<http://blog.csdn.net/alangdangjia/article/details/9495193>

# [MPEG2 PS和TS流格式](http://blog.csdn.net/alangdangjia/article/details/9495193)

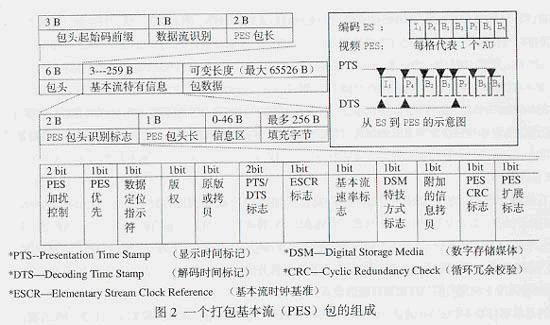
分类： [嵌入式/硬件电路](http://blog.csdn.net/alangdangjia/article/category/1237685)2013-07-26 13:58 3344人阅读 [评论](http://blog.csdn.net/alangdangjia/article/details/9495193#comments)(3) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/alangdangjia/article/details/9495193#report)

    应该说真正了解TS，还是看了朋友推荐的《数字电视业务信息及其编码》一书之后，MPEG2 TS和数字电视是紧密不可分割的，值得总结一下其中的一些关系。

    ISO/IEC－13818－1：  
系统部分；ISO/IEC－13818－2：视频；ISO/IEC－13818－3：音频；ISO/IEC－13818－4：一致性测试；ISO  
/IEC－13818－5：软件部分；ISO/IEC－13818－6：数字存储媒体命令与控制；ISO/IEC－13818－7：高级音频编码；ISO  
/IEC－13818－8：系统解码实时接口；

    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。

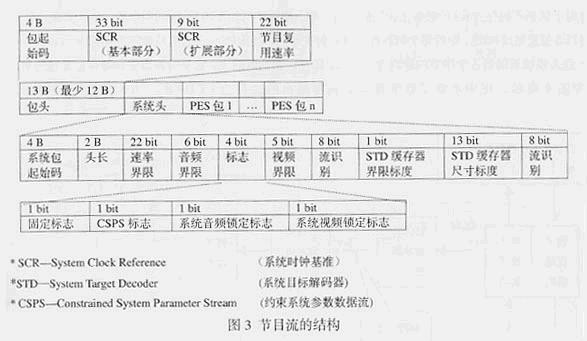
    ES是直接从编码器出来的数据流，可以是编码过的视频数据流，音频数据流，或其他编码数据流的统称。ES流经过PES打包器之后，被转换成PES包。PES包由包头和payload组成，具体格式摘录如下：



    可以看到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。

    上节介绍过，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 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。PS格式摘录如下：



    PS包的长度比较长且可变，主要用于无误码环境里，因为越长的话，同步越困难，且在丢包的情况下，重组也越困难。所以，PS适合于节目信息的编辑和本地内容应用的application。

    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。

    TS包由包头和包数据2部分组成，其中包头还可以包括扩展的自适用区。包头长度占4bytes，自使用区和包数据共占184bytes，整个TS包长度相当于4个ATM包长。TS包的包头由如下图摘录所示的同步字节、传输误码指示符、有效载荷单元起始指示符、传输优先、包识别（PID-Packet Identification）、传输加扰控制、自适应区控制和连续计数器8个部分组成。



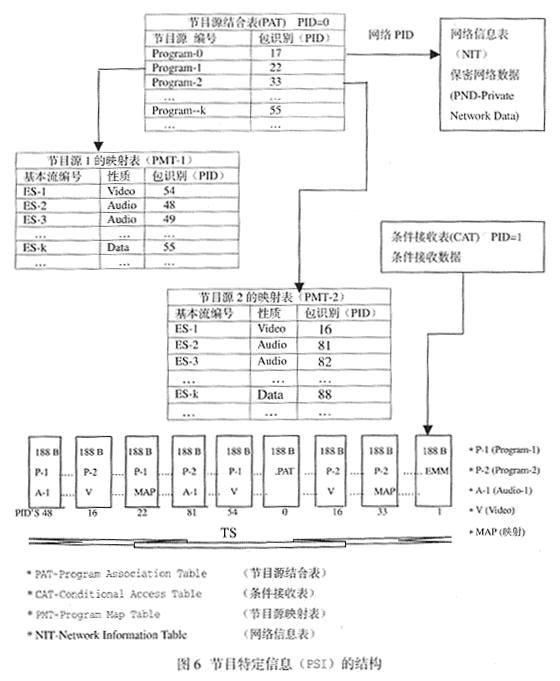
    其中，可用同步字节位串的自动相关特性，检测数据流中的包限制，建立包同步；传输误码指示符，是指有不能消除误码时，采用误码校正解码器可表示1bit 的误码，但无法校正；有效载荷单元起始指示符，表示该数据包是否存在确定的起始信息；传输优先，是给TS包分配优先权；PID值是由用户确定的，解码器根据PID将TS上从不同ES来的TS包区别出来，以重建原来的ES；传输加扰控制，可指示数据包内容是否加扰，但包头和自适应区永远不加扰；自适应区控制，用2 bit表示有否自适应区，即（01）表示有有用信息无自适应区，（10）表示无有用信息有自适应区，（11）表示有有用信息有自适应区，（00）无定义；连续计数器可对PID包传送顺序计数，据计数器读数，接收端可判断是否有包丢失及包传送顺序错误。显然，包头对TS包具有同步、识别、检错及加密功能。

