H.264 的句法和语义

H.264手册：

http://www.itu.int/rec/T-REC-H.264-200503-S/en

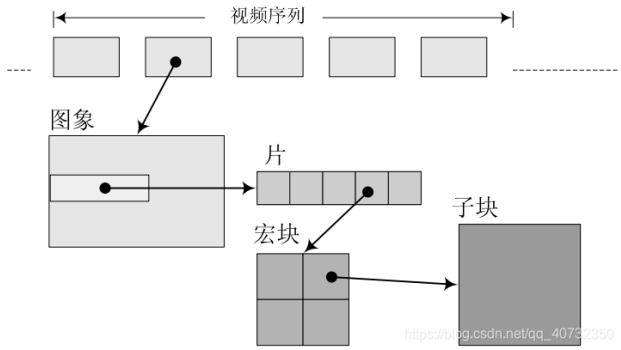
1 句法

在编码器输出的码流中，数据的基本单位是句法元素，每个句法元素由若干比特组成，它表示某个特定的物理意义，例如：宏块类型、量化参数等。 句法表征句法元素的组织结构，语义阐述句法元素的具体含义。 所有的视频编码标准都是通过定义句法和语义来规范编解码器的工作流程。

1.1 句法元素的分层结构

编码器输出的比特码流中，每个比特都隶属某个句法元素，也就是说，码流是由一个个句法元素依次衔接组成的，码流中除了句法元素并不存在专门用于控制或同步的内容。在 H.264 定义的码流中，句法元素被组织成有层次的结构，分别描述各个层次的信息。图1表现了这种结构。

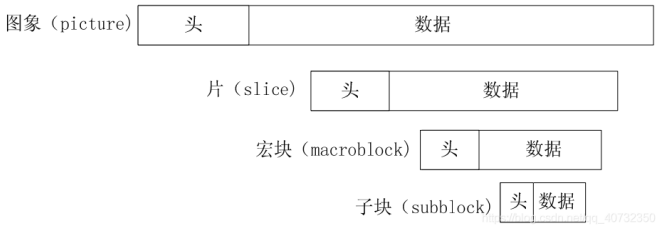
句法元素的分层结构



句法元素的分层结构有助于更有效地节省码流。例如，在一个图像中，经常会在各个片之间有相同的数据，如果每个片都同时携带这些数据，势必会造成码流的浪费。更为有效的做法是将该图像的公共信息抽取出来，形成图像一级的句法元素，而在片级只携带该片自身独有的句法元素。 在H.264 中，句法元素共被组织成 序列、图像、片、宏块、子宏块五个层次。

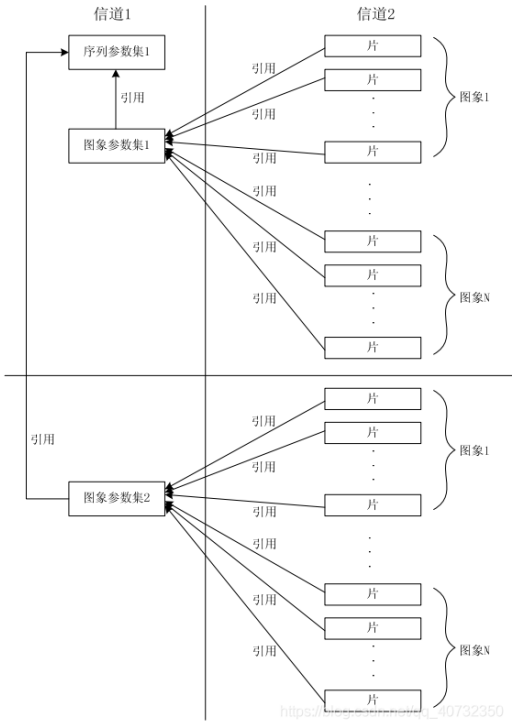
H.264 的分层结构是经过精心设计的，与以往的视频编码标准相比有很大的改进，这些改进主要针对传输中的错误掩藏，在有误码发生时可以提高图像重建的性能。在以往的标准中，分层的组织结构如图2，它们如同 TCP/IP 协议的结构，每一层都有头部，然后在每层的数据部分包含该层的数据。

