国 际 电 信 联 盟

ITU-T

国际电信联盟
电信标准化部门

H.264

(03/2005)

H系列：视听及多媒体系统

视听业务的基础设施 — 活动图像编码

通用视听业务的先进视频编码

ITU-T H.264建议书

国际电信联盟

ITU-T H系列建议书

视听及多媒体系统

可视电话系统的特性	H.100-H.199
视听业务的基础设施	
概述	H.200-H.219
传输多路复用和同步	H.220-H.229
系统概况	H.230-H.239
通信规程	H.240-H.259
活动图像编码	**H.260-H.279**
相关系统概况	H.280-H.299
视听业务的系统和终端设备	H.300-H.349
视听和多媒体业务的号码簿业务体系结构	H.350-H.359
视听和多媒体业务的服务质量体系结构	H.360-H.369
多媒体的补充业务	H.450-H.499
移动性和协作程序	
移动性和协作、定义、协议和程序概述	H.500-H.509
H系列多媒体系统和业务的移动性	H.510-H.519
移动多媒体协作应用和业务	H.520-H.529
移动多媒体应用和业务的安全性	H.530-H.539
移动多媒体协作应用和业务的安全性	H.540-H.549
移动性互通程序	H.550-H.559
移动多媒体协作互通程序	H.560-H.569
宽带和三网合一多媒体业务	
在VDSL上传送宽带多媒体业务	H.610-H.619

欲了解更详细信息，请查阅*ITU-T*建议书目录。

ITU-T H.264建议书

通用视听业务的先进的视频编码

摘 要

本建议书|国际标准是由已有视频编码标准发展而来(H.261，H.262和H.263)，以适应各种应用，如视频会议，数字存储媒体，电视广播，互联网流式传输和通信等，对运动图像压缩比提出的更高要求，本编码标准正是为了满足这种日益增长的需求而开发的。同时，本标准的设计也能够使编码图像能够以灵活的方式在不同的网络环境中使用。使用本建议书|国际标准可使得运动图像能以计算机数据的形式被使用，并能够存储在各种存储媒体上，在现有或未来网络中传送和接收，在现有或未来广播信道中分发。

本版本于2005年3月通过，包含的修改是对视频编码标准增加了四个简表，分别是高级，高级 10，高级 4:2:2和高级 4:4:4，提高了视频质量能力，从而扩展了标准的应用的范围（如，通过包括支持更大范围的图像样点精度和更高解析度的色度格式来实现）。另外，定义了补充数据的新类型，更进一步扩展视频编码标准的适用性。最后，对已印刷版本中的若干错误进行了修正。本版本除了增强了视频编码能力以外，还用来和联合研发的ISO/IEC 14496-10标准在技术上保持一致。

ITU-T H.264 建议书的勘误表1修改和更新了一些次要的方面，使之与于2005年4月通过的联合开发并且技术一致的ISO/IEC 14496-10的最新版本的文本相一致。它还改正了一些次要错误，进行了一些必要的澄清，并且定义了三个前面预留的幅型比标识符。

本版本包括 2005 年 3 月通过的文本以及在 2005 年 9 月通过的勘误表 1。

来 源

ITU-T 第 16 研究组（2005-2008 年）按照 ITU-T A.8 建议书规定的程序，于 2005 年 3 月 1 日批准了 ITU-T H.264（2005 年）建议书。它包括由 H.264 (2005) 勘误 1 引出的修订，该勘误按照 ITU-T A.8 建议书规定的程序，于 2005 年 9 月 13 日批准。

前　言

国际电信联盟（ITU）是从事电信领域工作的联合国专门机构。ITU-T（国际电信联盟电信标准化部门）是国际电信联盟的常设机构,负责研究技术、操作和资费问题，并且为在世界范围内实现电信标准化，发表有关上述研究项目的建议书。

每四年一届的世界电信标准化全会（WTSA）确定 ITU-T 各研究组的研究课题，再由各研究组制定有关这些课题的建议书。

WTSA 第 1 号决议规定了批准建议书须遵循的程序。

属 ITU-T 研究范围的某些信息技术领域的必要标准，是与国际标准化组织（ISO）和国际电工技术委员会（IEC）合作制定的。

注

本建议书为简要而使用的“主管部门”一词，既指电信主管部门，又指经认可的运营机构。

遵守本建议书的规定是以自愿为基础的，但建议书可能包含某些强制性条款（以确保例如互操作性或适用性等），只有满足所有强制性条款的规定，才能达到遵守建议书的目的。“应该”或“必须”等其它一些强制性用语及其否定形式被用于表达特定要求。使用此类用语不表示要求任何一方遵守本建议书。

知识产权

国际电联提请注意：本建议书的应用或实施可能涉及使用已申报的知识产权。国际电联对无论是其成员还是建议书制定程序之外的其它机构提出的有关已申报的知识产权的证据、有效性或适用性不表示意见。

至本建议书批准之日止，国际电联尚未收到实施本建议书可能需要的受专利保护的知识产权的通知。但需要提醒实施者注意的是，这可能不是最新信息，因此大力提倡他们查询电信标准化局（TSB）的专利数据库。

目 录

图的清单

表的清单

前言

国际电信联盟（ITU）是从事电信领域工作的联合国专门机构。ITU-T（国际电信联盟电信标准化部门）是国际电信联盟的常设机构，负责研究技术、操作和资费问题，并且为在世界范围内实现电信标准化，发表有关上述研究项目的建议书。每四年一届的世界电信标准化全会（WTSA）确定 ITU-T 各研究组的研究课题，再由各研究组制定有关这些课题的建议书。WTSA 第 1 号决议规定了批准建议书须遵循的程序。属 ITU-T 研究范围的某些信息技术领域的必要标准，是与国际标准化组织（ISO）和国际电工技术委员会（IEC）合作制定的。

ISO(国际标准化组织)和 IEC(国际电子技术委员会)共同构成了世界范围的标准专用系统。ISO 和 IEC 的成员实体参与国际标准的研究，是通过由各自组织机构设立的技术委员会来出来特定技术领域的活动。ISO 和 IEC 技术委员会在相互感兴趣的领域合作。其他国际组织，政府或者非政府性质的，可以通过与 ISO 和 IEC 发布联合声明的方式合作。在信息通信领域，ISO 和 IEC 成立了一个联合工作委员会 ISO/IEC JTC 1。被联合技术委员会接受的国际标准草案会在有投票权的国际实体中使用。作为国际标准的印刷本则需要至少 75%的国际实体投票通过。

本建议书|国际标准由 ITU-T SG 16 Q.6（也称为 VCEG，图像编码专家工作组）和 ISO/IEC JTC 1/SC 29/WG 11（也称为 MPEG，运动图像专家工作组）联合开发。VCEG 于 1997 年成立，主要是保持 ITU-T 图像编解码标准的先进性和发展新的图像编解码标准，可以适用于会话和非会话业务。MPEG 于 1988 年成立，主要是为了研究运动图像和相关音频的编码标准，以适应不同应用，如数字存储媒体、分发和通信。

本建议书|国际标准的附件 A 到 E 都包含了规范性要求是本建议书|国际标准的组成部分。

ITU-T H.264 建议书

通用视听业务的先进视频编码

0 引言

本节不是本建议书 | 国际标准的组成部分。

0.1 序言

本节不是本建议书 | 国际标准的组成部分。

随着处理能力和存储容量价格的下降，网络所能支持的编码视频数据的多样化以及视频编码技术的进步，对具有较高压缩效率，并且有更好的网络健壮性的视频压缩和表示的工业标准的需求非常迫切。为此，ITU-T 视频编码专家组（VCEG）和 ISO/IEC 运动图像专家组（MPEG）于 2001 年成立了联合视频小组（JVT），致力于开发一个新的建议书 | 国际标准。

0.2 目的

本节不是本建议书 | 国际标准的组成部分。

本建议书 | 国际标准是为了满足视频会议、数字存储媒体、电视广播、网络流媒体和通信等各种应用对高压缩比运动图像压缩日益迫切的需求而制定的。同时也是为不同的网络环境中的应用设计一种灵活的编码数据表示方式。本建议书 | 国际标准将使得运动视频能够作为一种计算机数据被处理，可以存储在各种不同的存储媒体上，能够在当前和未来网络上传送和接收，并且在现有和将来的广播信道上分配。

0.3 应用

本节不是本建议书 | 国际标准的组成部分。

本建议书 | 国际标准是针对使用广泛的视频应用设计的，包括但不限于下列各项：

CATV 在光网络、铜缆等介质上传输的有线电视
DBS 直接广播的卫星视频业务
DSL 数字用户线上的视频业务
DTTB 数字地面电视广播
ISM 交互式存储媒体（光盘等）
MMM 多媒体邮件
MSPN 分组交换网络上的多媒体业务
RTC 实时会话业务（视频会议、可视电话等）
RVS 远程视频监控
SSM 串行存储媒体（数字 VTR 等）

0.4 本规范的出版及其版本

本节不是本建议书 | 国际标准的组成部分。

本规范由 ITU-T 视频专家组（VCEG）和 ISO/IEC 运动图像专家组（MPEG）联合制定。分别在 ITU-T 和 ISO/IEC 两个组织中以技术对等的同文标准发布。

ITU-T H.264 建议书 | ISO/IEC 14496-10 版本 1 是指本建议书 | 国际标准的第一版（2003）。

ITU-T H.264 建议书 | ISO/IEC 14496-10 版本 2 是一份整合文本，包含根据第一份勘误表所指出的错误进行的修改。

ITU-T H.264 建议书 | ISO/IEC 14496-10 版本 3 是一份整合文本，包含第一份勘误表（2004），以及首次修订，即“保真度扩展”。

ITU-T H.264 建议书 | ISO/IEC 14496-10 版本 4（当前的规范）是一份整合文本，包含第一份勘误表（2004），首次修订（“保真度扩展”），以及第二份勘误表（2005）。ITU-T 在版本 2 后面发布的是版本 4（因为版本 4 的起草工作是在版本 3 最终文本批准之前完成的）。

0.5 简表和级别

本节不是本建议书 | 国际标准的组成部分。

本建议书 | 国际标准是为通用的应用场景设计的，适用于不同的应用、比特率、分辨率、质量和业务。应用领域包括数字存储媒体、电视广播和实时通信等。本规范的制定过程中考虑了来自不同典型应用的各种需求，开发了必要的算法，将他们整合在统一的语法规则之下。因此，本规范有利于视频数据在不同应用之间的交换。

考虑到实现本规范完整语法的可操作性，通过定义“简表”和“级别”的方法规定了很少的几个语法子集。相关术语的定义见第 3 节。

本建议书 | 国际标准规定的“简表”是指完整码流语法的一个子集。但是在给定简表的语法限定之下，编码器和解码器性能仍然可能差别很大，这取决于比特流中语法元素的取值，如解码图像大小等。目前在很多应用中，解码器能够处理一个简表下的所有可能的情况，这样做既不实用也不经济。

为了解决这一问题，每个简表下还定义了若干“级别”。级别是在某一简表下对语法元素和语法元素参数值的限定集合。这些限定可能仅针对量值。也可以以限定的组合形式出现（例如图像宽度乘以图像高度，再乘以每秒钟解码的图像数）。

符合本建议书 | 国际标准的已编码的视频内容使用统一的语法。为了构成完整语法的一个子集，码流中使用标志位、参数和其他语法元素，用来指示后续码流中某个语法元素的存在与否。

0.6 技术特征概述

本节不是本建议书 | 国际标准的组成部分。

编码表示的语法是为了能够在可接受的图像质量下获取较高压缩能力。除了高级 4:4:4 简表中无损编码采用的变换旁路的模式，和所有简表中的 I_PCM 模式外，其他算法并不是在编码和解码过程中始终保持样点原来的值，因而通常都不是无损的。获得高压缩能力可以采用很多技术。对每一帧中的矩形区域可以选择帧间或帧内编码算法（本建议书 | 国际标准不做规定）。通过使用基于块的运动矢量，帧间编码可以充分利用不同图像之间的时域统计依赖性。帧内编码采用不同的空间预测模式，对一幅图像中的空间统计依赖性加以利用。运动矢量和帧内预测模式可以通过图像中的不同块尺寸来定义。接下来是对预测残差进行变换，去除变换块内部像素之间的空间相关性以获得进一步的压缩，最后对变换系数进行量化。这个程是不可逆的，最终形成了与源图像非常接近的近似图像，同时丢弃一些视觉上不重要的信息。最后，再将运动矢量或帧内预测模式与量化后的变换系数信息合并在一起，进行变长编码或算术编码。

0.6.1 预测编码

本节不是本建议书 | 国际标准的组成部分。

随机访问与高效压缩这两个需求是相互矛盾的，因此本规范规定了两类主要的编码方式。帧内编码不需要参考其他图像。所以它可以充当随机访问点，解码可以从帧内编码图像开始，但是这种方式只能获得中等程度的压缩效率。帧间编码（预测和双向预测）可根据先前解码的图像对每个像素块进行帧间预测，这样压缩效率较高。另外，与其他视频编码标准不同是，本规范中的双向帧间预测图像也可以作为参考帧使用。

序列中的图像对上述三类编码方式的选用是很灵活的，并且解码的顺序通常不同于编码端源图像的捕获顺序或者解码端图像播放顺序。解码器根据具体应用的需求做出选择。对解码顺序做出规定是为了保证帧间预测图像的解码过程在其参考图像的解码后进行。

0.6.2 逐行和隔行视频的编码

本节不是本建议书 | 国际标准的组成部分。

本建议书 | 国际标准规定了对逐行和隔行扫描视频的语法和解码过程，二者是混合在同一个序列中的。隔行帧的两个场的捕获时间是不同的，而逐行帧的两个场捕获时间相同。两场可以独立编码，也可以联合编码。逐行帧一般按帧编码。而对隔行视频而言，编码器可以在帧编码和场编码之间选择。帧编码和场编码可以逐帧自适应地选择，并且可以在编码帧的局部选用。当视频场景包含大量细节且运动量较小的情况下一般首选帧编码。而在图像间存在剧烈运动时，场编码的效果比较好。

0.6.3 图像分割为宏块和更小的部分

本节不是本建议书 | 国际标准的组成部分。

本规范和以往的建议书和国际标准一样，也使用了宏块作为基本的视频编码处理单元，宏块包含一个 16×16 样点的亮度块和两个相应的色度块。

帧间预测时宏块可以进一步分割。帧间预测分割块大小的选择是在编码增益和表示运动补偿所需数据量之间折中的结果。本建议书 | 国际标准中，帧间预测过程可以使用 4×4 样点的块，1/4 像素精度的运动矢量进行运动补偿。一个样点块的帧间预测过程也可以包括从已存储的先前解码图像中选择参考图像。运动矢量利用相邻的已编码矢量做预测，进行差分编码。

通常由编码器计算得到合适的运动矢量和视频数据流中的其他数据元素。本建议书 | 国际标准不规定编码器上的运动估计过程和每个区域的帧间预测模式选择的方法。

0.6.4 时域冗余的削减

本节不是本建议书 | 国际标准的组成部分。

源图像和预测残差都含有大量的空间冗余。本建议书 | 国际标准采用基于块变换的方法削减空间冗余。根据先前编码图像进行帧间编码，或者根据当前图像中已编码样点进行帧内预测之后，得到的预测残差分成 4×4 大小的块。将他们变换到频域进行量化，量化之后很多变换系数为 0 或幅度很小，所以就可以用较少的编码数据来表示。本建议书 | 国际标准不规定变换和量化过程。

0.7 如何阅读本规范

本节不是本建议书 | 国际标准的组成部分。

建议读者从第 1 节（范围）开始，然后转到第 3 节（定义）。第 6 节规定了解码器源、输入和输出以及他们之间的关系。第 7 节（语法和语义）规定了从比特流中解析语法元素的顺序。7.1-7.3 节为语法顺序，7.4 节为语义描述；例如语法元素的范围、限制和条件等。第 9 节（解析过程）规定了大部分语法元素的解析过程。最后，第 8 节（解码过程）规定了语法元素如何映射到解码样值。在阅读本规范的过程中，读者可参阅第 2 节（规范参考文献）、第 4 节（缩写）和第 5 节（约定）。附件 A 到 E 也是本建议书 | 国际标准的组成部分。

附件 A 定义了 7 个简表（基准、主要、扩展、高级、高级 10、高级 4:2:2 和高级 4:4:4），每个简表适用于某个特定的应用领域，并且在这些简表下还定义了若干级别。附件 B 定义了字节流格式的语法和语义，以便检验比特流和解码器的一致性。附件 D 定义了补充增强信息的消息载荷的语法和语义。最后，附件 E 规定了视频可用性信息参数和序列参数集的语法和语义。

在本规范中，“注－”后面的语句是资料性的，不是本建议书 | 国际标准的组成部分。

1 范围

本建议书规定 ITU-T H.264 建议书 | ISO/IEC 国际标准 ISO/IEC 14496-10 视频编码。

2 规范性参考文献

下列 ITU-T 建议书和国际标准所包含的条款，通过在本建议书 | 国际标准中的引用而构成本建议书 | 国际标准的条款。在出版时，所指出的版本是有效的。所有的建议书和其他参考文献都面临修订，使用本建议书 | 国际标准的各方应探讨使用下列建议书和其他参考文献最新版本的可能性。国际电工委员会（IEC）和国际标准化组织（ISO）的成员负责维护当前有效的国际标准的注册。国际电信联盟电信标准化局负责维护当前有效的 ITU-T 建议书的注册。

- ITU-T Recommendation T.35 (2000), *Procedure for the allocation of ITU-T defined codes for non-standard facilities*.
- ISO/IEC 11578:1996, Annex A, *Universal Unique Identifier*.
- ISO/CIE 10527:1991, *Colorimetric Observers*.

3 定义

本建议书 | 国际标准采用下面的定义。

3.1 access unit 访问单元：一组 *NAL* 单元，通常包含一幅基本编码图像。除基本编码图像之外，也可能还包括一个或多个冗余编码图像或其他 *NAL* 单元，这些 *NAL* 单元中没有编码图像条带或条带数据分割块。一个访问单元的解码通常产生一幅解码图像。

3.2 transform coefficient 交流（AC）变换系数：一维或二维频率索引非零的变换系数。

3.3 adaptive binary arithmetic decoding process 自适应二进制算术解码过程：从自适应二进制算术编码过程产生的比特流中导出二进制位的值的熵解码过程。

3.4 adaptive binary arithmetic encoding process 自适应二进制算术编码过程：一种熵编码过程，在本建议书 | 国际标准中不做规定，对二进制位进行编码，能够产生可被自适应二进制算术解码过程所解码的比特流。

3.5 alpha blending α 混合：α 混合的处理过程在本建议书 | 国际标准中没有定义，此处理过程在显示过程中，联合使用了辅助解码图像和基本解码图像以及其他本规范未定义的数据。α 混合过程中，辅助解码图像的样点值解释为对应基本编码图像相应位置上亮度样点的不透明程度（或等同于透明度）。

3.6 arbitrary slice order 任意条带顺序：条带的解码顺序，图像中某些条带第一个宏块的宏块地址可能小于同一编码图像中的先前条带的第一个宏块地址。

3.7 auxiliary coded picture 辅助编码图像：基本编码图像的补充图像，在显示过程中可能与本建议书 | 国际标准未定义的其他数据一起使用。辅助编码图像与单色冗余编码图像有相同的语法规则。辅助编码图像必须包含和基本编码图像相同数目的宏块。辅助编码图像对解码过程没有影响。参阅基本编码图像和冗余编码图像。

3.8 B slice B 条带： 根据同一条带中已解码样点帧内预测，或从先前解码的参考图像经过帧间预测得到的条带，对每个块进行预测时最多使用两个运动矢量和参考索引。

3.9 Bin 二进制位：二进制串中的 1 比特。

3.10 Binarization 二值化：代表一个语法元素所有可能值的一组二进制串。

3.11 binarization process 二值化过程：语法元素所有可能值与一组二进制串之间的唯一映射过程。

3.12 bin string 二进制串：一串二进制位。二进制串是二值化的语法元素值的二进制表示。

3.13 bi-predictive slice 双向预测条带：见 *B* 条带。

3.14 bitstream 比特流：表示编码图像的及其相关数据，构成一个或多个编码视频序列的比特序列。比特流既可用来表示 NAL 单元流，也可表示字节流。

3.15 block 块：一个 M×N（M 行 N 列）的样点矩阵，或者一个 M×N 的变换系数矩阵。

3.16 bottom field 底场：组成一个帧的两个场中的一个。底场中的每一行在空间上位于顶场对应行的紧下方。

3.17 bottom macroblock (of a macroblock pair) （一个宏块对中的）底宏块：宏块对中的一个宏块，包含一个宏块对中所有底行的样点。对于一个场宏块对，底宏块表示位于宏块对空间区域内帧中底场区域的样点。对于一个帧宏块对，底宏块表示位于宏块对下半部分空间区域内帧中的样点。

3.18 broken link 断裂链接：比特流中的一个位置，它表示在解码顺序上其后面的一些图像可能包含严重的视觉瑕疵。这些瑕疵可能是在比特流生成时由未指明的操作导致的。

3.19 byte 字节：连续的 8 比特，读写时左边第一位是最高位，右边第一位是最低位。表示为数据位序列时，字节的最高有效位是第一位。

3.20 byte-aligned 字节对齐：从比特流的第一个比特开始的 8 的倍数的位置是字节对齐的位置。比特或字节或语法元素是字节对齐的，是指它出现在比特流字节对齐位置上。

3.21 byte stream 字节流：*NAL* 单元流的封装，包含开始码前缀和附件 B 定义的 *NAL* 单元。

3.22 can 可以：表示行为是允许的，但不一定是必须的。

3.23 category 类别：与每个语法元素相关联的数字。类别用来说明条带数据分割时语法元素在 *NAL* 单元中的分布。也可以用来按照具体应用所规定的方式（本建议书 | 国际标准未规定），代表语法元素的类别。

3.24 chroma 色度：一个形容词，它描述一个样点序列或单个样点，该描述一个样点序列或单个样点代表两个相对于基色的色差信号中的一个。色度序列或样点的符号为 Cb 和 Cr。

注 — 这里采用chroma，没有采用chrominance，是为了避免由于chrominance一词常和线性光学转移特性相联系而引起的混淆。

3.25 coded field 编码场：一个场的编码表示。

3.26 coded frame 编码帧：一个帧的编码表示。

3.27 coded picture 编码图像：一幅图像的编码表示。一个编码图像可以是一个编码场，也可以是一个编码帧。编码图像是一幅基本编码图像或一幅冗余编码图像的总称，但并不是二者的总称。

3.28 coded picture buffer (CPB) 编码图像缓存区：一个先进先出缓存区，包含附件 C 中假想参考解码器中规定的，按照解码顺序排列的访问单元。

3.29 coded representation 编码表示：以编码形式出现的数据元素。

3.30 coded video sequence 编码视频序列: 访问单元序列，由按照解码顺序排列的 *IDR* 访问单元和紧随其后的零个或多个非 *IDR* 访问单元组成，包括到下一个（不含）*IDR* 访问单元之前的所有访问单元。

3.31 component 分量：构成一个场或帧的一个距阵或 3 个距阵（1 个亮度矩阵，2 个色差矩阵）之一的单个样值。

3.32 complementary field pair 互补场对：互补的参考场对或互补的非参考场对的总称。

3.33 complementary non-reference field pair 互补的非参考场对： 两个位于解码顺序上相继的两个访问单元中非参考场，作为两个具有相反的奇偶性编码场，并且第一个场没有配对。

3.34 complementary reference field pair 互补的参考场对： 两个位于解码顺序上相继的两个访问单元中参考场，作为两个编码场，他们的 frame_num 语法元素值相同，解码顺序上的第二场不是 *IDR* 图像，并且不包含 memory_management_control_operation 值等于 5 的语法元素。

3.35 context variable 上下文变量：在某个二进制位的自适应二进制算术解码过程中，根据含有最近解码的二进制位的等式确定的变量。

3.36 DC transform coefficient 直流（DC）变换系数：所有维度上的频率索引均为 0 的变换系数。

3.37 decoded picture 解码图像：解码图像通过解码一幅编码图像得到。一幅解码图像既指一个解码帧，也指一个解码场。一个解码场可以是顶场，也可以是底场。

3.38 decoded picture buffer (DPB) 解码图像缓存区：保存解码图像的缓存区，用于附件 C 中假想参考解码器规定的预测参考、输出重排序或输出延时等。

3.39 decoder 解码器：解码过程的具体实现。

3.40 decoding order 解码顺序：解码过程中处理语法元素的顺序。

3.41 decoding process 解码过程：本建议书 | 国际标准规定的比特流读取和从中导出解码图像的过程。

3.42 direct prediction 直接预测：一种不用解码运动矢量的块的帧间预测模式。针对空域预测和时域预测又两种直接预测模式。

3.43 display process 显示过程：本建议书 | 国际标准不规定此过程。显示过程输入的经过剪裁的图像是解码过程的输出。

3.44 decoder under test (DUT) 被测解码器：进行与本建议书 | 国际标准一致性测试的解码器，操作假想流调度器同时向解码器和假想的参考解码器传送符合标准的比特流，比较两个解码器输出的值和定时信息。

3.45 emulation prevention byte 防伪字节：一个字节，值等于 0x03，可能在 *NAL* 单元中出现。防伪字节的出现可以保证在 *NAL* 单元的后续字节对齐的字节流中不会含有开始码前缀。

3.46 encoder 编码器：编码过程的具体实现。

3.47 encoding process 编码过程：产生符合本建议书 | 国际标准的比特流的过程，本建议书 | 国际标准对编码过程不做规定。

3.48 field 场：一帧的两个交替行组成的集合。一帧由两场组成，即一个顶场和一个底场。

3.49 field macroblock 场宏块：包含的样点仅来自一个场的宏块。一个编码场的所有宏块均为场宏块。当使用宏块自适应帧/场解码模式时，一个编码帧中的部分宏块可能是场宏块。

3.50 field macroblock pair 场宏块对：作为两个场宏块进行解码的一个宏块对。

3.51 field scan 场扫描：变换系数的排列顺序。与“*Z*”字形扫描顺序不同的是，它对列的扫描快于对行的扫描。场扫描用于场宏块中的变换系数。

3.52 flag 标志：可以取“0”或“1”两值的一个变量。

3.53 frame 帧：一个帧包含一个亮度矩阵样点和两个对应的色度矩阵样点。一帧包含两个场，即顶场和底场。

3.54 frame macroblock 帧宏块：一个宏块，它包含的样点来自同一个编码帧的两个场。当未使用宏块自适应帧/场解码模式时，一个编码帧中的所有宏块均为帧宏块。当使用了宏块自适应帧/场解码模式时，一个编码帧中的部分宏块可能为帧宏块。

3.55 frame macroblock pair 帧宏块对：作为两个帧宏块进行解码的一个宏块对。

3.56 frequency index 频率索引：与解码过程中反变换之前的变换系数相关的一维或二维索引。

3.57 hypothetical reference decoder (HRD) 假想参考解码器：一个假设的解码器模型，规定了对于编码过程中可能处理的符合标准的 *NAL* 单元流或字节流的可变性的约束。

3.58 hypothetical stream scheduler (HSS) 假想码流调度器：一个假想的向假想参考解码器传递输入比特流中定时信息和数据流的机制。HSS 用来检验比特流或解码器的一致性。

3.59 I slice 帧内（I）条带：非 *SI* 条带，并且该条带在解码时仅使用同一个条带内部的编码样点进行预测。

3.60 Informative 资料性：指本建议书 | 国际标准中提供的，但不是其组成部分的内容。资料性内容不设定任何需要与本建议书 | 国际标准保持一致的强制性要求。

3.61 instantaneous decoding refresh (IDR) access unit 即时解码刷新（IDR）访问单元：一个访问单元，其中的基本编码图像是一个即时解码刷新（IDR）图像。

3.62 instantaneous decoding refresh (IDR) picture 即时解码刷新（IDR）图像：一幅编码图像，其中所有条带为 I 或 SI 条带，导致在 IDR 图像解码之后解码过程将所有参考图像标记为“未用作参考”。IDR 图像解码之后，解码顺序上所有后续的编码图像都可以不用根据任何在 IDR 图像之前解码的图像来进行帧间预测解码。每个编码视频序列的第一幅图像为 IDR 图像。

3.63 inter coding 帧间编码：使用帧间预测对块、宏块、条带或图像进行编码。

3.64 inter prediction 帧间预测：不是根据当前解码图像，而是从已解码的参考图像得到的预测值。

3.65 interpretation sample value 解释性样点值：与辅助编码图像中已解码样值对应的可能替换值，可能在显示过程中用到。解释性样点值不用于解码过程，对解码过程没有影响。

3.66 intra coding 帧内编码：使用帧内预测对块、宏块、条带或图像进行编码。

3.67 intra prediction 帧内预测：在相同解码条带中使用先前已解码的样值生成当前样值的预测过程。

3.68 intra slice 帧内条带：见 *I* 条带。

3.69 inverse transform 反变换：解码过程的一部分，将一组变换系数转换为空域值，或者将一组变换系数转换为 *DC* 变换系数。

3.70 layer 层：没有分支的等级关系中的一组句法结构。高层包含低层。编码层指编码图像序列层、图像层、条带层和宏块层。

3.71 level 级别：对本建议书 | 国际标准中语法元素和变量取值所定义的一个限定集合。对所有简表定义了一组相同的级别，不同简表的每个级别大部分特性都是通用的。对于一个独立的实现，在一定的约束条件下，可以支持每个简表的不同级别。在另外的上下文中，级别指缩放之前变换系数的值。

3.72 list 0 (list 1) motion vector 列表 0（列表 1）运动矢量：与一个指向参考图像列表 *0*（列表 *1*）的参考索引相关联的运动矢量。

3.73 list 0 (list 1) prediction 列表 0（列表 1）预测：一个条带内容的帧间预测，使用指向参考图像列表 *0*（列表 *1*）的参考索引。

3.74 luma 亮度：说明单个样值或样值矩阵的形容词，这些样值代表原彩色图像中的单色信号。下文中用 Y 或 L 表示亮度。

注 — 这里采用luma，没有采用luminance，是为了避免由于luminance一词常和线性光学转移特性相联系而引起的混淆。有时用符号L而非Y是为了避免与代表纵坐标的y混淆。

3.75 macroblock 宏块：一个 16×16 的亮度样点块和相应的两个色度块样点。一个条带或宏块对划分为宏块称为分割。

3.76 macroblock-adaptive frame/field decoding 宏块自适应帧/场解码：编码帧的解码过程，其中一些宏块可以作为帧宏块解码，另外一些可以作为场宏块解码。

3.77 macroblock address 宏块地址：未使用宏块自适应帧/场解码模式时，宏块地址为图像的宏块光栅扫描的序号，该序号在图像中从左上角为 0 宏块开始编号。当使用了宏块自适应帧/场解码模式时，一个宏块对的顶宏块的宏块的地址是图像中宏块对光栅扫描编号的 2 倍，宏块对底宏块的宏块地址为相应顶宏块地址加 1。每个宏块对中的顶宏块地址均为偶数，底宏块地址均为奇数。

3.78 macroblock location 宏块位置：宏块在图像中的二维位置，以（x，y）表示。图像中左上角位置上的宏块其位置（x，y）等于（0，0）。对于每行宏块行 x 从左到右以 1 为步长递增。当未使用宏块自适应帧/场解码时，对于每列宏块 y 从上到下以 1 为步长递增。当使用了宏块自适应帧/场解码时，对于每列宏块 y 从上到下以 2 为步长递增，当一个宏块为底宏块时还要额外加 1。

3.79 macroblock pair 宏块对：使用宏块自适应帧/场解码时，一帧中配对的两个上下相接的宏块。条带分解为宏块对是一个分割块。

3.80 macroblock partition 宏块分割：为了进行帧间预测，从一个宏块的分割中得到的一个亮度块样点和两个相应的色度块样点。

3.81 macroblock to slice group map 宏块到条带的映射：将一幅图像的宏块映射到条带组的方法。宏块到条带组映射包含一个编号列表，每个编码宏块一个，规定了每个编码宏块所属的条带组。

3.82 map unit to slice group map 映射单元到条带组的映射:将一幅图像的条带组映射单元映射到条带组的一种方法。映射单元到条带组的映射包含一组编号，每个代表一个条带组映射单元，说明每个条带组映射单元对应的条带组。

3.83 may 可以：允许但不是必须的行为。在为了突出所述行为的可选特性时，用“可以或不可以”来表示强调。

3.84 memory management control operation 存储器管理控制操作：对参考图像的标记过程进行控制的 7 个操作。

3.85 motion vector 运动矢量：二维矢量，用于帧间预测，表示匹配对象在解码图像和在参考图像的位置偏移。

3.86 must 必须：表示对一个本建议书 | 国际标准中所指出的要求的观察或暗示。必须一词仅在资料性上下文中使用。

3.87 NAL unit NAL 单元：一个语法结构，包含后续数据的类型指示和所包含的字节数，数据以 RBSP 形式出现，必要时其中还散布有防伪字节。

3.88 NAL unit stream NAL 单元流：*NAL* 单元的序列。

3.89 non-paired field 未配对场：未配对参考场和未配对非参考场的总称。

3.90 non-paired non-reference field 未配对的非参考场：一个已解码的非参考场，它不是互补的非参考场对的一部分。

3.91 non-paired reference field 未配对的参考场：一个已解码的参考场，它不是互补的参考场对的一部分。

3.92 non-reference field 非参考场：以 nal_ref_idc 等于 0 进行编码的场。

3.93 non-reference frame 非参考帧：以 nal_ref_idc 等于 0 进行编码的帧。

3.94 non-reference picture 非参考图像：以 nal_ref_idc 等于 0 进行编码的图像。非参考图像不用于对任何图像进行帧间编码。

3.95 note 注：资料性注释的前缀词，仅用于资料性上下文中。

3.96 opposite parity 相反奇偶性：顶的相反奇偶性为底，反之亦然。

3.97 output order 输出顺序：从解码图像缓存区中输出解码图像的顺序。

3.98 P slice P 条带：可根据同一条带中已解码样点利用帧内预测进行解码，或者根据先前解码的参考图像利用帧间预测进行解码的条带，最多使用一个运动矢量和参考索引对每个块中的样点做预测。

3.99 Parameter 参数：序列参数集或图像参数集中的一个语法元素。参数也用于量化参数一词中。

3.100 Parity 奇偶性：场的奇偶性可为顶或底。

3.101 Partitioning 分割：将一个集合划分为不同的几个子集，使得每个元素均处于某个集合之中。

3.102 Picture 图像：场或帧的通称。

3.103 picture parameter set 图像参数集：一个语法结构，包含应用于零个或多个编码图像的语法元素，由每个条带头部中的语法元素 pic_parameter_set_id 确定。

3.104 picture order count 图像序号：一个变量，它随着图像位置在输出顺序上的递增具有非递减的值，此顺序递增相对于在解码顺序上的先前 *IDR* 图像，或者相对于将所有参考图像标志为“未用作参考”的包含存储管理控制操作的先前图像。

3.105 prediction 预测：预测过程的具体实现。

3.106 prediction process 预测过程：使用预测值得到当前解码的样点值或语法元素的一个估计。

3.107 predictive slice 预测条带：见 *P* 条带。

3.108 predictor 预测值：确定的值，或先前解码样点的值，或用于后续样点值或数据元素解码过程中的数据元素的联合。

3.109 primary coded picture 基本编码图像： 一幅图像的编码表示，用于与本建议书 | 国际标准相符合的比特流解码过程中。基本编码图像包含图像的所有宏块。只有基本编码图像才会影响解码过程。参见冗余编码图像。

3.110 profile 简表：本建议书 | 国际标准中的一个特定语法子集。

3.111 quantisation parameter 量化参数：一个变量，解码过程中对变换系数幅度进行缩放时使用。

3.112 random access 随机访问：比特流解码时的起始工作，不从流的开头位置开始。

3.113 raster scan 光栅扫描：矩形二维图案到一维图案的映射过程，一维图案的第一组值来自于二维图案最上边一行的从左到右扫描，然后依次是第二行、第三行等等。对于每种图案（由上到下）都是从左到右扫描的。

3.114 raw byte sequence payload (RBSP) 原始字节序列载荷：一个语法结构，包含整数个封装于 *NAL* 单元中的字节。RBSP 或者为空，或者包含具有数据比特串形式的语法元素，其后跟随 *RBSP* 截止位和零个或多个连续的 0 值比特。

3.115 raw byte sequence payload (RBSP) stop bit 原始字节序列载荷（RBSP）截止位：值为 1 的一个比特，出现在原始字节序列载荷（*RBSP*）中的数据比特串之后。*RBSP* 中数据比特串的结束位置可以通过搜索 *RBSP* 中最后一个非零比特－RBSP 截止位得到。

3.116 recovery point 恢复点：比特流中的一点，随机访问或断裂链接之后从此处得到一个该比特流所表示的精确或大致的解码图像。

3.117 redundant coded picture 冗余编码图像：图像或部分图像的编码表示。符合本建议书 | 国际标准的比特流的解码过程不应使用冗余编码图像。不要求冗余编码图像包含基本编码图像中的所有宏块。冗余编码图像不影响解码过程。参见基本编码图像。

3.118 reference field 参考场：参考场可以用于编码场中的 *P*、*SP* 和 *B* 条带或编码帧的场宏块解码时的帧间预测。参见参考图像。

3.119 reference frame 参考帧：参考帧可以用于编码帧中的 *P*、*SP* 和 *B* 条带解码时的帧间预测。参见参考图像。

3.120 reference index 参考索引：参考图像列表中的索引。

3.121 reference picture 参考图像：nal_ref_idc 不等于 0 的图像。参考图像包含可用于对解码顺序上后续图像的解码过程进行帧间预测的样点。

3.122 reference picture list 参考图像列表：用于 *P*、*B* 或 *SP* 条带帧间预测的一系列参考图像。*P* 或 *SP* 条带的解码过程只使用一个参考图像列表，而 *B* 条带的解码过程则要用到两个参考图像列表。

3.123 reference picture list 0 参考图像列表 0：用于 *P* 条带、*B* 条带或 *SP* 条带帧间预测的参考图像列表。*P* 条带和 *SP* 条带的所有帧间预测均使用参考图像列表 0。参考图像列表 0 是两个用于 *B* 条带帧间预测的参考图像列表之一，另外一个为参考图像列表 *1*。

3.124 reference picture list 1 参考图像列表 1：用于 *B* 条带帧间预测的参考图像列表。参考图像列表 1 是两个用于 *B* 条带帧间预测的参考图像列表之一，另外一个为参考图像列表 *0*。

