声控计算机制作与应用入门

陈龙三　编著

清华大学出版社

(京)新登字 158 号
北京市版权局著作权合同登记号：01-2001-2956 号

内 容 简 介

本书介绍了用 C 语言开发 PC 机声控系统的技术。

本书以最简单的一块声控接口卡入手，由浅入深地阐述了这块声卡硬件构成机理及制作过程，并且讲解了完整的 C 语言声卡驱动程序，同时讲解了完整 C 语言的录音、放音程序、语音识别程序以及数据存储和同时编译多个 C 语言文件。另外通过声控电话拨号、声控门禁设计和声控家电三个完整的实例讲解了 C 语言是如何控制 PC 机外围设备。

本书由浅入深，内容新颖、实例完整，适合于广大使用 C 语言进行微机系统开发的初学者。也适合用于大专院校计算机、自动控制、电子工程及相关专业的师生教学参考。

书　　名：声控计算机制作与应用入门
作　　者：陈龙三　编著
责任编辑：曾　刚
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.tsinghua.edu.cn
印 刷 者：北京鑫丰华彩印有限公司
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：11　**字数**：261 千字
版　　次：2001 年 10 月第 1 版　2001 年 10 月第 1 次印刷
书　　号：ISBN 7-302-04841-X/TP・2869
印　　数：0001～5000
定　　价：22.00 元(附软盘)

《工控与电子精品系列图书》策划委员会

《工控与电子精品系列图书》序

“以信息化带动工业化”是我国今后几年乃至更长时间第二产业发展的主题，也是我国科学技术发展的必由之路。世纪之初既有机遇又有挑战，作为一个工程技术人员怎样面对挑战而抓住机遇，使自己乘上工业化的快车！

每一位工程技术人员需要不断地去学习、去实践，丰富自己，才能跟上科技的步伐从而更适应激烈竞争的环境。本系列丛书完全从这个角度出发，使读者反复于学习与实践之间，不仅可以领会理论的精髓更可以掌握开发的技巧。

本系列丛书有以下特点：

实例丰富而详尽

针对目前市场图书现状，本系列丛书大多数以应用实例为主，其中有几本为应用实例集。文中所涉及硬件均有完整的电路图和源程序，更可贵的是大多数源程序都配有详尽的注释。涉及到操作步骤，更是详细而有序，手把手教习读者去开发真正的产品。

涉及范围广而精

本系列丛书针对目前乃至今后市场需求，由最底层的微电子技术到 EDA 工程，由信号处理技术到 DSP，由测控技术到单片机，由宽带网技术到智能建筑，讲解机理透彻，应用实例实用经典。本系列丛书还侧重于新技术的推广，为我国迎头赶上先进技术提供一些启发。

读者定位准确

本系列丛书中的每一本都是针对不同的工程技术人员，涉及到电子和工控行业大多数的技术人员，让每一位读者都可以找到适合自己的技术书籍。

本系列丛书的选题策划、稿件编辑，得到了广大高校教师和业内工程技术人员的大力支持与合作，才使得我们这个系列丛书能够以较高水准面向广大读者，在此表示衷心的感谢！

希望每一位工程技术人员走向各自事业的成功！

《工控与电子精品系列图书》策划委员会

2001 年 8 月

前　言

随着计算机在各个领域应用的逐渐深入，社会对计算机技术人才尤其是程序员的需求越来越多，要求也越来越高，为了适应社会的需要，程序员也希望自己掌握更多的技术尤其是计算机外围接口方面的知识，另外想从事计算机硬件和工业控制开发的初学者希望自己能够较快地掌握计算机接口方面的开发能力。

C 语言是目前唯一一种可以运行在从单片机、PC 机直到巨型机等各种机型的高级语言。目前 PC 机 C 语言的编译系统已经很完善了，用它开发一些底层控制、驱动等项目更是多快好省。C 语言功能强大，十分适用于控制系统的开发，它的优点主要有；

1. 作为一种高级语言，易于开发出功能复杂的控制驱动程序。
2. 良好的开发环境，调试方便。
3. 可移植性强，如在一种机型开发的程序可以轻松地移植到另一种机型上。
4. 作为一种编译语言，C 语言的效率很高，在一些规模较大的工控程序开发中，使用 C 语言开发程序的效率甚至高于汇编语言开发的程序。

但是，国内关于使用 C 语言开发 PC 机控制方面的技术资料极少，广大初学者无从下手，面对一大堆基础理论的技术资料，自己摸索且效率不高。相信本书的推出能为想学习 PC 机控制的初学者带来一点帮助，也为想深入学习 C 语言的程序员带来一些启发。

本书所涉及的一块声控卡纯粹为一块 PC 机的 I/O 卡，实际上就是早期的声卡，此卡的构成元件非常简单，是任何一本接口方面图书中必不可少的常用器件如 8255、A/D 和 D/A 转换芯片，而且无需对芯片编程，制作非常简单。书中均给出了完整电路图和完整的驱动程序，而且为了方便读者学习，本书所配磁盘中包括本书所有范例的 C 源程序文件、目标文件以及编译后的可执行文件。

为了使读者更深地了解 C 语言强大的控制功能，本书精选三个实例，如声控电话拨号、声控门禁设计和声控家电控制，均在书中予以详细讲述，相信这些应用都能给读者以启发，将 C 语言真正的应用于实践。

策划委员会

2001 年 8 月

序

研究语音识别及从事相关的实践工作已进入第七个年头了，曾经实验过PC、DSP及单片机8051的不同硬件工作平台，深深为其趣味性所吸引，在其间有一些大专院校的老师，对专题制作感兴趣的学生，业界工程师，参与讨论并提供宝贵的意见，于是将其整理成本书，使对语音识别感兴趣的朋友有一本入门的书可供参考，特别是书中所介绍的制作可以满足一般读者动手实践的乐趣。

声控计算机是利用语音识别的技术，以声音控制计算机完成特定的工作，使用者只要对着计算机自然的说话，便可以控制计算机动作，在以手不方便操作计算机键盘或是控制面板时，便是它派上用场的时机，只要事先了解一下它的使用限制及其操作方式，要识别那些单字、词、命令或是词组，便可以为我们带来操作上的方便及乐趣。

本书是一本语音识别实践入门的书，先介绍语音识别的基本控制原理再以PC机为主的硬件工作平台来做语音识别的相关接口的制作。在PC机上我们以基本的A/D、D/A组件及I/O接口，而设计了一片专供语音实验的语音卡，可以做声音录放音的相关实验外，进而可以将声音分析，取出特征参数做识别比对及处理，使学生在学过基本的接口理论后，可以加以利用，自行组装一套语音分析及识别系统，非常适合学生专题制作用。

PC机语音卡声控系统的硬件电路完全公开，至于软件方面，为了方便初学者自行设计声控应用程序，我们特别整理了PC语音卡声控链接库(DOS版)供读者使用，初学者只要使用TURBO C V2.0版，便可以在短时间内自行设计出属于自己的声控程序，减少尝试错误的次数而省下许多宝贵的时间。

声控的应用范围很多，包括计算机接口应用、自动化控制、消费性产品应用、门禁管理、仪器设备等。总之，您想到能以声音控制的微电脑装置都有可能实现。读者如果有学习及使用上的任何疑问，请与出版社接洽或与我们联络，谢谢。

网址　　：vic.seeder.net
E-MAIL ：ufvicwen@ms2.hinet.net

陈龙三
2000.5.4

目　录

第1章　声控计算机简介

声控计算机在十几年前可能只是出现在科幻影片中，人们可以跟计算机交谈。现在，您将会发现这些已经不是梦想了，市面上已经有产品开始在卖了！本章将介绍声控计算机的基本处理原理及应用等相关的知识，进而带领初学者很快地进入语音识别的应用领域中，一起来设计自己的声控应用系统。

1.1　何谓声控计算机

声控计算机可以声音来控制计算机，完成某些特定的动作，如此一来可以取代部分按键输入来执行指令，也就是说计算机可以听懂人们所讲的话，并且加以处理后可以完成特定的工作，更进一步可以让人与计算机进行交谈。

声控计算机的技术使用的是计算机语音识别技术，而语音识别的处理技术使用许多语音数字信号的分析方法。典型的语音信号处理技术可以分为以下几种：语音编码、语音合成、语音识别、语者辨识、语音了解，这些语音信号处理技术是未来发展计算机人机交谈的重要技术，其中语音识别是后两者的基本关键技术。

1.1.1　语音编码

语音编码是其他语音信号处理技术的基础，一般语音合成、识别、了解及语者辨识经常会用到语音编码的技术。原始语音数据经过编码压缩后，有助于进一步的数据分析及处理。应用在通讯传输系统上，将语音数据经过压缩后，原来的数据量会大量的降低，使得远程通讯传输速度得以加快，大幅降低通讯费用。此外语音数据经过编码后，可以保护私人谈话的隐密性，避免重要的谈话内容被窃听。

1.1.2　语音合成

语音合成的技术应用在语音的产生或还原上，通常是输入文字而产生流畅的语音输出，例如现在市面上的电子字典，可以输入英文单字而说出英文。个人计算机上可以输入中文字后，而计算机会说出来。因此在进行中英文输入时，会以发声音响应您的输入，增加文字输入的效率，大大的降低了因注视屏幕过久而产生的疲劳。因而一篇文章也不必用眼睛看，计算机会自动地念出来。

1.1.3　语音识别

语音识别技术是用来设计一台会听话的计算机，只要对着麦克风说话，便可以指挥计

算机动作，也就是要实现“芝麻开门”声控计算机的梦想，当然现在这已不再是梦想了。这部分是本书将要介绍的重点，并且将以软件及硬件来实现。语音识别系统应用的范围相当广泛，现在随着许多关键技术的突破及 VLSI 技术的进步，市面上已出现许多使用方便的声控应用产品，中文语音输入系统、声控手机语音拨号、声控汽车音响等，您只需动口，不必动手便可以享受科技带来的方便，相信未来还会有更多有趣的声控电子产品上市。

1.1.4 语者辨识

语者辨识的技术用在门禁管理上，用来做特定通行者身份的确认，其最后的结果是接受或拒绝。如果不是特定通行者输入语音时，系统会自动判别出来而发出警告的信息。门禁管理是保卫安全措施中重要的一项，传统的控制是使用钥匙，或是使用红外线，无线电遥控器，以及采用刷卡的方式来开启大门，多少会碰到钥匙或遥控器遗失的问题，如果语者辨识的效果稳定的话，这将是一种非常方便的门禁管理方式，相信未来的一些科技大楼会大量的采用。

1.1.5 语音了解

语音了解是计算机人机交谈系统中最复杂的技术，使用者可以与计算机直接交谈。此技术是以语音识别为基础，再结合语法分析、语意处理及词汇库等人工智能方面的技术，并配合语音合成，使得计算机能够明白说话者的意思，并以适当的语音来回答进行交谈。以下举个简单的计算机人机交谈的例子，应用在电话中产品价格的询问：

计算机提示语：您好，这是伟克工作室，您需要什么服务。
使用者回答　：我想订购语音卡，请问要多少钱？
计算机提示语：语音卡成品 2300 元。
使用者回答　：我还想订购红外线控制板，请问要多少钱？
计算机提示语：红外线控制板成品 2000 元。
使用者回答　：我考虑一下。
计算机提示语：谢谢您的来电。

1.2 声控计算机分类

声控计算机按照系统所能识别单字多少可以分类为以下三种：

- ✧ 特定词汇：几个单字、词或是词组。
- ✧ 少量词汇：数十个单字、词或是词组。
- ✧ 大量词汇：涵盖所有的单字、词或是词组发音。以中文语音识别来说就是所有中文字。

声控计算机的分类，按照使用者是否需要事先做训练分为三种：

- ✧ 特定语者：识别系统只能识别某一特定使用者的声音，使用者在第一次使用此系

统时需将所有要识别的字汇念过一到两次，当作语音参考样本。

- ✧ 语者调适：使用者只要曾经对识别系统训练过，此系统便可以识别出他的声音，是一种比较有弹性的做法，使用者不需要念完所有的音，只需要念过一部分的单音后，系统会自动将语音参考样本做调整。
- ✧ 不特定语者：任何使用者不需要事先对识别系统训练，都可以使用声控系统，此时系统中已经包含不同种性别、年龄的口音，这种声控系统是一种最完美实用的系统。

声控计算机按照语者说话的方式分类可以分为二种：

- ✧ 单音识别：系统只能识别单音，因此使用者所说出的每一个字必须分开来。
- ✧ 连续音识别：系统可以接受语者连续发音。

由以上的说明，读者可以了解到，一套最理想的声控计算机系统应该是拥有大量词汇、不特定语者连续音语音的识别系统，一般人不需要经过学习，便可以让计算机听懂他发出的语音，也就是说只要对着计算机说话便可以直接来控制计算机动作了，但是要完成这样的一套高识别率的系统实在不是一件容易的工作。

一般在应用上，特定语者，少量词汇的单音识别系统便可以满足我们的特定需求，若能先完成这样一套简单而高识别率的声控系统，在不影响识别率的情况下而后再逐渐加大词汇量，或是修改语者训练的方式，采用语者调适的方法，也可以提高声控系统的整体效能，增加使用的方便性。以下列举一般设计声控系统的基本要求以供参考，此系统是一个较易完成的系统，并且要有不错的辨识效果。

声控计算机基本要求有：

- ✧ 识别率高。
- ✧ 特定语者。
- ✧ 少量词汇。
- ✧ 单音识别。

1.3 声控计算机的应用

使用者可以对着计算机很自然地说话，做到以声音控制计算机动作，在用手不方便操作计算机键盘或是控制面板时，便是它派上用场的时候，只要事先先了解一下它的使用限制及其操作方式，要识别那些单字、词、命令或是词组，便可以为我们带来一些操作上的方便及乐趣。

声控的应用范围很多，一般可以分为以下几种：

- ✧ 计算机接口应用：利用声音控制屏幕显示，如演示文稿系统，多媒体展示，或利用声控来下达计算机指令与键盘同时操作，如应用在CAI，GAME中。
- ✧ 自动化控制：利用声音来控制机器人在高危险度的场所工作，或各种机械操作，或是声控仪表操作。
- ✧ 消费性产品应用：如家电控制，电视、音响、电灯或语音自动拨号、汽车声控设备、儿童玩具声控。

✧ 文字处理器：利用语音来输入文字，如听写机或声控文字处理器。

✧ 利用语者辨识技术设计门禁管理系统。

以下我们举些例子来做说明。

1.3.1 中文听写机

由于计算机的普及，现在已成了我们日常生活中不可缺少的工作伙伴了。而在操作计算机的时候，一般是使用键盘及鼠标，更先进的方式是使用数字板以手写输入方式来输入文字，许多人一定想过是否可以用语音的方式，其中以语音来做文字处理器是一种相当实用的中文输入方式，只需要对着计算机逐一将一篇文章“念”给计算机听，计算机便会将其转换为电子文字文件，是不是很神奇，想想我们每天要与人交谈要讲多少话，如果能以说话的方式来做文书输入，那么工作起来既轻松又有趣。

1.3.2 语音自动拨号

语音自动拨号则可以给经常联络的朋友用语音来拨通电话，只需要事先告诉计算机要识别那些朋友的名字及其对应的电话号码，便可以随便开口拨个电话出去，例如说出“小华”，便直接拨电话给小华了，此声控自动拨号特别适合于车上的移动电话使用，可以增加开车的安全性，使用上非常方便。

1.3.3 声控仪表操作

声控仪表操作，可以用声音直接控制仪表动作，例如量测电压时要人工转换到电压档，而量测电流时要又要转换到电流档，此时改为以声控的方式来操作，随口说出“电压”，或是“电流”，一切就完全自动转换设置了，未来的一些智能型高级仪器设备可能会增加这些方便的控制装置。

1.3.4 “0”至“9”，“YES”或“NO”的单音识别

不特定语者辨识系统在使用前并不需要做特别的训练，一般人可以直接说话来进行识别，但是通常的识别词汇不会很多，例如数字“0”至“9”，“YES”或“NO”的单音识别。如果这些单音的识别正确率高的话，便可以应用在很多需要输入数字的场合中，如语音查询系统中。

1.3.5 声控家电开关

如果将声控系统安装在家中控制灯光照明用，并事先输入电灯的相关声控命令，晚上回到家中，打开家门，一片漆黑，只需要说声“电灯”，电灯马上打开，不必摸黑找开关，一声命令马上为您带来光明。如果将声控系统安装在家中空调控制用，并事先输入空调遥

控的相关声控命令，大热天回到家中，打开家门，只需要说声“空调”，则空调马上打开，不用去开空调或是找遥控器按下开关来打开空调，一声命令，马上为您带来凉快。

1.4 声控计算机的处理步骤

声控计算机是由人的声音来控制计算机动作，可是计算机本身是完全不懂人的声音，因此必须要让计算机先了解熟悉人讲话的声音及腔调，此一步骤我们称为语音特征参数参考样本建模，将原先训练好的声音特点存成语音参考样本，以便将来做识别时，当做对比来参考。而实际在做识别时，使用者可以说出原先训练过的语音来操作，譬如说“目录”，计算机则做出 DOS 下“DIR”的动作，不过，原先必须先训练计算机一听到“目录”的声音则知道要做“DIR”的指令。当然您也可以直接说“DIR”。声控的处理流程如图1-1所示：

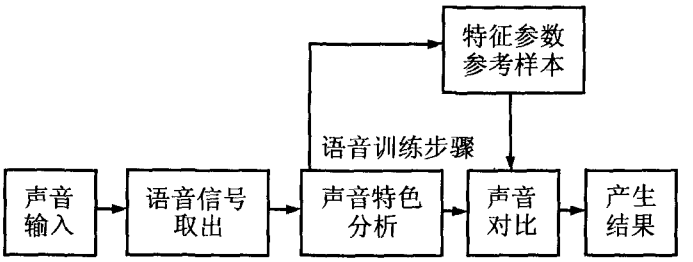


图 1-1 语音辨认步骤

如上图所示，语音识别可以分为五个步骤：

- ✧ 首先是声音输入，说话者用麦克风将声音经由语音卡输入计算机。
- ✧ 计算机把静音切除，而从中将代表声音的部分取出来，我们称为语音切割。
- ✧ 计算机进行声音的特征分析。
- ✧ 将输入的语音特征与原来训练好的声音特征进行对比，找出最相似的结果。
- ✧ 将结果转换成相对的动作而执行，完成声控的目的。

整个过程分为 2 个阶段：

- ✧ 语音训练：将输入的语音经过分析存为特征参数参考样本，即告诉计算机将来要辨识那些声音。
- ✧ 语音识别：将输入的语音经过分析与原先计算机内的参考样本做对比，找出最相近的声音当做辨认结果。

1.5 本书所制作的声控计算机

本书是一本介绍声控计算机实践的参考书，因此重点比较偏向实际制作方面，并且不谈及太深奥的理论。其实语音识别是一门相当有趣但是又需要一些理论的设计技术，有太多的实验及方法可以来做研究，为了让更多计算机 DIY 的使用者都能顺利地设计属于自己的声控计算机，我们特地花了些心思将过去曾实验过的一些效果还不错的语音识别方法，

整理成 TURBO C 声控链接库，让读者可以自行写程序来做控制。只要您懂得基本的语言程序，便可以参考本书所设计的声控接口，自行设计出特别的声控计算机应用系统。

本书所制作的声控计算机是以 PC 机为主的硬件工作平台，来做语音识别的相关接口的制作。在 PC 机上我们以基本的 A/D、D/A 组件及 I/O 接口为主，而设计了一片专供语音实验的语音卡，可以做声音录放音的相关实验外，进而可以将声音进行分析，取出特征参数做识别对比及处理，使学生在学过基本的接口理论后，可以加以利用，自行组装一套语音分析及识别系统。

PC 语音卡声控系统的硬件电路完全公开。至于软件方面，为了方便初学者自行设计声控应用程序，我们特别提供 PC 语音卡声控链接库供读者使用，初学者只要有 C 语言程序设计的基础便可以在短时间内自行设计出属于自己的声控程序，减少尝试错误的次数而省下许多宝贵的时间。

而 PC 机的等级也不必很高，只要您的 PC 机是 80486-DX33 等级以上的计算机(CPU 内含浮点运算)，都可以顺利地用来设计及制作此套声控计算机，此套系统特别适合学生声控计算机相关的专题制作及个人研究。

TURBO C 声控链接库特点：

- ✧ 利用自制 PC 语音卡接口做语音辨识。
- ✧ 提供特定语者的单音、字、词辨识功能。
- ✧ 识别率可达 90%以上。
- ✧ 不限定说话语言，方言也可以。
- ✧ 具有自动语音输入和自动侦测的功能。
- ✧ 链接库提供声音录音、放音、语音切割、分析、识别等功能。
- ✧ 含 TURBO C 声控链接库使用范例程序。

习　题

1. 什么是“声控计算机”？
2. 试描述语音信号处理技术的分类，并加以说明。
3. 什么是语音编码，试举例说明。
4. 什么是语音合成，试举例说明。
5. 什么是语者辨识，试举例说明。
6. 什么是语音了解，试举例说明。
7. 特定语者与不特定语者语音识别系统有何不同。
8. 试举例说明具有语者调适功能的语音识别系统。
9. 描述声控计算机基本要求。
10. 描述声控计算机使用的 4 种实例。
11. 什么是“文听写机”。
12. 画出声控的处理流程并加以说明。

第 2 章　实验环境设置

在真正进入声控计算机设计制作主题之前，本章先对书中所使用的实验环境和使用工具作一个说明，读者可以针对需要而加以添购，包括以下几大项：

- ✧ 实验硬件工具。
- ✧ 软件使用工具。
- ✧ 硬件适配卡。

2.1　实验硬件工具

做硬件实验时手上若有一些必要的硬件工具，则实验做起来会较为顺手，所使用的基本硬件工具如下：

- ✧ 示波器。
- ✧ 逻辑笔。
- ✧ 数字式万用表。
- ✧ PC插槽接口保护器。
- ✧ 直流电源。
- ✧ PC个人计算机。
- ✧ 面包板。
- ✧ 基本的焊接工具。

这是一般大专电子电机课实验室的基本设备，如果读者对微机硬件实验有兴趣的话，若经费许可的情况下，在家添购上述的一些简单的设备，可以增加实验的方便性。此外如果是做单片机 8051 控制程序的设计，需要 8051 在线仿真器(ICE)，来快速对控制程序下载并执行测试，读者可以依照自己的预算来选购哪一些工具来用。

2.2　软件使用工具

本书所使用的软件工具主要是PC上的TURBO C V2.0 C语言编译器DOS版，用来设计语音卡的驱动程序及声控程序，同时在声控程序方面支持有语音卡的声控链接库，使读者不必很深入地了解语音识别的整个技术，便可以直接来设计声控应用系统。读者所需要具备的相关知识，就是TURBO C的使用，如果还不清楚TURBO C整合的工作环境使用，可以直接参考附录中TURBO C简易使用说明。

C语言是在微机自动控制业界中，大家公认最佳的程控计算机语言，所以我们选择C语言设计语音识别的控制程序。基本上，使用C语言编译器做程序设计有以下的好处：

- ✧ 程序好写且具有结构化的特点。

- ✧ 程序具有自我批注功能，修改容易。
- ✧ 除错方便。
- ✧ 移植性高。

当然，我们也可以汇编语言来做设计，基本上使用汇编语言及C语言做设计都行，但是考虑到浮点运算，若以汇编语言来设计的话，那将是一个相当浩大的工程！现在在PC机上有如此好用的TURBO C编译器，为何不使用C语言来做程序设计？一旦熟悉了C语言后，您设计程序的效率会提高很多，甚至可以在PC机上以C语言测试正确的软件加以修改，改写成8051 C编译器可以接受的程序代码，而放到8051控制板上来做控制的工作，以此方法来做跨硬件平台的独立操作型控制系统设计，可以省下很多程序设计的时间。

2.3 硬件适配卡

配合本书的实验，建议使用的一块硬件接口卡：P_CARD PC语音识别卡。

SP_CARD PC语音识别卡为本书主要的识别工作平台，以基本的A/D、D/A组件及I/O接口，而设计了一片专供语音实验的语音卡，可以做声音录放音的相关实验外，进而可以将声音分析，取出特征参数做识别对比及处理，使学生在学过基本的接口理论后，可以加以利用，自行组装一套语音分析及识别系统，非常适合学生专题制作用。

以上所介绍的硬件适配卡的特性及使用，请参考本书各章及附录所做的说明，使用这块接口实验板，可以在PC机上设计出属于自己独一无二的声控计算机系统。

第3章　语音识别处理步骤

在第一章中曾经介绍过声控计算机的处理步骤，在本章中我们将进一步探讨其每一处理步骤的原理，读者有了这些基本的概念后，并配合实验来观察语音波形，可以达到最佳的学习效果。本章将分为以下几个小节来介绍语音识别处理步骤：

- ✧ 语音信号输入：使用应用程序 SI.EXE 及 VO.EXE 做实验。
- ✧ 语音信号切割：使用应用程序 SEG.EXE 做实验。
- ✧ 特征参数求取：使用应用程序 ANA.EXE 做实验。
- ✧ 语音识别对比：使用应用程序 MATCH.EXE 做实验。

在基本的概念说明后，我们再使用一些基本的应用程序来做实验，读者如果手上有一片语音卡的话，便可以马上进行验证，可以加深对语音信号处理及识别的印象，相关的应用原始 C 语言程序，在第 8 章中会做介绍，读者有兴趣的话，可以先自行翻阅一下，这些程序都是以语音卡声控链接库来完成的，这对初学者而言，只要以最少的程序，便可以设计许多控制的实验，非常方便实验及应用。

3.1　语音信号输入

在做语音信号处理前，必须先将语音输入计算机，在 PC 机上我们是以实验的观点来做相关的语音信号观察，于是我们想到利用一般的接口控制芯片来制作语音输入输出适配卡。不管在软件或是硬件学习上，都是相当好的研究题材，非常适合学生专题制作研究用。

在实验的设备上，我们只要一片语音卡及一支动圈式麦克风和一只喇叭，便可以从事语音信号相关的研究，一般对语音信号感兴趣的朋友可以负担得起的，有了以上的配备我们可以开始一步一步来研究。什么是计算机语音识别？首先让我们看一下语音的信号。

3.1.1　观察语音信号数值

执行应用程序 SI.EXE，计算机开始采集语音信号，并将相关的数值数据显示出来，此时可以对着麦克风说话，所显示出来的数值数据将会变化。当使用者按下任何键则结束程序执行，类似的执行结果如下页：

为了方便处理，语音信号我们以带正负号的字节来表示，其值的范围是 + 127 至 - 128，我们所看到的数值序列“0 1 1 0 0 1 0 1 0”是静音或是杂音的信号，振幅较小，而真正的语音信号值振幅较大，所以此程序可以用来测试所使用的麦克风是否适用于语音卡，同时也可以了解语音信号的数值表示，下一步我们将语音的信号波形画出来。

Voice data list:

1 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1
1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 1 1 0 1 0 1
0 0 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 4 -7 23 14 -16
51 -15 -15 31 -64 49 9 -36 19 -32 43 5 -28 20 -52 46 13 -2
-14 21 -19 18 -51 48 11 -15 -43 33 3 16 -40 24 -9 14 -30 2
12 -37 24 13 -19 -12 34 5 8 8 7 8 -24 8 -6 -12 7 -12 -32 1
-19 20 -5 -1 21 -7 -1 14 2 -2 -11 3 9 9 3 1 1 1 0 1 1 1 0
1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 1 1 0 0 0 1 1 1 1 0 1 1 0 0
1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 -1
9 -16 -97 61 -23 22 -49 0 35 -11 0 -17 -16 49 11 -38 31 -7
-30 26 -59 55 19 -29 16 -49 47 14 -30 7 -8 18 -26 -29 40 9
37 -13 21 8 12 -10 -41 34 -15 13 37 -4 10 12 10 4 -23
17 -17 -17 20 -3 -1 -12 -9 0 2 0 -1 2 1 0 2 1 0 1 0 -1
0 0 1 -1 1 1 1 1 0 0 1 1 1 1 1 0 0.....

.....

End of VOICE data list...

3.1.2 观察语音信号波形

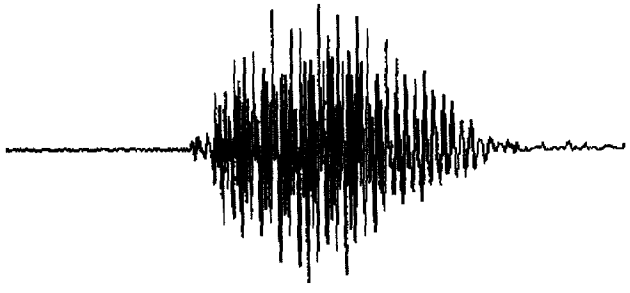
执行应用程序 VO.EXE，工作画面出现如下所示：

SP CARD SPEECH RECOG. LIB. DEMO --> mic i/p sp o/p

Any key to start MIC i/p ...

Please say something ...

Any key to stop MIC i/p ...



<ESC> to exit!
Any key to continue...

图 3-1 语音“1”的波形

当按下任何键开始输入语音，若再按下任何键则结束语音输入，此时计算机会将所录到的语音说出来，并将波形信号画出来，如上页图 3-1 所示，是笔者说出语音“1”的波形。再次按下按键可以重新再执行一次，若按下“ESC”键，则结束程序的执行。由图形中，我们可以看出语音波形的轮廓，此时，计算机变成一台储存式示波器，可以将语音信号采集到计算机中并且显示其波形，在第 7 章中我们还可以利用分析程序进一步将波形放大来看，分析各种波形的特性，以便利用其特性来做信号的处理。

3.2 语音信号切割

通常我们讲话时，会按照当时的情况而中间多少都会有一些停顿，以便能清楚地将意思表达出来，在我们做语音识别时便是利用中间停顿时的静音，将重要有意义的字词找出来，以便做进一步的分析及处理。

上一节中我们已经看到了语音信号的波形了，但是要做进一步的识别必须把所讲的字从语音信号中找出来，即找出语音的起始点及语音的结束点，此时就要应用到一点技巧称为语音切割，详细的语音切割技术已含括在声控链接库中，可以避免读者尝试错误的次数，让我们看一下说明的例子。

执行应用程序 SEG.EXE，工作画面出现如下：

```
SP CARD SPEECH RECOG. LIB. DEMO --> mic i/p   sp o/p
Any key to start MIC i/p ...
Please say something ...
Any key to stop MIC i/p ...
seg 0   --> 4560    7040
seg 1   --> 9360    11760
seg 2   --> 14000   16800
```

当按下任何键开始输入语音，若再按下任何键则结束语音输入，此时计算机会将所录到的语音播放出来，并在波形信号中利用语音切割的技术，将各个单音的端点找出来。例子中笔者说出“123”3 个单音，计算机便将此 3 个单音的端点定出来，如上所示，接着将波形画出来，并且也将相对的端点位置标出来，如图 3-2 所示。

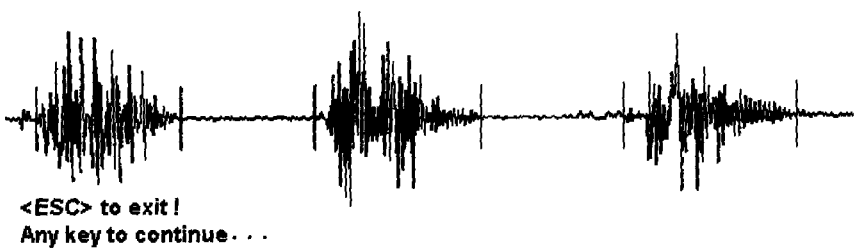


图 3-2　显示“123”各单音的端点位置

再次按下按键，可以重新再执行一次，若按下“ESC”键，则结束程序执行。由图形中，我们可以看出语音断点的判断相当合理，利用此方法可以正确地找出语音的真正端点位置，以利于后续的识别处理。如果一开始端点就找错，那么就不可能识别对了，所以语音信号切割处理，在声控应用中是一个相当重要的环节。

3.3 特征参数求取

我们可以很容易地分辨出一只白色的狗及一只黑色的狗，因为从颜色来看，便很清楚，所以颜色就是分辨狗的重要特征因素。两台不同厂牌的车子，一部是轿车，另一部是跑车，我们也可以从外型来做分辨，因此外型便是分辨车子的重要特征因素。相同的道理，语音识别也是要找出具有代表性的声音特征参数来做对比。

前面我们看过一些语音的应用实验程序，其中用到一些语音数据文件，我们随便说一个单音，令计算机采集语音信号，便要占用 2KB 至 3KB 的字节内存空间，试想我们怎样能够在这些数据中，找出具有代表性的声音特点做将来的分析识别用？这便是语音识别技术的困难所在，什么样的语音特征参数可以将某些声音的特征找出来，以后一旦出现类似的特征参数，便可以将其对比出来，实际上有许多论文及研究都在探讨这项技术，在此我们并不详加讨论其高深的理论基础，而是直接利用声控链接库中的功能来处理，详细语音特征参数求取的技术已包括在声控链接库中，以省下读者许多宝贵的时间。

我们的语音信号是由语音卡直接输入，并由计算机以 8KB 的采样频率来做数字化采样，即每秒拾取 8000 点语音数据，分辨率为 8 位，经过语音信号切割后，把要处理的单音位置找出来后，接着进行语音特征参数求取，将每一个单音的信号平均分为 16 个音阶框(FRAME)来做处理，而每一个音阶框再取出 10 个语音特征数值(浮点数值)，称为语音特征参数。

以下我们举例说明语音特征参数求取的过程，执行应用程序 ANA.EXE，工作画面如下：

```
Load SPEECH file COMP.SP...
I'm saying ...
Speech Feature list :
Frame  0:  0.39  0.97   0.98   1.39  0.97  - 0.87  - 0.24  - 0.61  - 0.14  - 0.22
Frame  1:  0.27  1.20   2.06   1.24  0.05  - 1.16  - 1.08  - 0.41  - 0.22  - 0.17
Frame  2:  0.32  1.04   2.32   1.96  0.37  - 1.44  - 1.26  - 0.22    0.01  - 0.19
Frame  3:  1.17  1.21   1.96   2.22  0.88    1.03    0.40    0.19  - 0.14  - 0.06
Frame  4:  0.64  2.37   1.57   1.92  1.77    1.89    0.04  - 0.41    0.08  - 0.02
Frame  5:  0.49  1.47   1.29   2.23  1.89    1.33    0.40  - 0.15  - 0.36    0.09
Frame  6:  1.04  1.98  - 0.98   0.22  0.89    0.65  - 1.13  - 0.40  - 0.22    0.11
Frame  7:  1.03  2.49  - 1.64  - 1.05  0.51    0.82  - 0.53    0.15  - 0.28    0.05
```

```
Frame  8 :  1.14  0.90  - 0.63  - 0.62  - 0.54   0.13   0.23  - 0.02  - 0.14  - 0.02
Frame  9 :  1.43  2.54  - 0.80  - 2.44  - 0.51  - 0.37   0.35   0.07  - 0.20   0.04
Frame  10 : 1.70  0.57   0.16  - 2.55  - 0.86  - 0.19  - 0.01  - 0.08  - 0.09   0.00
Frame  11 : 1.99  1.74  - 1.05  - 0.61  - 0.48  - 1.10   0.07   0.12   0.19   0.02
Frame  12 : 1.17  2.77   1.26  - 0.98  - 0.26  - 0.31  - 0.18   0.53   0.16   0.01
Frame  13 : 1.22  2.37   1.43  - 0.07   0.62  - 0.09  - 0.22  - 0.16   0.18  - 0.02
Frame  14 : 1.18  1.75   0.90   0.42   0.80   0.37   0.44   0.46   0.34  - 0.11
Frame  15 : 0.60  1.53   1.64   0.73   0.01   0.57   0.38  - 0.00   0.51   0.01
Any key to continue .....
```

系统自动加载语音输入文件，文件名为 COMP.SP，接着说出语音内容“计算机”，并对语音波形做语音切割的端点侦测，再将语音分为 16 个音阶框，进行特征参数求取，最后将每一个音阶框所取出的 10 个语音特征数值列出来，这些浮点数值并不会很大，而且有正有负。此求取特征参数在语音识别中有以下两大用途：

语音训练时需要对已知语音求取特征参数做为参考样本

将输入的已知固定的语音经过分析存为特征参数参考样本，即告诉计算机将来要识别哪些单音。并将这一些参考样本存盘，以便在语音识别时与未知语音特征参数比对，找出相对的误差或是失真。

语音识别时需要对未知语音求取特征参数

将输入的未知语音经过分析找出其特征参数，与原来计算机内的语音参考样本做对比，找出最相近的语音当作识别结果。此外，有一点要说明的是语音特征参数求取的过程中，需要做一连串的算术浮点运算，因此如果读者使用的计算机还是老式的计算机(CPU 为 80486 DX-33 未含浮点运算功能较早版本的处理器)，在执行以上的例子时，您可要等待，此时计算机并未完成运算，请耐心地等待结果，或是换一台较快的计算机来执行。

3.4 语音识别对比

在上一节中我们已经说明过语音特征参数的重要性及求取过程，本节将介绍如何利用特征参数来进行语音识别对比，最简单的方法是将未知测试语音的特征参数，逐一与已知固定的语音特征参数做对比，并计算出一失真值，如果已知语音特征参考样本有 10 组，那么经过对比后会产生 10 个失真值，取其最小失真值当作其失真度，而产生此失真度的该组语音特征参考样本，便是所对比出来的结果。

当然还有更复杂的语音识别对比方法，其实目前很多论文还在研究中，就在这方面还可以做许多的实验，在此我们并不打算进一步详加讨论这些高深的理论，而直接利用声控链接库中的内建功能来做实验，读者将可轻易的使用声控链接库中的内置功能做进行语音

识别对比的实验，可以省下许多宝贵的时间，以下我们以例子来做说明。

执行应用程序 MATCH.EXE，工作画面出现如下：

load DATA BASE ...

I'm saying ...

Recog ...