    TS包自适应区由自适应区长、各种标志指示符、与插入标志有关的信息和填充数据4部分组成。其中标志部分由间断指示符、随机存取指示符、ES优化指示符、PCR标志、接点标志、传输专用数据标志、原始PCR标志、自适应区扩展标志8个部分组成。重要的是标志部分的PCR字段，可给编解码器的27MHz时钟提供同步资料，进行同步。其过程是，通过PLL，用解码时本地用PCR相位与输入的瞬时PCR相位锁相比较，确定解码过程是否同步，若不同步，则用这个瞬时PCR调整时钟频率。因为，数字图像采用了复杂而不同的压缩编码算法，造成每幅图像的数据各不相同，使直接从压缩编码图像数据的开始部分获取时钟信息成为不可能。为此，选择了某些（而非全部）TS包的自适应区来传送定时信息。于是，被选中的TS包的自适应区，可用于测定包信息的控制bit和重要的控制信息。自适应区无须伴随每个包都发送，发送多少主要由选中的TS包的传输专用时标参数决定。标志中的随机存取指示符和接点标志，在节目变动时，为随机进入I帧压缩的数据流提供随机进入点，也为插入当地节目提供方便。自适应区中的填充数据是由于PES包长不可能正好转为TS包的整数倍，最后的TS包保留一小部分有用容量，通过填充字节加以填补，这样可以防止缓存器下溢，保持总码率恒定不变。

    前面3节总结了MPEG2  
TS的基本格式，其中包括PES，PS和TS，以及相关字段的介绍。那么作为一种传输流，TS将内容进行打包/复用，让其媒体内容变成TS传输，并最终在  
解码端解码。简单来看，TS是一个传输层的协议栈，它可以承载各种内容的传输，比如MPEG，WMV，H264，甚至是IP，那么其中的传输规范是如何定  
义的呢？这个即是PSI（节目特定信息）要做的事情。

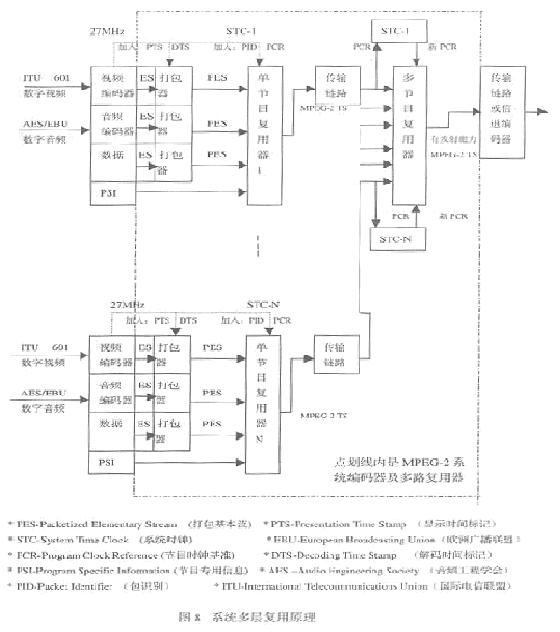
    PSI  
由四张表构成：PAT，PMT，CAT和NIT，这四张表分别描述了一个TS所包括的所有ES流的传输结构。首先的一个概念是，TS是以包形式传播，在编  
解码端都需要以一定的包ID来标识TS流里承载的内容，比如，PAT表会存在于一个或多个TS包里，所以要用一个特别的包ID来表示，另外，不同的ES流  
也需要不同的包ID来标识。我们有了PAT和PMT这两种表，解码器就可以根据PID，将TS上从不同ES来的TS包区分出来进行解码。

    TS的解码分两步进行，其一，是从PID为0  
的TS包里，解析出PAT表，然后从PAT表里找到各个节目源的PID，一般此类节目源都由若干个ES流组成，并描述在PMT表里面，然后通过节目源的  
PID，就可以在PMT表里检索到各个ES的PID。其二，解码器根据PMT表里的ES流的PID，将TS流上的包进行区分，并按不同的ES流进行解码。  
所以，TS是经过节目复用和传输复用两层完成的，即在节目复用时，加入了PMT，在传输复用时，加入了PAT。同样在节目解复用时，可以得到PMT，在传  
输解复用时，可以得到PAT。下图很好地概述了其思想。



    TS是支持多路复用的，所以它可用来传输经复用后的多层节目。在复用过程中，要注意的是，解码过程中所需要面对的时间参考和同步问题，因为解复用是需要各种信息同步进行的，所以在复用过程中，就需要插入相关的时间信息：PTS，DTS，PCR。

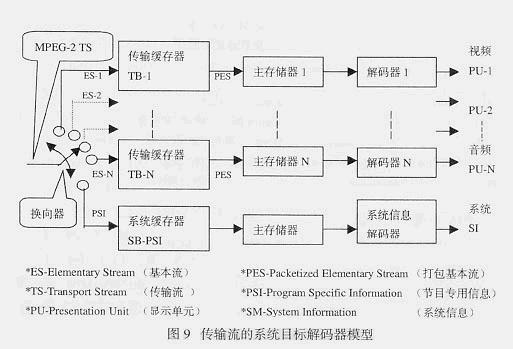
    在TS形成过程中，PTS和DTS是在ES打包成PES时，根据STC的参考，将其时钟信息注入PES包中的，而之后在PES切成TS时，再将PID和  
PCR信息注入到TS包中，当多路TS再进行复用的时候，各路TS的PCR将会被提取出来，再进行分析，然后再根据统一的STC参考，将新的PCR生成并  
注入到TS中去，最后，因为原来PAT表信息不在适用，所以新的PAT表需要再生成，并附加到新的TS流中去。经过这多层的复用之后，新的TS流即可以进  
入调制，传输阶段。过程可参见下图：



