# 音视频基础

## 基础概念介绍

### 1.1 GOP（Group of picture）

两个IDR帧之间的距离，一般而言，每一秒视频至少需要使用 1 个关键帧。增加关键帧个数可改善质量，但是同时增加带宽和网络负载。GOP=30，30FPS，即每个I帧之间相差1s。需要说明的是，通过提高GOP值来提高图像质量是有限度的，在遇到场景切换的情况时，H.264编码器会自动强制插入一个I帧，此时实际的GOP值被缩短了。另一方面，在一个GOP中，P、B帧是由I帧预测得到的，当I帧的图像质量比较差时，直到下一个GOP开始才有可能得以恢复，所以GOP值也不宜设置过大。过长的GOP还会影响Seek操作的响应速度，Seek操作需要先解码得到本GOP内的I帧及之前的N个预测帧才可以，GOP值越长，需要解码的预测帧就越多，seek响应的时间也越长。

### 1.2 CABAC / CAVLC

H.264/AVC标准中两种熵编码方法：CABAC叫自适应二进制算数编码，CAVLC叫前后自适应可变长度编码。其实就是帧编码和场编码。

CABAC：是一种无损编码方式，画质好，X264就会舍弃一些较小的DCT系数，码率降低，可以将码率再降低10-15%（特别是在高码率情况下），会降低编码和解码的速速。

CAVLC：占用更少的CPU资源，但会影响压缩性能。

帧：当采样视频信号时，如果是通过逐行扫描，那么得到的信号就是一帧图像，通常帧频为25帧每秒（PAL制）、30帧每秒（NTSC制）；

场：当采样视频信号时，如果是通过隔行扫描（奇、偶数行），那么一帧图像就被分成了两场，通常场频为50Hz（PAL制）、60Hz（NTSC制）；

帧频、场频的由来：最早由于抗干扰和滤波技术的限制，电视图像的场频通常与电网频率（交流电）相一致，于是根据各地交流电频率不同就有了欧洲和中国等PAL制的50Hz和北美等NTSC制的60Hz，但是现在并没有这样的限制了，帧频可以和场频一样，或者场频可以更高。

帧编码、场编码方式：逐行视频帧内邻近行空间相关性较强，因此当活动量非常小或者静止的图像比较适宜采用帧编码方式；而场内相邻行之间的时间相关性较强，对运动量较大的运动图像则适宜采用场编码方式。

### 1.3 Deblocking / FORCE\_IDR

Deblocking：开启会减少块效应

FORCE\_IDR：是否让每个I帧变成IDR帧，如果是IDR帧，支持随机访问。

### 1.4 frame / tff / bff

frame 将两场合并作为一帧进行编码

tff Enable interlaced mode (开启隔行编码并设置上半场在前)

bff Enable interlaced mode。

PAFF 和MBAFF：当对隔行扫描图像进行编码时，每帧包括两个场，由于两个场之间存在较大的扫描间隔，这样，对运动图像来说，帧中相邻两行之间的空间相关性相对于逐行扫描时就会减小，因此这时对两个场分别进行编码会更节省码流。

对帧来说，存在三种可选的编码方式：将两场合并作为一帧进行编码(frame 方式)或将两场分别编码(field 方式)或将两场合并起来作为一帧，但不同的是将帧中垂直相邻的两个宏块合并为宏块对进行编码；前两种称为PAFF 编码，对运动区域进行编码时field 方式有效，对非运区域编码时，由于相邻两行有较大的相关性，因而frame 方式会更有效。当图像同时存在运动区域和非运动区域时，在MB 层次上，对运动区域采取field 方式，对非运动区域采取frame 方式会更加有效，这种方式就称为MBAFF，预测的单位是宏块对。

### 1.5 码流 / 码率

码流(Data Rate)是指视频文件在单位时间内使用的数据流量，也叫码率或码流率，通俗就是取样率，画面质量控制中最重要的部分，一般是kb/s或者Mb/s。一般来说同样分辨率下，视频文件的码流越大，压缩比就越小，画面质量就越高，说明单位时间内取样率越大，精度就越高，处理出来的文件就越接近原始文件，图像质量越好，要求播放设备的解码能力也越高。

　　当然，码流越大，文件体积也越大，其计算公式是文件体积=时间X码率/8。例如，网络上常见的一部90分钟1Mbps码流的720P RMVB文件，其体积就=5400秒×1Mb/8=675MB。

　　通常来说，一个视频文件包括了画面及声音，而我们所说的一个视频文件码流率大小，一般是指视频文件中音频及视频信息码流率的总和。

以以国内最流行，大家最熟悉的RMVB视频文件为例，RMVB中的VB，指的是VBR，中文含义是可变比特率，表示RMVB采用的是动态编码的方式，把较高的采样率用于复杂的动态画面(歌舞、飞车、战争、动作等)，而把较低的采样率用于静态画面，达到画质与体积可兼得的效果。

### 1.6 采样率

采样率（也称为采样速度或者采样频率）定义了每秒从连续信号中提取并组成离散信号的采样个数，它用赫兹（Hz）来表示。采样率是指将模拟信号转换成数字信号时的采样频率，也就是单位时间内采样多少点。一个采样点数据有多少个比特。比特率是指每秒传送的比特(bit)数。单位为 bps(Bit Per Second)，比特率越高，传送的数据越大，音质越好.比特率 =采样率 x 采用位数 x声道数.

采样率类似于动态影像的帧数，比如电影的采样率是24赫兹，PAL制式的采样率是25赫兹，NTSC制式的采样率是30赫兹。当我们把采样到的一个个静止画面再以采样率同样的速度回放时，看到的就是连续的画面。同样的道理，把以44.1kHZ采样率记录的CD以同样的速率播放时，就能听到连续的声音。显然，这个采样率越高，听到的声音和看到的图像就越连贯。当然，人的听觉和视觉器官能分辨的采样率是有限的，基本上高于44.1kHZ采样的声音，绝大部分人已经觉察不到其中的分别了。

而声音的位数就相当于画面的颜色数，表示每个取样的数据量，当然数据量越大，回放的声音越准确，不过受人的器官的机能限制，16位的声音和24位的画面基本已经是普通人类的极限了，更高位数就只能靠仪器才能分辨出来了。比如电话就是3kHZ取样的7位声音，而CD是44.1kHZ取样的16位声音，所以CD就比电话更清楚。

当你理解了以上这两个概念，比特率就很容易理解了。以电话为例，每秒3000次取样，每个取样是7比特，那么电话的比特率是21000。 而CD是每秒 44100次取样，两个声道，每个取样是13位PCM编码，所以CD的比特率是44100\*2\*13=1146600，也就是说CD每秒的数据量大约是 144KB，而一张CD的容量是74分等于4440秒，就是639360KB＝640MB。

码率和取样率最根本的差别就是码率是针对源文件来讲的。

### 1.7 比特率

比特率是指每秒传送的比特(bit)数。单位为bps(Bit Per Second)，比特率越高，传送的数据越大。在视频领域,比特率常翻译为码率 !!!

比特率表示经过编码（压缩）后的音、视频数据每秒钟需要用多少个比特来表示，而比特就是二进制里面最小的单位，要么是0，要么是1。比特率就是比特率越高，音、视频的质量就越好，但编码后的文件就越大；如果比特率越少则情况刚好相反。

比特率是指将数字声音、视频由模拟格式转化成数字格式的采样率，采样率越高，还原后的音质、画质就越好。

### 1.8 常见编码模式

VBR（Variable Bitrate）动态比特率 也就是没有固定的比特率，压缩软件在压缩时根据音频数据即时确定使用什么比特率，这是以质量为前提兼顾文件大小的方式，推荐编码模式；

ABR（Average Bitrate）平均比特率 是VBR的一种插值参数。LAME针对CBR不佳的文件体积比和VBR生成文件大小不定的特点独创了这种编码模式。ABR在指定的文件大小内，以每50帧（30帧约1秒）为一段，低频和不敏感频率使用相对低的流量，高频和大动态表现时使用高流量，可以做为VBR和CBR的一种折衷选择。

CBR（Constant Bitrate），常数比特率 指文件从头到尾都是一种位速率。相对于VBR和ABR来讲，它压缩出来的文件体积很大，而且音质相对于VBR和ABR不会有明显的提高。

