The Data Communication Technology
of Single-chip Processor
for Both Beginners & Experts

单片机数据通信技术从入门到精通

[redacted] 主编
[redacted] 编著

单片机数据通信技术从入门到精通

石东海　　主编

扈啸　周旭升　　编著

西安电子科技大学出版社

2002

内 容 简 介

本书系统地介绍了单片机在数据通信方面的应用技术。第 1 章介绍了数据通信的基本概念及常见的通信媒质，第 2 章通过大量实例详细介绍了数据通信的调制与解调技术，第 3 章介绍了常用的编 / 解码技术，第 4 章着重介绍了单片机系统中常用的串行通信标准和接口技术，第 5 章介绍了 51 系列单片机之间通过标准串口通信的编程技术，第 6 章主要介绍了单片机与 PC 机之间的通信技术，包括在 Windows 环境下通过标准串口通信的编程技术，在 VB、VC、C++Builder 和 Delphi 等高级语言中实现串口通信的编程方法和参考程序，通过 PC 机标准键盘接口进行数据传输的技术，以及单片机同 PC 机并行传输数据的例子。

本书最大的特点是实用性强，其中很多实例可以直接拿来使用，极大地节省了设计人员的开发时间。

本书既可作为高等院校、培训班师生的教材，也可作为从事单片机应用技术人员的参考书。

图书在版编目（CIP）数据

单片机数据通信技术从入门到精通 / 扈啸等编著.

—西安：西安电子科技大学出版社，2002.11

(单片机应用技术设计与实例开发)

ISBN 7–5606–1173–7

Ⅰ. 单… Ⅱ. 扈… Ⅲ. 单片微型计算机–数据通信 Ⅳ. TN919

中国版本图书馆 CIP 数据核字(2002)第 068479 号

策　　划　毛红兵

责任编辑　钟宏萍

出版发行　西安电子科技大学出版社（西安市太白南路 2 号）

电　　话　(029)8227828　　邮　　编　710071

http://www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印　　刷　西安兰翔印刷厂

版　　次　2002 年 11 月第 1 版　　2002 年 11 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印张 19

字　　数　450 千字

印　　数　1～4 000 册

定　　价　27.00 元

ISBN 7 – 5606 – 1173 – 7／TP・0605

XDUP 1444001-1

前　言

随着信息时代的飞速发展，计算机应用技术日益渗透到社会生产生活的各个领域，家用电器的电脑化、计算机的电器化正逐渐成为计算机应用领域的下一个发展目标，在这一进程中，单片机起到了举足轻重的作用。以 51 系列单片机为代表的单片机家族也随之日益壮大起来，成为工业控制和家用电器智能化、微型化的排头兵，因此，如何最大限度地开发单片机的功能，如何提高其使用效能，成为了各个开发商关注的焦点。

如今，数据通信技术在各个领域得到了广泛的应用和发展，从工业控制、军事应用到医疗仪器、家用电器，到处都涉及到数据通信技术。在计算机主导工业生产并且日益走进家庭生活的今天，人们开始向更新的领域进军。控制智能化、仪器小型化、功耗微型化成为大家广泛关注的焦点，这就把单片机的地位提升到重要的地位，随之而来的单片机数据通信技术就成为新的技术焦点。掌握了单片机的数据通信技术也就是掌握了单片机的核心应用技术。

单片机数据通信技术的内容十分广泛，本书简单明了地介绍了有关通信的基本概念，以及单片机应用系统中常见的和最新的串行通信方式、串行接口芯片，结合大量工程应用开发的实例，向广大单片机学习者和开发者展现了实用的单片机数据通信技术。

本书具体结构如下：

第 1 章介绍数据通信的基本概念及常见的通信媒质。

第 2 章介绍数据通信的调制与解调技术，列举了大量实例。

第 3 章介绍常用的编 / 解码技术。

第 4 章重点介绍单片机系统中常用的串行通信标准和接口技术，列举了较多的例子来详细介绍 51 系列单片机同串行接口芯片通信的编程技术，包括了 I^2C 总线、SPI 总线、单总线、MPS 接口标准和 Microwire 接口标准等十多种常用芯片。

第 5 章介绍 51 系列单片机之间通过标准串口通信的编程技术。

第 6 章介绍单片机与 PC 机之间的通信技术，包括在 Windows 环境下通过标准串口通信的编程技术，在 VB、VC、C++Builder 和 Delphi 等高级语言中实现串口通信的编程方法和参考程序，通过 PC 机标准键盘接口进行数据传输的技术，以及单片机与 PC 机之间实现并行数据传输的例子。

石东海主持本书的全部编写工作，并认真审阅了全稿，本书前三章主要由周旭升编写，后三章主要由扈啸编写。张连超和张学成绘制了部分插图，王军和胡斌强搜集了部分资料。

另外，还要感谢本书所列参考文献的各位作者，由于他们的帮助才使我们的书稿变得更加完善。

由于水平有限，时间仓促，疏漏和不当之处在所难免，敬请广大读者批评指正！

编 者

2002 年 7 月

目　　录

第 1 章　数据通信基础

本章主要内容：

★ 数据通信概述
★ 通信协议
★ 数据传输模式
★ 数据通信系统的质量标准
★ 数据传输媒质

1.1　数据通信概述

1.1.1　数据通信概念及特点

数据通信是为了实现计算机与计算机或终端与计算机之间的信息交互而产生的一种通信技术，例如电报通信、计算机通信等。

典型的数据通信系统可用下面的等式描述，即

数据通信=数据处理+数据传输

由于数据通信是计算机之间的通信，所以它具有下列特点：

- 数据通信传输和处理离散的数字信号；
- 数据通信的通信速度很高，且通信量突发性强；
- 数据传输的可靠性要求高；
- 必须事先制定通信双方必须遵守的、功能齐备的通信协议；
- 数据通信的信息传输效率很高；
- 数据通信每次呼叫的平均持续时间短。

1.1.2　数据通信研究的内容

数据通信的功能是将数据从一点传输到另一点，而且数据是以某种方式编码和包装的。需要指出的是，两点间的数据传输必须是成功的，所以，数据是如何传输的，数据是如何编码的，数据通信是以什么样的规则去管理和控制的，所有这些问题都需要一一给予回答。为了简化，可以把数据通信划分为三个基本部分：

(1) 传输：解决如何为信息载体提供通路。研究适合传输的信号形式及相应的各种传输设备。

(2) 通信接口：主要研究接口的可靠性，解决怎样把发送端的信号变换为适合于传输的

形式，或者把传输到终点的信号变换为适合接收端设备接收的形式。

(3) 通信处理：通信处理是数据通信中最复杂的部分，本书只讲述其中一部分内容，大部分内容读者可参阅其它书籍。

1.1.3　数据通信系统的构成

数据通信系统是由数据终端设备和数据电路组成的，其构成示意图如图 1.1.1 所示。

数据源是数据的产生者或发送者，数据宿是数据的接收者和数据的终点，它们是各种类型的计算机或终端，统称为数据终端设备。

传输信道分为模拟信道和数字信道两种。模拟信道只能传输连续的模拟信号，如早期电话网等。数字信道传输离散的数字信号，即由“0”、“1”二进制码所构成的数字序列。

图 1.1.1　数据通信系统的构成示意图

数据信号要在信道上传输，必须采用信号变换设备。对模拟信道，信号变换设备即调制解调器，它把计算机或终端送来的数据信号变换为模拟信号再送往信道，或者反过来把信道送来的模拟信号变换成数据信号再送到计算机或终端。对数字信道，信号变换设备即接口设备，其作用是实现信号码型与电平的转换、信道特性的均衡、收发时钟的形成与供给以及码速控制等等。

1.2　通 信 协 议

1.2.1　通信协议的概念

数据通信是机器之间的通信，而大部分是利用数据通信网将若干台计算机连成计算机网络来实现的，所以数据通信也叫计算机通信。正由于数据通信是机器间的通信，所以和其它通信方式一样，应该在通信系统中规定一个统一的通信标准，即通信的内容是什么、如何通信、何时通信，都必须在通信的实体之间达成大家都能接受的协定，这些协定就被称为通信协议。也可将协议定义为监督和管理两个实体之间的数据交换的一整套规则。概括地说，通信协议是对数据传送方式的规定，包括数据格式定义和数据位定义等。

1.2.2　通信协议的内容和功能

1. 通信协议的内容

通信协议由下列三部分组成：

(1) 语法：规定通信双方彼此“如何讲”，即确定协议元素的格式，包括数据格式和信号电平等。如数据控制信息的结构或格式。

(2) 语义：规定通信双方彼此“讲什么”，即确定协议元素的类型和内容，包括用于相互协调及差错处理的控制信息。如规定通信双方要发出何种控制信息、执行什么动作和返回什么应答等。

(3) 定时关系：规定事件执行的顺序，即确定通信进程中通信的状态的变化，包括速度匹配和时序。如规定正确的应答关系等。

2. 通信协议的功能

由于数据通信是机器间的通信，因此通信协议应规范得十分详尽才能保证通信的正常进行，所以协议是一个复杂和庞大的通信规则的集合。其完成的功能主要有：

(1) 信号的传送与接收。应规定信息传送的格式、接口标准及启动控制、结束控制、超时控制等功能。

(2) 差错控制。使构成传输数据的源码或源码组具有一定的逻辑性，接收端根据收到的数据进行相应的检错和纠错操作。

(3) 顺序控制。对发送的信息进行编号，以免重复接收或丢失。

(4) 透明性。指对用户终端所使用的数据代码无任何约束性的限制，即对用户使用的代码保证编码的独立性与传输的透明性。

(5) 链路控制与管理。在全双工、半双工和多点线路等多种线路方式中，确定哪个站发送、哪个站接收、对多个用户同时呼叫的场合如何对其择优控制。

(6) 流量控制。为保证接收方和发送方在速率上的匹配而采用的方法。

(7) 路径选择。确定信息报文如何通过多个节点和链路到达目的节点的传送路径和最优的路径选择策略。

(8) 对话控制。指信息处理、信息安全和保密、应用服务等内容。

上述功能实际上应是从物理线路电气特性一直到各计算机进程之间共享资源的全部内容。要通过一个计算机网络来实现上述功能，绝非易事。将这种比较复杂的大任务分解成若干个较为简单的子任务，进而逐个加以解决，是简化工程设计中常用的方法。为此，通常采用结构化的设计方法，把上述这些总功能分解为多个子功能。

1.2.3　串行通信协议

串行通信协议包括同步协议和异步协议两种，本书只讨论异步串行通信协议。异步串行通信协议规定字符数据的传送格式，主要有下述内容：

(1) 起始位。通信线上没有数据被传送时处于逻辑“1”状态。当发送设备要发送一个字符数据时，首先发出一个逻辑“0”信号，这个逻辑低电平就是起始位。起始位通过通信线传向接收设备，接收设备检测到这个逻辑低电平后，就开始准备接收数据位信号。起始位所起的作用就是使设备同步，通信双方必须在传送数据位前协调同步。

(2) 数据位。当接收设备收到起始位后，紧接着就会收到数据位。数据位的个数可以是5、6、7、8 或 9 等，PC 机中经常采用 7 位或 8 位数据传送，8051 串行口采用 8 位或 9 位数据传送。这些数据位被接收到移位寄存器中，构成传送数据字符。在字符数据传送过程中，数据位从最低有效位开始发送，依次在接收设备中被转换为并行数据。

(3) 奇偶校验位。数据位发送完之后，便可以发送奇偶校验位。奇偶校验用于有限差错检测，通信双方应约定一致的奇偶校验方式。如果选择偶校验，那么组成数据位和奇偶位的逻辑“1”的个数必须是偶数；如果选择奇校验，那么逻辑“1”的个数必须是奇数。

(4) 停止位。在奇偶位或数据位(当无奇偶校验时)之后发送的是停止位。停止位是一个字符数据的结束标志，可以是 1 位、1.5 位或 2 位的低电平。接收设备收到停止位之后，通信线路上便又恢复逻辑“1”状态，直至下一个字符数据的起始位到来。

(5) 波特率设置。通信线路上传送的所有位信号都保持一致的信号持续时间，每一位的宽度都由数据传送速率确定，而传送速率是以每秒多少个二进制位来度量的，这个速率叫波特率。例如，如果数据以每秒 300 个二进制位在通信线路上传送，那么这个传送速率就为 300 波特。

1.3　数据传输模式

1.3.1　串行和并行传输

并行传输是构成字符的二进制代码在并行信道上同时传输的方式。并行传输时，一次传输一个字符，如微机内部总线上的数据码或地址码，收发双方不存在同步问题，而且速度也快，这是并行传输的优点。但是并行传输需要并行信道，所以线路投资大，不适合于远距离传输。图 1.3.1(a)所示为 8051 单片机与外设间 8 位数据并行通信的连接方法。

串行传输是构成字符的二进制代码序列在一条信道上以位(码元)为单位、按时间顺序且按位传输的方式。串行传输时，发送端按位发送，接收端按位接收，同时还要对所传输的字符加以确认，所以收发双方要采取同步措施，否则接收端将不能正确区分出所传输的字符，失去了通信的意义。串行传输的速度虽慢，但是只需要一条传输信道，且线路投资少，易于实现，所以是计算机通信采用的主要传输方式。图 1.3.1(b)所示为串行数据通信方式的连接方法。

图 1.3.1　两种通信方式连接示意图

计算机的并行数据信号变为串行数据信号，是由通信控制器完成的。通信控制既可由软件实现，也可由硬件实现。一般来说，硬件实现较为容易，由通信接口板(通信适配器)来完成。

1.3.2　异步和同步传输

1. 异步传输方式

异步传输方式是字符的异步传输技术。在异步传输方式下，传输数据以字符为单位。当发送一个字符代码时，字符前面要加一个“起”信号，其长度为 1 个码元，极性为“0”，即空号极性；字符后面要加一个“止”信号，其长度为 1、1.5 或 2 个码元(在国际 NO.2 码时用 1.5 码元长)，极性为“1”，即传号极性。加上“起”、“止”信号后，即可区分出所传输的字符。传送时，字符可以连续发送，也可以单独发送，不发字符时线路保持“1”状态。如图 1.3.2 所示，每个字符由 8 比特(位)组成，加“起”、“止”信号共有 11 位长(“止”信号为 8 位)，两字符之间的间隔长度可以不确定。异步传输方式适用于 1200 b/s 以下的低速传输，且实现起来比较简单。

图 1.3.2　起止式同步

异步传输方式的优点是收发双方不需要严格的位同步(因为每个字符开始时都要重新启动，位定时误差不会积累)，缺点是每一个字符都要加“起”、“止”信号，传输效率比较低。

2. 同步传输方式

同步传输方式即为位同步传输技术。在同步传输方式下，收发双方必须建立准确的位定时信号，正确地区分每位数据信号。在该方式中，每个字符不增加任何附加位，而是连续发送。但是在传输中，数据要分成组(或帧)，一组含多个字符代码或多个独立码元。为使收发双方建立和保持同步，在每组的开始和结束需加上规定的码元序列，作为标志序列。在发送数据之前必须先发送该标志序列，接收端通过检测出该标志序列来实现同步。

标志序列的码型因传输规程不同而异。例如在基本型传输规程中，利用国际 NO.5 代码中的“SYN”控制字符，可实现收发双方的同步。又如在高级数据链路控制规程(HDLC)中，是按帧格式传送，利用帧标志符“01111110”来实现收发双方的同步。

同步传输方式适用于 2400 b/s 以上的数据传输，不需加起、止信号，因此传输效率高，但是实现起来比较复杂。

1.3.3　双工通信

双工通信方式是对相互通信的两台通信设备(如计算机)间数据流向的描述，或者说是对一台通信设备执行收发操作能力的描述。

“双工”包括“全双工”、“半双工”和“单工”。双工通信方式及其链路结构如图 1.3.3 所示。

图 1.3.3　双工通信方式及链路结构

1. 全双工方式

全双工方式指相互通信的两台通信设备可以同时发送和接收数据，即数据同时可在两个方向上传送，所以它们之间至少需要两条通信线路。

2. 半双工方式

半双工方式指两台相互通信的通信设备均具有收发数据的能力，但在某一时间内它们只能执行一种操作(收或发)，不能同时执行收发两种操作；相应地，在它们之间的通信线路上可在两个方向上传输数据，但在某一时间内却只能在一个方向上传送数据。

为实现半双工通信，两台通信设备间只需一条通信线路，但各通信设备必须配备收发切换开关。

3. 单工方式

单工方式指两台通信设备间数据只能在一个方向上传送。在单工方式下，两台通信设备中一台为发送设备，另一台必为接收设备，它们之间有一条通信链路就够了。

1.4　数据通信系统的质量标准

数据通信和其它通信方式一样，首先要满足快速性和准确性的指标，在评价数据通信的质量时，主要用传输速率来评价快速性，用误码率来评价其准确性。至于其可靠性、有效性和经济性的有关指标也将一并介绍。

1.4.1　传输速率

传输速率是衡量数据通信系统通信能力的指标。它表征了单位时间内传送的信息量，在数据传输中，通常采用调制速率、数据信号速率和数据传输速率来表示。

1. 调制速率

调制是将基带的数字脉冲信号变换成适合于在线路上传输的某一频率载波信号的过程，我们把调制过程中单位时间内调制信号波形的变换次数，也就是单位时间内所能调制的调制次数，简称波特率。如果一个单位调制信号波的时间长度为 T (s)，则调制速率为：

$$R_B=\frac{1}{T} \tag{1.4.1}$$

在数据通信中，单位调制信号波又叫做码元，故调制速率又称作码元传输速率。

2. 数据传送速率

数据传送速率表示单位时间内通过信道的信息量，单位是比特/秒，用 b/s 表示。这是用来表示传输速率最常用的单位，简称比特率。在串行通信中定义数据传送速率为：

$$R_b=R_B\log_2 M=\frac{1}{T}\log_2 M \tag{1.4.2}$$

式中，R_B 为波特率，M 为调制信号波的状态数，T 为单位调制信号波的时间长度。

若为并行传输，则在相应的波特率 R_B 前乘以并行传输的通路数即可，在速率较高的情况下，可用千比特/秒(kb/s)或兆比特/秒(Mb/s)来表示。

调制速率(波特率)和数据传送速率(比特率)之间有一定的关系，在串行传输两状态调制中，二者的速率是相等的，即一个码元只携带一位二进制数的信息。但在实际的调制中，为提高信道的利用率和抗干扰性能，经常采用多进制或称多状态的调制(M 状态调制)，这时单位调制信号波里就包含了 $\log_2 M$ 个多比特信息，此时二者速率就差了 $\log_2 M$ 倍。

3. 信息传输速率

信息传输速率是数据通信系统中的数据源和数据宿之间单位时间传送的数据量，数据量的单位可以是比特、字符码组等，时间单位可以是秒、分、小时等，通常以 b/s 为单位。如果用 N 表示数据宿接受的信息比特数，t 表示传输 N 比特信息所需的时间，则信息传输速率为：

$$R_M=\frac{N}{t} \tag{1.4.3}$$

1.4.2　误码率

数据通信的目的在于使接收端获得正确的数据量，因此接收端数据的差错程度是数据通信质量的最重要的指标，一般可用误码率、误字率和误组率表示，通常用误码率表示。误码率计算公式为：

$$P_e=\frac{\text{接收差错的比特数}}{\text{总的传输比特数}} \tag{1.4.4}$$

为了降低误码率，不得不采用差错控制手段，以保证数据通信中系统的误码率 $P_e\leq 10^{-8}$。

1.4.3　可靠度

可靠度是衡量机器正常工作能力的一个指标，可用公式表示为：

$$P_r = \frac{\text{系统正常工作时间}}{\text{系统工作总时间}} \times 100\% \qquad (1.4.5)$$

影响可靠度的因素很多，主要是组成系统设备的可靠性、信道的质量、操作人员的水平和工作状态等。

1.4.4 功率利用率和频带利用率

在满足数据通信快速性、准确性的基础上，系统应追求较高的有效性，以提高系统的效率。其中功率利用率和频带利用率是从不同侧面来反映系统有效性的指标。

1. 功率利用率

功率利用率以比特差错率小于某一规定值时所要求的最低归一化信噪比(每比特的信号能量和噪声单边功率谱密度的比值)衡量。所要求的信噪比越低，则功率利用率越高。

2. 频带利用率

频带利用率是描述数据传输速率和带宽之间关系的一个指标，也是衡量数据通信系统有效性的指标，它是单位频带内所能传输的信息速率，其表示式为：

$$?_B = \frac{R_b}{B} = \frac{1}{BT}\log_2 M \ \ (\text{b/s}\cdot\text{Hz}) \qquad (1.4.6)$$

式中，B 是系统频带宽度，R_b 是系统的比特率，M 是调制的状态数。在频带宽度相同的条件下，比特传输速度越高，频带利用率也越高，反之则越低。

功率利用率和频带利用率这两项性能指标主要都决定于调制解调方式，在选择调制解调方式时应兼顾二者。如果在某些系统中主要功率受限，则可适当牺牲频带利用率来提高功率利用率；若主要频带受限，则着重提高频率利用率，而功率利用率可适当降低。

1.4.5 标准性

系统的标准性是缩短研制周期，降低生产成本，便于用户选购，便于维护的重要措施。特别是随着计算机通信网的日益发展，更显示出标准性的重要意义。我国信息产业部的有关机构也专门从事标准的研究和制定工作，这是发展我国数据通信不可缺少的。

1.4.6 通信建立

通信建立时间应尽可能短，它是反映系统同步性能的一个指标，对于间歇式的数据通信或瞬时通信来说，这项指标尤为重要。

1.4.7 标

其它指标还有经济性、操作简单、维修方便、能自动检测、体积小和重量轻等，在设计传输系统时也是要注意的。当然，这些指标也是相对的，要根据具体情况以及周围环境和服务对象等具体确定。

1.5　数据传输媒质

传输媒质是通信中实际传送信息的载体。数据通信系统中采用的传输媒质可分为有线和无线两大类。双绞线、同轴电缆、电力线、电话线、波导管和光纤是常用的几种有线传输媒质。短波通信、微波通信、卫星通信、红外通信、激光通信、散射通信以及蓝牙通信等的信息载体都属于无线传输媒质。

1.5.1　有线传输媒质

有线传输媒质是现代通信中最常用的媒质之一，它以有形的线路为传输介质，主要包括双绞线、同轴电缆和光纤。计算机间的远程数据通信一般要采用调制解调器，其传输介质本应使用专用的同轴通信电缆，但从成本角度考虑，还可使用双绞线、电话线和电力线。当然，采用光纤作为通信介质是最好不过了，但成本较高，可根据传输特性(包括其容量及传输频率范围)、连接性(点对点或多点)、地域范围(网络上点与点之间的最大距离)、抗干扰性(介质对干扰的屏蔽能力)、成本(包括组成部件、安装和维修成本等)几个方面对上述几种介质进行比较。下面便对这几种通信介质分别进行分析说明。

1. 双绞线

双绞线是由两条互相绝缘的铜导线扭绞起来构成的，一对线作为一条通信线路，其构成如图 1.5.1 所示。通常类似这样的一定数量的导线对捆成一个电缆，外面包上硬护套。之所以采用这种扭绞结构是为了减少相邻导线的电磁干扰，以提供相对稳定的导电特性。最近的研究结果表明，双绞线作为一个建筑物内微机局部网络的传输介质是有效且成本低廉的。

双绞线可用于传输模拟及数字信号，其通信距离一般为几公里到几十公里。当距离太长时，对于模拟信号，每隔 5～6 km 需加放大器以便将衰减了的信号放大到合适的数值；对于数字信号，每隔 2～3 km 需加转发器(中继器)以便将失真了的数字信号进行整形。导线越粗，其通信距离越远，但导线价格会越高。

图 1.5.1　双绞线的构成

与其它传输介质相比，双绞线的传输距离、带宽和数据率有限。当频率增大时，信号衰减也增大。此外它易于和电磁场耦合，对噪声和干扰较敏感。减少损耗的办法有两种：第一，可在双绞线外面加上一个金属编织网的屏蔽层减少干扰，相邻的线对采用不同扭绞长度减少串音；第二，使用平衡传输线，接收端用相位差判断数字 0 和 1，而不再用幅度差判断，因而可有效降低加性噪声干扰，增加传输距离。

对于点到点模拟信号传输，双绞线可以达到 250 kHz 的带宽。由于用户线路的衰减为每公里 1 dB，而电话线通用标准为最大损耗低于 6 dB，因此，电话线上每 6 km 内必须接放大器。对于数字的点到点线路，双绞线可以达到几 Mb/s 的数据率，且传输距离可达几公里。

由于双绞线成本低廉且性能较好，因而无论是在模拟还是数字数据通信中都是一种普遍采用的传输介质。目前，在某些专门系统中，双绞线在短距离传输中的速率已达 100～155 Mb/s。

2. 同轴电缆

同轴电缆也像双绞线那样由一对导体组成，但它们是按同轴的形式构成线对，其结构如图 1.5.2 所示。其中最里层是内导体，外包一层绝缘材料，外面再套一个空心的圆柱形外导体，最外层是起保护作用的塑料外皮。内导体和外导体构成一组线对。单根同轴电缆直径约为 0.5～2.5 cm，几个同轴电缆往往会在一个大电缆内，有些里面还装有二芯扭绞线或四芯线组用于传输控制信号。由于外导体是接地的，故同轴电缆具有很好的抗干扰性。

图 1.5.2　同轴电缆的构成

同轴电线与双绞线相比价格稍贵，因其具有带宽宽、数据传输速率高、传输距离长、抗干扰能力强等优点，尽管面临光纤、微波和卫星等传播信道的竞争，但仍是用途非常广泛的传输介质。由于它比双绞线具有优越的频率特性，现已被广泛用于较高速率和较高频率的数据传输，如长距离电报、电话传输，有线电视、局部网络和短距离系统连接的通信线路中。

3. 电力线

利用电力线载波实现计算机间的数据通信，将大大降低成本、缩短工时。下面举例说明电力线载波通信的原理。

可以单片机 8031 为控制核心，控制双音多频 DTMF 发生器(MT5087)，实现信号编码，再通过锁相环电路(LM567)进行信号调制。载波信号经放大后，耦合至电力线，完成数据发送。接收端将电力线上耦合进来的载波信号，经锁相环电路进行解调，再送入 DTMF 译码器(MT8870)，实现信号译码，由单片机接收，使信号得以还原，从而完成两地的数据通信。

电力线载波传输限于同一个电力变压器供电范围内，主从机最好接于同一相线上，这样传输衰减小、传输距离远、传输效果好。若传输线不同相，则应在它们之间就近跨接 0.1 μF 耐压 630 V 以上的电容，实现交连传送，但传送效果差些。

4. 电话线

利用现成的电话线进行串行数据通信，是非常经济的一个途径。

在远程通信时，传输线上的分布电容是使数据波形失真的主要因素。如果传输线是架在电线杆上的架空导线对，两导线间是由空气绝缘的，相互间距离有数厘米，它们之间就构成了分布电容。导线越长，电容越大，对数据波形的影响越严重。两条传输线距离越近，电容也越大，特别是电缆中的导线，不仅导线间距离小，而且绝缘材料的介电常数远大于空气的介电常数，所以同样长度的电容远比架空导线大，在选用传输线时应给予重视。这就是直接用电话线或普通导线进行数字通信时，距离受限制的原因。

如果传输的脉冲宽度太窄或使用的波特率太高，以致传输线上的电容尚未充满电荷之前就开始下降了，而电容上的电压尚未放电到最低电压又开始充电，这样在接收端无法判断它们，这便是用电话线直接传输数据时，应对波特率加以限制的原因。

实际上用电话线传输直流电压要考虑由电阻、电容和电感以及非理想绝缘所带来的漏电阻等综合影响。

5. 波导管

如果发送频率足够高，那么信号的电磁成分可在自由空间传播，从而不需要任何实际导体。尽管如此，为避免由于信号扩散而引起的干扰和损耗，同时为了使信号沿需要的路由传播，有时我们需要把这些电磁波禁闭在另外一种有界媒质——波导管中。

通常，波导管被用来把微波发射器和接收器连接到它们的天线，其工作频率范围为 2000～110 000 MHz。另外，由于湿气使微波衰减，因此波导管使用干燥空气或氯气来防止这些湿气。常见的波导管是环形波导管，如图 1.5.3 所示。目前波导管仍被用做高功率、高频信号的导体，但更新一些的系统采用了光纤电缆。

图 1.5.3　环形波导管
(a) 螺纹管；(b) 介质层

6. 光纤

光纤是光导纤维的简称，是传送光信号的媒质。光纤是利用各种玻璃和塑料制成的。光纤的结构呈圆柱形，内部是纤芯，外部是包层。纤芯采用二氧化硅掺以锗和磷等材料制成，直径约 5～75 μm；包层采用纯二氧化硅制成，直径约 100～150 μm。光纤的最外层是塑料护皮，用以保护纤芯。纤芯的折射率比包层的折射率高 1%左右，因而可以使光聚集在纤芯与包层的界面之内向前传播，形成光波导。如果纤芯的直径足够细(例如 5 μm 以下)，则光在光波导中的传播只有一种模式，这样的光纤称为单模光纤。如果纤芯的直径较粗，则光波导中可能同时有多种沿不同途径传播的模式，这样的光纤称为多模光纤。

光纤的主要传输特性是损耗和色散。损耗是光信号在光纤中传输时单位长度的衰减，单位是 dB/km。色散是光到达接收端的时延之差，即光脉冲展宽，单位是 ns/km。损耗会影响传输的中继距离，色散会影响传输速率(即传输带宽)，两者都是很重要的指标。

光纤通信的主要优点有：频带宽，通信容量大；在很宽的频带范围内，光纤对各频率的传输损耗和色散几乎相等，不需在接收端或中继站采取幅度或时延等的均衡措施；不受电磁干扰和静电干扰的影响，即在同一根光缆中，邻近各根光纤之间几乎没有串扰；构成光纤的主要原料是石英，石英的资源丰富且价格便宜；此外还有保密性好、线径细、体积

小、重量轻、损耗小、误码率低等优点。光纤已成为当今重要的传输媒质之一，为数字通信和计算机通信网的迅速发展提供了良好的传输环境。

1.5.2　无线传输媒质

所谓无线传输媒质，是指无须架设或铺埋电缆或光缆，而通过看不见摸不着的自由空间，将电信号转换成无线电波进行传送。发信端把待传的信息转换成无线电信号，依靠无线电波在空间传播，而收信端则要把无线电信号还原成发信端所传信息。

无线电波是一种电磁波，因为频率(波长)相差较远的无线电波往往具有不同的特性，通常用频率(或波长)作为无线电波最有表征意义的参量。例如，中长波沿地面传播，绕射能力较强；短波以电离层反射方式传播，传输距离很远；而微波只能在大气对流层中直线传播，绕射能力很弱。因此，把无线电波按其频率(或波长)来进行命名，如表 1.5.1 所示。下面简要介绍短波通信、微波通信、卫星通信、红外通信、激光通信、散射通信以及蓝牙技术等几种无线传输新技术。

表 1.5.1　无线电波频段划分

频　段	名　称	波长范围	主 要 应 用
30～300 kHz	LF(低频)，长波	1～10 km	导航
300 kHz～3 MHz	MF(中频)，中波	100 m～1 km	商用调幅无线电
3～30 MHz	HF(高频)，短波	10～100 m	短波无线电
30～300 MHz	VHF(甚高频)，超短波	1～10 m	甚高频电视，调频无线电
300 MHz～3 GHz	UHF(超高频)，微波	100 mm～1 m	超高频电视，地面微波
3～30 GHz	SHF(特高频)	10～100 mm	地面微波，卫星微波
30～300 GHz	EHF(极高频)	1～10 mm	实验用点点通信

1. 短波通信

短波通信是利用地面发射的无线电波在电离层反射，或电离层与地面之间多次反射而到达接收点的一种远距离通信方式，工作频率范围为 3～30 MHz。电离层的高度由数十公里到数百公里，分为多个不同的层次，而且随着季节、昼夜及太阳活动等情况不断变化，因此电离层的不稳定是造成短波通信质量不稳定的主要因素。同时，由于短波通信可能存在多条传播途径，各途径的时延不等，从而会产生多径效应及衰落现象。加之它的工作频段窄，通信容量小，所以在数据通信中很少使用。但是短波通信的突出优点是投资少、建设快、通信距离远，因而在军事通信及移动通信方面仍有实用价值。

2. 微波通信

微波通信是在对流层的视距范围内利用无线电波进行传输的一种通信方式，频率范围为 1～20 GHz。由于受地形和天线高度的限制，两微波站间的通信距离一般为 30～50 km，长途通信时必须建立多个中继站。中继站的功能是变频和放大，进行功率补偿。通过各中继站逐站将信息传送下去，很像接力赛跑，所以微波通信常称为微波接力通信。

微波通信分为模拟微波通信和数字微波通信两种。模拟微波通信主要采用调频制，每个射频信道可开通 300、600 至 3600 个话路。数字微波通信大都采用相移键控(PSK)，目前国内长途干线使用的数字微波主要有 4 GHz 的 960 路系统和 6 GHz 的 1800 路系统。微波通信的传输质量比较稳定，影响质量的主要因素是雨雪天气对微波产生的吸收损耗，不利地

形或环境对微波所造成的衰落现象。微波通信以其成本低、中继距离远等优点，成为长途通信的主要手段之一。

微波接力信道主要用于长途电信服务。工作在 1 GHz 以下的频段用于移动通信系统和某些数据采集系统；工作在 1.5～10 GHz 的频段用于传输电视和话音的同轴电缆的替代信道。大多数长途电话业务使用 4～6 GHz 的频率范围，由于在这些频率上越来越挤，因此目前也在使用其它较高的微波频率。

微波接力通信可传输电话、电报、图像、数据等信息，同时由于它具有频带宽、通信容量较大，灵活性、可靠性较好，投资少、见效快等优点，目前在各国应用都很广泛。但微波接力信道也存在一些缺点，如相邻站间必须直视、不得有障碍物，受气候干扰较大，保密性差，中继站的使用与维护带来的一些问题等等。

3. 卫星通信

简单地说，卫星通信就是地球上的无线电通信站之间利用人造卫星作中继站而进行的通信。人造卫星已成为当今通信的重要媒质之一。通信卫星被发射到地球赤道上空 36 000 km 处的对地静止轨道上，利用卫星上的通信转发器，可以接收由陆地卫星地球站发射的信号，该信号经放大、变频后，转发到其它的地球站，从而完成地球站之间的传输。一个卫星可以覆盖地球表面的 1/3 地区，只经过一次中继(即一个卫星的中继)，而且频带宽，因此卫星通信的容量大，中继站少。两个地球站间的直接通信距离可达 13 000 km，并且无论通信距离远近，通信的质量都相同，所以卫星通信已成为当今长途通信的主要手段之一，也是计算机通信的良好媒质之一。

4. 激光通信

激光通信是利用激光传输信息的通信方式。激光是一种新型光源，具有亮度高、方向性强、单色性好、相干性强等特征。它具有通信容量大、不受电磁干扰、保密性强、设备轻便、机动性好等优点，但使用时光学收发天线相互对准困难，通信距离限于视距(数公里至数十公里范围)，易受气候影响，在恶劣气候条件下甚至会造成通信中断。大气激光通信可用于江河湖泊、边防、海岛、高山峡谷等地的通信，还可用于微波通信或同轴电缆通信中断抢修时的临时顶替设备。波长为 0.5 m 附近的蓝绿激光可用于水下通信或对潜艇通信。

相比于微波通信，激光通信具有以下优势：发射光束窄，方向性好。激光通信中，激光光束的发散角通常都在毫弧度，甚至微弧度量级，它能较好地解决日益严重的卫星间的电磁波干扰和保密问题。激光通信中的天线尺寸小也是其优点之一。由于光波波长短(约零点几微米到几十微米)，在同样功能情况下，天线的尺寸比微波、毫米波通信天线尺寸要小许多，但其信息容量较大。光波(其相应光频率在 10^{13}～10^{17} Hz)作为信息载体可传输达 10 Gb/s 的数据速率，而且其功耗小、体积小、重量轻，此外，深空对于光波是一种无损耗、无干扰的良好传输介质，传输同样速率与信息的装备，光通信的性价比最高。

5. 红外通信

红外通信在人们日常生活中处处可见：从电视机、VCD 遥控器，到电梯、门禁系统，乃至便携式电脑，都是红外通信的实例。由于红外通信价格低廉，使用方便，解决了有线连接的许多不便，因而受到了家电设备厂商、电脑外围设备厂商以及通信设备厂商的高度重视。

使用发送器和接收器调制出不相干的红外线光就可以实现红外线通信。无论是直接传

输还是经一个浅色表面(如天花板等)的反射，收发器之间的距离都不能超过视线范围。

红外线传输与微波传输之间的一个重要区别是前者无法穿透墙体，因此，微波系统中的安全问题和干扰问题在红外线传输中都不存在。

为了实现各类产品的互连，国际上成立了一个红外数据通信协会(IrDA: Infrared Data Association)来协调各方面的工作，并制定了一系列的标准。这些标准包括：物理层协议、数据链路层协议、数据链路控制协议以及其它高层协议，如流控制协议、串行通信协议、类 HTTP 的实体交换协议、图像交换协议、局域网接入协议等等。这一系列标准的出台不仅规范了产品的设计，同时还进一步扩展了红外通信的功能和应用领域(如数据传输速率可高达到 4 Mb/s，甚至允许高达 16 Mb/s)。

6. 散射通信

散射通信是利用地面发射的无线电波在对流层散射而返回地面的一种通信方式，工作频率范围为 100 MHz～10 GHz，通信距离一般为 150～400 km，最高可达 800～1000 km。

与短波通信相比，散射通信的传输频带较宽，可达数百千赫兹到数兆赫兹，而且通信容量大，能进行多路复用。散射通信的主要缺点是传输损耗大，接收信号的强度变化大(因存在有快、慢两种衰落现象所致)。为了克服衰落现象，需要采取多种措施，如采用分集接收、加大天线、使用大功率发射机和低噪声接收机，致使成本提高。散射通信由于通信容量大，不受核爆、磁爆和极光的影响，因而在军事通信中以及在无法进行微波通信的条件下，有很高的实用价值。

7. 蓝牙技术

蓝牙技术是一种无线数据与语音通信的开放性全球规范，它以低成本的近距离无线连接为基础，为固定与移动设备通信环境建立一个特别连接，其程序写在一个 9×9 mm^2 的微芯片中。如果把蓝牙技术引入到移动电话和膝上型电脑中，就可以去掉移动电话与膝上型电脑之间的连接电缆而通过无线使其建立通信。打印机、PDA、桌上型电脑、传真机、键盘、游戏操纵杆以及所有其它的数字设备都可以成为蓝牙系统的一部分。除此之外，蓝牙无线技术还为已存在的数字网络和外设提供通用接口以组建一个远离固定网络的个人特别连接设备群。

蓝牙工作在全球通用的 2.4 GHz ISM(即工业、科学、医学)频段。蓝牙的数据速率为 1 Mb/s，时分双工传输方案被用来实现全双工传输。

ISM 频带是对所有无线电系统都开放的频带，因此使用其中的某个频段都会遇到不可预测的干扰源。例如某些家电、无绳电话、汽车房开门器、微波炉等等，都可能是干扰。为此，蓝牙特别设计了快速确认和跳频方案以确保链路稳定。跳频技术是把频带分成若干个跳频信道(hop channel)，在一次连接中，无线电收发器按一定的码序列(即一定的规律，技术上叫做“伪随机码”)不断地从一个信道“跳”到另一个信道，只有收发双方是按这个规律进行通信的，而其它的干扰不可能按同样的规律进行干扰；跳频的瞬时带宽是很窄的，但可以通过扩展频谱技术使这个窄带宽成百倍地扩展成宽频带，使干扰变小。

与其它工作在相同频段的系统相比，蓝牙跳频更快，数据包更短，这使蓝牙比其它系统都更稳定；前向纠错(Forward Error Correction)的使用抑制了长距离链路的随机噪声；采用了二进制调频技术的跳频收发器被用来抑制干扰和防止衰落。这些都是蓝牙技术迅速发展的原因。

第 2 章　数据通信中的调制解调技术及应用

本章主要内容：

★ 调制解调技术原理
★ 调制解调器概述
★ 调制解调器技术规范
★ 调制解调器应用实例

2.1　调制解调技术原理

2.1.1　引言

基带信号一般都包含有频率较低，甚至是直流的分量，在有线信道上传输时会因线路中串有的大量电容器或变压器等隔直流元件受到限制，在无线信道上传输时也因很难通过有限尺寸的天线而得到有效辐射。因此，目前大多数信道是不适合传输基带信号的。为使基带信号能利用现有的以传输模拟信号为主体的通信网进行传输，必须使代表信息的原始信号经过一种变换来进行传输，这种变换就是调制。

在实际的数据通信系统中，代表所传信息的原始信号(基带信号)称为调制信号，通常把用载波的一个参量的变化来反映调制信号变化的过程叫做“调制”。由于调制信号的三个参量(幅度、频率和相位)都能携带信息，因此有相应的调幅、调频和调相三种基本的调制形式。用载波幅度、频率、相位的变化来反映调制信号变化的调制分别叫幅度调制(简称调幅)、频率调制(简称调频)和相位调制(简称调相)。实现上述调制过程的设备叫调制器；从已调波中恢复调制信号的过程叫解调，相应的设备叫解调器。一般将调制器和解调器做成双向设备，称为调制解调器(Modulator and Demodulator)。

通常调制可分为模拟调制和数字调制两种方式。在数据通信中一般都采用数字调制。它与模拟调制的基本原理是相同的，区别在于模拟调制是对载波信号的参量进行连续调制，所以解调时必须对载波信号的受调参量进行连续估值。而数字调制是用载波信号参量的离散状态来表征所传送的数据信息，因此在解调时只需对载波的受调参量进行检测和判决即可恢复原始数据。由于数字调制产生的是一种模拟信号，其载波信号参量的离散状态是与数据信息相对应的，因此这种信号适宜于在带通模拟信道上传输。而实际通信系统中的传输媒质大都是带通型的，或是被多路复用分割为带通型的，所以，在数据通信系统中通常需要进行数字调制和解调。

2.1.2 数字振幅调制

数字振幅调制是各种数字调制的基础。所谓幅度调制就是以携带信息的基带信号去控制正弦载波的振幅，使载波的幅度随信息的变化而变化。采用幅度调制方式传输基带数据信号时，常以矩形的基带脉冲去控制正弦载波的振幅，因此称为幅度键控。

当基带脉冲有直流成分时，得到的已调信号是具有载波的双边带信号。由于载频分量不携带基带信号的任何信息，却又占有一定的功率，而上、下两个边带又包含相同的信息，因而它的频带利用率较低。在数据通信中，很少直接采用振幅调制方式。数据通信中通常采用的振幅调制方式有二进制振幅键控调制(2ASK)、多进制振幅调制、双边带抑制载波调制(DSBSC)、单边带调制(SSB)、残留边带调制(VSB)以及正交幅度调制(QAM)等。

虽然数字振幅调制在抗信道噪声的能力上要比数字频率调制和数字相位调制差些，但是，随着电路技术、滤波器技术以及均衡技术等的不断提高，数字振幅调制方式(特别是多电平振幅调制方式)又引起了人们的重视。

2.1.3 数字频率调制

用基带信号对载波的瞬时频率进行控制的调制方式叫做调频，在数据通信中则多称为频移键控（FSK）。频率调制是以携带信息的基带信号去控制正弦载波的频率，使载波的频率随信息的变化而变化，而它的幅度却恒定不变。其中最简单的一种方式是二进制频移键控调制，它是继振幅调制信号之后出现的较早的一种调制方式。由于其实现起来比较容易，解调时不需要本地载波，不需要与信号速度同步，设备不复杂，抗干扰和抗衰落的性能也较强，所以一直被广泛地应用于中、低速数据传输中。

近年来，数字频率调制技术有了相当大的发展，多进制频移键控（MFSK）、连续相位的二进制频移键控（CP-2FSK）、最小频移键控（MSK）以及最近出现的高斯最小频移键控（GMSK）、软化调频（TFM）等技术以其良好的功率利用率、抗码间串扰、带外辐射功率小等优点，在无线信道中得到广泛应用。

2.1.4 数字相位调制

数字相位调制是利用载波的相位变化来传递数字信息的调制方式。由于表征信息的载波相位变化只有有限个离散的取值，所以数字相位调制又叫相移键控(PSK)。相移键控可分为绝对移相的相移键控(PSK)和相对移相的相移键控(DPSK)。所谓绝对移相，是指利用载波的不同相位直接表示数据信息，即用未调载波的相位作为基准的调相；而相对移相是利用载波的相对相位，即前后码元载波相位的相对变化来表示信息。由于绝对移相在接收端解调时会产生信号的相位模糊，造成信号判决的二义性，影响了解调效果，故在实际传输系统中数字相位调制信号几乎都是 DPSK 信号。

数字调相中，载波本身并不携带信息，二相调相信号中没有载波分量，所以信号能量的利用率较高，而它所需的频带却和数字调幅的相等，因而数字调相优于数字调幅。数字调相与数字调频相比，需要较小的频带宽度，它特别适宜于在有衰落和多径信道中传输。

如果传输过程中干扰的扰动比脉冲待续的时间慢，两个脉冲将同样受到影响，但保存了两信号相位差所含有的数字信息。数字调相在恒参信道中比之振幅键控、频移键控具有较高的抗干扰性能，且可更经济有效地利用频带，所以是比较优越的调制方式，因而在实际中尤其是在中高速的数据传输中得到广泛应用。

2.2　调制解调器概述

2.2.1　调制解调器的功能

微机之间进行通信必须借助于传输媒介，即传输信道。当前普遍存在的电话通信网是模拟信道，传输的是模拟信号，呈带通或频带受限的低通特性。而微机输出的数字信号所包含的频率成分较多，频带较宽，并且含有直流和大量的低频成分，不能直接通过电话信道传输。若要通过电话信道传输数字信号，必须采取一定措施，方法是进行调制和解调。具体地说，调制过程是在发送端把数字信号变换成能被模拟信道传输的模拟信号，这是一种数/模变换过程，完成调制功能的设备是调制器(Modulator)；解调过程是在接收端再把接收到的模拟信号转换成数字信号，这是一种模/数变换过程，完成解调功能的设备是解调器(Demodulator)。调制和解调是一个事物的两个方面，缺一不可，因而把能实现信号调制和解调双重功能的设备称为调制解调器(Modulator-Demodulator，缩写为 MODEM)。

2.2.2　调制解调器的构成

调制解调器的构成框图如图 2.2.1 所示。调制解调器主要由基带处理、调制解调和信道形成三大部分组成。调制解调是调制解调器的核心，此外还有均衡和取样判决两部分。下面简单加以说明。

图 2.2.1　MODEM 构成框图

(1) 基带处理是在调制之前对数字信号进行的一些处理，用于消除码间干扰和适应不同调制方式的需要(如调相方式需要双极性码)。基带处理实际上是一种码型变换，因而也叫做基带波形形成。

(2) 信道形成是滤波器取出信号调制频谱并形成系统所要求的调制波形的过程，主要由

收发滤波器完成。其中，发送滤波器取出适合信道传输的调制频谱，该频谱经信道传输后，接收滤波器从中取出有用频谱并滤除噪声。

(3) 调制和解调由乘法器实现，基本过程是：数据信号与载波相乘(调制)，送入信道传输，接收端接收后还原出原数据信号(解调)。

(4) 均衡设备用于消除因信道特性不理想而造成的失真，取样判决器用于正确恢复出原来的数据信号。

2.2.3 调制解调器的标准

由图 2.2.1 可以看出，调制和解调是成对出现的。也就是说，调制和解调不仅要在传输速率、调制方式、通信方式等方面都必须一致，而且机械联结结构也必须相同。因此，为了实现系统间、地区间和国际间的互相通信，调制解调器的标准化、系列化是非常重要的，各厂家的产品都必须遵守共同的标准。当前制定调制解调器标准的组织和所制定的标准主要有以下一些：

(1) 国际电报电话咨询委员会(CCITT)及其制定的 V 系列建议；

(2) 美国贝尔通信公司及其制定的贝尔(Bell)标准；

(3) 美国联邦电信标准委员会(FTSC)及其联邦标准(FED-STD)；

(4) 美国国家军用标准(MIL－STD)及其它地域和国家标准等。

CCITT 是世界上有权威的国际电信组织，我国也参加了其中的工作。该组织每四年开一次会，修改和制定新的建议。我国 MODEM 的标准化工作也正在加紧进行之中，并已制定出一些国家标准(GB)和国家军用标准(GJB)。这些标准的共同特点是向国际标准靠拢，参照采用、等效采用或等同采用 CCITT 的 V 系列建议。CCITT 的最新建议是于 1989 年通过、1990 年正式出版的蓝皮书。蓝皮书增加的新建议有 V.13、V.14、V.33、V.42、V.20、V.230，同时对其它 V 系列建议进行了某些修改。

2.2.4 调制解调器的分类

调制解调器根据应用场合、使用方式、性能指标及调制方式等可以分成许多类型，主要有以下几种。

1. 按使用的传输信道类型分类

MODEM 是数据信号与传输信道的匹配设备，有不同类型的信道就有相应的 MODEM。MODEM 基本上有无线 MODEM 和有线 MODEM 之分。在有线 MODEM 中，根据使用信道的特点又可分为话路 MODEM 和宽带 MODEM。话路 MODEM 是指在一个话音频率范围内传输数据的 MODEM。宽带 MODEM 是指利用载波话路的群路传输数据的 MODEM，如利用载波基群(带宽为 60～108 kHz)以速率 64 kb/s 传输数据的 MODEM。

2. 按工作方式分类

(1) 双工通信、半双工通信和单工通信方式；

(2) 同步传输(又分同步数据同步传输和异步数据同步传输)及异步传输(起—止式编码)方式。

3. 按传输速率分类

(1) 低速：传输速率在 1200 b/s 以下。低速 MODEM 价格便宜，波特率通常为 600 b/s，常用于主计算机系统和它的外部设备之间的通信。例如 Bell 103(贝尔 103)调制解调器采用 FSK 调制技术，售价不到 200 美元，是一种使用十分广泛的 MODEM。

(2) 中速：传输速率为 1200～2400 b/s。这类 MODEM 采用 PSK 技术，常用于计算机系统间的快速串行通信。例如 Bell 201B 型调制解调器。

(3) 高速：传输速率为 4800 b/s 以上(指话音频带传输)。高速 MODEM 采用复杂的 PAM 技术，波特率可达 9600 b/s 以上，常用于高速计算机系统间的远程通信。

还有的分类方法把传输速率在 2400 b/s 以下的定为低速，传输速率在 4800～9600 b/s 的定为中速，传输速率在 14.4～19.2 kb/s 的定为高速。

4. 按电路分类

(1) 专用电路(租用电路)的 MODEM 分两线专用和四线专用两类；

(2) 普通交换电话电路(GSTN)的 MODEM 分人工呼叫/应答、自动呼叫/自动应答两类。

5. 按调制方式分类

按调制方式，通常分为调频 MODEM、调相 MODEM 和调幅 MODEM 三种。

6. 按应用及控制方式分类

按应用及控制方式，MODEM 分单机(独立)式和单板式两种。单板式 MODEM 不是独立的，一般装在计算机(微机)和终端设备内使用。

也有联机式的 MODEM，该型 MODEM 受计算机(微机)的控制。

7. 按集成化程度分类

按集成化程度，MODEM 可分为智能化 MODEM 和非智能化 MODEM 两种。

2.3　调制解调器技术规范

2.3.1　调制解调器的标准速率

调制解调器最主要的技术指标之一是数据传输速率。CCITT 提出了 V.5 和 V.6 两个关于调制解调器数据传输速率的标准建议。CCITT 的 V.5 建议是关于公用交换网中同步数据传输的数据传输速率的标准化建议，速率为 600、1200、2400、4800、9600 b/s。用户可以根据自己的需要以及接续所提供设备在这些速率中选择。CCITT 的 V.6 建议是关于租用电路上同步数据传输速率的标准，允许速率为 600 Nb/s(1≤N≤24)，又分为优先选用速率 600、1200、2400、4800、9600、14 400 b/s 和 3000、6000、7200、12 000 b/s。

关于 600～9600 b/s 的 MODEM 标准，有 V.22、V.23、V.26、V.26bis、V.27bis、V.27ter、V.27 及 V.29 等建议标准。表 2.3.1 给出了它们的基本调制参数。

表 2.3.1 话带 MODEM 的基本调制参数

CCITT 建议	调制方式	比特速率 /(b/s)	调制速率/Bd	特征频率/Hz	带宽/Hz	频带范围/Hz
V.22	2PSK	600	600	1200(低)	600	900～1500
				2700(高)		2100～2700
V.22	4PSK	1200	600	1200(低)	600	900～1500
				2400(高)		2100～2700
V.23	2FSK	最大 600	最大 600	1500	900	1050～1950
V.23	2FSK	最大 1200	最大 1200	1700	1800	800～2600
V.26bis	2DPSK	1200	1200	1800	1200	1200～2400
V.26	4PSK	2400	1200	1800	1200	1200～2400
V.27	8PSK	4800	1600	1800	1600	1000～2600
V.29	4PSK	4800	2400	1700	2400	500～2900
V.29	8PSK	7200	2400	1700	2400	500～2900
V.29	QAM	9600	2400	1700	2400	500～2900

2.3.2 TCM 技术

在频带有限有扰信道中如何解决信息传输的有效性与可靠性的矛盾，一直是大家所关注的课题。伴随着传输速率的提高，每个信道符号传输信息的比特数响应增加，信号点之间的间隔将不断缩小，即收端判决区缩小，判决条件变坏，如果发送信号的平均功率一定，则传输系统的性能急剧下降。由信息论知，从理论上说，为了在不增加信号平均功率的条件下改善系统的误码性能，可在调制前对信号进行纠错编码，以增加信号点之间的距离，增强抗干扰能力，从而可提高系统的性能。然而，在实现上却并非如此简单。通信系统中传统的差错控制方式是在输入信号进入发送机前进行纠错编码，然后对已编码的信号流进行调制传输，发送机并不区分信息比特和纠错编码插入的冗余比特，这样，为了保持数据信息传输速率不变，只有通过提高调制(码元)速率来补偿由于引入纠错编码而导致的传输比特率的提高，从而又引起误码率的上升。在 20 世纪 80 年代以前，人们也曾试图将传统的差错控制方式引入到调制解调器中，但都终因纠错编码获得的编码效益(信噪比增益)不足以补偿由于信号带宽的增加而付出的代价而告失败。20 世纪 70 年代末 80 年代初，Ungerboeck 等成功地将格形(Trellis)编码与调制技术相结合，创立了格形编码调制(Trellis Coding Modulation，TCM)方式。它的特点是在不增加信号传输带宽的情况下获得纠错编码增益。Ungerboeck 证明了在许多情况下，应用 TCM 技术可使调制解调器的系统性能提高 3～6 dB。

TCM 技术采用集分割映射方法，用欧氏距离替代汉明距离选择最佳信号星座，使所选择的码字集合具有最大的自由距离；利用码的冗余度，在接收端选择具有马尔科夫序列特性的有效码序列进行纠错译码，从而获得最大的编码增益。

TCM 调制解调能用卷积编码方法实现，用 QAM 方式调制，译码采用软判决最大似然概率译码——Viterbi 译码实现。

2.3.3　调制解调器新技术

1984 年 ITU-T 通过了 V.32 建议，规范了采用 TCM 技术实现 9600 b/s 话带 MODEM 的方案。之后人们对 TCM 技术进行了进一步的深入研究，并提出了许多相关的新技术，如带宽自适应调制技术、多维 TCM 技术、成形(shaping)技术和预编码(precoding)技术，为高速话带 MODEM 的发展奠定了理论和技术基础。

1. 带宽自适应调制技术

信道带宽是决定信息传输速率的重要因素，在以往的话带 MODEM 中通常是假设信道带宽为 2400 Hz(600～3000 Hz)，随着线路特性的改善，电话网传输质量也明显改善，一些高质量的电话信道的高端频中已超过 3500 Hz，可以充分利用这些频带资源传输更多的信息。带宽自适应调制技术就是基于这种考虑提出来的一种智能化方法，其实现方法是：在通信正式建立前，由发端的 MODEM 发送一定的前向探测信号，接收端根据收到的信号决定诸如信噪比、系统冲激响应和信道可用带宽等信道特性，然后通过反向信道将这些信息送给发送端，发端 MODEM 由这些信息来最佳地利用信道带宽，确定调制速率和调制方式等，保证 MODEM 始终工作在最佳状态和频带高利用率中。

2. 多维 TCM 技术

在最早的 V.32TCM MODEM 中采用的是三维 8 状态格形码，在其后的几年中人们对 TCM 技术进行了深入的研究，提出了多维编码的概念。随着 DSP 处理速度的迅猛提高，使多维 TCM 这类更复杂的技术的实现成为可能。多维 TCM 与二维 TCM 相比，在同等译码复杂度情况下，可获得更好的抗噪声性能；另外采用多维 TCM 可以降低调制信号的扩张率。例如，二维 TCM 系统的信号比不编码系统增加一倍，而四维 TCM 系统只增加 50%，八维 TCM 系统只增加 25%。

在 V.34 建议中，为了适应不同噪声环境下的信道，提出了 16 状态、32 状态和 64 状态三种四维 TCM 方案。

3. 成形技术

人们在研究多维信号星座中发现，在多维 TCM 系统中采用超球形边界的星座同采用超立方体边界的星座相比，可获得极限为 1.53 dB 的信噪比(即成形增益)，这种实现成形增益的技术即称为成形技术。成形技术选择传输信号点时，在多维信号星座的子星座中选择能量最小的信号点进行传输，这样就使得传输信号空间中的信号点呈高斯分布，而不是原来的均匀分布，从而使信号空间中的信号点更靠近坐标原点。信号传输时的平均能量降低，进而使系统的传输信息量更趋于信道容量。

尽管成形增益相对格形编码增益而言较小(前者为 3～6 dB，后者不超过 1.53 dB)，但事实上当编码增益达到一定程度后(例如 3～4 dB)，想进一步获得编码增益所付出的代价是很大的，通常为了再增加大约 0.4 dB 的编码增益，译码的复杂度要增加一倍。因此，从这个意义上看，采用并不太复杂的成形技术可获得 1 dB 左右的信噪比增益，是非常令人激动的。所以，在 V.34 建议中采用了成形技术。

4. 预编码技术

在 MODEM 中当传输速率提高时，信号所占带宽随之增大，由于话带边缘特性的劣化，

将引起严重的码间干扰(ISI)和信号衰减，为了消除这些影响，要采用均衡措施，最常用的方法是采用判决反馈均衡(DFE)器。DFE 的实现原理是：一方面接收端的接收信号通过一个前向均衡滤波器补偿信道特性的畸变；另一方面，接收端对经前向均衡后的信号进行判决，判决后的信息经过一个信道特性的逆滤波器获得误差信号反馈回来与接收信号相减，以抵消信道干扰。

由于 TCM 技术采用 Viterbi 译码，即在接收端对序列整体进行最大似然估计，而 DFE 采用的是逐符号判决，它与 Viterbi 译码方法是矛盾的，所以它不能很好地应用于 TCM 技术中。为解决这一问题，人们想起了 20 世纪 60 年代由 Tomlinson 和 Harashima 提出的预编码技术，它与 DFE 一样，也是一种均衡的方法，其性能与 DFE 相同。采用预编码技术后，仍然维持了与 DFE 相同的均衡特性，又解决了接收端逐符号判决与 Viterbi 译码的矛盾。由此可见，预编码技术可与 TCM 技术更好地结合，所以在 V.34 建议中采用了预编码技术。

此外，由于成形技术和预编码技术等都采用数字信号处理技术实现，它们可以与 TCM 编码、调制和均衡完美地结合在一起，有利于高效率的物理实现。

数据话带调制解调器经过几十年的发展，已经形成了一个比较完整的系列，其技术也由简单调制解调向各门类高新技术集合发展，并继续向智能化、芯片化和多功能方向发展。

V.34MODEM 获得成功的意义不仅在于达到了 28.8 kb/s 的前所未有的高速率，而且为利用话带调制解调器开展其它业务提供了可能。例如采用 V.42bis 建议规范的 4:1 的压缩比，可使数据传输速率达到 115.2 kb/s，这样，它就不仅可以传输一般的数据信号，还可传输可视电话、低速图像等业务的信号。

另外，高速调制解调器必须采用高速数字信号处理器实现。利用目前数字信号处理器的强大信号处理能力，可以将其它终端功能与调制解调器功能合并实现，向多媒质综合终端方向发展。

2.4 调制解调器应用实例

2.4.1 实例一：调制解调器芯片 AM7910 及其应用

AM7910 是美国 AMD(Advanced Micro Deviecs)公司在 20 世纪 80 年代初设计制造的可编程式异步频移键控(FSK)调制解调器，它在一片芯片内有数字滤波器、A/D 和 D/A 转换电路、联络信号、自动应答、反馈环路和反向通信信道，并设有多种工作方式，可与 Bell－103/202，CCITT V.21/V.23 等规程指标兼容，可用引脚选择的方式控制 300 b/s、600 b/s、1200 b/s 的传输速率，可实现全双工传输。同时，它提供了标准接口，可与许多种微处理器系统兼容。此外，其所有的数据接口都与 TTL 电平兼容。

1. AM7910 内部结构及引脚功能

(1) 引脚功能。AM7910 为 28 管脚双列直插式封装，其管脚排列如图 2.4.1 所示。

说明：

脚 1($\overline{\text{RING}}$)：振铃端；

脚 2(V_{DD})：＋5 V 端；

脚 3（$\overline{\text{RESET}}$）：**复位端；**
脚 4（V_{EE}）：**－**5 V **端；**
脚 5（RC）：**载波接收端；**
脚 6（CAP1）**和脚** 7（CAP2）：**外接补偿元件端；**
脚 8（TC）：**载波发送端；**
脚 9（AGND）：**模拟地端；**
脚 10（TD）：**数据发送端；**
脚 11（$\overline{\text{BRTS}}$）：**反向信道发送请求端；**
脚 12（$\overline{\text{RTS}}$）：**发送请求端；**
脚 13（$\overline{\text{CTS}}$）：**发送清零端；**
脚 14（$\overline{\text{BCTS}}$）：**反向信道发送清零端；**
脚 15（BRD）：**反向信道数据接收端；**
脚 16（$\overline{\text{DTR}}$）：**数据终端就绪端；**
脚 17（MC0）**～脚** 20（MC3）：**工作方式控制端；**
脚 21（MC4）：**反馈环方式控制端；**
脚 22（DGND）：**数字地端；**
脚 23（XTAL2）**和脚** 24（XTAL1）：**外接晶体端；**
脚 25（$\overline{\text{CD}}$）：**载波检测端；**
脚 26（RD）：**数据接收端；**
脚 27（$\overline{\text{BCD}}$）：**反向信道载波检测端；**
脚 28（BTD）：**反向信道数据发送端。**

图 2.4.1　AM7910 **引脚排列**

(2) **内部结构。**AM7910 **主要由发送器（调制器）、接收器（解调器）和接口控制逻辑电路（握手信号）三部分组成，如图** 2.4.2 **所示。这里面包括了片内的** A/D **和** D/A **转换电路、数字滤波器、微程序存储器、存储中间滤波器、结果数据存储器、**RS－232C **接口和** V.24 **接口信号电路等。片内集成了约** 5000 **个晶体管。**

图 2.4.2　AM9710 内部结构方框图

2. AM7910 的基本工作方式

用一片 AM7910 可完成 FSK 的调制与解调。FSK 调制解调的国际标准见表 2.4.1。

表 2.4.1　FSK 调制解调的国际标准

标　准	波特率 /(b/s)	双工特性	正向信道		反向信道	
			信号频率 f_m/Hz	空号频率 f_s/Hz	信号频率 f_m/Hz	空号频率 f_s/Hz
Bell－103	300	全双工	1270	1070	2225	2025
CCITT V.21	300	全双工	1180	980	1850	1650
	1200	半双工	1200	2200	调幅(载频 387 Hz)	
Bell－202C	1200	半双工	1200	2200	390	490
CCITT V.23 mode.1	600	半双工	2100	1300	450	390
CCITT V.23 mode.1	1200	半双工	1300	2100	390	450

(1) AM7910 工作方式的选择。AM7910 共有 5 条工作方式选择线，即 MC0～MC4，其中，根据 MC0～MC3 这 4 条线的不同组合可选择不同的工作方式，见表 2.4.2。

表 2.4.2　选择线的不同组合对应的工作方式

接高电平端子	工　作　频　率				相符的国际标准
	正 向 信 道		反 向 信 道		
	f_m/Hz	f_s/ Hz	f_m/Hz	f_s/Hz	
MC0	2225	2025	1270	1070	Bell－103
MC1	1200	2200	390	490	Bell－202
MC2	980	1180	1850	1650	CCITT V.21
MC3	2100	1300	450	390	CCITT V.23
MC4、MC1	1200	2200	1200	2200	1200 b/s 四线全双工

(2) 调制解调器中的反向信道。在选择 Bell－202 或 CCITT V.23 工作方式时，AM7910 提供了较低数码率的反向信道。反向信道通常用于传送应答信号和控制信号（包括状态信号和信令信号），而很少用于传送数字信息。

Bell－202 的反向信道通常工作在 5 b/s，它甚至不用 FSK 方式传送，而是用开关信号代替。当有信息时，发送一个 387 Hz 的模拟信号；没有信息时，不发送任何信号。

V.23 的反向信道是一个真正异步 FSK，其传输速率为 75 b/s。75 b/s 的传输速率可在每秒内传送约 7 个 ASCII 字符。此外，这种调制解调器可用于电视传真，这时就需要一个高速的主信道和一个类似打字机应答的低速反向信道。

(3) 反馈环工作方式。反馈环工作方式由管脚 MC4 来确定，这种工作方式是为自身测试而设计的。用编程的方式可通过选择来完成 10 种反馈环测试。

在通常的测试方式下，发送器和接收器工作在不同的频道上，即在某个频道上发射的同时，可在另一个频道上进行接收。为了检测调制解调器的工作状态是否正确，可将调制解调器置于反馈环工作方式，也就是让收/发工作在同一个频道上。若将接收的数据端（RD）与发射的数据端（TD）直接相连，可形成数据反馈环；若将接收载波输入（RC）与发射载波输出（TC）直接相连，可形成模拟反馈环。自身的控制器可给调制解调器发送一串数据，因为收/发工作在同一个频道，接收器能够解调出发送的一串数据，再自动将这串数据送回控制器。如果在对信号的调制解调过程中出现错误，可以被控制器检测出来。

AM7910 的每种工作方式都有一个反馈环工作方式，再加上一个反向信道反馈环工作方式，用于发送 CCITT V.23 的状态和信号信息。这样，共有 10 种反馈环工作方式。

(4) 双工操作。在 AM7910 的工作方式中，允许 300 b/s 的两线全双工和 1200 b/s 的四线全双工工作。

在 300 b/s 全双工工作时，需要同时占用两个不同方向的传输信道。因为数码率较低，所以需要占用的带宽相对窄些。1200 b/s 四线全双工工作是一种特殊的应用，此时，AM7910 置于 1200 b/s 反馈环工作方式。反馈环工作方式允许发送器和接收器工作在同一个频道上，同时又都工作于 1200 b/s 的传输速率上，但必须使用 4 条专用线，其中，两条线用于发送，另外两条线用于接收，只有这样才能把发送和接收分开，从而实现 1200 b/s 的四线全双工工作。

(5) 自动应答和握手信号。进行网络接口时，AM7910 可以做自动应答处理。

将调制解调器连接到电话线上通常需进行网络接口。当接口检测到一个有效的振铃信号时，表明有呼叫信号等待应答，接口将这个振铃信号送给 AM7910。此时，AM7910 先产生一个短暂的时间间隔（即空周期，它是为电话自动计费所设），然后从其发射载频上调制一个应答音，通过网络接口耦合到线路上，在线路的另一端，如果收到应答信号，即表明 AM7910 工作正常。AM7910 也提供了必备的握手信号，这是 RS-232C 或 CCITT V.24 协议的一部分。这些握手信号控制着与调制解调器相关的特定工作方式的动态操作。例如，“数据终端就绪”$\overline{\text{DTR}}$ 是芯片“使能”信号，当 $\overline{\text{DTR}}$ 无效时，芯片不能进行任何调制解调工作。“发送请求”$\overline{\text{RTS}}$ 是终端到调制解调器的信号，它通知调制解调器终端有数据要通过调制解调器发送出去，调制解调器用“发送清零”$\overline{\text{CTS}}$ 信号予以响应，即调制解调器已经准备接收从终端来的串行数据。当调制解调器从电话线路上接收到从另一个调制解调器发来的有效数据时，载波检测 $\overline{\text{CD}}$ 端被置位，表明控制器或终端已经收到了电话线路上的有效载

波。除非 $\overline{\text{DTR}}$ 被置位，调制解调器的 $\overline{\text{CTS}}$ 和 $\overline{\text{CD}}$ 端都不会开放，因而禁止数据的传输和接收。

AM7910 也提供了用于 1200 b/s 的 Bell-202 或 CCITT V.23 半双工方式的反向信道握手信号。$\overline{\text{RTS}}$ 为控制主信道或反向信道进行发送的控制端。当其为“0”时，在控制主信道上发送，在反向信道上接收；当其为“1”时，在控制主信道上接收，在反向信道上发送。

3. AM7910 在通信中的应用

AM7910 为单片 FSK 调制解调器，以有线或无线的形式在很短的时间内传送大量的数据。

(1) 利用电话线进行信息传输。AM7910 可以连接到电话线上，利用有线方式进行数据信息的传输，它与电话线的接口电路是数据耦合器。数据耦合器常用的一种应用形式是通过电话线路接口（PhoneLine Interface）直接连接到电话线路上去，如图 2.4.3 所示。电话线路接口有现成的模块电路芯片，如 Novation PLI、Cermetek CH1810 等。AM7910 与电话线路接口相连，在系统控制器的控制下可以完成自动呼叫、自动应答、自动拆线、载波检测、振铃信号的发送和接收以及反馈环路的测试等功能。

图 2.4.3　AM9710 与电话线路 Cermetek CH1810 的接口

(2) AM7910 可以直接与多种单片机相连。图 2.4.4 是 AM7910 与 8051 单片机的连接。DAA 也是电话线路接口芯片。

通过对 8051 异步串行通信口的编程可实现 AM7910 的全双工操作和半双工操作。如果采用脉冲或音频拨号，8051 可以作为自动呼叫单元。脉冲拨号由电话线路接口摘机端（OH）脉冲完成；音频拨号可通过 8051 的 P1.2 控制继电器将 MK5089 连接到电话线路接口电路上，由 MK5089 音频拨号芯片完成。

8051 也能方便地控制 AM7910 的工作方式。如 8051 可以使能 MC4 端，使 AM7910 工作在反馈环工作方式，这时 AM7910 的发送和接收滤波器工作在相同的频段上，这样，载波发送输出（TC）可以直接连接到载波接收输入（RC），构成模拟反馈环；或数据接收输出

(RD)可以直接连接到数据发送输入(TD)，构成数字反馈环；也可以通过 8051 并行输出口控制 AM7910 的脚 MC0～MC3，使其工作在不同的工作方式。图 2.4.4 所示的连接为 AM7910 工作在 Bell-103 方式。

