目 次

RS-232-C 详解

串行通信接口标准经过使用和发展，目前已经有几种。但都是在 RS-232 标准的基础上经过改进而形成的。所以，以 RS-232C 为主来讨论。RS-323C 标准是美国 EIA(电子工业联合会）与 BELL 等公司一起开发的 1969 年公布的通信协议。它适合于数据传输速率在 0～20000b/s 范围内的通信。这个标准对串行通信接口的有关问题，如信号线功能、电器特性都作了明确规定。由于通行设备厂商都生产与 RS-232C 制式兼容的通信设备，因此，它作为一种标准，目前已在微机通信接口中广泛采用。

在讨论 RS-232C 接口标准的内容之前，先说明两点：

首先，RS-232-C 标准最初是远程通信连接数据终端设备 DTE(Data Terminal Equipment)与数据通信设备 DCE（Data Communication Equipment)而制定的。因此这个标准的制定，并未考虑计算机系统的应用要求。但目前它又广泛地被借来用于计算机（更准确的说，是计算机接口）与终端或外设之间的近端连接标准。显然，这个标准的有些规定及和计算机系统是不一致的，甚至是相矛盾的。有了对这种背景的了解，我们对 RS-232C 标准与计算机不兼容的地方就不难理解了。

其次，RS-232C 标准中所提到的“发送”和“接收”，都是站在 DTE 立场上，而不是站在 DCE 的立场来定义的。由于在计算机系统中，往往是 CPU 和 I/O 设备之间传送信息，两者都是 DTE，因此双方都能发送和接收。

一、RS-232-C

RS-232C 标准（协议）的全称是 EIA-RS-232C 标准，其中 EIA(Electronic Industry Association)代表美国电子工业协会，RS（ecommeded standard）代表推荐标准，232 是标识号，C 代表 RS232 的最新一次修改（1969），在这之前，有 RS232B、RS232A。。它规定连接电缆和机械、电气特性、信号功能及传送过程。常用物理标准还有有 EIA&#0;RS-232-C、EIA&#0;RS-422-A、EIA&#0;RS-423A、EIA&#0;RS-485。 这里只介绍 EIA&#0;RS-232-C（简称 232，RS232）。 例如，目前在 IBM PC 机上的 COM1、COM2 接口，就是 RS-232C 接口。

1.电气特性

EIA-RS-232C 对电器特性、逻辑电平和各种信号线功能都作了规定。

在 TxD 和 RxD 上：逻辑 1(MARK)=-3V～-15V
逻辑 0(SPACE)=+3～＋15V
在 RTS、CTS、DSR、DTR 和 DCD 等控制线上：
信号有效（接通，ON 状态，正电压）＝+3V～+15V
信号无效（断开，OFF 状态，负电压)=-3V～-15V

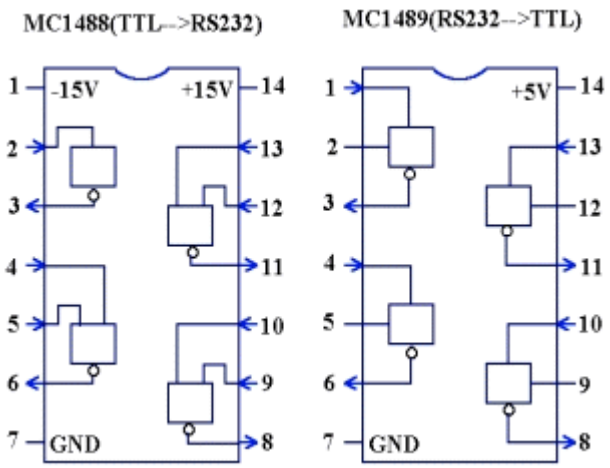


图 1

以上规定说明了 RS-323C 标准对逻辑电平的定义。对于数据（信息码）：逻辑“1”（传号）的电平低于-3V，逻辑“0”（空号）的电平告语+3V；对于控制信号；接通状态（ON）即信号有效的电平高于+3V，断开状态(OFF)即信号无效的电平低于-3V，也就是当传输电平的绝对值大于 3V 时，电路可以有效地检查出来，介于-3～+3V 之间的电压无意义，低于-15V 或高于+15V 的电压也认为无意义，因此，实际工作时，应保证电平在±(3～15)V 之间。

EIA-RS-232C 与 TTL 转换：EIA-RS-232C 是用正负电压来表示逻辑状态，与 TTL 以高低电平表示逻辑状态的规定不同。因此，为了能够同计算机接口或终端的 TTL 器件连接，必须在 EIA-RS-232C 与 TTL 电路之间进行电平和逻辑关系的变换。实现这种变换的方法可用分立元件，也可用集成电路芯片。目前较为广泛地使用集成电路转换器件，如 MC1488、SN75150 芯片可完成 TTL 电平到 EIA 电平的转换，而 MC1489、SN75154 可实现 EIA 电平到 TTL 电平的转换。MAX232 芯片可完成 TTL←→EIA 双向电平转换，图 1 显示了 1488 和 1489 的内部结构和引脚。MC1488 的引脚(2)、(4,5)、(9,10)和(12,13)接 TTL 输入。引脚 3、6、8、11 输出端接 EIA-RS-232C。MC1498 的 14 的 1、4、10、13 脚接 EIA 输入，而 3、6、8、11 脚接 TTL 输出。具体连接方法如图 2 所示。图中的左边是微机串行接口电路中的主芯片 UART，它是 TTL 器件，右边是 EIA-RS-232C 连接器，要求 EIA 高电压。因此，RS-232C 所有的输出、输入信号都要分别经过 MC1488 和 MC1498 转换器，进行电平转换后才能送到连接器上去或从连接器上送进来。

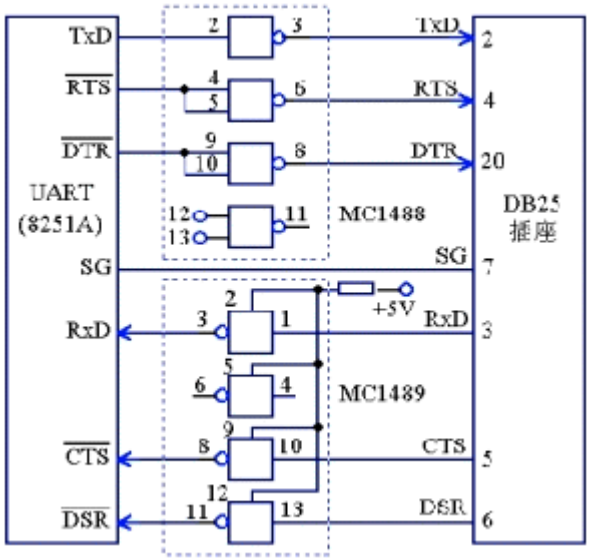


图 2

2、、连接器的机械特性：

连接器：由于 RS-232C 并未定义连接器的物理特性，因此，出现了 DB-25、DB-15 和 DB-9 各种类型的连接器，其引脚的定义也各不相同。下面分别介绍两种连接器。

（1）DB-25： PC 和 XT 机采用 DB-25 型连接器。DB-25 连接器定义了 25 根信号线，分为 4 组：

①异步通信的 9 个电压信号（含信号地 SG）2，3，4，5，6，7，8，20，22
②20mA 电流环信号 9 个（12，13，14，15，16，17，19,23，24）
③空 6 个（9，10，11，18，21，25）
④保护地（PE）1 个，作为设备接地端（1 脚）

DB-25 型连接器的外形及信号线分配如图 3 所示。注意，20mA 电流环信号仅 IBM PC 和 IBM PC/XT 机提供，至 AT 机及以后，已不支持。

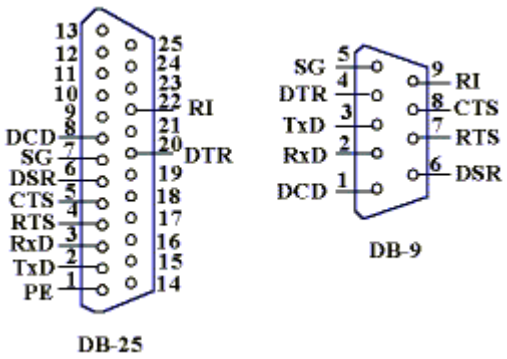


图 3

（2）DB-9 连接器

在 AT 机及以后，不支持 20mA 电流环接口，使用 DB-9 连接器，作为提供多功能 I/O 卡或主板上 COM1 和 COM2 两个串行接口的连接器。它只提供异步通信的 9 个信号。DB-25 型连

接器的引脚分配与 DB-25 型引脚信号完全不同。因此，若与配接 DB-25 型连接器的 DCE 设备连接，必须使用专门的电缆线。

电缆长度：在通信速率低于 20kb/s 时，RS-232C 所直接连接的最大物理距离为 15m（50 英尺）。

最大直接传输距离说明：RS-232C 标准规定，若不使用 MODEM，在码元畸变小于 4%的情况下，DTE 和 DCE 之间最大传输距离为 15m（50 英尺）。可见这个最大的距离是在码元畸变小于 4%的前提下给出的。为了保证码元畸变小于 4%的要求，接口标准在电气特性中规定，驱动器的负载电容应小于 2500pF。

3、RS-232C 的接口信号

RS-232C 规标准接口有 25 条线，4 条数据线、11 条控制线、3 条定时线、7 条备用和未定义线，常用的只有 9 根，它们是：

（1）联络控制信号线：

数据装置准备好（Data set ready-DSR)——有效时（ON）状态，表明 MODEM 处于可以使用的状态。

数据终端准备好(Data set ready-DTR)——有效时（ON）状态，表明数据终端可以使用。

这两个信号有时连到电源上，一上电就立即有效。这两个设备状态信号有效，只表示设备本身可用，并不说明通信链路可以开始进行通信了，能否开始进行通信要由下面的控制信号决定。

请求发送(Request to send-RTS)——用来表示 DTE 请求 DCE 发送数据，即当终端要发送数据时，使该信号有效（ON 状态），向 MODEM 请求发送。它用来控制 MODEM 是否要进入发送状态。

允许发送（Clear to send-CTS）——用来表示 DCE 准备好接收 DTE 发来的数据，是对请求发送信号 RTS 的响应信号。当 MODEM 已准备好接收终端传来的数据，并向前发送时，使该信号有效，通知终端开始沿发送数据线 TxD 发送数据。

这对 RTS/CTS 请求应答联络信号是用于半双工 MODEM 系统中发送方式和接收方式之间的切换。在全双工系统中作发送方式和接收方式之间的切换。在全双工系统中，因配置双向通道，故不需要 RTS/CTS 联络信号，使其变高。

接收线信号检出(Received Line detection-RLSD)——用来表示 DCE 已接通通信链路，告知 DTE 准备接收数据。当本地的 MODEM 收到由通信链路另一端（远地）的 MODEM 送来的载波信号时，使 RLSD 信号有效，通知终端准备接收，并且由 MODEM 将接收下来的载波信号解调成数字两数据后，沿接收数据线 RxD 送到终端。此线也叫做数据载波检出(Data Carrier dectection-DCD）线。

振铃指示(Ringing-RI)——当 MODEM 收到交换台送来的振铃呼叫信号时，使该信号有效（ON 状态），通知终端，已被呼叫。

（2）数据发送与接收线：

发送数据(Transmitted data-TxD)——通过 TxD 终端将串行数据发送到 MODEM，(DTE→DCE)。
接收数据(Received data-RxD)——通过 RxD 线终端接收从 MODEM 发来的串行数据，(DCE→DTE)。

（3）地线

有两根线 SG、PG——信号地和保护地信号线，无方向。

上述控制信号线何时有效，何时无效的顺序表示了接口信号的传送过程。例如，只有当 DSR 和 DTR 都处于有效（ON）状态时，才能在 DTE 和 DCE 之间进行传送操作。若 DTE 要发送数据，则预先将 DTR 线置成有效(ON)状态，等 CTS 线上收到有效(ON)状态的回答后，才能在 TxD 线上发送串行数据。这种顺序的规定对半双工的通信线路特别有用，因为半双工的通信才能确定 DCE 已由接收方向改为发送方向，这时线路才能开始发送。

2 个数据信号：发送 TXD；接收 RXD。
1 个信号地线：SG。
6 个控制信号：
DSR&#0;&#0;数传机（即 modem）准备好，Data Set Ready.
DTR&#0;&#0;数据终端（DTE，即微机接口电路，如 Intel8250/8251,16550）准备好，Data Terminal Ready。
RTS&#0;&#0;DTE 请求 DCE 发送(Request To Send)。
CTS&#0;&#0;DCE 允许 DTE 发送（Clear To Send）,该信号是对 RTS 信号的回答。
DCD&#0;&#0;数据载波检出，Data Carrier Detection 当本地 DCE 设备（Modem）收到对方的 DCE 设备送来的载波信号时，使 DCD 有效，通知 DTE 准备接收， 并且由 DCE 将接收到的载波信号解调为数字信号，经 RXD 线送给 DTE。
RI&#0;&#0;振铃信号 Ringing 当 DCE 收到交换机送来的振铃呼叫信号时，使该信号有效，通知 DTE 已被呼叫。

232 引脚	CCITT	Modem	名称	说明	用途	
					异步	同步
1	101	AA	保护地	设备外壳接地	PE	PE √
2	103	BA	发送数据	数据送 Modem	TXD	
3	104	BB	接收数据	从 Modem 接收数据	RXD	
4	105	CA	请求发送	在半双工时控制发送器的开和关	RTS	
5	106	CB	允许发送	Modem 允许发送	CTS	
6	107	CC	数据终端准备好	Modem 准备好	DSR	
7	102	AB	信号地	信号公共地	SG	SG √

8	109	CF	载波信号检测	Modem 正在接收另一端送来的信号	DCD	
9			空			
10			空			
11			空			
12			接收信号检测（2）	在第二通道检测到信号		√
13			允许发送（2）	第二通道允许发送		√
14	118		发送数据（2）	第二通道发送数据		√
15	113	DA	发送器定时	为 Modem 提供发送器定时信号		√
16	119		接收数据（2）	第二通道接收数据		√
17	115	DD	接收器定时	为接口和终端提供定时		√
18			空			
19			请求发送（2）	连接第二通道的发送器		√
20	108	CD	数据终端准备好	数据终端准备好	DTR	
21			空			
22	125		振铃	振铃指示	RI	
23	111	CH	数据率选择	选择两个同步数据率		√
24	114	DB	发送器定时	为接口和终端提供定时		√
25			空			

PART2

一、远距离通信

第 1 和第 2 中情况是属于远距离通信（传输距离大于 15m 的通信）的例子，故一般要加调制解调器 MODEM，因此使用的信号线较多。注意：在以下各图中，DTE 信号为 RS-232-C 信号，DTE 与计算机间的电平转换电路未画出。

1、采用 Modem(DCE)和电话网通信时的信号连接：

若在双方 MODEM 之间采用普通电话交换线进行通信，除了需要 2～8 号信号线外还要增加 RI(22 号)和 DTR(20 号)两个信号线进行联络，如图 1 所示。

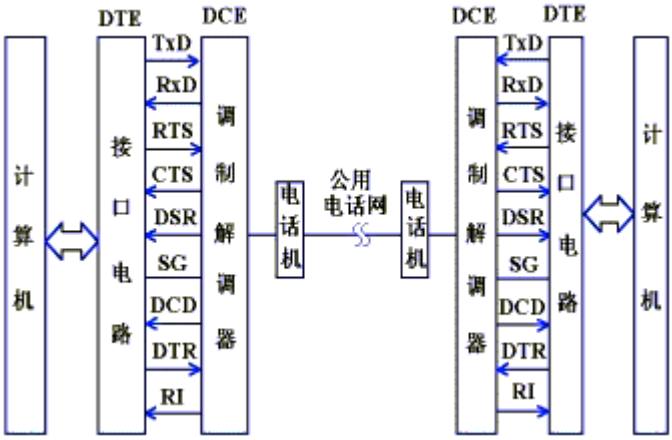


图 1

DSR、DTR：数传机（DCE）准备好、数据终端（DTE）准备好，只表示设备本身可用。

首先，通过电话机拨号呼叫对方，电话交换台向对方发出拨号呼叫信号，当对方 DCE 收到该信号后，使 RI（振铃信号）有效，通知 DTE，已被呼叫。当对方“摘机”后，两方建立了通信链路。

若计算机要发送数据至对方，首先通过接口电路（DTE）发出 RTS（请求发送）信号。此时，若 DCE（Modem）允许传送，则向 DTE 回答 CTS（允许发送）信号。一般可直接将 RTS/CTS 接高电平，即只要通信链路已建立，就可传送信号。（RTS/CTS 可只用于半双工系统中作发送方式和接收方式的切换。

当 DTE 获得 CTS 信号后，通过 TXD 线向 DCE 发出串行信号，DCE（Modem）将这些数字信号调制成模拟信号（又称载波信号），传向对方。

计算机向 DTE“数据输出寄存器”传送新的数据前，应检查 Modem 状态和数据输出寄存器为空。当对方的 DCE 收到载波信号后，向对方的 DTE 发出 DCD 信号（数据载波检出），通知其 DTE 准备接收，同时，将载波信号解调为数据信号，从 RXD 线上送给 DTE，DTE 通过串行接收移位寄存器对接收到的位流进行移位，当收到 1 个字符的全部位流后，把该字符的数据位送到数据输入寄存器，CPU 可以从数据输入寄存器读取字符。

2、采用专用电话线通信：在通信双方的 MODEM 之间采用电话线进行通信，则只要使用 2～8 号信号线进行联络与控制。不需要电话机、振铃信号 RI 和 DTR 信号，其信号线的连接如图 2 那样。

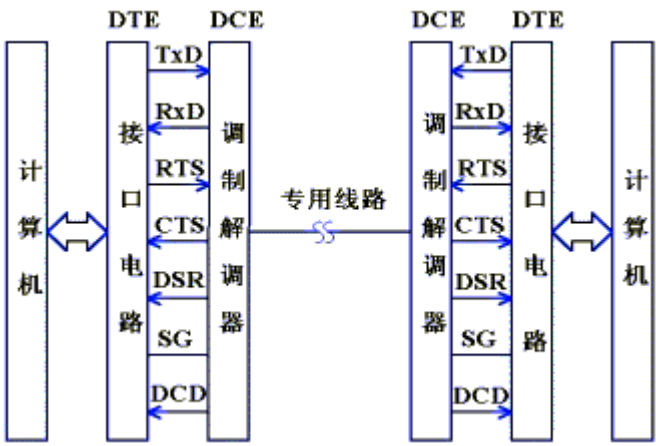


图 2

二、近距离通信：

当通信距离较近时，可不需要 Modem，通信双方可以直接连接，这种情况下，只需使用少数几根信号线。最简单的情况，在通信中根本不需要 RS-232C 的控制联络信号，只需三根线（发送线、接收线、信号地线）便可实现全双工异步串行通信，即是这里要讨论的第一种情况。

无 Modem 时，最大通信距离按如下方式计算：

RS-232C 标准规定：当误码率小于 4%时，要求导线的电容值应小于 2500PF。对于普通导线，其电容值约为 170PF/M。则允许距离 L=2500PF/（170PF/M）=15M

这一距离的计算，是偏于保守的，实际应用中，当使用 9600bps，普通双绞屏蔽线时，距离可达 30~35 米。

1、零 Modem 的最简连线（3 线制）

图 3 是零 MODEM 方式的最简单连接（即三线连接），图中的 2 号线与 3 号线交叉连接是因为在直连方式时，把通信双方都当作数据终端设备看待，双方都可发也可收。在这种方式下，通信双方的任何一方，只要请求发送 RTS 有效和数据终端准备好 DTR 有效就能开始发送和接收。

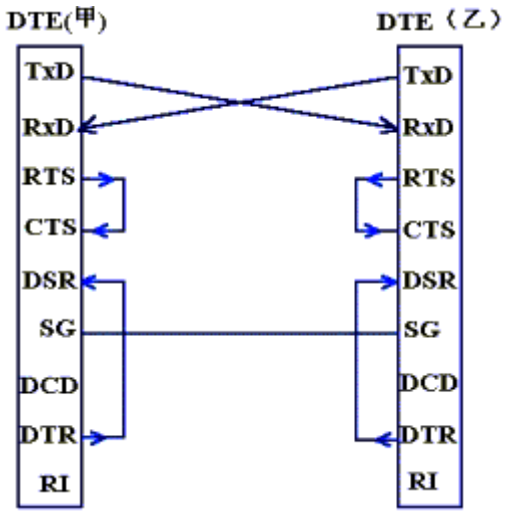


图 3

（1）RTS 与 CTS 互联：只要请求发送，立即得到允许

（2）DTR 与 DSR 互联：只要本端准备好，认为本端立即可以接收（DSR、数传机准备好）。

2、零 Modem 标准连接：

如果想在直接连接时，而又考虑到 RS-232C 的联络控制信号，则采用零 MODEM 方式的标准连接方法，其通信双方信号线安排如下 1-2-3-4-5 顺序所演示的那样。

