**R46CA01温度传感器设置协议**

主站发送

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 读取数量(2) | CRC16校验(2) |
|  | 03 读 |  |  |  |
|  | 06 写 |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 只读寄存器 读取功能码03 | | | | |
| 寄存器地址 | 寄存器内容 | 字节数 | 单位 | 备注 |
| 0x0000 | 温度值 | 2 | 0.1℃ | 没接传感器或者传感器坏时：  数据为0X8000(32768) |
| 读写寄存器 读取功能码03 写功能码 06 | | | | |
| 0x0002 | 485地址 | 2 |  | 读地址0XFF  写地址1-247 |
| 0x0003 | 波特率 | 2 |  | 0~4 0:1200  1:2400 2:4800  3:9600（默认）  4:19200 |
| 0x0004 | 温度修正值 | 2 | 0.1℃ | 2字节，输入正数温度增加；输入负数温度减小 |
| 0x0005 | 温度自动上报功能 | 2 | 秒 | 0 ：查询（默认）  1-255: 自动上报，单位秒。  1：每隔1秒上报  2: 每隔2秒上报  10: 每隔10秒上报  最多间隔255秒 |

**串口波特率：9600（可以设置），N，8，1**

**Modbus RTU 通信协议：**

1. **读取当前温度**

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 读取数量(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 长度(1) | 数据(n) | CRC16校验(2) |

功能码 0x03

寄存器地址：0x0000

读取数量：0x0001

返回的温度数据长度为两字节，高位在前低位再后，将这两字节转换成10进制数再除以10即为当前温度值；当最高位为1时表示负值，此时需将此值取补加1，也可将此值直接减去65536，即为当前温度值，下面举例说明：

发送帧（地址为1）：01 03 00 00 00 01 84 0A

返回帧：01 03 02 00 DB F8 1F

01地址码，03功能码，02长度，F8 1F crc16校验

00DB即为温度值，最高位为0，所以温度为正，将其转换为10进制=219，在将其除以10：21.9 即为当前温度值；

返回帧：01 03 02 FF 90 F2 3F

FF 90为温度值，最高位为1，所以温度为负，将其转换为10进制=65424，再减去65536= -11.2 即为当前温度值。

1. **读取485地址码**：

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 读取数量(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 长度(1) | 数据(n) | CRC16校验(2) |

地址码 0xff

功能码 0x03

寄存器地址：0x0002

读取数量：0x0001

例如：

发送帧：FF 03 00 02 00 01 30 14

返回帧：FF 03 02 00 01 50 50

FF地址码，03功能码，02长度，01当前模块地址，50 50 crc16校验

注意：使用此命令时485总线上只能接一个温度模块，超过一个将会出错！

1. **设置485地址：**

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 设置内容(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 寄存器值(2) | CRC16校验(2) |

功能码：0x06

寄存器地址：0x0002

设置内容：2字节(值1-247)

例如，当前的485地址是1，要把485地址改成3：

发送帧（地址为1）01 06 00 02 00 03 68 0B

返回帧：01 06 00 02 00 03 68 0B

1. **读取串口波特率**：

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 读取数量(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 长度(1) | 数据(n) | CRC16校验(2) |

功能码 0x03

寄存器地址：0x0003

读取数量：0x0001

例如：

发送帧（地址为1）：01 03 00 03 00 01 74 0A

返回帧：01 03 02 00 03 F8 45

01地址码，03功能码，02长度，03指当前波特率为9600，F8 45 crc16校验

波特率对应数字：0:1200 1:2400 2:4800 3:9600 4:19200

1. **设置串口波特率：**

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 设置内容(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 寄存器值(2) | CRC16校验(2) |

功能码：0x06

寄存器地址：0x0003

设置内容：2字节(值0-4)

例如，要把波特率改成4800：

发送帧（地址为1）01 06 00 03 00 02 F8 0B

返回帧：01 06 00 03 00 02 F8 0B

波特率对应数字：0:1200 1:2400 2:4800 3:9600 4:19200 5:恢复出厂设置

注意：1 使用此命令时模块重新上电，波特率才会更新！

2 波特率对应数字为5时可以恢复出厂设置

比如 ：01 06 00 03 00 05 B9 C9

1. **读取温度修正值**：

模块有可能与实际温度有误差，此修正值可以修正误差.单位是0.1℃。如果修正值是正数则在当前温度下加上此值，如果是负数则减去此值。

将其设置为0可使此功能失效。

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 读取数量(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 长度(1) | 数据(n) | CRC16校验(2) |