    解码过程要面对的问题是：解复用，视音频的同步，解码缓存器无上下溢。解复用即是将TS在同一信道里不同时序进行传输的节目分离出来；视音频同步由DTS,  
PTS和PCR三者协调完成，并且PCR是重建系统时间基准的绝对时标，而DTS和PTS是解码和重现时刻的相对时标；对解码缓存器无上下溢的问题，必须  
借助于系统目标解码器（STD）模型来对其进行实现，基本思想如下：

1. TS流进入解码器后，首先由换向器，按照一定的时序关系，将各种ES流分解出来（其中也包括PSI信息流）。
2. 分解过后的ES流会进入各自的传输缓存器，通过之后，其PES流进入各自的主存储器，注意的是：PSI信息流会进入系统缓存器，最后也到达主存储器。
3. 最后，解码器根据DTS信息，从各个主存储器分别提取媒体或系统信息，进行解码，并根据PTS信息，将媒体内容进行显示处理。

其过程可参见下图：

  
  
  
  
  
  
MPEG-2 学习笔记

  最近有点时间，看了一部分MPEG-2 的规范，看后想总结点东西，算是做了点作业，另外希望能和大家讨论讨论，请大家指点。

中文版很多概念翻译得很模糊，不易理解，但总体来说还算是不错，适合像我这种入门级别的看，不过建议和英文版对照看，对一些概念能比较准确的理解。整个规范包括三部分：系统，视频编码，音频编码。对应的标准号分别为ISO/IEC 13818-1，ISO/IEC 13818-2，ISO/IEC 13818-3，在规范中经常可以看到这几个字符串。

第一部分“系统”和我们现在的工作关系较紧密，我也主要学习了第一部分。后面两部分主要是讲解编码过程，编码部分看了实在让人犯晕，先偷一下懒吧，把第一部分搞清楚了再看去啃难啃的骨头吧。

下面进入正题了。

一、概念

规范中讲述的概念很多，容易让人糊涂，所以先把一些概念理清，弄清楚它们之间的关系，再看后面的就可提高很多的效率。

(1)ES- Elementary Streams (原始流)，对视频、音频信号及其他数据进行编码压缩后

的数据流称为原始流。原始流包括访问单元，比如视频原始流的访问单元就是一副图像的编

码数据。

(2) PES- Packetized Elementary Streams (分组的原始流)，原始流形成的分组称为PES分组，是用来传递原始流的一种数据结构

(3)节目是节目元素的集合。节目元素可能是原始流，这些原始流有共同的时间基点，用来做同步显示。

(4)传输流和节目流

TS-Transport Stream 翻译为“传输流”

PS-Program Stream  翻译为“节目流”

PS用来传输和保存一道节目的编码数据或其他数据。PS的组成单位是PES分组。

TS用来传输和保存多道节目的编码数据或其他数据，TS的组成单位是节目。

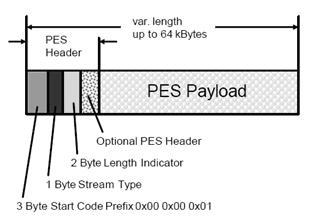
PS适用于不容易发生错误的环境，以及涉及到软件处理的应用，典型应用如DVD光盘的文件存储

TS适用于容易发生错误的环境，典型应用就是数字电视信号的传输。

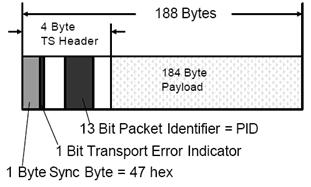
TS和PS是可以互相转换的，比如从TS中抽取一道节目的内容并产生有效的PS是可能。

(5)传输流分组和PES分组

原始流分成很多PES分组，保持串行顺序，一个PES分组只包含一个原始流的编码数据。PES分组长度很大，最大可为64K字节。

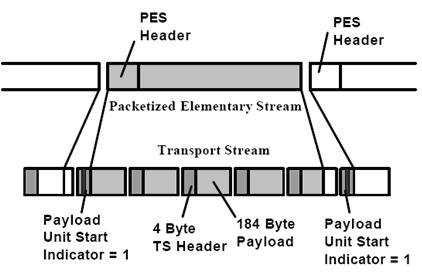
PES分组分为“分组首部(header)”和“有效负载(payload)”。“有效负载”指跟随在首部字节之后的字节。首部的前4个字节构成分组的起始码，标识了该分组所属原始流的类型和ID号。

TS分组也就是传输流数据形成的数据包。每个TS分组长度为188字节，包括“分组首部”和“有效负载，前4个字节是分组首部，包含了这个分组的一些信息。有些情况下需要更多的信息时，需在后面添加“调整字段(adaption field)”。



两者之间的关系：

    PES分组是插入到TS分组中的，每个PES分组首部的第一字节就是TS分组有效负载的第一字节。一个PID值的TS分组只带有来自一个原始流的数据。



(5)PSI

    全称Program Specific Information，意为节目专用信息。传输流中是多路节目复用的，那么，怎么知道这些节目在传输流中的位置，区分属于不同节目呢？所以就还需要一些附加信息，这就是PSI。PSI也是插入到TS分组中的，它们的PID是特定值。