图 2.4.4　AM7910 与 8051 单片机的接口

(3) AM7910 在无线通信中的应用。在工业测控系统中，经常需要将系统终端采集处理的数据送到远方的中心处理机中去。在距离遥远或无通信的情况下，必须借助无线信道进行数据传输。下面介绍两种可以构成无线计算机通信网络的调制解调器。将它们与 VHF 或 VHF 单工无线电对讲机和相应基地台结合，可以构成性能良好的、满足一般监控要求的无线数据传输系统，较之专业无线数据传输机，该系统的投资费用大大减少。

① 电路原理。中心计算机一般都带有 RS－232C 串行接口，前沿数据采集终端一般常采用单片机，通常也带串行接口。这里介绍的调制解调器有两种：一种用于中心机，简称 MODEM PC；另一种用于终端机，简称 MODEM 51。它们分别与中心机(PC 机)和 MCS-51 系列单片机配合使用。电路中主芯片为国际上通用的单片——MODEM 芯片 AM7910。

下面介绍两种调制解调器接口电路的设计，主要基于如下几点考虑：第一，节约频率资源；第二，结构简单，不改动电台；第三，可靠性高。因此，设计时采用半双工/手动工作状态选择、防射频发射阻塞电路。

图 2.4.5 所示为 MODEM PC 构成原理图。图 2.4.6 所示为 MODEM 51 构成原理图。由图 2.4.5 和图 2.4.6 可知，MODEM PC 与 MODEM 51 的不同之处仅在于两机给出的信号不同。两种调制解调器的电原理图如图 2.4.7 和图 2.4.8 所示。

② AM7910 与 PC 机接口。图 2.4.7 是中心机，即带有标准 RS－232C 串行接口的计算机，或计算机的终端所使用的调制解调器。

图 2.4.5　MODEM PC 构成原理图　　图 2.4.6　MODEM 51 构成原理图

电路中所有 1488、1489 芯片是将 TTL 电平与 EIA 标准 RS－232C 接口逻辑电平相互转换。

X1 是 4 脚晶体振荡电路，频率为 2.4576 MHz，输出方波信号直接送入 AM7910 的时钟输入端。AM7910 的 TC、RC 分别是模拟信号即载波的输出、输入端，经简单的 RC 网络与收发机（对讲机）的话筒 MIC 或耳机 EAR（或扬声器）插口相接。加入 RC 网络的目的是为了使语音信道的频率特性较好地与 AM7910 特性配合，增加信号传输的可靠性。

S1 为一组拨动开关，用它可以设置调制解调器工作方式，如 Bell－102、Bell－203、CCITT V.21、CCITT V.23 等各种方式，波特率有 300 b/s、600 b/s、1200 b/s 等等。这一状态应与对方调制解调器完全相同。

PTT 控制，即收发信机发送/接收状态的控制是通过 RTS 信号的高低电平实现的。本电路由 U10(A 与非门)、U7(A 可重触发单稳态电路)和 U9(A 与门及光电耦合器件 4N25)完成，同时该电路可以防止因发射阻塞而产生通信中断甚至烧毁收发信机。表 2.4.3 为通信时接口及有关逻辑电平。表中按正逻辑填号，×表示任意状态，\表示下降沿，/表示上升沿。

接通电源后电路的状态变化如表 2.4.3 所列。工作过程简述如下：串行接口初始状态时尚无数据发送，D 点被初始化为低电平，故 E 点输出低电平，因此，4N25 的 PTT 连接端呈高阻状态，即收发信机为接收状态。同时在 TxD 及 RTS 作用下 C 点为低电平。

当有请求发送信号时，RTS 由低电平变为高电平，C 端也由低电平变为高电平，该上升沿触发 74LS123 翻转，从而 D 点变为高电平，因这时 B 点也为高电平，故 E 点输出高电平，从而 4N25 输出端呈低阻状态，即收发信机进入发送状态。

表 2.4.3　通信时有关接口及逻辑电平表

状　　态	TxD	RxD	RTS	CTS	DTR	DCD	A	B	C	D	E
初始状态	L	L	L	H	L	L	H	L	L	L	L
请求发送(未发数据)	L	L	H	L	H	L	H	H	H	L	H
发送数据	×	L	H	L	H	L	×	H	×	H	H
阻塞状态	×	L	H	/	H	L	H	H	L	\	\

图 2.4.7　MODEM PC 电原理图(接 PC 机)

图 2.4.8　MODEM 51 电原理图(接 8031)

在发送数据时，A 点电平随数据流数据位的变化不断变化，因此，C 点也就随之变化，至少每帧变化一次。因为已设置该可重触发单稳电路的暂稳时间远大于每帧数据传输时间 T_D。D 点 $\tau >> T_D$ 将维持高电平，保证 E 点高电平，即保证收发信机在发数据期间一直处在发送状态。发送数据结束 RTS 使 B 点电平变低，E 点也随之变为低电平，收发信机变为接收状态，因为 TxD 已无数据发送，则经一段时间延时后 D 点也返回低电平状态，随后等待下一次通信的到来。

当某种因素导致未发送数据，但收发信机却处于发射状态（即“发射阻塞”）时，由于 A 点无数据流输入，因而经一段时间延时（由 $\tau = R_t C_t$ 决定）后，D 点电平返回低电平，迫使 E 点为低电平，从而将发射状态切换为接收状态，避免了“阻塞”的发生。

由电路可知，清除发送信号 CTS 由 D 点及 AM7910 的 CTS 脚相与非后送到 RS－232C 接口。因此，程序在读取调制解调器状态时要考虑延时的影响。

③ AM7910 与 MCS－51 接口。采用单片机系统作为终端机的调制解调器 MODEM 如图 2.4.8 所示。设计接口控制信号由 MCS－51 系列单片机的 I/O 口组成，电路原理与 MODEM PC 相同，但接口控制逻辑的电平均为 TTL 电平，不必使用 EIA RS－232C 串行接口的电平转换器件。因此，控制电平的有效/无效状态与 MODEM PC 不同。

2.4.2　实例二：调制解调器芯片 SSI73K222AL 及其应用

由 MODEM 芯片构成的单片机自动报警装置可以借助于工厂、企业内部的电话交换机网络，甚至公用电话交换机网络，远距离传送报警信息，不受地点和时间的限制，真正做到安全、迅速和准确。

1. SSI73K222AL 芯片简介

SSI73K222AL 是 TDK 公司近期推出的产品，它是一种高集成度的单片 MODEM 芯片。该芯片的主要特点是：可以和 8048 或 80C51 单片机对接，接口电路简单；串行口数据传输；既可以同步方式又可以异步方式工作，包括 V.22 扩充超速；与 CCITT V.22、V.21、BELL 212A、103 标准兼容；具有呼叫进程、载波、应答音、长回环检测的功能；能够通过编程产生 DTMF 信号及 550 Hz、1800 Hz 的防卫音信号；具有自动增益控制，动态范围达 45 dB；采用 CMOS 技术，低功耗、单电源供电。

图 2.4.9　SSI73K222AL 引脚排列

SSI73K222AL 具有 28DIP 封装，其引脚如图 2.4.9 所示。

SSI73K222AL 内部有四个寄存器可用于控制和状态的监视。其中，控制寄存器 CR0 用于控制电话线路上数据传输的方式。控制寄存器 CR1 用于控制 SSI73K222AL 内部状态与单片机之间的接口。检测寄存器 DR 是一个只读寄存器，它提供了监视 MODEM 工作状态的条件。音调寄存器 TR 则用于控制音频信号的产生，在 TR 的控制下，MODEM 可以产生 DTMF 信号、应答音信号和防卫音信号。还可以在 MODEM 启动和与对方联系过程中对 RXD 引脚进行控制。

有关寄存器各状态位的功能以及各寄存器的使用方法见表 2.4.4。

表 2.4.4　SSI73K222AL 寄存器各状态位的功能以及使用方法

寄存器名称	地址	数　据　位							
	AD2～AD0	D7	D6	D5	D4	D3	D2	D1	D0
控制寄存器 CR0	000	调制选择	0	发送模式：其中，1100=FSK 模式				发送允许	应答/始发
控制寄存器 CR1	001	数据发送方式		中断允许	旁路编码	时钟控制	复位操作	测试模式，其中：00=正常	
检测寄存器 DR	010	未用		接收数据	介码标志	载波检测	应答音	呼叫进程	长环检测
				条件检测到出“1”，否则出“0”					
音调控制寄存器 TR	011	RDX 控制	发防卫音	发应答音	发送 DTMF 音	该四位对应 1～16DTMF 信号，即：1=0001，2=0010，…，注意：0=1010			

2. SSI73K222AL 芯片在自动报警装置中的应用

在构成单片机自动报警装置时，可以有以下三种方案供用户选择：

(1) 直接拨通 BP 机号码报警。这是一个最简单的方案，硬件电路如图 2.4.10 所示。

图 2.4.10　直接拨通 BP 机号码报警方案硬件电路

首先，由单片机巡回监视报警信号的出现。图中，以 P1.3 口电位变低作为出现了报警信号。如有报警，则单片机立即通过 P1.7 口输出低电平，吸合继电器 J1，将装置与电话线

路接通。接着，单片机按照事先给定的 BP 机号码发 DTMF 信号即开始拨号，当接到传呼台的回音信号后即自动挂机(断开继电器 J1 的触点)。89C51 单片机控制子程序编制如例 2.4.1 所示。

例 2.4.1　本例中所拨打的 BP 机号码假设为 2065。

```
WAN：  JNB P1.3，DT              ；监视 P1.3 口
       SJMP WAN
  DT：ACALL DLY2                 ；延时 50 ms
    JNB P1.3，ARM                ；确认有报警信号，转处理程序
    SJMP WAN
  ARM：CLR P1.7                  ；吸合继电器 J1
    ACALL DLY2                   ；延时 50 ms
    MOV R6，#04H                 ；拨打四位电话号码，预置初值
    MOV DPTR，#7FF8H             ；地址指针指向 R0
    MOV A，#31H                  ；R0 按始发方式、FSK 模式设置。但禁止发送
    MOVX @DPTR，A
  LOOP：  MOV DPTR，#7FFBH       ；地址指针指向 TR
    MOV A，#0FH
    ADD A，R6                    ；取出电话号码
    MOVC A，@A+PC
    MOVX @DPTR，A                ；设置 TR
    MOV DPTR，#7FF8H             ；地址指针指向 R0
    MOV A，#33H                  ；允许发送
    MOVX @DPTR，A
    ACALL DLY3                   ；延时 250 ms
    MOV A，#31H                  ；停止发送
    MOVX @DPTR，A
    ACALL DLY3                   ；延时 250 ms
    DJNZ R6，LOOP                ；拨号未完，再拨出一个号码
    DB 95H，96H，9AH，92H        ；TR 设置及电话号码
  DTA：  MOV DPTR，#7FFAH        ；地址指针指向 DR
    MOVX A，@DPTR                ；监视 DRJNB ACC.2，DTA；检测应答音
    MOV DPTR，#7FF9H             ；地址指针指向 R1
    MOV A，#04H
    MOVX @DPTR，A                ；复位 MODEM
    SETB P1.7                    ；释放 J1
    RET
```

在这个方案中，持有该 BP 机的管理人员必须熟知各报警部门的电话号码，以便及时采取对策。

(2) 与语音电路相结合的报警。在这个方案中，应增设一块语音电路，我们在实验中采

用的是 ISD-1110 语音电路，该电路具有可随机录入、可循环播放的功能，每次放音时间为 10 s。该电路的引脚如图 2.4.11 所示。

录音时按下 AN 按钮，电路中 LED 发光，人对着话筒说话，话音就被录入芯片，录入的内容即使断电后仍不丢失。循环放音时只需使 PL 接低电位，早先录入的话音将通过喇叭被重复播放出来。现采用 89C51 的 P1.4 脚对其进行控制，可以在需要时刻投入工作。接线时可将输出端之一 SP+(或 SP–)接入图 2.4.10 中的 A 点，其它引脚按提示连接。

单片机编程方案与上例基本相同，区别在于：此处应按照事先给定的电话号码（例如“110”）发 DTMF 信号。拨号过程结束，延时一定时间之后即可启动语音电路工作。令其反复播放同一段预先录入的话语，如“我是某地某人，情况紧急，请求帮助”等。接电话的值班员，无论是谁，都可立即明白。

图 2.4.11 与语音电路相结合的报警电路引脚图

单片机控制方面，只须增加两条指令，对 P1.4 口进行控制即可。

(3) 接收端采用 MODEM 和单片机显示装置的报警。在接收端采用 MODEM 和单片机显示装置可以在无人值守的场合自动监视从各处发来的报警信息，将其存储并用数码显示出来，必要时还可增设警报音响等其它设施。

图 2.4.12 给出了一个简单的接收端采用 MODEM 和单片机接收装置的电路原理图。该装置可以通过电话线路与上述报警装置配合工作，进行数据通信。由于接收端无需 DTMF 拨号等功能，所以图中采用了 OKI 公司的低速 MODEM MSM6946，它结构简单、价格低廉、控制和使用都很方便，适用于 300B/S、FSK 工作方式，可以满足 Bell－103 标准。

图 2.4.12 中，接收端的 MODEM 按应答方式接线，单片机 89C51 平时处于巡回检测电话振铃信号的状态，一旦检测到该信号，则可将 J2 吸合，在两秒钟左右的沉默之后，启动 MODEM 发送应答音。双方经过简短的握手过程之后，89C51 便将收到的对方代码通过数码管显示出来。

为了使电路简单，图中采用了具有 BCD 转换、锁存、七段译码及驱动功能的 CMOS 电路 CD4511，当 89C51 在 P1.7～P1.4 口输出 0～9 的 BCD 码时，数码管能直接显示出来。由此看来，本电路可以区分九个报警点发来的报警信息。

在这种方案下，图 2.4.10 所示发送端的报警装置硬件线路不变，但控制软件应当作相应的补充，即在发送完 DTMF（拨号信号）之后，程序还应增加检测应答音、发送和接收握手信号、循环发送本机代码等内容。

图 2.4.12　**简单的** MODEM**、单片机接收装置**

2.4.3 实例三：HART 调制解调器 HT2012 的原理和应用

工业控制仪表普遍采用 4～20 mA **的模拟信号作为通信标准，在** 4～20 mA **的模拟信号基础上增加数字通信的能力，都采用通信协议，其中用得最多的是** Rosemount **公司研制的** HART(Highway Addressable Remote Transducer)**现场通信协议和** Honeywell **公司研制的** DE **现场通信协议。能实现** HART **协议通信规程的芯片是** Smar **公司生产的** HT2012**。**

1. HART **协议简介**

HART **协议是在** Bell-202 **标准通信基础上发展起来的，它使用了频率移相键控**(FSK)**技术。在传送** 4～20 mA **模拟信号的二线制中，还要传送数字信号。**HART **协议规定，在** 4～20 mA **电流上叠加一个** 1200 Hz **和** 2200 Hz **频率的正弦信号，分别代表“** 1 **”和“** 0 **”，其中** 1200 Hz **信号表示“** 1 **”，**2200 Hz **信号表示“** 0 **”。由于正弦信号的直流平均值为零，所以没有直流加到** 4～20 mA**。由此，在二根线上可以同时传送互不影响的模拟和数字信号。**

2. HT2012 **的引脚说明和特点**

(1) HT2012 **的引脚说明。**HT2012 **有两种封装，分别是** 16 **脚** DIP **和** 28 **脚** PLCC**，其引脚说明如表** 2.4.5 **所列。**

表 2.4.5 HT2012 **引脚说明**

信 号	**类型**	DIP **引脚**	PLCC **引脚**	**功 能 描 述**
V_{DD}		1	1	**＋**5 V **电源**
019_2k	**输出**	2	2	**用户时钟，**19.2 kHz
OCD	**输出**	3	3	**载体检测**
IRXA	**输入**	4	4	**解调器输入**
NC		5	5～12	**空**
TEST1	**输入**	6	13	**测试输入** 1**，正常使用时接** V_{SS}
NC		7		**空**
ORXD	**输出**	8	14	**解调器输出**
V_{SS}		9	15	**地**
NC		10		**空**
OTXA	**输出**	11	16	**调制器输出**
INRTS	**输入**	12	17	**请求发送**
TEST0	**输入**	13	26	**测试输入** 0**，正常使用时接** V_{SS}
ITXD	**输入**	14	27	**调制器输入**
NC		15	18～25	**空**
II 460K	**输入**	16	28	**输入时钟** 460.8 kHz

(2) HT2012 **的特点。**

① **单片** CMOS**，低功耗，**FSK **调制解调器；**

② **工作在** Bell－202 **标准上，传输率是** 1200 b/s**，半双工操作；**

③ **标准频率是** 1200 Hz(**表示** 1)**和** 2200 Hz(**表示** 0)**；**

④ **单一** 3～5 V **的电源，功耗很低，为** 40 μA**；**

⑤ CMOS **和** TTL **兼容；**

⑥ **在过程控制仪器和其它低功耗设备中提供** HART **通信协议；**

⑦ **需外部提供** 460.8 kHz **的时钟；**

⑧ 封装有 16DIP 和 28PLCC 两种。

3. 功能描述

HT2012 主要有 4 个功能模块，它们是时钟模块、调制器模块、解调器模块和载体检测模块。其原理如图 2.4.13 所示。

图 2.4.13　HT2012 的功能模块

(1) 时钟模块。时钟模块接受外部输入的 460.8 kHz 时钟信号，用于建立内部时钟。正常使用时，内部时钟为 19.2 kHz，输出到插针 2 上，供外部电路使用。

460.8 kHz 分别产生表示“0”(1200 Hz)和“1”(2200 Hz)两种频率，即

460.8 kHz÷3÷70=2194.3 Hz

460.8 kHz÷5.5÷70=1196.9 Hz

其相位误差分别为 9.5°和 5.2°。

(2) 调制器模块。从调制器模块的 ITXD 输入不归零制(NRZ)的数字信号，并将数据调制成 1200 Hz(逻辑“1”)和 2200 Hz(逻辑“0”)，从 OTXA 插针输出。

调制器输出频率(在 OTXA 插针上)：高频为 2194.3 Hz；低频为 1196.9 Hz；调制器相位连接误差为最大±10°。

(3) 解调器模块。从解调器模块的 IRXA 输入　　调制方波的数字脉冲，解调后的数字信号从 ORXD 输出，其条件为：

① 输入频率为 1200 Hz±10 Hz，2200 Hz±20 Hz；

② 时钟频率为 460.　Hz(1±0.1　)；

③ 输入(IRXA)的不对称性等于 0；

④ 解调器的偏差：一位时间的±12　。

其波形参看图 2.4.14。

图 2.4.14　解调器波形

(4) 载体检测模块。

① 载体检测的频率范围为 1000～2575 Hz。在 IRXA 上的信号频率在此范围内时，载体检测信号 OCD 必定为低(逻辑 0)；

② 从载体输入到载体检测的时间最小为 9.0 ms，最大为 14.4 ms，即从 IRXA 上有效载体信号开始到 OCD 变为逻辑“0”的时间；

③ 从载体卸载到停止载体检测的时间最大为 1.68 ms，即从 IRXA 上有效载体信号的消失开始到 OCD 变为逻辑“1”的时间；

④ 载体检测条件：时钟频率为 4.608 kHz (1±0.1%)，输入(IRXA)的最大不对称性为 5.0%。

4. HT2012 与微处理器和介质的接口

HT2012 HART 芯片用于外设到微处理器或微控制器的连接，所以使用时必须提供两个接口，一个是到微处理器或微控制器的接口；另一个是到介质(即外设)的接口，如图 2.4.15 所示。

图 2.4.15　HART 的接口

(1) 微处理器或微控制器接口。微处理器发出控制信号送到 HART 的 INRTS 插针。

如果 INRTS 为逻辑“0”，调制器使能，发送数据被调制，从 OTXA 插针输出。此时，解调器关闭，以节省电源功耗。

如果 INRTS 为逻辑“1”，解调器使能，从 IRXA 插针来的数据被解调，输出到 ORXD 插针上。此时，调制器关闭，以节省电源功耗。

解调器激活时，有效载体存在，则载体检测线 OCD 有效(低)，微处理器或微控制器查询此线有效，则开始输入解调数据。

(2) 介质接口。典型应用是半双工方式，即发送数据和接收数据在同一介质上进行。介质接口一般由两部分组成：一是调制输出要经过波形整形器；二是解调输入时要经过带通滤波器。

5. HT2012 的应用

HT2012 广泛用于工业控制仪表的现场通信，特别是在二线制的智能压力变送器中，采用 HART 通信协议，实现在两根线上同时传送模拟信号和数字信号。

2.4.4　实例四：嵌入式调制解调器与单片机的接口及编程

在分布式测控系统中，一般利用 DDN **的公用电话网来实现系统内部各子系统的通讯。而在** POS **系统、现代城市交通信号控制系统和** IC **卡电话机管理系统中，则宜采用嵌入式调制解调器，以便利用现有的电话线路进行联网。**

1. MSM7512B **嵌入式** MODEM **芯片的结构与特点**

MSM7512B **是用单一的** 3～5 V **电源供电的嵌入式调制解调器，其数据接口与** TTL **电平兼容，并可与单片机** AT89C2051 **直接连接。**MSM7512B **采用的是** CCITTV23 **标准，可进行** 1200 b/s **发送，**1200 b/s **接收与** 75 b/s **发送两种方式的数据传输。**

MSM7512B **内部主要由调制器、解调器、接口逻辑电平以及信号滤波放大电路组成。它的封装有两种形式，其中** DIP **封装的** MSM7512BRS **芯片的引脚排列如图** 2.4.16 **所示。**

图 2.4.16　MSM7512BRS **封装图**

说明：

VDD：3～5 V **电源，工作状态时的功耗为** 25 mW，**省电时功耗为** 0.1 mW。

AI：FSK **模拟信号输入端。**

AO：FSK **模拟信号输出端。**

EAI：**外部检测信号输入端。**

X1、X2：**晶振** 3.579 545 MHz **接入端。**

CLK：3.759 545 MHz **信号输出端。**

RD：**调制解调器串行信号输入端。**

$\overline{\text{CD}}$ ：**用于** FSK **接收信号及回答音的监测，当** $\overline{\text{CD}}$ **脚输出为低电平时，表示已接收到信号。**

XD：**调制解调器串行信号输出端。**

$\overline{\text{RS}}$ ：**当** $\overline{\text{RS}}$ **脚输入为低电平时，允许输出** FSK **信号和回答音。**

$\overline{\text{TEST}}$ ：**检测端口，一般接高电平或悬空。**

MOD1、MOD2：**工作模式选择端。其选择模式如表** 2.4.6 **所列。**

AOG：**为** AO **脚模拟输出信号电平选择端，该脚接高电平时，输出信号强度为**－10 dBm，**接低电平时，输出信号为**－4 dBm。

表 2.4.6　**工作模式选择**

MOD2	MOD1	**工 作 模 式**
0	0	1200 b/s FSK **发送模式**
0	1	1200 b/s FSK **接收和** 75 b/s **发送模式**
1	0	**模拟回路测试**
1	1	**省电模式**

MSM7512B 在进行通讯时，其正向信道传输速率为 1200 b/s，传号频率为 1300 Hz，空号频率为 2100 Hz ;而反向信道的传输速率为 75 b/s,传号频率为 390 Hz,空号频率为 450 Hz。

2. 接口电路

根据 MSM7512BRS 的内部结构，设计出它的外围电路以及与 AT89C2051 单片机的接口电路，如图 2.4.17 所示。

图 2.4.17　MSM7512BRS 与 AT89C2051 的接口电路

当电话线路有 25 Hz 振铃信号时，光耦的输出端变为低电平，触发单片机中断 INT0，并开始吸合继电器 R2 以形成直流通道。这时，接收的数据由串行输入口存放于单片机系统中。

同样，当需要发送数据时，首先吸合继电器，然后将发送的数据通过 XD 端输入到 MSM7512B，完成发送过程。

电源噪音对 MSM7512B 有很大影响，尤其当电源噪音是 SC 电路采样频率 28 kHz 的倍数时，MSM7512B 会误认为有载波信号，并触发载波检测端。因此，图 2.4.17 中，在 MODEM 芯片的电源旁增加了由电解电容与独石电容组成的旁路电容组，以消除电源噪音干扰。

3. 软件编程

该电路的接口软件程序的主要内容有：单片机波特率及串行口设置、外部中断处理、接收子程序、发送子程序。

下面是结合城市交通控制系统，对一主多从结构采用查询方式通讯协议的软件设计过程。

(1) 串行口设置及波特率 MODEM 芯片初始化。对于 AT89C2051 单片机，采用半双工通讯方式，当晶振为 11.0592 MHz 时，软件设置程序见例 2.4.2。

例 2.4.2

```
TA1 EQU P3.5
TA2 EQU P3.4
MOV TMOD，#20H
MOV TH1，#OE8H          ；晶振 11.0592 MHz
MOV TL1，#0E8H          ；波特率为 1200 b / s
MOV PCON，#0H           ；使 SMOD=0
MOV SCON，#50H          ；串行通信方式 1
SETB EX0                ；INT0 为高级中断
SETB PX1
SETB TR1                ；定时器 T1 开始工作
CLR TA1                 ；初始化为 1200 b / s 接收模式
SETB TA2
SETB EA                 ；开中断
```

(2) **外中断及数据接收。**MSM7512B **的接收响应如图** 2.4.18 **所示。在** FSK **信号成立后，经过** t_{CD1} **延迟，才有** $\overline{CD}$ **脚低电平输出。**t_{CD1} **一般为** 10 ms **左右。**

图 2.4.18　**接收响应**

例 2.4.3

```
CD EQU P1.3
JTQ EQU P1.4
RS EQU P3.3
INT0： PUSH DPH         ；保护返回信息
PUSH DPL
PUSH ACC
ACALL D100MS            ；调 100 ms 延时
JB CD，RETURN
REC： CLR JTQ           ；吸合继电器，摘机
```

```
CLR RS                    ；将 RS 清零
CLR TA2                   ；发回音信号
MOV R1，#0
REC1：CLR TI              ；60 个立即数#55H
MOV A，#55H
MOV SBUF，A
JNB TI，$
INC R1
LCALL D100MS              ；延时 l00 ms
CJNE R1，#60，REC1
CLR TI                    ；开始接收
SETB TA2
MOV R1，#50H
REC2：JNB RI，$
MOV A，SBUF               ；接收到一组数据
CLR RI                    ；保存于 RAM50H～6FH
MOV@R1，A
INC R1
CJNE R1，#70H，REC2
RETURN：SETB JTQ          ；断开继电器，挂机
POP ACC                   ；返回
POP DPL
POP DPH
RETI
D100MS：MOV R7，#100      ；延时 100 ms 子程序
D1：  MOV R6，#64H
D2：  NOP
NOP
NOP
DJNZ R6，D2
DJNZ R7，D1
RET
```

(3) **数据发送。**MSM7512B **的发送响应如图** 2.4.19 **所示。在** $\overline{\text{RS}}$ **信号成立后，经过** t_{RS1} **延迟，才有** AO **脚** FSK **信号输出。**t_{RS1} **一般为** 2 ms **左右。**

例 2.4.4

```
CLK EQU P1.1
DATE EQU P1.2
JTQ EQU P1.4
CE EQU P1.0
```

```
RS EQU P3.3
CD EQU P1.3
TRANSMIT：CLR JTQ                ；吸合继电器
CLR RS
CLR TA1
CLR TA2                          ；1200 b/s 发送模式
LCALL DTMFNO                     ；拨号，双音多频 CLR EA
MOV R1，#0
JNB CD，TRAN                     ；等待回音信号
TR1： LCALL D20MS                ；延时 20 ms
JNB CD，TRAN                     ；对方无响应，返回
INC R6
CJNE R1，#0FFH，TR1
SETB JTQ
SETB TA2
SETB EA
RET
TRAN： JNB CD，TRAN              ；等待对方回音结束
MOV A，#0AAH
MOV SBUF，A                      ；发#0AAH，以开始数据发送
JNB TI，$
CLR TI
LCALL DATEOUT                    ；发数据
SETB JTQ
SETB TA2
SETB RS
SETB EA
RET
DATEOUT：MOV R1，#50H              ；50H～6FH 为待发数据区
DATE1： MOV A，@R1
MOV SBUF，A
JNB TI，$
CLR TI
INC R1
CJNE R1，#70H，DATE1
RET
DTMFNO：MOV A，#6                ；被叫分机号在这里为 68
LCALL DTMF
LCALL DELAY
```

```
MOV A，#8
LCALL DTMF
LCALL DELAY
RET
DTMF： MOV R0，#0          ；拨号子程序
DAL： SETB CLK
RRC A
MOV DATA，C
LCALL D50μs
CLR CLK
LCALL D50μs
INC R0
CJNE R0，#5 DLA
SETB CLK
RET
D50μs：MOV R7，#20
D11： NOP
NOP
DJNZ，R7，D11
RET
```

图 2.4.19　**发送响应**

4. **典型应用**

由嵌入式 MODEM **组成的通讯系统在现代城市交通控制中已得到应用，它可把路口交通信号控制机与指挥中心的控制系统连接起来，并将路口的交通数据及时发送给指挥中心，以便中心计算机能够对这些数据进行综合分析，根据其处理结果来调节路口信号的绿信比、周期、相位差，并通过路口信息显示板，引导车辆，从而达到平衡交通流量，提高车辆运行速度的目的。**

2.4.5　实例五：基于 MODEM 的单片机与 PC 机之间的远程通信

随着计算机技术、通信技术的发展和成熟，数据通信已经成为一种广泛应用的通信方式，它是利用通信系统将数字、字母及字符等以二进制形式在计算机之间进行传输、交换和处理。数据通信可以在两台及两台以上的 PC 机之间、PC 机与单片机之间以及单片机与单片机之间进行，通过通信通道(如公用电话网、载波通道、光纤通道、微波通道、卫星通道)将两机联结。目前，单片机以其高性能价格比、高可靠性广泛用于自动监视、测量、控制等技术领域。单片机主要用作从机，安装在监视、测量和控制现场，而 PC 机则用作主机，安装在条件优越的环境(如温度和湿度适合、几乎无干扰源)中。单片机与 PC 机之间利用公用电话网通过调制解调器 MODEM 实现远程数字通信，其原理框图如图 2.4.20 所示。在实际中，PC 机和单片机之间通信距离近的小于 1 km，远的可达上千 km。

图 2.4.20　单片机与 PC 机基于 MODEM 通信的原理框图

1. PC 机串行通信结构

PC 机与单片机之间的通信一般采用串行异步通信方式。在 PC 机中设置四个(COM1、COM2、COM3、COM4)或两个(COM1、COM2)符合 RS－232C 接口标准的串行口(以下均以两个串行口来说明)。其中 COM2 为 25 针的连接器，COM1 为 9 针的连接器。

为实现异步通信，在 PC 机里设置了通用异步接收器和发送器，即 UART 通过编程可以设定通信格式和速度。PC 机中 UART 的电平为 TTL 电平，而串行口的电平为 RS－232C 的电平，为此 PC 机发送出去的数据要经电平转换器转换为 RS－232C 电平；PC 机接收的数据要经电平转换器转换为 TTL 电平。PC 机串行通信的硬件结构如图 2.4.21 所示。

图 2.4.21　PC 机串行通信的硬件结构

2. 单片机串行通信结构

在单片机中，一般只设置一个为 25 针或 9 针连接器的串行口，由于单片机 8098 中的电平为 TTL 电平，为了要经过 RS－232C 实现异步通信，也要用一片 1488 和两片 1489 来进行电平转换。单片机串行通信的硬件接口如图 2.4.22 所示。请求发送信号(RTS)和数据终端就绪信号(DTR)由单片机 8098 经 8255 的 A 口送出去。清除发送 CTS、数据设备就绪 DSR、载波检测 DCD 和振铃指示 RI 由 8255 的 B 口送入 8098。

图 2.4.22　单片机串行通信的硬件结构

3. 单片机与 PC 机之间的硬件接口

当单片机与 PC 机距离很近时(15 m 以内)，它们之间的数据通信可以通过其串行口以 3 线或多线直线相连；当距离很远(几十公里甚至上千公里)时，可以利用调制解调器 MODEM 通过公用电话网来实现数据通信，这时单片机与 PC 机之间的硬件接口如图 2.4.23 所示。

图 2.4.23　单片机与 PC 机基于 MODEM 通信的硬件接口

4. 软件设计与实现

下面以 PC 机呼叫单片机或单片机应答 PC 机为例来说明程序的设计。

(1) PC 机的程序设计。程序框图如图 2.4.24 所示。

① 初始化。为了实现异步串行通信，在 PC 机中要对其异步发送接收器 UART 进行初始化，以决定 PC 机异步串行通信的数据格式、传输速率、控制方式等。

例 2.4.5

```
outportb(COMU+30，0X83)      /*使 DLAB=1*/
outportb(COMU，0X60)         /*波特率=1*/
outportb(COMU+1，0X00)
outportb(COMU+3，0X03)       /*数据格式*/
outportb(COMU+40，0X03)      /*禁止 MODEM 循环反馈*/
```

```
outportb(COMU，0X00)          /*禁止中断*/
```

说明：COMU **为串行口端口地址。**

② **初始化** MODEM。

例 2.4.6

```
{    int key，i；
char *at1="ATZ"；
char *at2="ATE1Q0V1L3X4S0=1"；
for(i=0；i<=3；i++)
{
if(i<=2)key=at1[i]；          /*发出 at 命令*/
if(i==3)key=0x0d；            /*确认码*/
outportb(COMU，key)；         /*发出数据*/
delay(100)；                  /*延时 100 ms*/
}
delay(1000)；                 /*延时 1000 ms*/
for(i=0；i<=0；i++)
{
if(i<=15)key=at2[i]；
if(i==16)key=0x0d；
outportb(COMU，key)；
delay(100)；
}
delay(1000)；
}
```

图 2.4.24　PC **机程序流程图**

③ **拨号。**PC **机要实现与单片机进行数据通信，首先通过近程** MODEM **拨打单片机的电话号码，如** 38459620。PC **机首先发出命令，如** at+h0e1v0x2&c1dt，**然后发出电话号码，最后再发出确认码** 0x0d。PC **机通过串行口向近程** MODEM **发送命令或数字，每次只能送出一个字符或数字，而** MODEM **接收并响应需要一定的时间，故** PC **机每发出一个字符或数字都要延时** 100 ms。**当确认码发出后还要延时** 100 ms，**然后检测由近程** MODEM **反馈回** PC **机的回响码。若回响码为** 0X35，**则转入数字接收及处理程序，否则继续检测回响码。**

例 2.4.7

```
char at1="ath0q0v0l3x4&c1&d2dt"；  /*取 info 结构中的电话号码*/
char str=info->telephoneno；       /*电话号码位数*/
int ler1=strlen(str)；
for(i=0；i<=39；i++)
{
if(i>24)
{
```

```
    key=str[k]；                    /*发电话号码*/
    k++；
}
if(i<24)key=*at1；                  /*at 命令*/
if(i==39)key=0x0d；                 /*发确认码*/
outportb(COMU，key)；               /*向串口送数据*/
delay(100)；                        /*延时 100 ms*/
at1++；
}
delay(1000)；
do
{
key=bioskey(1)；
num1=inportb(COMU)；                /*检测回响码*/
delay(1)；
if(num1==0x35)drev( )；             /*回响码为 0x35，则转入数据接收与处理程序*/
}
while(key==0)；
key=bioskey(0)；
if(key==0x011b)return；             /*按 ESC 键则返回*/
```

(2) **单片机程序设计。单片机程序框图如图** 2.4.25 **所示，其程序见例** 2.4.8。

例 2.4.8

```
LD 72H，#4003H      ；8255 命令口地址
LD 70H；#0082HH     ；A 口为输出，B 口为输入
STB 70H，[72H]
LDB 16H，#20H       ；串口初始化
LDB 11H，#09H       ；方式 1
LDB 0EH，#4DH       ；波特率为 1200 b/s
LDB 0EH，80H
LDB 18H，#0CH       ；堆栈
ORB IOC1，#20H
LD 72H，#4000H      ；8255 的 A 口地址
LD 70H，#0003H      ；使 RTS 和 DTR 均为高电平
STB 70H，[72H]
LCALL COMZ0         ；连续发出四个 0
LCALL DT            ；延时 1000 ms
LCALL COMZ1         ；初始化 MODEM，发出 at 命令：ata
LCALL DT
```

图 2.4.25　单片机程序流程图

```
LCALL COMZ2        ；初始化 MODEM，发出 at 命令：ate1q0v1l3x4s0=1
LCALL DT
LCALL COMZ3        ；初始化 MODEM，发出 at 命令：atS7=30
LCALL DT
RING0：LDB SPCON，#09H
RING1：LDB 60H，SPSTAT
JBC 60H，6，RING1
ANDB 60H，#0BFH
LDB 70H，SBUF
CMPB 70H，#80H
JE TD
LJMP RING1
```

说明：

由于单片机向 MODEM **发出** at **命令的程序是相似的，故在此仅以** at **命令** ate1q0v1l3x4s0=1 **为例来加以说明。假设** at **命令存放在以** 0AB40H **为首地址的外部存储器中。程序如下：**

```
COMZ2：  LD 72H，#0AB40H      ；at 命令首地址
LDB 20H，#17H
COMZ2A：LDB 70H，[72H]        ；取 at 命令
LDB SBUF，70H                 ；向串口送 at 命令
DJNZ 20H，COMZ2A
LDB 70H，#0DH
LDB SBUF，70H                 ；发出确认码
RET
```

第 3 章　数据通信中的编/解码技术及应用

本章主要内容：

★ DTMF 编/解码技术
★ 三态逻辑编/解码技术
★ 红外遥控技术
★ 差错控制技术

3.1　DTMF 编/解码技术

3.1.1　DTMF 编/解码技术基础知识

1. DTMF(双音多频)编码方法

DTMF 是英文 Dual Tone Multiple Frequency 的缩写，意为“双音多频”，它在程控系统中应用最为广泛。

DTMF(双音多频)信令具有的传递速度，使得它不仅广泛应用于电话系统的语音通信中，而且在通信网中应用也极为普遍。一些系统中常常需要同时接收和发送 DTMF 信号，发送和接收均伴随着编码和解码过程。

电话机有两种拨号方式，即脉冲拨号方式和双音多频拨号方式。

双音多频拨号方式中的双音是指用两个特定的单音信号的组合叠加来代表数字或符号(功能)。两个单音的频率不同，所代表的数字和功能也不同。在双音多频电话机中，有 16 个按键，其中有 10 个数字键(0～9)，6 个功能键(*、#、A、B、C、D)。按照组合的原理，它必须有 8 种不同的单音频信号。由于采用的频率有 8 种，故称之为多频。又因从 8 种频率中任意抽出两种进行组合，又称其为 8 中取 2 的双音编码方法。

根据 CCITT 的建议，国际上采用 697 Hz、770 Hz、825 Hz、941 Hz、1209 Hz、1336 Hz、1477 Hz 和 1633 Hz。把这 8 种频率分成两个群，即高频群和低频群。从高频群和低频群中任意各抽出一种频率进行组合，共有 16 种不同的组合，代表 16 种不同的数字或功能，见表 3.1.1。

例如按“1”键时，由拨号电路产生 697 Hz 与 1209 Hz 叠加的信号电流输出；按“2”键时，产生 697 Hz 与 1336 Hz 的叠加输出；以此类推。

表 3.1.1　拨号数字与高、低频率的组合关系

低频组 f_L/Hz	高频组 f_H/Hz			
	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	0	*	#	D

2. 双音多频(DTMF)编解码器的原理

双音多频(DTMF)编码器是采用每位数字由一组低频和一组高频(f_L 和 f_H)按一定的组合叠加形成的一组双音多频率信号($\cos 2\pi f_L t+\cos 2\pi f_H t$)，实现快速数字拨号。一般拨号数字与高、低频率的组合关系如表 3.1.1 所列。

DTMF 解码器一般包括 DTMF 分组滤波器和 DTMF 译码器。DTMF 接收信号先经高、低群带通滤波进行 f_L/f_H 区分，然后过零检测、比较，得到相应于 DTMF 的两路 f_L、f_H 信号输出。该两路信号经译码、锁存、缓冲，恢复成对应于 16 种 DTMF 信号音对的 4 比特二进制码(D1～D4)，如表 3.1.2 所列。

表 3.1.2　16 种 DTMF 信号音对与二进制码的关系

f_L/Hz	f_H/Hz	数 字	二 进 制 码			
			D4	D3	D2	D1
697	1209	1	0	0	0	1
697	1336	2	0	0	1	0
697	1447	3	0	0	1	1
770	1209	4	0	1	0	0
770	1336	5	0	1	0	1
770	1447	6	0	1	1	0
852	1209	7	0	1	1	1
852	1336	8	1	0	0	0
852	1447	9	1	0	0	1
941	1209	0	1	0	1	0
941	1336	*	1	0	1	1
941	1447	#	1	1	0	0
697	1633	A	1	1	0	1
770	1633	B	1	1	1	0
852	1633	C	1	1	1	1
941	1633	D	0	0	0	0

目前 DTMF 产品多属于 CMOS 集成电路，国际上一些主要器件生产厂家或公司均有这方面的系列产品。有代表性的 DTMF 发送器包括 MITEL 公司的 MT5087、MT5089、MT5088、MT5091，Motorola 公司的 MC14410，AMI 公司的 S2860、S2559，TEXAS 公司的 TP5087、TCM5087、TCM5089，MOSTEK 公司的 MK5087、MK5089、MK5091 等。有代表性的 DTMF 接收器有 MT8870、M8870、MH88305 与 M957。有的 DTMF 发送器和接收器集成为一体，

如 MT8880、M8880、M8888、MT8888、MT8889 等。这些 DTMF 产品集成度高、体积小、抗干扰能力强，并且中间传输的是两个叠加的音频信号，最后输出的是二进制编码信号，便于与微型计算机接口，无需调制解调器。目前 DTMF 主要用于电话机及程控交换机中。实际上，DTMF 编译码电路可广泛用于遥控、遥测和数据传输等领域中。

3.1.2　DTMF 远程通信的软硬件实现技术及其应用

随着计算机技术和电信业的发展，通过电话线进行的远程通信越来越常见。在通信数据量不大、对通信速率要求不高的应用场合，可以采用 DTMF 通信方式，它具有接口简单、成本低廉且可靠性高的特点。下面分别论述其硬件、软件实现技术。

1. 硬件实现技术

(1) 通信接口电路设计。通信接口电路如图 3.1.1 所示。该电路中话机与接口电路并联，通过光耦输出电平检测用户是否摘机。用户摘机后通过 LINE1、LINE2 直接收码，降低了接口电路对拨号的影响。数据通信时 MPU 通过 I/O1 控制继电器断开话机，同时 I/O4 置高，电路模拟摘机，三极管组成恒流源维持摘机状态。在通信中断开话机可减少干扰，恒流源设计可保证电路具有较小的直流阻抗(小于 300 O)和较大的交流阻抗(大于 600 O)，使电路具有较好的收发码特性。

图 3.1.1　接口电路

LINE1、LINE2 间接入压敏电阻或瞬态抑制二极管可达到抗雷击保护作用。I/O2、I/O3 输出电平与相关软件配合可实现脉冲拨号接收和反极信号检测。

(2) 发码电路设计。发码电路如图 3.1.2 所示。该电路采用廉价的电话 DTMF 发生器 4087 芯片，它具有性能优良、接口简单的特点。用一片 373 代替键盘编码芯片来模拟按键，DTMF 发码电路使用芯片放大电路，片外采用 9014 作开关，发码时 9014 导通，120 Ω电阻与片内电路起输出放大作用，不发码时 9014 截止，可减少 4087 对收码电路的影响。

(3) 收码电路设计。收码电路如图 3.1.3 所示。该电路采用常规 8870 芯片，电路放大倍数取 3，在 IN-2 端接入 100 pF 电容可有效改善 8870 对 DTMF 中高频分量的接收。Q1～Q4 为数字量输出，可方便与 MPU 接口。

图 3.1.2　发码电路

图 3.1.3　收码电路

(4) 450 Hz 信号检测电路。该检测电路如图 3.1.4 所示。该电路采用 LM567 构成锁相环对线路中 450 Hz 信号检测，I/O8 为输出信号。该电路与相关软件配合可实现对拨号音、忙音、回铃音的检测。

图 3.1.4　450 Hz 信号检测电路

(5) 振铃与防盗检测电路。这部分电路如图 3.1.5 所示。该电路采用 LM339 电压比较器，当 I/O6 输出为高时有振铃信号。当用户没有摘机且 I/O5 输出为低时可判断有盗打行为。

图 3.1.5　振铃与防盗监测电路

2. 软件实现技术

(1) 从机发起通信程序设计。从机发起通信程序框图如图 3.1.6 所示，程序功能如下：

图 3.1.6　从机发起通信程序框图

① 在通信中断开话机可减少干扰，提高通信可靠性；

② 摘机后不能立即拨号，可延时或检测到拨号音后再拨号；

③ 通信中不允许无限等待，可限时接收，超时应退出通信。

(2) 从机应答主机程序设计。从机应答主机的程序框图如图 3.1.7 所示，程序功能如下：

① 判断是用户呼出摘机还是外线呼入用户接听摘机；

② 判断是主机呼入还是他人呼入；

③ 判断是用户正常拨号还是用户完成功能设置；

④ 具有振铃检测和自动摘机功能。振铃 4 次无人接听后电路即自动摘机。

图 3.1.7　从机应答主机程序框图

(3) 数据通信程序设计。数据通信程序框图如图 3.1.8 所示，程序功能如下：

① 采用固定格式报文方式，方便接收；

② 采用简单校验手段，实验发现 DTMF 通信中容易出现漏码，而重码、误码较少出现，所以采用固定字节接收方式和简单异或校验方式即可实现可靠通信；

③ 出错重发一次可提高通信成功率。

图 3.1.8　数据通信程序框图

(4) 发送码表与发码程序设计。373 输入值(HEX)与对应的 DTMF 输出见表 3.1.3。当输入为 F0H 时，4087 停止输出。

表 3.1.3　HEX 与 DTMF 对应表

HEX	DTMF	HEX	DTMF	HEX	DTMF	HEX	DTMF
78	0	D1	4	B2	8	74	C
E1	1	D2	5	B4	9	E8	D
E2	2	D3	6	72	A	D8	E
E4	3	B1	7	71	B	B8	F

标准发码程序为发送 100 ms 停发 100 ms，发码速率为 5 码/s。为提高发码速率可适当减少发送和停止时间，但停发不能少于 50 ms，所以最高发码速率可达 10 码/s。注意拨号时必须采用标准发码，否则交换机不会识别。接通后根据线路状况可适当提高发码速率。通信中也可采用自适应策略，根据误码率自动调整发码速率，从而达到最佳通信效果。

3. 技术应用

本套软硬件实现技术具有接口电路简单、可靠性高、成本低、灵活性强等优点，适用于数据通信量不大，速率要求不高的远程通信场合。通信中任一方均具有拨号音检测、振铃检测、自动摘机、拨号和数据通信功能，可自动实现语音通话与数据通信识别，并能双向呼叫，可应用于远程分布式数据采集系统、家用自动防盗报警装置、远程室内监控系统以及公话集中管理系统等。

该套技术现已成功应用于 JJF69 型公话集中管理系统，通信接口各项技术指标和软件各项功能均通过邮电部入网检测，经过实际运行证明该技术具有较好的推广应用价值。

3.1.3　DTMF 编/解码芯片在遥控系统中的应用

本节以 MT5087(DTMF **发送器)、**MC145436(DTMF **接收器)为例，介绍** DTMF **在无线控制系统中的应用实例。当用于有线遥控时，接法和电话线一样；当用于无线遥控时，将** DTMF **发送器产生的音频信号作为调制信号，只要能传输话音信号，就能传输** DTMF **输出的** 4 **位二进制码。**

1. **由** DTMF **构成的遥控发射电路**

由 MT5087 **发送器组成的遥控发射电路如图** 3.1.9 **所示。图中，**CD22100 **相当于电话机的按键，用于产生** 16 **个开关信号，它由** 4 **线至** 16 **线译码器、**16 **个控制存储器和排列成** 4 **行×**4 **列的交叉点开关矩阵组成，**A、B、C、D **为地址输入选择端，以选择不同的交叉点开关。**CD22100 **交叉点开关寻址真值表如表** 3.1.4 **所列。在图** 3.1.9 **中，用单片机** 8031 **的** P0 **口控制** CD22100 **的地址单元，以产生所需的** 16 **个** DTMF **编码信号。在实际应用中，控制盒和数据选择器用来选择不同的遥控指令。在下面将要介绍的“一点多路遥控接收电路”中可以是一个控制盒，在“多点多路遥控接收电路”中可以有多个控制盒。**

图 3.1.9　**遥控发射电路**

表 3.1.4　CD22100 **交叉点开关寻址真值表**

地址				选择通路	地址				选择通路
D	C	B	A		D	C	B	A	
0	0	0	0	Y1-X1	1	0	0	0	Y1-X3
0	0	0	1	Y2-X1	1	0	0	1	Y2-X3
0	0	1	0	Y3-X1	1	0	1	0	Y3-X3
0	0	1	1	Y4-X1	1	0	1	1	Y4-X3
0	1	0	0	Y1-X2	1	1	0	0	Y1-X4
0	1	0	1	Y2-X2	1	1	0	1	Y2-X4
0	1	1	0	Y3-X2	1	1	1	0	Y3-X4
0	1	1	1	Y4-X2	1	1	1	1	Y4-X4

2. **由** DTMF **构成的遥控接收电路**

由于 DTMF **电路只能输出** 4 **位二进制码即** 16 **个开关量，而且每次只有一路开关量有**

效，如果直接将它应用于遥控系统中，往往开关量不够用，也不能同时控制多个对象。为此，我们设计和调试出了一个较实用的 DTMF 接收扩展电路。通过这个扩展电路可以实现一点多路(可输出 150 个开关量)和多点多路(可同时控制 10 个对象，每个对象输出 15 个开关量)遥控，下面分别加以说明。

(1)“一点多路”DTMF 遥控接收电路。由 MC145436 DTMF 接收器构成的“一点多路”DTMF 遥控接收电路如图 3.1.10 所示。

图 3.1.10 “一点多路”DTMF 遥控接收电路

图中，CD4082 为 4 输入与门，主要完成复位功能，使遥控接收电路重新开始接收控制命令；CD4017 为十进制计数器，用于控制路号，本遥控接收电路共有 10 路，每路可输出 15 路开关量信号；CD4076 为 4 位 D 型寄存器，用于数据的锁存和输出。图中，CD4082、CD4017 和 CD4076 组成 DTMF 接收扩展电路。

“一点多路”DTMF 遥控接收电路的工作原理如下：

先复位，方法是让 MC145436 输出全 1，CD4082 输出 1。由于 CD4017 的复位端 R 为 1，因此 CD4017 的译码输出被清零，CD4076 的全部输出截止于高阻态。由于 MC145436 有一个输出数据有效端 DV，当它译码有效时，DV 输出为高电平，我们利用它作为 DTMF 接收扩展电路的时钟，用于计数。当 MC145436 译出第 1 组数据时，DV 输出第 1 个脉冲，计数器 CD4017 的 Q1 允许第 1 路的 CD4076 数据输出，第 1 路的控制指令有效。同理，当 MC145436 译出第 2 至 10 组数据时，DV 输出第 2 至 10 个脉冲，计数器 CD4017 的 Q2 至 Q10 允许第 2 至 10 路的 CD4076 数据输出，第 2 至 10 路的控制指令有效。一帧控制指令码的结构如下：

帧头(15) 复位码	目标 1 的 指令码	目标 2 的 指令码	… …	目标 10 的 指令码

每一帧指令码的形成完全由发射端的单片机 8031 根据控制盒和数据选择器产生，帧头不变，一直为 15(即 MC145436 的 4 位二进制输出为全 1)。每一路的指令码为 0～14 中的任何一个，若 10 路同时控制一个控制对象，可同时输出 150 个开关量，完全能够满足一般遥控目标的控制要求。

(2) “多点多路”DTMF 遥控接收电路。由 MC145436 DTMF 接收器构成的“多点多路”DTMF 遥控接收电路如图 3.1.11 所示。“多点多路”DTMF 遥控接收电路的工作原理与“一点多路”DTMF 遥控接收电路的工作原理基本相同，只是将控制一个目标的 10 路信号输出改为控制 10 个目标，每个目标可输出 15 个开关量信号。而且，10 个目标的接收机共用一套遥控发射电路，可对 10 个目标进行集中管理控制。计数器 CD4017 的 Q1 允许第 1 个目标的 CD4076 数据输出，第 1 个目标的控制指令有效。同理，计数器 CD4017 的 Q2 至 Q10 允许第 2 至第 10 个目标的 CD4076 数据输出，第 2 至第 10 个目标的控制指令有效。一帧控制指令码的结构如下：

帧头(15) 复位码	目标 1 的 指令码	目标 2 的 指令码	… …	目标 10 的 指令码

图 3.1.11　“多点多路”DTMF 遥控接收电路

3.1.4　几种新型 DTMF 编/解码芯片的应用实例

1. 用 DTMF 解码芯片 CSC5087 和 SC8870 实现单片机遥控键输入

单片机问世以来，在仪器、仪表、智能控制领域得到了广泛应用。绝大多数单片机应用系统(SCAS)都少不了键输入控制，通常使用的单片机系统按键可装在面板上，但对于那些工作于控制现场以及高低温、多灰尘场合的单片机应用系统，按键的寿命将会缩短，故障率也会大增。为提高单片机应用系统键输入的可靠性及耐久性，有关专家设计出了单片机遥控键输入电路，键码是通过音频感应方式输入 SCAS 系统的，且可实现一机多用，即一个键盘遥控器可对任意多个相同的 SCAS 系统进行键输入操作。

(1) 感应原理及电路结构。遥控键电路结构分为遥控器和译码接收电路两部分，其结构如图 3.1.12 所示。

图 3.1.12　键输入电路结构框图

遥控器部分对 16 个按键进行编码，对应每个按键都产生了一个唯一的双音频信号，此双音频信号由扬声器以声音形式发出。在译码接收电路中，话筒将遥控器发出的双音频声音信号接收下来，并送至译码器电路，译码器电路进行正确译码后输出一个二进制代码，此码即是由遥控器输入的按键代码。此外译码器还同时输出一个正确译码的标志信号，由此信号向 MCU 发出中断请求，MCU 就可读取键码并根据代码执行相应的程序模块。

(2) 电路设计原理。

① 硬件电路设计。遥控键输入电路如图 3.1.13 所示。电路核心是双音多频(DTMF)编解码芯片 CSC5087 和 SC8870。CSC5087 作为 DTMF 信号编码器。它可根据不同的按键产生一组双音频信号 $\cos 2\pi f_L t+\cos 2\pi f_H t$。国际电报电话咨询委员会(CCITT)和我国的标准规定按键与高、低频组频率的组合关系如表 3.1.1 所列。如按下按键“ 6 ”，发出的 DTMF 信号频率为 f_L=770 Hz、f_H=1477 Hz，此 DTMF 信号经音频功放 LM386 放大后由扬声器转换为声音信号发出。

由话筒接收并放大的 DTMF 信号经 SC8870 解码，将每一个 DTMF 信号译成一个 4 位二进制代码输出，16 个 DTMF 信号分别对应 0000～1111 共 16 个二进制代码，其对应关系如表 3.1.5 所列。如对遥控器发出频率为 f_L=770 Hz 和 f_H=1477 Hz 的 DTMF 信号，SC8870 译码后输出 0110 代码。

SC8870 有一延迟控制输出端 CID，若检测到一有效的 DTMF 信号，控制输入端 STO 电平超过门限电平 V_{Tst}，输入代码被更新，此时 CID 输出由低电平变为高电平；若 STO 低

于 V_{Tst}，则 CID 返回至低电平。而 STO 电平则由初始控制输出信号 ECO 决定，当 SC8870 检测到一有效的 DTMF 信号时，ECO 首先变为高电平，再经电阻使 STO 电平升高。在无输入的 DTMF 信号或输入信号连续失真时，ECO 输出低电平，这样 STO 也为低电平，CID 输出低电平。利用 CID 信号作为 MC68HC705 MCU 的中断请求信号，因 MCU 的中断触发为下降沿触发，故将 CID 信号经反相器反相后接入 MCU 的中断请求输入端 IQR。

图 3.1.13　遥控键输入电路图

SC8870 的 DO1～DO4 分别接 MCU 的口线 PA0～PA3，三态数据输出允许控制端 EN 接高电平，使 DO1～DO4 保持上次对 DTMF 信号的译码输出代码，这样 MCU 随时可读入输入的键值。

② 键输入中断和键码接收软件。软件流程如图 3.1.14 所示。MCU 由中断响应程序接收键码返回主程序后，主程序根据键码决定执行何种功能模块。

(3) 应用。由于 DTMF 信号具有编、译码可靠性高，传输误码率低等特性，因此本节所介绍的键输入电路具有抗干扰能力强的特点，已成功地应用于城市路灯微机监控系统中，所有开关柜测控子系统的键码都是采用这种方式输入，它们共用一个遥控器，这样使系统抗干扰性、抗灰尘、抗老化等性能都大为提高。

2. DTMF 解码芯片 MC145436 在公用电话网远程控制系统中的应用

(1) 系统结构。电话远程控制系统主要需要完成的功能是对电话双音多频信号进行解码，并自动驱动被控制电器设备进行指定操作。由于电话远程控制系统是利用电话进行控

表 3.1.5　SC8870 译码表

f_L/Hz	f_H/Hz	DO4	DO3	DO2	DO1	按键
941	1633	0	0	0	0	0
697	1209	0	0	0	1	1
697	1336	0	0	1	0	2
697	1477	0	0	1	1	3
770	1209	0	1	0	0	4
770	1336	0	1	0	1	5
770	1477	0	1	1	0	6
852	1209	0	1	1	1	7
852	1336	1	0	0	0	8
852	1477	1	0	0	1	9
941	1209	1	0	1	0	A
941	1336	1	0	1	1	B
941	1477	1	1	0	0	C
697	1633	1	1	0	1	D
770	1633	1	1	1	0	E
852	1633	1	1	1	1	F

图 3.1.14　键输入中断和键码接收软件流程

制的系统，因此，系统必须能识别电话的振铃信号，并能自动摘机和挂机。由于电话远程控制系统一般在无人值守的情况下工作，因此，必须能自动开机和关机，并且在用户出现误操作时，必须能自动复位及关机。一般被控制的电器设备有可能是强电驱动的电器，因此，真正控制电器设备开关的电路由继电器实现。另外，系统主要由集成电路和模拟电路组成，因此，必须设置直流电源电路，以提供系统正常运作所需的电力。

根据以上要求，电话远程控制系统主要设置了电话双音多频(DTMF)信号解码电路、系统控制电路、4/16 译码器、驱动电路、继电器开关电路、系统开启电路、系统关闭电路、电话摘机控制电路、电话挂机控制电路、自动复位电路和电源电路等，如图 3.1.15 所示。

图 3.1.15　电话远程控制系统结构示意图

(2) 电话双音多频信号解码。本系统采用 Motorola 公司的 MC145436 双音调多频接收机作电话双音多频解码核心(见图 3.1.16)。

MC145436 是硅栅 CMOS 大规模集成电路，包括有滤波器和译码器，用于检测一对音调是否符合十六进制输出双音多频标准。开关电容滤波器技术用于定时控制和输出电路的

数字化信号。MC145436 具有优良的电源线噪声指标和拨号音的抑制性能，很适合远端控制设备的电话双音多频信号的解码工作。

图 3.1.16　MC145436 原理方框图

利用 MC145436 及电话耦合电路、DTMF 信号放大电路，可构成一个电话双音多频信号的解码电路，如图 3.1.17 所示。

图 3.1.17　电话双音多频信号解码原理图

由电话线上传来的双音多频及电话直流供电混合信号，经过耦合器 T，滤除了电话线上的直流信号；然后，再将该信号送入放大器，将双音多频信号进行放大；之后，经过一个耦合电容，送入 MC145436 双音多频解码芯片。经 MC145436 芯片解码后，DV 信号变为有效(高电平)，同时输出 4 位代码(D8，D4，D2，D1)。4 位代码与电话键盘上按键的关系如表 3.1.6 所示。

表 3.1.6　MC145436 输出代码与电话键盘上按键的关系

按键	输出代码				按键	输出代码				按键	输出代码				按键	输出代码			
	D8	D4	D2	D1		D8	D4	D2	D1		D8	D4	D2	D1		D8	D4	D2	D1
1	0	0	0	1	5	0	1	0	1	9	1	0	0	1	A	1	1	0	1
2	0	0	1	0	6	0	1	1	0	0	1	0	1	0	B	1	1	1	0
3	0	0	1	1	7	0	1	1	1	*	1	0	1	1	C	1	1	1	1
4	0	1	0	0	8	1	0	0	0	#	1	1	0	0	D	0	0	0	0

(3) 系统开启和关闭电路。由于电话远程监测控制系统一般都放置在无人值守的环境下，因此，在不使用的情况下，系统应处于关闭状态；另外，当出现误操作时，系统应可以自动复位。因此，在电话远程控制系统中，设置了系统开启电路、系统关闭电路、自动复位开关电路。

① 系统开启电路。电话远程控制系统利用电话振铃信号作为系统开启信号，其实现方法如图 3.1.18 所示。

图 3.1.18　系统电源开启示意图

当电话振铃信号到来时，电话线路上的 90 V 振铃交变信号经耦合器 T 后，再经过一个桥电路 B 及滤波电路后，变成一个直流信号。该直流信号加在可控硅 Q 的 G 端上，打开可控硅 Q。动力电(220 V 交流)经可控硅 Q 后，驱动系统上的直流电源为整个系统提供电源，此时，系统的电源又反过来保持可控硅 Q 一直处于开启状态，从而保持整个系统处于开启工作状态。

② 系统关闭电路。系统关闭电路主要执行以下功能：当用户使用完控制系统后，需要关闭系统时，发送一个“关闭系统”命令，系统即自动关闭；另外，在出现误操作，如用户挂机时，未先发“关闭系统”命令，或一个非法用户无意中打开了系统，系统都可以利用自动复位开关送来的关机命令关闭系统。关闭系统电路如图 3.1.19 所示。

图 3.1.19　系统电源关闭示意图

当用户发出的关闭系统命令或系统自动复位关机命令到来时,通过与非门 U1 后,驱动继电器 KR 接通,使可控硅 Q 的 K、A 两端短路;当用户发出的关闭系统命令或系统自动复位关机命令失效时,通过与非门 U1 输出 0 电平,继电器 KR 断开,使可控硅 Q 的 K、A 两点开路,从而使可控硅 Q 进入关闭状态,切断动力电,关闭整个系统。

③ 自动复位开关电路。当用户挂机前,未关闭系统时,自动复位开关经一设定延迟后,会自动发出关机命令,关闭整个系统。自动复位开关电路如图 3.1.20 所示。

图 3.1.20　自动复位开关电路

自动复位开关的延迟时间由 R、C 组成的电路控制。当用户正在进行操作时,每当按下电话键盘上的一个按键,则 MC145436 的第 12 端(DV)变为高电平,该信号作为自动复位开关电路的 Sin 信号,快速对电容 C 进行充电。当用户松开电话按键,则 DV 变为低,即 Sin 为低,此时,电容 C 通过电阻 R 进行放电。当电容 C 上的电压(即芯片第 4 脚的电压 U_4)低于芯片第 5 脚的电压 U_5 上的电压时,Sout 输出一个低电平,即自动复位信号变为有效。

(4) 控制命令处理电路。由 MC145436 电话双音多频芯片输出的 4 位数字代码,根据系统安排,分别送至系统控制电路和控制命令处理电路。

控制命令处理电路的主要功能是将并行的数字控制信息(即 4 位数字代码)处理成对应的控制命令,并且实现弱电控制命令与强电控制命令之间的转换功能。控制命令处理电路具体包括三个部分,即 4/16 译码器、驱动电路和继电器开关电路。

① 4/16 译码器。4/16 译码器主要实现 4 位并行数字代码转换成对应的 16 位控制功能,可以由一块芯片,如 74LS4514 来实现。每当 4 位并行数字代码有效时,74LS4514 的 16 个输出端中即有一个唯一的输出端输出为 1,其余皆为 0。

② 驱动电路。驱动电路主要实现驱动继电器、命令复用和命令锁定三个功能。

驱动继电器。当一个 4/16 译码器被用作多个控制命令时，由一个 74LS4514 芯片驱动可能出现电力不足的情况，因此，设置驱动电路为继电器电路提供电力。

多命令选择。当 4/16 译码器的某个输出被用作多个控制命令时，可以将其接到不同的驱动门上，如 74LS245 等，选通某个驱动门，则该驱动门对应的命令有效，其它驱动门对应的命令无效。

命令锁定。命令锁定主要实现命令保持功能，即每接收到一次命令，就打开继电器或关闭继电器。命令锁定功能可以用 74LS74 芯片实现。

③ 继电器开关电路。继电器属强电电路，不能直接用集成电路芯片驱动，为此，在集成电路芯片与继电器之间必须设置一个驱动继电器的电路。本系统利用分立三极管的截止和饱和两个状态，来关闭继电器或打开继电器开关，其电路如图 3.1.21 所示。

图 3.1.21　继电器开关电路

本节介绍的电话远程控制系统，可以利用电话机方便地实现远程电器设备的控制操作，例如作为家用电器的远程控制器使用，使用者在任何地方，都可以使用电话机实现对居所的各种家用电器开关电源或其它的控制。系统实验表明，采用电话机作控制器，采用电话双音多频信号作为控制信号，可以可靠地实现远程系统的控制和操作。

3. DTMF 收发器 MT8888 及其应用

(1) 基本功能。MT8888 是一种具有 Intel 微处理器接口的功能较强的双音多频发送和接收器件，可用于寻呼系统、交换机系统和移动通信、
转账卡系统、互接拨号器、数字通信和计算机等领域。
其主要功能有：

① 完整的 DTMF 发送和接收功能；

② 高速 Intel 微处理器接口；

③ 可工作于自动音频突发模式；

④ 可调整保护时间；

⑤ 呼叫音检测到–30 dBm。

MT8888 引脚排列如图 3.1.22 所示。

图 3.1.22　MT8888 引脚图

说明：

IN＋、IN－（1，2）——运放的同相和反相输入端。

GS（3）——增益选择端。在该引脚与 IN－引脚之间接反馈电阻，可调节运放增益。

Vref（4）——基准电压输出端。通常为 VDD/2，作为运放的偏置电压。

VSS（5）——芯片电源负端，接地。

OSC1、OSC2（6、7）——时钟或振荡器的输入、输出端。两引脚间接 3.579545 MHz 晶体与内部电路构成芯片振荡器；若由外部电路提供时钟，则 OSC2 引脚开路。

TONE（8）—— DTMF 信号输出端，也可通过编程设置为单音输出。

$\overline{\text{WR}}$（9）——微处理器写输入端，低电平有效，与 TTL 电平兼容。

$\overline{\text{CS}}$（10）——片选信号输入端，低电平有效。该引脚可由微处理器的地址锁存信号（ALE）直接提供。

RS0（11）——寄存器选择控制输入端。

$\overline{\text{RD}}$（12）——微处理器读输入端，低电平有效，与 TTL 电平兼容。

$\overline{\text{IRQ}}$/CP（13）——中断请求信号，为开漏输出。在中断模式下，当突然发送或接收一个有效 DTMF 信号时，输出低电平信号。若控制寄存器设定电路工作于呼叫处理（CALL）模式和中断使能，则该端输出代表运放输入的方波信号音，但该信号频率必须落在呼叫处理滤波器的带宽内。

D0～D3（14～17）——数据总线，与 TTL 电平兼容。输入需发送的 DTMF 编码或输出译码的 DTMF 信号数据。当 CS=1 时呈高阻状态。

Est（18）——初始控制输出。若电路检测到一种有效的单音对时，Est 为高电平；若信号丢失，则 Est 返回低电平。

St/GT（19）——控制输入/时间监测输出。若 St 电压大于门限 V_{TSt}，电路寄存被检测的 DTMF 单音对，并更新输出锁存器内容。若 St 电压低于 V_{TSt}，则电路不接收一新单音对，GT 输出的作用是设置外部时间监测常数。

VDD（20）——芯片电源正端，典型值为+5 V。

(2) 工作原理。MT8888 是集 DTMF 发送和接收功能的器件，内带呼叫处理滤波器。接收部分与 DTMF 接收器件 MT8870 类似，发送部分包括行、列计数器和 D/A 变换器，另外增加了一些控制寄存器和接口、数据总线缓冲器，很容易实现与微处理器的直接接口。MT8888 通过微处理器接口可以由 RS0、$\overline{\text{WR}}$、$\overline{\text{RD}}$、D0～D3 等信号选择与设定内部寄存器，并控制电路的工作状态或工作模式。它共有 5 个不同作用的寄存器：发送数据寄存器(TDR)、接收数据寄存器(RDR)、状态寄存器(SR)、控制寄存器 A(CRA)和控制寄存器 B(CRB)，其控制关系如表 3.1.7 所示。

表 3.1.7　内部寄存器控制关系

RS0	$\overline{\text{WR}}$	$\overline{\text{RD}}$	功　能
0	0	1	数据写入 TDR
0	1	0	数据从 RDR 读出
1	0	1	数据写入 SR
1	1	0	数据从 SR 读出

MT8888 共有六种工作模式，下面进行简要介绍。

① DTMF 模式：发送与接收 DTMF 信号。输入数据经 TDR 控制可编程行及列计数器、D/A 变换器，合成需要发送的 DTMF 信号。或者，DTMF 信号经拨号音抑制、分离带通滤波器监频与确认，译成相应的 4 比特码，经 RDR 输至数据总线。DTMF 编译码对应关系如表 3.1.8 所示。

表 3.1.8　DTMF 编译码对应关系

双音频键	0	1	2	3	4	5	6	7	8	9	*	#	A	B	C	D
十进制数	10	1	2	3	4	5	6	7	8	9	11	12	13	14	15	0

② 呼叫处理(CALL)模式：电路可以检测电话呼叫过程中的各种信号音，只要信号的频率落在 320～510 Hz 范围内，片内呼叫处理滤波器便可滤出。经限幅得到的方波信号由 $\overline{\text{IRQ}}$/CP 端输出，以用于微处理器对呼叫性质和类别进行判断。若无信号滤出，则 $\overline{\text{IRQ}}$/CP 端始终保持低电平。

③ 突发(BURST)模式：在 DTMF 模式下工作于突发状态，信号突发和暂停时间各为 51±1 ms；在 CALL 模式下工作于突发状态，信号突发和暂停时间各为 102±2 ms，此时电路只可发送 DTMF 信号，但不能接收。

④ 单/双音(S/$\overline{\text{D}}$)产生模式：电路可产生单音或 DTMF 信号(由 CRB 控制)，用于测试和监测。

⑤ 测试(TEST)模式：使电路从 DTMF 接收部分得到延迟监测信号，并从 $\overline{\text{IRQ}}$/CP 端输出。

⑥ 中断模式：此模式下若选择 DTMF 状态，当 DTMF 信号被接收或出现在监测时间内，或准备发送更多数据(突发模式下)时，$\overline{\text{IRQ}}$/CP 端接至低电平。

各种模式的选择由控制寄存器(CRA 和 CRB)的相应位完成，如表 3.1.9 和表 3.1.10 所示。状态寄存器 SR 各位所表示的关系如表 3.1.11 所示。