### 1.9 帧速率

帧速率也称为FPS(Frames PerSecond)的缩写——帧/秒。是指每秒钟刷新的图片的帧数，也可以理解为图形处理器每秒钟能够刷新几次。越高的帧速率可以得到更流畅、更逼真的动画。每秒钟帧数(FPS)越多，所显示的动作就会越流畅。

### 1.10 分辨率

帧大小每一帧就是一副图像。640\*480分辨率的视频，建议视频的码速率设置在700以上，音频采样率44100就行了。

一个音频编码率为128Kbps，视频编码率为800Kbps的文件，其总编码率为928Kbps，意思是经过编码后的数据每秒钟需要用928K比特来表示。

计算输出文件大小公式：（音频编码率（KBit为单位）/8 +视频编码率（KBit为单位）/8）×影片总长度（秒为单位）=文件大小（MB为单位）。不同的编码需要不同的系统资源，大概可以认为是H.264>VC-1>MPEG2。相对于MPEG2，VC-1的压缩比更高，但相对于H.264而言，编码解码的计算则要稍小一些，一般来说，VC-1多为 “.wmv”后缀，但这都不是绝对的，具体的编码格式还是要通过软件来查询。

总的来说，从压缩比上来看，H.264的压缩比率更高一些，也就是同样的视频，通过H.264编码算法压出来的视频容量要比VC-1的更小，但是VC-1 格式的视频在解码计算方面则更小一些，一般通过高性能的CPU就可以很流畅的观看高清视频。相信这也是目前NVIDIA Geforce 8系列显卡不能完全解码VC-1视频的主要原因。

### 1.11 高清视频

目前的高清视频编码格式主要有H.264、VC-1、MPEG-2、MPEG-4、DivX、XviD、WMA-HD以及X264。

事实上，现在网络上流传的高清视频主要以两类文件的方式存在：一类是经过MPEG-2标准压缩，以tp和ts为后缀的视频流文件;一类是经过WMV-HD(Windows Media Video HighDefinition)标准压缩过的wmv文件，还有少数文件后缀为avi或mpg，其性质与wmv是一样的。真正效果好的高清视频更多地以H.264与VC-1这两种主流的编码格式流传。一般来说，H.264格式以“.avi”、“.mkv”以及“.ts”封装比较常见。

### 1.12 QP

介于0~31之间，值越小，量化越精细，图像质量就越高，而产生的码流也越长。

### 1.13 PSNR

允许计算峰值信噪比(PSNR,Peak signal-to-noise ratio),编码结束后在屏幕上显示PSNR计算结果。开启与否与输出的视频质量无关，关闭后会带来微小的速度提升。

### 1.14 profile level

 MPEG-2按压缩比大小的不同分成五个档次（profile），每一个档次又按图像清晰度的不同分成四种图像格式，或称为级别（level）。五个档次四种级别共有20种组合，但实际应用中有些组合不太可能出现，较常用的是11种组合（下表中标识“√”的项）。这11种组合分别应用在不同的场合，如MP@ML。

分别是BP、EP、MP、HP：  
　　1、BP-Baseline Profile：基本画质。支持I/P 帧，只支持无交错（Progressive）和CAVLC；  
　　2、EP-Extended profile：进阶画质。支持I/P/B/SP/SI 帧，只支持无交错（Progressive）和CAVLC；  
　　3、MP-Main profile：主流画质。提供I/P/B 帧，支持无交错（Progressive）和交错（Interlaced），也支持CAVLC 和CABAC 的支持；  
　　4、HP-High profile：高级画质。在main Profile 的基础上增加了8x8内部预测、自定义量化、无损视频编码和更多的YUV 格式；

H.264规定了三种档次，每个档次支持一组特定的编码功能，并支持一类特定的应用。

1）基本档次：利用I片和P片支持帧内和帧间编码，支持利用基于上下文的自适应的变长编码进行的熵编码（CAVLC）。主要用于可视电话、会议电视、无线通信等实时视频通信；

2）主要档次：支持隔行视频，采用B片的帧间编码和采用加权预测的帧内编码；支持利用基于上下文的自适应的算术编码（CABAC）。主要用于数字广播电视与数字视频存储；

3）扩展档次：支持码流之间有效的切换（SP和SI片）、改进误码性能（数据分割），但不支持隔行视频和CABAC。主要用于网络的视频流，如视频点播。

### 1.15 Reference

Reference：指两个P帧之间的距离。

### 1.16 主码流/副码流

主码流/副码流：主码流位率高，图像质量高，便于本地存储；副码流位率低，图像质量低，便于网络传输。

## 音视频入门

### 音视频通信流程



### 2.2视音频格式简介

视频、图像、音频的格式。这既包括MP4，FLV这样的封装格式，也包括H264，MPEG2这样的编码格式，还包括420 422 444这样的取样格式，甚至RGB，HSV，YUV这样的彩色空间也可以算作一种格式，将来在应用时还可能会遇到RTMP，HLS这样的流媒体传输协议或者其他的协议，都可以算作格式。

### 2.3 DTS&PTS

# 二．音视频传输封装格式

MPEG2系统任务包括：1. 规定以包传输数据的协议；2. 规定收发两端数据流同步的协议；3. 提供多个数据流的复用和解复用协议；3. 提供数据流加密的协议。以包形式存储和传送数据流是MPEG2系统之要点。

## 1. MPEG2 PS&TS

### 1.1简介

PS&TS：两种视频或影片封装格式，常用于高清片。扩展名分别为VOB/EVO和TS等；其文件编码一般用MPEG2/VC-1/H.264。TS流与PS流的区别在于TS流的包结构是固定长度的，而PS流的包结构是可变长度的。

MPEG2(PS)就是常见dvd上那种vob格式的（PS是Program Stream程序流缩写），TS的全称则是Transport Stream（传输流），TS格式在网络上现在主要在一些高清视频，至pal和ntsc是两种不同的制式,二者在分辨率和帧数上有区别。MPEG2-PS主要应用于存储的具有固定时长的节目，如DVD电影，而MPEG-TS则主要应用于实时传送的节目，比如实时广播的电视节目。这两种格式的主要区别是什么呢？简单地打个比喻说，你将DVD上的VOB文件的前面一截cut掉（或者干脆就是数据损坏），那么就会导致整个文件无法解码了，而电视节目是你任何时候打开电视机都能解码（收看）的，所以，MPEG2-TS格式的特点就是要求从视频流的任一片段开始都是可以独立解码的。

PS包与TS包在结构上的这种差异，导致了它们对传输误码具有不同的抵抗能力，因而应用的环境也有所不同。TS码流由于采用了固定长度的包结构，当传输误码破坏了某一TS包的同步信息时，接收机可在固定的位置检测它后面包中的同步信息，从而恢复同步，避免了信息丢失。而PS包由于长度是变化的，一旦某一PS包的同步信息丢失，接收机无法确定下一包的同步位置，就会造成失步，导致严重的信息丢失。因此，在信道环境较为恶劣，传输误码较高时，一般采用TS码流；而在信道环境较好，传输误码较低时，一般采用PS码流。

ES是直接从编码器出来的数据流，可以是编码过的视频数据流，音频数据流，或其他编码数据流的统称。ES流经过PES打包器之后，被转换成PES包。PES包由包头和payload组成。PTS/DTS是打在PES包里面的，这两个parameters是解决视音频同步显示，防止解码器输入缓存上溢或下溢的关键。PTS表示显示单元出现在系统目标解码器(STD: system target

decoder)的时间，DTS表示将存取单元全部字节从STD的ES解码缓存器移走的时刻。每个I、P、B帧的包头都有一个PTS和DTS，但PTS与DTS对B帧都是一样的，无须标出B帧的DTS。对I帧和P帧，显示前一定要存储于视频解码器的重新排序缓存器中，经过延迟（重新排序）后再显示，一定要分别标明PTS和DTS。

ES首先需打包成PES流包，然后PES根据需要打包成PS或TS包进行存储或传输。其每路ES只包含一路信源的编码数据流，所以每路PES也只包含相对应信源的数据流。

对PS流而言，每个PES包头含有PTS和DTS，流识别码，用于区别不同性质ES。然后通过PS复用器将PES包复用成PS包。实际上是将PES包分解为更细小的PS包。在解码的时候，解复用器将PS分解成一个个PES包，拆包器然后将PES包拆成视频和音频的ES，最后输入至各自解码器进行解码。一个问题是：各个ES在解码时，如何保证视音频的同步呢？除了PTS和DTS的配合工作外，还有一个重要的参数是SCR(system clockreference)。在编码的时候，PTS，DTS和SCR都是由STC(system timeclock)生成的，在解码时，STC会再生，并通过锁相环路（PLL－phase lockloop），用本地SCR相位与输入的瞬时SCR相位锁相比较，以确定解码过程是否同步，若不同步，则用这个瞬时SCR调整27MHz的本地时钟频率。最后，PTS，DTS和SCR一起配合，解决视音频同步播放的问题。