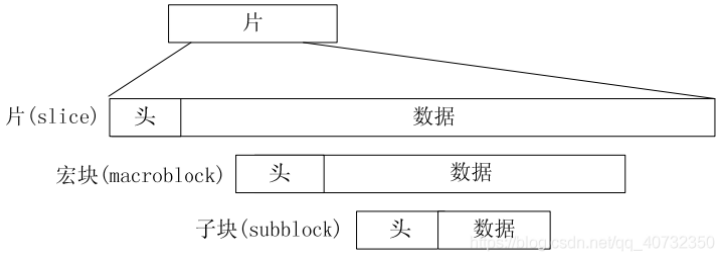
以往标准中句法元素的分层结构



上面的结构中，每一层的头部和它的数据部分形成管理与被管理的强依赖关系，头部的句法元素是该层数据的核心，而一旦头部丢失，数据部分的信息几乎不可能再被正确解码出来。尤其在序列层及图像层，由于网络中 MTU（最大传输单元）大小的限制，不可能将整个层的句法元素全部放入同一个分组中，这个时候如果头部所在的分组丢失，该层其他分组即使能被正确接收也无法解码，造成资源浪费。

于是有了：H.264 中句法元素的分层结构

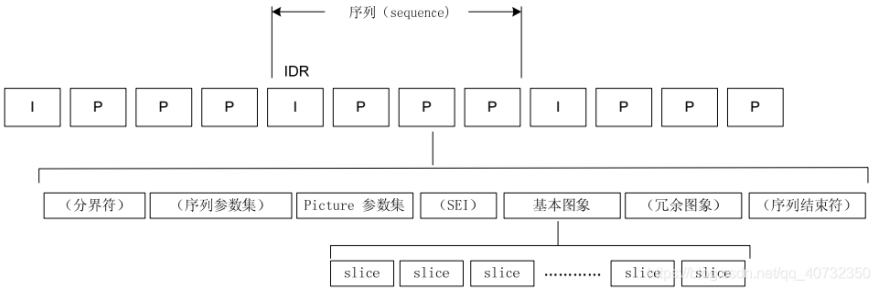




在 H.264 中，分层结构最大的不同是取消了序列层和图像层，并将原本属于序列和图像头部的大部分句法元素游离出来形成序列和图像两级参数集，其余的部分则放入片层。

上图所示的码流的结构是一种简化的模型，这个模型已经能够正确工作，但还不够完善，不适合复杂的场合。在复杂的通信环境中，除了片和参数集外还需要其他的数据单位来提供额外的信息。上图描述了在复杂通信中的码流中可能出现的数据单位。如前文所述，参数集可以被抽取出来使用其它信道。

H.264 码流中的数据单位



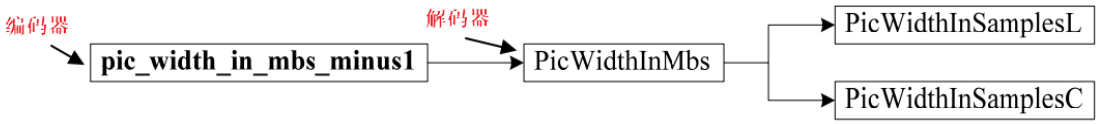
一个序列的第一个图像叫做 IDR 图像（立即刷新图像），IDR 图像都是 I图像。H.264 引入 IDR 图像是为了解码的重同步，当解码器解码到 IDR 图像时，立即将参考帧队列清空，将已解码的数据全部输出或抛弃，重新查找参数集，开始一个新的序列。这样，如果在前一个序列的传输中发生重大错误，如严重的丢包，或其他原因引起数据错位，在这里可以获得重新同步。 IDR 图像之后的图像永远不会引用 IDR 图像之前的图像的数据来解码。

要注意 IDR 图像和 I 图像的区别， IDR 图像一定是 I 图像， 但 I 图像不一定是 IDR 图像。一个序列中可以有很多的 I 图像， I 图像之后的图像可以引用 I 图像之间的图像做运动参考。

在上图中，除了参数集与片外还有其它的数据单位，这些数据单位可以提供额外的数据或同步信息，这些数据单位也是一系列句法元素的集合。它们在解码过程中不是必需的，但却可以适当提高同步性能或定义图像的复杂特征。

