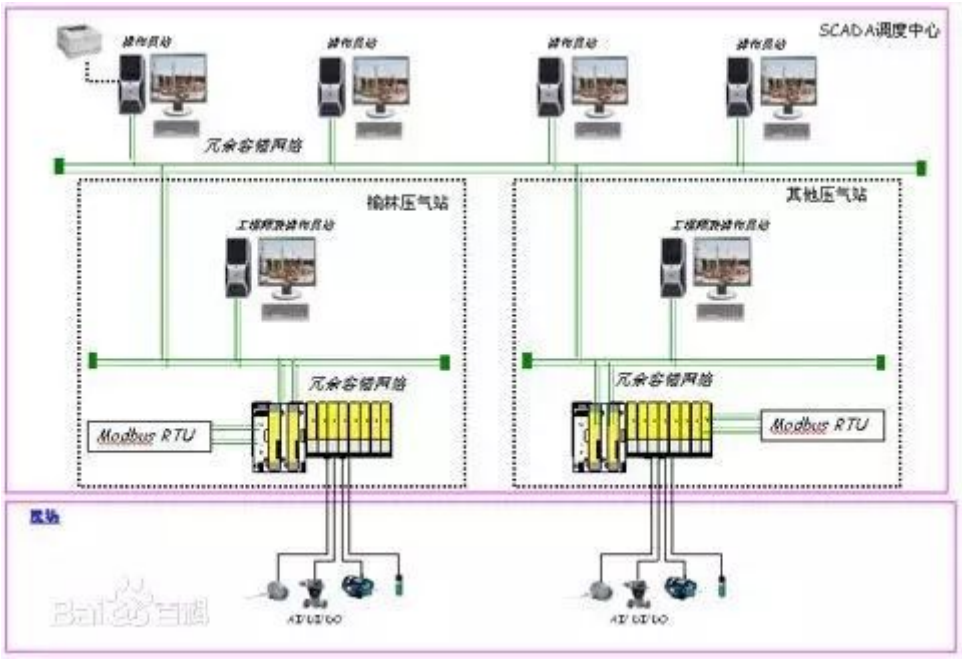
Modbus通訊協議是原來是這樣的!

愛上自動化 2022-04-06 10:45

ModBus網絡是一個工業通信系統，由帶智能終端的可編程序控制器和計算機通過公用線路或局部專用線路連接而成。其係統結構既包括硬件、亦包括軟件。它可應用於各種數據採集和過程監控。

ModBus網絡只有一個主機，所有通信都由他發出。網絡可支持247個之多的遠程從屬控制器，但實際所支持的從機數要由所用通信設備決定。採用這個系統，各PC可以和中心主機交換信息而不影響各PC執行本身的控制任務。

了解Modbus通訊協議是怎麼回事，在現場就可以用各種第三方的小軟件做通訊測試了。



Modbus協議包括ASCII、RTU、TCP等，並沒有規定物理層。此協議定義了控制器能夠認識和使用的消息結構，而不管它們是經過何種網絡進行通信的。標準的Modicon控制器使用RS232C實現串行的Modbus。Modbus的ASCII、RTU協議規定了消息、數據的結構、命令和就答的方式，數據

通訊採用Maser/Slave方式，Master端發出數據請求消息，Slave端接收到正確消息後就可以發送數據到Master端以響應請求；Master端也可以直接發消息修改Slave端的數據，實現雙向讀寫。

Modbus协议需要对数据进行校验，串行协议中除有奇偶校验外，ASCII模式采用LRC校验，RTU模式采用16位CRC校验，但TCP模式没有额外规定校验，因为TCP协议是一个面向连接的可靠协议。另外，Modbus采用主从方式定时收发数据，在实际使用中如果某Slave站点断开后（如故障或关机），Master端可以诊断出来，而当故障修复后，网络又可自动接通。因此，Modbus协议的可靠性较好。