         MPEG-2中规定了4个PSI，包括PAT(节目关联表)，CAT(条件访问表)，PMT(节目映射表)，

NIT(网络信息表)，这些PSI包含了进行多路解调和显示节目的必要的和足够的信

息。

         具体的应用中可能包括更多的信息，比如DVB-T中定义了SDT(服务描述表),EIT(环境信息表),BAT(节目组相关表),TDT(时间日期表)等，统称为DVB-SI(服务信息)。

l    PSI的PID是特定的，含PSI的数据包必须周期性的出现在传输流中。

PMT (Program Map Table )节目映射表

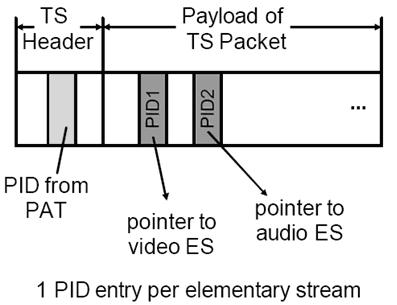
PMT所在分组的PID由PAT指定，所以要先解出PAT，再解PMT

PMT中包含了属于同一节目的视频、音频和数据原始流的PID。

找到了PMT，解多路复用器就可找到一道节目对应的每个原始流的PID，再根据原始流

PID，去获取原始流。如下图：PID1和PID2分别对应某道节目的视频原始流和音频原始流

的PID。



l      PAT (Program Association Table )节目关联表

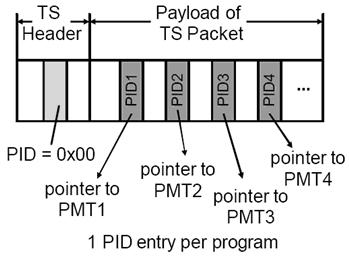
l      PAT所在分组的PID=0

      PAT中列出了传输流中存在的节目流

l

l      PAT指定了传输流中每个节目对应PMT所在分组的PID

l      PAT的第一条数据指定了NIT所在分组的PID ，其他数据指定了PMT所在分组的PID，如下图所示：



lCAT (Conditional Access Table )条件访问表

lCAT所在分组的PID=1

lCAT中列出了条件控制信息(ECM)和条件管理信息(EMM)所在分组的PID。

lCAT用于节目的加密和解密

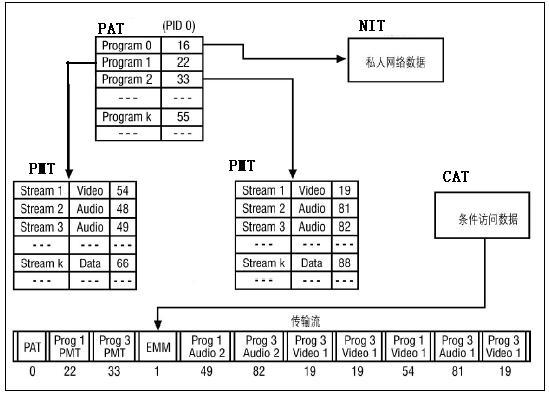
lNIT( Network Information Table)网络信息表

lNIT所在分组的PID由PAT指定

lNIT提供一组传输流的相关信息，以及于网络自身特性相关的信息，比如网络名称，传输参数(如频率,调制方式等)。

lNIT一般是解码器内部使用的数据，当然也可以做为EPG的一个显示数据提供给用户做为参考。

几种PSI之间的关系，如下图所示：首先PAT中指定了传输流中所存在的节目，及每个节目对应的PMT的PID号。 比如Program 1对应的PMT 的PID=22,然后找到PID=22的TS分组，解出PMT，得到这个节目中包含的原始流的PID，再根据原始流的PID去找相应的TS分组，获取原始流的数据，然后就可以送入解码器解码了。



二、数据结构

（1）TS分组

前面提到，TS分组由188个字节构成，其结构如下：

|  |
| --- |
| transport\_packet()  {   sync\_byte                                                                    // 8  transport\_error\_indicator                                          //1  payload\_unit\_start\_indicator                                    //1  transport\_priority                                                       // 1  PID                                                                             //13  transport\_scrambling\_control                                  // 2 adaptation\_field\_control                                            //2  continuity\_counter                                                      //4  if(adaptation\_field\_control=='10'  || adaptation\_field\_control=='11'){     adaptation\_field()   }   if(adaptation\_field\_control=='01' || adaptation\_field\_control=='11') {     for (i=0;i<N;i++){      data\_byte                                                                   //8    }   }   } |

前面32bit的数据即TS分组首部，它指出了这个分组的属性。

**sync\_byte**  同步字节，固定为0x47 ，表示后面的是一个TS分组，当然，后面包中的数据是不会出现0x47的

**transport\_error\_indicator** 传输错误标志位，一般传输错误的话就不会处理这个包了

**payload\_unit\_start\_indicator** 这个位功能有点复杂，字面意思是有效负载的开始标志，根据后面有效负载的内容不同功能也不同，后面用到的时候再说。

**transport\_priority**  传输优先级位，1表示高优先级，传输机制可能用到，解码好像用不着。

**PID**  这个比较重要，指出了这个包的有效负载数据的类型，告诉我们这个包传输的是什么内容。前面已经叙述过。

**transport\_scrambling\_control**加密标志位，表示TS分组有效负载的加密模式。TS分组首部(也就是前面这32bit)是不应被加密的，00表示未加密。

**adaption\_field\_control**  翻译为“调整字段控制”，表示TS分组首部后面是否跟随有调整字段和有效负载。01仅含有效负载，10仅含调整字段，11含有调整字段和有效负载。为00的话解码器不进行处理。空分组没有调整字段。