1.2 句法元素与变量——————————————————————————————————

编码器：将数据编码为句法元素然后依次发送。
解码器：将句法元素作求值计算，得出一些中间数据，这些中间数据就是 H.264 定义的变量。



图中， pic_width_in_mbs_minus1 是解码器直接从码流中提取的句法元素，这个句法元素表征图像的宽度，以宏块为单位。我们看到，为了提高编码效率， H.264 将图像实际的宽度减去 1 后再传送。

- PicWidthInMbs = pic_width_in_mbs_minus1 + 1
- PicWidthInSamplesL = PicWidthInMbs * 16
- PicWidthInSamplesC = PicWidthInMbs * 8

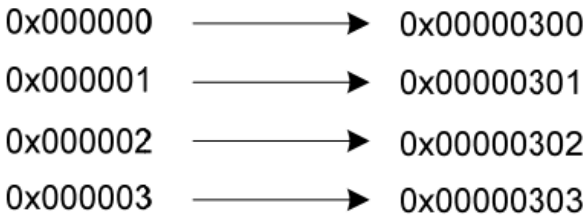
以上变量 PicWidthInMbs 表示图像以宏块为单位的宽，变量 PicWidthInSamplesL 、PicWidthInSamplesC 分别表示图像的亮度、色度分量以像素为单位的宽。 H.264 定义这些变量是因为在后续句法元素的提取算法或图像的重建中需要用到它们的值。

在 H.264 中，句法元素的名称是由小写字母和一系列的下划线组成，而变量名称是大小写字母组成，中间没有下划线。

2 NAL层语义

在网络传输的环境下，编码器将每个 NAL 各自独立、完整地放入一个分组，由于分组都有头部，解码器可以很方便地检测出 NAL 的分界，依次取出 NAL 进行解码。解码器为了在数据流中分辨每个 NAL的起始和终止，当数据流是存储在介质上时，在每个 NAL 前添加起始码： **0x 00 00 01**

为了“防止竞争”，在编码器编码完一个 NAL 时，应该检测是否出现图 7.6 左侧 中的四个字节序列，以防止它们和起始码竞争。如果检测到这些序列存在，编码器将在最后一个字节前插入一个新的字节： 0x03，从而使它们变成图 7.6 右测的样子。当解码器在 NAL 内部检测到有 0x000003 的序列时，将把 0x03 抛弃，恢复原始数据。



解码器在 NAL 层的处理步骤，其中变量 NumBytesInNALunit 是解码器计算出来的，解码器在逐个字节地读一个 NAL 时并不同时对它解码，而是要通过起始码机制将整个 NAL 读进、计算出长度后再开始解码。

3 句法表

解码器在 NAL 层的处理步骤，其中变量 **NumBytesInNALunit** 是解码器计算出来的，解码器在逐个字节地读一个 NAL 时并不同时对它解码，而是要通过起始码机制将整个 NAL 读进、计算出长度后再开始解码。

所有句法表——————————

```
1. nal_unit( NumBytesInNALunit );                    //表 7.1 NAL 层句法
2. seq_parameter_set_rbsp( );                        //表 7.2 序列参数集层句法
3. pic_parameter_set_rbsp( );                        //表 7.3 图像参数集层句法
4. slice_layer_without_partitioning_rbsp( );         //表 7.4 片层句法(不分区)
5. slice_data_partition_a_layer_rbsp( ) ;            //表 7.5 片层 A 分区句法
6. slice_data_partition_b_layer_rbsp( );             //表 7.6 片层 B 分区句法
7. slice_data_partition_c_layer_rbsp( ) ;            //表 7.7 片层 C 分区句法
8. rbsp_trailing_bits( ) ;                           //表 7.8 拖尾（trailing bits）句法
9. slice_header( );                                  //表 7.9 片头句法
10. ref_pic_list_reordering( );                      //表 7.10 参考帧队列重排序（reordering）句法
11. pred_weight_table( ) ;                           //表 7.11 加权预测句法
12. dec_ref_pic_marking( ) ;                         //表 7.12 参考帧队列标记(marking)句法
13. slice_data( ) ;                                  //表 7.13 片层数据句法
14. macroblock_layer( ) ;                            //表 7.14 宏块层句法
15. mb_pred( mb_type ) ;                             //表 7.15 宏块层预测句法
16. sub_mb_pred( mb_type );                          //表 7.16 子宏块预测句法
17. residual( ) ;                                    //表 7.17 残差句法
18. residual_block_cavlc( coeffLevel, maxNumCoeff ) ;//表 7.18 CAVLC 残差句法
19. residual_block_cabac( coeffLevel, maxNumCoeff ); //表 7.19 CABAC 残差句法
```

