MODBUS－RTU 通讯规约说明

力创科技 06-09-19

目录

一、MODBUS 通讯协议简介
二、通讯信息传输过程
三、MODBUS 功能码简介
四、错误校验码（CRC 校验）
五、通讯错误信息及数据的处理
附件：CRC 校验算法程序

一、MODBUS 通讯协议简介:

MODBUS 协议是应用于电子控制器上的一种通用语言。通过此协议，控制器相互之间、控制器经由网络（例如以太网）和其它设备之间可以通信。它已经成为一通用工业标准。有了它，不同厂商生产的控制设备可以连成工业网络，进行集中监控。

MODBUS 通讯规约允许 EDA 系列模块／仪表／变送器与多个品牌的 PLC、RTU、DCS 等或与第三方具有 MODBUS 兼容的监控系统之间进行信息交换和数据通讯。

MODBUS 是一个请求／应答协议，并且提供功能码规定的服务。MODBUS 是一种应用层报文传输协议，用于在通过不同类型的总线或网络连接的设备之间的客户机／服务器通信。

EDA 系列模块／仪表／变送器提供了标准的 RS-485／RS-232 通讯接口及 MODBUS-RTU 通讯协议；EDA 系列产品只要简单地增加一套基于计算机（或工控机）的监控软件（如：组态王、F1X 等）就可构成一套电力监控系统。

注:MODBUS 是 Modicon 公司的注册商标。

- 数据编码:

MODBUS 使用最高有效字节在低地址存储的方式表示地址与数据项。即当发送多个字节时，首先发送最高有效字节。例如:

寄存器大小　　值

16 位　　0x1234　　发送的第一字节为　0x12　然后发 0x34

- 通讯数据的类型及格式:

信息传输为异步方式，以字节为单位，每字节为 10 位的格式传输:

字格式（串行数据）	10 位二进制
起始位	1 位，0
数据位	8 位，最低的有效位先发送
奇偶校验位	无
停止位	1 位，1

通讯数据（信息帧）格式:

数据格式:	地址码	功能码	数据区	CRC 校验
数据长度:	1 字节	1 字节	N 字节	16 位 CRC 校验码（循环冗余码）

数据字节:　1 个字节由 8 位二进制数（8Bit）组成。

CRC 校验:　CRC 生成后，低字节在前，高字节在后。

- MODBUS-RTU 的帧结构:

在 RTU 模式中，新的信息总是以至少 3.5 个字符的静默时间开始。紧接着传送第一个域:设备地址。

整帧的信息必须以一个连续的数据流进行传输。如果信息结束前存在超过 1.5 个字符以上的间隔时间，则出错。

一帧信息的标准结构如下:

开始	地址域	功能域	数据域	CRC 校验	结束
T1-T2-T3-T4	8 位	8 位	n*8 位	16 位	T1-T2-T3-T4

二、通讯信息传输过程:

当通讯命令由发送设备（主机）发送至接收设备（从机）时，符合相应地址码的从机接收通讯命令，并根据功能码及相关要求读取信息，如果CRC 校验无误，则执行相应的任务，然后把执行结果（数据）返送给主机。返回的信息中包括地址码、功能码、数据区及CRC 校验码。如果CRC校验出错则不返回任何信息。

- **地址码:**

地址码是每次通讯信息帧的第一字节，从0 到255。这个字节表明由用户设置地址的从机将接收由主机发送来的信息。同一总线系统内的每个从机都必须有唯一的地址码，并且只有符合地址码的从机才能响应回送信息。当从机回送信息时，回送数据均以各自的地址码开始。主机发送的地址码表明将发送到的从机地址，而从机返回的地址码表明回送的从机地址。相应的地址码表明该信息来自于何处。

- **功能码:**

是每次通讯信息帧传送的第二个字节。MODBUS 通讯规约可定义的功能码为1到127。力创科技EDA系列模块／仪表／变送器仅用到其中的一部分功能码。 作为主机请求发送，通过功能码告诉从机应执行什么动作。作为从机响应，从机返回的功能码与从主机发送来的功能码一样，并表明从机已响应主机并且已进行相关的操作。

力创科技 EDA 系列 MODBUS 部分功能码如下表:

功能码	定　义	操　作
01	读开关量输出OUT	读取一路或多路开关量输出状态数据
02	读开关量输入DI	读取一路或多路开关量状态输入数（遥信）
03	读寄存器数据	读取一个或多个寄存器的数据
05	写开关量输出 OUT	控制一路继电器“分/合”输出，遥控
06	写单路寄存器	把一组二进制数据写入单个寄存器
0F	写多路开关量输出	写一路或多路开关量输出（遥控）
10	写多路寄存器	把多组二进制数据写入多个寄存器

