WAV

百科名片

WAV为微软公司(Microsoft)开发的一种声音文件格式，它符合RIFF(Resource Interchange File Format)文件规范,用于保存Windows平台的音频信息资源，被Windows平台及其应用程序所广泛支持，该格式也支持MSADPCM，CCITT A LAW等多种压缩运算法，支持多种音频数字，取样频率和声道，标准格式化的WAV文件和CD格式一样，也是44.1K的取样频率，16位量化数字，因此在声音文件质量和CD相差无几！ WAV打开工具是WINDOWS的媒体播放器。

目录

简介

通常使用三个参数来表示声音，量化位数，取样频率和声道数。声道有单声道和立体声之分，取样频率一般有11025Hz(11kHz) ，22050Hz(22kHz)和44100Hz(44kHz) 三种，不过尽管音质出色，但在压缩后的文件体积过大！相对其他音频格式而言是一个缺点，其文件大小的计算方式为:
WAV格式文件所占容量(KB) = (取样频率 X 量化位数 X 声道) X 时间 / 8 (字节 = 8bit) 每一分钟WAV格式的音频文件的大小为10MB，其大小不随音量大小及清晰度的变化而变化。

目前支持WAV设计的手机主要为智能手机，如索尼爱立信P910和诺基亚N90以及采用微软OS的多普达等手机，而其它一些非智能手机的产品，如果宣传支持WAV格式则多半属于只是支持单声道的。

剖析

WAVE是录音时用的标准的WINDOWS文件格式，文件的扩展名为“WAV”，数据本身的格式为PCM或压缩型。

WAV文件格式是一种由微软和IBM联合开发的用于音频数字存储的标准，它采用RIFF文件格式结构，非常接近

于AIFF和IFF格式。符合 RIFF(Resource Interchange File Format)规范。所有的WAV都有一个文件头，这个文件头音频流的编码参数。

表1 WAV文件的文件头

偏移地址	字节数	类型	内容
00H~03H	4	字符	资源交换文件标志（RIFF）
04H~07H	4	长整数	从下个地址开始到文件尾的总字节数
08H~0BH	4	字符	WAV文件标志（WAVE）
0CH~0FH	4	字符	波形格式标志（FMT）
10H~13H	4	整数	过滤字节（一般为00000010H）
14H~15H	2	整数	格式种类（值为1，表示数据PCMμ律编码的数据）
16H~17H	2	整数	通道数，单声道为1，双声音为2
18H~1BH	4	长整数	采样频率
1CH~1FH	4	长整数	波形数据传输速率（每秒平均字节数）
20H~21H	2	整数	数据的调整数（按字节计算）
22H~23H	2	整数	样本数据位数

表2 WAV声音文件的数据块

偏移地址	字节数	类型	内容
24H~27H	4	字符	数据标志符（data）
28H~2BH	4	长整型	采样数据总数
2CH...	...		采样数据

WAV文件作为最经典的Windows多媒体音频格式，应用非常广泛，它使用三个参数来表示声音：采样位数、采样频率和声道数。

声道有单声道和立体声之分，采样频率一般有11025Hz（11kHz）、22050Hz（22kHz）和44100Hz（44kHz）三种。WAV文件所占容量=（采样频率×采样位数×声道）×时间/8（1字节=8bit）。

WAV对音频流的编码没有硬性规定，除了PCM之外，还有几乎所有支持ACM规范的编码都可以为WAV的音频流进行编码。多媒体应用中使用了多种数据，包括位图、音频数据、视频数据以及外围设备控制信息等。RIFF为存储这些类型的数据提供了一种方法，RIFF文件所包含的数据类型由该文件的扩展名来标识，能以RIFF文件存储的数据包括：

音频视频交错格式数据(.AVI) 、波形格式数据(.WAV) 、位图格式数据(.RDI) 、MIDI格式数据(.RMI) 、调色板格式(.PAL) 、多媒体电影(.RMN) 、动画光标(.ANI) 、其它RIFF文件(.BND) 。