Frame 0 :	- 0.63	1.12	2.45	2.00	1.14	0.33	0.63	- 0.71	- 0.15	0.04
Frame 1 :	- 1.08	1.69	4.04	0.87	1.46	- 0.13	0.63	- 0.02	- 0.42	- 0.03
Frame 2 :	- 1.40	1.87	3.49	1.76	1.48	0.00	0.19	- 0.12	- 0.22	- 0.05
Frame 3 :	- 1.46	1.74	3.34	1.04	1.53	0.00	1.10	- 0.65	- 0.16	- 0.04
Frame 4 :	- 1.44	1.44	3.72	- 0.04	2.56	- 0.38	0.84	- 0.09	- 0.40	- 0.05
Frame 5 :	- 1.74	1.61	4.03	0.86	2.10	- 0.73	0.92	- 0.09	- 0.34	- 0.05
Frame 6 :	- 1.92	1.78	3.65	1.06	1.79	- 0.05	0.84	- 0.26	- 0.35	- 0.00
Frame 7 :	- 1.51	2.38	3.87	0.99	1.18	- 0.30	0.87	- 0.12	- 0.42	0.00
Frame 8 :	- 1.61	3.01	4.80	0.98	0.62	0.02	0.97	- 0.38	- 0.40	0.04
Frame 9 :	- 0.98	1.67	3.83	0.90	1.16	0.18	0.92	- 0.25	- 0.43	0.02
Frame 10 :	- 1.10	2.29	3.31	1.94	0.89	0.69	0.01	- 0.13	- 0.23	- 0.01
Frame 11 :	- 0.04	1.70	2.72	1.96	0.55	1.31	0.17	- 0.66	- 0.07	- 0.00
Frame 12 :	0.03	2.15	3.23	2.00	1.45	0.34	0.21	- 0.39	- 0.04	- 0.02
Frame 13 :	- 0.24	1.28	3.42	1.77	1.57	0.50	1.23	- 0.02	- 0.09	0.13
Frame 14 :	- 0.18	1.25	2.07	2.02	2.49	1.33	0.75	0.85	- 0.00	0.01
Frame 15 :	- 0.23	1.43	1.78	1.64	1.78	1.46	0.47	0.50	- 0.19	0.12

REF distortion list:

87 58 150 113 111 115 118 87 178 118 91 68 143 133 105 108 133 83
171 127 87 67 146 132 109 113 136 82 176 126

Recog. answer=1 distotion : 58

系统首先自动加载语音参考样本文件 DB.DAT，内部含有中文数字 0 至 9 各 3 组的语音特征参数，再来系统加载未知语音测试文件，文件名为 1.SP，其内容为中文数字 1。接着计算机说出语音内容“1”，并对未知的语音波形做语音切割的端点侦测，再做语音特征参数分析，最后将每一个音框所取出的 10 个语音特征数值列出来做参考，而后列出未知语音与各个已知语音参考样本的误差比对值，总共有 30 个失真值，取其中最小值当作最小失真值，该组语音便是识别的结果，识别的结果为“1”其失真值最小为 58。

习　题

1. 列举语音识别处理三大关键技术。
2. 请说明特征参数求取的主要目的。
3. 请说明语音识别对比的基本原理。

第 4 章　PC 语音卡设计

从本章开始，我们便进入声控计算机的制作主题，在本章中先介绍简单的声音录音放音基本原理，而后介绍如何以一些简单的硬件电路在 PC 机上设计一片语音卡，来完成基本的声音录音放音的相关实验，其中我们使用了模数转换芯片 AD0804 及数模转换芯片 DA08，这是学习数字接口控制及专题制作中经常会用到的芯片，只要结合一些相关的模拟电路便可以完成声音录放音的动作。

4.1　声音录音放音基本原理

声音信号是一种连续性的模拟信号，而要由计算机来做录放音控制，得先将声音信号数字化而进入到计算机的内存内，这数字化的过程我们以图 4-1 来做说明。原来的连续性的语音波形，经过数字采样的处理后可以得到数字形式的声音数据，其中所要的硬设备为 A/D（模数转换）接口，在软件处理上是每隔一小段时间到语音输入端口拾取一点声音数字信号数据至语音内存内，这一小段时间称为采样周期。

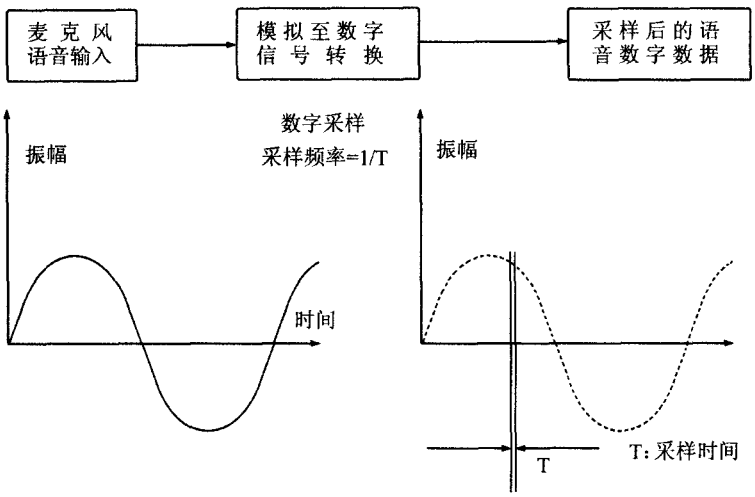


图 4-1　声音录音数字采样示意图

采样周期的倒数称为采样频率，即每秒钟对原始信号做多少次数字采样，由采样定理来看，采样频率不能小于原始信号最高频率的两倍，否则会造成采样失真，即所得到的数字信号数据不能忠实地将原来的模拟声音信号还原。

采样频率设置对声音的还原（声音品质）有重要的影响，当采样率越高时，所占用内存会增加，相对的放音品质更高，一般音乐的频率响应范围为 20Hz 至 20kHz，因此像 CD

音响其数字采样频率则高达 44kHz，保证可以将声音信号忠实的还原出来。

一旦声音经数字采样存入计算机内存中，可加以分析、取参数做压缩储存或是做语音识别，如要将声音还原，只要依原先的采样频率值经 D/A（数字至模拟转换）接口处理，便可以将声音重现从而播放出来，其过程如图 4-2。

图 4-2　数字放音控制

4.2　语音卡特性

语音卡是一片 PC 机上的 I/O 适配卡，此片适配卡上含有 A/D 接口（使用 AD0804）及 D/A 接口（使用 DA08）和模拟放大电路，可以做声音录放音的相关实验外，还可以将声音进一步做分析，取出特征参数做识别对比及应用，使学生在学过基本的数字微机接口理论后，可以加以利用，自行组装一套语音分析及识别系统，非常适合学生专题制作用。

4.2.1　语音卡硬件特点

语音卡的硬件特点有：

- ✧ 以 PC ISA 62 PIN I/O 适配卡来设计语音录音及放音实验接口。
- ✧ A/D 接口使用 AD0804 做模数转换。
- ✧ D/A 接口使用 DA08 做数模转换。
- ✧ 麦克风输入阻抗为 600Ω，喇叭输出阻抗为 4Ω。
- ✧ 提供模拟低通滤波器，截止频率为 4kHz。
- ✧ 卡上有 8 位 I/O 可供一般硬件扩充应用。
- ✧ 卡上 I/O 地址可以调整，不会与现有适配卡相冲突。
- ✧ 可以扩充做 8255 I/O 实验，提供 24 位 I/O 控制。
- ✧ 配合 PC 机 TURBO C V2.0 语音辨识声控链接库及语音辨识范例程序。
- ✧ 含实验用麦克风及实验用喇叭。

4.2.2　PC TURBO C 2.0 语音辨识声控链接库特点

TURBO C 2.0 语音辨识声控链接库特点有：

- ✧ 以语音输入卡做声音输入。
- ✧ 以 TURBO C 设计计算机语音控制系统。
- ✧ 提供特定语者的单音、字、词识别功能。
- ✧ 不限定说话语言，方言也可以。

- ✧ 具有自动语音输入侦测的功能。
- ✧ 链接库提供声音录音、放音、语音切割、分析、识别等功能。
- ✧ 几行程序即可完成语音识别的工作。
- ✧ 备有多个有趣的声控应用范例程序供参考，易学易用。
- ✧ 看了马上可以设计属于自己的计算机声控控制系统。
- ✧ 特别适合在自行设计的专题中加入声控的功能。
- ✧ 识别率可达 90%以上。

4.3 AD0804 功能说明

模拟到数字转换器，简称 ADC(Analog-Digital Converter)是将连续模拟信号转换为数字信号的器件，一般外界的物理量像电流、位移、温度、压力、重量、声音等均可以经过传感器接口处理而转换为模拟的电压信号，属于模拟信号，经过 ADC 接口将信号转换成为数字信号后，才能由计算机做数据的储存或是运算处理。ADC 接口一般用在数字接口或微机的接口输入控制上，典型的应用有以下几种：

- ✧ 对电压、电流的自动测量。
- ✧ 数字式万用表。
- ✧ 温度、照度测量。
- ✧ 电子秤设计。
- ✧ 声音数字化录音。
- ✧ 影像数字化录像。

其中最后两项在今天 PC 机上多媒体的应用尤其重要，像声霸卡即内含有声音录音的 ADC 接口，而视频捕捉卡上则有用于视频数字化转换的 ADC 处理芯片，由于视频信号频宽相当高，因此用在视频处理的 ADC 芯片其转换速度只有几十纳秒(ns)，相对的价格非常昂贵。在本节中主要是介绍最普遍的 ADC 芯片 AD0804 的功能。

AD0804 是一块使用相当普及的模数转换 IC，可以在一般的电子元器件商店买到，其主要性能如下：

- ✧ 使用逐次逼近法做 A/D 转换。
- ✧ 量化比是 8 位。
- ✧ 最大误差为 1LSB。
- ✧ 存取时间 135ns。
- ✧ 转换时间 100μs。
- ✧ 具有差动模拟电压输入。
- ✧ 0 到 5V 模拟标准电压输入。
- ✧ 输出锁定。

图 4-3 为其引脚图。

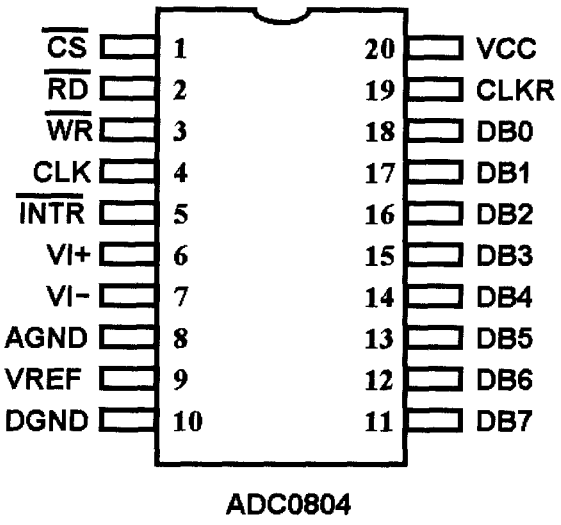


图 4-3　AD0804 引脚图

引脚说明如下：

- ✧ VCC：电源 5V 输入端。
- ✧ DGND：电源接地端，为数字电路的接地。
- ✧ AGND：为模拟电路的电源地端。
- ✧ VREF：参考电压输入引脚。
- ✧ VI+、VI－：差动模拟电压输入引脚。
- ✧ DB0～DB7：8 位的数据输出线，经过模数转换完成后的数字量，由此数据线送出来，当 CS 引脚为高电位时则呈现高阻抗的状态。
- ✧ CS：芯片选择使能信号，低电位有效。
- ✧ INTR：A/D 芯片转换完成的输出信号，低电位有效，用以通知外界（一般是 CPU）来读取转换完成的数字化后的数据。
- ✧ RD：芯片的读取控制信号，CS=0 且 RD=0 时，则读取数据线上的数字化后的数据。CS=0 且 RD 信号在上升沿时将 INTR 信号设定为 1。
- ✧ WR：使芯片开始转换的触发信号，低电位有效。
- ✧ CLK：A/D 芯片的工作时钟输入引脚，其最快的时钟输入不可太高（一般低于 1MHz），也可以配合 CLKR 引脚以 RC 电路组成振荡工作时钟，其振荡频率为 F=1/(1.1×RC)一般 R 取 10kΩ。
- ✧ CLKR：时钟输出引脚，可配合上述 CLK 引脚，以 RC 电路，提供芯片工作时的工作时钟。

4.4　DA08 功能说明

数字到模拟转换器简称 DAC(Digital-Analog Converter)是将数字信号转换成连续的模拟信号的器件，一般用在数字接口或微处理机的接口输出控制上，典型的应用有以下几种：

- ✧ 数字激光唱盘 CD 的放音转换。
- ✧ 计算机 VGA 显示适配卡中的图像输出转换电路。
- ✧ 直流电机速度控制。
- ✧ 数字式电源供给器。
- ✧ 任意波形发生器。
- ✧ 计算机音乐合成器控制。
- ✧ FM 声音源的输出转换控制。
- ✧ 计算机数字放音控制。

其中的计算机数字放音在许多的较新型的电子产品中我们都可以看到，几乎任何需要语音提示的产品，都可派上用场，熟悉这种控制接口技巧将可以在自行设计的产品中加入语音的功能，增加产品的附加价值。在本节中主要是介绍DAC转换芯片DAC08或DAC0800的功能。

在做数模转换接口时，常用到的 D/A 转换芯片是 DAC0800。其主要性能如下：

- ✧ 保持时间为 100ns。
- ✧ 最大误差为 1LSB。
- ✧ 输出电压可调整在 - 10V 至+18V 之间。
- ✧ 具有互补的输出电流，Iout 及 Iout\。
- ✧ 工作电压可从 4.5V 到 18V。

图 4-4 是其引脚图，图 4-5 为基本动作电路图。

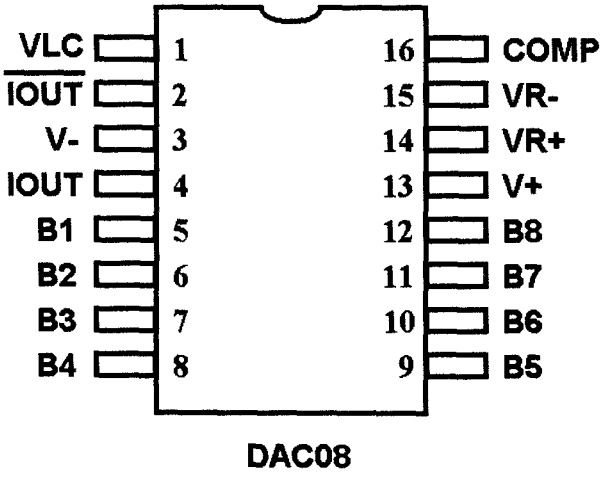


图 4-4　DAC08 引脚图

DAC0800 的引脚说明如下：

- ✧ V+：正电源输入引脚，电压范围为 4.5V 至 18V 之间，一般取+5V 输入。
- ✧ V - ：负电源输入电压范围为 - 18V 至 - 4.5V 之间，一般取 - 12V 输入。
- ✧ VR+、VR - ：参考电压的输入引脚，用来决定满刻度时电流的大小，其中参考电流 Iref、参考电压 Vref 及满刻度电流 Ifs 的关系如下：

0.2mA≦Iref≦4mA

Iref＝Vref / Rref

Ifs＝Iref×Ñ255/256

- ✧ VLC：逻辑工作临界值控制引脚，当 DAC0800 输入信号为 TTL 标准电位时，此引脚接地即可。
- ✧ COMP：引脚接上电容至 V - 负电源端，可形成频率补偿电路，以防止高频振荡。
- ✧ B1～B8：数字量数据输入，其中 B1 为 MSB（最高位），B8 为 LSB（最低位）。
- ✧ Iout、Iout\：模拟互补式电流输出，Iout 与 Iout\电流总和为定值。

Iout+Iout\=Ifs,其中 Ifs 为满刻度电流。

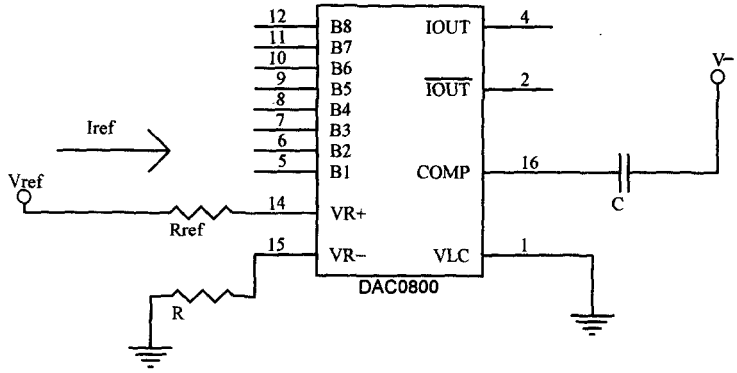


图 4-5　DAC08 基本工作线路图

4.5　8255 功能说明

8255 是一块可编程的 I/O 接口控制芯片，在很多电子产品微电脑硬件外围及工业自动控制板上均可以发现它，主要是用来支持 Intel 微机系统的外围控制芯片，由于其 I/O 工作具有一般标准的工作时间，可以适用在其他各种型号的微处理器上如 6502、Z80、8088，甚至单片机 8048、8051 来做额外的硬件扩充，因此我们称它为通用型多功能的可编程 I/O 接口控制芯片。

在语音卡的数字接口设计上，我们也使用了一块 8255I/O 控制芯片，本节先将此芯片的功能做一说明。

4.5.1　8255 简介

图 4-6 是其内部方块图，8255 有 3 个可编程的输入输出口（每个口提供 8 个 I/O 口），共可提供 24 支 I/O 线的控制引脚。而此 3 个 I/O 口的输入或输出分别由 A 组及 B 组的内部控制器加以设定。其中 A 组由口 A 的 PA0 至 PA7 及口的上半部 PC4 至 PC7 组成，B 组则由口 B 的 PB0 至 PB7 及口 C 的下半部 PC0 至 PC3 所组成。芯片控制的方式是通过双向的总线与微机系统连接，通过控制字组的写入来设置 8255 的工作方式。

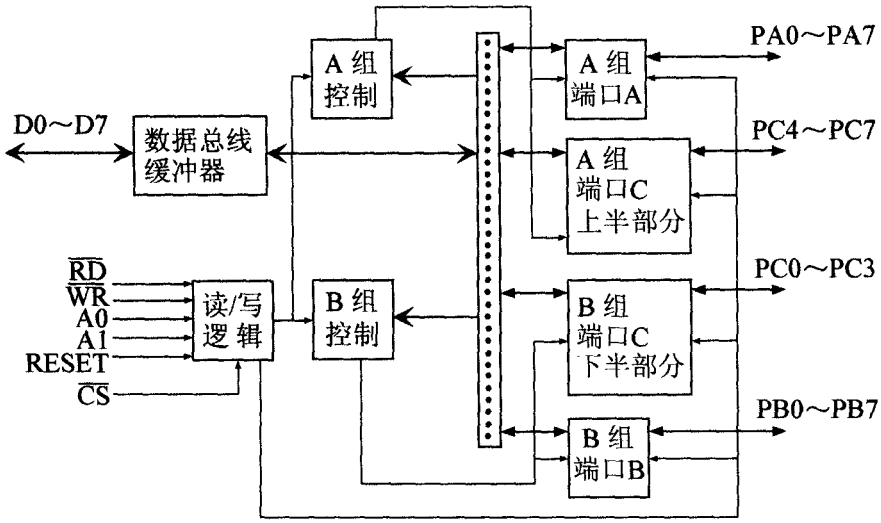


图 4-6　8255 内部方块图

4.5.2　8255 引脚说明

图 4-7 为 8255 的引脚图，其电气特性分别说明如下：

- A0，A1（I/O 口选择线）：输入此二信号（一般接系统地址总线的 A0 及 A1）用来选择 8255 内部 4 个控制端口寄存器，分别为端口 A、端口 B、端口 C 及命令控制寄存器，如：A1A0 为 00 则选择端口 A 寄存器，A1A0 为 01 则选择端口 B 寄存器，A1A0 为 10 则选择端口 C 寄存器，当 A1A0 为 11 时选择控制寄存器。
- D0～D7（数据总线）：双向输入输出
  8255 的 8 条数据线连至系统的数据总线，微处理器在执行数据输出或输入指令时，都是通过此总线。当芯片选择信号(CS)为高电位时此数据总线呈现高阻抗。
- RESET（复位信号）：输入
  高电位有效，用来清除内部控制寄存器，同时将 3 个 I/O 端口全部设为输入。
- CS（芯片选择信号）：输入
  低电位有效，CS=0 时控制线将内部数据总线与系统总线连接在一起，可以直接控制 8255 工作。当 CS=1 时则 D0～D7 与系统总线隔离，成为高阻抗状态。
- WR（写入信号）：输入
  低电位有效，配合 CS 信号将处理器数据写入 8255 内。
- RD（读取信号）：输入
  低电位有效，配合 CS 信号读取 8255 内部寄存器的值。
- PA0～PA7（端口 A）：输入或输出
- PB0～PB7（端口 B）：输入或输出
- PA0～PA7（端口 C）：输入或输出
  8255 的 3 个可编程设置的输入输出端口，做为与外围设备连接用。其中端口 A，

端口B可以全部设为输入或输出，端口C则可分别设置成为两组(PC0～PC3;PC4～PC7)4位的输入或输出端口。

✧ VCC（电源）

+5V 电源输入。

✧ GND（接地）

芯片接地。

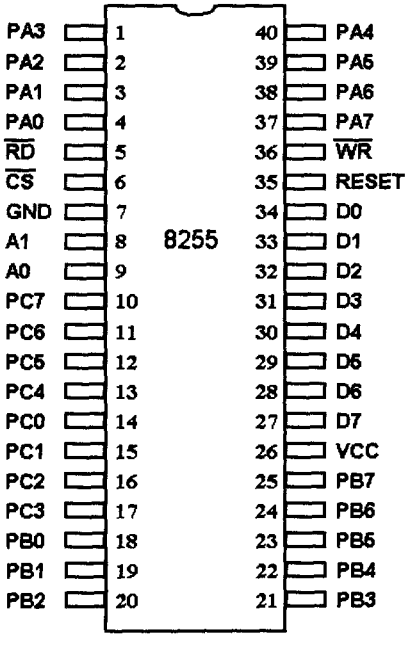


图 4-7　8255 引脚图

4.5.3　8255 工作说明

8255 的 A0 及 A1 引脚可以选取 4 个内部寄存器端口 A、端口 B、端口 C 和控制寄存器，其中 3 个 I/O 寄存器用来储存 3 个输入输出端口的数据，表 4-1 列出其工作情况。

表 4-1　8255 寄存器选择表

A1	A0	RD	WR	CS	动　作
0	0	0	0	0	读取端口 A 的数据
0	1	0	1	0	读取端口 B 的数据
1	0	0	1	0	读取端口 C 的数据
1	1	0	1	0	错误情况
0	0	1	0	0	数据写入端口 A
0	1	1	0	0	数据写入端口 B
1	0	1	0	0	数据写入端口 C
1	1	1	0	0	写入控制字
×	×	×	×	1	总线高阻抗
×	×	1	1	0	总线高阻抗

例如当（A1,A0）=(0,0)，选取端口 A 的寄存器，若 RD=0 是作端口 A 的数据读取（由端口 A 输入资料）；WR=0 便作端口 A 数据的写出（端口 A 输出数据）。当(A1,A0)=(1,1)，WR=0 是做控制命令字的写入，但无法进行读取。8255 共提供 3 种不同的操作模式来配合各种不同的 I/O 接口做控制。

- ✧ 模式 0：基本 I/O 控制；
- ✧ 模式 1：触发式 I/O 控制；
- ✧ 模式 2：触发式双向 I/O 控制。

至于如何设定，可以由控制字组来写入适当的动作命令字。

4.5.4 模式设定

图 4-9 为 8255 控制字格式

其中控制字的每一位含义如下：

- ✧ D7 ：D7=1，用来做控制字组设定用。
- ✧ D6,D5：选择端口 A 及端口 C 下半部的操作模式。
- ✧ D4 ：端口 A 的输入输出设定，“1”表输入，“0”表输出。“0”与“O”(out, 输出)相近，可帮助记忆。
- ✧ D3 ：PC4～PC7 输入输出设定。
- ✧ D2 ：端口 B 及端口 C 上半部的操作模式设定。
- ✧ D1 ：端口 B 的输入输出设定。
- ✧ D0 ：PC0～PC3 输入输出设定。

此外 8255 对端口 C 提供位设定/清除的功能，只要使用一个输出控制指令便可达到位控制的目的，在做状态设定时非常方便。在控制字组的 D7 位为 0 时是做端口 C 各别位控制用，至于设定哪一个位由 D3,D2,D13 个位来选择，设定情形如图 4-8 所示。

例子：设定位 2 为低电位，则送出控制字 0X04。

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	0	0

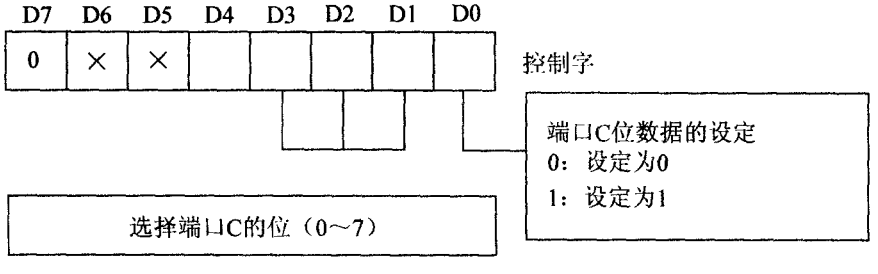


图 4-8 8255 位设定/清除格式

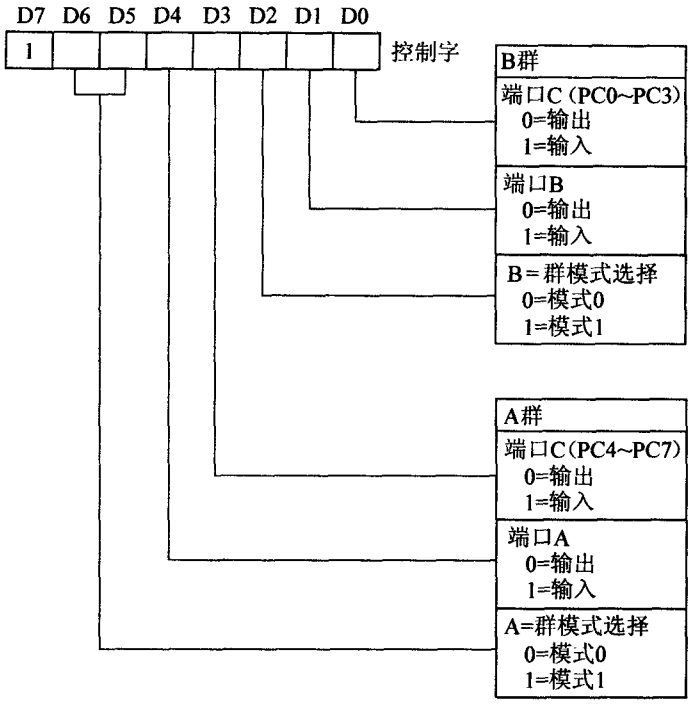


图 4-9　8255 控制字格式

4.5.5　8255 工作模式 0

8255 工作模式 0 为基本输入输出控制，通过控制字来设置 8255 的工作模式。

例子 1：控制字为 0X80

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0

则将 8255 设为模式 0 操作，3 个 I/O 端口全部设为输出。

语音卡 8255 的设计，使用 8255 模式 0 动作，当作基本的 I/O 控制。有关模式 1 及模式 2 的动作较为复杂，有兴趣的读者可以参考一般 PC 机 I/O 接口控制的书中关于 8255 接口设计的单元。

4.6　语音卡电路设计

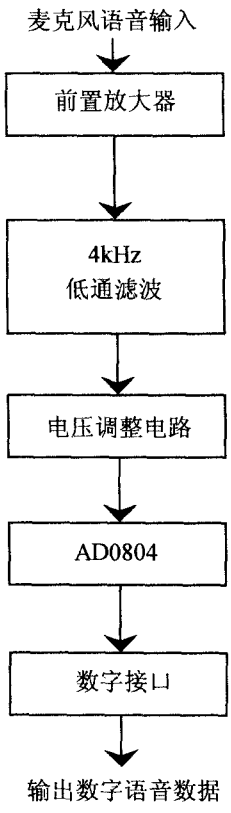
介绍过声音的录放音基本原理后，本节以实际电路设计来完成语音录音放音的功能，其使用到的硬件有 A/D 及 D/A 芯片，在前两节中已分别介绍过 AD0804 及 DA0800 芯片，只要结合这两块芯片，加上必要的模拟处理接口电路，即可完成接口的设计，为了实验方

便，我们加上 8255 芯片做数字接口处理而设计成一片适配卡，专门做声音录放音控制及做语音识别实验用，我们称为语音识别卡 SP_CARD。

语音卡整个电路可以分为以下 3 大部分：

- ✧ A/D 电路。
- ✧ 数字接口。
- ✧ D/A 电路。

图 4-10 为 A/D 转换的工作方块图，图 4-11 则为 D/A 转换的工作方块图表示。



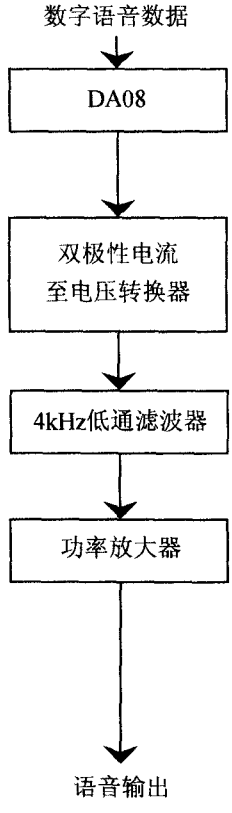


图 4-10　语音卡 A/D 转换工作原理图　　图 4-11　语音卡 D/A 转换工作原理图

4.6.1　A/D 转换电路

A/D 电路由模拟语音信号处理电路及 A/D 转换电路组成。图 4-12 为整个模拟语音信号处理的电路图，即包括图 4-9 所示的前端三部分：前置放大器、4kHz 低通滤波器及电位调整电路。首先语音信号由麦克风输入，一般麦克风的输入电压只有几十毫伏左右，因此通过增益约为 100 倍的前置放大器 U7B，放大至 1 伏特左右的电压，以便推动后面相关电路，而实际的放大率可以由可变电阻 VR1 来调整，以配合使用的需要。

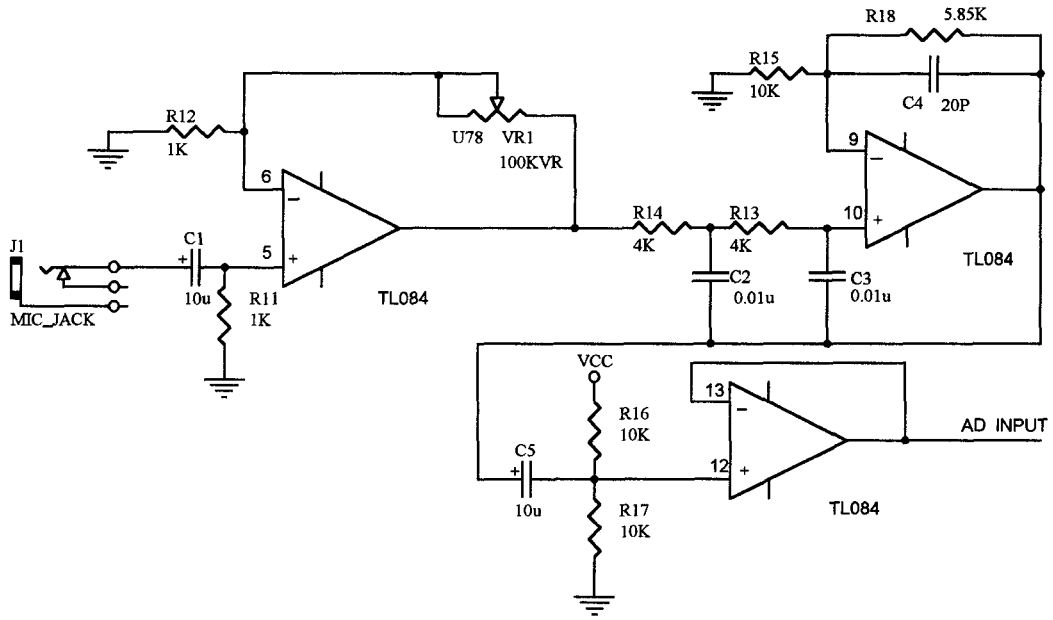


图 4-12　模拟语音信号放大电路

此外为了满足采样定理——至少要以信号最高工作频率的 2 倍来进行采样，而一般语音的说话频谱分布大部分小于 4kHz，因此低通滤波器（由 U7C 及相关 R、C 电路组成）的截止频率设计为 4kHz，而系统的采样率（由个人计算机的驱动程序控制）定为 8kHz，即每秒钟采样 8000 点语音数据，每一点的数据以一个字节来表示。

经过低通滤波器的语音信号然后接到由 R16、R17 及 U7D 所构成的电位调整电路，将原先具有双极性的语音信号电位转换至 0 到 5V 的范围，以满足 AD0804 的接口信号要求。图 4-13 为 8 位 A/D 转换电路，构成此电路的主要组件是 AD0804，将模拟信号（0 至 5V，由 Vin(+)输入）转换为 8 位的数字形式的数据，其转换时间为 100 μs。电路中 R1 及 C20 提供 A/D 所需的工作时序，实际的工作频率为：1/1.1RC=1/ [1.1×N10K×N100P]=910kHz。

此外为了简化与 PC 机的接口设计，在此 AD0804 芯片工作于自动信号转换的模式以此方式设计的话，一些控制信号，像 CS、RD、WR 和 INTR，便不必另外处理了。

其信号转换原理说明如下:

利用 555 产生单拍的起始触发信号，用来在电源接通时产生低电位工作脉冲，送入 WR 引脚，使芯片能开始转换。

在转换完后由 INTR 产生低电位工作脉冲，又促使 WR 引脚为低电位，使得芯片又重新进行转换的工作，如此一直循环下去。

由于 CS 及 RD 引脚接地，转换完的数字语音信号直接出现在数据总线上。

控制程序每隔一段时间(依采样率而定)，由数字接口读取数字语音数据到 PC 机中处理。

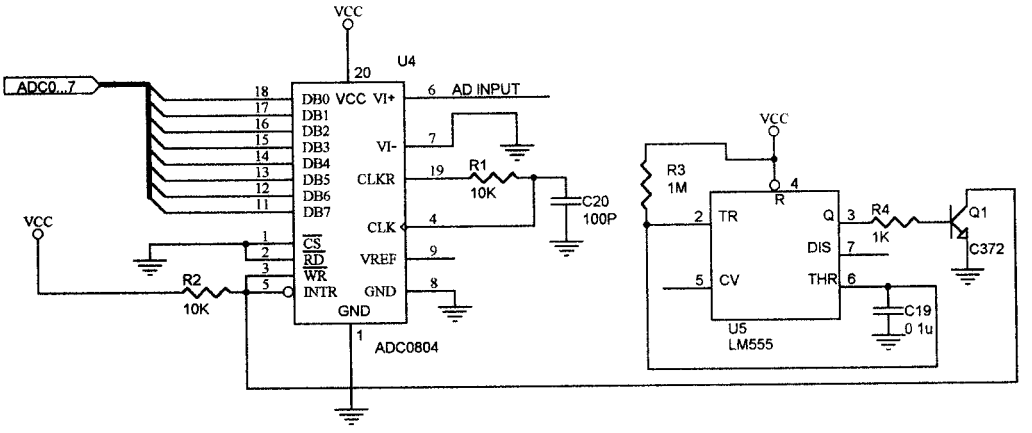


图 4-13　A/D 转换电路

4.6.2　数字接口

图 4-14 为数字接口的电路。PC 机方面的 I/O 译码是由 U1、U2 及 U3 来担任，其译码地址为 200H、210H 至 270H（采部分译码），可通过 DIP 开关 S1 来选择。避免与现有任何接口卡 I/O 口相冲突。任何时候 S1 只有一个位 ON，其余为 OFF。

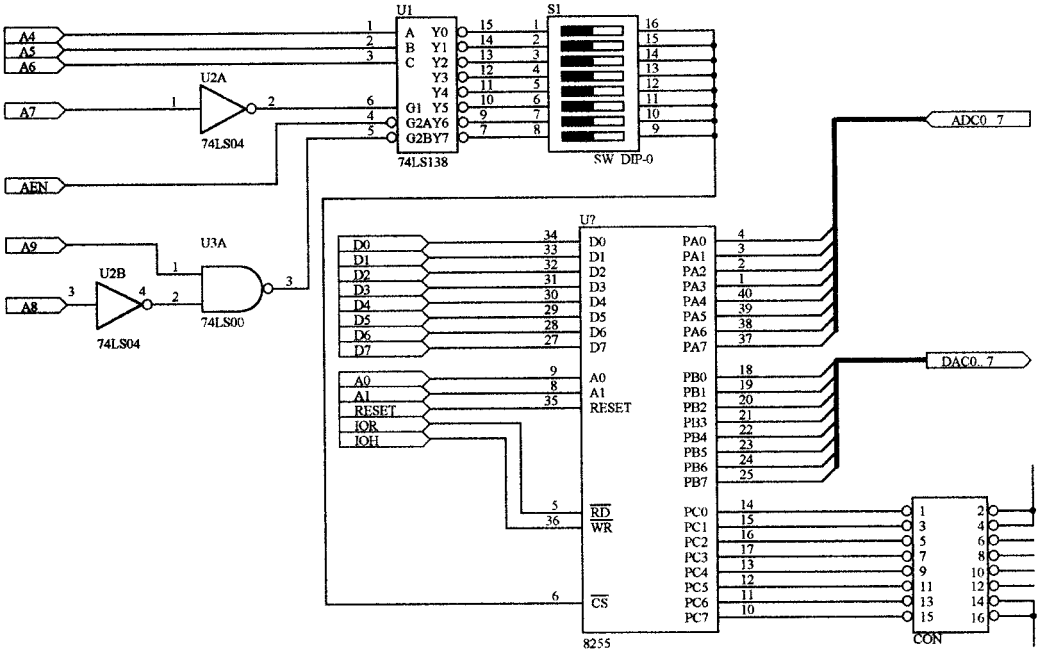


图 4-14　数字接口电路

在电路中主要靠 8255 担作数字接口的工作，8255 本身有 3 个 I/O 端口，PA、PB 及 PC。其中 PA 被设置为输入用作 A/D 语音输入采样，PB 被设置为输出提供给 D/A 作数字放音用，

