AVI是音频视频交错(Audio Video Interleaved)的英文缩写，它是Microsoft公司开发的一种符合RIFF文件规范的数字音频与视频文件格式。AVI格式允许视频和音频交错在一起同步播放，支持256色和RLE压缩，但AVI文件并未限定压缩标准，因此，AVI文件格式只是作为控制界面上的标准，不具有兼容性，用不同压缩算法生成的AVI文件，必须使用相应的解压缩算法才能播放出来。

AVI的主要参数

视频参数：

1. 视窗尺寸（Video size）：AVI的视窗大小可按4:3的比例或随意调整，视窗越大，数据量越大。
2. 帧率（Frames per second）：帧率也可以调整，而且与数据量成正比。不同的帧率会产生不同的效果。

|  |  |  |
| --- | --- | --- |
| 帧率（fps） | 文件大小（KB） | 效果 |
| 6 | 217 | 画面出现跳动的不连续感 |
| 15 | 637 | 画面基本连续，是实际应用中较常使用的参数 |
| 25 | 1134 | 理想的帧率，但数据量太大 |

音频参数：

在AVI文件中，视频和音频是分别存储的，因此可以把一段视频中的图像与另一段视频中的声音组合在一起。AVI 文件与WAV文件密切相关，因為WAV文件是AVI文件中音频信号的来源，音频的基本参数也即WAV格式的参数。除此以外，AVI文件还包括与音频有关的其他参数：

1. 视频与音频的交织参数（Interlace Audio Every X Frames）：AVI格式中每X帧交织存储的音频信号，也即音频和图象交替的频率。X是可调参数，X的最小值是一帧，即每个视频帧与音频数据交织组织，这是CD-ROM上使用的默认值。交织参数越小，回放AVI文件时读到内存中的数据流越少，回放越连续。因此，如果AVI文件的存储平台的数据传输率较大，则交错参数可设置得高一些，如几帧，甚至1秒。
2. 同步控制（Synchronization）：在AVI文件中，图像和音频是同步得很好的。但实际上由于CPU处理能力的不够，回放AVI时有可能出现图像和音频不同步的现象。当AVI文件的数据率较高，而MPC的处理速度不够时，容易出现图像和音频不同步的现象。如视频中人张嘴说话，但声音并没有发出来。设置同步控制可保证在不同的MPC环境下播放该AVI文件时都能同步。此时播放程序自动地丢掉一些中间帧以保证视频和音频的同步。

压缩参数：

在采集原始模拟视频时可以用不压缩的方式，这样可以获得最优秀的图像质量，但是代价就是文件极大。最原始的AVI每秒可达150M。生成AVI文件时需要根据应用环境的不同选择合适的压缩参数。

压缩算法：

压缩算法是首先要确定的一个参数。与MPEG标准不同的是，AVI采用的压缩算法并无统一的标准。也就是说，同样是以AVI为后缀的视频文件，其采用的压缩算法可能不同，需要相应的解压缩软件才能识别和回放该AVI文件。Microsoft公司推出AVI文件格式和VFW软件时，同时也推出了一种压缩算法，由于AVI和VFW的开放性，其它的公司也相应推出了其它压缩算法，只要把该算法的驱动加到Windows系统中，就可以在Windows系统中播放用该算法压缩的AVI文件。

图像深度：

与静态图像一样，视频的图像深度决定其可以显示的颜色数。某些编码（压缩算法）使用固定的图像深度，在这种情况下该参数不可调整。较小的图像深度可以减小文件的容量，但同时也降低了图像的质量。

压缩质量：

选择了一种压缩算法后还可以调整压缩质量，这个参数常用百分比来表示，100％表示最佳效果压缩。同一种压缩算法下，压缩质量越低，文件容量越小，丢失信息越多。

关键帧：

关键帧（Key Frame）是其他帧压缩时与之比较并产生差值的基准。关键帧可以不压缩，而中间帧（也称作差值帧）是根据其与关键帧的差异来压缩的。采用关键帧压缩可以使压缩比更小而回放速率更快，但在一段视频文件中访问某一帧的时间将延长。该参数只有在使用帧间压缩编码如帧间差值编码时才起作用。如果不设置关键帧，则编码器默认每一帧都是关键帧。

数据率：

根据其他参数，可以计算出AVI文件的数据率，一般以每秒兆比特计（MB/s）。数据率是AVI文件的一个重要参数。实际播放AVI文件时，从某种意义上说文件的数据率只能起到为播放平台设置初始的数据传输率的作用。如果AVI文件的数据率过高，而播放该AVI文件MPC达不到要求，则播放时可能出现不同步或者丢帧现象。因此，要根据播放环境的要求确定AVI的数据率，然后根据数据率的要求再确定其它参数。

