采集卡

twнонл5/

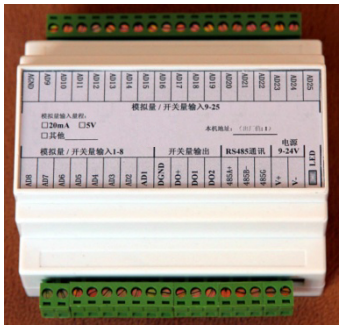
使用手册

V1.0

目录

一、硬件部分

1.1、简介



PR2320DC数据采集接口卡具有 23 路模拟量输入，可直接用作 23 路开关量输入，2 路独立雨量脉冲信号输入接口，集成电压校准电路和温度传感器，通讯采用 RS485 接口，使用 MODBUS 协议，电源电压 9~24V。该采集卡稳定性好、性价比高、功能强大，可广泛应用于工业过程控制系统以及实验室数据采集系统，可与 PLC、文本显示器、组态软件等进行连接。

1.2、特征

1) 23 路模拟量输入，使用 12 bit 分辨率 ADC，内部转换速度最大 1Msps，默认输入量程 0-20mA、0-5V，适用于大多数工业传感器和变送器。抗干扰能力强，具有过压过流保护，具有 RC 滤波。（量大可免费改为 0-10V，0-15V，0-20V，0-30V 及其他混合输入的量程）

2) 集成温度传感器和电压校准电路，随时进行校准采集数据，实时监测工作环境温度。

3) 23 路模拟量输入可直接作为开关量输入，无需设置。

4) 2 路独立雨量脉冲信号输入接口。

5) RS485 通讯接口，采用原装进口 TI 公司的 485 芯片，芯片内集成 TVS 管，可防止 400W 雷击浪涌电流，集成±15KV 的 HBM 静电释放电路，±8KV 的接触放电电路，±15KV 的气隙放电电路，性能远超普通的 MAX485 芯片。除此之外，板子上还加上了抗干扰滤波电路，大大提高 485 总线的稳定性，适合各种复杂环境应用。

6) 采用 MODBUS-RTU 协议，适用范围广，易于与其他设备如 PLC、文本显示器，组态软件等联网，具有自动处理错误命令功能，完全解决误操作问题。

7) 通讯可以软件更改从机地址和波特率，并具有硬件恢复出厂设置功能。

8) 采用工业级高速微处理器，运行速度快且稳定，具有内部看门狗，防止死机，防止程序跑飞。

9) 9-24V 宽范围电源电压，具有反接保护。

10) 为了加密，每个卡具有一个唯一不可更改的 16 个字节的序列号作为采集卡的唯一标识。另有 16 个字节的存储空间可以读写，掉电不丢失，用户可存储简单的信息。

11) 为提高稳定性，分别从硬件和软件方面做了很多保护及抗干扰的措施。软件方面，优化程序，做了周密复杂的测试，在以任何波特率，连续发送随机码或错误命令十分钟，未出现任何无操作，且停止发送后，立即发送正确命令，工作完全正常！

1.3、尺寸规格

PR2320DC用插拔式端子，使用常用的导轨式外壳，可方便卡在35mm导轨上，安装方便快捷。下面介绍一下外壳尺寸和电路板尺寸。

1.3.1、 外壳尺寸图

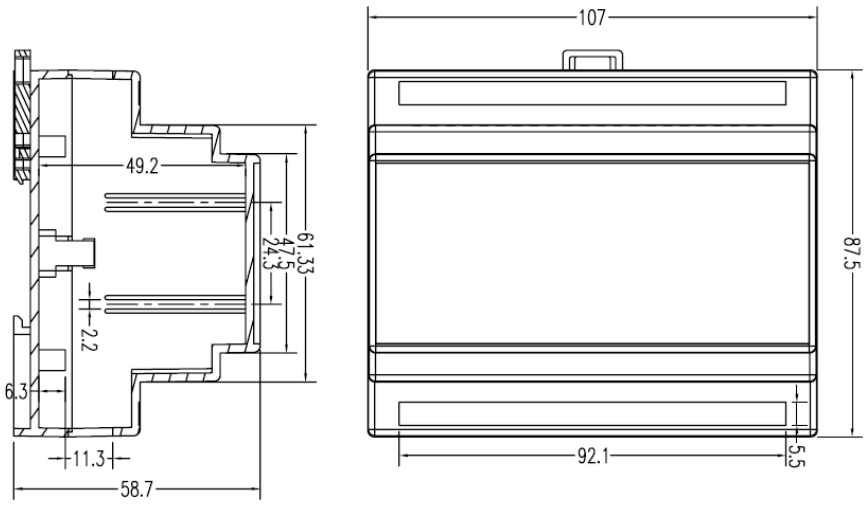


图 3-1. 外壳尺寸图 107×87.5×58.7mm（插上端子后，宽度从 87.5mm 变为 108mm）

1.3.2、 电路板尺寸图

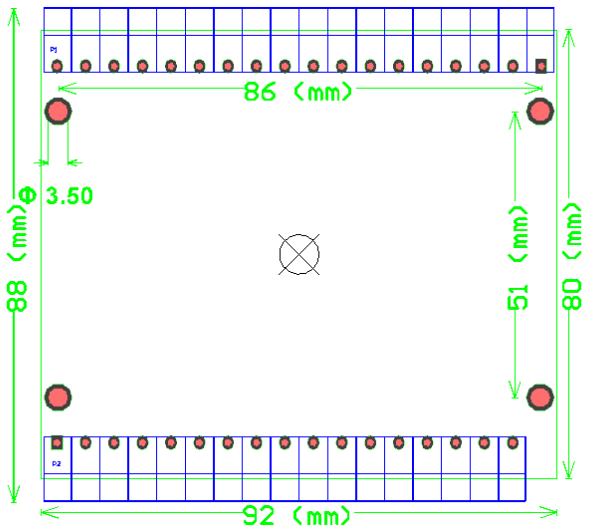


图 3-2. 电路板尺寸：92×80mm（PCB 板厚 1.6mm，上层元件最高 13mm）.