TS流也是由一个或多个PES组合而来的，他们可以具有相同的时间基准，也可以不同。其基本的复用思想是，对具有相同时间基准的多个PES现进行节目复用，然后再对相互有独立时间基准的各个PS进行传输复用，最终产生出TS。

#### 1.1.1参考链接：

1.PS流的格式和解析总结

https://blog.csdn.net/rankun1/article/details/69950873

2.MPEG-2 PS流

https://blog.csdn.net/duanbeibei/article/details/1698183

### 1.2 PS流格式

将具有共同时间基准的一个或多个PES组合（复合）而成的单一的数据流称为节目流（Program Stream）。ES是直接从编码器出来的数据流，可以是编码过的视频数据流，音频数据流，或其他编码数据流的统称。ES流经过PES打包器之后，被转换成PES包。

针对H264 做PS 封装：每个IDR NALU 前一般都会包含SPS、PPS 等NALU，因此将SPS、PPS、IDR 的NALU 封装为一个PS 包，包括ps 头，然后加上PS system header，PS system map，PES header+h264 raw data。所以一个IDR NALU PS 包由外到内顺序是：

PSheader | PS system header | PS system Map | PES header | h264 raw data。

对于其它非关键帧的PS 包，就简单多了，直接加上PS头和PES 头就可以了。顺序为：

PS header | PES header | h264raw data。

以上是对只有视频video 的情况，如果要把音频Audio也打包进PS 封装，也可以。当有音频数据时，将数据加上PES header 放到视频PES 后就可以了。顺序如下：PS 包=PS头|PES(video)|PES(audio)，再用RTP 封装发送就可以了。

PES包长度域有两个字节，共16比特，因此PES包的最大长度是65535字节。

#### 1.2.1 PS包结构

PS包结构由包头、系统头、系统映射、PES包四部分构成。具体参考以上。

包头由PS包起始码、系统时钟基准（SCR-System Clock Reference）的基本部分、SCR的扩展部分和PS复用速率4部分组成。

结构如下：

#### IMG_2561.2.2 PS包头PSheader

位串，用来标志一个包的开始PACK identifier -- 0xBA，marker\_bit=1



SCR和SCR\_ext：是系统时钟参考，一个以27MHz驱动的计数器，用作同步流的参考。时钟除以300以上（以匹配PTS / DTS等90KHz时钟），商为SCR（33位），余数余数为SCR\_ext（9位）

Program\_Mux\_Rate：这是一个22位整数，指定程序流目标解码器在包含它的包中接收节目流的速率。 program\_mux\_rate的值以50字节/秒为单位进行测量。值0被禁止。该字段所表示的值用于在2.5.2中定义P-STD输入端的字节到达时间。该字段值在本标准中的节目多路复合流的不同包中取值可能不同。

pack\_stuffing\_length：一个3位整数，指定此字段后面的填充字节数。

stuffing\_byte：填充字节，这是一个固定的8位值，等于'1111 1111'，可以由编码器插入，例如，以满足通道的要求。它被解码器丢弃。

#### 1.2.3 PS系统头 PSsystem header：

位串，BB指出系统标题的开始

|  |  |  |
| --- | --- | --- |
| 语  法 | 位数 | 助记符 |
| system\_header() { |  |  |
| system\_header\_start\_code | 32 | bslbf |
| header\_length | 16 | uimsbf |
| marker\_bit | 1 | bslbf |
| rate\_bound | 22 | uimsbf |
| marker\_bit | 1 | bslbf |
| audio\_bound | 6 | uimsbf |
| fixed\_flag | 1 | bslbf |
| CSPS\_flag | 1 | bslbf |
| system\_audio\_lock\_flag | 1 | bslbf |
| system\_video\_lock\_flag | 1 | bslbf |
| marker\_bit | 1 | bslbf |
| vedio\_bound | 5 | uimsbf |
| packet\_rate\_restriction\_flag | 1 | bslbf |
| reserved\_bits | 7 | bslbf |
| while (nextbits()=='1') { |  |  |
| stream\_id | 8 | uimsbf |
| '11' | 2 | bslbf |
| P-STD\_buffer\_bound\_scale | 1 | bslbf |
| P-STD\_buffer\_size\_bound | 13 | uimsbf |
| } |  |  |
| } |  |  |

header\_length：标题长度字段，16位字段。指出该字段后的系统标题的字节长度。在本规范将来的扩充中可能扩展该字段。

rate\_bound ：速率界限字段，22位字段，取值不小于编码在节目流的任何包中的program\_mux\_rate字段的最大值。该字段可被解码器用于估计是否有能力对整个流解码。

audio\_bound：音频界限字段，6位字段，取值是在从0到32的闭区间中的整数，且不小于节目流中解码过程同时活动的GB/T XXXX.3和GB/T AAAA.3音频流的最大数目。在本小节中，若STD缓冲区非空或展现单元正在P-STD模型中展现，则GB/T XXXX.3和GB/T AAAA.3音频流的解码过程是活动的。

fixed\_flag：固定标志字段，1位标志位。置'1'时表示比特率恒定的操作；置'0'时，表示操作的比特率可变。在恒定比特率的操作期间，复合的GB/T XXXX.1流中的system\_clock\_reference字段值应遵从下面的线性公式:

SCR\_base(i)＝((c1×i＋c2) DIV 300) % 233 (2-22)

SCR\_ext(i)＝((c1×i＋c2) DIV 300) % 300 (2-23)

其中：

c1 对所有i均有效的实型常数；

c2 对所有i均有效的实型常数；

i 在GB/T XXXX.1复合流中包含任何system\_clock\_reference字段的最后一位的字节索引。

CSPS\_flag：CSPS标志字段，1位字段。置'1'时，节目流符合2.7.9中定义的限制

system\_audio\_lock\_flag：系统音频锁定标志字段，1位字段。表示在系统目标解码器的音频采样率和system\_clock\_frequency之间存在规定的比率。system\_clock\_frequency在2.5.2.1中定义而音频采样率由GB/T XXXX.3规定。如果对节目流中所有音频基本流的所有展现单元，system\_clock\_frequency和实际音频采样率的比例SCASR是恒定的，且对音频流中所指出的标准采样率和下表中数值相等，则该字段只能为'1'。

SCASR＝(system\_clock\_frequency) / audio\_sample\_rate\_in\_the\_P-STD (2-24)

记号X/Y表示实数除法。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 标准音频采  样频率(kHz) | 16 | 32 | 22.05 | 44.1 | 24 | 48 |
| SCASR | 27 000 000  --------  16 000 | 27 000 000  -------  32 000 | 27 000 000  ------  22 050 | 27 000 000  ------  44 100 | 27 000 000  ------  24 000 | 27 000 000  --------  48 000 |

system\_video\_lock\_flag：系统视频锁定标志字段，1位字段。表示在系统目标解码器的视频帧速率和system\_clock\_frequency之间存在规定的比率。system\_clock\_frequency在2.5.2.1中定义而视频帧速率由GB/T XXXX.2规定。如果对GB/T XXXX.1中所有视频基本流的所有展现单元，system\_clock\_frequency和实际视频帧速率的比例SCFR是恒定的，且对视频流中所指出的标准帧速率和下表中数值相等，则该字段只能为'1'。

SCFR＝system\_clock\_frequency / frame\_rate\_in\_the\_P-STD (2-25)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 标准帧速率(Hz) | 23.976 | 24 | 25 | 29.97 | 30 | 50 | 59.94 | 60 |
| SCFR | 1 126 125 | 1 125 000 | 1 080 000 | 900 900 | 900 000 | 540 000 | 450 450 | 450 000 |