WAVE文件可以存储大量格式的数据，通常采用的音频编码方式是脉冲编码调制(PCM)。由于WAV格式源自Windows/Intel环境，因而采用Little-Endian字节顺序进行存储。

WAVE文件作为多媒体中使用的声波文件格式之一，它是以RIFF格式为标准的。RIFF是英文Resource Interchange File Format的缩写，每个WAVE文件的头四个字节便是“RIFF”。WAVE文件由文件头和数据体两大

部分组成。其中文件头又分为RIFF/WAV文件标识段和声音数据格式说明段两部分。WAVE文件各部分内容及格式见附表。

常见的声音文件主要有两种，分别对应于单声道（11.025KHz采样率、8Bit的采样值）和双声道（44.1KHz采样率、16Bit的采样值）。采样率是指：声音信号在“模→数”转换过程中单位时间内采样的次数。采样值是指每一次采样周期内声音模拟信号的积分值。

对于单声道声音文件，采样数据为八位的短整数（short int 00H-FFH）；而对于双声道立体声声音文件，每次采样数据为一个16位的整数（int），高八位和低八位分别代表左右两个声道。

WAVE文件数据块包含以脉冲编码调制（PCM）格式表示的样本。WAVE文件是由样本组织而成的。在单声道WAVE文件中，声道0代表左声道，声道1代表右声道。在多声道WAVE文件中，样本是交替出现的。

WAVE文件的每个样本值包含在一个整数i中，i的长度为容纳指定样本长度所需的最小字节数。首先存储低有效字节，表示样本幅度的位放在i的高有效位上，剩下的位置为0，这样8位和16位的PCM波形样本的数据格式。WAVE文件作为多媒体中使用的声波文件格式之一，它是以RIFF格式为标准的。

RIFF是英文Resource Interchange File Format的缩写，每个WAVE文件的头四个

字节便是“RIFF”。

WAVE文件是由若干个Chunk组成的。按照在文件中的出现位置包括：RIFF WAVE

Chunk, Format Chunk, Fact Chunk(可选), Data Chunk。具体见下图：

| RIFF WAVE Chunk |

| ID = 'RIFF' |

| RiffType = 'WAVE' |

| Format Chunk |

| ID = 'fmt ' |

| Fact Chunk(optional) |

| ID = 'fact' |

| Data Chunk |

| ID = 'data' |

图1 Wav格式包含Chunk示例

RIFF WAVE Chunk

==========

| |所占字节数| 具体内容 |

==========

| ID | 4 Bytes | 'RIFF' |

| Size | 4 Bytes | |

| Type | 4 Bytes | 'WAVE' |

图2 RIFF WAVE Chunk

以'RIFF'作为标示，然后紧跟着为size字段，该size是整个wav文件大小减去ID

和Size所占用的字节数，即FileLen - 8 = Size。然后是Type字段，为'WAVE'，表示是wav文件。

结构定义如下：

struct RIFF_HEADER

{

char szRiffID[4]; // 'R','I','F','F'

DWORD dwRiffSize;

char szRiffFormat[4]; // 'W','A','V','E'

};

Format Chunk

==========

| | 字节数 | 具体内容 |

==========

| ID | 4 Bytes | 'fmt ' |

| Size | 4 Bytes | 数值为16或18，18则最后又附加信息 |

| FormatTag | 2 Bytes | 编码方式，一般为0x0001 | |

| Channels | 2 Bytes | 声道数目，1--单声道；2--双声道 | |

| SamplesPerSec | 4 Bytes | 采样频率 | |

| AvgBytesPerSec| 4 Bytes | 每秒所需字节数 | |===> WAVE_FORMAT

| BlockAlign | 2 Bytes | 数据块对齐单位(每个采样需要的字节数) | |

| BitsPerSample | 2 Bytes | 每个采样需要的bit数 | |

| | 2 Bytes | 附加信息（可选，通过Size来判断有无） | |

图3 Format Chunk

以'fmt '作为标示。一般情况下Size为16，此时最后附加信息没有；如果为18则最后多了2个字节的附加信息。主要由一些软件制成的wav格式中含有该2个字节的

附加信息。

结构定义如下：

```
struct WAVE_FORMAT
{
WORD wFormatTag;
WORD wChannels;
DWORD dwSamplesPerSec;
DWORD dwAvgBytesPerSec;
WORD wBlockAlign;
WORD wBitsPerSample;
};
struct FMT_BLOCK
{
char szFmtID[4]; // 'f','m','t',' '
DWORD dwFmtSize;
WAVE_FORMAT wavFormat;
};
```