1.4、PR2320DC选型

PR2320DC（PR2320DC-Xnn）系列有以下几种:

X 为模拟量输入类型：C 表示电流，V 表示电压；nn 为量程，如 20 表示最大量程为 20mA 或 20V；

例如：选择模拟量输入 0-5V，型号为 PR2320DC-V05

型号	模拟量输入量程
PR2320DC-C20	0-20mA
PR2320DC-V05	0-5V
PR2320DC-V30	0-30V

1.5、引脚描述

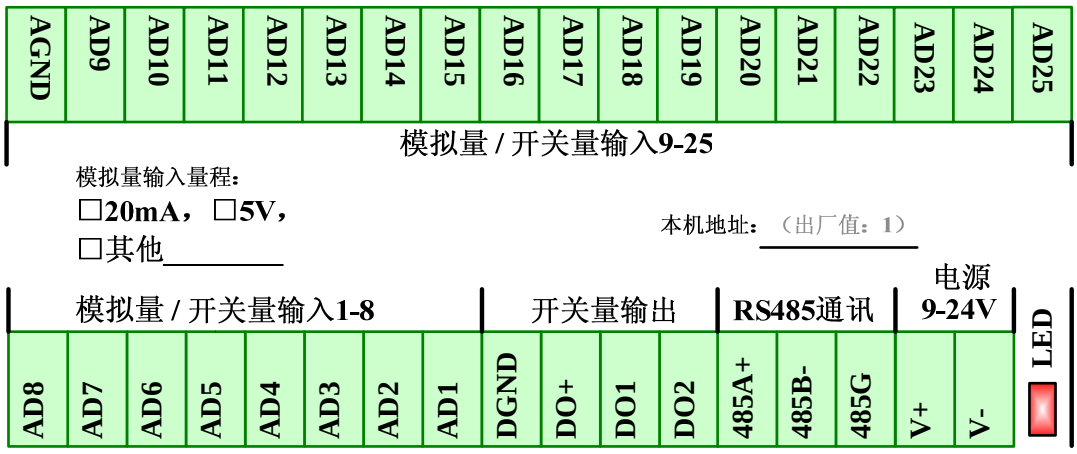


图 5-1. 采集卡PR2320DC　　引脚排列图

表 1.5.1：引脚功能描述表

引脚号	名称	功能描述
1	LED	指示灯，正常工作时亮，通讯时闪
2	V-	系统电源输入负极
3	V+	系统电源输入正极（推荐：9-24V，极限：8-30V）
4	485G	RS485 数据线参考地（内部与系统电源负极连）
5	485B-	RS485 数据线 B
6	485A+	RS485 数据线 A
7	DO2	开关量输出通道 2（工作电压 5-36V，极限 3.3-48V）
8	DO1	开关量输出通道 1（工作电压 5-36V，极限 3.3-48V）

9	DO+	开关量输出的电源正极
10	DGND	开关量输入输出的电源负极（内部与系统电源负极连）
11	AD1	模拟量或开关量输入通道 1
12	AD2	模拟量或开关量输入通道 2
13	AD3	模拟量或开关量输入通道 3
14	AD4	模拟量或开关量输入通道 4
15	AD5	模拟量或开关量输入通道 5
16	AD6	模拟量或开关量输入通道 6
17	AD7	模拟量或开关量输入通道 7
18	AD8	模拟量或开关量输入通道 8
19	AGND	模拟量或开关量输入参考地（内部与系统电源负极连）
20	AD9	模拟量或开关量输入通道 9
21	AD10	模拟量或开关量输入通道 10
22	AD11	模拟量或开关量输入通道 11
23	AD12	模拟量或开关量输入通道 12
24	AD13	模拟量或开关量输入通道 13
25	AD14	模拟量或开关量输入通道 14
26	AD15	模拟量或开关量输入通道 15
27	AD16	模拟量或开关量输入通道 16
28	AD17	模拟量或开关量输入通道 17
29	AD18	模拟量或开关量输入通道 18
30	AD19	模拟量或开关量输入通道 19
31	AD20	模拟量或开关量输入通道 20
32	AD21	模拟量或开关量输入通道 21
33	AD22	模拟量或开关量输入通道 22
34	AD23	模拟量或开关量输入通道 23
35	AD24	雨量脉冲信号24
36	AD25	雨量脉冲信号 25

1.6、硬件功能说明

1.6.1、 模拟量输入

本卡有 23 路模拟量输入，分辨率为 12 Bit，可分辨输入量程的 1/4096。采集卡电流型通道的输入阻抗为 150 欧姆，电压型通道的输入阻抗一般为几百 K 欧姆。ADC 的采样频率是 1Msps，卡内处理器对高速采集到的数据进行滤波去噪处理，保证信号的稳定性。

根据传感器的不同信号输出形式有不同的接线方法。下面分别列出二线制、三线制、四线制传感器的接线方法。注意：ADn 为 AD1-AD25 任何一个接线端子。

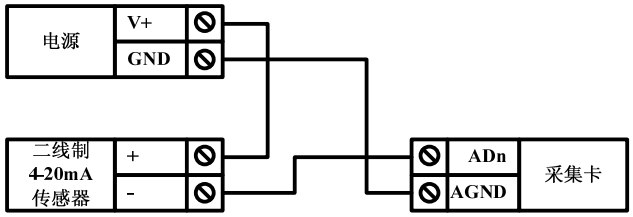


图 6-1A．二线制接线方法

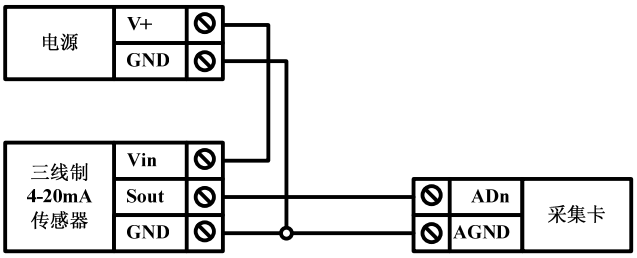


图 6-1B．三线制接线方法

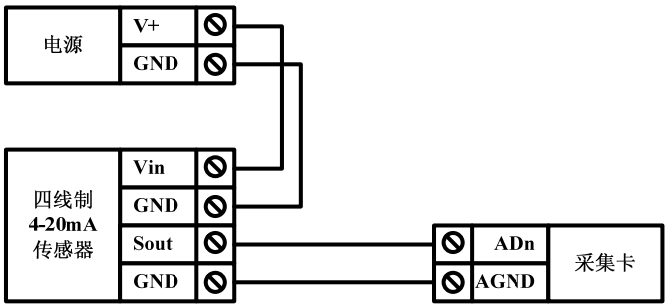


图 6-1C．四线制接线方法

1.6.2、 开关量输入

每个模拟量输入口都可以作为开关量输入口，输入模拟量的量程为 Amax，输入的实际值为 Ax，下表列出开关量输入的值。

输入口的值	开关量输入状态
$Ax < \frac{1}{3} \cdot Amax$	0
$Ax > \frac{2}{3} \cdot Amax$	1
$\frac{1}{3} \cdot Amax \leqslant Ax \leqslant \frac{2}{3} \cdot Amax$	不变