端口 C 则保留在特殊的硬件扩充用。

4.6.3 D/A 电路

图 4-15 为 D/A 的电路设计，我们所用的 D/A 芯片是 DA08，分辨率为 8 位，具有双极性电流驱动的数字到模拟转换器。上节 DA08 功能说明中，我们选择参考电流 Iref 为 2mA，即从 VR+提供 2mA 的电流，因此适当的选择 R5、R6 和 R7 以配合此工作条件，分析请参考图 4-16。

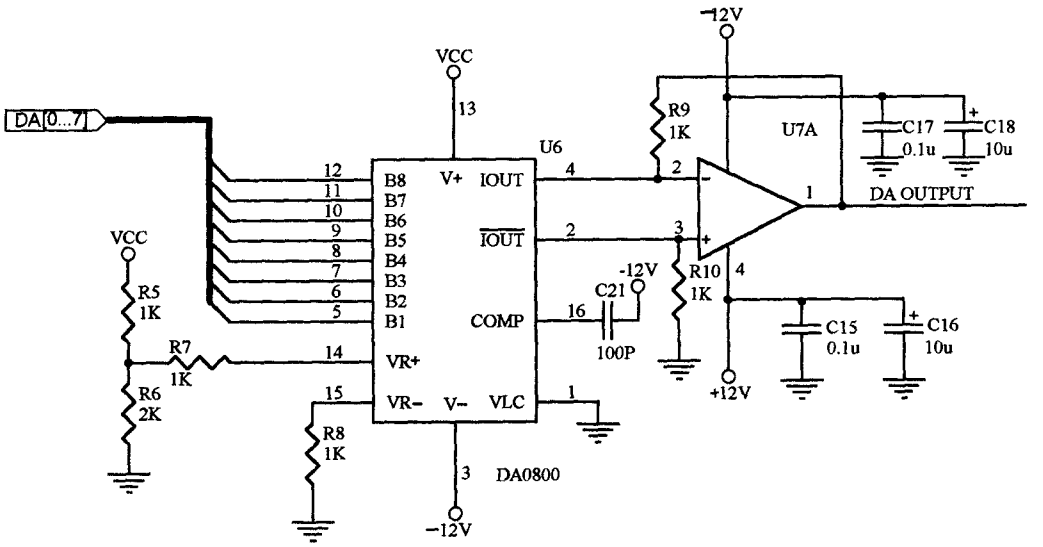


图 4-15 D/A 转换电路

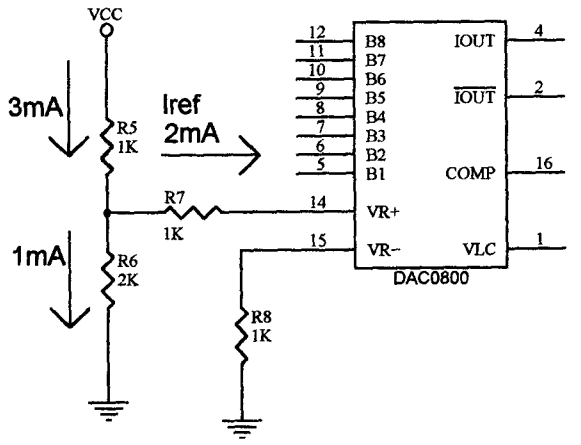


图 4-16 DAC08 Iref 电流分析

由于我们所要转换输出的信号为音频信号，其本身是一种双极性的信号，因此对杂音

的处理尤其重要。在无信号输出时，希望其输出对地是零电位，在此应用另一组运算放大器 U7A 做为电流到电压的转换器，U6 的正向输出至运算放大器的负端输入，而反向接至运算放大器的正端输入，达成双极性控制的目的。

原先由 VR+提供有 2mA 的电流，在满刻度输入时，I/O 引脚可以吸入 1.992mA（2×Ñ255/256）的电流，在理论上双极性控制时，可以达到正负 1.992V（1.992×Ñ1），此输出信号的振幅足以推动后级的相关模拟电路。

图 4-17 为语音输出滤波及放大电路。原先的数字信号经 D/A 转换、双极性电流至电压变换后成为模拟的语音信号，通过 4kHz 的低通滤波器（由 R20 及 C22 组成）以便滤除 4kHz 以上的高频刺耳杂音，再送往 U8 小功率放大器，做适当的功率放大而推动喇叭，其中可变电阻 VR2 为音量控制用。

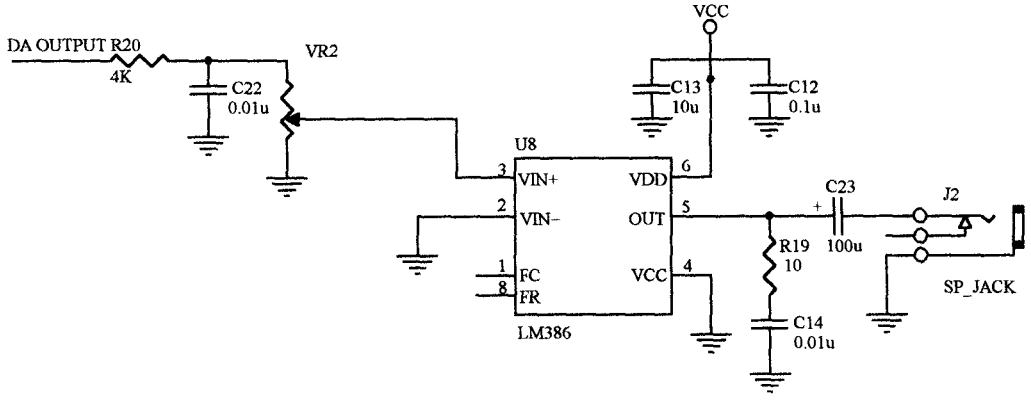


图 4-17　语音输出滤波器及放大器电路

习　题

1．若数字语音录音其采样频率为 8kHz，而分辨率为 8 位，则录音 1 分钟，将耗用多少内存？

2．若数字语音录音其采样频率为 8kHz，而放音频率各为 4kHz 及 16kHz，则得到的效果如何？

3．列举 ADC 的 4 种应用例子。

4．列举 DAC 的 4 种应用例子。

5．试描述 AD0804 芯片下列引脚功能：RD、WR、INTR。

6．试画出 AD0804 芯片工作于自动信号转换模式的电路及说明其工作原理。

7．试设计 PC 机 I/O 接口电路来控制 8255，其 I/O 基地址为 0X260。

8．若 8255 控制字为 0X99，则 8255 工作状态如何？

9．若要设置 8255 动作为模式 0 的操作，端口 A 输入，端口 B 及端口 C 为输出，则 8255 控制字为多少？

第5章　自行组装PC语音卡

上一章我们已经介绍过完整的语音卡设计原理，您如果想自己组装这样一片适配卡，可以参考本章所介绍的组装步骤，如果没有空自己组装，也可以直接订购此片语音卡成品，从制作适配卡及声控程序设计中，可以完全了解整个语音识别的信号处理过程，可以体会声控的趣味性，此片语音适配卡非常适合学生在声控系统专题制作方面的应用。

5.1　语音卡使用零件

表5-1　PC语音卡所需零件表

	零　件	数量	参数符号
IC	74LS138	1	U1
	74LS04	1	U2
	74LS00	1	U3
	ADC0804	1	U4
	LM555	1	U5
	DAC0800	1	U6
	TL084	1	U7
	LM386	1	U8
	8255-2	1	U9
IC插座	8 PIN	2	
	14 PIN	3	
	16 PIN	2	
	20 PIN	1	
	40 PIN	1	
电阻	1M　全部为1/4W	1	R3
	10K	3	R1，R2，R15
	5.6K	1	R18
	3.9K	3	R20，R13，R14
	2K	1	R6
	1K	7	R4，R5，R7-R11
	500	1	R12
	10	1	R19
	10K　%1精密电阻	2	R16，R17
	100KVR　可变电阻	2	VR1，VR2
电容	100u/16v　电解电容	1	C23
	10u/16v　电解电容	7	C1，C5，C13，C16，C18，C24，C25
	0.1u　积层电容	11	C6-C12，C14，C15，C17，C19
	0.01u　麦拉电容	3	C2，C3，C22
	100P　陶瓷电容	2	C20，C21
	20P　陶瓷电容	1	C4

续表

	零　件	数量	参数符号
三极管	2SC372	1	Q1
牛角座	16PIN	1	JP1
单声道插头	JACK	2	J1，J2
DIP 开关	DIP-8	1	S1

语音卡零件如上表 5-1 所示：

整片 PC 语音卡上使用的零件都可以在一般的电子元器件商店买到，建议自己组装的朋友按照以下语音卡使用零件表来买零件，一次把它买齐，免得为了个电阻还得跑一趟电子元器件商店。

5.2　DIY 自己装步骤

读者若对自己装语音卡有兴趣的话，首先是选购语音卡空的 PC 板，再来依所列的组件至电子元器件商店买齐所有零件后，便可以自己动手组装此片语音卡，有以下几个步骤：

步骤 1：焊上所有 IC 插座。由于 IC 组件可以重复使用，因此建议使用 IC 座，若板子出问题时，检修更换组件较为方便。

步骤 2：焊上所有电阻及可变电阻。

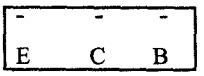
注意：R16 与 R17 为 10kΩ，误差为 1%的精密电阻，而其他电阻则为普通碳膜电阻。

步骤 3：装上 DIP 开关，单声道插座及牛角座。把 DIP 开关 1 至 8 全部拨于 OFF 处，唯独 6 在 ON，设定 I/O 端口地址为 250H。

步骤 4：焊上电容器。一共使用 4 种电容，分别为电解、积层、麦拉及陶瓷电容。其中只有电解电容为有极性的零件，长脚代表“+”，装上时注意脚位。

注意：C5（位于 MIC 接头的左侧）为 10μF 的极性电容，装上时右侧为“－”，请读者自行查看电路图来确认。

步骤 5：焊上晶体管 Q1，晶体管为有极性的零件，下图为焊接示意图。



晶体管 2SC372

步骤 6：按照 IC 参考符号，插上各 IC 组件。小心将 IC 完全插入 IC 座中，避免有 IC 接脚翘在外头。

步骤 7：以三用电表测量适配卡电源阻抗。将三用电表拨于欧姆档处，两个测试棒分别接于 VCC（+5V）及接地处，可以接在 U2 74LS04 脚位 7 及 14 代表地端及+5V 电源，其阻值应有数百欧姆，为 0 欧姆或是几欧姆，则可能是电源短路，请仔细查看一下是否有焊接短路的情形发生。若上述一切工作完成后，便可做下一单元的功能验证了。

5.3 功 能 验 证

在完成语音卡自行组装的工作后，如何做测试呢？可以使用本书下一章所介绍的以下几个应用程序，来确认适配卡上录音及放音动作是否正常。

- ✧ SPR.EXE ：语音录音程序。
- ✧ SPP.EXE ：语音放音程序。
- ✧ MP.EXE ：语音识别及编辑程序。

本语音识别卡由于软件及硬件公开，读者可以随心所欲地自行应用，从而发挥它的功能。目前支持此片语音识别卡的相关开发资源如下：

- ✧ 语音基本录音及放音控制程序。
- ✧ 语音识别及编辑整合程序 MP.EXE。
- ✧ TURBO C V2.0 语音卡声控链接库(SP.LIB)及语音辨识范例程序。

第 6 章　语音卡驱动程序设计

有了语音卡做语音数字录放音接口外，还得配合软件控制程序才能完成语音录放音的工作。为了方便程序设计，我们先介绍一个语音卡的录放音驱动程序。其中用到的程序技巧是PC机定时器应用，我们可以自己设计一个计时中断程序来处理声音数字化采样做语音录音，同理也可做数字语音播放。

6.1　录放音驱动程序

在微机系统设计上经常会用到定时与计数的功能，在PC机的主机板上也设计有 3 个定时/计数器，所使用的芯片是 8253(或 8254)定时/计数芯片，图 6-1 是PC机上定时器的使用情形，3 个通道都送入 1.19318MHz的工作时钟，但各设定为以下 3 个不同的用途：

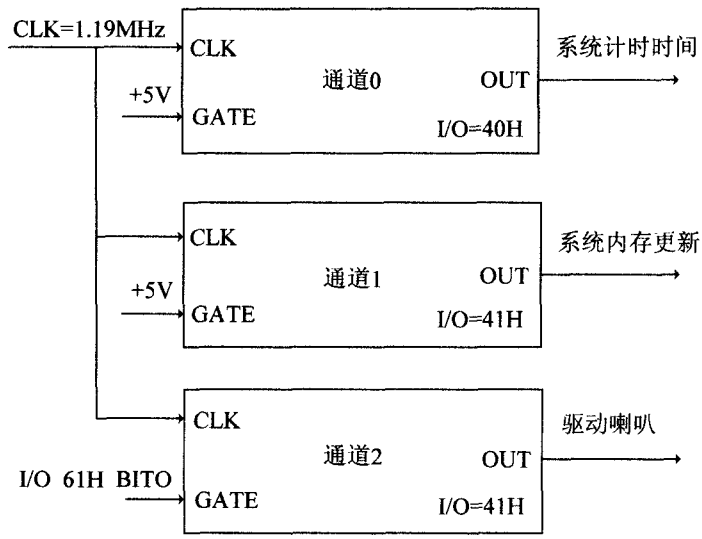


图 6-1　PC 机计时器的使用

- ✧ 通道 0：系统计时时间。
- ✧ 通道 1：系统内存更新。
- ✧ 通道 2：驱动喇叭发声。

1．通道 0：

通道 0 用来做系统一般定时器用，其闸控信号接高电位，使得定时器 0 一直工作。通道 0 的输出接到系统中断控制器 8259 的IRQ0 输入，BIOS设置为模式 3——方波发生器，赋初值为 0，因此在正常情况下每秒钟将产生 18.2 次(1193180/65536)的中断，BIOS就是利

用这个周期性的中断来维持计算机的时间计时。

2．通道 1：

通道 1 用来做系统内存更新用，闸控输入接高电位，定时器动作一直有效，PC机的BIOS在启动时即设定此通道工作于模式 2——做波特率发生器使用，定时器每 15μs产生一次内存更新信号，由于系统的主存储器采用动态RAM(DRAM)设计，因此内存更新周期与系统的正常运行有关，此通道的定时器不允许被使用者加以利用。

3．通道 2：

这个计时通道是用做喇叭驱动，闸控输入可以经由输出端口（I/O地址 61H）的位 0 加以控制启动或关闭。若启动后，设置为工作模式 3——做方波产生器，可以持续地由此通道送出固定频率的方波信号至喇叭，驱动喇叭发出声音。

6.1.1 重新设置定时器通道 0

了解PC机内部的定时器结构后，我们如何利用PC机定时器来设计自己的计时中断程序呢？那就是由 8253 的通道 0——系统计时时间着手，事实上PC机软件上一些常驻程序或是计算机病毒均是利用到这个地方。只不过一旦修改了系统计时工作的动作流程，计算机的系统时钟必然会不准确，得再想法子修正回来。另外如果自己的中断程序中有用到磁盘I/O的话，最好还是不要更改计时通道 0 的动作。

前面谈到通道 0 的输出是接到系统中断控制器 8259 的IRQ0 输入（鉴别码为 08H），而其真正中断服务程序的执行地址是存在地址(0:32)(8x4)起的 4 个字节处，此处地址(0:32)表示节区（Seqment）为 0，偏移值（Offset）为 32。也就是说中断程序的执行地址是一个向量，今天可以重新修改其向量地址便可以执行自己的中断程序了，但在修改计时中断向量前得将原来的中断向量保存起来，以便在程序执行完后将其还原。

而中断程序多长时间执行一次呢?根据我们程序需要而定，只要修改通道 0 的I/O地址40H的定时器初值即可。在完成了计时程序工作后，如果要结束整个控制程序，由于计算机上的时间被我们修改过了，于是最后得读取真实时钟(Real Time Clock，简称RTC)芯片上的数据来设定系统时间。一般PC机系统上均含有Motorola MC/146818A真实时钟或功能相同的芯片，此芯片可以通过外部的充电电池供电而不断地做计时工作，就如同家中的时钟，因此称它为“真实”时钟，此芯片可以独立担任计时的工作，并不会因为PC机的关机而停止计时。

通过以上的讨论，我们可以以下面几个子程序来完成计时中断程序的工作：

- ✧ getvect(no)：TURBO C函数用来取得中断编号no的程序向量地址。
- ✧ setvect(no,中断程序名称)：TURBO C函数用来设定中断编号值为no的程序执行中断程序执行地址，这一地址即为中断程序名称，可以利用getvect()函数先将原来的中断程序向量地址保存起来，再将自行设计的中断程序取代原来中断程序执行，待程序执行完后将中断程序向量还原回原来地址。
- ✧ set_timer(freq)：设定中断程序多长时间执行一次。
- ✧ restore()：还原原来的计时中断程序向量。

✧ read_rtc_time()：读取真实时钟的时间值，做系统时间的设定。

6.1.2 set_timer(freq)程序清单

```
set_timer(int freq)
{
unsigned p;
unsigned char plow, phi;
p=1193180/freq;                          /* 计算计数器赋值 */
plow=p%256;
phi=p/256;
disable();                               /* 禁止中断产生 */
outportb(0x43,0x36);                     /* 设置 8253 工作模式 */
outportb(0x40,plow);                     /* 写入计数值低字节 */
outportb(0x40,phi);                      /* 写入计数值高字节 */
enable();                                /* 允许中断产生 */
}
/*------------------------*/
restore()
{
disable();
outportb(0x43,0x36);                     /* 还原原来的计时中断时间常数 */
outportb(0x40,0);
outportb(0x40,0);
setvect(0x08, old_isr);                  /* 还原原来的计时中断程序向量 */
enable();
read_rtc_time();                         /* 读取真实时钟的时间值 */
}
/*-------------*/
read_rtc_time()
{
unsigned char cl,ch,dh;
struct time timept;
_AH=2;   geninterrupt(0x1a);             /* BIOS 插断呼叫控制真实时钟芯片 */
ch=_CH;  cl=_CL;  dh=_DH;
timept.ti_hour=10*(ch>>4)+(ch&0x0f);/* 取得小时数据 */
timept.ti_min=10*(cl>>4)+(cl&0x0f); /* 取得分钟数据 */
timept.ti_sec=10*(dh>>4)+(dh&0x0f); /* 取得秒数数据 */
timept.ti_hund=0;                        /* 百分之一秒设为 0 */
settime(&timept);                        /* 设定 MS-DOS 的系统时间 */
}
/*------------------*/
```

看过定时器的控制方法后，再来看看录放音驱动程序便容易多了。实际的录音中断程序为readadcisr()，而放音中断程序为talkisr()，真正录音控制程序为ad()，其原型被定义为：

```
unsigned ad(char *in, unsigned len);
```

- ✧ 传入值 ：*in为语音录音数据缓冲区地址指针。
- ✧ len ：最大语音数据缓冲区长度。
- ✧ 传回值 ：实际数字化录音的语音数据长度。

而放音控制程序为da()，其原型被定义为：

```
void da(char *out, unsigned bp, unsigned ep, unsigned len);
```

- ✧ 传入值：*out是语音放音数据缓冲区地址指针，bp是放音语音起点，ep是放音语音终点，len是最大语音数据缓冲区长度
- ✧ 传回值：无

6.1.3 SP.C 程序清单及其说明

6.1.3.1 SP.C 程序清单

```
1  /* SP.C   */
2  #include  <dos.h>
3
4  #define AD_PORT   0x250  /* 定义语音卡 A/D D/A I/O 地址 */
5  #define DA_PORT   AD_PORT+1
6  #define SILENCE   126
7
8  unsigned  in_count, count;
9  char  *buf;
10 /* 使用函数的原定义 */
11 void interrupt readadcisr(void);
12 void interrupt talkisr(void);
13 void interrupt (*old_isr)(void);
14
15 void set_timer(int freq);
16 void restore(void);
17 void read_rtc_time(void);
18 unsigned ad(char *in, unsigned len);
19 void    da(char  *out, unsigned bp, unsigned ep, unsigned len);
20 /*----------------------*/
21 void interrupt readadcisr(void)   /* 录音中断程序 */
22 {
23    buf[in_count]=inportb(AD_PORT);
24    in_count++;
25    outportb(0x20,0x20);
26 }  /* 送出中断程序执行完毕信号给中断控制器*/
```

```
/*-----------------------*/
void interrupt talkisr(void) /* 放音中断程序 */
{
  unsigned char s;

  s=buf[count]+128; /* 基址地址偏移量 */
  outportb(DA_PORT, s);
  count++;
  outportb(0x20,0x20);
} /* 送出中断程序执行完毕信号给中断控制器*/
/*-----------------------*/
unsigned ad(char *in, unsigned len) /* 录音子程序 */
{
unsigned buffer_end=len;
unsigned i;

 buf=in;
 in_count=0; /* 数字采样计数设为 0 */
 /* 将原来计时中断程序向量地址保留起来 */
 old_isr=getvect(0x08);
 setvect(0x08, readadcisr); /* 设定录音中断程序为计时中断程序 */
 set_timer(8000); /* 设定数字采样频率为 8K */

 do{    /* 循环做语音录音 */
    if(kbhit()) { getch(); break;}
 } while( in_count<buffer_end );

  restore(); /* 还原原来计时中断程序向量地址 */
  for(i=0; i<in_count; i++ )
   buf[i]=buf[i]-128;     /* 基址地址调整 */
  return in_count;  /* 传回数据计数值 */
}
/*---------------------------------*/
void da(out, bp, ep, len)  /* 语音放音控制程序 */
char  *out;
unsigned bp, ep, len;
{
unsigned buffer_end=len;

  buf=out; /* point to buffer for ISR */
  count=bp;
  /* 将原来计时中断程序向量地址保留起来 */
```

```
69   old_isr=getvect(0x08);
70   setvect(0x08, talkisr);/* 设定放音中断程序为计时中断程序*/
71   set_timer(8000); /* 设定放音速度为 8K */
72
73   do {   /* 循环做语音放音 */
74      if(kbhit()) { getch(); break;}
75   } while( ( count<ep) && (count<buffer_end) );
76   outportb(DA_PORT, SILENCE); /* 由 D/A 口送出静音的信号 */
77
78   restore(); /* 还原原来计时中断程序向量地址 */
79 }
80 /*------------------------------*/
81 void read_rtc_time(void)
82 {
83   unsigned char cl,ch,dh;
84   struct time timept;
85   /* BIOS 插断呼叫控制真实时钟芯片 */
86   _AH=2;   geninterrupt(0x1a);
87   ch=_CH;  cl=_CL;  dh=_DH;
88
89   timept.ti_hour=10*(ch>>4)+(ch&0x0f);  /* 取得小时数据 */
90   timept.ti_min=10*(cl>>4)+(cl&0x0f);  /* 取得分钟数据 */
91   timept.ti_sec=10*(dh>>4)+(dh&0x0f); /* 取得秒数数据 */
92   timept.ti_hund=0; /* 百分之一秒设为 0 */
93   settime(&timept); /* 设定 MS-DOS 的系统时间 */
94 }
95 /*------------------*/
96 void set_timer(int freq)
97 {
98 unsigned p;
99 unsigned char plow, phi;
100
101   p=1193180/freq;      /* 计算计数器赋值  */
102   plow=p%256;
103   phi=p/256;
104
105   disable();          /* 禁止中断产生 */
106   outportb(0x43,0x36); /* 设置 8253 工作模式 */
107   outportb(0x40,plow); /* 写入计数值低字节 */
108   outportb(0x40,phi);  /* 写入计数值高字节 */
109   enable();           /* 允许中断产生*/
110 }
```

```
111 /*------------------------*/
112 void restore(void)
113 {
114  disable();
115  outportb(0x43,0x36);     /* 还原原来的计时中断时间常数 */
116  outportb(0x40,0);
117  outportb(0x40,0);
118
119  setvect(0x08, old_isr); /* 还原原来的计时中断程序向量 */
120  enable();
121  read_rtc_time();        /* 读取真实时钟的时间值 */
122 }
123 /*--------------*/
```

6.1.3.2 程序说明

行号	说明
4	A/D端口地址定义。
5	D/A端口地址定义。
6	定义D/A静音数值。
11～19	所有使用函数的原型定义。
21～26	readadcisr()录音中断程序，每隔 0.125ms执行一次，即取样率为 8kHz。
23	由A/D端口输入语音数据放到缓冲区内。
24	计数加一。
25	送出中断程序执行完毕信号给中断控制器。
28～36	talkisr()放音中断程序，每隔 0.125ms执行一次。
32～33	将语音信号由缓冲区取出后加上基址地址偏移值后送往D/A口，之所以加上偏移值是为了满足DA08 接口信号要求。
38～58	ad()录音子程序。
43	将语音数据缓冲区指针传给工作缓冲区内以方便程序处理。
44	数字采样计数设为 0。
46	将原来计时中断程序向量地址保留起来，传给old_isr。
47	设定录音中断程序为计时中断程序。
48	设定数字采样率为 8k。
50～52	循环做语音录音，若是有键按下，或是语音数据缓冲区满了则跳离循环，结束录音的工作。
54	还原原来计时中断程序向量地址。
55～56	将存入缓冲区内的数据作基址地址调整，原先由A/D读进来的数据值为 0到 255，在减掉 128 后其范围为 - 128 到+127，以正负号的char型态表示。
57	传回数据计数值。
60～79	da()语音放音控制程序。

67　　将语音放音计数值指到放音的语音起点。
69　　将原来计时中断程序向量地址保留起来。
70　　设定放音中断程序为计时中断程序。
71　　设定放音速率为 8k。
73～75　　循环做语音放音，若是使用者按下了任意键，或是语音计数值到了放音终点，或计数值到了语音缓冲区的末端则结束放音的动作。
76　　由D/A端口送出静音的信号。
78　　还原原来计时中断程序向量地址。

6.2　PC 机上的录音程序

看过录放音驱动程序SP.C后，我们以TURBO C写一段程序做语音录音的工作。它的功能是在PC机上按下任意键则开始做语音录音，使用者可以对着麦克风输入语音，或是录下其他的音效，录音的时间长度约为 7 秒，欲结束录音按下任意键则停止录音，并把所录到的声音由语音卡放出来，并显示语音波形及录音总长度，当按下“ESC”键则结束程序执行。在此可以TURBO C的目标文件来整合，主程序为SPR.C，目标文件为SPR.PRJ，其内容为：SPR.C和SP.C。

即自动加载并编译SPR.C，最后连结成可执行文件SPR.EXE，在进入TURBO C的整合画面后，按下“ALT+P”键，在“project name”下输入“SPR”即可，再按下“ALT+R”键便可自动编译而执行。在执行的工作目录下，需要有以下两个图形画面的驱动程序：

- ✧ egavga.bgi：彩色屏幕。
- ✧ herc.bgi　：单色屏幕。

才可以看到语音波形，否则无法顺利开启图形接口。

6.2.1　SPR.C 程序清单及其说明

6.2.1.1　SPR.C 程序清单

```
1  /* SPR.C */
2
3  #include <graphics.h>
4  #include <stdio.h>
5  #include <alloc.h>
6  /* 定义语音卡 A/D D/A I/O 地址 */
7  #define  AD_PORT 0x250
8  #define  DA_PORT AD_PORT+1
9  #define  CR      AD_PORT+3
10 #define  CWORD 0x91
11 #define  BUFFER_LEN  64000 /* 定义语音缓冲区长度 */
12 #define  SILENCE     126
13
```

```
14 void getmemory(void);
15 void freememory(void);
16 void draw_signal(unsigned bb, unsigned ee);
17
18 char *title="SPEECH CARD A/D test.......";
19 char *signal;
20 int gd=DETECT, gm;
21 unsigned len;
22 /*-----------*/
23 main()
24 {
25 char ch;
26
27  getmemory();   /* 动态配置内存缓冲区 */
28  outportb(CR,CWORD);          /* 设定语音卡上 8255 的工作模式 */
29  outportb(DA_PORT, SILENCE); /* 端口 A 输入，端口 B 输出，端口 C 输出 */
30
31  while(1)
32   {
33    clrscr();
34    puts(title);
35    puts("Any key to record voice ....");
36    puts("ESC to exit ");
37
38    while(1) /* 使用循环来判断是否按下任意键 */
39      if( kbhit())
40       {
41        ch=getch();
42        if(ch==0x1b )  goto back;
43          else break;
44       }
45
46    puts("Please say .............");
47    puts("Any key to stop .......");
48    len=ad(signal, BUFFER_LEN);  /* 做声音录音 */
49
50    initgraph(&gd, &gm, ""); /* 初始化图形接口 */
51    draw_signal(0, len);  /* 将所录到的声音波形画出来 */
52    da(signal, 0, len, BUFFER_LEN); /* 将语音数据回放出来*/
53    getch();    /* 等待按下任意键离开 */
54    closegraph();  /* 关闭图形接口 */
55   } /* end while */
```

```
back:
 freememory(); /* 释放动态配置的内存缓冲区 */
}
/*-----------------*/
void getmemory(void) /* 动态配置内存缓冲区 */
{
 signal=(char far *)farcalloc(BUFFER_LEN ,sizeof(char));
 if (signal==NULL)
  { printf("out of memory !\n");exit(1);}
}
/*--------------------*/
void freememory(void)   /* 释放所配置的内存 */
{
  farfree((char  far *)signal);
}
/*----------------------------------------*/
void draw_signal(unsigned bb, unsigned ee)
{  /* 将语音波形画在屏幕上  */
  unsigned i, m, l;
  int maxx, y, ypos, xint;
  char mess[80];

  maxx=getmaxx();
  ypos=getmaxy()/2;
  line(5, ypos, maxx-5, ypos);
  setcolor(LIGHTGRAY);

  l=ee-bb;
  if(l<maxx) xint=1; else xint=l/maxx;

  moveto(5 ,ypos);
  for(i=5, m=bb; i<maxx-5; m+=xint, i++)
   {
    y=signal[m] + ypos;
    lineto(i, y);
   }

  sprintf(mess, "Voice length =%u", len);
  outtextxy(10, getmaxy()-80, mess);
  sprintf(mess, "Any key to continue ....");
  outtextxy(10, getmaxy()-50, mess);
}
```

```
98 /*---------------*/
```

6.2.1.2 程序说明

行号	说明
27	动态配置内存缓冲区。
28	设定语音卡上 8255 的工作模式，端口A为输入，端口B为输出，端口C为输出。
29	自D/A口送出静音信号。
33～36	显示操作说明。
38～44	循环来判断是否按下任意键，若是“ESC”键则结束程序执行，否则跳出循环做语音录音的工作。
46～47	提示使用者开始输入语音。
48	做声音录音。
50	初始化图形接口。
51	将所录到的声音波形画出来。
52	将语音数据回放出来。
53	循环等待按下任意键离开。
54	关闭图形接口。
57	释放动态配置的内存缓冲区。
60～65	getmemory()子程序，动态配置内存缓冲区来储存语音数据。
67～70	freememory()子程序，释放所配置的内存。
72～98	draw_signal()子程序，将语音数据按照起点及结束点的位置，将其波形画在屏幕上。

6.3 PC 机上的放音程序

本节我们以TURBO C写一段程序做语音放音的工作。若所储存的语音文件为SP.SP，放音程序为SPP.EXE，执行方法为：

SPP SP.SP <ENTER>

此时计算机将放出SP.SP的内容，目前SP.SP的声音内容为“语音卡”。如此一来，若我们自行录制了许多声音，可以利用此公用程序来检查声音的内容。此程序组成仍是以TURBO C的目标文件来完成，文件名为SPP.PRJ，内容如下：SPP.C和SP.C。

6.3.1 程序清单及其说明

6.3.1.1 程序清单

```
1 /* SPP.C */
2 #include <stdio.h>
3 #include <dos.h>
```

```
#include <alloc.h>
/*  定义语音卡A/D D/A I/O地址 */
#define  AD_PORT 0x250
#define  DA_PORT AD_PORT+1
#define  CR     AD_PORT+3

#define  CWORD     0x91
#define  BUFFER_LEN 64000
#define  SILENCE    126

char  *signal;

unsigned load_file(char *str);
void getmemory(void);
void freememory(void);
void show_help(void);
/*----------------*/
main(argc,argv)
int argc;
char *argv[];
{
unsigned len;

 getmemory(); /* 动态配置语音卡内存缓冲区 */
 outportb(CR,CWORD); /* 设定8255 */
 outportb(DA_PORT, SILENCE);

 if(argc==1)
   {
    show_help(); /* 告知SPP的使用方法 */
    exit(1);
   }

 len=load_file(argv[1]); /* 加载语音文件数据 */
 da(signal, 0, len, BUFFER_LEN);  /* 语音播放 */
 freememory();     /* 释放动态配置的内存 */
}
/*----------------------------*/
unsigned load_file(char *str) /* 加载语音数据文件 */
{
  FILE *in;
  unsigned i;

```

```
47   in=fopen(str,"rb");
48   if (in==NULL)
49    {
50     printf("Cannot open file %s !\n", str);
51     exit(1);
52    }
53   i=0;
54   while (!feof(in))
55    {
56      signal[i++]=getc(in);
57      if(i==BUFFER_LEN-1) break;
58    }
59   fclose(in);       return (i);
60  }
61  /*-----------------------*/
62  void getmemory(void)  /* 动态配置存储器缓冲区 */
63  {
64   signal=(char far *)farcalloc(BUFFER_LEN ,sizeof(char));
65   if (signal==NULL)
66    { printf("out of memory !\n");
67      printf("need %u bytes \n",BUFFER_LEN);
68      exit(1);
69    }
70  }
71  /*-------------------------*/
72  void freememory(void) /* 释放所配置的存储器 */
73  {
74    farfree((char  far *)signal);
75  }
76  /*---------------------*/
77  void show_help(void) /* 显示 SPP 放音程序的使用方法 */
78  {
79    clrscr();
80    puts("SP_CARD talk ..... ");
81    puts("Usage :  SPP voice filename");
82  }
```

6.3.1.2 程序说明

行号　　说明

27　　动态配置语音卡内存缓冲区。

31～35　若在输入执行文件SPP后未附加语音文件则告知SPP的使用方法。

37　　加载语音文件数据。

38　　做语音播放的功能。
39　　释放动态配置的内存。
42～60　load_file()子程序，加载语音数据文件。
77　　show_help()子程序，显示SPP放音程序的使用方法。

6.4　放音程序应用

上一节我们以已经介绍过支持语音卡的放音应用程序SPP.EXE，其执行方法为：

SPP　语音文件名 <ENTER>

我们可以写一个批处理文件来播放连续的几个语音文件。例如：SPP1.SP、SPP2.SP、SPP3.SP，则计算机会说出“1”、“2”、“3”。

那么在一般的应用程序中，如果不想直接驱动语音卡来说话，是不是也可以利用语音卡放音应用程序SPP.EXE呢？有一个简单的方法，那就是使用TURBO C的函数SYSTEM，来执行程序本身的外部执行文件。使用方法为：

system(可执行文件名字符串);

例如在C语言程序中，我们可以如下的程序来完成上述的功能：

```
system("SPP 1.SP");
system("SPP 2.SP");
system("SPP 3.SP");
```

完整的程序示范如下：

```
/* sp1.c */
/* spp.exe demo */

main()
{
  system("spp 1.sp");
  system("spp comp.sp");
  system("spp sp.sp");
}
```

此时计算机会播放出“1”、“计算机”、“语音卡”。利用此方法，我们可以事先录制一些语音文件或语音提示语，然后在程序的适当时机说出来，增加程序趣味性。

习　　题

1. 描述PC机 3 组定时器的用途。
2. 试说明如何利用PC机定时器来设计定时中断服务程序。
3. 试写一程序，用PC定时器来设计定时中断服务程序，由语音卡上 8255 的PC0 位产生 1ms宽的方波信号。
4. 何谓计时“RTC”？

第 7 章　语音分析及识别整合系统

本章将介绍一个实用的语音分析编辑及识别程序，在硬件方面完全支持语音卡，我们可以利用其内部的功能来做语音录音及放音的实验，并加以存盘，以备下回使用，此外可以做单音及连续语音的识别，对学习语音识别而言，是一个不错的学习软件。

7.1　整合系统功能介绍

语音分析及识别整合系统是一个以图形画面为向导的语音卡应用软件，主要设计来学习语音识别用，初学者可以很快的了解什么是计算机语音识别。先将此分析系统功能列举如下：

- ✧ 能对语音波形、瞬时能量、越零率、音高周期、自相关系数、线性预测编码及倒频谱等功能的分析。
- ✧ 以特定读者为语音识别的对象，并以中文数字0至9来做识别实验，可加以修改来识辨任何单字或词组。
- ✧ 具备单音及连续语音的识辨功能。
- ✧ 简单的参考样本建立程序及可修改的语音参考样本。
- ✧ 语音数据可储存至磁盘驱动器中供以后分析。
- ✧ 可由语音卡做实时线上的语音输入及识辨。
- ✧ 具有线上指令说明的功能。
- ✧ 以鼠标操作，容易学习。
- ✧ 指令功能以ICON触发方式或功能键来执行，方便使用。
- ✧ 具备语音输出的功能。
- ✧ 具有线上演示的功能，可辅助使用者学习。