表 3.1.9　控制寄存器 A(CRA)的功能

位	符　号	功　　能
b_0	TOUT	信号音输出控制。高电平有效，该位控制所有信号的发送
b_1	CP/$\overline{\text{DTMF}}$	呼叫处理或 DTMF 模式选择。低电平为 DTMF 模式；高电平为 CALL 模式。可检测呼叫信号音，从 $\overline{\text{IRQ}}$/CP 端输出方波(IRQ = 1 时)
b_2	$\overline{\text{IRQ}}$	中断允许位。高电平有效，使电路工作于中断模式
b_3	RSEL	寄存器选择位。高电平时，下一个写周期选 CRB，继而写周期返回选 CRA

表 3.1.10　控制寄存器 B(CRB)的功能

位	符　号	功　　能
b_0	$\overline{\text{BURST}}$	突发模式选择位。低电平选择突发模式。此时数据写入 TDR，产生突发/暂停各为 51 ± 1 ms 的 DTMF 信号，然后更新 SR，使 TDR 准备接收下一指令。若中断允许，则产生中断；若 CALL 模式允许，则产生 102 ± 2 ms 扩展突发信号
b_1	TEST	测试方式控制。高电平设定电路工作于测试方式
b_2	S/$\overline{\text{D}}$	单/双音产生选择位。低电平设定电路产生 DTMF 信号；高电平设定电路列或行(由 C/$\overline{\text{R}}$ 位决定)单音频信号输出
b_3	S/$\overline{\text{R}}$	列或行单音选择。高电平选择列单音输出；低电平选择行单音输出。该位与 S/$\overline{\text{D}}$ 位一起使用

表 3.1.11　状态寄存器(SR)的功能

位	名　称	状态标志设定	状态标志清除
b_0	中断请求	中断发生，b_1 或 b_2 置位	中断禁止，SR 读出后清除
b_1	突发模式下		
TDR 空	暂停时间结束，准备发送新数据	SR 读完数据后清除	
b_2	RDR 满	RDR 已有有效数据	SR 读完数据后清除
b_3	延迟控制	设定无 DTMF 信号有效检测功能	清除有效 DTMF 信号检测功能

(3) MT8888 与 80C32 的接口。MT8888 提供了与微处理器相连的接口，以对其发送、接收和工作模式进行控制。MT8888 可与 Intel 微处理器直接接口，即使使用 16 MHz 的单片机 80C51，也无需插入等待周期。与其它微处理器接口时，必须通过转换构造 MT8888 所需的时序。图 3.1.23 为 MT8888 的控制时序图。

图 3.1.23　MT8888 控制时序

(a) MT8888 读时序；(b) MT8888 写时序

图 3.1.24 是 MT8888 与单片机 80C32 的接口电路原理图，由于可以直接接口，因此无需构造控制信号。图中两片 MT8888(S1 和 S2)共用一个时钟振荡器。单片机的 P00～P03 口接 4 位数据总线，片选信号由单片机的地址锁存信号 ALE 提供，读写信号由微处理器的读写信号和译码信号经或门后产生。寄存器选择信号接到地址线 P20 口，这样，对每一片 MT8888 均有两个地址。两个中断信号经与门后送至单片机的 INT1 引脚。电路中扩展了一片 74365，软件可依此来判断是由哪一路 MT8888 产生的中断而扩展的。当 MT8888 向单片机 80C32 发出中断请求信号后，CPU 响应中断，执行中断服务程序。在中断服务程序中，首先读取 74365 的内容，以判断是哪一路 MT8888 所发出的中断请求后，再读取该路 MT8888 的状态寄存器，使中断自动清除以等待下一双音频信号。由于读完状态寄存器后，其内容即自动清除，重新读状态寄存器的内容是无效的，因此，应先将状态寄存器内容暂存于缓冲区内，再对标志位进行判断：该中断信号是发送中断还是接收中断，以执行下一步的操作。需要注意的是，单片机 80C32 的 INT1 中断方式应设置为电平中断，才能同时检测两片 MT8888 的中断请求，从而防止信号丢失。

图 3.1.24　MT8888 与 80C32 接口原理图

若将 MT8888 设置于呼叫处理工作模式，则通过对一定时间内中断次数的判断，可以识别不同的呼叫信号音，如振铃、回铃音、忙音、空号音以及拥塞音等。

软件程序包括 MT8888 初始化子程序、发送数据子程序和中断服务子程序。另外，在设计硬件电路时，由于 MT8888 发送 DTMF 信号同时又送到 MT8888 输入端，这样导致在发送数据时，要引起接收数据中断。为了正确判断，在程序中设置了一个发送数据标志 tflag，当 tflag=1 时，MT8888 处于发送数据状态。下面只给出第 1 片 MT8888(S1)的程序。

例 3.1.1　MT8888（S1）初始化子程序。

```
MOV DPTR，#0A001H
MOVX A@DPTR          ；读状态寄存器 SR
MOV A，#00H
MOVX A@DPTR，A       ；写控制寄存器
MOVX A@DPTR，A       ；写控制寄存器
MOV A，#08H
MOVX A@DPTR，A       ；写控制寄存器 A
MOV A，#00H
```

```
MOVX @DPTR，A        ；写控制寄存器 B
MOVX A，@DPTR        ；读状态寄存器 SR
MOV A，#0DH          ；设置 MT8888 工作方式
MOVX @DPTR，A        ；写控制寄存器 A
MOV A，#00H
MOVX @DPTR，A        ；写控制寄存器 B
RET
```

例 3.1.2　MT8888（S1）数据发送子程序。

入口参数：（R0）——待发送的 DTMF 数据。

```
SENDR02：
MOV A，R0
MOV DPTR，#0A000H
MOVX@DPTR，A              ；待发送数据送至 TDR
RET
```

例 3.1.3　80C32 INT1 中断服务子程序。

```
INT1： CLR EA             ；关中断
MOV DPTR，#8000H          ；读取 74365 内容
MOVX A，@DPTR
JB ACC.0，PATH1           ；转第 1 路 MT8888
JB ACC.2，PATH2           ；转第 2 路 MT8888
LJMP END
PATH1：MOV A，TFLAG       ；判断工作方式标志字
CJNE A，#01H，RECE1
MOV DPTR，#A001H
MOVX A，@DPTR             ；读状态寄存器 SR，清中断
ANL A，#02H
CJNE A，#02H，END
SETB TRANSEND             ；发送结束标志置位
LJMP END
RECEL:MOV DPTR，#A001H
MOVX A，@DPTR             ；读状态寄存器 SR，清中断
MOV DPTR，#A000H
MOVX A，@DPTR             ；读 DTMF 信号的数据编码
MOV R0，A                 ；结果存于 R0
SETB RECEIEND             ；收到结果标志置位
LJMP END
PATH2：(略)
END： SETB EA             ；开中断
RETI
```

3.2　三态逻辑编/解码技术

3.2.1　几种常用的三态逻辑编/解码芯片及其典型应用电路

1. MC145026

MC145026 是数字编码器，其内部电路由振荡器、计数器、编码器、检测器、模拟电子开关、数据选择及输出缓冲器等组成，其引脚排列如图 3.2.1 所示。A1/D1～A9/D9 为地址/数据并行输入端，可作地址端使用，也可作数据端使用。作地址端使用时，可由“ 1 ”、“ 0 ”进行二态编码或由“ 1 ”、“ 0 ”、“ 开路 ”(OPEN)进行三态编码。三态寻址码容量最大，最多可产生 3^9=19 683 种不重复的地址码，编码波形如图 3.2.2 所示。作数据端使用时，只能编成“ 1 ”和“ 0 ”两种状态，若将数据端编为“ 开路 ”，则译码器会自动译成“ 1 ”。两个脉冲代表一种编码：连续两个宽脉冲代表 1；连续两个窄脉冲代表“ 0 ”；先宽后窄两个连续脉冲代表“ 开路 ”。

图 3.2.1　MC145026 引脚排列

图 3.2.2　MC145026/27 地址码波形

MC145026 片内设有振荡电路，只要在 R_S、R_{TC}、C_{TC} 端外接 RC 元件即可形成振荡，为芯片提供时钟信号。从理论上讲，时钟频率可高达 10 MHz(V_{DD} = 15 V)以上，一般在 1 kHz～1 MHz 范围内选用。当时钟频率在 1～400 kHz 范围时，可按 $f \approx 1/(2.3R_{TC} \cdot C_{TC})$估算频率。通常选择 $R_S \geq 2R_{TC}$，$R_{TC} \geq 10$ kΩ，C_{TC} = 100 pF～15 μF。也可以使用外部时钟信号，此时时钟信号应从 R_S 端输入，而将 R_{TC}、C_{TC} 端置空不用。

DO 为数据信号输出端，根据地址/数据端输入状态所形成的编码数据流，由此端串行输出。$\overline{TE}$ 为发送使能控制端，低电平有效。当 $\overline{TE}$ = 0 时，允许 DO 端输出编码信号；当 $\overline{TE}$ =1 或悬空时，禁止 DO 输出，并使其呈高阻状态。$\overline{TE}$ 端内有上拉电阻，平时保持 $\overline{TE}$ =1。静态时，编码器被完全禁止，并控制时钟电路停止振荡，使芯片功耗降至最低，静态电流仅

为 0.2 μA，故无需另加电源开关。V_{DD}、V_{SS} 为正、负电源端，使用单电源供电时，应将 V_{SS} 接地。

2. MC145027

MC145027 是数字译码器，可与 MC145026 配对使用，它的技术参数和性能也与 MC145026 相同。

MC145027 由数据分离器、逻辑控制电路、序列发生器、模拟电子开关、移位寄存器、锁存器及输出驱动器等组成，其引脚排列如图 3.2.3 所示。A1～A5 为 5 个地址输入端，D6～D9 为四个数据输出端。地址可二态或三态编码，与编码器 MC145026 配用时，A1～A5 的编码输入状态应与 MC145026 的 A1/D1～A5/D5 输入状态完全相同，才能有效地进行译码。而 D6～D9 输出的数据，反映了编码器 MC145026 的 A6/D6～A9/D9 的输入状态，并且一一对应。如果发送不停顿，D6～D9 的输出状态将随 MC145026 的 A6/D6～A9/D9 输入状态不断改变而连续变化。

图 3.2.3　MC145027 引脚排列

DI 为编码信号串行输入端，VT 为译码有效输出端。当译码器收到地址编码相同、速率匹配的编码信号时，在 D6～D9 输出译码数据的同时，VT 输出便由低电平跳变为高电平，产生一个正脉冲，作为译码成功标志，指示发送有效。D6～D9 具有输出锁存功能，若不需要锁存，使 D6～D9 和 VT 相“与”即可。

R_1、C_1 端外接脉冲辨别定时元件，这两个元件(即 R_1、C_1)的取值大小决定着对宽、窄脉冲的识别。实际上是建立了一个门槛值，大于此值的译为宽脉冲，小于此值的译为窄脉冲。当编码器 MC145026 的 R_{TC}、C_{TC} 确定之后，应使 $R_1 \cdot C_1 = 3.95 R_{TC} \cdot C_{TC}$，其中 $R_1 \geq 10$ kΩ，$C_1 \geq 400$ pF。

R_2/C_2 端外接延时辨别定时元件，用以检测每个数据字的首尾和每次发送数据流的首尾，从而确定有效的各个单字。R_2、C_2 与编码器 MC145026 中 R_{TC}、C_{TC} 的关系是：$R_2 \cdot C_2 = 77\ R_{TC} \cdot C_{TC}$，其中 $R_2 \geq 100$ kΩ，$C_2 \geq 700$ pF。表 3.2.1 列出了一组定时元件选择参考值。由于 MC145000 系列编译码器对外围元件精度要求宽达±5%，因此定时电阻、电容只需选择相邻近的标称值即可，而不必苛求元件精度。MC145026/27 配对应用实例如图 3.2.4 所示。

表 3.2.1　MC145026/27 定时元件的选择

f/kHz	R_{TC}/kΩ	C_{TC}/pF	R_S/kΩ	R_1/kΩ	C_1/F	R_2/kΩ	C_2/F
362	10	100	20	10	470 p	100	910 p
181	10	220	20	10	910 p	100	1800 p
88.7	10	470	20	10	2000 p	100	3900 p
42.6	10	1000	20	10	3900 p	100	7500 p
21.5	10	2000	20	10	8200 p	100	0.015 μ
8.53	10	5100	20	10	0.02 μ	200	0.02 μ
1.71	50	5100	100	50	0.02 μ	200	0.1 μ

图 3.2.4 MC145026/27 配对应用实例

3. MC145030

MC145030 兼有编码和译码双重功能。工作电压为 2～6 V，工作温度为– 40～+85 ℃。V_{DD}=2 V 时最大静态电流为 20 μA，V_{DD} =2.5 V 时最大工作电流为 700 μA，时钟频率范围从直流到 500 kHz。其内部由带控制功能的振荡器、分频器、地址产生器、输入信号放大器、Manchester 编码和译码器、编码控制器、地址比较器及反转器等组成，其引脚排列如图 3.2.5 所示。

图 3.2.5 MC145030 引脚排列

说明：

MC145030 的编译码方式与 MC145026/27 不同，它采用 Manchester 编码和译码方式，基本规律是利用脉冲相位代表“1”和“0”：脉冲起始相位为 0° 时代表“0”；起始相位为 180° 时代表“1”。

A1～A9 为地址输入端，只允许进行二态编码，所以最多可提供 2^9=512 种不重复的地址码型。编码输出是根据地址位的输入状态，串行送出代表这些状态的数据流，首先是两位同步码，然后是 A1、A2、…、A9。译码时，将接收到的数据所代表的地址与本地（译码器）地址进行比较，以确定两者是否一致。每个地址输入端片内均有上拉电阻，可方便地通过编码开关或经跳线接 0。地址编码时，高电平可将地址端悬空或接 V_{DD}，低电平应接 V_{SS}。在静态待机时，A1～A9 内部上拉元件未被激活，地址输入端均呈高阻状态，以降低静态功耗。

EN 是编码使能控制端，脉冲上升沿触发有效，只要此脚有脉冲上升沿出现，任何译码时序均将被停止，并开始进入编码时序。DI 为接收数据输入端，内有输入信号放大器，可接收方波信号的最小幅值为 200 $mV_{p\text{-}p}$。输入信号通常经电容耦合，若信号电平在 V_{SS}～V_{DD} 之间亦可直接连接。DR 为译码复位端，加高电平时译码输出状态被清零，此端也可用来终止对输入信号的响应。ST 为编码状态指示输出端，编码时输出高电平，译码或空闲时输出低电平。当 ST= 0 时，编码输出端（EO）呈高阻状态。

DO 为译码输出端，片内是一个双稳态型反转输出器，当编码器连续两次发送相同的地址编码时，便完成一个工作时序。在收到有效的编码数据后，DO 输出便改变一次状态（1 或 0）。译码输出受 DR 控制，只有在 DR= 0 时才能有效地进行译码。

EO 为编码输出端，这是一个三态输出端，根据本地地址状态形成的编码数据流由此端串行输出，其中前两位为同步码，然后是 A1～A9。当 EN 有效时，连续两次发送地址编码才会完成一个编码工作时序。而后，EO 端便返回高阻状态。

OSC1～OSC3 为振荡端，外接 RC 元件便可形成振荡，振荡频率范围为直流至 500 kHz，表 3.2.2 列出了一组时钟频率和 RC 元件选择参数。在静态时，振荡器处于停振状态，芯片处于微功耗等待状态，只有在进行编码或译码时，振荡器才起振工作。若使用 500 kHz 以下的外部时钟信号，信号应由 OSC1 端送入，OSC2、OSC3 可置空不用。

表 3.2.2　MC145030 时钟频率及振荡元件的选择

f_{OSC} / kHz	*R*1 / kΩ	*R*2 / kΩ	*C* / pF	EN 开关延迟时间/ms
452	30	5.6	100	1.3
220	47	10	100	2.8
70	47	10	510	8.7
4.1	330	47	2200	148
*R1、C、R2 对应接至 OSC1～OSC3 端				

MC145030 可以很方便地实现双工数字信息传送，其典型应用电路见图 3.2.6。将两片 MC145030 的编码输出端 EO 和接收输入端 DI 互接，就构成了双工数字通信电路。S1、S2 为编/译码选择控制开关，平时 EN 被下拉为 0，芯片均处于等待译码状态。当按下 S1 或 S2

图 3.2.6　MC145030 组成的双工通信电路

时，EN 端被脉冲上升沿触发，开始将地址编码送入译码器。由于译码输出 EO 为三态输出端，因此在半双工通信中，可将同一片 MC145030 的 EO 和 DI 接在一起，用一根导线传送信号。当然，在使用其它载体传送信号时，也只需占用一个信道。

3.2.2 三态逻辑编/解码器在 PC 机与单片机通信中的应用

在遥测、遥控领域中，往往使用微机与单片机组成的多机系统完成测控任务。PC 机用作上位机，而单片机用作下位机。组建这样的系统时，PC 机与单片机的通信是整个系统成功的关键，常用的方法是使用微机的 RS-232C 串行接口与单片机进行串行通信。实际应用中，由于环境的影响以及 RS-232C 串行接口电气性能的限制，其通信的空间范围总是受到限制。针对上述问题，本部分将介绍如何在 PC 机和单片机上配置编/解码器，实现微机与单片机的通信，从而实现微机与单片机大范围、长距离的通信，并给出了在 PC 机和 8031 单片机间配置 MC145026/MC145027 编/解码器的实例，说明了其通信方法。

1. PC 机打印接口的利用

PC 机 CENTRONIC 打印机并行接口包括一个 8 位数据输出寄存器、一个 8 位数据输入寄存器、一个 5 位控制输出寄存器和一个 5 位状态输入寄存器。对该接口数据输出的写操作，可实现 8 位数据信息的输出；对接口控制寄存器的写操作，可改变输出控制线的状态；对接口状态输入寄存器的读操作，可得到外设的状态信息。CENTRONIC 并行接口与外设连接时使用 8 根数据线、4 根控制线和 5 根状态线。其数据线信号为 DATA0～DATA7，控制线信号为“选通”(—STROBE)、“初始化”(—INIT)、“打印机输入选择”(—SLCTIN)和“自动进纸”(—AUTOFDXT)。其中，在—SLCTIN 信号为低电平时，表示主机使用并行接口向打印机传送数据信息；当由—STROBE 产生负脉冲信号时，表示选通打印机，这时打印机接收数据线上的数据信号；—INIT 信号和—AUTOFDXT 信号在正常的数据传输过程中不使用，其状态保持不变。状态信号为：“忙”(—BUSY)、“出错”(—ERROR)、“缺纸”(—PE)、“选择”(—SLCT)和“应答”(—ACK)。由此可知，CENTRONIC 并行接口能完成一个 8 位的数据输出、一个 4 位的控制信号输出和 5 位状态信号的输入。这里，将该并行接口看作一般功能的 I/O 口，连接编码器与解码器。

2. 通信方法

使用编码器和解码器实现 PC 机与单片机的通信时，需要在 PC 机和单片机上分别配置编码器和解码器，以便完成数据信息的发送和接收。由于 MC145026/MC145027 一次能完成 4 位二进制数据信息的发送/接收，而计算机经常使用以字节为单位的数据，这样就需要将一个字节分为高半字节和低半字节，分两次发送/接收。图 3.2.7 为 PC 机与 8031 单片机使用编/解码器实现通信的原理图。

图 3.2.7 中，编码器 MC145026 的 U1 和 U4 的地址由微型开关 SW1 和 SW4 设置，解码器 MC145027 的 U2 和 U3 的地址由微型开关 SW2 和 SW3 设置。为了保证解码器能接收编码器发送的数据，其地址值应设置为相同数值，即 SW1 与 SW3 设置值相同，SW2 与 SW4 设置值相同。编码器和解码器外接的电阻、电容值应根据实际需要来选取。

PC 机与编码器的接口连线上，利用微机配置的打印机并行接口数据线的低 4 位 DATA0～DATA3 直接与编码器的 D6～D9 数据输入线相连，利用—INIT 控制线与编码器 $\overline{\text{TE}}$

数据发送允许脚相连；发送数据时，由 PC 机向并行接口数据输出寄存器低 4 位写入半个字节的数据信息，然后写控制输出寄存器，使得－INIT 信号为低电平，这样就完成 4 位数据的输出并由编码器发送。

图 3.2.7　PC 机与 8031 单片机使用编/解码器实现通信的原理图

在微机与解码器的接口连线上，使用并行接口的状态线－ACK、－ERROR、PE、SLCT 与解码器的数据线 D6～D9 相连，以便由 PC 机通过状态线读取解码器数据，并使用状态线 BUSY 与解码器的数据输出有效引脚 VT 相连，这样 PC 机可通过读取并行接口的状态寄存器检查解码器当前是否接收到有效的数据，以及得到由解码器接收的数据。

在编/解码器与 8031 单片机接口连线中，使用了单片机的 P1 口作为数据接口，使用了 P3 口中的两个引脚为控制状态线，其中 P1 口的低 4 位 P1.0～P1.3 与解码器 U2 的数据线 D6～D9 相连，以便读取由解码器接收的数据，并使用 P3.0 与解码器 U2 的 VT 引脚相连，由 P3.0 I/O 位的状态判定是否读取有效的数据。单片机发送数据是通过编码器 U4 实现的，使用 P1 口的高 4 位 P1.4～P1.7 与编码器 U4 的数据线 D6～D9 相连，使用 P3.1 的 I/O 引脚与 U4 的 $\overline{\text{TE}}$ 引脚相连。由单片机发数据信息时，可由 P1 口的高 4 位输出数据，再复位 P3 口 P3.1 I/O 位为低电平，命令由编码器 U4 发送输出数据信息。

使用编/解码器实现的 PC 机与单片机的通信，可应用于通信系统、报警系统、数据采集系统、LED 大屏幕显示系统、遥控系统等领域。这种方法既可使用有线方式，也可使用

无线方式。在与 PC 机的接口方面，巧妙地利用了微机打印机并行接口，因而既容易实现又简单方便。

3. 与 PC 机扩展槽接口

如果不利用 PC 机的打印接口，而利用 PC 机的扩展槽，则 MC145026 与 PC 机的接口如图 3.2.8 所示。MC145027 与 PC 机接口形式上也相同，只不过是 $\overline{\text{TE}}$ 引脚换成了 VT 引脚。

图 3.2.8　编码器与 PC 机扩展槽接口原理图

3.2.3　三态逻辑编/解码器在信号检测系统中的应用

在工业自动化控制系统中，往往需要对多点模拟量进行检测，传统的方法是在各检测点设置传感器，并以三线或二线连接到主机，通过多路模拟开关和模数转换器件对各个模拟量进行模数转换，取得相应的数据以供主机处理。这种方法存在如下缺陷：

① 模拟电压在通过电缆传到主机的过程中容易受到干扰；

② 主机要通过模拟开关选择传感器，这使探头的路数受到限制，主机的接口电路比较复杂；

③ 主机无法向各检测点传送控制指令；

④ 如果在检测点增加一个传感器，就必须增加一根电缆连至主机，因而增加了布线的复杂程度。

针对上述问题，可以设计一套基于 MC145027 的群模拟信号检测系统，使主机和各个探头之间只通过三根线即可进行双向的数据传输，如图 3.2.9 所示。由于 MC145027 特殊的译码方式能够消除瞬间的强电磁干扰，因而数据在传输过程中具有很高的可靠性。

图 3.2.9　群模拟信号检查系统图

1. 系统硬件电路的设计

该检测系统的硬件电路包括探头电路和主机接口电路两部分，主机和探头之间传输的格式遵循 MC145026 的编码格式。

(1) 探头电路。探头电路的原理框图如图 3.2.10 所示。主机发送至探头的编码信号线经过信号传至各个探头，经放大整形电路处理后送到 MC145027 进行解码，当地址判断一致后，VT 由低变高向单片机申请中断，由单片机读取解码后的数据。MC145027 能够解出 4 位数据码，4 位二进制的数据码可以表示 2^4=16 种命令，单片机根据命令的要求将采集到的数据(温度、压力、湿度等)按照 MC145026 的编码格式由 P1.7 输出，再经过驱动电路回送至主机。

图 3.2.10　探头电路

(2) 主机接口电路。以 486 或 586 微机作为主机，通过并行打印口与探头交换数据的主机接口电路如图 3.2.11 所示。MC145027 的 A1～A5 引脚的状态决定主机的地址码(00000)，D6～D9 分别和主机打印口的引脚 13、12、10、11 相连，上述 4 个引脚为打印机的状态输入口，口地址为 379H(279H)，分别对应于主机数据总线的 D4～D7。主机通过打印口的 14 脚(口地址 37AH/27AH，对应于数据位 D1)向各个探头发送命令，探头接到命令后向主机回传所要求的数据，并通过 MC145027 解码后由主机读取。MC145027 的 VT 引脚接至打印

口的 1 脚(口地址 37AH/27AH，对应于数据位 D0)，主机通过定时检测 VT 脚的状态来判断是否有应答数据到来。

图 3.2.11　主机接口电路

2. 软件的设计

(1) 数据格式的定义。根据 MC145026 的编码格式，在一个发送周期里可以发送 9 位数据信息，我们定义 A1～A5 为探头和主机的地址信息，由于总共可表示 243 个地址码，而主机的地址码定义为 00000，因而其余 242 个地址码可供探头使用；在主机发至探头的编码里，除了 A1～A5 表示探头的地址之外，尚有 A6～A9 共 4 位可以表示控制命令，共可组成 $\overline{\text{DSO}}$ 个命令码，设计时可以根据系统的要求将这 16 个命令码一一定义，以供系统使用。在探头发往主机的编码里，前五位 A1～A5 是固定格式 00000，表示主机的地址号，后四位 A6～A9 表示发往主机的数据，一个字节分两次发送，先发高半字节(高四位)，再发低半字节(低四位)。

(2) 探头地址码的设置。MC145027 的地址输入脚(A1～A5)有三个状态(高电平、平路、低电平)，也就是说地址线是三进制数据，而单片机的 I/O 口是二进制状态(二进制数据格式)。在探头电路中，为了使单片机发送的地址码与 MC145027 的地址码相对应，单片机必须能自动检测自身的地址。在图 3.2.10 所示的电路中，P1.0～P1.6 作为地址设定脚，它们所表示的地址信息应与 MC145027 的地址(A1～A5)相同，这涉及从二进制到三进制转换的问题。由于表示 243 个地址需要 8 位二进制数据，而单片机只有 7 位地址设定脚，另外的一位(最高位)只能由程序设定，这样，探头的地址就可以比较灵活地设置，因而具有一定的通用性。

(3) 单片机的软件设计。探头电路中的单片机主要用来完成以下几个功能：

① 完成探头电路的自检；

② 接收并执行主机发来的控制命令；

③ 根据主机的命令完成相应的动作；

④ 按照 MC145026 的编码格式向主机发送数据。

(4) 主机软件的设计。主机软件是整个控制系统的核心，在这里，我们只讨论和探头通信有关的内容。在主控软件中设置一个定时中断程序，以定时检测 MC145027 的 VT 脚的状态，当 VT 脚由低电平变为高电平时，通过读取 379H/279H 口的内容来接收探头发来的数据。当主机向各探头发送命令时，就可通过并行打印口的 14 脚发送控制命令编码。主机命令码的发送格式也应遵循 MC145026 的编码格式。

3.3　红外遥控技术

3.3.1　红外遥控原理

红外遥控本质上是利用红外线来传递数字编码信号。一般的红外遥控系统是由红外遥控信号发射器、红外遥控信号接收器和微控制器及其外围电路等三部分构成的，如图 3.3.1 所示。

图 3.3.1　红外遥控系统

遥控信号发射器用来产生遥控编码脉冲，驱动红外发射管输出红外遥控信号，遥控接收头完成对遥控信号的放大、检波、整形，解调出遥控编码脉冲。遥控编码脉冲是一组组串行二进制码，对于一般的红外遥控系统，此串行码输入到微控制器，由其内部 CPU 完成对遥控指令的解码，并执行相应的遥控功能。

在红外遥控系统中，解码的核心是 CPU。它接收解调出的串行二进制码，在内部根据本系统的遥控信号编码格式将串行码对应成遥控器上的按键。显然，这种在 CPU 内部解码出的遥控指令是不便于我们利用的，而且我们也不需要获取它。我们只需利用一般红外遥控系统中的遥控发射器、遥控接收头，自行设计解码电路，直接对遥控接收头解调出的遥控编码脉冲进行解码，就可以得到原始的按键信息。

3.3.2　红外编/解码方法

1. 红外遥控编码

实际应用中的各种红外遥控系统的原理都大同小异，区别只是在于各系统的信号编码格式不同。下面，我们举出红外遥控系统的实例来说明它的编码体制。

红外遥控发射器以 TC9012 为核心组成了键扫描、编码、发射电路。当按下遥控器上任一按键时，TC9012 即产生一串脉冲编码，如图 3.3.2 所示。

图 3.3.2　红外遥控编码脉冲波形

TC9012 形成的遥控编码脉冲对 40 kHz 载波进行脉冲幅度调制(PAM)后，便形成遥控信号，遥控信号经驱动电路由红外发射管发射出去。红外遥控接收头接收到调制后的遥控信号，经前置放大、限幅放大、带通滤波、峰值检波和波形整形，从而解调出与输入遥控信号反相的遥控脉冲。

在图 3.3.2 中，一次按键动作的遥控编码信息为 32 位串行二进制码。对于二进制信号“0”，一个脉冲占 1.2 ms；对于二进制信号“1”，一个脉冲占 2.4 ms，而每一脉冲内低电平均为 0.6 ms。从起始标志到 32 位编码脉冲发完，大约需 80 ms，此后遥控信号维持高电平。若按键未释放，则从起始标志起每隔 108 ms 发出 3 个脉冲的重复标志。

在 32 位的编码脉冲中，前 16 位码不随按键的不同而变化，我们称之为用户码。它是为了表示特定用户而设置的一个辨识标志，以区别不同机种和不同用户发射的遥控信号，防止误操作。后 16 位码随着按键的不同而改变，我们就是要读取这 16 位按键编码，经解码得到按键键号，转而执行相应控制动作。

那么，不同的按键编码脉冲是怎样和遥控器上不同的按键一一对应的呢？我们借助于逻辑分析仪记录下来遥控器上每一个按键的编码脉冲序列，破译出了各按键的编码。表 3.3.1 是解码后得到的红外遥控器上各键的编码(前 16 位用户码均为 0000001011111101，表 3.3.1 只列出后 16 位键码)。

由表 3.3.1 按键编码可看出，后 16 位键码的前 8 位与后 8 位互为补码，这样加大编码的冗余度的目的是为了增强遥控系统的抗干扰能力。实际上，我们只需截取 16 位键码的 8 位(比如后 8 位)，就可达到识别按键的目的。当然，要加强遥控系统的抗干扰能力，还需接收全 16 位键码甚至 16 位用户码加以识别。

表 3.3.1　红外遥控器按键编码的后 16 位键码

键 功 能	键　　码			
电源	0100	1000	1011	0111
静音	0000	1000	1111	0111
1	1000	0000	0111	1111
2	0100	0000	1011	1111
3	1100	0000	0011	1111
4	0010	0000	1101	1111
5	1010	0000	0101	1111
6	0110	0000	1001	1111
7	1110	0000	0001	1111

续表

键 功 能	键　　码			
8	0001	0000	1110	1111
频道＋	1101	1000	0010	0111
频道－	1111	1000	0000	0111
色度＋	0001	1000	1110	0111
色度－	0011	1000	1100	0111
对比＋	0110	1000	1001	0111
对比－	1110	1000	0001	0111
亮度＋	1001	1000	0110	0111
亮度－	1011	1000	0100	0111
音量＋	0101	1000	1010	0111
音量－	0111	1000	1000	0111
Sleep	0111	0000	1000	1111
→	1000	1000	0111	0111

2. **红外遥控解码**

红外遥控接收头解调出的编码是串行二进制码，包含着遥控器按键信息，但它还不便于 CPU **读取识别，因此需要先对这些串行二进制码进行解码。红外遥控信号解码电路如图** 3.3.3 **所示，它主要包括遥控编码脉冲串并转换电路与** PLD **解码电路。**

(1) **遥控编码脉冲的串并转换。红外遥控接收头解调出的遥控编码脉冲经一非门反相后引入计数器** 4020 **的复位端**(RST)，4020 **的脚** 10(CP)**端引入** 1 MHz **计数脉冲。遥控信号(已反相)中每一正脉冲到来时，其高电平对** 4020 **复位，经过** 0.6 ms **遥控信号变为低电平，**4020 **复位结束，开始以** 1 MHz **的频率计数，直到下一个正脉冲到来时为止。二进制码“** 0 **”每一脉冲周期低电平时间为** 0.6 ms**，二进制码“** 1 **”每一脉冲周期低电平时间为** 1.8 ms，4020 **的** Q11 **端即可以区分二进制码“** 0 **”或“** 1 **”。每一遥控编码正脉冲上升沿到来时，若** Q11 **端为“** 1 **”，说明前一位遥控码为“** 1 **”；若** Q11 **端为“** 0 **”，说明前一位遥控码为“** 0 **”。**

将 4020 **的** Q11 **端作为** 74HCS9S **的串行移位输入端**(SER)**，便可在每一个遥控编码脉冲上升沿到来时，并在** 4020 **复位之前，将** 74HC595 **中的数据沿** Q0～Q7 **方向依次移一位，且** 4020 **的** Q11 **端数据移入** 74HC595 **的** Q0 **端。对于一组遥控编码脉冲，共有** 33 **次上升沿(包括起标志)，而** 74HC595 **仅为** 8 **位移位寄存器，所以移位的最终结果，只有遥控编码脉冲的最后** 8 **位保留在** 74HC595 **中。**

当一组遥控编码脉冲(反相后)来到时，其起始标志的上跳沿触发了双单稳 74HC123 **的** 1B**，在** 1Q **上产生了一个宽度为** 120 ms **的正脉冲。**1Q **同时又触发了** 74HC123 **的** 2B**，在** $2\overline{Q}$ **产生一个宽度为** 80 ms **的负脉冲，**1Q **和** $2\overline{Q}$ **相与后作为锁存信号送至** 74HC595 **的** RCLK **端，即一组遥控编码脉冲到来** 80 ms **后，产生一个锁存信号。此时** 74HC595 **已经移过了一组遥控码，芯片中保留的是最后** 8 **位遥控码，锁存信号将这最后** 8 **位遥控码锁存。**

(2) **基于** EPROM **的遥控解码原理。经过串并转换，我们得到了** 8 **位并行遥控码。为了让** CPU **读取这个并行遥控码，通常的方法是在转换完成后产生一个中断，通知** CPU **来读取遥控信息。但这样做要占用** CPU **一个外部中断资源，并需编写额外的中断服务程序，显得比较烦琐。尤其是当仪器系统的软件不是由自己开发而又要加装遥控时更是无能为力。因此，我们想寻求一种不占用仪器** CPU **的软、硬件资源而实现遥控的方法，使键盘输入和遥**

控输入统一起来，占用同一个端口、同一个中断、同一个中断服务程序。简言之，要做到对 CPU 是透明的，似乎只有一个键盘输入单元在工作，只需访问它来进行键盘扫描、键码读出操作。但实际上却有遥控器与键盘两套键输入硬件在同时而独立地工作。

考察一下智能仪器的键盘扫描输入原理。在这种方式下，CPU 通过输出指令使键盘矩阵的行扫描线依次为“0”(低电平)，同时监测键盘矩阵的列扫描线。若无键按下，则列扫描线输出全“1”(高电平)；若有键按下，则此键所在列线输出为“0”，再结合行扫描线此时的状态，就可具体定位按键。

我们设想，可否将遥控接收头输出的含有按键信息的 8 位遥控码通过某种转换，并入键盘矩阵电路，当遥控器有键按下时，就会在机上键盘对应键处产生一个“模拟”按键动作，产生一个键码可供 CPU 读取。所谓“模拟”，是指没有机械按键动作，但对于键盘矩阵电路而言却产生一个低电平，其效果和机械按键动作完全一样。这样，就将遥控键盘和本机键盘统一起来，二者的键数和键功能定义都一样，一个相同的键在遥控器上按下和在本机键盘上按下对 CPU 而言没有任何区别，只不过对键盘矩阵来说，前者是软接触，后者是硬接触。

根据遥控器上按键与本机键盘按键的一一对应方案，我们可以导出实现“模拟”按键的逻辑真值表(其中 C0～C4 为列扫描线)，见表 3.3.2。

表 3.3.2 “模拟”按键逻辑真值表

输入变量		输出变量
键盘扫描码	8 位遥控码	C0C1C2C3C4
1110	遥控接收头解码、串并转换后得到的 20 个键的键码	20 种键码在此键盘行扫描码下 5 根列扫描线的输出
……		
1110		
1101	解调、串并转换后得到的 20 个键码	20 种键码下 5 根列扫描线的输出
……		
1101		
1011	解调、串并转换后得到的 20 个键码	20 种键码下 5 根列扫描线的输出
……		
1011		
0111	解调、串并转换后得到的 20 个键码	20 种键码下 5 根列扫描线的输出
……		
0111		
1100	解调、串并转换后得到的 20 个键码	20 种键码下 5 根列扫描线的输出
……		
1100		
……	……	……
……		
……		
0000	解调、串并转换后得到的 20 个键码	20 种键码下 5 根列扫描线的输出
……		
0000		

这是一个有 12 变量输入、S 变量输出的组合逻辑函数，最小项总数为 16×20＝320 个。若用普通逻辑门电路来实现这样的功能将是十分麻烦的，用 PLD(可编程逻辑器件)来做就要简单得多。EPROM 就是一种与阵列固定、或阵列可编程的逻辑器件。如果把 EPROM 的输入地址 A0，A1，…，AN 视为输入逻辑变量，同时把输出数据 D0，D1，…，DM 视为一

组多输出逻辑变量，那么输出与输入之间也就是一组多输出的组合逻辑函数。而且，EPROM 地址译码器的输出包含了全部输入变量的最小项，每一位数据输出又都是这些最小项之和，因而任何形式的组合逻辑函数均能通过向 EPROM 中写入相应的数据来实现。不难推想，具有N位输入地址和 M 位数据输出的 EPROM 可以获得一组(最多为 M 个)任何形式的 N 变量组合逻辑函数。

根据这个原理，选用 4 k×8 EPROM2732，可以实现任意 12 变量输入、8 变量输出的组合逻辑函数。在本机遥控系统中，利用了 EPROM 的 D0～D4 五根数据线和全部 12 根地址线，通过向 2732 中固化表 3.3.1 所示的逻辑真值表，从而实现了关键的遥控解码，使遥控器上按键与本机键盘按键一一对应起来。需要指出的是，EPROM 的地址译码是全译码，而在本方案中，占据地址线 A0～A7 的 8 位遥控码只有 20 种有效码值(20 个键)，即一页(256 字节)中只有 20 个有效数据，将剩余空间填入 0FFH。

由解码电路图 3.3.3 可见，EPROM2732 的地址线 A0～A7 接至 8 位输出锁存移位寄存器 74HCS9S 的输出(即 8 位遥控码)，A8～A11 接至键盘矩阵的行扫描线 R0～R3；2732 的 8 根数据线使用了其中的 5 根 D0～D4，接至键盘矩阵的列扫描线 C0～C4，2732 的 $\overline{CS}$ (片选

图 3.3.3　红外遥控解码电路原理图

端)接地，$\overline{OE}$(读信号)接至施密特与非门 4093 的 3 脚输出，此输出为双单稳 74HC123 的 1Q、$2\overline{Q}$ 与非的结果。

当遥控器上没有按键按下时，EPROM 2732 的 $\overline{OE}$ 端为“ 1 ”，使得 2732 的数据线 D0～D4 为高阻态，与键盘矩阵线脱离，而本机键盘的扫描与读出照常进行，不受影响。若遥控器上有键按下时，经红外发射、接收对应的 8 位遥控码出现在 74HC595 的输出端，并作为 EPROM 2732 的 A0～A7 输入，此时的行扫描码(CPU 发出)作为 A8～A11 输入，2732 的 $\overline{OE}$ 端为低电平，读出 A0～A11 指定单元的数据，将其中 D0～D4 放在键盘矩阵列线上。D0～D4 中只有一位为“ 0 ”，指示着在哪一列有键按下，这样就由遥控接收、解码电路模拟了一次“ 按键 ”动作。接下来，CPU 对这个“ 按键 ”动作的响应、处理就与本机键盘操作的处理完全一样了。

3.3.3　几种新型红外编/解码芯片及其应用

1. 8 路 BA5101/BA5201 红外遥控编 / 解码集成电路

BA5101/BA5201 引脚排列如图 3.3.4 所示，应用线路如图 3.3.5 所示。

图 3.3.4　BA5101/BA5201 引脚排列

图 3.3.5　BA5101/BA5201 应用线路图

其中，对应于 BA5101 每触发一次按钮，BA5201 相对应的通道便输出一个 65ms 的低脉宽信号。

说明：

RX 为红外线接收头。

2. 8 路 BA5048/BA5050 红外遥控编 / 解码集成电路

(1) 红外线遥控发射接收芯片 BA5048、BA5050 的特性。BA5048、BA5050 是配对使用的红外线遥控发射接收芯片。BA5048 是发射器，它采用 CMOS 结构，功耗极低，工作电压范围宽(1.5～5.0 V)；该芯片内置振荡电路，外围电路也极为简单；具有 18 种功能及 75 种指令；可以单键触发或多键触发(最高达六键)。

BA5048 的引脚排列如图 3.3.6 所示，各引脚的功能如表 3.3.3 所列。

图 3.3.6　BA5048 引脚排列

表 3.3.3　BA5048 引脚功能说明

引脚号	名　称	I/O	功 能 说 明
1，16	GND，VDD		电源 1.5～5.0 V
2，3	XT，XTB	I，O	振荡器接线端，可连接一个 455 kHz 陶瓷振荡器
4～9	K1～K6	I	按键矩阵的按键输入端，在 T1～T3×K1～K6 矩阵中 18 种按键被联系起来
10～12	T1～T3	O	按键矩阵的扫描输出端
13	CODE	I	发射器与接收器代码的输入端
14	TESTB	I	悬空
15	TXOUT	O	发射信号输入端，输出的是 12 位 1 周期的 38 kHz 的载波信号

BA5048 的按键输入是由 K1～K6 及 T1～T3 组成的 6×3 矩阵来构成的。K1～K6 与 T1 组合成的按键输出的是连续性信号，通过给 T1 到 K1～K6 加编码二极管可实现 63 种按键编码。连续性信号的输出特点是：当有一连续性按键按下时，指令发射 2 次信号，每次信号的周期是 20.25 ms，两次信号之间相隔 33.8 ms，然后有 87.8 ms 的暂停，之后另一周期开始直到按键抬起。K1～K6 与 T2～T3 组合成的按键输出的是一次性触发信号，按一次键只对应一次输出。一次性触发信号的输出特点是：当有一次性触发按键按下时，指令仅发射 2 次信号，每次信号的周期是 20.25 ms，两次信号之间相隔 33.8 ms，然后便停止。

信号每 1 个周期的发射指令是一串 12 位的字码，如图 3.3.7 所示。图中，C1～C3 是代码位，对应不同型号的解码器对应不同的代码；H、S1、S2 对应 T1、T2、T3，表示连续信号或是一次性触发信号；D1～D6 对应 K1～K6 按键，表示 6 位信息码。

C1	C2	C3	H	S1	S2	D1	D2	D3	D4	D5	D6

图 3.3.7　BA5048 发射指令序列的 12 位含义

图 3.3.8 是 BA5048 的应用电路原理图，1～6 键可加编码二极管，从而扩展出最多 63 个按键。

BA5050 的引脚排列如图 3.3.9 所示，各引脚的功能如表 3.3.4 所列。

图 3.3.8　BA5048 应用电路原理图　　图 3.3.9　BA5050 引脚排列

表 3.3.4　BA5050 引脚功能说明

引脚号	名称	I/O	功 能 说 明
1	VSS		电源地
2	RXIN	I	消除了载波信号的指令信号输入端
3～8	HP1～HP6	O	只要有连续性信号输入，该输出端就保持在高电平
10，9	CP1，CP2	O	当接收信号输入时，输出电平反向
20～11	SP1～SP10	O	当接收信号输入时，输出高电平脉冲
22，21	CODE1，CODE2	I	发射器代码与设于此端的代码进行比较，若一致，则输入被接收
23	OSC	I/O	一电阻和一电容并联接入此端和 VSS 之间，构成振荡电路
24	VDD		工作电源

BA5050 在接收信号时要连续接收 2 组发射信号，以确定信号是否正常。首先第一组信号的数码被存入片内的 12 位转换寄存器中，然后，当接收到第二组信号码时，立即与转换寄存器中的数码进行比较，看它们是否相同，如果不同系统马上被重置，反之当所有接收码正常时，各相应的输出端就从低电平升到高电平。

为防止其它机型的干扰，BA5050 提供 CODE1 和 CODE2 两个码位来核对发射器与接收器代码是否一致，只有两种代码一致时后面的数据位才有效产生相应的输出，否则不产生输出。

单脉冲输出引脚 SP1～SP10 是在连续接收到两组信号后立即输出一个宽度为 107 ms 的正脉冲。持续脉冲输出引脚 HP1～HP6 只要接收到有效的连续信号，相应的输出端就保持高电平，当松开按键连续信号终止后，再过大约 160 ms，输出才变为低电平。HP1～HP6 能同时并行地输出高电平。每当反向电平输出引脚 CP1、CP2 收到一次有效的信号时，输出端电平就反向一次。

BA5050 对电视机、日光灯、节能灯等干扰具有极强的抑制能力。

(2) BA5050 与单片机系统的接口电路。图 3.3.10 是 BA5050 与 MCS51 单片机系统的接口电路，该电路用两个端口来处理接收到的键码数据。

图 3.3.10　BA5050 **与** MCS51 **单片机系统接口电路**

HPR0 端口用来读取连续按键信号(HP1～HP6)和反转按键信号(CP1、CP2)。HP1～HP6 可有 63 种排列组合，其键码值的范围是 01H～3FH。在实际应用系统中，一般将 HP1～HP6 的排列组合按键设置成各种功能按键，CP1 与 CP2 设置成类似电源开关和静音等功能的按键。

SPR0 端口用来读取单次触发按键信号(SP1～SP10)。SP1～SP10 信号是经过十进制编码电路加到端口的，因此，没有 SP 键按下时，此端口读到的数据是 10H；有 SP 键按下时，SP1～SP10 对应键码是 00H～09H，在后面的子程序中已将它转化为 01H～0AH，在实际应用系统中一般将它与 1～9 和 0 十个数字键对应。

例 3.3.1 是读取遥控器键码的子程序，其中，连续性按键键码值存在 50H，反向电平输出 CP1、CP2 状态存在 51H，一次性触发按键键码存在 52H。读者可根据 50H、51H、52H 的值和具体的应用系统，自己定义按键功能和编写键码的处理程序，程序的基本结构是判断键值后作相应的跳转处理。

📖 **例 3.3.1　红外线遥控器接口子程序。**

```
READKEYSUB：          ；30 ms 中断服务程序中的一部分
MOV DPTR，#4000H      ；连续性信号输入端口
MOVX A，@DPTR         ；从端口读入数据
MOV B，A              ；暂存入 B
ANL A，#3FH           ；将 CP1、CP2 屏蔽掉
MOV 50H，A            ；将连续性按键键码写入 50H
MOV A，B              ；恢复端口数据
ANL A，#0C0H          ；将连续性按键键码屏蔽掉，读出 CP1、CP2
MOV 51H，A            ；存入 51H
MOV DPTR，#8000H      ；一次性触发按键输入端口地址
MOVX A，@DPTR         ；从端口读入数据
CJNE A，#10H，RDK10   ；如果端口数据不是 10H，说明有键按下，转 RDK10
MOV 52H，#00H         ；如果端口数据是 10H，说明无键按下，将键码置为 00H
JMP RDKEND
RDK10：INC A
MOV 52H，A            ；将读到的数加 1 作为键码存入 52H
RDKEND：RET
```

3. 6 路 LC2190/LC2200 红外遥控编 / 解码集成电路

6 路 LC2190/LC2200 红外遥控编 / 解码集成电路的最大特点是可选择自锁或互锁方式接收 IC 的六路输出。

所谓互锁，是指 IC 的六路输出只能有一路输出高电平，其余输出低电平。所谓自锁，是指六路中的每一路均可以自由变化，且在发射端按一下某一路按钮，在接收 IC 中对应的这一路输出电平就变化一次，原来为高电平现在变为低电平，原来为低电平现在就变为高电平。

这两种工作方式给实际应用带来了很大方便：互锁功能可用于电视、收音机遥控选台及其它“多中取一”的场合；自锁功能可用于控制多路开关而免去再加双稳态电路的麻烦。

发射电路的型号为 LC2190，其典型应用电路如图 3.3.11 所示，接在 2 脚的 RC 决定内部时钟振荡器的振荡频率，f_{osc}=38 kHz × 2 =76 kHz，微调 R 使频率稳定在 76 kHz。接在 3 脚的电容为脉冲间隔时间定时电容，因 LC2190 每次触发时连续发三遍，其间隔时间便由此电容来决定。5 脚输出有效指示，可外接 LED(对 GND)，当任一键按下时，输出一高电平，驱动 LED 发光。在按键 I1～I6 中，I1、I2 为连续发射，I3～I6 为单次发射。如果将 I1、I2 分别接一只开关二极管至 I6，则 I1、I2 亦成为单次发射。

图 3.3.11　LC2190 应用线路图

接收电路的型号为 LC2200，其典型应用电路如图 3.3.12 所示。发射端按钮开关 I1～I6 与接收端 OP1～OP6 对应。12 脚的 SW 为输出方式选择端。该脚接高电平时，OP1～OP6 输出为互锁状态；该脚接低电平时，OP1～OP6 输出为自锁状态。OP1～OP6 输出的驱动电流可达 2 mA，因此可直接驱动可控硅或通过三极管驱动继电器再次控制功率较大的电器。

4. PT2262/PT2272 红外遥控编 / 解码集成电路

(1) 红外遥控编 / 解码 IC 芯片 PT2262/PT2272 简介。PT226－IR 芯片是将载波振荡、编码、发射部分集于一身的集成电路，外围电路简单，使用方便。

PT2262－IR 芯片引脚排列如图 3.3.13 所示。各引脚功能如下：A0～A5 为地址输入，

图 3.3.12　LC2200 应用线路图

图 3.3.13　PT2262－IR 芯片引脚排列

可编制成三种状态：“ 1 ”、“ 0 ”和“ 开路 ”。A6/D0～A11/D5 为地址或数据输入，取决于接收端的译码器。做地址输入时，可编制成三种状态：“ 1 ”、“ 0 ”和“ 开路 ”；做数据输入时，可编成“ 1 ”和“ 0 ”两种状态。最大编码容量为 3^{12}=53141 种。$\overline{TE}$ 为发射使能端，低电平有效。OSC1、OSC2 外接振荡电阻，决定电路时钟频率。DOUT 为数据输出端，由各地址、数据的不同状态而决定的各位编码由此脚串行输出，编码波形如图 3.3.14 所示。DOUT 端输出的数据信号是调制在 38 kHz 的载波上的。要使载波频率为 38 kHz，应选择合适的电阻，使振荡频率为载波频率的 2 倍，即 76 kHz，振荡电阻一般选定在 430～470 O。V_{DD}、V_{SS} 为电源正负输入。PT2262－IR 芯片的工作电压范围为 3～15 V。PT2272 芯片的工作电压范围为－0.3～16 V，V_{DD}=12 V 时，功耗为 300 mW。PT2272 是和 PT2262－IR 配对使用的解码集成电路，引脚排列见图 3.3.15。A0～A5 是地址输入，要求与发射端 PT2262－IR 设定的状态一致。D0～D5 是 6 位数据输出。脉冲编码信号自 DI 输入，外接电阻一般选为 1 MO。当接收到有效信号时，VT 端由低电平变为高电平，起指示作用，同时亦可输出驱动负载。

图 3.3.14　编码波形

图 3.3.15　PT227 芯片引脚排列

(2) 实用电路。实用发射电路见图 3.3.16。发射电路由单稳态电路、编码电路、红外发

图 3.3.16　发射电路

射电路三部分组成。以 NE555 芯片为核心组成的单稳态电路，主要作用是当按下 AN 键后，在单稳态时间内，再次按键无效，以达到防抖动的目的，确保接收电路输出有效，并使接收电路输出与单稳态等时间长度的音频信号，给使用者一个报警器已处于警戒状态的提示。其工作原理是：当按下 AN 键后，外加正脉冲经由电容 C_1、电阻 R_4 组成的微分电路加到 NE555 芯片第 6 脚，使 7 脚输出一个单稳态负脉冲，其脉宽由电阻 R_3、电容 C_3 决定，单稳态时间为 $T=R_3 \cdot C_3$，取电阻 R_3=1 MΩ，电容 C_3=4.7 μF，则单稳态时间 T=4.7 s。电阻 R_4、R_5 与电容 C_1 构成放电回路。编码可由设定三态编码开关状态得到。红外发射电路中红外发射管 IR 可选用 LTE5208A。

接收电路如图 3.3.17 所示。接收电路由红外接收放大电路、解码电路、控制电路和音频输出电路四部分组成。IC1 选用 BA5302，它是红外放大解调一体化组件成品，当 IC1 的感光窗接收到由发射器发来的红外线调制信号时，经内部电路处理后从 IC1 的 OUT 端输出，经三极管 V1 放大倒相后从 IC2(PT2272)的 DI 端输入，解码正确时，IC2 的 VT 端输出高电平，送往控制电路。控制电路由 IC3 双 D 触发器 CD4013、IC4 四与非门芯片 CD4011 和水银开关 SQ 组成。三极管 V_2 放大音频讯号驱动讯响器 BUZ 发声。

图 3.3.17　接收电路

当开关接通电源时，接收放大电路和解码电路就处于待机状态，IC2 的 VT 端输出为低电平，同时电阻 R_1 与电容 C_2 组成的微分电路将触发器 IC3－1 的 S_1 端置位。由真值表(表 3.3.5)可知：R_1=0，S_1=1，$\overline{Q_1}$=0，Q_1=1，给 IC3－2 的 S_2 端置位成 S_2=1，因 R_2=0，故 Q_2=1，IC4-2 的 2 脚也为高电平。此时，IC4-1 的输入端因与 VT 端相连，输入为低电平，3 脚输出为高电平，IC4－2 的 3 脚输出为低电平，三极管 V2 截止，无音频信号输出。

当发射端按下 AN 键后，发射管按编码要求发射出一

表 3.3.5　真 值 表

输 入			输 出		
CP	R	S	D	Q	$\overline{Q}$
↑	0	0	0	Q0	$\overline{Q0}$
↑	0	0	1	1	0
↓	0	0	×	Q0	$\overline{Q0}$
×	1	0	×	0	1
×	0	1	×	1	0
×	1	1	×	1	1

组脉冲信号，接收头接收到的这一信号经内部处理输出，再经放大、解码，若解码正确，将使 PT2272 的输出端 VT 由低电平变为高电平，VT 端与 CP_1 相连，使 IC3-1 触发翻转(D1 端与 $\overline{Q_1}$ 端相连而使其状态方程为 $Q_{n+1}=\overline{Q_n}$)，$R_1=0$，$S_1=0$，$\overline{Q_1}=1$，$Q_1=0$。IC3-2 的 $S_2=0$，$R_2=0$，$Q_2=1$。接 $\overline{Q_1}$ 端的发光二极管被点亮。IC4-1 的输入端为高电平，3 脚输出为低电平，使 IC4-2 的 3 脚输出为高电平，三极管 V2 导通，音频信号输出，延迟时间与发射端单稳态时间相等。

接收器可安放于电视机、录像机等家用电器，当物品被搬动产生倾斜时，水银开关被接通，这一变化使 R_2 端被置为高电平，此时 $S_2=0$，$R_2=1$，$Q_2=0$，使得 IC4-2 的 3 脚输出高电平，讯响器鸣叫报警。这时，即使水银开关断开，由于 $S_2=0$，$R_2=0$，CP 不变，触发器 IC3-2 仍保持原状态不变，故一旦水银开关接通，讯响器便会长时间报警，不会因水银开关的再次通断而终止。

解除报警状态需由发射器再一次送出发射信号，接收器收到这一有效信号后，VT 端又产生一正脉冲，使 IC3-1 触发翻转，$\overline{Q_2}=0$，发光二极管熄灭，$Q_2=1$，此时 $S_2=1$，$R_2=0$。由真值表可知 $Q_2=1$，由于 IC4-1 的 3 脚输出为低电平，使 IC4-2 的 3 脚输出仍为高电平，保持单稳态工作时间长度后，VT 端跳变为低电平，Q_2 不变为高电平，IC4-1 的 3 脚变为高电平，所以 IC4-2 的 3 脚跳变为低电平，三极管 V2 截止，报警状态被解除。

实际应用中可并联安装多个水银开关，多方向调整角度，以提高报警器的灵敏度。

3.4 差错控制技术

在信息传输过程中，出现错误是不可避免的，所以，采取措施使误码率降低到最小程度(或满足系统要求)，是数据通信系统的重要内容之一。引起差错的原因是多方面的，主要原因是传输信道特性不理想以及外界干扰。因此，要减小误码率，提高传输质量，要在两个方面下功夫：第一，改善传输信道的电特性，使误码率达到要求。由于受经济因素和技术因素等条件的限制，这方面的努力往往不能得到理想的效果。第二，采取检错、纠错技术，即所谓差错控制技术。在差错控制技术中，对信息数据进行可靠有效的编码是很重要的途径之一。我们知道，外界干扰主要有随机噪声和突发脉冲噪声。随机噪声是有规律的，只要在电路中采取一定措施就可以减弱其干扰程度，但是，突发脉冲噪声是无规律可循的，抗干扰编码成为解决这个问题的重要手段。

3.4.1 抗干扰编码的基本概念

抗干扰编码是通过插入一些监督码元来实现纠(检)错功能的。为了更好地理解抗干扰编码的基本概念，有必要先了解抗干扰编码的机理。下面，就从二进制码的结构出发来讨论抗干扰编码的纠、检错的机理。

对二进制码而言，若一个独立的信息由 n 位二进制码元来表示，那么 n 位二进制码的所有组合可以表示 $N=2^n$ 种信息。如果这 N 种组合均用来传输信息，则这样构成的码是不具有抗干扰能力的。例如，电传电报所用的 5 单位码就是一个例子。因为其 $2^5=32$ 个码组全被信息序列占用，故在传输过程中无论码组中哪一位码出错，它仍属于传输码所定义的码

组内，这样，接收端就无法辨别接收码组是否出错。如果这 N 个码组不是全用的，情况就不同了。为说明这一点，下面用 3 位二进制码构成的码组来说明编码的抗干扰性能。

3 位二进制码构成的码组集合有 $2^3=8$ 种不同码组，这里分三种情况来讨论：

(1) 若每个码组都用作传输有用码组，则这样的编码不具有抗干扰能力；

(2) 若只采用(000)、(011)、(101)和(110)四个码组作为有用码组，则在传输过程中的任一码组中的任一位发生差错时，该码组都将变成废弃(无定义)码组，因而可以发现该出错码组，但是不能判断码组出错的具体位置，所以不能纠正，即该编码可以检出一个差错；

(3) 若只采用(000)和(111)作为有用码组，则在传输过程中任一码组出现两个差错时，接收端都能发现。若该有用码组用来纠错，则可以纠正发生在码组中任何位置的一个差错。其方法可以是：将 8 个码组分成两个子集，其中{(000)、(100)、(010)、(001)}与有用码组(000)对应，{(111)、(011)、(101)、(110)}与有用码组(111)对应，这样，在接收端只要收到第一子集中码组即判为(111)，从而纠正了传输码组中可能出现的任意一位差错。由此可以看出，该码能检出两个以下的差错，或纠正一个差错。

3.4.2　差错控制的基本工作方式

在数据通信中，目前主要有以下几种差错控制方式。

1. 反馈重发纠错(ARQ)方式

反馈重发纠错(ARQ)方式的工作原理是：发送端对发送序列进行差错编码，检测出错误的校验序列，接收端根据校验序列的编码规则判决是否传错，并把判决结果通过反馈信道传回给发送端。若无错，接收端确认接收，同时发送端缓冲器清除该序列；若有错，接收端拒收，同时发送端重新发送此序列，直到接收端接收正确为止。

在微机通信中，反馈重发纠错系统一般采用两种方式工作。一种是采用半双工通信方式的系统，该系统中发送端只有在接收到反馈应答的判决信号后，才能决定是否继续发下一组数据，故这种系统也称为发送等待系统，在面向字符的传输控制规程中应用较多。另一种是采用全双工通信方式的系统，该系统把需要应答的判决信号插到双方发送的信息帧中，这种系统称为连续发送系统，在面向位的传输控制规程中应用较多。

反馈重发纠错方式的缺点是实时性较差。

2. 前向纠错(FEC)方式

在前向纠错(FEC)方式中，发送端对数据进行检错和纠错编码，接收端收到这些编码后进行译码。译码不但能发现错误，而且能自动地纠正错误，因而不需要反馈信道。这种方式的缺点是译码设备复杂，而且纠错码的冗余码元多，效率低。

3. 混合纠错(HEC)方式

混合纠错(HEC)方式是上述前向纠错和反馈重发纠错两种纠错方式的结合。这种纠错方式中，发送端编码具有一定的纠错能力，接收端对收到的数据进行检测。若发现有错且未超过纠错能力，则能自动纠错；若超过纠错能力则发出反馈信息，命令发送端重发。这种方式在一定程度上弥补了反馈重发纠错和前向纠错两种方式的缺点。

4. 信息反馈(IRQ)方式

信息反馈(IRQ)，方式也称为回程校验方式，其特点是在发送端检测错误。其工作过程

为：发送端不对信息进行差错控制编码，而直接将用户信息传送给接收端；接收端收到信息后，将它们存储下来，再原封不动地通过反馈信道送回发送端；发送端将其与原发送信息进行比较，若无传输差错(或差错在允许范围内)，则发送新的信息；接收端收到信息后将上一次存储的信息传给用户。反之，若发现传输差错(或差错超出某个限度)，则发送端将信息重新传输，直到反馈信息被发送端认可为止。

3.4.3　几种常用检错码

1．奇偶校验码

奇偶校验码又称奇偶监督码，它只有一个监督元，是一种最简单、也是在数据通信中应用最多的一种检错码。

编码方法是在每个信息码组之后附加一位校验元(或称监督元)，使整个码组中“1”的个数成为奇数或偶数，分别称为奇校验或偶校验。接收端用一个模 2 加法器就可以很方便地完成检错工作。当错码为一个或奇数个时，因打乱了“1”数目的奇偶性，故能发现差错。但是，当错码为偶数个时，由于未破坏“1”数目的奇偶性，故不能发现偶数个错码。

2. 行列监督码

行列监督码也叫方阵校验码，这种码不仅可以克服奇偶监督码不能发现偶数个差错的缺点，还可以纠正某些位置的错码。其基本原理与简单的奇偶监督码相似，不同点在于每个码元都要受到纵、横两个方向的监督。行列监督码实质上是运用矩阵变换，把突发差错变成独立差错加以处理。因为这种方法比较简单，所以被认为是对抗突发差错很有效的手段。

3. 等比码

等比码又称定比码、恒比码或等重码(非零码组中“1”码的个数称为码重)。等比码的每个码组中，“1”和“0”的个数之比都是恒定的。例如，我国电报通信中电传机采用的 3:2 等比码，也叫五中取三码，即在 5 单位电传码的码组中(2^5=32)，取其“1”的数目恒为 3 的码组(C_5^3=10)，代表十个字符(0～9)，如表 3.4.1 所列。

表 3.4.1　3：2 等 比 码

十进制数字	1	2	3	4	5	6	7	8	9	0
3:2 等比码	01011	11001	10110	11010	00111	10101	11100	01110	10011	01101

在国际电传电报上通用的 ARQ 通信系统中，选用三个“1”、四个“0”的 3:4 码，又叫七中取三码。它有 C_7^3=35 个码组，可以完全替代电传机中 32 个不同的数字和符号。

在检测等比码时，通过计算接收码组中“1”的数目，判定传输有无错误。这种码除了“1”错成“0”和“0”错成“1”成对出现的错误以外，还能发现其它所有形式的错误，因此检错能力很强。实践中，应用这类码后，国际电报的误字率保持在 10^{-6} 以下。

4. 正反码

正反码主要用于 10 单位电码差错控制传输设备，它具有纠正码组中一个差错的能力，也能检查码组内所有两个以下的差错及大部分两个以上的差错。

每个正反码的码组由 10 个码元组成，前面 5 位是普通的五单位码(普通电传机码)；后 5

位是编码时加上去的监督码，其编码规则为：

(1) 当信息码组中“1”的个数为奇数时，监督码是信息码的重复；

(2) 当信息码组中“1”的个数为偶数时，监督码是信息码的反码。

3.4.4　数据通信中的纠错编码

随着编码理论的发展，现行的抗干扰编码发展成为两大类：分组码和卷积码。

分组码的特点是，每组监督元只监督本组信息码，与其它码组无关，具有分组独立监督的信用；卷积码的监督位可监督前后信息码组，具有连环监督的作用，因而有时也称之为连环码。它们的共同点是：所选用的监督码元和信息码元之间，都满足各自的数学关系。

1. 线性分组码

一个长为 n 的分组码，码字由两部分构成：信息码元(k 位)＋监督码元(n–k 位)。监督元是根据一定规则由信息码元变换得到的，变换规则不同就构成不同的分组码。如果码字中的监督位为信息位的线性组合——它们之间由一个线性方程来联系，就称其为线性分组码。

线性(n，k)分组码的每个码字有 n–k 个监督元，要从 k 个信息元中求出 n–k 个监督元，必须有(n–k)个独立的线性方程。根据不同的线性方程，可得到不同的(n，k)线性分组码。(n，k)分组码中信息元的全部组合(2^k)为码字的个数。

2. 循环码

线性分组码中一个很实用的子类是循环码。它可以用现代代数理论进行分析和构造，是计算机通信中常用的一种检、纠错码。循环码除了具有一般线性分组码的特性外，还具有循环性。

如果一个码组的每一次循环移位是另一码组，这种码叫做循环码。如(7，3)码就是一循环码。如果把信息 001，100，110，111，011，101，010，000 按照一定的规律加入监督码后，所得到新的码组为：

C0=0111001

C1=1011100

C2=0101110

C3=0010111

C4=1001011

C5=1100101

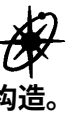
C6=1110010

C7=0000000

可见，它的一个码组的每一次循环移位是另一码组(全 0 码组除外)。循环码有两个显著特点：一是它既可以用线性方程确定，更适合于用代数方法进行分析研究；二是它具有循环移位特性，所需的编译码设备比较简单，容易实现。因此循环码在实际中得到了广泛的应用。

3. BCH 码

BCH 码(Bose-Chaudhuri-Hocquenghem codes)是自 1959 年发展起来的一种循环码，码的



生成多项式 g(x)与码的最小距离有关，因此，我们可以按照纠错的要求来直接确定码的构造。BCH 码的纠错能力很强，但译码方法比较复杂。由于译码方法的不断完善，BCH 码已经得到了广泛的应用。

第 4 章　串行通信总线标准及接口技术

本章主要内容：

★ **串行通信总线标准接口**
★ RS－232C **总线标准及应用**
★ RS－449/423/422/485 **标准总线接口及其应用**
★ I^2C **总线及应用实例**
★ SPI **总线及应用实例**
★ **单总线及应用实例**
★ USB **通用串行总线及应用**
★ IEEE-1394 **总线**
★ MPS **接口及应用**
★ Microwire **总线接口**
★ **其它串行接口的芯片操作**

4.1　串行通信总线标准接口

随着大规模集成电路技术的发展，在单个芯片上集成 CPU **以及组成一个单独工作系统所必须的** ROM、RAM、I/O **端口、**A/D、D/A **等外围电路已经实现，这就是常说的单片机或微控制器。任何一个微处理器都要与一定数量的部件和外围设备连接，但如果将各部件和每一种外围设备都分别用一组线路与** CPU **直接连接，那么连线将会错综复杂，甚至难以实现。为了简化硬件电路设计，简化系统结构，常用一组线路，配置以适当的接口电路，与各部件和外围设备连接，这组共用的连接线路被称为总线。采用总线结构便于部件和设备的扩充，统一的总线标准则容易使不同设备间实现互连。**

从广义上说，计算机通信方式可以分为并行通信和串行通信，相应的通信总线被称为并行总线和串行总线。并行通信速度快、实时性好，但由于占用的口线多，不适于小型化产品；而串行通信速率虽低，但在数据通信吞吐量不是很大的微处理电路中显得更加简易、方便、灵活。

串行通信是 CPU **与外界进行信息交换的一种方式，是指数据一位一位地按顺序传送的通信方式。串行通信有两种基本工作方式：异步传送和同步传送。串行通信的突出优点是只需一根或几根数据传输线，可大大降低硬件成本。虽然其传输速率低，但在通信中也得到了广泛使用。为保证可靠性高的通信要求，在选择接口标准时，须注意两点:**

① 通信速度和通信距离。标准串行接口的电气特性都有满足可靠传输时的最大通信速度和传送距离指标。但这两个指标之间具有相关性，适当地降低通信速度，可以提高通信距离，反之亦然。

② 抗干扰能力。对于标准接口，在保证不超过其使用范围时都有一定的抗干扰能力，可以保证可靠的信号传输。但在一些工业测控系统中，通信环境往往十分恶劣，因此在选择通信介质、接口标准时要充分注意其抗干扰能力，并采取必要的抗干扰措施。

所谓标准接口，就是明确定义若干信号线，使接口电路标准化、通用化，借助串行通信标准接口，不同类型的数据通信设备可以很容易实现它们之间的串行通信连接。采用标准接口后，能很方便地把各种计算机、外部设备、测量仪器等有机地连接起来，进行串行通信。标准异步串行通信接口有以下几类：RS－232C、RS－449(RS－422、RS－423 和 RS－485)，20 mA 电流环。RS－232C 是由美国电子工业协会(EIA)正式公布的、在异步串行通信中应用最广的标准总线，它包括了按位串行传输的电气和机械方面的规定，适合于短距离或带调制解调器的通信场合。为了提高数据传输速率和通信距离，EIA 又公布了 RS－422，RS－423 和 RS－485 串行总线接口标准。20 mA 电流环是一种非标准的串行接口电路，但由于它具有简单、对电器噪声不敏感的优点，因而在串行通信中也得到广泛使用。

4.2 RS－232C 总线标准及应用

4.2.1 RS－232C 总线标准接口及电器特性

RS－232C 是美国电子工业协会 EIA(Electronic Industry Association)制定的一种串行物理接口标准。RS 是英文“推荐标准”的缩写，232 为标识号，C 表示修改次数。RS－232C 总线标准规定了 21 个信号和 25 个引脚，包括一个主通道和一个辅助通道，在多数情况下主要使用主通道。对于一般双工通信，仅需几条信号线就可实现，包括一条发送线、一条接收线和一条地线。

RS－232C 标准规定的数据传输速率为 50、75、100、150、300、600、1200、2400、4800、9600、19200 b/s。驱动器允许有 2500 pF 的电容负载，通信距离将受此电容限制。信号传输速率为 20 kb/s 时，最大传输距离为 15 m。传输距离短的另一原因是 RS－232 属单端信号传送，存在共地噪声和不能抑制共模干扰等问题，因此一般用于短距离通信。