开关量输入口可接干湿触点，接近开关，电压电流信号等。

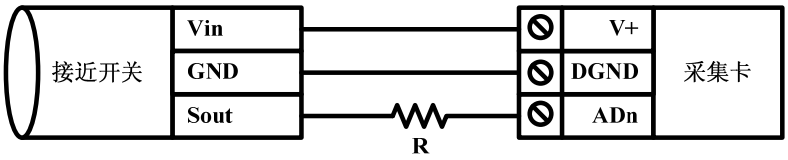


图 6-2. 接近开关接线图（PNP 型），其中 R 为限流电阻，若采集卡为电压输入型，R 可省去，直接用导线连通，若采集卡为电流型，R 要能够保证输入的最大电流小于 20mA，其值大约为 R=50*V-150，R 单位为欧姆，其中 V 为电源电压。为方便选择，其值可略大于理论值。推荐 24V 时 R=1.2K，12V 时 R=510 欧。

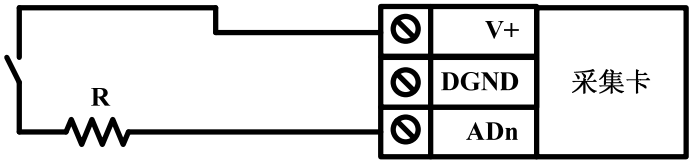


图 6-3. 干触点开关接线图，其中 R 为限流电阻，若采集卡为电压输入型，R 可省去，直接用导线连通，若采集卡为电流型，R 要能够保证输入的最大电流小于 20mA，其值大约为 R=50*V-150，R 单位为欧姆，其中 V 为电源电压。为方便选择，其值可略大于理论值。推荐 24V 时 R=1.2K，12V 时 R=510 欧。

1.6.3、 开关量输出

本卡有 2 个通道开关量输出，输出类型为 NPN 型的晶体管，内部原理图如图 6-4 所示，其中 RD 为上拉电阻，D 为续流二极管。设定开关量为 1 时，三极管导通，导通压降约 1V，最大负载电流为 500mA，可驱动报警器、指示灯、继电器、蜂鸣器、小型电机等。

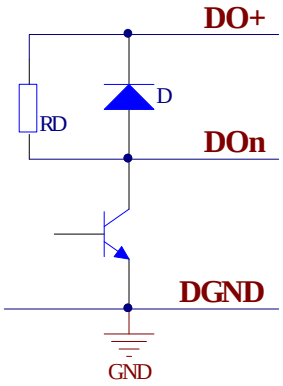


图 6-4. 采集卡中开关量输出内部原理图。

接负载时，接线原则是电流从负载中通过 DOn 流向 DGND 即可，如果是感性负载，DO+一定要接驱动负载的电源正极。如接继电器如图 6-5 所示。如果负载的电压与系统电压不同，可分开供电，接线方法如图 6-6 所示。

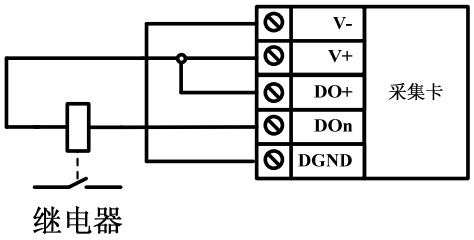


图 6-5. 采集卡开关量输出接线方法（负载与系统用同一电源）。

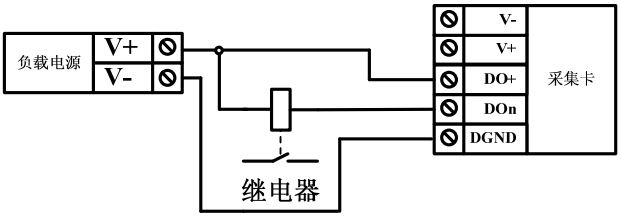


图 6-5. 采集卡开关量输出接线方法（负载与系统用不同电源）。

1.6.4、 RS485 通讯

本卡中 RS485 通讯有三个接口，485A+，485B-，485G，485A+接其他设备 RS485 通讯接口的正（常用的其他标记为 A，D+，T/R+等），485B-接其他设备 RS485 通讯接口的负（常用的其他标记为 B，D-，T/R-等），485G 接其他设备 RS485 通讯接口的参考地地线（常用的其他标记为 GND，0V，REF 等），如果有些设备没有参考地，可以接其电源的负极。一般要接 RS485 的地线，可大大提高数据传输的稳定性。

对于具有多个设备的 485 总线，采用手拉手的方式，可有效防止信号的反射，保证传输信号的完整性，如图 6-6 所示。

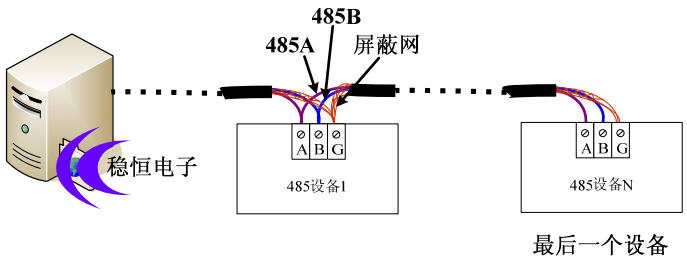


图 6-6. 多个 485 设备采用的手拉手接线方法

根据长期的使用经验及用户反馈，把一些注意事项列出，以供参考。

a) 接地

电子系统的接地是一个非常关键而又常常被忽视的问题，485 同样如此。一个典型的错误观点就是认为 RS-485 通信链路不需要信号地，而只是简单地用一对双绞线将各个接口的“A”、“B”端连接起来。这种处理方法在某些情况下也可以工作，但给系统埋下了隐患，主要有共模干扰和电磁辐射（EMI）两方面的问题。因此，尽管是差分传输，对于 RS-485 网络来讲，一条低阻的信号地还是必不可少的。一条低阻的信号地将两个接口的工作地连接起来，使共模干扰电压 VGPD 被短路。这条信号地可以是额外的一对线（非屏蔽双绞线）、或是屏蔽双绞线的屏蔽层。我们的 485 接口都提供接地端子。

b) 拓扑结构

这个可能大家都熟知，485 总线尽量避免环形拓扑结构和星形拓扑结构，最好采用一条总线将各个节点串接起来，从总线到每个节点的引出线长度应尽量短，以便使引出线中的反射信号对总线信号的影响最低。