Fact Chunk

	所占字节数	具体内容
ID	4 Bytes	'fact'
Size	4 Bytes	数值为4
data	4 Bytes	

图4 Fact Chunk

Fact Chunk是可选字段，一般当wav文件由某些软件转化而成，则包含该Chunk。

结构定义如下：

struct FACT_BLOCK

{

char szFactID[4]; // 'f','a','c','t'

DWORD dwFactSize;

};

Data Chunk

==========

| |所占字节数| 具体内容 |

==========

| ID | 4 Bytes | 'data' |

| Size | 4 Bytes | |

| data | | |

图5 Data Chunk

Data Chunk是真正保存wav数据的地方，以'data'作为该Chunk的标示。然后是

数据的大小。紧接着就是wav数据。根据Format Chunk中的声道数以及采样bit数，

wav数据的bit位置可以分成以下几种形式：

对于8位单声道，每个样本数据由8位（bit）表示；

对于8位立体声，每个声道的数据由一个8位（bit）数据表示，且第一个8位（bit）

数据表示0声道（左）数据，紧随其后的8位（bit）数据表示1声道（右）数据；

对于16位单声道，每个样本数据由16位（bit）表示；其中低字节存放高位，高字节存放低位

对于16位立体声，每个声道的数据由一个16位（bit）数据表示，且第一个16位（bit）

数据表示0声道（左）数据，紧随其后的16位（bit）数据表示1声道（右）数据。

| 单声道 | 取样1 | 取样2 | 取样3 | 取样4 |

| |----------

| 8bit量化 | 声道0 | 声道0 | 声道0 | 声道0 |

| 双声道 | 取样1 | 取样2 |

| |----------

| 8bit量化 | 声道0(左) | 声道1(右) | 声道0(左) | 声道1(右) |

| | 取样1 | 取样2 |

| 单声道 |----------

| 16bit量化 | 声道0 | 声道0 | 声道0 | 声道0 |

| | (低位字节) | (高位字节) | (低位字节) | (高位字节) |

| | 取样1 |

| 双声道 |----------

| 16bit量化 | 声道0(左) | 声道0(左) | 声道1(右) | 声道1(右) |

| | (低位字节) | (高位字节) | (低位字节) | (高位字节) |

图6 wav数据bit位置安排方式

Data Chunk头结构定义如下:

```
struct DATA_BLOCK {
char szDataID[4]; // 'd','a','t','a'
DWORD dwDataSize;
};
```

特点

WAV音频格式的优点包括：简单的编/解码(几乎直接存储来自模/数转换器(ADC)的信号)、普遍的认同/支持以及无损耗存储。WAV格式的主要缺点是需要音频存储空间。对于小的存储限制或小带宽应用而言，这可能是一个重要的问题。WAV格式的另外一个潜在缺陷是在32位WAV文件中的2G限制，这种限制已在为SoundForge开发的W64格式中得到了改善。

常见的WAV文件使用PCM无压缩编码，这使WAV文件的质量极高，体积也出奇大，对于PCM WAV，恐怕也只有无损压缩的音频才能和其有相同的质量，平时我们见的什么mp3,wma(不含 wmalossless)和wav的质量都是差很远的！这点可以通过频谱看出，即使320kbps的mp3和wav一比，也要自卑了！

Wav(1张)

支持

Wav格式支持MSADPCM、CCITTALaw、CCITT μ Law和其它压缩算法，支持多种音频位数、采样频率和声道，但其缺点是文件体积较大（一分钟44kHZ、16bit Stereo的WAV文件约要占用10MB左右的硬盘空间），所以不适合长时间记录。