NAL 层句法（NAL）

nal_unit(NumBytesInNALunit) {	C	Descriptor
forbidden_zero_bit	All	f(1)
nal_ref_idc	All	u(2)
nal_unit_type	All	u(5)
NumBytesInRBSP = 0		
for(i = 1; i < NumBytesInNALunit; i++) {		
if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {		
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
i += 2		
emulation_prevention_three_byte /* equal to 0x03 */	All	f(8)
} else		
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
}		
}		

---

forbidden_zero_bit 等于 0

nal_ref_idc 指示当前 NAL 的优先级。取值范围为 0-3, ,值越高,表示当前 NAL 越重要,需要优先受到保护。 H.264 规定如果当前 NAL 是属于参考帧的片，或是序列参数集，或是图像参数集这些重要的数据单位时，本句法元素必须大于 0。但在大于 0 时具体该取何值，却没有进一步规定,通信双方可以灵活地制定策略。

nal_unit_type 指明当前 NAL unit 的类型，具体类型的定义如表：

nal_unit_type	NAL类型	C
0	未使用	
1	不分区、非 IDR 图像的片	2, 3, 4
2	片分区 A	2
3	片分区 B	3
4	片分区 C	4
5	IDR 图像中的片	2, 3
6	补充增强信息单元（SEI）	5
7	序列参数集	0
8	图像参数集	1
9	分界符	6
10	序列结束	7
11	码流结束	8
12	填充	9
13..23	保留	
24..31	未使用	

nal_unit_type=5 时，表示当前 NAL 是 IDR 图像的一个片，在这种情况下， IDR 图像中的每个片的 nal_unit_type 都应该等于 5。注意 IDR 图像不能使用片分区。

rbsp_byte[i] RBSP 的第 i 个字节。 RBSP 指原始字节载荷，它是 NAL 单元的数据部分的封装格式，封装的数据来自 SODB（原始数据比特流）。 SODB 是编码后的原始数据， SODB 经封装为 RBSP 后放入 NAL 的数据部分。下面介绍一个 RBSP 的生成顺序。
从 SODB 到 RBSP 的生成过程：

- 如果 SODB 内容是空的，生成的 RBSP 也是空的
- 否则， RBSP 由如下的方式生成：

1. RBSP 的第一个字节直接取自 SODB 的第 1 到 8 个比特， （RBSP 字节内的比特按照从左到右对应为从高到低的顺序排列， most significant） ,以此类推， RBSP 其余的每个字节都直接取自 SODB 的相应比特。 RBSP 的最后一个字节包含 SODB 的最后几个比特，及如下的rbsp_trailing_bits()
2. rbsp_trailing_bits()的第一个比特是 1,接下来填充 0，直到字节对齐。 （填充 0 的目的也是为了字节对齐）
3. 最后添加若干个 cabac_zero_word(其值等于 0x0000)

emulation_prevention_three_byte NAL 内部为防止与起始码竞争而引入的填充字节 ,值为 0x03。

序列参数集层句法（SPS）

seq_parameter_set_rbsp( ) {	C	Descriptor
profile_idc	0	u(8)
constraint_set0_flag	0	u(1)
constraint_set1_flag	0	u(1)
constraint_set2_flag	0	u(1)
reserved_zero_5bits /* equal to 0 */	0	u(5)
level_idc	0	u(8)
seq_parameter_set_id	0	ue(v)
log2_max_frame_num_minus4	0	ue(v)
pic_order_cnt_type	0	ue(v)
if(pic_order_cnt_type == 0)		
log2_max_pic_order_cnt_lsb_minus4	0	ue(v)
else if(pic_order_cnt_type == 1) {		
delta_pic_order_always_zero_flag	0	u(1)
offset_for_non_ref_pic	0	se(v)
offset_for_top_to_bottom_field	0	se(v)
num_ref_frames_in_pic_order_cnt_cycle	0	ue(v)
for(i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)		
offset_for_ref_frame[i]	0	se(v)
}		

