HY-XSG 8通道导轨式称重变送模块

V2.1

使

用

说

明

书

蚌埠恒远传感器科技有限公司

[www.hychuangan.cn](http://www.hychuangan.cn)

0552-2806868

当前版本：V2.1

修改日期：2022-1-9

**目录**

[第一章 概述 2](#_Toc92713273)

[1.1产品简介 2](#_Toc92713274)

[1.2安全提示 2](#_Toc92713275)

[1.3技术参数以及外形尺寸 3](#_Toc92713276)

[1.4接口定义 3](#_Toc92713277)

[第二章 操作方法 4](#_Toc92713278)

[2.1按键以及显示区域定义 4](#_Toc92713279)

[2.2参数显示与设置 4](#_Toc92713280)

[2.2.1 01-SEt 系统参数 5](#_Toc92713281)

[2.2.2 02-Un 备用 6](#_Toc92713282)

[2.2.3 03-CAL 系统操作 6](#_Toc92713283)

[2.2.4 04-INF 系统信息 7](#_Toc92713284)

[第三章 辅助说明 8](#_Toc92713285)

[3.1 modbus通讯协议 8](#_Toc92713286)

[3.2 其他通讯 8](#_Toc92713287)

[3.2.1 主动发送之协议 8](#_Toc92713288)

[3.3 其他功能 9](#_Toc92713289)

[3.4 MODBUS RTU通信实例 9](#_Toc92713290)

# 第一章 概述

## 1.1产品简介

感谢您选择本公司的产品。在使用本产品之前，请仔细阅读本手册以使本产品能最大程度发挥作用。

本产品采用24位∑-△ADC，将桥式称重传感器的模拟信号转换为数字信号，共有8路。

装置采用宽工作电压供电方式，适用于10-30VDC电源系统。

**产品特点：**

1. 具有防射频RFI/电磁EMI干扰，具有很强的EMC特性；
2. 10-30V宽电压供电；
3. 高速24位∑-△ADC采样，高达500Hz以上；
4. 通讯接口完备，标配232和485。可选CAN,以太网等。

## 1.2安全提示



1. 本仪表具有抗干扰设计。请务必将仪表进行可靠接地，且与交流电源接地线分开
2. 不要在可燃性气体环境中使用
3. 避免阳光直射
4. 通讯站点建议使用与模块同一24Ｖ电源供电，否则通讯连接需通过隔离模块对外传输[例如PLC是AC220V，PLC与本模块需要增加通讯隔离模块]。

## 1.3技术参数以及外形尺寸

|  |  |
| --- | --- |
| **测量信号** | 8路-20mV~20mV，单路最大并联驱动4个350欧姆称重传感器 |
| **采样频率** | 500Hz |
| **检测精度** | III级 |
| **分辨率** | 1/500000 |
| **通讯接口** | 标配1路232,1路485。可选配CAN，以太网。 |
| **非线性度** | 0.005%FS |
| **工作电源** | 模块供电10-30V DC。传感器供电5V。 |
| **重量** | 约0.2kg |
| **外形尺寸** | 长\*宽\*高，单位mm |
| **功耗** | < 5W |
| **工作温度** | -20~+65℃ |

## 1.4接口定义

## QQ截图20220427151319

说明：

主画面显示重量值。8通道循环切换显示。CH01之后的值是1通道重量，CH02之后的值是2通道重量。

1：V+、V-为模块供电，建议24V直流；

2：E+、E-、为传感器激励接线，S+、S-、为1-8路传感器信号接线，；

3：B、A为485接口；GND、TX、RX为232接口；CANH、CANL为CAN接口；RJ45为以太网接口。

4:传感器屏蔽线可靠接地。

## 1.4按键以及显示区域定义

共4个按键：1575808054(1) 1575808085(1) 1575808103(1) 1575808125(1)

1575808054(1):进入菜单/返回上一级。主画面长按，输入密码000123解锁。

1575808085(1): 菜单画面为改变菜单选项；修改参数画面，移动光标。

1575808103(1)：菜单画面为改变菜单选项；修改参数画面，增加数值。

1575808125(1)：确认本次操作。

## 1.5参数显示与设置

输入参数之前，在主画面按1575808054(1)键，需要输入密码000123，按1575808125(1)键确定；

在主画面按1575808054(1)键进入参数设置画面，此时第一行显示01-SEt(系统参数)，按1575808085(1)1575808103(1)键，可按切换显示02—Un(备用)、03-CAL(系统操作)、04-INF(系统信息)。选定设置功能后，按下1575808125(1)键即可进入相应的参数表。此时，按1575808085(1)1575808103(1)可切换显示其他的参数。按1575808125(1)键，进入参数修改状态或者下一级显示。按住1575808054(1)键3秒以上，可直接退出到重量显示界面。