3.125 reference picture marking 参考图像标记：在比特流中规定解码图像如何被标记为用于帧间预测。

3.126 reserved 保留：当保留一词出现在说明某些特定语法元素取值的条款中时，表示该取值供 ITU-T|ISO/IEC 将来使用。符合本建议书 | 国际标准的比特流不应该使用这些值，但是这些值将来可能在本建议书 | 国际标准的扩展版本中用到。

3.127 residual 残差：样点或数据元素预测值与解码值之间的解码差值。

3.128 run 游程：解码过程中连续出现的数据元素的数目。在某些上下文中环境中，游程指“*Z*”字形扫描或场扫描后产生的变换系数度序列中非 0 系数之前的 0 值变换系数度的数目。在另外的上下文中则指宏块的数目。

3.129 sample aspect ratio 样点幅型比：帧中亮度样点阵列中，列之间的水平距离与行之间的垂直举例之间的规定比例，用于显示过程（该过程不在本建议书 | 国际标准中规定）。样点幅型比表示为 *h*:*v*，其中 *h* 为水平宽度，*v* 为垂直高度（任意的空间距离单位）。

3.130 scaling 缩放：变换系数幅度乘以一个因子的过程，得到变换系数。

3.131 sequence parameter set 序列参数集：一个语法结构，包含应用于 0 个或多个完整编码视频序列的语法元素，由条带头中的语法元素 pic_parameter_set_id 确定所引用的图像参数集，由图像参数集中的语法元素 seq_parameter_set_id 确定所引用的序列参数集。

3.132 shall 必须：用于表示需要与本建议书 | 国际标准保持一致的强制性要求。当表示对语法元素的值或通过特定解码过程获得的结果的强制性约束时，由编码器负责保证约束的执行。当用于说明由解码过程所执行的操作时，产生相同结果的任何解码过程都与本建议书 | 国际标准中对解码过程要求相符合。

3.133 should 应该：用于说明在预期的通常情况下鼓励采取的实施行为，但其并不是与本建议书 | 国际标准保持一致的强制要求。

3.134 SI slice SI 条带：仅使用同一条带中的已解码样点进行预测编码的条带，并使用预测样点的量化。SI 条带可以被编码从而其中的已解码样点可以和 SP 条带一样重建。

3.135 skipped macroblock 跳过的宏块：一个指示表明宏块除了以“跳过”方式解码外没有任何数据被编码的宏块。这样的指示对几个宏块是通用的。

3.136 slice 条带：特定条带组内部按照光栅扫描顺序排列的整数个宏块或宏块对。对于基本编码图像，条带组分割为条带称为分割。虽然一个条带包含条带组内部按照光栅扫描顺序排列的整数个宏块或宏块对，但这些宏块或宏块对在图像内部并不一定是按照光栅扫描顺序连续排列的。宏块地址是通过条带第一个宏块的地址（条带头中描述）以及宏块到条带组的映射得到的。

3.137 slice data partitioning 条带数据分割：基于一个与每个语法元素都相关的类别，对选定的语法元素进行分割为语法结构的方法。

3.138 slice group 条带组：图像中宏块或宏块对的子集。将图像分为条带组是图像的一个分割。该分割通过宏块到条带组映射规定。

3.139 slice group map units 条带组映射单元：映射单元到条带组映射的单元。

3.140 slice header 条带头：编码条带的一部分，包含与该条带中第一个或者全部宏块有关的数据元素。

3.141 source 源：表示编码前视频素材或者素材的某些属性。

3.142 SP slice SP 条带：使用同一条带中的已解码参考样点进行帧间预测编码的条带，SP 条带使用至少一个运动矢量和参考索引来预测每个块的样点值。SI 条带可以被编码从而其中的已解码样点可以和另一个 SP 条带或 *SI* 条带一样重建。

3.143 start code prefix 开始码前缀：字节流中唯一等于 0x000001 的 3 个字节的序列，作为每个 *NAL* 单元的前缀。解码器可以利用开始码前缀的位置来确定一个新的 *NAL* 单元的开始和前一个 *NAL* 单元的结束。*NAL* 单元中通过加入防伪字节来防止假冒的开始码前缀出现。

3.144 string of data bits (SODB) 数据比特串：表示语法元素的若干比特位的序列，出现在原始字节序列荷载中原始字节序列荷载结束位之前。在 SODB 中，最左边的比特位是第一位并且是最高位，最右边的比特位则是最后一位且是最低位。

3.145 sub-macroblock 子宏块：位于宏块的四个角上的宏块的四分之一样点，例如一个 8×8 亮度块和两个相应色度块。

3.146 sub-macroblock partition 子宏块分割：一个亮度样点块和两个相应色度块，帧间编码预测时通过子宏块的分割得到。

3.147 switching I slice 切换 I 条带：见 *SI* 条带。

3.148 switching P slice 切换 P 条带：见 *SP* 条带。

3.149 syntax element 语法元素：比特流中表示数据的元素。

3.150 syntax structure 语法结构：零个或多个语法元素按照规定顺序一起出现在比特流中。

3.151 top field 顶场：组成帧的两个场之一。顶场中的每一行在空间上均位于底场中行的紧上方。

3.152 top macroblock (of a macroblock pair) 顶宏块（一个宏块对中的）：宏块对中的一个宏块，包含宏块对中的顶行样点。对于一个场宏块对，顶宏块表示帧中位于该宏块对区域中顶场所包含的样点。对于帧宏块对，顶宏块表示帧中位于该宏块对区域中上半部分的样点。

3.153 transform coefficient 变换系数：频率域的标量，与解码过程的反变换部分中一个特定的一维或二维频率索引相关联。

3.154 transform coefficient level 变换系数幅值： 一个与特定二维频率索引相关联的整数量值，解码过程中缩放之前用于计算变换系数的值。

3.155 universal unique identifier (UUID) 通用唯一性标识符：通用唯一标识符空间中具有唯一性的标识符。

3.156 unspecified 未指明：说明一个特定语法元素的某些值时出现的“未指明”，是指这些值在本建议书 | 国际标准中没有明确的意义，并且在本建议书 | 国际标准的未来版本中也不会有明确的意义。

3.157 variable length coding (VLC) 可变长度编码： 可逆的熵编码过程，为出现概率大的符号分配较短的码字，为出现概率小的符号分配较长的码字。

3.158 zig-zag scan “z”字形扫描：变换系数级从较低的空间频率到较高空间频率（近似）的一个明确排列顺序。“z”字形扫描用于帧宏块中的变换系数级。

4 缩写

下列缩写适用于本建议书|国际标准：

CABAC	基于上下文的自适应二进制算术编码
CAVLC	基于上下文的自适应变长编码
CBR	恒定比特率
CPB	编码图像缓存区
DPB	解码图像缓存区
DUT	被测解码器
FIFO	先进先出
HRD	假想参考解码器
HSS	假想码流调度器
IDR	即时解码刷新
LSB	最低有效位
MB	宏块
MBAFF	宏块自适应帧－场编码
MSB	最高有效位
NAL	网络抽象层
RBSP	原始字节序列载荷
SEI	补充的增强信息
SODB	数据比特串
UUID	通用唯一性标识符
VBR	可变比特率
VCL	视频编码层
VLC	变长编码

VUI 视频可用性信息

5 约定

注 — 本规范中所用的数学运算符类似于C程序语言。但这里明确定义了整数除法和算术移位运算。除特别说明，编号和计数约定从0开始。

5.1 算术运算符

算术运算符定义如下：

+	加法运算。
−	减法运算（二元运算符）或取反（一元前缀运算符）。
*	乘法运算。
x^y	指数运算，表示 x 的 y 次幂。在不是旨在表示指数的情况下也可表示上标。
/	整除运算。沿向 0 的取值方向截断。例如，7/4 和−7/−4 截断至 1，−7/4 和 7/−4 截断至−1。
÷	除法运算，不做截断或四舍五入。
$\frac{x}{y}$	除法运算，不做截断或四舍五入。
$\sum_{i=x}^{y} f(i)$	自变量 i 取由 x 到 y（含 y）的所有整数值时，函数 $f(i)$ 的累加和。
x % y	模运算。x 除以 y 的余数，其中 x 与 y 都是正整数。

在没有以插入括号来明确指定运算优先次序的情况下，遵守如下规则：

— 乘法和除法运算先于加法和减法运算

— 乘法和除法运算从左到右进行

— 加法和加法运算从左到右进行

5.2 逻辑运算符

逻辑运算符定义如下：

x && y	x 和 y 之间的“与”逻辑运算
$x \| \| y$	x 和 y 之间的“或”逻辑运算
!	逻辑“非”运算
x ? y : z	如果 x 为真或非 0 值，则取值为 y；否则取值为 z

5.3 关系运算符

关系运算符定义如下：

>	大于
>=	大于或等于
<	小于
<=	小于或等于
==	等于
!=	不等于

当一个关系运算符用于一个被赋予“na”（不可用）值的语法元素或变量时，“na”作为该语法元素或变量的一个明确取值对待。“na”不等于任何其他值。

5.4 位运算符

位运算符定义如下：

& 按位“与”运算。对整数进行运算时，以整数的二进制补码形式进行操作。如果两个二进制运算数中一个位数小于另外一个，则较短的运算数高位加 0 补齐。

| 按位“或”运算。对整数进行运算时，以整数的二进制补码形式进行操作。如果两个二进制运算数中一个位数小于另外一个，则较短的运算数高位加 0 补齐。

x >> y 将 x 以 2 的补码整数表示的形式向右移 y 位。仅当 y 取正数时定义此运算。右移运算移入 MSB 的位应该等于移位运算前 x 的 MSB 的值。

x<< y 将 y 以 2 的补码整数表示的形式向左移 y 位。仅当 y 取正数时定义此运算。左移运算移入 LSB 的位值为 0。

5.5 赋值运算符

赋值运算定义如下：

= 赋值运算符。

++ 递增，例如 x++相当于 $x = x+1$；当用于数组下标时，在自加运算前先求变量值。

-- 递减，例如 x--相当于 $x = x - 1$；当用于数组下标时，在自减运算前先求变量值。

+= 自加指定值，例如 x += 3 相当于 $x = x + 3$，x += (−3)相当于 $x = x + (-3)$。

−= 自减指定值，例如 x −= 3 相当于 $x = x - 3$，x −= (−3)相当于 $x = x - (-3)$。

5.6 取值范围记号

取值范围记号定义如下：

$x = y..z$ x 取从 y 至 z（含 z）的值，其中 x、y 和 z 是整数。

5.7 数学函数

数学函数定义如下

$$\text{Abs}(x) = \begin{cases} x & ; \quad x >= 0 \\ -x & ; \quad x < 0 \end{cases} \tag{5-1}$$

Ceil(x) 取不小于 x 的最小整数。 (5-2)

$$\text{Clip1}_Y(x) = \text{Clip3}(0, (1 << \text{BitDepth}_Y) - 1, x) \tag{5-3}$$

$$\text{Clip1}_C(x) = \text{Clip3}(0, (1 << \text{BitDepth}_C) - 1, x) \tag{5-4}$$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \textbf{其他情况} \end{cases} \tag{5-5}$$

Floor(x) 取不大于 x 的最大整数。 (5-6)

$$\text{InverseRasterScan}(a, b, c, d, e) = \begin{cases} (a\%(d/b)) * b; & e == 0 \\ (a/(d/b)) * c; & e == 1 \end{cases} \tag{5-7}$$

Log2(x) 取以 2 为底的 x 的对数。 (5-8)

Log10(x)取以 10 为底的 x 的对数 (5-9)

Median(x, y, z) = x + y + z − Min(x, Min(y, z)) − Max(x, Max(y, z)) (5-10)

$$\text{Min(x, y)} = \begin{cases} x & ; \quad x <= y \\ y & ; \quad x > y \end{cases} \tag{5-11}$$

$$\text{Max(x, y)} = \begin{cases} x & ; \quad x >= y \\ y & ; \quad x < y \end{cases} \tag{5-12}$$

Round(x) = Sign(x) * Floor(Abs(x) + 0.5) (5-13)

$$\text{Sign(x)} = \begin{cases} 1 & ; \quad x >= 0 \\ -1 & ; \quad x < 0 \end{cases} \tag{5-14}$$

$$\text{Sqrt(x)} = \sqrt{x} \tag{5-15}$$

5.8 变量、语法元素和表格

语法元素在比特流中以**粗体**字出现。每个语法元素均表示为名称（所有字母小写，以下划线连接），和一个或两个语法类别，以及一到两个代表其编码表示方式的描述符。解码过程根据语法元素及其前一个已解码语法元素的值进行。当表格或文本中用到某个语法元素的具体值时，则以常规形式（例如非粗体）出现。

某些情况下语法表可能使用根据语法元素值导出的其他变量的值。这些变量出现在语法表或文字中，以小写和大写混合的形式命名，并且名字中不含下划线。以大写字母开头的变量是为了当前语法结构和所有相关语法结构的解码导出的。以大写字母开头的变量可用于后续语法结构的解码过程，这些语法结构引用了产生该变量的那个语法结构。而以小写字母开头的变量只能在导出它的小节中使用。

在某些情况下，语法元素值或变量值的“识记”名称可以与其数值互换。有时，“识记”名称的使用与数量值无关。值和名称的关联在文字中做出规定。名称由一组或多组字母由下划线连接而成。每组字母均以大写字母开头，可包括多个大写字母。

注 — 语法的描述风格与C语言的语法相似。

函数用名称来描述，函数名由语法元素名称和左右圆括号中的零个或多个以逗号（若有多个变量时）分隔的变量名称（用于定义）或值（用于使用）构成。

一维的阵列称为列表。二维的阵列称为矩阵。阵列可以是语法元素，也可以是变量。下标或方括号可用来表示一个阵列的索引。对于一个矩阵，第一个下标为行（垂直）索引，第二个下标为列（水平）索引。使用方括号表示时，索引的顺序则正好相反。因此一个矩阵中的水平位置 x 和垂直位置 y 上的元素可表示为 s[x, y]或 s_{yx}。

单引号之间的一串比特值为二进制记号。例如，‘01000001’表示一个第二位和最后一位等于 1 的 8 比特串。

十六进制记号，以前缀“0x”表示，当所表示的比特位数为 4 的整数倍时可替代二进制记号使用。例如，“0x41”表示一个第二位和最后一位等于 1 的 8 比特串。

不使用单引号括起来的或不带前缀“0x”的数值为十进制值。

测试语句中等于 0 的值代表假（FALSE）的情况。用其他非零值代表真（TRUE）。

5.9 逻辑运算符的文字描述

在文本中，含有逻辑运算符的下列伪码语句

```
  if( 条件 0 )
  语句 0
 else if ( 条件 1 )
  语句 1
…
else /* 解释其他情况的注释*/
  语句 n
```

可表示为如下形式：

... 如下 / ...应用下列规则
— 如果条件 0， 则语句 0
— 否则，如果条件 1，语句 1
— …
— 否则（说明性文字，表示其他情况），语句 n

文中的每个“如果…否则，如果…否则…”语句都是由“…如下…”或“…应用下列规则”引导的，后面紧跟“如果…”。最后一个“如果…否则，如果…否则…”语句的条件一般是“否则，…”。交替出现的“如果…否则，如果…否则…”语句可以通过将“…如下…”或“…应用下列规则”和最后的“否则，…”配对加以识别。

文中，一个以下列伪码描述的逻辑运算语句

```
  if( 条件 0a  &&  条件 0b )
  语句 0
else if ( 条件 1a  ||  条件 1b )
  语句 1
…
else
  语句 n
```

可描述如下：

…如下/…应用下列规则
— 如果下列所有条件为真，声明 0
  — 条件 0a
  — 条件 0b
— 否则，如果下列任何一个条件为真，声明 1
  — 条件 1a
  — 条件 1b
— …
— 否则，声明 n

文中，一个以下列伪码描述的逻辑运算语句

```
  if( 条件 0 )
 声明 0
if (条件 1 )
 声明 1
```

可描述如下：

当条件 0 时，声明 0
当条件 1 时，声明 1

5.10 过程

过程用于描述语法元素的解码。一个过程的规范和调用是分离的。所有属于当前语法结构的语法元素和大写的变量，以及相关的语法结构，在过程的规范和调用中都是可用的。过程的规范中可能还含有明确指定为输入的小写的变量。每个规范均明确的规定了输出。输出可以是大写的变量，也可以是小写的变量。

变量的分配规定如下：

— 如果是调用一个过程，在变量名称与规范中的名称不相同的情况下，变量被明确指定为过程规范中的小写输入或输出变量。

— 否则（当变量在调用和规范中有相同名称时）不指定。

在过程的规范中，一个特定宏块可用一个值与其宏块地址相等的变量名指代。

6 源、已编码、已解码以及输出数据的格式、扫描过程和相邻关系

6.1 比特流格式

本小节规定 NAL 单元流和字节流之间的关系，二者均称作比特流。

比特流可以有两种格式：NAL 单元流格式或字节流格式。从概念上来说，NAL 单元流格式是一个更为“基本”的类型。由一系列称为 NAL 单元的语法结构组成，按照解码顺序排序。NAL 单元流中 NAL 单元的解码顺序（和内容）是受约束的。

字节流可以用 NAL 单元流构造，通过将 NAL 单元按照解码顺序排列，并且为每个 NAL 单元添加一个起始码前缀和若干零值字节可以形成一个字节流。NAL 单元流格式可以通过在字节流中搜索唯一的起始码前缀，从字节流格式中提取出来。除字节流格式以外，构造 NAL 单元的其他方法，本建议书 | 国际标准不做规定。字节流格式在附件 B 中规定。

6.2 源、已解码的以及输出的图像格式

本小节规定由比特流确定的源与已解码帧和场之间的关系。

比特流所表示的视频源是一系列按解码顺序排列的帧或场或帧场的组合（称为混合图像）。

每个源或已解码图像（帧或场）都是由一个或多个视频样点阵列组成的：

— 仅亮度（Y）（单色），附带或不带辅助阵列。

— 亮度和两个色度（YCbCr或YCgCo），附带或不带辅助阵列。

— 绿、蓝和红（GBR，也称为RGB），附带或不带辅助阵列。

— 表示其他未定义的单色或3基色样点（例如YZX，也称为XYZ）的阵列，附带或不带辅助阵列。

为了便于标记和命名，本规范中规定，不考虑实际使用的颜色表示方法，与这些阵列相关的变量和词语均指亮度（或 L，或 Y）和色度，这里两个色度阵列用 Cb 和 Cr 表示。实际所使用的颜色表示方法可以用附件 E 规定的语法来指示。编码视频序列中表示为或不表示为辅助图像的单色（辅助）阵列，对于解码过程是可选的，可用于 α 混合等用途。

变量 SubWidthC 和 SubHeightC 在表 6-1 中规定，它们取决于通过 chroma_format_idc 表示的色度采样结构。表 6-1 中的“-”表示 SubWidthC 或 SubHeightC 的值未定义。chroma_format_idc、SubWidthC 和 SubHeightC 的其他值将由 ITU-T | ISO/IEC 规定。

表 6-1－由chroma_format_idc决定的SubWidthC和SubHeightC的值

chroma_format_idc	色彩格式	SubWidthC	SubHeightC
0	单色	-	-
1	4:2:0	2	2
2	4:2:2	2	1
3	4:4:4	1	1

在单色采样中只有一个样点阵列，名义上当作亮度阵列。

在 4:2:0 样点中，两个色度阵列的高度和宽度均为亮度阵列的一半。

在 4:2:2 样点中，两个色度阵列的高度等于亮度阵列的高度，宽度为亮度阵列的一半。

在 4:2:4 样点中，两个色度阵列的高度和宽度与亮度阵列的相等。

亮度样点阵列的高度和宽度为 16 的整数倍。在使用 4:2:0 采样格式的比特流中，色度样点阵列的高度和宽度为 8 的整数倍。而在使用 4:2:2 样点的比特流中，色度样点阵列的宽度为 8 的整数倍，高度为 16 的整数倍。作为两场独立编码，或者使用宏块自适应帧－场编码（见下文）模式的亮度阵列，其高度为 32 的整数倍。4:2:0 样点的比特流中，每个作为两场独立编码，或者使用宏块自适应帧－场编码（见下文）模式的色度样点矩阵，其高度 16 的整数倍。解码过程输出的图像，高和宽不一定必须是 16 的整数倍，可通过剪裁矩形来规定。

除非特别说明，亮度和色度（当出现时）阵列的语法顺序为：当 3 个色彩分量数据都出现时，首先是亮度阵列的数据，然后是 Cb 阵列数据，最后是 Cr 阵列数据。

对于使用同一个序列参数集（见下文）编码的场和帧，他们宽度相同，场的高度是帧的一半。

视频序列中用来表示每个亮度或色度样点的比特位数在 8-12 之间，表示亮度阵列的比特数和表示色度阵列的比特位数可能不相同。

当 chroma_format_idc 的值等于 1 时，一帧中亮度和色度样点的垂直和水平相对位置如图 6-1 所示。其他的色度样点相对位置由视频可用性信息指出（见附件 E）。

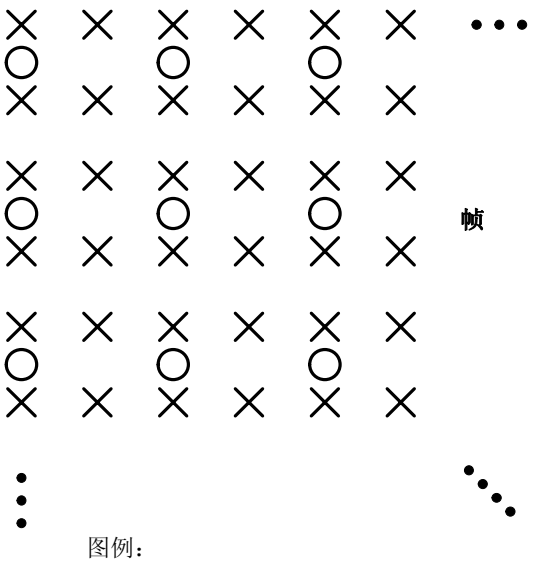


图 6-1－帧中4:2:0亮度和色度样点垂直和水平位置

一帧包含两场的情况如下文所述。一幅编码图像可以代表一个编码帧，也可代表单个编码场。符合本建议书 | 国际标准的编码视频序列可能含有编码帧和编码场的任意组合。通过宏块自适应帧－场编码模式的使用，解码过程也允许编码帧中的局部区域既可以帧方式编码，也可以场方式编码。

源和解码图像为两个类型之一：顶场和底场。当两场同时输出，或者联合起来被用做参考帧（见下文）时，它们（应该具有相反的奇偶性）是交织在一起的。解码帧中的第 1（以顶场为例）、3、5…行为顶场行。解码帧中的第 2、4、6…行为底场行。顶场仅包含解码帧中的顶场行。如果一个解码帧的顶场或底场被用做参考场（见下文），这时仅使用该解码帧中的奇数行（对顶场）或偶数行（对底场）。

当 chroma_format_idc 的值等于 1 时，顶场和底场中亮度和色度样点的垂直和水平相对位置如图 6-2 所示。顶场中色度样点的垂直样点位置相对于场的采样格点上移 1/4 个亮度样点的高度。底场中色度样点的垂直位置对于场的采样格点下移 1/4 个亮度样点的高度。其他的色度样点相对位置由视频可用性信息指出（见附件 E）。

注 — 色度样点的移位是为了使这些样点与图6-1中所示的整帧样点中的正常位置对齐。

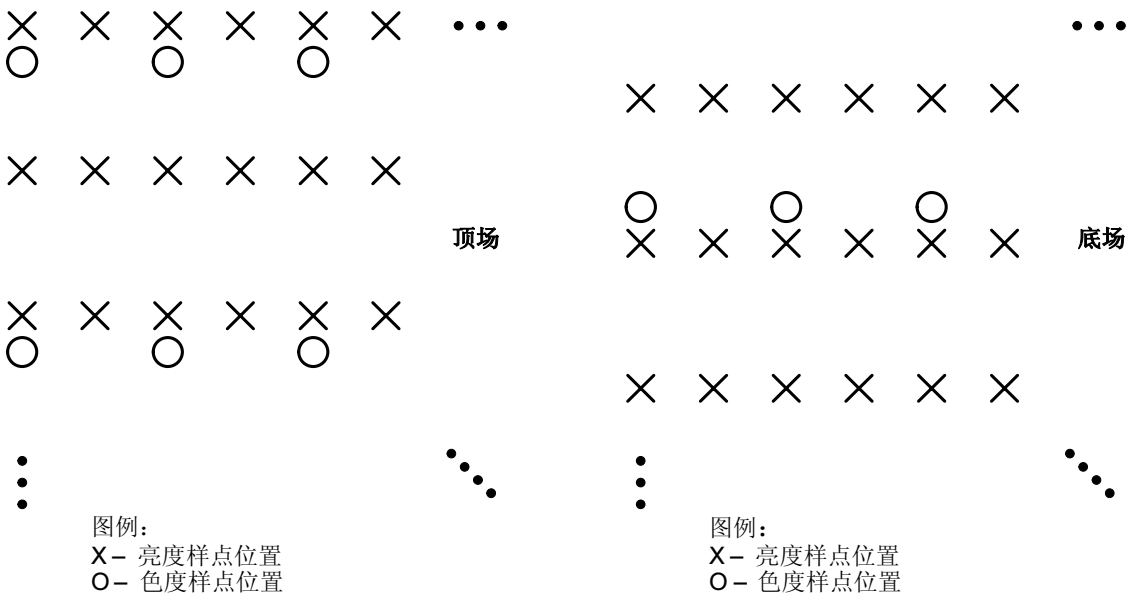


图 6-2－顶场和底场中4:2:0亮度和色度样点的垂直和水平位置

当 chroma_format_idc 的值等于 2 时，色度样点和对应的亮度样点处于同一位置上，帧和场中的样点位置分别如图 6-3－帧中 4:2:2 亮度和色度样点的垂直和水平位置和图 6-4 所示。

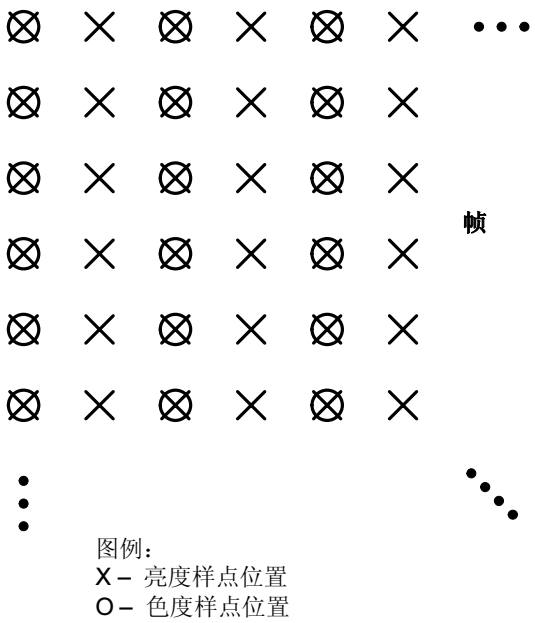


图 6-3－帧中4:2:2亮度和色度样点的垂直和水平位置

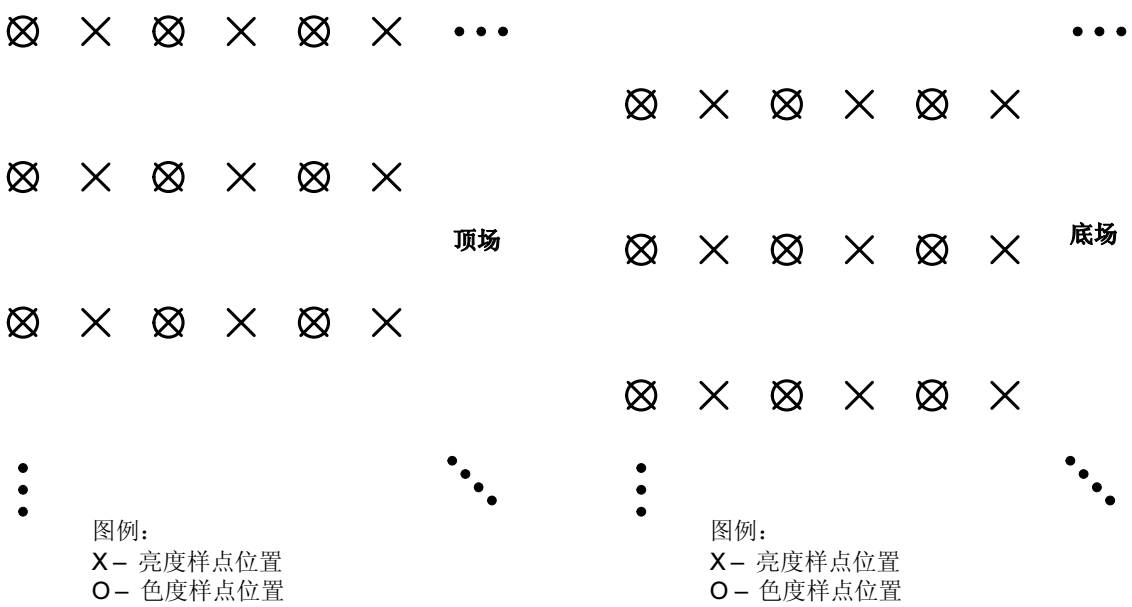


图 6-4－顶场和底场中4:2:2亮度和色度样点的垂直和水平位置

当 chroma_format_idc 的值等于 3 时，所有阵列中的样点均处于相同的位置上，帧和场中的样点位置分别如图 6-5 和图 6-6 所示。

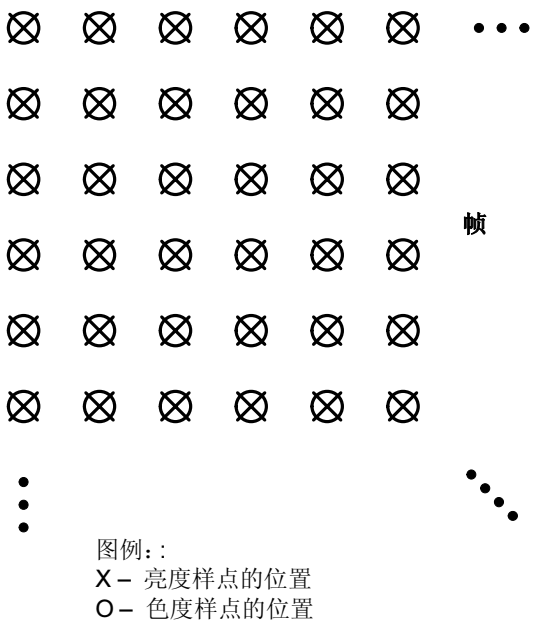


图 6-5－帧中4:4:4亮度和色度样点的垂直和水平位置

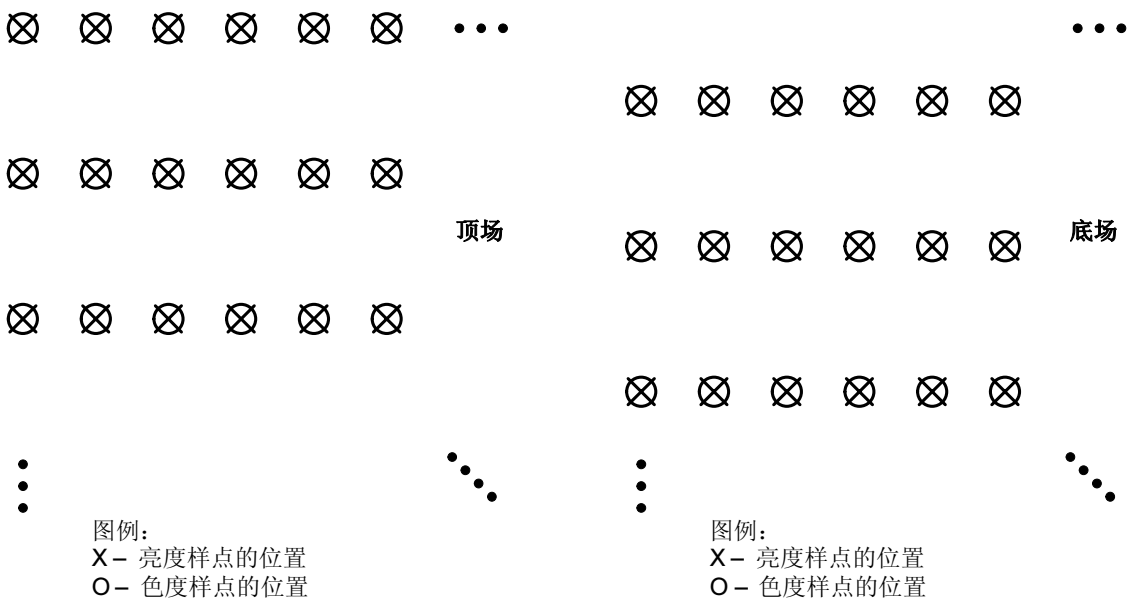


图 6-6－顶场和底场中4:4:4亮度和色度样点的垂直和水平位置

样点是以宏块为单元进行处理的。每个宏块中的样点阵列的高和宽均为 16 个样点。变量 MbWidthC 和 MbHeightC 分别规定了每个宏块中色度阵列的宽度和高度，其推导过程如下：

— 如果chroma_format_idc的值等于0（单色），则MbWidthC和MbHeightC均为0（单色视频没有色度阵列）。

— 否则，MbWidthC和MbHeightC按下式得到：

$$MbWidthC = 16 / SubWidthC \tag{6-1}$$

$$MbHeightC = 16 / SubHeightC \tag{6-2}$$

6.3 图像和条带的空间分割

本小节规定一幅图像如何分割为条带和宏块。图像被划分为条带。条带由一系列的宏块组成，当使用宏块自适应帧/场解码时则由一系列宏块对组成。

每个宏块均包含一个 16×16 的亮度阵列，当视频格式不是单色时，还包含和两个相应的色度阵列。如果没有使用宏块自适应帧/场解码，每个宏块代表图像中的一个空间矩形区域。例如，如图 6-7 所示，一幅图像被分为两个条带。

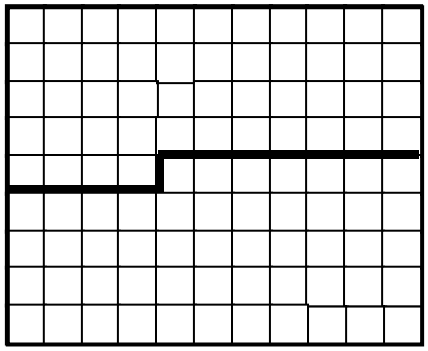


图 6-7－分割为两个条带的11×9个宏块的图像

当使用了宏块自适应帧/场解码时，图像被分割为若干含有整数个宏块对的条带，如图 6-8 所示。每个宏块对包含 2 个宏块。

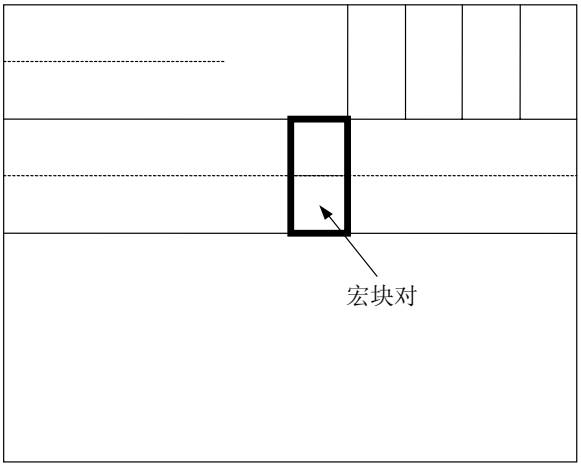


图 6-8－解码帧分割为宏块对

6.4 反向扫描过程和相邻数据的推导过程

本小节规定反向扫描过程，例如索引值到位置的映射，以及相邻数据的导出过程。

6.4.1 反向宏块扫描过程

本过程的输入是宏块地址 mbAddr。

本过程的输出为：地址是 mbAddr 的宏块中左上角亮度样点与所处图像左上角样点的相对位置（x，y）。

反向宏块扫描过程规定如下：

— 如果 MbaffFrameFlag 等于 0，

$$x = InverseRasterScan(mbAddr, 16, 16, PicWidthInSamples_L, 0) \quad (6\text{-}3)$$

$$y = InverseRasterScan(mbAddr, 16, 16, PicWidthInSamples_L, 1) \quad (6\text{-}4)$$

— 否则（MbaffFrameFlag 等于 1），按照下列规定：

$$xO = InverseRasterScan(mbAddr / 2, 16, 32, PicWidthInSamples_L, 0) \quad (6\text{-}5)$$

$$yO = InverseRasterScan(mbAddr / 2, 16, 32, PicWidthInSamples_L, 1) \quad (6\text{-}6)$$

根据当前宏块类型应用下列规则：

— 如果当前宏块为帧宏块，

$$x = xO \quad (6\text{-}7)$$

$$y = yO + (mbAddr \% 2) * 16 \quad (6\text{-}8)$$

— 否则（当前宏块为场宏块），

$$x = xO \quad (6\text{-}9)$$

$$y = yO + (mbAddr \% 2) \quad (6\text{-}10)$$

6.4.2 反向宏块分割和子宏块分割的扫描过程

宏块和子宏块可以进行分割，用于帧间预测的宏块和子宏块分割如图 6-9 所示。外边的矩形指向表示宏块或子宏块中的样点。矩形代表分割块。每个矩形中的编号表示反向宏块分割扫描或反向子宏块分割扫描的索引号。

表示宏块分割和子宏块分割宽度和高度的函数 MbPartWidth( )、MbPartHeight( )、SubMbPartWidth( )和 SubMbPartHeight( )分别在表 7-13、7-14、7-17 和 7-18 中规定。根据宏块的类型，对于每个宏块，函数 MbPartWidth( )和 MbPartHeight( ) 设定了适当的值。根据子宏块的类型，对 mb_type 值等于 P_8x8、P_8x8ref0 或 B_8x8 的宏块中的每个子宏块，函数 MbPartWidth( )和 MbPartHeight( ) 都设定了适当的值。