**continuity\_counter**  一个4bit的计数器，范围0-15，具有相同的PID的TS分组传输时每次加1，到15后清0。不过，有些情况下是不计数的。如下：(1)TS分组无有效负载(2)复制的TS分组和原分组这个值一样(3)后面讲到的一个标志discontinuity\_indicator为1时

**adaptation\_field()**        调整字段的处理

**data\_byte**                      有效负载的剩余部分，可能为PES分组，PSI，或一些自定义的数

                                            据。

(2)PAT

  PAT数据结构如下：

|  |
| --- |
| program\_association\_section() {  table\_id                                                                    // 8 section\_syntax\_indicator                                      //1  '0'                                                                           //1 reserved                                                              // 2 section\_length                                                    //12 transport\_stream\_id                                         // 16 reserved                                                           // 2 version\_number                                                 // 5 current\_next\_indicator                                     //1 section\_number                                                //8 last\_section\_number                                         // 8 for (i=0; i<N;i++) {    program\_number                                              // 16   reserved                                                          // 3   if(program\_number == '0') {     network\_PID                                                  //  13   }    else {     program\_map\_PID                                           // 13   }  }  CRC\_32                                                            //  32 } |

**table\_id**  固定为0x00 ，标志是该表是PAT

**section\_syntax\_indicator**段语法标志位，固定为1

**section\_length**         表示这个字节后面有用的字节数，包括CRC32。假如后面的字节加上前面的字节数少于188，后面会用0XFF填充。假如这个数值比较大，则PAT会分成几部分来传输。

**transport\_stream\_id**    该传输流的ID，区别于一个网络中其它多路复用的流。

**version\_number**       范围0-31，表示PAT的版本号，标注当前节目的版本．这是个非常有用的参数，当检测到这个字段改变时，说明TS流中的节目已经变化了，程序必须重新搜索节目．

**current\_next\_indicator**表示发送的PAT是当前有效还是下一个PAT有效。

**section\_number**分段的号码。PAT可能分为多段传输，第一段为00，以后每个分段

加1，最多可能有256个分段

**last\_section\_number**最后一个分段的号码

**program\_number**节目号

**network\_PID**网络信息表（NIT）的PID，网络信息表提供了该物理网络的一些信息，和电视台相关的。节目号为0时对应的PID为network\_PID

**program\_map\_PID**节目映射表的PID，节目号大于0时对应的PID，每个节目对应一个

**CRC\_32**   CRC32校验码

上面program\_number，network\_PID，program\_map\_PID 是循环出现的。program\_number等于0时对应network\_PID，program\_number等于其它值时对应program\_map\_PID。

举个例子，下述流为带PAT的TS分组：

**47 40 00 1c 00 00 b0 15 13 f6 e7 00 00 00 00 e0 10 00 01 e0 20 00 02 e0 21 1a 34 b4 77 ff…………..ff**

其中红色的四个字节是TS分组头部，用数据结构解出首部，得到PID=0x00，表示为该分组的有效负载是PAT。蓝色的00称为“指针域”----Pointer field，表示了一个偏移量，即从后面第几个字节开始是PAT部分。为00表示后面紧接着的就是PAT：**00 b0 15 13 f6 e7 00 00 00 00 e0 10 00 01 e0 20 00 02 e0 21 1a 34 b4 77**

再利用PAT的数据结构解出PAT，得到如下信息：

---------------PAT Information-------------

table\_id: 00

section\_syntax\_indicator: 01

section\_length: 0015

transport\_stream\_id: 13f6

version\_number: 13

current\_next\_indicator: 01

section\_number: 00

last\_section\_number: 00

program\_number: 0000

network\_PID: 0010

program\_number: 0001

program\_map\_PID: 0020

program\_number: 0002

program\_map\_PID: 0021

CRC\_32: 1a34b477

可以看出，此PAT只有一段，包含了三个节目，节目号0000对应于network\_PID=0010 ，节目号0001对应于program\_map\_PID =0020，节目号0002对应于program\_map\_PID =0021，从实际的角度，我们应该把这三个节目号理解为三个频道，第一个频道中的内容是网络信息，第二、三个频道包含了节目信息。在数字电视中，一个频道即对应于一个频点，如498MHZ，一个频道上可以有多个节目，后面的PMT即是告诉了我们某个频道中所有节目对应的PID。

于是现在就搜寻PID=0x0020的TS分组，即是频道2对应的PMT信息。

(其实楼主的理解不是完全正确，transport\_stream\_id标识了一个唯一的传输流（每一个传输流对应一个频点，如498MHz），一个PAT表表示一个流里面的信息。  
上面的三个节目号理解为三个频道是正确的，但在数字电视中，一个频道对应的也是一个节目，而不是一个频点（当然节目号为0时对应的是NIT的PID）。)