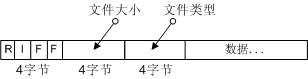
AVI格式

AVI文件采用的是RIFF文件结构方式，RIFF（Resource Interchange File Format，资源互换文件格式）是微软公司定义的一种用于管理windows环境中多媒体数据的文件格式，RIFF文件使用四字符码FOURCC，来表征数据类型，比如‘RIFF’、‘AVI ’、‘LIST’等。注意，Windows操作系统使用的字节顺序是little-endian，因此一个四字符码‘abcd’实际的DWORD值应为0x64636261。另外，四字符码中像‘AVI ’一样含有空格也是合法的。波形音频wave，MIDI和数字视频AVI都采用这种格式存储。

RIFF結構

RIFF文件的实际数据中，通常还使用了列表（List）和块（Chunk）的形式来组织。列表可以嵌套子列表和块。

1、”RIFF”表示字串  
2、RIFF文件大小  
3、形式类型或者列表类型 ”AVI”、”WAVE”…  
4、RIFF数据

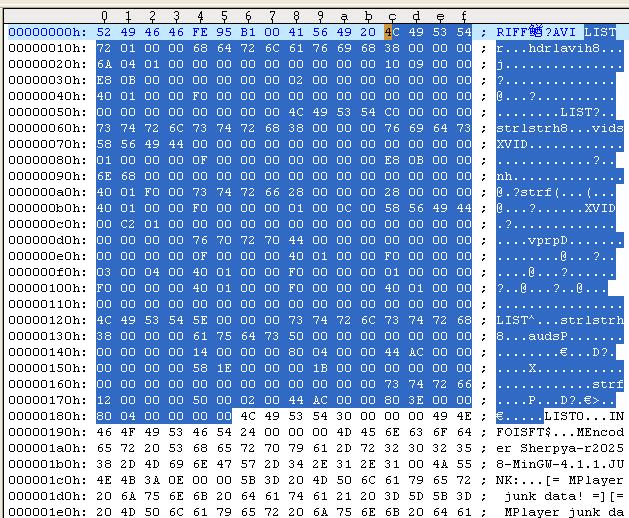


RIFF文件大小 = 實際資料長度 + 4（形式类型或者列表类型的大小）；也就是說，檔大小的值不包括‘RIFF’(‘LIST’)域和“檔大小”域本身的大小。（如该位置数据为28 69 6B 00，则实际为0x006B6928，转换成十进制为7039272，实际在windows下看到的文件大小为7039280 bytes，即换算成实际大小时要加上8）.

LIST結構

RIFF块中包含一系列的子块，其中有一种字块的ID为"LIST"，称为LIST，LIST块中可以再包含一系列的子块，但除了LIST块外的其他所有的子块都不能再包含子块。  
‘1、”LIST”表示字串  
2、LIST文件大小  
3、形式类型或者列表类型  
4、LIST数据

LIST文件大小 = 实际的列表数据长度 + 4（形式类型或者列表类型的大小）；也就是说listSize值不包括‘LIST’域和listSize域本身的大小。(如该位置数据为5E 00 00 00，即0x0000005E，转换成十进制为94，实际总长为102 bytes，即换算成列表总长是要加上8。但是有一点要特别注意的是，当前List具体包括到哪里，可能会有List嵌套。如截图中，选中的部分为一个hdrl的List；其中包含两个strl的List，一个音频，一个视频；每个strl List又包含一个strh和一个strf)