图 4.2.1 是计算机 25 芯串口引脚排列，表 4.2.1 给出其引脚信号功能。

图 4.2.2 是 9 芯串口引脚排列，表 4.2.2 给出其引脚信号功能。

表 4.2.3 列出 RS－232C 的电气特性。

图 4.2.1 计算机 25 芯串口引脚排列图　　图 4.2.2 计算机 9 芯串口引脚排列

表 4.2.1　计算机 25 芯串口引脚信号功能

引脚号	信号名称	方　向	信 号 功 能
1	ShieldGnd	—	接设备外壳，安全地
2	TXD	PC 机→对方	PC 机发送数据
3	RXD	PC 机←对方	PC 机接收数据
4	RTS	PC 机→对方	PC 机请求发送数据
5	CTS	PC 机←对方	对方已切换到接收状态(清除发送)
6	DSR	PC 机←对方	对方准备就绪
7	GND	—	信号地
8	DCD	PC 机←对方	PC 机收到远程信号(载波检测)
20	DTR	PC 机→对方	PC 机准备就绪
22	RI	PC 机←对方	通知 PC 机，线路正常(振铃指示)
9-19，21，23-25	NC	—	空

表 4.2.2　计算机 9 芯串口引脚信号功能

引脚号	信号名称	方　向	信 号 功 能
1	DCD	PC 机←对方	PC 机收到远程信号(载波检测)
2	RXD	PC 机←对方	PC 机接收数据
3	TXD	PC 机→对方	PC 机发送数据
4	DTR	PC 机→对方	PC 机准备就绪
5	GND	—	信号地
6	DSR	PC 机←对方	对方准备就绪
7	RTS	PC 机→对方	PC 机请求发送数据
8	CTS	PC 机←对方	对方已切换到接收状态(清除发送)
9	RI	PC 机←对方	通知 PC 机，线路正常(振铃指示)

表 4.2.3　RS－232C 电气特性

不带负载时驱动器输出电平 U_0	–25 V～+25 V
负载电阻 R_L 范围	3～7 kΩ
驱动器输出电阻 R_o	<300 Ω
负载电容(包括线间电容)C_L	<2500 pF
逻辑“0”时驱动器输出电平	5～15 V
逻辑“0”时负载端接收电平	>+3 V
逻辑“1”时驱动器输出电平	–15～–5 V
逻辑“1”时负载端接收电平	<–3 V
输出短路电流	<500 mA
驱动器转换速率	<30 V/μs

4.2.2 电平转换芯片介绍

由于 RS－232C 的逻辑 0 电平规定为+5～+15 V，逻辑 1 电平规定为–15～–5 V，因此在与 TTL 电路接口时必须经过电平转换。

电平转换可以用三极管等分离元件搭成，也可直接采用专用电平转换芯片。早期的常见芯片有需要±12 V 供电的 MC1488、MC1489 等，现在则出现了大量的单电源供电的电平转换芯片，其体积更小，连接简便，而且抗静电能力强。下面就常用的 MAX232 作一简单介绍。

MAX232 芯片是 MAXIM 公司生产的、包含两路接收器和驱动器的 RS－232 电平转换芯片，适用于各种 232 通信接口。MAX232 芯片内部有一个电源电压变换器，可以把输入的+5V 电源电压变换成为 RS－232C 输出电平所需的±10 V 电压。所以，采用此芯片接口的串行通信系统只需单一的+5 V 电源就可以了。对于没有±12 V 电源的场合，其适应性更强。加之其价格适中，硬件接口简单，所以被广泛采用。

MAX232 芯片的引脚排列如图 4.2.3 所示，其结构原理见图 4.2.4。

图 4.2.3 MAX232 芯片的引脚排列

图 4.2.4 中上半部分电容 C1、C2、C3、C4 及 V+、V–是电源变换部分。在实际应用中，器件对电源噪声很敏感。因此，Vcc 须要对地加去耦电容 C5，其值为 0.1 μF。电容 C1、C2、C3、C4 都选用钽电解电容，电容值为 1.0 μF(耐压值高于 16 V)，可以提高抗干扰能力。连接时电容必须尽量靠近器件，注意极性。

下半部分为发送和接收部分。实际应用中，T1IN、T2IN 和 R1OUT、R2OUT 可分别连接 TTL/CMOS 电平的 51 单片机的串行发送端 TxD 和接收端 RxD；T1OUT、T2OUT 和 R1IN、R2IN 分别接连至 RS－232 电平的 PC 机串行接收端和发送端。

采用 MAX232 芯片接口的 PC 机与 MCS－51 单片机串行通信接口电路如图 4.2.5 所示。从 MAX232 芯片中两路发送接收中任选一路连接。请注意其发送与接收引脚的对应，否则可能对器件或计算机串口造成永久性损坏。

图 4.2.4　MAX232 **芯片的结构原理图**

图 4.2.5　MAX232 **芯片的典型应用电路**

4.3 RS－449/423/422/485 标准总线接口及其应用

4.3.1 RS－232C 接口的主要缺点

RS－232C 接口的缺点主要表现在以下两个方面：

(1) 数据传输速率慢。RS－232C 规定的 20 kb/s 的传输速率虽然能满足异步通信要求(通常异步通信速率限制在 19.2 kb/s 以下)，但对某些同步系统来说，不能满足传送速率要求。

(2) 传送距离短。RS－232C 接口的一般装置之间电缆长度为 15 m，即使有较好的线路器件、优良的信号质量，电缆长度也不会超过 60 m。

鉴于 RS－232C 接口的上述缺点，1977 年 EIA 公布了新的标准接口 RS－449。RS－449 于 1980 年成为美国标准，在很多方面可代替 RS－232C。两者的主要差别是信号在导线上的传输方法不同。RS－232C 是利用传输信号线与公共地之间的电压差，RS－449 接口是利用信号导线之间的信号电压差，可在 1200 m 的双绞线上进行数字通信，速率可达 90 kb/s。RS－449 可以不使用调制解调器，它比 RS－232C 传输速率高，通信距离长，且由于 RS－449 系统用平衡信号差电路传输高速信号，所以噪声低，又可以多点或者使用公用线通信，此外，两台以上的设备可与 RS－449 通信电缆并联。RS－449 标准规定用 37 脚连接器。

RS－423/422(全双工)是 RS－449 标准的子集，规定了电气方面的要求。下面主要介绍单片机系统中常用的 RS－422A 和 RS－485 标准接口。

4.3.2 RS－422 串行总线标准及应用

RS－422A 标准是 EIA 公布的“平衡电压数字接口电路的电气特性”标准，这个标准是为改善 RS－232C 标准的电气特性，又考虑与 RS－232C 兼容而制定的。RS－422A 与 RS－232C 的关键不同在于把单端输入改为双端差分输入，信号地不再共用，双方的信号地也不再接在一起。

RS－422A 给出了对电缆、驱动器的要求，规定了双端电气接口形式，其标准是双绞线传送信号。它通过传输线驱动器，把逻辑电平变换成电位差；通过传输线接收器，由电位差转变成逻辑电平，实现信号接收。

RS－422A 比 RS－232C 传输信号距离长、速度快。传输率最大为 10 Mb/s，在此速率下，电缆允许长度为 120 米。如果采用较低传输速率，如 90 kb/s 时，最大距离可达 1200 米。

RS－422A 每个通道要用二条信号线，如果其中一条是逻辑“1”状态，另一条就为逻辑“0”。RS－422A 线路一般都需要两个通道，由发送器、平衡连接电缆、电缆终端负载、接收器几部分组成。在电路中规定只许有一个发送器，可有多个接收器，因此通常采用点对点通信方式。该标准允许驱动器输出为－6～6 V，接收器可以检测到的输入信号电平可低到 200 mV。

图 4.3.1 所示为平衡驱动差分接收电路。平衡驱动器的两个输出端分别为+VT 和–VT，故差分接收器的输入信号电压 VR=+VT–(–VT)=2VT，两者之间不共地，这样既可削弱干扰的影响，又可获得更长的传输距离及允许更大的信号衰减。

图 4.3.1　平衡驱动差分接收电路

4.3.3　RS－485 标准

RS－485 是 RS－422A 的变型：RS－422A 为全双工，可同时发送与接收；RS－485 则为半双工，在某一时刻，一个发送另一个接收。

真正的多点总线应由连接至总线的多个驱动器和接收器构成，并且其中任何一个均可发送或接收数据，也就是说两条信号线组成的单通道即可完成收发功能。RS-485 是一种多发送器的电路标准，它扩展了 RS-422A 的性能，允许双线总线上一个发送器驱动 32 个负载设备。负载设备可以是被动发送器、接收器或收发器(发送器和接收器的组合)。当用于多站互连时，可节省信号线，便于高速距离传送。许多智能仪器设备配有 RS-485 总线接口，便于将它们进行联网。

4.3.4　RS－232C、RS－422A、RS－485 性能比较

RS－232C、RS－422A 与 RS－485 的性能比较见表 4.3.1 所示。

表 4.3.1　RS－232C、RS－422A 与 RS－485 性能比较

接口	RS－232C	RS－422A	RS－485
操作方式	单端	差动方式	差动方式
最大距离/m	15(24 kb/s)	1 200(100 kb/s)	1 200(100 kb/s)
最大速率	200 kb/s	10 Mb/s	10 Mb/s
最大驱动器数目	1	1	32
最大接收器数目	1	10	32
接收灵敏度	±3 V	±200 mV	±200 mV
驱动器输出阻抗	300 Ω	60 kΩ	120 kΩ
接收器负载阻抗	3～7 kΩ	>4 kΩ	>12 kΩ
负载阻抗	3～7 kΩ	100 Ω	60 Ω
对共用点电压范围/V	±25	–0.25～+6	–7～+12

4.3.5 驱动芯片介绍

RS－485 接口可连接成半双工和全双工两种通信方式。半双工通信芯片有 SN75176、SN751276、SN75LBC184、MAX485、MAX1478、MAX3082、MAX1483 等。全双工通信的有 SN75179、SN75180、MAX488～491、MAX1482 等。下面以 MAX 系列芯片为例简要介绍。

1. 性能简介

MAXIM 公司生产的 MAX481/MAX483/MAX485/MAX487～MAX491 是用于 RS-485 和 RS－422 通信的低功率收发器件，每种芯片都由一个驱动器和一个收发器组成。主要特点如下：

(1) 单＋5 V 电源供电：

(2) 低功耗时，工作电流： 120～500 μA ;

静态电流： 120 μA (MAX483/487/488/489)；

300 μA (MAX481/485/490/491)；

(3) 关闭方式：MAX481/483/487 三种型号有低电流关机模式(驱动器和接收器处于禁止状态)消耗 0.1 μA 电流；

(4) 驱动器有过载保护功能;

(5) MAX 共模输入电压范围：–7～+12 V。

MAX481/MAX483/MAX485/MAX487～MAX491 性能比较见表 4.3.2。

表 4.3.2 MAX481/MAX483/MAX485/MAX487～MAX491 性能比较

型号	半/全双工	传输速率/(Mb/s)	转换率限制	低功耗关机	接收/驱动使能端	静态电流/mA	总线最大负载个数	管脚数
MAX481	半双工	2.5	无	有	有	300	32	8
MAX483	半双工	0.25	有	有	有	120	32	8
MAX485	半双工	2.5	无	无	有	300	32	8
MAX487	半双工	0.25	有	有	有	120	128	8
MAX488	全双工	0.25	有	无	无	120	32	8
MAX489	全双工	0.25	有	无	有	120	32	14
MAX490	全双工	2.5	无	无	无	300	32	8
MAX491	全双工	2.5	无	无	有	300	32	14

2. 管脚定义及工作电路

MAX481/MAX483/MAX485/MAX487 管脚图及典型工作电路见图 4.3.2，适于半双工通信。图中传输线为双绞线，Rt 为匹配电阻。

MAX488/MAX490 管脚图及典型工作电路见图 4.3.3，适于全双工通信。

MAX489/MAX491 管脚图及典型工作电路见图 4.3.4。

以上芯片引脚功能均见表 4.3.3，驱动器和接收器的输入输出关系见表 4.3.4 及表 4.3.5。

图 4.3.2　MAX481/MAX483/MAX485/MAX487 **管脚图及典型工作电路**

图 4.3.3　MAX488/MAX490 **管脚图及典型工作电路**

图 4.3.4　MAX489/MAX491 **管脚图及典型工作电路**

表 4.3.3　MAX481/MAX483/MAX485/MAX487～MAX491 **管脚说明**

管脚号			管脚名称	功能说明
MAX481/483 MAX485/487	MAX488 MAX490	MAX489 MAX491		
1	2	2	RO	**接收器输出：**A–B>+ 0.2V **时，**RO=1 **；** A–B<– 0.2V **时，**RO=0
2	—	3	$\overline{RE}$	**接收器输出使能：**$\overline{RE}$ =0 **时，允许接收器输出；** $\overline{RE}$ =1 **时，禁止接收器输出，**RO **为高阻**
3	—	4	DE	**驱动器输出使能：**DE=1 **时，允许驱动器工作；** DE=0 **时，禁止驱动器输出，**Y、Z **为高阻**
4	3	5	DI	**驱动器输入：**DI=1 **时，输出** Y **为高，**Z **为低** DI=0 **时，输出** Y **为低，**Z **为高** MAX481/483/485/487 **输出端为** A、B
5	4	6，7	GND	**地**
—	5	9	Y	**驱动器不反相输出端**
—	6	10	Z	**驱动器反相输出端**
6	—	—	A	**接收器同相输入/驱动器同相输出**
—	8	12	A	**接收器同相输入**
7	—	—	B	**接收器反相输入/驱动器反相输出**
—	7	11	B	**接收器反相输入**
8	1	14	V_{CC}	**电源**(4.75 V<V_{CC}<5.25 V)
—	—	1，8，13	NC	**空脚**

表 4.3.4　**发送功能表**

输入			输出	
$\overline{RE}$	DE	DI	Z	Y
X	1	1	0	1
X	1	0	1	0
0	0	X	**高阻**	**高阻**
1	0	X	**高阻***	**高阻***

表 4.3.5　**接收功能表**

输入			输出
$\overline{RE}$	DE	A－B	RO
0	0	>+ 0.2V	1
0	0	<– 0.2V	0
0	0	**输入开路**	1
1	0	X	**高阻***

说明：

***表示** MAX481/483/487 **处于关闭方式。**

4.3.6　应用电路

由 MAX48X/49X **系列收发器组成的差分平衡系统，抗干扰能力强，接收器可检测低达** 200 mV **的信号，传输数据可以从千米以外得到恢复，因此特别适用于远距离通信，可组成满足** RS－485/RS－422 **标准的通信网络。**

MAX481/MAX483/MAX485/MAX487 和 MAX489/MAX491 可用于总线(合用线)系统。图 4.3.5 电路中，驱动器有使能控制端 DE，各驱动器分时使用传输线(不发送数据的驱动器应被禁止)。当驱动器被禁止时，输出端 Y、Z 为高阻态，因而接收器具有高的输入阻抗。由于处于禁止状态的驱动器和多个接收器挂在传输线上不会影响信号的正常传送，故多个驱动器和接收器可共享一公用传输线，图 4.3.5 为典型半双工 RS-485 通信网络，网络上可挂 32 个站(MAX481/MAX483/MAX485)。如果使用 MAX487 作为站的收发器，由于它的输入阻抗是标准接收器的 4 倍，故网络上可挂 128 个站。

图 4.3.5　半双工 RS－485 通信网(合用线系统)

由 MAX489/MAX491 可组成全双工 RS－485 通信网，其线路连接如图 4.3.6 所示。

图 4.3.6　全双工 RS－485 通信网

4.3.7　RS－485 标准总线接口应用

RS－485 串行总线接口标准以差分平衡方式传输信号，具有很强的抗共模干扰的能力，允许一对双绞线上一个发送器驱动多个负载设备。利用单片机本身所提供的简单串行接口，加上总线驱动器，如 SN75176 等，可组合成简单的 RS－485 通信网络。

1. 总线驱动器芯片 SN75176

常用的 SN 系列 RS－485 总线驱动芯片有 SN75174、SN75175、SN75176。SN75176 芯片有一个发送器和一个接收器，适合作为 RS－485 总线驱动芯片。

SN75176 及其逻辑如图 4.3.7、表 4.3.6 及表 4.3.7 所示。

图 4.3.7　SN75176 芯片

表 4.3.6　驱动器逻辑表

输入 D	使能 DE	输出	
		A	B
H	H	H	L
L	H	L	H
X	L	Z	Z

表 4.3.7　接收器逻辑表

差分输入 A　　B	使能 $\overline{RE}$	输出 R
Vin≥0.2 V	L	H
– 0.2 V<Vin<0.2 V	L	?
Vin≤– 0.2 V	L	L
X	H	Z
开路	L	H

H=高电平，L=低电平，?=不定，X=无关系，Z=高阻抗(断开)

2. RS－485 方式构成的多机通信原理

在由单片机构成的多机串行通信系统中，一般采用主从式结构：从机不主动发送命令或数据，一切都由主机控制。因此在一个多机通信系统中，只有一台单机作为主机，各台从机之间不能相互通信，即使有信息交换也必须通过主机转发。采用 RS－485 构成的多机通信原理框图如图 4.3.8 所示。

图 4.3.8　采用 RS－485 构成的多机通信原理框图

在总线末端接一个匹配电阻，吸收总线上的反射信号，保证信号传输无毛刺。匹配电阻的取值应该与总线的特性阻抗相当。

当总线上没有信号传输时，总线处于悬浮状态，容易受干扰信号的影响。在总线上差分信号的正端 A^+和+5 电源间接一个 10 kΩ的电阻,正端 A^+和负端 B^-间接一个 10 kΩ的电阻，负端 B^-和地间接一个 10 kΩ的电阻，形成一个电阻网络。当总线上没有信号传输时，正端 A^+的电平大约为 3.2 V，负端 B－的电平大约为 1.6 V，即使有干扰信号，也很难产生串行通信的起始信号 0，从而增加了总线抗干扰的能力。

3. 通信规则

由于 RS－485 通信是一种半双工通信，发送和接收共用同一物理通道，在任意时刻只允许一台单机处于发送状态，因此要求应答的单机必须在侦听到总线上呼叫信号已经发送完毕，并且没有其它单机发出应答信号的情况下才能应答。半双工通信对主机和从机的发送和接收时序有严格的要求。如果在时序上配合不好，就会发生总线冲突，使整个系统的通信瘫痪，无法正常工作。要做到总线上的设备在时序上的严格配合，必须遵从以下几项原则：

(1) 复位时，主从机都应该处于接收状态。

SN75176 芯片的发送和接收功能转换是由芯片的 RE^*、DE 端控制的。RE^*=1，DE=1 时，SN75176 处于发送状态；RE*=0，DE=0 时，SN75176 处于接收状态。一般使用单片机的一根口线连接 RE^*、DE 端。在上电复位时，由于硬件电路稳定需要一定的时间，并且单片机各端口复位后处于高电平状态，这样就会使总线上各个分机处于发送状态，加上上电时各电路的不稳定，可能向总线发送信息。因此，如果用一根口线作发送和接收控制信号，应该将口线反向后接入 SN75176 的控制端，使上电时 SN75176 处于接收状态。

另外，在主从机软件上也应附加若干处理措施，如上电时或正式通信之前，对串行口做几次空操作，清除端口的非法数据和命令。

(2) 控制端 RE^*、DE 的信号的有效脉宽应该大于发送或接收一帧信号的宽度。

在 RS－232、RS－422 等全双工通信过程中，发送和接收信号分别在不同的物理链路上传输，发送端始终为发送端，接收端始终为接收端，不存在发送、接收控制信号切换问题。在 RS-485 半双工通信中，由于 SN75176 的发送和接收都由同一器件完成，并且发送和接收使用同一物理通道，必须对控制信号进行切换。控制信号何时为高电平，何时为低电平，一般以单片机的 TI、RI 信号作参考。发送时，检测 TI 是否建立起来，当 TI 为高电平后关闭发送功能转为接收功能；接收时，检测 RI 是否建立起来，当 RI 为高电平后，接收完毕，又可以转为发送。

4.4　I^2C 总线及应用实例

4.4.1　I^2C 总线介绍

I^2C(Inter-IC)总线是英文 INTER IC BUS 或 IC TO IC BUS 的简称，十多年前由 Philips 公司推出，是近年来在微电子通信控制领域广泛采用的一种新型总线标准。它是同步通信

的一种特殊形式，具有接口线少，控制方式简化，器件封装形式小，通信速率较高等优点。在主从通信中，可以有多个 I^2C 总线器件同时接到 I^2C 总线上，所有 I^2C 兼容的器件都有标准的接口，通过地址来识别通信对象， 使它们可以经由 I^2C 总线互相直接通信。此总线设计对系统设计及仪器制造都有利，因为可增加硬件的效率及简化电路，同时可提高仪器设备的可靠性，以及解决很多在设计数字控制电路上所遇到的接口问题。

I^2C 总线是由数据线 SDA 和时钟 SCL 构成的串行总线，可发送和接收数据。在 CPU 与被控 IC 之间、IC 与 IC 之间进行双向传送，最高传送速率为 400 kb/s。各种被控制电路均并联在这条总线上，就像电话机一样只有拨通各自的号码才能工作，所以每个电路和模块都有唯一的地址。在信息的传输过程中，I^2C 总线上并接的每一模块电路既是主控器(或被控器)，又是发送器(或接收器)，这取决于它所要完成的功能。CPU 发出的控制信号分为地址码和数据码两部分：地址码用来选址，即接通需要控制的电路，确定总线通信的器件；数据码是通信的内容。这样，各控制电路虽然挂在同一条总线上，却彼此独立，互不干扰。

1. I^2C 总线的电气特性与结构

I^2C 器件与 I^2C 总线的连接见图 4.4.1。为了避免总线信号的混乱，要求各设备连接到总线的输出端必须是开漏输出或集电极开路输出的结构。在图 4.4.1 中，SDA 及 SCL 为双向线，用上拉电阻接到电源正极。当总线空闲时，此两线都是“ 高 ”。设备上的串行数据线 SDA 接口电路应该是双向的，输出电路用于向总线上发数据，输入电路用于接收总线上的数据。串行时钟线也应是双向的，在总线上作为控制数据传送的主机要通过 SCL 输出电路发送时钟信号，同时要检测总线上的 SCL 上的电平，以决定什么时候发下一个时钟脉冲电平；作为接受主机命令的从机，要按总线上的 SCL 信号发出或接收 SDA 上的信号，也可以向 SCL 线发出低电平信号以延长总线时钟信号周期。总线空闲时，因各设备都是开漏输出，上拉电阻 *R*r 使 SDA 和 SCL 线都保持高电平。任一设备输出的低电平都使相应的总线信号线变低，也就是说各设备的 SDA 是“ 与 ”关系，SCL 也是“ 与 ”关系。由于不同的器件(CMOS、NMOS)都会接到 I^2C 总线，逻辑“ 0 ”(低)及“ 1 ”(高)的电平值能不能被固定，要视 V_{DD} 或 V_{CC} 的电平高低而定。

图 4.4.1　设备与总线的接口电路

总线对设备接口电路的制造工艺和电平都没有特殊的要求(NMOS、CMOS 都可以兼容)。数据传送率按 I^2C 总线可高达 100 kb/s，高速方式可高达 400 kb/s。总线上允许连接的设备数以总线上的电容量不超过 400 pF 为限。

总线的运行(数据传输)由主机控制。所谓主机，即启动数据传送的设备，它发出启动信号，发出时钟信号，传送结束时发出停止信号。通常，主机是微处理器，被主机寻访的设备都称为从机。为了进行通信，每个接到 I^2C 总线的设备都有一个唯一的地址，以便于主机寻访。主机和从机的数据传送，可以由主机发送数据到从机，也可以是从机发到主机。凡是发送数据到总线的设备称为发送器，从总线上接收数据的设备称为接受器。

I^2C 总线上允许连接多个微处理器及各种外围设备，如存储器、LED 及 LCD 驱动器、A/D 及 D/A 转换器等。为了保证数据可靠地传送，任一时刻总线只能由某一台主机控制，一个微处理器应该在总线空闲时发启数据。为了妥善解决多台微处理器同时发启数据而引起的传送冲突的问题，也就是决定总线控制权的问题，I^2C 总线允许连接不同传送速率的设备。多台设备之间时钟信号的同步过程称为同步化。

2. I^2C 总线的数据传输

在传输数据开始前，主控器件应发送起始位，通知从接收器件作好接收准备；在传输数据结束时，主控器件应发送停止位，通知从接收器件停止接收。这两种信号是启动和关闭 I^2C 器件的信号。以下分别为所需的起始位及停止位的时序条件(如图 4.4.2 所示)。

起始位时序：当 SCL 线在高位时，SDA 线由高转换至低。

停止位时序：当 SCL 线在高位时，SDA 线由低转换至高。

开始和停止条件由主控器产生。使用硬件接口可以很容易地检测开始和停止条件，没有这种接口的单片机必须以每时钟周期至少两次的频率对 SDA 取样，以便检测这种变化。

图 4.4.2　开始和停止条件

SDA 线上的数据在时钟高位时必须稳定。数据线上高低状态只有当 SCL 线的时钟信号为低电平时才可变换，如图 4.4.3 所示。输出到 SDA 线上的每个字节必须是 8 位，每次传输的字节不受限制，每个字节必须有一个确认位(又称应答位 ACK)。如果一接收器件在完成其它功能(如一内部中断)前不能接收另一数据的完整字节，它可以保持时钟线 SCL 为低，以促使发送器进入等待状态，当接收器件准备好接收数据的其它字节并释放时钟 SCL 后，数据传输继续进行。

图 4.4.3 I^2C 总线中的有效数据位

数据传送必须有确认位。与确认位对应的时钟脉冲由主控器产生，发送器在应答期间必须下拉 SDA 线，如图 4.4.4 所示。

图 4.4.4 I^2C 总线的确认位

当不能确认寻址的被控器件时，数据线保持为高，接着主控器产生停止条件终止传输。在传输结束时，主控接收器必须发出一个数据结束信号给被控发送器，被控发送器必须释放数据线，以允许主控器产生停止条件。合法的数据传输格式如下：

起始位	被控接收器地址	R/W	确认位	数据	确认位	…	停止位

I^2C 总线在起始位(开始条件)后的首字节决定哪个被控器将被主控器选择，例外的是“通用访问”地址，它可以寻址所有器件。当主控器输出一地址时，系统中的每一器件都将起始位后的前七位地址和自己的地址进行比较：如果相同，该器件认为自己被主控器寻址。该器件是作为被控接收器或是被控发送器则取决于第 8 位(R/W 位)。它是一个数据方向位(读／写)——“0”代表发送(写入)，“1”代表需求数据(读入)。数据传送通常以主控器所发出的停止位(停止条件)而终结，时序关系如图 4.4.5 所示。

3. C51 语言编写的 I^2C 总线通用读写函数

大多数 51 系列单片机没有设计专门的硬件电路，以实现在 I^2C 总线上的数据收发，但可以使用专用的 I^2C 总线控制芯片，或在 I/O 口上用软件模拟总线时序。后者的优点是不增加硬件成本，但占用 CPU 资源，因此常常应用在一些要求不高的场合，如与带有 I^2C 接口芯片的通信。

图 4.4.5　数据传输时序

在与 I^2C 接口芯片通信的过程中，单片机始终是主控器，参与主接收和主发送两种数据传输过程。

下面给出了 C51 语言编写的用单片机 I/O 口模拟 I^2C 总线的基本函数，具有一定的通用性。其中，SCL 接单片机 P1.0 口，SDA 接单片机 P1.1 口，DELAY_TIME 定义的是延时时间，“4”是根据 12M 晶振设置的，应用时可根据单片机工作频率和具体 I^2C 器件作适当调整。

例 4.4.1

```
/* 电平模拟函数和基本读写函数
void IIC_Start(void)；
void IIC_Stop(void)；
void SEND_0(void)；
void SEND_1(void)；
bit Check_Acknowledge(void)；
void Write_Byte(uchar b)reentrant；
bit Write_N_Bytes(uchar *buffer，uchar n)reentrant；
bit Read_N_Bytes(uchar SlaveAdr，uchar n，uchar *buffer)；
uchar Read_Byte(void)reentrant；
*/
#include<string.h>
#include<reg52.h>
#include<intrins.h>
#define DELAY_TIME 4
#define uchar unsigned char
#define uint unsigned int
sbit SCL=P1^0；
sbit SDA=P1^1；
void DELAY(uint t)
    {
```

```
        while(t!=0)
            t--；
        }
    void IIC_Start(void)
    {
        //启动 I²C 总线的函数，当 SCL 为高电平时使 SDA 产生一个负跳变
        SDA=1；
        SCL=1；
        DELAY(DELAY_TIME)；
        SDA=0；
        DELAY(DELAY_TIME)；
        SCL=0；
        DELAY(DELAY_TIME)；
    }
    void IIC_Stop(void)
    {
        //终止 I²C 总线，当 SCL 为高电平时使 SDA 产生一个正跳变
        SDA=0；
        SCL=1；
        DELAY(DELAY_TIME)；
        SDA=1；
        DELAY(DELAY_TIME)；
        SCL=0；
        DELAY(DELAY_TIME)；
    }
    void SEND_0(void)
    {
    //发送 0，在 SCL 为高电平时使 SDA 信号为低
        SDA=0；
        SCL=1；
        DELAY(DELAY_TIME)；
        SCL=0；
        DELAY(DELAY_TIME)；
    }
    void SEND_1(void)
    {
    //发送 1，在 SCL 为高电平时使 SDA 信号为高
        SDA=1；
        SCL=1；
```

```
    DELAY(DELAY_TIME)；
    SCL=0；
    DELAY(DELAY_TIME)；
}
bit Check_Acknowledge(void)
{
//发送完一个字节后检验设备的应答信号
    SDA=1；
    SCL=1；
    DELAY(DELAY_TIME/2)；
    F0=SDA；
    DELAY(DELAY_TIME/2)；
    SCL=0；
    DELAY(DELAY_TIME)；
    if(F0==1)
    return FALSE；
    return TRUE；
}
void Write_Byte(uchar b)reentrant
{
//向 I²C 总线写一个字节
    uchar i；
    for(i=0；i<8；i++)
    if((b<<i)&0x80)
        SEND_1( )；
    else
        SEND_0( )；
}
bit Write_N_Bytes(uchar *buffer，uchar n)reentrant
{
//向 I²C 总线写 n 个字节
    uchar i；
    IIC_Start( )；
    for(i=0；i<n；i++)
        {
            Write_Byte(buffer[i])；
            if(!Check_Acknowledge( ))
                {
                    IIC_Stop( )；
```

```
                    return(i==n)；
                }
            }
        IIC_Stop( )；
        return TRUE；
}
uchar Read_Byte(void)reentrant
{
//从 I²C 总线读一个字节
    uchar b=0，i；
    for(i=0；i<8；i++)
    {
    SDA=1；//释放总线
    SCL=1；//接受数据
    DELAY(10)；
    F0=SDA；
    DELAY(10)；
    SCL=0；
    if(F0==1)
        {
            b=b<<1；
            b=b|0x01；
        }
    else
        b=b<<1；
  }
  return b；
}
bit Read_N_Bytes(uchar SlaveAdr，uchar n，uchar *buffer)
{
//从 I²C 总线读 n 个字节
    uchar i；
    IIC_Start( )；
    Write_Byte(SlaveAdr)；          //向总线发送接收器地址
    if(!Check_Acknowledge( ))       //等待接收器应答信号
        return FALSE；
    for(i=0；i<n；i++)
        {
            buffer[i]=Read_Byte( )；
```

```
            if(i!=n)
                SEND_0( )；    //发送应答
            else
                SEND_1( )；    //发送非应答
            }
        IIC_Stop( )；
        return TRUE；
    }
```

4.4.2　24C 系列串行 E^2PROM 的应用

1. AT24C 系列串行 E^2PROM 简介

AT24C 系列串行 E^2PROM 具有 I^2C 总线接口功能，功耗小，电源电压宽(根据不同型号 2.5～6.0 V)，工作电流约为 3 mA，静态电流随电源电压不同为 30～110 μA，存储容量见表 4.4.1。

表 4.4.1　AT24C 系列串行 E^2PROM 参数

型号	容量	器件寻址字节(8 位)	一次装载字节数
AT24C01	128×8	$1010A_2A_1A_0$ R/W	4
AT24C02	256×8	$1010A_2A_1A_0$ R/W	8
AT24C04	512×8	$1010A_2A_1P_0$ R/W	16
AT24C08	1024×8	$1010A_2P_1P_0$ R/W	16
AT24C16	2048×8	$1010P_2P_1P_0$ R/W	16

(1) AT24C 系列 E^2PROM 接口及地址选择。由于 I^2C 总线可挂接多个串行接口器件，在 I^2C 总线中每个器件应有唯一的器件地址，按 I^2C 总线规则，器件地址为 7 位数据(即一个 I^2C 总线系统中理论上可挂接 128 个不同地址的器件)，它和 1 位数据方向位构成一个器件寻址字节，最低位 D_0 为方向位(读/写)。器件寻址字节中的最高 4 位(D_7～D_4)为器件型号地址，不同的 I^2C 总线接口器件的型号地址是厂家给定的，如 AT24C 系列 E^2PROM 的型号地址皆为 1010，器件地址中的低 3 位为引脚地址 $A_2A_1A_0$，对应器件寻址字节中的 D_3、D_2、D_1 位，在硬件设计时由连接的引脚电平给定。

对于 E^2PROM 的容量小于 256 B 的芯片(AT24C01/02)，8 位片内寻址(A_0～A_7)即可满足要求。然而对于容量大于 256 B 的芯片，8 位片内寻址范围不够，如 AT24C16，相应的寻址位数应为 11 位(2^{11}=2048)。若以 256 B 为 1 页，则多于 8 位的寻址视为页面寻址。在 AT24C 系列中，对页面寻址位采取占用器件引脚地址(A_2、A_1、A_0)的办法，如 AT24C16 将 A_2、A_1、A_0 作为页地址。凡在系统中引脚地址用作页地址后，该引脚在电路中不得使用，作悬空处理。AT24C 系列串行 E^2PROM 的器件地址寻址字节如表 4.4.1 所示，表中 $P_0P_1P_2$ 表示页面寻址位。

(2) AT24C 系列 E^2PROM 读写操作。对 AT24C 系列，E^2PROM 的读写操作完全遵守 I^2C 总线的主收从发和主发从收的规则。

连续写操作是对 E^2PROM 连续装载 n 个字节数据的写入操作，n 随型号不同而不同，

一次可装载的字节数见表 4.4.1。SDA 线上连续写操作的数据状态如图 4.4.6 所示，S 表示 START 起始位，A 表示 ACK 应答位。

图 4.4.6 SDA 线连续写操作数据状态

AT24C 系列片内地址在接收到每一个数据字节地址后自动加 1，故装载一页以内规定数据字节时，只须输入首地址，若装载字节多于规定的最多字节数，数据地址将“上卷”，前面的数据被覆盖。

连续读操作是为了指定首地址，需要两个“伪字节写”来给定器件地址和片内地址，重复一次启动信号和器件地址(读)，就可读出该地址的数据。由于“伪字节写”中并未执行写操作，因此地址没有加 1。以后每读取一个字节，地址自动加 1。在读操作中，接收器接收到最后一个数据字节后不返回肯定应答(保持 SDA 高电平)，随后发停止信号。SDA 上连续读操作的数据状态如图 4.4.7 所示。

图 4.4.7 SDA 线连续读操作数据状态

2. 51 单片机与 AT24C16 通信的硬件实现及汇编语言程序

(1) 硬件电路。图 4.4.8 是用 51 单片机 P1 口模拟 I^2C 总线与 E^2PROM 连接的电路图(以 AT24C16 为例)。由于 AT24C16 是漏极开路，图中 R_1、R_2 为上拉电阻(5.1 kO)；A_0～A_2 为地址引脚、TEST 为测试脚，它们均悬空。

图 4.4.8 51 单片机 P1 口与 E^2PROM 连接电路图

(2) 软件实现。由前述分析和图 4.4.8 的硬件电路，可编制 E^2PROM 的读写子程序。两者的主要区别在于读子程序需发送器件地址(写)和片内地址作为伪字节，之后再发送一次开始信号和器件地址(读命令)。

例 4.4.2 读写子程序

```
；写串行 E2PROM 子程序 EEPW
；  (R3)=器件地址
；  (R4)=片内字节地址
；  (R1)=欲写数据存放地址指针
```

```
；  (R7)=连续写字节数 n
EEPW:  MOV     P1，#0FFH
       CLR     P1.0          ；发开始信号
       MOV A，  R3           ；送器件地址
       ACALL   SUBS
       MOV A，  R4           ；送片内字节地址
       ACALL   SUBS
AGAIN:  MOV    A，@R1
       ACALL   SUBS          ；调发送单字节子程序
        INC    R1
        DJNZ   R7，AGAIN     ；连续写 n 个字节
        CLR    P1.0          ；SDA 置 0， 准备送停止信号
        ACALL  DELAY         ；延时以满足传输速率要求
        SETB   P1.1          ；发停止信号
        ACALL  DELAY
        SETB   P1.0
        RET
SUBS:   MOV    R0，#08H      ；发送单字节子程序
LOOP:   CLR    P1.1
        RLC    A
        MOV    P1.0，C
        NOP
        SETB   P1.1
        ACALL  DELAY
        DJNZ   R0，LOOP      ；循环 8 次送 8 个位
        CLR    P1.1
        ACALL  DELAY
        SETB   P1.1
REP:    MOV    C，P1.0
        JC     REP           ；判应答到否，未到则等待
        CLR    P1.1
        RET
DELAY:  NOP
        NOP
        RET

；读串行 E2PROM 子程序 EEPR
；(R1)=欲读数据存放地址指针
；(R3)=器件地址
```

```
;(R4)=片内字节地址
;(R7)=连续读字节数
EEPR:   MOV     P1，#0FFH
        CLR     P1.0            ;发开始信号
        MOV     A，R3           ;送器件地址
        ACALL   SUBS            ;调发送单字节子程序
        MOV     A，R4           ;送片内字节地址
        ACALL   SUBS
        MOV     P1，#0FFH
        CLR     P1.0            ;再发开始信号
        MOV     A，R3
        SETB    ACC.0           ;发读命令
        ACALL   SUBS
MORE:   ACALL   SUBR
        MOV     @R1，A
        INC     R1
        DJNZ    R7，MORE
        CLR     P1.0
        ACALL   DELAY
        SETB    P1.1
        ACALL   DELAY
        SETB    P1.0            ;送停止信号
        RET
SUBR:   MOV     R0，#08H        ;接受单字节子程序
LOOP2:  SETB    P1.1
        ACALL   DELAY
        MOV     C，P1.0
        RLC     A
        CLR     P1.1
        ACALL   DELAY
        DJNZ    R0，LOOP2
        CJNE    R7，#01H，LOW
        SETB    P1.0            ;若是最后一个字节，置 A=1
        AJMP    SETOK
LOW:    CLR     P1.0            ;否则置 A=0
SETOK:  ACALL   DELAY
        SETB    P1.1
        ACALL   DELAY
        CLR     P1.1
```

```
ACALL    DELAY
SETB     P1.0                ；应答完毕，SDA 置 1
RET
```

在程序中，多处调用了 DELAY 子程序(仅两条 NOP 指令)，这是为了满足 I^2C 总线上数据传送速率的要求，即只有当 SDA 数据线上的数据稳定下来之后才能进行读写(即 SCL 线发出正脉冲)。另外，在读最后一个数据字节时，置应答信号为“1”，表示读操作即将完成。

3. 51 单片机与 24C01 通信的 C51 语言程序

用 51 单片机 P1 口模拟 I^2C 总线与 24C01，连接的硬件电路与图 4.4.8 非常相似，所不同的是，24C01 无 TEST 脚，代之以 WP(硬件写保护)脚，如图 4.4.9 所示。R4 为 10 kΩ上拉电阻；A_0～A_2 为地址引脚，接地；SDA 接 P1.0，Sck 接 P1.1，WP 接 P1.2。单片机为 ATMEL 公司的 AT89C51。

图 4.4.9　24C01 接线图

例 4.4.3

C51 语言源程序

```
#include <reg51.h>
#include <intrins.h>
#define NOP _nop_  ( )；_nop_  ( )；_nop_  ( )；_nop_  ( ) /*空指令 延时*/
#define uchar unsigned char
#define uint unsigned int
#define AddWr 0xa0 /*器件地址选择及写标志*/
#define AddRd 0xa1 /*器件地址选择及读标志*/
/*有关全局变量*/
sbit Sda= P1^0；  /*串行数据*/
sbit Scl= P1^1；  /*串行时钟*/
sbit WP= P1^2；  /*硬件写保护*/
void mDelay(uchar j)
{     uint i；
for( ；j>0；j--)
  { for(i=0；i<125；i--)
    { ；}
  }
}
/*发送起始条件*/
void Start(void) /*起始条件*/
{
Sda=1；
Scl=1；
```

```
NOP；
Sda=0；
NOP；
}
void Stop(void) /*停止条件*/
{
Sda=0；
Scl=1；
NOP；
Sda=1；
NOP；
}
void Ack(void) /*应答位*/
{
Sda=0；
NOP；
Scl=1；
NOP；
Scl=0；
}
void NoAck(void) /*反向应答位*/
{
Sda=1；
NOP；
Scl=1；
NOP；
Scl=0；
}
void Send(uchar Data) /*发送数据子程序，Data 为要求发送的数据*/
{
uchar BitCounter=8； /*位数控制*/
uchar temp； /*中间变量控制*/
do
{
temp=Data；
Scl=0；
NOP；
if((temp&0x80)==0x80)/* 如果最高位是 1*/
Sda=1；
```

```
else
Sda=0；
Scl=1；
temp=Data<<1； /*RLC*/
Data=temp；
BitCounter--；
}while(BitCounter)；
Scl=0；
}
uchar Read(void) /*读一个字节的数据，并返回该字节值*/
{
uchar temp=0；
uchar temp1=0；
uchar BitCounter=8；
Sda=1；
do{
Scl=0；
NOP；
Scl=1；
NOP；
if(Sda) /*如果 Sda=1*/
temp=temp|0x01； /*temp 的最低位置 1*/
else
temp=temp&0xfe； /*否则 temp 的最低位清 0*/
if(BitCounter-1)
{    temp1=temp<<1；
temp=temp1；
}
BitCounter--；
}while(BitCounter)；
return(temp)；
}
void WrToROM(uchar Data[ ]，uchar Address，uchar Num)
{
uchar i；
uchar *Pdata；
Pdata=Data；
for(i=0；i<Num；i++)
{
```

```
Start( )； /*发送启动信号*/
Send(AddWr)； /*发送 SLA+W*/
Ack( )；
Send(Address+i)； /*发送地址*/
Ack( )；
Send(*(Pdata+i))；
Ack( )；
Stop( )；
mDelay(20)；
}
}
void RdFromROM(uchar Data[ ]，uchar Address，uchar Num)
{
uchar i；
uchar *Pdata；
Pdata=Data；
for(i=0；i<Num；i++)
{
Start( )；
Send(AddWr)；
Ack( )；
Send(Address+i)；
Ack( )；
Start( )；
Send(AddRd)；
Ack( )；
*(Pdata+i)=Read( )；
Scl=0；
NoAck( )；
Stop( )；
}
}
void main( )
{
uchar Number[4]={1，2，3，4}；
WP=1；
WrToROM(Number，4，4)； /*将初始化后的数值写入 E2PROM*/
mDelay(20)；
Number[0]=0；
```

```
Number[1]=0；
Number[2]=0；
Number[3]=0； /*将数组中的值清掉，以验证读出的数是否正确*/
RdFromROM(Number，4，4)；
}
```

4. **单片机与** 24C65 **通信的** PL/M51 **语言程序**

24C65 **是目前容量较大的一种串行** E^2PROM **芯片，容量为** 8 KB，I^2C **接口。芯片管脚排列见图** 4.4.10。A2、A1、A0 **为片选地址管脚，通过排列构成** 8 **个片选地址，存储空间最大为** 64 KB。

对 24C65 **的读写一个字节的操作由** 4 **个字节组成，如图** 4.4.11 **所示。**

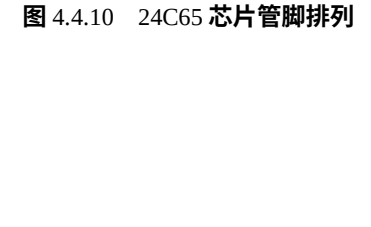


图 4.4.10　24C65 **芯片管脚排列**

图 4.4.11　24C65 **读写格式**

AT89C51 **单片机与** 24C65 **的连接如图** 4.4.10 **所示。**EEP_SCL **接单片机的** P1.0 **口，**EEP_SDA **接** P1.1 **口；**A2、A1、A0 **管脚均接地，即图** 4.4.11 **中第** 1 **个字节的** $A_2A_1A_0$ **位为** 000。

由 24C65 **的操作格式可知，按字节写入的方式，每写入一个字节的数据实际要对其写四个字节(程序流程见图** 4.4.12)，**因此写一个字节的子程序可作为一个基本“写”模块。遵循** I^2C **总线的时序要求，用** PL/M51 **语言编写的** 24C65 **“写”模块如下。**

例 4.4.4

```
；信号线连接与图 4.4.10 相同，单片机 P1.0 口接 SCL，P1.1 口接 SDA
WRITE:      PROCEDURE(X)；
            DECLARE X BYTE；
            DECLARE COUNT BYTE；
            DECLARE P10 BIT AT(90H) REG；
```

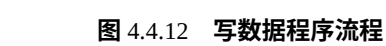


图 4.4.12　**写数据程序流程**

```
        DECLARE P11 BIT AT(91H) REG；
        P10=0；   启动总线
        ；
        P11=1；
        ；
        P10=1；
        ；
        P11=0；
        ；
        COUNT=0；
        DO WHILE(COUNT<8)；
            P10=0；
            ；
            P11=BOOLEAN(ROL(X，COUNT+1))AND 01H；
            ；
            P10=1；
            ；
            COUNT=COUNT+1；
            ；
        END；
        P10=0；  停止总线
        ；
        P11=0；
        ；
        P10=1；
        ；
        P11=1；
    END WRITE；
```

图 4.4.13　读数据程序流程

从 24C65 中读出一个数据的程序流程见图 4.4.13，程序既有写控制字和写地址操作，还有读出数据操作。编程时，既要调用上面的“写”模块，还要有读一个字节的“读”模块。PL/M51 语言编写的读模块程序如下：

例 4.4.5

```
；信号线连接与图 4.4.10 相同，单片机 P1.0 口接 SCL，P1.1 口接 SDA
READ:    PROCEDURE；
        DECLARE DATA BYTE；
        DECLARE TIME BYTE；
        DECLARE P10 BIT AT(90H) REG；
        DECLARE P11 BIT AT(91H) REG；
        P10=0；  启动总线
```

```
        ;
        P11=1;
        ;
        P10=1;
        ;
        P11=0;
        ;
        DATA=0;
        TIME=8;
        P10=0;
        DO WHILE(TIME>0);
            DATA=DATA OR (SHL(EXP AND(P11)，TIME-1));
            P10=1;
            ;
            P10=0;
            ;
            TIME=TIME-1;
            ;
        END;
        P10=0;   停止总线
        ;
        P11=0;
        ;
        P10=1;
        ;
        P11=1;
        RETURN(DATA);
    END READ;
```

4.4.3　数字温度传感器的应用实例

AD7XXX 系列是 AD 公司生产的数字温度传感器，其内部包括一个带隙温度传感器和一个 10 位 A/D 转换器，精度可达 0.25℃，是 LM75 的升级替换产品，可广泛应用于个人计算机、电子测试设备、办公设备、家用电器、过程控制等场合。该系列有 AD7414、AD7415、AD7416、AD7814 等四种型号，它们的工作原理相同，现以 AD7416 为例进行介绍。

1. AD7416 的主要特点

AD7416 包括下列特点：温度范围为–55～125 ℃；I^2C 接口；有超温指示；最大并联数 8 个；工作电压 2.7～5.5 V；转换时间 400 μs；小封装 SOT－23。

AD7416 的引脚排列如图 4.4.14 所示。每片 AD7416 的地址由 A_0、A_1、A_2 决定，可以由用户设置。地址格式为

1001 $A_2A_1A_0$ R/W

允许从 1001000 至 1001111 共 8 个地址中选择。OTI 为超温掉电输出引脚(漏极开路)。

AD7416 功能框图如图 4.4.15 所示，由带隙温度传感器、10 位 A/D 转换器、温度寄存器、可设点比较器、故障排队计数器等组成。传感器将温度转换成电压，再由 A/D 转换器转换成 10 位数字量送温度值寄存器。A/D 转换器的一次转换时间约 400 μs。

图 4.4.14 AD7416 引脚图

图 4.4.15 AD7416 功能框图

温度值寄存器是一个 16 位只读寄存器，它的高 10 位 D_{15}～D_6 由 A/D 转换器送来的数字量以补码格式储存，低 6 位 D_5～D_0 未用，小数点在 D_8、D_7 之间，如表 4.4.2 所示。

表 4.4.2 温度值寄存器

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5～D_0
B_9	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	未用

表 4.4.2 显示 A/D 转换器的全部理论范围为–128～+127 ℃。实际应用中，温度的测量范围将取决于器件的正常工作温度范围，表 4.4.3 中所列的数据为理论工作范围中的一些典型值。

表 4.4.3　温度值寄存器

温度	数字量输出
–128℃	1000000000
–125℃	1000001100
–100℃	1001110000
–75℃	1011010100
–50℃	1100111000
–25℃	1110011100
– 0.25℃	1111111111
0℃	0000000000
+ 0.25℃	0000000001
+10℃	0000101000
+25℃	0001100100
+50℃	0011001000
+75℃	0100101100
+100℃	0110010000
+125℃	0111110100
+127℃	0111111100

2. 51 单片机与 AD7416 连接的例子及汇编程序

图 4.4.16 给出了一个串行总线测温系统，由 AD7416、AD7414 和 AD7814 组成。AD7416、和 AD7414 的 OTI 输出“线与”，驱动一个公共的温度越限指示灯。

例 4.4.6

```
ADCH      EQU    32H      ；采样值高字节
ADCL      EQU    31H      ；采样值低字节
ADCNUM    EQU    30H      ；采样次数
ADCS      BIT    P1.5     ；AD7814 片选
ADSCLK    BIT    P1.1     ；AD7814 时钟
ADDOUT    BIT    P1.7     ；AD7814 数据输出
ADSCL     BIT    P1.1     ；AD7416 时钟
ADSDA     BIT    P1.7     ；AD7416 数据 I/O

；AD7416 的采样参考程序；AD7414、AD7415 的采样程序与 AD7416 相似，但地址不同
SAMPLE74:  MOV    ADCNUM，  #8        ；连续采样 8 次
SE074：    MOV    R6，#9EH            ；片选 AD7416 的地址写操作
           MOV    R5，#1              ；选中配置寄存器
           MOV    R4，#18H            ；给配置寄存器赋值
           LCALL  WRCOM               ；三字节的写操作
           MOV    R6，#10011110B
           MOV    R5，#3              ；选中温度上限寄存器
           MOV    R4，#40H            ；上限温度=64 ℃
           LCALL  WRCOM
           MOV    R6，#10011110B
           MOV    R5，#2              ；选中温度下限寄存器
           MOV    R4，#20H            ；上限温度=32 ℃
```

```
                LCALL   WRCOM
                MOV     R6，#10011110B
                MOV     R5，#0              ；选中温度寄存器
                MOV     WREXE               ；两字节的写操作
                MOV     DATA1，#10011111B   ；片选 AD7416，读操作
                LCALL   RDCOM
                ……  ；数据处理
                DJNZ    ADCNUM，SE074       ；采样未完，返回
WRCOM：         LCALL   BEGIN               ；三字节的写操作
                MOV     DATA1，R6
                LCALL   OUTBYTE             ；输出字节
                MOV     DATA1，R5
                LCALL   OUTBYTE
                MOV     DATA1，R4
                LCALL   OUTBYTE
                LCALL   STOP
                RET
REXE：          LCALL   BEGIN               ；两字节的写操作
                MOV     DATA1，R6
                LCALL   OUTBYTE
                MOV     DATA1，R5
                LCALL   OUTBYTE
                LCALL   STOP
                RET
RDCOM：         LCALL   BEGIN               ；读操作
                LCALL   OUTBYTE
                LCALL   INBYTE              ；输入字节
                MOV     ADCH，DATA1
                LCALL   NACK                ；MCU 使 ADSDA 数据线变为低电平
                LCALL   INBYTE
                MOV     ADCL，DATA1
                LCALL   ACK                 ；MCU 使 ADSDA 数据线变为高电平
                LCALL   STOP
                RET
OUTBYTE:        MOV     R7，#8              ；输出字节
OE1：           MOV     A，DATA1
                RLC     A
                MOV     ADSDA，C
                MOV     DATA1，A
                LCALL   CLOCK
                DJNZ    R7，OE1
                LCALL   ACK                 ；AD7416 产生应答
                RET
NBYTE：         SETB    ADSDA               ；输入字节
```

```
        MOV     R7，#8
INE1：  LCALL   CLOCK
        MOV     A，DATA1
        RLC     A
        MOV     DATA1，A
        DJNZ    R7，INE1
        RET
NACK：  CLR     ADSDA       ；AD7416 无应答
        LCALL   CLOCK       ；ADSDA 数据线为 0，一个时钟脉冲之后，ADSDA 为 1
        RET
ACK：   SETB    ADSDA；AD7416 有应答
        LCALL   CLOCK       ；ADSDA 为 1，一个时钟脉冲之后，ADSDA 为 0
        RET
STOP：  CLR     ADSDA       ；产生停止信号
        SETB    ADSCL
        NOP
        NOP
        NOP
        NOP
        SETB    ADSDA
        RET
BEGIN： SETB    ADSDA       ；产生开始信号
        SETB    ADSCL
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR     ADSDA
        NOP
        NOP
        NOP
        NOP
        CLR     ADSCL
        RET
CLOCK： NOP                 ；产生时钟脉冲
        SETB    ADSCL
        NOP
        NOP
        NOP
        NOP
        MOV     C，ADSDA
        CLR     ADSCL
        RET
```

图 4.4.16 串行总线测温系统

4.5　SPI 总线及应用实例

4.5.1　SPI 总线介绍

SPI(Serial Peripheral Interface 串行外设接口)总线系统是一种同步串行外设接口，允许 MCU 与各种外围设备以串行方式进行通信。外围设备包括简单的 TTL 移位寄存器(用作并行输入或输出口)、复杂的 LCD 显示驱动器或 A/D 转换器等。SPI 系统可直接与各个厂家生产的多种标准外围器件直接接口，它使用 4 条线：串行时钟线(SCK)、主机输入/从机输出数据线 MISO(简写为 SO)、主机输出/从机输入数据线 MOSI(简写为 SI)和低电平有效的从机选择线 $\overline{SS}$ (或 $\overline{OE}$)。由于 SPI 系统总线只需 3～4 根数据和控制线即可扩展具有 SPI 接口的各种 I/O 器件，而并行总线扩展方法需 8 根数据线、8～16 位地址线、2～3 位控制线，因而 SPI 总线的使用可以简化电路设计，省掉了很多常规电路中的接口器件，提高了设计的可靠性。

1. SPI 总线的组成

SPI 总线可在软件的控制下构成各种简单或复杂的系统，如：1 个主 MCU 和几个从 MCU；几个从 MCU 相互连接构成多主机系统(分布式系统)；1 个主 MCU 和 1 个或几个从 I/O 设备。在大多数应用场合中，使用 1 个 MCU 作为主机，它控制数据向 1 个或多个从外围器件的传送。从器件只能在主机发命令时才能接收或向主机传送数据，其数据的传输格式是高位(MSB)在前，低位(LSB)在后。SPI 典型结构如图 4.5.1 所示。

图 4.5.1　SPI 总线的组成

在把 SPI 与几种不同的串行 I/O 芯片相连时，必须使用每片的允许控制端，这可用 MCU 的 I/O 端口来控制。此时应特别注意这些串行 I/O 芯片的输入/输出特性：

(1) 输入芯片的串行数据输出是否有三态控制端。平时未选中芯片的输出端应处于高阻态。若没有三态控制端，应外加三态门；否则 MCU 的 MISO 端只能连接 1 个输入芯片。

(2) 输出芯片的串行数据输入是否有允许控制端。即应该只有在该芯片允许时，SCK 脉冲才把串行数据移入该芯片；芯片禁止时，SCK 对芯片无影响。若没有允许控制端，应在外部用门电路对 SCK 进行控制后，再加到芯片的时钟输入端，或者 SPI 只连接 1 个芯片，不能再连接其它输入或输出芯片。

2. SPI 总线在 8031 单片机中的实现方法

对于没有 SPI 接口的 8031 单片机来说，可使用软件来模拟 SPI 的操作，包括串行时钟、数据输入和输出。对于不同的串行接口外围芯片，它们的时钟时序有可能不同。

在图 4.5.1 中，假定 CPU 为 8031 单片机，用 P1.0 模拟 MOSI，P1.1 模拟 SCK，P1.2 模拟 $\overline{SS}$，P1.3 模拟 MISO 线。对于在 SCK 的上升沿输入(接收)数据和在下降沿输出(发送)数据的器件，一般应取图中的串行时钟输出 P1.1 的初始状态为 1，在允许接口芯片后，置 P1.1 为 0。因此，CPU 输出 1 位 SCK 时钟，同时，使接口芯片内的数据串行移位，从而输出 1 位数据至 8031 的 P1.3；再置 P1.1 为 1，使 8031 从 P1.0 输出 1 位数据(先为高位)至串行接口芯片。到此，模拟 1 位数据输入/输出完成。以后再置 P1.1 为 0，模拟下 1 位的输入/输出，依次循环 8 次，可完成 1 次通过 SPI 传输 1 个字节的操作。对于在 SCK 的下降沿输入数据和上升沿输出数据的器件，则应取串行时钟输出的初始状态为 0，在接口芯片允许时，先置 P1.1 为 1，此时，外围接口芯片输出 1 位数据(CPU 接收 1 位数据)，再置时钟为 0，外围接口芯片接收 1 位数据(CPU 发送 1 位数据)，从而完成 1 位数据的传送。

4.5.2 A/D 转换的实例

因为 51 单片机没有提供专用硬件电路来实现 SPI 总线，故需要用软件实现。下面以 TLC2453 为例，加以说明。

TLC2543 是 TI 公司的 12 位串行模数转换器，使用开关电容逐次逼近技术完成 A/D 转换过程。由于 TLC2543 是串行输出结构，能够节省 51 系列单片机 I/O 资源，且价格适中，分辨率较高，因此在仪器仪表中有较为广泛的应用。

1. TLC2543 的特点

(1) 12 位分辨率 A/D 转换器；
(2) 在工作温度范围内 10 μs 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 66 kb/s；
(6) 线性误差±1LSBmax；
(7) 有转换结束输出 EOC；
(8) 具有单、双极性输出；
(9) 可编程的 MSB 或 LSB 前导；
(10) 可编程输出数据长度。

2. TLC2543 的引脚排列及说明

TLC2543 有两种封装形式：DB、DW 或 N 封装以及 FN 封装，这两种封装的引脚排列如图 4.5.2 所示，引脚说明见表 4.5.1。

图 4.5.2　TLC2543 的封装

表 4.5.1　TLC2543 引脚说明

引脚号	名称	I/O	说明
1～9 11，12	AIN0～ AIN10	I	模拟量输入端。11 路输入信号由内部多路器选通。对于 4.1 MHz 的 I/O CLOCK，驱动源阻抗必须小于或等于 50 Ω，而且用 60 pF 电容来限制模拟输入电压的斜率
15	$\overline{\text{CS}}$	I	片选端。在 $\overline{\text{CS}}$ 端由高变低时，内部计数器复位。由低变高时，在设定时间内禁止 DATAINPUT 和 I/O CLOCK
17	DATAINPUT	I	串行数据输入端。由 4 位的串行地址输入来选择模拟量输入通道
16	DATAOUT	O	A/D 转换结果的三态串行输出端。$\overline{\text{CS}}$ 为高时处于高阻抗状态，$\overline{\text{CS}}$ 为低时处于激活状态
19	EOC	O	转换结束端。在最后的 I/O CLOCK 下降沿之后，EOC 从高电平变为低电平，并保持到转换完成和数据准备传输为止
10	GND	—	地。GND 是内部电路的地回路端。除另有说明外，所有电压测量都相对 GND 而言
18	I/OCLOCK	I	输入/输出时钟端。I/O CLOCK 接收串行输入信号并完成以下四个功能：(1) 在 I/O CLOCK 的前 8 个上升沿，8 位输入数据存入输入数据寄存器；(2) 在 I/O CLOCK 的第 4 个下降沿，被选通的模拟输入电压开始向电容器充电，直到 I/O CLOCK 的最后一个下降沿为止；(3) 将前一次转换数据的其余 11 位输出到 DATAOUT 端，在 I/O CLOCK 的下降沿时数据开始变化；(4) I/O CLOCK 的最后一个下降沿，将转换的控制信号传送到内部状态控制位
14	REF+	I	正基准电压端。基准电压的正端(通常为 VCC)被加到 REF+，最大的输入电压范围由加于本端与 REF–端的电压差决定
13	REF–	I	负基准电压端。基准电压的低端(通常为地)被加到 REF–
20	VCC	I	电源

3. 接口时序

可以用四种传输方法使 TLC2543 得到全 12 位分辨率，每次转换和数据传递可以使用 12 或 16 个时钟周期。

一个片选($\overline{CS}$)脉冲要插到每次转换的开始处，或是在转换时序的开始处变化一次后保持$\overline{CS}$为低，直到时序结束。

图 4.5.3 显示每次转换和数据传递使用 16 个时钟周期，和在每次传递周期之间插入$\overline{CS}$的时序，图 4.5.4 显示每次转换和数据传递使用 16 个时钟周期，仅在每次转换序列开始处插入一次$\overline{CS}$的时序。

图 4.5.3 16 个时钟传送时序图(使用$\overline{CS}$，MSB 在前)

GMS90C32 是 LGS 公司的 51 系列单片机之一，与 INTEL 公司的 80C32 完全兼容，不带 SP 接口。为了和 TLC2543 模数转换器接口，需要用软件来模拟 SPI 的时序操作。图 4.5.5 是 TLC2543 和 GMS90C32 的接口简图，TLC2543 的 I/O 时钟、数据输入、片选$\overline{CS}$由双向 I/O 口 P1 的引脚 P1.0、P1.1、P1.3 提供。TLC2543 的转换结果数据通过口 1 的 P1.2 脚接收，通道选择和方式数据通过 P3 口输入到微控制器。

图 4.5.4　16 **个时钟传送时序图(不使用** $\overline{\text{CS}}$ **，** MSB **在前**)

图 4.5.5　TLC2543 **和** GMS90C32 **接口**

4. 对 TLC2543 操作的汇编语言程序

例 4.5.1

```
        CLK   EQU P1.0
        DI    EQU P1.1
        DO    EQU P1.2
        NCS   EQU P1.3
        ORG 0030H
START:  MOV SP，#50H          ；初始化堆栈指针
        MOV P1，#04H          ；初始化 I/O 口
        CLR CLK
        SETB NCS
        MOV A，#0FFH
        ACALL TLC2543
        ACALL STORE
        JMP START
TLC2543: MOV R4，P3           ；读入通道选择和方式数据
        MOV A，R4
        CLR NCS               ；选通 TLC2543
        JB ACC.1，LSB
    ；处理低 8 位
MSB:    MOV R5，#08
LOOP1:  MOV C，DO             ；读入数据位
        RLC A                 ；循环左移一位
        MOV       DI，C       ；输出通道选择和方式数据位
        SETB CLK              ；置 I/O CLOCK 为高电平
        CLR CLK               ；置 I/O CLOCK 为低电平
        DJNZ R5，LOOP1        ；操作下一位
        MOV R2，A             ；存储转换结果的低字节
        MOV A，R4
        JB ACC.1，RETURN
LSB:    MOV R5，#08
LOOP2:  MOV C，DO             ；读入数据位
        RLC A                 ；循环左移一位
        MOV DI，C             ；输出通道选择和方式数据位
        SETB CLK              ；置 I/O CLOCK 为高电平
        CLR CLK               ；置 I/O CLOCK 为低电平
        DJNZ R5，LOOP2        ；操作下一位
        MOV R3，A             ；存储转换结果的低字节
        MOV A，R4             ；通道选择和方式数据位放入 A 寄存器中
```

```
            JB ACC.1，MSB          ；若 P3.1 为高，表示继续读取
RETURN: RET

STORE:      MOV A，R4              ；通道选择和方式数据位放入 A 寄存器中
            ANL A，#0F0H           ；留下通道号
            SWAP A
            MOV B，#02
            MUL AB
            ADD A，#030H
            MOV R1，A
            MOV A，R2
            MOV @R1，A             ；存储转换结果的高字节：
                                   ；通道 0 的存储地址为 30H
                                   ；通道 1 的存储地址为 32H
                                   ；…
            INCR1
            MOVA，R3
            MOV@R1，A              ；存储转换结果的低字节：
                                   ；通道 0 的存储地址为 31H
                                   ；通道 1 的存储地址为 33H
                                   ；…
            RET
            END
```

5. 对 TLC1543 操作的 C51 语言程序模块

TLC1543 是 TI 公司的 10 位串行模数转换器，除了位数的差别之外，其性能指标与 TLC2543 完全相同，管脚也兼容。它的详细资料请参考 TI 公司的产品数据手册。下面给出 51 单片机对其操作的 C51 语言模块。接口连线定义与前面介绍的第 3 点相同。

例 4.5.2

```
#define uchar unsigned char
#define uint unsigned int
#define CLOCK   P1^0
#define D_IN    P1^1
#define D_OUT   P1^2
#define _CS     P1^3
/*从 TLC1543 读取采样值，参数 port 是采样的通道号*/
uint read1543(uchar port)
{
uint data ad；
uint data i；
```

```
uchar data al=0，ah=0；
CLOCK=0；
_CS=0；
port<<=4；
for (i=0；i<4；i++)    /*输入通道号*/
{
      D_IN=(bit)(port&0x80)；CLOCK=1；CLOCK=0；
      port<<=1；
}

for (i=0；i<6；i++) /*填充 6 个 CLOCK*/
{
CLOCK=1；CLOCK=0；
}
_CS=1；_nop_( )；_nop_( )；_CS=0；  /*等待 AD 转换*/
for (i=0；i<2；i++) /*取 D9，D8*/
{
  D_OUT=1；
  CLOCK=1；
  ah<<=1；
  if (D_OUT) ah|=0x01；
  CLOCK=0；
}
for (i=0；i<8；i++) /*取 D7～D0*/
{
  D_OUT=1；
  CLOCK=1；
  al<<=1；
  if (D_OUT) al|=0x01；
  CLOCK=0；
}
_CS=1；
ad=(uint)ah；ad<<=8；ad|=al；  /*得到 AD 值*/
return (ad)；
}
```

4.5.3 MCM2814(E^2PROM)的例子

图 4.5.6 **为** 8031(MCU)**与** MCM2814(E^2PROM)**的硬件连接图，有关** MCM2814 **的详细情**

况可参考 Motorola 公司提供的芯片资料。图中，P1.0 模拟 MCU 的数据输出端(SPISI)，P1.1 模拟 SPI 的时钟输出端(SPICK)，P1.2 模拟 SPI 的从机选择端 $\overline{SS}$，P1.3 模拟 SPI 的数据输入端(SPISO)。下面介绍用 8031 汇编语言编写的模拟 SPI 串行输入、串行输出和串行输入/输出 3 个过程的子程序。这些子程序也适用于在串行时钟的上升沿输入和下降沿输出的各种串行外围接口芯片，如 8 位或 10 位 A/D 芯片、74LS 系列输出芯片等。对于下降沿输入、上升沿输出的各种串行外围接口芯片，只要改变 P1.1 的输出顺序，即先输出 0 再输入 1，再输出 0，则这些子程序也同样适用。

图 4.5.6　SPI 总线接口原理图

(1) MCU 串行输入子程序 SPIIN。

例 4.5.3　从 MCM2814 的 SPISO 线上接收一个十六进制数据“#1BH”，并放入寄存器 R0 中。

```
SPIIN：  SETB    P1.1          ；使 P1.1(时钟)输出为 1
         CLR     P1.2          ；选择从机
         MOV     R1，#08H      ；置循环次数
SPIN1：  CLR P1.1              ；使 P1.1(时钟)输出为 0
         NOP                   ；延时
         NOP
         MOV     C，P1.3       ；从机输出 SPISO 送进位 C
         RLC     A             ；左移至累加器 ACC
         SETB    P1.0          ；使 P1.0(时钟)输出为 1
         DJNZ    R1，SPIN1     ；判断是否循环 8 次
         MOV     R0，A         ；数据#1BH 送 R0
         RET                   ；返回
```

(2) MCU 串行输出子程序 SPIOUT。

例 4.5.4　将 8031 中 R0 寄存器的内容传送到 MCM2814 的 SPISI 线上。

```
SPIOUT： SETB    P1.1          ；使 P1.1(时钟)输出为 1
         CLR     P1.2          ；选择从机
         MOV     R1，#08H      ；置循环次数
         MOV     A，R0         ；R0 内容送累加器 ACC
SPIOT1： CLR     P1.1          ；使 P1.1(时钟)输出为 0
         NOP                   ；延时
```

```
        NOP
        RLC     A           ；左移至累加器 ACC，最高位至 C
        MOV     P1.0，C     ；进位 C 送从机输入 SPISI 线上
        SETB    P1.1        ；使 P1.1(时钟)输出为 1
        DJNZ    R1，SPIOT1  ；判断是否循环 8 次
        RET                 ；返回
```

(3) MCU 串行输入/输出子程序 SPIIO。

例 4.5.5 将 8031 中 R0 寄存器的内容传送到 MCM2814 的 SPISI 中，同时从 MCM2814 的 SPISO 接收数据#1BH。

```
SPIIO：  SETB    P1.1        ；使 P1.1(时钟)输出为 1
         CLR     P1.2        ；选择从机
         MOV     R1，#08H    ；置循环次数
         MOV     A，R0       ；R0 内容送累加器 ACC
SPIO1：  CLR     P1.1        ；使 P1.1(时钟)输出为 0
         NOP                 ；延时
         NOP
         MOV     C，P1.3     ；从机输出 SPISO 送进位 C
         RLC     A           ；左移至累加器 ACC，最高位至 C
         MOV     P1.0，C     ；进位 C 送从机输入
         SETB    P1.1        ；使 P1.1(时钟)输出为 1
         DJNZ    R1，SPIO1   ；判断是否循环 8 次
         RET                 ；返回
```

4.5.4 X25045 应用实例

X25045 是一种集电源监控、看门狗和高速非易失性存储器三功能于一块芯片的多功能器件。可选择看门狗定时周期，有 512*8 位的 RAM。X25045 的管脚排列、管脚功能及与 89C51 的连线分别见图 4.5.7、表 4.5.2 和图 4.5.8。详细资料参见 Xicro 公司提供的产品资料。此处仅给出其读写汇编程序。

表 4.5.2 X25045 管脚定义

管脚名称	功能
$\overline{CS}$	片选端(低有效)
SO	串行数据输出端
SI	串行数据输入端
SCK	串行时钟信号端
WP	写保护
V_{SS}	地
V_{CC}	电源
RESET	复位输出