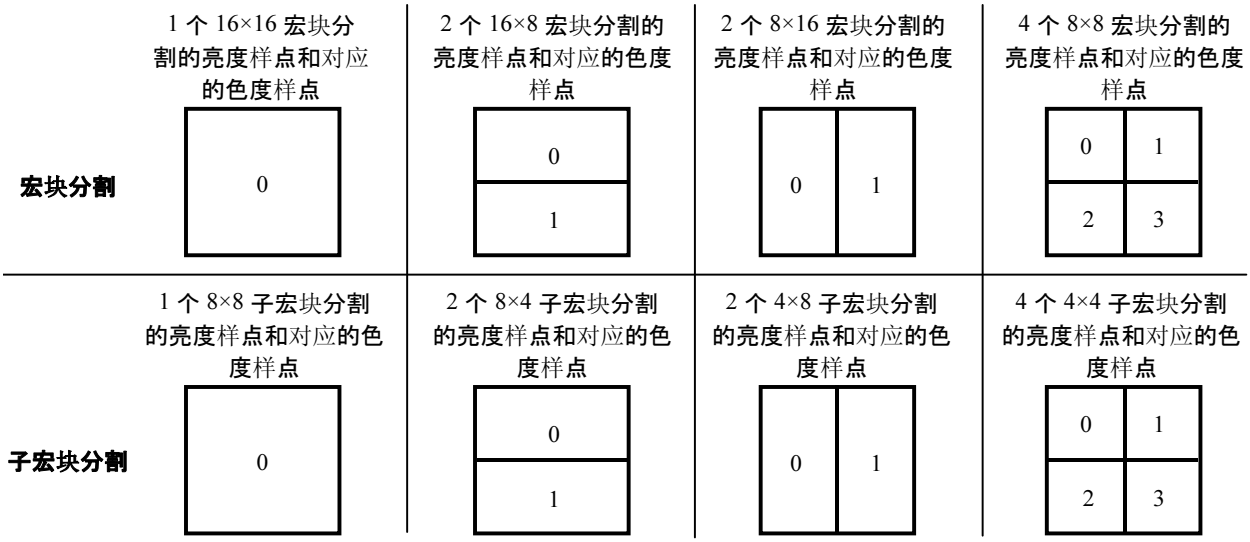


图 6-9－宏块分割，子宏块分割，宏块分割的扫描，子宏块分割的扫描

6.4.2.1 反向宏块分割扫描过程

本过程的输入是宏块分割的索引 mbPartIdx 。

本过程的输出为宏块分割 mbPartIdx 的左上角亮度样点与所处宏块左上角样点的相对位置（x，y）。

反向宏块分割扫描过程规定如下：

x = InverseRasterScan(mbPartIdx, MbPartWidth(mb_type), MbPartHeight(mb_type), 16, 0) (6-11)

y = InverseRasterScan(mbPartIdx, MbPartWidth(mb_type), MbPartHeight(mb_type), 16, 1) (6-12)

6.4.2.2 反向子宏块分割扫描过程

本过程的输入是宏块分割的索引 mbPartIdx 和子宏块分分割的索引 subMbPartIdx。

本过程的输出是子宏块分割 subMbPartIdx 中左上角亮度样点与所处子宏块左上角样点的相对位置（x，y）。

反向子宏块分割扫描过程规定如下：

— 如果 mb_type 等于 P_8x8、 P_8x8ref0 或 B_8x8，

x = InverseRasterScan(subMbPartIdx, SubMbPartWidth(sub_mb_type[mbPartIdx]),
SubMbPartHeight(sub_mb_type[mbPartIdx]), 8, 0) (6-13)

y = InverseRasterScan(subMbPartIdx, SubMbPartWidth(sub_mb_type[mbPartIdx]),
SubMbPartHeight(sub_mb_type[mbPartIdx]), 8, 1) (6-14)

— 否则，

x = InverseRasterScan(subMbPartIdx, 4, 4, 8, 0) (6-15)

y = InverseRasterScan(subMbPartIdx, 4, 4, 8, 1) (6-16)

6.4.3 反向4×4亮度块扫描过程

本过程的输入是 4×4 亮度块的索引 luma4x4BlkIdx 。

本过程的输出是索引为 luma4x4BlkIdx 的亮度块中左上角亮度样点与所处宏块左上角亮度样点的相对位置（x，y）。

图 6-10 表示 4×4 亮度块的扫描顺序。

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

图 6 -10－4×4亮度块的扫描顺序

4×4 亮度块的反向扫描过程规定为

x = InverseRasterScan(luma4x4BlkIdx / 4, 8, 8, 16, 0) + InverseRasterScan(luma4x4BlkIdx % 4, 4, 4, 8, 0) (6-17)

y = InverseRasterScan(luma4x4BlkIdx / 4, 8, 8, 16, 1) + InverseRasterScan(luma4x4BlkIdx % 4, 4, 4, 8, 1) (6-18)

6.4.4 反向8×8亮度块扫描过程

本过程的输入是 8×8 亮度块的索引 luma8x8BlkIdx。

本过程的输出为索引等于 luma8x8BlkIdx 的亮度块中左上角亮度样点与所处宏块左上角亮度样点的相对位置（x，y）。

图 6-11 表示 8×8 亮度块的扫描顺序。

0	1
2	3

图 6-11－8×8亮度块的扫描顺序

8×8 亮度块的反向扫描过程规定为：

x = InverseRasterScan(luma8x8BlkIdx, 8, 8, 16, 0) (6-19)

y = InverseRasterScan(luma8x8BlkIdx, 8, 8, 16, 1) (6-20)

6.4.5 宏块地址可用性的推导过程

此过程的输入是一个宏块地址 mbAddr。

此过程的输出是地址为 mbAddr 的宏块的可用性信息。

注 — 可用性的含义在该过程调用时给出。

宏块标志为可用，下述条件之一成立时，宏块被标志为不可用：

— mbAddr < 0

— mbAddr > CurrMbAddr

— 地址为mbAddr的宏块和地址为CurrMbAddr的宏块属于不同的条带

6.4.6 相邻宏块地址及其可用性的推导过程

仅当 MbaffFrameFlag 等于 0 时，调用本过程。

本过程的输出为：

— mbAddrA：当前宏块左侧宏块的地址和可用性状态。

— mbAddrB：当前宏块上侧宏块的地址和可用性状态。

— mbAddrC：当前宏块右上侧宏块的地址和可用性状态。

— mbAddrD：当前宏块左上侧宏块的地址和可用性状态。

图 6-12 表示地址为 mbAddrA、mbAddrB、mbAddrC 和 mbAddrD 的宏块与地址为 CurrMbAddr 的当前宏块在空间上的相对位置。

mbAddrD	mbAddrB	mbAddrC
mbAddrA	CurrMbAddr	

图 6-12－给定宏块的相邻宏块

6.4.5 节规定的过程的输入为 mbAddrA= CurrMbAddr – 1，输出为 mbAddrA 宏块是否可用。另外，当 CurrMbAddr % PicWidthInMbs 等于 0 时 mbAddrA 将被标识为不可用。

6.4.5 节规定的过程的输入为 mbAddrB = CurrMbAddr – PicWidthInMbs，输出为 mbAddrB 是否可用。

6.4.5 节规定的过程的输入为 mbAddrC = CurrMbAddr – PicWidthInMbs + 1，输出为 mbAddrC 是否可用。另外，当(CurrMbAddr + 1) % PicWidthInMbs 等于 0 时 mbAddrC 将被标识为不可用。

6.4.5 节规定的过程的输入为 mbAddrD = CurrMbAddr – PicWidthInMbs – 1，输出为 mbAddrD 是否可用。另外，当 CurrMbAddr % PicWidthInMbs 等于 0 时 mbAddrD 将被标识为不可用。

6.4.7 MBAFF帧中相邻宏块地址及其可用性的推导过程

仅当 MbaffFrameFlag 等于 1 时，调用本过程。

本过程的输出为：

— mbAddrA：当前宏块对左侧的宏块对中顶宏块的地址和可用性状态。

— mbAddrB：当前宏块对上侧的宏块对中顶宏块的地址和可用性状态。

— mbAddrC：当前宏块对右上侧的宏块对中顶宏块的地址和可用性状态。

— mbAddrD：当前宏块对左上侧的宏块对中顶宏块的地址和可用性状态。

图 6-13 表示地址为 mbAddrA、mbAddrB、mbAddrC 和 and mbAddrD 的宏块与当前宏块 CurrMbAddr 在空间上的相对位置。

不论当前宏块是宏块对中的顶宏块或底宏块，对于 mbAddrA、mbAddrB、mbAddrC 和 and mbAddrD 均有同样的值。

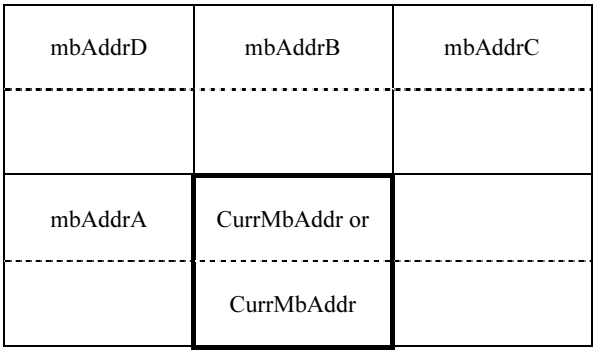


图 6-13－MBAFF帧中给定宏块的相邻宏块

6.4.5 小节规定的过程的输入为 mbAddrA=2 * (CurrMbAddr / 2 – 1)，输出为宏块 mbAddrA 是否可用。另外，mbAddrA 在(CurrMbAddr / 2) % PicWidthInMbs 等于 0 时将被标识为不可用。

6.4.5 小节规定的过程的输入为 mbAddrB =2 * (CurrMbAddr / 2 – PicWidthInMbs)，输出为宏块 mbAddrB 是否可用。

6.4.5 小节规定的过程的输入为 mbAddrC = 2 * (CurrMbAddr / 2 – PicWidthInMbs + 1)，输出为宏块 mbAddrC 是否可用。另外，mbAddrC 在(CurrMbAddr / 2 + 1) % PicWidthInMbs 等于 0 时将被标识为不可用。

6.4.5 小节规定的过程的输入为 mbAddrD = 2 * (CurrMbAddr / 2 – PicWidthInMbs －1)，输出为 mbAddrD 是否可用。另外，mbAddrD 在(CurrMbAddr / 2) % PicWidthInMbs 等于 0 时将被标识为不可用。

6.4.8 相邻宏块、块和分割块的推导过程

6.4.8.1规定了相邻宏块的推导过程。

6.4.8.2规定了相邻 8×8 亮度块的推导过程。

6.4.8.3规定了相邻 4×4 亮度块的推导过程。

6.4.8.4规定了相邻 4×4 色度块的推导过程。

6.4.8.5规定了相邻分割块的推导过程。

表 6-2 规定了输入的亮度位置的差分值（xD，yD）和 N 的对应关系。N 表示输出的 mbAddrN、mbPartIdxN、subMbPartIdxN、luma8x8BlkIdxN、luma4x4BlkIdxN 和 chroma4x4BlkIdxN 中 N 的替换值。6.4.8.1到6.4.8.5节使用了这里所规定的输入输出的分配关系。变量 predPartWidth 在使用表 6-2 时具体规定。

表 6-2－6.4.8.1到 6.4.8.5中输入输出对应关系的规范

N	xD	yD
A	−1	0
B	0	−1
C	predPartWidth	−1
D	−1	−1

图 6-14 表示当前宏块、块或分割使用帧编码模式时， 相邻的宏块、块或分割 A、B、C 和 D 与当前宏块、块或分割的相对位置。

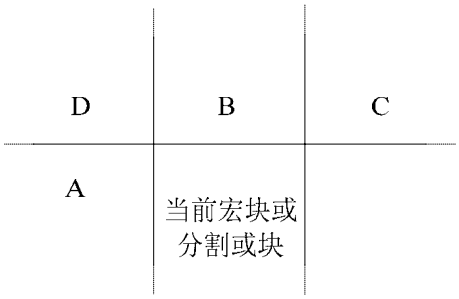


图 6-14－相邻宏块、块和分割的判决（资料性）

6.4.8.1 相邻宏块的推导过程

本过程的输出为：

— mbAddrA：当前宏块左侧宏块的地址及其可用性状态，以及

— mbAddrB：当前宏块上侧宏块的地址及其可用性状态。

mbAddrN（N为A或B）按如下规则导出：

— 亮度位置的差分值（xD，yD）按照表6-2设定。

— 对于（xN，yN）等于（xD，yD）的亮度位置，调用6.4.9规定的相邻位置的导出过程，得到的输出即为mbAddrN。

6.4.8.2 相邻的8×8亮度块的推导过程

本过程的输入为一个 8×8 亮度块索引 luma8x8BlkIdx。

luma8x8BlkIdx 指定了光栅扫描中宏块内的一个 8x8 亮度块。

本过程的输出为：

— mbAddrA：等于CurrMbAddr或当前宏块左侧宏块的地址及其可用性状态。

— luma8x8BlkIdxA：索引为luma8x8BlkIdx的8×8亮度块左侧的8×8亮度块的索引及其可用性状态。

— mbAddrB：等于CurrMbAddr或当前宏块上侧宏块的地址及其可用性状态。

— luma8x8BlkIdxB: 索引为luma8x8BlkIdx的8×8块上侧的8×8亮度块的及其可用性状态。

mbAddrN和luma8x8BlkIdxN（N为A或B）的导出过程如下：

— 亮度位置的差分值（xD，yD）根据表6-2设定。

— 亮度位置（xN，yN）按下式确定：

$$xN = (luma8x8BlkIdx \% 2) * 8 + xD \quad (6\text{-}21)$$

$$yN = (luma8x8BlkIdx / 2) * 8 + yD \quad (6\text{-}22)$$

— 调用6.4.9节的相邻位置推导过程时亮度位置（xN，yN）作为输入，输出则赋值给mbAddrN和（xW，yW）。

— 变量luma8x8BlkIdxN的推导如下：

— 如果mbAddrN不可用，则luma8x8BlkIdxN被标记为不可用。

— 否则（mbAddrN可用），宏块mbAddrN中覆盖了亮度位置（xW，yW）的8×8亮度块的索引被指定为luma8x8BlkIdxN的值。

6.4.8.3 相邻的4×4亮度块的推导过程

本过程的输入为一个 4×4 亮度块的索引。

本过程的输出为：

— mbAddrA：等于CurrMbAddr或等于左侧宏块的地址，及其可用性状态，

— luma4x4BlkIdxA：位于4×4块luma4x4BlkIdx左侧的4×4亮度块的索引及其可用性状态，

— mbAddrB：等于CurrMbAddr或等于上侧宏块的地址，及其可用性状态，

— luma4x4BlkIdxB：位于4×4块luma4x4BlkIdx上侧的4×4亮度块的编号及其可用性状态。

mbAddrN 和 luma4x4BlkIdxN（N 等于 A or B）推导如下：

— 亮度位置的差分值（ xD, yD ）按照表6-2设定。

— 以luma4x4BlkIdx为输入，调用6.4.3规定的4×4亮度块的反向扫描过程，输出为（x, y ）。

— 亮度位置（ xN, yN ）按下式规定：

$$xN = x + xD \quad (6\text{-}23)$$

$$yN = y + yD \quad (6\text{-}24)$$

— 以（xN, yN ）为输入，调用6.4.9规定的相邻位置推导过程，输出赋值给mbAddrN 和(xW, yW)。

— 变量luma4x4BlkIdxN的推导如下：

— 如果mbAddrN不可用，则将luma4x4BlkIdxN标记为不可用，

— 否则（mbAddrN可用），宏块mbAddrN中覆盖了亮度位置（xW, yW）的4×4亮度块指定为luma4x4BlkIdxN 。

6.4.8.4 相邻的4×4色度块的推导过程

本过程的输入为 4×4 色度块索引 chroma4x4BlkIdx 。

本过程的输出为：

— mbAddrA（等于CurrMbAddr或等于当前宏块左侧宏块的地址）及其可用性状态，

— chroma4x4BlkIdxA（索引为chroma4x4BlkIdx的4×4色度块左侧的4×4色度块的索引）及其可用性状态，

— mbAddrB（等于CurrMbAddr或等于当前宏块上侧宏块的地址）及其可用性状态，

— chroma4x4BlkIdxB（索引为chroma4x4BlkIdx的4×4色度块上侧的4×4色度块的索引）及其可用性状态，

— mbAddrN和chroma4x4BlkIdxN（N可以为A或B）推导如下：

— 色度位置的差分值(xD, yD)按照表6-2设定。

— 根据chroma_format_idc的不同取值，索引为chroma4x4BlkIdx的4×4色度块左上角位置（x，y）推导如下：

— 如果chroma_format_idc等于1或2，则

x = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (6-25)

y = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (6-26)

— 否则 (chroma_format_idc 等于3)

x = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 0) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 0) (6-27)

y = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 1) (6-28)

— 色度位置 (xN, yN)按下式规定：

xN = x + xD (6-29)

yN = y + yD (6-30)

— 调用6.4.9中规定的相邻位置的推导过程，以(xN, yN)为输入，输出分别赋值给mbAddrN和(xW, yW)。

— 变量chroma4x4BlkIdxN推导如下：

— 如果mbAddrN不可用，则chroma4x4BlkIdxN标志为不可用。

— 否则（mbAddrN可用），宏块mbAddrN中覆盖了样点(xW, yW)的4×4色度块即为chroma4x4BlkIdxN。

6.4.8.5 相邻分割块的推导过程

本过程的输入为：

— 宏块分割块的索引mbPartIdx

— 当前子宏块类型currSubMbType

— 子宏块分割块的索引subMbPartIdx

本过程的输出为：

— mbAddrA\mbPartIdxA\subMbPartIdxA：规定当前宏块左侧的宏块或子宏块分割块及其可用性状态，或者子宏块分割块CurrMbAddr\mbPartIdx\subMbPartIdx左侧的宏块或子宏块分割块及其可用性状态，

— mbAddrB\mbPartIdxB\subMbPartIdxB：规定当前宏块上侧的宏块或子宏块分割块及其可用性状态，或者子宏块分割块CurrMbAddr\mbPartIdx\subMbPartIdx上侧的的宏块或子宏块分割块及其可用性状态，

— mbAddrC\mbPartIdxC\subMbPartIdxC：规定当前宏块右上侧的宏块或子宏块分割块及其可用性状态，或者子宏块分割块CurrMbAddr\mbPartIdx\subMbPartIdx右上侧的宏块或子宏块分割块及其可用性状态，

— mbAddrD\mbPartIdxD\subMbPartIdxD：规定当前宏块左上侧的宏块或子宏块分割块及其可用性状态，或者子宏块分割块CurrMbAddr\mbPartIdx\subMbPartIdx左上侧的宏块或子宏块分割块及其可用性状态。

— mbAddrN、mbPartIdxN和subMbPartIdx （N等于A, B, C, 或D）推导如下：

— 以mbPartIdx作为输入，调用6.4.2.1规定的反向宏块分割扫描过程，(x, y)作为输出。

— 宏块分割中左上角的亮度样点位置（ xS, yS）推导如下：

— 如果mb_type等于 P_8x8、P_8x8ref0或B_8x8，则调用6.4.2.2中规定的反向子宏块分割的扫描过程，subMbPartIdx为输入，输出为（xS, yS）。

— 否则 （xS, yS）等于（0，0）

— 表6-2中的变量predPartWidth规定如下：

— 如果mb_type等于P_Skip、B_Skip或B_Direct_16x16，则predPartWidth = 16。

— 否则，如果mb_type等于B_8x8，应用如下过程：

— 如果currSubMbType等于B_Direct_8x8, 则predPartWidth = 16。

注 1 — 当currSubMbType等于B_Direct_8x8，并且direct_spatial_mv_pred_flag等于1时，预测的运动矢量为整个宏块的预测运动矢量。

— 否则，predPartWidth = SubMbPartWidth(sub_mb_type[mbPartIdx])。

— 否则，如果mb_type 等于P_8x8或P_8x8ref0， predPartWidth = SubMbPartWidth(sub_mb_type [mbPartIdx])。

— 否则，predPartWidth = MbPartWidth(mb_type)。

— 亮度位置的差分值（ xD, yD）按照表6-2设定。

— 相邻亮度位置（ xN, yN ）按下式确定：

$$xN = x + xS + xD \quad (6\text{-}31)$$

$$yN = y + yS + yD \quad (6\text{-}32)$$

— 为获得亮度位置，以（ xN, yN ）为输入，调用6.4.9规定的相邻位置的推导过程，输出即赋给 mbAddrN和(xW, yW)。

— 根据mbAddrN的可用性状态，应用下列规定：

— 如果mbAddrN不可用，则将宏块或子宏块分割块mbAddrN\mbPartIdxN\subMbPartIdxN 标志为不可用。

— 否则（mbAddrN可用），应用下列规定：

— 宏块mbAddrN中覆盖亮度位置（xW, yW）的宏块分割块被指定为mbPartIdxN，而且在宏块mbAddrN内，宏块分割mbPartIdxN中覆盖（xW, yW）的子宏块分割块被指定为subMbPartIdxN。

— 当由mbPartIdxN和subMbPartIdxN给出的分割块还没有被解码时，则将宏块分割块mbPartIdxN和子宏块分割块subMbPartIdxN标志为不可用。

注 2 — 后者的条件是如下情况：当mbPartIdx = 2，subMbPartIdx = 3，xD = 4，yD = -1，例如当第三个子宏块的最后一个4×4亮度块相邻的C被请求。

6.4.9 相邻位置的推导过程

本过程的输入为相对于当前宏块左上角样点位置的亮度或色度位置（xN，yN）。

本过程的输出为：

— mbAddrN：等于CurrMbAddr或等于包含（xN，yN）的相邻宏块的地址，及其可用性状态，

— (xW, yW)：表示与宏块mbAddrN左上角的相对（不是相对于当前宏块左上角）位置（xN，yN）。

令 maxW 和 maxH 为分别为表示位置分量 xN、xW 和 yN、yW 最大值的两个变量。maxW 和 maxH 的推导过程如下：

— 如果此过程为计算相邻亮度位置而调用，则

maxW = maxH = 16 (6-33)

— 否则（此过程为计算相邻色度位置而调用），

maxW = MbWidthC (6-34)

maxH = MbHeightC (6-35)

根据 MbaffFrameFlag 的不同取值，相邻位置的推导如下：

— 如果MbaffFrameFlag等于0，遵照6.4.9.1对场和非MBAFF帧中相邻位置的规定。

— 否则（MbaffFrameFlag等于1），遵照6.4.9.2对MBAFF帧中相邻位置的规定。

6.4.9.1 对场和非MBAFF帧中相邻位置的规范

本小节的规定当 MbaffFrameFlag 等于 0 时应用。

调用 6.4.6 所规定的相邻宏块地址及其可用性的推导过程，输出为 mbAddrA、mbAddrB、mbAddrC 和 mbAddrD 以及他们的可用性状态。

表 6-3 规定了(xN, yN)与 mbAddrN 之间的对应关系。

表 6-3－mbAddrN的规范

xN	YN	mbAddrN
< 0	< 0	mbAddrD
< 0	0 .. maxH －1	mbAddrA
0 .. maxW －1	< 0	mbAddrB
0 .. maxW －1	0 .. maxH －1	CurrMbAddr
> maxW －1	< 0	mbAddrC
> maxW －1	0 .. maxH －1	not available
	> maxH －1	not available

相对于宏块 mbAddrN 左上角的相邻位置（xW, yW ）按下式推导：

xW = (xN + maxW) % maxW (6-36)

yW = (yN + maxH) % maxH (6-37)

6.4.9.2 对MBAFF帧中相邻位置的规范

本小节的规定仅当 MbaffFrameFlag 等于 1 时应用。

调用 6.4.7 所规定的相邻宏块地址及其可用性的推导过程，输出为 mbAddrA、mbAddrB、mbAddrC 和 mbAddrD 以及他们的可用性状态。

表 6-4 按照下列两个步骤规定了宏块地址 mbAddrN 和 yM：

1．由（xN, yN ）决定的宏块地址 mbAddrX，以及下列变量：

— 变量currMbFrameFlag推导如下：

  — 如果地址为CurrMbAddr的宏块是帧宏块，currMbFrameFlag等于1；

  — 否则（地址为CurrMbAddr的宏块是场宏块），currMbFrameFlag等于0。

— 变量mbIsTopMbFlag推导如下：

  — 如果地址为CurrMbAddr的宏块是顶宏块（CurrMbAddr % 2等于0），则mbIsTopMbFlag等于1；

  — 否则（地址为CurrMbAddr的宏块是底宏块，CurrMbAddr % 2等于1），mbIsTopMbFlag 等于0。

2．根据mbAddrX的可用性，应用下列规则：

— 如果mbAddrX不可用，则将mbAddrN标志为不可用。

  — 否则（mbAddrX可用），将mbAddrN标志为可用，且表6-4根据（ xN, yN ）、currMbFrameFlag、mbIsTopMbFlag以及mbAddrXFrameFlag规定了mbAddrN，其中变量mbAddrXFrameFlag推导如下：

  — 如果宏块mbAddrX为帧宏块，则mbAddrXFrameFlag等于1；

  — 否则（宏块mbAddrX为场宏块），mbAddrXFrameFlag 等于0。

上述标志（flag）在表6-4中未规定值的（na），表示相应标志与表中所在的行无关。

表 6-4－mbAddrN和yM的规范

xN	yN	currMbFrameFlag	mbIsTopMbFlag	mbAddrX	mbAddrXFrameFlag	附加条件	mbAddrN	yM
< 0	< 0	1	1	mbAddrD			mbAddrD + 1	yN
			0	mbAddrA	1		mbAddrA	yN
					0		mbAddrA + 1	(yN + maxH) >> 1
		0	1	mbAddrD	1		mbAddrD + 1	2*yN
					0		mbAddrD	yN
			0	mbAddrD			mbAddrD + 1	yN
< 0	0 .. maxH －1	1	1	mbAddrA	1		mbAddrA	yN
					0	yN % 2 = = 0	mbAddrA	yN >> 1
						yN % 2 != 0	mbAddrA + 1	yN >> 1
			0	mbAddrA	1		mbAddrA + 1	yN
					0	yN % 2 = = 0	mbAddrA	(yN + maxH) >> 1
						yN % 2 != 0	mbAddrA + 1	(yN + maxH) >> 1
		0	1	mbAddrA	1	yN < (maxH / 2)	mbAddrA	yN <<1
						yN >= (maxH / 2)	mbAddrA + 1	(yN <<1) －maxH
					0		mbAddrA	yN
			0	mbAddrA	1	yN < (maxH / 2)	mbAddrA	(yN <<1) + 1
						yN >= (maxH / 2)	mbAddrA + 1	(yN <<1) + 1 – maxH
					0		mbAddrA + 1	yN
0 .. maxW – 1	< 0	1	1	mbAddrB			mbAddrB + 1	yN
			0	CurrMbAddr			CurrMbAddr －1	yN
		0	1	mbAddrB	1		mbAddrB + 1	2 * yN
					0		mbAddrB	yN
			0	mbAddrB			mbAddrB + 1	yN
0 .. maxW – 1	0 .. maxH －1			CurrMbAddr			CurrMbAddr	yN
> maxW – 1	<0	1	1	mbAddrC			mbAddrC + 1	yN
			0	不可用			不可用	na
		0	1	mbAddrC	1		mbAddrC + 1	2 * yN
					0		mbAddrC	yN
			0	mbAddrC			mbAddrC + 1	yN
> maxW – 1	0 .. maxH －1			不可用			不可用	na
	> maxH －1			不可用			不可用	na

按下式推导相对于宏块 mbAddrN 左上角的相邻位置（xW, yW ）：

xW = (xN + maxW) % maxW (6-38)

yW = (yM + maxH) % maxH (6-39)

7 语法和语义

7.1 以表格形式描述语法的方法

语法表格规定了所有允许输入的比特流的超集。附加的语法限定可能在其他章节中直接或间接规定。

注 — 实际的解码器应该有识别比特流入口点的方法，并且可以分辨和处理不一致的比特流。分辨和处理错误以及类似情形的方法不在本建议书中描述。

下面的表格给出了描述语法的伪代码例子。规定了当 **syntax_element** 出现时，从比特流中解析语法元素，并将指针移向比特流中下一个语法元素的位置上的过程。

	C	描述符
/* 语句可以是一个关联某一语法类别的语法元素和描述符，或者用于说明语法元素的存在、类型和数值的表达式，下面给出两个例子。*/		
syntax_element	3	ue(v)
条件语句		
/*花括号括起来的语句组是复合语句，在功能上视作单个语句。 */		
{		
语句		
语句		
…		
}		
/* “while”语句测试条件是否为 TRUE，如果为 TRUE，则重复执行循环体，直到条件不为 TRUE。 */		
While(条件)		
语句		
/* “do … while”语句先执行循环体一次，然后测试条件是否为 TRUE，如果为 TRUE，则重复执行循环体，直到条件不为 TRUE。*/		
Do		
语句		
while(条件)		
/*“if … else”语句首先测试条件，如果为 TRUE，则执行主要语句，否则执行另选语句。如果另选语句不需要执行，结构的“else”部分和相关的另选语句可忽略。*/		
if(条件)		
主要语句		
Else		
另选语句		
/* “for”语句首先执行最初语句，然后测试条件，如果条件为 TRUE，则重复执行主要语句和随后语句直到条件不为 TRUE。 */		
for(最初语句; 条件; 随后语句)		
主要语句		

7.2 语法函数、类别和描述符的规范

以下函数用于语法描述。这些函数假定解码器中存在一个比特流指针，这个指针指向比特流中解码过程要读取的下一个比特的位置。

byte_aligned( ) 的规定如下：

— 如果比特流的当前位置是在字节边界，即，比特流中的下一比特是字节第一个比特，byte_aligned( )的返回值为TRUE。

— 否则，byte_aligned( )的返回值为FALSE。

more_data_in_byte_stream( )，只有在附件 B 规定的字节流 NAL 单元语法结构中使用，规定如下：

— 如果字节流中后续还有更多数据，more_data_in_byte_stream( )的返回值为TRUE。

— 否则，more_data_in_byte_stream( ) 的返回值为 FALSE。

more_rbsp_data( ) 的规定如下：

— 如果在rbsp_trailing_bits( )之前的RBSP中有更多数据，more_rbsp_data( ) 的返回值为TRUE。

— 否则，more_rbsp_data( )的返回值为 FALSE。

判断 RBSP 中是否有更多数据的方法由应用规定（或者附件 B 中使用字节流格式的应用）。

more_rbsp_trailing_data( ) 的规定如下：

— 如果RBSP中有更多数据，more_rbsp_trailing_data( ) 的返回值为TRUE。

— 否则，more_rbsp_trailing_data( ) 的返回值为 FALSE。

next_bits(n)提供比特流中接下来的比特用于比较的目的，而不需要移动比特流指针。该函数使比特流中的下 n 个比特可见，n 在这里是函数的参数。当用在附件 B 规定的字节流中时，如果剩余的字节流已不足 n 个比特，next_bits(n)返回值为 0。

read_bits(n) 从比特流中读取下面的 n 个比特，并且将比特流指针向前移动 n 个比特。当 n 等于 0 时，read_bits(n)的返回值为 0 并且不移动比特流指针。

类别（在表中以 **C** 表示）规定条带数据可以至多划分为三种条带数据类别。条带数据类别 A 包含了类别 2 的所有语法元素。条带数据类别 B 包含了类别 3 的所有语法元素。条带数据类别 C 包含了类别 4 的所有语法元素。其他类别值的含义不作规定。某些语法元素需要使用两个类别值，这两个值通过竖线分开。在这些情况下，本文将会进一步说明应用的类别值的含义。对于在其他语法结构中使用的语法结构，它所包含的所有语法元素的类别值都应列出，通过竖线来分开。如果语法元素或者语法结构的类别标为"All"，它可以出现在所有的语法结构中。对于用在其他语法结构中的语法结构，语法表格中的数字类别值如果处于包含了一个类别值为“All”的语法结构中，那么该数字类别值被认为能够应用到类别为“All”的语法元素值。

如下描述符规定了每个语法元素的解析处理。对于某些语法元素，需要使用通过竖线分开的两个描述符。在这些情况下，左边的描述符在 entropy_coding_mode_flag 等于 0 的时候使用，右边的描述符在 entropy_coding_mode_flag 等于 1 的时候使用。

— ae(v)：上下文自适应算术熵编码语法元素。该描述符的解析过程在9.3节中规定。

— b(8)：任意形式的8比特字节。该描述符的解析过程通过函数read_bits(8)的返回值来规定。

— ce(v)：左位在先的上下文自适应可变长度熵编码语法元素。该描述符的解析过程在9.2节中规定。

— f(n)：n位固定模式比特串（由左至右），左位在先， 该描述符的解析过程通过函数read_bits(n)的返回值来规定。

— i(n)：使用n比特的有符号整数。在语法表中，如果*n*是‘v’，其比特数由其它语法元素值确定。解析过程由函数read_bits(*n*)的返回值规定，该返回值用最高有效位在前的2的补码表示。

— me(v)：映射的指数哥伦布码编码的语法元素，左位在先。解析过程在9.1中定义。

— se(v)：有符号整数指数哥伦布码编码的语法元素位在先。解析过程在9.1中定义。

— te(v)：舍位指数哥伦布码编码语法元素，左位在先。解析过程在9.1中定义。

— u(n)：*n*位无符号整数。在语法表中，如果*n*是‘v’，其比特数由其它语法元素值确定。解析过程由函数read_bits(*n*)的返回值规定，该返回值用最高有效位在前的二进制表示。

— ue(v)：无符号整数指数哥伦布码编码的语法元素，左位在先。解析过程在 9.1 中定义。

7.3 以表格形式表示的语法

7.3.1 NAL单元语法

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

7.3.2 原始字节序列载荷和RBSP尾比特语法

7.3.2.1 序列参数集RBSP 语法

seq_parameter_set_rbsp( ) {	C	描述符
profile_idc	0	u(8)
constraint_set0_flag	0	u(1)
constraint_set1_flag	0	u(1)
constraint_set2_flag	0	u(1)
constraint_set3_flag	0	u(1)
reserved_zero_4bits /* equal to 0 */	0	u(4)
level_idc	0	u(8)
seq_parameter_set_id	0	ue(v)
if(profile_idc == 100 || profile_idc == 110 || profile_idc == 122 || profile_idc == 144) {		
chroma_format_idc	0	ue(v)
if(chroma_format_idc == 3)		
residual_colour_transform_flag	0	u(1)
bit_depth_luma_minus8	0	ue(v)
bit_depth_chroma_minus8	0	ue(v)
qpprime_y_zero_transform_bypass_flag	0	u(1)
seq_scaling_matrix_present_flag	0	u(1)
if(seq_scaling_matrix_present_flag)		
for(i = 0; i < 8; i++) {		
seq_scaling_list_present_flag[i **]**	0	u(1)
if(seq_scaling_list_present_flag[i])		
if(i < 6)		
scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	0	
else		
scaling_list(ScalingList8x8[i – 6], 64, UseDefaultScalingMatrix8x8Flag[i – 6])	0	
}		
}		
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	
}		

7.3.2.1.1 缩放比例列表语法

scaling_list(scalingList, sizeOfScalingList, useDefaultScalingMatrixFlag) {	**C**	**描述符**
lastScale = 8		
nextScale = 8		
for(j = 0; j < sizeOfScalingList; j++) {		
if(nextScale != 0) {		
delta_scale	0 | 1	se(v)
nextScale = (lastScale + delta_scale + 256) % 256		
useDefaultScalingMatrixFlag = (j == 0 && nextScale == 0)		
}		
scalingList[j] = (nextScale == 0) ? lastScale : nextScale		
lastScale = scalingList[j]		
}		
}		

7.3.2.1.2 序列参数集扩展RBSP语法

seq_parameter_set_extension_rbsp( ) {	**C**	**描述符**
seq_parameter_set_id	10	ue(v)
aux_format_idc	10	ue(v)
if(aux_format_idc != 0) {		
bit_depth_aux_minus8	10	ue(v)
alpha_incr_flag	10	u(1)
alpha_opaque_value	10	u(v)
alpha_transparent_value	10	u(v)
}		
additional_extension_flag	10	u(1)
rbsp_trailing_bits()	10	
}		

7.3.2.2 图像参数集RBSP语法

pic_parameter_set_rbsp( ) {	C	描述符
pic_parameter_set_id	1	ue(v)
seq_parameter_set_id	1	ue(v)
entropy_coding_mode_flag	1	u(1)
pic_order_present_flag	1	u(1)
num_slice_groups_minus1	1	ue(v)
if(num_slice_groups_minus1 > 0) {		
slice_group_map_type	1	ue(v)
if(slice_group_map_type = = 0)		
for(iGroup = 0; iGroup <= num_slice_groups_minus1; iGroup++)		
run_length_minus1[iGroup **]**	1	ue(v)
else if(slice_group_map_type = = 2)		
for(iGroup = 0; iGroup < num_slice_groups_minus1; iGroup++) {		
top_left[iGroup **]**	1	ue(v)
bottom_right[iGroup **]**	1	ue(v)
}		
else if(slice_group_map_type = = 3 || slice_group_map_type = = 4 || slice_group_map_type = = 5) {		
slice_group_change_direction_flag	1	u(1)
slice_group_change_rate_minus1	1	ue(v)
} else if(slice_group_map_type = = 6) {		
pic_size_in_map_units_minus1	1	ue(v)
for(i = 0; i <= pic_size_in_map_units_minus1; i++)		
slice_group_id[i **]**	1	u(v)
}		
}		
num_ref_idx_l0_active_minus1	1	ue(v)
num_ref_idx_l1_active_minus1	1	ue(v)
weighted_pred_flag	1	u(1)
weighted_bipred_idc	1	u(2)
pic_init_qp_minus26 /* relative to 26 */	1	se(v)
pic_init_qs_minus26 /* relative to 26 */	1	se(v)
chroma_qp_index_offset	1	se(v)
deblocking_filter_control_present_flag	1	u(1)
constrained_intra_pred_flag	1	u(1)
redundant_pic_cnt_present_flag	1	u(1)
if(more_rbsp_data( )) {		
transform_8x8_mode_flag	1	u(1)
pic_scaling_matrix_present_flag	1	u(1)
if(pic_scaling_matrix_present_flag)		
for(i = 0; i < 6 + 2* transform_8x8_mode_flag; i++) {		
pic_scaling_list_present_flag[i **]**	1	u(1)
if(pic_scaling_list_present_flag[i])		
if(i < 6)		
scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	1	
else		

scaling_list(ScalingList8x8[i – 6], 64, UseDefaultScalingMatrix8x8Flag[i – 6])	1	
}		
second_chroma_qp_index_offset	1	se(v)
}		
rbsp_trailing_bits( )	1	
}		