无 Modem 的标准联线（7 线制）如图所示：

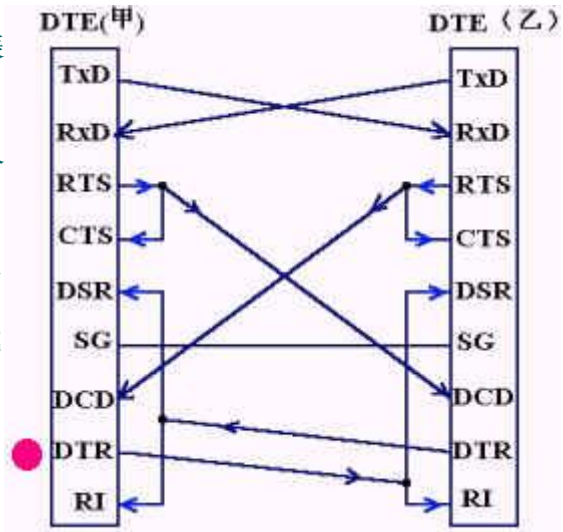
从中可以看出，RS-232C 接口标准定义的所有信号线都用到了，并且是按照 DTE 和 DCE 之间信息交换协议的要求进行连接的，只不过是把 DTE 自己发出的信号线送过来，当作对方 DCE 发来的信号，因此，又把这种连接称为双叉环回接口。

双方的握手信号关系如下（注：甲方乙方并未在图中标出）：

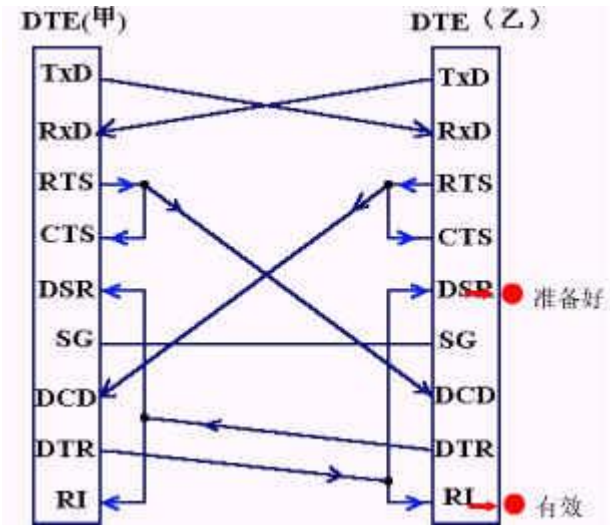
（1）当甲方的 DTE 准备好，发出 DTR 信号，该信号直接联至乙方的 RI（振铃信号）和 DSR（数传机准备好）。即只要甲方准备好，乙方立即产生呼叫（RI）有效，并同时准备好（DSR）。尽管此时乙方并不存在 DCE（数传机）。

（2）甲方的 RTS 和 CTS 相连，并与乙方的 DCD 互连。即：一旦甲方请求发送（RTS），便立即得到允许（CTS），同时，使乙方的 DCD 有效，即检测到载波信号。

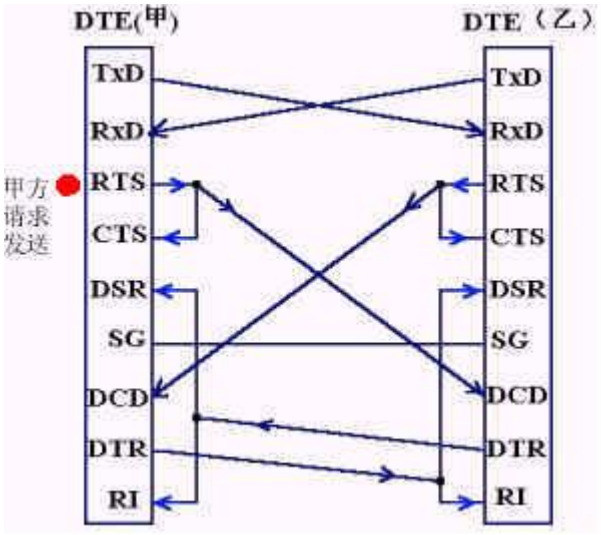
（3）甲方的 TXD 与乙方的 RXD 相连，一发一收。



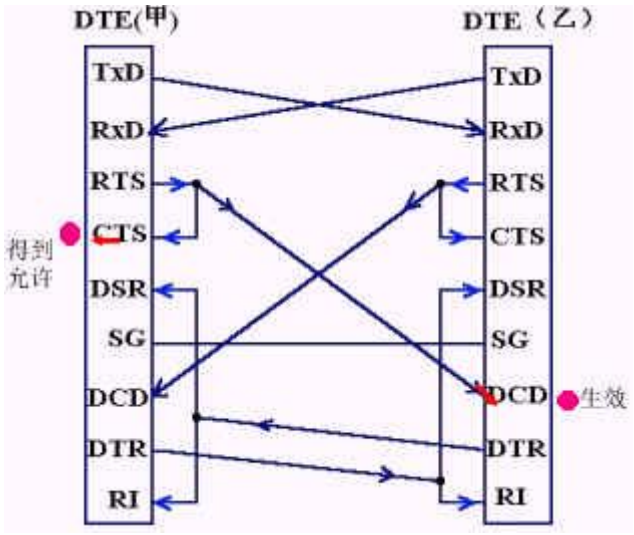
1



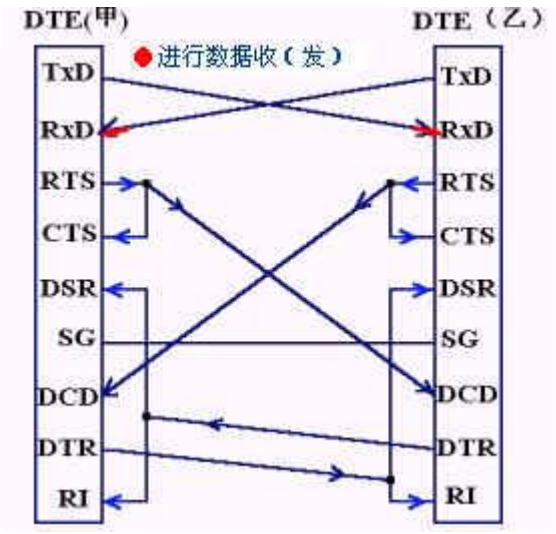
2


DTE(甲)
DTE（乙）
TxD
RxD
RTS
CTS
DSR
SG
DCD
DTR
RI
甲方
请求
发送

3


DTE(甲)
DTE（乙）
TxD
RxD
RTS
CTS
DSR
SG
DCD
DTR
RI
得到
允许
生效

4


DTE(甲)
DTE（乙）
进行数据收（发）
TxD
RxD
RTS
CTS
DSR
SG
DCD
DTR
RI

5

串口通信基本接线方法

目前较为常用的串口有 9 针串口（DB9）和 25 针串口（DB25），通信距离较近时(<12m)，可以用电缆线直接连接标准 RS232 端口(RS422,RS485 较远)，若距离较远，需附加调制解调器（MODEM）。最为简单且常用的是三线制接法，即地、接收数据和发送数据三脚相连，本文只涉及到最为基本的接法，且直接用 RS232 相连，以回答前段网友的咨询。

1.DB9 和 DB25 的常用信号脚说明

9 针串口（DB9）			25 针串口（DB25）		
针号	功能说明	缩写	针号	功能说明	缩写
1	数据载波检测	DCD	8	数据载波检测	DCD
2	接收数据	RXD	3	接收数据	RXD
3	发送数据	TXD	2	发送数据	TXD
4	数据终端准备	DTR	20	数据终端准备	DTR
5	信号地	GND	7	信号地	GND
6	数据设备准备好	DSR	6	数据准备好	DSR
7	请求发送	RTS	4	请求发送	RTS
8	清除发送	CTS	5	清除发送	CTS
9	振铃指示	DELL	22	振铃指示	DELL

2.RS232C 串口通信接线方法（三线制）

首先，串口传输数据只要有接收数据针脚和发送针脚就能实现：同一个串口的接收脚和发送脚直接用线相连，两个串口相连或一个串口和多个串口相连

- 同一个串口的接收脚和发送脚直接用线相连 对 9 针串口和 25 针串口，均是 2 与 3 直接相连；
- 两个不同串口（不论是同一台计算机的两个串口或分别是不同计算机的串口）

9 针－9 针		25 针－25 针		9 针－25 针	
2	3	3	2	2	2
3	2	2	3	3	3
5	5	7	7	5	7

上面表格是对微机标准串行口而言的，还有许多非标准设备，如接收 GPS 数据或电子罗盘数据，只要记住一个原则：接收数据针脚（或线）与发送数据针脚（或线）相连，彼些交叉，信号地对应相接。

3.串口调试中要注意的几点：

- 不同编码机制不能混接，如 **RS232C** 不能直接与 **RS422** 接口相连，市面上专门的各种转换器卖，必须通过转换器才能连接；
- 线路焊接要牢固，不然程序没问题，却因为接线问题误事；
- 串口调试时，准备一个好用的调试工具，如串口调试助手、串口精灵等，有事半功倍之效果；
- 强烈建议不要带电插拔串口，插拔时至少有一端是断电的，否则串口易损坏。

串口通讯的概念及接口电路

随着计算机系统的应用和微机网络的发展，通信功能越来越显的重要。这里所说的通信是只计算机与外界的信息交换。因此，通信既包括计算机与外部设备之间，也包括计算机和计算机之间的信息交换。由于串行通信是在一根传输线上一位一位的传送信息，所用的传输线少，并且可以借助现成的电话网进行信息传送，因此，特别适合于远距离传输。对于那些与计算机相距不远的人－机交换设备和串行存储的外部设备如终端、打印机、逻辑分析仪、磁盘等，采用串行方式交换数据也很普遍。在实时控制和管理方面，采用多台微机处理机组成分级分布控制系统中，各 **CPU** 之间的通信一般都是串行方式。所以串行接口是微机应用系统常用的接口。

许多外设和计算机按串行方式进行通信，这里所说的串行方式，是指外设与接口电路之间的信息传送方式，实际上，**CPU** 与接口之间仍按并行方式工作。

1 串行通信的概念

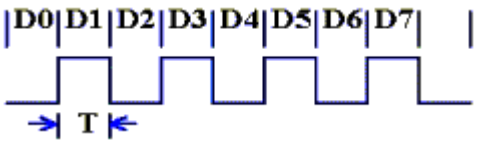


图 **1-1**

所谓“串行通信”是指外设和计算机间使用一根数据信号线(另外需要地线,可能还需要控制线),数据在一根数据信号线上一位一位地进行传输，每一位数据都占据一个固定的时间长度。如图 **1-1** 所示。这种通信方式使用的数据线少，在远距离通信中可以节约通信成本，当然，其传输速度比并行传输慢。

由于 **CPU** 与接口之间按并行方式传输，接口与外设之间按串行方式传输，因此，在串行接口中，必须要有“接收移位寄存器”（串→并）和“发送移位寄存器”（并→串）。典型的串行接口的结构如 **1-2** 所示。

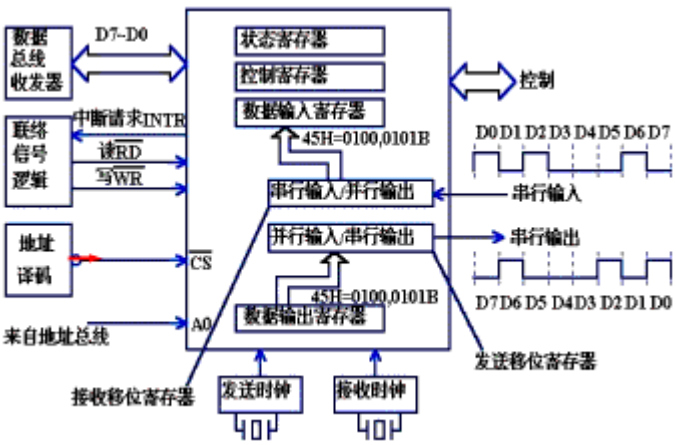


图 1-2

在数据输入过程中，数据 1 位 1 位地从外设进入接口的“接收移位寄存器”，当“接收移位寄存器”中已接收完 1 个字符的各位后，数据就从“接收移位寄存器”进入“数据输入寄存器”。CPU 从“数据输入寄存器”中读取接收到的字符。（并行读取，即 D7~D0 同时被读至累加器中）。“接收移位寄存器”的移位速度由“接收时钟”确定。

在数据输出过程中，CPU 把要输出的字符（并行地）送入“数据输出寄存器”，“数据输出寄存器”的内容传输到“发送移位寄存器”，然后由“发送移位寄存器”移位，把数据 1 位 1 位地送到外设。“发送移位寄存器”的移位速度由“发送时钟”确定。

接口中的“控制寄存器”用来容纳 CPU 送给此接口的各种控制信息，这些控制信息决定接口的工作方式。

“状态寄存器”的各位称为“状态位”，每一个状态位都可以用来指示数据传输过程中的状态或某种错误。例如，用状态寄存器的 D5 位为“1”表示“数据输出寄存器”空，用 D0 位表示“数据输入寄存器满”，用 D2 位表示“奇偶检验错”等。

能够完成上述“串<- ->并”转换功能的电路，通常称为“通用异步收发器”（UART：Universal Asynchronous Receiver and Transmitter）,典型的芯片有：Intel 8250/8251,16550。

有关 RS232 和 RS485 接口的问答

什么是 RS-232-C 接口？采用 RS-232-C 接口有何特点？传输电缆长度如何考虑？

答： 计算机与计算机或计算机与终端之间的数据传送可以采用串行通讯和并行通讯二种方式。由于串行通讯方式具有使用线路少、成本低，特别是在远程传输时，避免了多条线路特性的不一致而被广泛采用。 在串行通讯时，要求通讯双方都采用一个标准接口，使不同 的设备可以方便地连接起来进行通讯。 RS-232-C 接口（又称 EIA RS-232-C）是目前最常用的一种串行通讯接口。它是在 1970 年由美国电子工业协会（EIA）联合贝尔系统、 调制解调器厂家及计算机终端生产厂家共同制定的用于串行通讯的标 准。它的全名是“数据终端设备（DTE）和数据通讯设备（DCE）之间 串行二进制数据交换接口技术标准”该标准规定采用一个 25 个脚的 DB25 连接器，对连接器的每个引脚的信号内容加以规定，还对各种信 号的电平加以规定。

（1）接口的信号内容 实际上 RS-232-C 的 25 条引线中有许多是很少使用的，在计算机与终端通讯中一般只使用 3-9 条引线。RS-232-C 最常用的 9 条引线的信号内容见附表 1 所示

（2）接口的电气特性 在 RS-232-C 中任何一条信号线的电压均为负逻辑关系。即：逻 辑“1”，-5— -15V；逻辑“0” +5— +15V 。噪声容限为 2V。即 要求接收器能识别低至+3V 的信号作为逻辑“0”，高到-3V 的信号 作为逻辑“1” 附表 1

引脚序号	信号名称	符号	流向	功能
2	发送数据	TXD	DTE→DCE	DTE 发送串行数据
3	接收数据	RXD	DTE←DCE	DTE 接收串行数据
4	请求发送	RTS	DTE→DCE	DTE 请求 DCE 将线路切换到发送方式
5	允许发送	CTS	DTE←DCE	DCE 告诉 DTE 线路已接通可以发送数据
6	数据设备准备好	DSR	DTE←DCE	DCE 准备好
7	信号地			信号公共地

8	载波检测	DCD	DTE←DCE	表示 DCE 接收到远程载波
20	数据终端准备好	DTR	DTE→DCE	DTE 准备好
22	振铃指示	RI	DTE←DCE	表示 DCE 与线路接通,出现振铃

(3) 接口的物理结构 RS-232-C 接口连接器一般使用型号为 DB-25 的 25 芯插头座,通常插头在 DCE 端,插座在 DTE 端. 一些设备与 PC 机连接的 RS-232-C 接口,因为不使用对方的传送控制信号,只需三条接口线,即"发送数据"、"接收数据"和"信号地"。所以采用 DB-9 的 9 芯插头座,传输线采用屏蔽双绞线。

(4) 传输电缆长度 由 RS-232C 标准规定在码元畸变小于 4%的情况下,传输电缆长度应为 50 英尺,其实这个 4%的码元畸变是很保守的,在实际应用中,约有 99%的用户是按码元畸变 10-20%的范围工作的,所以实际使用中最大距离会远超过 50 英尺,美国 DEC 公司曾规定允许码元畸变为 10%而得出附表 2 的实验结果。其中 1 号电缆为屏蔽电缆,型号为 DECP.NO.9107723 内有三对双绞线,每对由 22# AWG 组成,其外覆以屏蔽网。2 号电缆为不带屏蔽的电缆。型号为 DECP.NO.9105856-04 是 22#AWG 的四芯电缆。 附表 2 DEC 公司的实验结果

波特率	1 号电缆传输距离(英尺)	2 号电缆传输距离(英尺)
110	5000	3000
300	5000	3000
1200	3000	3000
2400	1000	500
4800	1000	250
9600	250	250

2. 什么是 RS-485 接口?它比 RS-232-C 接口相比有何特点?

答: 由于 RS-232-C 接口标准出现较早,难免有不足之处,主要有以下四点:

(1) 接口的信号电平值较高,易损坏接口电路的芯片,又因为与 TTL 电平不兼容故需使用电平转换电路方能与 TTL 电路连接。

(2) 传输速率较低,在异步传输时,波特率为 20Kbps。

(3) 接口使用一根信号线和一根信号返回线而构成共地的传输形式, 这种共地传输容易产生共模干扰,所以抗噪声干扰性弱。

(4) 传输距离有限,最大传输距离标准值为 50 英尺,实际上也只能 用在 50 米左右。

针对 RS-232-C 的不足,于是就不断出现了一些新的接口标准,RS-485 就是其中之一,它具有以下特点:

1. RS-485 的电气特性:逻辑"1"以两线间的电压差为+(2—6) V 表示;逻辑"0"以两线间的电压差为-(2—6)V 表示。接口信号电平比 RS-232-C 降低了,就不易损坏接口电路的芯片, 且该电平与 TTL 电平兼容,可方便与 TTL 电路连接。

2. RS-485 的数据最高传输速率为 10Mbps

3．RS-485 接口是采用平衡驱动器和差分接收器的组合，抗共模干能力增强，即抗噪声干扰性好。

4．RS-485 接口的最大传输距离标准值为 4000 英尺，实际上可达 3000 米，另外 RS-232-C 接口在总线上只允许连接 1 个收发器， 即单站能力。而 RS-485 接口在总线上是允许连接多达 128 个收发器。即具有多站能力,这样用户可以利用单一的 RS-485 接口方便地建立起设备网络。

因 RS-485 接口具有良好的抗噪声干扰性，长的传输距离和多站能力等上述优点就使其成为首选的串行接口。因为 RS485 接口组成的半双工网络,一般只需二根连线,所以 RS485 接口均采用屏蔽双绞线传输。RS485 接口连接器采用 DB-9 的 9 芯插头座,与智能终端 RS485 接口采用 DB-9（孔）,与键盘连接的键盘接口 RS485 采用 DB-9（针）。

3．采用 RS485 接口时，传输电缆的长度如何考虑？

答： 在使用 RS485 接口时，对于特定的传输线经，从发生器到负载其数据信号传输所允许的最大电缆长度是数据信号速率的函数，这个 长度数据主要是受信号失真及噪声等影响所限制。下图所示的最大电缆长度与信号速率的关系曲线是使用 24AWG 铜芯双绞电话电缆（线 径为 0.51mm），线间旁路电容为 52.5PF/M，终端负载电阻为 100 欧 时所得出。（曲线引自 GB11014-89 附录 A）。由图中可知，当数据信 号速率降低到 90Kbit/S 以下时，假定最大允许的信号损失为 6dBV 时， 则电缆长度被限制在 1200M。实际上，图中的曲线是很保守的，在实 用时是完全可以取得比它大的电缆长度。 当使用不同线径的电缆。则取得的最大电缆长度是不相同的。例 如：当数据信号速率为 600Kbit/S 时，采用 24AWG 电缆，由图可知最 大电缆长度是 200m，若采用 19AWG 电缆（线径为 0。91mm）则电缆长 度将可以大于 200m； 若采用 28AWG 电缆（线径为 0。32mm）则电缆 长度只能小于 200m。

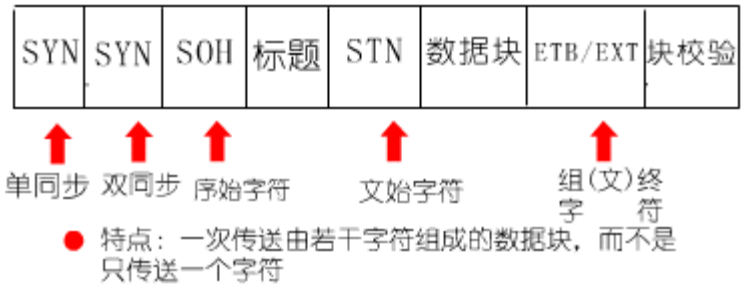
同步通信方式

1、同步通信方式的特点：

采用同步通信时，将许多字符组成一个信息组，这样，字符可以一个接一个地传输，但是，在每组信息（通常称为帧）的开始要加上同步字符，在没有信息要传输时，要填上空字符，因为同步传输不允许有间隙。在同步传输过程中，一个字符可以对应 5～8 位。当然，对同一个传输过程，所有字符对应同样的数位，比如说 n 位。这样，传输时，按每 n 位划分为一个时间片，发送端在一个时间片中发送一个字符，接收端则在一个时间片中接收一个字符。

同步传输时，一个信息帧中包含许多字符，每个信息帧用同步字符作为开始，一般将同步字符和空字符用同一个代码。在整个系统中，由一个统一的时钟控制发送端的发送和空字符用同一个代码。接收端当然是应该能识别同步字符的，当检测到有一串数位和同步字符相匹配时，就认为开始一个信息帧，于是，把此后的数位作为实际传输信息来处理。