地址码 0x01~0xFE

功能码 0x03

寄存器地址：0x0004

读取数量：0x0001

返回数据：摄氏度，需要将此值除以10。

例如 1：

发送帧：01 03 00 04 00 01 C5 CB

返回帧：01 03 02 00 64 B9 AF 其中0064为修正值，将其用十进制表示即为100，再除以10=10.0℃；

例如 2：

发送帧：01 03 00 04 00 01 C5 CB

返回帧：01 03 02 FF F1 38 30 FF F1其中为修正值，将其用十进制表示即为-15，再除以10=-1.5℃；

1. **设置温度修正值**：

如果模块的温度与实际温度有所偏差可以用此值修正，正数是加，负数是减。

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 设置值(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 设置值(2) | CRC16校验(2) |

地址码 0x01~0xFE

功能码 0x06

寄存器地址：0x0004

设置值：,2字节，最高位表示正负值符号，0表示正，1表示负，单位0.1℃。当最高位为1时表示负值，此时需将此值取补加1，也可将此值直接减去65536，即为当前温度值。禁止修正值将寄存器设置为“0X0000”

例如1：偏移值设置为2.0℃

发送帧：01 06 00 04 00 14 C8 04

返回帧：01 06 00 04 00 14 C8 04 返回帧与发送帧相同。

例如2：偏移值设置为-3.0℃，65536-30=65506 =0XFFE2

发送帧：01 06 00 04 FF E2 09 B2

返回帧：01 06 00 04 FF E2 09 B2 返回帧与发送帧相同。

例如3：禁止修正值将寄存器设置为“0X0000”

发送帧：01 06 00 04 00 00 C8 0B

返回帧：01 06 00 04 00 00 C8 0B 返回帧与发送帧相同。

1. **读取温度自动上报功能**：

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 读取数量(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 长度(1) | 数据(n) | CRC16校验(2) |

地址码 0x01~0xFE

功能码 0x03

寄存器地址：0x0005

读取数量：0x0001

例如：

发送帧：01 03 00 05 00 01 94 0B

返回帧：01 03 02 00 00 B8 44

01地址码，03功能码，02长度，00表示查询功能，B8 44 crc16校验

1. **设置温度自动上报功能：**

发送帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 设置内容(2) | CRC16校验(2) |

返回帧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址码(1) | 功能码(1) | 寄存器地址(2) | 寄存器值(2) | CRC16校验(2) |

功能码：0x06

寄存器地址：0x0005

设置内容：2字节(值1-247)