7.3.2.3 辅助增强信息RBSP语法

sei_rbsp( ) {	C	描述符
do		
sei_message( )	5	
while(more_rbsp_data( ))		
rbsp_trailing_bits( )	5	
}		

7.3.2.3.1 辅助增强信息消息语法

sei_message( ) {	C	描述符
payloadType = 0		
while(next_bits(8) = = 0xFF) {		
ff_byte /* equal to 0xFF */	5	f(8)
payloadType += 255		
}		
last_payload_type_byte	5	u(8)
payloadType += last_payload_type_byte		
payloadSize = 0		
while(next_bits(8) = = 0xFF) {		
ff_byte /* equal to 0xFF */	5	f(8)
payloadSize += 255		
}		
last_payload_size_byte	5	u(8)
payloadSize += last_payload_size_byte		
sei_payload(payloadType, payloadSize)	5	
}		

7.3.2.4 访问单元分隔符RBSP语法

access_unit_delimiter_rbsp( ) {	C	描述符
primary_pic_type	6	u(3)
rbsp_trailing_bits( )	6	
}		

7.3.2.5 序列结尾RBSP语法

end_of_seq_rbsp( ) {	C	描述符
}		

7.3.2.6 流结尾RBSP语法

end_of_stream_rbsp( ) {	C	描述符
}		

7.3.2.7 填充数据RBSP语法

filler_data_rbsp( ) {	C	描述符
while(next_bits(8) = = 0xFF)		
ff_byte /* equal to 0xFF */	9	f(8)
rbsp_trailing_bits( )	9	
}		

7.3.2.8 没有分割的条带层RBSP语法

slice_layer_without_partitioning_rbsp( ) {	C	描述符
slice_header( )	2	
slice_data( ) /* all categories of slice_data( ) syntax */	2 | 3 | 4	
rbsp_slice_trailing_bits( )	2	
}		

7.3.2.9 条带数据分割RBSP语法

7.3.2.9.1 条带数据分割块ARBSP语法

slice_data_partition_a_layer_rbsp( ) {	C	描述符
slice_header( )	2	
slice_id	全部	ue(v)
slice_data( ) /* only category 2 parts of slice_data( ) syntax */	2	
rbsp_slice_trailing_bits( )	2	
}		

7.3.2.9.2 条带数据分割块BRBSP语法

slice_data_partition_b_layer_rbsp( ) {	C	描述符
slice_id	全部	ue(v)
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	全部	ue(v)
slice_data( ) /* only category 3 parts of slice_data( ) syntax */	3	
rbsp_slice_trailing_bits( )	3	
}		

7.3.2.9.3 条带数据分割块CRBSP语法

slice_data_partition_c_layer_rbsp( ) {	C	描述符
slice_id	全部	ue(v)
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	全部	ue(v)
slice_data( ) /* only category 4 parts of slice_data( ) syntax */	4	
rbsp_slice_trailing_bits( )	4	
}		

7.3.2.10 条带尾比特RBSP语法

rbsp_slice_trailing_bits( ) {	C	描述符
rbsp_trailing_bits( )	全部	
if(entropy_coding_mode_flag)		
while(more_rbsp_trailing_data( ))		
cabac_zero_word /* equal to 0x0000 */	全部	f(16)
}		

7.3.2.11 尾比特RBSP语法

rbsp_trailing_bits( ) {	C	描述符
rbsp_stop_one_bit /* equal to 1 */	全部	f(1)
while(!byte_aligned( ))		
rbsp_alignment_zero_bit /* equal to 0 */	全部	f(1)
}		

7.3.3 条带头语法

slice_header( ) {	C	描述符
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)
if(!frame_mbs_only_flag) {		
field_pic_flag	2	u(1)
if(field_pic_flag)		
bottom_field_flag	2	u(1)
}		
if(nal_unit_type == 5)		
idr_pic_id	2	ue(v)
if(pic_order_cnt_type == 0) {		
pic_order_cnt_lsb	2	u(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt_bottom	2	se(v)
}		
if(pic_order_cnt_type = = 1 && !delta_pic_order_always_zero_flag) {		
delta_pic_order_cnt[0 **]**	2	se(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt[1 **]**	2	se(v)
}		
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	2	ue(v)
if(slice_type == B)		
direct_spatial_mv_pred_flag	2	u(1)
if(slice_type = = P | | slice_type = = SP | | slice_type = = B) {		
num_ref_idx_active_override_flag	2	u(1)
if(num_ref_idx_active_override_flag) {		
num_ref_idx_l0_active_minus1	2	ue(v)
if(slice_type == B)		
num_ref_idx_l1_active_minus1	2	ue(v)
}		
}		
ref_pic_list_reordering( )	2	
if((weighted_pred_flag && (slice_type = = P | | slice_type = = SP)) | | (weighted_bipred_idc == 1 && slice_type == B))		
pred_weight_table( )	2	
if(nal_ref_idc != 0)		
dec_ref_pic_marking( )	2	
if(entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)
if(slice_type == SP | | slice_type == SI) {		
if(slice_type == SP)		
sp_for_switch_flag	2	u(1)

slice_qs_delta	2	se(v)
}		
if(deblocking_filter_control_present_flag) {		
disable_deblocking_filter_idc	2	ue(v)
if(disable_deblocking_filter_idc != 1) {		
slice_alpha_c0_offset_div2	2	se(v)
slice_beta_offset_div2	2	se(v)
}		
}		
if(num_slice_groups_minus1 > 0 && slice_group_map_type >= 3 && slice_group_map_type <= 5)		
slice_group_change_cycle	2	u(v)
}		

7.3.3.1 参考图像列表重排序语法

ref_pic_list_reordering( ) {	C	描述符
if(slice_type != I && slice_type != SI) {		
ref_pic_list_reordering_flag_l0	2	u(1)
if(ref_pic_list_reordering_flag_l0)		
do {		
reordering_of_pic_nums_idc	2	ue(v)
if(reordering_of_pic_nums_idc == 0 || reordering_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
else if(reordering_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
} while(reordering_of_pic_nums_idc != 3)		
}		
if(slice_type == B) {		
ref_pic_list_reordering_flag_l1	2	u(1)
if(ref_pic_list_reordering_flag_l1)		
do {		
reordering_of_pic_nums_idc	2	ue(v)
if(reordering_of_pic_nums_idc == 0 || reordering_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
else if(reordering_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
} while(reordering_of_pic_nums_idc != 3)		
}		
}		

7.3.3.2 预测加权表格语法

pred_weight_table( ) {	C	描述符
luma_log2_weight_denom	2	ue(v)
if(chroma_format_idc != 0)		
chroma_log2_weight_denom	2	ue(v)
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++) {		
luma_weight_l0_flag	2	u(1)
if(luma_weight_l0_flag) {		
luma_weight_l0[i **]**	2	se(v)
luma_offset_l0[i **]**	2	se(v)
}		
if (chroma_format_idc != 0) {		
chroma_weight_l0_flag	2	u(1)
if(chroma_weight_l0_flag)		
for(j =0; j < 2; j++) {		
chroma_weight_l0[i **][** j **]**	2	se(v)
chroma_offset_l0[i **][** j **]**	2	se(v)
}		
}		
}		
if(slice_type = = B)		
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++) {		
luma_weight_l1_flag	2	u(1)
if(luma_weight_l1_flag) {		
luma_weight_l1[i **]**	2	se(v)
luma_offset_l1[i **]**	2	se(v)
}		
if(chroma_format_idc != 0) {		
chroma_weight_l1_flag	2	u(1)
if(chroma_weight_l1_flag)		
for(j = 0; j < 2; j++) {		
chroma_weight_l1[i **][** j **]**	2	se(v)
chroma_offset_l1[i **][** j **]**	2	se(v)
}		
}		
}		
}		

7.3.3.3 解码的参考图像标识语法

dec_ref_pic_marking( ) {	C	描述符
if(nal_unit_type == 5) {		
no_output_of_prior_pics_flag	2 | 5	u(1)
long_term_reference_flag	2 | 5	u(1)
} else {		
adaptive_ref_pic_marking_mode_flag	2 | 5	u(1)
if(adaptive_ref_pic_marking_mode_flag)		
do {		
memory_management_control_operation	2 | 5	ue(v)
if(memory_management_control_operation == 1 || memory_management_control_operation == 3)		
difference_of_pic_nums_minus1	2 | 5	ue(v)
if(memory_management_control_operation == 2)		
long_term_pic_num	2 | 5	ue(v)
if(memory_management_control_operation == 3 || memory_management_control_operation == 6)		
long_term_frame_idx	2 | 5	ue(v)
if(memory_management_control_operation == 4)		
max_long_term_frame_idx_plus1	2 | 5	ue(v)
} while(memory_management_control_operation != 0)		
}		
}		

7.3.4 条带数据语法

slice_data( ) {	C	描述符
if(entropy_coding_mode_flag)		
while(!byte_aligned( ))		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		
moreDataFlag = 1		
prevMbSkipped = 0		
do {		
if(slice_type != I && slice_type != SI)		
if(!entropy_coding_mode_flag) {		
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
for(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data( )		
} else {		
mb_skip_flag	2	ae(v)
moreDataFlag = !mb_skip_flag		
}		
if(moreDataFlag) {		
if(MbaffFrameFlag && (CurrMbAddr % 2 == 0 || (CurrMbAddr % 2 == 1 && prevMbSkipped)))		
mb_field_decoding_flag	2	u(1) | ae(v)
macroblock_layer( )	2 | 3 | 4	
}		
if(!entropy_coding_mode_flag)		
moreDataFlag = more_rbsp_data( )		
else {		
if(slice_type != I && slice_type != SI)		
prevMbSkipped = mb_skip_flag		
if(MbaffFrameFlag && CurrMbAddr % 2 == 0)		
moreDataFlag = 1		
else {		
end_of_slice_flag	2	ae(v)
moreDataFlag = !end_of_slice_flag		
}		
}		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
} while(moreDataFlag)		
}		

7.3.5 宏块层语法

macroblock_layer( ) {	C	描述符
mb_type	2	ue(v) | ae(v)
if(mb_type == I_PCM) {		
while(!byte_aligned( ))		
pcm_alignment_zero_bit	2	f(1)
for(i = 0; i < 256; i++)		
pcm_sample_luma[i **]**	2	u(v)
for(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i **]**	2	u(v)
} else {		
noSubMbPartSizeLessThan8x8Flag = 1		
if(mb_type != I_NxN && MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) == 4) {		
sub_mb_pred(mb_type)	2	
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8) {		
if(NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else if(!direct_8x8_inference_flag)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else {		
if(transform_8x8_mode_flag && mb_type == I_NxN)		
transform_size_8x8_flag	2	u(1) | ae(v)
mb_pred(mb_type)	2	
}		
if(MbPartPredMode(mb_type, 0) != Intra_16x16) {		
coded_block_pattern	2	me(v) | ae(v)
if(CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && (mb_type != B_Direct_16x16 || direct_8x8_inference_flag))		
transform_size_8x8_flag	2	u(1) | ae(v)
}		
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_type, 0) == Intra_16x16) {		
mb_qp_delta	2	se(v) | ae(v)
residual( )	3 | 4	
}		
}		
}		

7.3.5.1 宏块预测语法

mb_pred(mb_type) {	C	描述符
if(MbPartPredMode(mb_type, 0) == Intra_4x4 || MbPartPredMode(mb_type, 0) == Intra_8x8 || MbPartPredMode(mb_type, 0) == Intra_16x16) {		
if(MbPartPredMode(mb_type, 0) == Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
if(MbPartPredMode(mb_type, 0) == Intra_8x8)		
for(luma8x8BlkIdx=0; luma8x8BlkIdx<4; luma8x8BlkIdx++) {		
prev_intra8x8_pred_mode_flag[luma8x8BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra8x8_pred_mode_flag[luma8x8BlkIdx])		
rem_intra8x8_pred_mode[luma8x8BlkIdx **]**	2	u(3) | ae(v)
}		
if(chroma_format_idc != 0)		
intra_chroma_pred_mode	2	ue(v) | ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(MbPartPredMode (mb_type, mbPartIdx) != Pred_L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
}		
}		

7.3.5.2 子宏块预测语法

sub_mb_pred(mb_type) {	C	描述符
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
sub_mb_type[mbPartIdx **]**	2	ue(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
}		

7.3.5.3 残差数据语法

Residual( ) {	C	描述符
if(!entropy_coding_mode_flag)		
residual_block = residual_block_cavlc		
else		
residual_block = residual_block_cabac		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(Intra16x16DCLevel, 16)	3	
for(i8x8 = 0; i8x8 < 4; i8x8++) /* each luma 8x8 block */		
if(!transform_size_8x8_flag || !entropy_coding_mode_flag)		
for(i4x4 = 0; i4x4 < 4; i4x4++) { /* each 4x4 sub-block of block */		
if(CodedBlockPatternLuma & (1 << i8x8))		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(Intra16x16ACLevel[i8x8 * 4 + i4x4], 15)	3	
else		
residual_block(LumaLevel[i8x8 * 4 + i4x4], 16)	3 | 4	
else if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
for(i = 0; i < 15; i++)		
Intra16x16ACLevel[i8x8 * 4 + i4x4][i] = 0		
else		
for(i = 0; i < 16; i++)		
LumaLevel[i8x8 * 4 + i4x4][i] = 0		
if(!entropy_coding_mode_flag && transform_size_8x8_flag)		
for(i = 0; i < 16; i++)		
LumaLevel8x8[i8x8][4 * i + i4x4] = LumaLevel[i8x8 * 4 + i4x4][i]		
}		
else if(CodedBlockPatternLuma & (1 << i8x8))		
residual_block(LumaLevel8x8[i8x8], 64)	3 | 4	
else		
for(i = 0; i < 64; i++)		
LumaLevel8x8[i8x8][i] = 0		
if(chroma_format_idc != 0) {		
NumC8x8 = 4 / (SubWidthC * SubHeightC)		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
if(CodedBlockPatternChroma & 3) /* chroma DC residual present */		
residual_block(ChromaDCLevel[iCbCr], 4 * NumC8x8)	3 | 4	
else		
for(i = 0; i < 4 * NumC8x8; i++)		
ChromaDCLevel[iCbCr][i] = 0		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
for(i8x8 = 0; i8x8 < NumC8x8; i8x8++)		
for(i4x4 = 0; i4x4 < 4; i4x4++)		
if(CodedBlockPatternChroma & 2) /* chroma AC residual present */		
residual_block(ChromaACLevel[iCbCr][i8x8*4+i4x4], 15)	3 | 4	
else		
for(i = 0; i < 15; i++)		
ChromaACLevel[iCbCr][i8x8*4+i4x4][i] = 0		
}		

7.3.5.3.1 残差块CAVLC 语法

residual_block_cavlc(coeffLevel, maxNumCoeff) {	C	描述符
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
coeff_token	3 | 4	ce(v)
if(TotalCoeff(coeff_token) > 0) {		
if(TotalCoeff(coeff_token) > 10 && TrailingOnes(coeff_token) < 3)		
suffixLength = 1		
else		
suffixLength = 0		
for(i = 0; i < TotalCoeff(coeff_token); i++)		
if(i < TrailingOnes(coeff_token)) {		
trailing_ones_sign_flag	3 | 4	u(1)
level[i] = 1 – 2 * trailing_ones_sign_flag		
} else {		
level_prefix	3 | 4	ce(v)
levelCode = (Min(15, level_prefix) << suffixLength)		
if(suffixLength > 0 || level_prefix >= 14) {		
level_suffix	3 | 4	u(v)
levelCode += level_suffix		
}		
if(level_prefix >= 15 && suffixLength == 0)		
levelCode += 15		
if(level_prefix >= 16)		
levelCode += (1 << (level_prefix – 3)) – 4096		
if(i == TrailingOnes(coeff_token) && TrailingOnes(coeff_token) < 3)		
levelCode += 2		
if(levelCode % 2 == 0)		
level[i] = (levelCode + 2) >> 1		
else		
level[i] = (–levelCode – 1) >> 1		
if(suffixLength == 0)		
suffixLength = 1		
if(Abs(level[i]) > (3 << (suffixLength – 1)) && suffixLength < 6)		
suffixLength++		
}		
if(TotalCoeff(coeff_token) < maxNumCoeff) {		
total_zeros	3 | 4	ce(v)
zerosLeft = total_zeros		
} else		
zerosLeft = 0		
for(i = 0; i < TotalCoeff(coeff_token) – 1; i++) {		
if(zerosLeft > 0) {		
run_before	3 | 4	ce(v)

run[i] = run_before		
} else		
run[i] = 0		
zerosLeft = zerosLeft – run[i]		
}		
run[TotalCoeff(coeff_token) – 1] = zerosLeft		
coeffNum = -1		
for(i = TotalCoeff(coeff_token) – 1; i >= 0; i-－) {		
coeffNum += run[i] + 1		
coeffLevel[coeffNum] = level[i]		
}		
}		
}		

7.3.5.3.2 残差块CABAC语法

residual_block_cabac(coeffLevel, maxNumCoeff) {	C	描述符
if(maxNumCoeff = = 64)		
coded_block_flag = 1		
else		
coded_block_flag	3 | 4	ae(v)
if(coded_block_flag) {		
numCoeff = maxNumCoeff		
i = 0		
do {		
significant_coeff_flag[i **]**	3 | 4	ae(v)
if(significant_coeff_flag[i]) {		
last_significant_coeff_flag[i **]**	3 | 4	ae(v)
if(last_significant_coeff_flag[i]) {		
numCoeff = i + 1		
for(j = numCoeff; j < maxNumCoeff; j++)		
coeffLevel[j] = 0		
}		
}		
i++		
} while(i < numCoeff －1)		
coeff_abs_level_minus1[numCoeff - 1 **]**	3 | 4	ae(v)
coeff_sign_flag[numCoeff - 1 **]**	3 | 4	ae(v)
coeffLevel[numCoeff - 1] = (coeff_abs_level_minus1[numCoeff – 1] + 1) * (1 – 2 * coeff_sign_flag[numCoeff – 1])		
for(i = numCoeff - 2; i >= 0; i-－)		
if(significant_coeff_flag[i]) {		
coeff_abs_level_minus1[i **]**	3 | 4	ae(v)
coeff_sign_flag[i **]**	3 | 4	ae(v)
coeffLevel[i] = (coeff_abs_level_minus1[i] + 1) * (1 – 2 * coeff_sign_flag[i])		
} else		
coeffLevel[i] = 0		
} else		
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
}		

7.4 语义

本节规定了与语法结构和语法结构中的语法元素相关的语义。当一个语法元素的语义用一个或一组表格表示时，表格中未指定的任何值都不应出现在比特流中，除非在本建议书 | 国际标准中另外规定。

7.4.1 NAL单元语义

注 1 — 规定VCL是为了有效的表示视频数据的内容。规定NAL则是为了格式化数据，并以适用于存储介质或在多种通信信道上传输的格式提供头信息。NAL单元中包含了所有的数据，每个NAL单元都包含整数字节。NAL单元规定一种既适用于面向分组系统又适用于比特流系统的通用格式。用于分组传输和字节流的NAL单元的格式是一样的，不过字节流格式中的每个NAL单元前可以有一个起始码前缀和额外填充字节。

NumBytesInNALunit 规定了 NAL 单元的大小，单位是字节。在 NAL 单元解码时需要用到该值。为了能够推导出 NumBytesInNALunit，需要对 NAL 单元的边界进行划分。附件 B 规定了一种用于字节流格式的划分方法。其他划分方法可能会在本建议书 | 国际标准之外给出。

forbidden_zero_bit 应为 0。

nal_ref_idc 不等于 0 时，规定 NAL 单元的内容包含一个序列参数集，或一个图像参数集，或一个参考图像条带，或一个参考图像的条带数据分割。

如果一个包含一个条带或条带数据分割的 NAL 单元的 nal_ref_idc 等于 0 时，该条带或条带数据分割是一个非参考图像的一部分。

对于序列参数集或序列参数集扩展或图像参数集的 NAL 单元，nal_ref_idc 不应等于 0。当一个特定的图像的一个条带或条带数据分割 NAL 单元的 nal_ref_idc 等于 0，该图像的所有条带或条带数据划分 NAL 单元都应该等于 0。

IDR NAL 单元的 nal_ref_idc 不应等于 0，即 nal_unit_type 等于 5 的 NAL 单元。

所有 nal_unit_type 等于 6、9、10、11 或 12 的 NAL 单元其 nal_ref_idc 都应等于 0。

nal_unit_type 是指包含在 NAL 单元中的 RBSP 数据结构的类型，如表 7-1 所示。VCL NAL 单元是指那些 nal_unit_type 值等于 1 到 5（包括 1 和 5）的 NAL 单元。所有其他的 NAL 单元都称作非 VCL NAL 单元。

表 7-1 中标记为“C”的列列出可能出现在 NAL 单元中的语法元素的种类。另外，可能会出现语法种类是“All”的语法元素，这由 RBSP 数据结构的语法和语义决定。特别列出的语法种类的任何语法元素的出现与否由相关 RBSP 数据结构的语法和语义决定。nal_unit_type 的值不应等于 3 或 4，除非在 RBSP 数据结构中的至少有一个语法元素的语法元素种类值等于 nal_unit_type 的值而不是“All”。

表 7-1－NAL单元类型码

nal_unit_type	NAL 单元和 RBSP 语法结构的内容	C
0	未指定	
1	一个非 IDR 图像的编码条带 slice_layer_without_partitioning_rbsp( )	2, 3, 4
2	编码条带数据分割块 A slice_data_partition_a_layer_rbsp( )	2
3	编码条带数据分割块 B slice_data_partition_b_layer_rbsp( )	3
4	编码条带数据分割块 C slice_data_partition_c_layer_rbsp( )	4
5	IDR 图像的编码条带 slice_layer_without_partitioning_rbsp( )	2, 3
6	辅助增强信息 (SEI) sei_rbsp( )	5
7	序列参数集 seq_parameter_set_rbsp( )	0
8	图像参数集 pic_parameter_set_rbsp( )	1
9	访问单元分隔符 access_unit_delimiter_rbsp( )	6
10	序列结尾 end_of_seq_rbsp( )	7
11	流结尾 end_of_stream_rbsp( )	8
12	填充数据 filler_data_rbsp( )	9
13	序列参数集扩展 seq_parameter_set_extension_rbsp( )	10
14..18	保留	
19	未分割的辅助编码图像的编码条带 slice_layer_without_partitioning_rbsp( )	2, 3, 4
20..23	保留	
24..31	未指定	

在不影响 nal_unit_type 不等于 13 或 19 的 NAL 单元的解码过程和不影响本建议书 | 国际标准的一致性前提下，nal_unit_type 等于 13 和 19 的 NAL 单元可以被解码器丢弃。

采用 nal_unit_type 值等于 0 或在 24-31（包括 24 和 31）范围内的 NAL 单元不应影响本建议书 | 国际标准所规定的解码过程。

注 2 — 可以根据应用来使用NAL单元类型0和24-31，本建议书 | 国际标准不规定这些nal_unit_type值的解码过程。

解码器应忽略（从比特流中去除并丢弃）所有使用 nal_unit_type 为保留值的 NAL 单元的内容。

注 3 — 这允许未来对本建议书 | 国际标准做兼容性扩展的定义。

本文中编码的条带 NAL 单元全部指一个非 IDR 图像 NAL 单元的编码条带或一个 IDR 图像 NAL 单元的编码条带。

当一个编码图像条带 NAL 单元的 nal_unit_type 值等于 5 时，编码的同一图像其他所有的 VCL NAL 单元的 nal_unit_type 值都应该为 5。这样的图像称作 IDR 图像。

注 4 — 条带数据分割不可以用于IDR图像。

rbsp_byte[i **]** 是一个 RBSP 的第 i 个字节。一个 RBSP 定义为一个字节的有序序列，如下所示。

RBSP 包含一个 SODB，如下：

— 如果 SODB 为空（例如：长度是 0 比特），RBSP 也为空。

— 否则 RBSP 包括如下 SODB。

1）RBSP 的第一个字节包括（最高位的，最左边的）8 比特的 SODB；RBSP 的下一个字节应包括接下来的 8 比特的 SODB，等等，直到剩下的 SODB 少于 8 比特。

2）rbsp_trailing_bits( )用于 SODB 之后，如下：

i）最后 RBSP 字节的第 1 个的（最高位的，最左边的）比特包括 SODB 的剩下的比特（如果有的话）

ii）下一个比特为单个 rbsp_stop_one_bit，其值为 1，并且

iii）当 rbsp_stop_one_bit 不是一个字节对齐的字节的最后一个比特时，一个或更多的 rbsp_alignment_zero_bit 就会出现以形成一个字节对齐。

3）在某些 RBSP 结尾的 rbsp_trailing_bits( )之后，可以出现值为 0x0000 的一个或多个 16 比特的语法元素 cabac_zero_word。

具有这些 RBSP 属性的语法结构在语法表中用“_rbsp”前缀表示。这些结构在 NAL 单元中作为 rbsp_byte[i]数据字节的内容携带。NAL 单元到 RBSP 语法结构的关联如表 7-1 所规定。

注 5 — 当RBSP的边界已知时，解码器能够通过连接RBSP字节的比特并丢弃最后（最不重要的，最右边的）一个等于1的比特rbsp_stop_one_bit以及随后（次重要的，更靠近右边的）的任何等于0的比特从RBSP中解析出SODB。解码过程所必须的数据包含在RBSP的SODB部分。

emulation_prevention_three_byte 是一个等于 0x03 的字节。当一个 emulation_prevention_three_byte 出现在 NAL 单元中时，应该被解码过程丢弃。

NAL 单元的最后一个字节不能等于 0x00。

在 NAL 单元中，下面的三字节序列不应在任何字节对齐的位置出现：

— 0x000000

— 0x000001

— 0x000002

在一个 NAL 单元中，除了下列序列，任何以 0x000003 开头的四字节的序列都不能出现在任何字节对齐的位置：

— 0x00000300

— 0x00000301

— 0x00000302

— 0x00000303

注 6 — 当nal_unit_type等于0时，在设计编码器时必须要特别的小心以避免上面列出的三字节和四字节形式出现在NAL单元语法结构的开头，以免emulation_prevention_three_byte语法元素成为NAL单元的第三字节。

7.4.1.1 将一个SODB封装到RBSP中（资料性）

本节不是本建议书 | 国际标准的组成部分。

将一个 SODB 封装到一个 RBSP 中形式和使用 emulation_prevention_three_byte 将一个 RBSP 封装到一个 NAL 单元中的规定是为了以下的目的：

— 允许NAL单元中出现任意的SODB，但要防止在NAL单元中出现伪起始码。

— 通过在RBSP的结尾查找其比特rbsp_stop_one_bit，以便能够识别NAL单元中SODB的结尾，并且

— 使NAL单元的大小能够比某些情况（使用一个或多个cabac_zero_word）下的SODB的大小要大。

编码器通过下列步骤能够从一个 RBSP 中产生一个 NAL 单元：

从 RBSP 数据中查找字节对齐的下面二进制比特图案：

'00000000 00000000 000000xx' （其中 xx 代表任意2比特图案： 00、01、10或 11），

并且使用其中插入一个等于 0x03 的字节的图案来代替这些比特图案：

'00000000 00000000 00000011 000000xx',

最后，当 RBSP 数据的最后一个字节等于 0x00（只有在 RBSP 以 cabac_zero_word 结尾时才会出现），在数据的末尾添加一个等于 0x03 的字节。

得到的字节序列加上包含标识 RBSP 数据结构类型的 NAL 单元的首字节就形成了整个 NAL 单元。

该过程允许任何 SODB 在一个 NAL 单元中出现，同时可以确保：

— 该NAL单元中没有字节对齐的伪起始码，

— 无论是否字节对齐，在NAL单元中没有8个值为0的比特后跟随起始码的序列。

7.4.1.2 NAL单元的顺序及其与编码图像、访问单元和视频序列的关系

本节规定了对比特流中 NAL 单元顺序的要求。在本文中遵从该要求的比特流中任何 NAL 单元的顺序都是指 NAL 单元的解码顺序。在一个 NAL 单元中，7.3 节的语法，D.1 和 E.1 是指语法元素的解码顺序。符合本建议书 | 国际标准的解码器应能够按解码顺序接收 NAL 单元和它们的语法元素。

7.4.1.2.1 序列、图像参数集RBSP顺序及其激活

注 1 — 序列和图像参数集机制解除了传送编码的宏块数据与传送不经常变化的信息之间的耦合。在一些应用中，序列和图像参数集可以使用一种可靠的带外机制进行传送。

图像参数集 RBSP 包括一些参数，这些参数可以被一个或多个编码图像的编码条带 NAL 单元或编码条带数据分割块 A 的 NAL 单元使用。在解码过程操作的开始，每个图像参数集 RBSP 认为是未激活的。至多认为一个图像参数集 RBSP 在解码过程操作期间的指定时刻是激活的，并且任何特定图像参数集 RBSP 的激活会导致先前激活的图像参数集 RBSP（如果有的话）去激活。

当一个图像参数集 RBSP（具有特定的 pic_parameter_set_id 值）是未激活的，并且被编码条带 NAL 单元或编码条带数据分割块 A 的 NAL 单元使用时（使用值 pic_parameter_set_id），则该图像参数集 RBSP 激活。此图像参数集 RBSP 称作活动图像参数集 RBSP，直到由于另一个图像参数集 RBSP 的激活而去激活。一个具有特定 pic_parameter_set_id 值的图像参数集 RBSP 在激活之前对解码过程来说就应是可用的。

任何用于活动图像参数集 RBSP 的含有 pic_parameter_set_id 值的图像参数集 NAL 单元应与该活动图像参数集 RBSP 具有相同的内容，除非它是位于一个编码图像的最后一个 VCL NAL 单元之后，另一个编码图像的第一个 VCL NAL 单元之前。

一个序列参数集 RBSP 包括可能被一个或多个图像参数集 RBSP 或者一个或多个包含一个缓冲周期 SEI 消息的 SEI NAL 单元引用的参数。每个序列参数集 RBSP 在解码过程操作的开始认为是未激活的。至多都认为一个序列参数集 RBSP 在解码过程操作期间的指定时刻是激活的，并且任何特定的序列参数集 RBSP 的激活会导致先前激活的序列参数集 RBSP（如果有的话）去激活。

当一个序列参数集 RBSP（具有一个 seq_parameter_set_id 的特定值）并不是活动的而且由一个图像参数集 RBSP（使用该 seq_parameter_set_id 值）激活或由一个包括一个缓冲周期 SEI 消息（使用该 seq_parameter_set_id 值）的 SEI NAL 单元指示其激活时，该序列参数集 RBSP 激活。该序列参数集 RBSP 称作活动序列参数集 RBSP，直到由于另一个序列参数集 RBSP 的激活而去激活。一个具有特定 seq_parameter_set_id 值的序列参数集 RBSP 在激活之前对解码过程来说应就是可用的。一个活动的序列参数集 RBSP 在整个编码视频序列中保持激活。

注 2 — 由于一个IDR访问单元开始了一个新的编码视频序列并且一个激活的序列参数集RBSP在整个编码视频序列中必须保持活动状态，当缓冲周期SEI消息是IDR访问单元的一部分时，一个序列参数集RBSP可以仅由该缓冲周期SEI消息激活。

任何用于活动序列参数集 RBSP 的含有 seq_parameter_set_id 值的序列参数集 NAL 单元应与该活动序列参数集 RBSP 具有相同的内容，除非它是位于一个编码视频序列的最后一个访问单元之后，另一个编码视频序列的第一个 VCL NAL 单元和第一个包含一个缓冲周期 SEI 消息（当出现时）的 SEI NAL 单元之前。

注 3 — 如果图像参数集RBSP或序列参数集RBSP在比特流中传送，这些规定分别对包含图像参数集RBSP或序列参数集RBSP的NAL单元强加了一个顺序约束。否则（图像参数集RBSP或序列参数集RBSP由其他本建议书 | 国际标准未指定的方式传送），它们在解码的过程中必须适时的可用以保证这些规定被遵从。

当一个序列参数集扩展 RBSP 存在时，它包括具有与那些序列参数集 RBSP 类似功能的参数。为了为序列参数集扩展 RBSP 的语法元素建立规则以及为决定一个序列参数集扩展 RBSP 的激活，该序列参数集扩展 RBSP 应当作前面的具有同样 seq_parameter_set_id 值的序列参数集 RBSP 的一部分。当一个序列参数集 RBSP 存在，并且其后面没有跟随一个在序列参数集 RBSP 激活之前且具有同样 seq_parameter_set_id 值的序列参数集扩展 RBSP，对于活动序列参数集 RBSP 来说，该序列参数集扩展 RBSP 及其语法元素应被当作不存在。

所有针对序列参数集和图像参数集的语法元素的值与其他语法元素之间的关系作出的规定是仅仅适用于活动序列参数集和活动图像参数集的规定。如果任何序列参数集 RBSP 出现并且在比特流中没有激活，而是在其他的类似比特流中引用而被激活，该类序列参数集 RBSP 的语法元素应具有与指定的要求一样的值。如果任何图像参数集 RBSP 出现并从未在比特流中激活，而是在其他的类似比特流中引用而被激活，该类图像参数集 RBSP 的语法元素应具有与指定的要求一样的值。

在解码过程中（参见第 8 节），活动图像参数集和活动序列参数集的参数值应保持有效。为解释 SEI 消息，在同一个访问单元的基本编码图像的 VCL NAL 单元的解码过程中激活的图像参数集和序列参数集的参数值应当作有效的，除非在 SEI 消息语义中另外指定。

7.4.1.2.2 访问单元的顺序及其与编码视频序列的关系

遵从本建议书 | 国际标准的比特流由一个或多个编码视频序列组成。

编码视频序列由一个或多个访问单元组成。NAL 单元和编码图像的顺序以及它们与访问单元的关系在7.4.1.2.3节描述。

每个编码视频序列的第一个访问单元是 IDR 访问单元。该编码视频序列中所有后续的访问单元都是非 IDR 访问单元。

按解码顺序的连续访问单元的包含非参考图像的编码图像的图像顺序数不应递减。

出现在一个包含视频序列 NAL 单元结束符的访问单元后面的访问单元应该是一个 IDR 访问单元。

当一个 SEI NAL 单元包含属于多个访问单元（例如：当 SEI NAL 单元把编码视频序列作为其范围时）数据时，它应被包含到第一个适用的访问单元中。

当一个访问单元中存在一个流 NAL 单元的结尾时，该访问单元应该是该比特流的最后一个访问单元，并且流 NAL 单元的结尾应作为该访问单元中最后一个 NAL 单元。

7.4.1.2.3 NAL单元和编码图像的顺序及其与访问单元的关系

一个访问单元由一个基本编码图像、零或多个相应的冗余编码图像以及零或多个非 VCL NAL 单元组成。初始或冗余编码图像的 VCL NAL 单元之间的关系在7.4.1.2.5节描述。

比特流中第一个访问单元开始于该比特流的第一个 NAL 单元。

在基本编码图像的最后一个 VCL NAL 单元之后的第一个任何下列 NAL 单元代表了一个新的访问单元的开始。

— 访问单元分隔 NAL 单元（存在时）

— 序列参数集 NAL 单元（存在时）

— 图像参数集 NAL 单元（存在时）

— SEI NAL 单元（存在时）

— nal_unit_type 值在 14-18 之间（包括）的 NAL 单元

— 基本编码图像的第一个 VCL NAL 单元（总是存在）

对于基本编码图像的第一个 VCL NAL 单元的检测的规定在7.4.1.2.4节给出。

访问单元的编码图像和非 VCL NAL 单元的顺序应遵从下列规定：

— 当一个访问单元分隔NAL单元存在时，它应该是第一个NAL单元。在任何访问单元中应至多有一个访问单元分隔NAL单元。

— 当任何SEI NAL单元存在时，它们应在基本编码图像之前。

— 当一个包含缓冲周期SEI消息的SEI NAL单元存在时，该缓冲周期SEI消息应是该访问单元的第一个SEI NAL单元的第一个SEI消息载荷。

— 基本编码图像应在有关冗余编码图像之前。

— 当冗余编码图像存在时，它们的顺序应按redundant_pic_cnt值的升序排列。

— 当存在一个序列参数集扩展NAL单元时，它应该是一个与该序列参数集扩展NAL单元的seq_parameter_set_id值相同的序列参数集NAL单元之后的下一个NAL单元。

— 当一个没有分割NAL单元的辅助编码图像的一个或多个编码条带存在时，它们应在基本编码图像和所有冗余编码图像（如果有的话）之后。

— 当存在一个视频序列NAL单元的结尾时，它应该在基本编码图像和所有冗余编码图像（如果有的话）以及一个没有分割NAL单元（如果有的话）的辅助编码图像的所有编码条带之后。

— 当存在一个流NAL单元的结尾时，它应该是最后一个NAL单元。

— nal_unit_type值等于0、12或在20到31范围内（包括20和31）的NAL单元不应该出现在基本编码图像的第一个VCL NAL单元之前。

注 1 — 序列参数集NAL单元或图像参数集NAL单元可以在一个访问单元中出现，但不能在该访问单元的基本编码图像的最后一个VCL NAL单元之后，因为这种情况将代表一个新的访问单元的开始。

注 2 — 当在一个访问单元中存在一个nal_unit_type 值等于7或8的NAL单元时，它可以在它所在的访问单元的编码图像中引用或不引用，并且可以在后面的访问单元的编码图像中引用。

图 7-1 列出了不包含任何具有 nal_unit_type 值为 0、7、8 或在 12-18 或在 20-31 范围内（包括 12、18、20、31）的 NAL 单元的访问单元的结构。