num_ref_frames	0	ue(v)
gaps_in_frame_num_value_allowed_flag	0	u(1)
pic_width_in_mbs_minus1	0	ue(v)
pic_height_in_map_units_minus1	0	ue(v)
frame_mbs_only_flag	0	u(1)
if(!frame_mbs_only_flag)		
mb_adaptive_frame_field_flag	0	u(1)
direct_8x8_inference_flag	0	u(1)
frame_cropping_flag	0	u(1)
if(frame_cropping_flag) {		
frame_crop_left_offset	0	ue(v)
frame_crop_right_offset	0	ue(v)
frame_crop_top_offset	0	ue(v)
frame_crop_bottom_offset	0	ue(v)
}		
vui_parameters_present_flag	0	u(1)

if(vui_parameters_present_flag)		
vui_parameters( )	0	
rbsp_trailing_bits( )	0	
}		

profile_idc、 **level_idc** 指明所用 profile、 level。

参考：https://blog.csdn.net/xiaojun111111/article/details/52090185

constraint_set0_flag 等于 1 时表示必须遵从附录 A.2.1 所指明的所有制约条件。等于 0 时表示不必遵从所有条件。
constraint_set1_flag 等于 1 时表示必须遵从附录 A.2.2 所指明的所有制约条件。等于 0 时表示不必遵从所有条件。
constraint_set2_flag 等于 1 时表示必须遵从附录 A.2.3 所指明的所有制约条件。等于 0 时表示不必遵从所有条件。

reserved_zero_5bits 在目前的标准中本句法元素必须等于 0，其他的值保留做将来用，解码器应该忽略本句法元素的值。

seq_parameter_set_id 指明本序列参数集的 id 号，这个 id 号将被 picture 参数集引用，本句法元素的值应该在[0，31]。

log2_max_frame_num_minus4 这个句法元素主要是为读取另一个句法元素 frame_num 服务的，frame_num 是最重要的句法元素之一，它标识所属图像的解码顺序。可以在句法表看到， fram-num的解码函数是 ue（v），函数中的 v 在这里指定：

v = log2_max_frame_num_minus4 + 4
从另一个角度看，这个句法元素同时也指明了 frame_num 的所能达到的最大值：
MaxFrameNum = 2(log2_max_frame_num_minus4 + 4)

变量 MaxFrameNum 表示 frame_num 的最大值，在后文中可以看到，在解码过程中它也是一个非常重要的变量。

值得注意的是 frame_num 是循环计数的，即当它到达 MaxFrameNum 后又从 0 重新开始新一轮的计数。 解码器必须要有机制检测这种循环， 不然会引起类似千年虫的问题，在图像的顺序上造成混乱。

pic_order_cnt_type 指明了 poc (picture order count) 的编码方法， poc 标识图像的播放顺序。由于H.264 使用了 B 帧预测，使得图像的解码顺序并不一定等于播放顺序，但它们之间存在一定的映射关系。 poc 可以由 frame-num 通过映射关系计算得来，也可以索性由编码器显式地传送。 H.264 中一共定义了三种 poc 的编码方法，这个句法元素就是用来通知解码器该用哪种方法来计算 poc。 而以下的几个句法元素是分别在各种方法中用到的数据。

在如下的视频序列中本句法元素不应该等于 2:

- 一个非参考帧的接入单元后面紧跟着一个非参考图像(指参考帧或参考场)的接入单元
- 两个分别包含互补非参考场对的接入单元后面紧跟着一个非参考图像的接入单元.
- 一个非参考场的接入单元后面紧跟着另外一个非参考场,并且这两个场不能构成一个互补场对

log2_max_pic_order_cnt_lsb_minus4 指明了变量 MaxPicOrderCntLsb 的值:
MaxPicOrderCntLsb = 2(log2_max_pic_order_cnt_lsb_minus4 + 4)
该变量在 pic_order_cnt_type = 0 时使用。

delta_pic_order_always_zero_flag 等于 1 时,句法元素 delta_pic_order_cnt[0]和 delta_pic_order_cnt[1]不在片头出现,并且它们的值默认为 0; 本句法元素等于 0 时,上述的两个句法元素将在片头出现。

offset_for_non_ref_pic 被用来计算非参考帧或场的 picture order count (在 8.2.1),本句法元素的值应该在[-231 , 231 – 1]。

offset_for_top_to_bottom_field 被用来计算帧的底场的 picture order count (在 8.2.1), 本句法元素的值应该在[-231 , 231 – 1]。