(3)PMT

  PMT数据结构如下：

|  |
| --- |
| TS\_program\_map\_section() {  table\_id                                                                              // 8 section\_syntax\_indicator                                                  //1 '0'                                                                                      //  1 reserved                                                                           //    2 section\_length                                                                   //  12 program\_number                                                               //16 reserved                                                                              // 2 version\_number                                                                  //5 current\_next\_indicator                                                       //1 section\_number                                                                  // 8 last\_section\_number                                                          //8 reserved                                                                               //3 PCR\_PID                                                                             //13 reserved 4 program\_info\_length                                                          //12 for (i=0; i<N; i++) {    descriptor()  }  for (i=0;i<N1;i++) {    stream\_type                                                                        //8   reserved                                                                               //3   elementary\_PID                                                                  //13   reserved                                                                               //4   ES\_info\_length                                                                   //12   for (i=0; i<N2; i++) {     descriptor()    }  }  CRC\_32                                                                               //32 } |

**table\_id**  固定为0x02 ，标志是该表是PMT

**section\_syntax\_indicator**

**section\_length**

**version\_number**

**current\_next\_indicator**以上四个字段意思和PAT相同，可参考上面解释

**section\_number**

**last\_section\_number**以上两个字段意思和PAT相同，不过值都固定为0x00，我觉得这样的原因可能是因为PMT不需要有先后顺序，因为先定义哪个节目都是无所谓。

**program\_number**节目号，表示该PMT对应的节目

**PCR\_PID**PCR（节目时钟参考）所在TS分组的PID，根据PID可以去搜索相应的TS分组，解出PCR信息。

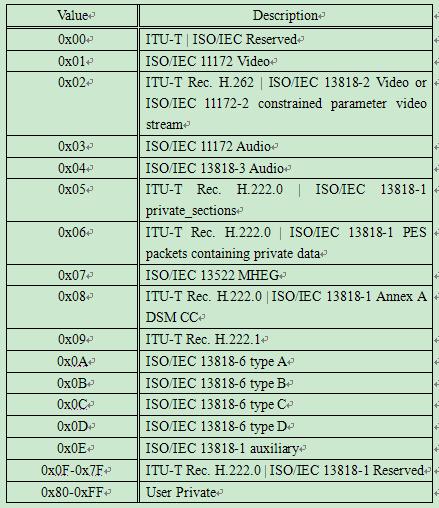
**program\_info\_length**该节目的信息长度，在此字段之后可能会有一些字节描述该节目的信息

**stream\_type**指示了PID为elementary\_PID的PES分组中原始流的类型，比如视频流，音频流等，见后面的表

**elementary\_PID**该节目中包括的视频流，音频流等对应的TS分组的PID

**ES\_info\_length**该节目相关原始流的描述符的信息长度。

stream\_type对应的类型：



还是举个例子，下述是一个包含PMT的TS分组，

**47 40 20 1c 00 02 b0 1f 00 01 e7 00 00 e1 00 f0 00 02 e1 00 f0 05 02 03 b2 44 5f 04 e1 10 f0 03 03 01 67 c9 ab c8 d2**

红色的四个字节是TS分组头部，蓝色的00是“指针域”，意义同PAT中的指针域。所以下面的数据就是PMT的内容：**02 b0 1f 00 01 e7 00 00 e1 00 f0 00 02 e1 00 f0 05 02 03 b2 44 5f 04 e1 10 f0 03 03 01 67 c9 ab c8 d2**

再解出PMT，得到下列信息：

table\_id: 02

section\_syntax\_indicator: 01

section\_length: 01f

program\_number: 0001

version\_number: 13

current\_next\_indicator: 01

section\_number: 00

last\_section\_number: 00

PCR\_PID: 0100

program\_info\_length: 000

descriptor:

steam\_type: 00

elementary\_PID: 0001

ES\_info\_length: 000

descriptor:

steam\_type: 02

elementary\_PID: 0001

ES\_info\_length: 005

descriptor: 02 03 b2 44 5f

steam\_type: 04

elementary\_PID: 0011

ES\_info\_length: 003

descriptor: 03 01 67

CRC\_32: c9abc8d2

可以看出，该节目号0001包含了三个流的信息，流类型分别为00，02，04，00的流为保留值，可以不考虑，02表示原始流为视频流，其elementary\_PID为0001，04表示原始流为音频流，其elementary\_PID为0011，两个流分别还带有descriptor（描述符），说明了该原始流的一些信息。

得到了这个elementary\_PID，再从后面的传输流中找到PID为这个值的TS分组，其有效负载即为这个原始流的数据，获取数据送到解码器，即可还原这个视频或音频了。

三、总结

上面的都是一些零散的知识，跟我们实际应用有什么关系呢？下面就是一个简易的应用过程---搜台。搜台过程大致如下：

先调整高频头到一个固定的频率(如498MHZ),如果此频率有数字信号,则相关芯片会自动把TS流数据传送给MPEG- 2 decoder. MPEG-2 decoder先进行数据的同步,也就是等待完整的Packet的到来.然后循环查找是否出现PID== 0x0000的Packet,如果出现了,则马上进入分析PAT的处理,获取了所有的PMT的PID.接着循环查找是否出现PMT,如果发现了,则自动进 入PMT分析,获取该频段所有的频道数据并保存.如果没有发现PAT或者没有发现PMT,说明该频段没有信号,进入下一个频率扫描。

上述过程主要涉及到PAT和PMT的一些解码和解复用知识，这也是目前我学习到的，当然，数字电视涉及到的知识远远不止这些，解码方面就还包括调整字段的处理，SI(业务信息)应用，时钟的处理，CA加密解MI系统等，还需要继续的学习和实践。