比率SCFR的值是精确的。对于23.976，29.97或59.94帧/秒的标准速率，实际的帧速率与标准速率略有不同。

video\_bound：视频界限字段。5位字段，取值是在从0到16的闭区间中的整数且不小于节目流中解码过程同时活动的GB/T XXXX.2和GB/T AAAA.2流的最大数目。在本小节中，若P-STD缓冲区非空或展现单元正在P-STD模型中展现，则GB/T XXXX.2和GB/T AAAA.2视频流的解码过程是活动的。

packet\_rate\_restriction\_flag：分组速率限制标志字段，1位标志位。若CSPS标识为'1'，则该字段表示2.7.9中规定的哪个限制适用于分组速率。若CSPS标识为'0'，则该字段的含义未定义。

reserved\_bits：保留位字段，7位字段。被保留供ISO/IEC将来使用。它的值应为'111 1111'，除非ISO/IEC对它作出其它规定。

stream\_id：流标识字段，8位字段。指示其后的P-STD\_buffer\_bound\_scale和P-STD\_buffer\_size\_bound字段所涉及的流的编码和基本流号码。

若取值'1011 1000'，则其后的P-STD\_buffer\_bound\_scale和P-STD\_buffer\_size\_bound字段指节目流中所有的音频流。

若取值'1011 1001'，则其后的P-STD\_buffer\_bound\_scale和P-STD\_buffer\_size\_bound字段指节目流中所有的视频流。

若stream\_id取其它值，则应该是大于或等于'1011 1100'的一字节值且应根据表2-18解释为流的编码和基本流号码。

节目流中的每个基本流应在每个系统标题中通过这种机制精确地规定一次它的P-STD\_buffer\_bound\_scale和P-STD\_buffer\_size\_bound。

P-STD\_buffer\_bound\_scale：P-STD缓冲区界限比例字段。1位字段。表示用于解释后续P-STD\_buffer\_size\_bound字段的比例系数。若前面的stream\_id表示一个音频流，则该字段值为'0'。若表示一个视频流，则该字段值为'1'。对于所有其它的流类型，该字段值可以为'0'也可以为'1'。

P-STD\_buffer\_size\_bound：P-STD缓冲区大小界限字段。13位无符号整数，取值不小于节目流中流n的所有分组的P-STD缓冲区大小BSn的最大值。若P-STD\_buffer\_bound\_scale的值为'0'，则该字段以128字节为单位来度量缓冲区大小的边界。若P-STD\_buffer\_bound\_scale的值为'1'，则该字段以1024字节为单位来度量缓冲区大小的边界。因此：

if (P-STD\_buffer\_bound\_scale = = 0)

BSn≤P-STD\_buffer\_size\_bound×128

else

BSn≤P-STD\_buffer\_size\_bound×1024

所以对于系统头部的解析，我们一般只要先首先判断是否存在系统头（根据系统头的起始码0x000001BB），然后我们读取系统头的头部长度,即PS SYSTEM HEADER LENGTH部分，然后根据头部的长度，跳过PS系统头。进入下一个部分，即PS 节目流映射头。

#### 1.2.4 PS 映射头 PS system Map

PS流的节目映射流部分(节目流映射)定义

|  |  |  |
| --- | --- | --- |
| 语  法 | 位数 | 助记符 |
| program\_stream\_map() { |  |  |
| packet\_start\_code\_prefix | 24 | bslbf |
| map\_stream\_id | 8 | uimsbf |
| program\_stream\_map\_length | 16 | uimsbf |
| current\_next\_indicator | 1 | bslbf |
| reserved | 2 | bslbf |
| program\_stream\_map\_version | 5 | uimsbf |
| reserved | 7 | bslbf |
| marker\_bit | 1 | bslbf |
| program\_stream\_info\_length | 16 | uimsbf |
| for (i=0;i<N;i++){ |  |  |
| descriptor() |  |  |
| } |  |  |
| elementary\_stream\_map\_length | 16 | uimsbf |
| for (i=0;i<N1;i++){ |  |  |
| stream\_type | 8 | uimsbf |
| elementary\_stream\_id | 8 | uimsbf |
| elementary\_stream\_info\_length | 16 | uimsbf |
| for (i=0;i<N2;i++) { |  |  |
| descriptor() |  |  |
| } |  |  |
| } |  |  |
| CRC\_32 | 32 | rpchof |
| } |  |  |

packet\_start\_code\_prefix：分组起始码前缀字段。24位码。它和跟随其后的map\_stream\_id共同组成一个分组起始码以标志分组的开始。该字段是值为'0000 0000 0000 0000 0000 0001' (0x000001)的位串。

map\_stream\_id：映射流标识字段。8位字段，值为0xBC。

program\_stream\_map\_length：节目流映射长度字段。16位字段。指示紧跟在该字段后的program\_stream\_map中的字节数。该字段的最大值为1018(0x3FA)。

current\_next\_indicator：当前下一个指示符字段。1位字段。置'1'时表示传送的节目流映射当前是可用的。置'0'时表示传送的节目流映射还不可用，但它将是下一个生效的表。

program\_stream\_map\_version：节目流映射版本字段。5位字段，表示整个节目流映射的版本号。一旦节目流映射的定义发生变化，该字段将递增1，并对32取模。在current\_next\_indicator为'1'时，该字段应该是当前适用的节目流映射的版本号；在current\_next\_indicator为'0'时，该字段应该是下一个适用的节目流映射的版本号。

program\_stream\_info\_length：节目流信息长度字段。16位字段，指出紧跟在该字段后的描述符的总长度。

marker\_bit：标记位字段。1位字段，取值为'1'。

elementary\_stream\_map\_length：基本流映射长度字段。16位字段，指出在该节目流映射中的所有基本流信息的字节长度。它只包括stream\_type、elementary\_stream\_id和elementary\_stream\_info\_length字段。(这里注意一下，这里的基本流映射长度，他只包括他后面的指定的那几个定义字段的总和，即从从这个长度，我们可以知道后面他根了几种类型的流定义，因为一种流的这个定义字段：stream\_type(1BYTE)、elementary\_stream\_id(1byte)和elementary\_stream\_info\_length(2byte)字段总和为4个字节，所以用elementary\_stream\_map\_length/4可以得到后面定义了几个流类型信息。)

stream\_type：流类型字段 。8位字段，根据表2-29规定了流的类型。该字段只能标志包含在PES分组中的基本流且取值不能为0x05。

(这里我们暂时根据国标GB28181中的定义可以知道

1、MPEG-4 视频流： 0x10；2、H.264 视频流： 0x1B；3、SVAC 视频流： 0x80；4、G.711 音频流： 0x90；5、G.722.1 音频流： 0x92；6、G.723.1 音频流： 0x93；7、G.729 音频流： 0x99；8、SVAC音频流： 0x9B。

因为节目映射流字段只有在关键帧打包的时候，才会存在，所以如果要判断PS打包的流编码类型，就根据这个字段来判断。)

elementary\_stream\_id：基本流标识字段。8位字段，指出该基本流所在PES分组的PES分组标题中stream\_id字段的值。（这个字段的定义，其中0x(C0~DF)指音频，0x(E0~EF)为视频）

elmentary\_stream\_info\_length：基本流信息长度字段。16位字段，指出紧跟在该字段后的描述符的字节长度。(即这个类型的流描述长度。这个后面的字段后面的指定长度不在elementary\_stream\_map\_length指定的范围类。)

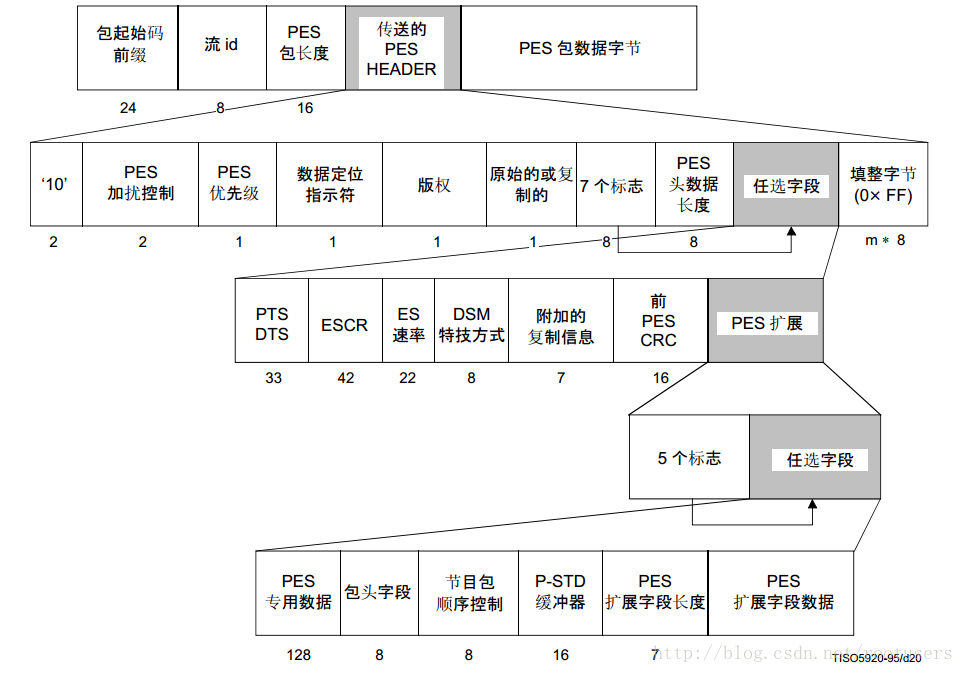
CRC\_32：CRC 32字段。32位字段，它包含CRC值以在处理完整个节目流映射后在附录A中定义的解码器寄存器产生0输出值。