num_ref_frames_in_pic_order_cnt_cycle 被用来解码 picture order count (在 8.2.1),本句法元素的值应该在[0,255]。

offset_for_ref__frame[i] 在 picture order count type=1 时用，用于解码 POC，本句法元素对循环 num_ref_frames_in_pic_order_cycle 中的每一个元素指定一个偏移。

max_num_ref_frames 指定参考帧队列可能达到的最大长度，解码器依照这个句法元素的值开辟存储区，这个存储区用于存放已解码的参考帧， H.264 规定最多可用 16 个参考帧，本句法元素的值最大为 16。值得注意的是这个长度以帧为单位，如果在场模式下，应该相应地扩展一倍。

gaps_in_frame_num_value_allowed_flag 这个句法元素等于 1 时，表示允许句法元素 frame_num 可以不连续。当传输信道堵塞严重时，编码器来不及将编码后的图像全部发出，这时允许丢弃若干帧图像。 在正常情况下每一帧图像都有依次连续的 **frame_num 值**，解码器检查到如果 frame_num 不连续，便能确定有图像被编码器丢弃。这时，解码器必须启动错误掩藏的机制来近似地恢复这些图像，因为这些图像有可能被后续图像用作参考帧。
当这个句法元素等于 0 时，表不允许 frame_num 不连续，即编码器在任何情况下都不能丢弃图像。这时， H.264 允许解码器可以不去检查 frame_num 的连续性以减少计算量。这种情况下如果依然发生 frame_num 不连续，表示在传输中发生丢包，解码器会通过其他机制检测到丢包的发生，然后启动错误掩藏的恢复图像。

pic_width_in_mbs_minus1本句法元素加 1 后指明图像宽度，以宏块为单位：

PicWidthInMbs = pic_width_in_mbs_minus1 + 1

通过这个句法元素解码器可以计算得到亮度分量以像素为单位的图像宽度：

PicWidthInSamplesL = PicWidthInMbs * 16

从而也可以得到色度分量以像素为单位的图像宽度：

PicWidthInSamplesC = PicWidthInMbs * 8

以上变量 PicWidthInSamplesL、 PicWidthInSamplesC 分别表示图像的亮度、色度分量以像素为单位的宽。

H.264 将图像的大小在序列参数集中定义，意味着可以在通信过程中随着序列参数集动态地改变图像的大小，在后文中可以看到，甚至可以将传送的图像剪裁后输出。

pic_height_in_map_units_minus1 本句法元素加 1 后指明图像高度：

PicHeightInMapUnits = pic_height_in_map_units_minus1 + 1

PicSizeInMapUnits = PicWidthInMbs * PicHeightInMapUnits

图像的高度的计算要比宽度的计算复杂，因为一个图像可以是帧也可以是场，从这个句法元素可以在帧模式和场模式下分别计算出出亮度、色度的高。值得注意的是，这里以 map_unit 为单位， map_unit的含义由后文叙述。

frame_mbs_only_flag 本句法元素等于 0 时表示本序列中所有图像的编码模式都是帧，没有其他编码模式存在；本句法元素等于 1 时 ，表示本序列中图像的编码模式可能是帧，也可能是场或帧场自适应，某个图像具体是哪一种要由其他句法元素决定。
结合 map_unit 的含义，这里给出上一个句法元素 pic_height_in_map_units_minus1 的进一步解析步骤：
当 frame_mbs_only_flag 等于１ ， pic_height_in_map_units_minus1 指的是一个 picture 中帧的高度；

当frame_mbs_only_flag 等于０ ， pic_heght_in_map_units_minus1 指的是一个 picture 中场的高度，所以可以得到如下以宏块为单位的图像高度：

FrameHeightInMbs = (2 – frame_mbs_only_flag) * PicHeightInMapUnits

PictureHeightInMbs= (2 – frame_mbs_only_flag) * PicHeightInMapUnits

mb_adaptive_frame_field_flag 指明本序列是否属于帧场自适应模式。 mb_adaptive_frame_field_flag等于１时表明在本序列中的图像如果不是场模式就是帧场自适应模式，等于０时表示本序列中的图如果不是场模式就是帧模式。。表列举了一个序列中可能出现的编码模式：
a. 全部是帧，对应于 frame_mbs_only_flag =1 的情况。
b. 帧和场共存。 frame_mbs_only_flag =0, mb_adaptive_frame_field_flag =0
c. 帧场自适应和场共存。 frame_mbs_only_flag =0, mb_adaptive_frame_field_flag =1
值得注意的是，帧和帧场自适应不能共存在一个序列中。