2、面向字符的同步协议（IBM 的 BSC 协议）



该协议规定了 **10** 个特殊字符（称为控制字符）作为信息传输的标志。其格式为

SYN	**SOH**	标题	**STX**	数据块	**ETB/ETX**	块校验

SYN：同步字符（**Synchronous character**），每帧可加 **1** 个（单同步）或 **2** 个（双同步）同步字符。

SOH：标题开始（**Start of Header**）。

标题：**Header**，包含源地址（发送方地址）、目的地址（接收方地址）、路由指示。

STX：正文开始（**Start of Text**）。

数据块：正文（**Text**），由多个字符组成。

ETB:块传输结束（**end of transmission block**）， 标识本数据块结束。

ETX：全文结束（**end of text**），（全文分为若干块传输）。

块校验：对从 **SOH** 开始，直到 **ETB/ETX** 字段的检验码。

3、面向 bit 的同步协议（ISO 的 HDLC）

一帧信息可以是任意位，用位组合标识帧的开始和结束。 帧格式为：

F 场	**A** 场	**C** 场	**I** 场	**FC** 场	**F** 场

F 场：标志场;作为一帧的开始和结束，标志字符为 **8** 位，**01111110**。

A 场：地址场，规定接收方地址，可为 **8** 的整倍位。接收方检查每个地址字节的第 **1** 位，如果为“**0**”，则后边跟着另一

个地址字节。若为“**1**”，则该字节为最后一个地址字节。

C 场：控制场。指示信息场的类型，**8** 位或 **16** 位。若第 **1** 字节的第 **1** 位为 **0**，则还有第 **2** 个字节也是控制场。

I 场：信息场。要传送的数据。

FC 场：帧校验场。**16** 位循环冗余校验码 **CRC**。除 **F** 场和自动插入的“**0**”位外，均参加 **CRC** 计算。

4、同步通信的“0 位插入和删除技术”

在同步通信中，一帧信息以一个（或几个）特殊字符开始，例如，**F** 场**=01111110B**。

但在信息帧的其他位置，完全可能出现这些特殊字符，为了避免接收方把这些特殊字符误认为帧的开始，发送方采用了“**0** 位插入技术”，相应地，接收方采用“**0** 位删除技术”。

发送方的 **0** 位插入：除了起始字符外，当连续出现 **5** 个 **1** 时，发送方自动插入一个 **0**。使得在整个信息帧中，只有起始字符含有连续的 **6** 个 **1**。

接收方的“**0** 位删除技术”：接收方收到连续 **6** 个 **1**，作为帧的起始，把连续出现 **5** 个 **1** 后的 **0** 自动删除。

5、同步通信的“字节填充技术”

设需要传送的原始信息帧为：

SOT	**DATA**	**EOT**

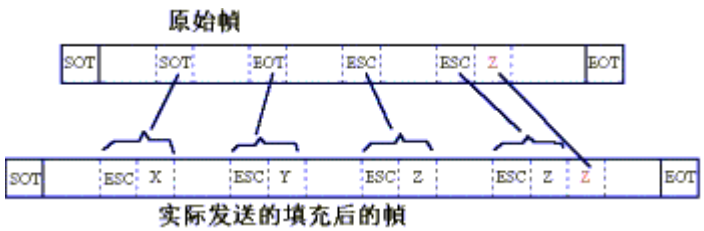
节填充技术采用字符替换方式，使信息帧的 **DATA** 中不出现起始字符 **SOT** 和结束字符 **EOT**。

设按下表方式进行替换：

DATA 中的原字符	替换为
SOT	**ESC X**
EOT	**ESC Y**
ESC	**ESC Z**

其中，**ESC=1AH**，**X**、**Y**、**Z** 可指定为任意字符（除 **SOT**、**EOT**、**ESC** 外）。

发送方按约定方式对需要发送的原始帧进行替换，并把替换后的新的帧发送给接收方。例如图所示：



接收方按约定方式进行相反替换，可以获得原始帧信息。

6、异步通信和同步通信的比较

（**1**）异步通信简单，双方时钟可允许一定误差。同步通信较复杂，双方时钟的允许误差较小。

（**2**）异步通信只适用于点**<-->** 点，同步通信可用于点**<-->** 多。

（**3**）通信效率：异步通信低，同步通信高。

通信协议

所谓通信协议是指通信双方的一种约定。约定包括对数据格式、同步方式、传送速度、传送步骤、检纠错方式以及控制字符定义等问题做出统一规定，通信双方必须共同遵守。因此，也叫做通信控制规程，或称传输控制规程，它属于 ISO'S OSI 七层参考模型中的数据链路层。

目前，采用的通信协议有两类：异步协议和同步协议。同步协议又有面向字符和面向比特以及面向字节计数三种。其中，面向字节计数的同步协议主要用于 DEC 公司的网络体系结构中。

一、物理接口标准

1.串行通信接口的基本任务

（1）实现数据格式化：因为来自 CPU 的是普通的并行数据，所以，接口电路应具有实现不同串行通信方式下的数据格式化的任务。在异步通信方式下，接口自动生成起止式的帧数据格式。在面向字符的同步方式下，接口要在待传送的数据块前加上同步字符。

（2）进行串－并转换：串行传送，数据是一位一位串行传送的，而计算机处理数据是并行数据。所以当数据由计算机送至数据发送器时，首先把串行数据转换为并行数才能送入计算机处理。因此串并转换是串行接口电路的重要任务。

（3）控制数据传输速率：串行通信接口电路应具有对数据传输速率——波特率进行选择和控制的能力。

（4）进行错误检测：在发送时接口电路对传送的字符数据自动生成奇偶校验位或其他校验码。在接收时，接口电路检查字符的奇偶校验或其他校验码，确定是否发生传送错误。

（5）进行 TTL 与 EIA 电平转换：CPU 和终端均采用 TTL 电平及正逻辑，它们与 EIA 采用的电平及负逻辑不兼容，需在接口电路中进行转换。

（6）提供 EIA-RS-232C 接口标准所要求的信号线：远距离通信采用 MODEM 时，需要 9 根信号线；近距离零 MODEM 方式，只需要 3 根信号线。这些信号线由接口电路提供，以便与 MODEM 或终端进行联络与控制。

2、串行通信接口电路的组成

为了完成上述串行接口的任务，串行通信接口电路一般由可编程的串行接口芯片、波特率发生器、EIA 与 TTL 电平转换器以及地址译码电路组成。其中，串行接口芯片，随着大规模继承电路技术的发展，通用的同步(USRT)和异步（UART）接口芯片种类越来越多，如下表所示。它们的基本功能是类似的，都能实现上面提出的串行通信接口基本任务的大部分工作，且都是可编程的。才用这些芯片作为串行通信接口电路的核心芯片，会使电路结构比较简单。

芯片	同步(USRT)		异步(UART)（起止式）	传输速率 b/s	
	面向字符	HDLC		同步	异步
INS8250			√		56K
MC6850			√		1M
MC6852	√			1.5M	

MC6854		√		1.5M	
Int8251A	√		√	64K	19.2K
Int8273		√		64K	
Z-80 SIO		√	√	800K	

3.有关串行通信的物理标准

为使计算机、电话以及其他通信设备互相沟通，现在，已经对串行通信建立了几个一致的概念和标准，这些概念和标准属于三个方面：传输率，电特性，信号名称和接口标准。

1、传输率：所谓传输率就是指每秒传输多少位，传输率也常叫波特率。国际上规定了一个标准波特率系列，标准波特率也是最常用的波特率，标准波特率系列为 **110**、**300**、**600**、**1200**、**4800**、**9600** 和 **19200**。大多数 **CRT** 终端都能够按 **110** 到 **9600** 范围中的任何一种波特率工作。打印机由于机械速度比较慢而使传输波特率受到限制，所以，一般的串行打印机工作在 **110** 波特率，点针式打印机由于其内部有较大的行缓冲区，所以可以按高达 **2400** 波特的速度接收打印信息。大多数接口的接收波特率和发送波特率可以分别设置，而且，可以通过编程来指定。

2、RS-232-C 标准：RS-232-C 标准对两个方面作了规定，即信号电平标准和控制信号线的定义。**RS-232—C** 采用负逻辑规定逻辑电平，信号电平与通常的 **TTL** 电平也不兼容，**RS-232-C** 将 **-5V～-15V** 规定为“**1**”，**+5V～+15V** 规定为“**0**”。图 **1** 是 **TTL** 标准和 **RS-232-C** 标准之间的电平转换。

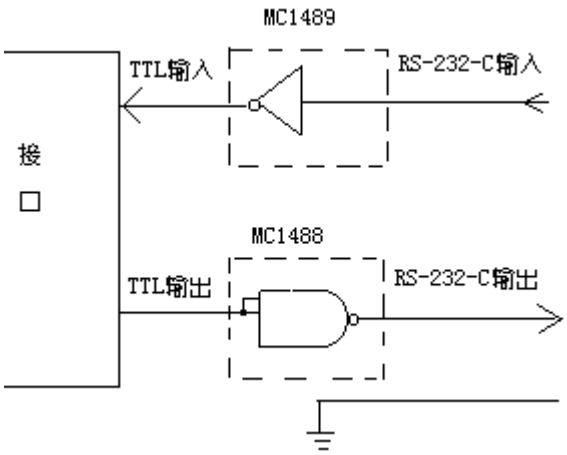


图 **1**

二、软件协议

1.OSI 协议和 TCP/IP 协议

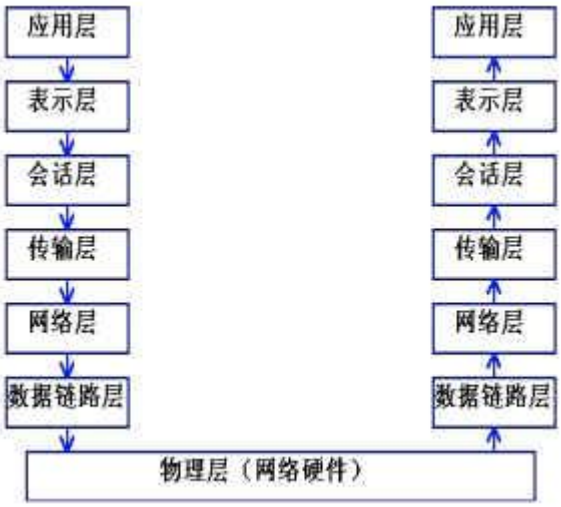


图 2

（1）OSI 协议

OSI 七层参考模型不是通讯标准，它只给出一个不会由于技术发展而必须修改的稳定模型，使有关标准和协议能在模型定义的范围内开发和相互配合。

一般的通讯协议只符合 OSI 七层模型的某几层，如： EIA-RS-232-C：实现了物理层。IBM 的 SDLC（同步数据链路控制规程）：数据链路层。ANSI 的 ADCCP（先进数据通讯规程）：数据链路层 IBM 的 BSC（二进制同步通讯协议）：数据链路层。应用层的电子邮件协议 SMTP 只负责寄信、POP3 只负责收信。

（2）TCP/IP 协议

实现了五层协议。

（1）物理层：对应 OSI 的物理层。
（2）网络接口层：类似于 OSI 的数据链路层。
（3）Internet 层：OSI 模型在 Internet 网使用前提出，未考虑网间连接。
（4）传输层：对应 OSI 的传输层。
（5）应用层：对应 OSI 的表示层和应用层。

2.串行通信协议

串行通信协议分同步协议和异步协议。

（1）异步通信协议的实例——起止式异步协议

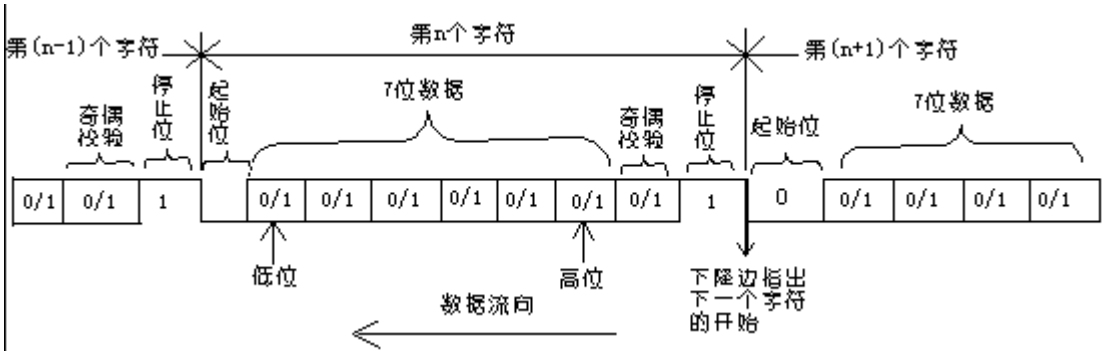


图 3

特点与格式：

起止式异步协议的特点是一个字符一个字符传输，并且传送一个字符总是以起始位开始，以停止位结束，字符之间没有固定的时间间隔要求。其格式如图 **3** 所示。每一个字符的前面都有一位起始位（低电平，逻辑值 **0**），字符本身有 **5**～**7** 位数据位组成，接着字符后面是一位校验位（也可以没有校验位），最后是一位，或意味半，或二位停止位，停止位后面是不定长度的空闲位。停止位和空闲位都规定为高电平（逻辑值），这样就保证起始位开始处一定有一个下跳沿。

从图中可以看出，这种格式是靠起始位和停止位来实现字符的界定或同步的，故称为起始式协议。传送时，数据的低位在前，高位在后，图 **4** 表示了传送一个字符 **E** 的 **ASCAII** 码的波形 **1010001**。当把它的最低有效位写到右边时，就是 **E** 的 **ASCII** 码 **1000101=45H**。

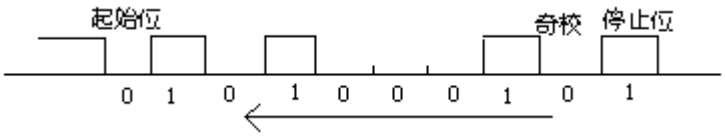


图 4

起／止位的作用：起始位实际上是作为联络信号附加进来的，当它变为低电平时，告诉收方传送开始。它的到来，表示下面接着是数据位来了，要准备接收。而停止位标志一个字符的结束，它的出现，表示一个字符传送完毕。这样就为通信双方提供了何时开始收发，何时结束的标志。传送开始前，发收双方把所采用的起止式格式（包括字符的数据位长度，停止位位数，有无校验位以及是奇校验还是偶校验等）和数据传输速率作统一规定。传送开始后，接收设备不断地检测传输线，看是否有起始位到来。当收到一系列的“**1**”（停止位或空闲位）之后，检测到一个下跳沿，说明起始位出现，起始位经确认后，就开始接收所规定的数据位和奇偶校验位以及停止位。经过处理将停止位去掉，把数据位拼装成一个并行字节，并且经校验后，无奇偶错才算正确的接收一个字符。一个字符接收完毕，接收设备有继续测试传输线，监视“**0**”电平的到来和下一个字符的开始，直到全部数据传送完毕。

由上述工作过程可看到，异步通信是按字符传输的，每传输一个字符，就用起始位来通知收方，以此来重新核对收发双方同步。若接收设备和发送设备两者的时钟频率略有偏差，这也不会因偏差的累积而导致错位，加之字符之间的空闲位也为这种偏差提供一种缓冲，所以异步串行通信的可靠性高。但由于要在每个字符的前后加上起始位和停止位这样一些附加位，使得传输效率变低了，只有约 **80%**。因此，起止协议一般用在数据速率较慢的场合（小于 **19.2kbit/s**）。在高速传送时，一般要采用同步协议。

（2）面向字符的同步协议

特点与格式：这种协议的典型代表是 IBM 公司的二进制同步通信协议(BSC）。它的特点是一次传送由若干个字符组成的数据块，而不是只传送一个字符，并规定了 10 个字符作为这个数据块的开头与结束标志以及整个传输过程的控制信息，它们也叫做通信控制字。由于被传送的数据块是由字符组成，故被称作面向字符的协议。

特定字符（控制字符）的定义：由上面的格式可以看出，数据块的前后都加了几个特定字符。SYN 是同步字符(synchronous Character），每一帧开始处都有 SYN，加一个 SYN 的称单同步，加两个 SYN 的称双同步设置同步字符是起联络作用，传送数据时，接收端不断检测，一旦出现同步字符，就知道是一帧开始了。接着的 SOH 是序始字符（Start Of Header），它表示标题的开始。标题中包括院地址、目的地址和路由指示等信息。STX 是文始字符(Start Of Text），它标志着传送的正文（数据块）开始。数据块就是被传送的正文内容，由多个字符组成。数据块后面是组终字符 ETB（End Of Transmission Block）或文终字符 ETX(End Of Text)，其中 ETB 用在正文很长、需要分成若干个分数据块、分别在不同帧中发送的场合，这时在每个分数据块后面用文终字符 ETX。一帧的最后是校验码，它对从 SOH 开始到 ETX（或 ETB）字段进行校验，校验方式可以是纵横奇偶校验或 CRC。另外，在面向字符协议中还采用了一些其他通信控制字，它们的名称如下表所示：

名 称	ASCII	EBCDIC
序始(SOH)	0000001	00000001
文始（STX)	0000010	00000010
组终(ETB)	0010111	00100110
文终（ETX)	0000011	00000011
同步(SYN)	0010110	00110010
送毕(EOT)	0000100	00110111
询问(ENQ)	0000101	00101101
确认(ACK)	0000110	00101110
否认（NAK)	0010101	00111101
转义(DLE)	0010000	00010000

数据透明的实现：面向字符的同步协议，不象异步起止协议那样，需要在每个字符前后附加起始和停止位，因此，传输效率提高了。同时，由于采用了一些传输控制字，故增强了通信控制能力和校验功能。但也存在一些问题，例如，如何区别数据字符代码和特定字符代码的问题，因为在数据块中完全有可能出现与特定字符代码相同的数据字符，这就会发生误解。比如正文有个与文终字符 ETX 的代码相同的数据字符，接收端就不会把它当作为普通数据处理，而误认为是正文结束，因而产生差错。因此，协议应具有将特定字符作为普通数据处理的能力，这种能力叫做“数据透明”。为此，协议中设置了转移字符 DLE(Data Link Escape)。当把一个特定字符看成数据时，在它前面要加一个 DLE，这样接收器收到一个 DLE 就可预知下一个字符是数据字符，而不会把它当作控制字符来处理了。DLE 本身也是特定字符，当它出现在数据块中时，也要在它前面加上另一个 DLE。这种方法叫字符填充。字符填

充实现起来相当麻烦，且依赖于字符的编码。正是由于以上的缺点，故又产生了新的面向比特的同步协议。

（3）面向比特的同步协议

特点与格式：面向比特的协议中最具有代表性的是 IBM 的同步数据链路控制规程 SDLC（Synchronous Data Link Control),国际标准化组织 ISO(International Standard Organization）的高级数据链路控制规程 HDLC（High Level Data link Control),美国国家标准协会(Americal National Standard Institute)的先进数据通信规程 ADCCP(Advanced Data Communication Control Procedure)。这些协议的特点是所传输的一帧数据可以是任意位，而且它是靠约定的位组合模式，而不是靠特定字符来标志帧的开始和结束，故称“面向比特”的协议。这中协议的一般帧格式如图 5 所示：

8位	8位	8位	≥0位	16位	8位
01111110	A	C	1	FC	01111110
开始标志	地址场所	控制场	信息场	校验场	结束标志

图 5

帧信息的分段：由图 5 可见，SDLC/HDLC 的一帧信息包括以下几个场(Filed），所有场都是从有效位开始传送。

（1）SDLC/HDLC 标志字符：SDLC/HDLC 协议规定，所有信息传输必须以一个标志字符开始，且以同一个字符结束。这个标志字符是 01111110，称标志场(F)。从开始标志到结束标志之间构成一个完整的信息单位，称为一帧(Frame)。所有的信息是以帧的形传输的，而标志字符提供了每一帧的边界。接收端可以通过搜索“01111110”来探知帧的开头和结束，以此建立帧同步。

（2）地址场和控制场：在标志场之后，可以有一个地址场 A(Address）和一个控制场 C(Control)。地址场用来规定与之通信的次站的地址。控制场可规定若干个命令。SDLC 规定 A 场和 C 场的宽度为 8 位或 16 位。接收方必须检查每个地址字节的第一位，如果为“0”，则后面跟着另一个地址字节；若为“1”，则该字节就是最后一个地址字节。同理，如果控制场第一个字节的第一位为为“0”，则还有第二个控制场字节，否则就只有一个字节。

（3）信息场：跟在控制场之后的是信息场 I(Information)。I 场包含有要传送的数据，并不是每一帧都必须有信息场。即数据场可以为 0，当它为 0 时，则这一帧主要是控制命令。

（4）帧校验信息：紧跟在信息场之后的是两字节的争校验，帧校验场称为 FC(Frame Check)场或称为帧校验序列 FCS(Frame check Squence)。SDLC/HDLC 均采用 16 位循环冗余校验码 CRC（Cyclic Redundancy Code)。除了标志场和自动插入的“0”以外，所有的信息都参加 CRC 计算。

实际应用时的两个技术问题：

（1）“0”位插入/删除：如上所述，SDLC/HDLC 协议规定以 01111110 为标志字节，但在信息场中也完全有可能有同一种模式的字符，为了把它与标志区分开来，所以采取了“0”位插入和删除技术。具体作法是发送端在发送所有信息（除标志字节外）时，只要遇到连续 5 个“1”，就自动插入一个“0”，当接收端在接收数据时（除标志字节）如果连续收到 5 个“1”，就自动将其后的一个“0”删除是，以恢复信息的原有形式。这种“0”位的插入和删除过程是由硬件自动完成的。