对于这个字段的解析，我们需要取值0x000001BC的位串，指出节目流映射的开始，暂时不需要处理，读取Header Length直接跳过即可，如果需要解析流编码类型，必须详细解析这个字段。

#### 1.2.5 PES包 PES\_packet

PES包=PES header+code raw data。







|  |  |  |
| --- | --- | --- |
| 语  法 | 位数 | 助记符 |
| PES\_packet(){ |  |  |
| packet\_start\_code\_prefix | 24 | bslbf |
| stream\_id | 8 | uimsbf |
| PES\_packet\_length | 16 | uimsbf |
| if(stream\_id != program\_stream\_map |  |  |
| && stream\_id !=padding\_stream |  |  |
| && stream\_id !=private\_stream\_2 |  |  |
| && stream\_id !=ECM |  |  |
| && stream\_id !=EMM |  |  |
| && stream\_id !=program\_stream\_directory |  |  |
| && stream\_id !=DSMCC\_stream |  |  |
| && stream\_id !=ITU-T Rec.H.222.1 type E stream){ |  |  |
| '10' | 2 | bslbf |
| PES\_scrambling\_control | 2 | bslbf |
| PES\_priority | 1 | bslbf |
| data\_alignment\_indicator | 1 | bslbf |
| copyright | 1 | bslbf |
| original\_or\_copy | 1 | bslbf |
| PTS\_DTS\_flags | 2 | bslbf |
| ESCR\_flag | 1 | bslbf |
| ES\_rate\_flag | 1 | bslbf |
| DSM\_trick\_mode\_flag | 1 | bslbf |
| additional\_copy\_info\_flag | 1 | bslbf |
| PES\_CRC\_flag | 1 | bslbf |
| PES\_extension\_flag | 1 | bslbf |
| PES\_header\_data\_length | 8 | uimsbf |
| if(PTS\_DTS\_flags =='10'){ |  |  |
| '0010' | 4 | bslbf |
| PTS[32..30] | 3 | bslbf |
| marker\_bit | 1 | bslbf |
| PTS[29..15] | 15 | bslbf |
| marker\_bit | 1 | bslbf |
| PTS[14..0] | 15 | bslbf |
| marker\_bit | 1 | bslbf |
| } |  |  |
| if(PTS\_DTS\_flags =='11'){ |  |  |
| '0011' | 4 | bslbf |
| PTS[32..30] | 3 | bslbf |
| marker\_bit | 1 | bslbf |
| PTS[29..15] | 15 | bslbf |
| marker\_bit | 1 | bslbf |
| PTS[14..0] | 15 | bslbf |
| marker\_bit | 1 | bslbf |
| '0001' | 4 | bslbf |
| PTS[32..30] | 3 | bslbf |
| marker\_bit | 1 | bslbf |
| PTS[29..15] | 15 | bslbf |
| marker\_bit | 1 | bslbf |
| PTS[14..0] | 15 | bslbf |
| marker\_bit | 1 | bslbf |
| } |  |  |
| if(ESCR\_flag =='1'){ |  |  |
| reserved | 2 | bslbf |
| ESCR\_base[32..30] | 3 | bslbf |
| marker\_bit | 1 | bslbf |
| ESCR\_base[29..15] | 15 | bslbf |
| marker\_bit | 1 | bslbf |
| ESCR\_base[14..0] | 15 | bslbf |
| marker\_bit | 1 | bslbf |
| ESCR\_extension | 9 | uimsbf |
| marker\_bit | 1 | bslbf |
| } |  |  |
| if(ES\_rate\_flag =='1'){ |  |  |
| marker\_bit | 1 | bslbf |
| ES\_rate | 22 | uimsbf |
| marker\_bit | 1 | bslbf |
| } |  |  |
| if (DSM\_trick\_mode\_flag =='1'){ |  |  |
| trick\_mode\_control | 3 | uimsbf |
| if ( trick\_mode\_control = =fast\_forward ) { |  |  |
| field\_id | 2 | bslbf |
| intra\_slice\_refresh | 1 | bslbf |
| frequency\_truncation | 2 | bslbf |
| } |  |  |
| else if ( trick\_mode\_control = = slow\_motion ) { |  |  |
| rep\_cntrl | 5 | uimsbf |
| } |  |  |
| else if ( trick\_mode \_control = = freeze\_frame ) { |  |  |
| field\_id | 2 | uimsbf |
| reserved | 3 | bslbf |
| } |  |  |
| else if ( trick\_mode \_control = = fast\_reverse ) { |  |  |
| field\_id | 2 | bslbf |
| intra\_slice\_refresh | 1 | bslbf |
| frequency\_truncation | 2 | bslbf |
| else if ( trick\_mode\_control = = slow\_reverse ) { |  |  |
| rep\_cntrl | 5 | uimsbf |
| } |  |  |
| else |  |  |
| reserved | 5 | bslbf |
| } |  |  |
| if ( additional\_copy\_info\_flag = ='1'){ |  |  |
| marker\_bit | 1 | bslbf |
| additional\_copy\_info | 7 | bslbf |
| } |  |  |
| if (PES\_CRC\_flag==‘1’){ |  |  |
| previous\_PES\_packet\_CRC | 16 | bslbf |
| } |  |  |
| if ( PES\_extension\_flag =='1') { |  |  |
| PES\_private\_data\_flag | 1 | bslbf |
| pack\_header\_field\_flag | 1 | bslbf |
| program\_packet\_sequence\_counter\_flag | 1 | bslbf |
| P-STD\_buffer\_flag | 1 | bslbf |
| reserved | 3 | bslbf |
| PES\_extension\_flag\_2 | 1 | bslbf |
| if(PES\_private\_data\_flag =='1'){ |  |  |
| PES\_private\_data | 128 | bslbf |
| } |  |  |
| if (pack\_header\_field\_flag == '1'){ |  |  |
| pack\_field\_length | 8 | uimsbf |
| pack\_header() |  |  |
| } |  |  |
| if (program\_packer\_sequence\_counter\_flag == '1'){ |  |  |
| marker\_bit | 1 | bslbf |
| program\_packet\_sequence\_counter | 7 | uimsbf |
| marker-bit | 1 | bslbf |
| MPEG1\_MPEG2\_indentifier | 1 | bslbf |
| original\_stuff\_length | 6 | uimsbf |
| } |  |  |
| if (P-STD\_buffer\_flag = = '1'({ |  |  |
| '01' | 2 | bslbf |
| P-STD\_buffer\_scale | 1 | bslbf |
| P-STD\_buffer\_size | 13 | uimsbf |
| } |  |  |
| if (PES\_extension\_flag\_2 == '1'{ |  |  |
| marker\_bit | 1 | bslbf |
| PES\_extension\_field\_length | 7 | uimsbf |
| for(i=0;i<PES\_extension\_field\_length;i++){ |  |  |
| reserved | 8 | bslbf |
| } |  |  |
| } |  |  |
| } |  |  |
| for (i=0;i<N1;i++){ |  |  |
| stuffing\_byte | 8 | bslbf |
| } |  |  |
| for (i=0;i<N2;i++){ |  |  |
| PES\_packet\_data\_byte | 8 | bslbf |
| } |  |  |
| } |  |  |
| else if (stream\_id = = program\_stream\_map |  |  |
| || stream\_id = = private\_stream\_2 |  |  |
| || stream\_id = = ECM |  |  |
| || stream\_id = = EMM |  |  |
| || stream\_id = = program\_stream\_directory |  |  |
| || stream\_id = = DSMCC\_stream |  |  |
| || stream\_id = = ITU-T Rec. H.222.1 type E stream ){ |  |  |
| for (i=0;i<PES\_packet\_length;i++){ |  |  |
| PES\_packet\_data\_byte | 8 | bslbf |
| } |  |  |
| } |  |  |
| else if (steam\_id = = padding\_stream){ |  |  |
| for (i=0;i<PES\_packet\_length;i++){ |  |  |
| padding\_byte | 8 | bslbf |
| } |  |  |
| } |  |  |
| } |  |  |