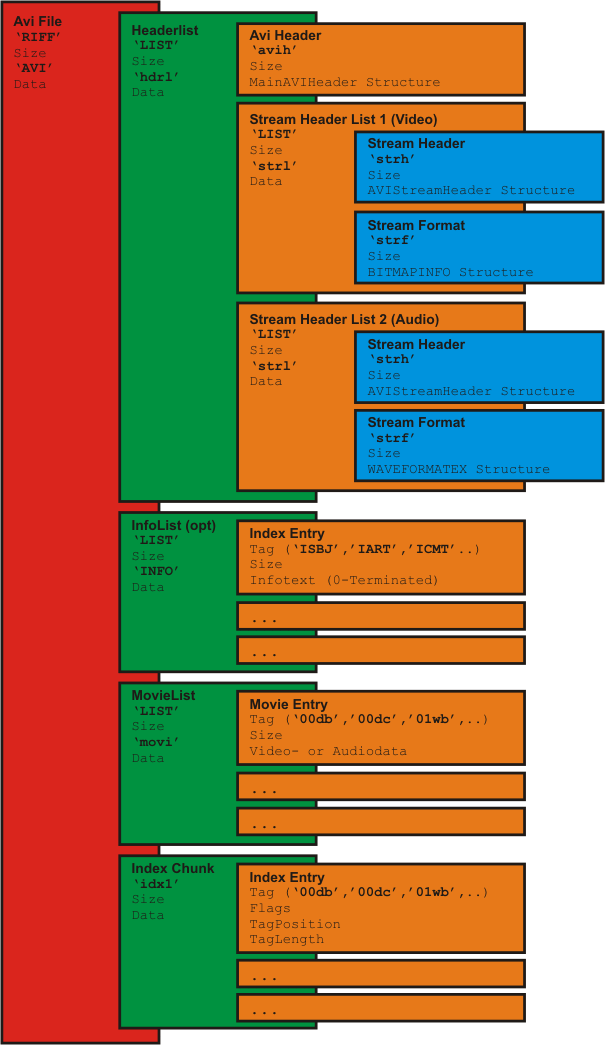
CHUNK結構

‘1、Chunk ID - 表示块类型的四字符码  
2、Chunk文件大小 -记录了整个块的大小  
3、Chunk数据

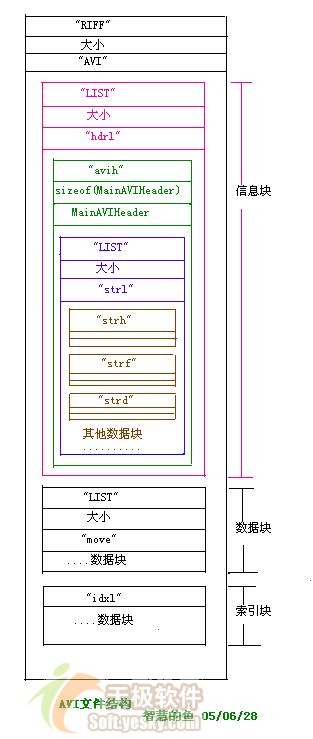
Chunk文件大小 = 实际的块数据长度，而不包括ckID域和ckSize域本身的大小。

从（00000000－000007F3）为一个WindowsAVI文件的信息区部分。   
它是文件的第一个LIST块。在它的内部记录着整个文件的系统构成，如告诉播放软件“我是一个AVI文件”；“在我的体内有几个数据流”；“每个数据流包含着什么数据类型——图像、声音或其他”；“如果是图像数据流，那么它的大小、颜色、压缩方式、播放速度等，等是怎样规定的”；“如果是声音数据流，那么它的压缩方式、播放效果等等又将有何规定”......在信息区中还有多个附属的LIST块，也就是前面提到的“子块”，它们用来记录每个数据流的全部信息。而这些附间LIST块与数据流之间保持着—一对应的关系，即 第一个附属LIST块对应于00号数据流;第二个附属LIST块对应于01号数据流......要想解释数据流，我们必须先了解AVI文件中数据块是什么。在AVI文件中，数据块是被放置在数据区中的一个有起始标志（由“数据流识别码”和“数据块存储方式识别码"组成,请参见对数据区部分的说明),并指明大小和数据内容的数据段.那么,数据流就是那些相互之间具有联系的同种数据类型的数据块集合.

00000000-00000003 多媒体文件识别码:RIFF   
00000004-00000007 文件大小（10EDICh字节）－8字节   
00000008-0000000B AVI文件识别码   
0000000C-0000000F 第一个LIST块识别码   
00000010-00000013 第一个LIST块的大小（168h字节）   
00000014-00000017 hdrl部分识别码，以下数据记录着此文件的格式   
00000018-0000001B hdrl部分所包含的avih块识别码，此模块记录着本文件的初始化信息   
0000001C-0000001F avih块大小（38h字节）   
00000020-00000023 每帧画面显示所维持多少个百万分之一秒，本例为1046Bh，即66667百万分之一秒，约0.07秒。所以在播放此文件时，你看到的画面约每秒15帧



