

Modbus 通信协议

一、Modbus 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。

此协议定义了一个控制器能认识使用的消息结构,而不管它们是经过何种网络进行通信的。它描述了一控制器请求访问其它设备的过程，如果回应来自其它设备的请求，以及怎样侦测错误并记录。它制定了消息域格局和内容的公共格式。

当在一Modbus 网络上通信时，此协议决定了每个控制器须要知道它们的设备地址，识别按地址发来的消息，决定要产生何种行动。如果需要回应，控制器将生成反馈信息并用Modbus 协议发出。在其它网络上，包含了Modbus 协议的消息转换为在此网络上使用的帧或包结构。这种转换也扩展了根据具体的网络解决节地址、路由路径及错误检测的方法。

1、在Modbus 网络上转输

标准的Modbus 口是使用一RS-232C 兼容串行接口，它定义了连接口的针脚、电缆、信号位、传输波特率、奇偶校验。控制器能直接或经由Modem 组网。

控制器通信使用主-从技术，即仅一设备（主设备）能初始化传输（查询）。其它设备（从设备）根据主设备查询提供的数据作出相应反应。典型的主设备：主机和可编程仪表。典型的从设备：可编程控制器。

主设备可单独和从设备通信，也能以广播方式和所有从设备通信。如果单独通信，从设备返回一消息作为回应，如果是广播方式查询的，则不作任何回应。Modbus 协议建立了主设备查询的格式：设备（或广播）地址、功能代码、所有要发送的数据、一错误检测域。

从设备回应消息也由**Modbus** 协议构成，包括确认要行动的域、任何要返回的数据、和一错误检测域。如果在消息接收过程中发生一错误，或从设备不能执行其命令，从设备将建立一错误消息并把它作为回应发送出去。

2、在其它类型网络上转输

在其它网络上，控制器使用对等技术通信，故任何控制都能初始和其它控制器的通信。这样在单独的通信过程中，控制器既可作为主设备也可作为从设备。提供的多个内部通道可 允许同时发生的传输进程。

在消息位，**Modbus** 协议仍提供了主—从原则，尽管网络通信方法是“对等”。如果一控制器发送一消息，它只是作为主设备，并期望从从设备得到回应。同样，当控制器接收到一消息，它将建立一从设备回应格式并返回给发送的控制器。