在Windows中，把声音文件存储到硬盘上的扩展名为WAV。WAV记录的是声音的本身，所以它占的硬盘空间大的很。例如：16位的44.1KHZ的立体声声音一分钟要占用大约10MB的容量，和MIDI相比就差的很远。

转换

AVI和WAV在文件结构上是非常相似的，不过AVI多了一个视频流而已。我们接触到的AVI有很多种，因此我们经常需要安装一些Decode才能观看一些AVI，我们接触到比较多的DivX就是一种视频编码，AVI可以采用DivX编码来压缩视频流，当然也可以使用其他的编码压缩。同样，WAV也可以使用多种音频编码来压缩其音频流，不过我们常见的都是音频流被PCM编码处理的WAV，但这不表示
WAV只能使用PCM编码，MP3编码同样也可以运用在WAV中，和AVI一样，只要安装好了相应的dDecode，就可以欣赏这些WAV了。

在Windows平台下，基于PCM编码的WAV是被支持得最好的音频格式，所有音频软件都能完美支持，由于本身可以达到较高的音质的要求，因此，WAV也是音乐编辑创作的首选格式，适合保存音乐素材。因此，基于PCM编码的WAV被作为了一种中介的格式，常常使用在其他编码的相互转换之中，例如MP3转换成WMA。

编解码器

WAV文件格式是一种由微软和IBM联合开发的用于音频数字存储的标准，它采用RIFF文件格式结构，非常接近于AIFF和IFF格式。多媒体应用中使用了多种数据，包括位图、音频数据、视频数据以及外围设备控制信息

等。RIFF为存储这些类型的数据提供了一种方法，RIFF文件所包含的数据类型由该文件的扩展名来标识，能以RIFF文件存储的数据包括：

音频视频交错格式数据(.AVI)

波形格式数据(.WAV)

位图格式数据(.RDI)

MIDI格式数据(.RMI)

调色板格式(.PAL)

多媒体电影(.RMN)

动画光标(.ANI)

其它RIFF文件(.BND)

RIFF是一种含有嵌套数据结构的二进制文件格式，每个数据结构都称为因一个chunk(块)。Chunk在RIFF文件中没有固定的位置，因而偏移量不能用于定位域值。一个块中的数据包括数据结构、数据流或其它组块(称为子块)等，每个RIFF块都具有如下结构：

```
typedef struct _Chunk
                    {
                    DWORD ChunkId; /*块标志*/
                    DWORD ChunkSize; /*块大小*/
                    BYTE ChunkData[ChunkSize]; /*块内容*/
                    } CHUNK;
```

ChunkId由4个ASCII字符组成，用以识别块中所包含的数据。字符RIFF用于标识RIFF数据块，间隔空格在右面是不超过4个字符的ID。由于这种文件结构最初是由Microsoft和IBM为PC机所定义，RIFF文件是按照little-endian字节顺序写入的，而采用big-endian字节顺序的文件则用‘RIFX’作为标志。

ChunkSize(块大小)是存储在ChunkData域中数据的长度，ChunkId与ChunkSize域的大小则不包括在该值内。

ChunkData(块内容)中所包含的数据是以字(WORD)为单位排列的，如果数据长度是奇数，则在最后添加一个空(NULL)字节。

子块(Subchunk)与块具有相同的结构。一个子块就是包含在其它块内部的一个块，只有RIFF文件块‘RIFF’和列表块‘List’才能含有子块，所有其它块仅能含有数据。一个RIFF文件就是一个RIFF块，文件中所有其它块和子块均包含在这个块中。

WAV文件可以存储大量格式的数据，通常采用的音频编码方式是脉冲编码调制(PCM)。由于WAV格式源

自Windows/Intel环境，因而采用Little-Endian字节顺序进行存储。

脉冲编码调制

Claude E. Shannon于1948年发表的“通信的数学理论”奠定了现代通信的基础。同年贝尔实验室的工程人员开发了PCM技术，虽然在当时是革命性的，但今天脉冲编码调制被视为是一种非常单纯的无损耗编码格式，音频在固定间隔内进行采集并量化为频带值，其它采用这种编码方法的应用包括电话和CD。PCM主要有三种方式：标准PCM、差分脉冲编码调制(DPCM)和自适应DPCM。在标准PCM中，频带被量化为线性步长的频带，用于存储绝对量值。在DPCM中存储的是前后电流值之差，因而存储量减少了约25%。自适应DPCM改变了DPCM的量化步长，在给定的信造比(SNR)下可压缩更多的信息。