（2）SDLC/HDLC 异常结束：若在发送过程中出现错误，则 SDLC/HDLC 协议常用异常结束(Abort)字符，或称为失效序列使本帧作废。在 HDLC 规程中，7 个连续的“1”被作为失效字符，而在 SDLC 中失效字符是 8 个连续的“1”。当然在试销序列中不使用“0”位插入/删除技术。SDLC/HDLC 协议规定，在一帧之内不允许出现数据间隔。在两帧之间，发送器可以连续输出标志字符序列，也可以输出连续的高电平，它被称为空闲（Idle）信号。

实战串行通讯

本文不是全面的讲述如何编写串行通讯程序，而是讨论一些实际遇到的问题。

1 选择通讯方式 -- 同步还是非同步

正如在《Serial communications in Microsoft Win32》等文章中提到的，同步（NonOverLapped）方式是比较简单的一种方式，编写起来代码的长度要明显少于异步（OverLapped）方式，我开始用同步方式编写了整个子程序，在 Windows98 下工作正常，但后来在 Windows2000 下测试，发现接收正常，但一发送数据，程序就会停在那里，原因应该在于同步方式下如果有一个通讯 Api 在操作中，另一个会阻塞直到上一个操作完成，所以当读数据的线程停留在 WaitCommEvent 的时候，WriteFile 就停在那里。我又测试了我手上所有有关串行通讯的例子程序，发现所有使用同步方式的程序在 Windows 2000 下全部工作不正常，对这个问题我一直找不到解决的办法，后来在 Iczelion 站点上发现一篇文章提到 NT 下对串行通讯的处理和 9x 有些不同，根本不要指望在 NT 或 Windows 2000 下用同步方式同时收发数据，我只好又用异步方式把整个通讯子程序重新写了一遍。

所以对于这个问题的建议是：如果程序只打算工作在 Win9x 下，为了简单起见，可以用同步方式写程序，如果程序打算在 NT 下也可以工作的话，就必须用异步方式写。

2 Win32 通讯 API Bug 之一 --- CommConfigDialog

CommConfigDialog 是弹出系统内置串口设置对话框的 API，我们在设备管理器中设置串口参数的对话框就是这个，使用这个 API 时不用先打开端口，它并不针对一个已打开的端口，而是仅仅是把 DCB 的内容填写到对话框中，当按了 OK 后把输入的结果存回到 DCB 数据结构中，至于什么时候把结果设置到串口上，那就是你自己要做的事情了。

CommCinfigDialog 的定义如下：

```
BOOL CommConfigDialog(

LPTSTR lpszName, // pointer to device name string
HWND hWnd, // handle to window
LPCOMMCONFIG lpCC // pointer to comm. configuration structure
);
```

但在使用中发现，对话框有时能出来，有时出不来，最后总结的经验是问题出在 COMMCONFIG 结构的 dwSize 字段上，COMMCONFIG 的定义如下：

```
typedef struct _COMM_CONFIG {
DWORD dwSize;
WORD wVersion;
```

```
WORD wReserved;
DCB dcb;
DWORD dwProviderSubType;
DWORD dwProviderOffset;
DWORD dwProviderSize;
WCHAR wcProviderData[1];
} COMMCONFIG, *LPCOMMCONFIG;
```

在参数中，wVersion 要填 100h，dwProviderSubType 要填 1，但 dwSize 就不能填 sizeof COMMCONFIG 了，我发现好象一定要把 dwSize 设置为比 sizeof COMMCONFIG 对话框才能出来，所以我用的代码中定义了一个足够大的缓冲区作为结构的地址：

```
_CommConfigDialog proc
local @stCC[256]:BYTE

pushad
invoke RtlZeroMemory,addr @stCC,sizeof @stCC
mov (COMMCONFIG ptr @stCC).dwSize,256
mov (COMMCONFIG ptr @stCC).wVersion,100h
mov (COMMCONFIG ptr @stCC).dwProviderSubType,1
invoke CommConfigDialog,addr [esi].szPortName,[esi].hWnd,addr @stCC
popad
ret

_CommConfigDialog endp
```

3 Win32 通讯 API Bug 之二--- BuildCommDCB

BuildCommDCB 的功能是把一个字符串如 com1:9600,n,8,1 这样的转换到具体的数据填写到 DCB 中，但使用中也存在问题，我发现我用它转换象 com1:9600,e,7,1 之类的带校验位的字符串，它总是无法把这个 e 给我转换过去，设置好串口一看，成了 9600,n,7,1，而上面提到的 CommConfigDialog 返回的结果用来设置串口却是正确的，经过比较，发现问题出在 DCB.fbits.fParity 这个 bit 上，只有把这个 bit 置 1，校验位才是有效的，而 BuildCommDCB 恰恰是漏了这个 bit，所有如果你要使用 BuildCommDCB，别忘了补充把 DCB.fbits.fParity 设置回去，我用的代码是：

```
_BuildCommDCB proc _lpszPara,_lpstDCB

pushad
mov esi,_lpstDCB
assume esi:ptr DCB

invoke RtlZeroMemory,esi,sizeof DCB
invoke BuildCommDCB,_lpszPara,esi
;*********************************************************************
; 根据校验位补充设置 DCB 中的 DCB.fbits.fParity 字段
;*********************************************************************
mov dword ptr [esi].fbits,0010b
```

```
cld
@@:
lodsb
or al,al
jz @F
cmp al,'='
jz _BCD_Check
cmp al,','
jnz @B
_BCD_Check:
lodsb
or al,al
jz @F
or al,20h
cmp al,'n'
jnz @B
;*********************************************************************
; 扫描到 =n 或 ,n 则取消校验位
;*********************************************************************
mov esi,_lpstDCB
and dword ptr [esi].fbits,not 0010b
@@:
popad
ret

_BuildCommDCB endp
```

4 Win32 通讯编程的一般流程

由于同步方式相对比较简单，在这里讲述的是异步方式的流程，在其他的很多文章里提到了 Windows 通讯 API 有二十多个，它们是：

```
BuildCommDCB
BuildCommDCBAndTimeouts
ClearCommBreak
ClearCommError
CommConfigDialog
EscapeCommFunction
GetCommConfig
GetCommMask
GetCommModemStatus
GetCommProperties
GetCommState
GetCommTimeouts
GetDefaultCommConfig
PurgeComm
```

SetCommBreak
SetCommConfig
SetCommMask
SetCommState
SetCommTimeouts
SetDefaultCommConfig
SetupComm
TransmitCommChar
WaitCommEvent

我刚看到这些 API 的时候，都不知道如何使用它们，但并不是所有这些 API 都是必须用的，比如说你要检测当前串口的设置可以只用 SetCommState 而不用 GetCommProperties 和 GetCommConfig，虽然它们返回的信息可能更多。同样，如果有些值你想用缺省的，比如缓冲区的大小和超时的时间等等，那么 SetupComm 和 BuildCommDCBAndTimeouts、SetCommTimeouts 也可以不用，TransmitCommChar 是马上在发送序列中优先插入发送一个字符用的，平时也很少用到，下面讲的是必须用到的 API 和使用步骤：

1. 建立 Event -- 用 CreateEvent

   invoke CreateEvent,NULL,TRUE,FALSE,NULL
   用异步方式操作串口必须要定义 OVERLAPPED 结构，其中的 hEvent 必须自己建立，你要定义两个 OVERLAPPED 结构，一个用于读一个用于写，当然也必须建立两个 Event，把它们放入 OVERLAPPED.hEvent

2. 打开串口 -- 用 CreateFile

   invoke CreateFile,addr szPortName,GENERIC_READ or GENERIC_WRITE,0,NULL,OPEN_EXISTING,FILE_FLAG_OVERLAPPED,NULL
   注意用异步方式必须指定 FILE_FLAG_OVERLAPPED，而文件方式必须 OPEN_EXISTING，读写必须是 GENERIC_READ or GENERIC_WRITE

3. 设置串口参数 -- 用 SetCommState

   invoke SetCommState,hCom,addr dcbx
   hCom 是前面打开成功后返回的句柄，dcbx 是数据结构 DCB，里面包括了通讯的具体参数，至于这个参数的建立，你可以自己填写，也可以用前面提到的 BuildCommDCB 或 CommConfigDialog 填写

4. 建立读数据的线程

   到这里，就可以开始读数据了，一般我们是在主线程中写数据，因为写是我们可以控制的，而读的时候我们不知道数据什么时候会到，所以要建立一个线程专门用来读数据，在这个线程中，我们循环地用 ReadFile 读串口，同时用 WaitCommEvent 检测线路状态。

5. 如果要检测通讯状态，如 CTS 信号，RingIn 等等 -- 用 SetCommMask、WaitCommEvent、ClearCommError、GetCommModemStatus

invoke SetCommMask,hCom,EV_BREAK or EV_CTS or EV_DSR or EV_ERR or EV_RING or EV_RLSD or EV_RXCHAR or EV_RXFLAG or EV_TXEMPTY
SetCommMask 指定 WaitCommEvent 要等待的事件名称，具体的参数请查手册

invoke WaitCommEvent,hCom,addr dwEvent,NULL
WaitCommEvent 等待一直到 SetCommMask 指定事件之一发生

invoke ClearCommError,hCom,addr dwError,addr stComStat
在 WaitCommEvent 以后，要用 ClearCommError 清除事件的 Flag，以便进行下一轮 WaitCommEvent，同时这个 API 可以获得更详细的事件信息

invoke GetCommModemStatus,hCom,addr dwModemStatus
同样，GetCommModemStatus 是用来获得串口线路状态的，如 CTS、RING 等等，当 WaitCommEvent 返回时，只是指出了如 CTS 等等状态有变化，但具体是变成 On 还是 Off 了还要靠这个 API 去取得更详细的信息

6. 读数据 -- 用 ReadFile

invoke ReadFile,hCom,addr szBuffer,sizeof szBuffer,addr dwBytesRead,addr stReadState
最后一个参数是开头定义的 OVERLAPPED 结构的地址，指定了它就表示是用异步方式的读方式，这个 API 会马上返回，接下去要用

invoke GetOverlappedResult,hCom,addr stReadState,addr dwBytesRead,FALSE
将其余的数据读完

7. 结束时关闭端口 -- 停止 WaitCommEvent 的等待以及关闭端口 CloseHandle

平时程序会停留在 WaitCommEvent 的等待中，当要终止线程的时候，必须是程序从 WaitCommEvent 中退出来，这时候要用

按照 Win32 手册上的说明，参数为 NULL 的 SetCommMask 会使另一个线程中的 WaitCommEvent 马上返回，然后就是用 CloseHandle 关闭端口
invoke CloseHandle,hCom

5 Win32 通讯 API Bug 之二--- SetCommMask 和 WaitCommEvent

严格的说这不应该是 Bug，而是偶然的情况，我发现有些时候我的读线程无法结束，跟踪发现是停在了 WaitCommEvent 上，这说明有时候 invoke SetCommMask,hCom,NULL 并不能使 WaitCommEvent 退出，我最后使用的办法是： 在 SetCommMask 以后再执行 invoke SetEvent,stReadState.hEvent，把读的 OVERLAPPED 结构中的 Event 置位，让

WaitCommEvent 认为有 Event 发生，它就会马上返回，也许这并不是普遍的情况，但如果你的程序也是停在了 WaitCommEvent 的地方，不妨一试。

6 如何编写读线程中的循环

按照《Serial communications in Microsoft Win32》一文中的例程，读循环可以用：

```
#define READ_TIMEOUT 500 // milliseconds

DWORD dwRes;
DWORD dwRead;
BOOL fWaitingOnRead = FALSE;
OVERLAPPED osReader = {0};

// Create the overlapped event. Must be closed before exiting
// to avoid a handle leak.
osReader.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

if (osReader.hEvent == NULL)
// Error creating overlapped event; abort.

if (!fWaitingOnRead) {
// Issue read operation.
if (!ReadFile(hComm, lpBuf, READ_BUF_SIZE, &dwRead, &osReader)) {
if (GetLastError() != ERROR_IO_PENDING) // read not delayed?
// Error in communications; report it.
else
fWaitingOnRead = TRUE;
}
else {
// read completed immediately
HandleASuccessfulRead(lpBuf, dwRead);
}
}



if (fWaitingOnRead) {
dwRes = WaitForSingleObject(osReader.hEvent, READ_TIMEOUT);
switch(dwRes)
{
// Read completed.
case WAIT_OBJECT_0:
if (!GetOverlappedResult(hComm, &osReader, &dwRead, FALSE))
// Error in communications; report it.
else
// Read completed successfully.
HandleASuccessfulRead(lpBuf, dwRead);
```

```
// Reset flag so that another opertion can be issued.
fWaitingOnRead = FALSE;
break;

case WAIT_TIMEOUT:
// Operation isn't complete yet. fWaitingOnRead flag isn't
// changed since I'll loop back around, and I don't want
// to issue another read until the first one finishes.
//
// This is a good time to do some background work.
break;

default:
// Error in the WaitForSingleObject; abort.
// This indicates a problem with the OVERLAPPED structure's
// event handle.
break;
}
}
```

这一段程序在 98 下正常，但非常不幸的是在 Win2000 下，ReadFile 总是返回读正确，并不返回 ERROR_IO_PENDING，使下面的 WaitForSingleObject 的循环形同虚设，要命的是，ReadFile 返回读正确却每次只读一个字节，结果程序工作得很奇怪，即使缓冲区中有很多的字符，程序也每次只能读一个字符，要等到发送字符或做其他的操作使线路状态改变了，才能读下一个字符，我不知道这个奇怪的现象是如何发生的，反正我解决的办法是在 ReadFile 前加 WaitCommEvent，真正等到 EV_RXCHAR 以后才去 ReadFile，到最后，我用的循环是这样的，虽然没有一篇文章中的例子是这样的，但它却同时在 windows9x 和 windows2000 下工作得很好：

```
.while dwFlag & IF_CONNECT
;*****************************************************************
; 检测其它的通信事件
; 如果检测到且定义了 lpProcessEvent 则调用 lpProcessEvent
;*****************************************************************
invoke WaitCommEvent,hCom,addr @dwEvent,NULL ;addr stReadState
push eax
invoke ClearCommError,hCom,addr @dwError,addr @stComStat
pop eax
.if eax == 0
invoke GetLastError
.if eax == ERROR_IO_PENDING
or dwFlag,IF_WAITING
.endif
.else
```

```
;这里是线路状态的处理
.endif
;*********************************************************************
; 如果没有在等待异步读的过程中，则读端口
;*********************************************************************
.if ! (dwFlag & IF_WAITING)
mov @dwBytesRead,0
invoke ReadFile,hCom,addr @szBuffer,sizeof @szBuffer,\
addr @dwBytesRead,addr stReadState
.if eax == FALSE
or dwFlag,IF_WAITING
invoke GetLastError
.if eax != ERROR_IO_PENDING
;这里是错误处理
.endif
.else
and dwFlag,not IF_WAITING
mov eax,@dwBytesRead
.if eax != 0
;这里是接收到的数据处理
.endif
.endif
.endif
;*********************************************************************
; 如果在异步读端口中，则等待一段时间
;*********************************************************************
.if dwFlag & IF_WAITING
invoke WaitForSingleObject,stReadState.hEvent,200
.if eax == WAIT_OBJECT_0
and dwFlag,not IF_WAITING
invoke GetOverlappedResult,hCom,addr stReadState,\
addr @dwBytesRead,FALSE
.if eax != 0
mov eax,@dwBytesRead
.if eax != 0
;这里是接收到的数据处理
.endif
.else
;这里是错误处理
invoke ClearCommError,hCom,addr @dwError,addr @stComStat
.endif
.else
;这里是错误处理
.endif
```

```
.endif
.endw
```

7 流控制的问题

在流控制方式为“无”和“软件控制”的情况下，基本上没有什么问题，但在“硬件控制”下，win32 手册中说明 RTS_CONTROL_HANDSHAKE 控制方式的含义是：

Enables RTS handshaking. The driver raises the RTS line when the "type-ahead" (input) buffer is less than one-half full and lowers the RTS line when the buffer is more than three-quarters full. If handshaking is enabled, it is an error for the application to adjust the line by using the EscapeCommFunction function.

也就是说，当缓冲区快满的时候 RTS 会自动 OFF 通知对方暂停发送，当缓冲区重新空出来的时候， RTS 会自动 ON，但我发现当 RTS 变 OFF 以后即使你已经清空了缓冲区， RTS 也不会自动的 ON，造成对方停在那里不发送了，所以，如果要用硬件流控制的话，还要在接收后最好加上检测缓冲区大小的判断，具体是使用 ClearCommError 后返回的 COMSTAT.cbInQue ， 当缓冲区已经空出来的时候， 要使用 invoke EscapeCommFunction,hCom,SETRTS 重新将 RTS 设置为 ON。

全双工和半双工方式

在串行通信中，数据通常是在两个站（如终端和微机）之间进行传送，按照数据流的方向Intel Mobile Pentium III-M(Tualatin)可分成三种基本的传送方式：全双工、半双工、和单工。但单工目前已很少采用，下面仅介绍前两种方式。

1、全双工方式（full duplex）

当数据的发送和接收分流，分别由两根不同的传输线传送时，通信双方都能在同一时刻进行发送和接收操作，这样的传送方式就是全双工制，如图 1 所示。在全双工方式下，通信系统的每一端都设置了发送器和接收器，因此，能控制数据同时在两个方向上传送。全双工方式无需进行方向的切换，因此，没有切换操作所产生的时间延迟，这对那些不能有时间延误的交互式应用（例如远程监测和控制系统）十分有利。这种方式要求通讯双方均有发送器和接收器，同时，需要 2 根数据线传送数据信号。（可能还需要控制线和状态线，以及地线）。

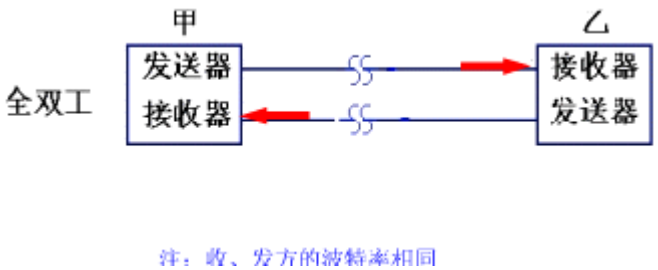


图 1

比如，计算机主机用串行接口连接显示终端，而显示终端带有键盘。这样，一方面键盘上输入的字符送到主机内存；另一方面，主机内存的信息可以送到屏幕显示。通常，往键盘上打入 1 个字符以后，先不显示，计算机主机收到字符后，立即回送到终端，然后终端再把这个字符显示出来。这样，前一个字符的回送过程和后一个字符的输入过程是同时进行的，即工作于全双工方式。

2、半双式方式（half duplex）

若使用同一根传输线既作接收又作发送，虽然数据可以在两个方向上传送，但通信双方不能同时收发数据，这样的传送方式就是半双工制，如图 2 所示。采用半双工方式时，通信系统每一端的发送器和接收器，通过收/发开关转接到通信线上，进行方向的切换，因此，会产生时间延迟。收/发开关实际上是由软件控制的电子开关。

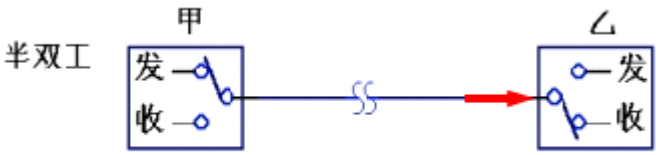


图 2

当计算机主机用串行接口连接显示终端时，在半双工方式中，输入过程和输出过程使用同一通路。有些计算机和显示终端之间采用半双工方式工作，这时，从键盘打入的字符在发送到主机的同时就被送到终端上显示出来，而不是用回送的办法，所以避免了接收过程和发送过程同时进行的情况。

目前多数终端和串行接口都为半双工方式提供了换向能力，也为全双工方式提供了两条独立的引脚。在实际使用时，一般并不需要通信双方同时既发送又接收，像打印机这类的单向传送设备，半双工甚至单工就能胜任，也无需倒向。

浅析 PC 机串口通讯流控制

我们在串行通讯处理中，常常看到 RTS/CTS 和 XON/XOFF 这两个选项，这就是两个流控制的选项，目前流控制主要应用于调制解调器的数据通讯中，但对普通 RS232 编程，了解一点这方面的知识是有好处的。那么，流控制在串行通讯中有何作用，在编制串行通讯程序怎样应用呢？这里我们就来谈谈这个问题。

1.流控制在串行通讯中的作用

这里讲到的“流”，当然指的是数据流。数据在两个串口之间传输时，常常会出现丢失数据的现象，或者两台计算机的处理速度不同，如台式机与单片机之间的通讯，接收端数据缓冲区已满，则此时继续发送来的数据就会丢失。现在我们在网络上通过 MODEM 进行数据传输，这个问题就尤为突出。流控制能解决这个问题，当接收端数据处理不过来时，就发出“不再接收”的信号，发送端就停止发送，直到收到“可以继续发送”的信号再发送数据。因此流控制可以控制数据传输的进程，防止数据的丢失。 PC 机中常用的两种流控制是硬件流控制（包括 RTS/CTS、DTR/CTS 等）和软件流控制 XON/XOFF（继续/停止），下面分别说明。