3、查询—回应周期

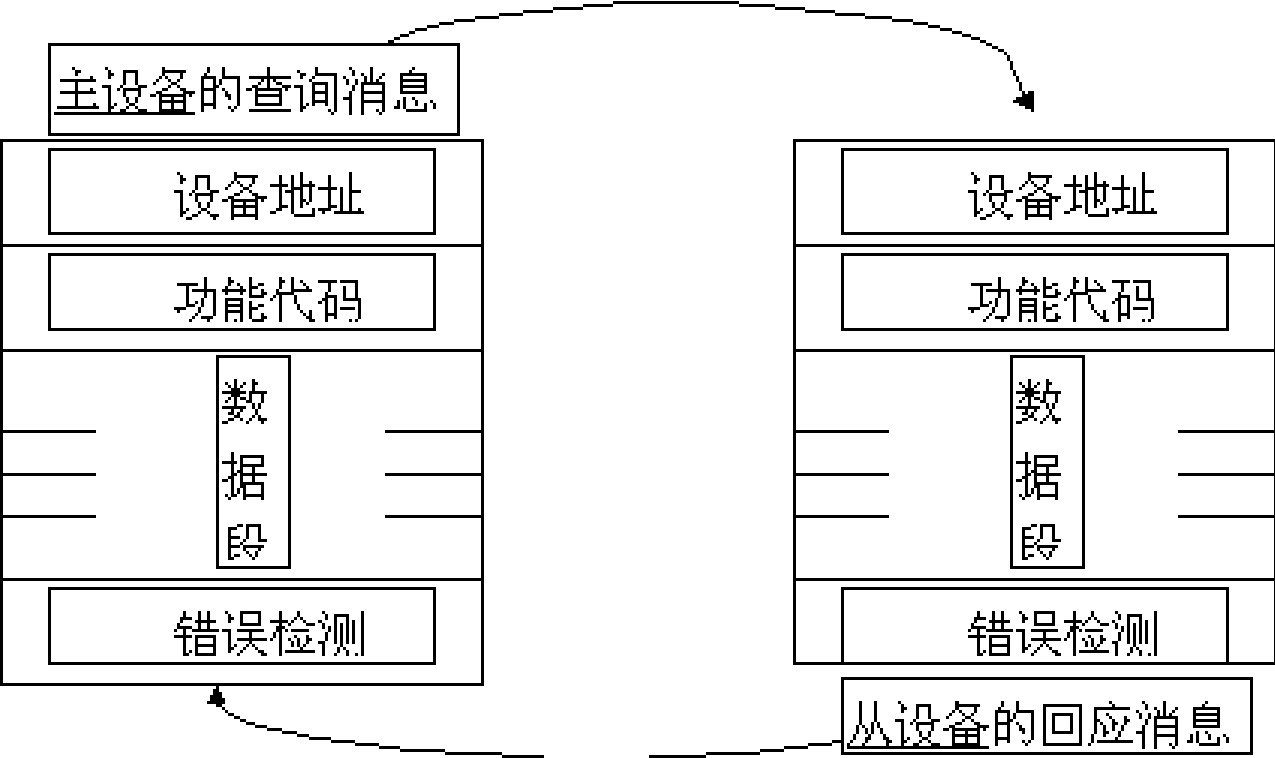


图 1 主—从 查询—回应周期表

(1) 查询

查询消息中的功能代码告之被选中的从设备要执行何种功能。数据段包含了从设备要 执行功能的任何附加信息。例如功能代码 **03** 是要求从设备读保持寄存器并返回它们的内容。

数据段必须包含要告之从设备的信息： 从何寄存器开始读及要读的寄存器数量。错误检测域为从设备提供了一种验证消息内容是否正确的方法。

(1) 回应

如果从设备产生一正常的回应，在回应消息中的功能代码是在查询消息中的功能代码的回应。数据段包括了从设备收集的数据： 象寄存器值或状态。如果有错误发生，功能代码将被修改以用于指出回应消息是错误的，同时数据段包含了描述此错误信息的代码。错误检测域允许主设备确认消息内容是否可用。

二、两种传输方式

控制器能设置为两种传输模式（ASCII 或 RTU）中的任何一种在标准的Modbus 网络通信。用户选择想要的模式，包括串口通信参数（波特率、校验方式等），在配置每个控制器的时候，在一个Modbus 网络上的所有设备都必须选择相同的传输模式和串口参数。

ASCII 模式

:	地址	功能代码	数据数量	数据 1	...	数据 n	LRC 高字节	LRC 低字节	回车	换行
---	----	------	------	------	-----	------	---------	---------	----	----

RTU 模式

地址	功能代码	数据数量	数据 1	...	数据 n	CRC 低字节	CRC 高字节
----	------	------	------	-----	------	---------	---------

所选的ASCII 或 RTU 方式仅适用于标准的Modbus 网络，它定义了在这些网络上连续传输的消息段的每一位，以及决定怎样将信息打包成消息域和如何解码。

在其它网络上（象 MAP 和 Modbus Plus）Modbus 消息被转成与串行传输无关的帧。

1、ASCII 模式

当控制器设为在Modbus 网络上以ASCII（美国标准信息交换代码）模式通信，在消息中的每个 8Bit 字节都作为两个ASCII 字符发送。这种方式的主要优点是字符发送的时间间隔可达到 1 秒而不产生错误。

代码系统

- 十六进制，ASCII 字符 0...9，A...F
- 消息中的每个ASCII 字符都是一个十六进制字符组成

每个字节的位

- 1 个起始位
- 7 个数据位，最小的有效位先发送
- 1 个奇偶校验位，无校验则无
- 1 个停止位（有校验时），2 个 Bit（无校验时）

错误检测域

- LRC(纵向冗长检测)

2、RTU 模式

当控制器设为在Modbus 网络上以RTU（远程终端单元）模式通信，在消息中的每个 8Bit 字节包含两个 4Bit 的十六进制字符。这种方式的主要优点是：在同样的波特率下，可比 ASCII 方式传送更多的数据。

代码系统

- 8 位二进制，十六进制数 0...9，A...F
- 消息中的每个 8 位域都是一个两个十六进制字符组成
- 每个字节的位
- 1 个起始位
- 8 个数据位，最小的有效位先发送
- 1 个奇偶校验位，无校验则无

- 1 个停止位（有校验时），2 个 Bit（无校验时）

错误检测域

- CRC(循环冗长检测)

三、Modbus 消息帧

两种传输模式中（ASCII 或 RTU），传输设备以将Modbus 消息转为有起点和终点的帧，这就允许接收的设备在消息起始处开始工作，读地址分配信息，判断哪一个设备被选中（广播方式则传给所有设备），判知何时信息已完成。部分的消息也能侦测到并且错误能设置为返回结果。