由以上说明看来，此套系统功能相当多，在下节中，我们再来说明如何使用。

7.2　系统操作方法

读者在使用前请先安装鼠标，语音分析及识别整合系统的执行文件为MP.EXE，输入

MP　<ENTER>

出现如图7-1的主画面。其中包含了几个区域：

- ✧ 波形区。
- ✧ 波形短时距能量显示区。
- ✧ 参数区。
- ✧ 识别结果显示区。

- ✧ 处理信息区。
- ✧ 状态区。
- ✧ 0至9 ICON选择区。
- ✧ 指令ICON选择区。

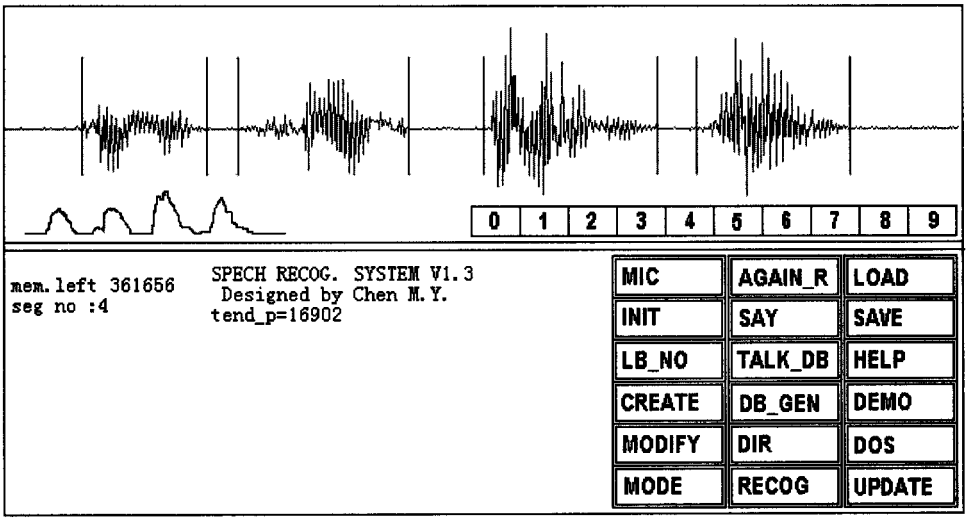


图 7-1 整合系统工作主画面

此套系统以图形接口构成，并配合鼠标操作，指令执行采用ICON触发方式或按键操作，并配合线上HELP说明，相信读者可以在短时间内熟悉本系统。

当读者第一次使用本系统时，请移动鼠标至指令区的HELP ICON并单击即可出现如下的线上求助使用说明：

====== SPEECH UTILITY KEY COMMAND ======

=	h -->	help	=
=	s -->	save speech file	=
=	l -->	load speech file	=
=	d -->	dir speech file	=
=	o -->	suspand to DOS	=
=	u -->	update data base	=
=	t -->	talk the speech	=
=	z -->	distortion list	=
=	MIC	microphone input	=
=	AGAIN_R	recog. again	=
=	INIT	clear screen	=
=	SAY	say buffer's data	=
=	TALK_DB	say data base file	=
=	CREATE	create data base	=

```
=      MODIFY        modify data base              =
=      MODE          change recog. mode            =
=      RECOG         recog. or not                 =
================================================
```

Right key to abort action
Left key to begin and end MIC input
Left key to pick 2 points to ZOOM the wave

当然除了操作鼠标外，也可按“h”键仍有此效果。看过此说明后，便能了解如何操作。先说明按键的功能：

- ✧ 按键“h”：线上指令功能说明。
- ✧ 按键“s”：将存在语音缓冲区中的数据存入磁盘驱动器中。
- ✧ 按键“l”：加载语音数据文件。
- ✧ 按键“d”：观看DB_VOICE目录下的所有语音文件名。
- ✧ 按键“o”：暂时离开识别系统而回到DOS下。
- ✧ 按键“u”：将存在内存中的语音参考样本写入标准语音数据库中。
- ✧ 按键“t”：将标准语音数据库的语音内容说出。
- ✧ 按键“z”：将经过单音识别后的误差结果显示出来。

本系统鼠标操作方法如下：

- ✧ 一般情况下右击表示放弃。
- ✧ 若为语音输入，则单击鼠标左键开始说话，系统并开始做语音采样，再次单击则停止采样，继续做下一步骤的处理。
- ✧ 任何时刻只要有波形出现即可移动鼠标指针至波形区内，屏幕中央会显示目前指针所指的波形位置的取样点数及时间长度，以毫秒（ms）表示。
- ✧ 任何时间只要有波形出现，便可直接在波形区上点出二点定一个范围（单击鼠标），将此范围内的波形予以放大来观察。

现将指令ICON区的18个操作动作详细介绍如下：

- ✧ MIC：为语音输入ICON，选到此ICON后单击鼠标开始说话，再按一次则停止语音输入，若已经设为直接识别模式（见RECOG功能）则直接进行识别，识别结果将逐一出现在结果显示区上及相对波形的附近。
- ✧ AGAIN_R：当语音缓冲区中已有数据，即波形出现在波形区时，可以改变成另一种识别方法（改MODE），对已知信号进行再次识别。
- ✧ LOAD：加载语音数据文件，可以先用DIR功能看看DB_VOICE目录中之所有文件名称再用此指令读入已经存在的数据。输入文件名称请输入全名，如1.sp。
- ✧ SAVE：将存在语音缓冲区中的数据存入磁盘驱动器中。*.SP为系统默认的语音文件名，所有30组参考样本的语音原始数据均存盘为这个文件扩展名，即0.SP至29.SP。读者可以自行将语音数据存为各种文件名。
- ✧ INIT：清除工作画面，如果工作画面过于混乱，可用此指令将画面清除。
- ✧ SAY：将缓冲区内的语音经D/A还原放出。

- ✧ TALK_DB：将标准语音数据库的语音内容放出，可以当作辅助说明使用，提示说那些话来进行识别。原始语音数据文件是以*.SP来储存在目录DB_VOICE中，目前设定为 0～30 组，其内容为中文数字 0～9 各 3 组。
- ✧ HELP：线上指令功能说明ICON。
- ✧ CREATE：用来建立语音数据库。原始语音数据存于DB_VOICE的目录下。进入此功能后系统会要求使用者输入 1 组数据库号码，此时以鼠标在 0 至 9 ICON处选取输入，而依照提示以语音输入 0 至 9 的单音。最后系统会将其数据求取语音特征参数，并存成标准语音数据库。
- ✧ DB_GEN：本功能可在修改语音参数后，重新产生一组新的DB.DAT。
- ✧ MODIFY：修改语音参考样本，进入此功能后，系统会询问修改哪一数字（0～9），使用鼠标在0至9 ICON区选取回答。接着系统询问修改哪一组参考样本(0、1、2)，同样使用鼠标在0至9 ICON区回答。在决定修改哪一数字、哪一组参考样本后，可重新输入语音了。
- ✧ DIR：观看DB_VOICE目录下的所有语音文件名。
- ✧ DOS：暂时离开识别系统而回到DOS下，输入“EXIT”可以返回系统。
- ✧ MODE：选择以下3种不同的语音识别方法：
  1、单音数字识别(使用DTW方法)
  2、连续数字识别(使用STA方法)
  3、特定字数的连续数字识别(使用LB方法)
- ✧ RECOG：显示于状态区，表示目前是在立即识别的状态下，即输入语音后马上进行识别，相对的显示为“no recog”。
- ✧ UPDATE：更新DB.DAT，将存在内存中的语音参考样本写入标准语音数据库中。
- ✧ LB_NO：设定使用LB方法当做连续语音识别单字的个数，即设定几阶做连音的识别。
- ✧ DEMO：对本套系统的功能作一连串的演示。

7.3 系统功能演示

在上一节中我们已经说明过分析系统的完整功能及操作，为使读者能更轻易上手，本节说明线上演示的功能，可辅助使用者学习。指令ICON区“DEMO”为线上演示的功能，移动鼠标至此区，并单击，则可以执行各种语音线上演示动作。演示动作可以分为以下几部分：

- ✧ HELP　　：说明系统线上的操作功能键，按下任何键可以继续做演示。
- ✧ TALK_DB：由语音卡放出语音数据库的内容。
- ✧ DIR　　：将DB_VOICE目录下的语音文件名列出来，如下所示。

```
Volume in drive C has no label
Volume Serial Number is 0151-0AF1
Directory of C:\M\DB_VOICE
```

```
[.]          [..]         0.SP        1.SP        10.SP
11.SP        12.SP        13.SP       14.SP       15.SP
16.SP        17.SP        18.SP       19.SP       2.SP
20.SP        21.SP        22.SP       23.SP       24.SP
25.SP        26.SP        27.SP       28.SP       29.SP
3.SP         4.SP          5.SP        6.SP        7.SP
8.SP         9.SP          COMP.SP     COMP1.SP    DEMO.SP
DEMO1.SP     VOICE.SP
            37 file(s)       124467 bytes
                          221511680 bytes free
```

✧ 单音识别：加载语音演示文件，文件名为DEMO.SP，内容为中文“1769”，均为单音，并使用单音识别的方法进行识别，并将识别的结果显示在相对波形的附近，结果如图 7-2 所示。

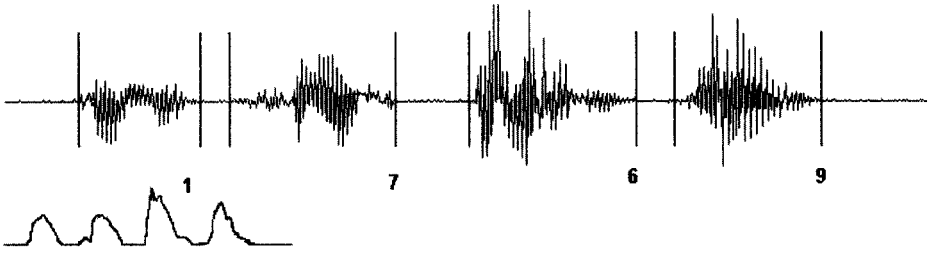


图 7-2　单音辨认结果

✧ SAY：将语音缓冲区的内容说出来，可以听见计算机说出“1769”。

✧ 单音比对结果列表：将单音“9”识别后的误差结果显示出来，如下所示。

```
==== SPEECH DISTORTION LIST ====
 0 -->133.905    1 -->141.022
 2 -->127.638    3 -->94.826
 4 -->124.833    5 -->104.316
 6 -->72.529     7 -->140.683
 8 -->148.437    9 -->72.331
10 -->117.521   11 -->133.084
12 -->117.102   13 -->127.596
14 -->120.581   15 -->105.527
16 -->95.767    17 -->141.120
```

18 -->139.259　19 -->85.495

20 -->120.112　21 -->138.878

22 -->112.227　23 -->127.905

24 -->141.169　25 -->105.368

26 -->85.981　27 -->139.222

28 -->161.580　29 -->93.621

top 5 list :　9　6 19 26 29

✧ 连音识别演示：系统先加载连音演示文件，文件名为DEMO1.SP，内容为中文“236”，数字“2”为单音，数字“36”为连音，并使用连音识别的方法进行识别，并将识别的结果显示在相对波形的附近，也将连音的断点自动标示出来，如图7-3所示，接着将语音缓冲区的内容放出来，可以听见计算机放出“236”的声音。

图 7-3　连续语音辨认及断点标示

✧ 波形观察：读者可以用鼠标来操作，进而观察语音波形的各种特性，可以在语音波形“2”的前后位置定出两点，则系统会将在此范围内的波形加以放大并且播放出这些内容，放大的波形如图 7-4 所示。同理可以在语音波形“36”的前后位置定出两点来观察其放大的波形，结果如图 7-5 所示。

图 7-4　语音“2”的波形放大

图 7-5　语音“36”的波形放大

7.4　分析声音波形的特性

整合系统的执行文件为MP.EXE，执行后自动加载语音演示文件DEMO.SP，工作画面会出现4个语音波形，本节将利用这些波形，来分析声音波形的特性，并说明如何操作鼠标来观察语音的信号。

首先确认4个语音波形的内容，移动鼠标至“TALK”ICON 处单击，此时计算机会说出4个单音“1769”，可以在语音“1”的前后位置定出两点，放大的波形如图7-6所示，波形除了放大外也显示出单字的端点位置，同时还播放出其内容。中文“1”的发音只有元音，可以明显地看出周期性的波形变化，右击会回到原先的波形。

图 7-6　语音“1”的波形放大

下一步我们来关察一下子音的特性，中文“7”的发音是子音加上只有元音，让我们看看数字“7”的波形，可以在语音“7”的前后位置定出两点，此时系统会放出“7”这个发音，并把完整的“7”的波形放大如图 7-7 所示，此时我们可以注意到子音的部分较短，一般称为摩擦音或是鼻音，振幅较小，而元音的振幅较大，并且会呈现周期性的变化。

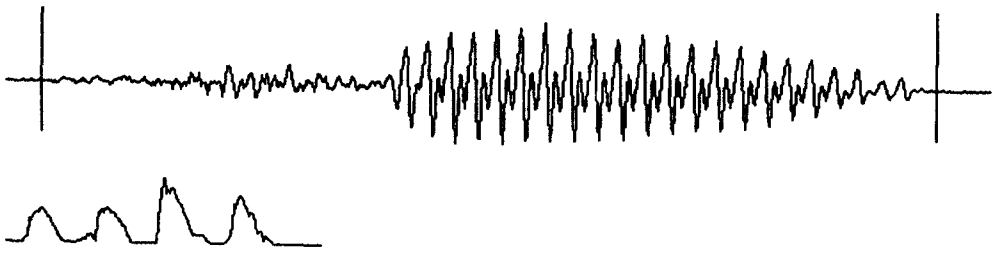


图 7-7　语音“7”的波形放大

7.5 线上进行语音识别

语音分析及识别系统具有单音及连续语音的识别功能，当然所识别的单字为中文数字0至9，读者如果备有语音卡，则可以线上输入语音而进行语音识别，本节将说明其操作步骤。识别系统提供以下3种语音识别：

- ✧ 单音数字识别。
- ✧ 连续数字识别。
- ✧ 特定字数的连续数字识别。

7.5.1 单音数字识别步骤

步骤1：移动鼠标至“MIC”ICON处单击，此时对着麦克风说一个数字，再单击一次则完成语音输入的动作。

步骤2：系统马上会显示所输入的语音波形，并经过语音信号切割的处理，定出语音数字的端点并标示出来。

步骤3：移动鼠标至“AGAIN_R”ICON处，单击进行单音数字识别，结果会出现在波形的附近。

步骤4：移动鼠标至“RECOG”ICON处单击，则一旦输入语音后马上进行识别。

此单音数字识别模式屏幕中会显示“DTW”，一次也可以同时说出数个单音，如“1234”则计算机会逐一将其识别出来。

7.5.2 连续数字识别

步骤1：移动鼠标至“MODE”ICON处单击，则识别模式变为“STA”，表示做连续数字识别。

步骤2：语音输入的操作动作同上，说出连续数字，如“236”，结果会出现在波形的附近的单音的端点或是连音的断点也会标示出来。

此连续数字识别模式屏幕中会显示“STA”，可以说数个单独数字，加上连续数字，但如果连续数字的字数超过两个以上时，识别正确率会降低。

7.5.3 特定字数的连续数字识别

步骤1：移动鼠标至“MODE”ICON处单击，则识别模式变为“LB”，表示做连续数字识别，原定是做连续2个数字的识别。

步骤2：语音输入的操作动作同上，说出连续数字，如“66”，结果会出现在波形的附近，其连音的断点也会标示出来。

步骤3：移动鼠标至“LB_NO”ICON处单击，则识别字数可以改变，此时可以移动鼠标至“0 ～ 9”数字选择ICON处单击来设定。如选择 3，则可以一次说出 3个连音数字，如“456”。

此特定字数的连续数字识别模式，屏幕中会显示“LB”，可以先设定要识别几个连续数字，一旦设定后，计算机一定会识别出所指定的连音数目字，但是若数字说的太快，则识别率不高。

7.6 产生及编辑语音波形文件

本节说明如何利用语音编辑程序来产生及编辑语音波形文件，以备下面几章使用，可以当作语音提示语使用，在程序执行的适当时机播放出来。例如我们要制作语音提示语“计算机”，并将编辑后的数据存盘为 COMP.SP，则操作如下：

步骤 1：移动鼠标至 MIC ICON 区单击，说出“计算机”，再次单击，则系统完成声音录音的工作，并将波形显示出来，如图 7-8 所示。

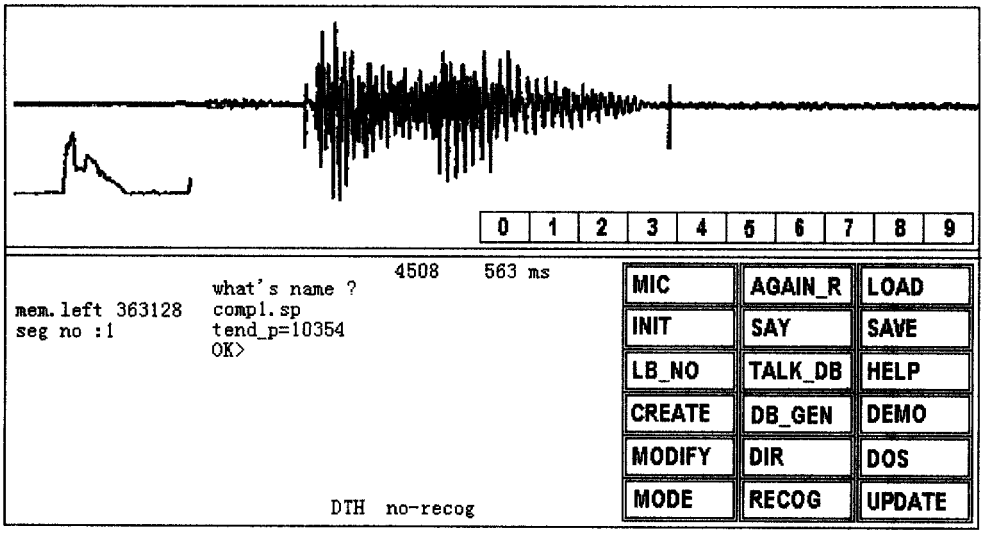


图 7-8 录制语音波形

步骤 2：移动鼠标在语音波形的前端位置单击，再移动鼠标至波形的末端位置，再次单击，可以将整段语音切割出来，系统将此范围内的语音回放，同时波形予以放大，执行结果如图 7-9 所示。此时可以按下“s”键，对所分离出来的语音数据进行存盘，按下“t”键，再次对语音回放。

步骤 3：按下“s”键，输入文件名 COMP.SP，存为名为“计算机”的语音文件。

步骤 4：右击鼠标回到主画面，按“ESC”键，返回DOS。

步骤 5：查看语音文件大小。

```
DIR    DB_VOICE\COMP   <ENTER>
 Volume in drive D has no label
 Volume Serial Number is 321B-1BDF
 Directory of D:\M\DB_VOICE
```

```
COMP        SP          4096 11-21-94    8:38p
      1 file(s)         4096 bytes
                        2584576 bytes free
```

只有 4096 字节。

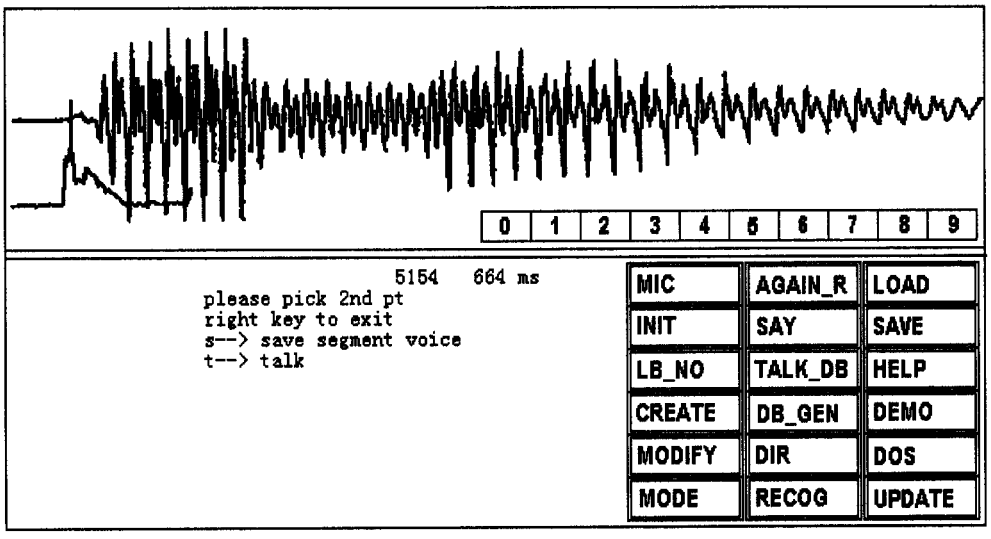


图 7-9　语音波形切割并放大

步骤 6：利用第 6 章所介绍的语音放音程序SPP.EXE来检查语音文件内容。

```
SPP    COMP.SP <ENTER>
```

可以听到计算机播放出“计算机”。至此我们已经制作好了一个语音文件，下回可以按照需要再制作其他的语音文件，在以后几章中可以使用。

第 8 章　语音卡声控链接库使用

看过语音卡的硬件设计原理及驱动程序设计后，读者是否有种冲动想要自己设计属于自己的声控程序？应从何处下手？语音辨识是一项热门的设计技术，由于包含的相关知识过于复杂，使许多人往往望而止步，现在为了推广相关的应用及方便学生的专题制作参考，我们特别推出了语音卡 TURBO C 声控链接库，只要您会 C 语言程序设计，即可让您自己设计声控程序，使用声控链接库可以让您轻易地实现“芝麻开门”的梦想。

8.1　声控链接库简介

8.1.1　声控链接库特色

- ✧ 以语音输入卡做声音输入。
- ✧ 以 TURBO C 设计计算机语音控制系统。
- ✧ 提供特定语者的单音、字、词识别功能。
- ✧ 不限定说话语言，方言也可以。
- ✧ 具有自动语音输入侦测的功能。
- ✧ 链接库提供声音录音、放音、语音切割、分析、识别等功能。
- ✧ 几行程序即可完成语音识别的工作。
- ✧ 备有多个有趣的声控应用范例程序供参考，易学易用。
- ✧ 看了马上可以设计属于自己的计算机声控控制系统。
- ✧ 特别适合在自己设计的专题中加入声控的功能。
- ✧ 识别率可达 90%以上。

8.1.2　系统配备

- ✧ 适用机型“PC 80486DX-33 以上的兼容机型。
- ✧ 操作系统 MS-DOS 3.0 以上
- ✧ 单色或彩色屏幕。
- ✧ 语音卡。
- ✧ 动圈式麦克风。
- ✧ 喇叭。
- ✧ TURBO 声控链接库 SP.LIB。
- ✧ TURBO C V2.0 C 编译器。

8.2 链接库内容介绍

本节说明语音卡 TURBO C 声控链接库所有相关控制函数的功能，有了这些特殊功能的函数，您不一定要完全了解语音识别的详细控制技术，便可以轻易地设计出声控应用程序了。先将基本的 15 个控制函数的功能简述如下：

1. SP_init_sp
   功能：侦测语音卡是否存在并对其初始化。
2. SP_mic_ip
   功能：由麦克风做声音手动控制输入采样。
3. SP_auto_mic_ip
   功能：由麦克风做声音自动输入采样。
4. SP_sp_out
   功能：经由语音卡将声音播放出去。
5. SP_separate
   功能：将语音信号进行语音切割为一段语音。
6. SP_separatex
   功能：将语音信号做多段语音的切割。
7. SP_analysis
   功能：将语音信号做特征分析。
8. SP_sp_match
   功能：语音识别函数。
9. SP_sp_match1
   功能：语音识别函数。
10. SP_load_voice
    功能：加载语音原始数据文件。
11. SP_save_voice
    功能：对语音数据进行存盘。
12. SP_save_voice1
    功能：对语音数据进行存盘。
13. SP_load_db
    功能：加载语音特征参数参考样本数据文件。
14. SP_save_db
    功能：对语音参考样本数据进行存盘。
15. SP_show_wave
    功能：显示语音原始波形文件

完整的使用说明如下：

8.2.1 SP_init_sp 函数

- ✧ 功能：侦测语音卡是否存在并对其初始化。
- ✧ 语法：int SP_init_sp(int io_add)。
- ✧ 说明：对语音卡按照指定的 I/O 地址做初始化的工作。
- ✧ 参数：io_add 语音卡的选定 I/O 地址。
- ✧ 传回值：1 表示成功而 0 表示失败。

8.2.2 SP_mic_ip

- ✧ 功能：由麦克风做声音手动控制输入采样。
- ✧ 语法：unsigned SP_mic_ip(char *in, unsigned len, int freq);。
- ✧ 说明：由麦克风做手动声音采样，执行后程序依所设定的采样频率。由语音卡进行语音采样，若按下任何按键则采样终止。
- ✧ 参数：in 存放输入语音数据的数组指针，
  len 语音数据缓冲区的长度，
  freq 所设定的采样频率。
- ✧ 传回值：实际语音长度。

8.2.3 SP_auto_mic_ip

- ✧ 功能：由麦克风做声音自动输入采样。
- ✧ 语法：unsigned SP_auto_mic_ip(char *in, unsigned len, int *exit_loop,unsigned en_begin, unsigned en_end, int freq);。
- ✧ 说明：由麦克风做自动声音采样，即程序自动侦测使用者是否有说话，若有则进行语音采样，若没有则继续等待，直至有声音输入。
- ✧ 参数：in 存放输入语音数据的数组指针，
  len 语音数据缓冲区的长度，
  exit_loop 若为 1 表示使用者按键，
  en_begin 语音起点的参考能量值，
  en_end 语音终点的参考能量值，
  freq 所设定的采样频率。
- ✧ 传回值：实际语音长度。

8.2.4 SP_sp_out

- ✧ 功能：经由语音卡将声音播放出去。
- ✧ 语法：void SP_sp_out(char *out, unsigned bp, unsigned ep, unsigned len);。
- ✧ 说明：将语音缓冲区内的声音经由语音卡回放出来。

✧ 参数：out　输出语音缓冲区的指针，
　　bp　将要说出语音数据的起点位置，
　　ep　将要说出语音数据的终点位置，
　　len　语音缓冲区最大长度。

✧ 传回值：无。

8.2.5 SP_separate

✧ 功能：将语音数据进行语音切割为一段语音。

✧ 语法：int SP_separate(char　*in, unsigned tend_p,unsigned *bp, unsigned *ep);。

✧ 说明：对语音信号做切割的动作。去除静音部分，而找出具有声音信息的位置。

✧ 参数：in　语音信号缓冲区的指针，
　　tend_p　实际采样到的语音长度，
　　bp　数组储存语音段起点位置，bp[0]表示第一个字的起点位置，
　　ep　数组储存语音段终点位置，ep[0]表示第一个字的终点位置，

✧ 传回值：1 表示该段语音可以切割出一个单音来，其起点及终点位置分别存于数组 bp[0]及 ep[0]中，0 表示该段语音没有有效的语音。

8.2.6 SP_separatex

✧ 功能：将语音数据做多段语音的切割。

✧ 语法：int SP_separatex(char　*in, unsigned tend_p,unsigned *bp, unsigned *ep);。

✧ 说明：对语音信号做切割的动作。去除静音部份，而找出具有声音讯息的位置。

✧ 参数：in　语音信号缓冲区的指针，
　　tend_p　实际采样到的语音长度，
　　bp　数组储存语音段起点位置，bp[0] 表示第一个字的起点位置，
　　ep　数组储存语音段终点位置，ep[0] 表示第一个字的终点位置。

✧ 传回值：x 表示该段语音可以切割出 x 个单音来，其起点及终点位置分别存于数组 bp[x]及 ep[x]中，0 表示该段语音没有有效的语音。

8.2.7 SP_analysis

✧ 功能：将语音信号做特征分析。

✧ 语法：int　SP_analysis(char　*sp, unsigned bb, unsigned ee,
float (*fea)[LEVEL+1]);。

✧ 说明：对语音信号做分析，找出其中的特征参数，当作对比识别用。

✧ 参数：sp　语音缓冲区指针，
　　bb　将要做分析的语音起点，
　　ee　将要做分析的语音终点，

fea　　数组储存经分析后求得的语音参数。

✧ 传回值：1 表示分析成功，而表示 0 分析失败

8.2.8 SP_sp_match

✧ 功能：语音识别函数。

✧ 语法：int SP_sp_match(float (*db)[FRAME][LEVEL+1], int db_no,
float (*fea)[LEVEL+1], int frames);。

✧ 说明：对所取得的待测语音特征参数(存于数组 fea 中)与原先数据库内的参考样本语音特征参数(存于 db 数组中)做识别对比，进而找出其中最相似的一组当做识别的结果。

✧ 参数：db　　原先已知的数据库内参考样本特征参数，
db_no　语音参考样本内的样本总数，
fea　　未知语音数据特征参数，
frames　未知语音数据共有几个音框将要进行对比。

✧ 传回值：对比结果数字编号，其值由 0 到(db_no-1)。

8.2.9 SP_sp_match1

✧ 功能：语音识别函数。

✧ 语法：int SP_sp_match1(float (*db)[FRAME][LEVEL+1], int db_no,
float (*fea)[LEVEL+1], int frames, float *dist);

✧ 说明：对所取得的待测语音特征参数(存于数组 fea 中)与原先数据库内的参考样本语音特征参数(存于 db 数组中)做识别对比，进而找出其中最相似的一组当做识别的结果。

✧ 参数：db　　原先已知的数据库内参考样本特征参数，
db_no　语音参考样本内的样本总数，
fea　　未知语音数据特征参数，
frames　未知语音数据共有几个音框将要进行对比，
dist　　列出待测语音特征参数与参考样本语音特征参数的各个对比失真值，最小的值在正常情况下表示识别结果的失真度。

✧ 传回值：对比结果数字编号，其值由 0 到(db_no-1)。

8.2.10 SP_load_voice

✧ 功能：加载语音原始数据文件。

✧ 语法：int SP_load_voice(char　*in,　unsigned len,
unsigned *rlen, char *fname);

✧ 说明：将文件名为 fname 的语音数据文件加载至语音数据缓冲区 in 内。

✧ 参数：in　　存放语音数据的缓冲区数组指针，
　　len　　语音缓冲区的最大长度，
　　rlen　　实际的语音文件长度，
　　fname　语音文件文件名。

✧ 传回值：1 表示文件加载成功，0 表示文件加载失败

8.2.11 SP_save_voice

✧ 功能：对语音数据进行存盘。

✧ 语法：int SP_save_voice(char　*out, unsigned rlen, char *fname);。

✧ 说明：将语音缓冲区内的数据存为语音文件，文件名为 fname。

✧ 参数：out　　语音数据缓冲区数组指针，
　　rlen　　语音文件长度，
　　fname　语音文件文件名，

✧ 传回值：1 表示存盘成功，0 表示存盘失败。

8.2.12 P_save_voice1

✧ 功能：对语音数据进行存盘。

✧ 语法：int SP_save_voice1(char *out, unsigned bp,　unsigned ep,
　　char *fname);。

✧ 说明：将语音缓冲区内的数据存为语音文件，文件名为 fname。

✧ 参数：out　　语音数据缓冲区数组指针，
　　bp　　欲储存的语音段起点位置，
　　ep　　欲储存的语音段终点位置，
　　fname　语音文件文件名。

✧ 传回值：1 表示存盘成功，0 表示存盘失败。

8.2.13 SP_load_db

✧ 功能：加载语音特征参数参考样本数据文件。

✧ 语法：int SP_load_db(float (*db)[FRAME][LEVEL+1],
　　int word_no, char *fname);。

✧ 说明：将文件名为 fname 的参考样本文件加载至已知特征参数数组 db 内。

✧ 参数：db　　已知特征参数数组指针，
　　word_no　已知单字个数，
　　fname　　参数样本文件名，

✧ 传回值：1 表示加载成功，0 表示加载失败。

8.2.14 SP_save_db

- ✧ 功能：对语音参考样本数据进行存盘。
- ✧ 语法：int SP_save_db(float (*db)[FRAME][LEVEL+1],
  int word_no, char *fname);。
- ✧ 说明：将已知参考样本数据存为数据库，文件名为 fname。
- ✧ 参数：db　已知特征参数数组指针，
  word_no　已知单字个数，
  fname　参数样本文件名。
- ✧ 传回值：1 表示存盘成功，0 表示存盘失败。

8.2.15 SP_show_wave

- ✧ 功能：显示语音原始波形文件。
- ✧ 语法：void SP_show_wave(char *out, unsigned bb, unsigned ee,
  int x1, int y1, int x2, int y2, float fact,
  unsigned *bp, unsigned *ep, int segs);。
- ✧ 说明：将语音数据缓冲区内的波形依指定的起点及终点，显示至屏幕上。
- ✧ 参数：out　语音数据缓冲区数组指针，
  bb　将要显示波形的起点位置，
  ee　将要显示波形的终点位置，
  x1　屏幕显示区域的左上角 X 坐标，
  y1　屏幕显示区域的左上角 Y 坐标，
  x2　屏幕显示区域的右下角 X 坐标，
  y2　屏幕显示区域的右下角 Y 坐标，
  fact　所显示波形振幅缩小比例，值越小波形越小，
  bp　经波形分析后的断点起点位置，bp[0]表示第一个单音的起点，
  ep　经波形分析后的断点结束点位置，ep[0]表示第一个单音的结束点，
  segs　语音一共可以被切割为几个单音。
- ✧ 传回值：无。

8.3 基本范例程序说明

看过声控链接库所有相关控制函数的功能介绍后，本节举一些简单的基本范例程序来说明声控链接库的使用。在前面第 3 章语音识别处理方法介绍中，我们提到语音识别处理步骤，并以几个简单的范例程序来做说明，在本节中将以声控链接库为主，来介绍这些程序怎么设计。以下是语音识别处理步骤的范例程序：

- ✧ 语音信号输入：使用范例程序 SI.EXE 及 VO.EXE 做实验。

✧ 语音信号切割：使用范例程序 SEG.EXE 做实验。
✧ 特征参数求取：使用范例程序 ANA.EXE 做实验。
✧ 语音识别对比：使用范例程序 MATCH.EXE 做实验。

有关执行结果说明，读者请自己到第 3 章查阅。

8.3.1 如何编译程序

步骤 1：TURBO C 使用大型(Large)内存模式来编译程序，设定如下：

ALT+O → Compiler → Model→ Large

步骤 2：使用 PRJ 文件进行整合编译，如编译 SI.C 程序，使用 SI.PRJ 文件。

ALT+P→ <ENTER> → SI

步骤 3：产生执行文件。

ALT+R

8.3.2 SI.EXE：观察语音信号数值

原始控制程序为 SI.C，TURBO C 目标文件名为 SI.PRJ。

```
1  /*************************************************
2  * Program: SI.C                                  *
3  * Description: PC SPEECH CARD use SP_LIB example.  *
4  *            MIC i/p  look data                  *
5  * Copyright (C) VICTOR SPEECH LAB.  1996, 1997    *
6  *************************************************/
7
8  #include <alloc.h>
9  #include <dos.h>
10 #include <stdio.h>
11
12 #define AD_PORT   0x250
13
14 #include "sp.h" /* 加载声控链接库头文件声明 */
15 #define   char  signed char
16 /*-----------------*/
17 main()
18 {
19 char c;
20 /* 检查语音卡是否存在，若存在则初始化语音卡 I/O 口 */
21  if(!SP_init_sp(0x250))
22   { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
23  clrscr();
24
```

```
25  puts("Voice data list :");
26  while(1) /* 无穷循环 */
27   {
28    c=inportb(AD_PORT)-128; /* 由A/D口输入语音数据 */
29    printf("%d ", c);  /* 输出语音数据数值 */
30    delay(200);        /* 延迟一段时间 */
31    if(kbhit()) break; /* 若有按键则跳离循环 */
32   }
33  puts("End of VOICE data list...");
34 }
```

8.3.3 VO.EXE：观察语音信号波形

原始控制程序为 VO.C，TURBO C 目标文件名为 VO.PRJ。

```
1  /*************************************************
2  * Program: VO.C                                  *
3  * Description: PC SPEECH CARD use SP_LIB example.  *
4  *            MIC i/p  speaker o/p                *
5  * Copyright (C) VICTOR SPEECH LAB.  1996, 1997    *
6  *************************************************/
7
8  #include <graphics.h>
9  #include <alloc.h>
10 #include <dos.h>
11 #include <stdio.h>
12
13 #include "sp.h" /* 加载声控链接库头文件声明 */
14 #define   LEN 64000
15 #define   char  signed char
16
17 char  *sp;        /* 语音数据的数组指针 */
18 unsigned sp_len;  /* 语音数据长度 */
19 unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
20 int segs;         /* 语音分段数 */
21 int gd=DETECT, gm;
22
23 /*-----------------*/
24 main()
25 {
26 int back;
27
28  if(!SP_init_sp(0x250))  /* 初始化语音卡I/O口 */
```

```
29   { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
30  /* 动态配置语音数据缓冲区 */
31  sp=(char  *)calloc(LEN ,sizeof(char));
32  if (sp==NULL) { printf("%c out of memory !\n", 7); exit(1);}
33
34  clrscr();
35
36  back=0;
37 while(1)
38  {
39   clrscr(); /* 显示操作信息 */
40   puts("SP CARD SPEECH RECOG. LIB. DEMO → mic i/p  sp o/p  ");
41   puts("Any key to start MIC i/p ....");
42   getch();
43   puts("Please say something ......");
44   puts("Any key to stop MIC i/p ....");
45   be(); /* 哔一声 */
46
47 /* 语音录音，若有按下任何按键则采样终止 */
48   sp_len=SP_mic_ip(sp, LEN, 8000);
49 /* 初始化图形接口 */
50   initgraph(&gd, &gm, "");
51   SP_show_wave(sp, 3000, sp_len, 0, 0, getmaxx(), getmaxy(),
52         1.0, bp, ep, segs); /* 显示语音信号波形 */
53 /* 播放出该段语音内容 */
54   SP_sp_out(sp, 0, sp_len, LEN, 8000);
55 /* 显示操作信息 */
56   outtextxy(10, getmaxy()-100, "Any key to continue ...");
57   outtextxy(10, getmaxy()-120, "<ESC> to exit ! ");
58 /* 等待按键处理 */
59   if(getch()==27) back=1;
60   closegraph();  /* 关闭图形接口 */
61   if(back) break;
62 }
63   free(sp); /* 释放动态配置的语音数据缓冲区 */
64 }
65 /*------------------*/
66 be() /* 哔一声 */
67 {
68   sound(500);
69   delay(100);
70   nosound();
```

```
71 }
72 /*---------------*/
```

