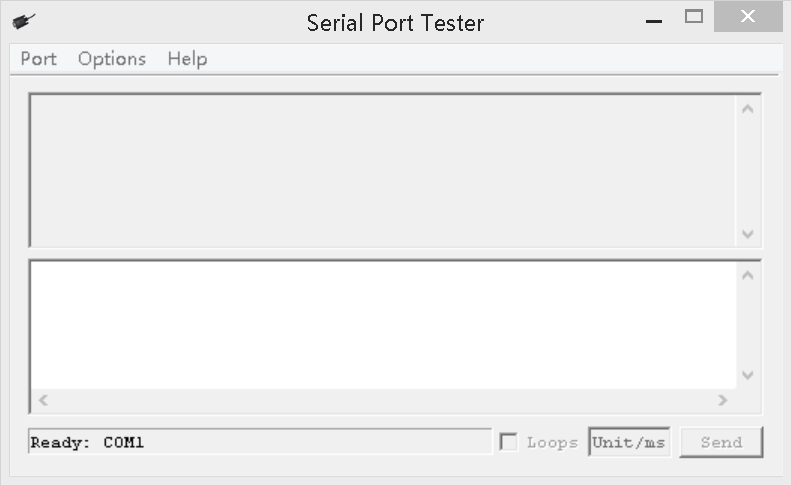
MODBUS commands you can use "Modbus Poll" input, as shown below

（CRC check generated automatically）



You can also use HyperTerminal serial input, as shown below

（Manually add CRC check）





Examples (Slave ID is 1,DIP switch state)

Channel 1 Open ：01 06 00 01 01 00 D9 9A

Channel 1 Close ：01 06 00 01 02 00 D9 6A

Channel 1 Toggle：01 06 00 01 03 00 D8 FA

Channel 1 Latch：01 06 00 01 04 00 DA CA

Channel 1 Momentary: 01 06 00 01 05 00 DB 5A

Channel 1 Delay 10 seconds : 01 06 00 01 06 0A 5B AD

Channel 1 Delay 100 seconds: 01 06 00 01 06 64 DA 41

Channel 2 Open ：01 06 00 02 01 00 29 9A

Channel 2 Close ：01 06 00 02 02 00 29 6A

Channel 2 Toggle ：01 06 00 02 03 00 28 FA

Channel 2 Latch ：01 06 00 02 04 00 2A CA

Channel 2 Momentary : 01 06 00 02 05 00 2B 5A

Channel 2 Delay 10 seconds : 01 06 00 02 06 0A AB AD

Channel 2 Delay 100 seconds : 01 06 00 02 06 64 2A 41

Read state (assuming that the channel 1 is open, the channel 2 is close).

Read channel 1 state ：01 03 00 01 00 01 D5 CA

Return open：01 03 02 00 01 79 84

Read channel 2 state ：01 03 00 02 00 01 25 CA

Return close：01 03 02 00 00 B8 44

Read channel 1 and channel 2 state ：01 03 00 01 00 02 95 CB

Return channel open and channel 2 close ：01 03 04 00 01 00 00 AB F3

**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) ;

}