对于Modbus的ASCII、RTU和TCP协议来说，其中TCP和RTU协议非常类似，我们只要把RTU协议的两个字节的校验码去掉，然后在RTU协议的开始加上5个0和一个6并通过TCP/IP网络协议发送出去即可。

1 通讯传送方式：

通讯传送分为独立的信息头，和发送的编码数据。以下的通讯传送方式定义也与ModBusRTU通讯规约相兼容：

初始结构 = ≥4字节的时间

地址码 = 1 字节

功能码 = 1 字节

数据区 = N 字节

错误校检 = 16位CRC码

结束结构 = ≥4字节的时间

地址码：地址码为通讯传送的第一个字节。这个字节表明由用户设定地址码的从机将接收由主机发送来的信息。并且每个从机都有具有唯一的地址码，并且响应回送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，而从机发送的地址码表明回送的从机地址。

功能码：通讯传送的第二个字节。ModBus通讯规约定义功能号为1到127。本仪表只利用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执行什么动作。作为从机响应，从机发送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进行操作。如果从机发送的功能码的最高位为 1 (比如功能码大与此同时127)，则表明从机没有响应操作或发送出错。

数据区：数据区是根据不同的功能码而不同。数据区可以是实际数值、设置点、主机发送给从机或从机发送给主机的地址。

CRC码：二字节的错误检测码。

2 通讯规约：

当通讯命令发送至仪器时，符合相应地址码的设备接通讯命令，并除去地址码，读取信息，如果没有出错，则执行相应的任务；然后把执行结果返送给发送者。返送的信息中包括地址码、执行动作的功能码、执行动作后结果的数据以及错误校验码。如果出错就不发送任何信息。

1．信息帧结构

地址码 功能码 数据区 错误校验码

8位 8位 N × 8位 16位

地址码：地址码是信息帧的第一字节(8位)，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。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。当从机回送信息时，相当的地址码表明该信息来自于何处。

功能码：主机发送的功能码告诉从机执行什么任务。表1-1列出的功能码都有具体的含义及操作。

数据区：数据区包含需要从机执行什么动作或由从机采集的返送信息。这些信息可以是数值、参考地址等等。例如，功能码告诉从机读取寄存器的值，则数据区必需包含要读取寄存器的起始地址及读取长度。对于不同的从机，地址和数据信息都不相同。

错误校验码：主机或从机可用校验码进行判别接收信息是否出错。有时，由于电子噪声或其它一些干扰，信息在传输过程中会发生细微的变化，错误校验码保证了主机或从机对在传送过程中出错的信息不起作用。这样增加了系统的安全和效率。错误校验采用CRC-16校验方法。

注：信息帧的格式都基本相同：地址码、功能码、数据区和错误校验码。

2．错误校验

冗余循环码（CRC）包含2个字节，即16位二进制。CRC码由发送设备计算，放置于发送信息的尾部。接收信息的设备再重新计算接收到信息的 CRC码，比较计算得到的CRC码是否与接收到的相符，如果两者不相符，则表明出错。