共同的执行过程

在对WAV音频文件进行编解码过程中，最一致的地方包括采样点和采样帧的处理和转换。一个采样点的值代表了给定时间内的音频信号，一个采样帧由适当数量的采样点组成并能构成音频信号的多个通道。对于立体声信号一个采样帧有两个采样点，一个采样点对应一个声道。一个采样帧作为单一的单元传送到数/模转换器(DAC)，以确保正确的信号能同时发送到各自的通道中。

VB中WAV

综述

在多媒体软件的开发设计中，声音是一个相当重要的多媒体元素，优秀的声音设计会为多媒体软件增色不少。而WAV格式的声音文件是一种最常用的声音文件格式，也最容易得到，比如通过Win 95中的“录音机”程序，利用麦克风就可以非常简单地录制WAV文件。VB是一个相当经典的多媒体开发的工具，在VB中播放WAV文件的方法主要有这样几种。

利用OLE控件

建一窗体，用鼠标选择OLE控件，在窗体上拖出OLE区域，在图一的窗口中选择新建和声音然后按确定键就完成了在窗口中添加OLE控件。

这样就可以在OLE控件的ResourceDoc属性中选择所要播放的文件，程序运行时双击OLE控件即可。

在实际的程序设计当中，往往需要单击某个图标或按钮来控制声音的播放，其实现方法是这样的：首先将OLE控件的VISIBLE属性设置为FALSE,然后在图标或按钮的单击事件中编写如下的程序：

```
Private Sub Contol_Click()

OLE1.Action = 7

End Sub
```

利用MMControl控件

VB5.0提供了许多设计多媒体的控件，在PROJECT/COMPENENTS/CONTROLS中选择MMControls 控件，窗

体上就出现了多媒体控件对象，在这个对象上有不同的图形功能标识，其名称从左到右分别是Pre、Next、Play、Pause、Back、Step、Stop、Record 、Eject。

这个多媒体控件可以播放多种格式的声音，播放WAV格式声音文件的程序代码

```
Private Sub form_load()

MMControl1.DeviceType = "waveaudio"

MMControl1.filename = "c:\win95\media\ding.wav"

MMControl1.Command = "open"

End Sub '以下是为图形标识Play事件编写的代码

Private Sub MMControl_playclick(cancel As Integer)

MMControl1.Command = "play"

End Sub
```

当运行这个程序时，MMControl控件中的Play键被激活，点取此按钮即可播放ding.wav文件。

在实际的软件设计当中，更多的情况是鼠标点击按钮或图标来控制声音的播放, 其实现方法是这样的：首先将MMControl控件的VISIBLE属性设置为FALSE,然后在图标或按钮的单击事件中编写如下的程序：

```
Private Sub Control1_Click()

MMControl1.Command = "play"

End Sub
```

运行程序时单击相用的按钮或图标，WAV文件照样可以播放。用自行设计的按钮或图标取代多媒体控件中的固定按钮，可设计出更加灵活方便的用户界面。

利用VB的API函数

在窗体的DECLARATIONS(声明)中输入如下代码：

```
P rivate Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal_lpszSoundName As String, ByVal uFlags As Long) As Long

'lpszSoundName是一个字符串变量，表示一个WAV格式的文件名。

'uFlags 用于设定播放状态的各种选项。参数值为0X00时，实现同步播放，参数值为0X01时实现非同步播放。
```

在命令按钮的单击事件中输入如下代码：

```
Private Sub Command1_Click()
```

```
Dim plays As Long

plays = sndPlaySound(“E:\WINDOWS\MEDIA\DING.WAV", &H0)

End Sub
```

运行时单击命令按钮即可播放WAV文件。