direct_8x8_inference_flag 用于指明 B 片的直接和 skip 模式下运动矢量的预测方法。

frame_cropping_flag 用于指明解码器是否要将图像裁剪后输出，如果是的话，后面紧跟着的四个句法元素分别指出左右、上下裁剪的宽度。

frame_crop_left_offset,frame_crop_right_offset,frame_crop_bottom_offset,frame_crop_bottom_offset 如上一句法元素所述。

vui_parameters_present_flag 指明 vui 子结构是否出现在码流中， vui 的码流结构在附录中指明，用以表征视频格式等额外信息。

图像参数集语义（PPS）

pic_parameter_set_id 用以指定本参数集的序号，该序号在各片的片头被引用。

seq_parameter_set_id 指明本图像参数集所引用的序列参数集的序号。

entropy_coding_mode_flag 指明熵编码的选择，本句法元素为０时，表示熵编码使用 CAVLC，本句法元素为１时表示熵编码使用 CABAC

pic_order_present_flag POC 的三种计算方法在片层还各需要用一些句法元素作为参数，本句法元素等于１时表示在片头会有句法元素指明这些参数；本句法元素等于０时，表示片头不会给出这些参数，这些参数使用默认值

num_slice_groups_minus1 本句法元素加１后指明图像中片组的个数。 H.264 中没有专门的句法元素用于指明是否使用片组模式，当本句法元素等于０（即只有一个片组），表示不使用片组模式，后面也不会跟有用于计算片组映射的句法元素。

slice_group_map_type 当 num_slice_group_minus1 大于０，既使用片组模式时，本句法元素出现在码流中，用以指明片组分割类型。
map_units 的定义：

- 当 frame_mbs_only_flag 等于１时， map_units 指的就是宏块
- 当 frame_mbs_only_falg 等于０时
- 帧场自适应模式时， map_units 指的是宏块对
- 场模式时， map_units 指的是宏块
- 帧模式时， map_units 指的是与宏块对相类似的，上下两个连续宏块的组合体。

run_length_minus1[i] 用以指明当片组类型等于０时，每个片组连续的 map_units 个数。

top_left[i],bottom_right[i] 用以指明当片组类型等于２时，矩形区域的左上及右下位置。

slice_group_change_direction_flag 当片组类型等于３、４、５时，本句法元素与下一个句法元素一起指明确切的片组分割方法。

slice_group_change_rate_minus1 用以指明变量 SliceGroupChangeRAte

pic_size_in_map_units_minus1 在片组类型等于６时，用以指明图像以 map_units 为单位的大小。

slice_group_id[i] 在片组类型等于６时，用以指明某个 map_units 属于哪个片组。

num_ref_idx_lo_active_minus1 加１后指明目前参考帧队列的长度，即有多少个参考帧（包括短期和长期）。值得注意的是，当目前解码图像是场模式下，参考帧队列的长度应该是本句法元素再乘以２，因为场模式下各帧必须被分解以场对形式存在。（这里所说的场模式包括图像的场及帧场自适应下的处于场模式的宏块对） 本句法元素的值有可能在片头被重载。
读者可能还记得在序列参数集中有句法元素 num_ref_frames 也是跟参考帧队列有关，它们的区别 是 num_ref_frames

指明参考帧队列的最大值，解码器用它的值来分配内存空间；
num_ref_idx_lo_active_minus1 指明在这个队列中当前实际的、已存在的参考帧数目，这从它的名字“active”中也可以看出来。

这个句法元素是 H.264 中最重要的句法元素之一，在第章我们可以看到，编码器要通知解码器某个运动矢量所指向的是哪个参考图像时，并不是直接传送该图像的编号，而是传送该图像在参考帧队列中的序号。这个序号并不是在码流中传送的，而是编码器和解码器同步地、用相同的方法将参考图像放入队列，从而获得一个序号。这个队列在每解一个图像，甚至是每个片后都会动态地更新。维护参考帧队列是编解码器十分重要的工作，而本句法元素是维护参考帧队列的重要依据。参考帧队列的复杂的维护机制是 H.264 重要也是很有特色的组成部分