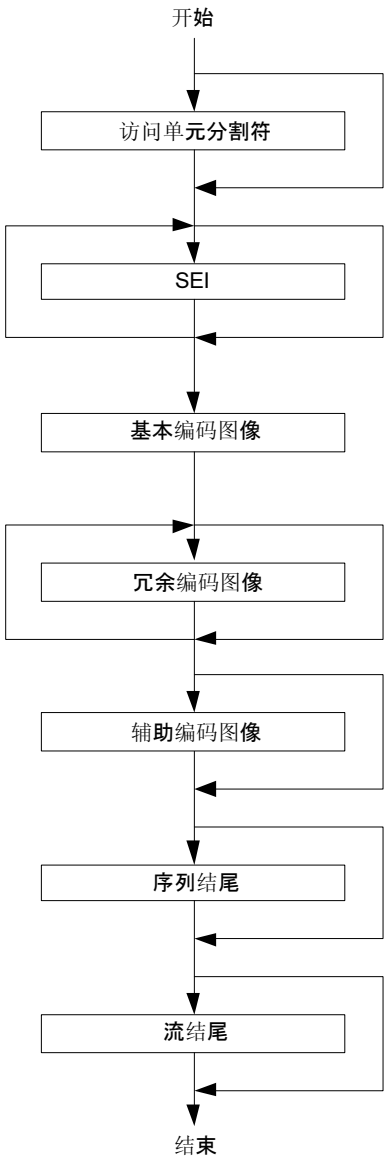


图 7-1－不包含任何具有nal_unit_type值为0、7、8或在12-18或在20-31范围内（包括12、18、20、31）的NAL单元的访问单元的结构

7.4.1.2.4 基本编码图像的第一个VCL NAL单元的检测

本节规定了 VCL NAL 单元的语法，使其能够检测到每个基本编码图像的第一个 VCL NAL 单元。

当前访问单元的基本编码图像的任何编码条带 NAL 单元或编码条带数据分割块 A 的 NAL 单元应与前一个访问单元的基本编码图像的任何编码条带 NAL 单元或编码条带数据分割块 A 的 NAL 单元以下列方式中的一种或多种进行区分：

— frame_num的值不同。不管由于memory_management_control_operation等于5的情况出现而为了在后续解码过程中使用frame_num的值是否已经等于0，该值通常在测试条件中是在条带头的语法中出现的frame_num的值。

注 1 — 上述情况的一个推理是一个包含frame_num值等于1的基本编码图像不能包含一个等于5的memory_management_control_operation，除非跟随其后的下一个基本编码图像（如果有的话）能够满足下面列出的其它条件。

— pic_parameter_set_id 值不同。

— field_pic_flag 值不同。

— bottom_field_flag 在两个访问单元中都出现而且值不同。

— nal_ref_idc 值不同，而且其中一个的nal_ref_idc 值等于0。

— 两个访问单元的pic_order_cnt_type 都等于0，并且两个pic_order_cnt_lsb值不同或delta_pic_order_cnt_bottom 值不同。

— 两个访问单元的pic_order_cnt_type都等于1，并且两个delta_pic_order_cnt[0] 值不同或者delta_pic_order_cnt[1] 值不同。

— nal_unit_type 值不同，而且其中一个的nal_unit_type 值等于5。

— 两个访问单元的nal_unit_type都等于5，并且idr_pic_id 值不同。

注 2 — 在冗余编码图像中的一些VCL NAL单元或一些非VCL NAL单元（例如：一个访问单元分隔符NAL单元）也可用于检测访问单元的边界，而且可以因此辅助检测一个新的基本编码图像的开始。

7.4.1.2.5 VCL NAL单元的顺序及其与编码图像的关系

每个 VCL NAL 单元都是一个编码图像的一部分。

一个编码 IDR 图像中的 VCL NAL 单元的顺序规定如下：

— 如果像在附件A中规定的那样允许任意条带顺序的话，一个IDR图像NAL单元的编码条带相互间可以按任何顺序。

— 否则（不允许随意的条带顺序），一个IDR图像NAL单元的编码条带顺序应按一个IDR图像NAL单元的每个编码条带的第一个宏块地址递增的顺序。

一个编码的非IDR图像中的VCL NAL单元的顺序按下列规定：

— 如果像在附件A中规定的那样允许任意条带顺序的话，一个非IDR图像NAL单元或编码条带数据分割块A的NAL单元的编码条带相互间可以按任何顺序。具有特定的slice_id值的一个编码条带数据分割块A的NAL单元应在任何已有的具有相同slice_id值的编码条带数据分割块B的NAL单元之前。一个具有特定的slice_id值的编码条带数据分割块A的NAL单元应在任何已有的具有相同slice_id值的编码条带数据分割块C的NAL单元之前。当一个具有特定的slice_id值的编码条带数据分割块B的NAL单元存在时，它应在任何已有的具有相同slice_id值的编码条带数据分割块C的NAL单元之前。

— 否则（不允许随意的条带顺序），一个非IDR图像NAL单元或编码条带数据分割块A的NAL单元的编码条带顺序应按一个非IDR图像NAL单元或编码条带数据分割块A的NAL单元的每个编码条带的第一个宏块地址递增的顺序。具有特定slice_id值的编码条带数据分割块A的NAL单元应直接在任何已有的具有相同slice_id值的编码条带数据分割块B的NAL单元之前。当不存在一个具有相同slice_id值的编码条带数据分割块B的NAL单元时，一个具有特定的slice_id值的编码条带数据分割块A的NAL单元应直接在任何已有的具有相同slice_id值的编码条带数据分割块C的NAL单元之前。当一个具有特定的slice_id值的编码条带数据分割块B的NAL单元存在时，它应直接在任何已有的具有相同slice_id值的编码条带数据分割块C的NAL单元之前。

具有 nal_unit_type 等于 12 的 NAL 单元可能会出现在访问单元中但不能位于该访问单元中的基本编码图像的第一个 VCL NAL 单元之前。

具有 nal_unit_type 等于 0 或其值在 24-31（包括 24 和 31，未特别指定）范围内的 NAL 单元可能会出现在访问单元中，但不能位于该访问单元中的基本编码图像的第一个 VCL NAL 单元之前。

NAL 单元具有的 nal_unit_type 值如果在 20-23（包括 20 和 23，保留）范围内，则其不能位于该访问单元中基本编码图像的第一个 VCL NAL 单元之前（当将来由 ITU-T | ISO/IEC 规定时）。

7.4.2 原始字节序列载荷及RBSP拖尾比特语义

7.4.2.1 序列参数集RBSP语义

profile_idc 和 **level_idc** 是指比特流所遵守的配置和级别，如附件 A 中所给出的。

constraint_set0_flag 等于 1 是指比特流遵从 A.2.1 节中的所有规定。constraint_set0_flag 等于 0 是指该比特流可以遵从也可以不遵从 A.2.1 节中的所有规定。

constraint_set1_flag 等于 1 是指比特流遵从 A.2.2 节中的所有规定。constraint_set1_flag 等于 0 是指该比特流可以遵从也可以不遵从 A.2.2 节中的所有规定。

constraint_set2_flag 等于 1 是指该比特流遵从 A.2.3 节中的所有规定。constraint_set2_flag 等于 0 是指该比特流可以遵从也可以不遵从 A.2.3 节中的所有规定。

注 1 — 当constraint_set0_flag、constraint_set1_flag 或 constraint_set2_flag中的一个或多个等于1，该比特流必须遵从A.2节下面的所有规定。当profile_idc等于100、110、122或144时，constraint_set0_flag、constraint_set1_flag 和 constraint_set2_flag都应等于0。

constraint_set3_flag 含义如下：

— 如果 profile_idc等于66、77或88并且level_ idc等于 11，constraint_set3_flag 等于 1 是指该比特流遵从附件A中对级别1b的所有规定，constraint_set3_flag等于0是指该比特流可以遵从也可以不遵从附件A中有关1b级别的所有规定。

— 否则（profile_idc等于100、110、122或144或level_ idc不等于 11），constraint_set3_flag 等于1留作未来 ITU-T | ISO/IEC 使用。根据本建议书 | 国际标准的规定，当profile_idc等于100、110、122或144或level_ idc不等于 11时，比特流中constraint_set3_flag 应等于0。当profile_idc等于100、110、122或144或level_ idc不等于 11时，遵从本建议书 | 国际标准的解码器将忽略constraint_set3_flag 的值。

reserved_zero_4bits 应等于 0。reserved_zero_4bits 的其他取值将由 ITU-T | ISO/IEC 未来规定。解码器将忽略 reserved_zero_4bits 的值。

seq_parameter_set_id 用于识别图像参数集所指的序列参数集。seq_parameter_set_id 的值应在 0-31 的范围内，包括 0 和 31。

注 2 — 当可行时，如果其他序列参数集语法元素的值不一致，编码器将使用不同的seq_parameter_set_id值，而不是改变与特定seq_parameter_set_id值相关的语法元素的值。

chroma_format_idc 是指如 6.2 节所提出的，与亮度取样对应的色度取样。chroma_format_idc 的值应该在 0 到 3 的范围内（包括 0 和 3）。当 chroma_format_idc 不存在时，应推断其值为 1（4：2：0 的色度格式）。

residual_colour_transform_flag 值等于 1 时，应用 8.5 节规定的残余颜色变换。residual_colour_transform_flag 等于 0 时则不使用残余颜色变换。当 residual_colour_transform_flag 不存在时，默认其值为 0。

bit_depth_luma_minus8 是指亮度队列样值的比特深度以及亮度量化参数范围的取值偏移 $QpBdOffset_Y$，如下所示：

$$BitDepth_Y = 8 + bit\_depth\_luma\_minus8 \tag{7-1}$$

$$QpBdOffset_Y = 6 * bit\_depth\_luma\_minus8 \tag{7-2}$$

当 bit_depth_luma_minus8 不存在时，应推定其值为 0。bit_depth_luma_minus8 取值范围应该在 0 到 4 之间（包括 0 和 4）。

bit_depth_chroma_minus8 是指色度队列样值的比特深度以及色度量化参数范围的取值偏移 $QpBdOffset_C$，如下所示：

$$BitDepth_C = 8 + bit\_depth\_chroma\_minus8 \tag{7-3}$$

$$QpBdOffset_C = 6 * (bit\_depth\_chroma\_minus8 + residual\_colour\_transform\_flag) \tag{7-4}$$

当 bit_depth_chroma_minus8 不存在时，应推定其值为 0。bit_depth_chroma_minus8 取值范围应该在 0 到 4 之间（包括 0 和 4）。

变量 RawMbBits 按下列公式得出：

$$RawMbBits = 256 * BitDepth_Y + 2 * MbWidthC * MbHeightC * BitDepth_C \tag{7-5}$$

qpprime_y_zero_transform_bypass_flag 等于 1 是指当 QP'_Y 等于 0 时变换系数解码过程的变换旁路操作和图像构建过程将会在第 8.5 节给出的去块效应滤波过程之前执行。qpprime_y_zero_transform_bypass_flag 等于 0 是指变换系数解码过程和图像构建过程在去块效应滤波过程之前执行而不使用变换旁路操作。当 qpprime_y_zero_transform_bypass_flag 没有特别指定时，应推定其值为 0。

seq_scaling_matrix_present_flag 等于 1 表示存在 i=0..7 的标志 seq_scaling_list_present_flag[i]。seq_scaling_matrix_present_flag 等于 0 则表示不存在这些标志并且由 Flat_4x4_16 表示的序列级别的缩放比例列表应被推断出来（对应 i=0..5），由 Flat_8x8_16 表示的序列级别的缩放比例列表应被推断出来（对应 i=6..7）。当 seq_scaling_matrix_present_flag 没有特别指定时，应推定其值为 0。

缩放比例列表 Flat_4x4_16 和 Flat_8x8_16 规定如下：

Flat_4x4_16[i] = 16, i = 0..15, (7-6)

Flat_8x8_16[i] = 16, i = 0..63. (7-7)

seq_scaling_list_present_flag[i] 等于 1 是指视频序列参数集中存在缩放比例列表 i 的语法结构。seq_scaling_list_present_flag[i] 等于 0 表示视频序列参数集中不存在缩放比例列表 i 的语法结构并且表 7-2 中列出的缩放比例序列后退规则集 A 应用于以 i 值为索引的序列级别的缩放比例列表。

表 7-2－缩放比例列表的记忆名索引号分配以及后退规则的规范

缩放比例索引号	记忆名	块的大小	MB 预期类型	组成	缩放比例序列后退规则集 A	缩放比例序列后退规则集 B	默认缩放比例序列
0	Sl_4x4_Intra_Y	4x4	Intra	Y	默认缩放比例序列	视频序列级别缩放比例序列	Default_4x4_Intra
1	Sl_4x4_Intra_Cb	4x4	Intra	Cb	i = 0 的缩放比例序列	i = 0 的缩放比例序列	Default_4x4_Intra
2	Sl_4x4_Intra_Cr	4x4	Intra	Cr	i = 1 的缩放比例序列	i = 1 的缩放比例序列	Default_4x4_Intra
3	Sl_4x4_Inter_Y	4x4	Inter	Y	默认缩放比例序列	视频序列级别缩放比例序列	Default_4x4_Inter
4	Sl_4x4_Inter_Cb	4x4	Inter	Cb	i = 3 的缩放比例序列	i = 3 的缩放比例序列	Default_4x4_Inter
5	Sl_4x4_Inter_Cr	4x4	Inter	Cr	i = 4 的缩放比例序列	i = 4 的缩放比例序列	Default_4x4_Inter
6	Sl_8x8_Intra_Y	8x8	Intra	Y	默认缩放比例序列	视频序列级别缩放比例序列	Default_8x8_Intra
7	Sl_8x8_Inter_Y	8x8	Inter	Y	默认缩放比例序列	视频序列级别缩放比例序列	Default_8x8_Inter

表 7-3 给出了默认缩放比例列表 Default_4x4_Intra 和 Default_4x4_Inter 的规范。 表 7-4 给出了默认缩放比例列表 Default_8x8_Intra 和 Default_8x8_Inter 的规范。

表 7-3－默认缩放比例列表 Default_4x4_Intra和 Default_4x4_Inter的规范

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Default_4x4_Intra[idx]	6	13	13	20	20	20	28	28	28	28	32	32	32	37	37	42
Default_4x4_Inter[idx]	10	14	14	20	20	20	24	24	24	24	27	27	27	30	30	34

表 7-4－默认缩放比例列表Default _8x8_Intra 和 Default_8x8_Inter的规范

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Default_8x8_Intra[idx]	6	10	10	13	11	13	16	16	16	16	18	18	18	18	18	23
Default_8x8_Inter[idx]	9	13	13	15	13	15	17	17	17	17	19	19	19	19	19	21

表 7-4（续）－默认缩放比例列表Default _8x8_Intra 和 Default_8x8_Inter的规范

idx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Default_8x8_Intra[idx]	23	23	23	23	23	25	25	25	25	25	25	25	27	27	27	27
Default_8x8_Inter[idx]	21	21	21	21	21	22	22	22	22	22	22	22	24	24	24	24

表 7-4（续）－默认缩放比例列表Default _8x8_Intra 和 Default_8x8_Inter的规范

idx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Default_8x8_Intra[idx]	27	27	27	27	29	29	29	29	29	29	29	31	31	31	31	31
Default_8x8_Inter[idx]	24	24	24	24	25	25	25	25	25	25	25	27	27	27	27	27

表 7-4（结束）－默认缩放比例列表Default _8x8_Intra 和 Default_8x8_Inter的规范

idx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Default_8x8_Intra[idx]	31	33	33	33	33	33	36	36	36	36	38	38	38	40	40	42
Default_8x8_Inter[idx]	27	28	28	28	28	28	30	30	30	30	32	32	32	33	33	35

log2_max_frame_num_minus4 按下列公式可得出与 frame_num 相关的变量 MaxFrameNum 的值。

$$MaxFrameNum = 2^{(log2\_max\_frame\_num\_minus4 + 4)} \quad (7\text{-}8)$$

log2_max_frame_num_minus4 的值应在 0-12 范围内（包括 0 和 12）。

pic_order_cnt_type 是指解码图像顺序的计数方法（如 8.2.1 节所述）。pic_order_cnt_type 的取值范围是 0 到 2（包括 0 和 2）。

包含下列任何情况的一个编码视频序列中 pic_order_cnt_type 都不能等于 2：

- 一个包含非参考图像的访问单元紧跟在一个包含非参考帧的访问单元之后；
- 一个包含非参考图像的访问单元紧跟在分别包含一个由两个场一起组成一个互补的非参考场对的两个访问单元之后；
- 一个包含非参考场的访问单元之后紧跟在一个包含另一个非参考图像的访问单元（该访问单元没有与两个访问单元中的第一个组成一个互补的非参考场对）

log2_max_pic_order_cnt_lsb_minus4 表示用于 8.2.1 节规定的图像顺序数解码过程中的变量 MaxPicOrderCntLsb 的值，公式如下：

$$MaxPicOrderCntLsb = 2^{(\ log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4\ )} \quad (7\text{-}9)$$

log2_max_pic_order_cnt_lsb_minus4 的取值范围应该在 0-12 内（包括 0 和 12）。

delta_pic_order_always_zero_flag 等于 1 表示视频序列的条带头中没有 delta_pic_order_cnt[0] 和 delta_pic_order_cnt[1] 两个字段，它们的值都默认为 0。delta_pic_order_always_zero_flag 等于 0 表示视频序列的条带头中包含 delta_pic_order_cnt[0]字段，并可能包括 delta_pic_order_cnt[1]的字段（也可以不包括）。

offset_for_non_ref_pic 用于计算 8.2.1 节规定的一个非参考图像的图像顺序号。offset_for_non_ref_pic 的取值范围是 -2^{31} 到 $2^{31}-1$（包括-2^{31} 和 $2^{31}-1$）。

offset_for_top_to_bottom_field 用于计算 8.2.1 节规定的一个帧的底场的图像顺序号。offset_for_top_to_bottom_field 的取值范围是 -2^{31} 到 $2^{31}-1$（包括-2^{31} 和 $2^{31}-1$）。

num_ref_frames_in_pic_order_cnt_cycle 用于 8.2.1 节规定的图像顺序号的解码过程。num_ref_frames_in_pic_order_cnt_cycle 的取值范围是 0 到 255（包括 0 和 255）。

offset_for_ref_frame[i] 是在 8.2.1 节规定的图像顺序号的解码过程中使用的一个 num_ref_frames_in_pic_order_cnt_cycle 值的列表中的一个元素。offset_for_ref_frame[i] 的取值范围是-2^{31} 到 $2^{31}-1$（包括-2^{31} 和 $2^{31}-1$）。

num_ref_frames 规定了可能在视频序列中任何图像帧间预测的解码过程中用到的短期参考帧和长期参考帧、互补参考场对以及不成对的参考场的最大数量。num_ref_frames 字段也决定了 8.2.5.3 节规定的滑动窗口操作的大小。num_ref_frames 的取值范围应该在 0 到 MaxDpbSize （参见 A.3.1 或 A.3.2 节的定义）范围内，包括 0 和 MaxDpbSize 。

gaps_in_frame_num_value_allowed_flag 表示 7.4.3 节给出的 frame_num 的允许值以及在 8.2.5.2 节给出的 frame_num 值之间存在推测的差异的情况下进行的解码过程。

pic_width_in_mbs_minus1 加 1 是指以宏块为单元的每个解码图像的宽度。

以宏块为单元的图像宽度变量由下列公式可得：

$$PicWidthInMbs = pic\_width\_in\_mbs\_minus1 + 1 \quad (7\text{-}10)$$

亮度分量的图像宽度变量由下列公式得出：

$$PicWidthInSamples_L = PicWidthInMbs * 16 \quad (7\text{-}11)$$

色度分量的图像宽度变量由下列公式得出：

$$PicWidthInSamples_C = PicWidthInMbs * MbWidthC \quad (7\text{-}12)$$

pic_height_in_map_units_minus1 加 1 表示以条带组映射为单位的一个解码帧或场的高度。

变量 PicHeightInMapUnits 和 PicSizeInMapUnits 由下列公式得出：

$$PicHeightInMapUnits = pic\_height\_in\_map\_units\_minus1 + 1 \quad (7\text{-}13)$$

$$PicSizeInMapUnits = PicWidthInMbs * PicHeightInMapUnits \quad (7\text{-}14)$$

frame_mbs_only_flag 等于 0 表示编码视频序列的编码图像可能是编码场或编码帧。frame_mbs_only_flag 等于 1 表示编码视频序列的每个编码图像都是一个仅包含帧宏块的编码帧。

变量 pic_width_in_mbs_minus1、pic_height_in_map_units_minus1 和 frame_mbs_only_flag 的允许取值范围在附件 A 中规定。

变量 pic_height_in_map_units_minus1 的语义依赖于变量 frame_mbs_only_flag，规定如下：

-— 如果 frame_mbs_only_flag 等于0，pic_height_in_map_units_minus1加1就表示以宏块为单位的一场的高度。

-— 否则（frame_mbs_only_flag等于1），pic_height_in_map_units_minus1加1就表示以宏块为单位的一帧的高度。

变量 FrameHeightInMbs 由下列公式得出：

FrameHeightInMbs = (2 – frame_mbs_only_flag) * PicHeightInMapUnits (7-15)

mb_adaptive_frame_field_flag 等于 0 表示在一个图像的帧和场宏块之间没有交换。mb_adaptive_frame_field_flag 等于 1 表示在帧和帧内的场宏块之间可能会有交换。当 mb_adaptive_frame_field_flag 没有特别规定时，默认其值为 0。

direct_8x8_inference_flag 表示在 8.4.1.2 节中规定的 B_Skip、B_Direct_16x16 和 B_Direct_8x8 亮度运动矢量的计算过程使用的方法。当 frame_mbs_only_flag 等于 0 时 direct_8x8_inference_flag 应等于 1。

frame_cropping_flag 等于 1 表示帧剪切偏移参数遵从视频序列参数集中的下一个值。frame_cropping_flag 等于 0 表示不存在帧剪切偏移参数。

frame_crop_left_offset, frame_crop_right_offset, frame_crop_top_offset, frame_crop_bottom_offset 是指从解码过程中输出的编码图像序列中的图像样值以帧坐标中的一个矩阵区域的形式输出。

变量 CropUnitX 和 CropUnitY 由下列公式得出：

— 如果 chroma_format_idc 等于0， CropUnitX 和 CropUnitY 按下列方式计算：

CropUnitX = 1 (7-16)

CropUnitY = 2 – frame_mbs_only_flag (7-17)

— 否则（chroma_format_idc 等于 1、 2 或 3），CropUnitX 和CropUnitY 按下列公式计算：

CropUnitX = SubWidthC (7-18)

CropUnitY = SubHeightC * (2 – frame_mbs_only_flag) (7-19)

帧剪切矩形包括水平帧坐标从 CropUnitX * frame_crop_left_offset 到 $PicWidthInSamples_L$ – (CropUnitX * frame_crop_right_offset + 1) 且垂直帧坐标从 CropUnitY * frame_crop_top_offset 到 (16 * FrameHeightInMbs) – (CropUnitY * frame_crop_bottom_offset + 1)的亮度样点。frame_crop_left_offset 的值应在 0 到 ($PicWidthInSamples_L$ / CropUnitX) – (frame_crop_right_offset + 1) 范围内（包括边界值），frame_crop_top_offset 的值应在 0 到(16 * FrameHeightInMbs / CropUnitY) – (frame_crop_bottom_offset + 1)范围内（包括边界值）。

当 frame_cropping_flag 等于 0 时，frame_crop_left_offset、frame_crop_right_offset、 frame_crop_top_offset 和 frame_crop_bottom_offset 的值应等于 0。

当 chroma_format_idc 不等于 0 时，两个色度队列的相应的指定样点是那些帧坐标为（x / SubWidthC, y / SubHeightC）的样点，其中（x，y）是指定的亮度样点的帧坐标。

对于解码场，解码场的指定样点是那些帧坐标落在指定的矩形当中的样点。

vui_parameters_present_flag 等于 1 表示存在如附录 E 提到的 vui_parameters( ) 语法结构。vui_parameters_present_flag 等于 0 表示不存在如附录 E 提到的 vui_parameters( ) 语法结构。

~~0.1.1.1.1~~7.4.2.1.1 缩放比例列表的语义

delta_scale 是用于计算缩放比例列表中的第 j 个元素，j 为 0 到 sizeOfScalingList – 1（包括边界值）。delta_scale 的值应在-128 到 +127 之间（包括-128 和+127）。

当计算出 useDefaultScalingMatrixFlag 等于 1 时，应推定缩放比例列表等于表 7-2 给出的默认的缩放比例列表。

~~0.1.1.1.2~~7.4.2.1.2 序列参数集扩展RBSP语义

seq_parameter_set_id 标识与序列参数集扩展有关的序列参数集。变量 seq_parameter_set_id 的值应在 0 到 31 范围内（包括 0 和 31）。

aux_format_idc 等于 0 表示在编码视频序列中没有辅助编码图像。变量 aux_format_idc 等于 1 表示编码视频序列中的每个访问单元都仅有一个辅助编码图像，并且为 alpha 混色的目的，在从解码过程输出之后的显示过程中，每个访问单元中相关的基本编码图像的解码样点值应乘以该访问单元中的辅助编码图像的解释样点值。aux_format_idc 等于 2 表示编码视频序列中的每个访问单元都仅有一个辅助编码图像，并且为 alpha 混色的目的，在从解码过程输出之后的显示过程中，每个访问单元中相关的基本编码图像的解码样点值不应乘以该访问单元中的辅助编码图像的解释样点值。aux_format_idc 等于 3 表示编码视频序列中的每个访问单元都仅有一个辅助编码图像，并且不规定辅助编码图像的使用。aux_format_idc 的值应在 0 到 3 的范围内（包括 0 和 3）。表示编码视频序列中的每个访问单元都仅有一个辅助编码图像的 aux_format_idc 大于 3 的值为将来由 ITU-T | ISO/IEC 使用而保留。当 aux_format_idc 未做特别规定时，其值默认为 0。

注 1 — 遵照本建议书 | 国际标准的解码器不必需对辅助编码图像解码。

bit_depth_aux_minus8 表示辅助编码图像的样点队列中样值的比特深度。bit_depth_aux_minus8 的取值范围是 0 到 4（包括 0 和 4）。

alpha_incr_flag 等于 0 表示为 alpha 混合的目的，每个解码辅助编码图像样点值的解释样点值等于解码辅助编码图像样点的值。alpha_incr_flag 等于 1 表示为 alpha 混色的目的，在解码辅助编码图像样点之后，任何大于 Min（alpha_opaque_value, alpha_transparent_value ）的辅助编码图像样点值应加 1 以得到辅助编码图像样点的解释样点值，并且任何小于等于 Min（alpha_opaque_value, alpha_transparent_value ）的辅助解码图像样点值应不改变即可使用，像解码辅助编码图像样点值的解释样点值一样。

alpha_opaque_value 表示一个辅助编码图像样点的解释样点值，并且为 alpha 混色的目的该解释样点值的同一个访问单元的相关的亮度和色度样点认为是不透明的。用于代表 alpha_opaque_value 语法元素的比特数是 bit_depth_aux_minus8 + 9 比特。

alpha_transparent_value 表示一个辅助编码图像样点的解释样点值，并且为 alpha 混色的目的该解释样点值的同一个访问单元的相关的亮度和色度样点认为是透明的。用于代表 alpha_transparent_value 语法元素的比特数是 bit_depth_aux_minus8 + 9 比特。

当 alpha_incr_flag 等于 1 时，变量 alpha_transparent_value 不应等于变量 alpha_opaque_value，并且 Log2(Abs(alpha_opaque_value – alpha_transparent_value))应是一个整数。一个 alpha_transparent_value 的值等于 alpha_opaque_value 表示辅助编码图像不是用于 alpha 混色的。

注 2 — 为alpha混色的目的，变量alpha_opaque_value可以大于变量alpha_transparent_value，或者小于变量alpha_transparent_value。解释样点值应严格限于alpha_opaque_value 和alpha_transparent_value范围内（包括边界值）。

序列参数集扩展的解码以及辅助编码图像的解码不需要与本建议书 | 国际标准一致。

辅助编码图像的每个编码条带的定义应遵从与一个冗余图像的编码条带一样的规定，下列不同除外：

–— 下列应用与基本编码图像是否是一个 IDR 图像有关：

– — 如果基本编码图像是一个 IDR 图像，辅助编码条带语法应符合一个具有 nal_unit_type 等于 5 的条带（一个 IDR 图像的条带）

– — 否则（基本编码图像不是一个 IDR 图像），辅助编码条带的定义应符合一个具有 nal_unit_type 等于 1 的条带（一个非 IDR 图像的条带）。

– — 一个辅助编码图像（当存在时）的条带应包含那些与基本编码图像相关的所有宏块。

– — 在所有辅助编码条带中 redundant_pic_cnt 应等于 0。

辅助编码图像的解码过程（可选）就好像辅助编码图像是一个独立的编码视频流中的基本编码图像一样，这与该基本编码图像在当前的编码视频流中有下列不同：

– 每个辅助编码图像的IDR或非IDR状态应与同一个访问单元的初始图像的IDR或非IDR状态一样，而不是由nal_ref_idc 的值推定。

– 在辅助编码图像解码时chroma_format_idc 的值应推定等于0。

– 在辅助编码图像解码时变量 bit_depth_luma_minus8 应等于变量 bit_depth_aux_minus8 的值。

注 3 — 正常情况下，alpha混色由大小全部相同的一个背景图像B、一个前景图像F和一个解码辅助解码图像A组成。为举例说明假定B和F的色度分辨率被过取样成与亮度分辨率一样。相应的b、f和a分别表示B、F和A的样点值，而下标Y、Cb和Cr表示亮度和色度。

变量 alphaRange、alphaFwt 和 alphaBwt 由下列公式定义：

alphaRange = Abs(alpha_opaque_value －alpha_transparent_value)

alphaFwt = Abs(a －alpha_transparent_value)

alphaBwt = Abs(a －alpha_opaque_value)

这样，在 alpha 混色中，表示显示图像 D 的样点 d 可以由下列公式得出：

$d_Y = (alphaFwt*f_Y + alphaBwt*b_Y + alphaRange/2) / alphaRange$

$d_{CB} = (alphaFwt*f_{CB} + alphaBwt*b_{CB} + alphaRange/2) / alphaRange$

$d_{CR} = (alphaFwt*f_{CR} + alphaBwt*b_{CR} + alphaRange/2) / alphaRange$

图像 D、F 和 B 的样点也可以由红、绿和蓝分量值（见 E.2.1 节）表示。这里我们假定用 Y、Cb 和 Cr 分量值。每个分量，例如 Y，为上述举例说明的目的而被假设为在每个图像 D、F 和 B 中都具有同样的比特深度。然而不同的分量，例如在本例中的 Y 和 Cb 不需要具有同样的比特深度。

当 aux_format_idc 等于 1 时，F 将是从解码亮度和色度中获取的解码图像，A 将是从解码辅助编码图像中获取的解码图像。在这种情况下，举例的 alpha 混色合成涉及把 F 的样点值乘以从 A 的样点值中获取的因子。

在编辑或直接观看时十分有用的一个通用图像格式称作预乘法-黑视频。如果前景图像是 F，那么预乘法-黑视频 S 通过下列方式得到：

$s_Y = (alphaFwt*f_Y) / alphaRange$

$s_{CB} = (alphaFwt*f_{CB}) / alphaRange$

$s_{CR} = (alphaFwt*f_{CR}) / alphaRange$

预乘法-黑视频 S 具有图像在一个黑背景下将正确显示的特征。对于一个非黑的背景 B，显示图像 D 的成分可以计算如下：

$d_Y = s_Y + (alphaBwt*b_Y + alphaRange/2) / alphaRange$

$d_{CB} = s_{CB} + (alphaBwt*b_{CB} + alphaRange/2) / alphaRange$

$d_{CR} = s_{CR} + (alphaBwt*b_{CR} + alphaRange/2) / alphaRange$

当 aux_format_idc 等于 2 时，S 将表示从解码亮度和色度中获取的解码图像，A 将仍然表示从解码辅助编码图像中获取的解码图像。在这种情况下，alpha 混色合成不涉及把 F 的样点值乘以从 A 的样点值中获取的因子。

additional_extension_flag 等于 0 表示在 RBSP 拖尾比特之前的视频序列参数集扩展语法结构中没有跟随额外的数据。变量 additional_extension_flag 的值应等于 0。additional_extension_flag 等于 1 的值保留用于 ITU-T | ISO/IEC 将来定义。符合本建议书 | 国际标准的解码器应忽略所有在一个视频序列参数集扩展 NAL 单元中 additional_extension_flag 值为 1 之后跟随的数据。

~~0.1.1.2~~7.4.2.2 图像参数集RBSP语义

pic_parameter_set_id 标识在条带头中提到的图像参数集。变量 pic_parameter_set_id 的值应该在 0 到 255 的范围内（包括 0 和 255）。

seq_parameter_set_id 是指活动的序列参数集。变量 seq_parameter_set_id 的值应该在 0 到 31 的范围内（包括 0 和 31）。

entropy_coding_mode_flag 用于选取语法元素的熵编码方式，在语法表中由两个标识符代表，具体如下：

– 如果entropy_coding_mode_flag 等于0，那么采用语法表中左边的描述符所指定的方法（Exp-Golomb编码，见9.1节，或CAVLC，见9.2节）。

– 否则（entropy_coding_mode_flag 等于1），就采用语法表中右边的描述符所指定的方法（CABAC，见9.3节）。

pic_order_present_flag 等于 1 表示与图像顺序数有关的语法元素将出现于条带头中，如第 7.3.3 节所规定。pic_order_present_flag 等于 0 表示条带头中不会出现与图像顺序数有关的语法元素。

num_slice_groups_minus1 加 1 表示一个图像中的条带组数。当 num_slice_groups_minus1 等于 0 时，图像中所有的条带属于同一个条带组。num_slice_groups_minus1 的允许取值范围在附件 A 中给出。

slice_group_map_type 表示条带组中条带组映射单元的映射是如何编码的。slice_group_map_type 的取值范围应该在 0 到 6 内（包括 0 和 6）。

slice_group_map_type 等于 0 表示隔行扫描的条带组。

slice_group_map_type 等于 1 表示一种分散的条带组映射。

slice_group_map_type 等于 2 表示一个或多个前景条带组和一个残余条带组。

slice_group_map_type 的值等于 3、4 和 5 表示变换的条带组。当 num_slice_groups_minus1 不等于 1 时，slice_group_map_type 不应等于 3、4 或 5。

slice_group_map_type 等于 6 表示对每个条带组映射单元清楚地分配一个条带组。

条带组映射单元规定如下：

– 如果 frame_mbs_only_flag 等于0，mb_adaptive_frame_field_flag等于1，而且编码图像是一个帧，那么条带组映射单元就是宏块对单元。

– 否则，如果frame_mbs_only_flag等于1或者一个编码图像是一个场，条带组映射单元就是宏块的单元。

– 否则（frame_mbs_only_flag 等于0， mb_adaptive_frame_field_flag 等于0，并且编码图像是一个帧），条带组映射单元就是像在一个MBAFF帧中的一个帧宏块对中一样垂直相邻的两个宏块的单元。

run_length_minus1[i **]** 用来指定条带组映射单元的光栅扫描顺序中分配给第 i 个条带组的连续条带组映射单元的数目。run_length_minus1[i] 的取值范围应该在 0 到 PicSizeInMapUnits – 1 内（包括边界值）。

top_left[i **]** 和 **bottom_right[** i **]** 分别表示一个矩形的左上角和右下角。top_left[i] 和 bottom_right[i] 是条带组映射单元所在图像的光栅扫描中的条带组映射单元位置。对于每个矩形 i，语法元素 top_left[i] 和 bottom_right[i]的值都应该遵从下面所有的规定。

– top_left[i] 应小于或等于bottom_right[i] 并且 bottom_right[i] 应小于 PicSizeInMapUnits。

– （ top_left[i] % PicWidthInMbs ）应小于或等于 （ bottom_right[i] % PicWidthInMbs ）的值。

slice_group_change_direction_flag 通常与 slice_group_map_type 一起用来表示当 slice_group_map_type 的值为 3、4 或 5 时精确的映射类型。

slice_group_change_rate_minus1 用来指定变量 SliceGroupChangeRate。SliceGroupChangeRate 表示一个条带组的大小从一个图像到下一个的改变的倍数，以条带组映射单元为单位。slice_group_change_rate_minus1 的值应该在 0 到 PicSizeInMapUnits – 1 的范围内（包括边界值）。变量 SliceGroupChangeRate 规定如下：

SliceGroupChangeRate = slice_group_change_rate_minus1 + 1 (7-20)

pic_size_in_map_units_minus1 用来指定图像中的条带组映射单元数。pic_size_in_map_units_minus1 应该等于 PicSizeInMapUnits – 1。

slice_group_id[i **]** 表示光栅扫描顺序中的第 i 个条带组映射单元的一个条带组。slice_group_id[i] 语法元素的大小是 Ceil(Log2(num_slice_groups_minus1 + 1)) 比特。slice_group_id[i] 的值应该在 0 到 num_slice_groups_minus1 范围内（包括边界值）。

num_ref_idx_l0_active_minus1 表示参考图像列表 0 的最大参考索引号，该索引号将用来在一幅图像中 num_ref_idx_active_override_flag 等于 0 的条带使用列表 0 预测时，解码该图像的这些条带。当 MbaffFrameFlag 等于 1 时，num_ref_idx_l0_active_minus1 是帧宏块解码的最大索引号值，而 2 * num_ref_idx_l0_active_minus1 + 1 是场宏块解码的最大索引号值。num_ref_idx_l0_active_minus1 的值应该在 0 到 31 的范围内（包括 0 和 31）。

num_ref_idx_l1_active_minus1 与 num_ref_idx_l0_active_minus1 具有同样的定义，只是分别用 11 和列表 1 取代 10 和列表 0。

weighted_pred_flag 等于 0 表示加权的预测不应用于 P 和 SP 条带。weighted_pred_flag 等于 1 表示在 P 和 SP 条带中应使用加权的预测。

weighted_bipred_idc 等于 0 表示 B 条带应该采用默认的加权预测。weighted_bipred_idc 等于 1 表示 B 条带应该采用具体指明的加权预测。weighted_bipred_idc 等于 2 表示 B 条带应该采用隐含的加权预测。weighted_bipred_idc 的值应该在 0 到 2 之间（包括 0 和 2）。

pic_init_qp_minus26 表示每个条带的 $SliceQP_Y$ 初始值减 26。当解码非 0 值的 slice_qp_delta 时，该初始值在条带层被修正，并且在宏块层解码非 0 值的 mb_qp_delta 时进一步被修正。pic_init_qp_minus26 的值应该在－(26 + $QpBdOffset_Y$) 到 +25 之间（包括边界值）。

pic_init_qs_minus26 表示在 SP 或 SI 条带中的所有宏块的 $SliceQS_Y$ 初始值减 26。当解码非 0 值的 slice_qs_delta 时，该初始值在条带层被修正。pic_init_qs_minus26 的值应该在-26 到 +25 之间（包括边界值）。

chroma_qp_index_offset 表示为在 QP_C 值的表格中寻找 Cb 色度分量而应加到参数 QP_Y 和 QS_Y 上的偏移。chroma_qp_index_offset 的值应在-12 到 +12 范围内（包括边界值）。

deblocking_filter_control_present_flag 等于 1 表示控制去块效应滤波器的特征的一组语法元素将出现在条带头中。deblocking_filter_control_present_flag 等于 0 表示控制去块效应滤波器的特征的一组语法元素不会出现在条带头中，并且它们的推定值将会生效。

constrained_intra_pred_flag 等于 0 表示帧内预测允许使用残余数据，且使用帧内宏块预测模式编码的宏块的预测可以使用帧间宏块预测模式编码的相邻宏块的解码样值。constrained_intra_pred_flag 等于 1 表示受限制的帧内预测，在这种情况下，使用帧内宏块预测模式编码的宏块的预测仅使用残余数据和来自 I 或 SI 宏块类型的解码样值。

redundant_pic_cnt_present_flag 等于 0 表示 redundant_pic_cnt 语法元素不会在条带头、图像参数集中指明（直接或与相应的数据分割块 A 关联）的数据分割块 B 和数据分割块 C 中出现。redundant_pic_cnt_present_flag 等于 1 表示 redundant_pic_cnt 语法元素将出现在条带头、图像参数集中指明（直接或与相应的数据分割块 A 关联）的数据分割块 B 和数据分割块 C 中。

transform_8x8_mode_flag 等于 1 表示 8x8 变换解码过程可能正在使用（参见 8.5 节）。transform_8x8_mode_flag 等于 0 表示未使用 8x8 变换解码过程。当 transform_8x8_mode_flag 不存在时，默认其值为 0。

pic_scaling_matrix_present_flag 等于 1 表示存在用来修改在序列参数集中指定的缩放比例列表的参数。pic_scaling_matrix_present_flag 等于 0 表示用于该图像中的缩放比例列表应等于那些由序列参数集规定的。当 pic_scaling_matrix_present_flag 不存在时，默认其值为 0。

pic_scaling_list_present_flag[i **]** 等于 1 表示存在缩放比例列表的语法结构并用于指定序号为 i 的缩放比例列表。pic_scaling_list_present_flag[i]等于 0 表示在图像参数集中不存在缩放比例列表 i 的语法结构，并且根据 seq_scaling_matrix_present_flag 的值，应用下列条款：