图 4.5.7 X25045 管脚图

图 4.5.8　8031 **单片机与** X25045 **连线图**

例 4.5.6

X25045 **的读写程序如下：**

```
        XCS     EQU P1.0
        XSI     EQU P1.1
        XSCK    EQU P1.2
        XSO     EQU P1.3

        ORG     0000H
        LJMP    START
        ORG     0030H
START:
        MOV     P1，#0FFH
        CLR     XSCK
        CLR     XSI
        CLR     XSO
        CLR     XCS
        LCALL   XWE
        LCALL   XWC
        LCALL   XRC
```

```
        LCALL   XWE
        LCALL   XWD
        LCALL   XRD
        LJMP    START

XWD:                            ;子程序：写入一个字节数据
        CLR     XSCK
        CLR     XCS
        MOV     A，#2           ;“写数据”命令
        LCALL   XSSI
        MOV     A，#0
        LCALL   XSSI            ;输入数据地址
        MOV     A，#0FH
        LCALL   XSSI            ;写数据
        SETB    XCS
        RET

XRD:                            ;子程序：读出一个字节数据
        CLR     XSCK
        CLR     XCS
        MOV     A，#3           ;读数据命令
        LCALL   XSSI
        MOV     A，#0
        LCALL   XSSI            ;地址
        CLR     A
        LCALL   XSSO            ;读出一个字节数据
        MOV     71H，A          ;存入 71H 地址中
        RET

XWC:                            ;子程序：写入控制命令
        CLR     XCS
        MOV     A，#1
        LCALL   XSSI
        MOV     A，#20H         ;NOBL0，1 看门狗定时：1.4 s
        LCALL   XSSI
        SETB    XCS
        RET

XRC:                            ;子程序：读出状态字
```

```
        CLR     XCS
        MOV     A，#05H
        LCALL   XSSI
        LCALL   XSSO
        SETB    XCS
        MOV     70H，A
        RET

XWE:                                    ；子程序：写使能
        CLR     XCS
        MOV     A，#06H
        LCALL   XSSI
        SETB    XCS
        RET

XSSI:                                   ；子程序：写入一个字节
        MOV     R7，#8
LP1:
        CLR     XSCK
        RLC     A
        MOV     XSI，C
        SETB    XSCK
        NOP
        DJNZ    R7，LP1
        RET

XSSO:                                   ；子程序：读出一个字节
        MOV     R7，#8
        CLR     A
LP2:
        RLC     A
        CLR     XSCK
        NOP
        MOV     C，XSO
        RLC     A
        SETB    XSCK
        DJNZ    R7，LP2
        CLR     XSCK
        RET
```

4.6 单总线及应用实例

4.6.1 单总线技术简介

单总线技术是美国达拉斯半导体公司近年推出的新技术。它将地址线、数据线、控制线、电源线合为一根信号线，允许在这根信号线上挂数百个测控对象，这些测控对象所用芯片是由该公司提供的。每个芯片均有一个 64 位的 ROM，也称之为身份证号，确保挂在单总线上后，可以被唯一地区分识别出来，这是定位和寻址器件实现单总线测控功能的前提条件。ROM 中含有 CRC 检验码，能确保数据交换可靠。芯片内还含有收、发控制和电源存储电路，一般不用另附电源。这些芯片在检测点就把模拟信号数字化，单总线上传送的是数字信号，使系统的抗干扰性能好、可靠性高。

单总线系统是由一个总线命令者和一个或多个从者组成的计算机应用系统。系统按单总线协议规定的时序和信号波形进行初始化、识别器件和交换数据。

下面从硬件配置、总线协议和总线信号三个方面介绍单总线技术。

1. 硬件配置

单总线系统只定义了一根信号线。总线上的每个器件都能够在合适的时间驱动它，相当于把地址线、数据线、控制线合为一根线对外进行数据交换。为了区分这些芯片，厂家在生产每个芯片时，都编制了唯一的序列号，通过寻址就能把芯片识别出来。这样做，能使这些器件挂在一根信号线上进行码分多址、串行分时数据交换，组成一个自动测控系统，甚至还可以组成一个微型局域网。厂家对每个芯片用激光刻录了一个 64 位二进制 ROM 代码。从最低位开始，前 8 位是族码，表示产品的分类编号；接着的 48 位是一个序列号，这个 15 位的十进制编码完全可为每个芯片编制一个全世界唯一的号码；最后 8 位是前 56 位的 CRC 校验码。CRC(Cyclic Redundancy Check)称为循环冗余码检测，是数据通信中校验数据传输是否正确的一种常用方法。在使用时，总线命令者读入 ROM 中 64 位二进制码后，将前 56 位按 CRC 多项式(这里是 $X^8+X^5+X^4+1$)计算出 CRC 值，然后与 ROM 中高 8 位的 CRC 值比较，若相同，则表明数据传送正确；否则，要求重传。

芯片内部的收、发控制和电源存储电路如图 4.6.1 所示。

图 4.6.1 单总线芯片入口的示意图

由于这些芯片采用 CMOS 技术，耗电量都很小(空闲时几μW，工作时几 mW)，从单总线上吸收一点电流储存在芯片内的电容中就可正常工作，故一般不用另附电源。单总线上通常处于高电位(5 V 左右)，每个器件都能在需要时驱动它。所以，为了避免在不工作时给总线增加功耗，挂在总线上的每个器件必须是漏极开路或者是三态输出的。当单总线上所挂器件超过 8 个时，就需要注意器件的总线驱动问题，这一点在进行系统设计时要加以注意。

连接单总线的总线电缆是有长度限制的。当采用普通信号电缆传输长度超过 50 m 时，读取的测温数据将可能发生错误。这种情况主要是由总线分布电容使信号波形产生畸变造成的。将总线电缆改为双绞线带屏蔽电缆时，正常通信距离可达 150 m，当采用每米绞合次数更多的双绞线带屏蔽电缆时，正常通信距离进一步加长。

单总线的数据传输有两种模式：通常情况下，以 16.3 kb/s 的速率通信，超速模式可达 142 kb/s。因此，单总线数据传输只能用于对速度要求不高的场合，一般用于 100 kb/s 以下速率的测控或数据交换系统中。

2. 总线协议

总线协议是软件设计的任务。在单总线系统中，软件设计是技术的关键。简洁的硬件配置是靠复杂的软件来支撑的。在 PC 机作主控机时，单总线软件设计基于 Dallas 公司授权的软件开发商提供的成套开发工具，为软件开发应用带来很大的便利。而用单片机作主控机时，需依据单总线协议，用汇编语言或高级语言来编写全部软件，给开发应用增加了一定的难度。

总线协议保证了数据可靠的传输，任一时刻单总线上只能有一个控制信号或数据。一次数据传输可分为以下四个操作过程:①初始化；②传送 ROM 命令；③传送 RAM 命令；④数据交换。

单总线上所有的处理都从初始化开始。初始化时序是由一个复位脉冲(总线命令者发出)和一个或多个从者发出的应答脉冲(总线从者发出)组成。应答脉冲的作用是：从器件让总线命令者知道该器件是在总线上的，并准备好开始工作。应答脉冲的信号波形如图 4.6.2 所示。

图 4.6.2　单总线的时序信号波形

当总线命令者检测到某器件的存在时，首先发送 7 个 ROM 功能命令中的一个命令：①读 ROM(总线上只有一个器件时，即读出其序列号)；②匹配 ROM(总线上有多个器件时，寻址某个器件)；③查找 ROM(系统首次启动后，须识别总线上各器件)；④跳过 ROM(总线上只有一个器件时，可跳过读 ROM 命令直接向器件发送命令，以节省时间)；⑤超速匹配

ROM(超速模式下寻址某个器件)；⑥超速跳过 ROM(超速模式下跳过读 ROM 命令)；⑦条件查找 ROM(只查找输入电压超过设置的报警门限值的器件)。这些操作在器件数据手册中都有具体的命令码供编程使用。当成功执行上述命令之一后，总线命令者可发送任何一个可使用的命令来访问存储和控制功能，进行数据交换。所有数据的读写都是从最低位开始的。

3. 总线信号

单总线传送的数据或命令是由一系列的时序信号组成的，单总线上共有 4 种时序信号: ①初始化信号；②写 0 信号；③写 1 信号；④读信号。图 4.6.2 给出了常规模式下这 4 种波形的示意图。各器件的数据手册对这 4 种波形参数(如脉冲上升时间、宽度和间隙等)都作了具体的要求，设计中应保证指令执行时间小于或等于时序信号中的最小时间。这部分软件必须用单片机的汇编语言进行编程，以确保严格的时间关系。需要注意的是，当单片机工作频率不同时，同一段延时程序在单总线上产生的时延值是不同的。

4.6.2 结合 1820 介绍具体编程

1. DS1820 简介

DS1820 是美国 DALLAS 半导体公司生产的可组网数字式温度传感器，在其内部使用了在板(ON-BOARD)专利技术。全部传感元件及转换电路集成在形如一只三极管的集成电路内，封装形式如图 4.6.3 所示。与其它温度传感器相比，DS1820 具有以下特性：

(1) 独特的单线接口方式，DS1820 在与微处理器连接时仅需要一条口线即可实现微处理器与 DS1820 的双向通信;

(2) DS1820 支持多点组网功能，多个 DS1820 可以并联在唯一的总线上，实现多点测温;

(3) DS1820 在使用中不需要任何外围元件；

(4) 温度范围–55～+125 ℃，固有测温分辨率为 0.5 ℃；

(5) 测量结果以 9 位数字量方式进行串行传送。

2. DS1820 内部结构及测温原理

DS1820 内部结构框图如图 4.6.4 所示。

图 4.6.3 DS1820 封装形式　　图 4.6.4 DS1820 内部结构框图

DS1820 测温原理如图 4.6.5 所示。图中低温度系数晶振的振荡频率受温度影响很小，用于产生固定频率的脉冲信号，并将其送给计数器 1。高温度系数晶振随温度变化，其振荡

率明显改变，所产生的信号作为计数器 2 的脉冲输入。计数器 1 和温度寄存器被预置在–55℃所对应的一个基数值。计数器 1 对低温度系数晶振产生的脉冲信号进行减法计数，当计数器 1 的预置值减到 0 时，温度寄存器的值将加 1，计数器 1 的预置将重新被装入，计数器 1 重新开始对低温度系数晶振产生的脉冲信号进行计数，如此循环，直到计数器 2 计数到 0 时，停止温度寄存器值的累加，此时温度寄存器中的数值即为所测温度。图 4.6.5 中的斜率累加器用于补偿和修正测温过程中的非线性，其输出用于修正计数器 1 的预置值。

图 4.6.5　DS1820 测温原理

在正常测温情况下，DS1820 的测温分辨率为 0.5 ℃，以 9 位数据格式表示，其中最低有效位(LSB)由比较器进行 0.25℃比较，当计数器 1 中的余值转化成低于 0.25 ℃的温度后，清除温度寄存器的最低位(LSB)；当计数器 1 中的余值转化成高于 0.25 ℃的温度后，置位温度寄存器的最低位(LSB)。例如–25.5 ℃对应的 9 位数据格式如图 4.6.6 所示。表 4.6.1 列出了 DS1820 温度数字的部分对应关系。

图 4.6.6　–25.5 ℃对应的 9 位数据格式

表 4.6.1　DS1820 温度数字对应关系

温度/℃	输出的二进制码	对应的十六进制码
+125	0000000011111010	00FAH
+25	0000000000110010	0032H
+1/2	0000000000000001	0001H
0	0000000000000000	0000H
–1/2	1111111111111111	FFFFH
–25	1111111111001110	FFCEH
–55	1111111110010010	FF92H

3. DS1820 的操作命令

DS1820 的 64 位 ROM 的结构如图 4.6.7 所示。图中，开始 8 位是 DS1820 的产品类型编号 10H，接着是每个器件的唯一的序号，共有 48 位，最后 8 位是前 56 位的 CRC 校验码，这也是多个 DS1820 可以采用一根线进行通信的原因。主机操作 ROM 的命令有五种，如表 4.6.2 所列。

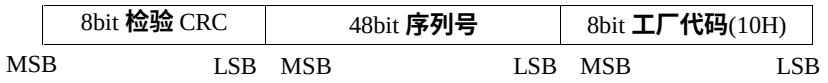


图 4.6.7 DS1820 的 64 位 ROM 结构

表 4.6.2 DS1820 的 ROM 命令

指 令	说 明
读 ROM(33H)	读 DS1820 的序列号
匹配 ROM(55H)	继续读完 64 位序列号的命令，用于多个 DS1820 时定位
跳过 ROM(CCH)	此命令执行后的存储器操作将针对在线的所有 DS1820
搜 ROM(F0H)	识别总线上各器件的编码，为操作各器件作好准备
报警搜索(ECH)	仅温度越限的器件对此命令作出响应

DS1820 的高速暂存器由便笺式 RAM 和非易失性电擦写 EERAM 组成，后者用于存储 TH、TL 值。数据先写入便笺式 RAM，经校验后再传给 EERAM。便笺式 RAM 占 9 个字节，包括温度信息(第 1、2 字节)、TH 和 TL 值(3、4 字节)、计数寄存器(7、8 字节)、CRC(第 9 字节)等，第 5、6 字节不用。暂存器的命令共 6 条，见表 4.6.3 所列。

表 4.6.3 DS1820 存储控制命令

指 令	说 明
温度转换(44H)	启动在线 DS1820 做温度 A/D 转换
读数据(BEH)	从高速暂存器读 9 位温度值和 CRC 值
写数据(4EH)	将数据写入高速暂存器的第 3 和第 4 字节中
复制(48H)	将高速暂存器中第 3 和第 4 字节复制到 EERAM
读 EERAM(B8H)	将 EERAM 内容写入高速暂存器中第 3 和第 4 字节
读电源供电方式(B4H)	了解 DS1820 的供电方式

在正常测温情况下，DS1820 的测温分辨力为 0.5℃，可采用下述方法获得高分辨率的温度测量结果：首先，用 DS1820 提供的读暂存器指令(BEH)读出以 0.5℃为分辨率的温度测量结果；其次，切去测量结果中的最低有效位(LSB)，得到所测实际温度的整数部分 Tz；然后，再用 BEH 指令取计数器 1 的计数剩余值 Cs 和每度计数值 CD。考虑到 DS1820 测量温度的整数部分以 0.25℃、0.75℃为进位界限的关系，实际温度 Ts 可用下式计算：

$$Ts=(Tz-0.25℃)+(CD-Cs)/CD$$

DS1820 完成温度转换后，就把测得的温度值与 TH、TL 作比较。若 T>TH 或 T<TL，则将该器件内的告警标志置位，并对主机发出的告警搜索命令作出响应。因此，可用多只 DS1820 同时测量温度，主机进行告警搜索。一旦某测温点越限，主机利用告警搜索命令即可识别正在告警的器件，并读出其序号，而不必考虑非告警器件。

4. 51 单片机操作 DS1820 的参考程序

根据单总线的时序信号波形，用汇编语言写出接口程序，C51 部分将其作为外部函数调用即可。本程序要求 8051 的 P1.0 与 DS1820 连接，工作频率为 12 MHz。

例 4.6.1

```
#include <reg51.h>
#include <stdio.h>
#include <math.h>
#define uchar unsigned char
#define uint unsigned int
extern WDS1820(uchar x)；        /* 外部函数声明：写 DS1820 命令*/
extern RDS1820(uchar *pt)；      /* 外部函数声明：读 DS1820 数据 */
extern bit RTDS1820(void)；      /* 外部函数声明：复位 DS1820 */
extern Delay15(uchar n)；        /* 外部函数声明：延时 15 μs */
sbit P1_0=P1^0；                 /*定义 P1.0 为 I/O 口 */

void main (void)                 /* 主函数 */
{
  uint i；
  float data tempF；
  uchar data temp[10]，disbuf[10]；  /* 存放温度数据和显示数据的局部数组变量声明 */
  if(RTDS1820()!=1) error(0x3)；     /* 复位并判 DS1820 是否存在 */
  Delay15(0xff)；                    /* 延时约 15×255 μs */
  WDS1820(0xcc)；                    /* 向 DS1820 发跳读 ROM 命令 */
  WDS1820(0x44)；                    /* 向 DS1820 发启动温度变换命令 */
  P1_0=1；                           /* P1.0 口置线高电平 */
  do{ Delay15(0xff)；i++；}while(i<=400)；    /* 延时约 1.5 s */
  if(RTDS1820()!=1) error(0x3)；             /* 复位并判 DS1820 是否存在 */
  Delay15(0xff)；                            /* 延时约 15×255 μs */
  WDS1820(0xcc)；                            /* 向 DS1820 发跳读 ROM 命令 */
  WDS1820(0xbe)；                            /* 向 DS1820 发读 9 字节数据命令 */
  RDS1820(&temp)；                           /* 9 字节数据读入数组 temp */
tempF=(((temp[1]<<8)+temp[0])>>1)-0.25+((temp[7]-temp[6])/temp[7])；    /* 温度值修正计算 */
  sprintf(&disbuf，"T=%+4.1f%c"，tempF，'C')； /* 按 T=±XXX.XC 格式组织数据送 disbuf */
  …… /* 插入程序其它部分 */
}
```

以下是用汇编语言编写的 DS1820 接口源程序清单，共有 4 个子程序，其中 RTDS 1820 无参数传递，但具有 bit 对象的返回值，DELAY15 和 WDS1820 带有一个经 R7 传递的无符号 char 类参数，DS1820 带有一个经 R7 传递的 1 字节指针类参数。

例 4.6.2

```
        NAME    RW1820  ；定义模块名
?PR?RDS1820?RW1820      SEGMENT CODE    ；RDS1820 子程序代码段声明
?PR?WDS1820?RW1820      SEGMENT CODE    ；WDS1820 子程序代码段声明
```

```
?PR?RTDS1820?RW1820      SEGMENT CODE    ；RTDS1820 子程序代码段声明
?PR?DELAY15?RW1820       SEGMENT CODE    ；DELAY15 子程序代码段声明

PUBLIC
_RTDS1820，_WDS1820，_RDS1820，_DELAY15
  ；公开函数名以便 C 模块可调用它们

        RSEG      ?PR?RDS1820?RW1820_RDS1820:
  ；RDS1820 代码段起始，完成 9 字节温度数据的读取
    MOV   R1，#9                          ；置 9 字节数据计数器初值
    MOV   A，R7                           ；取经 R7 传递的数组 temp 首址(C 中定义)
    MOV   R0，A
RD18201:  MOV   R2，#8                    ；置 1 字节移位计数器初值
RD18202:  SETB   P1.0                     ；P1.0 置为高电平
          NOP
          NOP
          CLR P1.0                        ；P1.0 置为低电平
          NOP
          NOP
          SETB P1.0                       ；P1.0 置为高电平，准备输入数据
          MOV R7，#1                      ；延时 15 μs
           LCALL DELAY15
                MOV C，P1.0               ；P1.0 状态读入位累加器
                RRC A                     ；累加器 A 右移
          DJNZ R2，RD18202                ；判断一个字节是否读完
            MOV @R0，A                    ；保存结果
    INC R0                                ；地址指针加 1
    DJNZ R1，RD18201                      ；判断 9 字节是否读完
       RET                                ；返回
  RSEG      ?PR?WDS1820?RW1820_WDS1820:
  ；WDS1820 代码段起始，完成 1 字节命令的写入
                MOV R1，#8                ；置 1 字节移位计数器初值
                CLR C                     ；清位累加器
                MOV A，R7                 ；取经 R7 传递的命令参数
WR18201:  CLR P1.0                        ；P1.0 置为低电平
                MOV R7，#1                ；延时 15 μs
    LCALL DELAY15
          RRC A                           ；累加器 A 右移 1 位
          MOV P1.0，C                     ；发送 1 位数据给 DS1820
```

```
        MOV R7，#1                    ；延时 15 μs
        LCALL DELAY15
        SETB P1.0                     ；P1.0 置为高电平
        NOP
          DJNZ R1，WR18201            ；判断 1 字节数据是否发送完毕
        SETB P1.0                     ；P1.0 置为高电平
       RET                            ；返回

        RSEG     ?PR?RTDS1820?RW1820
；RTDS1820 代码段起始，判 DS1820 是否存在
RTDS1820:
              CLR P1.0                ；P1.0 置为低电平
        MOV R7，#40                   ；延时约 600 μs
              LCALL DELAY15
              SETB P1.0               ；P1.0 置为高电平
        MOV R7，#4                    ；延时约 60 μs
        LCALL DELAY15
              MOV R7，#100            ；置循环读初值
              SETB C                  ；位累加器置 1
RST0:      JNB P1.0，RST1             ；P1.0 为低表明 DS1820 存在并返回
        DJNZ R7，RST0                 ；判断循环读 100 次结束否
        CLR C                         ；无 DS1820 存在脉冲，位累加器清零
RST1:RET                              ；DS1820 存在标志经位累加器返回

RSEG    ?PR?DELAY15?RW1820_DELAY15:
；DELAY15 代码段起始，延时 15 μs 功能
DELAY15:   MOV R6，#6                 ；2 个时钟周期
DEL151:    DJNZ R6，DEL151            ；2 个时钟周期*R6
             DJNZ R7，DELAY15         ；2 个时钟周期
       RET                            ；1 个时钟周期
       END
```

4.6.3　其它单总线器件简介及系统应用示例

单总线专用芯片的种类和型号很多，可以参阅达拉斯(DALLAS，现为美信 MAXIM 公司)公司的数据手册，根据需要选用，也可从互联网上访问。这里简单介绍单总线测控系统中用到的一些器件芯片。

1. A/D 转换器

DS2450 是达拉斯公司 1999 年推出的具有单总线接口的 A/D 转换器，封装见图 4.6.8。

其主要特性为:

(1) 4 路模拟输入通道，两种模拟输入量程为 0～2.56 V 和 0～5.12 V；

(2) 未用作输入的通道可作为输出通道使用；

(3) 一个数据口可以 16.3 kb/s 的速率通信，超速模式可达 142 kb/s；

(4) 利用逐次逼近的变换原理，可选择 8 位转换精度；

(5) 不用另接电源和外围电路；

(6) 8 脚 SOIC 封装。

图 4.6.8 DS2450 封装图

这样，温度湿度的检测也可用 A/D 转换器 DS2450 和模拟式温度湿度传感器串接来实现。同理，其它类型的传感器，如防火、防盗等传感器，其输出电压也可通过 DS1820 进行判定，从而实现报警功能。

2. 可寻址控制开关

在测控系统中，开关量控制是应用得最多的。对计算机来讲，只要送出一位 0 或 1 控制码信号，就可用它去触发被控电路。通常是先触发光电耦合器，然后启动继电器、晶闸管或固态继电器，应视被控设备功率大小选用合适的开关器件。

达拉斯公司提供了一些可寻址的控制开关，如 DS2405、DS2406、DS2409 等。

图 4.6.9 DS2405 原理示意图

DS2405 原理示意图见图 4.6.9，主要特点为:

(1) 单总线上的命令决定漏极开路的输出逻辑；

(2) 电平作为开关控制信号；

(3) PIO 引脚吸收电流能力大于 4 mA(0.4 V)；

(4) 不用外接电源。

3. 硅序列号 DS2401

DS2401 芯片实际上是块符合单总线协议的 ROM 硅片，厂家在其中写入了唯一的序列号，用作寻址定位的标识。DS2401 广泛用于位置判定、身份识别和巡检监督等场合。

4. 串口 RS232 到单总线适配器

当采用 PC 机进行低速测控时，用串行口 RS－232 连接是很方便的，但它不能同时挂上多个测控对象。为此，要采用 RS－232 到单总线的适配器，实现与 PC 机方便的连接。达拉斯公司提供的 DS9097 等型号的适配器，可完成多线对一线的转换及电平的变换。

5. 防静电保护二极管

在单总线线路的末端，为防止开路状态时静电等的干扰侵入，通常都接上 DS9502 之类的防静电(可高达 27 kV)保护二极管。

6. 单总线系统应用

用单片机实现单总线应用，硬件连接简单，单机 I/O 口 P1、P2、P3 中的任一位端口都可以与总线进行双向数据传输。用单片机对单总线系统进行现场长期监控是比较经济实惠的方案，而且还可通过 RS－232(或 485)串行口与上位机 PC 连接，在 Windows 平台上进行更高一级的软件管理。单总线技术可广泛应用到社会各领域，这里只列举了环境状态监控的应用情况，其方法也完全可以用于其它领域。

采用单总线技术设计环境状态监控系统，只需要一条双绞线(一根为信号线，一根为地线)从单片机拉向监控现场，然后将各种监控对象挂在其上就可以了，如图 4.6.10 所示。图中只画出了一个监控现场的配置，布线接头可用通常电话线路的接头，插入和拔出都很方便。

图 4.6.10　单总线测控系统示意图

4.7　USB 通用串行总线及应用

4.7.1　USB 总线概述

连接计算机外设的串行数据总线(简称串行总线)的发展一直非常缓慢。1969 年美国电子工业协会推出的 RS－232(后改为 RS－232C)串行总线，至今仍是连接计算机外设的主流串行总线。尽管在 20 世纪 70 年代和 80 年代，陆续推出过 RS－422A/423A、RS－449、RS－485 和 RS－530 等串行总线，其中，RS－449 的设计初衷就是想取代 RS－232，RS－530 则是想取代 RS－449，但由于种种原因，都没有改变 RS－232 先入为主的主导地位。RS－232C 的最高传输速率只有 20 kb/s，RS－530 虽高一些，也只有 2 Mb/s，与 20 世纪 70 年代初 HP 公司推出的通用接口总线 GPIB(即 IEEE－488)这种并行总线相比，RS－530 的传输速率只有 GPIB 的 1/4，更无法与性能先进的并行总线相比。因此，长期以来，串行总线只用于连接慢速外设，或用作低速网络的总线。

USB(Universal Serial Bus)是 1995 年 Microsoft、Compaq、IBM 等公司联合制定的一种

新的 PC 串行通信协议。USB 协议出台后，得到各 PC 厂商、芯片制造商和 PC 外设厂商的广泛支持。从当初的 0.7、0.8 版本到现在广泛采用的 1.0、1.1 版本，甚至到已正式公布的的 2.0 版本，USB 本身也处于不断的发展和完善中。USB 外设在国外以惊人的速度发展，迄今为止，各种 USB 的外设已经有上千种。

USB 1.0 通用串行总线规范是由 Intel、Compaq、Digital、IBM、Microsoft、NEC、Nor-thern-Telecom 这些著名的计算机公司和通信公司于 1996 年 1 月提出的。通用串行总线是一种将 USB 外围设备连接到主机的外部总线结构，它是通过 PCI 总线和 PC 的内部系统数据线连接，实现数据的传送。USB 同时又是一种通信协议，它支持主系统(host)和 USB 的外围设备(device)之间的数据传送，通过一个 4 针的标准插头，采用菊花链形式把所有的外设连接起来。

USB 的优点有以下几条：

(1) USB 为所有的 USB 外设提供了单一的、易于操作的标准连接类型，排除了外部设备对系统资源的需求，因而减少了硬件的复杂性和对端口的占用，整个 USB 系统只有一个端口和一个中断，节省了系统资源。

(2) USB 支持热插拔(hot plug)和即插即用(Plug-and-Play)，也就是说，在不关闭计算机的情况下可以安全地插上和断开 USB 设备，动态地加载驱动程序。

(3) USB 在设备供电方面有很大的灵活性。USB 直接连接到 Hub 或者是连接到 host 的设备可以通过 USB 电缆供电，也可以通过电池或者其它的电力设备来供电，还可使用两种供电方式的组合，并且支持挂起和唤醒的节能模式。

(4) USBv1.1 规范提供全速 12 Mb/s 和低速 1.5 Mb/s 的模式，USB2.0 规范提供高达 480 Mb/s 的数据传输速率，可以适应各种不同类型的外设。

(5) 为了适应各种不同类型外围设备的要求，USB 提供了四种不同的数据传送类型。

(6) USB 使得多个外围设备可以跟主机通信，最多可支持 127 个设备，但需要使用 USB Hub 增加分支。

4.7.2 USB 总线的硬件结构

USB 采用四线电缆，其中两根是用来传送数据的串行通道，另两根为下游(Downstream)设备提供电源，如图 4.7.1 所示。

图 4.7.1 USB 连接线

D+、D–是一对差模信号线，它支持两种数据传输率，对于高速需要高带宽的外设，USB 以全速 12 Mb/s 传输数据，但必须使用屏蔽的双绞线，且长度不超过 5 m。对于低速外设，USB 则以 1.5 Mb/s 的传输速率传输数据，这种模式下可以使用无屏蔽的非双绞线，但长度

不超过 3 m。为保证能提供一定电平的信号并且与终端的负载匹配，在电缆的每一端都使用不平衡终端负载。这种终端负载也保证能检测出外设与端口的连接和分离，且能区分高速与低速 USB 总线，可根据外设情况在两种传输模式中自动动态转换。V_{BUS} 通常为+5 V 的电源，GND 是地线。

USB 是基于令牌的总线，类似于令牌环网络。USB 主控制器广播令牌，总线上的设备检测令牌中的地址是否与自身相符，通过接收或发送数据给主机来响应。USB 通过支持悬挂/恢复操作来管理 USB 总线电源。USB 系统采用级联星型拓扑，该拓扑由三个基本部分组成：主机(Host)、集线器(Hub)和功能设备，如图 4.7.2 所示。

图 4.7.2　USB 系统级联结构

主机(Host)，也称为根、根结或 Root Hub，它做在主板上或作为适配卡安装在计算机上。主机包含有主控制器和根集线器(Root Hub)，控制着 USB 总线上的数据和控制信息的流动，每个 USB 系统只能有一个根集线器，它连接在主控制器上。

集线器是 USB 结构中的特定成分，它提供叫作端口(Port)的点将设备连接到 USB 总线上，同时检测连接在总线上的设备，并为这些设备提供电源管理，负责总线的故障检测和恢复。集线器可为总线提供能源，亦可为自身提供能源(从外部得到电源)。自身提供能源的设备可插入总线提供能源的集线器中，但总线提供能源的设备不能插入自身提供能源的集线器或支持超过四个的下游端口中。如果总线提供能源的设备需要超过 100 mA 电源时，不能同总线提供电源的集线器连接。

4.7.3　USB 总线的软件结构

每个 USB 有且只有一个主机，它包括 USB 总线接口、USB 设备层和功能层三层结构，如图 4.7.3 所示。

图 4.7.3 USB 的软件结构

1．USB 总线接口

USB 总线接口处理电气层与协议层的互连。从互连的角度来看，相似的总线接口由设备和主机同时给出，例如串行接口机(SIE)。USB 总线接口由主控制器实现。

2．USB 系统

USB 系统用主控制器管理主机与 USB 设备间的数据传输。它与主控制器间的接口依赖于主控制器的硬件定义。同时，USB 系统也负责管理 USB 资源，例如带宽和总线能量，这使客户访问 USB 成为可能。USB 系统还有三个基本组件。

(1) 主控制器驱动程序(HCD)。该程序可把不同主控制器设备映射到 USB 系统中。HCD 与 USB 之间的接口叫 HCDI，特定的 HCDI 由支持不同主控制器的操作系统定义。通用主控制器驱动器(UHCD)处于软结构的最底层，由它来管理和控制主控制器。UHCD 实现了与 USB 主控制器通信和控制 USB 主控制器，并且它对系统软件的其它部分是隐蔽的。系统软件中的最高层通过 UHCD 的软件接口与主控制器通信。

(2) USB 驱动程序(USBD)。该程序在 UHCD 驱动器之上，它提供驱动器级的接口，满足现有设备驱动器设计的要求。USBD 以 I/O 请求包(IRPs)的形式提供数据传输架构，由通过特定管道(Pipe)传输数据的需求组成。此外，USBD 使客户端出现设备的一个抽象，以便于抽象和管理。作为抽象的一部分，USBD 拥有缺省的管道。通过它可以访问所有的 USB 设备以进行标准的 USB 控制。该缺省管道描述了一条 USBD 和 USB 设备间通信的逻辑通道。

(3) 主机软件。在某些操作系统中，没有提供 USB 系统软件。这些软件本来是用于向设备驱动程序提供配置信息和装载结构的。在这些操作系统中，设备驱动程序将应用操作系统提供的接口而不是直接访问 USBDI(USB 驱动程序接口)结构。

3．USB 客户软件

USB 客户软件位于软件结构的最高层，负责处理特定 USB 设备驱动器。客户程序层描述所有直接作用于设备的软件入口。当设备被系统检测到后，这些客户程序将直接作用于外围硬件。这个共享的特性将 USB 系统软件置于客户和它的设备之间，客户程序要根据 USBD 在客户端形成的设备映像对它进行处理。

主机各层有以下功能：

(1) 检测连接和移去的 USB 设备；

(2) 管理主机和 USB 设备间的数据流；

(3) 连接 USB 状态和活动统计；

(4) 控制主控制器和 USB 设备间的电气接口，包括限量能量供应。

HCD 提供了主控制器的抽象和通过 USB 传输的数据的主控制器视角的一个抽象。USBD 提供了 USB 设备的抽象和 USBD 客户与 USB 功能间数据传输的一个抽象。USB 系统促进客户和功能间的数据传输，并作为 USB 设备的规范接口的一个控制点。USB 系统提供缓冲区管理能力并允许数据传输同步于客户和功能的需求。

4.7.4　USB 总线的数据传输方式

USB 的数据传输方式有四种：控制(control)、同步(isochronous)、中断(interrupt)和大量(bulk)。通常所有传送方式的主动权都在 PC，也就是 Host。

控制传送是双向传送，数据量通常较小。USB 系统软件主要用来进行查询、配置和给 USB 设备发送通用的命令。控制传送方式可以包括 8、16、32 和 64 字节的数据，这依赖于设备和传输速度。控制传输典型应用在主计算机和 USB 外设之间的端点(Endpoint)之间的传输，但是指定控制传输可能用到其它的端点。

同步传输提供了确定的带宽和间隔时间(latency)。它主要用于时间要求严格并具有较强容错性的流数据传输，或者用于要求恒定数据传送率的即时应用中。例如，即时通话的网络电话使用同步传输模式，就是很好的选择。同步数据要求确定的带宽值和确定的最大传送次数。对于同步传送来说，即时的数据传递比数据的完整性更重要一些。

中断方式传输主要用于定时查询设备是否有中断数据要传送。设备的端点模式器的结构决定了它的查询频率范围为 1～255 ms。这种传输方式典型地应用在少量的、分散的、不可预测数据的传输，键盘、操纵杆和鼠标就属于这一类型。中断方式传送是单向的，并且对于 Host 来说只有输入的方式。

大量方式传送主要应用在大量传送和接收数据，同时又没有带宽和间隔时间要求的情况下。打印机和扫描仪属于这种类型。这种传输类型可以等到所有其它类型的数据传送完成之后再传送和接收数据。

USB 将其有效的带宽分成各个不同的帧(frame)，每帧通常是 1 ms 时间长。每个设备每帧只能传送一个同步的传送包。在完成了系统的配置信息和连接之后，USB 的 Host 就会对不同的传送点和传送方式作一个统筹安排，用来适应整个的 USB 的带宽。通常情况下，同步方式和中断方式的传送会占据整个带宽的 90%，剩下的就安排给控制方式来传送数据。

4.7.5　USB 接口器件及应用

在单片机系统中，常用的 USB 接口方法有两种：一是采用专用的 USB 接口器件；一种是选用内部集成 USB 接口的单片机。下面就举两个例子分别介绍。

1．USB 专用接口芯片 USBN9602

对于 USB 通信接口，较流行的专用芯片是 National Semiconductor 公司的 USBN9602。

USBN9602 是标准双列直插式 28 引脚芯片。芯片内部总共带有 7 个传送/接收 FIFO 缓冲器：1 个双向传送和接收 FIFO 缓冲器，3 个单向传送 FIFO 缓冲器，3 个单向接收 FIFO 缓冲器。芯片引脚结构如图 4.7.4，其主要管脚的功能列于表 4.7.1。

表 4.7.1 USBN9602 管脚功能

名称	引脚号	功能描述
$\overline{WR}$ /SK	3	写信号/串行时钟
INTR	4	中断输出
DRQ	5	DMA 方式请求
$\overline{DACK}$	6	DMA 方式应答
A0/ALE/SI	7	地址线/地址锁存/串行输入
D0/SO	8	数据线或地址线/串行输出
D1～D7	9～15	数据线或地址线
V3.3	18	3.3 V 电压
D+	19	USB D+
D–	20	USB D–
MODE1	24	方式选择
MODE0	25	方式选择

图 4.7.4 USBN9602 引脚结构

USBN9602 作为 USB 接口的专用芯片，广泛用于测控技术、数据采集、信号处理等。普通单片机与 USBN9602 接口很简洁，参考电路如图 4.7.5。

图 4.7.5 普通单片机与 USBN9602 接口电路

2. 集成 USB 接口的单片机 EZ－USB

EZ－USB 是美国 Cypress 公司推出的带智能 USB 接口的单片机，它降低了 USB 外设的开发难度，提供了一个性能优良且价格较低的设计方案。以下简要介绍该单片机的结构、特点及使用方法。

(1) 芯片结构。EZ－USB 的内部结构如图 4.7.6 所示。一个加强的 8051 内核，性能可达到标准 8051 的 5～10 倍；一个智能 USB 引擎，可以代替 USB 外设开发者完成 USB 协议中规定的 80%～90%的通信工作，这使得开发者不需要深入了解 USB 的低级协议，即可顺利地开发出所需要的 USB 外设；一个 USB 收发模块；一个锁相环；对外有 24 个 I/O、16 位的地址线、8 位的数据线、一个 I^2C 口和一对 USB 口(D+、D–)；其内部还集成有非易失性存储器(如 EPROM、E^2PROM、FLASH ROM)、微处理器、RAM、SIE、DMA，减少了各芯片接口部分的时序配合的麻烦。EZ－USB 遵从 USB1.0 规范(12 Mb/s)，支持远程唤醒。

图 4.7.6　EZ－USB 的内部结构

外设未通过 USB 接口接到 PC 机之前，外设上的固件(firmware)存储在 PC 上，一旦外设接到 PC 机上，PC 先询问该外设是“谁”(即读设备的标识符)，然后将该外设的固件下载到 EZ-USB 的 RAM 中，这个过程叫作再枚举。

这个特性给 USB 外设开发者带来很多方便，比如，在开发过程中，当固件需要修改时，可以先在 PC 机上修改，然后下载到 EZ－USB，从而避免了烧片子的麻烦。当开发好的外设已经批量售出，开发商需要向用户提供固件的升级版本时，就可以把代码在网上发布，各地的用户自行将其下载到本地 PC 机上，即完成了升级，而无需用户具备硬件开发的专业技能。

EZ－USB 的固件在测试阶段可独立于驱动程序，驱动程序和固件的开发及调试相互独立，可加快开发的速度。

(2) 软件设计。EZ－USB 具有高集成度的 Windows 界面开发系统，C 风格的代码，自动生成的程序调度机制等特点，大大降低了开发人员的工作量。EZ－USB 的软件包中包括一个通用驱动程序(General Purpose Driver，简称 GPD)、一个固件代码框架及一些例子代码。PC 机上的应用程序以不同的参数调用 GPD 的函数，实现对外设的控制。在 USB 外设上，其固件设计开发工具中的框架是自动生成的，USB 外设开发者根据外设功能的具体要求，选择感兴趣的函数逐一填写函数体，从而开发出高效、简洁的 USB 驱动程序。

4.8 IEEE 1394 总 线

IEEE 1394 的正式公布名称是 IEEE 1394 高性能串行总线。这是美国国家标准协会于 1995 年制定的标准的高速、低成本串行总线，用于连接计算机外设，或作为计算机底板并行总线的备份总线。Apple 公司称之为 FireWire(火线)，Sony 公司称之为 i.Link，TexasInstruments 公司称之为 Lynx。尽管各厂商注册的商标名称不同，但其实质都是指一项技术，即 IEEE 1394。

尽管 IEEE 1394 目前还没有被 PC 厂商所广泛采用，但其作为一种数据传输的开放式技术标准，被应用在众多的领域。目前来说，IEEE 1394 技术使用得最广的领域还是数字成像领域，支持的产品包括数字照相机和摄像机等。

4.8.1 IEEE 1394 的特点与结构

IEEE 1394 有主要以下特点：

(1) 高的传输速率 在 TTL 总线底板上，它的传输速率为 24.576 Mb/s 在 BTL(Backplane Transceiver Logic，总线底板收发器逻辑)总线底板上，为 49.152 Mb/s；用电缆传输时可达 98.304 Mb/s、196.608 Mb/s 和 393.216 Mb/s。

(2) 节点之间的最大距离不能超过 4.5 m，不过可以使用 IEEE 1394 中继器克服这一局限。一台 IEEE 1394 中继器可以将节点之间的距离延长 4.5 m。因为 IEEE 1394 最多只能支持 16 层树形网段，所以两个端点之间的最大距离为 72(16×4.5)m。

(3) 每个网段最多可以连接 63 台设备，每台 IEEE 1394 可以连接 10 231 394 个网段，从而可以实现各种复杂的网络结构。不过，考虑到两个节点之间 4.5 m 的最大距离限制，IEEE 1394 并不适合在广域网中使用。

(4) 因为 IEEE 1394 设备支持热插拔，所以可以在任何时候向 IEEE 1394 网络添加或拆除设备。这样一来既不用担心影响数据的传输，也不需要进行重新配置，系统可以根据变化的环境进行自动调节。

(5) IEEE 1394 网络使用的是对等结构，可自动分配结点地址，不需要地址开关，也不需要设置专门的服务器。不过，对于那些集中进行管理或数据存储的系统来说，IEEE 1394 并不是一个理想的选择。

(6) 同一网络中的数据可以不同的速度进行传输，目前可以实现的速度为 100 Mb/s、200 Mb/s 和 400 Mb/s。这一特点决定了在设计网络时一定要考虑到不同设备的传输性能。如果在两台传输速度可达 400 Mb/s 的设备之间放置一台 100 Mb/s 的设备，无疑会使实际的传输速度大大降低。

IEEE 1394 总线的物理拓扑如图 4.8.1 所示。它有两种形式：一种是与计算机并行总线同处在总线底板上(如图中结点 1 和 2)，称为总线底板环境(backplane environment)串行总线；另一种是用电缆连接的串行总线，称为电缆环境(cable environment)串行总线。前者可作为并行总线的备份，必要时可代替并行总线工作，后者用于连接外设。

图 4.8.1 中的结点 1 和结点 2 可以是不同总线结构的计算机，它们可以通过总线桥彼此互连，协同工作。

图 4.8.1　IEEE 1394 总线的物理拓扑

IEEE 1394 开放式主控制器接口(OHCI)是向所有准备支持 IEEE 1394 技术的厂商提供的开放式标准。OHCI 由物理层、链路层、交易层和串行总线管理 4 个部分组成。具体功能如下：

(1) 物理层(Physical Layer)。该层提供设备和线缆之间的电气和机械连接，处理数据传输和接收，确保所有设备可以正常访问总线。物理层功能由硬件实现。

(2) 链路层(Link Layer)。该层提供同步和异步模式下的数据包接收确认、定址、数据校验以及数据分帧等。链路层功能由硬件实现。

(3) 交易层(Transaction Layer)。该层只处理异步数据包，提供 Read、Write 和 Lock 命令。Read 命令向命令发出方传回数据；Write 命令向接收方发送数据；Lock 命令通过生成往返通路实现 Read 和 Write 功能。交易层功能由固件实现。

(4) 串行总线管理(Serial Bus Management)。该层提供全部总线的控制功能，包括确保向所有总线连接设备供应电力，优化定时机制，分配同步通道 ID 以及处理基本错误等。

在实际操作过程中，设备首先要求控制物理层。如果进行异步传输，数据发送和接收方互换地址，然后进行数据传输。当接收方收到数据包时，会向发送方传回确认信息。如果接收方没有收到数据包，则启动错误修复机制。

如果进行同步传输，发送方首先要求获得一个特定带宽的数据通道，然后将通道 ID 附加在所要传输的数据中一起发送。接收方对数据流进行检测，只有当发现具有特定 ID 号的数据时才进行接收。

同步数据传输模式在优先级上要高于异步传输模式。当一台设备发送同步数据时，将获得一个专用的数据通道，直到数据传送完毕为止。而同一时刻发生的异步数据传输则只能使用当前所剩的可用带宽。上述机制充分保证了像视频流这样的对时间延迟要求很高的应用，可以在不受其它应用干扰的情况下实时完成。

在 OHCI 规范中没有任何对数据调制或解调的规定。这是因为，IEEE 1394 是一种全数字协议，在数据传输过程中不需要进行任何的数模转换，从而大大节省了系统开销。

4.8.2 IEEE 1394 的连接方式

IEEE 1394 支持两种不同的连接器。最为常用的一种是直接与 6 条铜质导线进行连接，如图 4.8.2 所示。其中一对线用于传送数据，一对线用于传送时钟信号，还有一对线传送电源。电缆通过端口把各结点连接起来。端口由收发器、终端匹配器和一些逻辑电路组成，起中继器作用。通过串行总线可提供的电源电压范围为 8～40VDC，电流为 1.5 A。由于通过串行总线可提供电源，因此即使结点的本地电源出现故障，也不致影响总线的正常工作。另一种是 Sony 公司推出的只包含 4 条数据线的小型线缆，并专门设计了与之搭配的新型连接器。这种连接器如果要与标准的 6 导线线缆连接的话，需要使用转换器。因为小型线缆不提供电源线，所以与之连接的设备只能使用外部电源供电。

图 4.8.2 IEEE 1394 串联线的切面示意图

虽然 IEEE 1394 可以通过串联线为接驳设备供电，但是对于各种连接设备来说，只靠连接线供电还是远远不够的。例如，像硬盘这种对于电量要求较高的设备，就很难从所接入的设备中得到充足的电力供应。

通过 IEEE 1394 连接的各种设备可以采用任何一种拓扑结构。下面以图 4.8.3 为例简单地进行介绍。

图中所示的连接结构充分显示出 IEEE 1394 网络的灵活性。IEEE 1394 网桥左、右两边的所有设备各自组成一个独立的网段。网桥可以对网段之间的数据进行有效的隔离。这样，左边网段中的视频相机等对网络带宽要求较高的设备或应用就不会影响右边网段的传输性能。不过网桥也允许网段之间进行数据互换，1 号计算机可以使用 2 号打印机，而 2 号计算机也可以访问视频相机中的数据。图中的网络还使用了 IEEE 1394 分离器将 1 号打印机单独隔离，这样就不会因为打印机的速度过慢而影响到 2 号计算机和数字摄像机之间的高速连接。

图 4.8.3　IEEE 1394 连接设备图

4.8.3　IEEE 1394 与 USB 发展前景比较

IEEE 1394 标准目前仍处于不断改进和发展之中，不久前改进的 IEEE 1394 已将数据传输速率提高到 800 Mb/s～3.2 Gb/s。

USB1.1 和后来速度更高的 USB2.0 都是具有与 1394 相似的技术特性的串行总线。但是，1394 和 USB 是面向不同的市场应用程序的。USB 完全是 PC 内部的总线。USB 的目的是提供一个低成本的、即插即用的方式来连接计算机外设和 PC。USB 符合 PC 业界的简单 PC 倡议(Easy PC Initiative)，因为它的接口对用户非常友好，所以它的性能特性允许移除许多传统的接口，如串口、并口、游戏端口、PS2 等等。USB1.1 可以在目前的 PC 上通用，USB2.0 将继续保持这一特性。

正如计算机界 USB 的兴起一样，消费性电子界也越来越倾向于使用 1394 作为新一代数码消费性电子设备的互连方式。整个家庭影院的设备都可以用一根 1394 线缆在各个设备之间进行互连，消除了目前家庭娱乐系统的某些不良的线路特性。有些电子公司将 1394 视作实现“家庭 A/V 网络”的潜在动力。

虽然 USB2.0 和 P1394－2000 有许多相似的技术特性，但它们是从不同的应用领域发展起来的，因此将来的发展方向会大不相同：USB 用于计算机的外设；1394 用于数码消费性电子设备。当然，有时这个界限可能会模糊不清，同一设备上可能同时使用了 USB 和 1394。这两种总线会共存并成功地发展下去。

4.9 MPS 接口及应用

4.9.1 具有 MPS 接口的 X84041 (E^2PROM)简介

MPS 是 Micro Port Saver(节省微处理器接口的器件)的缩写，代表产品是 Xicor 公司的串行 E^2PROM 存储器的 X84XXX 系列。MPS 接口器件可直接连到处理器总线上，并通过标准总线读写时序进行数据传输。这样，在微控制器的控制线如 $\overline{WR}$、$\overline{RD}$ 等的配合下，可以最大限度地节省 I/O 口，并简化逻辑编程。下面就以 X84041 为例加以介绍。

X84041 是 MPS 接口的 4096 位串行 E^2PROM，可组织成 512×8、256×16 或 128×32 的结构；3.3 Mb/s 数据传输率；低功耗 CMOS 结构，2.7～5.5 V 工作电压；读取时间 4.5 ns，典型写周期 5 ms；10 万次擦写次数，大于 100 年的数据保存期。其 8 脚封装形式如图 4.9.1 所示，管脚说明见表 4.9.1，与 51 单片机连接电路如图 4.9.2 所示。有关 X84041 的详细内容请参考 Xicor 公司提供的产品资料。

图 4.9.1 X84041 管脚定义

表 4.9.1 X84041 管脚说明

符 号	说 明
I/O	数据输入/输出
$\overline{CE}$	芯片使能输入
$\overline{OE}$	输出使能输入
$\overline{WE}$	写使能输入
$\overline{WP}$	写保护输入
Vcc	电源
Vss	地
NC	没连接

图 4.9.2 51 单片机与 X84041 的连接

4.9.2　X84041 的接口时序关系

X84041 的工作时序分别如图 4.9.3～图 4.9.7 所示。X84041 的具体操作命令可参考 4.9.3 节的测试程序。

图 4.9.3　X84041 的“读”周期时序关系

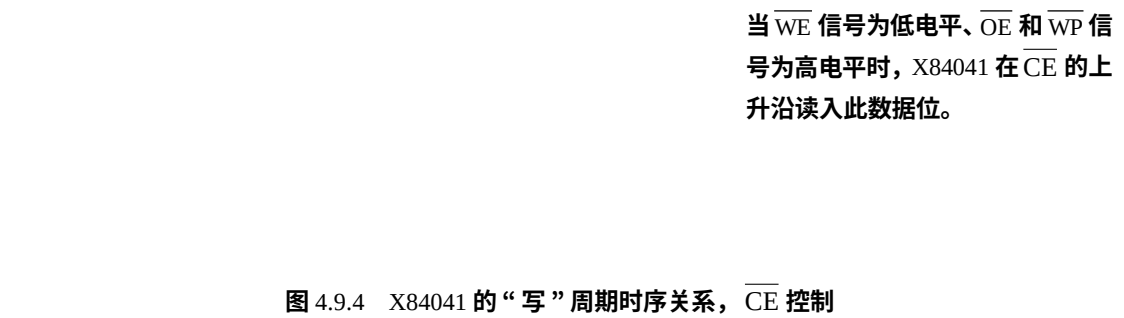


图 4.9.4　X84041 的“写”周期时序关系，$\overline{CE}$ 控制

图 4.9.5　X84041 的“写”周期时序关系，$\overline{WE}$ 控制

图 4.9.6 X84041 的读操作时序

图 4.9.7 X84041 的写操作时序

4.9.3 X84041 与 51 系列单片机接口的测试程序

按图 4.9.2，将 89C51 单片机与 X84041 连接，地址译码使用 A13、A14、A15 三根地址线，即在程序存储空间 E000H～FFFFH 范围内可选通/$\overline{\text{CS}}$ 脚。

下面这段程序的功能是：

(1) 写单字节：将#11H 写入 E^2PROM 的 000H 地址。

(2) 读单字节：从 E^2PROM 的 00H 地址读出数据。

(3) 整页写：用页写的方法将#22H，#33H，# 44H 三个数据写入 E^2PROM 的 100H，101H，102H 地址。

(4) 连续读：用连续读的方法从 E^2PROM 的 100H、101H、102H 地址中读出数据。

具体汇编程序如下：

例 4.9.1

```
BYTE_ADDR equ 0F000H          ；单字节操作地址
BYTE_DATA equ 11H
PAGE_ADDR equ 0F100H          ；页操作地址
PAGE_DATA1 equ 22H
PAGE_DATA2 equ 33H
PAGE_DATA3 equ 44H
STACK_TOP equ 060H            ；堆栈地址

ORG 0000H
        jmp main
ORG 0030H
MAIN:
        mov SP，#STACK_TOP
        clr EA                ；禁止中断
        lcall byte_write
        lcall byte_read
        lcall page_write
        lcall sequ_read
        ljmp $

byte_write:；写单字节
        mov DPTR， #BYTE_ADDR
        lcall rst             ；送入复位时序信号，表示操作开始
        mov A， DPH
        lcall load_byte       ；送入地址的高字节
        mov A， DPL
        lcall load_byte       ；送入地址的低字节
        mov A， #BYTE_DATA
        lcall load_byte       ；送入数据
        lcall nv_start        ；开始非易失性写周期
        lcall nv_poll         ；等待非易失性写周期结束
            ret

page_write:                   ；整页写
        mov DPTR， #PAGE_ADDR
        lcall rst             ；送入复位时序信号，表示操作开始
        mov A， DPH
        lcall load_byte       ；送入地址的高字节
```

```
        mov A， DPL
        lcall load_byte          ；送入地址的低字节
        mov A， #PAGE_DATA1
        lcall load_byte          ；送入第一个数据
        mov A， #PAGE_DATA2
        lcall load_byte          ；送入第二个数据
        mov A， #PAGE_DATA3
        lcall load_byte          ；送入第三个数据
        lcall nv_start           ；开始非易失性写周期
        lcall nv_poll            ；等待非易失性写周期结束
        ret

byte_read:                       ；读单字节
        mov DPTR， #BYTE_ADDR
        lcall rst                ；送入复位时序信号，表示操作开始
        mov A， DPH
        lcall load_byte          ；送入地址的高字节
        mov A， DPL
        lcall load_byte          ；送入地址的低字节
        lcall rec_byte           ；接收一个字节的数据
            ret

sequ_read:                       ；连续读
        mov DPTR， #PAGE_ADDR
        lcall rst                ；送入复位时序信号，表示操作开始
        mov A， DPH
        lcall load_byte          ；送入地址的高字节
        mov A， DPL
        lcall load_byte          ；送入地址的高字节
        lcall rec_byte           ；接收数据的第一个字节
        lcall rec_byte           ；接收数据的第二个字节
        lcall rec_byte           ；接收数据的第三个字节
        ret

load_byte: ；
        mov R0， #08             ；循环 8 次
    loop1:
        rl A
        movx @DPTR， A           ；将 A 寄存器的最低位送入 E2PROM
```

```
        djnz R0，  loop1
        ret

rec_byte:  ；
        mov R0，  #08          ；循环 8 次
        mov R1，  #00          ；R1 清零
loop2:
        clr C
            movx A，  @DPTR    ；读数据位，存入 A 寄存器的最低位
        jnb ACC.0，  zero_bit
        setb C
        zero_bit:
        mov A，  R1            ；将读入的数据位存入 R1
        rlc A
        mov R1，  A
        djnz R0，  loop2
        ret

rst:    ；复位
        movx A，  @DPTR        ；读一位数据
        mov A，  #00H
        movx @DPTR，  A        ；写入位“ 0 ”
        movx A，  @DPTR        ；产生“ 读 ”周期时序关系
        ret

nv_start:                      ；非易失性写周期开始
        movx A，  @DPTR        ；读数据位
        mov A，  #01H
        movx @DPTR，  A        ；写入数据位“ 1 ”
        movx A，  @DPTR        ；读数据位
        ret

nv_poll:
            mov R0，  #0FFH    ；设置延迟时间
loop3:
        movx A，  @DPTR
        nop
        nop
        nop
```

```
        jb ACC.0，  end_loop        ；若数据位为“1”，表示非易失性写周期结束
        djnz R0，  loop3            ；非易失性写周期结束
end_loop:
        ret
END
```

4.10 Microwire 总线接口

4.10.1 Microwire 总线简介

Microwire 总线是 NS(National Semiconductor)公司推出的三线同步串行总线。这种总线由一根数据输出线(SO)、一根数据输入线(SI)和一根时钟线(SK)组成(但每个器件还要接一根片选线)。原始的 Microwire 总线上只能连接一台单片机作为主机，由它控制时钟线，总线上的其它设备都是从机。此后，NS 公司推出了 8 位的 COP800 单片机系列，仍采用原来的 Microwire 总线，但单片机上的总线接口改成既可由自身发出时钟，也可由外部输入时钟信号，也就是说，连接到总线上的单片机既可以是主机，也可以是从机。为了区别于原有 Microwire 总线，称这种新产品为增强型的 Microwire/PLUS。增强型的 Microwire/PLUS 总线上允许连接多台单片机和外围器件，因此，总线具有更大的灵活性和可变性，可以应用于分布式、多处理器的复杂系统。由于该总线只需三根信号线，因此可以大大缩小整个系统的尺寸；同时，由于该总线的连接和拆卸都很方便，使系统的设计大大简化，也就是说，要改变一个系统，仅需改变连接到总线上的单片机及外围器件的数量和型号。

每台接到总线上的设备应有三根信号线，即数据输出线(SO)、数据输入线(SI)和时钟线(SK)。所有设备的时钟线连接到同一根 SK 线上，设备连接如图 4.10.1 所示。主机向 SK 线

图 4.10.1 Microwire 总线设备连接图

发送时钟脉冲信号，从机设备在时钟信号的同步沿输出/输入数据。主机设备的数据输出线 SO 和所有从机设备的数据输入线相接，从机设备的数据输出线都接到主机输入线上。主机经过另外的片选线选通某一从机设备，然后发出时钟脉冲，主机和选通的从机在时钟的下降沿从各自的 SO 线输出一位数据，在时钟的上升沿从各自的 SI 端读入一位数据；每个时钟周期发送一位数据，接收一位数据，从而实现数据的交换。

Microwire 总线接口的典型产品是 93CXX 系列的 E^2PROM。

4.10.2　93C46 系列串行 E^2PROM 的应用

下面以仙童公司的 FM93C46 为例介绍串行 E^2PROM 的具体应用。

1. FM93C46 的特点及封装

FM93C46 是 1024 位 COMS 非易失性 E^2PROM。它的特点如下：

(1) 组织形式为 64×16bit；

(2) 采用 Microwire4 线制串行接口；

(3) 宽电压范围为 2.7～5.5 V；

(4) 可完成 1 百万次擦写周期;

(5) 可以保持 40 年数据;

(6) 可以具有 10 ms 的写入周期。

FM93C46 的 DIP 封装如图 4.10.2 所示，管脚功能见表 4.10.1。

表 4.10.1　FM93C46 的管脚功能

管脚名称	功能
CS	片选端(高有效)
SK	串行时钟信号端
DI	串行数据输入端
DO	串行数据输出端
GND	地
NC	空脚
V_{CC}	电源

图 4.10.2　FM93C46 的 DIP 封装形式

2．FM93C46 的操作

FM93C46 共有 7 条控制命令。控制命令由 1 位起始位开始，接着是 2 位操作码，然后是 6 位的地址段，最后是 16 位的数据段。各控制命令代码见表 4.10.2。

表 4.10.2　FM93C46 控制命令

操作命令	起始位	操作码	地址段						数据段
读	1	10	A5	A4	A3	A2	A1	A0	
写使能	1	00	1	1	X	X	X	X	
写	1	01	A5	A4	A3	A2	A1	A0	D15～D0
全写	1	00	0	1	X	X	X	X	D15～D0
写禁止	1	00	0	0	X	X	X	X	
擦除	1	11	A5	A4	A3	A2	A1	A0	
全擦除	1	00	1	0	X	X	X	X	

FM93C46 同步时序如图 4.10.3 所示。在 CS 高电平时，DI、DO 都在 SK 的上跳沿输入有效。

FM93C46 读周期时序如图 4.10.4 所示。FM93C46 写使能时序如图 4.10.5 所示。FM93C46 写周期时序如图 4.10.6 所示。

图 4.10.3 FM93C46 同步时序

图 4.10.4 FM93C46 读周期时序

图 4.10.5　FM93C46 写使能时序

图 4.10.6　FM93C46 写周期时序

3. FM93C46 读写程序

下面给出 FM93C46 的读写参考模块，用 C51 语言编写。

单片机采用 89C52，选用 12 MHz 晶振。信号线定义如下：CS 接 P1.0，SK 接 P1.1，DI 接 P1.2，DO 接 P1.3。

例 4.10.1

```
#include <reg52.h>
#define uchar unsigned char
#define uint unsigned int
sbit CS=P1^0；
sbit SK=P1^1；
sbit DI=P1^2；
sbit DO=P1^3；
```

```
/*
uchar ReadChar(uchar address)；
void WriteChar(uchar address，uchar InData)；
void ReadString(uchar data *RamAddress，uchar RomAddress，uchar Number)；
void WriteString(uchar data *RamAddress，uchar RomAddress，uchar Number)；
*/

/*写使能函数(擦除或写操作之前执行必须)*/
void Ewen(void) {
  uchar temp，InData；
  CS=0；
  SK=0；
  CS=1；
  InData=0x98；// 10011XXXX
  for(temp=9；temp!=0；temp--) {  // 9
      DI=InData&0x80；
      SK=1；    SK=0；
      InData<<=1；
  }
  CS=0；
}

/*写禁止(此操作执行后可有效地保护数据)*/
void Ewds(void) {
  uchar temp，InData；
  CS=0；
  SK=0；
  CS=1；
  InData=0x80；// 10000XXXX
  for(temp=9；temp!=0；temp--) {  // 9
      DI=InData&0x80；
      SK=1；    SK=0；
      InData<<=1；
  }
  CS=0；
}

/*读某个地址的的数据(16 位)*/
uint Read(uchar address) {
```

```
    uchar temp；
    uint result；
    Ewen( )；
    SK=0；    DI=1；// 110 A5-A0
    CS=0；    CS=1；
    SK=1；    SK=0；              // 1
    address=address&0x3f|0x80；
    for(temp=8；temp!=0；temp--) {  // 8
        DI=address&0x80；
        SK=1；    SK=0；
        address<<=1；
    }
    DO=1；
    for(temp=16；temp!=0；temp--) {// 16
        SK=1；
        result=(result<<<=1；
    }
    for(temp=16；temp!=0；temp--) {// 16
        DI=InData&0x8000；
        SK=1；    SK=0；
        InData<<=1；
    }
    CS=0；    DO=1；
    CS=1；    SK=1；
    while(DO==0) {              // 忙(正在写)检测
        SK=0；    SK=1；
    }
    SK=0；    CS=0；
    Ewds( )；
}

/*擦除数据(地址 An～A0)*/
void Erase(uchar address) {
    uchar temp；
    Ewen( )；
    SK=0；    DI=1；// 111 A5-A0
    CS=0；    CS=1；
    SK=1；    SK=0；              // 1
    address|=0xc0；
```

```
  for(temp=8；temp!=0；temp--) {  // 8
      DI=address&0x80；
      SK=1；    SK=0；
      address<<=1；
  }
  CS=0；    DO=1；
  CS=1；    SK=1；
  while(DO==0) {
      SK=0；    SK=1；
  }
  SK=0；    CS=0；
  Ewds( )；
}

/*擦除整个芯片(仅在电源电压 VCC = 4.5～5.5 V 时有效)*/
void Eral(void) {
  uchar temp，InData；
  Ewen( )；
  CS=0；
  SK=0；
  CS=1；
  InData=0x90；// 10010XXXX
  for(temp=9；temp!=0；temp--) {  // 9
      DI=InData&0x80；
      SK=1；    SK=0；
      InData<<=1；
  }
  CS=0；    DO=1；
  CS=1；    SK=1；
  while(DO==0) {
      SK=0；    SK=1；
  }
  SK=0；    CS=0；
  Ewds( )；
}

/*写整个芯片(仅在电源电压 VCC = 4.5～5.5 V 时有效)*/
void Wral(uint InData) {
  uchar temp，address；
```

```
  Ewen( )；
  CS=0；
  SK=0；
  CS=1；
  address=0x88；    // 10001XXXX
  for(temp=9；temp!=0；temp--) {  // 9
      DI=address&0x80；
      SK=1；    SK=0；
      address<<=1；
  }
  for(temp=16；temp!=0；temp--) {// 16
      DI=InData&0x8000；
      SK=1；    SK=0；
      InData<<=1；
  }
  CS=0；    DO=1；
  CS=1；    SK=1；
  while(DO==0) {
      SK=0；    SK=1；
  }
  SK=0；    CS=0；
  Ewds( )；
}

uchar ReadChar(uchar address) {
  uchar temp=address>>1；
  if(address&0x01) return((uchar)(Read(temp)>>8))；
  else return((uchar)(Read(temp)))；
}

void WriteChar(uchar address，uchar InData) {
  uchar temp=address>>1；
  if(address&0x01) Write(temp，(uint)(Read(temp)&0x00ff|(InData<>1) {
      temp=*RamAddress；
      RamAddress++；
      temp=temp|(*RamAddress)<>1，temp)；
      RomAddress++；
      RomAddress++；
      Number--；
```

```
        Number--；
    }
    if(Number) WriteChar(RomAddress，*RamAddress)；
}
```

4.10.3 A/D 转换器 TLC0831 的应用

TLC0831 是 TI 公司生产的八位逐次逼近模数转换器，它有一个差分输入通道，串行输出配置为与标准移位寄存器或微处理器兼容的 Microwire 总线接口。差分的模拟电压输入有利于共模抑制，并使模拟输入电压偏置值为零。输入基准电压(REF 端输入电压)可以调整，使较小的模拟输入电压可以编码到全 8 位分辨率。也可以使基准电压等于最大模拟输入值，从而得到满比例尺转换，获得最高的转换分辨率。典型情况下，REF 端连接至 VCC。TLC0832 内部已设定 REF 等于 VCC，未引出 REF 端。TLC0831 的操作和不同输入通道数的 TLC0832、TLC0834、TLC0838 非常相似，但由于 TLC0831 不需要设置通道选择，所以 DI 端口没有使用，也没有引出，这一点请读者注意。

TLC0831 的主要特点：

(1) 具有 8 位分辨率；

(2) 易于和微处理器接口或独立使用;

(3) 满比例尺操作或用 5 V 基准电压;

(4) 当单 5 V 供电时，模拟输入范围为 0～5 V；

(5) 输入和输出与 TTL 和 CMOS 兼容；

(6) 在 f_{CLOCK}=250 kHz 时，转换时间为 32 μs

(7) 兼容于美国国家半导体公司的 ADC083X 系列器件；

(8) 总非调整误差 1LSB;

(9) TLC083xC 的工作温度范围为 0～70 ℃，TLC083xI 的工作温度范围为– 40～85 ℃

TLC0831 的封装形式如图 4.10.7(双列直插式)所示，管脚功能见表 4.10.3。限于篇幅，TLC0831 的详细介绍请参阅 TI 公司的产品资料手册。本节仅给出 C51 程序供参考。

表 4.10.3 TLC0831 的管脚功能

管脚名称	功　　能
$\overline{CS}$	片选端(低电平有效)
IN+	差模输入正端
IN–	差模输入负端
GND	地
REF	输入基准电压
DO	串行数据输出端
CLK	串行时钟信号端
V_{CC}	电源

图 4.10.7 TLC0831 的封装形式

例 4.10.2

本节仅给出 C51 **程序供参考。单片机采用** 89C52，**选用** 12 MHz **晶振。信号线定义如下：** DO **接** P1.0；$\overline{\text{CS}}$ **接** P1.1；CLK **接** P1.2。

```
#include <reg52.h>
#include <stdio.h>
#include <absacc.h>
#define uchar unsigned char
#define uint unsigned int

sbit adcdo=P1^0；
sbit adccs=P1^1；
sbit adcclk=P1^2；

void delay1(uchar x)；
uchar readadc(void)；
void adcck(void)；

void delay1(uchar x) {
    uchar i；
    for(i=0；i<x；i++)；
}

void adcck(void) {
      adcclk=1；
delay1(2)；
      adcclk=0；
delay1(2)；
}

uchar readadc(void)
{     uchar i；
      adccs=0；
adcck( )；
ch=0；
      for (；adcdo==1；) adcck( )；
      for (i=0；i<8；i++)
{
```

```
adcck( )；
ch=(ch<<1)|adcdo；
}
    adccs=1；
    return(ch)；
}
```

4.11 其它串行接口的芯片操作

4.11.1 DS1302 串行时钟芯片的结构及工作原理

DS1302 是美国 DALLAS 公司推出的一种高性能、低功耗、带 RAM 的实时时钟芯片，它可以对年、月、日、星期、时、分、秒进行计时，且具有闰年补偿功能，工作电压为 2.5～5.5 V。DS1302 采用三线接口，与 CPU 进行同步通信，并可采用突发方式一次传送多个字节的时间数据或 RAM 数据。DS1302 内部有一个 31×8 的用于临时性存放数据的 RAM 寄存器。DS1302 是 DS1202 的升级产品，与 DS1202 兼容，但增加了主电源/后备电源双电源引脚，同时提供了对后备电源进行涓细电流充电的能力。

DS1302 的引脚如图 4.11.1 所示，引脚功能如表 4.11.1 所示，控制字如图 4.11.2 所示。

图 4.11.1 DS1302 引脚

表 4.11.1 DS1302 引脚功能

管脚号	管脚名称	功　　能
1	V_{CC2}	主电源
2，3	X1，X2	32.768 kHz 晶振接口
4	GND	地
5	$\overline{RST}$	复位兼片选端，读写操作时必为高电平
6	I/O	串行数据输入/输出
7	SCLK	串行时钟输入端，是串行数据的同步信号
8	V_{CC1}	后备电源

图 4.11.2 DS1302 控制字含义

控制字节的最高位(位 7)必须是逻辑 1，如果它为 0，则不能把数据写入到 DS1302 中; 第 6 位如果为 0，则表示存取日历时钟数据，为 1 表示存取 RAM 数据；位 5～位 1 指示操作单元的地址；最低位(位 0)如为 0 表示要进行写操作，为 1 表示进行读操作，控制字节总是从最低位开始输出。

DS1302 通过把 $\overline{\text{RST}}$ 输入驱动置高电平来启动所有的数据传送。$\overline{\text{RST}}$ 输入有两种功能：首先，$\overline{\text{RST}}$ 接通控制逻辑，允许地址/命令序列送入移位寄存器；其次，$\overline{\text{RST}}$ 提供了终止单字节或多字节数据的传送手段。当 $\overline{\text{RST}}$ 为高电平时，所有的数据传送被初始化，允许对 DS1302 进行操作。如果在传送过程中置 $\overline{\text{RST}}$ 为低电平，则会终止此次数据传送，并且 I/O 引脚变为高阻态。上电运行时，在 Vcc≥2.5 V 之前，$\overline{\text{RST}}$ 必须保持低电平。只有在 SCLK 为低电平时，才能将 $\overline{\text{RST}}$ 置为高电平。

在控制指令字输入后的下一个 SCLK 时钟的上升沿时，数据被写入 DS1302，数据输入从低位即 D0 开始。同样，在紧跟 8 位的控制指令字后的下一个 SCLK 脉冲的下降沿读出 DS1302 的数据。读出数据的顺序为从低位 D0～高位 D7，数据读写时序见图 4.11.3 和图 4.11.4。