以上都是根据steam\_id进行判断处理

表2-18 Stream\_id赋值

|  |  |  |
| --- | --- | --- |
| stream\_id | 注 | 流 编 码 |
| 1011 1100 | 1 | program\_stream\_map（0xBC） |
| 1011 1101 | 2 | private\_stream\_1(0xBD) |
| 1011 1110 |  | padding\_stream(0xBE) |
| 1011 1111 | 3 | private\_stream-2(0xBF) |
| 110x xxxx |  | GB/T XXXX.3或GB/T AAAA.3音频流编号xxxx(0xC0~0xDF) |
| 1110 xxxx |  | GB/T XXXX.2或GB/T AAAA.2视频流编号xxxx(0xE0~0xEF) |
| 1111 0000 | 3 | ECM\_stream(0xF0) |
| 1111 0001 | 3 | EMM\_stream(0xF1) |
| 1111 0010 | 5 | GB/T XXXX.1附录B或GB/T XXXX.6\_DSMCC\_stream(0xF2) |
| 1111 0011 | 2 | ISO/IEC\_13522\_stream(0xF3) |
| 1111 0100 | 6 | ITU-T Rec. H.222.1类型A |
| 1111 0101 | 6 | ITU-T Rec. H.222.1类型B |
| 1111 0110 | 6 | ITU-T Rec. H.222.1类型C |
| 1111 0111 | 6 | ITU-T Rec. H.222.1类型D |
| 1111 1000 | 6 | ITU-T Rec. H.222.1类型E |
| 1111 1001 | 7 | ancillary\_stream(0xF9) |
| 1111 1010…1111 1110 |  | 保留数据流 |
| 1111 1111 | 4 | program\_stream\_directory(0xFF) |

符号x表示值'0'或'1'均被允许且可产生相同的流类型。流号码由x的取值决定。

注

1 类型为program\_stream\_map的PES分组有唯一的语法，在2.5.4.1中作了规定。

2 类型为private\_stream\_1和ISO/IEC\_13352\_stream的PES分组与GB/T XXXX.2及GB/T XXXX.3音频流服从相同的PES分组语法。

3 类型为private\_stream\_2，ECM\_stream和EMM\_stream的PES分组与private\_stream\_1相似，除了在PES\_packet\_length字段后未规定语法。

4 类型为program\_stream\_directory的PES分组有唯一的语法，在2.5.5中作了规定。

5 类型为DSM\_CC\_stream的PES分组有唯一的语法，在GB/T XXXX.6中作了规定。

6 stream\_id与表2-29中的stream\_type 0x09相关联。

7 stream\_id仅用于PES分组。PES分组在传输流中携带了来源于节目流或GB/T AAAA.1系统流的数据(参见2.4.3.7)。

0x0000 01B9结束，即MPEG\_Program\_end\_code

PES分组中各字段的语义定义：

packet\_start\_code\_prefix：分组起始码前缀字段。24位代码，它和后面的stream\_id构成了标识分组开始的分组起始码。它是一个值为'0000 0000 0000 0000 0000 0001' (0x000001)的位串。

stream\_id：流标识字段。在节目流中，它规定了基本流的号码和类型。定义参见表2-18。在传输流中，它可以被设定为正确描述表2-18中定义的基本流类型的任何有效值。在传输流中，基本流类型在2.4.4的节目特定信息中作了规定。

PES\_packet\_length：PES分组长度字段。16位字段，指出了PES分组中跟在该字段后的字节数目。值为0表示PES分组长度要么没有规定要么没有限制。这种情况只允许出现在有效负载包含来源于传输流分组中某个视频基本流的字节的PES分组中。

PES\_scrambling\_control：PES加扰控制字段。2位字段，表示PES分组有效负载的加扰方式。当加扰发生在PES层，PES分组标题，如果有可选字段的话也包括在内，不应被加扰(参见2-19)。

PES\_priority：PES优先级字段。1位字段，指示PES分组中有效负载的优先级。'1'表示PES分组中有效负载的优先级高于该字段为'0'的PES分组有效负载。多路复合器能使用该字段来区分安排基本流中数据的优先级。传输机制不应改动该字段。

data\_alignment\_indicator：数据对齐指示符字段。1位标志。置'1'时表示PES分组标题后紧跟着在2.6.10中的data\_alignment\_indicator所指出的视频起始码或音频同步字，如果有data\_alignment\_indicator描述符的话。若其值为'1'且无该描述符，则需要在表2-47和2-48中alignment\_type '01'所表示的对齐。当值为'0'时，没有定义是否有任何此种的对齐。

Copyright：版权字段。1位字段。置'1'时表示相关PES分组有效负载的材料受到版权保护。当值为'0'时，没有定义该材料是否受到版权保护。2.6.24中描述的版权描述符与包含PES分组的基本流相关。若描述符作用于包含PES分组的材料，则版权标志被置为'1'。

original\_or\_copy：原始或拷贝字段。1位字段。置'1'时表示相关PES分组有效负载的内容是原始的；值为'0'表示相关PES分组有效负载的内容是一份拷贝。

PTS\_DTS\_flags：PTS DTS标志字段。2位字段。当值为'10'时，PTS字段应出现在PES分组标题中；当值为'11'时，PTS字段和DTS字段都应出现在PES分组标题中；当值为'00'时，PTS字段和DTS字段都不出现在PES分组标题中。值'01'是不允许的。

ESCR\_flag：ESCR标志字段。1位标志。置'1'时表示ESCR基础和扩展字段出现在PES分组标题中；值为'0'表示没有ESCR字段。

ES\_rate\_flag：ES速率标志字段。1位标志。置'1'时表示ES\_rate字段出现在PES分组标题中；值为'0'表示没有ES\_rate字段。

DSM\_trick\_mode\_flag：DSM特技方式标志字段。1位标志。置'1'时表示有8位特技方式字段；值为'0'表示没有该字段。

additional\_copy\_info\_flag：附加版权信息标志字段。1位标志。置'1'时表示有附加拷贝信息字段；值为'0'表示没有该字段。

PES\_CRC\_flag：PES CRC标志字段。1位标志。置'1'时表示CRC字段出现在PES分组标题中；值为'0'表示没有该字段。

PES\_extension\_flag ：PES扩展标志字段。1位标志。置'1'时表示PES分组标题中有扩展字段；值为'0'表示没有该字段。

PES\_header\_data\_length：PES标题数据长度字段。8位字段。指出包含在PES分组标题中的可选字段和任何填充字节所占用的总字节数。该字段之前的字节指出了有无可选字段。

marker\_bit：标记位字段。值为'1'的1位字段。

PTS：展现时间戳字段。展现时间与解码时间的关系如下：PTS是一个编码在三个分离字段中的33位数字。它指出了基本流n的第k个展现单元在系统目标解码器中的展现时间tpn(k)。PTS的值以系统时钟频率的1/300(即90 kHz)为单位。展现时间由PTS根据式2-11计算而来。对编码展现时间戳频率的约束参见2.7.4。

PTS(k)＝((system\_clock\_frequency×tpn(k)) DIV 300) % 233 (2-11)

其中，tpn(k)是展现单元Pn(k)的展现时间。对音频而言，若PES分组标题中有PTS，则它是指PES分组中开始的第一个存取单元。若PES分组中有音频存取单元的首字节，则有一个音频存取单元开始于该PES分组中。对视频而言，若PES分组标题中有PTS，则它是指包含PES分组中开始的第一个画面起始码的存取单元。若PES分组中有画面起始码的首字节，则有一个画面起始码开始于该PES分组中。对音频展现单元(PU)，low\_delay序列中的视频PU以及B画面，展现时间tpn(k)应等于tdn(k)。对于非low\_delay中的I画面和P画面，在存取单元(AU) k和k'之间无解码不连续时，展现时间tpn(k)应等于下一个传输的I画面或P画面的解码时间tdn(k) (参见2.7.5)。若有解码不连续或流终止，则tpn(k)和tdn(k)之间的差别应与初始流一直延续，没有不连续也没有终止时完全相同。