8.3.4 SEG.EXE：语音信号切割

原始控制程序为 SEG.C，TURBO C 目标文件名为 SEG.PRJ。

```
1  /*****************************************************
2  * Program: SEG.C                                    *
3  * Description: PC SPEECH CARD use SP_LIB example.    *
4  *        MIC i/p speaker o/p show seg. wave         *
5  * Copyright (C) VICTOR SPEECH LAB.  1996, 1997      *
6  *****************************************************/
7  #include <graphics.h>
8  #include <alloc.h>
9  #include <dos.h>
10 #include <stdio.h>
11
12 #include "sp.h" /* 加载声控链接库头文件声明 */
13 #define   LEN 64000
14 #define   char     signed char
15
16 char  *sp;       /* 语音数据的数组指针 */
17 unsigned sp_len; /* 语音数据长度 */
18 unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
19 int segs;        /* 语音分段数 */
20 int gd=DETECT, gm;
21 /*----------------*/
22 main()
23 {
24 int back, i;
25
26  if(!SP_init_sp(0x250)) /* 初始化语音卡 I/O 口 */
27   { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
28
29  /* 动态配置语音数据缓冲区 */
30  sp=(char  *)calloc(LEN ,sizeof(char));
31  if (sp==NULL) { printf("%c out of memory !\n", 7); exit(1);}
32
33  clrscr();
34
35  back=0; /* 清除脱离标志位 */
36 while(1)
```

```
37  {
38   clrscr(); /* 显示操作信息 */
39   puts("SP CARD SPEECH RECOG. LIB. DEMO → mic i/p  sp o/p  ");
40   puts("Any key to start MIC i/p ....");
41   getch(); /* 等待按键 */
42   puts("Please say something ......");
43   puts("Any key to stop MIC i/p ....");
44   be(); /* 哔一声 */
45
46  /* 语音录音，若有按下任何按键则采样终止 */
47   sp_len=SP_mic_ip(sp, LEN, 8000); /* 语音切割为数段 */
48   segs=SP_separatex(sp, sp_len, bp, ep);
49
50  /* 显示语音切割各段的断点 */
51   for(i=0; i<segs; i++)
52    printf("seg %d → %u   %u \n", i, bp[i], ep[i]);
53   getch(); /* 等待按键 */
54  /* 初始化图形接口 */
55   initgraph(&gd, &gm, "");
56   SP_show_wave(sp, 3000, sp_len, 0, 0, getmaxx(), getmaxy(),
57         1.0, bp, ep, segs); /* 显示语音信号波形 */
58  /* 播放出第一段语音内容 */
59   SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
60   delay(1000);  /* 延迟一下 */
61  /* 播放出全部语音内容 */
62   SP_sp_out(sp, 0, sp_len, LEN, 8000);
63  /* 显示操作信息 */
64   outtextxy(10, getmaxy()-100, "Any key to continue ...");
65   outtextxy(10, getmaxy()-120, "<ESC> to exit ! ");
66  /* 等待按键处理 */
67   if(getch()==27) back=1;
68   closegraph(); /* 关闭图形接口 */
69   if(back) break;
70  }
71   free(sp); /* 释放动态配置的语音数据缓冲区 */
72  }
73  /*-----------------*/
74  be() /* 哔一声 */
75  {
76   sound(500);
77   delay(100);
78   nosound();
```

```
}
/*--------------*/
```

8.3.5 ANA.EXE：语音信号切割

原始控制程序为 ANA.C，TURBO C 目标文件名为 ANA.PRJ。

```
/*************************************************
* Program: ANA.C                                 *
* Description: PC SPEECH CARD use SP_LIB example. *
*        Load sp file comp.sp and take feature  *
* Copyright (C)    VICTOR SPEECH LAB.  1996, 1997 *
*************************************************/

#include <alloc.h>
#include <dos.h>
#include <stdio.h>

#include "sp.h" /* 加载声控链接库头文件声明 */

#define   LEN 64000
#define   char signed char

char  *sp;       /* 语音数据的数组指针 */
unsigned sp_len; /* 语音数据长度 */
unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;        /* 语音分段数 */
float  test_frame[MAX_FRAME][LEVEL+1];
/*-----------------*/
main()
{
int i,j;

 if(!SP_init_sp(0x250)) /* 初始化语音卡 I/O 口 */
  { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
 /* 动态配置语音数据缓冲区 */
 sp=(char  *)calloc(LEN ,sizeof(char));
 if (sp==NULL) { printf("%c out of memory !\n", 7); exit(1);}

 clrscr(); /* 显示操作信息 */
 puts("Load SPEECH file COMP.SP....");
 /* 加载测试语音文件 COMP.SP */
 if(!SP_load_voice(sp, LEN, &sp_len, "COMP.SP"))
```

```
  { puts("Cannot find file COMP.SP"); exit(1); }
 /* 播放出该段语音内容 */
 puts("I'm saying ......");
 SP_sp_out(sp, 0, sp_len, LEN, 8000);
 /* 语音切割为一段 */
 segs=SP_separate(sp, sp_len, bp, ep);

/* 语音信号特征参数分析 */
 if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
   { printf("%c cannot take feature ! !\n", 7); exit(1); }
/* 显示特征参数值 */
 puts("Speech Feature list :");

 for(i=0; i<FRAME; i++)
  {
   printf("Frame %2d : ", i);
   for(j=0; j<LEVEL; j++)
    printf("%2.2f ", test_frame[i][j]);
   puts("");
  }

 puts("Any key to continue ........");
 getch();   /* 等待按键 */
 free(sp);  /* 释放动态配置的语音数据缓冲区 */
}
/*-----------------*/
```

8.3.6 MATCH.EXE：语音识别对比

原始控制程序为 MATCH.C，TURBO C 目标文件名为 MATCH.PRJ。

```
/*************************************************
 * Program: MATCH.C                              *
 * Description: PC SPEECH CARD use SP_LIB example.  *
 *            Load 1.sp file and recog. it       *
 * Copyright (C)   VICTOR SPEECH LAB.  1996, 1997  *
 *************************************************/
#include <alloc.h>
#include <dos.h>
#include <stdio.h>
#include "sp.h" /* 加载声控链接库头文件声明 */

#define   LEN 32000
```

```
#define   char signed char

#define   DB_FILE  "DB.DAT" /* 0~29 */
#define   MAX_DB   5

char  *sp;      /* 语音数据的数组指针 */
unsigned sp_len;/* 语音数据长度 */

float  (*DB)[FRAME][LEVEL+1]; /* 参考样本特征参数数组 */
float  test_frame[MAX_FRAME][LEVEL+1]; /* 测试样本特征参数数组 */

unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;  /* 语音分段数 */

int vc_no;
/*----------------------*/
main()
{
 vc_no=MAX_DB; /* 动态配置参考样本缓冲区 */
 DB=farcalloc(MAX_DB, FRAME*(LEVEL+1)*4);
   if(DB==NULL) { printf("%c out of memory !\n", 7); exit(1);}
 /* 动态配置语音数据缓冲区 */
 sp=(char  *)calloc(LEN ,sizeof(char));
 if (sp==NULL) { printf("%c out of memory !\n", 7); exit(1);}
 /* 加载参考样本数据文件 */
 if(SP_load_db(DB, vc_no, DB_FILE))
    puts("load DATA BASE .......");
 else  { printf("%c NO %s  DATA BASE !\n", 7, DB_FILE); exit(1); }
 /* 加载测试语音文件 1.SP */
 if(!SP_load_voice(sp, LEN, &sp_len, "1.SP"))
  { puts("Cannot find file COMP.SP"); exit(1); }
 /* 语音识别处理 */
 match();
 free(sp);   /* 释放动态配置的语音数据缓冲区 */
 farfree(DB);/* 释放动态配置的参考样本缓冲区 */
}
/*------------------------*/
match() /* 语音识别处理 */
{
int ans, frames, i;
float dist[MAX_DB];

```

```
55  clrscr();
56  puts("I'm saying ......");
57  /* 播放出该组语音内容 */
58  SP_sp_out(sp, 0, sp_len, LEN, 8000);
59  puts("Recog.....");
60  segs=SP_separate(sp, sp_len, bp, ep); /* 语音切割 */
61  /* 语音信号特征参数分析 */
62  if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
63    { printf("%c cannot take feature ! !\n", 7); exit(1); }
64  look(); /* 显示特征参数值 */
65  frames=FRAME; /* 语音识别对比 */
66  ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
67  /* 显示识别对比测试结果 */
68  puts("REF distortion list:");
69  for(i=0; i<vc_no; i++)
70   printf("%2.0f ", dist[i]);
71
72  printf("Recog. answer=%d  distotion : %2.0f  \n",
73         ans, dist[ans]);
74  getch(); /* 等待按键 */
75 }
76 /*-------------------------*/
77 look()  /* 显示特征参数值 */
78 {
79 int i,j;
80
81  for(i=0; i<FRAME; i++)
82   {
83    printf("Frame %2d : ", i);
84    for(j=0; j<LEVEL; j++)
85     printf("%2.2f ", test_frame[i][j]);
86    puts("");
87   }
88 }
```

8.4 应用范例程序说明

使用者拿到语音卡声控链接库磁盘时，可以先浏览一下有哪些文件中的范例程序，经过研究应用范例程序，让您很快地了解控制程序的设计。

8.4.1 语音卡声控链接库磁盘内容

SP.LIB：声控链接库
SP.H：声控链接库函数原型声明
5 支演示程序：

1. WAVE.EXE ：观察多段声音波形执行文件。
   WAVE.C ：观察多段声音波形原始程序。
   WAVE.PRJ ：TURBO C 的目标文件。
   DEMO.SP ：演示的声音文件。
2. SR.EXE ：声控演示采样频率执行文件。
   SR.C ：声控演示采样频率原始程序。
   SR.PRJ ：TURBO C 的目标文件。
3. COMP.EXE ：声控演示语音训练执行文件。
   COMP.C ：声控展示语音训练原始程序。
   COMP.PRJ ：TURBO C 的目标文件。
   COMP.DAT ：声控演示的声音参考样本文件。
   PRODD?.SP ：参考样本语音文件。
4. BALLC.EXE ：声控滚球语音训练执行文件。
   BALLC.C ：声控滚球语音训练原始程序。
   BALLC.PRJ ：TURBO C 的目标文件。
   BALL.DAT ：声控滚球的声音参考样本文件。
   PRODB?.SP ：参考样本语音文件。
5. BALL.EXE ：声控滚球执行文件。
   BALL.C ：声控滚球原始程序。
   BALL.PRJ ：TURBO C 的目标文件。
   BALL.DAT ：声控滚球的声音参考样本文件。

8.4.2 范例程序使用

8.4.2.1 程序 1 WAVE.EXE：观察多段声音波形执行文件

程序动作说明：

(1) 读取语音文件 DEMO.SP。
(2) 由计算机说出。
(3) 将语音存盘。
(4) 按下任何键由麦克风输入语音，可以说出多个单音。再按下任何键结束语音输入。
(5) 将语音各个单音断点位置显示出来。
(6) 将语音波形显示出来。
(7) 把语音回放出来。

(8) 若按下“ESC”键结束程序，否则继续由麦克风输入语音。

程序清单如下：

```
/*************************************************
* Program: WAVE.C                                *
* Description: PC SPEECH CARD use SP_LIB example. *
*            Look wave example                   *
* Copyright (C) VICTOR SPEECH LAB.  1996, 1997    *
*************************************************/

#include <graphics.h>
#include <alloc.h>
#include <dos.h>
#include <stdio.h>

#define   LEN 64000
#define   char  signed char

char  *sp;       /* 语音数据的数组指针 */
unsigned sp_len; /* 语音数据长度   */
unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;
int gd=DETECT, gm;
FILE *in, *out;

#include "sp.h" /* 加载声控链接库头文件声明 */
/*----------------*/
main()
{
int back, i;

 /* 初始化语音卡 I/O 口 */
 if(!SP_init_sp(0x250))
  { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
 /* 动态配置语音数据缓冲区 */
 sp=(char  *)calloc(LEN ,sizeof(char));
 if (sp==NULL) { printf("%c out of memory !\n", 7); exit(1);}

 clrscr();
 puts("Test load SPEECH file DEMO.SP....");
 puts("I'm saying ......");
 /* 加载演示的声音文件 */
 if(!SP_load_voice(sp,  LEN, &sp_len, "DEMO.SP"))
```

```
41    { puts("Cannot find file DEMO.SP"); exit(1); }
42   /* 播放出语音内容 */
43   SP_sp_out(sp, 0, sp_len, LEN, 8000);
44   puts("Any key to continue ........\n");
45   getch(); /* 暂停等待按键 */
46   /* 将语音数据存盘 */
47   puts("Test save SPEECH file DEMO1.SP....");
48   puts("I'm saving SPEECH file DEMO1.SP ......");
49   if(!SP_save_voice(sp, sp_len, "DEMO1.SP"))
50    { puts("Cannot save file DEMO1.SP"); exit(1); }
51
52   puts("Any key to continue ........");
53   getch(); /* 暂停等待按键 */
54
55   back=0; /* 清除脱离标志位 */
56  while(1) /* 无穷循环 */
57   {
58    clrscr(); /* 显示工作画面 */
59    puts("SP CARD SPEECH RECOG. LIB. DEMO → Look MIC wave  ");
60    puts("Any key to start MIC i/p ....");
61    getch(); /* 暂停等待按键 */
62    puts("Please say something x words........");
63    puts("Any key to stop MIC i/p ....");
64    be();  /* 哔一声 */
65
66  /* 自动侦测是否有声音输入，若有则开始录音 */
67    sp_len=SP_mic_ip(sp, LEN, 8000);
68  /* 多段语音切割 */
69    segs=SP_separatex(sp, sp_len, bp, ep);
70  /* 显示语音切割的位置*/
71    for(i=0; i<segs; i++)
72     printf("seg %d → %u   %u \n", i, bp[i], ep[i]);
73    getch(); /* 暂停等待按键 */
74  /* 初始化图形接口 */
75    initgraph(&gd, &gm, "");
76    SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
77          0.6, bp, ep, segs);  /* 显示语音信号波形 */
78  /* 播放出第一段语音内容 */
79    SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
80    delay(1000);
81  /* 播放出第一段语音内容 */
82    SP_sp_out(sp, 0, sp_len, LEN, 8000);
```

```
83  /* 显示操作信息 */
84    outtextxy(10, getmaxy()-100, "Any key to continue ...");
85    outtextxy(10, getmaxy()-120, "<ESC> to exit ! ");
86    if(getch()==27) back=1;
87    closegraph();  /* 关闭图形接口 */
88    if(back) break;
89  }
90    free(sp); /* 释放动态配置的语音数据缓冲区 */
91  }
92  /*------------------*/
93  be() /* 哔一声 */
94  {
95    sound(500);
96    delay(100);
97    nosound();
98  }
99  /*----------*/
```

8.4.2.2 程序 2 SR.EXE: 声控演示采样频率执行文件

程序动作说明:

(1) 录音的采样频率为每秒 8000 个点的语音数据。

(2) 按下任何键由麦克风输入语音，再按下任何键结束语音输入。

(3) 按下以下的功能键可以做不同放音速度的控制:

1 → 8kHz　normal speed

2 → 9kHz

3 → 10kHz

4 → 11kHz

5 → 12kHz

6 → 15kHz

7 → 20kHz

8 → 25kHz

9 → 30kHz

q → 7kHz

w → 6kHz

e → 5kHz

r → 4kHz

t → 3kHz

SPACE → say again

(4) 按下空格键可以再次输入语音。

程序清单如下:

```
/************************************************
* Program: SR.C                                 *
* Description:PC SPEECH CARD use SP_LIB example.  *
*           Sample rate demo.                    *
* Copyright (C) VICTOR SPEECH LAB.  1996, 1997    *
************************************************/

#include <graphics.h>
#include <alloc.h>
#include <dos.h>
#include <stdio.h>

#define   LEN 64000
#define   char signed char

char  *sp;  /* 语音数据的数组指针 */
unsigned sp_len;    /* 语音数据长度 */
unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;       /* 语音分段数 */
int gd=DETECT, gm;
FILE *in, *out;

#include "sp.h" /* 加载声控链接库头文件声明 */
/*-----------------*/
main()
{
char c;
int freq;
 /* 初始化语音卡 I/O 口 */
 if(!SP_init_sp(0x250))
  { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
 /* 动态配置语音数据缓冲区 */
 sp=(char  *)calloc(LEN ,sizeof(char));
 if (sp==NULL)   { printf("%c out of memory !\n", 7); exit(1);}

while(1) /* 无穷循环 */
 {
  clrscr();
  puts("SPEECH CARD VOICE RECORD/PLAY test  ");
  puts("Any key to start MIC i/p   <ESC> back....");
  if(getch()==27) break; /* 若按下 ESC 键则跳离循环 */

```

```
43   puts("Please say 1 word.........");
44   puts("Any key to stop MIC i/p ....");
45   be(); /* 哔一声 */
46
47  /* 语音录音，若有按下任何按键则采样终止 */
48   sp_len=SP_mic_ip(sp, LEN, 8000);
49   segs=SP_separate(sp, sp_len, bp, ep); /* 语音切割 */
50   SP_sp_out(sp, bp[0], ep[0], LEN, 8000); /* 播放出该段语音内容 */
51   menu(); /* 显示工作画面 */
52
53  wait:
54   c=getch();/* 等待按键 */
55   if(c==' ') continue; /*重新播放一遍 */
56   switch(c)
57    {  /* 设定语音输出控制频率 */
58     case '1' : freq= 8000;  break;
59     case '2' : freq= 9000;  break;
60     case '3' : freq=10000;  break;
61     case '4' : freq=11000;  break;
62     case '5' : freq=12000;  break;
63     case '6' : freq=15000;  break;
64     case '7' : freq=20000;  break;
65     case '8' : freq=25000;  break;
66     case '9' : freq=30000;  break;
67
68     case 'q' : freq= 7000;  break;
69     case 'w' : freq= 6000;  break;
70     case 'e' : freq= 5000;  break;
71     case 'r' : freq= 4000;  break;
72     case 't' : freq= 3000;  break;
73
74     default  : break;
75    }
76   gotoxy(5, 20);
77   printf("Play speed : %5d ", freq);
78   SP_sp_out(sp, bp[0], ep[0], LEN, freq); /* 播放出该段语音内容 */
79   goto wait;
80  }
81   free(sp); /* 释放动态配置的语音数据缓冲区 */
82  }
83  /*-----------------*/
84  be() /* 哔一声 */
```

```
{
  sound(500);
  delay(100);
  nosound();
}
/*--------------*/
menu()  /* 显示工作画面 */
{
   clrscr();
   puts("SPEECH CARD talk test ........");
   puts("1 → 8K HZ normal speed ");
   puts("2 → 9K HZ  ");
   puts("3 → 10K HZ  ");
   puts("4 → 11K HZ  ");
   puts("5 → 12K HZ  ");
   puts("6 → 15K HZ  ");
   puts("7 → 20K HZ  ");
   puts("8 → 25K HZ  ");
   puts("9 → 30K HZ  ");

   puts("q → 7K HZ  ");
   puts("w → 6K HZ  ");
   puts("e → 5K HZ  ");
   puts("r → 4K HZ  ");
   puts("t → 3K HZ  ");
   puts("SPACE → say again ");
}
```

8.4.2.3 程序 3 COMP.EXE：声控演示语音训练执行文件

(1) 本声控展示的语音内容有 5 个词：

当显示结果为 COMPUTER1 时，播放声音为“计算机”。当显示结果为 COMPUTER2 时，播放声音为“COMPUTER”。当显示结果为 COMPUTER3 时，播放日文发音“COMPUTER”。当显示结果为 COMPUTER4 时，播放台语发音“计算机”。当显示结果为 JOHN 时，播放声音为“JOHN”。前 4 个音都代表的是“计算机”。

(2) 执行后画面出现如下内容：

SP_LIB demo COMPUTER

c → create DATA BASE

l → listen DATA BASE

m → modify DATA BASE

SPACE SPEECH test

Select ?

(3) 按“c”键可以产生语音样本。

(4) 按“l”键可以聆听语音参考样本内容。

(5) 按“m”键可以做语音样本修改。

(6) 按空格键则开始识别。

语音样本产生

- ✧ 依序输入语音。
- ✧ 所输入的语音会存为参考样本语音文件 PRODDX.SP(X=0~4)。
- ✧ 语音输入时使用者不必按任何键，系统会自动进行采样。
- ✧ 计算机显示出切割后的波形并进行测试识别。类似结果如下：

Test recog : COMPUTER1　　dist.=61.2　　(Normal < 80.0)

表示刚输入的语音与原来的参考样本进行测试识别，识别结果为“COMPUTER1”，失真度(dist)为 61.2，失真度越低表示输入的语音越像识别结果，但有时也有可能是混淆音，通常失真度小于 80.0 表示可以接受，如此可以作为训练时的一项参考，看看是否要重新输入语音或是接受此一组语音样本。

- ✧ 使用者判断是否接受，若是按“y”键或“ENTER”键表示接受此语音，否则按“ESC”键放弃，则要重新输入语音。一般若语音切割结果太离谱(切割位置错误)则放弃，得重新输入语音样本。

语音样本修改

- ✧ 输入某一样本编号。
- ✧ 系统说出语音文件的内容。
- ✧ 输入语音。
- ✧ 计算机显示出切割后的波形并进行测试识别。类似结果如上说明。
- ✧ 使用者判断是否接受，若是按“y”键表示接受此语音，按“ENTER”键重新输入语音，则按“ESC”键放弃。一般若语音切割结果太离谱(切割位置错误)则放弃，得重新输入语音样本。

语音识别

- ✧ 输入语音。
- ✧ 计算机将语音波形进行分析。
- ✧ 计算机做语音识别。
- ✧ 将识别结果显示出来，例如说“计算机”则经识别后类似的结果显示如下：

Recog.....

COMPUTER1 > 59　　94　　92　　61　　121

表示识别结果正确，并把失真度(dist)显示出来为 59，同时也将其他的各个参考失真度显示出来，其中失真度最低的一组为识别结果，由此看来第 1 组失真度(dist=59)与第 4 组(dist=61)很接近，也表示这两组会造成混淆音的机会很大，所幸识别结果都是代表计算机。将各个参考失真度显示出来有助于我们对识别结果的分析。若失真度大 90 则显示“NOT”，表示所说语音与参考样本中的内容不相干，拒绝识别，例如我们咳嗽一声，则会显示“NOT”，拒绝识别。

注意

此一失真度临界值(90)为一实验值，如果设的太小则某些识别正确的音会被拒绝识别。反之设的太大，只要咳嗽一声或是说一些不相干的语音，系统也可能会接受它，而造成错误识别。因此必须经过几次的实验才能真正确定其值。

程序清单如下：

```
/*****************************************************
* Program: COMP.C                                    *
* Description: PC SPEECH CARD use SP_LIB example.  *
*            "COMPUTER" word DEMO.                   *
* Copyright (C) VICTOR SPEECH LAB.  1996, 1997     *
*****************************************************/

/* 加载包括函数文件 */
#include <graphics.h>
#include <alloc.h>
#include <dos.h>
#include <stdio.h>
#include "sp.h" /* 加载声控链接库头文件声明 */

#define   LEN 32000
#define   char  signed char

#define   DB_FILE  "COMP.DAT"
#define   NO      300
#define   MAX_DB   5

char  *sp, *sppro;  /* 语音数据的数组指针 */
unsigned sp_len, pro_len; /* 语音数据长度 */

float  (*DB)[FRAME][LEVEL+1];  /* 参考样本特征参数数组 */
float  test_frame[MAX_FRAME][LEVEL+1]; /* 测试样本特征参数数组 */

unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;            /* 语音分段数 */
int gd=DETECT, gm;

char fname[80];

FILE *in, *out;
int vc_no=5;

char string[80];
```

```
38  char *vc_note[]={"COMPUTER1", "COMPUTER2","COMPUTER3",COMPUTER4",
39                   "JOHN     "}; /* 所要识别的单音内容 */
40  /*-----------------*/
41  main()
42  {
43  char ch;
44
45   if(!SP_init_sp(0x250)) /* 初始化语音卡 I/O 口 */
46    { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
47
48  DB=farcalloc(MAX_DB, FRAME*(LEVEL+1)*4);/* 动态配置参考样本缓冲区*/
49     if(DB==NULL) { printf("%c out of memory !\n", 7); exit(1);}
50
51   sp=(char  *)calloc(LEN ,sizeof(char)); /* 动态配置语音数据缓冲区 */
52   if (sp==NULL)  { printf("%c out of memory !\n", 7); exit(1);}
53
54 sppro=(char  *)calloc(LEN ,sizeof(char)); /* 动态配置语音提示语缓冲区 */
55   if (sppro==NULL)  { printf("%c out of memory !\n", 7); exit(1);}
56
57   /* 加载参考样本数据文件 */
58   if(SP_load_db(DB, vc_no, DB_FILE))
59     puts("load DATA BASE .......");
60   else  { printf("%c NO %s  DATA BASE !\n", 7, DB_FILE); getch(); }
61
62  while(1) /* 无穷循环 */
63   {
64    clrscr(); /* 显示工作画面 */
65    puts("SP_LIB demo COMPUTER   .......");
66    puts(" c   → create DATA BASE");
67    puts(" l   → listen DATA BASE");
68    puts(" m   → modify DATA BASE");
69
70    puts(" SPACE   SPEECH test ");
71    puts("Select ? ");
72
73    ch=getch(); /* 等待按键 */
74    switch(ch)
75      {
76        case 27 : exit(0); /* 若按下 ESC 键则结束程序执行 */
77        case 'c': create_db(); break; /* 产生并录制参考样本数据文件 */
79        case 'l': prompt();    break; /* 听取所要识别的语音数据内容 */
80        case 'm': modify_db(); break; /* 修改参考样本语音数据文件 */
```

```
81          case ' ': comp();      break; /* 语音识别 */
82          default : break;
83        }
84     }
85      free(sp);      /* 释放动态配置的语音数据缓冲区 */
86      farfree(DB); /* 释放动态配置的参考样本缓冲区 */
87   }
88   /*------------------------------*/
89   modify_db()  /* 修改参考样本语音数据文件 */
90   {
91   int i, exit_f,no, ans, frames;
92   char c, mess[80];
93   float dist[10];
94   /* 显示将要识别的语音内容 */
95     puts("\n0 → COMP1");
96     puts("1 → COMP2");
97     puts("2 → COMP3");
98     puts("3 → COMP4");
99     puts("4 → JOHN ");
100
101  /* 输入修改参考样本编号 */
102    puts("=== modify word no ===");
103    do{printf("modify which no ");
104    scanf("%d", &no);
105    } while( no>=vc_no); /* 若编号无效则重新输入编号 */
106
107  again:
108  /* 说出所要修改的该组语音内容 */
109    printf("please say <  %s >\n", vc_note[no]);
110    sprintf(fname, "PRODD%d.SP",no);
111    if(!SP_load_voice(sppro,  LEN, &pro_len, fname))
112     { puts("Cannot find prompt file"); exit(1); }
113    SP_sp_out(sppro, 0, pro_len, LEN, 8000);
114
115  /* 取得语音数据..... */
116      be();       /* 哔一声 */
117      exit_f=0;   /* 清除脱离标志位 */
118     /* 语音输入自动侦测并检查是否有键按下*/
119      sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
120  /* 语音切割 */
121     segs=SP_separate(sp, sp_len, bp, ep);
122     if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
```

```
/* 初始化图形接口 */
   initgraph(&gd, &gm, "");
   SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
         0.6, bp, ep, segs); /* 显示语音信号波形 */
   SP_sp_out(sp, bp[0], ep[0], LEN, 8000); /* 播放出该段语音内容 */

/* 语音信号特征参数分析 */
   if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
    { printf("%c cannot take feature ! !\n", 7); exit(1); }

/* 语音识别对比测试 */
   frames=16;
   ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
/* 显示识别对比测试结果 */
   sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
         vc_note[ans], dist[ans]);
   outtextxy(10, getmaxy()-80, mess);
/* 显示操作讯息 */
   outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
   outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");

/* 等待按键处理 */
   c=getch();

   if(c==0x1b)  {  closegraph(); return; }
   if(c==0x0d)  {  closegraph(); goto next; }
   if(c!='y')   {  closegraph(); goto again; }
     else closegraph();

next:;
/* 语音信号特征参数分析 */
   if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
    { printf("%c cannot take feature ! !\n", 7); exit(1); }

/* 存为语音参考样本文件 */
   puts("Saving data base ......");
   if( SP_save_db(DB, vc_no, DB_FILE) )
{ printf("Modify data base  <<<%d  %s  >>> ok ! ", no, vc_note[no]);
     delay(2000); }
   else { printf("%c cannot save data base !\n", 7); exit(1); }
}
/*--------------------*/
```

```
comp() /* 语音识别 */
{
int ans, i;
int exit_f, frames;
float dist[10];

 puts("Recog.....");
while(1) /* 无穷循环 */
 {
again:
  be(); /* 哔一声 */
  exit_f=0;  /* 清除脱离标志位 */
/* 语音输入自动侦测并检查是否有键按下*/
  sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
  if(exit_f) break; /* 若有按键则跳离循环 */
/* 语音切割 */
  printf(".");
  segs=SP_separate(sp, sp_len, bp, ep);
  if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
/* 语音信号特征参数分析 */
  if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
     { printf("%c cannot take feature ! !\n", 7); exit(1); }
/* 语音识别对比 */
 frames=16;
 ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
 if(dist[ans]>=90.0) printf("NOT"); /* 失真度大于临界值则告知无法识别 */
   else
    { /* 显示识别结果 */
     printf(" %s ", vc_note[ans]);
/* 播放出识别结果 */
     sprintf(fname, "PRODD%d.SP",ans);
     if(!SP_load_voice(sppro,  LEN, &pro_len, fname))
      { puts("Cannot find prompt file"); exit(1); }
     SP_sp_out(sppro, 0, pro_len, LEN, 8000);
    }

/* 列出语音识别的失真度 */
  for(i=0; i<vc_no; i++)
   {
    if(ans==i) printf(">");
    printf("%3.0f  ",dist[i]);
   }
```

```
207   puts("");
208  }/* end while */
209 }
210 /*--------------------*/
211 be()  /* 哔一声 */
212 {
213   sound(500);
214   delay(100);
215   nosound();
216 }
217 /*---------------*/
218 create_db()  /* 产生并录制参考样本数据文件 */
219 {
220 int i, exit_f,no;
221 char c;
222 int ans, frames;
223 float dist[10];
224 char mess[80];
225
226   clrscr(); /* 显示将要识别的语音内容 */
227   puts("Create 1st data base ");
228   puts("0 → COMP1");
229   puts("1 → COMP2");
230   puts("2 → COMP3");
231   puts("3 → COMP4");
232   puts("4 → JOHN  ");
233
234  for(no=0; no<vc_no; no++)
235   {
236 again:
237    printf("please say <  %s >\n", vc_note[no]);
238 /* 取得语音数据..... */
239    be();      /* 哔一声 */
240    exit_f=0;  /* 清除脱离标志位 */
241    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
242 /* 语音输入自动侦测并检查是否有按键 */
243 /* 语音切割 */
244    segs=SP_separate(sp, sp_len, bp, ep);
245    if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
246 /* 初始化图形接口 */
247    initgraph(&gd, &gm, "");
248    SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
```

```
249          0.6, bp, ep, segs); /* 显示语音信号波形 */
250 /* 播放出该段语音内容 */
251   SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
252 /* 语音信号特征参数分析 */
253   if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
254    { printf("%c cannot take feature ! !\n", 7); exit(1); }
255
256 /* 语音识别对比测试 */
257   frames=16;
258   ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
259 /* 显示识别对比测试结果 */
260   sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
261          vc_note[ans], dist[ans]);
262   outtextxy(10, getmaxy()-80, mess);
263 /* 显示操作信息 */
264   outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
265   outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
266
267 /* 等待按键处理 */
268   c=getch();
269   if(c==0x1b)  {  closegraph(); return; }
270   if(c==0x0d)  {  closegraph(); goto next; }
271   if(c!='y')   {  closegraph(); goto again; }
272     else closegraph();
273 next:;
274 /* 原始语音数据存盘 */
275   sprintf(fname, "PRODD%d.SP",no);
276
277   if(!SP_save_voice1(sp, bp[0], ep[0], fname))
278    { puts("Cannot save prompt file "); exit(1); }
279
280 /* 语音信号特征参数分析 */
281   if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
282    { printf("%c cannot take feature ! !\n", 7); exit(1); }
283 }/* end of no */
284
285 /* 存为语音参考样本文件 */
286   puts("Saving data base ......");
287   if( SP_save_db(DB, vc_no, DB_FILE) )
288     { puts("Save data base  OK !");   delay(2000); }
289   else { printf("%c cannot save data base !\n", 7); exit(1); }
290
```

```
291   prompt(); /* 播放出所有识别的语音数据内容 */
292  }
293  /*--------------------*/
294  prompt()  /* 听取所要识别的语音数据内容 */
295  {
296  int no;
297
298   puts("Say data base .....");
299   for(no=0; no<vc_no; no++)
300    { /* 播放出某一参考样本语音 */
301     sprintf(fname, "PRODD%d.SP",no);
302     if(!SP_load_voice(sppro,  LEN, &pro_len, fname))
303      { puts("Cannot find prompt file"); exit(1); }
304     SP_sp_out(sppro, 0, pro_len, LEN, 8000);
305     delay(5000);
306    }
307  }
308  /*--------------------*/
```

8.4.2.4 程序 4 BALLC.EXE: 声控滚球语音训练执行文件

整个声控滚球语音训练操作与 COMP.EXE 一样，所不一样的是语音文件内容不一样，如下所示：

表 8.1 声控滚球语音操作训练和内容

显示结果	语音内容	参考样本语音文件名
UP	上	PRODB0.SP
DOWN	下	PRODB1.SP
LEFT	左	PRODB2.SP
RIGH T	右	PRODB3.SP