图 4.11.3　DS1302 的读时序图

图 4.11.4　DS1302 的写时序图

DS1302 共有 12 个寄存器，其中有 7 个寄存器与日历、时钟相关，存放的数据位为 BCD 码形式，其日历、时间寄存器及其控制字见表 4.11.2。

DS1302 还有充电寄存器、时钟突发寄存器及与 RAM 相关的寄存器等。时钟突发寄存器可一次性顺序读写除充电寄存器外的所有寄存器内容。DS1302 与 RAM 相关的寄存器分为两类：一类是单个 RAM 单元，共 31 个，每个单元组态为一个 8 位的字节，其命令控制字为 C0H~FDH，其中奇数为读操作，偶数为写操作；另一类为突发方式下的控制寄存器，此方式下可一次性读写所有的 RAM 的 31 个字节，命令控制字为 FEH(写)、FFH(读)。

表 4.11.2 DS1302 的日历、时钟寄存器及其控制字

寄存器名		命令字格式		取值范围	位内容							
		写操作	读操作		7	6	5	4	3	2	1	0
秒寄存器		80H	81H	00～59	CH	10SEC			SEC			
分寄存器		82H	83H	00～59	0	10MIN			MIN			
小时寄存器		84H	85H	01～12 或 11～23	12/24	0	$\frac{10}{AP}$	HR	HR			
日期寄存器		86H	87H	01～28，29，30，31	0	0	10DATA		DATA			
月份寄存器		88H	89H	01～12	0	0	0	10M	MONTH			
星期寄存器		8AH	8BH	01～07	0	0	0	0	0	DAY		
年份寄存器		8CH	8DH	00～99	10YEAR				YEAR			
写保护寄存器		8EH	8F	00H/80H	WP	0						
涓流充电寄存器		90H	91H	—	TCS				DS		RS	
时钟突发寄存器		BEH	BFH	—	—							
RAM 突发寄存器		FEH	FFH	—	—							
RAM 寄存器	0	C0H	C1H	00～FFH	RAM 数据							
	⋮	⋮	⋮	00～FFH								
	30	FCH	FDH	00～FFH								

4.11.2 DS1302 在单片机系统中的应用

1. 硬件电路

DS1302 与 CPU 的连接仅需要三条线，即 SCLK(7)、I/O(6)、$\overline{RST}$(5)。DS1302 与 89C51 单片机连接的电路原理图如图 4.11.5 所示。图中 R1、R2、R3 为上拉电阻，阻值可选 5.1 kO。V_{CC1} 在单电源与电池供电的系统中提供低电源并提供低功率的电池备份。V_{CC2} 在双电源系统中提供主电源，在这种运用方式下，V_{CC1} 连接到备份电源，以便在没有主电源的情况下能保存时间信息以及数据，DS1302 由 V_{CC1} 或 V_{CC2} 两者中的较大者供电：当 V_{CC2} 大于 V_{CC1}+0.2V 时，V_{CC2} 给 DS1302 供电 当 V_{CC2} 小于 V_{CC1} 时，DS1302 由 V_{CC1} 供电。DS1302 时钟的产生基于外接晶体振荡器，振荡器的频率为 32768Hz，要求其自身的负载电容为 6 pF。如果测得振荡频率大于 32768 Hz，说明负载电容偏小；测荡频率小于 32768 Hz 说明负载电容偏大，可以用调整负载电容的方法来提高 DS1302 的计时精度。

图 4.11.5　DS1302 与 C51 单片机连接的电路图

2．参考程序

例 4.11.1

操作 DS1302 时钟芯片的 C51 语言程序如下：

```
#include <reg52.h>
#include <stdio.h>
#include <absacc.h>
#define uchar unsigned char
/*引脚连接关系定义*/
sbit clock_clk=P1^0
sbit clock_dat=P1^1；
sbit clock_rst=P1^2；

sbit a0=ACC^0；
sbit a1=ACC^1；
sbit a2=ACC^2；
sbit a3=ACC^3；
sbit a4=ACC^4；
sbit a5=ACC^5；
```

```
sbit a6=ACC^6；
sbit a7=ACC^7；

void clock_out(uchar dd) {
  ACC=dd；
  clock_dat=a0；clock_clk=1；clock_clk=0；
  clock_dat=a1；clock_clk=1；clock_clk=0；
  clock_dat=a2；clock_clk=1；clock_clk=0；
  clock_dat=a3；clock_clk=1；clock_clk=0；
  clock_dat=a4；clock_clk=1；clock_clk=0；
  clock_dat=a5；clock_clk=1；clock_clk=0；
  clock_dat=a6；clock_clk=1；clock_clk=0；
  clock_dat=a7；clock_clk=1；clock_clk=0；
}
uchar clock_in(void) {
  clock_dat=1；
  a0=clock_dat；
  clock_clk=1；clock_clk=0；a1=clock_dat；
  clock_clk=1；clock_clk=0；a2=clock_dat；
  clock_clk=1；clock_clk=0；a3=clock_dat；
  clock_clk=1；clock_clk=0；a4=clock_dat；
  clock_clk=1；clock_clk=0；a5=clock_dat；
  clock_clk=1；clock_clk=0；a6=clock_dat；
  clock_clk=1；clock_clk=0；a7=clock_dat；
  return(ACC)；
}
uchar read_clock(uchar ord) {
  uchar dd=0；
  clock_clk=0；
  clock_rst=0；
  clock_rst=1；
  clock_out(ord)；
  dd=clock_in( )；
  clock_rst=0；
  clock_clk=1；
  return(dd)；
}
void write_clock(uchar ord，uchar dd) {
  clock_clk=0；
```

```
  clock_rst=0；
  clock_rst=1；
  clock_out(ord)；
  clock_out(dd)；
  clock_rst=0；
  clock_clk=1；
}

void main(void) {
  uchar address，d；
  address=0；          /*读内部 RAM  地址为 00H*/
  d=read_clock(0xc0|address)；
  address=1；          /*读内部 RAM 地址为 01H*/
  d=read_clock(0xc0|address)；
write_clock(0xc1|0x7，0)； /*写保护寄存器设置成为 0，允许写*/
  address=0；d=5；    /*写内部 RAM   地址为 00H，写入数据 5*/
  write_clock(0xc1|address，d)；
  address=1；d=123； /*写内部 RAM   地址为 01H，写入数据 123*/
  d=read_clock(0xc1|address，d)；
  address=0；          /*读内部时钟日历数据  0：s*/
  d=read_clock(0x80|address)；
  address=1；          /*读内部时钟日历数据  1：min*/
  d=read_clock(0x80|address)；
  address=0；d=3；    /*写内部时钟日历数据 0：s，写入 03 s*/
  write_clock(0x80|address，d)；
}
```

说明：

(1) **每次上电，必须把秒寄存器高位(第 7 位 CH)设置为 0，时钟才能走时。**

(2) **如果需要写入数据和时钟日历信息，必须把写保护寄存器 WP 位设置成为 0。**

(3) **使 DS1302 开始涓细电流充电的语句如下：**

```
write_clock(0xc1|0x8，0xaa)；
```

(4) **请读者注意，本例中的基本读写函数 CLOCK_IN 和 CLOCK_OUT 没有用循环移位的方法。虽然代码长度增加，但是，由于延时短，因此时序易于严格控制，并且时序简单直观，不易出错，因此在综合考虑编程效率、执行速度、代码长度后，这也是一种不错的方法。**

第 5 章　51 单片机之间的通信技术

本章主要内容：

★ 51 系列单片机串行口简介
★ 单片机点到点通信的实例
★ 单片机多机通信实例
★ 用软件方法提高单片机通信距离

5.1　51 系列单片机串行口简介

在正式讲解之前，首先需要了解下面几个基本概念：

(1) 串行通信：指数据一位一位地按顺序传送的通信方式。其优点是所需传输线路少，适合远距离通信；其缺点是传输速度慢。

(2) 波特率：指串行通信中每秒能发送或接收的二进制位数。若发送一位时间为 T，则波特率为 $1/T$。

(3) 单工串行口：只能接收或只能发送的串行口。

(4) 半双工串行口：既可接收又可发送，但两个动作不能同时进行的串行口。

(5) 全双工串行口：能同时进行接收和发送的串行口。

(6) f_{osc} :单片机系统时钟频率,通常为所用晶体振荡器的频率(可选范围为 1.2～40 MHz，视单片机具体情况而定)。

(7) 奇偶校验：异步接收/发送器(UART)在发送时，电路自动检测发送字符位中“1”的个数，并在奇偶校验位上添加“1”或“0”，使得“1”的总和(包括奇偶校验位)为偶数(偶校验)或奇数(奇校验)。

5.1.1　MCS－51 串行口结构

8051 系列单片机片上构造有可编程的全双工通用异步接收/发送器(UART)，可用于串行通信，发送时数据由 TxD(P3.1)端送出，接收时数据由 RxD(P3.0)端输入。有两个数据缓冲器(SBUF)，一个用作发送，另一个用作接收。短距离的机间通信可直接使用 UART 的 TTL 电平，使用驱动芯片(MAX232 等)可接成 RS－232C 和通用微机串口进行通信。波特率时钟必须从内部定时器 1 或定时器 2 获得。若在应用时要求 RS－232C 完全的握手功能，则须借助单片机其它管脚利用软件进行处理。

(1) 数据缓冲器 SBUF。串行口有两个物理上各自独立的发送缓冲器和接收缓冲器。这两个缓冲器共用一个地址 99H，发送缓冲器只写不读，接收缓冲器只读不写。接收缓冲器

是双缓冲的，以避免在接收下一帧数据之前，CPU 未能及时响应接收器中断，没有把上一帧数据读走而产生两帧数据重叠问题。

(2) 串行口控制寄存器 SCON。SCON 的字节地址为 98H，可按位寻址，位地址为 98H～9FH。其格式为：

SM0(D7)	SM1	SM2	REN	TB8	RB8	TI	RI(D0)

说明：

SM0、SM1：串行口工作方式控制位，具体工作方式如表 5.1.1 所示。

SM2：允许方式 2 和方式 3 多机通信控制位。在方式 2 和方式 3 中，若 SM2=1，则接收到的第 9 位数据(RB8)为“0”时，不启动接收中断标志 RI(RI=0)。在方式 1 中，若 SM2=1，则只有接收到有效停止位时才启动 RI，若没接收到有效停止位，则 RI 清“0”。在方式 0 中，SM2 应为“0”。

表 5.1.1　串行口工作方式表

SM0	SM1	工作方式	功 能 说 明	所用波特率
0	0	0	同步移位寄存器方式 (用于 I/O 扩展或同步传输)	f_{osc} /12
0	1	1	10 位异步收发 (波特率可变)	由定时控制
1	0	2	11 位异步收发 (波特率固定)	f_{osc} /32 或 f_{osc} /64
1	1	3	11 位异步收发 (波特率可变)	由定时控制

REN：串口接收允许位。由软件置位。置“1”是允许接收，置“0”时禁止接收。

TB8：在方式 2 和方式 3 中是要发送的第 9 位数据。由软件置位或清零。

RB8：在方式 2 和方式 3 中是接收到的第 9 位数据。在方式 1 时，若 SM2=0，RB8 是接收到的停止位。在方式 0 中不使用 RB8。

TI：发送中断标志。由硬件在方式 0 发送到第 8 位时置“1”，或在其它方式中发送停止位时置“1”。必须由软件清零。

RI：接收中断标志。由硬件在方式 0 接收到第 8 位时置“1”，或在其它方式中接收停止位时置“1”。必须由软件清零。

(3) 特殊功能寄存器 PCON。PCON 的字节地址是 87H，没有位寻址功能。PCON 只有第 7 位 SMOD 是与串行口的波特率设置有关的选择位。其格式为：

SM0(D7)	空	空	空	GF1	GF0	PD	IDL(D0)

 **说明：**

当 SMOD=1 **时，串行口波特率加倍。**

GF0、GF1：**两个通用标志位。**

PD、IDL：CMOS **器件的低功耗控制位。**

(4) UART **出错标志。为了使数据传输更为可靠，**UART **常设置如下三种出错标志：**

① **奇偶错误** PE(Parity Error)：**奇偶错误** PE **由奇偶错误标志触发器指示，该触发器由奇偶校验结果信号置位。**

② **帧错误** FE(Frame Error)：**帧错误由帧错误标志触发器** FE **指示。该触发器在** UART **检测到帧的停止位不是“** 1 **”而为“** 0 **”时** FE **被置位。**

③ **溢出错误** OE(Overrun Error)：UART **接收端在接收到第一个字符后便放入“接收数据缓冲器”，然后就继续从** RxD **线上接收第二个字符，并等待** CPU **从“接收数据缓冲器”中取走第一个字符。如果** CPU **很忙，一直没有机会取走第一个字符，以致接收到的第二字符进入“接收数据缓冲器”而造成第一个字符丢失。发生这种错误时，**UART **自动使“溢出错误标志触发器”** OE **置位。**

5.1.2 串行口工作方式

串行口具有四种工作方式，下面介绍各种方式的功能和外特性，同时简要说明串行口的内部逻辑和时序。

1．**方式** 0

方式 0 **为寄存器输入/输出方式，可外接移位寄存器，以扩展** I/O **接口，或以较高的速率接同步输入/输出设备。**

在此方式下，波特率固定为 f_{osc}/12**。这时传送的数据，无论是输入还是输出，均通过引脚** RxD(P3.0)**端，移位同步脉冲均由** TxD(P3.1)**输出。每一帧数据为** 8 **位二进制数，低位在先，高位在后。**

(1) **方式** 0 **发送。当一个数据写入缓冲** SBUF **时，串口以振荡频率的十二分之一的波特率，将数据从** RxD **端串行输出，**TxD **端输出移位同步信号，发送完毕时，中断标志** TI **置“** 1 **”。**

(2) **方式** 0 **接收。串行口定义为方式** 0**，并将** REN **置“** 1 **”后，便启动串口以振荡频率的十二分之一的波特率接收数据，**TxD **端输出移位同步信号，**RxD **为数据输入端，当接收器接收到** 8 **位数据时，中断标志** RI **置“** 1 **”。**

2．**方式** 1

串行口定义为方式 1 **时，传送一帧数据为** 10 **位，其中包括** 1 **位起始位、**8 **位数据位(先低位后高位)和** 1 **位停止位。方式** 1 **的波特率可变，波特率=(**T1 **的溢出率)×**2^{SMOD}/32**。**

(1) **方式** 1 **发送。数据由** TxD **端输出，发送一帧信息为** 10 **位，其中包括** 1 **位起始位、**8 **位数据位和** 1 **位停止位。数据字节写入** SBUF **后，便启动串行口发送器发送，当发送数据完成后，中断标志** TI **置“** 1 **”。**

(2) **方式** 1 **接收。数据从** RxD **端输入，在** REN **置“** 1 **”后，就允许接收器接收。接收器以波特率** 16 **倍的速率采样** RxD **端电平。当采样到** RxD **引脚上“** 1 **”到“** 0 **”的跳变时，**

启动接收器接收并复位内部的 16 分频计数器，以便实现同步。计数器的 16 个状态把一位的时间分成 16 等份，在每位时间的第 7、8 和 9 计数状态，位检测器采样 RxD 的值，接收的值是 3 次采样中至少有两次是相同的值，以排除噪声干扰。若起始位接收的值不是“0”，则起始位无效，复位接收电路；若检测到起始位有效时，则将接收的数据输入移位寄存器，开始接收本帧其余数据信息。当 RI=0，同时接收到停止位为“1”(SM2=0)时，停止位进入 RB8，置“1”中断标志 RI。若以上两个条件中的任一条件得不到满足，则所有接收信息将丢失，因此中断标志 RI 必须在中断服务程序中由用户清零。通常串行口以方式 1 工作时，SM2 置为“0”。

3．方式 2

串行口定义为方式 2 时，串行口被定义为 9 位异步通信接口。传送一帧信息为 11 位，其中包括 1 位起始位、8 位数据位(先低位后高位)、可程控为“1”或“0”的第 9 位数据位和 1 位停止位。方式 2 的波特率是固定的，波特率为 $2^{SMOD}/64*f_{osc}$。具体帧结构见图 5.1.1。

图 5.1.1　方式 2 帧结构

(a) 无空闲位字符帧；(b) 有空闲位字符帧

(1) 方式 2 发送。数据由 TxD 端输出，发送一帧信息为 11 位，附加的第 9 位数据是 SCON 中的 TB8，可由软件置位或清零，可作为多机通信中的地址、数据标志，也可作为数据的奇偶校验位。可以用如下指令中的一条来完成：

```
SETB TB8
CLR TB8
```

CPU 执行一条写入发送缓存器指令(例如“MOV SBUF，A”)，就启动发送器发送，发送完一帧信息，置 TI 中断标志为“1”。

(2) 方式 2 接收。数据由 RxD 端输入，REN 被置“1”后，接收器开始以波特率 16 倍的速率采样 RxD 电平，检测到 RxD 端由高到低的负跳变时，启动器启动接收，复位内部 16 分频计数器，以实现同步。计数器的 16 个状态把一位时间分成 16 等份，在每位时间的

7、8、9 个状态位检测采样 RxD 的值：若接收的值不是“0”，则起始位无效并复位接收电路；当采样到 RxD 端从“1”到“0”的跳变时，确认位起始位有效，接着开始接收本帧其余信息。接收完一帧信息后，在 RI=0、SM2=0 时，或接收到的第 9 位数据为“1”时，8 位数据装入接收缓冲器，第 9 位数据装入 SCON 的 RB8，并置位中断标志 RI。若不满足上述两个条件，接收到的信息将丢失。

4．方式 3

串行口被定义为方式 3 时，为波特率可变的 9 位异步通信方式。除了波特率外，方式 3 与方式 2 相似。方式 3 的波特率为：(T1 溢出率)×2^{SMOD}/32。

5.1.3 波特率的设计

波特率的设计表如表 5.1.2 所示。

表 5.1.2 波特率设计表

工作方式	SMOD 设定	波特率	操 作 码
0	0/1	f_{osc}/12	MOV SCON， #00H
2	0	f_{osc}/64	MOV SCON， #80H CLR SMOD
	1	f_{osc}/32	MOV SCON， #80H SETB SMOD
1，3	0	T1 溢出率/32	MOV SCON， #40H (方式 1) MOV SCON， #0C0H(方式 3) CLR SMOD
	1	T1 溢出率/16	MOV SCON， #40H (方式 1) MOV SCON， #0C0H(方式 3) SETB SMOD

 说明：

8031 系列单片机的工作频率为系统时钟频率的 12 分频。

若采用定时器 1 作为波特率发生器(当然定时器 0 也可以)，其工作原理是：设置定时器的初值→决定定时器溢出率(计数值溢出)→将溢出率分频→得到波特率(控制了 UART 发送或接收一位数据的时间)。根据串行口的四种工作方式得波特率设计表(表 5.1.2)和 T1 溢出率计算表(表 5.1.3)。注意此时需 TMOD 寄存器计数/定时选择位 C/T=0。

表 5.1.3 T1 溢出率计算表

定时器 1 工作方式	T1 溢出率
0	(f_{osc}/12)/(2^{13} －初值)
1	(f_{osc}/12)/(2^{16} －初值)
2，3	(f_{osc}/12)/(2^{8} －初值)

对 52 系列单片机，可通过式 T2CON 选择定时器 2 作为串行口波特率发生器。当 TCLK=1 时，定时器 2 为发送波特率发生器；当 RCLK=1 时，定时器 2 为接收波特率发生器；当 TCLK=1、RCLK=1 时，定时器 2 同时作发送和接收波特率发生器。此时波特率与 PCON 寄存器无关，计算公式为：

$$\text{波特率}=(f_{osc}/12)/[2^{16}-(\text{RCAP2H，RCAP2L})]$$

其中，RCAP2H 和 RCAP2L 寄存器构成一个 16 位寄存器，为自动重装入的数值。

5.1.4　波特率误差及选择

波特率是串行通信中的一个重要参数，通信双方波特率的一致性直接影响数据传输的正确性。一般情况下，对于 11 位的串行数据帧(带奇偶校验位)，通信双方波特率的最大误差不应超过 4.5%。

由上节波特率的计算方法可得，当系统时钟为 6 MHz 时，若要产生 4800 b/s 的波特率，采用定时器 1、工作方式 2 自动装载，SMOD=0 时，定时常数为：

$$n = 2^8 - \frac{f_{\text{ocs}}}{32 \times 12 \times 4800} \approx 252.75$$

因 n 必为整数，四舍五入后为 n=253(FDH)，因而，实际的波特率为：

$$B = \frac{f_{\text{ocs}}}{32 \times 12 \times (2^8 - n)} = \frac{6.000 \times 10^6}{32 \times 12 \times 3} \approx 5208$$

波特率误差为：

$$\Delta B = \frac{5208 - 4800}{4800} = 8.5\%$$

当用 6 MHz 晶振设定 1200 b/s 的波特率时，其误差仅为 0.1%。而使用 11.059 MHz 晶振时，设定各标准波特率误差都很小。

在 51 系列单片机之间进行通信时，若各单片机的晶振和定时常数相同，尽管实际波特率可能与设定值相差较远，但双方之间的误差却很小，只是出现由晶振引起的误差。

当单片机之间的晶振和定时常数不同，或单片机同 DSP 处理器以及 PC 机之间进行通信时，必须考虑到波特率误差问题。这时，可能需要用到非标准波特率，表 5.1.4 列出了使用 6 MHz 晶振时，误差较小的波特率。

表 5.1.4　供选择的部分非标准波特率(SMOD=0)

设定波特率	N	相对误差
1302	F4H	0.5%
2232	F9H	0.7%
3125	FBH	0.4%
5026	FDH	0.5%
1200	F3H	0.1%

经过计算，还可得到 SMOD=1 时的各波特率的误差情况。

5.1.5　波特率的自动检测

在分布式多波特率通信系统中，常常要求从设备在软件上做到自身的波特率能随主设备自动调整，使系统适应性更强，更具智能化。波特率自动检测的范围仅限于标准波特率。

常见的从设备实现波特率自动检测的方法有三种：

(1) 从设备启动通信程序后，逐一选择标准波特率，向主设备发送某个事先商定的握

手码，直到收到主设备发回的确认码。

(2) 利用串行异步通信每一帧起始位的低电平和停止位的高电平，用定时器记录每帧长度，由此确定波特率。

(3) 利用主设备发送某一特殊码型的字符，如回车键“Enter”，从设备收到的码值会随主设备的波特率不同而不同。

下面介绍如何利用方法(2)给出编程方法和程序清单，从设备为 89C51 单片机。

通信开始时，主设备发出一个测试字符。当单片机检测到起始位(下降沿)时，启动计数器 T0。为了记录停止位，单片机必须记录串行数据的每一个上跳沿，直至在相对很长的时间内都不再有上跳沿(用 T0 计数器的溢出来指示)为止，那么，最后一个被记录的上跳沿即为停止位。将此计数值与事先计算好的波特率索引表的标准波特率对应值相比较，从而得到正确的波特率。需要注意的是，为确保 T0 溢出时所记录的最后一个上跳沿为停止位，主设备发送测试字符后应使传输线保持一段高电平。

对每种波特率，可通过下式计算出对应的最大计数器预置值，存入波特率索引表(BdTab)中：

$$\text{最大定时器预置值} = \frac{\text{晶振频率(MHz)}}{\text{波特率(b/s)}} \times \frac{5}{12}$$

以串行口工作方式 1(8 位数据位，无奇偶校验，1 位停止位)为例，上式推导如下：

$$\text{最大定时器预置值} = \frac{\text{最小识别时间}}{\text{机器周期}}$$

$$\text{最小识别时间} = \frac{\text{识别位数} \times \text{字节时间}}{9}$$

式中，机器周期=12/晶振频率；字节时间=9/波特率。

例 5.1.1

程序清单如下：

```
        ；预定义
        RX BIT P3.0                ；串行数据接收端
        CHARH DATA 30H             ；保存 TH0 值
        CHARL DATA 31H             ；保存 TL0 值
        BDRT   DATA   32H          ；保存波特率最终检测值
        DISPLAY EQU PI             ；显示调试结果
        ；复位及中断向量
        ORG 0030H
BEGING:                            ；检测波特率值
        LCALL AUTOBAUD             ；送波特率索引值
        MOV DISPLAY，BDRT
        SJMP BEGING

        ；波特率检测子程序
        ；返回 ACC=波特率索引指针
```

```
AUTOBAUD:
        MOV TMOD，#01H          ；置 T0 工作方式 1
        MOV TH0，#0             ；初值写入 T0
        MOV TL0，#0
        MOV TCON，#0
        MOV CHARH，#0           ；初始化寄存单元
        MOV CHARL，#0
CHK0:
        JB RX，CHK0             ；等待字符起始位
        SETB TR0                ；启动 T0
CHK1:
        JB TF0，CHK3            ；T0 溢出否？
        JNB RX，CHK1            ；检测串行数据上升沿
        MOV CHARH，TH0          ；“捕获”在上升沿的 T0 值
        MOV CHARL，TL0
CHK2:
        JB TF0，CHK3            ；T0 溢出否？
        JB RX，CHK2             ；检测串行数据下降沿
        SJMP CHK1               ；重复采样
CHK3:
        CLR TR0                 ；T0 溢出处理
        CLR TF0                 ；清标志，停止计数
        MOV BDRT，#19           ；建立索引表指针
LOOP:
        MOV A，BDRT
        MOV DPTR，#BDTAB
        MOVC A，@A+DPTR         ；查表比较
        DEC BDRT
        CJNE A，CHARH，CMP1     ；检查结果范围
        SJMP CMPLOW             ；高字节相等，检查低字节
CMP1:
        JC MATCH                ；若表中值<计数值，匹配
        DJNZ BDRT，LOOP         ；表是否比较完毕?
        SJMP MATCH
CMPLOW:
        MOV A，BDRT
        MOVC A，@A+DPTR         ；查表比较
        CJNE A，CHARL，CMP2     ；检查数据范围
        SETB C                  ；若相等，则匹配
```

```
CMP2:
        JC MATCH                    ；若 A<结果低字节，置 C
        DJNZ BDRT，LOOP             ；表是否比较完毕？
MATCH:
        MOV A，BDRT                 ；比较完
        CLR C                       ；取最终波特率索引值
        RR A
        MOV BDRT，A                 ；保存
        RET
        ；索引表保存标准波特率所对应的计数值
        ；晶振频率为 11.0592 MHz
        ；排列顺序：LSB，MSB

BDTAB:
 DB 2CH，00H                        ；0 – 越限，值太小
        DB 78H，00H                 ；1 – 波特率 38400
        DB 0F0H，00H                ；2 – 波特率 19200
        DB 0E0H，01H                ；3 – 波特率 9600
        DB 0C0H，03H                ；4 – 波特率 4800
        DB 80H，07H                 ；5 – 波特率 2400
        DB 00H，0FH                 ；6 – 波特率 1200
        DB 00H，1EH                 ；7 – 波特率 600
        DB 00H，3CH                 ；8 – 波特率 300
        DB 00H，78H                 ；9 – 越限，值太大
END
```

5.2 单片机点到点通信的实例

5.2.1 供测试的简单通信程序

1. **发送单个字符，供测试硬件线路和波特率**

例 5.2.1 **将**#0AAH **握手信号由甲机发送到乙机，甲乙机设置相同：波特率均为** 1200 b/s，**晶振** f_{osc}=11.059 MHz，**串口工作方式** 1，**定时器** T1 **工作方式** 2。

甲机发送程序：

```
        MOV TMOD，#20H              ；定时器 T1 工作方式 2
        MOV TH1，#0E8H
        MOV TL1，#0E8H              ；赋定时器初值
        SETB TR1                    ；启动 T1
```

```
MOV SCON，#50H          ；串口工作方式 1，允许接收
MOV A，#0AAH
MOV SBUF，A             ；发送
JNB TI，$               ；等待发送结束
CLR TI                  ；允许再次发送
SJMP $                  ；程序结束
```

乙机接收程序：

```
MOV TMOD，#20H          ；定时器 T1 工作方式 2
MOV TH1，#0E8H
MOV TL1，#0E8H          ；赋定时器初值
SETB TR1                ；启动 T1
MOV SCON，#50H          ；串口工作方式 1，允许接收
JNB RI，$               ；等待接收
MOV A，SBUF             ；读出接收的数据
CLR RI                  ；允许再次接收
SJMP $                  ；程序结束
```

2. **连续发送和接收数据，供测试通信稳定性，并介绍和校验方法**

 **说明：**

A 机发送，B 机接收，均为查询方式。协议如下：约定双方波特率为 1200 b/s，晶振为 11.059 MHz，串行口工作方式 1，设 T1 工作模式 2，SMOD=0，TH1=0E8H。

A 机开始发送时，先发#0AAH，B 机收到后回答#0BBH，表示接收。

当 A 机收到#0BBH 后，开始发送数据，每发送一次求一次“检查和”。“检查和”是发送的每一个字节数据都累加到一个寄存单元中去，累计过程中多次发生向高位进位(溢出丢失)，最后在累加单元中剩余的结果即为累加和。设定数据块长为 16 字节，起始地址为 30H，一个数据块发完再发出“累加和”。

B 机接收数据并转存到数据区，起始地址为 30H。同时，每接收一次也计算一次“检查和”，当一个数据块接收完毕后，再接收从 A 机发来的“检查和”，并将它与 B 机计算的“检查和”进行比较。若两者相等，说明接收正确，B 机回答#00H；若两者不等，B 机回答#0FFH，要求重发。

A 机收到#0FFH 的答复结束发送，否则重复发送此数据块。

例 5.2.2　A 机发送子程序清单。

```
        0RG 4000H
ASTART:
        CLR EA
        MOV TMOD，#20H          ；设 T1 为定时模式 2
        MOV TH1，#0E8H          ；设定时器初值
        MOV TL1，#0E8H
```

```
        M0V PCON，#00H          ；波特率为 1200 b/s
        SETB TR1                ；启动定时器
        M0V SCON，#50H          ；设串行口为方式 1，允许接收
ATT1:
        M0V SBUF，#0AAH         ；发联络信号 AA
AWAIT1:
        JBC TI，ARR1
        SJMP AWAIT1
ARR1:
        JBC RI，ARR2            ；等待 B 机应答
        SJMP ARR1
ARR2:
        MOV A，SUBF

        XRL A，#0BBH            ；判断 B 机准备信号(BB)
        JNZ ATT1                ；B 机未准备好，继续联络
ATT2:
        MOV R0，#30H            ；建立要发送的数据块地址指针
        MOV R7，#10H            ；块长度计数初值 16
        MOV R6，#00H            ；清检查和单元
ATT3:
        MOV SBUF，@R0           ；发送一个数据字节
        MOV A，R6
        ADD A，@R0              ；求检查和
        M0V R6，A               ；保存检查和
        INC R0
AWAIT2:
        JBC TI，ATT4
        SJMP AWAIT2
ATT4:
        DJNZ R7，ATT3           ；整个数据收送完了吗?
        MOV SBUF，R6            ；发送检查和
AWAIT3:
        JBC TI，ARR3
        SJMP AWAIT3
ARR3:
        JBC RI，ARR4            ；等待 B 机应答
        SJMP ARR3
ARR4:
```

```
        MOV A，SBUF
        JNZ ATT2                    ；若 B 机回答 00 返回，否则重发
AEND：
    RET
```

例 5.2.3　B 机接收子程序清单。

```
        0RG 4000H
BSTART:
        CLR EA
        MOV TM0D，#20H              ；设 T1 为定时模式 2
        MOV TH1，#0E8H
        MOV TL1，#0E8H
        M0V PCON，#00H              ；波特率为 1200 b/s
        SETB TR1
        MOV SCON，#50H              ；设串行口为方式 1，允许接收
BRR1:
        JBC RI，BRR2                ；等待联络信号
        SJMP BRR1
BRR2:
        MOV A，SBUF
        XRL A，#0AAH                ；判断 A 机请求信号(AA)
        JNZ BRR1
BTT1:
        MOV SBUF，#0BBH             ；是 A 机请求，发应答信号"BB"
BWAIT1:
        JBC TI，BRR3
        SJMP BWAIT1
BRR3:
        MOV R0，#30H                ；设数据地址指针
        M0V R7，#10H                ；块长度计数初值
        MOV R6，#00H                ；检查和单元初始化
BRR4:
        JBC RI，BRR5
        SJMP BRR4
BRR5:
        MOV A，SBUF
        MOV @R0，A                  ；将接收的数据转存 30H 开始的 RAM
        INC R0
        ADD A，R6                   ；求检查和
        MOV R6.A
```

```
            DJNZ R7，BRR4                ；数据块接收完了吗?
BWAIT2:
            JBC RI，BRR6                 ；接收 A 机检查和
            SJMP BWAIT2
BRR6:
            MOV A，SBUF
            XRL A，R6                    ；比较检查和
            JZ BEND                      ；检查和相等
            MOV SBUF，#0FFH              ；检查和不相等，发错误标志
BWAIT3:
            JBC TI，BRR3                 ；转重新接收
            SJMP BWAIT3
BEND:
            MOV SBUF，#00H               ；接收正确，向 A 机回发 00
        RET
```

3. 可能出现的问题

在实际中，通信不成功的原因可以归为两类：线路问题和软件问题。下面就检查方法进行说明。

如怀疑是线路问题，则按下述方法进行检查：

(1) 通信线路是否有断路或短路现象；

(2) 通信点之间的中继器或驱动器(如 MAX232 或 75176)工作是否正常(如果有的话)；

(3) 本机的 RXD 是否连接到对方的 TXD 端(特别是用了 RS－232 串口 9 芯或 25 芯标准接头的时候)；

(4) 如果是与 PC 机通信，还应检查 PC 机的串口是否正常(PC 机串口比较容易损坏)；

(5) 检查单片机的工作频率，因为，个别情况下晶振的振荡频率会与标称值相差较大。具体操作为测量在无 MOVX 指令时的 ALE 信号是否为晶振频率的 6 分频(基本 51 系列单片机为 6 分频，其它扩充型号的分频数请参阅其数据手册)。

如怀疑是软件问题，则按下述方法进行检查：

(1) 波特率是否一致，以及误差是否在允许范围之内(<5%)；

(2) 是否设置了 SMOD 位，中断允许位是否设置(如果用了中断方式)；

(3) 数据格式是否一致，有无校验位，起始位和停止位的个数是否相同；

(4) 清除 TI、RI 标志位后的延时是否足够。一般的通信程序都采用半双工的工作方式：发送时，检测 TI 是否建立起来，当 TI 为高电平后关闭发送功能，转为接收功能；接收时，检测 RI 是否建立起来，当 RI 为高电平后，接收完毕，又可以转为发送。

在实际情况中会出现传输的数据时对时错的现象，此时可以查看单片机的时序，如图 5.2.1 所示。

单片机在串行口发送数据时，只要将 8 位数据位传送完毕，TI 标志即建立，但此时应发送的第 9 位数据位(若发送地址帧时)和停止位尚未发出。如果在这时关闭发送控制，势必

图 5.2.1　串行中模式 3 时序图

造成发送帧数据不完整。如果单片机通信采用较高的波特率，几条操作指令的延时就可能超过 2 位(或 1 位)数据的发送时间，问题或许不会出现。但是，如果采用较低波特率，如 9600 b/s，发送一位数据需 100 μs 左右，单靠几条操作指令的延时远远不够，问题就明显地暴露出来。接收数据时也同样如此，单片机在接收完 8 个数据位后就建立起 RI 信号，但此时还未接收到第 9 位数据位和停止位。所以，接收端必须延时大于 2 位数据位的时间(1 位数据位时间=1/波特率)，再作应答，否则会发生总线冲突。

标志(RI、TI)建立与清除之间的延时 T 可参考如下取值：T>2×(1/波特率)，可以选取 T=2.5×(1/波特率)。

5.2.2　较完整的通信程序

假设用 A 机发送信号，用 B 机接收信号，且均为查询方式。协议如下：约定双方波特率为 2400 b/s，晶振为 6 MHz，串行口工作方式 3，奇偶校验，T1 工作模式 2。

由 5.1.5 节公式，得：

$$n = 2^8 - \frac{f_{\text{osc}} \times 2^{\text{SMOD}}}{\text{波特率} \times 12 \times 32}$$

将 f_{osc}=6×10^6 Hz，SMOD=1，波特率=2400 b/s 代入，进行计算得 n=242.98≈243=F3H。将计算得到的 n 又回代到公式中，得实际波特率为 2403.85 b/s，误差为 0.16%。

A 机将外部 RAM 4000H～407FH 单元的内容向 B 机发送。B 机对接收的每字节数据偶校验，校验位放入 TB8 中：校验正确，发回#00H，并将数据存入 4000H 为首址的外部 RAM

中；校验不正确，发回#0FFH，表示错误，要求重发。A 机收到#00H 后，发送下一个字节；收到#0FFH 后，重新发送原数据，直至正确。

例 5.2.4　A 机源程序。

主程序

```
        ORG 0000H
        LJMP MAIN              ；上电，转向主程序
        ORG 0023H              ；串行口的中断入口地址
        LJMP SERVE1            ；转向 A 机中断服务程序
        ORG 0030H              ；主程序
MAIN:
        MOV TM0D，#20H         ；设 T1 工作方式 2
        MOV TH1，#0F3H         ；赋计数初值
        MOV TL1，#0F3H         ；赋计数值
        SETB TR1               ；启动定时器 T1
        MOV PCON.#80H          ；设 SMOD=1
        MOV SCON，#0D0H        ；置串行口方式 3，允许接收
        MOV DPTR，#4000H       ；置数据块首址
        MOV R0，#80H           ；置发送字节数初值
        SETB ES                ；允许串行口中断
        SETB EA                ；CPU 开中断
        MOVX A，@DPTR          ；取第一个数据发送
        MOV C，P
        MOV TB8，C             ；奇偶标志送 TB8
        MOV SBUF，A            ；发送数据
        SJMP $                 ；等待中断
```

中断服务程序

```
SERVE1:
        JNB RI，LOOP           ；是接收中断，清除 RI，转入接收 B 机的应答信息
        CLR TI                 ；是发送中断，清除此中断标志
        SJMP ENDT
LOOP:
        M0V A，SBUF            ；取 B 机的应答信息
        JZ    LOOP1            ；判应答信号是#00H
        MOVX A，@DPTR          ；否则 A 机重发
        MOV C，P
        MOV TB8，c
        MOV SBUF，A            ；A 机重发原数据
        SJMP ENDT
LOOPl:
```

```
        INC DPTR                ；修改地址指针，准备发送下一个数据
        MOV X A，@DPTR
        MOV C，P
        MOV TB8，C
        MOV SBUF，A             ；发送
        DJNZ R0，ENDT           ；数据块未发送完，返回继续发送
        CLR ES                  ；全部发送完，禁止串行口中断
ENDT:
        RETI                    ；中断返回
    END
```

例 5.2.5　B 机源程序。

主程序

```
        ORG 0000H
        LJMP MAIN               ；上电，转向主程序
        ORG 0023H               ；串行口的中断入口地址
        LJMP SERVE2             ；转向 B 机中断服务程序
        ORG 2000H               ；主程序
MAIN:
        MOV TMOD，#20H          ；设 T1 工作方式 2
        MOV TH1，#0F3H          ；赋计数初值
        MOV TL1，#0F3H          ；赋计数值
        SETB TR1                ；启动定时器 T1
        MOV PCON，#80H          ；设 SMOD=1
        MOV SCON，#0D0H         ；置串行口方式 3，允许接收
        MOV DPTR，#4000H        ；置数据区首址
        MOV R0，#80H            ；置接收字节数初值
        SETB ES                 ；允许串行口中断
        SETB EA，CPU            ；开中断
        SJMP $                  ；等待中断
```

中断服务程序

```
SERVE2:
        JNB RI，LOOP            ；是接收中断，清除此标志，转 LOOP 接收
        CLR TI                  ；是发送中断，清除此标志，中断返回
        SJMP ENDT
LOOP:
        MOV A，SBUF             ；接收(读入)数据
        MOV C，P                ；奇偶标志送 C
        JC LOOP1                ；为奇数，转 LOOP1
```

```
            ORL C，RB8                ；为偶数，检测 RB8
            JC LOOP2                  ；奇偶校验错，转 LOOP2
            SJMP LOOP3
LOOP1:
            ANL C，RB8                ；检测 RB8
            JC LOOP3                  ；奇偶校验正确，转 LOOP3
LOOP2:
            MOV A，#0FFH
            MOV SBUR，A               ；发送“不正确”应答信号
            SJMP ENDT
LOOP3:
            MOVX @DPTR，A             ；存放接收数据
            MOV A，#00H
            MOV SBUF，A               ；发送"正确"应答信号
            INC DPTR                  ；修改数据区指针
            DJNZ R0，ENDT             ；数据块尚未接完，返回
            CLR ES                    ；所有数据接收完毕，禁止串行中断
ENDT:
            RETI                      ；中断返回
        END
```

5.3 单片机多机通信实例

5.3.1 多机通信原理

主从式多机通信是多机通信系统中最简单的一种。在主从式多机系统中，只有一台主机，但从机可以有多台。主机发送的信息可以传送到所有的从机或指定的从机，从机发送的信息只能为主机所接收。各从机之间不能直接通信。主机通常由 PC **机充任，但也可由单片机担当；从机通常为单片机，如图** 5.3.1 **所示。**51 **单片机用于多机通信时，必须在方式** 2 **或方式** 3 **下工作，作主机的** 51 **单片机的** SM2 **应设定为** 0**，作从机的** SM2 **应设定为** 1**。主机发送并为从机接收的信息有两类：一类是地址，当串行数据第** 9 **位为“** 1 **”时，指示需要与主机通话的从机地址；另一类是数据，当串行数据第** 9 **位为“** 0 **”时，指示需与主机交换的数据。由于所有从机的** SM2＝1**，故每个从机总能在** RI＝0 **时收到主机发来的地址，并进入各自的中断服务程序。在中断服务程序中，每台从机把接收到的从机地址和它的本机地址(系统设计时所分配)进行比较。所有经过比较后地址不相等的从机均自动从各自的中断服务程序中退出(**SM2 **仍为** 1)**，只有比较成功的从机才是被主机寻址通信的从机。被寻址从机在程序中使** SM2＝0**，以便接收随之而来的数据或命令(**RB8＝0)**。上述过程进一步归结如下：**

图 5.3.1　多机通信示意图

(1) 主机的 SM2＝0；所有从机的 SM2=1，以便接收主机发来的地址。

(2) 主机给从机发送地址时，第 9 数据位上发送“ 1 ”，以指示从机接收这个地址。

(3) 所有从机在 SM2＝1、RB8＝1 和 RI＝0 时，接收主机发来的从机地址，进入相应中断服务程序，并和本机地址比较以确认是否为被寻址从机。

(4) 被寻址从机通过指令清除 SM2，以正常接收数据，并向主机发回接收到的从机地址，供主机核对。未被寻址从机保持 SM2=“ 1 ”，并退出各自中断服务程序。

(5) 完成主机和被寻址从机之间的数据通信，被寻址从机在通信完成后重新使 SM2＝1，并退出中断服务程序，等待下次通信。

5.3.2　主从式多机通信实例

在多机通信中，主机通常把从机地址作为 8 位数据(第 9 数据位为 1)发送，因此，51 单片机构成的多机通信系统最多允许 255 台从机(地址为 00H～FEH)，FFH 作为一条控制命令由主机发送给从机，以便使被寻址从机的 SM2＝1。

下面给出一个主机和从机的通信程序，波特率设为 1200 b/s。

在多机通信中，主从机之间除传送从机地址和数据外，还应传送一些供主机或从机识别的命令和状态字。

本例中设定的两条控制命令为：① 00H——主机发送，从机接收命令；② 01H——从机发送，主机接收命令。这两条命令均以数据形式发送(即第 9 位数据为 0)。

本例中设定的从机状态字由被寻址从机发送，为主机所接收，用于指示从机的工作状态，其格式如图 5.3.2 所示。

1. 主机程序

主机程序由主机主程序和主机通信子程序组成。主机主程序用于定时器 T1 初始化、串行口初始化和传递主机通信子程序所需入口参数。主机通信子程序用于主机和从机间一个数据块的传送。主机主程序、通信子程序流程图分别如图 5.3.3 及图 5.3.4 所示。

图 5.3.2 从机状态字格式

说明：

程序中所用寄存器分配如下：

R0——存放主机发送数据块始址；

R1——存放主机接收数据块始址；

R2——存放被寻址的从机地址；

R3——存放主机发出的命令；

R4——存放发送数据块长度；

R5——存放接收数据块长度。

例 5.3.1

主机程序：

```
        ORG   0030H
START: MOV TMOD，#20H
        ；定时器 Tl 为方式 2
        MOV TH1，#0F4H
        ；波特率为 1200 b/s
        MOV TL1，#0F4H
        SETB TR
        ；启动 T1 工作
        MOV SCON，#0D8H
        ；串行口工作方式 3，允许接收，SM2=0，TB8＝1
        MOV PCOH，#00H
        MOV R0，#04H            ；接收数据块始址
        MOV R1，#20H            ；发送数据块始址
        MOV R2，#SLAVE          ；被寻址从机地址
        MOV R3，#00H/01H        ；若为 00H，则主机发、从机收命令
                                ；若为 01H，则从机发、主机收命令
        MOV R4，#20H            ；发送数据块长度
        MOV R5，#20H            ；接收数据块长度
        ACALL MCOMMU            ；调用主机通信子程序
        SJMP $                  ；结束
```

图 5.3.3 主机主程序流程图

图 5.3.4　**通信子程序流程图**

例 5.3.2

主机通信子程序：

```
        ORG 0130H
MCOMMU:
        MOV A，R2          ；从机地址送 A
        MOV SBUF，A        ；发送从机地址
        JNB RI，$          ；等待接收从机应答地址
        CLR RI             ；从机应答后清 RI
        MOV A，SBUF        ；从机应答地址送 A
        XRL A，SBUF        ；核对两个地址
        JZ MTXD2           ；相符，则转 MTXD2
MTXD1:
```

```
        MOV SBUF，#0FFH      ；发送从机复位信号
        SETB TB8             ；地址帧标志送 TB8
        SJMP MCOMMU          ；重发从机地址
MTXD2:
        CLR TB8              ；准备发送命令
        MOV SBUF，R3         ；送出命令
        JNB RI，$            ；等待从机应答
        CLR RI               ；从机应答后清 RI
        MOV A，SBUF          ；从机应答命令送 A
        JNB ACC.7，MTXD3     ；核对命令后无错，则命令分类
        SJMP MTXD1           ；若命令收错，则重新联络
MTXD3:
        CJNE R3，#00H，MRXD  ；若为从机发送主机接收命令，则 MRXD
        JNB ACC.0，MTXD1     ；若从机接收未就绪，则重新联络
MTXD4:
        MOV SBUF，@R0        ；若从机接收就绪，则开始发送
        JNB TI，$            ；等待发送结束
        CLR TI               ；发送结束后清 TI
        INC R0               ；R0 指向下一个发送数据
        DJNZ R4，MTXD4       ；若数据块未发完，则继续
        RET
MRXD:
        JNB ACC.1，MTXD1     ；若为从机发送未就绪，则重新联络
MRXD1:
        JNB RI，$            ；等待接收完毕
        CLR RI               ；接收到一帧后清 RI
        MOV A，SBUF          ；收到的数据送 A
        MOV @R1，A           ；存储数据
        INC R1               ；接收数据区指针加 1
        DJNZ R5，MRXD1       ；若未接收完，则继续
    RET
END
```

2. **从机程序**

从机程序由从机主程序和从机中断服务程序组成。从机主程序用于定时器 T1 **初始化、串行口初始化和中断初始化。从机中断服务程序用于对主机的通信。从机主程序、中断服务程序框图分别如图** 5.3.5 **和图** 5.3.6 **所示。现分述如下。**

图 5.3.5　从机主程序框图

图 5.3.6　从机中断服务程序框图

例 5.3.3

从机主程序：

```
        ORG 0030H
START:
        MOV TMOD，#20H      ；定时器 T1 方式 2
        MOV TH1，#0F4H      ；波特率为 1200 b/s
        MOV TL1，#0F4H
        SETB TR1            ；启动 T1 工作
```

```
        MOV SCON，#0F8H      ；串行口工作方式 3，允许接收，SM2=1，TB8=1
        MOV PCON，#00H
        MOV R0，#20H R0      ；指向发送数据块始址
        MOV R1，#40H         ；R1 指向接收数据区始址
        MOV R2，#20H         ；发送数据块长度
        MOV R3，#20H         ；接收数据块长度
        SETB EA              ；开 CPU 中断
        SETB ES              ；允许串行口中断
        CLR RI                   ；清 RI
    SJMP $
```

由于从机串行口设定为方式 3，SM2＝1 和 RI＝0，且串行口中断已经开放，因此从机的接收中断总能被响应。在中断服务程序中，SLAVE 是从机的本机地址，F0H(即 PSW.5)为本机发送就绪位地址(即 F0H 中为 1 表示从机发送准备就绪)，PSW.1 为本机接收就绪状态位(即 PSW.1＝1 为本机已推备好接收)。从机中断服务程序流程如图 5.3.6 所示。

寄存器分配为:

R0——存放发送数据块始址；

R1——存放接收数据块始址；

R2——存放发送数据块长度；

R3——存放接收数据块长度。

例 5.3.4　从机中断服务程序。

```
ORG   0023H
        SJMP SINTSBV
ORG 0100H
SINTSBV:
        CLR RI               ；接收到地址后清 RI
        PUSH ACC             ；保护 A 于堆栈
        PUSH PSW             ；保护 PSW 于堆栈
        MOV A，SBUF          ；接收的从机地址送 A
        XRL A，#SLAVE        ；和本机地址核对
        JZ SRXD1             ；若是呼叫本机，则继续
RETURN:
        POP PSW              ；若不是呼叫本机，则恢复 PSW
        POP ACC              ；恢复 ACC
        RETI                 ；中断返回
SRXD1:
        CLR SM2              ；准备接收数据／命令
        MOV SBUF，#SLAVE     ；发回本机地址以供核对
        JNB RI               ；等待接收主机发来的数据／命令
        CLR RI               ；接收到后清 RI
```

```
        JNB RB8              ；是数据／命令，则继续
        SETB SM2             ；若是恢复信号，则令 SM2＝1
        SJMP RETURN          ；返回主程序
SRXD2:
        MOV A，SBUF          ；接收命令送 A
        CJNE A，#02H，NEXT   ；命令是否合法？
NEXT:
        JC SRXD3             ；若命令合法，则继续
        CLR TI               ；若命令不合法，则清 TI
        MOV SBUF，#80H       ；发送 ERR=1 的状态字
        SETB SM2             ；令 SM2=1
        SJMP RETURN          ；返回主程序
SRXD3:
        JZ SCHRX             ；若为接收命令，则转 SCHRX
        JB F0，STXD          ；若本机发送就绪，则转 STXD
        MOV SBUF，#00H       ；若本机发送未就绪，则发 TRDY=0
        SJMP RETURN          ；返回主程序
        MOV SBUF，#02H       ；发送 TRDY=1 的状态字
        JNB TI，$            ；等待发送完毕
        CLR TI               ；发送完后清 TI
LOOP1:
        MOV SBUF，@R0        ；发送一个字符数据
        JNB   TI，$          ；等待发送完毕
        CLR TI               ；发送完毕后清 TI
        INC   R0             ；发送数据块始址加 1
        DJNZ R2，LOOP1       ；字符未发完，则继续
        SETB SM2
        SJMP RETURN          ；返回
SCHRX:
        JB PSW.1，SRXD       ；若本机接收就绪，则转 SRXD
        MOV SBUF，#00H       ；若本机接收未就绪，则发 RRDY=0
        SETB SM2
        SJMP RETURN          ；返回主程序
SRXD:
        MOV SBUF，#01H       ；发出 RRDY＝1 状态字
LOOP2:
        JNB RI，$            ；接收一个字符
        CLR RI               ；接收一帧字符后清 RI
        MOV @R1，SBUF
        INC R1               ；接收数据块指针加 1
```

```
        DJNZ R3，LOOP2          ；若未接收完，则继续
        SETB SM2
        SJMP RETURN             ；返回主程序
END
```

5.3.3 C51 语言通信的例子

一个主机与多个从机进行单工通信，主机发送，从机接收。主机先向从机发送一帧地址信息，然后再发送 10 **位数据信息。从机接收主机发来的地址，并与本机的地址相比较：若不相同，则仍保持** SM2＝1 **不变，自动抛弃数据帧；若地址相同，则使** SM2＝0，**准备接收主机发来的数据信息，直至接收完** 10 **位数据。通信双方均采用** 11.0592 MHz **的晶振，用** T1 **产生** 9600 b/s **的波特率，采用中断方式传送数据。从机的地址为** 0～255 **的编码。实际通信中，还应考虑通信协议，为简单起见下面的程序中未予考虑。主机发送程序，从机接收程序文件名为** RECEIVE.C。

例 5.3.5

主机发送程序(文件名为 SEND.C)。

```
#include<reg51.h>
#define COUNT 10                    /*定义发送缓冲区大小*/
#define NODE_ADDR 64                /*定义目的节点位置*/
unsigned char buffer[COUNT]；       /*定义发送缓冲区*/
int pointer；                       /*定义当前指针*/
main( )
{
/*发送缓冲器初始化*/
    while(pointer<COUNT)
    {
        buffer[pointer]='A'；
       pointer++；
    }
/*初始化串行口和波特率发生器*/
    SCON=0xc0；
    TMOD=0x20；
    THl=0xfd；
    TRl=1；
    ET1=0；
    ES=l；
    EA=1；
    pointer＝–1；
/*发送地址帧*/
```

```
    TB8=1；
    SBUF=NODE_ADDR；
/*等待全部数据帧发送完毕*/
      while(pointer<COUNT)；
          /*  ……  */
/*添加其它程序行*/
}
/*发送中断服务函数*/
void send(void)interrupt 4 using 3
   {
/*清发送中断标志，并修改发送缓冲区当前位置指针*/
     T1=0；
     pointer++；
/*如果全部数据发送完毕则返回；否则发送下一帧数据*/
     if(pointer>=COUNT)return；
      else
        {
          TB8=0；                          /*设置数据帧标志*/
          SBUF=buffer[pointer]；           /*启动发送*/
        }
 }
```

接收程序(文件名为 RECEIVE.C)。

```
#include＜reg51.h＞
#define COUNT 10                           /*定义接收缓冲区*/
#define NODE_ADDR 64                       /*定义本节点地址*/
unsigned char buffer[COUNT]；              /*定义接收缓冲区和当前位置指针*/
int pointer；
main( )
{
/*初始化串行口和波特率发生器，并允许串行口接收地址帧*/
     SCON＝0xf0；                  /*MODEL 3，REN=1，SM2=1*/
     TMOD=0X20；
     TH1=0Xfd；
     TR1=1；
     ET1=0；
     ES=1；
     EA=1；
/*等待接收地址帧和全部数据帧*/
```

```
        pointer＝0；
        while(pointer＜COUNT＝；
       ／*  ……  *／
/*添加其它程序行*/
}

/*接收中断服务函数*／
void receive(void)interrupt 4 using 3
  {
       RI＝0；      /*清接收中断标志*/
/*如果为本节点地址帧，则置 SM2＝0，以便接收数据帧*/
        if(RB8==1)
        {
          if(SBUF==NODE_ADDR)SM2=0；
          return；
        }
/*将接收到的数据帧送接收缓冲区，并修改当前位置指针*/
       buffer[pointer++]＝SBUF；
/*如果已接收完全部数据帧，则此次通信结束，置 SM2=1，准备下一次通信*/
        if(pointer>=COUNT)SM2=1；
}
```

5.4 用软件方法提高单片机通信距离

用硬件方法提高单片机的通信距离在第 4 章已介绍，本节介绍如何用 I/O 口模拟差分总线及异步通信协议，以此提高串行通信的距离。

由于单片机的通信信号为 TTL 电平，如果不采取其它的措施，其通信速率在 9600 b/s 时的通信距离不超过 5 m。为了延长单片机的通信距离，通常采用 RS－232/RS－485 转换器、RS－232/RS－422 转换器或光隔远程收发器。要用纯软件方法实现 51 单片机之间的远程通信，首先必须将单片机的 TTL 电平用软件方法转换为差分电平，其次要用普通 I/O 口线来构成软件串行口，并且在软件上进行正确的配合。

5.4.1 TTL 电平转换成差分电平的纯软件方法

51 单片机的信号均为 TTL 电平。TTL 电平信号传输距离非常有限，而差分电平信号则取决于两种信号线之间的电平差值。如果某条信号线的电平高于另一条，则信号为 1，否则为 0。由于差分电平信号可以避免长距离传输导线上的电荷积累，并且具有更宽的电平范围，所以传输距离远得多。RS485 差分电平信号在 9600 b/s 传输率时，传输距离可达 1200 m。

为了用纯软件实现差分电平传输，就不能使用 51 单片机本身的硬件串行口，而是用普

通 I/O 线来实现串行通信。差分电平要用两个 I/O 口来实现，如 P1.2 和 P1.3，参见图 5.4.1。当传输信号为 1 时，P1.2 为+5 V，而 P1.3 为 0 V。当传输信号为 0 时，P1.2 为 0 V，而 P1.3 为+5 V。注意，P1.2 和 P1.3 不得同时置+5 V 或同时置 0 V。

图 5.4.1　I/O 口实现差分电平

5.4.2　软件串行口的实现方法

由普通 I/O 口来实现串行通信，习惯上称为软件串行口。采用差分电平通信的软件串行口的硬件接线图如图 5.4.2 所示。A、B 两机均为 51 单片机。P1.2 和 P1.3 用于发送，P1.0 和 P1.1 用于接收，P1.0 同时接本机中 INT0。软件串行口一般采用标准的 10 位异步通信格式：1 位起始位(信号 0)，8 位数据位，1 位停止位(信号 1)。接收时均是低位数据在前，高位数据在后。

软件串行口接收和发送的工作原理和过程如下：单片机执行初始化程序时，定义 P1 口为位操作方式，其中 P1.0 和 P1.1 为输入(初始化为“1”)，P1.2 和 P1.3 为输出。P1.2 初始化为“1”，P1.3 初始化为“0”，这样发送信号处于停止位(差分电平“1”)。定义 INT0 为负沿触发。允许中断且定义成高优先级，然后开中断，两机进入随时可开始串行通信的等待状态。A、B 不通信时，两机的收发均为“1”，一旦某机(假设为 A)需要与对方通信，A 机以约定的波特率(假定为 9600 b/s)通过 P1.2 和 P1.3 发送。

图 5.4.2　采用差分电平通信硬件接线图

发送和接收一个字节的过程如下：A 机发送端首先发送起始位(差分“0”电平)，B 机 INT0 引脚产生下跳沿后，产生中断申请。B 机响应此中断后，执行 INT0 中断服务子程序。

在中断服务子程序开始，用位输入指令读入 P1.0 和 P1.1 状态，如果分别是“ 1 ”电平和“ 0 ”电平，则表明此次中断是受干扰所致，因而取消此次接收过程，中断返回；如果 P1.0 和 P1.1 状态读入分别是“ 0 ”电平和“ 1 ”电平，则表明本次中断确是 A 机发送起始位所引起，经精确延时，在 A 机发送各数据位中间处进行采样，获得各数据位的状态，最后生成一个字节，送有关单元之后中断返回。虽然送往 INT0 的信号为 TTL 电平，但是由于它的传输速率非常之低，加上软件抗干扰措施，所以仍然可以有效地传输 1200 m。

5.4.3 软件串行通信的编程

串行通信要实现成功接收必须解决以下关键技术：

(1) 要准确、快速检测出对方发出的起始位以及起始位负跳变的时刻；

(2) 保证在每个数据位中间采样；

(3) 具有有效的校验手段。

我们有针对性地采取以下措施：用定义成高级中断的外中断引脚与接收线相连，及时捕捉起始位信息，并在确定采样时刻的计算中扣除中断响应滞后的延时时间；精心设计、编写程序，逐条计算指令实际执行时间，用关闭中断的方法排除其它中断干扰，从而保证在每个数据位中间处进行采样；每位重复采样多次，确定各数据位的状态，从而可大大减少远程通信常见的瞬态干扰(其特点是幅度大、作用时间短、随机性强)对通信的不良影响。综合采用上述技术措施，接收 1 帧数据的中断服务程序见例 5.4.1。

例 5.4.1

```
        CLR EX0                 ；关中断
        PUSH A
        CLR A
        MOV R0，#03H            ；置采样次数
LP0：                           ；读入数据
        JB P1.0，LP1
        JB P1.1，LP2
        SJMP END
LP1:
        JNB P1.1，LP3
        SJMP END
LP2:
        INC A
        SJMP LP4
LP3:
        NOP
        NOP
        NOP
LP4:
```

```
        DJNZ R0，LP0          ；连续采样三次
        CLR C
        SUBB A，#02H          ；若数据两次为１，则中断返回
        JC END
        MOV R1，#08H
        MOV R0，#2CH
LP5:
        DJNZ R0，$
        NOP
        MOV R2，#03H
LP6:                          ；采样三次
        JB P1.0，LP7
        JB P1.1，LP8
        SJMP END
LP7:
        JNB P1.1，LP9
        SJMP END
LP8:
        INC R0                ；记采样为 0 的次数
LP9:
        DJNZ R2，LP6
        MOV A，R0
        CLR C
        SUBB A，#02H
        RRC A
        MOV R0，#02H
    DJNZ R1，LP5              ；采样八位数据
        POP A
END:
        RETI
```

第 6 章　51 单片机与 PC 机的通信技术

由于 Windows 操作系统限制了软件设计者对 PC 机底层硬件的直接操作，因此在串口通信方面，本章没有介绍汇编语言编程。通信的硬件结构请参看 4.2 节。

本章主要内容：

★ DOS 环境下的串口通信程序设计
★ Windows 环境下的串口通信程序设计
★ 通过 PC 机标准键盘接口传输数据
★ 单片机与 PC 机并行传输技术的应用

6.1　DOS 环境下的串口通信程序设计

在 DOS 环境下实现异步串行通信可以有三种方法：

(1) 汇编语言直接驱动硬件。

(2) DOS 功能调用。

(3) BIOS 功能调用。

本节给出一个 BIOS 功能调用的例子。

6.1.1　bioscom 函数功能介绍

PC 机中提供了一个有关串行口的 BIOS 软中断，中断号是 14H。Turbo C 的库函数中提供了专门调用 BIOS 串行口软中断的函数 bioscom( )。

函数 bioscom(int cmd，char byte，int port)用于实现由 port 指定的在 I/O 端口上进行的各种 RS－232C 异步通信。

port 为 0 表示串口 1(COM1)，port 为 1 表示串口 2(COM2)。

cmd 决定通信类型(它实际上是软中断 INT 14H 的子功能号)，为 0 表示初始化串行口；为 1 表示发送一个字符(写串口)；为 2 表示接收一个字符(读串口)；为 3 表示返回串口当前状态。

在使用串口通信之前，要对其进行初始化，此时 cmd=0。byte 值决定了串口工作方式。byte(D7 D6 D5 D4 D3 D2 D1 D0)初始化参数含义见表 6.1.1。

例如，若想把串行口 1(COM1)设置为 1200 b/s 的波特率，无奇偶校验，使用 1 个停止位和 8 个数据位时，应使 byte 参数编码为 10000011，用十六进制表示为 0x83。初始化语句为 bioscom(0，0x83，1)。

表 6.1.1　byte 初始化参数表

D1D0	数据位个数
00	无效
01	无效
10	7 位数据
11	8 位数据
D2	停止位个数
0	使用 1 位停止位
1	使用 2 位停止位
D4D3	校验方式
00	无校验位
01	奇校验
10	无校验位
11	偶校验
D7D6D5	通信波特率(b/s)
000	110
001	150
010	300
011	600
100	1200
101	2400
110	4800
111	9600

从串行口发送一个字符时，令 cmd=1，byte 为发送的字符。

从串行口接收一个字符时，令 cmd=2，byte 参数被忽略，可以写成 0。

取串行口状态时，令 cmd=3，byte 参数也被忽略，可以写成 0。

bioscom( )总是返回一个 16 位的二进制数，它通常反映了串行口的某些重要的状态信息，这也是编写查询方式通信程序的重要依据：其返回值的高 8 位字节描述了线路状态，见表 6.1.2 所示；返回值的低 8 位随 cmd 值而定，常用于描述调制解调器的状态。cmd=1 时，低 8 位返回值是原发送的字符；cmd=2 时，低 8 位返回的是由串行口接收的字符；cmd 分别为 0 或 3 时，低 8 位含义如表 6.1.2 所示。

表 6.1.2　bioscom( )返回值含义表

bioscom( )返回值高 8 位的意义		cmd 为 0 或 3 时 bioscom( )返回值低 8 位的意义	
位	含　义	位	含　义
15	起时错误	7	线路信号被检测
14	发送移位寄存器空	6	振铃指示
13	发送保持寄存器空	5	数据装置就绪
12	间断检测错误	4	清除发送
11	接收帧的格式错误	3	线路信号变化
10	接收奇偶错误	2	脉冲后沿振铃检测
9	接收重叠错误	1	数据装置就绪变化
8	接收数据就绪	0	清除发送信号变化

6.1.2 调用 bioscom 函数实现通信的例子

1. **基本的收发子程序**

例 6.1.1

下面程序是利用 Turbo C **编写的，从** COM1 **串口发送一个字符的是函数** sport( )**。调用** bioscom( )**函数，采用查询方式。**

```
void sport(char ch)
{
while(!(bioscom(3，0，0)&0x2000))；
if(bioscom(1，ch，0)&0x8000)
    {
        print("超时错误\n")；
        exit(1)；
    }
}
```

从 COM1 **串口接收一个字符的函数** rport( )**，采用查询方式**

```
rpor( )
{
int a；
while(!(bioscom(3，0，0)&0x0100))
    {
        if(kbhit( ))
            {
                getch( )；
                exit(1)；
            }
        a=bioscom(2，0，0)&0x00ff；
        return(a)；
    }
}
```

该函数不断查询“ 数据准备就绪 ”位：若为 1**，说明串口收到了一个字符的数据；否则，用按下任一键退出的方法避免程序陷入死循环。**

sport( )**和** rport( )**这两个函数将作为后面发送和接收文件程序中的函数被反复调用。**

2. PC **机接收、发送子程序**

(1) **通信协议。波特率为** 1200 b/s**，为** 8 **位数据位，** 1 **位停止位，无奇偶检验，**PC **机采用查询方式收发数据；**8051 **采用中断方式接收，查询方式发送，校验数据的累加和。**

软件握手方式：发送方在正式发送数据之前先发一个“ ? **”号**(ASCII **码**)**作为联络信号，接收方接到“** ? **”号后回送一个“** . **”号作为应答信号。随后，发送方依次发送数据长度**(**字**

节数)、数据和校验和。接收方在收到发送方发过来的校验和后，将其与自己所累加的校验和相比较：相同则回送一个字母“ O ”，表示传送正确并结束本次的通信过程；若不相同则回送一个“ F ”，并要求发送方重新发送数据，直到接收正确为止。

(2) PC 机发送子程序。规定欲发送的文件存在当前目录下，并且，为了说明问题的方便，只传送总字节小于 256 个字符的文件。

例 6.1.2　PC 机发送文件函数 sendf( )程序清单。

```
void sendf(char *fname)
{
FILE *fp；
char ch；
int handle，count，sum=0；
if((fp=fopen(fname，"r"))==NULL)
      {
            printf("不能打开输出文件!\n")；
            exit(1)；
      }
handle=fileno(fp)；                    /*取得文件句柄*/
count=filelength(handle)；             /*取得文件总字节数*/
printf("准备发送文件...\n")；
do
      {
            ch=?’；                    /*发送联络信号?*/
            sport(ch)；
      }While(rport( )!=’.’，)；        /*直到接到应答信号为止*/
sport(count)；                         /*发送总字节数*/
rep:
for(；count；count--)
{
ch=getc(fp)；                          /*文件中取一个字符*/
sum=sum+ch；                           /*累加校验和*/
if(ferror(fp))
      {
            printf("读文件有错误＼n")；
            break；
      }
sport(ch)；                            /*从串口发一个字符*/
}
sport(sum)；                           /*发送累加校验和*/
if(rport( )=='F')                      /*发送错误则重发*/
```

```
        {
            count=filelength(handle)；
            sum=0；
            fseek(fp，-count，1)；      /*文件指针回退 COUNT 字节*/
            goto rep；
        }
    else
        {
            fclose(fp)；
            printf("发送文件结束＼n")；
        }
    }
```

(3) PC 机接收文件子程序。接收函数 receivef( )用查询方式从串口接收一个总字节数小于 256 个字符的文件，接收的文件也存于当前目录下。

例 6.1.3

```
void receivef(char *fname)
{
FILE *fp；
char ch；
int count，temp，sum=0；
remove(fname)；                    /*盘上有同名文件将被删掉*/
if((fp=fopen(fname，"w"))==NULL)；
    {
        printf("不能打开输入文件\n")；
        exit(1)；
    }
printf("接收文件名:%s\n"，fname)；
while(rport( )!=? ')；             /*收到联络信号?*/
sport('$')；
ch='. '；
sport(ch)；                        /*发应答信号'. '*/
temp=rport( )；                    /*收总字节数*/
count=temp；
rep:
for(；count；count--)
    {
        ch=rport( )；                  /*从串口接收一个字符*/
        putc(ch，fp)；                 /*将一个字符写入文件*/
        sum=sum+ch；                   /*累加校验和*/
```

```
            if(ferror(fp))
                {
                    printf("写文件有错误\n")；
                    exit(1)；
                }
        }
    if(rport()!=sum)
        {
            ch='F'；
            sport(ch)；                    /*校验和有错误，发F*/
            count=temp；
            sum=0；
            fseek(fp，– count，1)；        /*文件指针回退 COUNT 个字节*/
            goto rep；
        }
    else
        {
            ch='O'；
            sport(ch)；                    /*校验正确，发'O' (注意是字母 O)*/
            fclose(fp)；
            printf("接收文件结束\n")；
        }
    }
```

(4) PC 机主程序。在有了上述发送和接收文件两个函数之后，主程序的工作只是在完成串口初始化后，根据键入的命令来决定是发送文件还是接收文件。

例 6.1.4　PC 机主程序。

```
main(int argc，char *argv[ ])
{
while(argc!=3)
    {
        printf("命令行命令不正确，请重新键入命令!\n")；
        exit(1)；
    }
bioscom(0，0x83，0)；/*串口初始化*/
if(tolower((argv[1])=='s')
sendf(argv[2])；
else if (tolower(*argv[1]=='r')
receivef(argv[2])；
}
```

这里采用的是带参主函数 main(int argc，char *argv[ ])。其中 argc 是一个整型变量，argv[ ] 是一个字符型指针数组。利用 main 函数的参数可以使主程序从系统得到所需数据(也就是说带参函数可直接从 DOS 命令行中得到参数值，当然，这些值是字符串)。当程序运行时(在 DOS 下执行.EXE 文件)，可以根据输入的命令行参数进行相应的处理。

例如，当 C 语言程序命名为 mycomm 时，源文件 mycomm.c 经编译链接后生成可执行文件 mycomm.exe。若要执行程序 mycomm，将当前目录下名为 fl.c 的文件从串口发送出去，需键入下述命令:

mycomm　s　f1.c

从串口接收若干字符，并写到当前目录下名为 f2.c 的文件中去，可键入命令：

mycomm　r　f2.c

3. 8051 通信软件设计

在这个例子中，单片机采用查询发送、中断接收的方式。

(1) 单片机查询发送子程序。本程序将片外 RAM 中从 1000H 开始的小于 256 字节的数据从串口发送出去，发送的数据字节数在 R7 中，用 R6 作累加和寄存器。

例 6.1.5　单片机查询发送子程序。

```
SEND：
        MOV A，#3FH  ；发?'(3FH)
        MOV SBUF，A
        JNB TI，$
        CLR TI
        JNB RI，$
        CLR RI
        MOV A，SBUF
        CJNE A，#2EH，SEND          ；应答信号是"."(2EH)，则发字节数
        MOV A，R7
        MOV R3，A                   ；暂存总字节数
        MOV SBUF，A
        JNB TI，$
        CLR TI
        MOV R6，#00H
        MOV DPTR，#1000H
SEND1：
        MOVX A，@DPTR
        MOV SBUF，A
        JNB TI，$
        CLR TI
        ADD A，R6                   ；计算校验和
        MOV R6，A
```

```
        INC DPTR
        DJNZ R7，SEND1          ；未发送完毕 R7 个数据，则继续
        MOV A，R6
        MOV SBUF，A
        JNB TI，$
        CLR TI                  ；发送校验和
        JNB RI，$
        CLR RI
        MOV A，SBUF
        CJNE A，#46H，SEND2     ；如收到应答是"F"(46H)，则重发数据
        RET
SEND2:
        MOV DPTR，#1000H
        MOV R6，#00H
        MOV A，R3
        MOV R7，A
        LJMP SEND1
```

(2) **单片机接收中断服务子程序。在中断服务子程序中，为了区别所接收的信号是联络信号还是字节数，是数据还是校验和，需要设立不同的标志位，为此在可进行位寻址的** RAM **中设定。详见表** 6.1.3。

表 6.1.3　**标志位内容表**

位 地 址	内　容
00H	接收联络信号标志位
01H	接收字节数标志位
02H	接收数据标志位
03H	接收文件结束标志位

在初始化时，这些位均为 0。**在中断服务子程序中，将接收到的字节数存入** R7 **中，接收的数据存入片外** RAM **从** 1000H **开始的单元中。**

例 6.1.6　**单片机接收中断服务子程序。**

```
RECE：
        CLR ES
        CLR RI
        JB 00H，RECE1
        MOV A，SBUF
        CJNE A，#3FH，RECE2     ；接收的不是?号，则退出
        MOV A，#2EH
        MOV SBUF，A             ；发送应答信号.(2EH)
        JNB TI，$
        CLR TI
```

```
        SETB 00H
        SETB ES
        RETI
        RECE2：
        MOV A，#24H
        MOV SBUF，A          ；发送应答信号$(24H)
        JNB TI，$
        CLR TI
        SETB ES
        RETI
RECE1:
        JB 01H，RECE4
        MOV A，SBUF          ；接收字节数
        MOV R7，A
        MOV R3，A            ；暂存字节总数
        SETB 01H
        SETB ES
        RETI
RECE4:
        JB 02H，RECE5
        MOV A，SBUF          ；接收一个字符
        MOVX @DPTR，A        ；存入片外 RAM 中
        ADD A，R6
        MOV R6，A
        INC DPTR
        DJNZ R7，RECE7
        SETB 02H
RECE7：
        SETB ES
        RETI
RECE5：
        MOV A，SBUF
        CJNE A，06H，RECE8   ；06H 为 R6 的字节地址
        MOV A，#4FH
        MOV SBUF，A          ；校验和正确，发O(4FH)
        JNB TI，$
        CLR TI
        SETB 03H
        SETB ES
        RETI
```

```
RECE8：
        MOV DPTR，#1000H
        MOV R6，#00H
        MOV A，R3
        MOV R7，A
        MOV A，#46H
        MOV SBUF，A          ；校验和不正确，发F’(46H)
        JNB TI，$
        CLR TI
        CLR 02H
        SETB ES
        RETI
```

(3) **单片机主程序。**

例 6.1.7　**单片机主程序。**

```
        ORG 0000H
        LJMP MAIN
        ORG 0023H
        LJMP RECE
        ORG 0030H

MAIN:
        MOV SP，#60H
        MOV SCON，#50H          ；串口初始化
        MOV TMOD，#20H
        MOV TH1，#0F3H
        MOV TL1，#0F3H
        MOV PCON，#00H
        SETB TR1
        SETB EA
        SETB ES
LP:
        CLR 00H                 ；初始化
        CLR 01H
        CLR 02H
        CLR 03H
        MOV R6，#00H
        MOV DPTR，#1000H
        JNB 03H，$      ；等待接收文件
        LCALL SEND      ；发送文件
        LJMP LP
```

6.2 Windows 环境下的串口通信程序设计

6.2.1 三种常用的通信方法 ActiveX \API\DLL

1. Windows 串行通信机制

在 Windows 环境下，系统的各种资源是共享的，不允许用户程序以独占的方式使用共享资源，以保证不同的应用程序共享串行口资源。Windows 系统函数包含了支持通信中断的功能。Windows 系统为每个通信设备开辟了用户定义的输入输出缓冲区(即为读/写缓冲区)，数据进出通信口均由系统后台来完成。应用程序只需完成对输入输出缓冲区操作就可以了。实际过程是每接收一个字符就产生一个低级硬件中断，Windows 系统中的串行驱动程序就取得了控制权，并将接收到的字符放入输入数据缓冲区，然后将控制权返还正在运行的应用程序。如果输入缓冲区数据已满，串行驱动程序用当前定义的流控制机制通知发送方停止发送数据。队列中的数据按“先进先出”的顺序处理。这一切由 Windows 在后台执行，速度非常快，且不允许用户直接介入。

2. Windows 提供给用户实现串行通信的三种方法

(1) 通过串行通信控件实现，如微软提供的 MSComm 或 SPComm 等通信控件。使用控件的属性进行串行口配置，使用控件的事件驱动进行串口响应，使用控件的方法完成串行口接收和发送数据。其中最常用的是 MSComm32.OCX 控件，其属性见表 6.2.1。

表 6.2.1 MSComm32.OCX 控件属性表

属 性	描 述	数据类型	举 例
CommPort	设置并返回通信端口号	Integer	2
Settings	设置并返回波特率、校验位、数据位、停止位	String	“9600，N，8，1”
PortOpen	设置并返回端口状态，也可以用于打开和关闭串口	Boolean	True
OutBufferSize	设置并返回发送缓冲区的大小，以字符为单位	Integer	512
InBufferSize	设置并返回接收缓冲区的大小，以字符为单位	Integer	1024
Sthreshold	设置并返回发送时产生 OnComm 事件的字符数	Integer	0 不产生 OnComm 事件
Rthreshold	设置并返回接收时产生 OnComm 事件的字符数	Integer	8(收到 8 个字符时发生 OnComm 事件)
NullDiscard	设定是否忽略发送 0(Null)字符	Bollean	True(不发送) False(发送)
InputMode	设置并返回接收类型	ComInputModeText (字符方式) ComInputModeBinary (二进制方式)	ComInputModeBinary
InputLen	设置并返回从接收缓冲区读取的字符	Integer	256
ParityReplace	当发生奇偶校验错误时，设置并返回替换数据流中一个非法字符的字符	String	?