最后，给出一个典型的系统接线图。

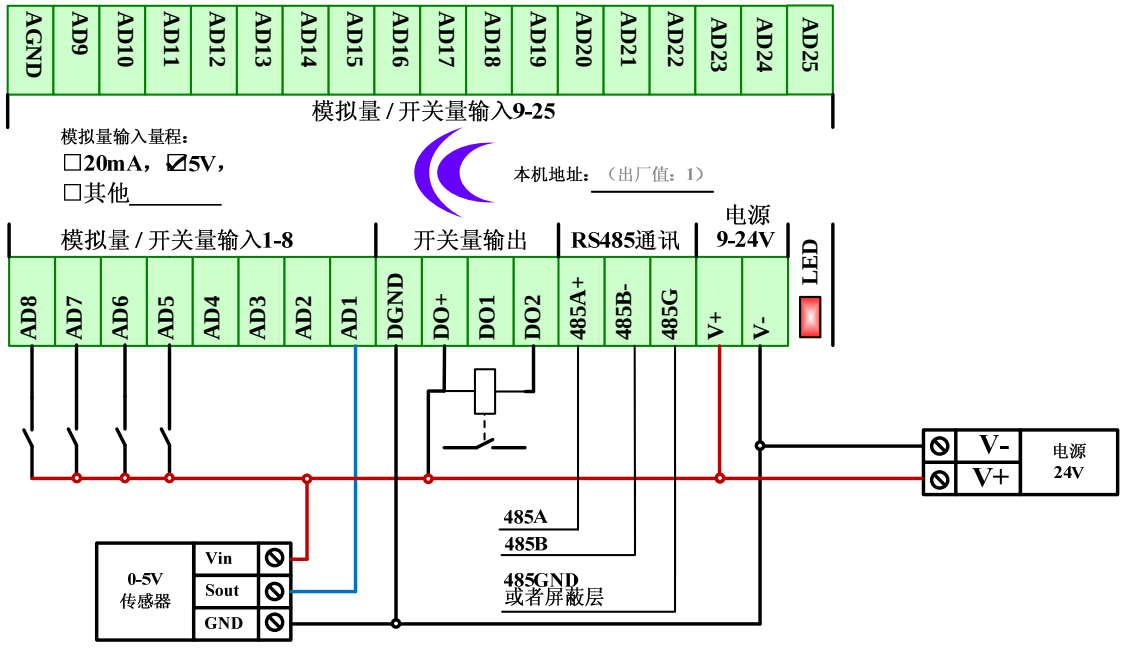


图 6-7. 系统典型接线图（采集卡为 0-5V 电压型）

二、软件部分

2.1、通讯出厂默认参数：

- 通讯接口：　RS485。
- 通讯格式：　1 个起始位，8 个数据位，无校验，1 个停止位。
- 波特率：　9600（可软件更改）。
- 地址：　1（可软件更改）。
- 通讯方式：　监控主机与本装置采用一对一（或一对多）主从查询方式。
- 数据协议：　MODBUS-RTU

数据格式：标准的 MODBUS 协议 RTU 方式，16 进制编码，若数字不是一个字节，则高字节在前低字节在后，主从式传输，每一帧有固定的格式，包括几个字节的数据，

2.2、数据存储格式及算法

采集卡中的数据及配置参数存储在卡内自带的内存中，可通过软件协议进行读写，本节中逐一对存储的数据及数据的含义、算法转换等进行解释。对存储数据进行读写操作的方法，将在后续章节介绍。

2.2.1、模拟量输入

模拟量输入经过 12bit 分辨率的 ADC 进行模数转换，得到的数据为 2 个字节，数据范围是 0-4095（0-0xFFF），所存储的 MODBUS 地址为 30001~30029，可用功能码 4 读取。为了有更好的兼容性，同时也备份存储在 40001~40029 的 MODBUS 地址中，可用功能码 3 读取。其中 30001~30025（或 40001~40025）为 1-25 通道采集到的 ADC 值，30026（或 40026）为板卡内基准电压的 ADC 值（D_{REFX}），30027（或 40027）为板卡内温度传感器的 ADC 值，30028（或 40028）为板卡内基准电压的出厂校准值（D_{REF0}），30029（或 40029）为板卡内温度传感器在 90℃是的校准值。

1~25 通道模拟量的算法为：

$$A_X = D_X \times \frac{A_{max}}{4095}$$

其中：A_X为模拟量输入值，D_X为读取的 ADC 值，A_{max}为模拟量的最大量程。

例如：采集卡模拟量输入为 0-20mA 电流型（板卡的最大量程为 22mA），我们从 30001（或 40001）读取 ADC 值为 2578，那么把 22mA 代入上式中A_{max}，2578 代入上式中D_X，计算可得采集到的电流为 2578*22mA/4095 = 13.85mA。

为了提高采集精度，可利用板卡内部基准电压来校准采集到的模拟量值，校准的算法为：

$$A_X = D_X \times \frac{A_0}{4095} \times \frac{D_{REF0}}{D_{REFX}}$$

其中：A_X为模拟量输入值，D_X为读取的 ADC 值，A_0为模拟量的额定量程，D_{REF0}为板卡内基准电压的出厂校准值，D_{REFX} 为板卡内基准电压的实时测量值。

内部温度传感器的算法为：

$$T = D_T \times \frac{3300}{4095 \times 1.62} - 273.15$$

其中：T为温度，单位为摄氏度，D_T为读取的 ADC 值。

若使用内部基准电压进行校准，算法为：

$$T = D_T \times \frac{3000}{4095 \times 1.62} \times \frac{D_{REF0}}{D_{REFX}} - 273.15$$

其中：T为温度，单位为摄氏度，D_T为读取的 ADC 值，D_{REF0}为板卡内基准电压的出厂校准值，D_{REFX}为板卡内基准电压的实时测量值。

2.2.2、开关量输入

每一路模拟量输入都可以作为开关量输入，采集到的开关量输入的数据所存储的 MODBUS 地址为 10001~10025，每个地址中存储一个位（bit），当输入为高电平时，其值为 1，当输入为低电平时，其值为 0，可通过功能码 2 来读取。为了有更好的兼容性，同时也备份存储在 40030 和 40031 的 MODBUS 地址中，它们以字的方式存储，每个字包含 16 个 bit，可表示 16 路开关量输入的值，若开关量输入为高电平，对应位为 1，反之对应位为 0。40030 存储开关量输入通道 1 到通道 16 的值，最低位为第 1 通道，最高位为第 16 通道，40031 存储开关量输入通道 17 到通道 125 的值，最低位为第 17 通道，第 9 位为第 25 通道，10~16 位未用，补充 0。40030 和 40031 可用功能码 3 读取。