### 2.2.1 01-SEt 系统参数

在主画面按1575808054(1)键，当显示01-SEt时，按1575808125(1)键，进入系统参数显示，参数如下表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 显示符 | 定义 | 缺省值(范围) | 描述 | PLC寄存器地址  备注：MODBUS地址减一 |
| 01-000 | 小数点 | 2(0-4) |  | 1001 |
| 01-001 | 零点1 | 0(0-999999) | 保存的零点采样值。 | 1003 |
| 01-002 | 零点2 | 1005 |
| 01-003 | 零点3 | 1007 |
| 01-004 | 零点4 | 1009 |
| 01-005 | 零点5 | 1011 |
| 01-006 | 零点6 | 1013 |
| 01-007 | 零点7 | 1015 |
| 01-008 | 零点8 | 1017 |
| 01-009 | 线性系数1 | 1000(1-999999) | 校满时形成的系数。 | 1019 |
| 01-010 | 线性系数2 | 1021 |
| 01-011 | 线性系数3 | 1023 |
| 01-012 | 线性系数4 | 1025 |
| 01-013 | 线性系数5 | 1027 |
| 01-014 | 线性系数6 | 1029 |
| 01-015 | 线性系数7 | 1031 |
| 01-016 | 线性系数8 | 1033 |
| 01-017 | 滤波等级 | 16(0-19) | 数值越大，滤波效果越好，但重量显示更滞后。 | 1035 |
| 01-018 | 分度值 | 0(0-5) | 0:1 1:2 2:5 3:10 4:20 5:50。 | 1037 |
| 01-019 | 稳定范围 | 0.01(0.00-99.99) | 这个值大于0时，开始判断稳定。 | 1039 |
| 01-020 | 稳定时间 | 0.30(0.00-9.99) | 此时间内，重量变化量在稳定范围内，则稳定。 | 1041 |
| 01-021 | 蠕变范围 | 0.00(0.00-99.99) | 这个值大于0时，进行蠕变修正。 | 1043 |
| 01-022 | 蠕变时间 | 10.00(0.00-99.99) | 此时间内，重量变化量在蠕变范围内，且一直稳定，则进行蠕变修正。 | 1045 |
| 01-023 | 置零范围 | 0.00(0.00-99.99) | 这个值大于0时，进行自动置零操作。 | 1047 |
| 01-024 | 置零时间 | 1.00(0.00-9.99) | 此时间内，重量在该范围内，且一直稳定，则进行自动置零。持续稳定只置零一次。 | 1049 |
| 01-025 | 通讯地址 | 1(0-128) |  | 1051 |
| 01-026 | 232口波特率 | 1(0-4) | 0:9600 1:19200 2:38400 3:57600 4:115200 | 1053 |
| 01-027 | 232口校验 | 0(0-2) | 0:无校验 1:偶校验 2:奇校验 | 1055 |
| 01-028 | 232口功能 | 0(0-9) | 0:RTU 1:主动发送 其余：备用 | 1057 |
| 01-029 | 232口32位顺序 | 0(0-3) | 0:1234 1:2143 2:3412 34321 | 1059 |
| 01-030 | 485口波特率 | 1(0-4) | 0:9600 1:19200 2:38400 3:57600 4:115200 | 1061 |
| 01-031 | 485口校验 | 0(0-2) | 0:无校验 1:偶校验 2:奇校验 | 1063 |
| 01-032 | 485口功能 | 0(0-9) | 0:RTU 1:主动发送 2：TCP(有以太网模块时有效)  其余：备用 | 1065 |
| 01-033 | 485口32位顺序 | 0(0-3) | 0:1234 1:2143 2:3412 34321 | 1067 |
| 01-041 | 采集速度 | 1(0-2) | 0:10Hz 1:40Hz 2:640Hz | 1083 |
| 其余 |  |  | 备用 |  |

### 2.2.2 02-Un 备用

### 2.2.3 03-CAL 系统操作

在主画面按1575808054(1)键，当显示01-SEt时，按1575808085(1)1575808103(1)可切换显示为03-CAL,按1575808125(1)键进入模块的功能操作，例如校零、校满等。包含的操作如下表：

|  |  |  |
| --- | --- | --- |
| 显示符 | 功能 | 描述 |
| 03-000 | 校零1 | 1-8通道校零 |
| 03-001 | 校零2 |
| 03-002 | 校零3 |
| 03-003 | 校零4 |
| 03-004 | 校零5 |
| 03-005 | 校零6 |
| 03-006 | 校零7 |
| 03-007 | 校零8 |
| 03-008 | 校满1 | 1-8通道校满 |
| 03-009 | 校满2 |
| 03-010 | 校满3 |
| 03-011 | 校满4 |
| 03-012 | 校满5 |
| 03-013 | 校满6 |
| 03-014 | 校满7 |
| 03-015 | 校满8 |