(2) 以文件方式打开串口，通过 Windows 应用编程 API 提供的一整套串行通信函数来实现。在 Windows 系统中，串行口和串行通信驱动程序是通过一个数据结构进行配置的，这个数据结构被称为设备控制块(Device Control Block)，简称 DCB。

(3) 使用动态链接库 DLL。自己编制端口驱动程序或使用第三方提供的 DLL 例程。动态连接库是一些过程或函数的集合。这些过程或函数在程序运行期间动态地链接到应用程序，而不是在编译期间静态地链接到可执行文件。Windows 应用程序大量使用了动态链接库文件，内容包括了系统应用程序需要的许多过程，例如低级操作函数、显示窗口和图形、管理内存等等。共有数千条函数和信息可以为应用程序所引用。

3. ActiveX 控件的注册方法

一般来说，一个外来 ActiveX 控件要在 Windows 中被正确使用，首先必须将控件文件(*.OCX)复制到硬盘 Windows\system\目录下(Win2000 和 WinNT 下是 system32 目录下)，并注册。未在 Windows 中注册过的 ActiveX 控件是不能使用的(微软提供的 MSComm 控件在安装 VB 时已经注册)。

注册 ActiveX 控件一般来说有三种途径：使用 Regsvr32.exe 程序对 ActiveX 控件进行注册；使用安装程序制作软件 InstallShield；在应用程序中加入注册代码。

上述方法适用范围不同，各有各的优点，下面分别讨论。

 说明：

此控件在 VB 和 VC 各版本中稍有不同，详细请查 MSDN Library。

4. 使用 Regsvr32.exe 程序对 ActiveX 控件进行注册

对于未注册过的 ActiveX 控件，可使用此软件对其进行注册，外来 ActiveX 控件要应用到自己的程序中也必须进行注册。该文件位于 Windows 目录的 system 子目录下。使用方法如下：

(1) 点击【开始】→【运行】；

(2) 在运行对话框中输入以下命令：

regsvr32 < 文件名 > 注册一个 ActiveX 控件

regsvr32 /u < 文件名 > 解除某 ActiveX 控件的注册

我们使用一些带 ActiveX 控件的应用程序时，有时会出现不能运行的情况，这有可能是其自带 ActiveX 控件未注册所致，这时可用上述命令。另外，VC++使用者若对 regsvr32.exe 的编码感兴趣的话，可在 VC++的联机帮助中找到其源代码。若使用 VC++6.0，则在 MSDN 光盘的\sample\VC98\MFC\controls\regsvr 中可找到其源代码。

5. 使用安装程序制作软件 InstallShield

使用 regsvr32.exe 来注册 ActiveX 控件虽然简单，但需要用户手工注册，在不用时还得手工解除注册，因此，这对一个应用程序来说并非好的解决方案。大型应用软件一般都有一个安装程序，在安装程序中解决 ActiveX 控件注册是较为理想的一种方案。使用 InstallShield 可以制作出专业级的 Setup，还可注册其中的 ActiveX 控件，而且在以后下载软件时，可自动注销掉以前注册的 ActiveX 控件。其方法如下：

(1) 启动 InstallShield，使用 Project Wizard 建立一个新的项目；

(2) 新建一个“File Group”，将需要注册的 ActiveX 控件文件放入此“File Group”中；

(3) 将上述“File Group”的“Self-Registered”属性设置成“Yes”。

上述方法仅为制作 Setup 中设置自动注册 ActiveX 控件的几个步骤，至于使用 InstallShield 怎样制作一个完整的 Setup，请读者自己参看有关书籍。

6. 在应用程序中加入注册代码

对于小型程序，不宜采取第二种方法，较好的方法是在程序中嵌入注册代码，实现应用程序自动注册。其编程方法是：

(1) 使用 Windows API 函数 LoadLibrary 载入 ActiveX 控件；

(2) 使用 GetProcAddress 函数获取 ActiveX 控件中注册函数 DllRegisterServer(注销函数为 DllUnregisterServer)指针；

(3) 调用注册函数 DllRegisterServer(或注销函数 DllUnregisterServer)。

下面以用 VC++6.0 编写的 RegActivex 程序为例，介绍怎样在程序中自动注册(注销)ActiveX 控件。

RegActivex 的主体框架由 VC++6.0 的 AppWizard 自动产生，在其基础上增加两个菜单项：【注册 ActiveX 控件】、【注销 ActiveX 控件】。当选择【注册 ActiveX 控件】菜单时，将对控件 toweratl.ocx(一个汉诺游戏)进行注册；当选择【注销 ActiveX 控件】菜单时，将解除控件 toweratl.ocx 的注册。

具体步骤如下：

(1) 使用 VC++ 6.0 建立一个单文档的应用程序 RegActivex，其它项目接受默认设置；

(2) 给应用程序增加【注册 ActiveX 控件】和【注销 ActiveX 控件】两个菜单项：

① 单击 VC++ 6.0 左边窗口的【ResoureView】，双击 MENU 项目下的 IDR_MAINFRAME 以打开菜单编辑器；

② 在菜单编辑器中加入一个主菜单【注册】，在【注册】主菜单下加入【注册 ActiveX 控件】和【注销 ActiveX 控件】两个菜单项；

(3) 给新建的菜单项【注册 ActiveX 控件】和【注销 ActiveX 控件】增加响应函数：

① 在资源编辑器中，双击 MENU 中的 IDR_MAINFRAME 打开菜单编辑器，单击【注册 ActiveX 控件】，按 Ctrl+W 键打开 MFC Class Wizard；

② 双击 Messages 框中的 COMMAND，给【注册 ActiveX 控件】菜单项增加响应函数 OnRegisterReg( )，单击右边的【Edit】按钮给 OnRegisterReg( )函数增加如下代码：

```
void CmainFrame::OnRegisterReg( )
{ //ActiveX 控件的路径及文件名
LPCTSTR pszDllName="toweratl.ocx"； //装载 ActiveX 控件
HINSTANCE hLib = LoadLibrary(pszDllName)；
if (hLib < (HINSTANCE)HINSTANCE_ERROR)
{
MessageBox("不能载入 Dll 文件!", "结果"，MB_OK)；return；} //获取注册函数 DllRegisterServer 地址
FARPROC lpDllEntryPoint；
```

```
lpDllEntryPoint = GetProcAddress(hLib，_T("DllRegisterServer"))； //调用注册函数 DllRegisterServer
if(lpDllEntryPoint!=NULL)
{
      if(FAILED((*lpDllEntryPoint)()))
    {
       MessageBox("调用 DllRegisterServer 失败！"，"结果"，MB_OK)；
      FreeLibrary(hLib)；
      return；
     }
   MessageBox("注册成功"，"结果"，MB_OK)；
}
  else MessageBox("调用 DllRegisterServer 失败！"，"结果"，MB_OK)；
}
```

③ 【注销 ActiveX 控件】菜单项响应函数的编写方法同上，代码也相似，只是将“lpDllEntryPoint=GetProcAddress(hLib，_T("DllRegisterServer"))；”改成“lpDllEntryPoint = GetProcAddress(hLib，_T("DllUnregisterServer"))；”。

6.2.2　用 VB 开发串口通信软件

用 VisualBasic6.0(简称 VB6)来开发 Windows 环境下的串口通信应用软件十分方便，编程工作量相对较小，只需进行主要应用功能的编程和少量界面控制的编程。

VB6 提供了许多供用户选择的控件(Customcontrol)，这些控件以.OCX 为文件后缀名，其中 MSComm32.OCX 即是用于串行通信的控件。如果需要使用该控件，可将该控件添加到工具箱(toolbox)内，这样就可以利用该控件进行串行通信程序的设计。MSComm 通信控件具有丰富的用于串口通信的属性及事件，提供了一系列标准通信命令的接口，可以用它创建全双工、事件驱动、高效实用的通信程序。该控件屏蔽了通信过程中的底层操作，程序员可以设置、监视 MSComm 的属性和事件。它提供了两种处理串行通信的方法：一是事件驱动法；二是查询法。

(1) 事件驱动法。这是一种很强的处理串口活动的方法。当串口接收到或发送完指定数量的数据时，或当状态发生改变时，MSComm 控件都将触发 OnComm 事件，该事件也可以捕获通信中的错误。当应用程序捕获到这些事件后，可通过检查 MSComm 控件的 CommEvent 属性的值来获知所发生的事件或错误，从而执行相应的处理。这种方法具有程序响应及时、可靠性高等优点。

(2) 查询法。可以在每个重要的程序之后查询 MSComm 控件某些属性(如 CommEvent 属性和 InBufferCount 属性)的值来检测事件和通信错误。这对小程序可能比较常用。

MSComm 事件的主要属性重述如下：

- Commport 设置并返回通信端口号。端口号可以设置为 1~16 的任何数，如 Mscomm.Commport=2 表示设置当前通信端口为 COM2。

- Setting 设置并返回波特率、奇偶校验、数据位、停止位。格式为 Mscomm.Setting =String。String 是一个包含四部分的字符串：第一部分为波特率；第二部分为奇偶校验，N 表示不校验，M 表示符号校验，E 表示偶校验，O 表示奇校验，S 表示空格校验；第三部分为数据位数，其可选值为 4、5、6、7、8；第四部分为停止位位数，其可选值为 1、1.5、2。Setting 属性的缺省值为“9600，N，8，1”。
- Portopen 设置并返回通信端口的状态，也可以打开和关闭端口。
- Input 从接收缓冲区返回和删除字符。该属性在运行时为只读。
- InputLen 设置并返回每次 Input 属性从接收缓冲区中读取的字符数。InputLen 属性的缺省值为 0。设置 InputLen 为 0 时，Input 将读取接收缓冲区的全部字符。
- Output 向传送缓冲区写数据。要传送的数据可以是文本数据或二进制数据。
- CommEvent 返回最近的通信事件或错误。只要有通信事件发生错误时就会产生 Oncomm 事件。CommEvent 属性中存有该错误或事件的数值代码。
- InputMode 设置返回类型。该属性为 0 时，Input 属性所检取的数据是文本；为 1 时，Input 属性所检取的数据是二进制数据。这个属性对于与单片机的通信尤为重要。

1. 两台 PC 机之间的串行通信程序设计

(1) 设计用户界面。

首先新建一个“标准 EXE”工程(如图 6.2.1 所示)，然后选择【工程/部件】菜单项(如图 6.2.2 所示)，在【部件】对话框中选择 Microsoft Comm Control6.0 即可添加 MsComm 控件(如图 6.2.3 所示)；再在窗体中依次布置如表 6.2.2 中的控件并设置其属性。完成后的界面如图 6.2.4 所示。

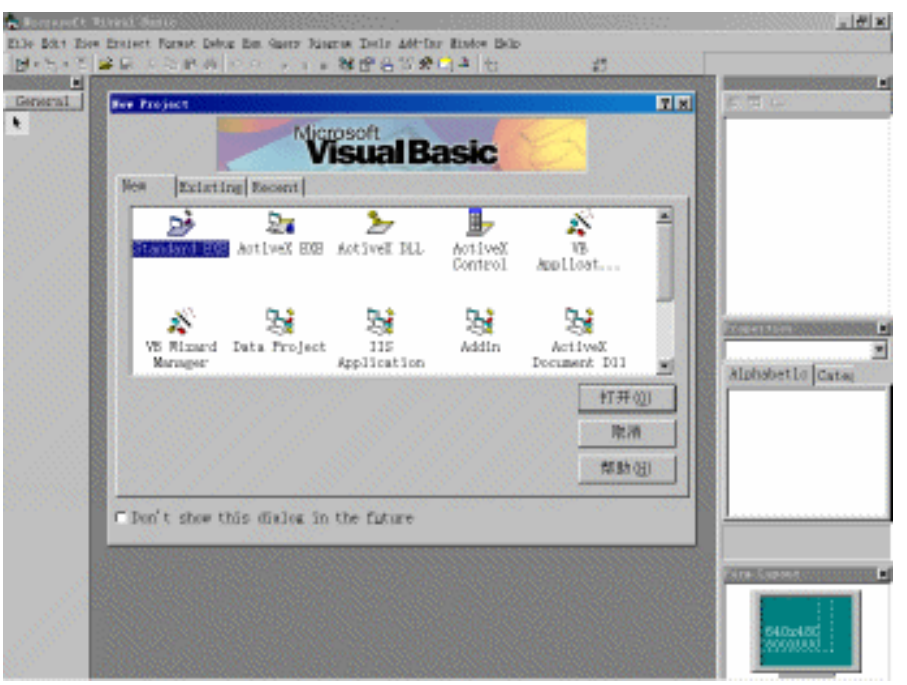


图 6.2.1 新建一个“标准 EXE”工程

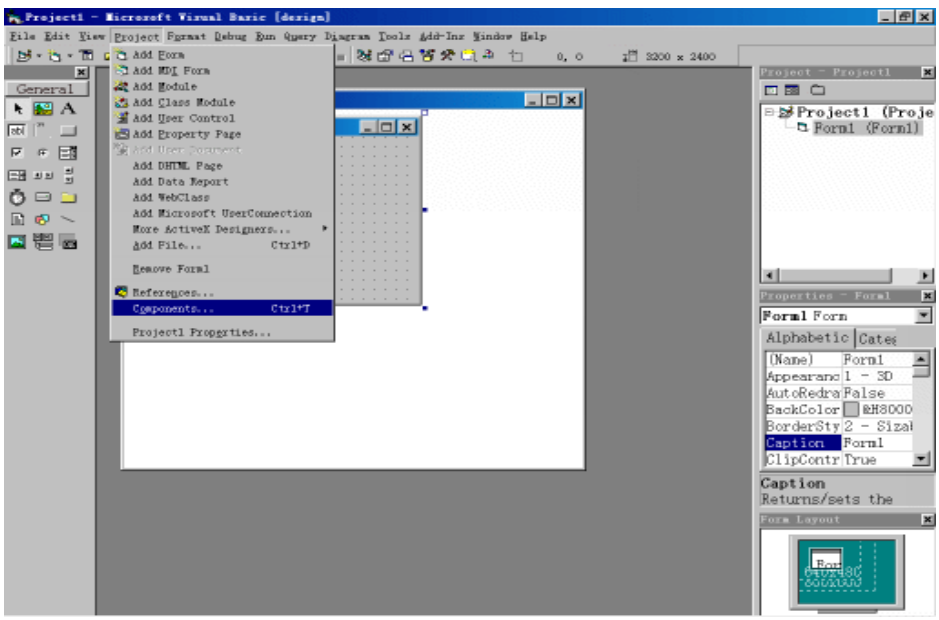


图 6.2.2　选择【部件】对话框

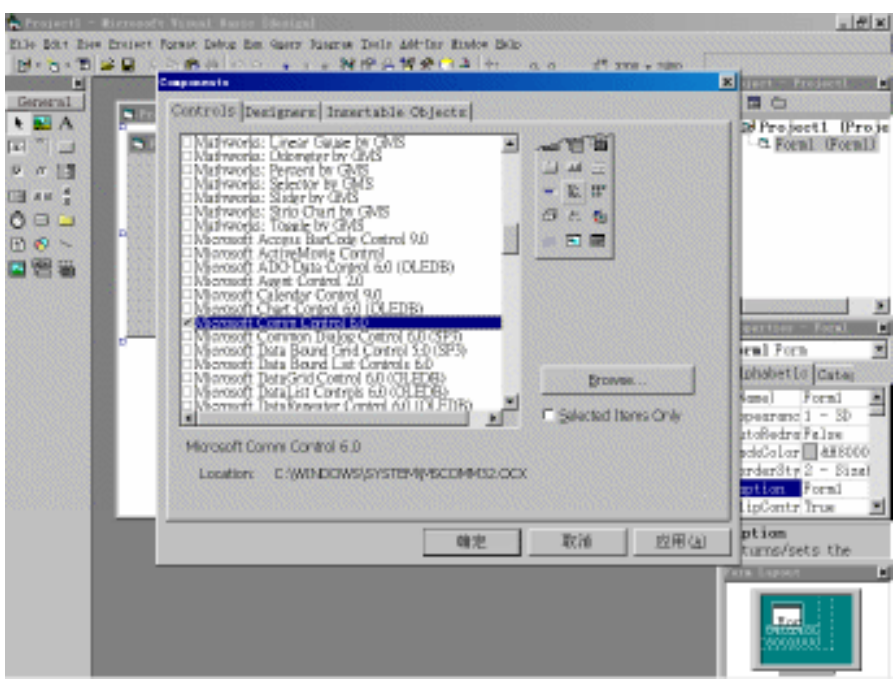


图 6.2.3　添加 MsComm 控件

表 6.2.2 控件属性表

控件名称	Name 属性	Caption 属性
Form	Forml	空
Command1	Comd1	确定
Command2	Comd2	退出
Command3	Comd3	确定
Command4	Comd4	退出
Command5	Comd5	退出程序
MSComm	msc1	空
Label1	Lab1	RS－232 接口功能检测
Label2	Lab2	发送数据测试
Label3	Lab3	接收数据测试
Text1	Txt1	空
Text2	Txt2	空

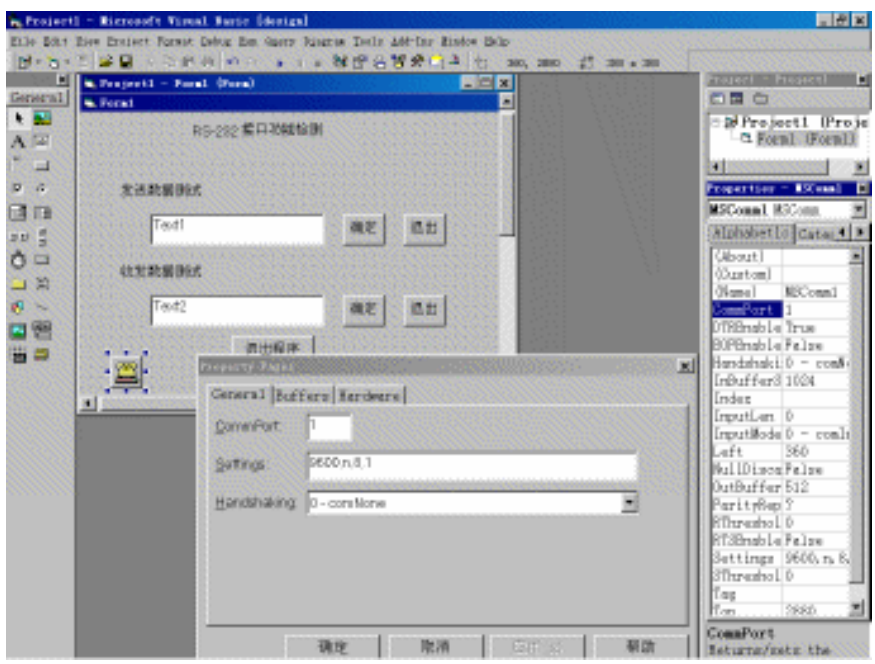


图 6.2.4 完成后的界面

例 6.2.1

```
Private Sub Form_Load( )
    Msc1.CommPort=1                    ’使用串行口 1
                                       ’波特率 9600 b/s，偶校验，8 个数据位，1 个停止位
```

```
    Msc1.Settings="9600，E，8，1"
                                    ’当使用 Msc1.Input 时，每次从接收缓冲区取一个字节
    Msc1.InputLen=0
    Msc1.Portopen=True              ’打开串行口
End Sub

Private Sub Comd1_Click( )
        Dim x As string
    On Error Resume Next            ’简单的错误处理
    If Txt1.Text=""Then
        x=MsgBox("发送数据不能为空"，16)
        Exit Sub
    End If
    Msc1.Output=Txt1.Text+Chr$(13)
                                    发送数据
    For i=1 To 20000000
    Next
End Sub

Private Sub Comd2_Click( )
    Txt1.Text=""
        Txt1.SetFocus
End Sub

Private Sub Comd3_Click( )
    Dim instring As string
    instring=Msc1.Input
    Msc1.Output=Txt2.Text+Chr$(13)
    Do
    DoEvents
    Loop Until Msc1.InBufferCount>=20
    instring=Msc1.Input             ’从接收队列中读入字符串
    Txt2，Text=""
    Txt2.Text=instring              ’显示读入的字符串
End Sub

Private Sub Comd4_Click( )
    Txt2.Text=""
    Txt2.SetFocus
```

```
End Sub

Private Sub Comd5_Click( )
    Msc1.PortOpen=False
    End
End Sub
```

说明：

上述程序设置的端口通信协议为：9600 b/s **的波特率，偶校验，**8 **个数据位，**1 **个停止位。当退出程序时，一定要关闭串行口，通过** Comm1.PortOpen=False **完成。在这个程序的基础上，再参考第** 5 **章相关内容，可以很容易地编写出单片机部分的通信程序。**

2. **利用** OnComm **事件实现串口通信**

初始化设置 MSComm **控件的** CommPort、Settings、EOFEnable **属性。设置** MSComm **控件的** PortOpen **属性，通过** Output **方法发送字符串数据。设置并读取** MSComm **控件的** InputLen **和** InBufferCount **属性，通过** Input **方法接收字符串数据。**MSComm1_OnComm **事件检测发送或接收数据时哪种事件触发或出错的原因。测试前用电缆接好串口头后连接两台计算机，同时运行此程序发送和接收** send.txt **文本中的数据。程序界面如图** 6.2.5 **所示。**

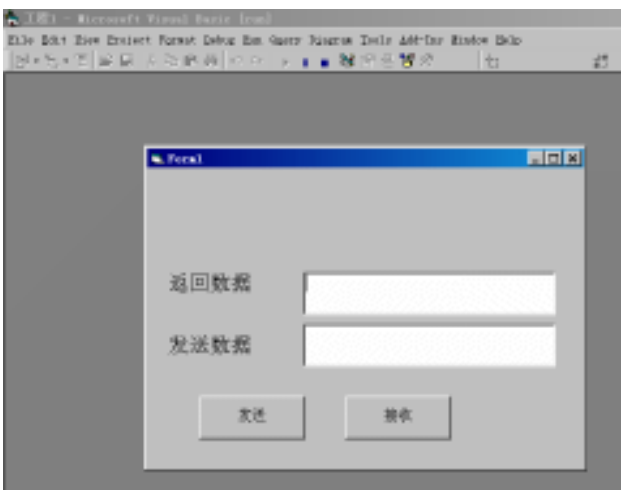


图 6.2.5 **程序运行界面**

例 6.2.2

```
Private Sub Command1_Click( )
Dim x1， x2， sline As String
Dim nfilehandle As Integer
Const filename = "send.txt"

nfilehandle = FreeFile
Open filename For Input As #nfilehandle
```

```
Do While Not EOF(nfilehandle)     ’循环至文件尾
    Input #nfilehandle，  sline
     x2 = x2 + sline + Chr(26)
Loop

Close #nfilehandle       ’关闭文件
X1 = sends((x2))
Text2.Text = x1
End Sub

Private Sub Command2_Click( )
Dim x As String
x = receive( )
Text1.Text = x
End Sub

Private Sub Form_Load( )
 Form1.MSComm1.CommPort = 1
      ’9600 b/s 波特率，无奇偶校验，8 个数据位，一个停止位
 Form1.MSComm1.Settings = "9600，N，8，1"
 Form1.MSComm1.EOFEnable = True   ’只有先设其为真 comEvEOF 才能触发 oncomm 事件
 Form1.MSComm1.Rthreshold = 1       ’当输入字符小于此值时  comEvReceive 才能触发 oncomm 事件
 Form1.Timer2.Enabled = True
 Form1.Timer2.Interval = 3000
End Sub

Private Sub Timer1_Timer( )
Form1.Timer1.Enabled = False
End Sub

Private Sub Timer2_Timer( )
Dim x As String
x = receive( )
Text1.Text = x
End Sub
Private Sub MSComm1_OnComm( )
     Select Case MSComm1.CommEvent
    ’可在每个 Case 后面添加处理错误和事件的程序代码
’错误
```

```
            Case comEventBreak          ' 收到 Break
            Case comEventCDTO           'CD (RLSD) 超时
            Case comEventCTSTO          'CTS 超时
            Case comEventDSRTO          'DSR 超时
            Case comEventFrame          'Framing 错误
            Case comEventOverrun        ' 数据丢失
            Case comEventRxOver         ' 接收缓冲区溢出
            Case comEventRxParity       'Parity 错误
            Case comEventTxFull         ' 传输缓冲区已满
            Case comEventDCB            ' 获取 DCB 时意外错误

        ' 事件
            Case comEvCD                'CD 线状态变化
            Case comEvCTS               'CTS 线状态变化
            Case comEvDSR               'DSR 线状态变化
            Case comEvRing              'Ring Indicator 变化
            Case comEvReceive           ' 收到 Rthreshold # of  chars
              X = receive( )
              Text1.Text = x
            Case comEvSend              ' 传输缓冲区有 Sthreshold 个字符
            Case comEvEOF               ' 输入数据流中发现 EOF 字符
              x = receive( )
              Text1.Text = x
          End Select
End Sub

Function sends(str As String)
    Dim count As Integer
    Dim instring As String
     If Not Form1.MSComm1.PortOpen Then
           sends = "CAN NOT OPEN PORT!"
           Form1.MSComm1.PortOpen = True
     End If
     ' 当输入占用时，
     ' 告诉控件读入整个缓冲区
     Form1.MSComm1.InputLen = 0
     Form1.MSComm1.Output = str
     ' 关闭串行端口
     Form1.MSComm1.PortOpen = False
```

```
sends = str
End Function
Function rec( ) As String
    Dim insting As String
    Dim count As Integer
    Dim CONT1 As Boolean
    Dim cont2 As Boolean
    CONT1 = False
    cont2 = True
    Form1.Timer1.Enabled = True
    Form1.Timer1.Interval = 2000
    If Not Form1.MSComm1.PortOpen Then
        Form1.MSComm1.PortOpen = True
    End If
    Form1.MSComm1.InputLen = 0
    Do
        DoEvents
        If Form1.Timer1.Enabled = False Then
            CONT1 = True
            Exit Do
        End If
        cont2 = True
    Loop Until Form1.MSComm1.InBufferCount >= 3
      If CONT1 Then
       rec = "TIME    OUT!缓冲区未收到数据！"
       Else
      If cont2 Then
        rec = "缓冲区已收到数据！"
      End If
    ’从串行端口读 "OK" 响应
       Count = Form1.MSComm1.InBufferCount
    ’关闭串行端口
     Instring = Form1.MSComm1.Input
     rec = instring
     End If
  Form1.MSComm1.PortOpen = False
End Function
Function receive( ) As String
    Dim insting As String
```

```
    Dim count As Integer
    If Not Form1.MSComm1.PortOpen Then
        Form1.MSComm1.PortOpen = True
    End If
    Form1.MSComm1.InputLen = 0
    ' 检查数据
    If Form1.MSComm1.InBufferCount Then
    ' 读数据
    Instring = Form1.MSComm1.Input
    Else
    instring = "缓冲区还未收到数据！"
    End If
     count = Form1.MSComm1.InBufferCount
    ' 关闭串行端口
     Receive = instring
End Function
```

3. 使用 Timer 控件进行事件驱动

Timer 控件的主要属性及用法如下：

Interval 属性：它返回或设置对 Timer 控件的计时事件调用间隔的毫秒数。

Enabled 属性：它决定该控件是否对时间的推移作出响应。将 Enabled 设置为“ False ”，会关闭 Timer 控件；设置为“ True ”，则打开 Timer 控件。

创建 Timer 事件程序，可通知 VB 在每次 Interval 到时该做什么。当 Timer Enabled 属性为“ True ”时，程序从其 Interval 属性的设置值开始倒计时。VB 将在 Interval 时间到后自动访问 Timer_Timer 过程。

为实现通信程序，须在 VB 开发环境下设置一个用作控制通信的窗体。窗体上主要有一个通信控件 Mscomm1 和两个 Timer 控件。VB 的特点是事件驱动，定时器控件会定时触发相应事件的驱动程序。

(1) 发送单片机命令。为了使主机能够对整个检测过程进行实时控制，需要在发送命令以后设定等待的时间，也可以通过条件判断下一步是发送还是接收命令。对发送的命令，可能是文本方式或二进制代码。在发送二进制代码时，应特别注意发送的格式。发送命令过程是一个带参过程，这样可使发送命令简便易行。

例 6.2.3

```
Sub 发送单片机命令过程(command As Byte)
Dim 输出命令(1To1)As Byte
DoEvents
输出命令(1)=command
        MSComm1.OutBufferCount=0
        MSComm1.Output=输出命令
```

```
        MSComm1.InBufferCount=0
    End sub
```

(2) **接收数据。接收数据是一个被动的过程，可以通过函数来实现，由定时器开启。在接收过程中，多数用特征字符，如“** OK”、“ # ”**等，这些需要在通信协议中约定。**

例 6.2.4

```
Function 接收数据( )
Do
    DoEvents
    In_buffer$=In_buffer$ & MSComm2.Input
    Loop Until InStr(In_buffer$，"OK")     '从串行端口读"OK"响应
    In_buffer=Left(In_buffer，len(In_buffer)-2)
    接收数据=In_buffer$
End Function
```

(3) Timer **控件控制。通过** Timer **控件来控制通信中的发送命令和接收数据过程，在通信程序中设置两个** Timer **控件分别控制发送单片机命令和接收单片机数据。为了实现一台** PC **机和多单片机之间的通信，可在一个** Timer **控件的过程中，在发送命令之前设定命令参数和要接收数据的单片机号，然后发送单片机命令；在另一个** Timer **控件的过程中，根据发送前设定的单片机号，接收不同单片机的数据。**

例 6.2.5　Timer **控件控制程序。**

```
'发送命令主控程序
Private Sub Timersend_Timer( )
Timersend.Enabled=False
  Select Case command
   Case 1
      Call 发送单片机命令过程(任务 1)
      TimerReceive.Enabled=False          '启动自动接收
   Case 2
     Call 发送单片机命令过程(任务 2)
     MSComm1.Rthreshold=0                 '关闭自动接收
     TimerReceive.Interval=500
     TimerReceive.Enabled=True            '启动定时器接收
     机号=1
   Case 3
     Call 发送单片机命令过程(任务 3)
     MSComm1.Rthreshold=0                 '关闭自动接收
     TimerReceive.Interval=500
     TimerReceive.Enabled=True            '启动定时器接收
     机号=2
```

```
    Case 4
    …
    Case n
    …
  End Select
  End Sub

  ' 接收数据主程序
  Private Sub TimerReceive_Timer( )
  TimerReceive.Enabled=False
    Select Case 机号
      Case 1
        In_buffer$=接收数据(机号)
        Call 任务 1
      Case 2
        In_buffer$=接收数据(机号)
        Call 任务 2
      Case 3
      …
      Case n
        In_buffer$=接收数据(机号)
        Call 任务 n
    End Select
  End Sub
```

4. 与单片机通信的演示例子

(1) 通信线路连接。PC 机串行口 9 针连接器的 2 脚和 3 脚通过电缆线与 RS232 收发器 MAX202 的 232 电平端口三线交叉连接。MAX202 的逻辑电平端口与 89C51 单片机(未画出)的串行口相连，如图 6.2.6 所示。

(2) PC 机编程。PC 机方用 VB 5.0 编写，其中串行通信模块仍然使用 ActiveX 控件 MSComm32.OCX 编程，该过程的步骤如下：

① 进入 VB，选择标准 EXE 项。

② 选主菜单【工程】中的【部件】，在其中的【控件】表中选 Microsoft Comm Control 5.0 项(用鼠标在该项前的方框内点击)；关闭菜单，这时在【工具箱】的图标窗口出现一电话机图标。

③ 用鼠标双击“电话机”图标，把它引入到当前“form”窗口中。

④ 从【工具箱】引入一个按钮控键，命名为“command1”。

⑤ 双击“comman1”控键，在编程窗口编写程序。

图 6.2.6　PC 机与单片机的串口连接

```
Private Sub Command1_Click( )
Dim outstring As String                          '定义输出字符串 outstring
Dim message As String，title As String， default As String   '定义输入窗口 InputBox
title = "输入窗口"
message = "请输入 ASCII 码字符"                   '提示信息
default = "no input"                             '无输入时的缺省字符串
Cls
outstring = InputBox(message，title，default)     '从键盘输入的窗口取得字符串 outstring
n = Len(outstring)                               '取 outstring 的长度
Print "outstringLenth=";n
Call send(outstring， n)                         '调用串行通信子程序
End Sub
```

⑥ 编写串行通信子程序

```
Private Sub send(outstring， n)
Dim instring As String                           '定义接收字符串 instring
MSComm1.CommPort = 1                             '定义该通信控件属性，用端口 COM1
MSComm1.Settings = "19200，n，8，1"
'波特率为 19200 b/s，无奇偶校验，8 位数据，1 个停止位
MSComm1.InputLen = 0          '当输入占用时，告诉控件读入整个缓冲区
MSComm1.PortOpen = True       '打开端口
```

```
MSComm1.Rthreshold = n          接收的字符串长度此处将是 n
MSComm1.Output = outstring + Chr(&HD)
'发送以回车符结尾的字符串
以下是采用查询方式从端口获得单片机方发来的数据
Do
DoEvents
Loop Until MSComm1.InBufferCount >= n
'等待，直到输入缓冲区计数大于或等于 n
Print "InBufferCount=";MSComm1.InBufferCount
'显示接收到的字符串长度
instring = MSComm1.Input            '取得接收到的字符串
Print "Ok，get:";instring          '显示接收到的字符串
MSComm1.PortOpen = False            '关闭端口
End Sub
```

上述子程序中也可以采用事件驱动的方法从端口获得单片机方发来的数据。把上述查询部分(从 Do 开始到关闭端口)改为下列子程序：

```
Private Sub MSComm1_OnComm()
Dim instring As String
Select Case MSComm1.CommEvent
'errer 捕捉每一个事件或错误，把代码写在对应的 case 语句后面
Case comEventCDTO       'CD(RLSD)超时
Case comEventOverrun    '数据丢失
Case comEventRxOver     '接收缓冲区溢出
Case comEventTxFull     '传输缓冲区满
'……还有其它一些错误 Case，省略
'event
Case comEvCD            'CD 线状态变化
Case comEvCTS           'CTS 线状态变化
Case comEvDSR           'DSR 线状态变化
Case comEvRing          '振铃
Case comEvReceive       '这里我们只处理接收到 Rthreshold 个字符的事件
instring = MSComm1.Input
Print "Ok!get:"；instring
MSComm1.PortOpen = False
Case comEvSend          '传输缓冲区有 sthreshold 个字符
Case comEvEOF           '输入数据流中发现 EOF 字符
End Select
End Sub
```

(3) 51 单片机编程。设晶振频率为 11.0592 MHz，为了简单直观演示双方串行通信的结果，在单片机接收到 PC 机发来的一个字符串后，再以相反的顺序把该字符串发送出去，PC 机接收后在屏幕上显示。程序如下：

```
ORG     0000H
        LJMP MAIN
        ORG 0023H
        LJMP SRT            ；串行中断入口
        ORG 0030H
MAIN：
        MOV SP，#60H
        MOV TMOD，#20H      ；设 T1 为方式 2，作波特率发生器
        MOV PSON，#80H      ；位 SMOD=1
        MOV TH0，#0FDH      ；波特率=19.2 kb/s
        MOV R0，#30H        ；串行接收、发送缓冲区首址
        SETB TR1
        CLR ET1             ；禁止 T1 中断
        MOV SCON，#50H      ；串行方式 1，REN=1
        SETB ES             ；允许串行中断
        SJMP $
        SRT：               ；串行中断子程序
        CLR ES              ；关闭串行中断
        CLR TI
        CLR RI
        GET：
        MOV A，SUBF
        CJNE A，#13，GO     ；若接收到回车符，结束
        SJMP EXIT
GO：
        MOV @R0，A          ；接收到的字符存入缓冲区
        INC R0
        WAIT：
        JBC RI，GET
        SJMP WAIT
EXIT：
        MOV A，R0
        SUB A，#30H         ；接收到的有效字符个数
        MOV R2，A
        DEC R0
        SEND：MOV SUBF，@R0     ；按反序发送刚接收到的字符串
```

```
WAIT1：
        JNB TI，WAIT1
        CLR TI
        DEC  R0
        DJNZ R2，SEND
WAIT2：
        MOV SUBF，#13              ；发送回车，结束标志
        JNB TI，WAIT2
        CLR      TI
        SETB ES                    ；允许串行中断，准备下次通信
        RETI
```

(4) 运行。在 VB 环境下运行上述程序，出现带“command1”按钮的窗口，用鼠标点击它，出现以“输入窗口”为名的窗口，从键盘输入字符串。例如‘ABCDEFG’，用鼠标点击“确定”按钮，该字符串连同回车符被发送出去。然后，在第一个窗口出现几行输出，最后一行是接收到单片机发回的‘GFEDCBA’，说明双方通信成功。

本程序仅为演示参考程序，并无实际意义。但它演示了 PC 机和单片机串行通信的一般方法以及在 VB 中处理数据的方法，使得单片机和 PC 机的应用能够更加紧密地结合在一起。读者可在此基础上加入一些握手信号和检错码，如奇偶校验、累加和校验及循环冗余校验(CRC)等，便可成为自己的应用程序。

6.2.3 用 VC 开发串口通信软件

我们将结合一个具体的例子来说明如何用 VC 开发串口通信软件。本程序的编程环境是 Win98 和 Visual C++6.0。这个编程示例可以由用户选定进行传输的通信端口，并设定这个端口的相关参数，包括波特率、数据位、停止位、奇偶校验和流量控制等。还具有发送数据和接收数据的双重功能。

1．在程序中嵌入通信控件

启动 Visual C++6.0，利用 MFC AppWizard(exe)新建一个项目文件，并命名为 RS232(图 6.2.7)，在 AppWizard 第一步选择基于对话框的应用程序类型(Dialog based)，在第二步将 ActiveX Controls 复选框选中，表示本程序支持 ActiveX 控件。其它均接收缺省设置，AppWizard 将自动生成一个以对话框为主窗口的应用程序。下面我们将在该程序中加入通信控件。在 Resource View 中打开对话框(IDD_RS232_DIALOG)，将其修改为如图 6.2.8 所示的对话框。注意将对话框的语言属性改为 Chinese(P.R.C)。

单击【Project】菜单，从菜单中选择【Add to Project】，再单击【Components and Controls】(图 6.2.8)，从弹出的对话框中单击【Registered ActiveX Controls】，然后在列表框中选择【Microsoft Communications Control，version 6.0】(图 6.2.9)，单击 OK。这时窗口会询问你是否加入 CMSComm 类，单击 OK，返回后就会看到控件工具条上添加了一个通信控件(形状如电话机)，将其拖放到对话框上，同时注意到 Project 中新增加了一个类。通过查看类

MSComm 的源文件 mscomm.cpp，我们可以了解这个控件的属性和使用方法，其中的 Get 函数可以用来访问该属性的当前值，而 Set 函数则用来设置该属性的新值。

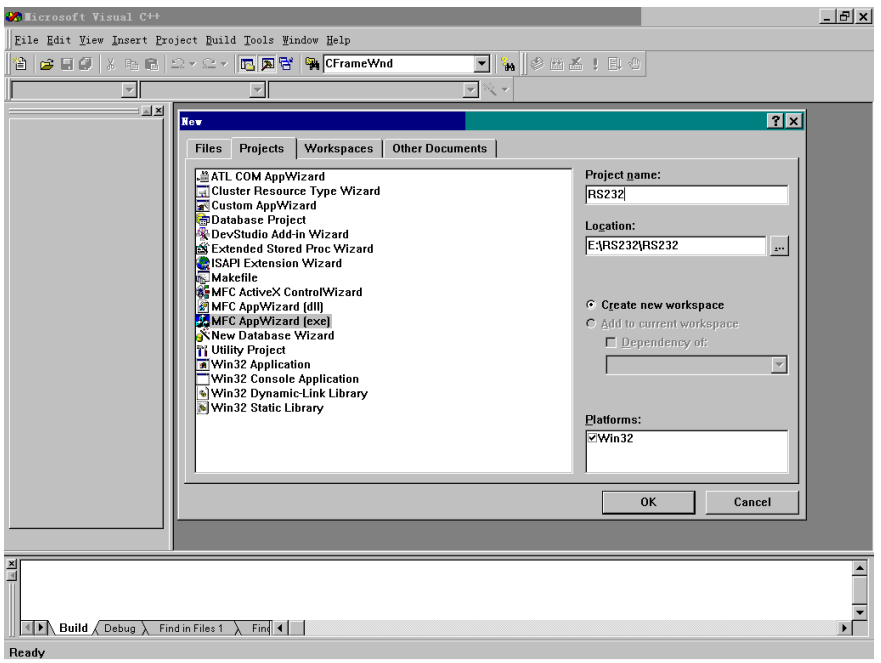


图 6.2.7　新建项目文件

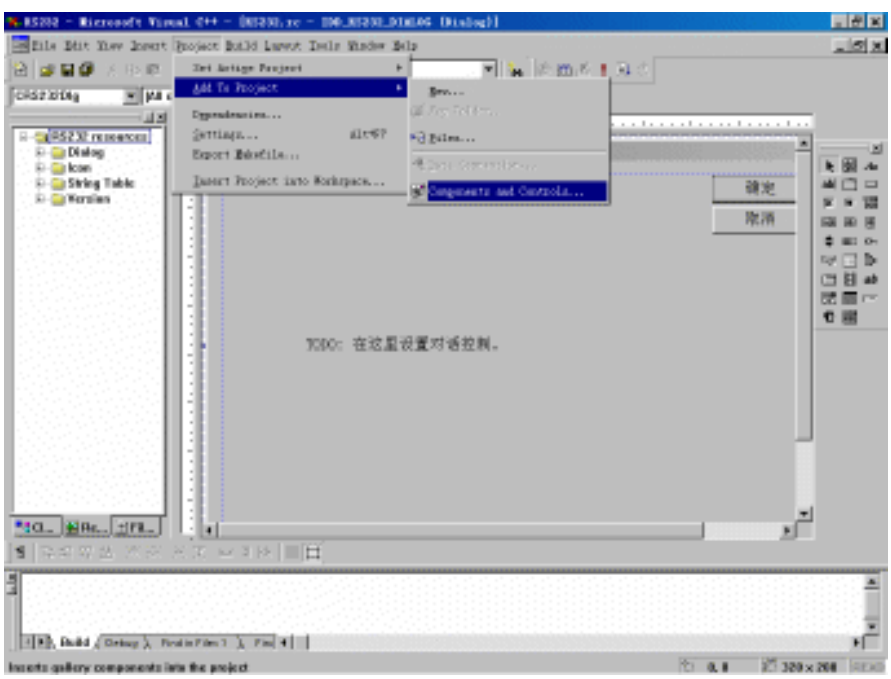


图 6.2.8　选择【Components and Controls】对话框

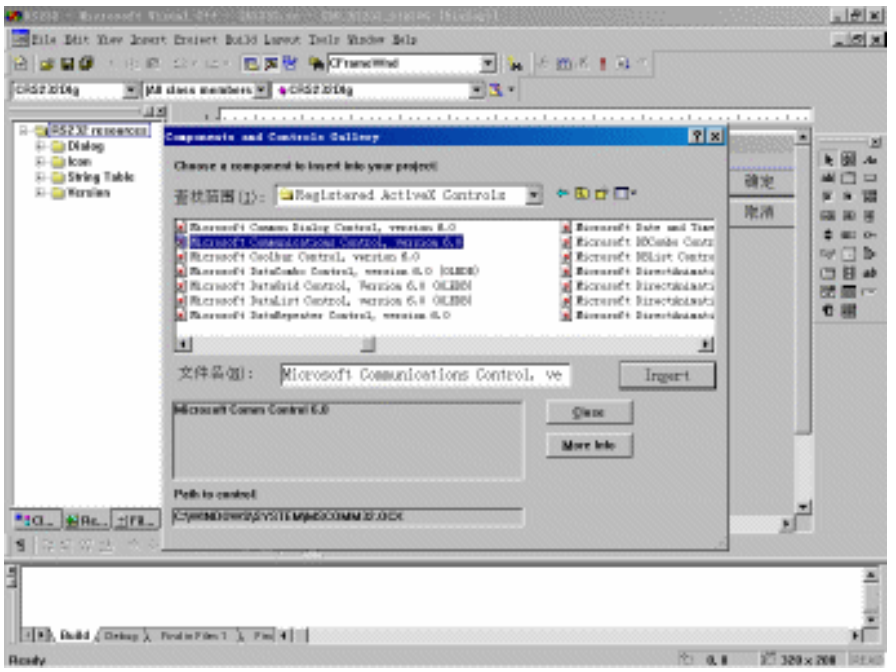


图 6.2.9　添加 MSComm 控件

2．设置通信控件的属性值

激活通信控件的属性对话框，一些重要的属性及其说明如表 6.2.3 所示。

表 6.2.3　串口属性及功能

属　性	设定值	功　能
CommPort	2	串口号，一般串口 1 为鼠标所用，故用串口 2
InBufferSize	1024	接收缓冲区大小
InputLen	0	从接收缓冲区读取的字节数，0 表示全部读取
InputMode	1	接收数据的类型，0 表示文本类型，1 表示二进制类型
OutBufferSize	1024	发送缓冲区大小
RThreshold	1	设定当接收几个字符时触发 OnComm 事件，0 表示不产生事件，1 表示每接收一个字符就产生一个事件
SThreshold	0	设定在触发 OnComm 事件前，发送缓冲区内所允许的最少的字符数，0 表示发送数据时不产生事件，1 表示当发送缓冲区空时产生 OnComm 事件
Settings	9600，n，8，1	串口的参数设置，依次为波特率、奇偶校验(n—无校验，e—偶校验，o—奇校验)、数据位数、停止位数

表 6.2.3 中的属性设定值是本例程中所用的值，可根据需要灵活设定。至于其它的属性可采用缺省值，其说明可参考联机帮助文件。

通信控件的工作原理类似于中断方式，当有通信事件发生时(如发送数据、接收数据等)，就会触发 OnComm 事件，在该事件的处理函数中调用 GetCommEvent()函数，通过返回值即可确定是哪类事件，再作出相应的处理。

3．主程序的编写

首先我们为对话框中的控件添加对应的变量和响应函数。具体做法如表 6.2.4 所示，完成后的窗体如图 6.2.10 所示。

表 6.2.4　添加控件表

控　件	Caption	控件 ID	对应变量或函数
文本框	发送数据	IDC_STATIC1	无
文本框	接收数据	IDC_STATIC2	无
发送数据编辑框	无	IDC_SENDDATA	m_SendData
接收数据编辑框	无	IDC_RECEIVEDATA	m_ ReceiveData
发送按钮	发送	IDC_SEND	OnSend( )
清除按钮	清除	IDC_CLEAR	OnClear( )
通信控件	无	IDC_MSCOMM	m_Comm

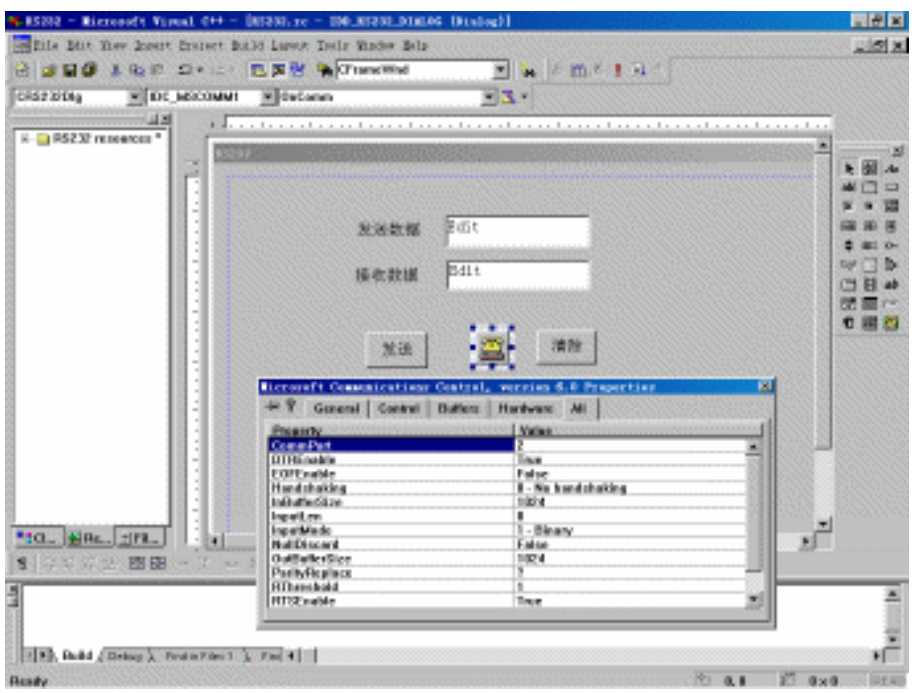


图 6.2.10　完成后的窗体和串口控件属性框

再打开 ClassWizard，选中 IDC_MSCOMM，可看到该控件能响应的消息为 OnComm，添加该函数并将其改名为 OnComm( )，在该函数中添加代码，就能实现对串口事件的处理。其中一个需要注意的问题是函数 GetInput( )返回的是 VARIANT 型变量，而在编辑框中显示的是 CString 型变量，因此必须进行转换。先将 VARIANT 型变量转换为 COleSafeArray 型变量，再将其转换为 BYTE 型数组，然后将数组转换为 CString 型变量。这个转换过程看起来比较复杂，但它可以用不同的变量类型来显示接收数据。该程序的主要代码添加在 RS232Dlg.cpp 中，如下所示：

```
void CRS232Dlg::OnSend( )
{
```

```
        if(!m_Comm.GetPortOpen( ))
                m_Comm.SetPortOpen(TRUE)；//打开串口
        UpdateData(TRUE)；
       m_Comm.SetOutput(COleVariant(m_SendData))；//发送数据
}

void CRS232Dlg::OnClear( )
{
        m_ReceiveData.Empty( )；     //清除接收对话框中的数据
        m_SendData.Empty( )；        //清除发送对话框中的数据
    UpdateData(FALSE)；
}

void CRS232Dlg::OnComm( )
{
      VARIANT m_Input1；
       COleSafeArray m_Input2；
       long length，i；
       BYTE data[1024]；
       CString str；
      if(m_Comm.GetCommEvent( )==2)   //接收缓冲区内有字符
       {
              m_Input1=m_Comm.GetInput()；//读取缓冲区内的数据
              m_Input2=m_Input1；
//将 VARIANT 型变量转换为 COleSafeArray 型变量
             length=m_Input2.GetOneDimSize( )；//确定数据长度
             for(i=0；i<length；i++)
                    m_Input2.GetElement(&i，data+i)；
//将数据转换为 BYTE 型数组
            for(i=0；i<length；i++) //将数组转换为 CString 型变量
              {
                    char a=*(char*)(data+i)；
                   str.Format("%c"，a)；
                    m_ReceiveData+=str；
              }
       }
      UpdateData(FALSE)；//更新编辑框内容
}
```

6.2.4　用 C++Builder 开发串口通信软件

1. **使用** VB ActiveX **控件**

在 C++Builder **中插入** MSComm **控件的过程如下：**

(1) **确认已将控件文件**(MSCOMM32.OCX)**复制到硬盘** Windows\system**目录下**(Win2000 **和** WinNT **下是** system32 **目录下**)**。注册方法请参见** 6.2.1 **注册控件一节。**

(2) **在** C++Builder **中新建一个项目。**

(3) **单击** Project **菜单，从菜单中选择【**Import ActiveX Control…**】(图** 6.2.11)**。**

(4) **在弹出的** Import ActiveX **对话框中选择【**Microsoft Comm Control6.0(Version 1.1)**】，在** Palette Name **下拉框中选择默认项【**ActiveX**】，然后单击【**Install…**】(图** 6.2.12)**。**

(5) **完成后就会看到** ActiveX **工具条上添加了一个通信控件(形状如电话机)，将其拖放到对话框上即可(图** 6.2.13)**。**

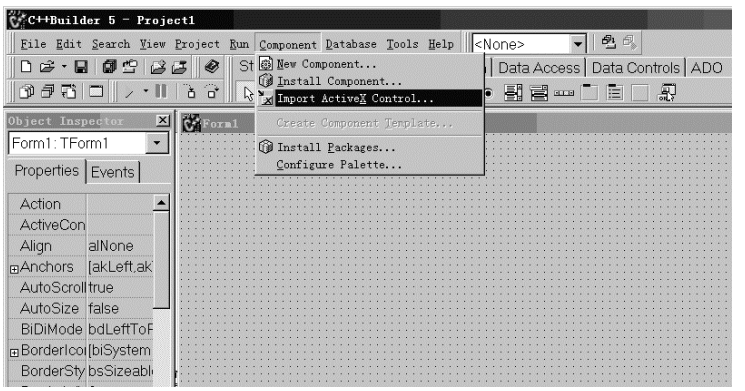


图 6.2.11　**选择【**Import ActiveX Control…**】菜单项**

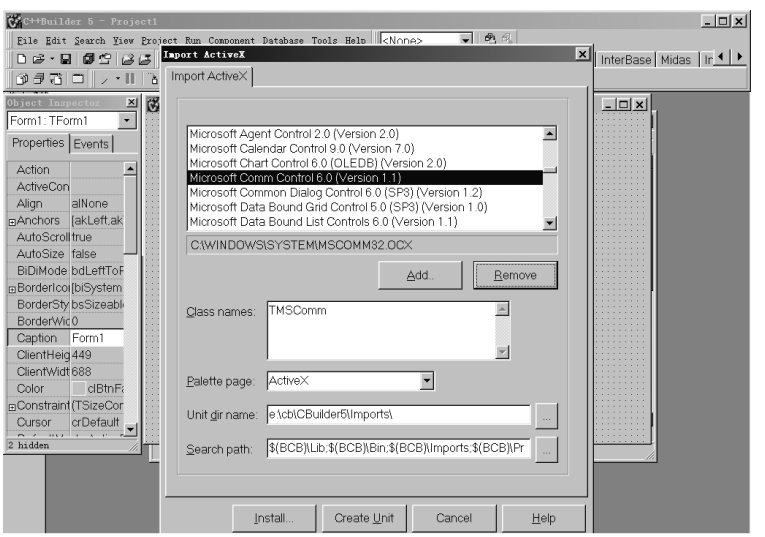


图 6.2.12　**添加串口通信控件**

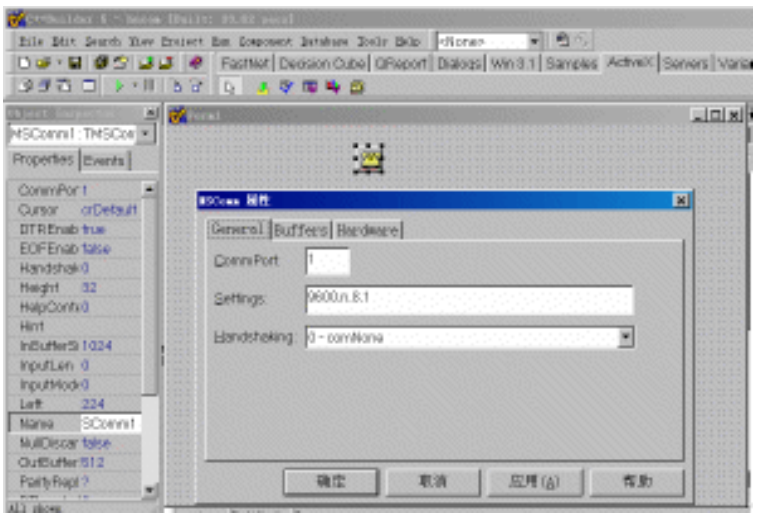


图 6.2.13 串口通信控件属性框

在要发送数据时，我们声明一个发送数据缓冲区，然后重置它的大小，填充它的元素，发送它，例如：

```
    unsigned char buff[200]；
//声明为全局变量，应该是动态数组，否则会出现乱码
    OleVariant TxBuff；//声明一个 OleVariant 变量
    TxBuff=VarArrayCreate(OPENARRAY(int，(0，200))，varByte)；
//重置它的大小，为 0～n，int 为 n 的类型
    //varByte 为 TxBuff 每一个元素的类型
    for(int i=0；ibr<200＋1；i＋＋)TxBuff.PutElement(buff[i]，i)；
    //填充元素，其中 buff 为定义的一个固定数组，其中有要发的数据
    MSComm1－>Output=TxBuff；
//发送数据，MSComm1 为放在窗体上的一个 MSComm 控件
    接收数据时请看下面的例子：
    unsigned char buff[200]；//声明一个存储接收数据的缓冲区，全局变量
    int ByteNum；//收到的字节数
    int BuffPtr；//接收缓冲区的指针，请声明为全局变量
    OleVariant RxBuff；//一个用于接收的 OleVariant 变量
    if(MSComm1－>InBufferCount>0)RxBuff=Communica1－>Input；
//如果缓冲区中有多于一个字节的数据
    ByteNum=RxBuff.Array HighBound(1)；//将实际读的字节数取出
    for(int i=0；ibr<ByteNum＋1；i＋＋)
        {buff[BuffPtr＋＋]=RxBuff.GetElement(i)；}
//将接收数据读入自己的缓冲区
```

在 Object Inspector 的 Event 标签中只有一个事件 OnComm，这个事件在 MSComm32 控件收到数据时会被调用，但必须设置 RThreshold 属性。这是一个门槛，表示收到几个字节就发送通知消息，如果为零，就不发送通知消息，这样 OnComm 函数就不会得到执行。

2. 使用 API 函数

在 Win32 下，对串口的操作就如同对文件一样打开或关闭，对串行数据的读写可在用户定义的读写缓冲区中进行。具体使用的函数如下。

首先，用 CreateFile( )打开通信串口，其中参数 lpFileName 指向串口逻辑名，如“ COM1 ”或“ COM2 ”等，参数 dwDesiredAccess 定义文件的读写权限，一般设为 ENERIC_READ|GENERIC_WRITE；参数 dwShareMode 定义资源共享方式，此处必须设为 0，为独占方式；lpSecurityAttributes 定义安全属性，Win 95 下为 NULL；dwCreationDistribution 定义文件创建方式；dwFlagsAndAttributes 定义文件属性和标记，应设为 FILE_FLAG_OVERLAPPED，表示异步通信方式；hTemplateFile 指向一个模板文件的句柄，在 Windows 95 下为 NULL。

然后，用 BuildCommDCB( )和 SetCommState( )函数通过通信设备控制块 DCB(Device Control Block)设置串口通信参数(如波特率、停止位、数据位、校验位等)，其中 BuildCommDCB( )中的字符串参数 lpDef 定义同 DOS 命令中 MODE 的参数格式，关于 DCB 更具体的设置需要根据用户对数据流定义、握手信号及通信控制要求具体定义，参见有关 Windows 技术资料。用 GetCommState( )可以得到当前的 DCB 参数值。如果需要还可通过 SetCommTimeouts( )和 GetCommTomeouts( )重新设置读写的超时参数；读写缓冲区的设置使用 SetupComm( )，参数 dwInQueue 和 dwOutQueue 分别定义为输入和输出缓冲区的大小。

在串口初始化完毕后，还要建立与通信有关的事件对象。一般使用 CreateEvent( )函数，它返回一事件句柄，其中参数 lpEventAttributes 指向安全属性结构地址，在 Win95(无安全属性)中为 NULL；布尔参数 bManualReset 定义事件重置方式，true 表示手工重置，false 表示自动重置(相关函数为 SetEvent( )和 ResetEvent( ))；参数 bInitialState 定义事件初始状态，true 表示发信号，false 为不发信号；lpName 是为多进程设置的事件名，对于单进程定义为 NULL。然后用 SetCommMask( )定义用户程序可监视的通信事件类别。

以上设置完成后，用户程序就可以等待通信事件的产生，一般调用函数 WaitCommEvent()监视通信事件，其中参数 lpEvtMask 指向产生事件的掩码地址，用于判断事件产生的性质，lpOverlapped 指向重叠结构地址，可简单定义为 NULL。对于串口事件的响应一般有四种方式：查询、同步 I/O、异步 I/O 和事件驱动 I/O，需要根据用户不同控制要求而定。查询方式占用较长的计算机时间，同步 I/O 方式直到读取完指定的字节数或超时时才返回，容易造成线程阻塞，异步 I/O 用于后台处理，事件驱动是由系统通知用户程序发生的事件并进行串口操作。比较而言，事件驱动 I/O 方式较灵活。

当有通信事件产生时，就可用函数 ReadFile( )和 WriteFile( )直接对串口缓冲区进行读写操作了。其中，lpBuffer 指向读写缓冲区，nNumberOfBytes 为要读写的字节数，lpNumberOfBytes 为实际读写的字节数，lpOverlapped 指定同步或异步操作。通信结束后，调用函数 CloseHandle( )将串口关闭。

例 6.2.6

```
HANDLE hcom；//定义句柄
DCB dcb；
OVERLAPPED e； //定义重叠结构
void _fastcall TForm1::OkBtnClick(TObject？Sender)
{ hcom=CreateFile("COM2"，GENERIC_READ|GENERIC_WRITE，0，NULL，OPEN_EXISTING，
FILE_ATTRIBUTE_NORMAL|FILE_FLAG_OVERLAPPED，NULL)； //打开通信口
BuildCommDCB("9600，O，8，1"，&dcb)；
//第一个字符串参数实际使用时由实际情况选择后组合，这里仅简单说明其格式
SetCommState(hcom，&dcb)；
SetupComm(hcom，512，512)；//设置读写缓冲区
e.hEvent=CreateEvent(NULL，false，false，NULL)；//设置事件
SetCommMask(hcom，EV_RXCHAR| EV_TXEMPTY)；//设置事件掩码
OkBtn—>Enabled=false；
}
```

6.2.5 用 Delphi 开发串口通信软件

1. **用汇编语言实现串行通信**

由于 Win2000 **和** WinNT **不允许嵌入汇编，因而给这种方法造成了一定的局限。**

在 Delphi **语言的任何一个过程或函数中均可使用汇编语句，但必须将它们放在** asm **和** end **关键字之间。使用汇编命令的函数或过程必须按下面的要求返回它们的值：**

(1) **字节型的返回值放在** AL，BL **等寄存器中**(8 **位**)；

(2) **字型的返回值放在** AX，BX **等寄存器中**(16 **位**)；

(3) 32 **位的返回值放在** EAX，ECX **等寄存器中。**

例 6.2.7

```
Var
ByteVar:   Byte；
WordVar:   Word；
IntVar:    Integer；
……
MOV AL，ByteVar
MOV BX，WordVar
MOV ECX，IntVar
End；
```

采用汇编语言对串口操作有三种方法：直接对串口芯片进行操作；调用 DOS **中断；调用** BIOS **中断。下面主要以** BIOS **中断为例，**BIOS **即** Basic Input/Output System **服务程序。使用** BIOS **而不直接访问硬件，将使程序更为简练。对串口而言，**BIOS **比** DOS **功能强些。**

初始化串口：使用 COM1 **口，波特率为** 2400 b/s，8 **位数据位，无奇偶校验。**

例 6.2.8

```
asm
  MOV AH，0
  MOV AL，0A3H
  INT 14H
  End；
```

发送和接收与汇编语言环境下编程完全一样。

例 6.2.9

把 x **表示的** 8 **位数发送出去。**

```
Var
x :Byte；
begin
  asm
  MOV ah，01h
  MOV al，x
  MOV dx，01h
  INT   14h
  end；
end；
```

把接收到的数据放在 y **中。**

```
Var
y:Byte；
begin
      asm
      MOV ah，02h
      MOV dx，01h
      INT   14h
      MOV y，al
      end；
end；
```

2. **使用通信控件**

(1) VB **的** MSComm **控件。这里介绍如何把** MSComm **这个** ActiveX **控件加入到** Delphi **中，充分发挥出二者的优越性。**

首先，在 VB **安装目录下找到三个文件**: Mscomm.srg，Mscomm32.ocx，Mscomm32.dep。**把这三个文件拷贝到** Windows **的** system **目录下(注意** WinNT **下是** System32)。**然后，用** Windows **下的注册工具** regsvr32 **注册该** OCX **控件，例如：**

Regsvr32 … system **目录**.ocx .