2.2.3、开关量输出

本卡有 2 路开关量输出，其值所存储的 MODBUS 地址为 00001 和 00002，可用功能码 1 读，可用功能码 5 和 15 进行写。读到的值为当前开关量输出的值，写可以改变当前开关量输出的值。当其值为 1 时，开关量输出的 NPN 晶体管导通，即输出通道与地导通。为了有更好的兼容性，开关量输出的值也备份存储在 40032 的 MODBUS 地址中，最低第 1 位为第 1 路开关量输出，第 2 位为第 2 路开关量输出，其他位未用，恒为 0。40032 可用功能码 3 进行读，可用功能码 6 和 16 进行写。

2.2.4、板卡参数设置

本卡的参数包括序列号、卡内信息、波特率、从机地址、用户存储空间等。

序列号 本卡有 16 个字节（8 个字）的序列号，存储在 48193~48200 的 MODBUS 地址中，可通过功能码 3 来读取。序列号为板卡的唯一标识，每个卡都不一样，用户可以以此分辨是哪个卡，也可以用此序列号来加密配套的软件等，从而保护用户的知识产权。

卡内信息 MODBUS 地址 48201~48204 中存储卡内的信息，包含开关量模拟量输入输出通道数、版本号和版本日期，可通过功能码 3 来读取，具体含义参考附录 1。

波特率 波特率是表示 RS485 通讯的速率，存储在 MODBUS 地址 48205 中，可用功能码 3 进行读，可用功能码 6 和 16 进行写，写过之后重启板卡生效，断电后数据不丢失。其中的数据为 0~12，分别表示 300bps~921600bps。具体对应关系请参考下表。该地址中出厂默认值为 5，即波特率为 9600。

表 1：MODBUS 地址 48205 中设置值与波特率对应表（并不是所有可设置的数值都可用，为温度性考虑，建议在 115200bps 以下）

数据	0	1	2	3	4	5	6
波特率	300	600	1200	2400	4800	9600	19200
数据	7	8	9	10	11	12	
波特率	38400	57600	115200	230400	460800	921600	

从机地址 从机地址在通讯协议中用来区分不同的卡，挂在同一条 RS485 总线上的从机必须地址不同。从机地址存储在 MODBUS 地址 48206 中，可用功能码 3 进行读，可用功能码 6 和 16 进行写，写过之后重启板卡生效，断电后数据不丢失。其中的数据为 1~255，该地址中出厂默认值为 1。

注意：

i. 设定完地址和波特率后，仍然按照原地址或者波特率进行通讯，只有断电后重启新设定的波特率和地址才生效。

ii. 如果忘记原地址，可以用广播方式修改，但是必须保证只有一个从机连在主机总线上，否则会把所有从机改为同一个地址。

iii. 如果地址和波特率都忘记，可恢复出厂设置，方法为先断电，把板子上 RESET 处（如右图所示）的两个孔短路然后重新上电，上电后断开即可。

工厂参数 MODBUS 地址 48207~48210 中存储了工厂参数，为工厂调试所用，进制对其写操作，否则可能会产生工作异常等不可预期后果。

用户存储空间 卡中有 16 个字节的用户存储空间，留给用户存储数据所用，如可存储公司信息，板卡所处位置等，可随意进行读写，对板卡工作不产生任何影响。存储在 MODBUS 地址 48211~48218 中，可用功能码 3 进行读，可用功能码 6 和 16 进行写，断电后数据不丢失。

2.3、MODBUS 协议帧格式

标准的 MODBUS-RTU 协议包含从机地址、功能码、寄存器地址、寄存器数目、数据长度、数据及 CRC 校验等。最后两个字节为 CRC 校验，用来校验所收到数据帧的正确性。其中，从机地址是指要操作的采集卡的地址，只有对应地址的板卡执行该帧的命令，若数据帧中的地址为 0，则为广播地址，即所有板卡都执行命令；功能码用来指示从机要执行的功能，如读写寄存器等；寄存器地址表示对应功能码下需要访问的寄存器的起始地址；寄存器数目是指对应功能码下需要访问的寄存器的数目；数据长度表示数据区的字节数，一般为 1 个字节；数据即所要读写的数据，CRC 用来校验一帧数据的正确性，防止数据在传输过程中出错。CRC 校验有两个字节，高字节在前，所校验的字节包括数据帧中 CRC 前面所有的字节。CRC 的计算方法见后面附录校验值 CRC16 生成的计算方法。

在介绍通讯协议之前，首先解释一下 MODBUS 地址和寄存器地址的区别。MODBUS 地址是以 0,1,3,4 开头的，例如 0xxxx，1xxxx，3xxxx，4xxxx，前面的“0,1,3,4”只是地址的类型，不是地址的数值，而寄存器地址没有前面的“0,1,3,4”。MODBUS 地址是从 1 开始计，寄存器地址是从 0 开始计，MODBUS 地址转换为寄存器地址就是去掉开头的一位数字，剩下的 4 位再减 1。如 MODBUS 地址 40005、30001，寄存器地址是 4、0，不同的 MODBUS 地址可能寄存器地址是相同的，但是操作这些寄存器的功能码不同。

下面对本板卡支持的功能码逐一进行详细解释，并给出一些例子，方便对本板卡的开发和使用，用户可参考本部分利用编程语言进行软件开发。若使用组态软件或者 PLC 进行通讯，可跳过该部分，直接参考 MODBUS 寄存器一览表，表中描述了数据所存储的 MODBUS 地址及地址中数据的含义。

声明：以下所有示例命令中的数据为 16 进制，文中若有 16 进制用 H 结尾或者 0x 开头，如 0FH、0x10，地址为默认的 01。

2.3.1、 功能码 1（01H）读开关量输出——读地址 0XXXX

功能码 1 可以读取 MODBUS 地址 0xxxx 的数据。

主机发送帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	CRC 校验（2Byte）
SlaveID	01	regH regL	numH numL	crcH crcL