注1： low\_delay序列是low\_delay标志被设置的视频序列(参见GB/T XXXX.2中的6.2.2.3)。

若音频中有滤波，则系统模型假定滤波不会导致延迟。因此，编码时PTS所涉及的采样与解码时PTS所涉及的采样是相同的。对于可伸缩编码，参见2.7.6。

DTS：解码时间戳字段。DTS是一个编码在三个分离字段中的33位数字。它指出了基本流n的第j个展现单元在系统目标解码器中的解码时间tdn(j)。DTS的值以系统时钟频率的1/300 (即90 kHz)为单位。解码时间由DTS根据式2-12计算而来：

DTS(j)＝((system\_clock\_frequency×tdn(j)) DIV 300) % 233 (2-12)

其中，tdn(j)是存取单元An(j)的解码时间。

对视频而言，若PES分组标题中有DTS，则它是指包含PES分组中开始的第一个画面起始码的存取单元。若PES分组中有画面起始码的首字节，则该画面起始码开始于该PES分组中。

对于可伸缩编码，参见2.7.6。

ESCR\_base，ESCR\_extension：ESCR字段。42位字段，分两部分编码。第一部分是一个长度为33位的字段，其值ESCR\_base(i)由式2-14给出；第二部分是一个长度为9位的字段，其值ESCR\_ext(i)由式2-15给出。ESCR字段指出了基本流中包含ESCR\_base最后一个比特的字节到达PES-STD输出端的期望时间(参见2.5.2.4)。

特别地

ESCR(i)＝ESCR\_base(i)×300＋ESCR\_ext(i) (2-13)

其中：

ESCR\_base(i)＝((system\_clock\_frequency×t(i)) DIV 300) % 233 (2-14)

ESCR \_ext(i)＝((system\_clock\_frequency×t(i)) DIV 1) % 300 (2-15)

ESCR和ES\_rate字段(参见下面紧接的语义)包含与PES流序列相关的时间信息。这些字段应满足2.7.3中定义的约束。

ES\_rate：基本流速率字段 。22位无符号整数。对于PES流而言，它指出了系统目标解码器接收PES分组的速率。该字段在它所属的PES分组以及同一个PES流的后续PES分组中一直有效，直到遇到一个新的ES\_rate字段。该字段的值以50字节/秒为单位，且不能为0。该字段用于定义PES流的字节到达P-STD输入端的时间(参见2.5.2.4中的定义)。在各个PES分组中，编码在该字段中的值可能不同。

trick\_mode\_control：特技方式控制字段。3位字段。它表示作用于相关视频流的特技方式。对其它类型的基本流，该字段及其后5位的含义没有定义。trick\_mode状态的定义参见2.4.2.3的特技模式部分。当trick\_mode状态为假时，对GB/T XXXX.2视频而言，解码过程输出渐进序列中每幅画面的次数N由repeat\_first\_field和top\_field\_first字段来规定。对GB/T AAAA.2视频而言，由序列标题决定。对于隔行序列，当trick\_mode状态为假时，对GB/T XXXX.2视频而言，次数N由repeat\_first\_field和progressive\_frame字段来规定。

当trick\_mode状态为真时，画面的播放次数依赖于值N。

表2-20 特技方式控制值

|  |  |
| --- | --- |
| 值 | 描  述 |
| '000' | 快进 |
| '001' | 慢动作 |
| '010' | 冻结帧 |
| '011' | 快倒 |
| '100' | 慢倒 |
| '101'－'111' | 保留 |

当该字段值发生变化或特技模式操作停止时，可能会出现下列情况的任意组合：

l 时基不连续；

l 解码不连续；

l 连续性计数器不连续；

在特技模式的情况下，解码和展现的非标准速度可能会导致视频基本流数据中定义的某些字段值不正确。同样，片段结构的语义约束也可能无效。这些例外所涉及的视频语法元素为：

l bit\_rate;

l vbv\_delay;

l repeat\_first\_field;

l v\_axis\_positive;

l field\_sequence;

l subcarrier;

l burst\_amplitude;

l subcarrier\_phase;

在特技模式中，解码器不应该依赖于编码在这些字段中的值。

标准并不要求解码器能解码trick\_mode\_control字段。但是，能解码该字段的解码器应能满足以下标准要求。

fast forword：快进。trick\_mode\_control字段中的值'000'。当该值出现时，它表示一个快进视频流并定义了PES分组标题中后续5位的含义。intra\_slice\_refresh位可以被设定为'1'以指出可能有丢失的宏块。解码器可以用前一个解码画面中相同位置的宏块来代替。表2-21中定义的field\_id字段，表示应该显示哪个或哪些字段。frequency\_truncation字段指出了可能包括的一个系数受限集合。该字段值的含义如表2-22所示。

slow motion：慢动作。trick\_mode\_control字段中的值'001'。当该值出现时，它表示一个慢动作视频流，并定义了PES分组标题中后续5个比特的含义。对渐进序列而言，该画面应被显示N×rep\_cntrl时间，其中N定义如上。

对GB/T AAAA.2视频和GB/T XXXX.2视频渐进序列而言，该画面应被显示N×rep\_cntrl时间，其中N定义如上。

对GB/T XXXX.2隔行序列而言，该画面应被显示N×rep\_cntrl时间。若该画面是一个帧画面，则待显示的第一个字段在top\_field\_first为1时应该是顶字段，在top\_field\_first为0时，应该是底字段(参见GB/T XXXX.2)。该字段被显示N×rep\_cntrl / 2时间。该画面的其它字段被显示N－N×rep\_cntrl / 2时间。

freeze frame：冻结帧。trick\_mode\_control字段中的值'010'。当该值出现时，它表示冻结帧视频流，并定义了PES分组标题中后续5位的含义。表2-21中定义的field\_id字段，表示应该显示哪个(些)字段。field\_id字段指出了包含该字段的PES分组中开始的第一个视频存取单元，除非该PES分组包含0个有效负载字节。在后一种情况下，field\_id字段指出了最近的前一个视频存取单元。

fast reverse：快倒。trick\_mode\_control字段中的值'011'。当该值出现时，它表示一个快倒视频流并定义了PES分组标题中后续5位的含义。intra\_slice\_refresh位可以被设定为'1'以指出可能有丢失的宏块。解码器可以用前一个解码画面中相同位置的宏块来代替。表2-21中定义的field\_id字段，表示应该显示哪个或哪些字段。frequency\_truncation字段指出了可能包括的一个系数受限集合。该字段值的含义如表2-22“系数选择值”所示。

slow reverse：慢倒。trick\_mode\_control字段中的值'100'。当该值出现时，它表示一个慢倒视频流并定义了PES分组标题中后续5位的含义。对GB/T AAAA.2视频和GB/T XXXX.2视频渐进序列而言，该画面应被显示N×rep\_cntrl时间，其中N定义如上。

对GB/T XXXX.2隔行序列而言，该画面应被显示N×rep\_cntrl时间。若该画面是一个帧画面，则待显示的第一个字段在top\_field\_first为1时应该是底字段，在top\_field\_first为'0'时，应该是顶字段(参见GB/T XXXX.2)。该字段被显示N×rep\_cntrl/2时间。该画面的其它字段被显示N－N×rep\_cntrl / 2时间。

field\_id：字段标识字段。2位字段，表示应该显示哪个(些)字段。根据表2-21对其进行编码。

表2-21 field\_id字段控制值

|  |  |
| --- | --- |
| 值 | 描  述 |
| '00' | 仅自顶向下播放 |
| '01' | 仅自底向上播放 |
| '10' | 播放所有帧 |
| '11' | 保留 |

intra\_slice\_refresh：片内参考字段。1位标志。置'1'时表示PES分组的视频数据编码片中可能有丢失的宏块；置'0'时，表示上述情况可能不出现。更多的信息可参见GB/T XXXX.2。解码器可以用前一个解码画面中同一个位置的宏块来代替丢失的宏块。

frequency\_truncation：频率截断字段。2位字段。指出在对PES分组中数据进行编码时可能用到受限系数集合。其值定义于表2-22。

表2-22 系数选择值

|  |  |
| --- | --- |
| 值 | 描述 |
| '00' | 仅DC系数非0 |
| '01' | 仅前三个系数非0 |
| '10' | 仅前六个系数非0 |
| '11' | 所有系数均可能非0 |

rep\_cntrl：显示次数控制字段。5位字段，指出隔行画面中每一字段的显示次数或渐进画面显示次数。对隔行画面而言，顶字段或底字段是否应首先显示是视频序列标题中trick\_mode\_control字段和top\_field\_first字段的功能。该字段值不能为'0'。

additional\_copy\_info：附加版权信息字段。7位字段，包含与版权信息有关的专用数据。

previous\_PES\_packet\_CRC：前PES分组CRC字段。16位字段。在对前一个PES分组(不包括该PES分组的标题)进行处理后，该字段包含一个在解码器的16个寄存器中生成0输出的CRC值。该CRC值与附录A中所定义的相类似，但具有以下多项式：

x16＋x12＋x5＋1

注2： 该CRC值是为了用于网络维护，例如将有间隙性错误的源隔离开来，而不是为了供基本流解码器使用。它仅用于计算数据字节，因为在传输过程中PES分组标题数据可能被修改。