1、ASCII 帧

使用ASCII 模式，消息以冒号（:）字符（ASCII 码 3AH）开始，以回车换行符结束（ASCII 码 0DH,0AH）。

其它域可以使用的传输字符是十六进制的0...9,A...F。网络上的设备不断侦测“:”字符，当有一个冒号接收到时，每个设备都解码下个域（地址域）来判断是否发给自己的。

消息中字符间发送的时间间隔最长不能超过1 秒，否则接收的设备将认为传输错误。
一个典型消息帧如下所示：

起始位	设备地址	功能代码	数据	LRC 校验	结束符
1 个字符	2 个字符	2 个字符	n 个字符	2 个字符	2 个字符

图 2 ASCII 消息帧

2、RTU 帧

使用RTU 模式，消息发送至少要以 3.5 个字符时间的停顿间隔开始。在网络波特率下多样的字符时间，这是最容易实现的(如下图的 T1-T2-T3-T4 所示)。传输的第一个域是设备

地址。可以使用的传输字符是十六进制的0...9,A...F。网络设备不断侦测网络总线，包括停顿间隔时间内。当第一个域（地址域）接收到，每个设备都进行解码以判断是否发往自己的。在最后一个传输字符之后，一个至少3.5 个字符时间的停顿标定了消息的结束。一个新的消息可在此停顿后开始。

整个消息帧必须作为一连续的流转输。如果在帧完成之前有超过1.5 个字符时间的停顿时间，接收设备将刷新不完整的消息并假定下一字节是一个新消息的地址域。同样地，如果一个新消息在小于 3.5 个字符时间内接着前个消息开始，接收的设备将认为它是前一消息的延续。这将导致一个错误，因为在最后的CRC 域的值不可能是正确的。一典型的消息帧如下所示：

起始位	设备地址	功能代码	数据	CRC 校验	结束符
T1-T2-T3-T4	8Bit	8Bit	n 个 8Bit	16Bit	T1-T2-T3-T4

图 3 RTU 消息帧

3、地址域

消息帧的地址域包含两个字符（ASCII）或 8Bit（RTU）。可能的从设备地址是 0...247 (十进制)。单个设备的地址范围是1...247。主设备通过将要联络的从设备的地址放入消息中的地址域来选通从设备。当从设备发送回应消息时，它把自己的地址放入回应的地址域中，以便主设备知道是哪一个设备作出回应。

地址 0 是用作广播地址，以使所有的从设备都能认识。当Modbus 协议用于更高水准的网络，广播可能不允许或以其它方式代替。

4、如何处理功能域

消息帧中的功能代码域包含了两个字符（ASCII）或 8Bits（RTU）。可能的代码范围是十进制的 1...255。当然，有些代码是适用于所有控制器，有此是应用于某种控制器，还有些保留以备后用。

当消息从主设备发往从设备时，功能代码域将告之从设备需要执行哪些行为。例如去读取输入的开关状态，读一组寄存器的数据内容，读从设备的诊断状态，允许调入、记录、校验在从设备中的程序等。

当从设备回应时，它使用功能代码域来指示是正常回应(无误)还是有某种错误发生（称作异议回应）。对正常回应，从设备仅回应相应的功能代码。对异议回应，从设备返回一等同于正常代码的代码，但最重要的位置为逻辑1。

例如：一从主设备发往从设备的消息要求读一组保持寄存器，将产生如下功能代码：

0 0 0 0 0 0 1 1 （十六进制 03H）

对正常回应，从设备仅回应同样的功能代码。对异议回应，它返回：

1 0 0 0 0 0 1 1 （十六进制 83H）

除功能代码因异议错误作了修改外，从设备将一独特的代码放到回应消息的数据域中，这能告诉主设备发生了什么错误。

主设备应用程序得到异议的回应后，典型的处理过程是重发消息，或者诊断发给从设备的消息并报告给操作员。

5、数据域

数据域是由两个十六进制数集合构成的，范围00...FF。根据网络传输模式，这可以由一对ASCII 字符组成或由一RTU 字符组成。

从主设备发给从设备消息的数据域包含附加的信息：从设备必须用于进行执行由功能代码所定义的所为。这包括了象不连续的寄存器地址，要处理项的数目，域中实际数据字节数。

例如，如果主设备需要从设备读取一组保持寄存器（功能代码03），数据域指定了起始寄存器以及要读的寄存器数量。如果主设备写一组从设备的寄存器（功能代码10 十六进制），数据域则指明了要写的起始寄存器以及要写的寄存器数量，数据域的数据字节数，要写入寄存器的数据。