从机回复帧格式：

从机地址（1Byte）	功能码（1Byte）	数据长度（1Byte）	数据（nByte）	CRC 校验（2Byte）
SlaveID	01	n	D_0~D_{n-1}	crcH crcL

解释：regH regL 表示要访问寄存器的起始地址，numH numL 表示要读取的开关量个数。本卡有 2 路开关量输出，回复的数据中有 1 个字节数据。如，要读取 00001、00002 的开关量输出状态，寄存器起始地址为 0（00 00H），开关量个数为 2（00 02H）。回复数据帧中数据长度为 1（01H），数据为 1 个字节，字节中按位表示开关量的状态，bit0 表示第 1 路开关，bit1 表示第 2 路开关，对应位为 1 表示闭合，为 0 表示断开，bit3~bit7 未用，为 0。

假设第 2 路闭合，第 1 路断开，则回复的数据为 02H。

例 1：读取 2 路开关量输出（第 2 路闭合）

主机发送：01 01 00 00 00 02 BD CB

从机返回：01 01 01 02 D0 49

2.3.2、 功能码 2（02H） 读取开关量输入——读地址 1XXXX

功能码 2 可以读取 MODBUS 地址 1xxxx 的数据。

主机发送帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	CRC 校验（2Byte）
SlaveID	02	regH regL	numH numL	crcH crcL

从机回复帧格式：

从机地址（1Byte）	功能码（1Byte）	数据长度（1Byte）	数据（nByte）	CRC 校验（2Byte）
SlaveID	02	n	D_0~D_{n-1}	crcH crcL

解释：regH regL 表示要访问寄存器的起始地址，numH numL 表示要读取的开关量个数。本卡有 25 路开关量输入，回复的数据中一个字节中包含 8 个 bit（bit0~bit7）表示 8 个开关量输入的状态，所以 25 路状态最多有 4 个字节数据。如（例 1），要读取 10001~10025 的开关量输入状态，寄存器起始地址为 0（00 00H），开关量个数为 25（00 19H）。回复数据帧中数据长度为 04H，数据为 4 个字节，字节中按位表示开关量的状态，第一个字节的 bit0 表示从寄存器起始地址开始的第 1 路开关，bit1 表示从寄存器起始地址开始的第 2 路开关，第 2 个字节的 bit0 表示从寄存器起始地址开始的第 9 路开关状态，以此类推。数据中对应位为 1 表示闭合，为 0 表示断开。又如（例 2），要读取 10023、10024、10025 的开关量输入状态，寄存器起始地址为 22（00 16H），开关量个数为 3（00 03H）。回复数据帧中数据长度为 01H，数据为 1 个字节，字节中按位表示开关量的状态，第一个字节的 bit0 表示从寄存器起始地址开始的第 1 路开关，即第 23 路开关量输入，bit1、bit2 表示第 24、25 路开关输入状态。

注：此处的“闭合”是指开关量输入端与电源正极接通，或者输入端为高电平。

假设第 2、4、25 路闭合，其他断开。

例 1，读取所有 25 路开关量输入状态：

主机发送：01 02 00 00 00 19 B9 C0

从机返回：01 02 04 0A 00 00 01 39 FA

例 2，读取第 23、24、25 路开关量输入状态：

主机发送：01 02 00 16 00 03 D9 CF

从机返回：01 02 01 04 A0 4B

2.3.3、 功能码 3（03H）读取保持寄存器——读地址 4XXXX

功能码 3 可以读取 MODBUS 地址 4xxxx 的数据。

主机发送帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	CRC 校验（2Byte）
SlaveID	03	regH regL	numH numL	crcH crcL

从机回复帧格式：

从机地址（1Byte）	功能码（1Byte）	数据长度（1Byte）	数据（nByte）	CRC 校验（2Byte）
SlaveID	03	2n	$D_{0H}D_{0L}$~ $D_{nH}D_{nL}$	crcH crcL

解释：regH regL 表示要访问寄存器的起始地址，numH numL 表示要读取寄存器的个数。回复的帧中数据长度为 2n，因为 4xxxx 是以字（16bit）的方式存储，一个字为 2 个字节，所以数据肯定为偶数个字节。数据中 D_{nH} 和 D_{nL} 分别表示第 n 个寄存器的高字节和低字节，高字节在前。D_{0H} 和 D_{0L} 是从寄存器的起始地址算起的第一个字的高低字节。

例 1：读取第 2、3、4 通道的模拟量输入（40002、40003、40004）

主机发送：01 03 00 01 00 03 54 0B

从机返回：01 03 06 D_{0H} D_{0L} D_{1H} D_{1L} D_{2H} D_{2L} crcH crcL

回复的数据中 D_{0H} 和 D_{0L} 是起始寄存器地址 40002 的数值，其模拟量的 ADC 算法为

$D = 256 \times D_{0H} + D_{0L}$

例 2：读取 1-25 通道的开关量输入（40030、40031）

主机发送：01 03 00 1D 00 02 54 0D

从机返回：01 03 04 D_{0H} D_{0L} D_{1H} D_{1L} crcH crcL

回复的数据中 D_{0H} 和 D_{0L} 是起始寄存器地址 40030 的数值，D_{0H} 和 D_{0L} 是寄存器地址 40031 的数值，D_{0H} 和 D_{0L} 组成一个 16 位的字 D_0，其 bit0 表示第 1 通道的开关量输入的值，bit15 表示第 16 路开关量输入值，同理 D_1 的 bit0 表示第 17 路的开关量输入值，bit8 表示第 25 路开关量输入值，bit9~bit15 未用，为 0。对应位为 1 表示输入高电平，为 0 表示输入低电平。

例 3：读取波特率（48205）

主机发送：01 03 20 0C 00 01 4F C9

从机返回：01 03 02 D_{0H} D_{0L} crcH crcL

回复的数据中 D_{0H} 和 D_{0L} 是起始寄存器地址 48205 的数值， D_{0H} 和 D_{0L} 组成一个 16 位的字 D_0，为波特率的代码。

2.3.4、 功能码 4（04H） 读取模拟量输入——读地址 3XXXX

功能码 4 可以读取 MODBUS 地址 3xxxx 的数据。

主机发送帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	CRC 校验（2Byte）
SlaveID	04	regH regL	numH numL	crcH crcL

从机回复帧格式：

从机地址（1Byte）	功能码（1Byte）	数据长度（1Byte）	数据（nByte）	CRC 校验（2Byte）
SlaveID	04	2n	$D_{0H}D_{0L}$~ $D_{nH}D_{nL}$	crcH crcL