- **数据区:**

数据区包括需要由从机返回何种信息或执行什么动作。这些信息可以是数据（如: 开关量输入/输出、模拟量输入/输出、寄存器等等）、参考地址等。例如， 主机通过功能码 03 告诉从机返回寄存器的值（包含要读取寄存器的起始地址及读取寄存器的长度），则返回的数据包括寄存器的数据长度及数据内容。对于不同的从机，地址和数据信息都不相同（可参照通讯信息表）。

EDA 系列模块／仪表／变送器采用 MODBUS－RTU 通讯规约，主机（PLC、RTU、PC 机、 DCS 等）利用通讯命令（功能码 03），可以任意读取其数据寄存器（其数据信息表详见相应说明书）。一次最多可读取寄存器个数是 *100* 个。EDA91 系列模块／仪表的数据寄存器存储的电量多达几百个（如: 电流、电压、功率、0～31 次谐波分量、需量等），每个参数都是 16 位（2 字节）的二进制数 据，并且高位在前;

- **CRC 校验:**

MODBUS－RTU 通讯协议的 CRC（冗余循环码）包含 2 个字节，即 16 位二进制数。低字节在前，高字节在后。其详细说明见后页;

- **静止时间要求:**

在 MODBUS-RTU 模式中，发送数据前要求数据总线静止时间即无数据发送时间至少大于 3.5 个字符的时间（如波特率为 9600 时为 3.6mS）；整帧的信息必须以一个连续的数据流进行传输。如果信息结束前存在超过 1.5 个字符以上的间隔时间，则出错。

三、MODBUS 功能码简介:

3.1 功能码 01（0x01）: 读 1 路或多路开关量输出状态

例 1: 主机要读取地址为 01，输出开关量第 0～15 路的共 16 路输出状态

主机发送的报文格式:

主机发送	字节数	发送信息	备注

从机地址	1	01	发送到地址为 01 的从机
功能码	1	01	读开关量输出状态
起始位	2	0000	起始 BIT 位地址为 0000
读开关量个数	2	0010	读取 16 路开关量输出状态
CRC 码	2	3DC6	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式:

从机响应	字节数	返回信息	备注
从机地址	1	01	来至从机 01
功能码	1	01	读开关量输出状态
数据长度	1	02	2 个字节（16 个 BIT 位）
OUT 状态数据	1	02 00	第一个字节的 BIT0 位对应开关量开始地址的状态位; 第一个字节对应开关量 7～0 状态位; 第二个字节对应开关量 15～8 状态位; 返回数据 0200 表示第 1 路开关量输出为“1”，其余为“0”;
CRC 码	2	B89C	由 EDA 模块计算得到的 CRC 码

例 2：主机要读取地址为 01，输出开关量 2 和 3 的输出状态（开关量 0、1、2、3……）

主机发送的报文格式:

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	01	读开关量输出状态
起始位	2	0002	起始 Bit 位地址为 0002
读开关量个数	2	0002	读取 2 路开关量输出状态
CRC 码	2	1C0B	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式:

从机响应	字节数	返回信息	备注
从机地址	1	01	来至从机 01
功能码	1	01	读开关量输出状态
数据长度	1	01	1 个字节（8 个 BIT 位）
OUT 状态数据	1	02	寄存器内容 BIT0 对应开关量输出 2 状态为“0”，BIT1 对应开关量输出 3 状态为“1”; Bit7～2 为用 0 填充的 6 个剩余位;
CRC 码	2	D04°	由 EDA 模块计算得到的 CRC 码

3.2 功能码 02（0x02）：读 1 路或多路开关量输入状态 DI

例 3：主机要读取地址为 01，开关量 DI0～15 的输入状态

主机发送的报文格式:

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	02	读开关量输入状态
起始位	2	0000	起始 BIT 位地址为 0000
读开关量个数	2	0010	读取 16 路开关量输入状态
CRC 码	2	79C6	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式:

从机响应	字节数	返回信息	备注
从机地址	1	01	来至从机 01
功能码	1	02	读开关量输入状态
数据长度	1	02	2 个字节（16 个 BIT 位）
DI 状态数据	1	0200	第一个字节的 BIT0 位对应开关量开始地址的状态位; 第一个字节对应开关量 7～0 状态位; 第二个字节对应开关量 15～8 状态位; 返回数据 0200 表示第 1 路开关量输入为“1”，其余为“0”;
CRC 码	2	B8D8	由 EDA 模块计算得到的 CRC 码