如果没有错误发生，从从设备返回的数据域包含请求的数据。如果有错误发生，此域包含一异议代码，主设备应用程序可以用来判断采取下一步行动。

在某种消息中数据域可以是不存在的（0 长度）。例如，主设备要求从设备回应通信事件记录（功能代码 0B 十六进制），从设备不需任何附加的信息。

6、错误检测域

标准的Modbus 网络有两种错误检测方法。错误检测域的内容视所选的检测方法而定。

ASCII

当选用ASCII 模式作字符帧，错误检测域包含两个ASCII 字符。这是使用LRC（纵向冗长检测）方法对消息内容计算得出的，不包括开始的冒号符及回车换行符。LRC 字符附加在回车换行符前面。

RTU

当选用RTU 模式作字符帧，错误检测域包含一16Bits 值(用两个 8 位的字符来实现)。错误检测域的内容是通过对消息内容进行循环冗长检测方法得出的。CRC 域附加在消息的最后，添加时先是低字节然后是高字节。故CRC 的高位字节是发送消息的最后一个字节。

7、字符的连续传输

当消息在标准的Modbus 系列网络传输时，每个字符或字节以如下方式发送（从左到右）：

最低有效位...最高有效位

使用ASCII 字符帧时，位的序列是：

有奇偶校验

起始位	1	2	3	4	5	6	7	奇偶位	停止位
-----	---	---	---	---	---	---	---	-----	-----

无奇偶校验

起始位	1	2	3	4	5	6	7	停止位	停止位
-----	---	---	---	---	---	---	---	-----	-----

图 4. 位顺序（ASCII）

使用RTU 字符帧时，位的序列是：

有奇偶校验

起始位	1	2	3	4	5	6	7	8	奇偶位	停止位
-----	---	---	---	---	---	---	---	---	-----	-----

无奇偶校验

起始位	1	2	3	4	5	6	7	8	停止位	停止位
-----	---	---	---	---	---	---	---	---	-----	-----

图 4. 位顺序（RTU）

四、错误检测方法

标准的Modbus 串行网络采用两种错误检测方法。奇偶校验对每个字符都可用，帧检测（LRC 或 CRC）应用于整个消息。它们都是在消息发送前由主设备产生的，从设备在接收过程中检测每个字符和整个消息帧。

用户要给主设备配置一预先定义的超时时间间隔，这个时间间隔要足够长，以使任何从设备都能作为正常反应。如果从设备测到一传输错误，消息将不会接收，也不会向主设备

作出回应。这样超时事件将触发主设备来处理错误。发往不存在的从设备的地址也会产生超时。

1、奇偶校验

用户可以配置控制器是奇或偶校验，或无校验。这将决定了每个字符中的奇偶校验位是如何设置的。

如果指定了奇或偶校验，“1”的位数将算到每个字符的位数中（ASCII 模式 7 个数据位，RTU 中 8 个数据位）。例如RTU 字符帧中包含以下 8 个数据位：

1 1 0 0 0 1 0 1

整个“1”的数目是 4 个。如果便用了偶校验，帧的奇偶校验位将是0，使得整个“1”的个数仍是 4 个。如果便用了奇校验，帧的奇偶校验位将是1，使得整个“1”的个数是 5 个。

如果没有指定奇偶校验位，传输时就没有校验位，也不进行校验检测。代替一附加的停止位填充至要传输的字符帧中。

2、LRC 检测

使用ASCII 模式，消息包括了一基于LRC 方法的错误检测域。LRC 域检测了消息域中除开始的冒号及结束的回车换行号外的内容。

LRC 域是一个包含一个 8 位二进制值的字节。LRC 值由传输设备来计算并放到消息帧中，接收设备在接收消息的过程中计算LRC，并将它和接收到消息中 LRC 域中的值比较，如果两值不等，说明有错误。

LRC 方法是将消息中的 8Bit 的字节连续累加，丢弃了进位。

LRC 简单函数如下：

```

static unsigned char LRC(auchMsg,usDataLen) unsigned
char *auchMsg ; /* 要进行计算的消息 */
unsigned short usDataLen ; /* LRC 要处理的字节的数量*/
{unsigned char uchLRC=0; /*LRC 字节初始化 */ while
(usDataLen--)/ * 传送消息 */
uchLRC += *auchMsg++ ; /* 累加*/ return
((unsigned char)(-((char_uchLRC))));
}

```

3、CRC 检测

使用 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。

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。

CRC 是先调入一值是全“1”的 16 位寄存器，然后调用一过程将消息中连续的8 位字节各当前寄存器中的值进行处理。仅每个字符中的8Bit 数据对 CRC 有效，起始位和停止位以及奇偶校验位均无效。