**AVI Layout：**



**信息塊 -** 信息塊包括文件的通用信息，定义数据格式，所用的压缩算法等参数。

**数据块 -** 数据块包含实际数据流，即图像和声音序列数据。这是文件的主体，也是决定文件容量的主要部分。视频文件的大小等于该文件的数据率乘以该视频播放的时间长度

**索引块 -** 索引块包括数据块列表和它们在文件中的位置，以提供文件内数据随机存取能力。

AVI文件是目前使用的最复杂的RIFF文件，它能同时存储同步表现的音频视频数据。AVI的RIFF块的形式类型是AVI，它包含3个子块，如下所述：  
1、信息块，ID为 "hdrl" 的LIST块，定义AVI文件的数据格式。

‘hdrl’列表嵌套了一系列块和子列表 — 首先是一个‘avih’块，用于记录AVI文件的全局信息，比如流的数量、视频图像的宽和高等：

|  |
| --- |
| typedef struct {  DWORD ChunkID; // 必需為'avih'  DWORD ChunkSize; // 本資料結構的大小，不包括最初的8個位元組（ID和Size兩個域）      DWORD   MicroSecPerFrame;   // 視頻幀間隔時間（以毫秒為單位）     DWORD   MaxBytesPerSec;     // 這個AVI檔的最大資料率     DWORD   PaddingGranularity; // 數據填充的粒度     DWORD   Flags;         // AVI檔的全局標記，比如是否含有索引塊等     DWORD   TotalFrames;   // 總幀數     DWORD   InitialFrames; // 為交互格式指定初始幀數（非交互格式應該指定為0）     DWORD   Streams;       // 本檔包含的流的個數     DWORD SuggestedBufferSize; // 建議讀取本檔的緩存大小（應能容納最大的塊）     DWORD Width;         // 視頻圖像的寬（以圖元為單位）     DWORD   Height;        // 視頻圖像的高（以圖元為單位）     DWORD   Reserved[4];   // 保留 }MainAVIHeader; |

接著，就是一个或多个‘strl’子列表。（文件中有多少个流，这里就对应有多少个‘strl’子列表。）每个‘strl’子列表至少包含一个‘strh’块和一个‘strf’块，而‘strd’块（保存编解码器需要的一些配置信息）和‘strn’块（保存流的名字）是可选的。（注意：‘strl’子列表出现的顺序与媒体流的编号是对应的，比如第一个‘strl’子列表说明的是第一个流（Stream 0），第二个‘strl’子列表说明的是第二个流（Stream 1），以此类推）

首先是‘strh’块，用于说明这个流的头信息：

|  |
| --- |
| typedef struct { FOURCC fcc;  // 必須為‘strh’ DWORD   cb;   // 本資料結構的大小，不包括最初的8個位元組（fcc和cb兩個域） FOURCC fccType; //流的种類，如：vids(视频流), auds(音频流), mids(MIDI流), txts(文字流） FOURCC fccHandler;//表示数据流解压缩的驱动程序代号 DWORD dwFlags; //数据流属性,標記：是否允許這個流輸出？調色板是否變化？ WORD wPriority; //此数据流的播放优先级 WORD wLanguage; //音频的语言代号 DWORD dwInitalFrames;//说明在开始播放前需要多少桢 DWORD dwScale; //数据量，视频每桢的大小或者音频的采样大小 DWORD dwRate; //dwScale /dwRate = 每秒的采样数 DWORD dwStart; //数据流开始播放的位置，以dwScale为单位 DWORD dwLength; //数据流的数据量，（单位与dwScale和dwRate的定义有关） DWORD dwSuggestedBufferSize; //建议缓冲区的大小 DWORD dwQuality; //解压缩质量参数，值越大，质量越好（0 ~ 10,000） DWORD dwSampleSize; //音频的采样大小  struct {  short int left;  short int top;  short int right;  short int bottom;  } rcFrame; // 指定这个流（视频流或文字流）在视频主窗口中的显示位置  //视频主窗口由AVIMAINHEADER结构中的dwWidth和dwHeight决定 }AVIStreamHeader; |

