单通道数字式称重变送器

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：V1.1

修改日期：2020-5-10

**目录**

[第一章 概述 1](#_Toc47085131)

[1.1产品简介 1](#_Toc47085132)

[1.2安全提示 2](#_Toc47085133)

[1.3技术参数以及外形尺寸 2](#_Toc47085134)

[1.4接口定义 2](#_Toc47085135)

[第二章 辅助说明 3](#_Toc47085136)

[2.1 modbus通讯协议 3](#_Toc47085137)

#

# 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，且装置采用宽工作电压供电方式，适用于10-30VDC电源系统。

本产品还具备传感器线路检测功能，即当未接传感器或者传感器故障(包括接线脱落等)时，进行对应的报警提示。

**产品特点：**

1. 完善的硬件抗干扰设计， EMC性能优异。
2. 宽电压供电，供电电压范围10-30VDC。
3. 高速24位∑-△ADC采样，高达500Hz以上。
4. 丰富的传感器故障检测功能。如信号超限，模块采样故障，传感器线路连接故障等。
5. 通讯接口多样化设计，同时具有RS232和RS485通讯接口。

## 1.2安全提示

1. 本产品具有抗干扰设计。请务必将产品外壳可靠接地。
2. 不要在可燃性气体环境中使用。
3. 避免阳光直射。

## 1.3技术参数以及外形尺寸

|  |  |
| --- | --- |
| **测量信号** | -20mV~20mV，可最大并联驱动6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RS232,1路RS485 |
| **非线性度** | 0.005%FS |
| **工作电源** | 10-30VDC（传感器激励电源5VDC） |
| **重量** | 约0.2kg |
| **外形尺寸** | 100\*64\*24，长\*宽\*高，单位mm |
| **功耗** | < 3W |
| **工作温度** | -20~+65℃ |

## 1.4接口定义

说明

1：DC+、DC-为变送器供电端子，建议使用12V或24V直流电源供电；

2：E+、E-、S+、S-为传感器接线端子，屏蔽线可与变送器外壳直接连接接地；

3：A、B为RS485接口端子； RX、TX、GND为RS232接口端子；

# 第二章 辅助说明

## 2.1 modbus通讯协议

默认9600波特率，8个数据位，无校验，1个停止位[9600,8,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 | 默认[范围] | 描述 | 寄存器地址 |
| 毛重 |  | 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 |
| 采样值 |  |  | 7 |
| 开入开出状态 |  | 0-2位是开入状态，3-9位为开出状态。 | 9 |
| 其他状态 |  | 采样错误。第0,1位，激励线可能断；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。 | 11 |
| 小数点 | 2[0-3] |  | 1001 |
| 零点 | 0[-500000~500000] |  | 1005 |
| 满度系数 | 10 |  | 1007 |
| 滤波 | 14[0-19] | 滤波值越大，数据越稳，响应速度越慢 | 1013 |
| 分度值 | 0[0~5] | 0:1 1:2 2:5 3:10 4: 20 5:50 | 1017 |
| 稳定范围 | 0.01(0.00-99.99) | 这个值大于0时，开始判断稳定。 | 1019 |
| 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。 | 1021 |
| 蠕变范围 | 0.00(0.00-99.99) | 这个值大于0时，进行蠕变修正。 | 1023 |
| 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。 | 1025 |
| 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。 | 1027 |
| 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。 | 1029 |
| 通讯地址 | 1(0-128) |  | 1031 |
| 1口波特率 | 0(0-4) | 0:9600 1:19200 2:38400 3:57600 4:115200 | 1033 |
| 1口校验 | 0(0-2) | 0:无校验 1:偶校验 2:奇校验  | 1035 |
| 1口功能 | 0(0-9) | 0:RTU 其余：备用  | 1037 |
| 1口32位顺序 | 0(0-3) | 0:1234 1:2143 2:3412 34321 | 1039 |
| 2口波特率 | 1(0-4) | 0:9600 1:19200 2:38400 3:57600 3:115200 | 1041 |
| 2口校验 | 0(0-2) | 0:无校验 1:偶校验 2:奇校验  | 1043 |
| 2口功能 | 0(0-9) | 0:RTU 其余：备用  | 1045 |
| 2口32位顺序 | 0(0-3) | 0:1234 1:2143 2:3412 34321 | 1047 |
| 主动发送间隔 | 200(1-1000) | 单位为ms | 1049 |

**MODBUS RTU通信协议**

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。

**主机对从机读数据操作**
主机进行读1号寄存器32位的数据 操作，则报文是：
    01             03            00 00           00 02          C4 0B
 从机地址        功能号          数据地址      读取数据个数       CRC校验
那么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，则结果是：返回信息给主机，返回的信息也是有格式的：
返回内容：
    01         03            04             00 01 E2 40          E2 A3
  从机地址   功能号     数据字节个数    四个字节数据    CRC校验

四个16进制数据转换为10进制，就是123456

**主机对从机写数据操作**

主机进行写1号寄存器32位的数据 操作

写入砝码重量123456，则报文是：
    01          10       00 00         00 02       04   00 01 E2 40  EB 3F
 从机地址    功能号   数据地址     寄存器数量  字节数    四个字节数据   CRC校验

校零操作，则报文是：
    01          10       00 00         00 02       04   00 00 00 00  F3 AF
 从机地址    功能号   数据地址     寄存器数量  字节数    四个字节数据   CRC校验

返回内容：

01          10       00 00         00 02         41 C8
 从机地址    功能号   数据地址     寄存器数量     CRC校验

**Modbus RTU CRC校验码计算方法**

在CRC计算时只用8个数据位，起始位及停止位，如有奇偶校验位也包括奇偶校验位，都不参与CRC计算。

CRC计算方法是：

1、  加载一值为0XFFFF的16位寄存器，此寄存器为CRC寄存器。

2、  把第一个8位二进制数据（即通讯信息帧的第一个字节）与16位的CRC寄存器的相异或，异或的结果仍存放于该CRC寄存器中。

3、  把CRC寄存器的内容右移一位，用0填补最高位，并检测移出位是0还是1。

4、  如果移出位为零，则重复第三步（再次右移一位）；如果移出位为1，CRC寄存器与0XA001进行异或。

5、  重复步骤3和4，直到右移8次，这样整个8位数据全部进行了处理。

6、  重复步骤2和5，进行通讯信息帧下一个字节的处理。

7、  将该通讯信息帧所有字节按上述步骤计算完成后，得到的16位CRC寄存器的高、低字节进行交换

8、  最后得到的CRC寄存器内容即为：CRC校验码

代码：

///<summary>

/// 转换成CRC码

//modbus CRC16

publicvoid CRC16Calc(byte[] dataBuff, int dataLen)

{

int CRCResult = 0xFFFF;

if (dataLen < 2)

{

   return;

}

for (int i = 0; i < (dataLen - 2); i++)

{

    CRCResult = CRCResult ^ dataBuff[i];

for (int j = 0; j < 8; j++)

{

if ((CRCResult & 1) == 1)

CRCResult = (CRCResult >> 1) ^ 0xA001;

else CRCResult >>= 1;

}

}

dataBuff[dataLen - 1] =Convert.ToByte(CRCResult >> 8);

dataBuff[dataLen - 2] =Convert.ToByte(CRCResult & 0xff);

}