CRC 产生过程中，每个 8 位字符都单独和寄存器内容相或（OR），结果向最低有效位方向移动，最高有效位以 0 填充。LSB 被提取出来检测，如果 LSB 为 1，寄存器单独和预置的值或一下，如果 LSB 为 0，则不进行。整个过程要重复 8 次。在最后一位（第 8 位）完成后，下一个 8 位字节又单独和寄存器的当前值相或。最终寄存器中的值，是消息中所有的字节都执行之后的 CRC 值。

CRC 添加到消息中时，低字节先加入，然后高字节。

CRC 简单函数如下：

```

unsigned short CRC16(puchMsg, usDataLen)
unsigned char*puchMsg; /* 要进行 CRC校验的消息 */ unsigned
short usDataLen ; /* 消息中字节数 */

```


/* CRC 低位字节值表*/

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
};

RS-232、RS-422 与 RS-485 标准及应用

一、RS-232、RS-422 与 RS-485 的由来

RS-232、RS-422 与 RS-485 都是串行数据接口标准，最初都是由电子工业协会（EIA）制订并发布的，RS-232 在 1962 年发布，命名为 EIA-232-E，作为工业标准，以保证不同厂家产品之间的兼容。RS-422 由 RS-232 发展而来，它是为弥补 RS-232 之不足而提出的。为改进 RS-232 通信距离短、速率低的缺点，RS-422 定义了一种平衡通信接口，将传输速率提高到 10Mb/s，传输距离延长到 4000 英尺（速率低于 100kb/s 时），并允许在一条平衡总线上连接最多 10 个接收器。RS-422 是一种单机发送、多机接收的单向、平衡传输规范，被命名为 TIA/EIA-422-A 标准。为扩展应用范围，EIA 又于 1983 年在 RS-422 基础上制定了 RS-485 标准，增加了多点、双向通信能力，即允许多个发送器连接到同一条总线上，同时增加了发送器的驱动能力和冲突保护特性，扩展了总线共模范围，后命名为 TIA/EIA-485-A 标准。由于 EIA 提出的建议标准都是以“RS”作为前缀，所以在通讯工业领域，仍然习惯将上述标准以 RS 作前缀称谓。

RS-232、RS-422 与 RS-485 标准只对接口的电气特性做出规定，而不涉及接插件、电缆或协议，在此基础上用户可以建立自己的高层通信协议。因此在视频界的应用，许多厂家都建立了一套高层通信协议，或公开或厂家独家使用。如录像机厂家中的 Sony 与松下对录像机的 RS-422 控制协议是有差异的，视频服务器上的控制协议则更多了，如 Louth、Odetis 协议是公开的，而 ProLINK 则是基于 Profile 上的。

二、RS-232 串行接口标准

目前 RS-232 是 PC 机与通信工业中应用最广泛的一种串行接口。RS-232 被定义为一种在低速率串行通讯中增加通讯距离的单端标准。RS-232 采取不平衡传输方式，即所谓单端通讯