2.硬件流控制

硬件流控制常用的有 **RTS/CTS** 流控制和 **DTR/DSR**（数据终端就绪/数据设置就绪）流控制。

硬件流控制必须将相应的电缆线连上，用 **RTS/CTS**（请求发送/清除发送）流控制时，应将通讯两端的 **RTS**、**CTS** 线对应相连，数据终端设备（如计算机）使用 **RTS** 来起始调制解调器或其它数据通讯设备的数据流，而数据通讯设备（如调制解调器）则用 **CTS** 来起动和暂停来自计算机的数据流。这种硬件握手方式的过程为：我们在编程时根据接收端缓冲区大小设置一个高位标志（可为缓冲区大小的 **75**%）和一个低位标志（可为缓冲区大小的 **25**%），当缓冲区内数据量达到高位时，我们在接收端将 **CTS** 线置低电平（送逻辑 **0**），当发送端的程序检测到 **CTS** 为低后，就停止发送数据，直到接收端缓冲区的数据量低于低位而将 **CTS** 置高电平。**RTS** 则用来标明接收设备有没有准备好接收数据。

常用的流控制还有还有 **DTR/DSR**（数据终端就绪/数据设置就绪）。我们在此不再详述。由于流控制的多样性，我个人认为，当软件里用了流控制时，应做详细的说明，如何接线，如何应用。

3.软件流控制

由于电缆线的限制，我们在普通的控制通讯中一般不用硬件流控制，而用软件流控制。一般通过 **XON/XOFF** 来实现软件流控制。常用方法是：当接收端的输入缓冲区内数据量超过设定的高位时，就向数据发送端发出 **XOFF** 字符（十进制的 **19** 或 **Control-S**，设备编程说明书应该有详细阐述），发送端收到 **XOFF** 字符后就立即停止发送数据；当接收端的输入缓冲区内数据量低于设定的低位时，就向数据发送端发出 **XON** 字符（十进制的 **17** 或 **Control-Q**），发送端收到 **XON** 字符后就立即开始发送数据。一般可以从设备配套源程序中找到发送的是什么字符。

应该注意，若传输的是二进制数据，标志字符也有可能在数据流中出现而引起误操作，这是软件流控制的缺陷，而硬件流控制不会有这个问题。

顺便说明一下，有不少朋友问到，为什么不在我编写的软件串口调试助手中将流控制加进去，我最初将这个调试工具定位在各种自动控制的串口程序调试上，经过计算和实验验证，在设置的特定采样周期内可以完成通讯任务，就干脆不用流控制。而且在工控中您即使不懂流控制，也能编写出简单的串口通讯程序来，就如我写的串口调试助手。

奇偶校验

串行数据在传输过程中，由于干扰可能引起信息的出错，例如，传输字符‘**E**’，其各位为：

0100，0101=45H

D7 D0

由于干扰，可能使位变为 **1**，这种情况，我们称为出现了“误码”。我们把如何发现传输中的错误，叫“检错”。发现错误后，如何消除错误，叫“纠错”。

最简单的检错方法是“奇偶校验”，即在传送字符的各位之外，再传送 **1** 位奇/偶校验位。可采用奇校验或偶校验。

奇校验：所有传送的数位（含字符的各数位和校验位）中，“**1**”的个数为奇数，如：

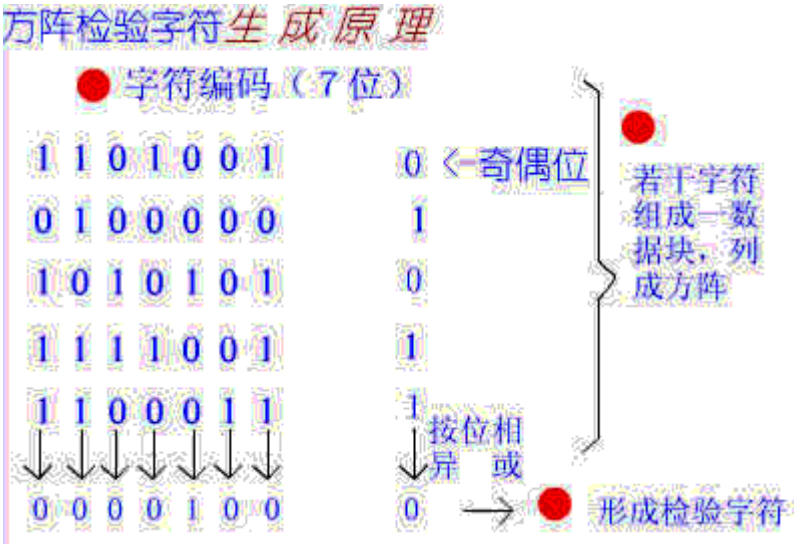
1 0110，0101

0 0110，0001

偶校验：所有传送的数位（含字符的各数位和校验位）中，“**1**”的个数为偶数，如：

1 0100，0101

0 0100，0001



奇偶校验能够检测出信息传输过程中的部分误码（1 位误码能检出，2 位及 2 位以上误码不能检出），同时，它不能纠错。在发现错误后，只能要求重发。但由于其实现简单，仍得到了广泛使用。

有些检错方法，具有自动纠错能力。如循环冗余码（CRC）检错等。

开发通信软件的技术与技巧

【提要】随着计算机应用领域的不断扩展，计算机之间的远程通信用得也越来越广泛，计算机间的远程通信所使用的通信软件，在市面上可以买到，但是通用的通信软件虽然能发送和接收文件，在许方情况下这些通信软件并不能满足实际工作的需要。本文就这一技术进行了探讨，介绍了有关的知识和技术，并由实例分步骤说明如何进行通信软件的开发。

一、前言

本文将以 VB5.0 所带的通信控件 MSCOMM 进行通信软件的开发，它有约 30 个属性和事件（可以从联机帮助中找到其使用资料）。本文简述它的主要属性及事件，并归类整理。

利用串行端口与调制解调器进行连接时，对于用 MSCOMM 控件编制通信软件来说，只需了解以下五根线的代号及作用。以下五根线的高电平/低电平状态分别对应 MScomm 控件的相应属性的 True/False 值。

(1)DTR 线：PC 发往 MODEM，表示 PC 机是否已准备好。

(2)RTS 线：PC 发往 MODEM，表示 PC 机是否允许 modem 发回数据.

(3)DSR 线:MODEM 发往 PC,表示 MODEM 是否已做好操作准备

(4)CTS 线:MODEM 发往 PC,表示 MODEM 是否允许发送数据

(5)CD 线:MODEM 发往 PC,表示 MOEDM 已经与呼叫的远方 MODEM 处于连结状态

二、MSCOMM 控件的属性

用 1,2,...表示串口 COM1,COM2....

设置或返回联接 MODEM 的串口的编号

Settings

例用"19200,N,8,1"表示传输速率为 19200bps,没有奇偶校验位，8 位数据位，1 位停止位。

设置或返回通信参数。

Handshaking

0 没有握手协议，不考虑流量控制。

1XON/XOFF,即在数据流中嵌入控制符来进行流控。
2RTS/CTS,既由信号线 **RTS/CTS** 自动进行流量控制（常用）。
3 两者皆可。
设置或返回硬件握手协议，指的是 **PC** 机 **MODEM** 之间为了控制流速而约定的内部协议。
PortOpen
True/False 可以打开**/**关闭端口。
打开或关闭端口。
OutBufferSize
传输缓冲区的字节数**,**如选 **1024**。
设置或返回传输缓冲区大小。
OutPut
Variant 型变量。
向传输缓冲区写数据流。
传输文本数据时**,**应将 **String** 型数据放入 **Variant** 变量，传输二进制数据**(**即按字节**)**时**,**应将 **Byte** 型数组数据放入 **Variant** 变量
InBufferSize
接收缓冲区的字节数**,**如选 **1024**。
设置或返回接收缓冲区大小。
InputMode
0 用 **Input** 属性接收文本型数据。
1 用 **Input** 属性接收二进制数据。
设置或返回接收数据的数据类型。
InBufferCount
Integer 型
返回接收缓冲区中已传到但还未取走的字符个数。
Input
当 **InputMode** 属性值为 **0**（文本模式）时**,**变量中含 **String** 型数据。
当 **InputMode** 属性值为 **1**（二进制模式）时**,**变量中含 **Byte** 型数组数据。
将接收缓冲区中收到的数据读入变量。
DTREnabled
RTSEnabled
DSRHolding
CTSHolding
CDHolding
均取值 **TRUE/FALSE**
用于读取或控制 **pc** 机与 **modem** 之间的交互状态。需运用好。例如，应在读取到 **DSRHolding** 属性值为 **TRUE** 时再向 **MODEM** 发出命令。应当在载波检测到以后**(CDHolding** 属性为 **TRUE)**时再向 **MODEM** 发送数据。

三、**MSCOMM** 控件的触发事件

MSCOMM 控件只使用一个事件 **OnComm**，用属性 **CommEvent** 的十七个值来区分不同的触发时机。主要有以下几个：

（**1**）**CommEvent=1** 时：传输缓冲区中的字符个数已少于 **Sthreshold(**可设置的属性值**)**个。

（2）CommEvent=2 时：接收缓冲区中收到 hreshold(可设置的属性值)个个字符，利用此事件可编写接收数据的过程。

（3）CommEvent=3 时：CTS 线发生变化。

（4）CommEvent=4 时：DSR 线发生变化。

（5）CommEvent=5 时：CD 线发生变化。

（6）CommEvent=6 时：检测到振铃信号。

另外十种情况是通信错误时产生，即错误代码。

四、通信软件程序实现

1、首先是通信参数设置，主要就是可以设置端口号，波特率，数据位，停止位，奇偶校验位及设置硬件握手协议，这些设置较为简单。

2、向 MODEM 发出 DTR(已准备好)信号，如下例程：

```
If MSComm1.PORTOPEN Then
MSComm1.DTREnable=True
Else
MSComm1.DTREnable=False
EndIf
```

3、打开时向 MODEM 发出一些命令来设置参数，其中 S0=n(n>=1)自动应答.n 为响铃次数；E0/E1 关闭/打开命令字符回应；Q0/Q1modem 返回/不返回结果码；M0/M1 关闭/打开 MODEM 扬声器，例程如下：

```
If MSComm1.PORTOPEN Then
Do While not MSComm1.CTSHolding : loop
Outstring="ATS0=1E1Q0M0"+Chr(13)
MSComm1.Output=Outstring
End if
```

4、进行拨号设计，需向 MODEM 发出 ATDT 命令，如下语句：

```
MSComm1.Output="ATDT"+Trim(“电话号码”)+Chr(13)
```

5、拨号以后发送数据文件，程序要循环等待并随时判定是否接通。如果 MODEM 向 PC 的回应字符串中含有"Connect"或 CDHolding 属性值变为 True(检测出载波)，则表示已与远方 MODEM 连机了，此时可以传输数据。

程序设计发送及接收程序时，需要以下定义：

```
S_FILENAME = "NAME" + Chr(5) + Chr(13) + Chr(10)
S_FILELEN = "LENTH" + Chr(5) + Chr(13) + Chr(10)
S_FILESEND = "BEGIN" + Chr(5) + Chr(13) + Chr(10)

Sub OpenFileToSend() '打开一个欲发送的文件
HSend = FreeFile
Open SENDFN For Binary As hSend 'SENDFN 中含有由用户选定的要传送的文件名。
LF&=LOF(hSend) '文件长度为 LF&
'开始发送文件名，文件长度,文件开始等信息字符串。
Dim Data as Vrait
Data = S_FILENAME
MSComm1.Output=Data '发出"FILENAME"文件名字串的提示信息
Data = SENDFN +Chr(13)+Chr(10)
```

```
MSComm1.Output=Data '发出文件名
Data = S_FILELEN
MSComm1.Output=DATA '发出"FILELEN"提示字串
Data = Trim(Str(LF&))+Chr(13)+Chr(10)
MSComm1.Output=Data '发出文件大小
Data = S_FILESTAR
MSComm1.Output=Data '发出"FILESTART"提示信息,表示下面文件开始。

Dim Sendarr() as byte '定义字节型数组
Sum=0 '记录累计发送的字节数
BSIZE=MSComm1.OutBufferSize '每次发送的块大小
ReDim Sendarr(1 To BSIZE) '重新定义读取缓冲
Do While Sum<LF& '循环发送
If LF&-Loc(hSend)<BSIZE Then
BSIZE=LF&-Loc(hSend)
ReDim Sendarr(1 To BSIZE)
End If
Get hSend ,, SENDARR '从文件取字节放入字节数组
Sendvar=Sendarr '转放到 Variant 型变量
'当 CTS 线及 CD 线为高电平时才可发送，否则需等待。
T=Timer+60
L:
If MSComm1.CTSHolding And MSComm1.CDHolding Then
MSComm1.Output=Sendvar '发送
Sum = Sum + BSIZE'累加计数
Else
If Timer < T Then
Go To L '循环等待
Else
Go To CLOSEFILE '等待时间超过 60 秒则退出
End If
End If
'等待系统处理完
Do
RET = DoEvents()
Loop Until MSComm1.OutBufferCount = 0
Loop '循环发送完毕。
```

6、接收 MODEM 送回的信息和数据文件，该过程是编写 mscomm1 控件的 OnCOMM 事件的处理程序来完成。为了接收文本类型的握手信号，通常使 InputMode 属性为文本模式。当发现接收到的字符串中有"FILESTAR"+Chr(5)+Chr(13)+Chr(10)，则将 InputMode 属性改为二进制模式。当文件内容接收完（由接收的字节数判断）再将 InputMode 属性改为文本模式。例程如下：

```
Private Static Sub MSComm1_OnComm()
```

```
Select Case MSComm1.CommEvent
Case comEvReceive '接收缓冲区收到 Rthreshold 个字符时触发
Dim DATA As Variant
Dim N As Long
Dim SJARR() As Byte
N=MSComm1.InBufferCount '接收缓冲区字符总数
If MSComm1.InputMode=0 Then'文本模式时，将收到的数据放到字符串变量。
MSComm1.InputLen=0
DATA = Space(N)
DATA = MSComm1.Input
Else '二进制模式时，将收到的数据放到字节数组。
ReDim S JARR(1ToN)
DATA=ARR
MSComm1.InputLen=N
DATA=MSComm1.Input
End If
'其它 case 情况略
End Select
End Sub
```

７、关于接收到的数据的处理例程：

```
Public Static Sub HandleData (Disp As Control，N As Long，DATA As Variant)
'参数：Disp(文本框，用于显示接受数据)
'N 为本次接收到的字节数
'DATA(接收到的数据.Variant 型)
If not mscomm1.InputMode=0 Then '接收的是字符串
Go To L2
End If
Disp.SelStart = Len ( Disp.Text)
Disp.SelLength = 0
Disp.SelText = DATA '显示字符数据
If InStr(1,Disp.Text,S_FILESTAR,0)=0 Then '若没有开始标志就结束此过程
ExitSub
EndIf
V_FILENAME = InStr(1,Disp.Text,S_FILENAME,0) '找文件名及文件长度
V_FILELEN = InStr(1,Disp.Text,S_FILELEN,0)
FN = Mid(Disp.Text,V_FILENAME+11,(V_FILELEN-V_FILENAME-13))
HJS = FreeFile'打开接收文件
JSFN = Pathc+"\SJFILE\S"+Trim(Str(NO))+"_"+FN

Open JSFN For Binary As HJS
V_FILENAME = InStr(1,Disp.Text,S_FILENAME,0)
V_FILELEN = InStr(1,Disp.Text,S_FILELEN,0)
FN = Mid(Disp.Text,V_FILENAME+11,(V_FILELEN-1)-(V_FILENAME+11))
```

```
FL = Mid(Disp.Text,V_FILELEN+11,V_FILESTAR-(V_FILELEN+10))
SENDLEN = Val(FL) '应收总字节数 SENDLEN

ReDim JSARR(0 To N-1)
JSARR=DATA '将字节流放入字节型数组
Puth JS , , JSARR ' 写入已打开的接收文件
JSLEN=JSLEN+N '本次已累计收到的字节数
Close HJS
End Sub
```

接口技术的基本知识

CPU 与外部设备、存储器的连接和数据交换都需要通过接口设备来实现，前者被称为 I/O 接口，而后者则被称为存储器接口。存储器通常在 CPU 的同步控制下工作，接口电路比较简单；而 I/O 设备品种繁多，其相应的接口电路也各不相同，因此，习惯上说到接口只是指 I/O 接口。

一、I/0 接口的概念

1.接口的分类

I/O 接口的功能是负责实现 CPU 通过系统总线把 I/O 电路和 外围设备联系在一起，按照电路和设备的复杂程度，I/O 接口的硬件主要分为两大类：

1）I/O 接口芯片

这些芯片大都是集成电路，通过 CPU 输入不同的命令和参数，并控制相关的 I/O 电路和简单的外设作相应的操作，常见的接口芯片如定时／计数器、中断控制器、DMA 控制器、并行接口等。

2）I/O 接口控制卡

有若干个集成电路按一定的逻辑组成为一个部件，或者直接与 CPU 同在主板上，或是一个插件插在系统总线插槽上。

按照接口的连接对象来分，又可以将他们分为串行接口、并行接口、键盘接口和磁盘接口等。

2.接口的功能

由于计算机的外围设备品种繁多，几乎都采用了机电传动设备，因此，CPU 在与 I/O 设备进行数据交换时存在以下问题：

速度不匹配:I/O 设备的工作速度要比 CPU 慢许多，而且由于种类的不 同，他们之间的速度差异也很大，例如硬盘的传输速度就要比打印机快出很多。

时序不匹配:各个 I/O 设备都有自己的定时控制电路，以自己的速度传 输数据，无法与 CPU 的时序取得统一。

信息格式不匹配:不同的 I/O 设备存储和处理信息的格式不同，例如可以分为串行和并行两种；也可以分为二进制格式、ACSII 编码和 BCD 编码等。

信息类型不匹配:不同 I／O 设备采用的信号类型不同，有些是数字信号，而 有些是模拟信号，因此所采用的处理方式也不同。

基于以上原因，CPU 与外设之间的数据交换必须通过接口来完成，通常接口有以下一些功能：

1）设置数据的寄存、缓冲逻辑，以适应 CPU 与外设之间的速度差异，接口通常由一些寄存器或 RAM 芯片组成，如果芯片足够大还可以实现批量数据的传输；

2）能够进行信息格式的转换，例如串行和并行的转换；

3）能够协调 CPU 和外设两者在信息的类型和电平的差异，如电平转换驱动器、数 / 模或模 / 数转换器等；

4）协调时序差异；

5）地址译码和设备选择功能；

6）设置中断和 DMA 控制逻辑，以保证在中断和 DMA 允许的情况下产生中断和 DMA 请求信号，并在接受到中断和 DMA 应答之后完成中断处理和 DMA 传输。

3.接口的控制方式

CPU 通过接口对外设进行控制的方式有以下几种：

1）程序查询方式

这种方式下，CPU 通过 I/O 指令询问指定外设当前的状态，如果外设准备就绪，则进行数据的输入或输出，否则 CPU 等待，循环查询。

这种方式的优点是结构简单，只需要少量的硬件电路即可，缺点是由于 CPU 的速度远远高于外设，因此通常处于等待状态，工作效率很低。

2）中断处理方式

在这种方式下，CPU 不再被动等待，而是可以执行其他程序，一旦外设为数据交换准备就绪，可以向 CPU 提出服务请求，CPU 如果响应该请求，便暂时停止当前程序的执行，转去执行与该请求对应的服务程序，完成后，再继续执行原来被中断的程序。

中断处理方式的优点是显而易见的，它不但为 CPU 省去了查询外设状态和等待外设就绪所花费的时间，提高了 CPU 的工作效率，还满足了外设的实时要求。但需要为每个 I / O 设备分配一个中断请求号和相应的中断服务程序，此外还需要一个中断控制器（I / O 接口芯片）管理 I / O 设备提出的中断请求，例如设置中断屏蔽、中断请求优先级等。

此外，中断处理方式的缺点是每传送一个字符都要进行中断，启动中断控制器，还要保留和恢复现场以便能继续原程序的执行，花费的工作量很大，这样如果需要大量数据交换，系统的性能会很低。

3）DMA（直接存储器存取）传送方式

DMA 最明显的一个特点是它不是用软件而是采用一个专门的控制器来控制内存与外设之间的数据交流，无须 CPU 介入，大大提高 CPU 的工作效率。

在进行 DMA 数据传送之前，DMA 控制器会向 CPU 申请总线控制 权，CPU 如果允许，则将控制权交出，因此，在数据交换时，总线控制权由 DMA 控制器掌握，在传输结束后，DMA 控制器将总线控制权交还给 CPU。

二、常见接口

1.并行接口

目前，计算机中的并行接口主要作为打印机端口，接口使用的不再是 36 针接头而是 25 针 D 形接头。所谓“并行”，是指 8 位数据同时通过并行线进行传送，这样数据传送速度大大提高，但并行传送的线路长度受到限制，因为长度增加，干扰就会增加，容易出错。

现在有五种常见的并口：4 位、8 位、半 8 位、EPP 和 ECP，大多数 PC 机配有 4 位或 8 位的并口，许多利用 Intel386 芯片组的便携机配有 EPP 口，支持全部 IEEE1284 并口规格的计算机配有 ECP 并口。

标准并行口 4 位、8 位、半 8 位:

4 位口一次只能输入 4 位数据，但可以输出 8 位数据；8 位口可以一次输入和输出 8 位数据；半 8 位也可以。

EPP 口（增强并行口）:由 Intel 等公司开发，允许 8 位双向数据传送，可以连接各种非打印机设备，如扫描仪、LAN 适配器、磁盘驱动器和 CDROM 驱动器等。

ECP 口（扩展并行口）:由 Microsoft、HP 公司开发，能支持命令周期、数据周期和多个逻辑设备寻址，在多任务环境下可以使用 DMA（直接存储器 访问）。