**校零：**当显示03-000时，按1575808125(1)键，显示采样值，此时再按1575808125(1)键，显示3秒倒计时，计时结束，自动保存零点，并且返回03-000,1号通道校零结束。

**校满：**当显示03-008时，先在称台上放重物（砝码），再按1575808125(1)键，输入重物的重量，按1575808125(1)键确认，此时将显示重物重量。如果信号有错，将提示ERR错误。此时，再按1575808125(1)键，显示3秒倒计时，计时结束，自动保存满度系数值，并且返回03-001。1号通道校满结束。

### 2.2.4 04-INF 系统信息

在主画面按1575808054(1)键，当显示01-SEt时，按1575808085(1)1575808103(1)可切换显示为04-INF,按1575808125(1)键进入模块的功能操作，例如校零、校满等。包含的操作如下表：

|  |  |  |
| --- | --- | --- |
| 显示符 | 功能 | 描述 |
| 04-000 | 版本等查询 | 查询版本、仪表错误等信息 |
| 04-001 | 密码管理等 | 设置密码，恢复默认等 |
| 04-002 | 出厂测试 | 出厂测试以及相关出厂操作 |

**版本等查询：**仅供厂家使用

**密码管理等：**当显示04-001时，按1575808125(1)键，之后可通过按1575808085(1)1575808103(1)循环显示“01-PASS”,“02-dEF”,“03-FAC”。

显示“01-PASS”时，按1575808125(1)键，可修改密码。首先输入原密码，再输入新密码。

显示“02-dEF”时，按1575808125(1)键，然后选择YES，再按1575808125(1)键，可恢复默认。

显示“03-FAC”时，仅供厂家使用。

**出厂测试：**当显示04-002时，按1575808125(1)键，之后可通过按1575808085(1)1575808103(1)循环显示“CH01”、“CH02”,…“CH08”。按1575808125(1)键可查看对应采样值。

# 辅助说明

## 3.1 modbus通讯协议[232、485和以太网共用]

默认19200波特率，8个数据位，无校验，1个停止位[19200,8,N,1]通讯设置，所有数据皆为32位整形数据，占用2个寄存器，共4个字节。

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 数据类长度 | 描述 | PLC寄存器地址  备注：MODBUS地址减一 |
| 重量1 | 32位整形 | 读写寄存器：  写入0:校零；写入其他数值，表示输入称台重物重量，校满。假如重量2个小数点，砝码10.00，则写入1000。 | 1 |
| 重量2 | 3 |
| 重量3 | 5 |
| 重量4 | 7 |
| 重量5 | 9 |
| 重量6 | 11 |
| 重量7 | 13 |
| 重量8 | 15 |
| 其他状态1 | 32位整形 | 采样错误。第2位，信号溢出，可能传感器坏或者信号线断；第3位，采样模块错误。 | 17 |
| 其他状态2 | 19 |
| 其他状态3 | 21 |
| 其他状态4 | 23 |
| 其他状态5 | 29 |
| 其他状态6 | 31 |
| 其他状态7 | 33 |
| 其他状态8 | 35 |
| 采样值1 | 32位整形 | 只读 | 37 |
| 采样值2 | 39 |
| 采样值3 | 41 |
| 采样值4 | 43 |
| 采样值5 | 45 |
| 采样值6 | 47 |
| 采样值7 | 49 |
| 采样值8 | 51 |
| 批量置零指令 | 32位整形 | 写入8，1-8通道批量置零。 | 31 |

## 3.2 其他通讯

### 3.2.1 主动发送之协议

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 起始符 | 符号[+/-] | 数据[6位] | 小数点[0-3] | 异或校验 | 结束 |
| 0x02 | 0x2B/0X2D | 6个字节 | 0x30-0x33 | 2个字节 | 0xFF |

1:数据采用ASCII码进行传递。例如显示为1234，则传递16进制30 30 31 32 33 34

2:异或校验位之前的除去起始符的所有数据进行异或运算，会得到一个字节的数据，然后把这个字节转换为两个ASCII码，例如，计算得到的校验为0x4A，其对应的16进制ASCII为34 41。