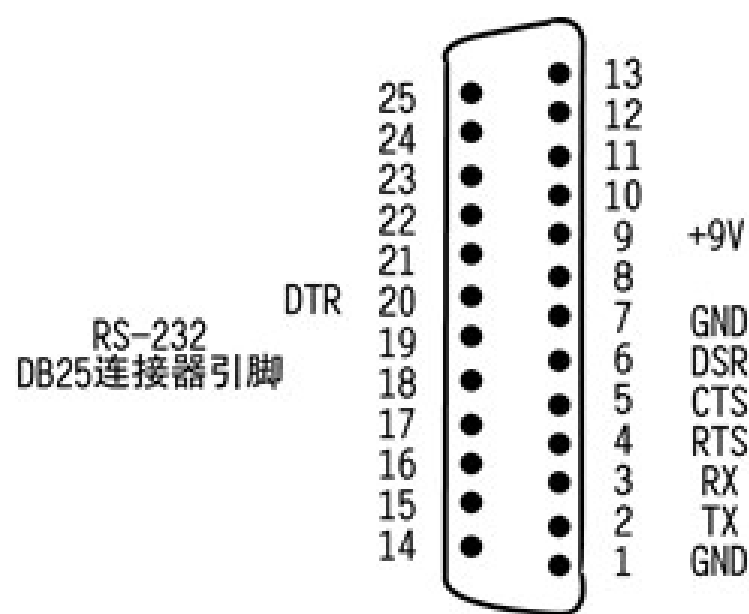


图 1

收、发端的数据信号是相对于信号地，如从 DTE 设备发出的数据在使用DB25 连接器时是 2 脚相对 7 脚（信号地）的电平，DB25 各引脚定义参见图 1。典型的RS-232 信号在正负电平之间摆动，在发送数据时，发送端驱动器输出正电平在+5~+15V，负电平在-5~-15V 电平。当无数据传输时，线上为TTL，从开始传送数据到结束，线上电平从TTL 电平到RS-232电平再返回TTL 电平。接收器典型的工作电平在+3~+12V 与-3~-12V。由于发送电平与接收电平的差仅为 2V 至 3V 左右，所以其共模抑制能力差，再加上双绞线上的分布电容，其传送距离最大为约 15 米，最高速率为 20kb/s。RS-232 是为点对点（即只用一对收、发设备） 通讯而设计的，其驱动器负载为 3~7kΩ。所以 RS-232 适合本地设备之间的通信。其有关电气参数参见表 1。

规定		RS232	RS422	R485
工作方式		单端	差分	差分
节点数		1 收、1 发	1 发 10 收	1 发 32 收
最大传输电缆长度		50 英尺	400 英尺	400 英尺
最大传输速率		20Kb/S	10Mb/s	10Mb/s
最大驱动输出电压		+/-25V	-0.25V~+6V	-7V~+12V
驱动器输出信号电平 (负载最小值)	负载	+/-5V~+/-15V	+/-2.0V	+/-1.5V
驱动器输出信号电平 (空载最大值)	空载	+/-25V	+/-6V	+/-6V
驱动器负载阻抗(Ω)		3K~7K	100	54
摆率(最大值)		30V/μs	N/A	N/A
接收器输入电压范围		+/-15V	-10V~+10V	-7V~+12V

接收器输入门限	+/-3V	+/-200mV	+/-200mV
接收器输入电阻(Ω)	3K~7K	4K(最小)	$\geq 12K$
驱动器共模电压		-3V~+3V	-1V~+3V
接收器共模电压		-7V~+7V	-7V~+12V

表 1

三、RS-422 与 RS-485 串行接口标准

1. 平衡传输

RS-422、RS-485 与 RS-232 不一样，数据信号采用差分传输方式，也称作平衡传输，它使用一对双绞线，将其中一线定义为A，另一线定义为B，如图 2。

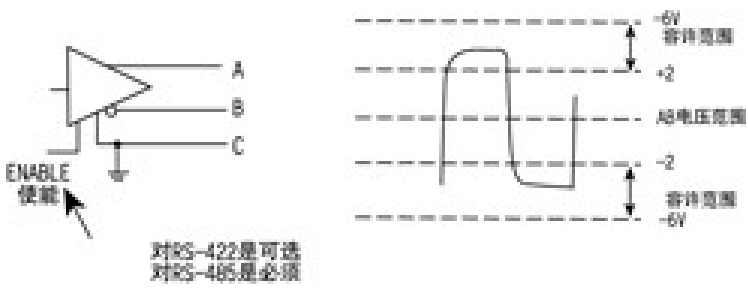


图 2

通常情况下，发送驱动器A、B 之间的正电平在+2~+6V，是一个逻辑状态，负电平在-2~6V，是另一个逻辑状态。另有一个信号地C，在RS-485 中还有一“使能”端，而在RS-422 中这是可用可不用的。“使能”端是用于控制发送驱动器与传输线的切断与连接。当“使能”端起作用时，发送驱动器处于高阻状态，称作“第三态”，即它是有别于逻辑“1”与“0”的第三态。

接收器也作与发送端相对的规定，收、发端通过平衡双绞线将AA 与 BB 对应相连，当在收端AB 之间有大于+200mV 的电平时，输出正逻辑电平，小于-200mV 时，输出负逻辑电平。接收器接收平衡线上的电平范围通常在200mV 至 6V 之间。参见图 3。