目前几乎所有的 586 机的主板都集成了并行口插座，标注为 Paralle1 或 LPT1，是一个 26 针的双排针插座。

2.串行接口

计算机的另一种标准接口是串行口，现在的 PC 机一般至少有两个串行口 COM1 和 COM2。串行口不同于并行口之处在于它的数据和控制信息是一位接一位串行地传送下去。这样，虽然速度会慢一些，但传送距离较并行口更长，因此长距离的通信应使用串行口。通常 COM1 使用的是 9 针 D 形连接器，而 COM2 有些使 用的是老式的 DB25 针连接器。

3.磁盘接口

1）IDE 接口

IDE 接口也叫做 ATA 端口，只可以接两个容量不超过 528M 的硬盘驱动器，接口的成本很低，因此在 386、486 时期非常流行。但大多数 IDE 接口不支持 DMA 数据传送，只能使用标准的 PCI / O 端口指令来传送所有的命令、状态、数据。几乎所有的 586 主板上都集成了两个 40 针的双排针 IDE 接口插座，分别标注为 IDE1 和 IDE2。

2）EIDE 接口

EIDE 接口较 IDE 接口有了很大改进，是目前最流行的接口。

首先，它所支持的外设不再是 2 个而是 4 个了，所支持的设备除了硬盘，还包括 CD－ROM 驱动器磁盘备份设备等。

其次，EIDE 标准取消了 528MB 的限制，代之以 8GP 限制。

第三，EIDE 有更高的数据传送速率，支持 PIO 模式 3 和模式 4 标准。

4.SCSI 接口

SCSI（SmallComputerSystemInterface）小计算机系统接口，在做图形处理和网络服务的计算机中被广泛采用 SCSI 接口的硬盘。除了硬盘以外，SCSI 接口还可以连接 CD－ROM 驱动器、扫描仪和打印机等，它具有以下特点：

＊可同时连接 7 个外设；

＊总线配置为并行 8 位、16 位或 32 位；

＊允许最大硬盘空间为 8.4GB（有些已达到 9.09GB）；

＊更高的数据传输速率，IDE 是 2MB 每秒，SCSI 通常可以达到 5MB 每秒，FASTSCSI（SCSI－2）能达到 10MB 每秒，最新的 SCSI－3 甚至能够达到 40MB 每秒，而 EIDE 最高只能达到 16.6MB 每秒；

＊成本较 IDE 和 EIDE 接口高很多，而且，SCSI 接口硬盘必须和 SCSI 接口卡配合使用，SCSI 接口卡也比 IED 和 EIDE 接口贵很多。

＊SCSI 接口是智能化的，可以彼此通信而不增加 CPU 的负担。在 IDE 和 EIDE 设备之间传输数据时，CPU 必须介入，而 SCSI 设备在数据传输过程中起主动作用，并能在 SCSI 总线内部具体执行，直至完成再通知 CPU。

5.USB 接口

最新的 USB 串行接口标准是由 Microsoft、Intel、Compaq、IBM 等大公司共同推出，它提供机箱外的热即插即用连接，用户在连接外设时不用再打开机箱、关闭电源，而是采用“级联”方式，每个 USB 设备用一个 USB 插头连接到一个外设的 USB 插座上，而其本身又提供一个 USB 插座给下一个 USB 设备使用，通过 这种方式的连接，一个 USB 控制器可以连接多达 127 个外设，而每个外设间的距离可达 5 米。USB 统一的 4 针圆形插头将取代机箱

后的众多的串/并口（鼠标、MODEM）键盘等插头。USB 能智能识别 USB 链上外围设备的插入或拆卸。　　除了能够连接键盘、鼠标等，USB 还可以连接 ISDN、电话系统、数字音响、打印机以及扫描仪等低速外设。

三、I/O 扩展槽

I/O 扩展槽即 I/O 信号传输的路径，是系统总线的延伸，可以插入任意的标准选件，如显示卡、解压卡、MODEM 卡和声卡等。通过 I/O 扩展槽，CPU 可对连接到该通道的所有 I／O 接口芯片和控制卡寻址访问，进行读写。

根据总线的类型不同，主板上的扩展槽可分为 ISA、EISA、MAC、VESA 和 PCI 几种。

1）ISA 插槽

黑色，分为 8 位、16 位两种。16 位的扩展槽可以插 8 位和 16 位的控制卡，但 8 位的扩展槽只能插 8 位卡。

2）EISA 插槽

棕色，外型、长度与 16 位的 ISA 卡一样，但深度较大，可插入 ISA 与 EISA 控制卡。

3）VESA 插槽

棕色，位于 16 位 ISA 扩展插槽的下方，与 ISA 插槽配合使用。

4）PCI 插槽

白色，与 VESA 插槽一样长，与 ISA 插槽平行，不需要与 ISA 插槽配合使用，而且只能插入 PCI 控制卡。由于主板的空间有限，PCI 插槽要占用 ISA 插槽的位置

一个单片机串行数据采集/传输模块的设计

摘 要　以 GMS97C2051 单片机为核心，采用 TLC2543 12 位串行 A/D 转换器，设计了一个串行数据采集/传输模块，给出了硬件原理图和主要源程序。

关键词　串行 A/D 转换器　串行数据传输　GMS97C2051 单片机

在微机测控系统中，经常要用到 A/D 转换。常用的方法是扩展一块或多块 A/D 采集卡。当模拟量较少或是温度、压力等缓变信号场合，采用总线型 A/D 卡并不是最合适、最经济的方案。这里介绍一种以 GNS97C2051 单片机为核心，采用 TLC2543 12 位串行 A/D 转换器构成的采样模块，该模块的采样数据由单片机串口经电平转换后送到上位机（IBM PC 兼容机）的串口 COM1 或 COM2，形成一种串行数据采集串行数据传输的方式。经实践调试证实：该模块功耗低、采样精度高、可靠性好、接口简便，有一定实用价值。

1 主要器件介绍

1.1 TLC2543 串行 A/D 转换器

模块采用 TI 公司的 TLC2543 12 位串行 A/D 转换器，使用开关电容逐次逼近技术完成 A/D 转换过程。由于是串行输入结构，能够节省 51 系列单片机 I/O 资源，且价格适中。其特点有：

（1）12 位分辨率 A/D 转换器；
（2）在工作温度范围内 10μs 转换时间；
（3）11 个模拟输入通道；
（4）3 路内置自测试方式；
（5）采样率为 66kbps；
（6）线性误差+1LSB（max）
（7）有转换结束（EOC）输出；
（8）具有单、双极性输出；
（9）可编程的 MSB 或 LSB 前导；

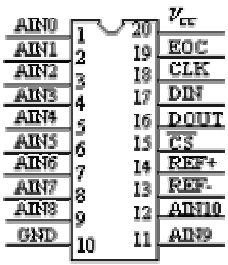


图 1　TLC2543 引脚

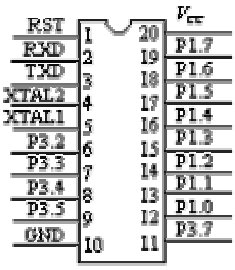


图 2　GMS97C2051 引脚

（10）可编程的输出数据长度。

TLC2543 的引脚排列如图 1 所示。图 1 中 AIN0～AIN10 为模拟输入端；$\overline{CS}$ 为片选端；DIN 为串行数据输入端；DOUT 为 A/D 转换结果的三态串行输出端；EOC 为转换结束端；CLK 为 I/O 时钟；REF+为正基准电压端；REF-为负基准电压端；V_{cc} 为电源；GND 为地。

1.2 GMS97C2051 单片机

GMS97C2051 是武汉力源公司和韩国 LG 公司联合推出的一种性能价格比极高的 8 位单片机，其指令系统与 MCS-51 系列完全兼容。GMS97C2051 与 AT89C2051 兼容(可直接替换)，但其性能价格比优于 AT89C2051。引脚排列如图 2 所示。

1.3 电平转换器 MAX3232

MAX3232 为 RS-232 收发器，简单易用，单+5V 电源供电，仅需外接几个电容即可完成从 TTL 电平到 RS-232 电平的转换，引脚排列如图 3 所示。

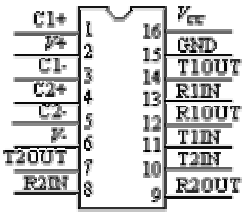


图 3　MAX3232 引脚图

2 硬件设计

硬件电路如图 4 所示。

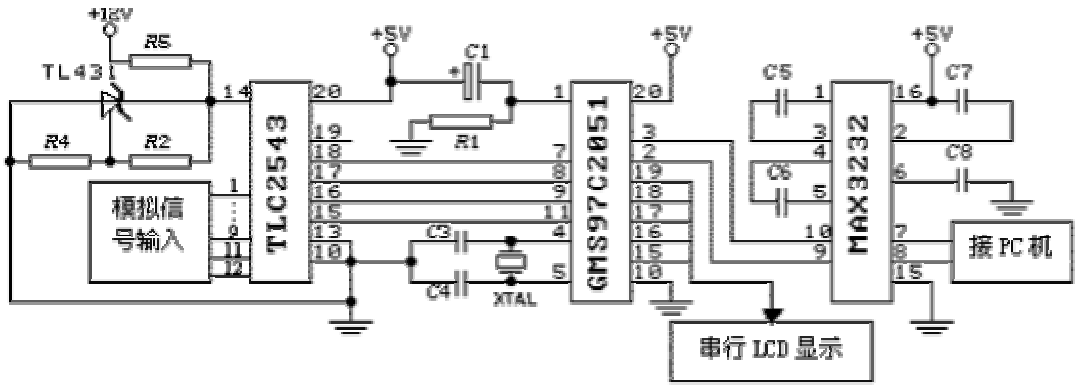


图 4　串行采集/传输电路原理图

单片机 GMS97C2051 是整个系统的核心，TLC2543 对输入的模拟信号进行采集，转换结果由单片机通过 P3.5（9 脚）接收，AD 芯片的通道选择和方式数据通过 P3.4（8 脚）输入到其内部的一个 8 位地址和控制寄存器，单片机采集的数据通过串口（3、2 脚）经 MAX3232 转换成 RS232 电平向上位机传输。图中串行 LCD 显示电路仅用于调试，对采集/传输的数据进行监测。

3 单片机软件设计

单片机程序主要包括串行数据采集模块“DATA_SAM”和串行数据传输模块“RS232”，调试所用到的显示子程序在此略去。

TLC2543 的通道选择和方式数据为 8 位，其功能为：D7、D6、D5 和 D4 用来选择要求转换的通道，D7D6D5D4=0000 时选择 0 通道，D7D6D5D4=0001 时选择 1 通道，依次类推；D3 和 D2 用来选择输出数据长度，本程序选择输出数据长度为 12 位，即 D3D2=00 或 D3D2=10；D1，D0 选择输入数据的导前位，D1D0=00 选择高位导前。

TLC2543 在每次 I/O 周期读取的数据都是上次转换的结果，当前的转换结果在下一个 I/O 周期中被串行移出。第一次读数由于内部调整，读取的转换结果可能不准确，应丢弃。

数据采集程序如下：

```
DATA_SAM:
    MOV     R0,#30H             ；数据缓冲区首地址 30H→R0
    MOV     R1,#00000000B       ；0 通道方式/通道数据
    ACALL   RD_AD               ；第一次读取的转换结果可能不准确,丢弃。
    MOV     R1,#00010000B       ；1 通道方式/通道数据
    ACALL   RD_AD               ；送 1 通道方式/通道数据并读第 0 通道转换结果
    MOV     @R0,R2              ；转换结果存放到数据缓冲区,下同
    INC     R0
    MOV     @R0,R3
    INC     R0
    MOV     R1,#00100000B       ；2 通道方式/通道数据
    ACALL   RD_AD               ；送 2 通道方式/通道数据并读第 1 通道转换结果
    MOV     @R0,R2
    INC     R0
    MOV     @R0,R3
    INC     R0
    ……                          ；其它通道操作方式类推
    RET
```

单片机通过编程产生串行时钟，并按时序发送与接收数据位，完成通道方式/通道数据的写入和转换结果的读出，程序如下，供数据采集模块“DATA_SAM” 调用。

```
    CLK     EQU     P3.3
    DIN     EQU     P3.4
    DOUT    EQU     P3.5
    CS      EQU     P3.7
RD_AD:
    CLR     CLK             ；清 I/O 时钟
    SETB    CS              ；设置片选为高
    CLR     CS              ；设置片选为低
    MOV     R4,#08          ；先读高 8 位
    MOV     A, R1           ；把方式/通道控制字放到 A
LOP1:
    MOV     C,DOUT          ；读转换结果
    RLC     A               ；A 寄存器左移，移入结果数据位，移出方式/通道控制位
    MOV     DIN,C           ；输出方式/通道位
    SETB    CLK             ；设置 I/O 时钟为高
    CLR     CLK             ；清 I/O 时钟
    DJNZ    R4,LOP1         ；R4 不为 0，则返回 LOP1
    MOV     R2,A            ；转换结果的高 8 位放到 R2 中
    MOV     A,#00H          ；复位 A 寄存器
    MOV     R4,#04          ；再读低 4 位
LOP2:
    MOV     C,DOUT          ；读转换结果
    RLC     A               ；A 寄存器左移，移入结果数据位
    SETB    CLK             ；设置 I/O 时钟为高
    CLR     CLK             ；清 I/O 时钟
```

```
    DJNZ   R4,LOP2          ；R4 不为 0，则返回 LOP2
    MOV     R3,A            ；转换结果的低 4 位放到 R3 中
    SETB   CS               ；设置片选为高
    RET
```

串行数据传输模块包括串行口初始化子程序和数据传输子程序，各子程序分别如下。其中数据传输采用查询方式，也可以方便地改为中断方式。

```
INIT_COM:
    MOV    SCON,#50H        ；串口方式 1 工作，8 位数据位，1 位停止位，无奇偶校验
    MOV    PCON,#80H        ；SMOD=1，波特率增倍
    MOV    TMOD,#20H        ；波特率设置，fosc=12MHz，波特率=2* 2400，N=0F3H
    MOV    TH1,#0F3H
    MOV    TL1,#0F3H
    SETB   TR1              ；启动定时器 T1
    RET
RS232:
    MOV    R0,#30H          ；缓冲区首地址 30H→R0
    MOV    R5,#22           ；发送数据长度→R5，11* 2=22
LOOP:
    MOV    A,@R0            ；取数据→A
    MOV    SBUF,A           ；数据→SBUF
WAIT:
    JBC    TI,CONT          ；判断发送中断标志，是 1 则转到 CONT，并清 TI
    SJMP   WAIT
CONT:
    INC    R0
    DJNZ   R5,LOOP
    RET
```

4 上位机串口接收程序设计

上位机接收数据所用 C 语言程序包括初始化子程序和接收子程序。各子程序分别如下：

```
void init_com1(void) /*初始化子程序*/
{
outportb(0x3fb,0x80); /*线控制寄存器高位置 1，使波特率设置有效*/
outportb(0x3f8,0x18); /*波特率设置，与单片机波特率一致为 4800bps*/
outportb(0x3f9,0x00);
outportb(0x3fb,0x03); /*线控制寄存器设置，8 位数据位，1 位停止位,无奇偶校验*/
outportb(0x3fc,0x03); /*Modem 控制寄存器设置,使 DTR 和 RTS 输出有效*/
outportb(0x3f9,0x00); /*设置中断允许寄存器，禁止一切中断*/
}
void receive_data(void) /*查询方式接收数据子程序*/
{
    while(!kbhit())
    {
            while(!(inportb(0x3fd)&0x01));/*若接收寄存器为空，则等待*/
            printf("%x ",inportb(0x3f8)); /*读取结果并显示*/
    }
    getch();
}
```

5 结论

本文给出的硬件和软件均经过实践检验，并且已经按照 PC/104 总线制作成数据采集卡，使用很方便，能够满足对数据采样频率要求不是特别高的应用场合。

参 考 文 献

1 TLC2543 模数转换器数据手册及应用笔记．武汉力源电子股份有限公司，1999
2 一九九九年产品目录（第一期）．武汉力源电子股份有限公司，1999
3 何立民．MS-51 系列单片机应用系统设计．北京：北京航空航天大学出版社，1999
4 NEW RELEASES DATA BOOK（Volume V）．MAXIM，1996：2-61～2-72

单工、半双工和全双工的定义

串行通讯的基本概念：与外界的信息交换称为通讯。基本的通讯方式有并行通讯和串行通讯两种。

一条信息的各位数据被同时传送的通讯方式称为并行通讯。并行通讯的特点是：各数据位同时传送，传送速度快、效率高，但有多少数据位就需多少根数据线，因此传送成本高，且只适用于近距离（相距数米）的通讯。

一条信息的各位数据被逐位按顺序传送的通讯方式称为串行通讯。串行通讯的特点是：数据位传送，传按位顺序进行，最少只需一根传输线即可完成，成本低但送速度慢。串行通讯的距离可以从几米到几千米。

根据信息的传送方向，串行通讯可以进一步分为单工、半双工和全双工三种。信息只能单向传送为单工；信息能双向传送但不能同时双向传送称为半双工；信息能够同时双向传送则称为全双工。

串行通讯又分为异步通讯和同步通讯两种方式。在单片机中，主要使用异步通讯方式。

MCS_51 单片机有一个全双工串行口。全双工的串行通讯只需要一根输出线和一根输入线。数据的输出又称发送数据（TXD），数据的输入又称接收数据（RXD）。串行通讯中主要有两个技术问题，一个是数据传送、另一个是数据转换。数据传送主要解决传送中的标准、格式及工作方式等问题。数据转换是指数据的串并行转换。具体说，在发送端，要把并行数据转换为串行数据；而在接收端，却要把接收到的串行数据转换为并行数据。

单工、半双工和全双工的定义

如果在通信过程的任意时刻，信息只能由一方 A 传到另一方 B，则称为单工。

如果在任意时刻，信息既可由 A 传到 B，又能由 B 传 A，但只能由一个方向上的传输存在，称为半双工传输。

如果在任意时刻，线路上存在 A 到 B 和 B 到 A 的双向信号传输，则称为全双工。

电话线就是二线全双工信道。 由于采用了回波抵消技术，双向的传输信号不致混淆不清。双工信道有时也将收、发信道分开，采用分离的线路或频带传输相反方向的信号，如回线传输。

-------->	<-------->	-------->
A---------B	A----------B	A---------B

		<--------
单工	半双工	全双工

从 RS232 端口获得电源

图 1 的电路从一个 RS-232 端口产生半稳压 5V 输出。与 PC 鼠标电源或依靠调制解调器控制信号 DTR 和 RTS 的可比较电路不同的是，该电路采用 3 线端口（GND,Rx 和 Tx)工作，并仅从 Tx 线获得功率。（除非高占空比时，Tx 线，RCV-232 在提供功率时仍然可供使用）。输出电流大约 8mA，对 CMOS 微控制器和其它低功耗电路足够了。IC1 是一个开关电容充电泵电压转换器，它既可以使输入电压反向，也可以使输入电压加倍。图中的电路连接提供倍压配置,使输入电压极性反转：正输入电压一般在 GND 和 OUT 之间连接，但是该电路却在 OUT 和 GND 之间连接一个负输入电压。IC 使负 Vin 在正向加倍，产生一个与 Vin 相等的正输出（在 Vdd 端）。

齐纳二级管 D1 用作并联稳压器，使 Vin“半稳压”至-5V(实际为-4.7V)。图中的 33uF 电容比一般的电容值要大一些，它在最坏传送模式（全零）下支持输出电压。例如，在 9600 波特时，零特性使输出电压下降大约 0.2V。对于更低的波特率，需要将 C1 换成一个更大的电容。

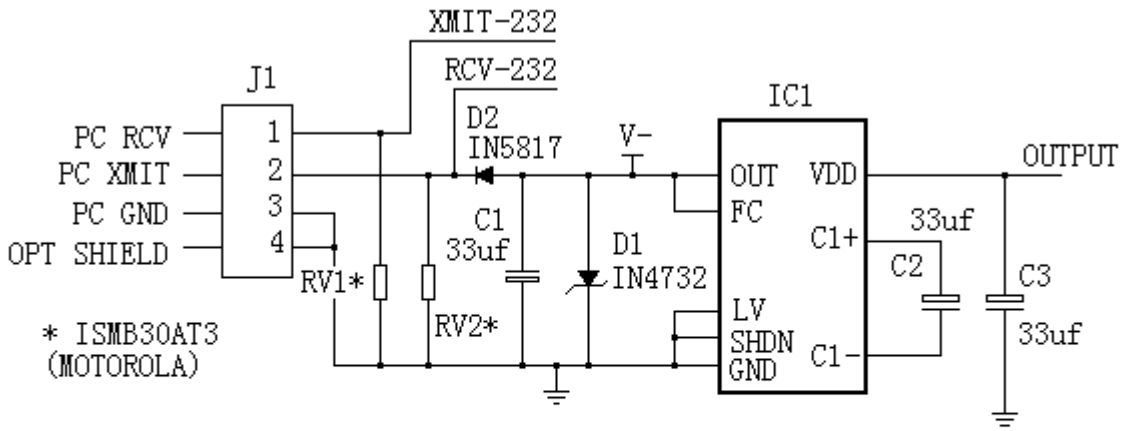


图1 工作在反极性输入电压的倍压模式，这一开关电容电压转换器在8mA时，从RS232端口的Tx线得到一个半稳压的5V电压。

串行同步通信的应用

摘要：该文给出利用 8251A 实现串行同步通讯设计的方法
关键词：串行同步 8251A 同步时钟 Modem

1 引言

在分布式测控系统中，上位机常常采用工业 PC 而工作站则用 STD/PC 总线工业控机，它们之间的数据通信很多采用串行异步方式，而串行同步方式则鲜为人用。在一次为用户开发 NEC 终端机仿真系统过程中为给系统提供同步通信模块，以 STD5221 通信板，配合 MultiModem224 调制解调器实现远程串行同步通信。（如图 1）