–— 如果 seq_scaling_matrix_present_flag 等于0，表7-2中指定的缩放比例列表后退规则集A应用于获取序号为i的图像级缩放比例列表。

–— 否则（seq_scaling_matrix_present_flag 等于1），表7-2中指定的缩放比例列表后退规则集B应用来获取序号为i的图像级缩放比例列表。

second_chroma_qp_index_offset 表示为在 QP_C 值的表格中寻找 Cr 色度分量而应加到参数 QP_Y 和 QS_Y 上的偏移。second_chroma_qp_index_offset 的值应在-12 到 +12 范围内（包括边界值）。

当 second_chroma_qp_index_offset 不存在时，默认其值等于 chroma_qp_index_offset。

~~0.1.1.3~~7.4.2.3 补充增强信息RBSP语义

补充增强信息（SEI）包括的信息在解码来自 VCl NAL 单元的编码图像的样点时不是必要的。

~~0.1.1.3.1~~7.4.2.3.1 补充增强信息消息语义

一个 SEI NAL 单元包括一个或多个 SEI 消息。每个 SEI 消息由表示 SEI 载荷类型的 payloadType 和表示 SEI 载荷大小的 payloadSize 变量组成。SEI 载荷在附录 D 中规定。表示 SEI 载荷大小的 payloadSize 以字节为单位，并且等于 SEI 载荷的字节数。

ff_byte 是等于 0xFF 的一个字节，用来识别在其所使用的语法结构中是否需要一种更长的表示法。

last_payload_type_byte 是一个 SEI 消息的载荷类型的最后一个字节。

last_payload_size_byte 是一个 SEI 消息大小的最后一个字节。

~~0.1.1.4~~7.4.2.4 访问单元分隔符RBSP语义

访问单元分隔符将用于识别一个基本编码图像中存在的条带的类型以及简化访问单元之间边界的检测。没有与访问单元分隔符相关的标准化的解码过程。

primary_pic_type 表示基本编码图像所有条带的 slice_type 值都是表 7-5 列出的组的元素，并对应给出的 primary_pic_type 值。

表 7-5－primary_pic_type的含义

primary_pic_type	可能会出现在基本编码图像中的 slice_type 值
0	I
1	I, P
2	I, P, B
3	SI
4	SI, SP
5	I, SI
6	I, SI, P, SP
7	I, SI, P, SP, B

~~0.1.1.5~~7.4.2.5 序列结尾RBSP语义

视频序列结尾 RBSP 表示按解码顺序（如果存在）在比特流中下一个紧跟的访问单元将是一个 IDR 访问单元。视频序列 RBSP 结尾的 SODB 和 RBSP 的内容为空。没有规定一个视频序列 RBSP 结尾的标准化解码过程。

~~0.1.1.6~~7.4.2.6 流结尾RBSP语义

流结尾的 RBSP 表示按解码顺序，在流结尾的 RBSP 之后没有任何其他的 NAL 单元。在流结尾 RBSP 的 SODB 和 RBSP 的内容为空。没有规定一个流结尾 RBSP 的标准化解码过程。

~~0.1.1.7~~7.4.2.7 填充数据RBSP语义

填充数据 RBSP 包括那些值等于 0xFF 的字节。没有为一个填充数据 RBSP 规定标准化的解码过程。

ff_byte 是一个等于 0xFF 的字节。

~~0.1.1.8~~7.4.2.8 未分割的条带层RBSP语义

没有对 RBSP 分割的条带层由一个条带头和条带数据组成。

~~0.1.1.9~~7.4.2.9 条带数据分割的RBSP语义

~~0.1.1.9.1~~7.4.2.9.1 条带数据分割块A的RBSP语义

当使用条带数据分割块时，单个条带的编码数据将被分成三个独立的分割块。分割块 A 包括类型 2 的所有语法元素。

类型 2 的语法元素包括在条带头和条带数据语法结构中的所有语法元素，除了在 residual( )语法结构中的语法元素。

slice_id 标识与数据分割有关的条带。每个条带在包含该条带的编码图像中应具有一个唯一的 slice_id。当如附录 A 中规定那样不允许任意条带顺序时，按解码顺序，一个编码图像的第一个条带的 slice_id 值应等于 0，并且该编码图像的后续每个条带的 slice_id 值应逐 1 递增。

slice_id 的取值范围规定如下：

– 如果 MbaffFrameFlag 等于0， slice_id 的取值范围应该在0到PicSizeInMbs – 1内（包括边界值）。

– 否则（MbaffFrameFlag 等于1）， slice_id 的取值范围应在0 到 PicSizeInMbs / 2 – 1内（包括边界值）。

~~0.1.1.9.2~~7.4.2.9.2 条带数据分割块B的RBSP语义

当使用条带数据分割块时，单个条带的编码数据将被分成 1 到 3 个独立的分割块。条带数据分割块 B 包括类型 3 的所有语法元素。

类型 3 语法元素包括语法结构 residual( )以及那些用于宏块类型 I 和 SI（如表 7-10 所示）的语法结构中的所有语法元素。

slice_id 具有在 7.4.2.9.1 节规定的一样的语义。

redundant_pic_cnt 对于那些属于基本编码图像的条带和条带数据分割块应等于 0。在冗余编码图像中的编码条带和编码条带数据分割块的 redundant_pic_cnt 的值应大于 0。当 redundant_pic_cnt 不存在时，默认其值为 0。redundant_pic_cnt 的值应该在 0 到 127 范围内（包括 0 和 127）。

条带数据分割块 B 的 RBSP 是否存在规定如下：

– 如果一个条带数据分割块A的RBSP的语法元素表示在一个条带的条带数据类型3的任何语法元素的存在，一个条带数据分割块B的RBSP应存在并与条带数据分割块A的RBSP具有相同的slice_id 和redundant_pic_cnt 的值。

– 否则（一个条带数据分割块A的RBSP的语法元素不表示在一个条带的条带数据类型3的任何语法元素的存在），与条带数据分割块A的RBSP具有相同的slice_id 和redundant_pic_cnt 的值的条带数据分割块B的RBSP应不存在。

~~0.1.1.9.3~~7.4.2.9.3 条带数据分割块C的RBSP语义

当使用条带数据分割块时，单个条带的编码数据将被分成 3 个独立的分割块。条带数据分割块 C 包括类型 4 的所有语法元素。

类型 4 语法元素包括语法结构 residual( )以及那些用于宏块类型 P 和 B（如表 7-10 所示）的语法结构中的所有语法元素。

slice_id 具有在 7.4.2.9.1 节规定的一样的语义。

redundant_pic_cnt 具有在 7.4.2.9.2 节规定的一样的语义。

关于条带数据分割块 C 的 RBSP 是否存在的规定如下：

– 如果一个条带数据分割块A的RBSP的语法元素表示在一个条带的条带数据类型4的任何语法元素的存在，一个条带数据分割块C的RBSP应存在并与条带数据分割块A的RBSP具有相同的slice_id 和redundant_pic_cnt 的值。

– 否则（一个条带数据分割块A的RBSP的语法元素不表示在一个条带的条带数据类型4的任何语法元素的存在），与条带数据分割块A的RBSP具有相同的slice_id 和redundant_pic_cnt 的值的条带数据分割块C的RBSP应不存在。

~~0.1.1.10~~7.4.2.10 RBSP条带尾比特的语义

cabac_zero_word 是两个等于 0x0000 的字节的排列。

规定 NumBytesInVclNALunits 作为一个编码图像所有 VCL NAL 单元的 NumBytesInNALunit 值的总和。

规定 BinCountsInNALunits 作为 9.3.3.2 节规定的解析过程函数 DecodeBin( )被调用来解码一个编码图像中所有 VCL NAL 单元的内容的次数。当 entropy_coding_mode_flag 等于 1 时，BinCountsInNALunits 不应超过 (32 ÷ 3) * NumBytesInVclNALunits + (RawMbBits * PicSizeInMbs) ÷ 32。

注 — 解码条带层NAL单元的内容得到的分组结果的最大数的限制，可以通过插入一个cabac_zero_word语法元素来增加NumBytesInVclNALunits的值得到满足。一个NAL单元的每个cabac_zero_word都由三字节的序列0x000003代表（作为对NAL单元内容限制规定的结果，该规定要求每个cabac_zero_word包括一个emulation_prevention_three_byte）。

~~0.1.1.11~~7.4.2.11 RBSP尾比特语义

rbsp_stop_one_bit 应等于 1。

rbsp_alignment_zero_bit 应等于 0。

~~0.1.2~~7.4.3 条带头语义

如果存在，条带头语法元素 pic_parameter_set_id、 frame_num、 field_pic_flag、bottom_field_flag、idr_pic_id、 pic_order_cnt_lsb、 delta_pic_order_cnt_bottom、delta_pic_order_cnt[0]、delta_pic_order_cnt[1]、sp_for_switch_flag 和 slice_group_change_cycle 的值在一个编码图像的所有条带头中都应一样。

first_mb_in_slice 表示在条带中第一个宏块的地址。当如附件 A 中规定的那样不允许任意的条带顺序时，本条带的 first_mb_in_slice 的值应不小于当前图像的任何在该条带之前（按解码顺序）的其他条带的 first_mb_in_slice 的值。

条带中第一个宏块的地址按如下方式得到：

–— 如果 MbaffFrameFlag 等于0， first_mb_in_slice就是该条带中第一个宏块的地址，并且first_mb_in_slice 的值应在0到 PicSizeInMbs – 1的范围内（包括边界值）。

–— 否则（MbaffFrameFlag 等于1），first_mb_in_slice * 2 就是该条带中的第一个宏块地址，该宏块是该条带中第一个宏块对中的顶宏块，并且first_mb_in_slice 的值应该在 0 到 PicSizeInMbs / 2 – 1的范围内（包括边界值）。

slice_type **表示条带的编码类型，如表 7-6 所示：**

表 7-6－slice_type的名称关联

slice_type	slice_type 的名称
0	P (P 条带)
1	B (B 条带)
2	I (I 条带)
3	SP (SP 条带)
4	SI (SI 条带)
5	P (P 条带)
6	B (B 条带)
7	I (I 条带)
8	SP (SP 条带)
9	SI (SI 条带)

slice_type 的值在 5 到 9 范围内表示，除了当前条带的编码类型，所有当前编码图像的其他条带的 slice_type 值应与当前条带的 slice_type 值 一样，或者等于当前条带的 slice_type 值减 5。

当 nal_unit_type 等于 5（IDR 图像）时， slice_type 应等于 2、 4、 7 或 9。

当 num_ref_frames 等于 0 时， slice_type 应等于 2、 4、 7 或 9。

pic_parameter_set_id 指定使用的图像参数集。pic_parameter_set_id 的值应该在 0 到 255 范围内（包括 0 和 255）。

frame_num 用作一个图像标识符，在比特流中应由 log2_max_frame_num_minus4 + 4 个比特表示。关于 frame_num 的规定如下：

变量 PrevRefFrameNum 按如下方式得到：

-— 如果当前的图像是一个 IDR 图像， PrevRefFrameNum 将设为 0。

-— 否则（当前图像不是一个 IDR 图像），PrevRefFrameNum 设置如下：

- — 如果对包含一个非参考图像，且按解码顺序其前面一个访问单元包含一个参考图像的访问单元解码的过程调用 8.2.5.2 节规定的 frame_num 出现中断的解码过程，PrevRefFrameNum 应设为等于最后一个“不存在的”，由 8.2.5.2 节规定的 frame_num 出现中断的解码过程推定的参考帧的 frame_num 值。
- — 否则 PrevRefFrameNum 设为等于按解码过程包含参考图像的前一个访问单元的 frame_num 值。

frame_num 的取值规定如下：

-— 如果当前图像是一个IDR图像， frame_num 应等于0。

-— 否则（当前图像不是一个IDR图像），则按解码顺序前一个访问单元包含一个参考图像作为前导参考图像，该访问单元中有基本编码图像，当前图像的frame_num值不应等于PrevRefFrameNum ，除非满足下面所有三个条件：

- — 按解码顺序当前图像和前导参考图像属于连续的访问单元。
- — 当前图像和前导参考图像是具有相反奇偶性的参考场。
- — 满足下面的一个或多个条件：
  - — 前导参考图像是一个 IDR 图像
  - — 前导参考图像包括一个值等于 5 的 memory_management_control_语法元素

    注 1 — 当前导参考图像包括一个memory_management_control_operation 语法元素等于5时，PrevRefFrameNum 应等于0。
  - — 在前导参考图像之前存在一个基本编码图像，并且该基本编码图像的 frame_num 不等于 PrevRefFrameNum。
  - — 在前导参考图像之前存在一个基本编码图像，并且该基本编码图像不是一个参考图像。

当 frame_num 的值不等于 PrevRefFrameNum 时，下列几条将被应用：

— 按解码顺序，不应有任何前面的场或帧当前被标记为“用于短期参考”并具有一个frame_num 值等于由变量UnusedShortTermFrameNum按下列公式得出的任何值：

UnusedShortTermFrameNum = (PrevRefFrameNum + 1) % MaxFrameNumwhile(UnusedShortTermFrameNum != frame_num) (7-21)

UnusedShortTermFrameNum = (UnusedShortTermFrameNum + 1) % MaxFrameNum

— frame_num 的值规定如下：

  — 如果 gaps_in_frame_num_value_allowed_flag 等于 0，当前图像的 frame_num 值将等于 (PrevRefFrameNum + 1) % MaxFrameNum。

  — 否则（gaps_in_frame_num_value_allowed_flag 等于 1），下面几条将应用：

    — 如果 frame_num 大于 PrevRefFrameNum，按解码顺序比特流中在前面参考图像之后应不会有任何非参考图像在当前图像之前，且此段解码序列满足下面两个条件。

      — 非参考图像的 frame_num 值小于 PrevRefFrameNum。

      — 非参考图像的 frame_num 值大于当前图像的 frame_num 值。

— 否则（frame_num 小于 PrevRefFrameNum），按解码顺序比特流中在前面参考图像之后应不会有任何非参考图像在当前图像之前，且此段解码序列满足下面两个条件。

— 非参考图像的 frame_num 的值小于 PrevRefFrameNum。

— 非参考图像的 frame_num 值大于当前图像的 frame_num 值。

包含语法元素 memory_management_control_operation 值等于 5 的一个图像其 frame_num 按上述规定，并且在解码当前图像和存储管理控制操作处理之后，该图像在解码过程中的后续所有应用都将推定为 frame_num 等于 0，除非如 7.4.1.2.4 中的规定。

注 2 — 当基本编码图像不是一个IDR图像并且不包含值等于5的memory_management_control_operation 语法元素时，相应的冗余编码图像的frame_num值与前面一个基本编码图像的frame_num值相同。冗余编码图像包含的 memory_management_control_operation 语法元素值等于5，或者相应的基本编码图像是一个IDR图像，两者必居其一。

field_pic_flag 等于 1 表示该条带是一个编码场的条带。field_pic_flag 等于 0 表示该条带是一个编码帧的条带。当 field_pic_flag 不存在时，应推定其值为 0。

变量 MbaffFrameFlag 由下列公式得到：

MbaffFrameFlag = (mb_adaptive_frame_field_flag && !field_pic_flag) (7-22)

以宏块为单位的图像高度的变量由下列公式得出：

PicHeightInMbs = FrameHeightInMbs / (1 + field_pic_flag) (7-23)

用于亮度分量的图像高度变量由下列公式得出：

$PicHeightInSamples_L = PicHeightInMbs * 16$ (7-24)

用于色度分量的图像高度变量由下列公式得出：

$PicHeightInSamples_C = PicHeightInMbs * MbHeightC$ (7-25)

当前图像的 PicSizeInMbs 变量由下列公式得出：

PicSizeInMbs = PicWidthInMbs * PicHeightInMbs (7-26)

变量 MaxPicNum 由下列公式得出：

- 如果 field_pic_flag 等于 0，MaxPicNum 设为等于 MaxFrameNum。
- 否则（field_pic_flag 等于 1），MaxPicNum 设为等于 2*MaxFrameNum。

变量 CurrPicNum 由下列公式得出：

- 如果 field_pic_flag 等于 0，CurrPicNum 设为等于 frame_num。
- 否则（field_pic_flag 等于 1），CurrPicNum 设为等于 2 * frame_num + 1。

bottom_field_flag 等于 1 表示该条带是一个编码底场的一部分。bottom_field_flag 等于 0 表示该图像是一个编码的顶场。当该条带不存在此语法元素时，应推定其值为 0。

idr_pic_id 标识一个 IDR 图像。一个 IDR 图像的所有条带中的 idr_pic_id 值应保持不变。当按解码顺序的两个连续访问单元都是 IDR 访问单元时，第一个 IDR 访问单元的条带的 idr_pic_id 值应与第二个 IDR 访问单元的 idr_pic_id 值不同。idr_pic_id 的值应在 0 到 65535 的范围内（包括 0 和 65535）。

pic_order_cnt_lsb 表示一个编码帧的顶场或一个编码场的图像顺序数对 MaxPicOrderCntLsb 取模。pic_order_cnt_lsb 语法元素的大小是 log2_max_pic_order_cnt_lsb_minus4 + 4 个比特。pic_order_cnt_lsb 的值应该在 0 到 MaxPicOrderCntLsb – 1 的范围内（包括边界值）。

delta_pic_order_cnt_bottom 表示一个编码帧的底场和顶场的图像顺序数之间的差，具体如下：

— 如果当前图像包含一个memory_management_control_operation等于5， delta_pic_order_cnt_bottom 的值应在 (1 – MaxPicOrderCntLsb) 到 $2^{31} – 1$范围内（包括边界值）。

— 否则（当前图像不包括一个memory_management_control_operation等于5），delta_pic_order_cnt_bottom 的值应该在$–2^{31}$ 到 $2^{31} – 1$的范围内（包括边界值）。

当当前条带的比特流中不存在该语法元素时，应推定其值为 0。

delta_pic_order_cnt[0 **]** 表示图像顺序数与 8.2.1 节规定的一个编码帧的顶场或一个编码场的预期图像顺序数之间的差异。delta_pic_order_cnt[0] 的值应该在 $– 2^{31}$ 到 $2^{31} – 1$ 的范围内（包括边界值）。当该语法元素在当前条带的比特流中不存在时，应推定其值为 0。

delta_pic_order_cnt[1 **]** 表示图像顺序数与 8.2.1 节规定的一个编码帧的底场的预期图像顺序数之间的差异。delta_pic_order_cnt[1] 的值应该在 $– 2^{31}$ 到 $2^{31} – 1$ 的范围内（包括边界值）。当该语法元素在当前条带的比特流中不存在时，应推定其值为 0。

redundant_pic_cnt 对于属于基本编码图像的条带和条带数据隔离带应等于 0。对于一个冗余编码图像的编码条带或编码条带数据隔离带的 redundant_pic_cnt 的值应大于 0。当 redundant_pic_cnt 在比特流中不存在时，应推定其值为 0。redundant_pic_cnt 的值应该在 0 到 127 范围内（包括 0 和 127）。

注 3 — 解码基本图像的任何区域以及第8节中规定的任何相同访问单元的冗余图像的解码过程中得到的相应的区域从外观上看应该是相似的。

一个冗余编码图像的编码条带或编码条带数据隔离带的 pic_parameter_set_id 值应该使得在一个冗余编码图像中使用的图像参数集里的 pic_order_present_flag 值等于相应基本编码图像中使用的图像参数集中的 pic_order_present_flag 的值。

当在基本编码图像以及任何冗余编码图像中出现时，下列语法元素应具有同样的值：field_pic_flag、bottom_field_flag 和 idr_pic_id。

当在一个访问单元中的 VCL NAL 单元的 nal_ref_idc 值等于 0 时，同一个访问单元的所有其他 VCL NAL 单元的 nal_ref_idc 值都应等于 0。

注 4 — 上述规定同时具有下列含义。如果基本编码图像的VCL NAL单元的nal_ref_idc值等于0，那么任何相应的冗余编码图像的VCL NAL单元的nal_ref_idc值都等于0；否则（基本编码图像的VCL NAL单元的nal_ref_idc值大于0），任何相应的冗余编码图像的VCL NAL单元的nal_ref_idc值也都大于0。

如 8.2.5 节的规定，在同一个访问单元的基本编码图像或任何冗余编码图像中调用解码参考图像标记过程之后，参考图像的标记状态和 frame_num 的值应一致，而不管访问单元的基本编码图像或任何冗余编码图像（代替基本编码图像）是否被解码。

注 5 — 上述规定同时具有下列含义：

如果一个基本编码图像不是一个IDR图像，dec_ref_pic_marking( )语法结构的内容在基本编码图像和对应基本编码图像的所有冗余编码图像的所有条带头中必须一致。

否则（一个基本编码图像是一个IDR图像），下面几条将应用：

如果根据基本编码图像得来的一个冗余编码图像是一个IDR图像，dec_ref_pic_marking( ) 语法结构的内容在基本编码图像和对应基本编码图像的冗余编码图像的所有条带头中必须一致。

否则（基本编码图像相应的一个冗余编码图像不是一个IDR图像），该冗余图像的所有条带头必须包括一个包含memory_management_control_operation语法元素等于5的 dec_ref_pic_marking ( )语法结构，并且下面几条将适用：

如果基本编码图像中long_term_reference_flag的值等于0，冗余编码图像的dec_ref_pic_marking语法结构必须不能包括值等于6的 memory_management_control_operation语法元素。

否则（基本编码图像中long_term_reference_flag的值等于1），冗余编码图像的dec_ref_pic_marking语法结构必须包括值等于 5、4 和 6（按解码顺序）的 memory_management_control_operation 的语法元素，并且 max_long_term_frame_idx_plus1的值必须等于1，而且long_term_frame_idx的值必须等于0。

同一个访问单元的任何冗余编码图像或基本编码图像的解码过程完成后得到的 TopFieldOrderCnt 和 BottomFieldOrderCnt（如果可用）的值应该一样，不管访问单元的基本编码图像或任何冗余编码图像（代替基本编码图像）是否被解码。

一个冗余编码图像的编码条带或编码条带数据隔离带的解码过程不是必需的。当一个编码条带的条带头中的redundant_pic_cnt 大于 0 时，解码器应丢弃该编码条带。然而任何冗余编码图像的一个编码条带或编码条带数据隔离带应遵从与一个基本编码图像的编码条带或编码条带数据隔离带一样的规定。

注 6 — 当序列由于在传输过程中出现错误或丢失导致解码基本图像中的一些样点不能被正确解码而一个编码的冗余条带能够被正确解码时，解码器将用解码冗余条带的相关样点代替解码初始图像的样点。当多个冗余条带覆盖初始图像的相关区域时，冗余条带将使用redundant_pic_cnt的最小值。

具有同样的 redundant_pic_cnt 值的冗余条带和条带数据划分属于同一个冗余图像。在同一个冗余图像中的解码条带不需要覆盖整个图像区域而且不能交迭。

direct_spatial_mv_pred_flag 表示为得到帧间预测的动作矢量和参考序号而使用的方法，具体如下：

-— 如果direct_spatial_mv_pred_flag 等于1，8.4.1.2节给出的亮度运动矢量B_Skip、B_Direct_16x16和B_Direct_8x8将使用空间指引的模式来预期，参见8.4.1.2.2节。

-— 否则（direct_spatial_mv_pred_flag 等于 0），8.4.1.2节给出的亮度运动矢量B_Skip、B_Direct_16x16和B_Direct_8x8的获取过程将使用临时指引模式来预期，参见8.4.1.2.3节。

num_ref_idx_active_override_flag 等于 0 表示在提到的图像参数集中规定的语法元素 num_ref_idx_l0_active_minus1 和 num_ref_idx_l1_active_minus1 的值将生效。num_ref_idx_active_override_flag 等于 1 表示在提到的图像参数集中规定的语法元素 num_ref_idx_l0_active_minus1 和 num_ref_idx_l1_active_minus1 在条带头中超过当前条带（且仅指当前条带）的值如下：

当当前条带是一个 P、SP 或 B 条带并且 field_pic_flag 等于 0 而且图像参数集中的 num_ref_idx_l0_active_minus1 的值超过 15 时，num_ref_idx_active_override_flag 将等于 1。

当当前条带是一个 B 条带并且 field_pic_flag 等于 0 而且图像参数集中的 num_ref_idx_l1_active_minus1 的值超过 15 时，num_ref_idx_active_override_flag 将等于 1。

num_ref_idx_l0_active_minus1 表示将用于解码该条带的参考图像列表 0 的最大参考序号。

num_ref_idx_l0_active_minus1 的取值范围规定如下：

–— 如果field_pic_flag 等于0，num_ref_idx_l0_active_minus1的值将在0到15的范围内（包括0和15）。当MbaffFrameFlag 等于 1 时，num_ref_idx_l0_active_minus1 就是解码帧宏块的最大序号值，而 2 * num_ref_idx_l0_active_minus1 + 1是解码场宏块的最大序号值。

–— 否则（field_pic_flag 等于1），num_ref_idx_l0_active_minus1 的值应该在0到31的范围内（包括0和31）。

num_ref_idx_l1_active_minus1 具有与 num_ref_idx_l0_active_minus1 同样的语义，只是分别用 11 和列表 1 代替 10 和列表 1。

cabac_init_idc 表示用于决定关联变量的初始化过程中使用的初始化表格的序号。变量 cabac_init_idc 的值应该在 0 到 2 的范围内（包括 0 和 2）。

slice_qp_delta 表示用于条带中的所有宏块的 QP_Y 的初始值，该值在宏块层将被 mb_qp_delta 的值修改。该条带初始 QP_Y 量化参数按下面的公式计算：

$$SliceQP_Y = 26 + pic\_init\_qp\_minus26 + slice\_qp\_delta \quad (7\text{-}27)$$

slice_qp_delta 应该受限，这样 $SliceQP_Y$ 的值将在$-QpBdOffset_Y$ 到 +51 的范围内（包括边界值）。

sp_for_switch_flag 表示用来解码 SP 条带中的 P 宏块的解码过程，具体如下：

-— 如果sp_for_switch_flag 等于0，对于在8.6.1节中提到的非交换图像SP条带中的P宏块将使用SP解码过程解码。

-— 否则（sp_for_switch_flag等于1），对于在8.6.2节中提到的交换图像SP条带中的P宏块将使用SP和SI解码过程解码。

slice_qs_delta 表示 SP 和 SI 条带中所有宏块的 QS_Y 值。该条带的 QS_Y 量化参数按下面的公式计算：

$$QS_Y = 26 + pic\_init\_qs\_minus26 + slice\_qs\_delta \quad (7\text{-}28)$$

slice_qs_delta 的值应该首先限制使 QS_Y 的值在 0 到 51 的范围内（包括 0 和 51）。QS_Y 的值用于解码 mb_type 等 SI 的 SI 条带的所有宏块以及预测模式为帧间的 SP 条带的所有宏块。

disable_deblocking_filter_idc 表示去块效应滤波器的操作在经过条带的一些块边缘时是否会被废弃，并指定该滤波器针对哪个边缘被废弃。当条带头中不存在 disable_deblocking_filter_idc 时，其值默认为 0。

disable_deblocking_filter_idc 的值应该在 0 到 2 范围内（包括 0 和 2）。

slice_alpha_c0_offset_div2 表示访问 α 和 t_{C0} 去块效应滤波器表格来滤波条带中的宏块所控制的操作使用的偏移。对于这个含义，当在这些表格中寻址时应使用该偏移，该偏移可按下面的公式计算：

$$FilterOffsetA = slice\_alpha\_c0\_offset\_div2 << 1 \quad (7\text{-}29)$$

slice_alpha_c0_offset_div2 的值应该在-6 到 +6 范围内（包括边界值）。当条带头中不存在 slice_alpha_c0_offset_div2 时，默认其值为 0。

slice_beta_offset_div2 表示访问 β 去块效应滤波器表格来滤波带中的宏块所控制的操作使用的偏移。对于这个含义，当在 β 表格中寻址时应使用该偏移，该偏移可按下面的公式计算：

$$FilterOffsetB = slice\_beta\_offset\_div2 << 1 \quad (7\text{-}30)$$

slice_beta_offset_div2 的值应该在-6 到 +6 范围内（包括边界值）。当条带头中不存在 slice_beta_offset_div2 时，默认其值为 0。

slice_group_change_cycle 用来在 slice_group_map_type 等于 3、4 或 5 时得到条带组 0 中的条带组映射单元的数目，计算公式如下：

$$MapUnitsInSliceGroup0 = Min(slice\_group\_change\_cycle * SliceGroupChangeRate, PicSizeInMapUnits) \quad (7\text{-}31)$$

slice_group_change_cycle 的值在比特流中由比特数来表示。

$$Ceil(Log2(PicSizeInMapUnits \div SliceGroupChangeRate + 1)) \quad (7\text{-}32)$$

slice_group_change_cycle 的值应该在 0 到 Ceil(PicSizeInMapUnits÷SliceGroupChangeRate)的范围内（包括边界值）。

~~0.1.2.1~~7.4.3.1 参考图像列表重新排序语义

语法元素 reordering_of_pic_nums_idc、abs_diff_pic_num_minus1 和 long_term_pic_num 表示从初始参考图像列表到用于解码该条带的参考图像列表的改变。

ref_pic_list_reordering_flag_l0 等于 1 表示语法元素 reordering_of_pic_nums_idc 存在用来表示参考图像列表 0。ref_pic_list_reordering_flag_l0 等于 0 表示该语法元素不存在。

当 ref_pic_list_reordering_flag_l0 等于 1 时， reordering_of_pic_nums_idc 的次数不等于 3 而跟随的 ref_pic_list_reordering_flag_l0 不应超过 num_ref_idx_l0_active_minus1 + 1。

当如 8.2.4.2 节规定的那样产生初始参考图像列表中的 RefPicList0[num_ref_idx_l0_active_minus1]等于“无参考图像”时， ref_pic_list_reordering_flag_l0 应等于 1 而 reordering_of_pic_nums_idc 不应等于 3，直到在像 8.2.4.3 节规定的那样产生的重新排序列表中的 RefPicList0[num_ref_idx_l0_active_minus1] 不等于“无参考图像”。

ref_pic_list_reordering_flag_l1 等于 1 表示语法元素 reordering_of_pic_nums_idc 存在用来表示参考图像列表 1. ref_pic_list_reordering_flag_l1 等于 0 表示该语法元素不存在。

当 ref_pic_list_reordering_flag_l1 等于 1 时， reordering_of_pic_nums_idc 的次数不等于 3，而跟随的 ref_pic_list_reordering_flag_l1 不应超过 num_ref_idx_l1_active_minus1 + 1。

当在解码一个 B 条带并且如 8.2.4.2 节规定的那样产生初始参考图像列表中的 RefPicList0[num_ref_idx_l1_active_minus1]等于“无参考图像”时，ref_pic_list_reordering_flag_l1 应等于 1 而 reordering_of_pic_nums_idc 不应等于 3，直到在像 8.2.4.3 节规定的那样产生的重新排序列表中的 RefPicList1[num_ref_idx_l1_active_minus1] 不等于“无参考图像”。

reordering_of_pic_nums_idc 与 abs_diff_pic_num_minus1 或 long_term_pic_num 一起表示参考图像中的哪一个被重新映射。reordering_of_pic_nums_idc 的值在表 7-7 中给出。直接跟随在 ref_pic_list_reordering_flag_l0 或 ref_pic_list_reordering_flag_l1 之后的第一个 reordering_of_pic_nums_idc 的值不应等于 3。

表 7-7－用来重新排序参考图像列表的reordering_of_pic_nums_idc 操作

reordering_of_pic_nums_idc	表示的重新排序
0	abs_diff_pic_num_minus1 存在并且相当于从一个图像数的预期值减去一个差值
1	abs_diff_pic_num_minus1 存在并相当于对一个图像数的预期值加上一个差值
2	long_term_pic_num 存在并表示一个参考图像的长期图像编号
3	初始参考图像列表重新排序结束循环

abs_diff_pic_num_minus1 加 1 表示被移动到当前列表中序号对应图像的图像数与图像数预期值的绝对差值。abs_diff_pic_num_minus1 的值应该在 0 到 MaxPicNum – 1 的范围内。abs_diff_pic_num_minus1 的允许取值范围在 8.2.4.3.1 节中进一步限制。

long_term_pic_num 表示被移动到当前列表中序号对应图像的长期图像编号。当解码一个编码帧时，long_term_pic_num 应等于分配给标记为“用于长期参考”的参考帧或附加参考场对当中的一个的 LongTermPicNum 。当解码一个编码场时，long_term_pic_num 应等于分配给标记为“用于长期参考”的参考场中的一个的 LongTermPicNum。

~~0.1.2.2~~7.4.3.2 预测加权表语义

luma_log2_weight_denom 是所有亮度加权因子中分母的 2 的对数。luma_log2_weight_denom 的值应该在 0 到 7 的范围内（包括 0 和 7）。

chroma_log2_weight_denom 是所有色度加权因子中分母的 2 的对数。chroma_log2_weight_denom 的值应该在 0 到 7 的范围内（包括 0 和 7）。

luma_weight_l0_flag 等于 1 表示存在列表 0 预期的亮度成分中的加权因子。luma_weight_l0_flag 等于 0 表示不存在这些加权因子。

luma_weight_l0[i **]** 是用于亮度预期值的加权因子，该亮度预期值是使用 RefPicList0[i]的列表 0 预期中的亮度预期值。当 luma_weight_l0_flag 等于 1 时， luma_weight_l0[i] 的值应该在–128 到 127 的范围内（包括边界值）。当 luma_weight_l0_flag 等于 0 时，可推断对于 RefPicList0[i] 的 luma_weight_l0[i] 等于 $2^{luma\_log2\_weight\_denom}$。

luma_offset_l0[i **]** 是亮度预期值的附加偏移，该亮度预期值是使用 RefPicList0[i]的列表 0 预期中的亮度预期值。luma_offset_l0[i] 的值应该在–128 到 127 的范围内。当 luma_weight_l0_flag 等于 0 时，可推断对于 RefPicList0[i]的 luma_offset_l0[i]也等于 0。

chroma_weight_l0_flag 等于 1 表示存在列表 0 预期的色度预期值的加权因子。chroma_weight_l0_flag 等于 0 表示不存在这些加权因子。

chroma_weight_l0[i][j] 是用于色度预期值的加权因子，当中 Cb 的 j 等于 0 而 Cr 的 j 等于 1，该色度预期值是使用 RefPicList0[i] 的列表 0 预期中的色度预期值。当 chroma_weight_l0_flag 等于 1 时，chroma_weight_l0[i][j]的值应该在–128 到 127 的范围内（包括边界值）。当 chroma_weight_l0_flag 等于 0 时，可推断对于 RefPicList0[i]的 chroma_weight_l0[i][j] 等于 $2^{\text{chroma\_log2\_weight\_denom}}$。

chroma_offset_l0[i][j] 是用于色度预期值的附加偏移，当中 Cb 的 j 等于 0 而 Cr 的 j 等于 1，该色度预期值是使用 RefPicList0[i]的列表 0 预期中的色度预期值。chroma_offset_l0[i][j] 的值应该在-128 到 127 的范围内（包括边界值）。当 chroma_weight_l0_flag 等于 0 时，可推断对于 RefPicList0[i]的 chroma_offset_l0[i][j] 也等于 0。

luma_weight_l1_flag、**luma_weight_l1**、**luma_offset_l1**、**chroma_weight_l1_flag**，**chroma_weight_l1**、**chroma_offset_l1** 具有分别与 luma_weight_l0_flag、luma_weight_l0、luma_offset_l0、chroma_weight_l0_flag、chroma_weight_l0、chroma_offset_l0 一样的语义，只是用 11、列表 1 和 List1 分别代替 10、列表 0 和 List0。

~~0.1.2.3~~7.4.3.3 解码参考图像符号语义

语法元素 no_output_of_prior_pics_flag、long_term_reference_flag、adaptive_ref_pic_marking_mode_flag、memory_management_control_operation、difference_of_pic_nums_minus1、long_term_frame_idx、long_term_pic_num 和 max_long_term_frame_idx_plus1 表示参考图像的符号。

参考图像的符号可以是“未用于参考”、“用于短期参考”或“用于长期参考”，但是只能是这三个中的一个。当一个参考图像是指被标记为“用于参考”时，表示共同引用了被标记为“用于长期参考”或“用于短期参考”的图像（但不是同时引用两者）。一个被标记为“用于短期参考”的参考图像表示一个短期参考图像，被标记为“用于长期参考”的图像表示一个长期参考图像。