<http://blog.csdn.net/alangdangjia/article/details/9495171>

# [PS流格式](http://blog.csdn.net/alangdangjia/article/details/9495171)

分类： [嵌入式/硬件电路](http://blog.csdn.net/alangdangjia/article/category/1237685)2013-07-26 13:55 357人阅读 [评论](http://blog.csdn.net/alangdangjia/article/details/9495171#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/alangdangjia/article/details/9495171#report)

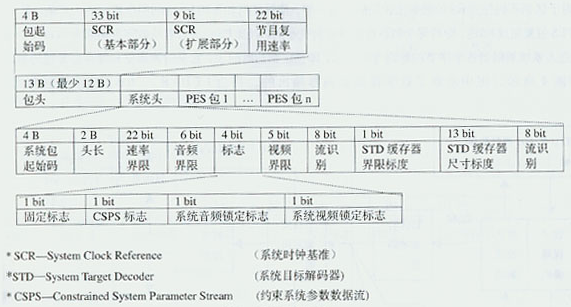
**概念：**

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。

**构成：**

PS包由包头、系统头、PES包3部分构成。**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。



维基百科对应的图表（包头、系统头）：





**字节顺序，如下所示：**

**4B的包起始码：**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **byte 0** | | | | | | | | **byte 1** | | | | | | | | **byte 2** | | | | | | | | **byte 3** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **0000 0000 0000 0000 0000 0001 start code** | | | | | | | | | | | | | | | | | | | | | | | | 1011 1010 PACK identifier | | | | | | | |

PACK identifier -- 0xBA

**系统时钟基准（SCR-System Clock Reference）的基本部分**、**SCR的扩展部分**：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **byte 4** | | | | | | | | **byte 5** | | | | | | | | **byte 6** | | | | | | | | **byte 7** | | | | | | | | **byte 8** | | | | | | | | **byte 9** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **01** | | SCR 32..30 | | | **1** | SCR 29..15 | | | | | | | | | | | | | | | **1** | SCR 14..00 | | | | | | | | | | | | | | | **1** | SCR\_ext | | | | | | | | | **1** |

**PS复用速率：**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **byte 10** | | | | | | | | **byte 11** | | | | | | | | **byte 12** | | | | | | | | **byte 13** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Program\_Mux\_Rate | | | | | | | | | | | | | | | | | | | | | | **1** | **1** | reserved | | | | | pack\_stuffing\_length | | |

* SCR and SCR\_ext together are the System Clock Reference, a counter driven at 27MHz, used as a reference to synchronize streams. The clock is divided by 除以300 (to match the 90KHz clocks such as PTS/DTS), the quotient 商 is SCR (33 bits), the remainder 余数 is SCR\_ext (9 bits)
* Program\_Mux\_Rate -- This is a 22 bit integer specifying the rate at which the program stream target decoder receives the Program Stream during the pack in which it is included. The value of program\_mux\_rate is measured in units of 50 bytes/second. The value 0 is forbidden.
* pack\_stuffing\_length -- A 3 bit integer specifying the number of stuffing bytes which follow this field.
* stuffing byte -- This is a fixed 8-bit value equal to '1111 1111' that can be inserted by the encoder, for example to meet the requirements of the channel. It is discarded by the decoder.

两个头之后便是PES包（payload）：



参考一段代码理解：http://read.pudn.com/downloads104/sourcecode/multimedia/mpeg/427188/PESdecode/pesdecode.cpp\_\_.htm

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。

<http://blog.csdn.net/alangdangjia/article/details/9706419>

# [传输流(TS)](http://blog.csdn.net/alangdangjia/article/details/9706419)

分类： [程序设计](http://blog.csdn.net/alangdangjia/article/category/1124146) [嵌入式/硬件电路](http://blog.csdn.net/alangdangjia/article/category/1237685)2013-08-01 18:43 412人阅读 [评论](http://blog.csdn.net/alangdangjia/article/details/9706419#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/alangdangjia/article/details/9706419#report)

**传输流**(TS)

将具有共同时间基准或具有独立时间基准的一个或多个PES组合而成的单一的数据流称为传输流（Transport Stream）。TS实际是面向数字化分配媒介（有线、卫星、地面网）的传输层接口。对具有共同时间基准的两个以上的PES先进行节目复用,然后再对相互可有独立时间基准的各个PS进行传输复用，即将每个PES再细分为更小的TS包

   TS包由包头、自适应区和包数据3部分组成。每个包长度为固定的188B，包头长度占4 B，自适应区和包数据长度占184B。184B为有用信息空间，用于传送已编码的视音频数据流。当节目时钟基准（PCR-Program Clock Reference）存在时，包头还包括可变长度的自适应区，包头的长度就会大于4B。考虑到与通信的关系，整个传输包固定长度应相当于4个ATM包。考虑到加密是按照8B顺序加扰的，代表有用信息的自适应区和包数据的长度应该是8B的整数倍，即自适应区和包数据为23×8B =184B。