例 4：主机要读取地址为 01，开关量 DI1～3 的输入状态

主机发送的报文格式：

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	02	读开关量输入状态
起始位	2	0001	起始 BIT 位地址为 0001
读开关量个数	2	0003	读取 3 路开关量输入状态
CRC 码	2	69CB	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式：

从机响应	字节数	返回信息	备注
从机地址	1	01	来至从机 01
功能码	1	02	读开关量输入状态
数据长度	1	01	1 个字节（8 个 BIT 位）
DI 状态数据	1	02	数据 02H 的 Bit2、Bit0 对应开关量输入的 DI3、DI1 其状态为“0”，Bit1 对应开关量输入的 DI2 其状态为“1”；（因命令是从开关量输入的第 1 位开始读取）Bit7～3 为用 0 填充的 5 个剩余位；
CRC 码	2	2049	由 EDA 模块计算得到的 CRC 码

3.3 功能码 03（0x03）：读多路寄存器

例 5：主机要读取地址为 01，开始地址为 0106H 的 2 个从机寄存器数据

主机发送的报文格式：

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	03	读取寄存器
起始地址	2	0106	起始地址为 0106H
数据长度	2	0002	读取 2 个寄存器（共 4 字节）
CRC 码	2	25F6	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式：

从机响应	字节数	返回信息	备注
从机地址	1	01	来至从机 01
功能码	1	03	读取寄存器
返回字节数	1	04	2 个寄存器共 4 字节
寄存器数据 1	2	2710	地址为 0106 寄存器的内容
寄存器数据 2	2	1388	地址为 0107 寄存器的内容
CRC 码	2	FC14	由 EDA 模块计算得到的 CRC 码

3.4 功能码 05（0x05）：写 1 路开关量输出（遥控）

控制命令为：

“FF00”为输出开关量为“1”，即控制继电器“合”；“0000”为输出开关量为“0”，即控制继电器“分”。

例 6：主机要控制地址为 01，第 0 路开关量 DO0（或继电器）“合”

主机发送的报文格式：

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	05	写开关量输出状态
输出 Bit 位	2	0000	对应输出继电器 BIT0 位（DO0）
控制命令	2	FF00	控制该路继电器输出为“合”
CRC 码	2	8C3A	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式：与主机发送的报文格式及数据内容完全相同。

例 7：主机要控制地址为 01，第 1 路开关量 DO1（或继电器）“分”

主机发送的报文格式：

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	05	写开关量输出状态
输出 Bit 位	2	0001	对应输出继电器 BIT1 位（DO1）
控制命令	2	0000	控制该路继电器输出为“分”
CRC 码	2	9C0A	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式：与主机发送的报文格式及数据内容完全相同。

3.5 功能码 06（0x06）：写单路寄存器

例 8：主机要把数据 1388，保存到 1 号从机地址为 0001 的寄存器中去。

主机发送的报文格式：

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	06	写单路寄存器
起始地址	2	0001	要写入的寄存器地址
写入数据	2	1388	对应的写入数据
CRC 码	2	D55C	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式：与主机发送的报文格式及数据内容完全相同。

3.6 功能码 0F（0x0F）：写多路开关量输出（遥控）

例 9：主机要控制地址为 01，第 0、2 路继电器闭合，第 1、3 路继电器断开。

主机发送的报文格式：

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	0F	写多路开关量输出
起始位地址	2	0000	对应输出继电器起始 BIT0 位（DO0）
输出数量	2	0004	控制 DO0～DO3 共 4 路继电器
字节计数	1	01	1 字节数；字节数 N＝输出数量／8，若余数不等于 0，则 N=N+1。
输出数据	2	05	Bit3～Bit0：0101 为 0、2 路合，1、3 路分；其他未使用位用 0 填充
CRC 码	2	FE95	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式：

从机响应	字节数	返回信息	备注
从机地址	1	01	来至从机 01
功能码	1	0F	写多路开关量输出
起始位地址	2	0000	对应输出继电器起始 BIT0 位（DO0）
输出数量	2	0004	控制 DO0～DO3 共 4 路继电器
CRC 码	2	5408	由 EDA 模块计算得到的 CRC 码