解释：regH regL 表示要访问寄存器的起始地址，numH numL 表示要读取寄存器的个数。回复的帧中数据长度为 2n，因为 3xxxx 是以字（16bit）的方式存储，一个字为 2 个字节，所以数据肯定为偶数个字节。数据中 D_{nH} 和 D_{nL} 分别表示第 n 个寄存器的高字节和低字节，高字节在前。D_{0H} 和 D_{0L} 是从寄存器的起始地址算起的第一个字的高低字节。

例 1：读取第 2、3、4 通道的模拟量输入（30002、30003、30004）

主机发送：01 04 00 01 00 03 E1 CB

从机返回：01 04 06 D_{0H} D_{0L} D_{1H} D_{1L} D_{2H} D_{2L} crcH crcL

回复的数据中 D_{0H} 和 D_{0L} 是起始寄存器地址 30002 的数值，其模拟量的 ADC 算法为

$D = 256 \times D_{0H} + D_{0L}$

2.3.5、 功能码 5（05H） 设置单路开关量输出——写地址 0XXXX

功能码 5 可以写 MODBUS 地址 0xxxx 的数据。

主机发送帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数据（2Byte）	CRC 校验（2Byte）
SlaveID	05	regH regL	D_H D_L	crcH crcL

从机回复帧格式（与发送帧相同）：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数据（2Byte）	CRC 校验（2Byte）
SlaveID	05	regH regL	D_H D_L	crcH crcL

解释：regH regL 表示要访问寄存器的地址，D_H D_L 为操作寄存器的数据，当其为 FF00H 时寄存器的数据位置 1 闭合，当为 0000H 时置 0 断开，其他值非法。例如，要设置第 2 路开关量输出的状态为闭合，则寄存器地址为 00 01H，寄存器数据 FF00H。

例：第 2 路闭合

主机发送：01 05 00 01 FF 00 DD FA

从机返回：01 05 00 01 FF 00 DD FA

2.3.6、 功能码 6（06H） 设置单个保持寄存器——写地址 4XXXX

功能码 6 可以写单个 MODBUS 地址 4xxxx 的数据。

主机发送帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数据（2Byte）	CRC 校验（2Byte）
SlaveID	06	regH regL	D_H D_L	crcH crcL

从机回复帧格式（与发送帧相同）：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数据（2Byte）	CRC 校验（2Byte）
SlaveID	06	regH regL	D_H D_L	crcH crcL

解释：regH regL 表示要访问寄存器的地址，D_H D_L 为要写入寄存器的数据。例如，要设置从机地址（MODBUS 地址 48206）为 17，则寄存器地址为 20 0DH，寄存器数据 0011H。

例：在 MODBUS 地址 48206 中写入 17，即改从机地址为 17。

主机发送：01 06 20 0D 00 11 D3 C5

从机返回：01 06 20 0D 00 11 D3 C5

2.3.7、 功能码 15（0FH） 设置多路开关量输出——写地址 0XXXX

功能码 15 可以设置 MODBUS 地址 0xxxx 的数据。

主机发送帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	数据长度（1Byte）	数据（nByte）	CRC 校验（2Byte）
SlaveID	0F	regH regL	numH numL	n	D_0~D_{n-1}	crcH crcL

从机回复帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	CRC 校验（2Byte）
SlaveID	0F	regH regL	numH numL	crcH crcL

解释：regH regL 表示要访问寄存器的起始地址，numH numL 表示要写的开关量个数，n 为写入数据的字节数，D_0~D_{n-1} 为要写入的数据。本卡有 2 路开关量输出，写入的数据中有 1 个字节数据。如，要写 00001、00002 的开关量输出状态，寄存器起始地址为 0（00 00H），开关量个数为 2（00 02H），数据长度为 1（01H），数据为 1 个字节，数据字节中按位表示开关量的状态，bit0 表示第 1 路开关，bit1 表示第 2 路开关，对应位为 1 表示闭合，为 0 表示断开，bit3~bit7 未用，为 0。

假设第 2 路闭合，第 1 路断开，则回复的数据为 02H。

例：设置开关量输出第 2 路闭合，第 1 路断开

主机发送：01 0F 00 00 00 02 01 02 5F 56

从机返回：01 0F 00 00 00 02 D4 0A

2.3.8、 功能码 16（10H） 设置多个保持寄存器——写地址 4XXXX

功能码 16 可以设置 MODBUS 地址 4xxxx 的数据。

主机发送帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	数据长度（1Byte）	数据（nByte）	CRC 校验（2Byte）
SlaveID	10	regH regL	numH numL	2n	$D_{0H}D_{0L}$~ $D_{nH}D_{nL}$	crcH crcL

从机回复帧格式：

从机地址（1Byte）	功能码（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	CRC 校验（2Byte）
SlaveID	10	regH regL	numH numL	crcH crcL

解释：regH regL 表示要访问寄存器的起始地址，numH numL 表示要读取寄存器的个数，数据长度为 2n，因为 4xxxx 是以字（16bit）的方式存储，一个字为 2 个字节，所以数据肯定为偶数个字节。数据中 D_{nH} 和 D_{nL} 分别表示第 n 个寄存器的高字节和低字节，高字节在前。D_{0H} 和 D_{0L} 是从寄存器的起始地址算起的第一个字的高低字节。

例：写入用户存储空间（MODBUS 地址 48211~48218）为 0102H、 0304H、0506H、0708H、090AH、0B0CH、0D0EH、FFFFH

主机发送：01 10 20 12 00 08 10 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E FF FF 16 99

从机返回：01 10 20 12 00 08 6A 0A

附录

附录 1：MODBUS 寄存器地址一览表

寄存器	名称	备注
开关量输出		支持功能码 1,5,15
00001	DO1	开关量输出通道 1
00002	DO2	开关量输出通道 2
00003	-	保留
开关量输入		支持功能码 2
10001	DI1	开关量输入通道 1
10002	DI2	开关量输入通道 2
10003	DI3	开关量输入通道 3
10004	DI4	开关量输入通道 4
10005	DI5	开关量输入通道 5
10006	DI6	开关量输入通道 6
10007	DI7	开关量输入通道 7
10008	DI8	开关量输入通道 8
10009	DI9	开关量输入通道 9
10010	DI10	开关量输入通道 10
10011	DI11	开关量输入通道 11
10012	DI12	开关量输入通道 12
10013	DI13	开关量输入通道 13
10014	DI14	开关量输入通道 14
10015	DI15	开关量输入通道 15
10016	DI16	开关量输入通道 16
10017	DI17	开关量输入通道 17
10018	DI18	开关量输入通道 18
10019	DI19	开关量输入通道 19
10020	DI20	开关量输入通道 20
10021	DI21	开关量输入通道 21
10022	DI22	开关量输入通道 22
10023	DI23	开关量输入通道 23
10024	DI24	开关量输入通道 24
10025	DI25	开关量输入通道 25
模拟量输入		支持功能码 4
30001	AI1	模拟量输入通道 1
30002	AI2	模拟量输入通道 2
30003	AI3	模拟量输入通道 3