也就是说我们说出此四个方向的音，由计算机进行识别。

程序清单如下：

```
1   /*********************************************************
2   * Program: BALLC.C                                      *
3   * Description: PC SPEECH CARD use SP_LIB example.       *
4   *            Voice control ball data base editor       *
5   * Copyright (C)    VICTOR SPEECH LAB.  1996, 1997       *
6   *********************************************************/
7
8    #include <graphics.h>
9    #include <alloc.h>
10   #include <dos.h>
```

```
#include <stdio.h>
#include "sp.h"   /* 加载声控链接库头文件声明 */

#define   LEN 8000
#define   char    signed char

#define   DB_FILE  "BALL.DAT"
#define   NO         300
#define   MAX_DB   5

char  *sp, *sppro;  /* 语音数据的数组指针 */
unsigned sp_len, pro_len; /* 语音数据长度 */

float  (*DB)[FRAME][LEVEL+1]; /* 参考样本特征参数数组 */
float  test_frame[MAX_FRAME][LEVEL+1];  /* 测试样本特征参数数组 */

unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;               /* 语音分段数 */
int gd=DETECT, gm;

char fname[80];

FILE *in, *out;
int vc_no=4;    /* 所要识别的单音数 */

char string[80];  /* 所要识别的单音 */
char *vc_note[]={"UP     ", "DOWN    ", "LEFT    ", "RIGHT   "};
/*-----------------*/
main()
{
char ch;

 if(!SP_init_sp(0x250))  /* 初始化语音卡 I/O 口 */
  { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
 /* 动态配置参考样本缓冲区 */
 DB=farcalloc(MAX_DB, FRAME*(LEVEL+1)*4);
   if(DB==NULL) { printf("%c out of memory !\n", 7); exit(1);}
 /* 动态配置语音数据缓冲区 */
 sp=(char  *)calloc(LEN ,sizeof(char));
 if (sp==NULL) { printf("%c out of memory !\n", 7); exit(1);}
 /* 动态配置语音提示语缓冲区 */
 sppro=(char  *)calloc(LEN ,sizeof(char));
```

```
53   if (sppro==NULL)  { printf("%c out of memory !\n", 7); exit(1);}
54
55   /* 加载参考样本数据文件 */
56   if(SP_load_db(DB, vc_no, DB_FILE))
57     puts("load DATA BASE .......");
58   else  { printf("%c no %s  DATA BASE !\n", 7, DB_FILE); getch(); }
59
60  while(1) /* 无穷循环 */
61   {
62    clrscr(); /* 显示工作画面 */
63    puts("SP_LIB demo rolling ball .......");
64    puts(" c   → create DATA BASE");
65    puts(" l   → listen DATA BASE");
66    puts(" m   → modify DATA BASE");
67
68    puts(" SPACE   SPEECH test ");
69    puts("Select ? ");
70
71    ch=getch(); /* 等待按键 */
72    switch(ch)
73      {
74        case 27 : exit(0);  /* 若按下 ESC 键则结束程序执行 */
75        case 'c': create_db(); break; /* 产生并录制参考样本数据文件 */
76
77        case 'l': prompt();    break; /* 听取所要识别的语音数据内容 */
78        case 'm': modify_db(); break; /* 修改参考样本语音数据文件 */
79        case ' ': ball();      break; /* 语音识别 */
80        default : break;
81      }
82    }
83     free(sp);     /* 释放动态配置的语音数据缓冲区 */
84     farfree(DB);  /* 释放动态配置的参考样本缓冲区 */
85  }
86  /*------------------------------*/
87  modify_db() /* 修改参考样本语音数据文件  */
88  {
89  int i, exit_f,no, frames, ans;
90  char c, mess[80];
91  float dist[10];
92  /* 显示欲识别的语音内容 */
93    puts("\n0 → up     ");
94    puts("1 → down    ");
```

```
95    puts("2 → left");
96    puts("3 → right");
97
98  /* 输入修改参考样本编号 */
99    puts("=== modify word no ===");
100
101   do{printf("modify which no ");
102   scanf("%d", &no);
103   } while( no>=vc_no); /* 若编号无效则重新输入编号 */
104
105 again:
106 /* 播放出所要修改的该组语音内容 */
107   printf("please say <  %s >\n", vc_note[no]);
108   sprintf(fname, "PRODB%d.SP",no);
109   if(!SP_load_voice(sppro,  LEN, &pro_len, fname))
110    { puts("Cannot find prompt file"); exit(1); }
111   SP_sp_out(sppro, 0, pro_len, LEN, 8000);
112
113 /* 取得语音数据..... */
114    be();      /* 哔一声 */
115    exit_f=0;  /* 清除脱离标志位 */
116 /* 语音输入自动侦测并检查是否有按键 */
117    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
118 /* 语音切割 */
119   segs=SP_separate(sp, sp_len, bp, ep);
120   if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
121 /* 初始化图形接口 */
122   initgraph(&gd, &gm, "");
123   SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
124         0.6, bp, ep, segs); /* 显示语音信号波形 */
125   SP_sp_out(sp, bp[0], ep[0], LEN, 8000); /* 播放出该段语音内容 */
126
127 /* 语音信号特征参数分析 */
128   if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
129    { printf("%c cannot take feature ! !\n", 7); exit(1); }
130
131 /* 语音识别对比测试 */
132   frames=16;
133   ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
134 /* 显示识别对比测试结果 */
135   sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
136         vc_note[ans], dist[ans]);
```

```
137    outtextxy(10, getmaxy()-80, mess);
138  /* 显示操作信息 */
139    outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
140    outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
141
142  /* 等待按键处理 */
143    c=getch();
144
145    if(c==0x1b)  {  closegraph(); return; }
146    if(c==0x0d)  {  closegraph(); goto next; }
147    if(c!='y')   {  closegraph(); goto again; }
148      else closegraph();
149
150  next:;
151  /* 语音信号特征参数分析 */
152    if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
153     { printf("%c cannot take feature ! !\n", 7); exit(1); }
154
155  /* 存为语音参考样本文件 */
156    puts("Saving data base ......");
157    if( SP_save_db(DB, vc_no, DB_FILE) )
158    { printf("Modify data base  <<<%d  %s  >>> ok ! ", no, vc_note[no]);
159      delay(2000); }
160    else { printf("%c cannot save data base !\n", 7); exit(1); }
161  }
162  /*-------------------*/
163  ball()  /* 语音识别 */
164  {
165  int ans, i;
166  int exit_f, frames;
167  float dist[10];
168
169   puts("Recog.....");
170  while(1) /* 无穷循环 */
171   {
172  again:
173    be();    /* 哔一声 */
174    exit_f=0;/* 清除脱离标志位 */
175  /* 语音输入自动侦测并检查是否有按键 */
176    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
177    if(exit_f) break;  /* 若有按键则跳离循环 */
178
```

```
  printf(".");
  segs=SP_separate(sp, sp_len, bp, ep); /* 语音切割 */
  if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
/* 语音信号特征参数分析 */
  if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
     { printf("%c cannot take feature ! !\n", 7);
       exit(1);
     }
  frames=16;
/* 语音识别对比 */
  ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
 if(dist[ans]>=90.0) printf("NOT ");/* 失真度大于临界值则告知无法识别 */
   else
    {  /* 显示识别结果 */
     printf(" %s ", vc_note[ans]);
/* prompt */
     sprintf(fname, "PRODB%d.SP",ans);
     if(!SP_load_voice(sppro,  LEN, &pro_len, fname))
      { puts("Cannot find prompt file"); exit(1); }
     SP_sp_out(sppro, 0, pro_len, LEN, 8000);
    } /* 播放出该组语音内容 */

/* 列出语音识别的失真度 */
  for(i=0; i<vc_no; i++)
   {
    if(ans==i) printf(">");
    printf("%3.0f   ",dist[i]);
   }
  puts("");
 }/* end while */
}
/*--------------------*/
be()  /* 哔一声 */
{
  sound(500);
  delay(100);
  nosound();
}
/*---------------*/
create_db()  /* 产生并录制参考样本数据文件 */
{
int i, exit_f,no;
```

```
221 char c;
222 int ans, frames;
223 float dist[10];
224 char mess[80];
225
226   clrscr();/* 显示将要识别的语音内容 */
227   puts("create 1st data base ");
228   puts("0 → up     ");
229   puts("1 → down    ");
230   puts("2 → left");
231   puts("3 → right");
232
233  for(no=0; no<vc_no; no++)
234   {
235 again:
236    printf("please say <  %s >\n", vc_note[no]);
237 /* 取得语音数据..... */
238    be();     /* 哔一声 */
239    exit_f=0; /* 清除脱离标志位 */
240 /* 语音输入自动侦测并检查是否有按键 */
241    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
242    if(exit_f) return; /* 若有按键则跳离循环 */
243 /* 语音切割 */
244    segs=SP_separate(sp, sp_len, bp, ep);
245    if(segs==0) goto again;/* 若无有效语音分段则重新做语音输入 */
246 /* 初始化图形接口 */
247    initgraph(&gd, &gm, "");
248    SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
249           0.6, bp, ep, segs); /* 显示语音信号波形 */
250 /* 播放出该段语音内容 */
251    SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
252 /* 语音信号特征参数分析 */
253    if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
254     { printf("%c cannot take feature ! !\n", 7); exit(1); }
255
256 /* 语音识别对比测试 */
257    frames=16;
258    ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
259 /* 显示识别对比测试结果 */
260    sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
261           vc_note[ans], dist[ans]);
262    outtextxy(10, getmaxy()-80, mess);
```

```
263 /* 显示操作信息 */
264    outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
265    outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
266
267 /* 等待按键处理 */
268    c=getch();
269    if(c==0x1b)  {  closegraph(); return; }
270    if(c==0x0d)  {  closegraph(); goto next; }
271    if(c!='y')   {  closegraph(); goto again; }
272       else closegraph();
273 next:;
274 /* 原始语音数据存盘 */
275    sprintf(fname, "PRODB%d.SP",no);
276
277    if(!SP_save_voice1(sp, bp[0], ep[0], fname))
278     { puts("Cannot save prompt file "); exit(1); }
279
280 /* 语音信号特征参数分析 */
281    if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
282     { printf("%c cannot take feature ! !\n", 7); exit(1); }
283  }/* end of no */
284
285 /* 存为语音参考样本文件 */
286    puts("Saving data base ......");
287    if( SP_save_db(DB, vc_no, DB_FILE) )
288      { puts("Save data base  OK !");   delay(2000); }
289    else { printf("%c cannot save data base !\n", 7); exit(1); }
290 /* 说出所有识别的语音数据内容 */
291  prompt();
292 }
293 /*--------------------*/
294 prompt() /* 听取所要识别的语音数据内容 */
295 {
296 int no;
297
298  puts("Say data base .....");
299  for(no=0; no<vc_no; no++)
300   {
301    sprintf(fname, "PRODB%d.SP",no);
302    if(!SP_load_voice(sppro,  LEN, &pro_len, fname))
303     { puts("Cannot find prompt file"); exit(1); }
304    SP_sp_out(sppro, 0, pro_len, LEN, 8000);
```

```
305     delay(5000);
306    }
307  }
308 /*-------------------*/
```

8.4.2.5 程序 5 BALL.EXE：声控滚球执行文件

✧ 程序执行后，只要对麦克风说出“上”、“下”、“左”、“右”，各个单音，滚球即会做出相应的动作。

✧ 语音样本的产生及修改使用 BALLC.EXE。

程序清单如下：

```
1   /*****************************************************
2   * Program: BALL.C                                   *
3   * Description: PC SPEECH CARD use SP_LIB example.    *
4   *            Voice control ball                     *
5   * Copyright (C)  VICTOR SPEECH LAB.  1996, 1997     *
6   *****************************************************/
7
8   #include <graphics.h>
9   #include <alloc.h>
10  #include <dos.h>
11  #include <stdio.h>
12  #include "sp.h"   /* 加载声控链接库头文件声明 */
13
14  #define   LEN 8000
15  #define   char  signed char
16
17  #define   DB_FILE  "BALL.DAT"
18  #define   NO       300
19  #define   MAX_DB    5
20
21  char  *sp;        /* 语音数据的数组指针 */
22  unsigned sp_len;  /* 语音数据长度 */
23  float  (*DB)[FRAME][LEVEL+1]; /* 参考样本特征参数数组 */
24  float  test_frame[MAX_FRAME][LEVEL+1]; /* 测试样本特征参数数组 */
25
26  unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
27  int segs;               /* 语音分段数 */
28  int gd=DETECT, gm;
29  int vc_no=4;    /* 所要识别的单音数 */
30  char *buffer;
31  int bx, by;
32  int maxx, maxy;
```

```
/*----------------*/
main()
{
char ch;
/* 初始化语音卡 I/O 口 */
  if(!SP_init_sp(0x250))
    { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }
/* 动态配置参考样本缓冲区 */
 DB=farcalloc(MAX_DB, FRAME*(LEVEL+1)*4);
   if(DB==NULL) { printf("%c out of memory !\n", 7); exit(1);}
/* 动态配置语音数据缓冲区 */
 sp=(char  *)calloc(LEN ,sizeof(char));
 if (sp==NULL)  { printf("%c out of memory !\n", 7); exit(1);}
/* 加载参考样本数据文件 */
 if(SP_load_db(DB, vc_no, DB_FILE))
    puts("load DATA BASE .......");
 else  { printf("%c no %s  DATA BASE !\n", 7, DB_FILE); getch(); }

  ball();       /* 声控滚球语音识别 */
  free(sp);     /* 释放动态配置的语音数据缓冲区 */
  farfree(DB);  /* 释放动态配置的参考样本缓冲区 */
}
/*-------------------------------*/
create_ball() /* 产生声控滚球影像 */
{
  int size;

  setcolor(1);
  setfillstyle(SOLID_FILL,1);
  circle(100,100, 15);
  floodfill(100,100, 1);

  size=imagesize(83, 83, 117, 117);
  buffer=malloc(size);
  getimage(83, 83, 117, 117, buffer);
  putimage(83, 83, buffer, XOR_PUT);
}
/*---------------*/
ball()   /* 声控滚球语音识别 */
{
int ans;
char mess[40];
```

```
int exit_f;
float dist[10];
 /* 初始化图形接口 */
   initgraph(&gd, &gm, "");
   maxx=getmaxx(); maxy=getmaxy();
   create_ball();

   bx=maxx/2; by=maxy/2;
   create_ball();  /* 产生声控滚球影像 */
   putimage(bx, by, buffer, XOR_PUT);

   setcolor(WHITE);
   outtextxy(10, getmaxy()-10, " <ESC> to exit !");
   outtextxy(10, getmaxy()-20, " VOICE CONTROL ROLLING BALL....");

while(1) /* 无穷循环 */
 {
again:
  be();      /* 哔一声 */
  exit_f=0;  /* 清除脱离标志位 */
  sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
/* 语音输入自动侦测并检查是否有按键 */
  if(exit_f) break; /* 若有按键则跳离循环 */
  segs=SP_separate(sp, sp_len, bp, ep); /* 语音切割 */
  if(segs==0) goto again;/* 若无有效语音分段则重新做语音输入 */
/* 语音信号特征参数分析 */
  if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
     { printf("%c cannot take feature ! !\n", 7); exit(1); }
/* 语音识别对比 */
  ans=SP_sp_match1(DB, vc_no, test_frame, 16, dist);
  if(dist[ans]>=120.0) continue; /* 失真度大于临界值则告知无法识别 */
/* 执行动作 */
  action(ans);
  if(kbhit()) break;
 }/* end of while */

back:  closegraph(); /* 关闭图形接口 */
}
/*-----------------*/
action(int dir) /* 执行相对动作 */
{
int step=50,x;
```

```
117
118 /* 擦去旧的影像 */
119     putimage(bx, by, buffer, XOR_PUT);
120
121 /* 决定球的方向 */
122     switch(dir)
123        {
124         case 0 : by-=step; break;
125         case 1 : by+=step; break;
126         case 2 : bx-=step; break;
127         case 3 : bx+=step; break;
128         default: break;
129        }
130 /* 检查球是否超出屏幕的范围 */
131    if(bx<30) bx=30;
132    if(bx >(maxx-60)) bx=maxx-60;
133    if(by<30) by=30;
134    if(by >(maxy-30)) by=maxy-30;
135 /* 发出音效 */
136    for(x=4000; x>500; x-=500)
137     { sound(x); delay(30); nosound(); }
138
139 /* 擦去新的影像 */
140    putimage(bx, by, buffer, XOR_PUT);
141 }
142 /*--------------------*/
143 be()  /*哔一声 */
144 {
145   sound(500);
146   delay(100);
147   nosound();
148 }
149 /*---------------*/
```

8.5 如何提高识别率

本语音卡声控链接库可以让使用者，以C语言来自己设计属于自己的声控程序，适当地操作本控制系统可以使整体识别率达到90% 以上。至于如何提高辨识率，以下有一些建议提供给读者作参考。

✧ 尽量避免使用容易混淆的音当作识别的字词，如中文数“1”和“7”。

✧ 同一辨识对象使用多组参考样本。例如：说“美国”，“America”，“USA”均识别

为美国。

- ✧ 若为多人使用的系统请各自加载每人独立的语音参考样本进行识别。
- ✧ 不限使用语言，讲方言、国语、台语、英语皆可。
- ✧ 语音输入品质十分重要，太大声、太小声、背景杂音太吵皆不宜。
- ✧ 由于语音输入的麦克风是使用动圈式麦克风，为指向式麦克风，因此请对着麦克风，以固定的距离说话。
- ✧ 加入语音提示语可以改进人机接口操作的方便。

第9章 声控指令

以上各章我们已经将语音卡硬件、控制软件及声控链接库作了详细的说明，由本章起将利用声控链接库来设计几个简单实用又有趣的声控应用程序，使读者可以充分地使用语音卡结合声控链接库来设计自己的声控专题。

本章先介绍声控指令专题设计及使用方法。

9.1 基本功能

声控指令系统，其功能是用说话可以控制计算机来执行指令的动作，计算机还会以语音来引导您如何操作，并响应您识别的结果。原先所设定要识别的5个单音(以英文发音)及语音存盘如下：

- ✧ DBSP0.SP 语音文件的内容为：“DIR”。
- ✧ DBSP1.SP 语音文件的内容为：“PE2”。
- ✧ DBSP2.SP 语音文件的内容为：“MEMORY”。
- ✧ DBSP3.SP 语音文件的内容为：“DOS”。
- ✧ DBSP4.SP 语音文件的内容为：“SOUND”。
- ✧ 使用者只要说出以上的几个单音，系统会自动识别出您所说出的那一个单音，以语音响应出识别结果并执行相对的动作，如果您所说的那一个单音并不存在语音参考文件中，则系统会告知“无法识别”。

所执行的相对动作如下：

- ✧ DIR ：显示目前工作目录内容。
- ✧ PE2 ：执行PE2文字处理器。
- ✧ MEMORY ：显示目前PC机的内存使用状况。
- ✧ DOS ：暂时离开控制程序返回DOS。
- ✧ SOUND ：发出一声音效。

例如使用者只要说出“DIR”，则系统马上会显示目前工作目录的内容。

本系统的使用注意事项：

- ✧ 本系统为特定语者5个单音或词的语音识别系统。
- ✧ 使用自己的语音参考样本可以获得很高的辨识效果。

9.2 系统组成

整个声控指令系统组成如下：

- ✧ PC语音卡。

✧ 有方向性动圈式麦克风一支。
✧ 4Ω喇叭一支或连到扩音机输入端。
✧ 控制程序COM.EXE及相关程序文件。

相关程序文件说明如下：

✧ COM.C ：声控指令C语言控制程序。
✧ COM.DAT ：语音参考样本文件。
✧ COM.EXE ：声控指令执行文件。
✧ COM.PRJ ：声控指令TURBO C目标文件。
✧ DBSP0.SP--DBSP4.SP：语音参考样本语音数据文件。
✧ EGAVGA.BGI ：TURBO C彩色屏幕驱动程序。
✧ HERC.BGI ：TURBO C单色屏幕驱动程序。
✧ SP.H ：TURBO C声控链接库声明头文件。
✧ SP.LIB ：TURBO C声控链接库展示版。
✧ P0.SP--P4.SP ：语音提示语语音数据文件。

9.3 产生语音提示语

为了使声控应用程序可以提供交谈式的语音识别功能，我们先录制了一些常用的语音提示语，以供后面几章声控应用程序来使用，这些语音提示语是利用第 7 章所介绍的语音编辑程序来录制并经过编辑而产生的语音波形文件，有兴趣修改语音提示语的朋友，可以自行翻阅 7.6 节的使用说明。

目前所录制的语音提示语内容如下：

✧ P0.SP的语音提示内容为：“请说”。
✧ P1.SP的语音提示内容为：“您可以说”。
✧ P2.SP的语音提示内容为：“无法识别”。
✧ P3.SP的语音提示内容为：“您说”。
✧ P4.SP的语音提示内容为：“语音训练”。

9.4 执 行 例 子

执行控制程序COM.EXE，工作画面出现：

```
COMMAND CONTROL     .....
Copyright (C) VICTOR SPEECH LAB.   1996, 1997
 c     --> create DATA BASE
 l     --> listen DATA BASE
 m     --> modify DATA BASE
 SPACE -->   SPEECH recog ....
Select ?
```

共提供有以下的一些控制指令键：

- ✧ 按键“c” ：产生并录制参考样本语音数据文件。
- ✧ 按键“l” ：聆听原设定所要识别的语音数据内容，可以听见 5 个单字音。
- ✧ 按键“m” ：修改参考样本语音数据文件。
- ✧ 空格键 ：进行语音识别。

一般操作步骤如下：

步骤 1：按按键“l”，听一下计算机会识别哪几个单音。

步骤 2：按空格键，直接进行语音识别，系统会提示说“请说”，此时可以对着麦克风说出“MEMOR”，则计算机会进行语音识别对比，将结果说出来，并执行DOS“MEM”的指令，也会显示识别的失真度。由于“MEM”的单音是存在第 3 组的语音参考样本中，其失真度最小，以“>”符号标示出来。

```
Recog.....
. MEM      <执行 MEM 指令>

    655360 bytes total conventional memory
    655360 bytes available to MS-DOS
    119728 largest executable program size

  15728640 bytes total contiguous extended memory
         0 bytes available contiguous extended memory
  12372992 bytes available XMS memory
           MS-DOS resident in High Memory Area

 <显示识别的失真度>
 120   120   > 57   137   144
```

若按下任何键则结束语音识别，返回工作画面。

步骤 3：如果计算机识别出“DIR”，则会显示出目前工作目录下的文件内容。

步骤 4：如果计算机识别出“PE2”，则系统会执行PE2 文字处理器，当然使用者必须要事先设定PE2.EXE程序的执行路径。

步骤 5：如果计算机识别出“DOS”，则系统会暂时离开控制程序而返回DOS，当使用者输入“EXIT”，便可以返回控制程序。

步骤 6：如果计算机识别出“SOUND”，则PC机会发出一声音效。

步骤 7：如果识别率不好，则可以按键“m”，修改某一组参考样本语音数据文件。修改参考样本的操作说明可参考 8.4 节语音卡声控链接库应用范例程序说明。

步骤 8：对于另外一个使用者要进行识别时，或是想修改语音识别的内容时，可以按按键“c”，产生并录制新的参考样本语音数据文件。产生新的参考样本数据文件的操作说明同样参考 8.4 节的说明。

9.5 程 序 设 计

声控指令原始控制程序为COM.C，TURBO C目标文件名为COM.PRJ，其主要语音识别子程序是使用声控链接库，详细的说明请参考第 8 章语音卡声控链接库使用。各主要控制子程序说明如下：

- ✧ menu() ：显示将要识别的语音内容。
- ✧ modify_db() ：修改参考样本语音数据文件。
- ✧ recog() ：语音识别。
- ✧ create_db() ：产生并录制参考样本数据文件。
- ✧ be() ：哔一声。
- ✧ so() ：发出一声音效。
- ✧ prompt() ：听取所要识别的语音数据内容。
- ✧ say_pro(char no) ：播放出某一语音提示语内容。
- ✧ say_db(char no) ：播放出某一语音参考样本内容。
- ✧ action(char no) ：依据识别结果产生相应的动作。

在识别出某一单音编号no后，由动作执行子程序action()控制以便执行相对的指令，在此使用TURBO C函数system()，来执行程序本身的外部执行文件。

例如：

```
system("MEM"); /执行 DOS “MEM” 指令 */
system("COMMAND"); /* 执行 DOS “COMMAND” 指令 */
```

程序清单如下：

```
1  ***********************************************
2  * Program: COM.C
3  * Description: PC SPEECH CARD use SP_LIB example.
4  *         COMMAND control
5  * Copyright (C) VICTOR SPEECH LAB.  1996, 1997
6  ***********************************************/
7   /* c0: DIR   所要识别的单音
8     c1: PE2
9     c2: MEM
10    c3: DOS
11    c4: SOUND
12  */
13  #include <graphics.h>
14  #include <alloc.h>
15  #include <dos.h>
16  #include <stdio.h>
17  #include "sp.h"
18
19  /* 语音卡 I/O 口基地址 */
```

```
#define  AD_PORT  0x250

#define   LEN 32000
#define   char  signed char
#define   DB_FILE  "COM.DAT"
/* SP.LIB 演示版一次最多 5 组进行识别 */
#define   MAX_DB   5

/* 语音提示语内容指针 */
#define PSAY   0  /* 请说      */
#define YOUPRO 1  /* 您可以说  */
#define ANO    2  /* 无法识别  */
#define YOUSAY 3  /* 您说      */
#define VTRAIN 4  /* 语音训练  */
#define REJECT 80.0   /* 无法识别的失真度临界值 */

char  *sp, *sppro;    /* 语音数据的数组指针 */
unsigned sp_len, pro_len; /* 语音数据长度   */

float  (*DB)[FRAME][LEVEL+1];  /* 参考样本特征参数数组 */
float  test_frame[MAX_FRAME][LEVEL+1]; /*  测试样本特征参数数组 */

unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;
int gd=DETECT, gm;

char fname[80];
FILE *in, *out;
int vc_no=5;    /* 所要识别的单音数 */
char string[80];
char *vc_note[]={"DIR", "PE2", "MEM", "DOS",  "SOUND"};
char *co="Copyright (C) VICTOR SPEECH LAB.  1996, 1997";
/*-----------------*/
main()
{
char ch;

 if(!SP_init_sp(AD_PORT)) /* 初始化语音卡 I/O 口 */
  { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }

 DB=farcalloc(MAX_DB, FRAME*(LEVEL+1)*4); /* 动态配置参考样本缓冲区 */
   if(DB==NULL) { printf("%c out of memory !\n", 7); exit(1);}
```

```
62
63   sp=(char  *)calloc(LEN ,sizeof(char));  /* 动态配置语音数据缓冲区 */
64   if (sp==NULL)  { printf("%c out of memory !\n", 7); exit(1);}
65
66  sppro=(char  *)calloc(LEN ,sizeof(char)); /* 动态配置语音提示语缓冲区 */
67   if (sppro==NULL)  { printf("%c out of memory !\n", 7); exit(1);}
68
69   if(SP_load_db(DB, vc_no, DB_FILE)) /* 加载参考样本数据文件 */
70     puts("load DATA BASE .......");
71   else  { printf("%c NO %s  DATA BASE !\n", 7, DB_FILE); getch(); }
72
73  while(1) /* 无穷循环 */
74   {
75    clrscr(); /* 显示工作画面 */
76    puts("COMMAND CONTROL   .......");
77    puts(co);
78    puts(" c   --> create DATA BASE");
79    puts(" l   --> listen DATA BASE");
80    puts(" m   --> modify DATA BASE");
81    puts(" SPACE -->  SPEECH recog ....");
82    puts("Select ? ");
83
84    ch=getch(); /* 等待按键 */
85     switch(ch)
86      {
87        case 27 : exit(0); /* 若按下 ESC 键则结束程序执行 */
88        case 'c': create_db(); break; /* 产生并录制参考样本数据文件 */
89        case 'l': prompt();    break; /* 听取所要识别的语音数据内容 */
90        case 'm': modify_db(); break; /* 修改参考样本语音数据文件 */
91        case ' ': recog();     break; /* 语音识别 */
92        default : break;
93     }
94    }
95     free(sp);   /* 释放动态配置的语音数据缓冲区 */
96     free(sppro);/* 释放动态配置的语音提示语数据缓冲区 */
97     farfree(DB);/* 释放动态配置的参考样本缓冲区 */
98
99  /*------------------------------*/
100  menu()  /* 显示欲识别的语音内容 */
101  {
102   puts("0 --> DIR  ");
103   puts("1 --> PE2  ");
```

```
104   puts("2 --> MEM  ");
105   puts("3 --> DOS  ");
106   puts("4 --> SOUND");
107  }
108  /*------------------*/
109  modify_db()  /* 修改参考样本语音数据文件 */
110  {
111  int i, exit_f,no, ans, frames;
112  char c, mess[80];
113  float dist[10];
114
115    puts("");
116    menu();
117  /* 输入修改参考样本编号 */
118    puts("=== modify word no ===");
119    do{ printf("modify which no ");
120    scanf("%d", &no);
121    } while( no>=vc_no); /* 若编号无效则重新输入编号 */
122
123  again:
124  /* 发出“请说”提示语 */
125    say_pro(PSAY);
126    printf("please say <  %s >\n", vc_note[no]);
127    say_db(no); /* 播放出所要修改的该组语音内容 */
128
129  /* 取得语音数据..... */
130    be();   /* 哔一声 */
131    exit_f=0;  /* 清除脱离标志位 */
132    /* 语音输入自动侦测并检查是否有按键 */
133    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
134  /* 语音切割 */
135     segs=SP_separate(sp, sp_len, bp, ep);
136     if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
137  /* 初始化图形接口 */
138     initgraph(&gd, &gm, "");
139     SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
140           0.6, bp, ep, segs); /* 显示语音信号波形 */
141  /* 播放出该段语音内容 */
142     SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
143  /* 语音信号特征参数分析 */
144     if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
145      { printf("%c cannot take feature ! !\n", 7); exit(1); _7d
```

```
146
147  /* 语音识别比对测试 */
148     frames=16;
149     ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
150  /* 显示识别比对测试结果 */
151     sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
152             vc_note[ans], dist[ans]);
153  /* 显示操作信息 */
154     outtextxy(10, getmaxy()-80, mess);
155     outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
156     outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
157
158  /* 等待按键处理 */
159     c=getch();
160
161     if(c==0x1b)  {  closegraph(); return; }
162     if(c==0x0d)  {  closegraph(); goto next; }
163     if(c!='y')   {  closegraph(); goto again; }
164       else closegraph();
165
166  next:;
167  /* 语音信号特征参数分析 */
168     if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
169      { printf("%c cannot take feature ! !\n", 7); exit(1); }
170
171  /* 存为语音参考样本文件 */
172     puts("Saving data base .....");
173     if( SP_save_db(DB, vc_no, DB_FILE) )
174       { printf("Modify data base  <<<%d  %s  >>> ok ! ", no,
175         vc_note[no]); delay(2000); }
176     else { printf("%c cannot save data base !\n", 7); exit(1); }
177  }
178  /*--------------------*/
179  recog() /* 语音识别 */
180  {
181  int ans, i;
182  int exit_f, frames;
183  float dist[10];
184
185   puts("Recog.....");
186  while(1) /* 无穷循环 */
187   {
```

```
again:
  be(); /* 哔一声 */
  exit_f=0; /* 清除脱离标志位 */
  say_pro(PSAY);  /* 发出“请说”提示语 */
  sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
  /* 语音输入自动侦测并检查是否有按键 */
  if(exit_f) break; /* 若有按键则跳离循环 */

  printf(".");
  segs=SP_separate(sp, sp_len, bp, ep);/* 语音切割 */
  if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
  /* 语音信号特征参数分析 */
  if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
     { printf("%c cannot take feature ! !\n", 7); exit(1); _7d
  frames=16;
  /* 语音识别对比 */
  ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
  if(dist[ans]>=REJECT)  /* 失真度大于临界值则告知无法识别 */
    { printf("NOT ");  say_pro(ANO); }
   else
    { /* 显示识别结果 */
     printf(" %s ", vc_note[ans]);
      /* 播放出识别结果 */
     say_pro(YOUSAY); delay(300);
     say_db(ans); /* 播放出该组语音内容 */
     action(ans); /* 执行动作 */ delay(1000);
    }

/* 列出语音识别的失真度 */
  for(i=0; i<vc_no; i++)
   {
    if(ans==i) printf(">");
    printf("%3.0f   ",dist[i]);
   }
  puts("");

 }/* 循环 */
}
/*--------------------*/
create_db()  /* 产生并录制参考样本数据文件 */
{
int i, exit_f,no;
```

```
229 char c;
230 int ans, frames;
231 float dist[10];
232 char mess[80];
233
234  clrscr();
235  puts("Create data base ");
236  menu(); /* 显示将要识别的语音内容 */
237
238  say_pro(VTRAIN); /* 发出“语音训练”提示语  */
239 for(no=0; no<vc_no; no++)
240  {
241 again:
242   printf("please say <  %s >\n", vc_note[no]);
243 /* 取得语音数据..... */
244   be();     /* 哔一声 */
245   exit_f=0; /* 清除脱离标志位 */
246 /* 语音输入自动侦测并检查是否有按键 */
247   sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
248 /* 语音切割 */
249   segs=SP_separate(sp, sp_len, bp, ep);
250   if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
251 /* 初始化图形接口 */
252   initgraph(&gd, &gm, "");
253   SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
254          0.6, bp, ep, segs); /* 显示语音信号波形 */
255 /* 播放出该段语音内容 */
256   SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
257 /* 语音信号特征参数分析 */
258   if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
259    { printf("%c cannot take feature ! !\n", 7); exit(1); _7d
260
261 /* 语音识别对比测试 */
262   frames=16;
263   ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
264 /* 显示识别对比测试结果 */
265   sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
266          vc_note[ans], dist[ans]);
267 /* 显示操作信息 */
268   outtextxy(10, getmaxy()-80, mess);
269   outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
270   outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
```

```
271
272 /* 等待按键处理 */
273    c=getch();
274    if(c==0x1b)  {  closegraph(); return; }
275    if(c==0x0d)  {  closegraph(); goto next; }
276    if(c!='y')   {  closegraph(); goto again; }
277       else closegraph();
278 next:;
279 /* 原始语音数据存盘 */
280    sprintf(fname, "DBSP%d.SP",no);
281    if(!SP_save_voice1(sp, bp[0], ep[0], fname))
282     { puts("Cannot save prompt file "); exit(1); }
283
284 /* 语音信号特征参数分析 */
285    if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
286     { printf("%c cannot take feature ! !\n", 7); exit(1); }
287  }/* end of no */
288
289 /* 存为语音参考样本文件 */
290    puts("Saving data base ......");
291    if( SP_save_db(DB, vc_no, DB_FILE) )
292      { puts("Save data base  OK !");   delay(2000); }
293    else { printf("%c cannot save data base !\n", 7); exit(1); }
294
295  prompt(); /* 播放出所有识别的语音数据内容 */
296 }
297 /*--------------------*/
298 be()  /* 哔一声 */
299 {
300   sound(500);
301   delay(100);
302   nosound();
303 }
304 /*---------------*/
305 prompt()  /* 听取所要识别的语音数据内容 */
306 {
307 int no;
308
309  printf("Say data base .....");
310  say_pro(YOUPRO);
311  for(no=0; no<vc_no; no++)
312   {
```

```
313    printf("%s ",vc_note[no]);
314    say_db(no);  /* 播放出某一参考样本语音 */
315    delay(500);
316   }
317  }
318  /*-------------------*/
319  say_pro(char no)  /* 播放出某一语音提示语内容 */
320  {
321    sprintf(fname, "P%d.SP",no);  /* 产生语音提示语文件名 */
322  if(!SP_load_voice(sppro, LEN, &pro_len, fname)) /* 加载语音数据文件 */
323     { puts("Cannot find prompt file"); exit(1); }
324    SP_sp_out(sppro, 0, pro_len, LEN, 8000); /* 播放出语音数据内容 */
325  }
326  /*-------------------*/
327  say_db(char no)  /* 播放出某一语音参考样本内容 */
328  {
329    sprintf(fname, "DBSP%d.SP",no); /* 产生语音参考样本语音文件名 */
330  if(!SP_load_voice(sppro, LEN, &pro_len, fname)) /* 加载语音数据文件 */
331     { puts("Cannot find prompt file"); exit(1); }
332    SP_sp_out(sppro, 0, pro_len, LEN, 8000); /* 播放出语音数据内容 */
333  }
334  /*----------------------*/
335  action(char no) /* 依据识别结果产生相对的动作 */
336  {
337   switch(no)
338    {
339     case 0 : system("DIR");  break;  /* 显示工作目录下文件内容 */
340     case 1 : system("PE2");  break;  /* 执行文字处理器 */
341     case 2 : system("MEM");  break;  /* 执行 DOS "MEM" 指令*/
342     case 3 : puts("TYPE 'exit' to back ....");
343              system("COMMAND");  break; /* 执行 DOS "COMMAND" 指令*/
344     case 4 : so();  break;   /* 发出一声音效*/
345     default : break;
346    }
347  }
348  /*---------------*/
349  so()  /* 发出一声音效*/
350  {
351  int i;
352
353   for(i=500; i<2000; i+=100)
354    {
```

```
355     sound(i);
356     delay(300);
357    }
358   nosound();
359  }
```

习　题

原始声控指令程序COM.EXE的语音参考样本文件名为COM.DAT，修改原始程序，使得每一个不同使用者，可以自行加载属于自己的语音参考样本文件。

第 10 章　声控电话拨号

本章将以声控链接库来设计声控电话拨号系统，本系统可以用声音来控制电话拨号，不必记电话号码，免拨号，直接说出想要打电话的对象，系统会自动控制MODEM来进行电话拨号，这是一个简单却很实用且有趣的声控专题制作。在本章中除了可以学习声控程序的设计外，还可以控制MODEM来做一些简单的电话拨号实验，本章先介绍MODEM电话拨号的原理及使用方法。

10.1　MODEM 电话拨号

调制解调器(MODEM)是计算机外设中一项重要的产品，现在个人计算机只要连上MODEM，便可以通过电话线连上国际互联网，与世界各地的朋友或是厂商交换信息。MODEM的主要用途如下：

- ✧ 计算机自动拨号。
- ✧ 远程数据文件传送。
- ✧ 远程接口自动控制。
- ✧ 连上电子公告牌BBS站。
- ✧ 连上国际互联网。

现在新型的MODEM还具备有传真及语音供能，配合PC机可以建立语音信箱，或是计算机数据自动传真回复查询，是办公自动化中一项不可缺少的计算机外设。在本章声控电话拨号专题制作中，我们是用来做计算机自动拨号。

10.1.1　MODEM 的连接

MODEM是连接到个人计算机的串行通讯端口上，它介于PC机与电话线之间，利用现成普及全世界的电话网络可以将两台计算机做数据的传送。实际上以PC机来控制MODEM，主要是经由串行通讯端口RS232 接口来达成，由于个人计算机有两个通讯端口COM1 以及COM2。COM1 通常连接的是鼠标，因此MODEM一般是连至COM2，在程控方面可以直接控制RS232 通讯端口接口进而控制MODEM动作。

10.2　TURBO C RS232 通讯端口接口

TURBO C函数中对于通讯端口的控制是使用bioscom函数，其原型声明如下：

```
int bioscom (int cmd, char brte, int port);
```

在输入参数有 3 种：

- cmd：控制函数的工作命令。
- byte：通讯协议参数或发送或接收的数据。
- port：通讯端口设定。

10.2.1 工作命令 cmd

- cmd=0：设定通讯协议参数，以byte定义来完成通讯协议的设定。
- cmd=1：将byte的值写入通讯端口中。
- cmd=2：从通讯端口中读取一个字节数据。
- cmd=3：传回通讯端口目前的状态。

10.2.2 通讯协议参数 byte

- 传输率（波特率）有如下几种：
  0X00=110 bps
  0X20=150 bps
  0X40=300 bps
  0X60=600 bps
  0X80=1200 bps
  0XA0=2400 bps
  0XC0=4800 bps
  0XE0=9600 bps
- 奇偶校验位：
  0X00=无奇偶校验位
  0X08=采用奇校验
  0X18=采用偶校验
- 数据字符长度：
  0X02=7 个数据位
  0X03=8 个数据位
- 停止位个数：
  0X00=1 位停止位
  0X04=2 位停止位

其中个别的要项可以任意组合，例如，常用的通讯协议为（9600,8,N,1），即：

- 波特率 9600 bps ：0XE0
- 8 个数据位 ：0X03
- 没有同位检查位 ：0X00
- 1 个停止位 ：0X00

则参数设定为 0XE3，计算方式为：0XE0+0X00+0X03+0X00=0XE3

10.2.3 通讯端口 port 指定

0=com1，则为通讯端口 1；1=com2，则为通讯端口 2。执行bioscom函数后，传回值为一整数，其高字节（B15～B8，位 15 至位 8）为通讯端口状态，低字节（B7～B0）则定义如下：

- ✧ cmd=0 时，表示调制解调器状态。
- ✧ cmd=1 时，无意义。
- ✧ cmd=2 时，表示所接收到的数据。
- ✧ cmd=3 时，调制解调器状态。

10.2.4 通讯端口状态

若该位为 1 表示所描述的状态设定。

- ✧ B15：传输时间用完，时间到了（Time Out）。
- ✧ B14：传送移位寄存器（Transmit Shift Register，TSR）空。
- ✧ B13：传送保持寄存器（Transmit Holding Register，THR）空。
- ✧ B12：侦测到间断（Break）信号。
- ✧ B11：发生帧格式（Framing）错误。
- ✧ B10：奇偶检验错误。
- ✧ B9 ：发生重叠（Overrun）错误。
- ✧ B8 ：接收数据备妥。