STD 总线 DTE DCE DTE STD 总线

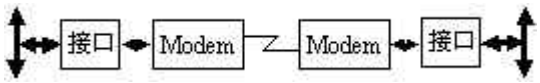


图 1

2 8251A 初始化

在不同系统或计算机之间进行数据通信主要采用并行和串行两种方式。8251A 是一种通用的同 / 异步接收 / 发送器(USART)。 在异步方式下的应用在有关书刊上已屡见不鲜，这里就不加重复。下面我们根据 8251A 芯片的使用体会对其在串行同步方式下的通信原理及应用进行着重介绍。在开始发送或接收之前，8251A 必须装入一组由 CPU 产生的控制字。这些控制信号定义了 8251A 的完整功能含义，并且必须紧跟在一个复位操作之后（内部的或外部的）。控制字分为两种格式：方式字和命令字。图 2 为定义 8251A 方式字和命令字设定的初始化与发送或接收数据流程图。

inital 为初始化程序的主要部分，方式字设置为止 3CH(内同步方式，双同步字符,8 位数据,奇校验方式) 。 命令字设置为 B7H（进入同步字符搜索方式，请求发送，接收就绪，数据端就绪，发送允许）。

```
inital proc ; 初始化程序.
: ; 清状态口和数据口.
mov dx,port ; port 为发送口/接收口地址
mov al,40h ; 复位操作，目的迫使 8251 out dx,al ; 进入方式字格式，
                                mov al,3ch ; 设置方式字操
                                作
out dx,al
mov al,55h ; 设置同步字符操作
out dx,al ; 同步字符 1 为 55h
out dx,al ; 同步字符 2 为 55h
mov al,0b7h ; 设置命令字操作
ret
inital endp
```

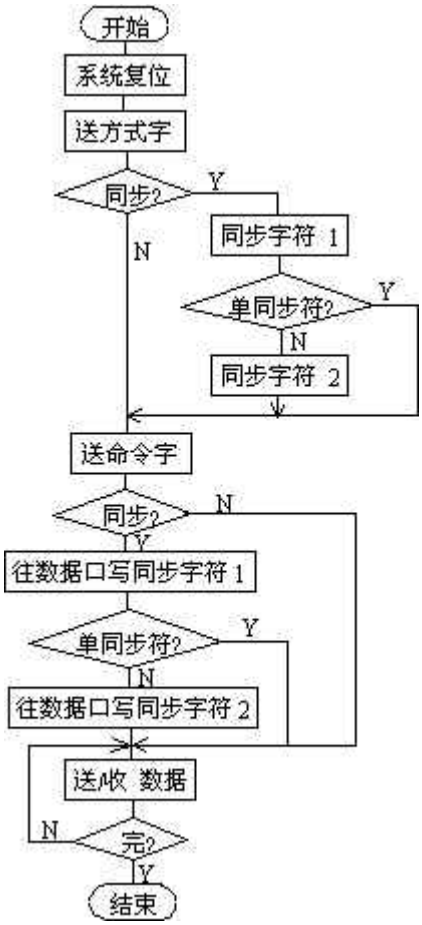


图 2

3 同步通信体验

(1) 时钟

在串行同步通信中，需要使发送的数据同时带有同步信息，因此，在硬件电路的设计中需要保证数据流中每一个连续不断的数据位均由一个基本时钟控制，并定时在某个特定的间隔上，所以对时钟要求甚严，即使两个工作站的通讯模板上 8251A 的晶振频率标称值相同，但实际上每个晶振的频率有所差别，8251A 时钟频率的误差将导致同步时钟相位的移动，离开要求的位置。为了保证进入同步后相位一直被锁定，我们将 Modem 置为同步方式，利用 Modem 的 RXC（接收时钟）和 TXC(传送时钟)作为 8251A 的接收／传送时钟，以此来达到传送时钟和接收时钟的同步。

另外 8251A 的 CLK 这个输入信号用作产生器件内部的定时,它的频率必须比 RXC 与 TXC 高 30 倍。

(2) 发送

8251A 被初始化完后,在 CPU 向 8251A 写一个字符启动发送前,TXD 输出端一直处于高平状态，作为 8251A 启动发送的第一个字符应是 SYNC(同步字符)符号。一旦启动了发送,TXD 输出端上的数据一定以 TXC 的频率连续不停地发送。在 CPU 不能及时向传送缓冲器写数之时,SYNC 符号将自动插入到 TXD 数据流中,以保持 TXD 上有数据连续不断的发送。

(3) 接收

同步接收有外同步和内同步两种方式（初始化时，在方式字中设定，本例设为内同步）。内同步方式下：

命令字的 ENTER HUNT 位置上的数据在 RXC 的上升没被采样 RXC 缓冲器与同步字符比较，直到相同为止（若 8251A 被设置为双 SYNC 方式，则要与第二同步字符比较）。8251A 结束 HUNT 搜索同步字符状态进入同步状态，处于字符同步中，然后把 SYNDET 引脚置为高电平。表明接收方已与发送方同步上。

外同步方式：

外同步方式是发送方接收主 SYNDET 脚施加一高电平的方法，迫使脱离 HUNT 方式，实现发送方与接收方的同步。

(4) MutiModem224 的接口芯脚分配

为使不同厂家的设备兼容，1969 年由电子工业协会（Electronic Industries Association)公布 RS-232-C 标准。最初拟制为终端设备和调制解调器之间的连接规定。它规定了两设备间的电器特性和所需连线的名称及编号。

在串行通讯链路中,将通信设备分为两类，以线“2”作为数据输出的通讯设备称为 DTE(Data Terminal Equipment)。象调制解调器将线“2”作为数据输入的通讯设备称为 DCE (Data Communication Equipment)。这样假如知道一个设备是 DTE,一个是 DCE,即可“2”到“2”,“3”到“3”的一一对应地将它们连接起来。这就是公认的直接连接。但厂家不一定总遵守这个规定，所以一个给出的通讯设备是 DTE,还是 DCE 并不能分清。因此在连接两个通讯设备时,最有效的方法是根据 RS232-C 出脚的名称,按实际应用需要相联。图 3 为调制解调器与通讯模板的 RS232-C 25 芯接口的连接图。

计算机首先发出数据终端就绪信号，然后指示调制解调器呼叫远程站，当调制解调器完成联通后，它就发出调制解调器就绪信号，通知计算机调制解调器已完成通讯准备，此时计算机就发出请求传送信号，等调制解调器应答了允许发送信号后，即开始数据传送。

4 控制字设置与软件实现

8251A 的引脚上有一“控制/信号”信号 C/D,此信号和“读/写”信号合起来通知 8251A 当前读写的是数据还是控制字.状态字。当 C/D=0 进行读写时,读出和写入的是数据。当 C/D=1 进行写入时,写入的是控制字、方式字和同步字符;C/D=1 进行读出时,是从状态寄存器中读出的状态。那么,在 C/D=1 写入时,到底写到哪一个寄存器呢?这涉及 8251A 初始化的有关约定。这个约定有三条: (1)芯片复位后,第一次用 C/D=1 写入的值是方式字;(2)如果方式字中规定了同部方式,接着用 C/D=1 写入的就是同部字符;(3)在此之后,以 C/D=1 写入的都被作为命令字。

图 3

从原则上来说，象 8251A 这样的 8 位接口芯片,连接在 16 位系统时,低 8 位的数据总写到偶地址，高 8 位的数据总写到奇地址. STD5221 也遵从这个原则,不将地址总线的最低位 A0 连到 8251A 的地址线上,而将地址总线 A1 作为 8251A 的地址最低位地址总线 A0 经过反相后连到 8251A 的 C/D 端.(在常见的具有 USART 的 PC 系统中,A0 是直接连接到 8251A 的 C/D 端,与 STD5221 相反,这一点在应用时要注意)。注：据实验结果,在 C/D=1 写入的第一个命令字之后，先向数据口写同步字符,才能启动同步发送(在流程图中有标示)。

5 通讯模板及程序说明

STD5221 是一种通用的串行数据通讯插件，它提供了两套完全独立的 RS232-C 串行数据通道。本例子在 STD V40 系统Ⅱ下开发，以 STD5221 作为通讯模板经过 Modem 以同步方式互发一串字符。限于篇幅未能完全收录。

参考文献

[1] 魏庆福.STD 总线工业控制机设计于应用
[2] 康拓公司.STD 总线工业控制机 双串行通信板（STD5221）
[3] MultiModem224E7 Owner’sManual
[4] 王仲文译.精通串行通信，电子工业出版社

串行通信波特率的一种自动检测方法

摘 要：给出了一种利用接收到的字符信息检测串行终端通信波特率 的方法。此方法简单、可靠、易行，并给出了实现这种检测方法的伪代码。

关键词：自动检测；波特率

串行通信是终端和主机之间的主要通信方式，通信波特率一般选择 1800、4800、9600 和 19200 等。终端的类型有很多种，其通信速率也有很多种选择。主机怎样确定终端的通信速率呢？本文给出了一种简单、易行的方法：设定主机的接收波特率（以 9600 波特为例），终端发送一个特定的字符（以回车符为例），主机根据接收到的字符信息就可以确定终端的通信波特率。本文对这种方法予以详述。

1 基本方法

回车符的 ASCII 值为 0x0D。串行通信时附加一个起始位和终止位，位的传输顺序一般是 先传低位再传高位。此时回车符的二进制表示方式为：

图 1 回车符的位序列

串行通信中一个二进制位的传输时间（记为 T）取决于通信的波特率，9600 波特时一个 二进制位的传输时间是 19200 波特时一个二进制位传输时间的两倍，即：$2*T_{19200}=T_{9600}$。因此，9600 波特时一个位的传输时间，19200 波特时可以传输两个位。同样地 ，9600 波特传输两个位的时间在 4800 波特时只能传送一个位。主机设定接收波特率为 9600， 终端只有也以 9600 波特发送的字符，主机才能正确地接收。发送波特

率高于或低于 9600 都会 使主机接收到的字符发生错误。接收波特率为 9600，终端以不同的波特率发送回车符时，主 机接收到的二进制序列如表 1 所示。

从表 1 中可以看出，除了 19200 和 1800 波特时两种特例情况，其他情形的二进制序列都是 9600 波特时二进制序列的变换。取前十个二进制位与 9600 波特时的二进制位相对应。忽略缺 少停止位‘1’引发的数据帧错误，把接收到的字符表示成字节方式（如表 1 的最右列所示） 。例如：在发送速率为 1200 波特，接收速率为 9600 波特时，主机得到的字节是 0x80，而不 是正确的回车符 0x0D。因为在不同的发送速率下（9600，4800，2400，1200）得到的字节 不同，所以通过接收字符的判定就可以确定发送波特率。

发送波特率为 19200 时，其发送速度正好是接收速度（9600 波特）的两倍，因此发送端 的两个二进制位会被接收端看作一个。取决于不同的串行接口硬件，‘01’和‘10’这两种 二进制位组合可能被认为是‘1’或者‘0’。幸运的是，只有 0～4 位存在这样的歧义问题， 后面的位因为都是停止位，所以都是‘1’。因此，发送速率为 19200 波特时接收到的字符其高半个字节为 0xF。低半个字节可能是多个值中的一个，但不会是 0x0，因为 0x0D 中有相邻 的两个‘1’，这就会至少在低半个字节中产生一个‘1’。因此，整个字节的形式为 0xF?， 且低半个字节不为 0。

表 1 不同波特率下的二进制序列

波特率	接收到的二进制位序列	字节表示
19200	0 1 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1	0xF?
9600	0 1 0 1 1 0 0 0 1	0x0D
4800	0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1	0xE6
2400	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1	0x78
1800	0 0 0 0 0 x 1 1 1 1 x 0 0 0 0 0 1 1 1 1	0xE0
1800	0 0 0 0 0 x 1 1 1 1 x 0 0 0 0 0 1 1 1 1	0xF0
1200	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0	0x80
600	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1	0x00
300	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0x00
150	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0x00
110	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0x00

发送速率为 1800 波特时，因为

$T_{1800}=T_{9600}*16/3$，

而 16/3 不是整数，接收端二进制位的状态转换时刻和 9600 波特不一一对应，引起在接收端 的一个位接收周期内有状态发生变化的可能。表 1 中给出的第六个位（表示为 x）就是这种情 况。因为 x 有可能被看作‘1’，也有可能被看作‘0’，所以发送速率为 1800 波特时接收到 的字节可能是 0xE0 或者 0xF0。波特率为 3600 和 7200 时也有同样的问题，也可以采用同样的方 法，但不确定的位数会增加，需要检测的字节种类也会更多。3600 波特和 7200 波特的传输速 率几乎不采用，因此这个问题并不严重。只要发送波特率在 1200～19200 之间，我们都可以 通过接收到的一个字符对此波特率进行唯一的判定。

2 低波特率的检测

当发送速率低于 1200 波特时，接收端收到的字节都是 0x00，因此只能确定其速率低于 12 00 波特，

而不可能再得到更多的信息。为了解决这个问题，可以在 9600 波特的速率下继续接 收下一个字节信息。发送速率为 600 波特或更低时，一个位的发送时间要大于 9600 波特时整 个字节的接收时间。因此，发送端每一个从‘1’（终止位）到‘0’（起始位）的跳变都会 让接收端认为一个新的字节开始了。表 2 所示为 600 波特或更低的传输速率时接收端回车符的 二进制序列（只给出开始的一些位）。

表 2 低波特率回车符的接收方式

波特率	9600 波特二进制序列	时间差（周期）	时间差（实时间）
600	16 0's 16 1's 16 0's	32	3.33ms
300	32 0's 32 1's 32 0's	64	6.66ms
150	64 0's 64 1's 64 0's	128	13.33ms
110	87 0's 87 1's 87 0's	174	18.13ms
75	128 0's 128 1's 128 0's	256	26.66ms
50	192 0's 192 1's 192 0's	384	4 0.00ms

600 波特时，第一个从‘1’到‘0’的跳变在初始化以后即刻发生。这个跳变让接收端 得到字节 0x00。第二个跳变在初始化(16+16)*T_{9600}秒以后发生，这会让接收端认 为另外一个字节开始接收了。一个二进制位的接收时间是 T_{9600}，所以串行接口电路 会在第一个跳变以后 10* T_{9600}秒提示第一个字节接收完毕，在(16+16+10)* T_{9600}秒以后提示第二个字节接收完毕。因此 600 波特时，第一个字节接收完毕和第二个字节 接收完毕的时间差是(16+16+10-10)* T_{9600}=32* T_{9600}秒。表 2 的第三列所示 是把这个时间差以 T_{9600}的个数表示。因为 T_{9600}=1/9600 秒=104.16 毫秒，相 乘可以得到两个字节接收完毕的实时间差。不同发送波特率的时间差如表 2 的最后一列所示 。有了这个时间差信息，就可以确定低传输速率时的波特率了：测定第一个和第二个字节的 接收时间差，然后在时间差常数表（表 2）里查出哪个波特率下的时间差与之最相近，对应 的就是终端发送波特率。即使测定的时间差有些误差，一般也可以正确地确定波特率。

3 实现方式

通过以上分析，各种波特率都可以通过回车符的发送和接收信息来测定，算法实现的伪 代码在本文的最后给出。应用实践证明了这种方法的有效性。

```
;  Pseudo code to determine what baud rate a transmitter is at,
                                                    on the b asis of a single

;  RETURN (0x0D) character received from it.

Initialise receive baud rate to 9600
Wait for Byte to be received
IF Byte = 0x00 THEN
      Start Timer
      REPEAT
      UNTIL (Timer > 50 ms OR New Byte Received)
      CASE Timer IN
        1 ms-4 ms:□ 600 Baud
        5 ms-10 ms:□ 300 Baud
```

```
        11 ms-15 ms:□ 150 Baud
        16 ms-22 ms:□ 110 Baud
        23 ms-32 ms:□ 75 Baud
        33 ms-49 ms:□ 50 Baud
            ELSE:□ Timed out; reset
    END CASE;
ELSIF Byte >= 0xF1 THEN
    □ 19200 Baud
ELSE
    CASE Byte IN
        0x0D:□ 9600 Baud
        0xE6:□ 4800 Baud
        0x78:□ 2400 Baud
    0xE0,0xF0:□ 1800 Baud
        0x80:□ 1200 Baud
        ELSE:□ Line noise; reset
    END CASE
END IF■
```

参考文献：

［1］赵依军等．单片微机接口技术［M］.北京： 人民邮电出版社，1989.

［2］刘利．软硬件技术参考大全［M］.北京： 学苑出版社，1993.

［3］张世一．数字信号处理［M］．北京：北京工业学院出版社，1987.

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。

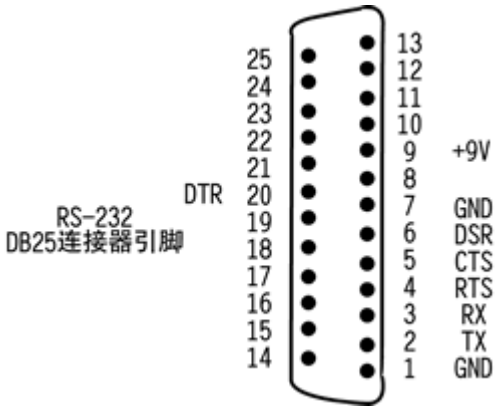


图 **1**

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**～**7k**Ω。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

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)	≥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。

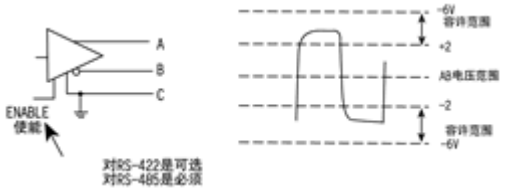


图 2

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。

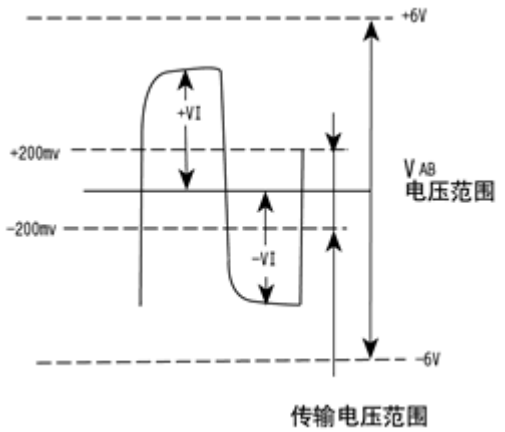


图 3

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 10×4k+100Ω（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）实现。

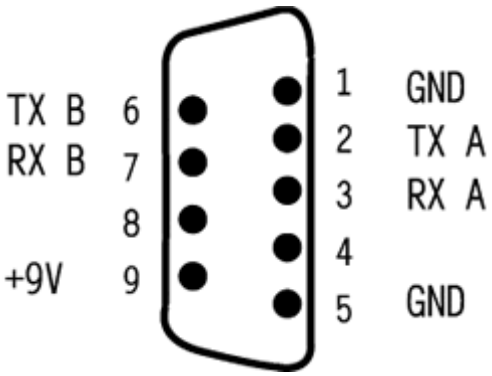


图 4

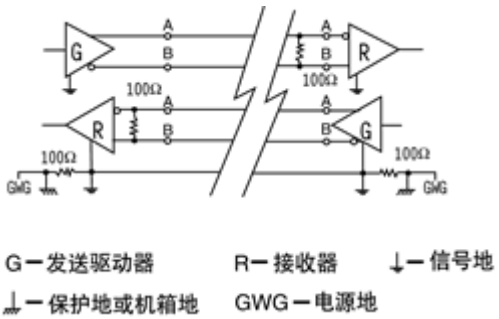


图 5

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。在矩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

3．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。

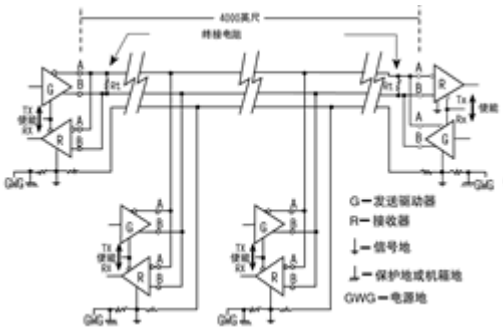


图 6

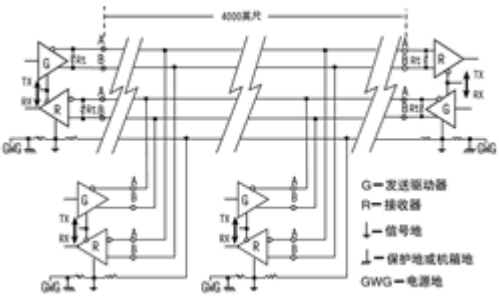


图 7

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 剑□鸣 S-422 是 4k 健；□旧峡梢运磚 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。

RS-485 需要 2 个终接电阻，其阻值要求等于传输电缆的特性阻抗。在矩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。终接电阻接在传输总线的两端。

四、RS-422 与 RS-485 的网络安装注意要点

RS-422 可支持 10 个节点，RS-485 支持 32 个节点，因此多节点构成网络。网络拓扑一般采用终端匹配的总线型结构，不支持环形或星形网络。在构建网络时，应注意如下几点：