PES\_private\_data\_flag：PES专用数据标志字段。1位标志。置'1'时表示PES分组标题中包含专用数据；置'0'时表示PES分组标题中无专用数据。

pack\_header\_field\_flag：包标题字段标志字段。1位标志。置'1'时表示PES分组标题中有GB/T AAAA.2包标题或节目流包标题。若该字段在包含于节目流中的PES分组中，其值应为'0'。在传输流中，当值为'0'时表示PES标题中无包标题。

program\_packet\_sequence\_counter\_flag：节目分组序列计数标志字段。1位标志。值为'1'时表示PES分组有program\_packet\_sequence\_counter，MPEG1\_MPEG2\_identifier和original\_stuff\_length字段。值为'0'时表示PES分组标题中无这些字段。

P-STD\_buffer\_flag：P-STD缓冲区标志字段。1位标志。置'1'时表示PES分组标题中有P-STD\_buffer\_scale和P-STD\_buffer\_size字段。值为'0'时表示PES标题中无这些字段。

PES\_extension\_flag\_2：PES扩展标志字。1位标志，置'1'时表示有PES\_extension\_field\_length及相关字段。

PES\_private\_data：PES专用数据字段。16位字段。包含专用数据。这些数据与其前后的字段组合在一起时，不能与packet\_start\_code\_prefix (0x000001)冲突。

pack\_field\_length：包字段长度字段。8位字段。表示pack\_header\_field()以字节为单位时的长度。

program\_packet\_sequence\_counter：节目分组序列计数字段。7位字段。它是一个可选的计数器，随着来自于节目流或GB/T AAAA.1流的每一个后续的PES分组或传输流中具有单个节目定义的PES分组而递增，以提供与连续性计数器(参见2.4.3.2)相似的功能。它能用于检索节目流或原始GB/T AAAA.1流中的初始PES分组序列。该计数器在达到最大值后回卷为0。PES分组不能出现重复。因此，复合节目中任何两个连续的PES分组不应具有相同的program\_packet\_sequence\_counter值。

MPEG1\_MPEG2\_identifier：MPEG1 MPEG2标识符字段。1位标志。置'1'时表示PES分组携带的信息来自于GB/T AAAA.1流；置'0'时表示PES分组携带的信息来自于节目流。

original\_stuff\_length：初始填充长度字段。6位字段。指定用于初始GB/T XXXX.1分组标题或初始GB/T AAAA.1分组标题中的填充字节数。

P-STD\_ buffer\_scale：P-STD缓冲区比例字段。1位字段。仅当该PES分组包含于节目流中时才有意义。它指出了用来解释后续P-STD\_buffer\_size字段的比例因子。若前面的stream\_id表示一个音频流，该字段值应为'0'；若前面的stream\_id表示一个视频流，该字段值应为'1'。对于所有的其它流类型，其值可以为'0'或'1'。

P-STD\_buffer\_size：P-STD缓冲区大小字段 。13位无符号整数。仅当该PES分组包含于节目流中时才有意义。它定义了P-STD输入缓冲区的大小BSn。若P-STD\_ buffer\_scale的值为'0'，那么P-STD\_buffer\_size以128字节为单位来度量缓冲区的大小。若P-STD\_buffer\_scale的值为'1'，那么P-STD\_buffer\_size以1024字节为单位来度量缓冲区的大小。因此：

if ( P-STD\_buffer\_scale == 0)

BSn＝P-STD\_buffer\_size×128

else

BSn＝P-STD\_buffer\_size×1024

当该字段被GB/T XXXX.1系统目标解码器收到后，其编码值立即生效。

PES\_extension\_field\_length：PES扩展字段长度字段。7位字段。指出了跟在该字段之后在PES扩展字段中直到且包括任何保留字节的数据的字节长度。

stuffing\_byte ：填充字节字段。8位字段，其值恒定为'1111 1111'。可以由编码器插入以满足通道的需求等。解码器丢弃该字段。一个PES分组标题中只能出现32个填充字节。

PES\_packet\_data\_byte ：PES分组数据字节字段。该字段应该是来自于由分组的stream\_id或PID所指定的基本流的连续数据字节。当基本流数据符合GB/T XXXX.2或GB/T XXXX.3时，该字段应该是与本标准的字节相对齐的字节。基本流的字节序应得到保持。该字段的字节数N由PES\_packet\_length字段规定。N应等于PES\_packet\_length减去在PES\_packet\_length字段的最后一个字节与第一个 PES\_packet\_data\_byte间的字节数。

padding\_byte：填料字节字段 ，8位字段，其值恒定为'1111 1111'。该字段被解码器丢弃。

#### 1.2.6 PES中DTS&PTS

可以看到PTS/DTS（流识别码，用于区别不同性质ES）是打在PES包里面的，这两个参数是解决视音频同步显示，防止解码器输入缓存上溢或下溢的关键。PTS表示显示单元出现在系统目标解码器(STD: system target decoder)的时间，DTS表示将存取单元全部字节从STD的ES解码缓存器移走的时刻。每个I、P、B帧的包头都有一个PTS和DTS，但PTS与DTS对B帧都是一样的，无须标出B帧的DTS。对I帧和P帧，显示前一定要存储于视频解码器的重新排序缓存器中，经过延迟（重新排序）后再显示，一定要分别标明PTS和DTS。

关于音视频的同步：

除了PTS和DTS的配合工作外，还有一个重要的参数是SCR(system clock reference)。在编码的时候，PTS，DTS和SCR都是由STC(system time clock)生成的，在解码时，STC会再生，并通过锁相环路（PLL－phase lock loop），用本地SCR相位与输入的瞬时SCR相位锁相比较，以确定解码过程是否同步，若不同步，则用这个瞬时SCR调整27MHz的本地时钟频率。最后，PTS，DTS和SCR一起配合，解决视音频同步播放的问题。

### 1.3 TS流格式

#### 1.3.1 简介

因为Apple公司提出的HLS(http live streaming)格式的流行，mpeg-ts封装的文件在互联网上已经随处可见。这套体系的强大之处就在于它的简单，只要你有工具可以把一个或多个视频文件切割成一堆的小ts文件，并且生成一个内容非常简单的m3u8文件，然后只要有一个标准的web server（apache, nginx）就能对外提供点播视频流媒体服务了。假如可以对直播数据流进行切片存储成小文件，那就又具备了直播流媒体服务能力。

介绍一个非常好用的解析库bitstream，并提供据此库实现的ts解析工具与大家共同学习探讨。在MPEG-2系统中编码端实现单一码流的详细过程如下：

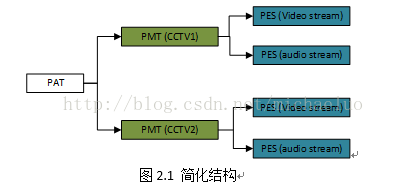
（1）. A/D转换后，通过MPEG-2压缩编码得到ES基本流。该流数据量很大，并且只是I P B帧的取样信息。

（2）.通过PES打包器，打包并在每个帧中插入PTS/DTS标识，编程PES。原来是流的格式，现在成了数据包的分割形式。

（3）.PES根据需要打包成PS或者TS包进行存储

#### IMG_2561.3.2 TS结构

TS流的最小数据单元是188字节，所以可以认为一个TS文件的长度一定是188的整数倍。



## h264

### 3.1

## 3.RTP