用于一个编码图像的所有编码条带的语法元素 adaptive_ref_pic_marking_mode_flag 和解码参考图像符号语法结构的内容应该是相同的。

解码参考图像符合语法结构的语法种类应该按下面几条推断：

-— 如果解码参考图像符号语法结构是一个条带头，该解码参考图像符号语法结构的语法种类应等于2。

-— 否则（该解码参考图像符号语法结构是在如附件D中指出的一个解码参考图像符号的副本SEI消息中），该解码参考图像符号语法结构的语法种类应等于5。

no_output_of_prior_pics_flag 表示在解码图像缓冲器中先前解码的图像在一个 IDR 图像解码之后是如何处理的。参见附件 C。但该 IDR 图像是比特流中第一个 IDR 图像时，no_output_of_prior_pics_flag 的值在解码过程中将不生效。当该 IDR 图像不是比特流中的第一个 IDR 图像并且从活动的视频序列参数集中得到的 PicWidthInMbs,、FrameHeightInMbs 或 max_dec_frame_buffering 的值与从先前序列活动时的视频序列参数集中得到的 PicWidthInMbs, 、FrameHeightInMbs 或 max_dec_frame_buffering 的值不同时，不管 no_output_of_prior_pics_flag 的实际值是什么，解码器可能会推断 no_output_of_prior_pics_flag 等于 1。

long_term_reference_flag 等于 0 表示变量 MaxLongTermFrameIdx 被设为等于“无长期帧序号”而且该 IDR 图像被标记为“用于短期参考”。long_term_reference_flag 等于 1 表示变量 MaxLongTermFrameIdx 被设为等于 0 而当前的 IDR 图像被标记为“用于长期参考”并且 LongTermFrameIdx 被分配为等于 0。当 num_ref_frames 等于 0 时，long_term_reference_flag 应该也等于 0。

adaptive_ref_pic_marking_mode_flag 选择当前解码图像的参考图像符号的模式如表 7-8 所示。当当前被标记为“用于长期参考”的帧数、附加场对和非成对的场等于 Max(num_ref_frames, 1) 时，adaptive_ref_pic_marking_mode_flag 将等于 1。

表 7-8－adaptive_ref_pic_marking_mode_flag的解释

adaptive_ref_pic_marking_mode_flag	指定的参考图像符号模式
0	变化窗口的参考图像符号模式：符号模式为短期参考图像提供一个先进先出的机制。
1	适应参考图像符号模式：参考图像符号模式提供语法元素来表示参考图像的符号为“未使用的参考”并分配长期帧序号。

memory_management_control_operation 表示一个用于影响参考图像符号的一个控制操作。语法元素 memory_management_control_operation 后面紧跟着是由 memory_management_control_operation 值指定的操作所必须的数据。与 memory_management_control_operation 相关的值和控制操作在表 7-9 中给出。语法元素 memory_management_control_operation 是通过按它们出现在条带头中的顺序进行的解码过程来处理的，为每个 memory_management_control_operation 表达的语义限制以每个 memory_management_control_operation 处理的顺序应用于特定的位置。

对于 memory_management_control_operation 的解释，术语参考图像解释如下：

— 如果当前图像是一个帧，属于参考图像就表示其或者是一个参考帧或者是一个附加参考场对。

— 否则（当前图像是一个场），术语参考图像表示或者是一个参考场或者是一个参考帧的场。

条带头中的 memory_management_control_operation 不应等于 1，除非当 memory_management_control_operation 被解码处理时，指定的参考图像被标记为“用于短期参考”。

条带头中的 memory_management_control_operation 不应等于 2，除非当 memory_management_control_operation 被解码处理时，指定的长期图像编号表示一个被标记为“用于长期参考”的参考图像。

条带头中的 memory_management_control_operation 不应等于 3，除非当 memory_management_control_operation 被解码处理时，指定的参考图像被标记为“用于短期参考”。

当 memory_management_control_operation 被解码处理时，如果变量 MaxLongTermFrameIdx 的值等于“非长期帧序号”，那么 memory_management_control_operation 不应等于 3 或 6。

一个条带头中不会有多于 1 个的 memory_management_control_operation 等于 4。

一个条带头中不会有多于 1 个的 memory_management_control_operation 等于 5。

一个条带头中不会有多于 1 个的 memory_management_control_operation 等于 6。

条带头中的 memory_management_control_operation 不应等于 5，除非同一个解码参考图像符号 语法结构中不存在值在 1 到 3 范围内的 memory_management_control_operation 。

同一个条带头中值为 6 的 memory_management_control_operation 后面不应跟随一个值为 5 的 memory_management_control_operation 。

当存在一个值为 6 的 memory_management_control_operation 时，任何同一个条带头中跟随在值为 6 的 memory_management_control_operation 之后的值为 2、3 或 4 的 memory_management_control_operation 不应表示当前的图像被标记为“未用于参考”。

注 1 — 这些规定禁止任何多个将表示当前图像被标记为“未用于参考”的memory_management_control_operation 语法元素组合。然而一些其他memory_management_control_operation语法元素的组合是允许的，这将不止一次的影响在同一个条带头中其他参考图像的符号状态。尤其是，允许memory_management_control_operation等于3表示将被分配给一个特定短期参考图像的一个长期帧序号，在同一个条带头中此值为3的 memory_management_control_operation 后面跟随一个值为2、3、4或6表示同一个参考图像随后被标记为“未用于参考”的memory_management_control_operation。

表 7-9－存储管理控制操作（memory_management_control_operation）的值

memory_management_control_operation	存储管理控制操作
0	结束 memory_management_control_operation 语法元素循环
1	标记一个短期参考图像为“未用于参考”
2	标记一个长期参考图像为“未用于参考”
3	标记一个短期参考图像为“用于长期参考”并为其分配一个长期帧序号
4	表示最大长期帧序号并标记所有具有大于最大值长期帧序号的长期参考图像为“未用于参考”
5	标记所有参考图像为“未用于参考”并设置变量 MaxLongTermFrameIdx 为“非长期帧序号”
6	标记当前图像为“用于长期参考”并为其分配一个长期帧序号

当解码一个场并且一个等于 3 的 memory_management_control_operation 命令存在来给作为一个短期参考帧的一部分或一个短期附加参考场对的一部分的场分配一个长期帧序号时，另一个 memory_management_control_operation 命令将存在于同一个解码参考图像符号语法结构中，并给同一个帧或附加参考场对的其他场分配同样的长期帧序号。

注 2 — 上述要求必须完成，甚至当由等于3的memory_management_control_operation 表示的场随后被标记为“未用于参考”时（例如，当memory_management_control_operation等于2存在于同一个条带头中并导致该场被标记为“未用于参考”）也必须完成。

当一个附加参考场对中的第一个场（按解码顺序）包括一个等于 1 的 long_term_reference_flag 或一个等于 6 的 memory_management_control_operation 命令时，该用于附加参考场对的其他场的解码参考图像符号语法结构将包括一个等于 6 的 memory_management_control_operation 给其他场分配同样的长期帧序号。

注 3 — 上述要求必须完成，甚至当附加参考场对的第一个场随后被标记为“未用于参考”时（例如，当一个等于2的memory_management_control_operation存在于第二个场的条带头中并导致第一个场被标记为“未用于参考”）也必须完成。

difference_of_pic_nums_minus1 用来（其中 memory_management_control_operation 等于 3 或 1）给一个短期参考图像分配一个长期帧序号或者标记一个短期参考图像为“未用于参考”。当相关的 memory_management_control_operation 被解码处理时，导致从 difference_of_pic_nums_minus1 获取的图像编号是分配给标记为“用于参考”的参考图像中的一个并且先前并没有分配一个长期帧序号的图像编号。

结果图像编号规定如下：

- — 如果field_pic_flag 等于0，结果图像编号将是分配给参考帧或附加参考场对的一组图像编号中的一个。

  注 4 — 当field_pic_flag等于0时，结果图像编号必须是分配给一个两个场都被标记为“用于参考”的附加参考场对或者一个两个场都被标记为“用于参考”的帧的一个图像编号。尤其是当field_pic_flag等于0时，一个非成对的场或者一个包括单个场被标记为“用于参考”的帧的符号不能被一个等于1的 memory_management_control_operation影响。

- — 否则（field_pic_flag等于1），结果图像编号将是分配给参考场的一组图像编号中的一个。

long_term_pic_num 是用于（其中 memory_management_control_operation 等于 2）将一个长期参考图像标记为“未用于参考”。当有关 memory_management_control_operation 被解码处理时，long_term_pic_num 将等于分配给当前被标记为“用于长期参考”的参考图像中的一个的长期图像编号。

结果长期图像编号规定如下：

— 如果field_pic_flag 等于0，结果长期图像编号将是分配给参考帧或附加参考场对的一组长期图像编号中的一个。

注 5 — 当field_pic_flag等于0时，结果长期图像编号必须是分配给一个两个场都被标记为“用于参考”的附加参考场对或者一个两个场都被标记为“用于参考”的帧的一个长期图像编号。尤其是当field_pic_flag等于0时，一个非成对的场或者一个包括单个场被标记为“用于参考”的帧的符号不能被一个等于2的memory_management_control_operation影响。

— 否则（field_pic_flag等于1），结果长期图像编号将是分配给参考场的一组长期图像编号中的一个。

long_term_frame_idx 是用于（其中 memory_management_control_operation 等于 3 或 6）给一个图像分配一个长期帧序号。当有关的 memory_management_control_operation 被解码处理时，long_term_frame_idx 的值应该在 0 到 MaxLongTermFrameIdx 范围内（包括边界值）。

max_long_term_frame_idx_plus1 减去 1 表示长期参考图像中允许的最大长期帧序号值（直到接收到另一个 max_long_term_frame_idx_plus1 值）。max_long_term_frame_idx_plus1 的值应该在 0 到 num_ref_frames 范围内（包括边界值）。

~~0.1.3~~7.4.4 条带数据语义

cabac_alignment_one_bit 是一个等于 1 的比特。

mb_skip_run 表示连续跳跃的宏块数，对于连续跳跃的宏块，当解码一个 P 或 SP 条带时，mb_type 应该是指 P_Skip 而该宏块的类型共同表示为一个 P 宏块类型；或者当解码一个 B 条带时，mb_type 应该是指 B_Skip 而该宏块的类型共同表示为一个 B 宏块类型。mb_skip_run 的值应该在 0 到 PicSizeInMbs – CurrMbAddr 范围内（包括边界值）。

mb_skip_flag 等于 1 表示对于当前宏块，当解码一个 P 或 SP 条带时，mb_type 应该是指 P_Skip 而该宏块的类型共同表示为一个 P 宏块类型；或者当解码一个 B 条带时，mb_type 应该是指 B_Skip 而该宏块的类型共同表示为一个 B 宏块类型。mb_skip_flag 等于 0 表示当前宏块不是跳跃的。

mb_field_decoding_flag 等于 0 表示当前宏块对是一个帧宏块对。mb_field_decoding_flag 等于 1 表示该宏块对是一个场宏块对。一个帧宏块对中的两个宏块都是指在文本中是作为帧宏块，然而一个场宏块对中的两个宏块是指在文本中是作为场宏块。

当一个宏块对中的每个宏块中都不存在 mb_field_decoding_flag 时， mb_field_decoding_flag 的值由下列方式得出：

— 如果同一个条带中有一个邻居宏块对直接位于当前宏块对的左侧，mb_field_decoding_flag 的值将等于直接位于当前宏块对左侧的邻居宏块对的mb_field_decoding_flag 值。

— 否则，如果同一个条带中没有一个邻居宏块对直接位于当前宏块对的左侧并且同一个条带中有一个邻居宏块对直接位于当前宏块对之上， mb_field_decoding_flag 的值将等于直接位于当前宏块对之上的邻居宏块对的mb_field_decoding_flag 值。

— 否则（在同一个条带中无论是在当前宏块对的左侧或上方都没有邻居宏块对），mb_field_decoding_flag 的值将等于0。

end_of_slice_flag 等于 0 表示该条带中跟随有另一个宏块。end_of_slice_flag 等于 1 表示条带的结尾并且后面不再有宏块跟随。

条带数据语法表中使用的 NextMbAddress( )功能在 8.2.2 节规定。

~~0.1.4~~7.4.5 宏块层语义

mb_type 是指宏块的类型。mb_type 的语义依赖于其条带类型。

表和语义由 I, SI, P, SP 和 B 条带的不同宏块类型指定。每张表格代表了 mb_type 的值，mb_type 的名称，使用的宏块划分的数目（由 NumMbPart(mb_type)函数给出），宏块的预测模式（当宏块不划分时）或第一划分（由 MbPartPredMode(mb_type, 0)函数给出）和第二划分的预测模式（由 MbPartPredMode(mb_type, 1)函数给出）。无效的值用"na"表示。在正文中，mb_type 的值可表示为宏块的类型，而 MbPartPredMode( )中的值 X 可表示为"X 宏块（划分）预测模式"或"X 预测宏块"。

表 7-10 列出了每种条带类型的允许的宏块类型集合。

注 1 — 有一些取值为Pred_L0预测模式的宏块类型可归类为B宏块类型。

表7-10－slice_type允许的宏块类型集合

slice_type	允许的宏块类型集合
I (条带)	I (参见表 7-11) (宏块类型)
P (条带)	P (参见表 7-13) 和 I (参见表 7-11) (宏块类型)
B (条带)	B (参见表 7-14) 和 I (参见表 7-11) (宏块类型)
SI (条带)	SI (参见表 7-12) 和 I (参见表 7-11) (宏块类型)
SP (条带)	P (参见表 7-13) 和 I (参见表 7-11) (宏块类型)

transform_size_8x8_flag 等于 1 表示对于当前宏块，亮度样点中变换系数解码过程和图像构建过程的调用，应在残余的 8x8 块的去块效应滤波过程之前完成。transform_size_8x8_flag 等于 0 表示对于当前宏块，亮度样点当中变换系数解码过程和图像构建过程的调用在剩余的 4x4 块的去块效应滤波过程之前完成。如果 transform_size_8x8_flag 在比特流中不存在，则默认其值为 0。

注 2 — 在当前宏块预测模式MbPartPredMode(mb_type, 0)等于帧内_16x16时，比特流中不存在transform_size_8x8_flag且其默认值为0。

当所有序号为 mbPartIdx = 0..3 的 8x8 块的比特流中存在 sub_mb_type[mbPartIdx] (参见 7.4.5.2 小节)时，变量 noSubMbPartSizeLessThan8x8Flag 表示每四个 8x8 块对应的 SubMbPartWidth(sub_mb_type[mbPartIdx])和 SubMbPartHeight(sub_mb_type[mbPartIdx])都等于 8。

注 3 — 当noSubMbPartSizeLessThan8x8Flag等于0且当前宏块类型不等于I_NxN时，比特流中不存在transform_size_8x8_flag且其默认值为0。

表 7-11 中列出了所有表示为 I 宏块类型的宏块类型。

所有 I 条带的宏块类型都是 I 宏块类型。

表7-11－I 条带的宏块类型

mb_type	mb_type 的名称	transform_size_8x8_flag	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	I_NxN	0	Intra_4x4	na	公式 7-33	公式 7-33
0	I_NxN	1	Intra_8x8	na	公式 7-33	公式 7-33
1	I_16x16_0_0_0	na	Intra_16x16	0	0	0
2	I_16x16_1_0_0	na	Intra_16x16	1	0	0
3	I_16x16_2_0_0	na	Intra_16x16	2	0	0
4	I_16x16_3_0_0	na	Intra_16x16	3	0	0
5	I_16x16_0_1_0	na	Intra_16x16	0	1	0
6	I_16x16_1_1_0	na	Intra_16x16	1	1	0
7	I_16x16_2_1_0	na	Intra_16x16	2	1	0
8	I_16x16_3_1_0	na	Intra_16x16	3	1	0
9	I_16x16_0_2_0	na	Intra_16x16	0	2	0
10	I_16x16_1_2_0	na	Intra_16x16	1	2	0
11	I_16x16_2_2_0	na	Intra_16x16	2	2	0
12	I_16x16_3_2_0	na	Intra_16x16	3	2	0
13	I_16x16_0_0_1	na	Intra_16x16	0	0	15
14	I_16x16_1_0_1	na	Intra_16x16	1	0	15
15	I_16x16_2_0_1	na	Intra_16x16	2	0	15
16	I_16x16_3_0_1	na	Intra_16x16	3	0	15
17	I_16x16_0_1_1	na	Intra_16x16	0	1	15
18	I_16x16_1_1_1	na	Intra_16x16	1	1	15
19	I_16x16_2_1_1	na	Intra_16x16	2	1	15
20	I_16x16_3_1_1	na	Intra_16x16	3	1	15
21	I_16x16_0_2_1	na	Intra_16x16	0	2	15
22	I_16x16_1_2_1	na	Intra_16x16	1	2	15
23	I_16x16_2_2_1	na	Intra_16x16	2	2	15
24	I_16x16_3_2_1	na	Intra_16x16	3	2	15
25	I_PCM	na	na	na	na	na

~~A~~

在表 7-11 中如下语义分配给宏块类型：

I_NxN 是 mb_type 等于零且 MbPartPredMode(mb_type, 0)等于 Intra_4x4 或 Intra_8x8 的助记名

I_16x16_0_0_0, I_16x16_1_0_0, I_16x16_2_0_0, I_16x16_3_0_0, I_16x16_0_1_0, I_16x16_1_1_0, I_16x16_2_1_0, I_16x16_3_1_0, I_16x16_0_2_0, I_16x16_1_2_0, I_16x16_2_2_0, I_16x16_3_2_0, I_16x16_0_0_1, I_16x16_1_0_1, I_16x16_2_0_1, I_16x16_3_0_1, I_16x16_0_1_1, I_16x16_1_1_1, I_16x16_2_1_1, I_16x16_3_1_1, I_16x16_0_2_1, I_16x16_1_2_1, I_16x16_2_2_1, I_16x16_3_2_1 是指宏块按 Intra_16x16 预测模式宏块编码。

对每一个 Intra_16x16 预测宏块，取值为 Intra16x16PredMode 表示它为 Intra_16x16 预测模式。CodedBlockPatternChroma 包含了色度码块图的值，其解释参见表 7-15。当 chroma_format_idc 等于 0 时，CodedBlockPatternChroma 也等于 0。CodedBlockPatternLuma 表示对于亮度分量，非零的 AC 变换系数幅值是存在的。CodedBlockPatternLuma 等于 0 表示宏块的亮度分量中的所有 AC 变换系数幅值为 0。CodedBlockPatternLuma 等于 15 表示宏块的亮度分量中至少有一个 AC 变换系数幅值不是 0，这就要求扫描所有 16 个 16x16 块中的 4x4 块的 AC 变换系数幅值。

Intra_4x4 表示宏块预测模式，它表示 Intra_4x4 预测过程按 8.3.1 小节中所述被调用。Intra_4x4 是一种帧内宏块预测模式。

Intra_8x8 表示宏块预测模式，它表示 Intra_8x8 预测过程按 8.3.2 小节中所述被调用。Intra_8x8 是一种帧内宏块预测模式。

Intra_16x16 表示宏块预测模式，它表示 Intra_16x16 预测过程按 8.3.3 小节中所述被调用。Intra_16x16 是一种帧内宏块预测模式。

对于用 mb_type 编码且 mb_type 等于 I_PCM 的宏块，可推断此宏块为帧内宏块预测编码模式。

对于可被称为 SI 宏块类型的宏块类型，其详细描述见表 7-12。

SI 条带的宏块类型可参见表 7-12 和表 7-11。mb_type 值为 0 请参见表 7-12，mb_type 值为 1 到 26 请参见表 7-11，其序号为 mb_type 的值减 1。

表7-12－SI条带的值为0的宏块类型

mb_type	mb_type 的名称	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	SI	Intra_4x4	na	公式 7-33	公式 7-33

在表 7-12 中如下语义分配给宏块类型。SI 宏块按 Intra_4x4 预测宏块编码。

表 7-13 中列出了全部为 P 宏块类型的宏块类型。

P 和 SP 条带的宏块类型可参见表 7-13 和表 7-11。mb_type 值为 0 到 4 请参见表 7-13，mb_type 值为 5 到 30 请参见表 7-11，其序号为 mb_type 的值减 5。

表7-13－P和SP条带的值为0到4的宏块类型

mb_type	mb_type 的名称	NumMbPart (mb_type)	MbPartPredMode (mb_type, 0)	MbPartPredMode (mb_type, 1)	MbPartWidth (mb_type)	MbPartHeight (mb_type)
0	P_L0_16x16	1	Pred_L0	na	16	16
1	P_L0_L0_16x8	2	Pred_L0	Pred_L0	16	8
2	P_L0_L0_8x16	2	Pred_L0	Pred_L0	8	16
3	P_8x8	4	na	na	8	8
4	P_8x8ref0	4	na	na	8	8
推测值	P_Skip	1	Pred_L0	na	16	16

在表 7-13 中如下语义分配给宏块类型：

–— P_L0_16x16表示大小为16x16的亮度样点的一个亮度宏块划分与相关的色度样点一起可以预测出宏块样点。

–— P_L0_L0_MxN, MxN可以是16x8或8x16，它表示宏块样点的预测可以分别由两个大小为MxN(MxN等于16x8)的亮度划分或两个大小为MxN(MxN等于8x16) 亮度划分与相关的色度样点一起来实现。

–— P_8x8表示对每一个子宏块，比特流中有一个附加的语法元素来指明相应子宏块的类型（参见7.4.5.2小节）。

–— P_8x8ref0的语义与P_8x8相同，区别是对于P_8x8ref0，比特流中没有参考索引（ref_idx_l0）的语法元素，且对应所有宏块的子宏块的ref_idx_l0[mbPartIdx]推导后应该等于0（其中mbPartIdx的索引值为0..3）。

–— P_Skip表示比特流中的宏块没有更多的数据了。

在表7-13中，如下语义分配给宏块预测模式（MbPartPredMode( )）。

–— Pred_L0表示使用列表0预测调用帧间预测过程。Pred_L0是一种帧间宏块预测模式。

表 7-14 列出了所有的可能与 B 宏块类型有关的宏块类型。

表 7-14 和表 7-11 中列出了 B 条带的宏块类型。mb_type 值为 0 到 22 的在表 7-14 种列出，mb_type 值为 23 到 48 的在表 7-11 种列出，其序号值为 mb_type 减去 23。

表7-14－B条带中值为0到22的宏块类型

mb_type	mb_type 的名称	NumMbPart (mb_type)	MbPartPredMode (mb_type, 0)	MbPartPredMode (mb_type, 1)	MbPartWidth (mb_type)	MbPartHeight (mb_type)
0	B_Direct_16x16	na	Direct	Na	8	8
1	B_L0_16x16	1	Pred_L0	Na	16	16
2	B_L1_16x16	1	Pred_L1	Na	16	16
3	B_Bi_16x16	1	BiPred	Na	16	16
4	B_L0_L0_16x8	2	Pred_L0	Pred_L0	16	8
5	B_L0_L0_8x16	2	Pred_L0	Pred_L0	8	16
6	B_L1_L1_16x8	2	Pred_L1	Pred_L1	16	8
7	B_L1_L1_8x16	2	Pred_L1	Pred_L1	8	16
8	B_L0_L1_16x8	2	Pred_L0	Pred_L1	16	8
9	B_L0_L1_8x16	2	Pred_L0	Pred_L1	8	16
10	B_L1_L0_16x8	2	Pred_L1	Pred_L0	16	8
11	B_L1_L0_8x16	2	Pred_L1	Pred_L0	8	16
12	B_L0_Bi_16x8	2	Pred_L0	BiPred	16	8
13	B_L0_Bi_8x16	2	Pred_L0	BiPred	8	16
14	B_L1_Bi_16x8	2	Pred_L1	BiPred	16	8
15	B_L1_Bi_8x16	2	Pred_L1	BiPred	8	16
16	B_Bi_L0_16x8	2	BiPred	Pred_L0	16	8
17	B_Bi_L0_8x16	2	BiPred	Pred_L0	8	16
18	B_Bi_L1_16x8	2	BiPred	Pred_L1	16	8
19	B_Bi_L1_8x16	2	BiPred	Pred_L1	8	16
20	B_Bi_Bi_16x8	2	BiPred	BiPred	16	8
21	B_Bi_Bi_8x16	2	BiPred	BiPred	8	16
22	B_8x8	4	na	na	8	8
推测值	B_Skip	na	直接值	na	8	8

在表 7-14 中，如下语义分配给宏块类型：

– B_Direct_16x16 表示比特流中的宏块没有运动矢量差别或参考序号。函数MbPartWidth(B_Direct_16x16)和MbPartHeight(B_Direct_16x16)用于8.4.1小节中直接模式预测的运动矢量和参考帧序号的推导过程。

– B_X_16x16（其中X可以是L0, L1, or Bi）表示用一个大小为16x16亮度样点的亮度宏块划分和色度样点一起来预测宏块样点。对于一个类型为B_X_16x16（其中X为L0或L1）的宏块，其比特流中存在一个运动矢量差和一个参考序号。对于一个类型为B_X_16x16（其中X为Bi）的宏块，宏块的比特流中存在两个运动矢量差和两个参考序号。

–— B_X0_X1_MxN（其中X0, X1表示第一和第二宏块划分，它们都可以是L0或L1；MxN可以是16x18或8x16）表示用两个大小为MxN(MxN等于16x8或8x16)亮度样点的亮度宏块划分和色度样点一起来预测宏块样点。对于一个宏块划分X0或X1（X0或X1都可以是L0或L1），其比特流中存在一个运动矢量差和一个参考序号。对于一个宏块划分X0或X1（X0或X1均为Bi），宏块划分的比特流中存在两个运动矢量差和两个参考序号。

–— B_8x8表示对每一个子宏块，比特流中存在一个附加的语法元素（sub_mb_type），它指示出相应子宏块的类型（见7.4.5.2小节）。

–— B_Skip表示比特流中的宏块没有更多的数据了。函数MbPartWidth(B_Skip)和MbPartHeight(B_Skip)用于8.4.1小节中直接模式预测的运动矢量和参考帧序号的推导过程。

在表 7-14 中，如下语义分配给宏块预测模式(MbPartPredMode( ))。

–— 直接表示比特流中的宏块（当_Skip或B_Direct_16x16时）没有运动矢量差异或参考序号。直接是一种帧间宏块预测模式。

–— Pred_L0：参见表7-13的语义。

–— Pred_L1表示使用列表1的预测调用帧间预测过程。Pred_L1是一种帧间宏块预测模式。

–— BiPred表示使用列表0和列表1预测调用帧间预测过程。BiPred是一种帧间宏块预测模式。

pcm_alignment_zero_bit 是一个等于 0 的比特。

pcm_sample_luma[i]是一个样点值。第一个 pcm_sample_luma[i]值代表宏块里光栅扫描中的亮度样点值。比特的数目通常代表这些样点每一个都是 $BitDepth_Y$。当 profile_idc 不等于 100, 110, 122 或 144 时，pcm_sample_luma[i]不能等于 0。

pcm_sample_ chroma [i]是一个样点值。第一个 MbWidthC * MbHeightC pcm_sample_ chroma [i]值代表宏块里光栅扫描中的 Cb 样点值且其余的 MbWidthC * MbHeightC pcm_sample_chroma[i]值代表宏块里光栅扫描中的 Cr 样点值。比特的数目通常代表这些样点每一个都是 $BitDepth_C$。当 profile_idc 不等于 100, 110, 122 或 144 时，pcm_sample_ chroma [i]不能等于 0。

coded_block_pattern 表示六个亮度和色度 8x8 块可能含有非零的变换系数幅值。对预测模式不等于 Intra_16x16 的宏块，比特流中存在 coded_block_pattern 且变量 CodedBlockPatternLuma 和 CodedBlockPatternChroma 由如下公式得出：

CodedBlockPatternLuma = coded_block_pattern % 16

CodedBlockPatternChroma = coded_block_pattern / 16 (7-33)

当存在 coded_block_pattern 时，CodedBlockPatternLuma 表示了宏块的每四个 8x8 亮度块的下列情形之一：

-— 8x8 亮度块中的四个 4x4 亮度块的所有变换系数幅值都等于 0。

- — 8x8 亮度块中的一个或多个 4x4 亮度块的一个或多个变换系数幅值不为 0。

表 7-15 中列出了 CodedBlockPatternChroma 的含义。

表7-15－CodedBlockPatternChroma 取值规范

CodedBlockPatternChroma	描述
0	所有色度变换系数幅值都等于 0。
1	一个或多个色度 DC 变换系数幅值不为 0。所有色度 AC 变换系数幅值都等于 0。
2	零个或多个色度 DC 变换系数幅值不为 0。一个或多个色度 AC 变换系数幅值不为 0。

mb_qp_delta 能改变宏块层中 QP_Y 的值。mb_qp_delta 解码后的值应包含在–(26 + $QpBdOffset_Y$ / 2) 到 +(25 + $QpBdOffset_Y$ / 2)范围内。当任何宏块（包括 P_Skip 和 B_Skip 宏块类型）中都不存在 mb_qp_delta 时，mb_qp_delta 默认为 0。

QP_Y 的值为：

$$QP_Y = ((QP_{Y,PREV} + mb\_qp\_delta + 52 + 2 * QpBdOffset_Y) \% (52 + QpBdOffset_Y)) - QpBdOffset_Y \quad (7\text{-}34)$$

其中 $QP_{Y,PREV}$ 是当前条带中按解码顺序排列的前一个宏块的亮度量化参数 QP_Y。在每一条带的起始处，对于条带中的第一个宏块，$QP_{Y,PREV}$ 的初始值为 $SliceQP_Y$，它可由式 7-27 得出。

QP'_Y 的值由下式得出：

$$QP'_Y = QP_Y + QpBdOffset_Y \quad (7\text{-}35)$$

~~0.1.4.1~~7.4.5.1 宏块预测语义

宏块的所有样点都可预测。使用如下语法元素可得出预测模式。

prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]** 和 **rem_intra4x4_pred_mode[** luma4x4BlkIdx **]** 表示序号为 luma4x4BlkIdx = 0..15 的 4x4 亮度块的 Intra_4x4 预测。

prev_intra8x8_pred_mode_flag[luma8x8BlkIdx] 和 **rem_intra8x8_pred_mode[** luma8x8BlkIdx] 表示序号为 luma8x8BlkIdx = 0..3 的 8x8 亮度块的 Intra_8x8 预测。

intra_chroma_pred_mode 表示如表 7-16 所示，宏块中用于色度的空间预测类型使用 Intra_4x4 或 Intra_16x16 预测。intra_chroma_pred_mode 的取值范围为 0 到 3。

表 7-16－intra_chroma_pred_mode 和空间预测模式间的关系

intra_chroma_pred_mode	帧内色度预测模式
0	DC
1	水平的
2	垂直的
3	平面的

当 **ref_idx_l0[** mbPartIdx **]** 存在时, 它表示参考图像列表序号为 0 的参考图像被用于预测。

ref_idx_l0[mbPartIdx]的范围，参考图像列表 0 中的序号和用于预测的参考图像中的场的奇偶如下所述：

— 如果MbaffFrameFlag等于0或mb_field_decoding_flag等于0，ref_idx_l0[mbPartIdx]的取值范围将是0到num_ref_idx_l0_active_minus1，包括边界值。

— 否则（MbaffFrameFlag等于1且mb_field_decoding_flag也等于1），ref_idx_l0[mbPartIdx]的取值范围将是0到2 * num_ref_idx_l0_active_minus1 + 1。

当帧间预测只用 1 幅参考图像时，ref_idx_l0[mbPartIdx]的值应等于 0。

ref_idx_l1[mbPartIdx **]** 与 ref_idx_l0 的语义相同，只需用 l1 和列表 1 分别取代 l0 和列表 0。

mvd_l0[mbPartIdx **][** 0 **][** compIdx **]**表示了使用的矢量分量和其预测值间的差异。序号 mbPartIdx 表示 mvd_l0 属于哪一个宏块划分。宏块的划分以 mb_type 表示。水平运动矢量分量差值按解码顺序首先被解码且其 CompIdx = 0。垂直运动矢量分量按解码顺序第二个被解码且其 CompIdx = 1。按附录 A 所述得出的运动矢量变量的值的约束表示了 mvd_l0[mbPartIdx][0][compIdx]的分量的取值范围。

mvd_l1[mbPartIdx **][** 0 **][** compIdx **]**与 mvd_l0 的语义相同，只需用 11 和 L1 分配取代 l0 和 L0。

~~0.1.4.2~~7.4.5.2 子宏块预测语义

sub_mb_type[mbPartIdx **]** 表示子宏块的类型。

表和语义用于表示 P 和 B 宏块类型的各种各样的子宏块类型。每一张表表示了 sub_mb_type 的值，sub_mb_type 的名称，使用的子宏块划分数目（由 NumSubMbPart(sub_mb_type)函数给出）和子宏块的预测模式（由函数 SubMbPredMode(sub_mb_type)给出）。在正文中，sub_mb_type 的值可能被称为“子宏块类型”，SubMbPredMode( )的值可能被称为“子宏块预测模式”。

P 宏块类型中 sub_mb_type[mbPartIdx] 的解释见表 7-17，其中“推测值”那一行表示当 sub_mb_type[mbPartIdx]不存在时的推测值。

表 7-17—P宏块中的子宏块类型

sub_mb_type[mbPartIdx]	sub_mb_type[mbPartIdx] 的名称	NumSubMbPart (sub_mb_type[mbPartIdx])	SubMbPredMode (sub_mb_type[mbPartIdx])	SubMbPartWidth (sub_mb_type[mbPartIdx])	SubMbPartHeight (sub_mb_type[mbPartIdx])
推测值	Na	na	na	na	na
0	P_L0_8x8	1	Pred_L0	8	8
1	P_L0_8x4	2	Pred_L0	8	4
2	P_L0_4x8	2	Pred_L0	4	8
3	P_L0_4x4	4	Pred_L0	4	4

在表 7-17 中如下语义分配给子宏块类型。

– — P_L0_MxN，其中 MxN 可以是 8x8, 8x4, 4x8, 或 4x4，它表示子宏块样点的预测可以分别由一个大小为 MxN(MxN 等于 8x8)的亮度划分，两个大小为 MxN(MxN 等于 8x4)的亮度划分，或两个大小为 MxN(MxN 等于 4x8)的亮度划分，或四个大小为 MxN(MxN 等于 4x4)的亮度划分与其相关的色度样点一起实现。

在表 7-17 中，如下语义分配给子宏块预测模式(SubMbPredMode( ))。

– — Pred_L0: 参见表 7-13 中的语义。

B 宏块类型中 sub_mb_type[mbPartIdx] 的解释见表 7-18，其中“推测值”一行表示当 sub_mb_type[mbPartIdx]不存在时的推测值，推测值"mb_type"表示在这种情形下 sub_mb_type[mbPartIdx]的名称与 mb_type 的名称相同。

表7-18－B宏块中的子宏块类型

sub_mb_type[mbPartIdx]	sub_mb_type[mbPartIdx]的名称	NumSubMbPart (sub_mb_type[mbPartIdx])	SubMbPredMode (sub_mb_type[mbPartIdx])	SubMbPartWidth (sub_mb_type[mbPartIdx])	SubMbPartHeight (sub_mb_type[mbPartIdx])
推测值	mb_type	4	直接值	4	4
0	B_Direct_8x8	4	直接值	4	4
1	B_L0_8x8	1	Pred_L0	8	8
2	B_L1_8x8	1	Pred_L1	8	8
3	B_Bi_8x8	1	BiPred	8	8
4	B_L0_8x4	2	Pred_L0	8	4
5	B_L0_4x8	2	Pred_L0	4	8
6	B_L1_8x4	2	Pred_L1	8	4
7	B_L1_4x8	2	Pred_L1	4	8
8	B_Bi_8x4	2	BiPred	8	4
9	B_Bi_4x8	2	BiPred	4	8
10	B_L0_4x4	4	Pred_L0	4	4
11	B_L1_4x4	4	Pred_L1	4	4
12	B_Bi_4x4	4	BiPred	4	4

表 7-18 中，如下语义分配给子宏块类型：

– — B_Skip 和 B_Direct_16x16 表示比特流中的子宏块没有运动矢量差或参考序号。对于直接模式预测, 函数 SubMbPartWidth( )和 SubMbPartHeight( )用于在 8.4.1 小节中运动矢量和参考帧序号的推导过程。

– — B_Direct_8x8 表示比特流中的子宏块没有运动矢量差或参考序号。对于直接模式预测, 函数 SubMbPartWidth(B_Direct_8x8)和 SubMbPartHeight(B_Direct_8x8)用于在 8.4.1 小节中运动矢量和参考帧序号的推导过程。

– — B_X_MxN, 其中 X 可以是 L0, L1 或 Bi，MxN 可以是 8x8, 8x4, 4x8 或 4x4，它表示子宏块样点的预测可分别由一个大小为 MxN 等于 8x8 的亮度划分，或两个大小为 MxN 等于 8x4 的亮度划分，或两个大小为 MxN 等于 4x8 的亮度划分，或四个大小为 MxN 等于 4x4 的亮度划分与其相关的色度样点一起来实现。所有子宏块划分共享相同的参考序号。对于位于 sub_mb_type 为 B_X_MxN（其中 X 不是 L0 就是 L1）的子宏块中的一个 MxN 子宏块划分，比特流中存在一个运动矢量差。对于位于 sub_mb_type 为 B_Bi_MxN 的子宏块中的一个 MxN 子宏块划分，比特流中存在两个运动矢量差。

表 7-18 中，如下语义分配给子宏块预测模式(SubMbPredMode( ))：

– — 直接值：参见表 7-14 中的语义。

– — Pred_L0：参见表 7-13 中的语义。

– — Pred_L1：参见表 7-14 中的语义。

– — BiPred：参见表 7-14 中的语义。

ref_idx_l0[mbPartIdx **]**的语义和 7.4.5.1 节中 ref_idx_l0 的相同。

ref_idx_l1[mbPartIdx **]**的语义和 7.4.5.1 节中 ref_idx_l1 的相同。

除了应用于具有 subMbPartIdx 的子宏块分割块索引，**mvd_l0[** mbPartIdx **][** subMbPartIdx **][** compIdx **]**的语义和 7.4.5.1 节中 mvd_l0 的相同。索引值 mbPartIdx 和 subMbPartIdx 表明 mvd_l0 被指派给哪一个宏块划分和子宏块划分。

mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]**的语义和 7.4.5.1 节中 mvd_l1 的相同。