num_ref_idx_l1_active_minus1 与上一个句法元素的语义一致，只是本句法元素用于 list１，而上一句法元素用于 list0

weighted_pred_flag 用以指明是否允许 P 和 S P 片的加权预测，如果允许，在片头会出现用以计算加权预测的句法元素。

weighted_bipred_flag 用以指明是否允许 B 片的加权预测，本句法元素等于 0 时表示使用默认加权预测模式，等于 1 时表示使用显式加权预测模式，等于 2 时表示使用隐式加权预测模式。

pic_init_qp_minus26 加 26 后用以指明亮度分量的量化参数的初始值。在 H.264 中，量化参数分三个级别给出：图像参数集、片头、宏块。在图像参数集给出的是一个初始值。

pic_init_qs_minus26 与上一个句法元素语义一致，只是用于 SP 和 SI。

chroma_qp_index_offset 色度分量的量化参数是根据亮度分量的量化参数计算出来的，本句法元素用以指明计算时用到的参数。

deblocking_filter_control_present_flag 编码器可以通过句法元素显式地控制去块滤波的强度，本句法元素指明是在片头是否会有句法元素传递这个控制信息。如果本句法元素等于 0，那些用于传递滤波强度的句法元素不会出现，解码器将独立地计算出滤波强度。

constrained_intra_pred_flag 在 P 和 B 片中，帧内编码的宏块的邻近宏块可能是采用的帧间编码。当本句法元素等于 1 时，表示帧内编码的宏块不能用帧间编码的宏块的像素作为自己的预测，即帧内编码的宏块只能用邻近帧内编码的宏块的像素作为自己的预测；而本句法元素等于 0 时，表示不存在这种限制。

redundant_pic_cnt_present_flag指明是否会出现 redundant_pic_cnt 句法元素。

片头语义————————————————————

first_mb_in_slice 片中的第一个宏块的地址, 片通过这个句法元素来标定它自己的地址。要注意的是在帧场自适应模式下，宏块都是成对出现，这时本句法元素表示的是第几个宏块对，对应的第一个宏块的真实地址应该是

2 * first_mb_in_slice

slice_type 指明片的类型，IDR 图像时, slice_type 等于 2, 4, 7, 9。

pic_parameter_set_id 图像参数集的索引号. 范围 0 到 255。

frame_num 每个参考帧都有一个依次连续的 frame_num 作为它们的标识,这指明了各图像的解码顺序。但事实上我们在表 中可以看到，frame_num 的出现没有 if 语句限定条件，这表明非参考帧的片头也会出现 frame_num。只是当该个图像是参考帧时，它所携带的这个句法元素在解码时才有意义。如表：

slice_type	Name of slice_type
0	P (P slice)
1	B (B slice)
2	I (I slice)
3	SP (SP slice)
4	SI (SI slice)
5	P (P slice)
6	B (B slice)
7	I (I slice)
8	SP (SP slice)
9	SI (SI slice)

图像序号	图像类型	是否用作参考	frame_num
1	I	是	0
2	P	是	1
3	B	否	2
4	P	是	2
5	B	否	3
6	P	是	3
7	B	否	4
8	P	是	4
…	…	…	…

field_pic_flag 这是在片层标识图像编码模式的唯一一个句法元素。所谓的编码模式是指的帧编码、场编码、帧场自适应编码。当这个句法元素取值为 1 时 属于场编码； 0 时为非场编码。

bottom_field_flag 等于 1 时表示当前图像是属于底场；等于 0 时表示当前图像是属于顶场。

idr_pic_id IDR 图像的标识。不同的 IDR 图像有不同的 idr_pic_id 值。值得注意的是， IDR 图像有不等价于 I 图像，只有在作为 IDR 图像的 I 帧才有这个句法元素，在场模式下， IDR 帧的两个场有相同的 idr_pic_id 值。 idr_pic_id 的取值范围是 [0， 65535]，和 frame_num 类似，当它的值超出这个范围时，它会以循环的方式重新开始计数。

slice_qp_delta 指出在用于当前片的所有宏块的量化参数的初始值。
SliceQPY = 26 + pic_init_qp_minus26 + slice_qp_delta
QPY 的范围是 0 to 51。
我们前文已经提到， H.264 中量化参数是分图像参数集、片头、宏块头三层给出的，前两层各自给出一个偏移值，这个句法元素就是片层的偏移。