3.7 功能码 10（0x10）：写多路寄存器

主机利用这个功能码把多个数据保存到 EDA 表的数据寄存器中去。

MODBUS 通讯规约中的寄存器指的是 16 位（即 2 字节），并且高位在前。

例 10：主机要把 0001，0014 保存到地址为 0002，0003 的从机寄存器中去（从机地址码为 01）。

主机发送的报文格式：

主机发送	字节数	发送信息	备注
从机地址	1	01	发送到地址为 01 的从机
功能码	1	10	写多路寄存器
起始地址	2	0002	要写入的寄存器起始地址

写寄存器数量	2	0002	要写入的寄存器个数
字节计数	1	04	要写入的数据字节长度
保存数据 1	2	0001	数据 0001 写入地址为 0002 的寄存器
保存数据 2	2	0014	数据 0014 写入地址为 0003 的寄存器
CRC 码	2	23B9	由主机计算出的 CRC 码

从机（EDA）响应返回的报文格式:

从机响应	字节数	返回信息	备注
从机地址	1	01	来至从机 01
功能码	1	10	写多路寄存器
起始地址	2	0002	要写入的寄存器起始地址
写寄存器数量	2	0002	要写入的寄存器个数
CRC 码	2	E008	由 EDA 模块计算得到的 CRC 码

四、错误校验码（CRC 校验）:

使用 MODBUS-RTU 模式，消息包括了一基于 CRC 方法的错误检测域。CRC 域检测了整个消息的内容。

主机或从机可用校验码进行判别接收信息是否正确。由于电子噪声或一些其它干扰，信息在传输过程中有时会发生错误，错误校验码（CRC）可以检验主机或从机在通讯数据传送过程中的信息是否有误，错误的数据可以放弃，这样增加了系统的可靠性及通讯效率。

CRC 域是两个字节，包含一 16 位的二进制值。它由传输设备计算后加入到消息中。接收设备重新计算收到消息的 CRC，并与接收到的 CRC 域中的值比较，如果两值不同，则有误。

在进行 CRC 计算时只用 8 个数据位，起始位及停止位和奇偶校验位都不参与 CRC 计算。

CRC 码的计算方法是:

1. 预置 1 个全“1”的 16 位 CRC 寄存器（0xFFFF）（即全为 1）;

2. 把第一个 8 位二进制数据（既信息帧的第一个字节）与 16 位的 CRC 寄存器的低 8 位相异或（XOR），把结果放于 CRC 寄存器的低 8 位;

3. 把 CRC 寄存器的内容右移一位（朝低位），用 0 填补最高位，并检查右移后的移出位;

4. 如果移出位为 1,则 CRC 寄存器与预置的值 A001（1010 0000 0000 0001）异或一下; 如果移出位为 0，则不进行。

5. 重复 8 次步骤 3 和 4，对整个 8 位数据全部进行处理;

6. 重复按步骤 2 到 5 的方法，进行通讯信息帧的下一个字节处理;

7. 将该通讯信息帧所有字节按上述步骤计算完成后，得到 16 位的 CRC 值;

8. CRC 添加到消息中时，低字节先加入，然后高字节。

五、通讯错误信息及数据的处理:

当 EDA 系列模块／仪表／变送器 检测到除了 CRC 码出错以外的错误时，则向主机回送信息，功能码的最高位置为 1，即从机返送给主机的功能码是在主机发送的功能码的基础上加 128 。以下的这些代码表明有意外的错误发生。

EDA 从主机接收到的信息如有 CRC 错误，则将被 EDA 从机忽略。

EDA 从机返送的错误码的格式如下（CRC 码除外）:

地址码: 1 字节

功能码: 1 字节（最高位为 1）

错误码: 1 字节

CRC 码: 2 字节。

EDA 响应回送如下错误码:

81: 非法的功能码。 接收到的功能码 EDA 模块不支持。

82: 读取非法的数据地址。 指定的数据位置超出 EDA 模块的可读取的地址范围。

83：非法的数据值。　接收到主机发送的数据值超出 EDA 模块相应地址的数据范围。

附件：CRC 校验算法程序（直接计算）

```
function CalcCRC16(str: string): Word;
    procedure CRC16(Data: Byte);
    var
        i: Integer;
    begin
        Result := Result xor Data;
        begin
            if ((Result and 1)=1) then
                Result := (Result shr 1) XOR $A001
            else
                Result := Result shr 1;
        end;
    end;
var
    i: Integer;
begin
    Result := $FFFF;
for i:=1 to Length(str) do
        CRC16(Byte(str[i]));
end;
```