注册成功后用记事本打开 Mscomm.srg，可以看到类似下面的内容:

[HKEY_CLASSES_ROOT4250E830-6AC2-11cf-8ADB-00AA00C00905]

@ = "kjljvjjjoquqmjjjvpqqkqmqykypoqjquoun"

为了能正确使用该控件，需要修改注册表的信息，在注册表的 HKEY_CLASSES_ROOT 下建一主键:

4250E830-6AC2-11cf-8ADB-00AA00C00905

主键的内容为：

kjljvjjjoquqmjjjvpqqkqmqykypoqjquoun

至此，已经完成了控件的系统注册工作，剩下的工作是在 Delphi 中导入这个 ActiveX 控件，进入 Delphi 环境，在【Component】菜单下选择【Import ActiveX Control】，这时在列表中可以看到 Microsoft Comm Control 6.0 选项，选中这个选项，点击【Install】按钮，完成安装。在 ActiveX 控件页上可以看到一个以电话机为图标的控件，这就是 MSComm 控件。这样，就可以像在 VB 中一样，方便地使用 MSComm 控件了。

(2) SPCOMM 控件。该控件具有丰富的与串口通信密切相关的属性及事件，提供了对串口的各种操作，而且还支持多线程，可从相关站点下载。下面，结合实例介绍 SPCOMM 控件的使用。

- SPCOMM 的安装

① 选择下拉菜单【Component】中的【Install Component】选项，弹出如图 6.2.14 所示的窗口。在 Unit file name 处填写 SPCOMM 控件所在的路径，其它各项可用默认值，点击 OK 按钮。

② 安装后，在 System 控件面板中将出现一个红色控件 COM，如图 6.2.15 所示。现在就可以像 Delphi 自带控件一样使用 COM 控件了。如果不能使用，请参考 MSComm 控件的注册方法进行注册。

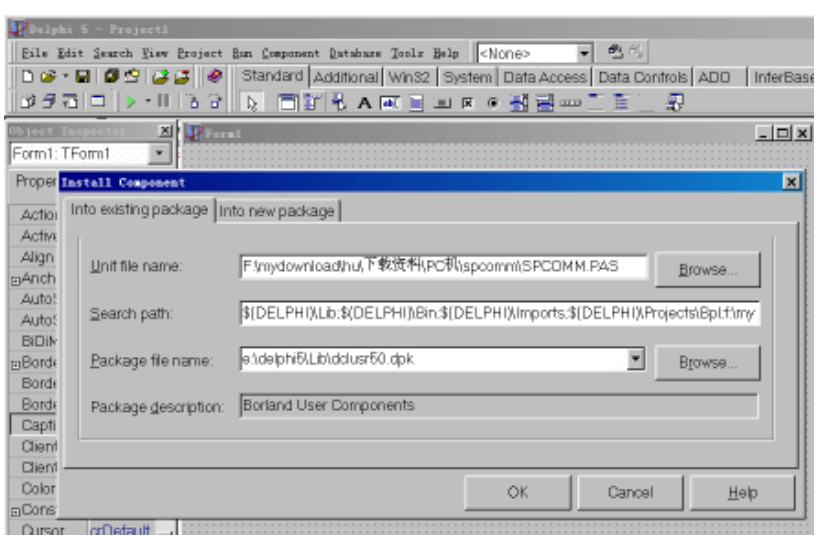


图 6.2.14 安装 SPCOMM 控件

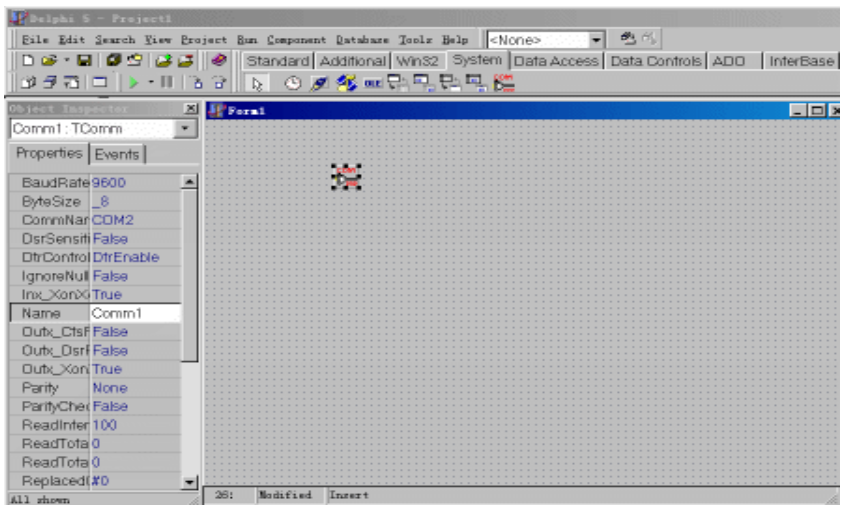


图 6.2.15　SPCOMM 控件

- SPCOMM **的属性、方法和事件**

①　属性。

CommName：**表示** COM1、COM2 **等串口的名字；**

BaudRate：**根据实际需要设定的波特率，在串口打开后也可更改此值，实际波特率随之更改；**

ParityCheck：**表示是否需要奇偶校验；**

ByteSize：**根据实际情况设定的字节长度；**

Parity：**奇偶校验位；**

StopBits：**停止位；**

SendDataEmpty：**这是一个布尔型属性，为** true **时表示发送缓存为空，或者发送队列里没有信息；为** false **时表示发送缓存不为空，或者发送队列里有信息。**

②　方法。

Startcomm：**用于打开串口，当打开失败时通常会报错。错误主要有** 7 **种：串口已经打开；打开串口错误；文件句柄不是通信句柄；不能够安装通信缓存；不能产生事件；不能产生读进程；不能产生写进程。**

StopComm：**用于关闭串口，没有返回值。**

WriteCommData(pDataToWrite:PChar；dwSizeofDataToWrite:Word)：**表示带有布尔型返回值的函数，用于将一个字符串发送到写进程，发送成功返回** true，**发送失败返回** false。**执行此函数将立即得到返回值，发送操作随后执行。该函数有两个参数，其中** pDataToWrite **是要发送的字符串，**dwSizeofDataToWrite **是发送字符串的长度。**

③　事件。

OnReceiveData :procedure (Sender: TObject；Buffer: Pointer；BufferLength: Word) of object **当有数据输入缓存时将触发该事件，在这里可以对从串口收到的数据进行处理。**Buffer **中是收到的数据，**BufferLength **是收到的数据长度。**

OnReceiveError : procedure(Sender: TObject； EventMask : DWORD) **当接收数据出现错**

误时将触发该事件。

- SPCOMM 的使用

下面是一个利用 SPCOMM 控件的串口通信的例子。

例 6.2.10 以实现 PC 机与单片机 89C51 之间的通信为例，首先要调通它们之间的握手信号。假定它们之间的通信协议是：PC 发送到 89C51 一帧数据 6 个字节，89C1 发送到 PC 一帧数据也为 6 个字节。当 PC 发出(F0，01，FF，FF，01，F0)后，89C51 能收到一帧(F0，01，FF，FF，01，F0)，表示数据通信握手成功，两者之间就可以按照协议相互传输数据。

创建一个新的工程 COMM32.DPR，把窗体的 NAME 属性定为 FCOMM，把窗体的标题命名为测试通信，按照图 6.2.16 添加控件，图中 SPCOMM 控件命名为 COMM1。

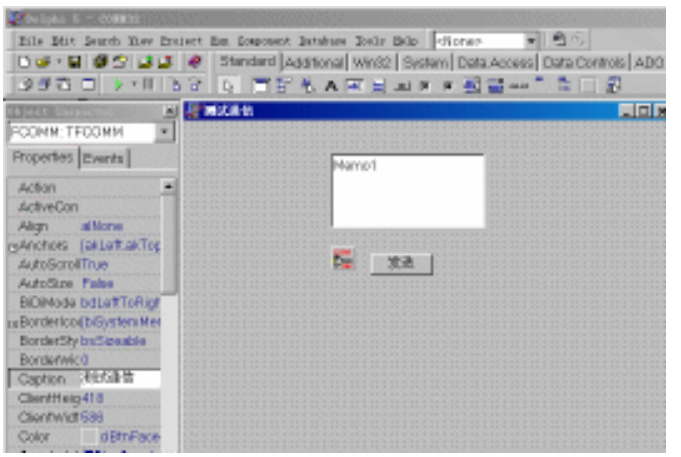


图 6.2.16 测试通信对话框

① 设定 COMM1 属性。

波特率：4800 b/s；奇偶校验位：无；字节长度：8；停止位：1；串口：COM1。

Memo1 中将显示发送和接收的数据。将新的窗体存储为 Comm.pas。

② 编写源代码。

```
//变量说明
var
fcomm: TFCOMM；
viewstring:string；
i:integer；
rbuf，sbuf:array[16] of byte；
//打开串口
procedure TFCOMM.FormShow(Sender: TObject)；
begin
comm1.StartComm；
end；
//关闭串口
```

```
procedure TFCOMM.FormClose(Sender: TObject； var Action: TCloseAction)；
begin
comm1.StopComm；
end；
//自定义发送数据过程
procedure senddata；
var
i:integer；
commflg:boolean；
begin
    viewstring:=''；
    commflg:=true；
    for i:=1 to 6 do
    begin
    if not fcomm.comm1.writecommdata(@sbuf[i]，1) then
        begin
        commflg:=false；
        break；
    end；
//发送时字节间的延时
    sleep(2)；
    viewstring:=viewstring＋ inttohex(sbuf[i]，2)＋''；end；
    viewstring:= '发送'＋ viewstring；
    fcomm.memo1.lines.add(viewstring)；
    fcomm.memo1.lines.add('')；
    if not commflg then messagedlg('发送失败!'，mterror，[mbyes]，0)；
end；
//发送按钮的点击事件
procedure TFCOMM.Btn_sendClick(Sender: TObject)；
    begin
    sbuf[1]:=byte($f0)；//帧头
    sbuf[2]:=byte($01)；//命令号
    sbuf[3]:=byte($ff)；
    sbuf[4]:=byte($ff)；
    sbuf[5]:=byte($01)；
    sbuf[6]:=byte($f0)；//帧尾
    senddata；//调用发送函数
end；
//接收过程
```

```
procedure TFCOMM.Comm1ReceiveData(Sender: TObject； Buffer: Pointer；BufferLength: Word)；
var
i:integer；
begin
viewstring:= ''；
move(buffer^，pchar(@rbuf^)，bufferlength)；
for i:=1 to bufferlength do
viewstring:=viewstring＋ inttohex(rbuf[i]，2)＋''；
viewstring:= 接收'＋ viewstring；
memo1.lines.add(viewstring)；
memo1.lines.add('')；
end；
```

如果 memo1 **上显示发送** F0 01 FF FF 01 F0 **和接收到** F0 01 FF FF 01 F0**，这表示串口已正确地发送出数据并正确地接收到数据，串口通信成功。**

3. **使用** API **函数**

Delphi **的强大功能和支持多线程的面向对象编程技术，使得实现串行通信非常简单方便。主要步骤如下：首先，利用** CreateFile **函数打开串行口，以确定本应用程序对此串行口的占有权，并封锁其它应用程序对此串口的操作；其次，通过** GetCommState **函数填充设备控制块** DCB**，再通过调用** SetCommState **函数配置串行口的波特率、数据位、校验位和停止位，然后，创建串行口监视线程监视串行口事件，在此基础上就可以在相应的串口上操作数据的传输；最后，用** CloseHandle **函数关闭串行口。例** 6.2.11 **所示程序用** Delphi 3.0 **编制在** Win95 **环境下调试通过，供读者参考。**

例 6.2.11

```
unit comdemou；
interface
uses
Windows， Messages， SysUtils， Classes， Graphics， Controls， Forms， Dialogs；
const
Wm_commNotify=Wm_User+12；
type
TForm1 = class(TForm)
procedure FormCreate(Sender: TObject)；
private
Procedure comminitialize；
Procedure MsgcommProcess(Var Message:Tmessage)；Message Wm_commnotify；
{ Private declarations }
public
{ Public declarations }
end；
```

```
// 线程声明
TComm=Class(TThread)
protected
procedure Execute；override；
end；
var
Form1: TForm1；
hcom，Post_Event:Thandle；
lpol:Poverlapped；
implementation
{$R *.DFM}
Procedure TComm.Execute； // 线程执行过程
var
dwEvtMask:Dword；
Wait:Boolean；
Begin
fillchar(lpol，sizeof(toverlapped)，0)；
While True do Begin
dwEvtMask:=0；
Wait:=WaitCommEvent(hcom，dwevtmask，lpol)； // 等待串行口事件
if Wait Then Begin
waitforsingleobject(post_event，infinite)； // 等待同步事件置位
resetevent(post_event)； // 同步事件复位
PostMessage(Form1.Handle，WM_COMMNOTIFY，0，0)；// 发送消息
end；
end；
end；

procedure Tform1.comminitialize； // 串行口初始化
var
lpdcb:Tdcb；
Begin
hcom:=createfile('com2'，generic_read or generic_write，0，nil，open_existing，
file_attribute_normal or file_flag_overlapped，0)；// 打开串行口
if hcom=invalid_handle_value then
else
setupcomm(hcom，4096，4096)； // 设置输入，输出缓冲区皆为 4096 字节
getcommstate(hcom，lpdcb)； // 获取串行口当前默认设置
lpdcb.baudrate:=2400；
```

```
lpdcb.StopBits:=1；
lpdcb.ByteSize:=8；
lpdcb.Parity:=EvenParity； // 偶校验
Setcommstate(hcom，lpdcb)；
setcommMask(hcom，ev_rxchar)；
// 指定串行口事件为接收到字符
end；
Procedure TForm1.MsgcommProcess(Var Message:Tmessage)；
var
Clear:Boolean；
Coms:Tcomstat；
cbNum，ReadNumber，lpErrors:Integer；
Read_Buffer:array[1..100]of char；
Begin
Clear:=Clearcommerror(hcom，lpErrors，@Coms)；
if Clear Then Begin
cbNum:=Coms.cbInQue；
ReadFile(hCom，Read_Buffer，cbNum，ReadNumber，lpol)；
// 处理接收数据
SetEvent(Post_Event)； // 同步事件置位
end；
end；
procedure TForm1.FormCreate(Sender: TObject)；
begin
comminitialize；
post_event:=CreateEvent(nil，true，true，nil)； // 创建同步事件
Tcomm.Create(False)； // 创建串行口监视线程
end；
end.
```

6.2.6 Windows 环境下多机通信的例子

随着微机和单片机技术的不断发展，采用一台微机与多台单片机构成的主从式测控系统越来越多，它既利用了单片机价格低、功能强、抗干扰性能好等优点，又利用了 Windows **的高级用户界面，操作方便灵活。在这样的主从系统中，微机和多单片机之间的通信是其中的关键之一，本小节着重阐述用** VB **通信控件实现它们之间的串行通信。**

1. **硬件结构与通信协议**

硬件结构如图 6.2.17 **所示，为实现远距离通信，采用** RS－422A **标准接口，在微机端，将** RS－232 **电平转换为** RS－422 **电平，也可直接使用** RDS－100 **异步四线远程数据收发器。**

图 6.2.17　多机通信的接口图

在此测控系统中，由于各从机共享串行总线，为避免冲突，由主机轮流联络各从机，而从机不能主动向主机或其它从机进行联络，即使有数据要上报给主机，也必须等待主机与它联络之后才能发送数据。

为了保证主机与所选择的从机实现可靠的通信，必须给每一个从机分配一个唯一的地址。本系统中，为了将从机地址与通信过程中的有效信息、命令区别开，规定 ASCII 码 00H～0FH 为地址码。

在通信协议中，串行数据帧由 11 位组成：1 位起始位，8 位数据位，1 位可编程位，1 位停止位。由主机首先发送从机地址联络；从机联络上后，回发本机地址作为响应帧；主机收到正确的响应帧后，发出发送参数命令帧；从机收到发送参数命令帧，开始发送参数帧，依次为：参数个数(1 Byte)、参数(1～64 Byte)、校验和(1 Byte)；主机正确接收完参数后，发送结束通信命令帧；从机收到结束通信命令帧后结束通信状态，主机继续与下一从机联络。

2. 微机通信程序设计

由于 Windows 95/98 系统管理了所有的串行口，并负责处理串行通信的事件，所以用户不能像 DOS 环境下那样直接操作串行口地址。而 Visual Basic 6.0 中名为 MSCOMM32.OCX 的通信控件(若工具箱中无此控件，可通过 Project 菜单下的 Component…子菜单将该控件加入到工具箱中)提供了标准的事件处理函数和过程，并通过属性的方法设置串行通信口参数，较容易地解决了串行口通信问题。

由于 VB 是事件驱动的，所以具体程序的编制必须围绕相应的事件进行，本系统中有关通信的工作过程主要是：启动系统，轮流联络各单片机，发送命令，接收参数。

启动系统也即窗体加载，主要完成一些初始化工作。由于微机联络某单片机时，首先发送 1 帧该单片机的地址帧，第 9 位必须为 1，故在通信开始前，微机必须置为传号状态。

例 6.2.12　初始化程序。

```
Sub FormLoad( )
```

```
MSComm1.CommPort=1                    ' 选择串行口 1
MSComm1.Settings="9600，M，8，1"      ' 9600 波特率，传号状态，8 位数据位，1 位停止位
MSComm1.RTSEnable=True                ' RTS 有效
MSComm1.DTREnable=True                ' DTR 有效
MSComm1.InBufferSize=512              ' 接收缓冲区为 512 字节
MSComm1.NullDiscard=False             ' 接收空字符
MSComm1.InBufferCount=0               ' 清接收缓冲区
MSComm1.OutBufferCount=0              ' 清发送缓冲区
MSComm1.PortOpen=True                 ' 打开串行口 1
End Sub
```

初始化完成后，微机可以轮流联络各单片机通信了，其结构如下：

```
For I = 0 to 15
Select Case I
Case 0
    add = add0                        ' add0 为 0 号从机地址
    …
Case 15
    add = add15 ' add15 为 15 号从机地址
End Select
    …(联络通信程序)
    …(命令发送和参数接收程序)
Nex5 … Next I
```

以下为联络通信程序(以 1 号从机为例)：

```
add = add1
buffer = Chr(add)
MSComm1.Output = buffer               ' 发送 1 号从机地址
Do Until MSComm1.InBufferCount > 0    ' 如收到从机的响应帧，跳出循环
    DoEvents
    fg = fg + 1
    If fg > 1000 Then
    yy1="通信失败或"+Str(add)+ "号从机没准备好，请联络下一从机!"
    MsgBox (yy1)
    …(1 号从机故障登记、处理后，Goto Nex5)
    End If
Loop
si = MSComm1.Input                    ' 读接收缓冲区内容，即从机的响应帧
If si= buffer Then                    ' 收到的响应帧与发送的地址帧相符，联络成功
    passsetting=1                     ' 允许发送或接收命令/参数帧
   MSComm1.PortOpen = False           ' 关闭串行口(以便进行发送命令/参数帧的设置)
else
```

```
yy2="通信正常，但"+Str(add)+ "号从机的响应帧错误，请检查，并联络下一从机！"
MsgBox (yy2)
…(1 号从机故障登记、处理后，Goto Nex5)
End If
…
```

微机联络通信成功后，置微机为空号状态，再打开通信端口，即可向选择的单片机发送命令帧或从选择的单片机接收参数帧。设定：单片机发送参数的命令为 20H，单片机结束通信的命令为 26H。程序如下：

```
MSComm1.Settings = "9600，S，8，1"      ' 9600 b/s 的波特率，空号状态，8 位数据位，1 位停止位
MSComm1.PortOpen = True                 ' 重新打开串行口，可以发送命令/参数帧了
If passsetting=0 Then
    MsgBox ("请与从机联络上之后，再发送命令/参数帧！")
    Goto Nex5
else
    Com1=&H20
    C0m1c=Chr(Com1)
    MSComm1.Output = Com1c              ' 发出发送参数的命令帧
…(接收参数程序，可参考联络通信程序中接收响应帧部分)
Com2=&H26
C0m2c=Chr(Com2)
MSComm1.Output = Com2c                  ' 发送结束本次通信的命令帧
passsetting=0                           ' 不允许发送或接收命令/参数帧
MSComm1.PortOpen = False                ' 关闭串行口(以便联络下一从机的设置)
End if
…
```

3. 单片机通信程序设计

为使微机与单片机可靠地通信，首先必须保证波特率一致，单片机采用 18.432 MHz 晶振频率，取 SMOD=1，当波特率为 9600 b/s 时，根据公式：

$$B=\frac{f_{\mathrm{ocs}}\times 2^{\mathrm{SMOD}}}{32\times 12\times(2^{8}-n)}$$

计算得到时间常数 n=246=F6H。

单片机置为工作方式 3，且多机通信控制位 SM2 为 1。

例 6.2.13　初始化程序。

```
MOV SP，#60H          ；设置堆栈指针
MOV TMOD，#20H        ；定时器 1 为定时方式 2
MOV TL1，# 0F6H       ；设置定时器 1 的初始值
MOV TH1，#0F6H
MOV PCON，#80H        ；SMOD=1
```

```
MOV SCON，#0F0H        ；串行口工作于方式 3，置 SM2=1，允许接收
SETB TR1               ；开定时器 1
SETB EA                ；允许中断
SETB ES                ；开串行中断
```

单片机通过串行中断方式接收微机发送的地址帧和命令帧。当微机发送地址帧时，第 9 位为 1，所有的单片机都产生中断，并将接收到的地址与各自的地址对比：若相符，则置 TB8 为 1，清 SM2 为 0，并回送给微机响应帧；若其地址不相符，则直接中断返回。

当微机发送命令帧时，由于第 9 位为 0，故 SM2 为 0 的单片机产生中断，接收到微机的命令帧；而 SM2 为 1 的单片机，不能产生中断，无法接收到微机的命令帧，达到微机与它所选择的某台单片机通信的目的。

单片机接收到发送参数的命令帧后，置 TB8 为 0，直接发送有关参数；若接收到结束通信的命令帧，则置 SM2 为 1 后中断返回，结束本次通信。中断服务程序的流程框图见图 6.2.18，具体设计参见第 5 章。

图 6.2.18　中断服务程序框图

6.3　通过 PC 机标准键盘接口传输数据

单片机应用系统与 PC 机之间的通信主要采用异步串行通信方式。但 RS－232C、RS－422、RS－449 等标准接口都规定了自己的电气标准，而要能与单片机的 TTL 接口电平相连，必须在系统中加入接口转换芯片。

键盘接口是 PC 机标准接口，以其作为单片机系统与 PC 机的通信接口，具有无需电平转换和 PC 机提供+5 V 电源等优点。在一些无需键盘或键盘使用不多的场合，可考虑这种通信方式。

6.3.1　PC 机键盘接口结构

目前，PC 机广泛采用的键盘是一种非编码键盘，它用一块单片机担负起键盘矩阵扫描和扫描码传输的功能。按通信协议的不同，键盘可分两大类。

一类是过去常用的 83/84 键标准键盘(下面简称标准键盘)，如 PC/XT 机键盘，使用 XT 传输协议。此种协议只允许键盘发送数据和接收来自 PC 机的复位指令，功能有限。

另一类是 101/102 或更多键的扩展键盘(下面简称扩展键盘)，如 PC/AT 机键盘，以及现在常见的 104 键 PS/2 接口 Windows 标准键盘，使用 AT－PS/2 兼容的传输协议。Windows 标准键盘增添了一些用于 Windows 环境下的功能键，如弹出 Windows 开始菜单、开关机和进入休眠状态等功能。有的 Windows 键盘设计了上网操作的快捷键，有的家用品牌机还在键盘上增加了多媒质功能，如音量大小调节、播放 CD 等，并预留几个键供用户自己定义功能。

虽然键盘上的新功能很多，但其基本工作原理和接口都是一样的，下面主要就扩展键盘加以介绍。

1. 键盘接口

键盘接口有大小两种，小口接头是 PS/2 接口，为 6 针 DIN 母插，外观见图 6.3.1(a)，与 PS/2 鼠标接口相同；大口接头为 5 针 DIN 母插，外观见图 6.3.1(b)，是老式 XT/AT 键盘的接头。键盘接口引脚定义见表 6.3.1。

图 6.3.1　DIN

(a) 6 针 DIN；(b) 5 针 DIN

表 6.3.1 键盘接口引脚定义

引脚号(6 针)	引脚号(5 针)	名称	方向	功 能
1	3	DATA	←→	串行数据
2	1	NC	---	未连接
3	4	GND	---	地
4	2	VCC	→	电源 +5 V
5	5	CLK	→	串行时钟
6	---	NC	---	未连接

PC 机扩展键盘接口如图 6.3.2 所示。键盘扫描处理器 8048 将按键扫描编码后，通过 PS/2 接口传送到 PC 机键盘处理器 8042。8042 接收到键盘扫描码后，将其转换成系统扫描码，放入 8042 的内部并行输出缓冲器，同时产生硬件中断。PC 机的 CPU 通过 I/O 口 60H 将扫描码读入，转换成 ASCII 码。

图 6.3.2 PC 机与键盘接口

2. 键盘结构及信号分析

PC 机键盘一般用 8049(8048)单片机作控制器，虽然现在生产的 Windows 键盘控制器已经高度集成化，但其内部结构仍可由图 6.3.3 来解释。

电阻 R1 和电容 C1 使系统上电时键盘有不小于 300 ms 的上电复位时间。键盘复位后进行自检，并把检测结果送往系统主板。键盘工作在扫描方式，由两片 74LS138(3-8 译码器)构成了带使能端的 4-16 线译码器。单片机 8048 通过 Pl.0～P1.4 口扫描键盘矩阵中的一列，使其变为低电平，同时通过 DB0～DB7 读入当时每一行的状态。如果当时正好该列有键按下，则由于相应行列接触，读入的该行状态就为低电平，而其余情况下都是高电平。由控制器查找对应的键盘扫描码，再进行数据发送。

键盘与系统主板之间由时钟(CLK)和数据(DATA)两条双向线进行通信。图 6.3.3 中，端口 P2.6 和 P2.5 分别发送键盘送往主板的时钟和数据；由 T0 和/INT 分别接收系统主板送入键盘的时钟和数据，并对 P2.6 和 P2.5 的输出进行检测。

两端发送的每帧数据具有相同的 11 位串行格式，第一位是起始位(低电平)，后面依次为 8 位数据，1 位奇偶校验位(奇校验)和 1 位停止位(高电平)，如下所示。

起始位	D0	D1	D2	D3	D4	D5	D6	D7	奇偶校验位	停止位

键盘数据的收发都由键盘时钟同步，也就是说 8042 并不输出时钟信号，数据传输的时钟信号总是由键盘控制器 8048 的串行时钟端 CLK 提供。

图 6.3.3　键盘原理图

键盘在准备发送数据之前，先检测时钟线和数据线的状态。如果时钟线是低电平，表示禁止态，数据先放到数据缓冲器中；如果时钟线为高电平而数据线为低电平，则表示系统要发送命令给键盘，也要把数据存入键盘缓冲器，并开始接收来自系统的命令。在检测到时钟线和数据线都是高电平的情况下，键盘开始按数据串行格式发送数据。每位数据在时钟的下降沿有效，在时钟脉冲高电平期间变化。传送之后，系统将禁止键盘接收信号，直到系统已把输入处理完毕或要求键盘送出相应信号为止。

系统在准备向键盘传发数据时，先检查键盘是否正在发送数据。如果键盘正在发送数据，若还没有发送到第 10 个时钟脉冲(对应奇校验位)，系统可强迫时钟线为低电平超过 60 μs，从而使键盘停止输出。但如果键盘已经传送到 10 个脉冲，则系统将接收完本次数据串的传送。

当键盘没有发送数据或系统强迫键盘停止发送后，系统将数据线拉低准备向键盘传送命令，键盘则发出时钟信号，进入接收数据状态。每位数据在时钟的上升沿有效，在时钟脉冲低电平期间变化。键盘接收完 11 位数据后，将停止位拉低作为应答信号，表示数据接收完毕。输入键盘的每一个系统命令或数据，键盘必须要在 20 ms 之内作出响应，系统才向其作下一个传送。如果键盘响应无效，或者有奇偶校验错，系统会再一次传送命令或数据。

图 6.3.4 给出了以上数据收发过程的时序图。

图 6.3.4 键盘与 PC 机数据收发过程时序图

3. 键盘扫描码

键盘上的每个按键在敲击后所发出的扫描码实际上含两部分内容，一是按键接通扫描码，二是放键断开扫描码。若按住键不放，则不断发出按键扫描码，直到松开键后才发出放键扫描码。下面分别介绍两种键盘的扫描格式。

由于扩展键盘扫描方式与老式 83/84 键标准键盘有较大的区别，所以它们在硬件上互不兼容，因扫描码不同而不能直接互换使用。

标准键盘的键位置号与发送的按键扫描码是等值的，均用 1 个字节表示。如字符“ A ”键，键号为“ 30 ”(十进制)，接通扫描码为“ 1EH ”(十六进制)，两者相同。标准键盘的断开扫描码也是 1 个字节，它是由接通扫描码加上 80H 而得出的。如字符“ A ”键，先按下后又松开，则先输出“ 1EH ”，后输出“ 9EH ”。

扩展键盘的键位置号与发送的按键扫描码不等值。就绝大部分按键来看，扩展键盘的接通扫描码为 1 个字节；断开扫描码为 2 个字节，形式是在接通扫描码前加一个键盘控制器命令字“ F0H ”。如字符“ A ”键，键号为“ 31 ”(十进制)，十六进制为“ 1FH ”；接通扫描码为“ 1CH ”，断开扫描码则为“ F0H ”和“ lCH ”。

标准键盘和扩展键盘的键盘扫描码格式如表 6.3.2 所列。

扩展键盘还有一部分按键的接通和断开扫描码都是多字节的。以 104 键 Windows 键盘为例，其“ 上下左右 ”四个方向键就是多字节的。

表 6.3.2 标准键盘和扩展键盘键盘扫描码格式

		起始位	8 位数据(低位在前，D0～D7)	奇校验位	停止位	起始位	8 位数据(低位在前，D0～D7)	奇校验位	停止位
标准键盘	接通扫描码	1	0XXX XXXX	无	0	无			
	断开扫描码	1	1XXX XXXX	无	1	无			
扩展键盘	接通扫描码	0	XXXX XXXX	有	1	无			
	断开扫描码	0	0 0 0 0 1 1 1 1	1	1	0	XXXX XXXX	有	1

4. 通信命令字

键盘与 PC 机通信命令字较多，下面介绍几条主要命令，如表 6.3.3 所示。

表 6.3.3　PC 机和键盘部分通信命令字

PC 机发送的命令字	16 进制值	键盘发送的命令字	16 进制值
置位/复位方式指示	ED	自检成功码	AA
允许	F4	自检错误码	FC
ECHO	EE	ECHO	EE
复位	FF	确认	FA
重新发送	FE	重新发送	FE

PC 机向键盘发出的命令共有 10 条，都是单字节命令：EDH～FFH。其中 FDH～F7H、F2H～EFH 保留未用。PC 机接收到的其它字节都示为无效。

键盘向 PC 机发出的键盘响应共有 7 条。其中，上电复位后键盘发出自检码“AA”或“FC”，表示自检结果。当 PC 机发送“ECHO”指令时，键盘回答“ECHO”，表示通信正常。当 PC 机发送“重新发送”指令时，键盘重新发送上次发送的数据。当收到 PC 机发送的其它指令时，键盘一般要在 20 ms 之内回答“确认”。

5. 应用

了解了键盘扫描码的编码格式，可以容易地编写出单片机读 AT－PS/2 键盘单字节扫描码的函数。以 12 MHz 晶振工作的 89C2051 单片机为例，键盘信号线 DATA 连至 P1.2，时钟线 CLK 连至 P1.3，参见图 6.3.5。单片机读取单个字节键盘扫描码的 C51 函数如下：

```
uchar Atrdbyt( )
{
    uchar nn=0xff,mm=0x01,uu=0xfe;
    uchar j;
    if(DATA==1)reutrn(255)；
//SDA 为高电平，表示没有按键扫描码，返回 255
    for(j=0;j<9;j++)
       {
        while(CLK==1);
          if(DATA==0)
            nn=(nn&uu);
          else
            nn=(nn|mm);
          nn=_cror_ (nn,1);
          while(CLK==0);
        }
    return(nn);    //返回键盘扫描码
}
```

图 6.3.5　键盘接口电路图

6.3.2 利用键盘接口与 PC 机通信

经前面键盘和 PC 机通信的原理分析，可以设计出利用键盘标准接口与 PC 机通信的应用实例。下面就以一种 IC 卡读写器为例加以介绍。

IC 卡读写器以 24 MHz 晶振的 89C2051 单片机为核心，相当于在键盘和 PC 机之间用了 4 个 I/O 口，连接如图 6.3.6 所示，接口电路可参考图 6.3.5。

图 6.3.6 IC 卡读写器与 PC 机和键盘的连接

89C2051 除了要保证原键盘的正常工作外，还要实现和 PC 机的通信。它执行的通信功能有四种：接收 PC 机的数据和命令；传送数据和命令给键盘；接收键盘数据和命令响应；传送数据和命令响应给 PC 机。

IC 卡读写器对于键盘发送来的数据和命令响应字，一律不加分析处理，以原样转发 PC 机。

IC 卡读写器对于检测到的 PC 机命令进行分析，如果是给键盘的命令，则转发给键盘；如果是 7FH，则表示后续的命令是给 IC 卡读写器的控制命令，调用相关的处理程序。在这里，采用键盘命令中未使用的 7FH 为给 IC 卡读写器的命令前缀。

IC 卡读写器收到键盘的控制命令后，要将 IC 卡内的数据发送给 PC 机。由上节分析可知，PC 机接收到键盘命令响应字时，内部处理不会引起键盘中断响应(09H)；而只有键盘扫描码才会被转换成系统扫描码，引起操作系统的响应。因此，为防止少量 IC 卡数据和键盘的命令响应字冲突，必须将发送给 PC 机的数据预处理，可采用如下方法：先将数据字节分为高低两个字节，然后调整为两个 BCD 码字节，再将这两个 BCD 字节作一次系统扫描码到键盘扫描码的变换，才发送给 PC 机。

PC 机接收到这两个 BCD 码字节后，将其当作键盘扫描码转换成系统扫描码，送给操作系统，存入键盘数据缓冲区(60H)。同 IC 卡读写器通信的 PC 机应用程序读入这两个字节，先把 BCD 码调整为 16 进制数据，再合成为一个字节，得到原数据，完成整个通信过程。

6.4 51 单片机与 PC 机并行传输技术的应用

在一般的主从分布式微机系统中，都存在主机与分机之间的通信问题。一般主机由处理能力强、有丰富的应用软件以及良好的人机操作界面的 PC 机担任。分机常常由体积小、价格便宜的单片机为核心。PC 机与单片机之间的通信，可采用串行通信方式，也可采用并行通信方式。串行通信一般使用在信息量不大的远程通信上；并行通信一般是基于总线级的传输，它传输速度快，一般用在信息量大、实时性强的系统中。本节介绍几个单片机通过双端口 RAM 和 PC 机并行数据传输的例子。

6.4.1　双端口 RAM 简介

IDT 公司的双端口 RAM 时序与 INTEL 公司的单片机系统兼容，特别适用于单片机与单片机、单片机与 PC 机之间大量数据的高速双向传送。

IDT7132 是一种高速 2K×8 bit CMOS 双端口静态 RAM，它提供了两个带有自身控制、地址和 I/O 引脚的独立端口，可独立地读写存储器中的任何单元。其一个端口的使用和普通静态 RAM 基本相同。为防止读写数据冲突，IDT7132 带有硬件端口总线仲裁电路，提供了 BUSY 总线仲裁方式，可以允许双机同步地读或写存储器中的任何单元。即每个端口(左端口，右端口)都可以对内部单元进行读和写。为避免两端口同时对同一单元进行“写”，操作，每一端口内部设置“BUSY”信号，即“BUSY®”和“BUSY(L)”，用于表示该端口对内部单元的操作状态。当某个端口对内部单元操作时，其信号“BUSY”呈现高电平。其管脚图参见图 6.4.1，结构功能参见图 6.4.2，非竞争的读写控制如表 6.4.1 所示。

图 6.4.1　IDT7132 管脚图

图 6.4.2 IDT7132 结构功能图

表 6.4.1 IDT7132 非竞争读写控制字

左或右端口				功 能
R/$\overline{W}$	$\overline{CE}$	$\overline{OE}$	D_0～D_7	
X	H	X	Z	掉电模式
L	L	X	数据输入	数据写入存储器
H	L	L	数据输出	存储器中数据输出
H	L	H	Z	输出呈高阻

说明：

L=低电平，H=高电平，X=任意，Z=高阻。

6.4.2 51 单片机通过双口 RAM 与 PC 机 ISA 插槽进行数据传输

PC 机与多只下位机通过双口进行数据传输示意图见图 6.4.3。PC 机通过左端口读写双口 RAM，不同区域的地址单元存放的是不同下位机的信息，以便下位机各自读写自己区域的数据。下位机分别占用不同的双口 RAM 左端口地址单元，以便同 PC 机进行独立的数据传输。PC 机与双口 RAM 的接口连接图如图 6.4.4 所示。

图 6.4.3 PC 机与多只下位机通过双口 RAM 进行数据传输示意图

图 6.4.4　PC 机与双口 RAM 的接口示意图

根据 PC 机的软件设置，系统选用地址为 A0000H～A7FFFH，相应译码为 $\overline{CS}$=A19*/A18*A17*/A16*/A15。

下位机选用 89C51 单片机。其中一片与双口 RAM 的接口图如图 6.4.5 所示。

图 6.4.5　89C51 单片机与双口 RAM 的接口图

6.4.3　51 单片机通过双口 RAM 与 PC 机并口数据传输

利用双端口 RAM IDT7132 和 PC 机的并行接口(PC 机并行打印机适配器接口 LPT)，可实现单片机与 PC 机之间并行通信，通信速度也可满足要求。

1. PC 机并行打印机接口各信号作用

PC 机都配有一个标准的 CENTRONICS 并行打印机适配器接口，长期以来一直是作为打印机接口而应用的，通过对其分析发现，它也可以作为通用 I/O 并行口来使用。通过对该适配器内的三个 I/O 端口的控制，可实现数据和控制信号的输入、输出。下面对各 I/O 端口及并行口各引线功能进行说明。

PC 并行接口外观是 25 针母插座，如图 6.4.6 所示，各针功能见表 6.4.2。

图 6.4.6　25 针 PC 并口

表 6.4.2　25 针 PC 并口功能定义

信号线	名　称	传输方向	含　义
1	$\overline{\text{STROBE}}$	ー>	选通端，低电平有效
2～9	D0～D7	ー>	数据
10	$\overline{\text{ACK}}$	<ー	确认，低电平有效
11	BUSY	<ー	忙
12	PE	<ー	缺纸
13	SEL	<ー	选择
14	$\overline{\text{AUTOFED}}$	ー>	自动换行，低电平有效
15	$\overline{\text{ERROR}}$	<ー	错误，低电平有效
16	$\overline{\text{INTI}}$	ー>	初始化，低电平有效
17	$\overline{\text{SELIN}}$	ー>	选择输入，低电平有效
18～25	GND	ーー	信号地

PC 机并行接口共三个 I/O 地址：378H、379H、37AH，可以用汇编指令“ in ”与“ out ”来控制。C 语言可用读或写端口指令来操作。

PC 机并行打印机适配器接口具有 8 位数据输入输出线，可通过读写 378H I/O 端口对外引线上的数据进行读写操作。对该端口进行一次写数据操作，意味着一个 8 位数据被写到一个 8 位寄存器锁存并输出到外引线上。而对该端口进行一次读操作，读入的只是由写操作锁存在输出寄存器中的值，所以读操作只能用于打印机接口的自检。对于双机通信来说，8 位数据线只能起到从 PC 机向外发送数据的作用。

PC 机并行打印机适配器接口具有 5 根控制信号输出线，可通过写 37AH I/O 端口把控制信号输出到外引线上，该输出信号也经过寄存器锁存。控制字格式如表 6.4.3 所示。

表 6.4.3　37AH I/O 端口控制字格式

输出数据	D7	D6	D5	D4	D3	D2	D1	D0
外引线	X	X	X	IRQEN	$\overline{\text{SELIN}}$	INIT	$\overline{\text{AUTOFD}}$	$\overline{\text{STROBE}}$

D4 位 IRQEN 为中断允许控制，如果不采用中断方式，应使 D4=0。另外，还要注意，外引线 INIT 输出的信号逻辑与 D2 相同，而其它 3 位相反。

PC 机并行打印机适配器接口具有 5 根状态输入线，可通过读 379H I/O 端口把外引线上的状态信号读入 PC 机，该端口输入缓冲器与状态输入线的对应情况如表 6.4.4 所示。

表 6.4.4　379H I/O 端口状态信号格式

输入数据	D7	D6	D5	D4	D3	D2	D1	D0
外引线	BUSY	$\overline{\text{ACK}}$	PE	SEL	$\overline{\text{ERROR}}$	X	X	X

对于双机通信来说，PC 机可通过该端口读入数据，并且一般只使用 D4 到 D7 这 4 位。另外，由于 D7 位 BUSY 信号反相，可接一非门加以校正。注意：如果采用中断方式，中断请求信号必须从 D6 位 $\overline{\text{ACK}}$ 引入，那么输入数据线只能使用 D3、D4、D5、D7，数据的重组稍微麻烦些。

2. 硬件接口电路设计

该并行通信的原理框图如图 6.4.7 所示，由 PC 机、并行通信接口以及包括双端口 RAM 的单片机控制系统组成。

图 6.4.7　并行通信原理框图

单片机控制系统与 PC 机之间的并行通信是通过对双端口 RAM 的读写实现的。单片机对双端口 RAM 的访问与片外扩展 RAM 没有区别，而 PC 机侧对双端口 RAM 的读写则需设计一接口电路才能实现。从以上的分析可以看出，PC 机并行打印机适配器接口共有 13 根输出线和 5 根输入线可供访问双端口 RAM，而双端口 RAM IDT7132 的地址总线宽度是 11 位、数据总线的宽度是 8 位，因此必须使用锁存器 74LS373 实现地址和数据总线的共享，使用双路开关 74LS157 实现将 IDT7132 的 8 位数据分两次读入 PC 机。硬件接口电路如图 6.4.8 所示。

端口 RAM 的控制字、地址、输出数据都是通过 PC 机对 378H 端口写操作发出的，3 个 74LS373 的锁存命令以及 74LS157 的控制命令则是通过对 37AH 端口写操作实现的。另需注意：当 PC 机从双端口 RAM 中读数据时，应控制 H3 的 $\overline{\text{OE}}$ 为高电平，使 H3 输出呈高阻状态，以避免对读入数据产生影响。而 PC 机向双端口 RAM 中写数据时，H3 的 $\overline{\text{OE}}$ 应为低电平。

为方便实时性的要求，可将双端口 RAM 右侧的片选信号 $\overline{\text{CE}_\text{R}}$ 与/ACK 相连，借用 PC 机打印中断 IRQ7 采用中断方式进行并行通信。单片机控制系统侧是主动方，单片机在每个控制周期最后将各项数据写入双端口 RAM，然后由 $\overline{\text{CE}_\text{R}}$ 发出中断请求。

3. 软件设计

下面给出用 Turbo C2.0 编写的非中断方式用于读写操作的函数。

图 6.4.8 **硬件接口电路**

例 6.4.1

```
Void readdata(unsigned char read-address-L， unsigned char read-address-H，
    unsigned char read-data)
    {
    outportb(0x378，0x70)； /*写双端口 RAM 控制字且使 H3 的 OE 为高电平*/
    outportb(0x37a，0xf)；
    outportb(0x37a，0xb)；
    outportb(0x378，read-address-L)；
    outportb(0x37a，0xa)；
    outportb(0x37a，0xb)； /*低 8 位地址送 H2 锁存*/
```

```
outportb(0x378，read-address-H)；
outportb(0x37a，0xf)；
outportb(0x37a，0xb)；    /*高 3 位地址送 H1 锁存*/
outportb(0x378，0x60)；   /*发读命令*/
outportb(0x37a，0xf)；
outportb(0x37a，0xb)；
read-data=inportb(0x379)&0xf；          /*读低 4 位数据*/
outportb(0x37a，0x9)；
read-data+=(inportb(0x379)&0xf)<<4；    /*读高 4 位，再与低 4 位合并*/
}

Void writedata(unsigned char write-address-L， unsigned char write-address-H，
unsigned char write-data)
{
outportb(0x378，0x20)；   /*写双端口 RAM 控制字且使 H3 的 OE 为低电平*/
outportb(0x37a，0xf)；
outportb(0x37a，0xb)；
outportb(0x378，write-address-L)；
outportb(0x37a，0xa)；
outportb(0x37a，0xb)；    /*低 8 位地址送 H2 锁存*/
outportb(0x378，write-address-H)；
outportb(0x37a，0xf)；
outportb(0x37a，0xb)；    /*高 3 位地址送 H1 锁存*/
outportb(0x378，write-data)；
outportb(0x37a，0x3)；
outportb(0x37a，0xb)；    /*输出数据送 H3 锁存*/
outportb(0x378，0x0)；    /*写命令*/
outportb(0x37a，0xf)；
outportb(0x37a，0xb)；
}
```

附录 A 部分常用网址

A.1 部分国际标准协会的网址

IEEE (Institute of Electrical and Electronics Engineers) :
http://www.ieee.org
ETSI (European Telecommunications Standards Institute):
http://www.etsi.org
ANSI (American National Standards Institute) :
http://www.ansi.org
EIA(Electronic Industries Association)：
http://www.eia.org
DISA(Data Interchange Standards Association):
http://www.disa.org
CSR(Communications Standard Review):
http://www.csrstds.com/sM
论坛: http://wwww.atmforum.org
ACM-TSC(Association for Computing Machinery-Technical Standards Committee):
http://www.acm.org/tsc
AIW(Advanced Peer-to-Peer Networking Implementers Workgroup):
http://www.releigh.ibm.com/app/aiwhome.htm
IETF(Internet Engineering Task Force):
http://www.ietf.org
IISP(Unite States Information Infrastruture Standards Panel):
http://www.ansi.org/iisp
GigaNet: http://www.giganet-ethernet.org

A.2 部分 IC 查询站点

http://www.21ic.com/(**中文**)
http://www.p8s.com/ (**中文**)
http://www.etimag.com.tw/Semi/default.asp (**中文**)
http://www.edtn.com/search
http://www.embeddedlinks.com/chipdir/

http://www.questlink.com/
http://www.semidex.com/

A.3 部分国外电子公司网址

Acctel Technologies
http://www.acceltech.com
Actel Corp.
http://www.actel.com
Aldec Inc.
http://www.aldec.com
Altera Corp.
http://www.altera.com
AMD
http://www.amd.com
AMP
http://www.amp.com
Anadigics,Inc
http://www.anadigics.com/anadigics/
Analog Device
http://www.analog.com
Asahi Kasei Microsystems Co.Ltd
http://www.akm.com
Asymtek
http://www.asymtek.com
AVX/Kyocera
http://www.avxcorp.com
Bourns Asia
http://www.bourns.com
BTU International
http://www.btu.com
Burr Brown Corp.
http://www.burr-brown.com
Bussman Division Cooper Industries
http://www.bussmann.com
Chip Express
http://www.chipexpress.com
Compass Design Automation

http://www.compass-da.com
Consilium,Inc
http://www.consilium.com
Consumer Microcircuits
http://www.cmlmicro.com
CyberOptics Corp.
http://www.cyberoptics.com
Cypress
http://www.cypress.com
Dallas Semiconductor
http://www.dalsemi.com
Design Acceleration
http://www.designaa.com
Elantec Semiconductor Inc
http://www.elantec.com
Epson
http://www.epson.com.jp
Exemplar Logic
http://www.exemplar.com
Force Computers GmbH
http://www.forcecomputers.com
Frontline Design Automation
http://www.frontline.com
Fluke Corp.
http://www.fluke.com
Fujitsu Microelectronics
http://www.fujitsu.com
Genrad Inc
http://www.genrad.com
Harris Semiconductor
http://www.semi.harris.com
Hewlett Packard Asia Pacific Ltd
http://www.hp.com
Hitachi Ltd
http://www.hitachi.com
IAR Systems
http://www.iar.se
IDT
http://www.idt.com

IKOS Systems
http://www.ikos.com
International Rectifier
http://www.irf.com
Interpoint
http://www.intp.com
Iwatsu Elec Co.,Ltd
http://www.iwatsu.com
Ismeca
http://www.ismeca.com
LATTICE
http://www.latticesemi.com
Maxim Integrated Products
http://www.maxim-ic.com
Micrel Semiconductor
http://www.micrel.com
Microchip Technology
http://www.mchip.com/microchip
Micron Technology Inc
http://www.micron.com/mti
MicroSim Corp.
http://www.microsim.com
Mosel Vitelic
http://www.moselvitelic.com
Motorola Semiconductor HK Ltd
http://www.apspg.com
MPM
http://www.mpmcorp.com
National Instruments
http://www.natinst.com
National Semiconductor HK Ltd
http://www.nsc.com
Oki Electronic (HK) Ltd
http://www.oki.com
Orbit Semiconductor
http://www.orbitsemi.com
OrCAD
http://www.orcad.com
Pads Software

http://www.pads.com
Philips Semiconductors
http://www.semiconductors.philips.com
Quality Semiconductor
http://www.qualitysemi.com
QuickLogic
http://www.quicklogic.com
Samtec Asia Pacific
http://www.samtec.com
Schurter S.A.
http://www.schurter.com.ch
SGS-Thomson
http://www.st.com
Sharp-Roxy(HK)Ltd
http://www.sharp.co.jp
Siemens Semiconductor
http://www.siemens.de/Semiconductor
Silicon Systems
http://www.ssil.com
Sony/Tektronix Corp.
http://www.infoweb.or.jp/sonytek
Speed S.A.
http://www.speed.com
Stanford Research Systems
http://www.srs.com
Synergy Semiconductor
http://www.synergysemi.com
Synopsys
http://www.synopsys.com
Synrad
http://www.synrad.com
Teccor
http://www.teccor.com
Tektronix
http://www.tek.com
TEMIC/Siliconix
http://www.temic.com
TEMIC Semiconductors
http://www.temic.de

Teradyne
http://www.teradyne.com
Tescon Co Ltd
http://www.ksp.or.jp/tescon
Texas Instruments HK Ltd
http://www.ti.com
The Mosis Service
http://www.mosis.org/mosis
TransTech Parallel Systems Ltd
http://www.is.org/mosis
TransTech Parallel Systems Ltd
http://www.transtech.com
Tritech Microelectronics
http://www.tritech.sg.com
Toshiba
http://www.toshiba.com
Vicor Corp.
http://www.vicr.com
Xicor Inc
http://www.xicor.com
Xilinx Asia Pacific
http://www.xilinx.com
Zilog Inc
http://www.zilog.com

附录 B　集成电路前缀和生产公司

目前，集成电路的命名国际上还没有一个统一的标准，各制造公司都有自己的一套命名方法，给识别集成电路带来很大的困难，但各制造公司对集成电路的命名还存在一些规律。下面列出一些常见的集成电路生产公司的命名方法供读者参考。(只写了前缀。)

AN	PANASONIC 日本松下电器公司	AN5123
AC	TEXAS INSTRUMENTS(TI)美国德克萨斯仪器公司	AC5944
AD	ANALOG DEVICES 美国模拟器件公司	AD7118
AY	GENERAL INSTRUMENTS(GI)美国通用仪器公司	AY3-8118
BA	ROHM 日本东洋电具制作所	BA328
CA	RCA 美国无线电公司	CA3123
CA	PHILIPS 荷兰飞利浦公司	CA3046
CA	SIGNETICS 美国西格尼蒂克公司	CA3089
CD	RCA 美国无线电公司	CD4081BE
CS	CHERRY SEMICONDUCTOR 美国齐瑞半导体器件公司	CS263
CT	PLESSEY 英国普利西半导体公司	CT1010
CX	SONY 日本索尼公司	CX108
CXA	SONY 日本索尼公司	CXA1034
DN	PANASONIC 日本松下电器公司	DN74LS04
EEA	SIGNETICS 美国西格尼蒂克公司	EEA5550
EF	THOMSON-CSF 法国汤姆逊半导体公司	EF4443
EFB	THOMSON-CSF 法国汤姆逊半导体公司	EFB7510
ESM	THOMSON-CSF 法国汤姆逊半导体公司	ESM523
F	FAIRCHILD 美国仙童公司	F4001
FCM	FAIRCHILD 美国仙童公司	FCM7040
HA	HITACHI 日本日立公司	HA1361
HD	HITACHI 日本日立公司	HD74LS02
HM	HITACHI 日本日立公司	HM472114AP
ICL	INTERSIL 美国英特锡尔公司	ICL8063C
IR	SHARP 日本夏普(声宝)公司	IR9393
ITT	ITT 德国 ITT 半导体公司	ITT3064
KA	SAMSUNG 韩国三星电子公司	KA2101
KC	SONY 日本索尼公司	KC583
LA	SANYO 日本三洋电气公司	LA4102
LB	SANYO 日本三洋电气公司	LB1405
LC	SANYO 日本三洋电气公司	LC4001B
LM	SANYO 日本三洋电气公司	LM8523
LM	NATIONAL SEMICONDUCTOR 美国国家半导体公司	LM1800A
LM	SIGNETICS 美国西格尼蒂克公司	LM387
LM	FAIRCHILD 美国仙童公司	LM1014A
LM	SGS-ATES SEMICONDUCTOR 意大利 SGS 亚特斯电子公司	LM317
M	SGS-ATES SEMICONDUCTOR 意大利 SGS 亚特斯电子公司	M192

MA	ANALOGSYS 美国模拟系统公司	MA106
MB	FUJITSU 日本富士通公司	MB74LS00P
MBM	FUJITSU 日本富士通公司	MBM100470
MC	MOTOROLA 美国摩托罗拉半导体产品公司	MC13007
MC	ANALOGSYS 美国模拟系统公司	MC114
MF	MITSUBISHI 日本三菱电机公司	MF1011
MK	MOSTEK 美国摩斯特卡公司	MK4116
ML	PLESSEY 英国普利西半导体公司	ML231B
ML	MITEL EMICONDUCTOR 加拿大米特尔半导体公司	ML8804
MLM	MOTOROLA 美国摩托罗拉半导体产品公司	MLM301AG
MM	NATIONAL CONDUCTOR 美国国家半导体公司	MM5430
MN	PANASONIC 日本松下电气公司	MN3207
MP	MICRO ROWERYSTEMS 美国微功耗系统公司	MP5401
MPS	MICRO ROWERYSTEMS 美国微功耗系统公司	MPS5003
MSM	OKI 美国 OKI 半导体公司	MSM5525
MSM	OKI 日本冲电气有限公司	MSM4001RS
M5	MITSUBISHI 日本三菱电机公司	M58981
N	SIGNETICS 美国西格尼蒂克公司	N74LS123
NE	SIGNETICS 美国西格尼蒂克公司	NE540
NE	PHILIPS 荷兰飞利浦公司	NE541PHA
NE	MULLARD 英国麦拉迪公司	NE541PHA
NE	SGS-ESMICONDUCTOR 意大利 SGS-亚特斯电子公司	NE555
NJM	NEW JAPAN RADIO 新日本无线电公司	NJM387
OM	PANASONIC 日本松下电器公司	OM200
OM	SIGNETICS 美国西格尼蒂克公司	OM200
RC	PAYTHEON 美国雷声公司	RC4194TK
RH-IX	SHARP 日本夏普(声宝)公司	RXIX0212CE
S	SIEMENS 德国西门子公司	S572
S	AMERICAN MICROSYSTEMS 美国微系统公司	S2743
SAA	SIGNETICS 美国西格尼蒂克公司	SAA1070
SAA	GENERAL NSTRUMENTS(GI)美国通用仪器公司	SAA1025-01
SAA	ITT 德国 ITT-半导体公司	SAA1173
SAB	SIGNETICS 美国西格尼蒂克公司	SAB2024
SAB	AGE-TELEFUNKEN 德国德律风根公司	SAB2010
SAF	SIGNETICS 美国西格尼蒂克公司	SAF1032
SAK	PHILIPS 荷兰飞利浦公司	SAK150BT
SAS	HITACHI 日本日立公司	SAS560
SAS	AGE-TELEFUNKEN 德国德律风根公司	SAS6600
SAS	SIEMENS 德国西门子公司	SAS5800
SDA	SIEMENS 德国西门子公司	SDA5680
SE	SIGNETICS 美国西格尼蒂克公司	SE540
SG	SILICON GENERAL 美国硅系统公司	SG3731
SH	FAIRCHILD 美国仙童公司	SH741
SI	SANKEN 日本三垦电子公司	SI1030
SK	RCA 美国无线电公司	SK9199
SL	PLESSEY 英国普利西半导体公司	SL624C
SN	TEXAS INSTRUMENTS(TI)美国德克萨斯仪器公司	SN76635
SND	SSS 美国固体科学公司	SND5027
SO	SIEMENS 德国西门子公司	SO41E

SP	PLESSEY 英国普利西半导体公司	SP8735P
STK	SANYO 日本三洋电器公司	STKO40A
SW	PLESSEY 英国普利西半导体公司	SW450
T	TOSHIBA 日本东芝公司	T1400
T	GENERAL INSTRUMENTS(GI)美国通用仪器公司	T1102
TA	TOSHIBA 日本东芝公司	TA7628
TAA	SIGNETICS 美国西格尼蒂克公司	TAA370
TAA	SIEMENS 德国西门子公司	TAA991D
TAA	SGS-ATES SEMICONDUCTOR 意大利 SGS-亚特斯电子公司	TAA611
TAA	PHILIPS 荷兰飞利浦公司	TAA630S
TAA	PLESSEY 英国普利西半导体公司	TAA570
TAA	MULLARD 英国麦拉迪公司	TAA570
TBA	FAIRCHILD 美国仙童公司	TBA641
TBA	SIGNETICS 美国西格尼蒂克公司	TBA700
TBA	SGS-ATES SEMICONDUCTOR 意大利 SGS-亚特斯电子公司	TBA800
TBA	HITACHI 日本日立公司	TBA810
TBA	NEC ELECTRON 日本电气公司	TBA810S
TBA	ITT 德国 ITT-半导体公司	TBA940
TBA	AGE-TELEFUNKEN 德国德律风根公司	TBA540
TBA	SIEMENS 德国西门子公司	TBA120SA
TBA	PLESSEY 英国普利西半导体公司	TBA750
TBA	NATIONAL EMICONDUCTOR 美国国家半导体公司	TBA970
TBA	THOMSON-CSF 法国汤姆逊半导体公司 PHILIPS 荷兰飞利浦公司	TBA970NSD
TBA	PHILIPS 荷兰飞利浦公司	TBA570A
TBA	MULLARD 英国麦拉迪公司	TBA530
TC	TOSHIBA 日本东芝公司	TC9121
TCA	ITT 德国 ITT-半导体公司	TCA260S
TCA	SIGNETICS 美国西格尼蒂克公司	TCA770
TCA	SPRAGUE ELECTRICS 美国史普拉格电子公司	TCA3089
TCA	MOTOROLA 美国摩托罗拉半导体产品公司	TCA4500
TCA	PLESSEY 英国普利西半导体公司	TCA800
TCA	SGS-ATES SEMICONDUCTOR 意大利 SGS-亚特斯电子公司	TCA3189
TCA	MULLARD 英国麦拉迪公司	TCA770SQ
TCA	PHILIPS 荷兰飞利浦公司	TCA750Q
TCA	AGE-TELEFUNKEN 德国德律风根公司	TCA270
TCA	SIEMENS 德国西门子公司	TCA440
TCM	TEXAS INSTRUMENTS(TI)美国德克萨斯仪器公司	TCM2912B
TDA	SIGNETICS 美国西格尼蒂克公司	TDA1580
TDA	SPRAGUE ELECTRICS 美国史普拉格电子公司	TDA1051
TDA	MOTOROLA 美国摩托罗拉半导体产品公司	TDA3190P
TDA	NATIONAL SEMICONDUCTOR 美国国家半导体公司	TDA2590
TDA	PLESSEY 英国普利西半导体公司	TDA2560
TDA	SIEMENS 德国西门子公司	TDA1048
TDA	NEC ELECTRON 日本电气公司	TDA1051
TDA	AGE-TELEFUNKEN 德国德律风根公司	TDA1083
TDA	ITT 德国 ITT-半导体公司	TDA1950
TDA	HITACHI 日本日立公司	TDA2002
TDA	SGS-ATES SEMICONDUCTOR 意大利 SGS-亚特斯电子公司	TDA1910
TDA	PRO ELECTRON 欧洲电子联盟	TDA1000

TDA	PHILIPS 荷兰飞利浦公司	TDA4500
TDA	RCA 美国无线电公司	TDA1170A
TDA	MULLARD 英国麦拉迪公司	TDA1051
TDA	THOMSON-CSF 法国汤姆逊半导体公司	TDA2540
TDC	TPW LSI PRODUCTS 美国 TRW 大规模集成电路公司	TDC1010
TEA	THOMSON-CSF 法国汤姆逊半导体公司	TEA5620
TEA	PHILIPS 荷兰飞利浦公司	TEA2026T
TL	TEXAS INSTRUMENTS(TI)美国德克萨斯仪器公司	TL489
TM	TOSHIBA 日本东芝公司	TM4503
TMM	TOSHIBA 日本东芝公司	TMM841P
TMS	TEXAS INSTRUMENTS(TI)美国德克萨斯仪器公司	TMS1976
TP	TEXAS INSTRUMENTS(TI)美国德克萨斯仪器公司	TP4011P
TP	NATIONAL CONDUCTOR 美国国家半导体公司	TP4011B
TPA	SIEMENS 德国西门子公司	TPA1047
TUA	SIEMENS 德国西门子公司	TUA1000
U	AGE-TELEFUNKEN 德国德律风根公司	U416
UAA	SIEMENS 德国西门子公司	UAA190
ULN	SPRAGUE ELECTRICS 美国史普拉格电子公司	ULN2204A
ULN	SIGNETICS 美国西格尼蒂克公司	ULN2209
ULX	SPRAGUE ELECTRICS 美国史普拉格电子公司	ULX3840A
uA	MOTOROLA 美国摩托罗拉半导体产品公司	uA7584
uA	SIGNETICS 美国西格尼蒂克公司	uA758
uA	FAIRCHILD 美国仙童公司	uA1310PC
uAA	THOMSON-CSF 法国汤姆逊半导体公司	uAA4000
uPC	NEC ELECTRON 日本电气公司	uPC1018C
uPD	NEC ELECTRON 日本电气公司	uPD4016D
XR	EXAR INTEGRATED SYSTEMS 美国埃克亚集成系统公司	XR4739

附录 C 波 特 率 表

当选用最常用的串行口工作方式 1、3，定时/计数器工作方式 2 时，8031 内核单片机波特率参数选取表如表 C-1～表 C-9 所示。

表 C-1 6M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
2400	2403.8	243	1	0.16%
1200	1201.9	243	0	0.16%
1200	1201.9	230	1	0.16%
600	600.9	230	0	0.16%
600	600.9	204	1	0.16%
300	300.4	204	0	0.16%
300	300.4	152	1	0.16%

表 C-2 11.0592M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
19200	19200	253	1	0.00%
9600	9600	253	0	0.00%
9600	9600	250	1	0.00%
4800	4800	250	0	0.00%
4800	4800	244	1	0.00%
2400	2400	244	0	0.00%
2400	2400	232	1	0.00%
1200	1200	232	0	0.00%
1200	1200	208	1	0.00%
600	600	208	0	0.00%
600	600	160	1	0.00%
300	300	160	0	0.00%
300	300	64	1	0.00%

表 C-3 12M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
4800	4807.6	243	1	0.16%
2400	2403.8	243	0	0.16%
2400	2403.8	230	1	0.16%
1200	1201.9	230	0	0.16%
1200	1201.9	204	1	0.16%
600	600.9	204	0	0.16%
600	600.9	152	1	0.16%
300	300.4	152	0	0.16%
300	300.4	48	1	0.16%

表 C-4　16.384M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
9600	9481.4	247	1	1.23%
4800	4740.7	247	0	1.23%
4800	4740.7	238	1	1.23%
2400	2370.3	238	0	1.23%
2400	2370.3	220	1	1.23%
1200	1185.1	220	0	1.23%
1200	1201.8	185	1	0.15%
600	600.9	185	0	0.15%
600	600.9	114	1	0.15%
300	300.4	114	0	0.15%

表 C-5　20M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
9600	9469.6	245	1	1.35%
4800	4734.8	245	0	1.35%
4800	4734.8	234	1	1.35%
2400	2367.4	234	0	1.35%
2400	2422.4	213	1	0.93%
1200	1211.2	213	0	0.93%
1200	1197.3	169	1	0.22%
600	598.6	169	0	0.22%
600	598.6	82	1	0.22%
300	299.3	82	0	0.22%

表 C-6　22.184M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
19200	19256.9	253	0	0.29%
19200	19256.9	250	1	0.29%
9600	9628.4	250	0	0.29%
9600	9628.4	244	1	0.29%
4800	4814.2	244	0	0.29%
4800	4814.2	232	1	0.29%
2400	2407.1	232	0	0.29%
2400	2407.1	208	1	0.29%
1200	1203.5	208	0	0.29%
1200	1203.5	160	1	0.29%
600	601.7	160	0	0.29%
600	598.6	63	1	0.22%
300	299.3	63	0	0.22%

表 C-7　24M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
9600	9615.3	243	1	0.16%
4800	4807.6	243	0	0.16%
4800	4807.6	230	1	0.16%
2400	2403.8	230	0	0.16%
2400	2403.8	204	1	0.16%
1200	1201.9	204	0	0.16%
1200	1201.9	152	1	0.16%
600	600.9	152	0	0.16%
600	600.9	48	1	0.16%
300	300.4	48	0	0.16%

表 C-8　33.3M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
19200	19270.8	247	1	0.36%
9600	9635.4	247	0	0.36%
9600	9635.4	238	1	0.36%
4800	4811.7	238	0	0.36%
4800	4811.7	220	1	0.36%
2400	2408.8	220	0	0.36%
2400	2408.8	184	1	0.36%
1200	1204.4	184	0	0.36%
1200	1196.1	111	1	0.32%
600	598.0	111	0	0.32%

表 C-9　40M 晶振波特率选取表

设定波特率	实际波特率	定时器载入值	SMOD 位	波特率误差
19200	18939.3	245	1	1.35%
9600	9469.6	245	0	1.35%
9600	9469.6	234	1	1.35%
4800	4734.8	234	0	1.35%
4800	4844.9	213	1	0.93%
2400	2422.4	213	0	0.93%
2400	2394.6	169	1	0.22%
1200	1197.3	169	0	0.22%
1200	1197.3	82	1	0.22%
600	598.6	82	0	0.22%

参 考 文 献

[1] 汤吉群等. 数据通信技术. 北京：人民邮电出版社，1999

[2] Gilbert Held 著，魏桂英等译. 数据通信技术. 北京：清华大学出版社，1995

[3] 宋建新等. 数据通信. 北京：人民邮电出版社，1996

[4] 李朝青. PC 机及单片机数据通信技术. 北京：北京航空航天大学出版社，2000

[5] 马宏杰等. 微机通信原理与实用技术. 北京：清华大学出版社，1994

[6] 瞿贵荣. 常用数字编译码器的特性及应用. 无线电（增刊），2000

[7] 何立民. 单片机应用技术选编①～⑧. 北京：北京航空航天大学出版社，1993～2000

[8] 万福君等. 单片微机原理系统设计与开发应用. 合肥：中国科学技术大学出版社，1995

[9] 季绍陵等. 由 MODEM 芯片与 89C51 构成的自动报警装置. 数据采集与处理，1999

[10] 胡嗣云. 嵌入式调制解调器与单片机的接口及编程. 国外电子元器件，2001

[11] 吴杰康等. 基于 MODEM 的单片机与 PC 机间的远程通信. 国外电子元器件，2001

[12] 曾刚等. DTMF 远程通信的软硬件实现技术. 电子技术应用，2000

[13] 蔡茂国. 公用电话网远程控制系统的开发与应用. 电子技术应用，2000

[14] 阎石等. PLD 器件在红外遥控解码中的应用. 电子技术应用，2000

[15] 王立锦等. 单片机控制系统的红外线遥控器接口. 电子技术，1998

[16] 温强等. 双音多频发送接收器 MT8888 及其应用. 自动化技术及应用，2000

[17] 朱金刚等. 一种基于 MC145027 的群模拟信号检测系统. 国外电子元器件，2000

[18] 李坚. 专用编解码集成电路及其应用. 电子技术应用，1997

[19] 周明德 . 微型计算机系统原理及应用 . 北京：清华大学出版社，1998

[20] 马忠梅等. 单片机的 C 语言应用程序设计. 北京：北京航空航天大学出版社，1999

[21] 梁翎，李爱齐. C 语言程序设计实用技术与程序实例. 上海：上海科普出版社，1996

[22] 胡汉才. 单片机原理及其接口技术. 北京：清华大学出版社，1999

[23] 孙国盟等译. PC 硬件金典. 北京：电子工业出版社，2001

[24] 邬宽明. 单片机外围器件实用手册（数据传输及接口器件分册）. 北京：北京航空航天大学出版社，1998

[25] 黄宇飞等. 单片机单总线技术. 单片机与嵌入式系统应用，2001

[26] 赵志国，冯建武. 利用 VB 实现 PC 机与多单片机通讯. 单片机与嵌入式系统应用，2001

[27] 朱旻. 在 VB 下 PC 机与 MSC-51 单片机的串行通信. 单片机与嵌入式系统应用，2001

[28] 沈乃财. IBM-PC 与多台 AT89C52 快速并行通讯研究. 自动化与仪表，1999

[29] 薛惠珍. PC 机与单片机之间并行通讯的设计. 自动化与仪表，1999

[30] 曹国义，梁汉滨. DEPHI 在工业控制通讯程序的应用. 微机发展，1999

[31] 刘建国等. 用 AT 键盘接口进行双机通讯的并行测控系统. 微计算机信息，1997

[32] 殷志文等. 利用 IBM-PC 机的键盘接口通信的新型 IC 卡读写器. 电气自动化，1999

[33] 江征战. 串行通信接口标准 RS-423/422/485 及应用. 电子技术应用，1994

[34] 李华贵，杨师宣. 用 PLM 编程实现串行 EEPROM 24LC65 与 89C52 的接口. 微计算机信息，1999

[35] 李成斌，胡生清. I^2C 总线串行通信技术及其应用. 自动化与仪表，1999
[36] 张紫蓬，王淑清. 解决 DS1820 通讯误码问题的方法. 电测与仪表，2001
[37] 吴江，陈尚松. 用单总线技术设计环境状态监控系统. 电子技术应用，2000
[38] 贾振国. DS1820 及其高精度温度测量的实现. 电子技术应用，2000
[39] 金伟正. 单线数字温度传感器的原理与应用. 电子技术应用，2000
[40] 耿建平，陈尚松. DS2450 A/D 转换器的特性与应用，2000
[41] 赵文敏，寿彩丽. 用 VB6.0 开发微机与多台单片机之间的串行通信程序. 华东地质学院学报，1999
[42] 吴江，陈尚松. 用单总线技术设计环境状态监控系统. 电子技术应用，2000
[43] 通用串行总线(USB)技术规范. 电子测试，1999
[44] 宋方等. 通用串行总线(USB)及其芯片的使用. 测控技术，2000