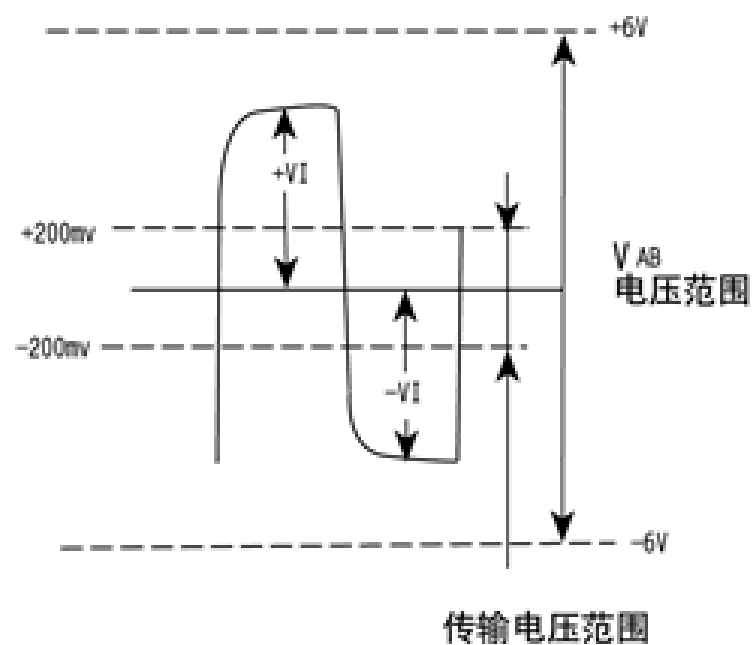


图 3

2. RS-422 电气规定

RS-422 标准全称是“平衡电压数字接口电路的电气特性”，它定义了接口电路的特性。

图 5 是典型的RS-422 四线接口。实际上还有一根信号地线，共 5 根线。图 4 是其DB9 连接器引脚定义。由于接收器采用高输入阻抗和发送驱动器比RS232 更强的驱动能力，故允许在相同传输线上连接多个接收节点，最多可接10 个节点。即一个主设备（Master），其余为从设备（Salve），从设备之间不能通信，所以RS-422 支持点对多的双向通信。接收器输入阻抗为 4k，故发端最大负载能力是 $10 \times 4k + 100\Omega$ （终接电阻）。RS-422 四线接口由于采用单独的发送和接收通道，因此不必控制数据方向，各装置之间任何必须的信号交换均可以 按软件方式（XON/XOFF 握手）或硬件方式（一对单独的双绞线）。

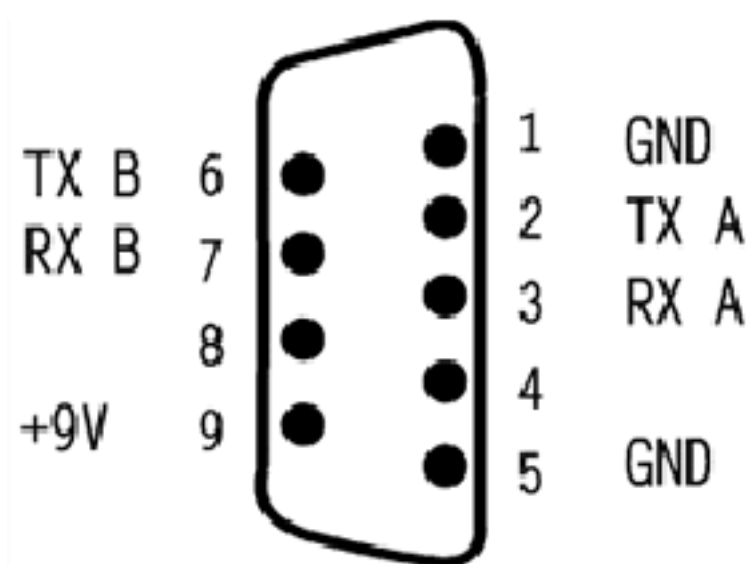


图 4

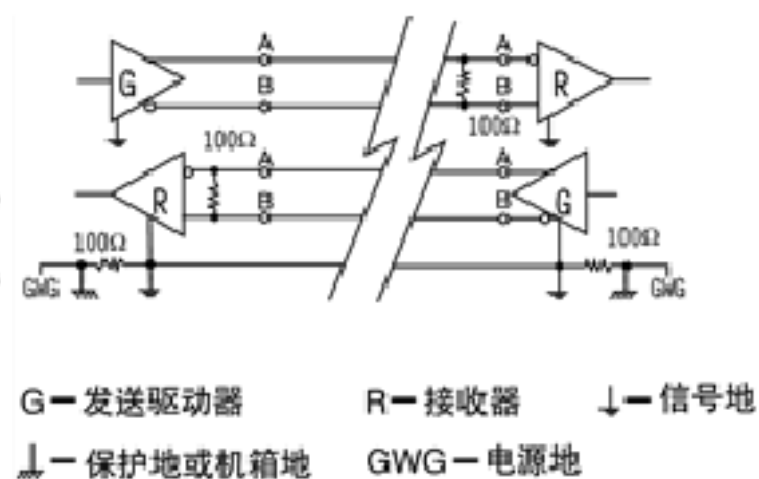


图 5

RS-422 的最大传输距离为 4000 英尺（约 1219 米），最大传输速率为 10Mb/s。其平衡双绞线的长度与传输速率成反比，在100kb/s 速率以下，才可能达到最大传输距离。只有