TS包的包头由如图所示的同步字节、传输误码指示符、有效载荷单元起始指示符、传输优先、包识别（PID-Packet Identification）、传输加扰控制、自适应区控制和连续计数器8个部分组成。其中，可用同步字节位串的自动相关特性，检测数据流中的包限制，建立包同步；传输误码指示符，是指有不能消除误码时，采用误码校正解码器可表示1bit 的误码，但无法校正；有效载荷单元起始指示符，表示该数据包是否存在确定的起始信息；传输优先，是给TS包分配优先权；PID值是由用户确定的，解码器根据PID将TS上从不同ES来的TS包区别出来，以重建原来的ES；传输加扰控制，可指示数据包内容是否加扰，但包头和自适应区永远不加扰；自适应区控制，用2 bit表示有否自适应区，即（01）表示有有用信息无自适应区，（10）表示无有用信息有自适应区，（11）表示有有用信息有自适应区，（00）无定义；连续计数器可对PID包传送顺序计数，据计数器读数，接收端可判断是否有包丢失及包传送顺序错误。显然，包头对TS包具有同步、识别、检错及加密功能。

TS包自适应区由自适应区长、各种标志指示符、与插入标志有关的信息和填充数据4部分组成。其中标志部分由间断指示符、随机存取指示符、ES优化指示符、PCR标志、接点标志、传输专用数据标志、原始PCR标志、自适应区扩展标志8个部分组成。

TS包语法结构如下：

|  |  |  |
| --- | --- | --- |
| **Syntax** | **No. of bits** | **Mnemonic** |
| transport\_packet(){ |  |  |
| **sync\_byte** | **8** | **bslbf** |
| **transport\_error\_indicator** | **1** | **bslbf** |
| **payload\_unit\_start\_indicator** | **1** | **bslbf** |
| **transport\_priority** | **1** | **bslbf** |
| **PID** | **13** | **uimsbf** |
| **transport\_scrambling\_control** | **2** | **bslbf** |
| **adaptation\_field\_control** | **2** | **bslbf** |
| **continuity\_counter** | **4** | **uimsbf** |
| if(adaptation\_field\_control=='10'  || adaptation\_field\_control=='11'){ |  |  |
| adaptation\_field() |  |  |
| } |  |  |
| if(adaptation\_field\_control=='01' || adaptation\_field\_control=='11') { |  |  |
| for (i=0;i<N;i++){ |  |  |
| **data\_byte** | **8** | **bslbf** |
| } |  |  |
| } |  |  |
| } |  |  |

自适应控制区语法结构如下：

| **Syntax** | **No. of Bits** | **Mnemonic** |
| --- | --- | --- |
| adaptation\_field() { |  |  |
| **adaptation\_field\_length** | **8** | **uimsbf** |
| if(adaptation\_field\_length >0) { |  |  |
| **discontinuity\_indicator** | **1** | **bslbf** |
| **random\_access\_indicator** | **1** | **bslbf** |
| **elementary\_stream\_priority\_indicator** | **1** | **bslbf** |
| **PCR\_flag** | **1** | **bslbf** |
| **OPCR\_flag** | **1** | **bslbf** |
| **splicing\_point\_flag** | **1** | **bslbf** |
| **transport\_private\_data\_flag** | **1** | **bslbf** |
| **adaptation\_field\_extension\_flag** | **1** | **bslbf** |
| if(PCR\_flag == '1') { |  |  |
| **program\_clock\_reference\_base** | **33** | **uimsbf** |
| **reserved** | **6** | **bslbf** |
| **program\_clock\_reference\_extension** | **9** | **uimsbf** |
| } |  |  |
| if(OPCR\_flag == '1') { |  |  |
| **original\_program\_clock\_reference\_base** | **33** | **uimsbf** |
| **reserved** | **6** | **bslbf** |
| **original\_program\_clock\_reference\_extension** | **9** | **uimsbf** |
| } |  |  |
| if (splicing\_point\_flag == '1') { |  |  |
| **splice\_countdown** | **8** | **tcimsbf** |
| } |  |  |
| if(transport\_private\_data\_flag == '1') { |  |  |
| **transport\_private\_data\_length** | **8** | **uimsbf** |
| for (i=0; i<transport\_private\_data\_length;i++){ |  |  |
| **private\_data\_byte** | **8** | **bslbf** |
| } |  |  |
| } |  |  |
| if (adaptation\_field\_extension\_flag == '1' ) { |  |  |
| **adaptation\_field\_extension\_length** | **8** | **uimsbf** |
| **ltw\_flag** | **1** | **bslbf** |
| **piecewise\_rate\_flag** | **1** | **bslbf** |
| **seamless\_splice\_flag** | **1** | **bslbf** |
| **reserved** | **5** | **bslbf** |
| if (ltw\_flag == '1') { |  |  |
| **ltw\_valid\_flag** | **1** | **bslbf** |
| **ltw\_offset** | **15** | **uimsbf** |
| } |  |  |
| if (piecewise\_rate\_flag == '1') { |  |  |
| **reserved** | **2** | **bslbf** |
| **piecewise\_rate** | **22** | **uimsbf** |
| } |  |  |
| if (seamless\_splice\_flag == '1'){ |  |  |
| **splice\_type** | **4** | **bslbf** |
| **DTS\_next\_AU[32..30]** | **3** | **bslbf** |
| **marker\_bit** | **1** | **bslbf** |
| **DTS\_next\_AU[29..15]** | **15** | **bslbf** |
| **marker\_bit** | **1** | **bslbf** |
| **DTS\_next\_AU[14..0]** | **15** | **bslbf** |
| **marker\_bit** | **1** | **bslbf** |
| } |  |  |
| for ( i=0;i<N;i++) { |  |  |
| **reserved** | **8** | **bslbf** |
| } |  |  |
| } |  |  |
| for (i=0;i<N;i++){ |  |  |
| **stuffing\_byte** | **8** | **bslbf** |
| } |  |  |
| } |  |  |
| } |  |  |