例如，当前是查询功能，要改成自动上报：

1秒自动上报，发送帧（地址为1）01 06 00 05 00 01 58 0B

2秒自动上报，发送帧（地址为1）01 06 00 05 00 02 18 0A

3秒自动上报，发送帧（地址为1）01 06 00 05 00 03 D9 CA

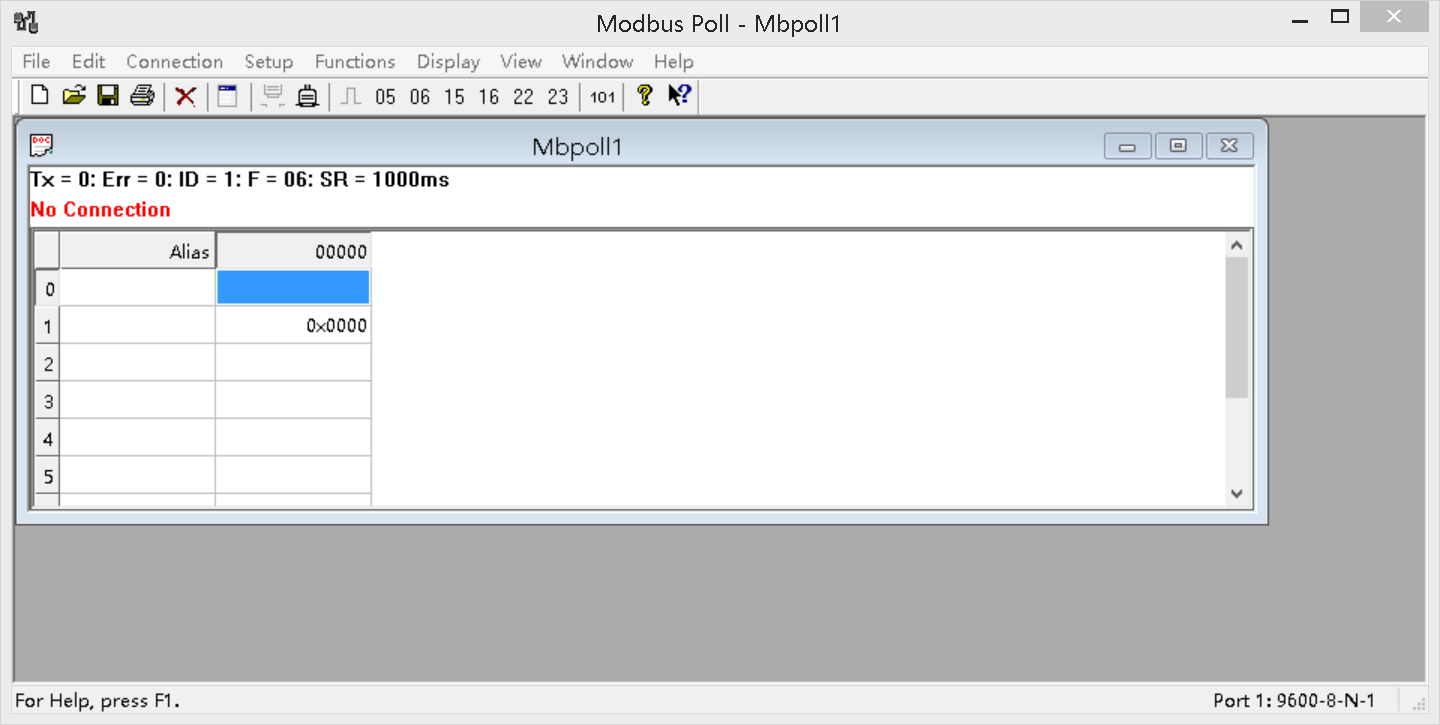
4秒自动上报，发送帧（地址为1）01 06 00 05 00 04 98 08

5秒自动上报，发送帧（地址为1）01 06 00 05 00 05 59 C8

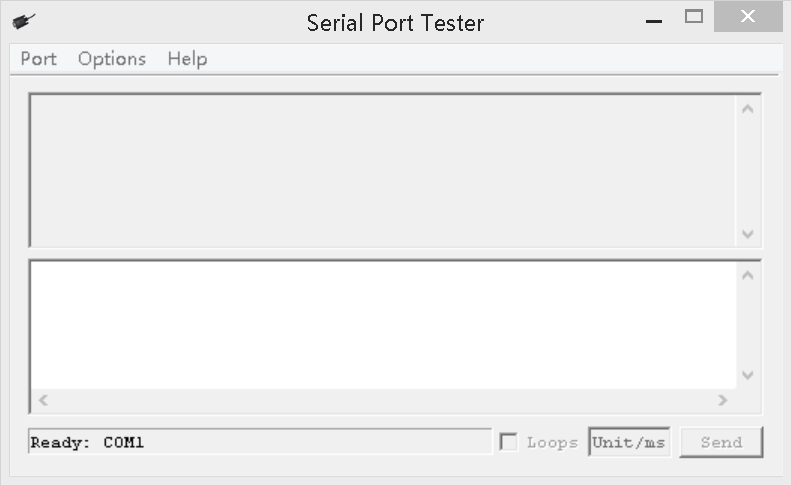
10秒自动上报，发送帧（地址为1）01 06 00 05 00 0A 19 CC

禁止查询功能：发送帧(地址为1) 01 06 00 05 00 00 99 CB

MODBUS指令可以用"Modbus Poll"输入，如下图



也可以使用串口超级终端输入，如下图



**CRC16校验程序：**

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) ;

}