在很短的距离下才能获得最高速率传输。一般 100 米长的双绞线上所能获得的最大传输速率仅为 1Mb/s。

RS-422 需要一终接电阻，要求其阻值约等于传输电缆的特性阻抗。在短距离传输时可不需终接电阻，即一般在 300 米以下不需终接电阻。终接电阻接在传输电缆的最远端。

RS-422 有关电气参数见表 1

1. RS-485 电气规定

由于RS-485 是从RS-422 基础上发展而来的，所以RS-485 许多电气规定与RS-422 相仿。如都采用平衡传输方式、都需要在传输线上接终接电阻等。RS-485 可以采用二线与四线方式，二线制可实现真正的多点双向通信，参见图6。

而采用四线连接时，与 RS-422 一样只能实现点对多的通信，即只能有一个主（Master）设备，其余为从设备，但它比RS-422 有改进， 无论四线还是二线连接方式总线上可多接到 32 个设备。参见图 7。

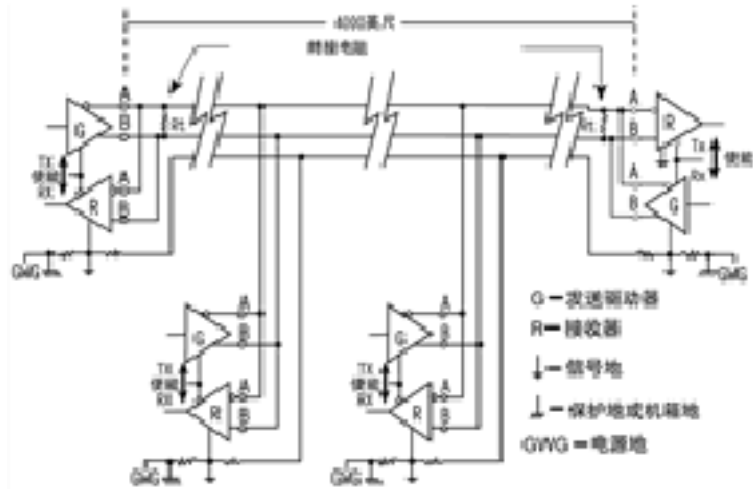


图 6

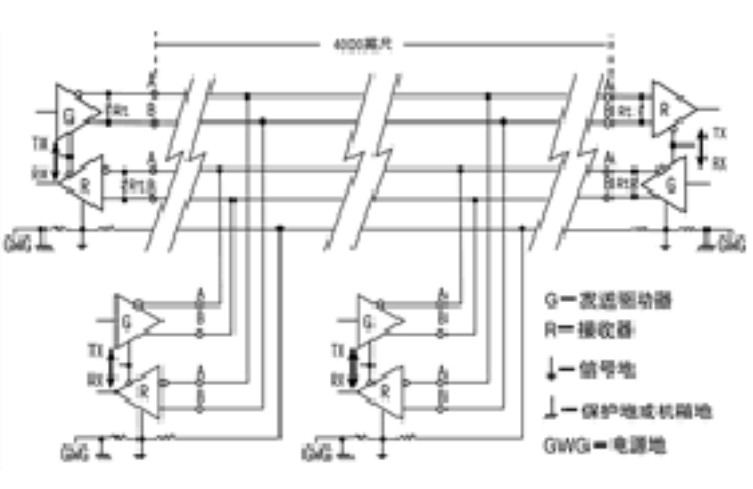


图 7

RS-485 与 RS-422 的不同还在于其共模输出电压是不同的，RS-485 是-7V 至+12V 之间，而 RS-422 在-7V 至+7V 之间，RS-485 接收器最小输入阻抗为 12k 欧，RS-422 是 4k 欧； 旧型号运放 S-485 满足所有 RS-422 的规范，所以 RS-485 的驱动器可以用在 RS-422 网络中应用。

RS-485 有关电气规定参见表 1。

RS-485 与 RS-422 一样，其最大传输距离约为1219 米，最大传输速率为10Mb/s。平衡双绞线的长度与传输速率成反比，在100kb/s 速率以下，才可能使用规定最长的电缆长度。只有在很短的距离下才能获得最高速率传输。一般100 米长双绞线最大传输速率仅为1Mb/s。