3 Modbus支持的功能码：

功能码	名称	作用
1	读取线圈状态	取得一组逻辑线圈的当前状态（ON/OFF）

2	读取输入状态	取得一组开关输入的当前状态（ON/OFF)
3	读取保持寄存器	在一个或多个保持寄存器中取得当前的二进制值
4	读取输入寄存器	在一个或多个输入寄存器中取得当前的二进制值
5	强置单线圈	强置一个逻辑线圈的通断状态
6	预置单寄存器	把具体二进值装入一个保持寄存器
7	读取异常状态	取得8个内部线圈的通断状态，这8个线圈的地址由控制器决定
8	回送诊断校验	把诊断校验报文送从机，以对通信处理进行评鉴
9	编程（只用于484）	使主机模拟编程器作用，修改PC从机逻辑
10	控询（只用于484）	可使主机与一台正在执行长程序任务从机通信，探询该从机是否已完成其操作任务，仅在含有功能码9的报文发送后，本功能码才发送
11	读取事件计数	可使主机发出单询问，并随即判定操作是否成功，尤其是该命令或其他应答产生通信错误时
12	读取通信事件记录	可是主机检索每台从机的ModBus事务处理通信事件记录。如果某项事务处理完成，记录会给出有关错误
13	编程（184/384 484 584)	可使主机模拟编程器功能修改PC从机逻辑
14	探询（184/384 484 584)	可使主机与正在执行任务的从机通信，定期控询该从机是否已完成其程序操作，仅在含有功能13的报文发送后，本功能码才得发送
15	强置多线圈	强置一串连续逻辑线圈的通断
16	预置多寄存器	把具体的二进制值装入一串连续的保持寄存器
17	报告从机标识	可使主机判断编址从机的类型及该从机运行指示灯的状态
18	（884和MICRO 84）	可使主机模拟编程功能，修改PC状态逻辑

19	重置通信链路	发生非可修改错误后，是从机复位于已知状态，可重置顺序字节
20	读取通用参数（584L）	显示扩展存储器文件中的数据信息
21	写入通用参数（584L）	把通用参数写入扩展存储文件，或修改之
22～64	保留作扩展功能备用	
65～72	保留以备用户功能所用	留作用户功能的扩展编码
73～119	非法功能	
120～127	保留	留作内部作用
128～255	保留	用于异常应答

4 功能码命令详解：

在这些功能码中较长使用的是1、2、3、4、5、6号功能码，使用它们即可实现对下位机的数字量和模拟量的读写操作。

1、01号命令，读可读写数字量寄存器（线圈状态）：

计算机发送命令：[设备地址] [命令号01] [起始寄存器地址高8位] [低8位] [读取的寄存器数高8位] [低8位] [CRC校验的低8位] [CRC校验的高8位]

例：[11][01][00][13][00][25][CRC低][CRC高]

意义如下：

<1>设备地址：在一个485总线上可以挂接多个设备，此处的设备地址表示想和哪一个设备通讯。例子中为想和17号(十进制的17是十六进制的11)通讯。

<2>命令号01：读取数字量的命令号固定为01。

<3>起始地址高8位、低8位：表示想读取的开关量的起始地址(起始地址为0)。比如例子中的起始地址为19。

<4>寄存器数高8位、低8位：表示从起始地址开始读多少个开关量。例子中为37个开关量。

<5>CRC校验：是从开头一直校验到此之前。

设备响应：[设备地址] [命令号01] [返回的字节个数][数据1][数据2]...[数据n] [CRC校验的高8位] [CRC校验的低8位]

例：[11][01][05][CD][6B][B2][0E][1B] [CRC高] [CRC低]

意义如下：

<1>设备地址和命令号和上面的相同。

<2>返回的字节个数：表示数据的字节个数，也就是数据1，2...n中的n的值。

<3>数据1...n：由于每一个数据是一个8位的数，所以每一个数据表示8个开关量的值，每一位为0表示对应的开关断开，为1表示闭合。比如例子中，表示20号(索引号为19)开关闭合，21号断开，22闭合，23闭合，24断开，25断开，26闭合，27闭合...如果询问的开关量不是8的整倍数，那么最后一个字节的高位部分无意义，置为0。

<4>CRC校验同上。

2、05号命令，写数字量（线圈状态）：

计算机发送命令：[设备地址] [命令号05] [需下置的寄存器地址高8位] [低8位] [下置的数据高8位] [低8位] [CRC校验的低8位] [CRC校验的高8位]

例：[11][05][00][AC][FF][00][CRC高][CRC低]