"strf"子块紧跟在"strh"子块之后，其结构视"strh"子块的类型而定，如下所述；

1、如果 strh子块是视频数据流，则 strf子块的内容是一个BIMAPINFO结构，如下：

|  |
| --- |
| typedef struct tagBITMAPINFO { 　BITMAPINFOHEADER bmiHeader; 　RGBQUAD bmiColors[1]; //颜色表 }BITMAPINFO;  typedef struct tagBITMAPINFOHEADER { 　DWORD biSize; 　LONG biWidth; // 说明图象的宽度，以象素为单位 　LONG biHeight;// 说明图象的高度，以象素为单位, 如果该值是一个正数，说明图像是倒向的，如果该值是一个负数，则说明图像是正向的。 　WORD biPlanes;// 为目标设备说明位面数，其值将总是被设为1 　WORD biBitCount;// 说明比特数/象素，其值为1、4、8、16、24、或32 　DWORD biCompression;// 说明图象数据压缩的类型。其值可以是下述值之一： 　DWORD biSizeImage;// 说明图象的大小，以字节为单位。当用BI\_RGB格式时，可设置为0 　LONG biXPelsPerMeter;// 说明水平分辨率，用象素/米表示 　LONG biYPelsPerMeter;// 说明垂直分辨率，用象素/米表示 　DWORD biClrUsed;// 说明位图实际使用的彩色表中的颜色索引数（设为0的话，则说明使用所有调色板项） 　DWORD biClrImportant;// 说明对图象显示有重要影响的颜色索引的数目，如果是0，表示都重要 }BITMAPINFOHEADER; |

2、如果 strh子块是音频数据流，则strf子块的内容是一个WAVEFORMAT结构，如下：

|  |
| --- |
| typedef struct { 　WORD wFormatTag;  　WORD nChannels; //声道数 　DWORD nSamplesPerSec; //采样率 　DWORD nAvgBytesPerSec; //WAVE声音中每秒的数据量 　WORD nBlockAlign; //数据块的对齐标志 　WORD biSize; //此结构的大小 }WAVEFORMAT |

2、数据块，ID为 "movi" 的LIST块，包含AVI的音视频序列数据。

用于保存真正的媒体流数据（视频图像帧数据或音频采样数据等）。保存的方式為：

1. 将数据块直接嵌在‘movi’列表里面
2. 将几个数据块分组成一个‘rec ’列表后再编排进‘movi’列表。

（注意：在读取AVI文件内容时，建议将一个‘rec ’列表中的所有数据块一次性读出。）但是，当AVI文件中包含有多个流的时候，数据块与数据块之间如何来区别呢？于是数据块使用了一个四字符码来表征它的类型，这个四字符码由2个字节的类型码和2个字节的流编号组成。

标准的类型码定义如下：

1. ‘db’（非压缩视频帧）、
2. ‘dc’（压缩视频帧）、
3. ‘pc’（改用新的调色板）、
4. ‘wb’（音缩视频）。

比如第一个流（Stream 0）是音频，则表征音频数据块的四字符码为‘00wb’；第二个流（Stream 1）是视频，则表征视频数据块的四字符码为‘00db’或‘00dc’。对于视频数据来说，在AVI数据序列中间还可以定义一个新的调色板，每个改变的调色板数据块用‘xxpc’来表征，新的调色板使用一个数据结构AVIPALCHANGE来定义。（注意：如果一个流的调色办中途可能改变，则应在这个流格式的描述中，也就是AVISTREAMHEADER结构的dwFlags中包含一个AVISF\_VIDEO\_PALCHANGES标记。）另外，文字流数据块可以使用随意的类型码表征。  
  
3、索引块，ID为 "idxl" 的子块，定义 "movi" LIST块的索引数据，是可选块。  
最后，紧跟在‘hdrl’列表和‘movi’列表之后的，就是AVI文件可选的索引块。这个索引块为AVI文件中每一个媒体数据块进行索引，并且记录它们在文件中的偏移（可能相对于‘movi’列表，也可能相对于AVI文件开头）。索引块使用一个四字符码‘idx1’来表征，索引信息使用一个数据结构来AVIOLDINDEX定义。

|  |
| --- |
| typedef struct \_avioldindex { FOURCC fcc; // 必须为‘idx1’ DWORD cb; // 本数据结构的大小，不包括最初的8个字节（fcc和cb两个域）  struct \_avioldindex\_entry {  DWORD dwChunkId; // 表征本数据块的四字符码  DWORD dwFlags; // 说明本数据块是不是关键帧、是不是‘rec ’列表等信息  DWORD dwOffset; // 本数据块在文件中的偏移量  DWORD dwSize; // 本数据块的大小  } aIndex[]; // 这是一个数组！为每个媒体数据块都定义一个索引信息 } AVIOLDINDEX; |

注意：如果一个AVI文件包含有索引块，则应在主AVI信息头的描述中，也就是AVIMAINHEADER结构的dwFlags中包含一个AVIF\_HASINDEX标记。  
还有一种特殊的数据块，用一个四字符码‘JUNK’来表征，它用于内部数据的队齐（填充），应用程序应该忽略这些数据块的实际意义。