1. 采用一条双绞线电缆作总线，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。图 8 所示为实际应用中常见的一些错误连接方式（a，c，e）和正确的连接方式（b，d，f）。a，c，e 这三种网络连接尽管不正确，在短距离、低速率仍可能正常工作，但随着通信距离的延长或通信速率的提高，其不良影响会越来越严重，主要原因是信号在各支路末端反射后与原信号叠加，会造成信号质量下降。

2. 应注意总线特性阻抗的连续性，在阻抗不连续点就会发生信号的反射。下列几种情况易产生这种不连续性：总线的不同区段采用了不同电缆，或某一段总线上有过多收发器紧靠在一起安装，再者是过长的分支线引出到总线。
总之，应该提供一条单一、连续的信号通道作为总线。

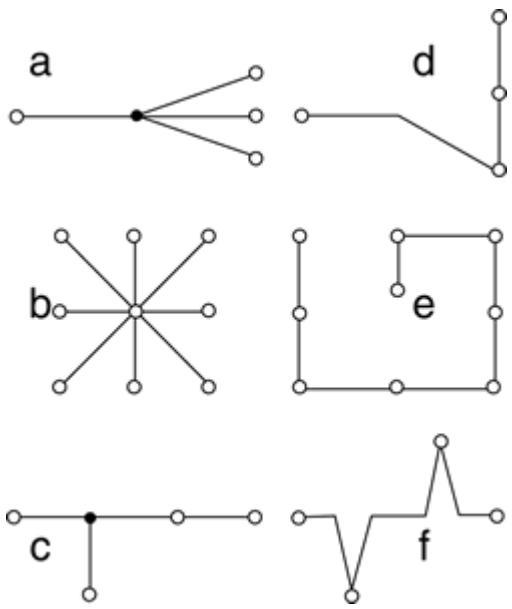


图 8

五、RS-422 与 RS-485 传输线上匹配的一些说明

对 RS-422 与 RS-485 总线网络一般要使用终接电阻进行匹配。但在短距离与低速率下可以不用考虑终端匹配。那么在什么情况下不用考虑匹配呢？理论上，在每个接收数据信号的中点进行采样时，只要反射信号在开始采样时衰减到足够低就可以不考虑匹配。但这在实际上难以掌握，美国 MAXIM 公司有篇文章提到一条经验性的原则可以用来判断在什么样的数据速率和电缆长度时需要进行匹配：当信号的转换时间（上升或下降时间）超过电信号沿总线单向传输所需时间的 3 倍以上时就可以不加匹配。例如具有限斜率特性的 RS-485 接口 MAX483 输出信号的上升或下降时间最小为 250ns，典型双绞线上的信号传输速率约为 0.2m/ns（24AWG PVC 电缆），那么只要数据速率在 250kb/s 以内、电缆长度不超过 16 米，采用 MAX483 作为 RS-485 接口时就可以不加终端匹配。

一般终端匹配采用终接电阻方法，前文已有提及，RS-422 在总线电缆的远端并接电阻，RS-485 则应在总线电缆的开始和末端都需并接终接电阻。终接电阻一般在 RS-422 网络中取 100Ω，在 RS-485 网络中取 120Ω。相当于电缆特性阻抗的电阻，因为大多数双绞线电缆特性阻抗大约在 100～120Ω。这种匹配方法简单有效，但有一个缺点，匹配电阻要消耗较大功率，对于功耗限制比较严格的系统不太适合。

另外一种比较省电的匹配方式是 RC 匹配，如图 9。利用一只电容 C 隔断直流成分可以节省大部分功率。但电容 C 的取值是个难点，需要在功耗和匹配质量间进行折衷。

还有一种采用二极管的匹配方法，如图 10。这种方案虽未实现真正的“匹配”，但它利用二极管的钳位作用能迅速削弱反射信号，达到改善信号质量的目的。节能效果显著。

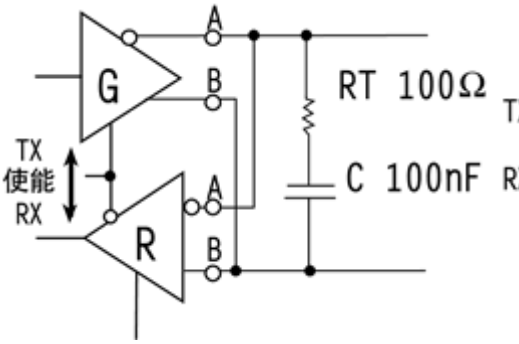


图 9

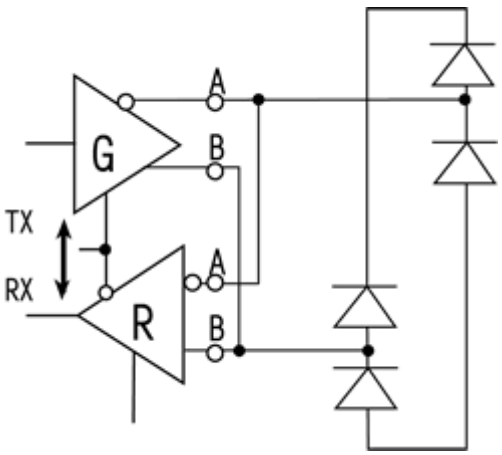


图 10

六、RS-422 与 RS-485 的接地问题

电子系统接地是很重要的，但常常被忽视。接地处理不当往往会导致电子系统不能稳定工作甚至危及系统安全。RS-422 与 RS-485 传输网络的接地同样也是很重要的，因为接地系统不合理会影响整个网络的稳定性，尤其是在工作环境比较恶劣和传输距离较远的情况下，对于接地的要求更为严格。否则接口损坏率较高。很多情况下，连接 RS-422、RS-485 通信链路时只是简单地用一对双绞线将各个接口的“A”、“B”端连接起来。而忽略了信号地的连接，这种连接方法在许多场合是能正常工作的，但却埋下了很大的隐患，这有下面二个原因：

1．共模干扰问题：正如前文已述，RS-422 与 RS-485 接口均采用差分方式传输信号方式，并不需要相对于某个参照点来检测信号，系统只需检测两线之间的电位差就可以了。但人们往往忽视了收发器有一定的共模电压范围，如 RS-422 共模电压范围为-7～+7V，而 RS-485 收发器共模电压范围为-7～+12V，只有满足上述条件，整个网络才能正常工作。当网络线路中共模电压超出此范围时就会影响通信的稳定可靠，甚至损

坏接口。以图 11 为例，当发送驱动器 A 向接收器 B 发送数据时，发送驱动器 A 的输出共模电压为 VOS，由于两个系统具有各自独立的接地系统，存在着地电位差 VGPD。那么，接收器输入端的共模电压 VCM 就会达到 VCM=VOS+VGPD。RS-422 与 RS-485 标准均规定 VOS≤3V，但 VGPD 可能会有很大幅度（十几伏甚至数十伏），并可能伴有强干扰信号，致使接收器共模输入 VCM 超出正常范围，并在传输线路上产生干扰电流，轻则影响正常通信，重则损坏通信接口电路。

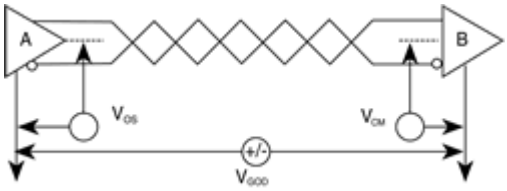


图 11

2.（EMI）问题：发送驱动器输出信号中的共模部分需要一个返回通路，如没有一个低阻的返回通道（信号地），就会以辐射的形式返回源端，整个总线就会像一个巨大的天线向外辐射电磁波。

由于上述原因，RS-422、RS-485 尽管采用差分平衡传输方式，但对整个 RS-422 或 RS-485 网络，必须有一条低阻的信号地。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条线（非屏蔽双绞线），或者是屏蔽双绞线的屏蔽层。这是最通常的接地方法。

值得注意的是，这种做法仅对高阻型共模干扰有效，由于干扰源内阻大，短接后不会形成很大的接地环路电流，对于通信不会有很大影响。当共模干扰源内阻较低时，会在接地线上形成较大的环路电流，影响正常通信。笔者认为，可以采取以下三种措施：

（1） 如果干扰源内阻不是非常小，可以在接地线上加限流电阻以限制干扰电流。接地电阻的增加可能会使共模电压升高，但只要控制在适当的范围内就不会影响正常通信。

（2） 采用浮地技术，隔断接地环路。这是较常用也是十分有效的一种方法，当共模干扰内阻很小时上述方法已不能奏效，此时可以考虑将引入干扰的节点（例如处于恶劣的工作环境的现场设备）浮置起来（也就是系统的电路地与机壳或大地隔离），这样就隔断了接地环路，不会形成很大的环路电流。

（3） 采用隔离接口。有些情况下，出于安全或其它方面的考虑，电路地必须与机壳或大地相连，不能悬浮，这时可以采用隔离接口来隔断接地回路，但是仍然应该有一条地线将隔离侧的公共端与其它接口的工作地相连。参见图 12。

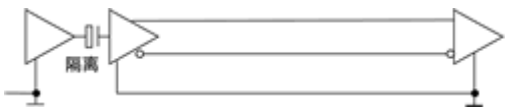
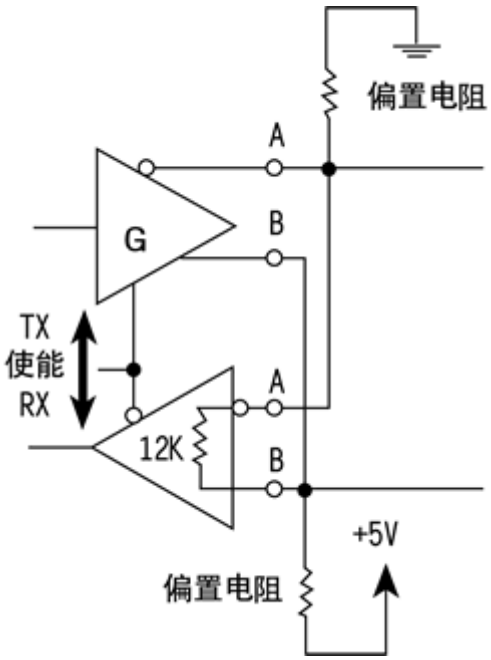


图 12

七、RS-422 与 RS-485 的网络失效保护

RS-422 与 RS-485 标准都规定了接收器门限为±200mV。这样规定能够提供比较高的噪声抑制能力，如前文所述，当接收器 A 电平比 B 电平高+200mV 以上时，输出为正逻辑，反之，则输出为负逻辑。但由于第三态的存在，即在主机在发端发完一个信息数据后，将总线置于第三态，即总线空闲时没有任何信号驱动总线，使 AB 之间的电压在-200～+200mV 直至趋于 0V，这带来了一个问题：接收器输出状态不确定。如果接收机的输出为 0V，网络中从机将把其解释为一个新的启动位，并试图读取后续字节，由于永远不会有停止位，

产生一个帧错误结果，不再有设备请求总线，网络陷于瘫痪状态。除上述所述的总线空闲会造成两线电压差低于 200mV 的情况外，开路或短路时也会出现这种情况。故应采取一定的措施避免接收器处于不确定状态。



通常是在总线上加偏置，当总线空闲或开路时，利用偏置电阻将总线偏置在一个确定的状态（差分电压≥-200mV）。如图 13。将 A 上拉到地，B 下拉到 5V，电阻的典型值是 1kΩ，具体数值随电缆的电容变化而变化。

上述方法是比较经典的方法，但它仍然不能解决总线短路时的问题，有些厂家将接收门限移到 -200mV/-50mV，可解决这个问题。例如 Maxim 公司的 MAX3080 系列 RS-485 接口，不仅省去了外部偏置电阻，而且解决了总线短路情况下的失效保护问题。

八、RS-422 与 RS-485 的瞬态保护

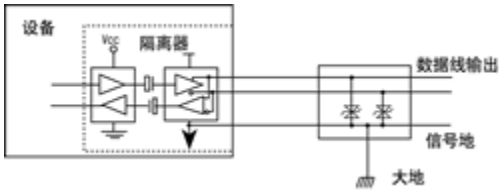
前文提到的信号接地措施，只对低频率的共模干扰有保护作用，对于频率很高的瞬态干扰就无能为力了。由于传输线对高频信号而言就是相当于电感，因此对于高频瞬态干扰，接地线实际等同于开路。这样的瞬态干扰虽然持续时间短暂，但可能会有成百上千伏的电压。

实际应用环境下还是存在高频瞬态干扰的可能。一般在切换大功率感性负载如电机、变压器、继电器等或闪电过程中都会产生幅度很高的瞬态干扰，如果不加以适当防护就会损坏 RS-422 或 RS-485 通信接口。对于这种瞬态干扰可以采用隔离或旁路的方法加以防护。

1. 隔离保护方法。这种方案实际上将瞬态高压转移到隔离接口中的电隔离层上，由于隔离层的高绝缘电阻，不会产生损害性的浪涌电流，起到保护接口的作用。通常采用高频变压器、光耦等元件实现接口的电气隔离，已有器件厂商将所有这些元件集成在一片 IC 中，使用起来非常简便，如 Maxim 公司的 MAX1480/MAX1490，隔离电压可达 2500V。这种方案的优点是可以承受高电压、持续时间较长的瞬态干扰，实现起来也比较容易，缺点是成本较高。

2. 旁路保护方法。这种方案利用瞬态抑制元件（如 TVS、MOV、气体放电管等）将危害性的瞬态能量旁路到大地，优点是成本较低，缺点是保护能力有限，只能保护一定能量以内的瞬态干扰，持续时间不能很长，

而且需要有一条良好的连接大地的通道，实现起来比较困难。实际应用中是将上述两种方案结合起来灵活加以运用，如图 **14**。在这种方法中，隔离接口对大幅度瞬态干扰进行隔离，旁路元件则保护隔离接口不被过高的瞬态电压击穿。



串口泵

[摘要]将 **RS232** 接口转换成双端平衡传送和差分接收方法，并对信号进行光电隔离，无需外接电源，实现延长 **RS232** 通讯距离和抗干扰保护接口之目的。

[关键词]平衡传送 差分接收 串口窃电 瞬态电压抑制

一、引言

串口泵也称为 **RS232** 隔离长线驱动器，是德阳四星电子技术开发中心研制的 **RS232/RS422/RS485** 系列产品之一，外形采用 **DB25** 转接盒，外插 **RS232** 串口，即插即用，使用极为方便。独特的串口窃电技术使得该 系列产品不需要外接电源，也不需靠初始化设置串口来供电，使用本系列产品后均不会对软件作任何修改，确保适合一切软件！该产品全部采用工业级元器件，并设有抗雷击等瞬态电压抑制电路。特别适用于工业现场，野外等恶劣环境，外形如图 **1** 所示：

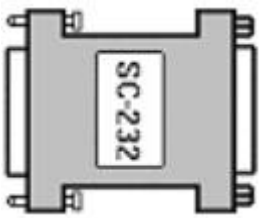


图 **1**

- 外形尺寸：**60×50×17**
- 重量：**40** 克
- 工作温度：**-40～+85**℃
- 工作湿度：**0～95%**

二、工作原理

采用 **RS232** 接口单端信号传送方式，通讯距离只能达到 **15** 米，很大程度上限制了计算机的应用范围。影响计算机正常通讯的因素主要有电磁辐射干扰（噪声干扰）和电源干扰（串扰）。电磁辐射干扰主要来自于工业现场的各种用电器，它在通讯电缆线上产生感应电势等噪声干扰使系统不能正常工作。电源干扰的主要原因是设备的电源电路不够理想，如两台设备所用的电源不同相；设备接地不好；或者两台设备处在不同的电力分布网，地电位不同等原因在两台设备之间产

生地电位差而形成串扰。

由图 **2** 可见：**Er=Et+En+Eg,Et** 为十几伏，**Eg** 一般为几十伏至几百伏，**En** 若是来自用电器一般不超过几十伏，若是来自雷电则可达几千伏。 **RS232** 接口所能承受的最大电压是 **25** 伏，由此可见，几十伏以上的干扰信号不仅会影响正常通讯而且足可将接口烧毁。

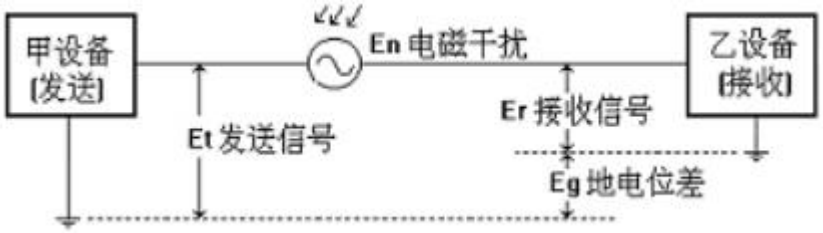


图 **2**

SC-232 系列光隔离长线收发器是将 **RS232** 接口数据传送方式转换成双端平衡信号传输，并用高速光隔离器件将两台设备隔离开，在差分接收器接收到双端平衡信号后再还原成 **RS232** 信号送至 **RS232** 接口。见图 **3** 所示：

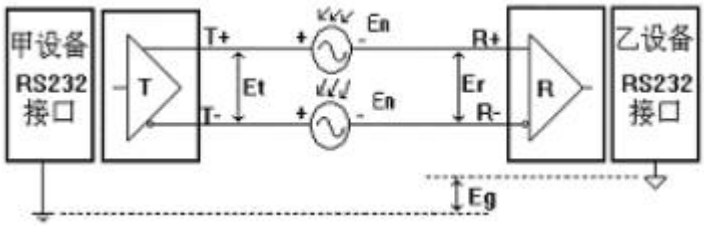


图 **3**

Et 为发送端电压，**Er** 为接收端电压，**En** 为干扰电压，**Eg** 为地电位差。

Er=Et+En-En=Et

由于采用双端平衡传输和差分接收，使得加在两根导线上的干扰信号可以相互抵消而不影响正常信号的传输，因而有效克服了噪声干扰。同时由于光隔离器的作用，使得两台通讯设备之间的地电位差 **Eg** 不会对信号有任何影响。从而彻底解决了通讯中的串扰和噪声干扰。

经实测，使用串口泵后在波特率为 **9600bps** 时通讯距离可达 **2** 公里，若降低通讯速率，则可相应延长通讯距离。

三、工作电源的窃取

本装置不需外接电源，其工作电源从 **RS232** 接口上窃取。如图 **4** 所示：图中 **TXD**、**RTS**、**DTR** 是 **RS232** 的输出信号，每根信号线可提供约 **9mA** 的输出电源，通讯时其电压在**+10V** 和**-10V** 之间跳变，通讯停止后其电压极性不定，但开机上电时全部为**-10V**。该三个信号经 **D1**～**D6** 全波整流和 **C1**、**C2** 滤波后得到**+9V** 和**-9V** 直流电压，若是开机上电状态则只有**-9**Ⅴ电压。**IC1** 和 **IC2** 分别是负电压到正电压转换器和正电压到负电压转换器，将输入电压转换极性并升压，这样无论 **TXD**、

RTS、**DTR** 是什么极性都能在 **DW1** 和 **DW2** 端得到＋９**V** 和－９Ｖ的稳定电压。亦即不需靠软件设置来得到工作电压，确保适合所有软件！其实只需以上任意一根信号线即可满足要求。

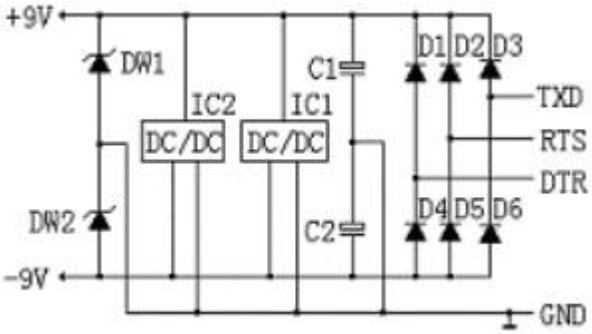


图 **4**

四、瞬态电压干扰接口保护

由于本装置采用平衡差分传输方式，因而具有良好的抗电磁干扰能力，光电隔离电路能有效的抑制地电位差的干扰。采用屏幕电缆对抑制射频干扰很有效，但屏蔽电缆线的电容较大，会相应降低通讯速率。除上述常见干扰因素外，瞬态电压干扰是威胁通讯接口安全的元凶之一。

瞬态电压抑制器 **TVS** 是一种高效能的电路保护器件，外形及符号同普通稳压管，所不同的是，它是一种特制的齐纳二级管，能经受高达数千伏的脉冲电压和数十乃至数百安培的浪涌电流，能承受的功率高达数千瓦。同时 **TVS** 具有极小的极间电容，并不会影响信号的正常传输。

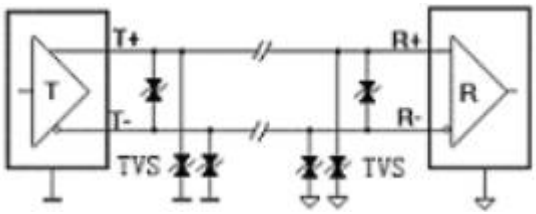


图 **5**

图 5 给出了本采用 **TVS** 的保护电路。在发送端和接受端各采用 3 个 **TVS** 单元，分别对线路之间、线路对地之间的瞬态电压干扰进行抑制，消除了由于强电进入而产生通讯口被烧毁的现象，从而有效的保证了整个通讯系统的安全运行。

五、结束语

串口泵可广泛用于各种工业控制系统、计算机通讯系统等一切采用 **RS232** 通讯的设备，对延长通讯距离、抗干扰和保护接口等方面不失为一种经济可靠的装置。根据以上原理，还可设计出 **RS232** 到 **RS422/RS485** 等各种通讯规程转换器，并且不需要外接电源。