意义如下：

<1>设备地址和上面的相同。

<2>命令号:写数字量的命令号固定为05。

<3>需下置的寄存器地址高8位，低8位：表明了需要下置的开关的地址。

<4>下置的数据高8位，低8位：表明需要下置的开关量的状态。例子中为把该开关闭合。注意，此处只可以是[FF][00]表示闭合[00][00]表示断开，其他数值非法。

<5>注意此命令一条只能下置一个开关量的状态。

设备响应：如果成功把计算机发送的命令原样返回，否则不响应。

3、03号命令，读可读写模拟量寄存器（保持寄存器）：

计算机发送命令：[设备地址] [命令号03] [起始寄存器地址高8位] [低8位] [读取的寄存器数高8位] [低8位] [CRC校验的高8位] [CRC校验的低8位]

例：[11][03][00][6B][00][03] [CRC高][CRC低]

意义如下：

<1>设备地址和上面的相同。

<2>命令号:读模拟量的命令号固定为03。

<3>起始地址高8位、低8位：表示想读取的模拟量的起始地址(起始地址为0)。比如例子中的起始地址为107。

<4>寄存器数高8位、低8位：表示从起始地址开始读多少个模拟量。例子中为3个模拟量。注意，在返回的信息中一个模拟量需要返回两个字节。

设备响应：[设备地址] [命令号03] [返回的字节个数][数据1][数据2]...[数据n] [CRC校验的高8位] [CRC校验的低8位]

例：[11][03][06][02][2B][00][00][00][64] [CRC高] [CRC低]

意义如下：

<1>设备地址和命令号和上面的相同。

<2>返回的字节个数：表示数据的字节个数，也就是数据1，2...n中的n的值。例子中返回了3个模拟量的数据，因为一个模拟量需要2个字节所以共6个字节。

<3>数据1...n：其中[数据1][数据2]分别是第1个模拟量的高8位和低8位，[数据3][数据4]是第2个模拟量的高8位和低8位，以此类推。例子中返回的值分别是555，0，100。

<4>CRC校验同上。

4、06号命令，写单个模拟量寄存器（保持寄存器）：

计算机发送命令：[设备地址] [命令号06] [需下置的寄存器地址高8位] [低8位] [下置的数据高8位] [低8位] [CRC校验的高8位] [CRC校验的低8位]

例：[11][06][00][01][00][03] [CRC高] [CRC低]

意义如下：

<1>设备地址和上面的相同。

<2>命令号:写模拟量的命令号固定为06。

<3>需下置的寄存器地址高8位，低8位：表明了需要下置的模拟量寄存器的地址。

<4>下置的数据高8位，低8位：表明需要下置的模拟量数据。比如例子中就把1号寄存器的值设为3。

<5>注意此命令一条只能下置一个模拟量的状态。

设备响应：如果成功把计算机发送的命令原样返回，否则不响应。

5、16号命令，写多个模拟量寄存器（保持寄存器）：

计算机发送命令：[设备地址] [命令号16] [需下置的寄存器地址高8位] [低8位] [数据数量高8位] [数据数量低8位] [下置的数据高8位] [低8位][……][……] [CRC校验的高8位] [CRC校验的低8位]

例：[11][16][00][01][00][01][00][05] [CRC高] [CRC低]

意义如下：

<1>设备地址和上面的相同。

<2>命令号:写模拟量的命令号固定为16。

<3>需下置的寄存器地址高8位，低8位：表明了需要下置的模拟量寄存器的地址。

<4>需下置的数据数量高8位，低8位：表明了需要下置的数据数量，这里为1。

<5>下置的数据高8位，低8位：表明需要下置的模拟量数据。比如例子中就把1号寄存器的值设为5。

設備響應：如果成功把計算機返回的如下命令，否則不響應。

設備響應：[設備地址] [命令號16] [需下置的寄存器地址高8位] [低8位] [數據數量高8位] [數據數量低8位] [CRC校驗的高8位] [CRC校驗的低8位]，如上例返回：

[11][16][00][01][00][01] [CRC高] [CRC低]