30004	AI4	模拟量输入通道 4
30005	AI5	模拟量输入通道 5
30006	AI6	模拟量输入通道 6
30007	AI7	模拟量输入通道 7
30008	AI8	模拟量输入通道 8
30009	AI9	模拟量输入通道 9
30010	AI10	模拟量输入通道 10
30011	AI11	模拟量输入通道 11
30012	AI12	模拟量输入通道 12
30013	AI13	模拟量输入通道 13
30014	AI14	模拟量输入通道 14
30015	AI15	模拟量输入通道 15
30016	AI16	模拟量输入通道 16
30017	AI17	模拟量输入通道 17
30018	AI18	模拟量输入通道 18
30019	AI19	模拟量输入通道 19
30020	AI20	模拟量输入通道 20
30021	AI21	模拟量输入通道 21
30022	AI22	模拟量输入通道 22
30023	AI23	模拟量输入通道 23
30024	AI24	模拟量输入通道 24
30025	AI25	模拟量输入通道 25
30026	VREF	模拟量输入基准电压
30027	TEMPER	模拟量输入温度传感器
30028	VREF0	模拟量输入基准电压出厂校正值
30029	TEMPER90	模拟量输入温度校正值
保持寄存器		支持功能码 3,6,16
40001	AI1	模拟量输入通道 1
40002	AI2	模拟量输入通道 2
40003	AI3	模拟量输入通道 3
40004	AI4	模拟量输入通道 4
40005	AI5	模拟量输入通道 5
40006	AI6	模拟量输入通道 6
40007	AI7	模拟量输入通道 7
40008	AI8	模拟量输入通道 8
40009	AI9	模拟量输入通道 9
40010	AI10	模拟量输入通道 10
40011	AI11	模拟量输入通道 11
40012	AI12	模拟量输入通道 12
40013	AI13	模拟量输入通道 13
40014	AI14	模拟量输入通道 14

40015	AI15	模拟量输入通道 15
40016	AI16	模拟量输入通道 16
40017	AI17	模拟量输入通道 17
40018	AI18	模拟量输入通道 18
40019	AI19	模拟量输入通道 19
40020	AI20	模拟量输入通道 20
40021	AI21	模拟量输入通道 21
40022	AI22	模拟量输入通道 22
40023	AI23	模拟量输入通道 23
40024	AI24	模拟量输入通道 24
40025	AI25	模拟量输入通道 25
40026	VREF	模拟量输入基准电压
40027	TEMPER	模拟量输入温度传感器
40028	VREF0	模拟量输入基准电压出厂校正值
40029	TEMPER90	模拟量输入温度校正值
40030	DIN1-16	开关量输入通道 1 到通道 16 的值，最低位为第 1 通道
40031	DIN17-25	开关量输入通道 17 到通道 25 的值，最低位为第 17 通道
40032	DOUT1-2	开关量输出通道 1 和 2 的值，最低位为第 1 通道
40033	RES	无意义
40034-40092	forbidden	不可读写，否则产生不可预测后果
48193	CARD_SN0	板卡唯一序列号 0
48194	CARD_SN1	板卡唯一序列号 1
48195	CARD_SN2	板卡唯一序列号 2
48196	CARD_SN3	板卡唯一序列号 3
48197	CARD_SN4	板卡唯一序列号 4
48198	CARD_SN5	板卡唯一序列号 5
48199	CARD_SN6	板卡唯一序列号 6
48200	CARD_SN7	板卡唯一序列号 7
48201	CARD_INF0	板卡信息 0，高字节开关量输入数，低字节开关量输出数
48202	CARD_INF1	板卡信息 1，高字节模拟量输入数，低字节模拟量输出数
48203	CARD_INF2	板卡信息 2，高字节版本号，低字节年份
48204	CARD_INF3	板卡信息 3，高字节月份，低字节日期
48205	BAND_RATE	波特率：0-12，表示 300bps-921600bps
48206	SLAVE_ID	从机地址：1-255,0 为广播地址
48207	FAC0	工厂调试所用，不要写，否则产生不可预期后果
48208	FAC1	工厂调试所用，不要写，否则产生不可预期后果
48209	FAC2	工厂调试所用，不要写，否则产生不可预期后果
48210	FAC3	工厂调试所用，不要写，否则产生不可预期后果
48211	USER0	为用户存储所用，可以读写，写入后掉电保存
48212	USER1	为用户存储所用，可以读写，写入后掉电保存
48213	USER2	为用户存储所用，可以读写，写入后掉电保存

48214	USER3	为用户存储所用，可以读写，写入后掉电保存
48215	USER4	为用户存储所用，可以读写，写入后掉电保存
48216	USER5	为用户存储所用，可以读写，写入后掉电保存
48217	USER6	为用户存储所用，可以读写，写入后掉电保存
48218	USER7	为用户存储所用，可以读写，写入后掉电保存

附录 2：校验值 CRC16 生成的计算方法（C 语言）

CRC Generation

Example

The function takes two arguments:

`unsigned char *puchMsg ;` A pointer to the message buffer containing binary data to be used for generating the CRC

`unsigned short usDataLen ;` The quantity of bytes in the message buffer.

The function returns the CRC as a type `unsigned short`.

CRC Generation Function

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ; /* message to calculate CRC upon */
unsigned short usDataLen ; /* quantity of bytes in message */
{
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 ^ *puchMsgg++ ; /* calculate the CRC */
uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex} ;
uchCRCLo = auchCRCLo[uIndex] ;
}
return (uchCRCHi << 8 | uchCRCLo) ;
}
```

High-Order Byte Table

```
/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
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
} ;
```

Low-Order Byte Table

```
/* Table of CRC values for low-order byte */
static char auchCRCLo[] = {
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
} ;
```