其中帧格式错误表示串行输入缓冲器接收到一个完整的字符数据后，却侦测不到停止位。而重叠错误表示存在串行输入缓冲器内的数据未被读取又收到下一个接收进来的字符数据，因此前一个字符数据被新的字符数据覆盖掉而遗失。

10.2.5 调制解调器状态

- ✧ B7：指示DCD（Data Carrier Detect）的状态。
- ✧ B6：指示RI（Ring Indicator）的状态。
- ✧ B5：指DSR（Data Set Ready）的状态。
- ✧ B4：指示CTS（Clear To Send）的状态。
- ✧ B3：侦测到DCD信号有变化。
- ✧ B2：侦测到RI信号有变化。
- ✧ B1：侦测到DSR信号有变化。
- ✧ B0：侦测到CTS信号有变化。

10.3 MODEM 电话拨号控制

由计算机对MODEM做控制是使用“AT”命令组，目前市面上所买到的MODEM都可

以执行“AT命令组”的功能，之所以称为“AT命令组”，是因为在下达命令组给MODEM时，所有命令起头都加上“AT”这两个字符。例如：接通电话(拿起电话筒)“AT命令组”为“ATH1”，切断电话(挂下电话)“AT命令组”为“ATH0”。一般电话拨号控制常用到的控制动作如下：

- ✧ MODEM系统开机复位。
- ✧ 接通电话。
- ✧ 切断电话。
- ✧ 电话拨号。

相对的“AT命令组”如表 10.1：

表 10.1 动作所对应 AT 命令组

控制动作	AT 命令组
系统复位	ATZ
接通电话	ATH1
切断电话	ATH0
电话拨号	ATDXXXX

注：其中 XXXX 表示所要拨出去的电话号码。

在看过以上的简易MODEM控制拨号说明后，我们先来写一个利用MODEM来做计算机拨号的程序，程序文件名为TE.C，在执行控制程序前，请先连接MODEM 于通讯端口2(COM2)，并开启MODEM电源。执行控制程序TE.EXE后，工作画面出现：

```
==========================

PC TEL dial via MODEM

==========================

Copyright (C) VICTOR SPEECH LAB.   1996, 1997


 1    → TEL ON
 2    → TEL OFF
 d    → Dial 117 ....
 t    → Dial VICTOR LAB
ESC   → Exit
Select ?
```

共提供有以下的一些控制指令键：

- ✧ 按键“1”：令MODEM接通电话，此时可以听见MODEM接通电话后的嘟声。
- ✧ 按键“2”：令MODEM切断电话，此时电话的嘟声停止。
- ✧ 按键“d”：电话拨号，拨出“117”为报时台。
- ✧ 按键“t”：电话拨号至本工作室。

程序清单如下：

```
1   /*************************************************
```

```
2  * Program: TE.C                                    *
3  * Description:  PC TEL dial via MODEM              *
4  *         PC / RS232 COM2  <9600 N 8 1>            *
5  * Copyright (C) VICTOR SPEECH LAB.  1996, 1997      *
6  ****************************************************/
7
8  #include <bios.h>
9  #include <stdio.h>
10
11 #define PROTOCAL 0xe3 /* 通讯协议句柄 */
12 #define PROT   0
13 #define TX     1
14 #define RX     2
15 #define STATUS 3
16 int port=1;  /* 使用 COM2 */
17
18 char *co="Copyright (C) VICTOR SPEECH LAB.  1996, 1997";
19 /*------------------------------*/
20 main()
21 {
22 char c;
23  /* 设定 RS232 通讯协议 */
24  bioscom(PROT, PROTOCAL, port);
25  clrscr();
26  puts("COM2 : <9600 N 8 1>  ");
27  puts("Please connect MODEM to COM2 ");
28  reset_modem(); /* 初始化调制解调器 */
29
30  while(1) /* 无穷循环 */
31   {
32    clrscr(); /* 显示工作画面 */
33    puts("========================");
34    puts("PC TEL dial via MODEM");
35    puts("========================");
36    puts(co);
37    puts("");
38    puts(" 1   → TEL ON       ");/* 接通电话 */
39    puts(" 2   → TEL OFF      ");/* 切断电话 */
40    puts(" d   → Dial 117 ....");/* 电话拨号 117 报时 */
41    puts(" t   → Dial VICTOR LAB "); /* 电话拨号至本工作室 */
42    puts("ESC  → Exit   ");
43    puts("Select ? ");
```

```
44    c=getch(); /* 等待按键 */
45    switch(c)
46      { /* 若按下 ESC 键则结束程序执行 */
47        case 27 : exit(0);
48                  /* 接通电话 */
49        case '1': printf("TEL ON"); send("ATH1");
50                  delay(5000);  break;
51                  /* 切断电话 */
52        case '2': printf("TEL OFF"); send("ATH0");
53                  delay(5000);  break;
54                  /* 电话拨号 117 报时 */
55        case 'd': printf("Dial 117 ..."); send("ATDT117");
56                  delay(5000);  break;
57                  /* 电话拨号至本工作室 */
58        case 't': printf("Dial VICTOR LAB."); send("ATDT2260258");
59                  delay(5000);  break;
60        default : break;
61       _7d
62    }/* loop */
63  }
64  /*-------------------*/
65  tx(unsigned char c) /* 由 RS232 接口送出字符 */
66  {
67    bioscom(TX, c, port);
68  }
69  /*------------------*/
70  send(char *str) /* 由 RS232 接口送出字符串 */
71  {
72  int i;
73    i=0;
74    do{ tx(str[i]); i++; }
75    while(str[i]!='\0');
76    tx(0x0d);
77  }
78  /*--------------------*/
79  reset_modem() /* 初始化调制解调器 */
80  {
81   send("ATZ");
82  }
83  /*-----------------*/
```

10.4 声控电话拨号基本功能

声控指令系统，其功能是用语音来控制计算机经由MODEM来进行电话拨号，计算机还会以语音来引导您如何操作，并响应您识别的结果。原先所设定要识别的5个单音及语音文件如表10.2。

表10.2 单个音及语音文件

语音文件名	内容
DBSP0.SP	TIME(报时)
DBSP1.SP	VICTOR(伟克)
DBSP2.SP	JACK
DBSP3.SP	AFA(阿发)
DBSP4.SP	SLIN(小林)

表10.3 单个音所对应的动作

内容	动作
报时	拨电话至报时台 117
伟克	拨电话至本工作室 TEL: 2260258
JACK	拨电话给朋友 JACK TEL: 2234567
阿发	拨电话给朋友 AFA TEL: 3731234
小林	拨电话给朋友 SLIN TEL: 3879876

使用者只要说出以上的几个单音，系统会自动识别出所说出的那一个单音，以语音响应出识别结果并进行电话拨号的动作，如果所说的那一个单音并不存在语音参考文件中，则系统会告知“无法识别”。所执行的相对动作如表10.3。

例如使用者只要说出“报时”，则系统马上会拨电话至报时台，而使用者可以马上听见现在的时间。

本系统的使用注意事项：

- ✧ 本系统为特定语者5个单音或词的语音识别系统。
- ✧ 使用自己的语音参考样本可以获得很高的辨识效果。
- ✧ 所使用的MODEM内部必须接有喇叭，否则要经由外接喇叭才能听见电话拨通后对方的声音。
- ✧ 如果电话拨通后想与对方通话，可以拿起电话听筒进行对谈，但是基本上您的电话必须与MODEM并连至外线。

10.5 系 统 组 成

整个声控指令系统组成如下：

- ✧ PC语音卡。
- ✧ 有方向性动圈式麦克风一支。
- ✧ 4Ω喇叭一支或连至扩音机。
- ✧ 控制程序VTE.EXE及相关程序文件。
- ✧ MODEM一台。

相关程序文件说明如下：

- ✧ VTE.C ：声控拨号C语言控制程序。
- ✧ VTE.DAT ：语音参考样本文件。
- ✧ VTE.EXE ：声控拨号执行文件。
- ✧ VTE.PRJ ：声控拨号TURBO C目标文件。

- ✧ DBSP0.SP--DBSP4.SP ：语音参考样本语音数据文件。
- ✧ EGAVGA.BGI ：TURBO C彩色屏幕驱动程序。
- ✧ HERC.BGI ：TURBO C单色屏幕驱动程序。
- ✧ SP.H ：TURBO C声控链接库头文件声明。
- ✧ SP.LIB ：TURBO C声控链接库演示版。
- ✧ P0.SP--P4.SP ：语音提示语语音数据文件。

10.6 执 行 例 子

执行控制程序VTE.EXE，工作画面如下：

```
VOICE CONTROL TEL DIAL  .....
Copyright (C) VICTOR SPEECH LAB.  1996, 1997
 c   → create DATA BASE
 l   → listen DATA BASE
 m   → modify DATA BASE
 SPACE →  SPEECH recog ....
------------------------
0 → TIME        TEL: 117
1 → VICTOR      TEL: 2260258
2 → JACK        TEL: 2234567
3 → AFA         TEL: 3731234
4 → SLIN        TEL: 3879876
Select ?
```

共提供有以下的一些控制指令键：

- ✧ 按键“c” ：产生并录制参考样本语音数据文件。
- ✧ 按键“l” ：聆听原设定所要识别的语音数据内容，可以听见5个单字音。
- ✧ 按键“m” ：修改参考样本语音数据文件。
- ✧ 空格键 ：进行语音识别。

一般操作步骤如下：

步骤1：按按键“l”，听一下计算机会识别哪几个单音。

步骤2：按空格键，直接进行语音识别，系统会提示说“请说”，此时可以对着麦克风说出“报时”，则计算机会进行语音识别对比，将结果说出来，并执行电话拨号至报时台，也会显示识别的失真度。若按下任何键则结束语音识别，返回工作画面。

步骤3：如果识别效果不好，则可以按键“m”，修改某一组参考样本语音数据文件。修改参考样本的操作说明可参考8.4节语音卡声控链接库应用范例程序说明。

步骤4：对于另外一个使用者要进行识别时，或是想修改语音识别的内容时，可以按按键“c”，产生并录制新的参考样本语音数据文件。产生新的参考样本数据文件的操作说明同样参考8.4节的说明。

10.7 程 序 设 计

声控指令原始控制程序为VTE.C，TURBO C目标文件名为VTE.PRJ，其主要语音识别子程序是使用声控链接库，详细的说明请参考第 8 章语音卡声控链接库使用。各主要控制子程序说明如下：

- ✧ menu() ：显示将要识别的语音内容。
- ✧ modify_db() ：修改参考样本语音数据文件。
- ✧ recog() ：语音识别。
- ✧ create_db() ：产生并录制参考样本数据文件。
- ✧ be() ：哔一声。
- ✧ prompt() ：听取所要识别的语音数据内容。
- ✧ say_pro(char no) ：说出某一语音提示语内容。
- ✧ say_db(char no) ：说出某一语音参考样本内容。
- ✧ action(char no) ：依据识别结果进行电话拨号。
- ✧ send(char *str) ：由RS232 接口送出字符串。
- ✧ reset_modem() ：初始化调制解调器。
- ✧ tx(unsigned char c)：由RS232 接口送出字符。

在识别出某一单音编号后，由动作执行子程序action()控制以便执行电话拨号，以子程序send()由RS232 接口送出控制字符串，来控制MODEM进行电话拨号，达成语音声控自动拨号的目的。

程序清单如下：

```
/*******************************************************
* Program: VTE.C                                       *
* Description:  PC SPEECH CARD use SP_LIB example.   *
*         VOICE control dial via MODEM                 *
* Copyright (C) VICTOR SPEECH LAB.  1996, 1997       *
*******************************************************/
/* c0: TIME  所要识别的单音
   c1: VICTOR
   c2: JACK
   c3: AFA
   c4: SLIN
 */
#include <graphics.h>
#include <alloc.h>
#include <dos.h>
#include <stdio.h>
#include <bios.h>

#include "sp.h"
```

```
20
21  /* 语音卡 I/O 口基地址 */
22  #define  AD_PORT  0x250
23
24  #define   LEN 32000
25  #define   DB_FILE  "VTE.DAT"
26  #define   MAX_DB   5
27  /* SP.LIB 演示版一次最多 5 组进行识别 */
28  /* 语音提示语内容指针 */
29  #define PSAY   0  /* 请说     */
30  #define YOUPRO 1  /* 您可以说 */
31  #define ANO    2  /* 无法识别 */
32  #define YOUSAY 3  /* 您说     */
33  #define VTRAIN 4  /* 语音训练  */
34  #define REJECT 80.0   /* 无法识别的失真度临界值 */
35
36  #define PROTOCAL 0xe3 /* 通讯协议句柄 */
37  #define PROT   0
38  #define TX     1
39  #define RX     2
40  #define STATUS 3
41  int port=1;  /* 使用 COM2 */
42
43  char  *sp, *sppro; /* 语音数据的数组指针 */
44  unsigned sp_len, pro_len; /* 语音数据长度  */
45
46  float  (*DB)[FRAME][LEVEL+1]; /* 参考样本特征参数数组 */
47  float  test_frame[MAX_FRAME][LEVEL+1]; /*  测试样本特征参数数组 */
48
49  unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
50  int segs;   /*  所切割出来的单音数  */
51  int gd=DETECT, gm;
52
53  char fname[80];
54  FILE *in, *out;
55  int vc_no=5; /* 所要识别的单音数 */
56  char string[80];
57
58  char *vc_note[]={"TIME" , "VICTOR", "JACK", "AFA",  "SLIN"};
59  char *co="Copyright (C) VICTOR SPEECH LAB.  1996, 1997";
60  /*-----------------*/
61  main()
```

```
{
char ch;

 clrscr();
 puts("COM2 : <9600 N 8 1>  ");
 puts("Please connect MODEM to COM2 ");
 reset_modem(); /* 初始化 MODEM  */
 /* 初始化语音卡 I/O 口 */
 if(!SP_init_sp(AD_PORT))
  { puts("CANNOT FIND SPEECH CARD !"); getch(); }
 /* 动态配置参考样本缓冲区 */
 DB=farcalloc(MAX_DB, FRAME*(LEVEL+1)*4);
   if(DB==NULL) { printf("%c out of memory !\n", 7); exit(1);}
 /* 动态配置语音数据缓冲区 */
 sp=(char  *)calloc(LEN ,sizeof(char));
 if (sp==NULL)     { printf("%c out of memory !\n", 7); exit(1);}
 /* 动态配置语音提示语缓冲区 */
 sppro=(char  *)calloc(LEN ,sizeof(char));
 if (sppro==NULL)  { printf("%c out of memory !\n", 7); exit(1);}
 /* 加载参考样本数据文件 */
 if(SP_load_db(DB, vc_no, DB_FILE))
    puts("load DATA BASE .......");
 else  { printf("%c NO %s  DATA BASE !\n", 7, DB_FILE); getch(); }

while(1) /* 无穷循环 */
 {
  clrscr();  /* 显示工作画面 */
  puts("VOICE CONTROL TEL DIAL  .......");
  puts(co);
  puts(" c   → create DATA BASE");
  puts(" l   → listen DATA BASE");
  puts(" m   → modify DATA BASE");
  puts(" SPACE →  SPEECH recog ....");
  puts("------------------------");
  menu();
  puts("Select ? ");

  ch=getch(); /* 等待按键 */
  switch(ch)
    {
      case 27 : exit(0);  /* 若按下 ESC 键则结束程序执行 */
      case 'c': create_db(); break; /* 产生并录制参考样本数据文件 */
```

```
104       case 'l': prompt();    break; /* 听取所要识别的语音数据内容 */
105       case 'm': modify_db(); break; /* 修改参考样本语音数据文件 */
106       case ' ': recog();     break; /* 语音识别 */
107       default : break;
108      }
109    }
110     free(sp);    /* 释放动态配置的语音数据缓冲区 */
111     free(sppro); /* 释放动态配置的语音提示语数据缓冲区 */
112     farfree(DB); /* 释放动态配置的参考样本缓冲区 */
113  }
114  /*-------------------------------*/
115  menu()  /* 显示将要识别的人名及相应的电话号码 */
116  {
117   puts("0 → TIME    TEL: 117     ");
118   puts("1 → VICTOR  TEL: 2260258 ");
119   puts("2 → JACK    TEL: 2234567 ");
120   puts("3 → AFA     TEL: 3731234 ");
121   puts("4 → SLIN    TEL: 3879876 ");
122  }
123  /*-------------------*/
124  modify_db() /* 修改参考样本语音数据文件 */
125  {
126  int i, exit_f,no, ans, frames;
127  char c, mess[80];
128  float dist[10];
129
130    puts("");
131    menu();
132  /* 输入修改参考样本编号 */
133    puts("=== modify word no ===");
134    do{printf("modify which no ");
135    scanf("%d", &no);
136    } while( no>=vc_no); /* 若编号无效则重新输入编号 */
137
138  again:
139  /* 发出“请说”提示语 */
140    say_pro(PSAY);
141    printf("please say <  %s >\n", vc_note[no]);
142    say_db(no); /* 说出所要修改的该组语音内容 */
143
144  /* 取得语音数据..... */
145    be();     /* 哔一声 */
```

```
146    exit_f=0; /* 清除脱离标志位 */
147    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
148  /* 语音输入自动侦测并检查是否有按键按下 */
149  /* 语音切割 */
150    segs=SP_separate(sp, sp_len, bp, ep);
151    if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
152  /* 初始化图形接口 */
153    initgraph(&gd, &gm, "");
154    SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
155           0.6, bp, ep, segs); /* 显示语音信号波形 */
156    SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
157  /* 播放出该段语音内容 */
158  /* 语音信号特征参数分析 */
159    if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
160     { printf("%c cannot take feature ! !\n", 7); exit(1); _7d
161
162  /* 语音识别对比测试 */
163    frames=16;
164    ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
165  /* 显示识别对比测试结果 */
166    sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
167           vc_note[ans], dist[ans]);
168    outtextxy(10, getmaxy()-80, mess);
169  /* 显示操作信息 */
170    outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
171    outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
172
173  /* 等待按键处理 */
174    c=getch();
175
176    if(c==0x1b)     {  closegraph(); return; }
177    if(c==0x0d)     {  closegraph(); goto next; }
178    if(c!='y')   {  closegraph(); goto again; }
179      else closegraph();
180
181  next:;
182  /* 语音信号特征参数分析 */
183    if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
184     { printf("%c cannot take feature ! !\n", 7); exit(1); }
185
186  /* 存为语音参考样本文件 */
187    puts("Saving data base ......");
```

```
188    if( SP_save_db(DB, vc_no, DB_FILE) )
189    { printf("Modify data base  <<<%d  %s  >>> ok ! ", no, vc_note[no]);
190      delay(2000); }
191    else { printf("%c cannot save data base !\n", 7); exit(1); }
192  }
193  /*--------------------*/
194  recog() /* 语音识别 */
195  {
196  int ans, i;
197  int exit_f, frames;
198  float dist[10];
199
200   puts("Recog.....");
201  while(1) /* 无穷循环 */
202   {
203  again:
204    be(); /* 哔一声 */
205    exit_f=0;   /* 清除脱离标志位 */
206    say_pro(PSAY);  /* 发出“请说”提示语 */
207    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
208    if(exit_f) break; /* 语音输入自动侦测并检查是否有按键按下 */
209  /* 若有按键则跳离循环 */
210
211    printf(".");
212    segs=SP_separate(sp, sp_len, bp, ep); /* 语音切割 */
213    if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
214  /* 语音信号特征参数分析 */
215    if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
216      { printf("%c cannot take feature ! !\n", 7); exit(1); }
217  /* 语音识别对比 */
218    frames=16;
219    ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
220    if(dist[ans]>=REJECT){ printf("NOT "); say_pro(ANO); }
221     else /* 失真度大于临界值则告知无法识别 */
222      { /* 显示识别结果 */
223       printf(" %s ", vc_note[ans]);
224  /* 播放出识别结果 */
225       say_pro(YOUSAY);delay(300);
226       say_db(ans); /* 播放出该组语音内容 */
227       action(ans); delay(1000); /* 执行动作 */
228      }
229
```

```
/* 列出语音识别的失真度 */
  for(i=0; i<vc_no; i++)
   {
    if(ans==i) printf(">");
    printf("%3.0f   ",dist[i]);
   }
  puts("");

 }/* 循环 */
}
/*--------------------*/
create_db()  /* 产生并录制参考样本数据文件 */
{
int i, exit_f,no;
char c;
int ans, frames;
float dist[10];
char mess[80];

  clrscr();
  puts("Create data base ");
  menu();  /* 显示将要识别的语音内容 */

  say_pro(VTRAIN); /* 发出“语音训练”提示语 */
 for(no=0; no<vc_no; no++)
  {
again:
   printf("please say <  %s >\n", vc_note[no]);
/* 取得语音数据..... */
   be();     /* 哔一声 */
   exit_f=0; /* 清除脱离标志位 */
   sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
/* 语音输入自动侦测并检查是否有按键 */
/* 语音切割 */
   segs=SP_separate(sp, sp_len, bp, ep);
   if(segs==0) goto again;/* 若无有效语音分段则重新做语音输入 */
/* 初始化图形接口 */
   initgraph(&gd, &gm, "");
   SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
         0.6, bp, ep, segs); /* 显示语音信号波形 */
   SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
/* 播放出该段语音内容 */
```

```
/* 语音信号特征参数分析 */
  if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
   { printf("%c cannot take feature ! !\n", 7); exit(1); }

/* 语音识别对比测试 */
  frames=16;
  ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
/* 显示识别对比测试结果 */
  sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
          vc_note[ans], dist[ans]);
  outtextxy(10, getmaxy()-80, mess);
/* 显示操作信息 */
  outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
  outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");

/* 等待按键处理 */
  c=getch();
  if(c==0x1b)     {  closegraph(); return; }
  if(c==0x0d)     {  closegraph(); goto next; }
  if(c!='y')   {  closegraph(); goto again; }
    else closegraph();
next:;
/* 原始语音数据存盘 */
  sprintf(fname, "DBSP%d.SP",no);
  if(!SP_save_voice1(sp, bp[0], ep[0], fname))
   { puts("Cannot save prompt file "); exit(1); }

/* 语音信号特征参数分析 */
  if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
   { printf("%c cannot take feature ! !\n", 7); exit(1); }
 }/* end of no */

/* 存为语音参考样本文件 */
  puts("Saving data base ......");
  if( SP_save_db(DB, vc_no, DB_FILE) )
    { puts("Save data base  OK !");   delay(2000); }
  else { printf("%c cannot save data base !\n", 7); exit(1); }

 prompt(); /* 播放出所有识别的语音数据内容 */
}
/*--------------------*/
be() /* 哔一声 */
```

```
{
  sound(500);
  delay(100);
  nosound();
}
/*--------------*/
prompt() /* 听取所要识别的语音数据内容 */
{
int no;

 printf("Say data base .....");
 say_pro(YOUPRO);
 for(no=0; no<vc_no; no++)
  {
   printf("%s ",vc_note[no]);
   say_db(no); /* 播放出某一参考样本语音 */
   delay(1500);
  }
}
/*--------------------*/
say_pro(char no) /* 播放出某一语音提示语内容 */
{
   sprintf(fname, "P%d.SP",no); /* 产生语音提示语文件名 */
   if(!SP_load_voice(sppro,  LEN, &pro_len, fname)) /* 加载语音数据文
件 */
    { puts("Cannot find prompt file"); exit(1); }
   SP_sp_out(sppro, 0, pro_len, LEN, 8000); /* 播放出语音数据内容 */
}
/*-------------------*/
say_db(char no) /* 播放出某一语音参考样本内容 */
{
   sprintf(fname, "DBSP%d.SP",no); /* 产生语音参考样本语音文件名 */
if(!SP_load_voice(sppro, LEN, &pro_len, fname)) /* 加载语音数据文件 */
    { puts("Cannot find prompt file"); exit(1); }
   SP_sp_out(sppro, 0, pro_len, LEN, 8000); /* 播放出语音数据内容 */
}
/*----------------------*/
action(char no) /* 依据识别结果进行电话拨号 */
{
  switch(no)
   {
    case 0 : send("ATDT117");    break;
```

```
355     case 1 : send("ATDT2260258"); break;
356     case 2 : send("ATDT2234567"); break;
357     case 3 : send("ATDT3731234"); break;
358     case 4 : send("ATDT3879876"); break;
359     default : break;
360    }
361 }
362 /*-------------------------------------------------*/
363 tx(unsigned char c) /* 由RS232接口送出字符 */
364 {
365   bioscom(TX, c, port);
366 }
367 /*-----------------*/
368 send(char *str) /* 由RS232接口送出字符串 */
369 {
370 int i;
371   i=0;
372   do{ tx(str[i]); i++; }
373   while(str[i]!='\0');
374   tx(0x0d);
375 }
376 /*-------------------*/
377 reset_modem()  /* 初始化调制解调器 */
378 {
379  send("ATZ");
380 }
381 /*-----------------*/
```

习　题

1．MODEM 电话拨号测试程序TE.C，是使用COM2 通讯端口 2 来控制MODEM，修改原始程序，使得可以使用COM1 通讯端口 1 来控制MODEM做电话拨号。执行方式为：

TE　1　→ 通讯端口 1 来控制MODEM做电话拨号

TE　2　→ 通讯端口 2 来控制MODEM做电话拨号

2．修改MODEM电话拨号测试程序TE.C，可以线上输入电话号码而做电话拨号。

3．声控电话拨号程序VTE.EXE的语音参考样本文件名为VTE.DAT，修改原始程序，使得每一个不同使用者，可以自行加载属于自己的语音参考样本文件。

第 11 章　声控门禁设计

门禁管制是保卫安全措施中重要的一项措施，传统的控制是使用钥匙，或是使用红外线，无线电遥控器，及使用刷卡的形式来开启大门。在结合新的科技后有人提出指纹对比，人体面部影像对比及声控门禁。传统的控制方式必须随身携带钥匙或卡片，而后者不必携带任何东西，但是先前须要“注册”，使控制系统了解您个人的指纹，面部影像或是声音特征。

本章将以声控链接库来设计简易的声控门禁系统，本系统可以用声音来分辨是否为使用者本人，如果是的话，才放行，基本上这是一个结合特定语者辨识及声控密码辨识的例子，别人如果要通过门禁，必需要“模仿”您的声音，并破解您的个人声音密码。当然系统本身还可以做很多的实验。

11.1　声控门禁基本功能

声控门禁系统，其功能是只要说出原先设定的 2 个语句 (连续语音)，计算机便可以分辨出您的身份，计算机会以语音来引导您如何操作，并响应您识别的结果。原先所设定要识别的 2 个语句存盘为：语音文件 DBSP0.SP 的内容为 COM1，语音文件 DBSP1.SP 的内容为 COM2。

原系统设定 COM1 内容为“芝麻开门”，COM2 内容为“543”，使用者可以自行更改。使用者第一次要说出“芝麻开门”，第二次则说出“543”，才能通过此声控门禁系统。系统会自动识别出您所说出的那一个单音，以语音响应出识别结果看看是否您本人，如果您所说的那一个单音并不存在语音参考文件中，则系统会告知“无法识别”。例如使用者说出“HELLO”，则系统会响应您“无法识别”。

本系统的使用注意事项：

- ✧ 所设定的 2 个语句必须是连续语音，中间不可有间断音。
- ✧ 识别时 2 个连续语音的顺序必须一致，才可通过测试。

11.2　系 统 组 成

整个声控门禁系统组成如下：

- ✧ PC 语音卡。
- ✧ 有方向性动圈式麦克风一支。
- ✧ 4Ω 喇叭一支或连至扩音机。
- ✧ 控制程序 VDO.EXE 及相关程序文件。

相关程序文件说明如下：

① VDO.C：声控门禁 C 语言控制程序。
② VDO.DAT：语音参考样本文件。
③ VDO.EXE：声控门禁执行文件。
④ VDO.PRJ：声控门禁 TURBO C 目标文件。
⑤ DBSP0.SP、DBSP1.SP：语音参考样本语音数据文件。
⑥ EGAVGA.BGI：TURBO C 彩色屏幕驱动程序。
⑦ HERC.BGI：TURBO C 单色屏幕驱动程序。
⑧ SP.H：TURBO C 声控链接库头文件声明。
⑨ SP.LIB：TURBO C 声控链接库演示版。
⑩ P0.SP--P4.SP：语音提示语语音数据文件。

11.3 执 行 例 子

执行声控门禁控制程序 VDO.EXE，工作画面出现：

```
VOICE CONTROL SECURITY ...
Say the correct commands to pass...
 c    --> create DATA BASE
 l    --> listen DATA BASE
 SPACE -->  SPEECH recog ...
Select ?
```

声控门禁系统共提供有以下的一些控制指令键：

✧ 按键“c”：产生并录制参考样本语音数据文件。
✧ 按键“l”：聆听原设定所要识别的语音数据内容，可以听见 2 个语句。
✧ 空格键　：进行语音识别。

一般操作步骤如下：

步骤 1：按按键“1”，听一下计算机会识别那 2 个语句。

步骤 2：按空格键，直接进行语音识别，系统会哔一声提示您输语音，此时可以对着麦克风说出测试语句，则计算机会进行语音识别对比，例如使用者说出“HELLO”，则系统会响应您“无法识别”。执行结果如下：

```
Say the correct commands to pass.....
Please say    first   voice command ....NOT
NO GO !!!!!!!!
```

步骤 3：例如使用者说出的第 1 句及第 2 句语音与原先设定的一样，则系统会响应您通过测试的信息。执行结果如下：

```
Say the correct commands to pass...
Please say    first   voice command ... COM1
Please say    second voice command ... COM2
VOICE COMMAND OK  !!!       PASS   !!!
```

步骤 4：对于另外一个使用者要进行识别时，或是想修改语音识别的内容时，可以按按键“c”，产生并录制新的参考样本语音数据文件。产生新的参考样本数据文件的操作说明同样参考 8.4 节的说明。

步骤 5：找几位朋友来测试，随意说出两句语音，测试系统的识别特性，或是请朋友模仿自己的声音看看系统能不能接受。

11.4 程 序 设 计

声控门禁原始控制程序为 VDO.C，TURBO C 目标文件名为 VDO.PRJ，其主要语音识别子程序是使用声控链接库，详细的说明请参考第 8 章语音卡声控链接库使用。各主要控制子程序说明如下：

- ✧ menu()　　：显示将要识别的语音内容。
- ✧ recog()　　：语音识别。
- ✧ create_db()　　：产生并录制参考样本数据文件。
- ✧ be()　　：哔一声。
- ✧ sound_pass()　　：发出测试成功声响。
- ✧ sound_fail()　　：发出测试失败声响。
- ✧ prompt()　　：听取所要识别的语音数据内容。
- ✧ say_pro(char no)：播放出某一语音提示语内容。
- ✧ say_db(char no)：播放某一语音参考样本内容。

在识别出相对的失真度后，会检查其失真度大小，如果失真度太大了，大过某一临界值则为未知的语音输入便告知“无法识别”，并发出测试失败的声响。如果失真度在临界范围内则告知您说出了哪一个句子，但是若该句子的顺序不对，也是测试失败，于是发出测试失败的声响。一旦失真度很小同时 2 个句子的顺序都对，则是测试成功，系统会发出测试成功的声响，并显示讯息“VOICE COMMAND OK !!!　　PASS　!!!”。

程序清单如下：

```
1   /*************************************************
2   * Program: VDO.C                                 *
3   * Description: PC SPEECH CARD use SP_LIB example.*
4   *         VOICE COMMAND SECURITY                 *
5   * Copyright (C) VICTOR SPEECH LAB.  1996, 1997   *
6   *************************************************/
7
8   /* c0: COM1  所要识别的句子 1
9      c1: COM2  所要识别的句子 2
10     c2: XXX
11     c3: XXX
12     c4: XXX
13  */
```

```
#include <graphics.h>
#include <alloc.h>
#include <dos.h>
#include <stdio.h>
#include "sp.h"

/* 语音卡 I/O 口基地址 */
#define  SP_PORT  0x250

#define   LEN 32000
#define   char     signed char

#define   DB_FILE  "VDO.DAT"
/* SP.LIB 演示版一次最多 5 组进行识别 */
#define   MAX_DB   5
/* 语音提示语内容指针 */
#define PSAY   0  /* 请说      */
#define YOUPRO 1  /* 您可以说  */
#define ANO    2  /* 无法识别  */
#define YOUSAY 3  /* 您说      */
#define VTRAIN 4  /* 语音训练  */
#define REJECT 80.0   /* 无法识别的失真度临界值 */

char  *sp, *sppro;        /* 语音数据的数组指针 */
unsigned sp_len, pro_len; /* 语音数据长度 */

float  (*DB)[FRAME][LEVEL+1]; /* 参考样本特征参数数组 */
float  test_frame[MAX_FRAME][LEVEL+1]; /*  测试样本特征参数数组 */

unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;
int gd=DETECT, gm;

char fname[80];

FILE *in, *out;
int vc_no=5; /* 所要识别的单音数 */

char string[80];
char *vc_note[]={"COM1", "COM2", "XXX", "XXX",  "XXX"};
/*----------------*/
main()
```

```
{
char ch;

 if(!SP_init_sp(SP_PORT))  /* 初始化语音卡 I/O 口 */
  { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }

 DB=farcalloc(MAX_DB, FRAME*(LEVEL+1)*4); /* 动态配置参考样本缓冲区 */
   if(DB==NULL) { printf("%c out of memory !\n", 7); exit(1);}

 sp=(char  *)calloc(LEN ,sizeof(char)); /* 动态配置语音数据缓冲区 */
 if (sp==NULL)     { printf("%c out of memory !\n", 7); exit(1);}

 sppro=(char  *)calloc(LEN ,sizeof(char));  /* 动态配置语音提示语缓冲区
*/
 if (sppro==NULL)  { printf("%c out of memory !\n", 7); exit(1);}

 if(SP_load_db(DB, vc_no, DB_FILE)) /* 加载参考样本数据文件 */
   puts("load DATA BASE …");
 else  { printf("%c NO %s  DATA BASE !\n", 7, DB_FILE); getch(); }

while(1) /* 无穷循环 */
 {
  clrscr(); /* 显示工作画面 */
  puts("VOICE CONTROL SECURITY …");
  puts("Say the correct commands to pass…");

  puts(" c   --> create DATA BASE");
  puts(" l   --> listen DATA BASE");
  puts(" SPACE -->  SPEECH recog …");
  puts("Select ? ");

  ch=getch(); /* 等待按键 */
  switch(ch)
    {
      case 27 : exit(0); /* 若按下 ESC 键则结束程序执行 */
      case 'c': create_db(); break; /* 产生并录制参考样本数据文件 */
      case 'l': prompt();    break; /* 听取所要识别的语音数据内容 */
      case ' ': recog();     break; /* 语音识别 */
      default : break;
    }
 }
  free(sp);    /* 释放动态配置的语音数据缓冲区 */
```

```
97      free(sppro); /* 释放动态配置的语音提示语数据缓冲区 */
98      farfree(DB); /* 释放动态配置的参考样本缓冲区 */
99    }
100   /*------------------------------*/
101   menu()   /* 显示欲识别的语音内容 */
102   {
103    puts("0 --> COM1 ");
104    puts("1 --> COM2 ");
105    puts("2 --> XXX  ");
106    puts("3 --> XXX  ");
107    puts("4 --> XXX  ");
108   }
109   /*------------------*/
110   recog()   /* 语音识别 */
111   {
112   int ans, i;
113   int exit_f, frames;
114   float dist[10];
115   char p,an[2];
116
117    an[0]=an[1]=0;
118    clrscr();
119    puts("Say the correct commands to pass…");
120
121     p=0;
122     while(1) /* 无穷循环 */
123      {
124   again:
125       if(p==0)
126        printf("\nPlease say   first  voice command …");
127       else
128        printf("\nPlease say   second voice command …");
129       be();     /* 哔一声 */
130       exit_f=0; /* 清除脱离标志位 */
131   /* 语音输入自动侦测并检查是否有按键 */
132       sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
133       if(exit_f) /* 若有按键则跳离循环 */
134        { puts("USER BREAK !"); return; }
135   /* 语音切割 */
136       printf(".");
137       segs=SP_separate(sp, sp_len, bp, ep);
138       if(segs==0) goto again; /* 若无有效语音分段则重新做语音输入 */
```

```
/* 语音信号特征参数分析 */
    if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
     { printf("%c cannot take feature ! !\n", 7); exit(1); }
/* 语音识别对比 */
    frames=16;
    ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
    if(dist[ans]>=REJECT) /* 失真度大于临界值则告知无法识别 */
      {
       printf("NOT ");
       say_pro(ANO);
       sound_fail();
       return;
      }

     else
      {
/* 显示识别结果 */
       printf(" %s ", vc_note[ans]);
/* prompt */
       say_pro(YOUSAY);
       say_db(ans); /* 播放出该组语音内容 */
       if(ans!=p) { sound_fail(); return; }
       an[p]=ans;
       p++;
       if(an[0]==0 && an[1]==1)
         { sound_pass(); delay(2000); return; }
     }
   }/* end while */
}
/*-------------------*/
create_db()  /* 产生并录制参考样本数据文件 */
{
int i, exit_f,no;
char c;
int ans, frames;
float dist[10];
char mess[80];

  clrscr();
  puts("Create DOOR SECURITY data base ");
  menu();  /* 显示将要识别的语音内容 */