3：4通道数据，符号和数据共4帧，每帧数据包含了7个字节，分别是符号+数据。

## 3.3 其他功能

如果需要以太网网功能，请提前联系厂家，关于以太网的配置和测试工具，可向厂家获取。

**MODBUS RTU通信实例**

本公司地址采用西门子系统地址描述规则，实际发送指令，指令为16进制，地址需要减1。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 设备地址站号 | 功能号 | 数据地址 | 读取数据个数 | CRC校验 |  |
| 01 | 03 | 00 00 | 00 02 | C4 0B | 读取1通道重量 |
| 01 | 03 | 00 02 | 00 02 | 65 CB | 读取2通道重量 |
| 01 | 03 | 00 04 | 00 02 | 85 CA | 读取3通道重量 |
| 01 | 03 | 00 06 | 00 02 | 24 0A | 读取4通道重量 |
| 01 | 03 | 00 08 | 00 02 | 45 C9 | 读取5通道重量 |
| 01 | 03 | 00 0A | 00 02 | E4 09 | 读取6通道重量 |
| 01 | 03 | 00 0C | 00 02 | 04 08 | 读取7通道重量 |
| 01 | 03 | 00 0E | 00 02 | A5 C8 | 读取8通道重量 |
| 01 | 03 | 00 00 | 00 10 | 44 06 | 读取1-8通道全部重量 |

**主机对从机读重量操作**  
主机进行读1-8通道对应1.3.5.7.9.11.13.15号重量寄存器32位重量操作，发送指令读取时，寄存器地址-1，默认起始寄存器地址从0开始，则报文是：

单片机接收到这串数据根据数据计算CRC校验判断数据是否正确，如果判断数据无误，返回信息给主机，返回的信息也是有格式的。

例：返回内容：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 设备地址站号 | 功能号 | 数据字节个数 | 四个字节数据 | CRC校验 |  |
| 01 | 03 | 04 | 00 01 E2 40 | E2 A3 | 单通道返回重量数据 |
| 01 | 03 | 20 | 每4个字节为一通道重量 | CRC | 返回1-8通道全部重量数据 |

返回的四个16进制字节数据00 01 E2 40就是重量，转换为10进制就是123456。

**主机对从机写数据操作**

**清零和校准说明：**

主机进行写1.3.5.7.9.11.13.15号寄存器32位的数据 操作

例：清零操作，则16进制报文是： 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 设备地址站号 | 功能号 | 数据地址 | 寄存器数量 | 字节数 | 四个字节数据 | CRC校验 |  |
| 01 | 10 | 00 00 | 00 02 | 04 | 00 00 00 00 | F3 AF | 1通道清零 |
| 01 | 10 | 00 02 | 00 02 | 04 | 00 00 00 00 | 72 76 | 2通道清零 |
| 01 | 10 | 00 04 | 00 02 | 04 | 00 00 00 00 | F2 5C | 3通道清零 |
| 01 | 10 | 00 06 | 00 02 | 04 | 00 00 00 00 | 73 85 | 4通道清零 |
| 01 | 10 | 00 08 | 00 02 | 04 | 00 00 00 00 | F2 09 | 5通道清零 |
| 01 | 10 | 00 0A | 00 02 | 04 | 00 00 00 00 | 73 D0 | 6通道清零 |
| 01 | 10 | 00 0C | 00 02 | 04 | 00 00 00 00 | F3 FA | 7通道清零 |
| 01 | 10 | 00 0E | 00 02 | 04 | 00 00 00 00 | 72 23 | 8通道清零 |
| 01 | 10 | 00 1E | 00 02 | 04 | 00 00 00 08 | 72 E9 | 1-8通道批量清零 |

例：校准砝码重量100，如需加一位小数点则写入1000，如需加2位则写入10000，报文以16进制100.00为例则是： 

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 设备地址站号 | 功能号 | 数据地址 | 寄存器数量 | 字节数 | 四个字节数据 | CRC校验 |
| 01 | 10 | 00 00 | 00 02 | 04 | 00 00 27 10 | E9 93 |

返回内容：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 设备地址站号 | 功能号 | 数据地址 | 寄存器数量 | CRC校验 |
| 01 | 10 | 00 00 | 00 02 | 41 C8 |

**Modbus RTU CRC校验码计算方法**

下面是我编写的C#版CRC16校验(逆序):

// 逆序CRC表

private byte[] aucCRCHi = new byte[] {

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

};

private byte[] aucCRCLo = new byte[] {

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

};

//通过逆序CRC16表, 逆序CRC计算

int CRC16(byte[] Frame, int Length) // 传入要校验的数组名及其长度

{

byte CRCHi = 0xFF;

byte CRCLo = 0xFF;

int iIndex = 0;

for (int i = 0; i < Length; i++ )

{

iIndex = CRCLo ^ (Frame[i++]);

CRCLo = (byte)(CRCHi ^ aucCRCHi[iIndex]);

CRCHi = aucCRCLo[iIndex];

}

return ( int )( CRCHi << 8 | CRCLo );// CRC校验返回值 // CRCHI 向左移动，就是逆序计算的代表