```

```
181    say_pro(VTRAIN); /* 发出“语音训练”提示语 */
182   for(no=0; no<2; no++)
183    {
184  again:
185     printf("Please say %d command <  %s >\n", no, vc_note[no]);
186  /* 取得语音数据..... */
187     be();     /* 哔一声 */
188     exit_f=0; /* 清除脱离标志位 */
189  /* 语音输入自动侦测并检查是否有按键按下*/
190     sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
191     segs=SP_separate(sp, sp_len, bp, ep); /* 语音切割 */
192  /* 若无有效语音分段则重新做语音输入 */
193     if(segs==0) goto again;
194     initgraph(&gd, &gm, ""); /* 初始化图形接口 */
195  /* 显示语音信号波形 */
196     SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
197           0.6, bp, ep, segs);
198  /* 播放出该段语音内容 */
199     SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
200  /* 语音信号特征参数分析 */
201     if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
202      { printf("%c cannot take feature ! !\n", 7); exit(1); }
203
204  /* 语音识别对比测试 */
205     frames=16;
206     ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
207  /* 显示识别对比测试结果 */
208     sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
209           vc_note[ans], dist[ans]);
210     outtextxy(10, getmaxy()-80, mess);
211  /* 显示操作信息 */
212     outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
213     outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
214
215  /* 等待按键处理 */
216     c=getch();
217     if(c==0x1b)  {  closegraph(); return; }
218     if(c==0x0d)  {  closegraph(); goto next; }
219     if(c!='y')   {  closegraph(); goto again; }
220       else closegraph();
221  next:;
222  /* 原始语音数据存盘 */
```

```
223    sprintf(fname, "DBSP%d.SP",no);
224
225    if(!SP_save_voice1(sp, bp[0], ep[0], fname))
226     { puts("Cannot save prompt file "); exit(1);  }
227
228  /* 语音信号特征参数分析 */
229    if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
230     { printf("%c cannot take feature ! !\n", 7); exit(1); }
231   }/* end of no */
232
233  /* 存为语音参考样本文件 */
234    puts("Saving data base ......");
235    if( SP_save_db(DB, vc_no, DB_FILE) )
236      { puts("Save data base  OK !");   delay(2000); }
237    else { printf("%c cannot save data base !\n", 7); exit(1); }
238  /* 说出所有识别的语音数据内容 */
239   prompt();
240  }
241  /*--------------------*/
242  be() /* 哔一声 */
243  {
244    sound(500);
245    delay(100);
246    nosound();
247  }
248  /*---------------*/
249  prompt() /* 听取所要识别的语音数据内容 */
250  {
251  int no;
252
253   puts("Say data base .....");
254   say_pro(YOUPRO);
255   for(no=0; no<2; no++)
256    {
257     say_db(no); /* 说出某一参考样本语音 */
258     delay(500);
259    }
260  }
261  /*--------------------*/
262  say_pro(char no) /* 播放出某一语音提示语内容 */
263  {
264     sprintf(fname, "P%d.SP",no); /* 产生语音提示语文件名 */
```

```
265 if(!SP_load_voice(sppro,  LEN, &pro_len, fname)) /* 加载语音数据文件 */
266     { puts("Cannot find prompt file"); exit(1); }
267    SP_sp_out(sppro, 0, pro_len, LEN, 8000); /* 播放出语音数据内容 */
268 }
269 /*------------------*/
270 say_db(char no) /* 播放出某一语音参考样本内容 */
271 {
272    sprintf(fname, "DBSP%d.SP",no); /* 产生语音参考样本语音文件名 */
273 if(!SP_load_voice(sppro,  LEN, &pro_len, fname)) /* 加载语音数据文件 */
274     { puts("Cannot find prompt file"); exit(1);  }
275    SP_sp_out(sppro, 0, pro_len, LEN, 8000); /* 播放出语音数据内容 */
276 }
277 /*----------------------*/
278 sound_pass()  /* 发出测试成功声响*/
279 {
280 int i;
281
282  puts("\nVOICE COMMAND OK !!!    PASS  !!!");
283  for(i=500; i<2000; i+=100)
284   {
285    sound(i);
286    delay(300);
287   }
288  nosound();  delay(1000);
289
290  for(i=2000; i>0; i-=100)
291   {
292    sound(i);
293    delay(500);
294   }
295  nosound();
296 }
297 /*--------------*/
298 sound_fail() /* 发出测试失败声响*/
299 {
300 char i;
301
302  puts("\nNO GO !!!!!!!!!");
303  for(i=0; i<5; i++)
304   { be(); delay(500); }
305 }
```

习　题

1. 列举门禁管制控制的 4 种方式。

2. 声控门禁控制程序 VDO.EXE 的语音参考样本文件名为 VDO.DA，修改原始程序，使得每一个不同使用者，可以自行加载属于自己的语音参考样本文件。

第 12 章　声控家电的控制系统

本章将以声控链接库来设计声控家电控制系统，本系统可以直接用声音来控制家电或一般电源开关的开启或关闭，不用动到开关直接以声音来控制家电的开关，这是一个简单却很实用有趣的家电声控专题。在本章中除了可以学习声控程序的设计外，还可以了解语音卡如何做I/O的控制，并了解如何以继电器来控制市电或直流电源。

12.1　基 本 功 能

声控家电系统，其功能是用说话可以控制计算机来开启或关闭一般家电电源，计算机会以语音来引导您如何操作，并响应您识别的结果。原先所设定要识别的 5 个单音(以中文发音)及语音存盘如表 12-1。

表 12-1　语音文件对应的声音内容

语音文件名	中文
DBSP0.SP	电灯
DBSP1.SP	电扇
DBSP2.SP	音响
DBSP3.SP	收音机
DBSP4.SP	电视

表 12-2　声音内容所对应的执行动作

内容	动　作
电灯	电灯开启或关闭
电扇	电扇开启或关闭
音响	音响开启或关闭
收音机	收音机开启或关闭
电视	电视开启或关闭

使用者只要说出以上的几个单音，系统会自动识别出您所说出的那一个单音，以语音响应出识别结果并执行相对的动作，如果您所说的那一个单音并不存在语音参考文件中，则系统会告知“无法识别”。所执行的相对动作如表 12-2。

本系统的使用注意事项：

- ✧ 本系统为特定语者 5 个单音或词的语音识别系统。
- ✧ 使用自己的语音参考样本可以获得很高的辨识效果。
- ✧ I/O线直接控制继电器，可以控制市电或直流电源开关。
- ✧ 本系统只能控制家电产品的开启或关闭，无法进一步做其他遥控的动作。

12.2　系 统 组 成

整个声控指令系统组成如下：

- ✧ PC语音卡。
- ✧ 有方向性动圈式麦克风一支。
- ✧ 4Ω喇叭一支或连至扩音机。

✧ 控制程序EL.EXE 及相关程序文件。
✧ 16PIN排线一条。
✧ I/O控制实验板一片(自行制作)。

相关程序文件说明如下：

✧ EL.C ：声控家电C语言原始控制程序。
✧ EL.DAT ：语音参考样本文件。
✧ EL.EXE ：声控家电执行文件。
✧ EL.PRJ ：声控家电TURBO C目标文件。
✧ DBSP0.SP--DBSP4.SP ：语音参考样本语音数据文件。
✧ EGAVGA.BGI ：TURBO C彩色屏幕驱动程序。
✧ HERC.BGI ：TURBO C单色屏幕驱动程序。
✧ SP.H ：TURBO C声控链接库头文件声明。
✧ SP.LIB ：TURBO C声控链接库演示版。
✧ 1P0.SP--P4.SP ：语音提示语语音数据文件。

12.3 执 行 例 子

执行控制程序EL.EXE，工作画面出现:

```
LIGHT/FAN CONTROL   ...
Copyright (C) VICTOR SPEECH LAB.   1996, 1997
 c    --> create DATA BASE
 l    --> listen DATA BASE
 m    --> modify DATA BASE
 SPACE -->   SPEECH recog ...
Select ?

    LIGHT   : OFF
    FAN     : OFF
    AMP     : OFF
    RADIO   : OFF
    TV      : OFF
```

画面上显示出控制指令键及目前所控制的 5 组 I/O 线。目前的开启状态 ON 表示开，OFF 表示关，原设定为全部 OFF。声控家电系统共提供有以下的一些控制指令键:

✧ 按键“c” ：产生并录制参考样本语音数据文件。
✧ 按键“l” ：听原设定所要识别的语音数据内容，可以听见 5 个可识别的单音。
✧ 按键“m” ：修改参考样本语音数据文件。
✧ 空格键 ：进行语音识别。

操作说明如下：

1. 按按键“1”，听一下计算机会识别那几个单音。

2. 按空格键，直接进行语音识别，系统会哔一声提示您输语音，此时可以对着麦克风说出“电灯”，则计算机会进行语音识别对比，将结果说出来，如果识别正确则电灯的控制线会开启，并显示“LIGHT : ON”；再次说出“电灯”，则电灯的控制线会关闭，并显示“LIGHT : OFF”。当电灯连至I/O控制板时则可以用声音来开关电灯。若在识别时，按下任何按键可以结束语音识别动作，返回工作画面。

3. 如果计算机识别出“电扇”，可以控制电扇的开启。

4. 如果计算机识别出“音响”，可以控制音响的开启。

5. 如果计算机识别出“收音机”，可以控制收音机的开启。

6. 如果计算机识别出“电视”，可以控制电视的开启。

7. 如果识别率不好，则可以按键“m”，修改某一组参考样本语音数据文件。修改参考样本的操作说明可参考 8.4 节语音卡声控链接库应用范例程序说明。

8. 对于另外一个使用者要进行识别时或是想修改语音识别的内容时，可以按按键“c”，产生并录制新的参考样本语音数据文件。产生新的参考样本数据文件的操作说明请参考 8.4 节的说明。

12.4 系 统 设 计

声控家电原始控制程序为EL.C，TURBO C目标文件名为EL.PRJ，其主要语音识别子程序是使用声控链接库，详细的说明请参考第 8 章语音卡声控链接库使用。各主要控制子程序说明如下：

- ✧ menu() ：显示将要识别的语音内容。
- ✧ modify_db() ：修改参考样本语音数据文件。
- ✧ recog() ：语音识别。
- ✧ create_db() ：产生并录制参考样本数据文件。
- ✧ be() ：哔一声。
- ✧ prompt() ：听取所要识别的语音数据内容。
- ✧ show_status() ：显示I/O线动作状态。
- ✧ say_pro(char no) ：播放出某一语音提示语内容。
- ✧ say_db(char no) ：播放出某一语音参考样本内容。
- ✧ action(char no) ：依据识别结果产生相应的动作。

在识别出某一单音编号 no 后，由动作执行子程序 action( )控制以便执行相应的指令，在此使用语音卡上的 8255 做一般 I/O 的输出控制，由 8255 的端口 C 经由 20 PIN 排线，连至自行制作的 I/O 控制实验板，来做家电的电源控制实验。8255 的端口 C 所控制的 I/O 线分配如表 12-3。

控制信号是高电位动作，例如将 PC0 设为 1，则电灯将点亮，将 PC0 设为 0，则电灯将熄灭。图 12-1 为 I/O 控制实验板上 8255 PC0 控制继电器 ON/OFF 的电路，相同的电路做 5 组便可以控制 5 组家电的开关，分别由 8255 PC0 至 PC4 来做控制。

表 12-3　8255 的端口 C 所控制的 I/O 线分配表

8255 端口 C	控制线
PC0	电灯
PC1	电扇
PC2	音响
PC3	收音机
PC4	电视

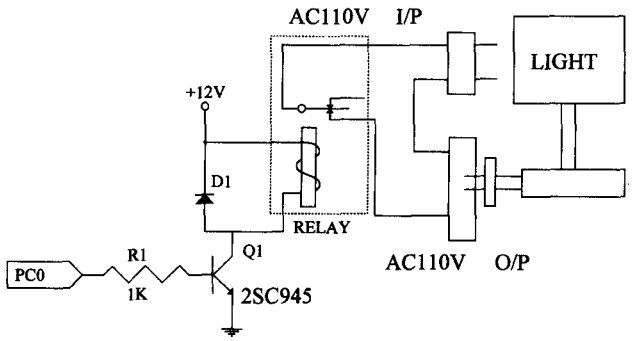


图 12-1　I/O 控制实验板电路

12.4.1　继电器如何连接

在实验板上连接有 5 只继电器，如何连接出来做控制呢？首先我们介绍一下继电器的工作接点，如图 12-2，LINE接脚表示直流控制信号送入接点，其余 3 个控制接点说明如下：

- ✧ NC　：Normal Close，常闭点。以COM为共同点，NC与COM在平时是呈导通的状态。
- ✧ COM ：Common，共通点。输出控制接点的共同接点。
- ✧ NO　：Normal Open常开点。NO与COM在平时是呈开路的状态，当继电器动作时，NO与COM导通，NC与COM则呈开路（不导通）状态。

因此继电器是串联到回路中当作开关使用从而控制电灯，当继电器导通时使市电AC 110V进入回路，电灯将被点亮。

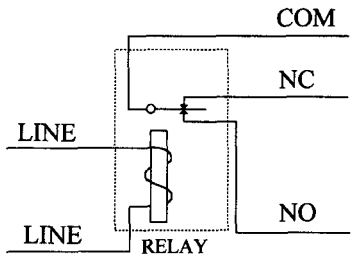


图 12-2　继电器的工作接点

12.4.2 8255 如何控制

系统在激活前必须对 8255 进行初始化设置，以设定其工作模式，将其设定为模式 0，一般I/O控制，端口A输入(A/D语音输入)，端口B输出(D/A语音输出)，端口C输出(I/O位控制)，其控制字为 0x90，可以如下的程序来完成设定。

```
#define  AD_PORT  0x250       /* 8255 PA  */
#define  IO       AD_PORT+2   /* 8255 PC  */
#define  CON      AD_PORT+3   /* 8255 CW  */

outportb(CON, 0x90); /* 初始化 8255 I/O 端口 */
```

至于端口C输出控制可以用位设定/清除的功能来完成。

例如：将PC0 设为 1，程序为：outportb(CON, 1); /* PC0=1 */ 则电灯将点亮。将PC0 设为 0，程序为：outportb(CON, 0); /* PC0=0 */ 则电灯将熄灭。

有关 8255 进一步的控制使用说明，请参考 4.5 节 8255 功能说明。

程序清单如下：

```
1   /*************************************************
2   * Program: EL.C                                  *
3   * Description:PC SPEECH CARD use SP_LIB example.*
4   *           Electric device control              *
5   * Copyright (C) VICTOR SPEECH LAB.  1996, 1997  *
6   *************************************************/
7   /* c0: LIGHT  所要识别的单音
8      c1: FAN
9      c2: AMP
10     c3: RADIO
11     c4: TV
12  */
13  #include <graphics.h>
14  #include <alloc.h>
15  #include <dos.h>
16  #include <stdio.h>
17  #include "sp.h"
18
19  /* 语音卡 I/O 口基地址 */
20  #define  AD_PORT  0x250       /* 8255 PA */
21  #define  IO       AD_PORT+2   /* 8255 PC */
22  #define  CON      AD_PORT+3   /* 8255 CW */
23
24  #define   LEN 32000
```

```
#define   char     signed char

#define   DB_FILE  "EL.DAT"
#define   MAX_DB   5
/* SP.LIB 演示版一次最多 5 组进行识别 */
/* 语音提示语内容指针 */
#define PSAY   0  /* 请说      */
#define YOUPRO 1  /* 您可以说  */
#define ANO    2  /* 无法识别  */
#define YOUSAY 3  /* 您说      */
#define VTRAIN 4  /* 语音训练  */
#define REJECT 80.0 /* 无法识别的失真度临界值 */

char  *sp, *sppro;    /* 语音数据的数组指针 */
unsigned sp_len, pro_len; /* 语音数据长度 */

float  (*DB)[FRAME][LEVEL+1];  /* 参考样本特征参数数组 */
float  test_frame[MAX_FRAME][LEVEL+1]; /* 测试样本特征参数数组 */

unsigned bp[10], ep[10]; /* 数组储存语音段断点位置 */
int segs;
int gd=DETECT, gm;

char fname[80];

FILE *in, *out;
int vc_no=5;    /* 所要识别的单音数 */

char string[80];
char *vc_note[]=_7b"LIGHT", "FAN", "AMP", "RADIO",  "TV"};
char *co="Copyright (C) VICTOR SPEECH LAB.  1996, 1997";
char b0,b1,b2,b3,b4 ;      /* 5 条 I/O 线动作标志位 */
/*----------------*/
main()
{
char ch;

 if(!SP_init_sp(0x250)) /* 检查语音卡是否存在 */
  { puts("CANNOT FIND SPEECH CARD !"); getch(); exit(1); }

 outportb(CON, 0x90); /* 初始化 8255 I/O 口 */
 outportb(IO, 0); /* I/O 线输出低电位 */
```

```
67
68   b0=b1=b2=b3=b4=0; /* I/O 线动作标志位全部清除 */
69
70   DB=farcalloc(MAX_DB, FRAME*(LEVEL+1)*4); /* 动态配置参考样本缓冲区 */
71     if(DB==NULL) { printf("%c out of memory !\n", 7); exit(1);}
72
73   sp=(char  *)calloc(LEN ,sizeof(char));/* 动态配置语音数据缓冲区 */
74   if (sp==NULL) { printf("%c out of memory !\n", 7); exit(1);}
75
76   /* 动态配置语音提示语缓冲区 */
77   sppro=(char  *)calloc(LEN ,sizeof(char));
78   if (sppro==NULL)  { printf("%c out of memory !\n", 7); exit(1);}
79
80   if(SP_load_db(DB, vc_no, DB_FILE)) /* 加载参考样本数据文件 */
81     puts("load DATA BASE .......");
82   else  { printf("%c NO %s  DATA BASE !\n", 7, DB_FILE); getch(); }
83
84  while(1) /* 无穷循环 */
85   {
86    clrscr();  /* 显示工作画面 */
87    puts("LIGHT/FAN CONTROL   .......");
88    puts(co);
89    puts(" c   --> create DATA BASE");
90    puts(" l   --> listen DATA BASE");
91    puts(" m   --> modify DATA BASE");
92    puts(" SPACE -->  SPEECH recog ....");
93    puts("Select ? ");
94    show_status(); /* 显示 I/O 动作线状态 */
95
96    ch=getch();  /* 等待按键 */
97    switch(ch)
98      {
99       case 27 : exit(0); /* 若按下 ESC 键则结束程序执行 */
100      case 'c': create_db(); break; /* 产生并录制参考样本数据文件 */
101      case 'l': prompt();    break; /* 听取所要识别的语音数据内容 */
102      case 'm': modify_db(); break; /* 修改参考样本语音数据文件 */
103      case ' ': recog();     break; /* 语音识别 */
104      default : break;
105     }
106   }
107    free(sp);    /* 释放动态配置的语音数据缓冲区 */
108    free(sppro); /* 释放动态配置的语音提示语数据缓冲区 */
```

```
109     farfree(DB); /* 释放动态配置的参考样本缓冲区 */
110  }
111  /*------------------------------*/
112  menu() /* 显示将要识别的语音内容 */
113  {
114   puts("0 --> LIGHT");
115   puts("1 --> FAN  ");
116   puts("2 --> AMP  ");
117   puts("3 --> RADIO");
118   puts("4 --> TV   ");
119  }
120  /*--------------*/
121  modify_db()    /* 修改参考样本语音数据文件 */
122  {
123  int i, exit_f,no, ans, frames;
124  char c, mess[80];
125  float dist[10];
126
127    menu();
128  /* 输入修改参考样本编号 */
129    puts("=== modify word no ===");
130    do{printf("modify which no ");
131    scanf("%d", &no);
132    } while( no>=vc_no);/* 若编号无效则重新输入编号 */
133
134  again:
135  /* 发出“请说”提示语 */
136    say_pro(PSAY);
137    printf("please say <  %s >\n", vc_note[no]);
138    say_db(no);/* 播放出所要修改的该组语音内容 */
139
140  /* 取得语音数据..... */
141    be();     /* 哔一声 */
142    exit_f=0; /* 清除脱离标志位 */
143  /* 语音输入自动侦测并检查是否有按键 */
144    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
145  /* 语音切割 */
146     segs=SP_separate(sp, sp_len, bp, ep);
147  /* 若无有效语音分段则重新做语音输入 */
148     if(segs==0) goto again;
149     initgraph(&gd, &gm, ""); /* 初始化图形接口 */
150     SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
```

```
151           0.6, bp, ep, segs); /* 显示语音信号波形 */
152  /* 播放出该段语音内容 */
153    SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
154  /* 语音信号特征参数分析 */
155    if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
156     { printf("%c cannot take feature ! !\n", 7); exit(1); }
157
158  /* 语音识别对比测试 */
159    frames=16;
160    ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
161  /* 显示识别对比测试结果 */
162    sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
163           vc_note[ans], dist[ans]);
164    outtextxy(10, getmaxy()-80, mess);
165  /* 显示操作信息 */
166    outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
167    outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
168
169  /* 等待按键处理 */
170    c=getch();
171
172    if(c==0x1b)     {  closegraph(); return; }
173    if(c==0x0d)     {  closegraph(); goto next; }
174    if(c!='y')   {  closegraph(); goto again; }
175      else closegraph();
176
177  next:;
178  /* 语音信号特征参数分析 */
179    if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
180     { printf("%c cannot take feature ! !\n", 7); exit(1); }
181
182  /* 存为语音参考样本文件 */
183    puts("Saving data base ......");
184    if( SP_save_db(DB, vc_no, DB_FILE) )
185    { printf("Modify data base  <<<%d  %s  >>> ok ! ", no, vc_note[no]);
186      delay(2000); }
187    else { printf("%c cannot save data base !\n", 7); exit(1); }
188  }
189  /*--------------------*/
190  recog() /* 语音识别 */
191  {
192  int ans, i;
```

```
int exit_f, frames;
float dist[10];

 clrscr();
 puts("LIGHT/FAN CONTROL   .......");
 puts(co);
 printf("recog.......");
 show_status();

while(1)  /* 无穷循环 */
 {
again:
  be();     /* 哔一声 */
  exit_f=0; /* 清除脱离标志位 */
   /* 语音输入自动侦测并检查是否有按键 */
  sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
  if(exit_f) break; /* 若有按键则跳离循环 */
  gotoxy(10, 18); printf("        ");
  gotoxy(10, 18); printf(".");
/* sound detect ......*/
  segs=SP_separate(sp, sp_len, bp, ep); /* 语音切割 */
/* 若无有效语音分段则重新做语音输入 */
  if(segs==0) goto again;
/* 语音信号特征参数分析 */
  if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
     { printf("%c cannot take feature ! !\n", 7); exit(1); }
/* 语音识别对比 */
  frames=16;
  ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
  if(dist[ans]>=REJECT) /* 失真度大于临界值则告知无法识别 */
    { printf("NOT ");  say_pro(ANO); }
   else
    {  /* 显示识别结果 */
     printf(" %s ", vc_note[ans]);
/* 播放出识别结果 */
     say_pro(YOUSAY);
     action(ans);   /* 执行动作 */
     show_status(); /* 显示 I/O 线状态 */
     say_db(ans);   /* 播放出该组语音内容 */
    }

/* 列出语音识别的失真度
```

```
235   for(i=0; i<vc_no; i++)
236    {
237     if(ans==i) printf(">");
238     printf("%3.0f   ",dist[i]);
239    }
240   puts("");
241  */
242   }/* 循环 */
243  }
244  /*--------------------*/
245  create_db()  /* 产生并录制参考样本数据文件 */
246  {
247  int i, exit_f,no;
248  char c;
249  int ans, frames;
250  float dist[10];
251  char mess[80];
252
253   clrscr();
254   puts("Create data base ");
255   menu();  /* 显示将要识别的语音内容 */
256   say_pro(VTRAIN);  /* 发出“语音训练”提示语 */
257
258  for(no=0; no<vc_no; no++)
259   {
260  again:
261    printf("please say <  %s >\n", vc_note[no]);
262  /* 取得语音数据..... */
263    be();       /* 哔一声 */
264    exit_f=0;   /* 清除脱离标志位 */
265  /* 语音输入自动侦测并检查是否有按键 */
266    sp_len=SP_auto_mic_ip(sp, LEN, &exit_f, 500, 400, 8000);
267  /* 语音切割 */
268    segs=SP_separate(sp, sp_len, bp, ep);
269  /* 若无有效语音分段则重新做语音输入 */
270    if(segs==0) goto again;
271  /* 初始化图形接口 */
272    initgraph(&gd, &gm, "");
273  /* 显示语音信号波形 */
274    SP_show_wave(sp, 0, sp_len, 0, 0, getmaxx(), getmaxy(),
275          0.6, bp, ep, segs);
276    SP_sp_out(sp, bp[0], ep[0], LEN, 8000);
```

```
277 /* 语音信号特征参数分析 */
278    if(!SP_analysis(sp, bp[0], ep[0], test_frame) )
279     { printf("%c cannot take feature ! !\n", 7); exit(1); }
280 /* 语音识别对比测试 */
281    frames=16;
282    ans=SP_sp_match1(DB, vc_no, test_frame, frames, dist);
283 /* 显示识别对比测试结果 */
284    sprintf(mess, "Test recog : %s  dist.=%3.1f (Normal < 80.0)",
285            vc_note[ans], dist[ans]);
286    outtextxy(10, getmaxy()-80, mess);
287 /* 显示操作信息 */
288    outtextxy(10, getmaxy()-100, "<y> or <enter> to accept");
289    outtextxy(10, getmaxy()-120, "<ESC> to abort ! ");
290
291 /* 等待按键处理 */
292    c=getch();
293    if(c==0x1b)  {  closegraph(); return; }
294    if(c==0x0d)  {  closegraph(); goto next; }
295    if(c!='y')   {  closegraph(); goto again; }
296      else closegraph();
297 next:;
298 /* 原始语音数据存盘 */
299    sprintf(fname, "DBSP%d.SP",no);
300    if(!SP_save_voice1(sp, bp[0], ep[0], fname))
301     { puts("Cannot save prompt file "); exit(1); }
302
303 /* 语音信号特征参数分析 */
304    if(!SP_analysis(sp, bp[0], ep[0], DB[no]) )
305     { printf("%c cannot take feature ! !\n", 7); exit(1); }
306  }/* end of no */
307
308 /* 存为语音参考样本文件 */
309    puts("Saving data base ......");
310    if( SP_save_db(DB, vc_no, DB_FILE) )
311      { puts("Save data base  OK !");  delay(2000); }
312    else { printf("%c cannot save data base !\n", 7); exit(1); }
313 /* 播放出所有识别的语音数据内容 */
314  prompt();
315 }
316 /*-------------------*/
317 be()  /* 哔一声 */
318 {
```

```
319   sound(500);
320   delay(100);
321   nosound();
322  }
323  /*---------------*/
324  prompt() /* 听取所要识别的语音数据内容 */
325  {
326  int no;
327
328   printf("Say data base .....");
329   say_pro(YOUPRO);
330   for(no=0; no<vc_no; no++)
331    {
332     printf("%s ", vc_note[no]);
333     say_db(no);
334     delay(500);
335    }
336  }
337  /*--------------------*/
338  show_status() /* 显示I/O线状态  */
339  {
340   gotoxy(2,11); printf("LIGHT : ");
341   if(b0) printf("ON "); else  printf("OFF");
342
343   gotoxy(2,12); printf("FAN   : ");
344   if(b1) printf("ON "); else  printf("OFF");
345
346   gotoxy(2,13); printf("AMP   : ");
347   if(b2) printf("ON "); else  printf("OFF");
348
349   gotoxy(2,14); printf("RADIO : ");
350   if(b3) printf("ON "); else  printf("OFF");
351
352   gotoxy(2,15); printf("TV    : ");
353   if(b4) printf("ON "); else  printf("OFF");
354  }
355  /*---------------*/
356  action(int no)  /* 依据识别结果产生相对的动作 */
357  {
358  /*  printf("%d ", no); */
359   if(no==0) b0=1-b0;  /* 位反向处理 */
360   if(no==1) b1=1-b1;  /* 1变0, 0变1 */
361   if(no==2) b2=1-b2;
```

```
362   if(no==3) b3=1-b3;
363   if(no==4) b4=1-b4;
364
365  /* 8255端口C位控制格式：0xxx___0/1*/
366
367   if(b0)  outportb(CON, 1); /* PC0=1 */
368     else  outportb(CON, 0); /* PC0=0 */
369
370   if(b1)  outportb(CON, 3); /* PC1=1 */
371     else  outportb(CON, 2); /* PC1=0 */
372
373   if(b2)  outportb(CON, 5); /* PC2=1 */
374     else  outportb(CON, 4); /* PC2=0 */
375
376   if(b3)  outportb(CON, 7); /* PC3=1 */
377     else  outportb(CON, 6); /* PC3=0 */
378
379   if(b4)  outportb(CON, 9); /* PC4=1 */
380     else  outportb(CON, 8); /* PC4=0 */
381  }
382  /*---------------*/
383  say_pro(char no)  /* 播放出某一语音提示语内容 */
384  {
385     sprintf(fname, "P%d.SP",no);
386     if(!SP_load_voice(sppro,  LEN, &pro_len, fname))
387      { puts("Cannot find prompt file"); exit(1); }
388     SP_sp_out(sppro, 0, pro_len, LEN, 8000);
389  }
390  /*------------------*/
391  say_db(char no)  /* 播放出某一语音参考样本内容 */
392  {
393     sprintf(fname, "DBSP%d.SP",no);
394     if(!SP_load_voice(sppro,  LEN, &pro_len, fname))
395      { puts("Cannot find prompt file"); exit(1); }
396     SP_sp_out(sppro, 0, pro_len, LEN, 8000);
397  }
398  /*----------------------*/
```

习　题

原始声控指令程序EL.EXE的语音参考样本文件名为EL.DAT，修改原始程序，使得每一个不同使用者，可以自行加载属于自己的语音参考样本文件。

附　录

附录 A　TURBO C 简易使用说明

在进入TURBO C整合的操作窗口环境中，通常我们是完成一些基本的操作程序，便可以建立C原始程序并编译而执行程序。有哪些基本的操作项目呢？如下所列：

1. 编辑原始程序。
2. 加载PE2已编辑好的原始程序。
3. 存盘。
4. 编译C程序。
5. 执行C程序。
6. 以目标文件编译程序。

A.1　编辑原始程序

步骤1. 进入TURBO C并编辑原始C程序TEST.C，可以在DOS命令下打入

TC TEST <ENTER>

若TEST.C已存在则系统自动加载程序进入编辑功能，如果TEST.C不存在则是一个新文件。

步骤2. 在编辑上常用的功能键有以下几种：

- ✧ 上下左右方向键：4个方向光标的移动。
- ✧ Page Up　：使光标往上移动一页。
- ✧ Page Down　：使光标往下移动一页。
- ✧ Home　：移动光标至行首。
- ✧ End　：移动光标至行尾。
- ✧ Ctrl+Y　：删除光标所在的该行文字。
- ✧ Ctrl+K B　：先按住“Ctrl”接再按“K”键，而后放开，再按“B”键，可以设定区段的起始位置，准备做区段复制用。
- ✧ Ctrl+K K　：设定区段的结束位置。
- ✧ Ctrl-K Y　：删除整个区段。
- ✧ Ctrl+K C　：把整个区段内容拷贝到光标所在的位置。
- ✧ Ctrl+K V　：把整个区段内容移动到光标所在的位置。
- ✧ Ctrl+K R　：把一个文件内容读到目前光标所在的位置上。
- ✧ Ctrl+K W　：将目前设定的一个区段写入磁盘成为一个文件。

A.2 加载 PE2 已编辑好的原始程序

若您的C程序是以PE2做编辑则可以如下操作：
“ALT-F” → “L”→输入欲加载的程序文件名

A.3 存盘

按下功能键“F2”则做存盘动作，要写入新文件则可以按下“ALT-F”、“W”功能键。

A.4 编译 C 程序

按下“ALT-C”、“ENTER”则进行编译。

A.5 执行 C 程序

按下“ALT-R”、“ENTER”则执行程序。

A.6 以目标文件编译程序

按下“ALT+P”、“ENTER”可输入目标文件的文件名，其扩展名为*.PRJ，如果为TEST.PRJ，则产生的可执行文件为 TEST.EXE，其中 TEST.PRJ 内容如下：

- TEST1.C
- TEST2.C
- TEST3.C

只要一执行“ALT+R”，则系统会自动编译以上 3 支程序而产生可执行文件 TEST.EXE。以目标文件来整合程序可以使较复杂的大型程序得以有效率地进行编译，在编辑上较为方便而且迅速。

附录 B PC 语音卡快速安装使用说明

1. 准备动圈式麦克风，即唱卡拉OK的那一型麦克风一支，必要时可以经过转接头连至MIC接头输入。
2. 准备一小型喇叭，喇叭阻抗最好为 4Ω。
3. 关闭 PC 机电源。
4. 打开 PC 机机箱，选择一空插槽，小心插入语音卡，连上麦克风及喇叭线。
5. 打开 PC 机电源，PC 机应该可以正常执行一般程序。
6. 执行测式程序：SPR.EXE

在 PC 机上按下任何键则开始做语音录音，使用者可以对着麦克风输入语音，录音的时间长度约为 7 秒，按下任何键则停止录音，并把所录到的声音由语音卡播放出来，同时显示语音波形及录音总长度，当按下“ESC”键，则结束程序执行。

7. 声控链接库应用请参考说明文件 SP_LIB.DOC。
8. 语音编辑/识别整合控制程序请执行 MP.EXE。

附录 C　PC 语音卡使用

读者在使用PC语音卡做实验时，如果遇到一些问题时，可以参考本文说明。

C.1　PC I/O 口的使用

PC 语音卡上的I/O端口译码，可由卡上的DIP开关来调整(部份译码范围200H至270H)，以避免和其他您现有的适配卡相冲突，任何的设定，只能有一个位置于 ON，其余则必需置于 OFF 位置。一般 I/O 口使用如下：

表 C-1　PC I/O 口的使用

开关位置	译码基地址	适配卡使用
1	200H	游戏端口地址
2	210H	
3	220H	声霸卡选用地址
4	230H	
5	240H	声霸卡选用地址
6	250H	语音卡出厂时原设定地址
7	260H	
8	270H	

以上是硬件上的设置，至于软件上的设置修改如下：

C.1.1　语音分析及识别整合系统 MP.EXE 使用

MP　→I/O端口译码基地址=0X250
MP 5→I/O端口译码基地址=0X250
MP 6→I/O端口译码基地址=0X260
MP 7→I/O端口译码基地址=0X270

C.1.2　TURBO C 声控链接库 I/O 端口译码基底使用

函数：int SP_init_sp(int io_add)。
动作：对语音卡依指定的 I/O 地址做初始化的工作。
参数：io_add　语音卡的选定 I/O 地址。
SP_init_sp(0X250)→I/O 端口译码基地址=0X250
SP_init_sp(0X260)→I/O 端口译码基地址=0X260
SP_init_sp(0X270)→I/O 端口译码基地址=0X270

C.2　麦克风增益控制

适合语音卡上使用的麦克风最好为动圈式(非电容式)麦克风，即唱卡拉 OK 的那一型麦

克风，如果您使用的麦克风灵敏度不够，以致于所拾取到的语音波形太小，请微调可变电阻(VR1)，顺时针增益增加，语音卡出厂时已经调整至适当位置了，如非必要请勿任意调整。有时候，如果语音卡录不到语音的波形，有可能是麦克风输入接头松动，以致于接触不良，请自行检查。

C.3 喇叭音量控制

喇叭的使用请用阻抗 4Ω 的小型外接喇叭，可以得到不错的音效输出，如果要产生更大的声音输出，请连至功放输入，并适当地调整可变电阻(VR2)，顺时针音量增加。如果语音卡没有声音输出，有可能是喇叭输出接头松动，以致于接触不良，请自行检查。

C.4 I/O 扩充引脚图

语音卡使用 8255 控制芯片来做数字接口的工作，8255 本身有 3 个 I/O 口，PA、PB 及 PC。其中 PA 被设置为输入用做 A/D 语音输入采样，PB 被设置为输出提供给 D/A 做数字放音用，端口 C 则保留在特殊的硬件扩充用。

8255 端口 C 有 8 个 BIT 的 I/O，可以由 16PIN 脚座经由排线拉至 PC 外面来做实验。16PIN 脚座脚位分配如下：

PC0	P1	P2	+5V
PC1	P3	P4	+5V
PC2	P5	P6	
PC3	P7	P8	
PC4	P9	P10	
PC5	P11	P12	
PC6	P13	P14	GND
PC7	P15	P16	GND

附录 D PC 语音卡的其他应用

PC语音卡由于软件及硬件公开，读者可以随心所欲的自行应用，而发挥它的功能。目前支持此片语音识别卡的相关开发资源如下：

- ✧ SPR.EXE ：语音录音程序。
- ✧ SPP.EXE ：语音放音程序。
- ✧ MP.EXE ：语音识别及编辑程序。
- ✧ 可做声控指令设计。
- ✧ 可做声控电话拨号。
- ✧ 可做声控门禁设计。

- ✧ 可做声控家电控制。
- ✧ 可做学习型红外线家电遥控。
- ✧ TURBO C V2.0 语音卡声控链接库 (DOS 版)及语音辨识范例程序。

读者意见征询表

尊敬的读者，请您抽空填写此表，并邮寄或传真至我编辑部。在此，我们先向您表示诚挚的谢意，并将给积极参与的读者赠送精美小礼品。

1. 姓名 _____ 2. 性别 _____ 3. 年龄 _____ 4. 电话 _____
5. 单位 _____ 6. 职务/职称 _____
7. 通信地址 _____ _____邮编 _____E-mail_____
8. 您的文化程度: □中专、以下 □大专 □本科 □研究生以上
9. 您所属专业: □电子工程 □通信 □机械 □自动化 □计算机
10. 您所在行业: □硬件开发 □通信 □制造业 □其他
11. 您的工作性质: □设计开发 □教师 □生产 □其他
12. 您感兴趣的领域: □仪器仪表 □家电 □通信 □工业控制
13. 您经常使用哪个公司的工控、电子和通讯等硬件产品

_____

14. 您经常使用哪个公司的工控、电子和计算机辅助设计等软件产品

_____

15. 您目前或今后需要哪方面书籍

_____

16. 您对本书或本编辑部有何建议

_____

_____

请填好本表后寄给:

清华大学校内金地公司

《声控计算机制作与应用入门》编辑部收

邮编: 100084

联系电话:（010）62781736　　传真:（010）62788903

如需本书可与本编辑部联系邮购（汇款请按以上地址填写，另加邮费 10%）