~~0.1.4.3~~7.4.5.3 残差数据语义

用于分析变换系数幅值的语法结构 residual_block( )，其赋值如下：

- — 如果 entropy_coding_mode_flag 等于 0，residual_block 将被设置为等于 residual_block_cavlc，用于分析变换系数幅值的语法元素。
- — 否则（entropy_coding_mode_flag 等于 1），residual_block 将被设置为等于 residual_block_ cabac，用于分析变换系数幅值的语法元素。

依赖于 mb_type，亮度或色度，和色度格式，语法结构 residual_block(coeffLevel, maxNumCoeff)，与参数 coeffLevel 一起使用如下（coeffLevel 是一个包含分解入 residual_block( )和 maxNumCoeff 的 maxNumCoeff 级变换系数幅值的列表）：

— 依赖于 MbPartPredMode(mb_type, 0)，应用如下规则：

– — 如果 MbPartPredMode(mb_type, 0) 等于 Intra_16x16，变换系数幅值被解析成 Intra16x16DCLevel列表和16个Intra16x16ACLevel[i]列表。Intra16x16DCLevel包含16级用于每个4x4亮度块的DC的变换系数幅值。对16个索引值为i=1..15的4x4亮度块中的每一个亮度块，第i个块的15个AC变换系数幅值被解析成第i个列表Intra16x16ACLevel[i]。

– — 否则（MbPartPredMode(mb_type, 0)不等于Intra_16x16），应用如下规则：

– — 如果transform_size_8x8_flag等于0，对16个序号值为i=0..15的4x4亮度块中的每一个亮度块，第i个块的16级变换系数幅值被解析成第i个列表LumaLevel[i]。

– — 否则（transform_size_8x8_flag等于1），对4个序号值为i=0..3的8x8亮度块中的每一个亮度块，应用如下规则：

– — 如果entropy_coding_mode_flag等于0，首先，对每个序号为i4x4 = 0..3的4个4x4亮度块中的每一个亮度块，第i4x4个块的16级变换系数幅值被解析成第(i8x8 * 4 + i4x4)个列表LumaLevel[i8x8 * 4 + i4x4]。其次，序号值为4 * i + i4x4的第i8x8个亮度块的64级变换系数幅值（其中i = 0..15，i4x4 = 0..3）可以从公式LumaLevel8x8[i8x8][4 * i + i4x4] = LumaLevel[i8x8 * 4 + i4x4][i]得到。

注 — 假定luma4x4BlkIdx = i8x8 * 4 + i4x4（其中包含偏移量为i4x4的相应的第i8x8个8x8亮度块的每一个四阶变换系数幅值）的4x4亮度块代表由6.4.3小节中的逆向4x4亮度块扫描过程给出的空间位置。

– — 否则（entropy_coding_mode_flag等于1），第i8x8个块的64级变换系数幅值被解析成第i8x8个列表LumaLevel8x8[i8x8]。

– — 对每一个序号值为iCbCr = 0..1的色度分量，第4 * NumC8x8个4x4色度块的DC变换系数幅值被解析成第iCbCr个列表ChromaDCLevel[iCbCr]。

– — 对每一个色度分量（序号值为iCbCr = 0..1）的4x4色度块(序号值为i4x4 = 0..3和i8x8 = 0..NumC8x8 – 1)，15 级 AC 变换系数幅值被解析成第 iCbCr 个色度分量的第 (i8x8*4 + i4x4) 个列表 ChromaACLevel [iCbCr][i8x8*4 + i4x4]。

~~0.1.4.3.1~~7.4.5.3.1 残差块CAVLC语义

用在 7.3.5.3.1 小节中的函数 TotalCoeff(coeff_token)，其返回值为从 coeff_token 中导出的非零变换系数幅值的个数。

用在 7.3.5.3.1 小节中的函数 TrailingOnes (coeff_token)，其返回值为从 coeff_token 中导出的那个拖尾。

coeff_token 是指一次变换系数幅值扫描中的非零变换系数幅值总数和拖尾变换系数幅值的个数。拖尾变换系数幅值是指非零变换系数扫描结尾处绝对值为 1 的多至连续三个非零变换系数级别中的一个。coeff_token 的取值范围在 9.2.1 节给出。

trailing_ones_sign_flag 是指拖尾的变换系数幅值的的正负符号，详情如下：

- 如果 trailing_ones_sign_flag 等于 0，则相应的变换系数幅值解码值为+1。
- 否则（trailing_ones_sign_flag 等于 1），则相应的变换系数幅值解码值为-1。

level_prefix 和 level_suffix 是指非零变换系数幅值的值。9.2.2 小节中描述了 level_prefix 和 level_suffix 的取值范围。

total_zeros 是指位于变换系数幅值扫描中最后的非零变换系数幅值的位置之前的取值为 0 的变换系数幅值的总数。total_zeros 的取值范围如 9.2.3 小节所述。

run_before 是指扫描中位于非零变换系数幅值之前的连续的取值为 0 的变换系数幅值的个数。run_before 的取值范围如 9.2.3 小节所述。

coeffLevel 包含了当前的变换系数幅值列表的 maxNumCoeff 个变换系数幅值。

~~0.1.4.3.2~~7.4.5.3.2 残差块CABAC语义

coded_block_flag 是指块中是否包含非零变换系数幅值，详情如下：

- 如果 coded_block_flag 等于 0，则块中不包含非零变换系数幅值。
- 否则（coded_block_flag 等于 1），块中至少包含一个非零变换系数幅值。

significant_coeff_flag[i **]**是指扫描位置 i 上的变换系数幅值是否非零，详情如下：

- 如果 significant_coeff_flag[i]等于 0，则扫描位置 i 上的变换系数幅值等于 0。
- 否则（significant_coeff_flag[i]等于 1），扫描位置 i 上的变换系数幅值不等于零。

last_significant_coeff_flag[i **]**是指对于扫描位置 i，其后续的扫描位置（从 i + 1 到 maxNumCoeff – 1）上是否有非零的变换系数幅值，详情如下：

- 如果 last_significant_coeff_flag[i]等于 1，则块中的所有后续（按扫描顺序排序）的变换系数幅值其值均为 0。
- 否则（last_significant_coeff_flag[i]等于 0），扫描路径上有更多的非零变换系数幅值。

coeff_abs_level_minus1[i **]**是变换系数幅值减 1 的绝对值。coeff_abs_level_minus1 的值受限于 8.5 小节中的限制方式。

coeff_sign_flag[i **]**是指变换系数幅值的正负符号，详情如下：

- 如果 coeff_sign_flag 等于 0，则相应的变换系数幅值为正。
- 否则（coeff_sign_flag 等于 1），相应的变换系数幅值为负。

coeffLevel 包含了当前的变换系数幅值列表的 maxNumCoeff 个变换系数幅值。

8 解码过程

本过程的输出为当前图像（有时以变量 CurrPic 表示）的已解码样点。

本节在第 7 节给定语法元素和大写变量的基础上，描述解码过程。

本节规定的解码过程可以使所有的解码器得到一致的数字结果。任何解码过程，若产生与本节规定的相同的结果，就符合本建议书|国际标准的解码过程需求。

本节中所有参考图像都是基本图像。本节中的所有参考条带都是基本图像中的条带。本节中所有的参考条带分割都是基本图像中的条带数据分割。

解码过程的概述如下：

— 8.1 中定义 NAL 单元的解码过程。

— 8.2 节定义了使用条带层及条带层以上各层的语法元素进行解码的过程。

  — 与图像顺序排列有关的变量和函数在8.2.1中推导。(仅对图像中的某条带解码时才需要调用)。

  — 与宏块到条带组映射有关的变量和函数在8.2.2中推导。（仅对图像中的某条带解码时才需要调用）。

  — 当使用条带数据分割时，组合多个条带数据分割的方法在8.2.3定义。

  — 在当前图像的frame_num不等于PrevRefFrameNum，且不等于(PrevRefFrameNum + 1) % MaxFrameNum 时，frame_num上的空隙的解码过程应在解码图像中的所有条带之前，按照8.2.5.2的规定执行。

  — 在 P、 SP 或B 条带的解码开始阶段，需要执行8.2.4节定义的参考图像列表重建过程，从而可以得到参考图像列表0 (RefPicList0)，当解码B条带时，得到参考图像列表1（RefPicList1）。

  — 当当前图像是参考图像，且在所有本图像中的条带都已解码后，8.2.5中的已解码图像的标记过程描述当前图像如何在后续图像解码过程的帧间预测过程中使用。

— 8.3，8.4，8.5，8.6 和 8.7 定义使用宏块层及宏块层以上层的语法元素进行解码的过程。

  — 除8.3节所定义的I_PCM宏块以外，I 和SI宏块的帧内预测过程都以帧内预测得到的样点作为输出。8.3节直接定义了I_PCM宏块的重建过程。输出的结果是在去块效应滤波过程之前的重建样点。

  — P和B宏块的帧间预测过程在8.4中定义，帧间预测结果作为输出。

  — 去块效应滤波前变换系数的解码过程和图像重建过程在8.5中定义。此过程产生I和B宏块以及P条带中的P宏块样点。输出结果为去块效应滤波前的重建样点。

  — 8.6定义了SP条带和SI宏块中的P宏块的解码过程。该过程产生SP、SI宏块中的P宏块的样点值。输出结果是去块效应滤波之前的重建样点。

  — 8.7定义了对靠近块和宏块边缘的重建图像进行的去块效应滤波的过程，输出是已解码样点。

8.1 NAL单元解码过程

本过程的输入是 NAL 单元。

本过程的输出是封装在 NAL 单元中的 RBSP 语法结构。

每一个 NAL 单元的解码过程都是从 NAL 单元中提取出 RBSP 语法结构，然后指示解码过程对 NAL 单元中的 RBSP 语法结构按照以下的方式进行解码。

8.2 节描述 nal_unit_type 的值为 1 到 5 时 NAL 单元的解码过程。

8.3 节描述 nal_unit_type 的值为 1，2 和 5 时 NAL 单元中的宏块或宏块分割的解码过程。

8.4 节描述 nal_unit_type 的值为 1 和 2 时 NAL 单元中的宏块或宏块分割的解码过程。

8.5 节描述 nal_unit_type 的值为 1，3，4 和 5 时 NAL 单元中的宏块或宏块分割的解码过程。

8.6 节描述 nal_unit_type 的值为 1，3，4 和 5 时 NAL 单元中的宏块或宏块分割的解码过程。

8.7 节描述 nal_unit_type 的值为 1 到 5 时 NAL 单元中的宏块或宏块分割的解码过程。

nal_unit_type 值为 7 和 8 时 NAL 单元里封装的分别是序列参数集和图像参数集。图像参数集用于由每一个条带头中图像参数集决定的 NAL 单元的解码过程。序列参数集用于由图像参数集中的序列参数集决定的其它 NAL 单元的解码过程。

不规定 NAL 单元中的 nal_unit_type 值为 6，9，10，11，12 的解码过程。

8.2 条带解码过程

8.2.1 图像顺序号的解码过程

本过程的输出是 TopFieldOrderCnt (如果可用) 和 BottomFieldOrderCnt (如果可用)。

图像顺序号用来在解码 B 条带时（参考 8.2.4.2.3 和 8.2.4.2.4 节）决定参考图像的初始图像顺序，在 B 条带的显式加权预测（参见 8.4.2.3.2）中以及在解码器的一致性检查中（参见 C.4），用来表示时域直接模式（参见 8.4.1.2.3）下运动矢量推导过程中的帧或场之间的图像序号差别。

对每一帧、场（或者由编码场解码得到，或者作为解码帧的一部分）和互补场对都要产生图像顺序号信息，过程如下：

— 每一个编码帧有两个图像顺序号，其顶场和底场分别称为TopFieldOrderCnt和BottomFieldOrderCnt。

— 每一个编码场有一个图像顺序号，其顶场或底场分别称为TopFieldOrderCnt或BottomFieldOrderCnt。

— 每一个互补参考场对有两个图像顺序号，其顶场和底场分别称为 TopFieldOrderCnt 和 BottomFieldOrderCnt。

TopFieldOrderCnt 和 BottomFieldOrderCnt 表示了相应的顶场和底场的相对图像顺序，这一顺序是相对于前一个 IDR 图像的第一个输出场的图像顺序，或相对于包括 memory_management_control_operation 值为 5 的图像在内的解码顺序上前一个参考图像。

TopFieldOrderCnt 和 BottomFieldOrderCnt 是通过分别调用 8.2.1.1、8.2.1.2 和 8.2.1.3 节中图像顺序类型为 0、1、2 的解码过程推导出的。如果当前图像包含有值等于 5 的存储管理控制操作（memory_management_control_operation）时，在当前图像解码完成后，tempPicOrderCnt 的值设置为当前图像的 PicOrderCnt 值(CurrPic)，当前图像的 TopFieldOrderCnt（如果存在）值设置为 TopFieldOrderCnt－tempPicOrderCnt，当前图像的 BottomFieldOrderCnt（如果存在）的值设置为 BottomFieldOrderCnt－tempPicOrderCnt。

对一个已编码的 IDR 帧，比特流中不应该含有导致 Min(TopFieldOrderCnt, BottomFieldOrderCnt)不等于 0 的数据，对一个已编码的 IDR 顶场，比特流中不应该含有导致 TopFieldOrderCnt 不等于 0 的数据，或对一个已编码的 IDR 底场，比特流中不应该含有导致 BottomFieldOrderCnt 不等于 0 的数据。这样，对 IDR 编码帧的场，TopFieldOrderCnt 和 BottomFieldOrderCnt 中至少要有一个的值等于 0。

当当前图像不是 IDR 图像时，应用如下过程：

— 认为列表变量包含TopFieldOrderCnt和BottomFieldOrderCnt元素，这些元素的值与包括了下列所有内容的图像列表相关：

  — 列表中的第一个图像是位于以下任一类型图像之前：

    — 一个IDR图像

    — 一个memory_management_control_operation值为5的图像

— 以及下列额外图像：

— 如果 pic_order_cnt_type等于0，应该是按照解码顺序列表中第一个图像之后的所有其它图像，这些图像不能是由 8.2.5.2节所述frame_num间隙的解码过程中得到的非"不存在"图像，或者是解码顺序中当前图像之前的图像或者是当前图像。当pic_order_cnt_type等于0，并且当前图像不是 8.2.5.2节规定的frame_num间隙的解码过程中得到的"不存在"图像时，当前图像被包含在在解码参考图像标记过程之前的listD中。

否则(pic_order_cnt_type 不等于 0)，应该是按照解码顺序列表第一个图像之后的所有其它图像，这些图像或者是解码顺序中当前图像之前的图像或者是当前图像。当 pic_order_cnt_type 不等于 0 时，当前图像被包含在在解码参考图像标记过程之前的 listD 中。

— 考虑变量列表ListO，它包含按照升序排列的ListD中的元素。ListO不应该含有如下列任何值：

— 对用于一帧或者互补参考场对的TopFieldOrderCnt和BottomFieldOrderCnt，它们在ListO中处于不连续的位置上。

— 值等于另一个TopFieldOrderCnt的TopFieldOrderCnt。

— 值等于另一个BottomFieldOrderCnt的BottomFieldOrderCnt。

— 值等于另一个TopFieldOrderCnt的BottomFieldOrderCnt，TopFieldOrderCnt和BottomFieldOrderCnt属于同一帧或互补参考场对的情况除外。

比特流中不应该含有导致 8.2.1.1 到 8.2.1.3 中定义的解码过程用到的 TopFieldOrderCnt 、BottomFieldOrderCnt、PicOrderCntMsb、或 FrameNumOffset 的值超过（含）-2^{31} 到 $2^{31}-1$ 范围的数据。

函数 PicOrderCnt(picX)如下：

if(picX 是一帧或互补场对)
PicOrderCnt(picX)= picX 互补场对的 Min(TopFieldOrderCnt, BottomFieldOrderCnt)
else if(picX 是顶场)

PicOrderCnt(picX) =场 picX 的 TopFieldOrderCnt (8-1)

else if(picX 是底场)
PicOrderCnt(picX) =场 picX 的 BottomFieldOrderCnt

DiffPicOrderCnt(picA, picB)如下：

DiffPicOrderCnt(picA, picB)= PicOrderCnt(picA) − PicOrderCnt(picB) (8-2)

比特流中不应该含有导致用于解码过程的 DiffPicOrderCnt(picA, picB)的值超过-2^{15} 到 $2^{15}-1$ 范围的数据。

注 1 — 设X是当前图像，Y, Z是同一序列中的另外两幅图像，当DiffPicOrderCnt(X, Y)和 DiffPicOrderCnt(X, Z)的值都为正或都为负值时，Y, Z被认为是相对于X是同一个输出顺序方向。

注 2 — 很多种应用中PicOrderCnt(X)的值设为与相对于IDR图像的样点时间的X图像的样点时间成比例。

当当前图像包含一个等于 5 的 memory_management_control_operation 时，PicOrderCnt(CurrPic)应该大于 PicOrderCnt(listD中的任何其他图像)。

8.2.1.1 图像顺序类型为0时的解码过程

当 pic_order_cnt_type 等于 0 时调用本过程。

本过程的输入是在本节规定的解码顺序中前一图像的 PicOrderCntMsb。

本过程的输出是 TopFieldOrderCnt 和 BottomFieldOrderCnt，或者其中之一。

变量 prevPicOrderCntMsb 和 prevPicOrderCntLsb 的值由以下过程得到：

— 如果当前图像是IDR图像，prevPicOrderCntMsb设为0，prevPicOrderCntLsb也等于0。

— 否则 (当前图像为非 IDR 图像)，应用如下规则：

— 如果解码顺序中的前一个已解码图像包含的memory_management_control_operation等于5，应用如下规则：

— 如果在解码顺序上前一个参考图像不是底场，则prevPicOrderCntMsb 设置为0，prevPicOrderCntLsb设置为解码顺序上前一个参考图像的TopFieldOrderCnt值。

— 否则（按照解码顺序前一个参考图像为底场），prevPicOrderCntMsb 设为 0 并且prevPicOrderCntLsb设为0。

— 否则（解码顺序上前一个参考图像不包含等于5的memory_management_control_operation），prevPicOrderCntMsb的值设置为等于解码顺序中前一个图像的PicOrderCntMsb，prevPicOrderCntLsb的值设为解码顺序中前一参考图像的pic_order_cnt_lsb。

当前图像的 PicOrderCntMsb 由以下过程得到：

```
if( ( pic_order_cnt_lsb  <  prevPicOrderCntLsb )  &&
  ( ( prevPicOrderCntLsb − pic_order_cnt_lsb )  >=  ( MaxPicOrderCntLsb / 2 ) ) )
   PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb                    (8-3)
else if( ( pic_order_cnt_lsb  >  prevPicOrderCntLsb )  &&
        ( ( pic_order_cnt_lsb − prevPicOrderCntLsb )  >  ( MaxPicOrderCntLsb / 2 ) ) )
   PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb
else
   PicOrderCntMsb = prevPicOrderCntMsb
```

当当前图像不是底场时，TopFieldOrderCnt 由以下过程得到：

```
if( !field_pic_flag  ||  !bottom_field_flag )
   TopFieldOrderCnt = PicOrderCntMsb + pic_order_cnt_lsb                      (8-4)
```

当当前图像不是顶场时，BottomFieldOrderCnt 由以下过程得到：

```
if( !field_pic_flag )
   BottomFieldOrderCnt = TopFieldOrderCnt + delta_pic_order_cnt_bottom
else if( bottom_field_flag )                                                  (8-5)
   BottomFieldOrderCnt = PicOrderCntMsb + pic_order_cnt_lsb
```

8.2.1.2 图像顺序类型为1时的解码过程

当 pic_order_cnt_type 等于 1 时调用本过程。

本过程的输入是在本节规定的解码顺序中前一图像的 FrameNumOffset。

本过程的输出是 TopFieldOrderCnt 和 BottomFieldOrderCnt，或者其中之一。

TopFieldOrderCnt 和 BottomFieldOrderCnt 的值由本节所规定的原理得到。令 prevFrameNum 等于解码顺序上前一个图像的 frame_num。

当当前图像不是 IDR 图像时，变量 prevFrameNumOffset 的值由如下过程得到：

— 如果解码顺序上前一个图像包含的 memory_management_control_operation 等于 5，则prevFrameNumOffset等于0。

— 否则（按照解码顺序前一图像不包含等于 5 的 memory_management_control_operation），prevFrameNumOffset的值等于解码顺序上前一图像的FrameNumOffset。

注 — 当gaps_in_frame_num_value_allowed_flag等于1时，按照8..5.2节的规定，通过frame_num间隔的解码过程可能会推断出解码顺序中的前一幅图像为“不存在”帧。

推导过程按照下列步骤顺序进行：

1．变量 FrameNumOffset 由如下方法得到：

```
if( nal_unit_type  ==  5 )
    FrameNumOffset = 0
else if( prevFrameNum > frame_num )                                           (8-6)
    FrameNumOffset = prevFrameNumOffset + MaxFrameNum
```

```
else
    FrameNumOffset = prevFrameNumOffset
```

2．变量 absFrameNum 由以下过程得到：

```
if( num_ref_frames_in_pic_order_cnt_cycle  !=  0 )
    absFrameNum = FrameNumOffset + frame_num
else                                                                    (8-7)
    absFrameNum = 0
if( nal_ref_idc  ==  0  &&  absFrameNum  >  0 )
    absFrameNum = absFrameNum − 1
```

3．当 absFrameNum 大于 0 时，picOrderCntCycleCnt 和 frameNumInPicOrderCntCycle 的值由如下方法得到：

```
if( absFrameNum  >  0 ) {
    picOrderCntCycleCnt = ( absFrameNum − 1 ) / num_ref_frames_in_pic_order_cnt_cycle
    frameNumInPicOrderCntCycle = ( absFrameNum − 1 ) % num_ref_frames_in_pic_order_cnt_cycle        (8-8)
}
```

4．变量 expectedDeltaPerPicOrderCntCycle 的值由如下方法得到：

```
expectedDeltaPerPicOrderCntCycle = 0
for( i = 0;  i < num_ref_frames_in_pic_order_cnt_cycle;  i++ )
    expectedDeltaPerPicOrderCntCycle += offset_for_ref_frame[ i ]       (8-9)
```

5．变量 expectedPicOrderCnt 的值由如下方法得到：

```
if( absFrameNum > 0 ){
    expectedPicOrderCnt = picOrderCntCycleCnt * expectedDeltaPerPicOrderCntCycle
    for( i = 0; i <= frameNumInPicOrderCntCycle; i++ )
        expectedPicOrderCnt = expectedPicOrderCnt + offset_for_ref_frame[ i ]
} else
    expectedPicOrderCnt = 0
if( nal_ref_idc  ==  0 )                                                (8-10)
    expectedPicOrderCnt = expectedPicOrderCnt + offset_for_non_ref_pic
```

6．变量 TopFieldOrderCnt 或 BottomFieldOrderCnt 的值由如下方法得到：

```
if( !field_pic_flag ) {
    TopFieldOrderCnt = expectedPicOrderCnt + delta_pic_order_cnt[ 0 ]
    BottomFieldOrderCnt = TopFieldOrderCnt +
            offset_for_top_to_bottom_field + delta_pic_order_cnt[ 1 ]   (8-11)
} else if( !bottom_field_flag )
     TopFieldOrderCnt = expectedPicOrderCnt + delta_pic_order_cnt[ 0 ]
else
    BottomFieldOrderCnt =          expectedPicOrderCnt + offset_for_top_to_bottom_field + delta_pic_order_cnt[ 0 ]
```

8.2.1.3 图像顺序类型为2时的解码过程

当 pic_order_cnt_type 等于 2 时调用本过程。

本过程的输出是 TopFieldOrderCnt 和 BottomFieldOrderCnt，或者其中的某一个。

设 prevFrameNum 等于按照解码顺序前一图像的 frame_num。

当当前图像不是 IDR 图像时，变量 prevFrameNumOffset 由如下方法得到：

— 如果解码顺序中的前一图像包含的 memory_management_control_operation 等于 5，则 prevFrameNumOffset的值等于0。

— 否则（解码顺序中的前一图像不包含等于 5 的 memory_management_control_operation），则 prevFrameNumOffset等于前一图像的FrameNumOffset。

注 1 — 当gaps_in_frame_num_value_allowed_flag等于1时，按照8.2.5.2节的规定，通过frame_num间隔的解码过程可能会推断出解码顺序中的前一幅图像为"不存在"帧。

变量 FrameNumOffset 由如下过程得到：

```
if( nal_unit_type == 5 )
  FrameNumOffset = 0
else if( prevFrameNum > frame_num )                                    (8-12)
  FrameNumOffset = prevFrameNumOffset + MaxFrameNum
else
  FrameNumOffset = prevFrameNumOffset
```

变量 tempPicOrderCnt 由如下过程得到：

```
if( nal_unit_type == 5 )
   tempPicOrderCnt = 0
else if( nal_ref_idc == 0 )                                            (8-13)
    tempPicOrderCnt = 2 * ( FrameNumOffset + frame_num ) − 1
else
    tempPicOrderCnt = 2 * ( FrameNumOffset + frame_num )
```

变量 TopFieldOrderCnt 或 BottomFieldOrderCnt 由如下过程得到：

```
if( !field_pic_flag ) {
   TopFieldOrderCnt = tempPicOrderCnt
   BottomFieldOrderCnt = tempPicOrderCnt                               (8-14)
} else if( bottom_field_flag )
   BottomFieldOrderCnt = tempPicOrderCnt
else
   TopFieldOrderCnt = tempPicOrderCnt
```

注 2 — 图像顺序类型2不能用在包含一系列连续非参考图像的视频序列中，那样会导致其中的多个图像具有相同TopFieldOrderCnt值，或者其中的多个图像具有相同BottomFieldOrderCnt值。

注 3 — 图像顺序类型2会产生与解码顺序一致的输出顺序。

8.2.2 宏块到条带组的映射的解码过程

本过程的输入是当前活动图像参数集和待解码的条带的条带头。

本过程的输出是宏块到条带组的映射 MbToSliceGroupMap。

在所有条带开始时都要调用本过程。

注 — 本过程的输出对于所有图像的条带都相同。

当 num_slice_groups_minus1 等于 1 并且 slice_group_map_type 等于 3、4 和 5 时，条带组 0 和 1 的大小和形状由 slice_group_change_direction_flag 决定，如表 8-1，具体规定见 8.2.2.4 至 8.2.2.6。

表 8-1－精确条带组映射类型

slice_group_map_type	slice_group_change_direction_flag	精确条带组映射类型
3	0	Box-out clockwise
3	1	Box-out counter-clockwise
4	0	Raster scan
4	1	Reverse raster scan
5	0	Wipe right
5	1	Wipe left

在这种情况下，MapUnitsInSliceGroup0 条带组映射单元按照规定的增长顺序分配给条带组 0，图像的其他的 PicSizeInMapUnits − MapUnitsInSliceGroup0 个条带组映射单元分配到条带组 1。

当 num_slice_groups_minus1 等于 1 并且 slice_group_map_type 等于 4、5 时，变量 sizeOfUpperLeftGroup 的值定义如下：

```
sizeOfUpperLeftGroup = ( slice_group_change_direction_flag ?
                         ( PicSizeInMapUnits − MapUnitsInSliceGroup0 ) : MapUnitsInSliceGroup0 )   (8-15)
```

变量 mapUnitToSliceGroupMap 的值通过以下方式得到：

— 如果num_slice_groups_minus1的值等于0，对所有i, i从0到（PicSizeInMapUnits − 1）,条带组映射单元到条带组的映射方法如下，i从0到（PicSizeInMapUnits − 1），包括0和（PicSizeInMapUnits − 1）：

```
mapUnitToSliceGroupMap[ i ] = 0                                                    (8-16)
```

— 否则(如果num_slice_groups_minus1的值不等于0), mapUnitToSliceGroupMap通过如下方式得到：

  — 如果slice_group_map_type为0, 则mapUnitToSliceGroupMap的推导过程在8.2.2.1规定。

  — 否则，如果slice_group_map_type等于 1，mapUnitToSliceGroupMap的推导过程在8.2.2.2中规定。

  — 否则，如果slice_group_map_type等于 2，mapUnitToSliceGroupMap的推导过程在8.2.2.3中规定。

  — 否则，如果slice_group_map_type等于 3，mapUnitToSliceGroupMap的推导过程在8.2.2.4中规定。

  — 否则，如果slice_group_map_type等于 4，mapUnitToSliceGroupMap的推导过程在8.2.2.5中规定。

  — 否则，如果slice_group_map_type等于 5，mapUnitToSliceGroupMap的推导过程在8.2.2.6中规定。

  — 否则(slice_group_map_type等于 6)，mapUnitToSliceGroupMap的推导过程在8.2.2.7中规定。

在得到 mapUnitToSliceGroupMap 之后,要调用 8.2.2.8 中规定的过程，将映射单元到条带组的映射 mapUnitToSliceGroupMap 转变为宏块到条带组的映射 MbToSliceGroupMap。当得到 8.2.2.8 中规定的宏块到条带组的映射以后，通过函数 NextMbAddress(n)得到 NextMbAddress，如下：

```
    i = n + 1
  while( i < PicSizeInMbs  &&  MbToSliceGroupMap[ i ]  !=  MbToSliceGroupMap[ n ] )
    i++;
nextMbAddress = i                                                                  (8-17)
```

8.2.2.1 隔行扫描型条带组映射类型的规范

slice_group_map_type 等于 0 时，应用本节规定的过程。

映射单元到条带组的映射由以下方法得到：

```
i = 0
do
  for( iGroup = 0; iGroup <= num_slice_groups_minus1 && i < PicSizeInMapUnits;
       i += run_length_minus1[ iGroup++ ] + 1 )
    for( j = 0; j <= run_length_minus1[ iGroup ] && i + j < PicSizeInMapUnits; j++ )
      mapUnitToSliceGroupMap[ i + j ] = iGroup                                     (8-18)
  while( i < PicSizeInMapUnits )
```

8.2.2.2 分散型条带组映射类型的规范

slice_group_map_type 等于 1 时，应用本节规定的过程。

映射单元到条带组的映射由以下方法得到：

```
for( i = 0; i < PicSizeInMapUnits; i++ )
  mapUnitToSliceGroupMap[ i ] = ( ( i % PicWidthInMbs ) +
                                  ( ( ( i / PicWidthInMbs ) * ( num_slice_groups_minus1 + 1 ) ) / 2 ) )
                                % ( num_slice_groups_minus1 + 1 )                                    (8-19)
```

8.2.2.3 具有残余条带组映射类型的前景规范

slice_group_map_type 等于 2 时，应用本节规定的过程。

映射单元到条带组的映射由以下方法得到：

```
for( i = 0; i < PicSizeInMapUnits; i++ )
  mapUnitToSliceGroupMap[ i ] = num_slice_groups_minus1
for( iGroup = num_slice_groups_minus1 − 1; iGroup >= 0; iGroup-−) {
  yTopLeft = top_left[ iGroup ] / PicWidthInMbs
  xTopLeft = top_left[ iGroup ] % PicWidthInMbs
  yBottomRight = bottom_right[ iGroup ] / PicWidthInMbs
  xBottomRight = bottom_right[ iGroup ] % PicWidthInMbs
  for( y = yTopLeft; y <= yBottomRight; y++ )
      for( x = xTopLeft; x <= xBottomRight; x++ )
          mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] = iGroup                                   (8-20)
}
```

注 — 矩形可能重叠。条带组0包含的宏块在由top_left[0] 和bottom_right[0]定义的矩形内。一个条带组号如果大于0并且小于num_slice_groups_minus1，条带组包含那些在由这一条带组确定的矩形内，但又不在比这一条带组条带组号大的条带组内的宏块。条带组号等于num_slice_groups_minus1的条带组包含那些不在其他条带组内的宏块。

8.2.2.4 box-out条带组类型的规范

slice_group_map_type 等于 3 时，应用本节规定的过程。

映射单元到条带组的映射由以下方法得到：

```
for( i = 0; i < PicSizeInMapUnits; i++ )
  mapUnitToSliceGroupMap[ i ] = 1
x = ( PicWidthInMbs − slice_group_change_direction_flag ) / 2
y = ( PicHeightInMapUnits − slice_group_change_direction_flag ) / 2
( leftBound, topBound ) = ( x, y )
( rightBound, bottomBound ) = ( x, y )
( xDir, yDir ) = ( slice_group_change_direction_flag − 1, slice_group_change_direction_flag )
for( k = 0; k < MapUnitsInSliceGroup0; k += mapUnitVacant ) {
  mapUnitVacant = ( mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] == 1 )
  if( mapUnitVacant )
      mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] = 0                                            (8-21)
  if( xDir == −1 && x == leftBound ) {
      leftBound = Max( leftBound − 1, 0 )
      x = leftBound
      ( xDir, yDir ) = ( 0, 2 * slice_group_change_direction_flag − 1 )
  } else if( xDir == 1 && x == rightBound ) {
      rightBound = Min( rightBound + 1, PicWidthInMbs − 1 )
      x = rightBound
      ( xDir, yDir ) = ( 0, 1 − 2 * slice_group_change_direction_flag )
  } else if( yDir == −1 && y == topBound ) {
      topBound = Max( topBound − 1, 0 )
      y = topBound
      ( xDir, yDir ) = ( 1 − 2 * slice_group_change_direction_flag, 0 )
  } else if( yDir == 1 && y == bottomBound ) {
      bottomBound = Min( bottomBound + 1, PicHeightInMapUnits − 1 )
      y = bottomBound
      ( xDir, yDir ) = ( 2 * slice_group_change_direction_flag − 1, 0 )
  } else
      ( x, y ) = ( x + xDir, y + yDir )
}
```

8.2.2.5 光栅扫描条带组类型的规范

当 slice_group_map_type 等于 4 时，应用本节规定。

映射单元到条带组的映射由以下方法得到：

```
for( i = 0; i < PicSizeInMapUnits; i++ )
  if( i < sizeOfUpperLeftGroup )
    mapUnitToSliceGroupMap[ i ] = slice_group_change_direction_flag
  else                                                                    (8-22)
    mapUnitToSliceGroupMap[ i ] = 1 − slice_group_change_direction_flag
```

8.2.2.6 消除条带组类型的规范

当 slice_group_map_type 等于 5 时，应用本节规定。

映射单元到条带组的映射由以下方法得到

```
k = 0;
for( j = 0; j < PicWidthInMbs; j++ )
  for( i = 0; i < PicHeightInMapUnits; i++ )
    if( k++ < sizeOfUpperLeftGroup )
      mapUnitToSliceGroupMap[ i * PicWidthInMbs + j ] = slice_group_change_direction_flag
    else                                                                  (8-23)
      mapUnitToSliceGroupMap[ i * PicWidthInMbs + j ] = 1 − slice_group_change_direction_flag
```

8.2.2.7 显式条带组类型的规范

当 slice_group_map_type 等于 6 时，应用本节规定。

映射单元到条带组的映射由以下方法得到：

```
mapUnitToSliceGroupMap[ i ] = slice_group_id[ i ]                         (8-24)
```

对所有 i 从 0 到 PicSizeInMapUnits − 1 范围, 包括 0 和 PicSizeInMapUnits − 1。

8.2.2.8 由映射单元到条带组的映射到宏块到条带组的映射转换的规范

i 从 0 到 PicSizeInMapUnits − 1（包括 0 和 PicSizeInMapUnits − 1），宏块到条带组的映射由以下过程产生：

— 如果frame_mbs_only_flag等于1或者field_pic_flag等于1，宏块到条带组的映射如下：

```
MbToSliceGroupMap[ i ] = mapUnitToSliceGroupMap[ i ]                      (8-25)
```

— 否则，如果MbaffFrameFlag等于1，宏块到条带组的映射如下：

```
MbToSliceGroupMap[ i ] = mapUnitToSliceGroupMap[ i / 2 ]                  (8-26)
```

— 否则（frame_mbs_only_flag等于0，并且mb_adaptive_frame_field_flag等于0，并且field_pic_flag等于0），宏块到条带组的映射如下：

```
MbToSliceGroupMap[ i ] = mapUnitToSliceGroupMap[ ( i / ( 2 * PicWidthInMbs ) ) * PicWidthInMbs
                                                 + ( i % PicWidthInMbs ) ]  (8-27)
```

8.2.3 条带数据分割的解码过程

本过程的输入是：

— 条带数据分割A层RBSP，

— 当条带数据中包含类别3的语法元素时，与条带数据A层RBSP的条带具有相同slice_id的条带数据分割B层RBSP，以及

— 当条带数据中包含类别4的语法元素时，与条带数据分割A层RBSP的条带具有相同slice_id的条带数据分割C层RBSP。

注 1 — 条带数据分割B层RBSP和条带数据分割C层RBSP不一定需要存在。

本过程的输出是已经编码的条带。

没有使用条带数据分割时，由未分割 RBSP 的一个条带层表示的已编码条带包含了一个条带头，在条带头后是包含条带宏块数据的类别 2、3、4（参见 7.3 节类型表）的所有语法元素在内的条带数据语法结构。

当使用了条带数据分割时，条带的宏块数据分割成包含在各自 NAL 单元内的三个分割块。分割块 A 包含一个条带数据分割块 A 的头和类别 2 所有的语法元素。当分割块 B 存在时，包含一个条带数据分割块 B 头和类别 2 的所有语法元素。当分割块 C 存在时，包含一个条带数据分割块 C 头和类别 4 的所有语法元素。

当使用了条带数据分割时，每一类别的语法元素从不同的 NAL 单元分别单独解析，当没有各自类别的符号存在时，相应解析可以不存在。解码过程应该以与处理相应的没有进行分割的 RBSP 条带同样的方式处理这些分割，在这个过程中，需要将语法元素从条带数据分割中提取出来，在这些分割里，语法元素的出现依赖于 7.3 节中的条带数据分割的方式。

注 2 — 类别3的语法元素与I和SI宏块类型的残留数据解码相关。类别4的语法元素与P和B宏块类型的残留数据解码相关。类型2包含所有其他的与宏块解码有关的语法元素，并且它们的信息经常表示为头信息。条带数据分割块A的头包含条带头里的所有语法元素，还包含一个用于将条带数据分割块B和C和A 联系起来的slice_id。条带数据分割块B和C仅包含一个用于与条带数据分割A联系起来的slice_id。

8.2.4 参考图像列表解码过程

在解码每一个 P，SP，或 B 条带开始时，调用本过程。

解码参考图像按照 8.2.5 节的规定和比特流的说明，被标记为“用于短期参考”或“用于长期参考”。短期参考图像用 frame_num 值标识。长期参考图像按照 8.2.5 节的规定和比特流的说明，被分配一个长期帧索引。

调用 8.2.4.1 节来说明下述内容：

— 将变量FrameNum 、FrameNumWrap和PicNum的值赋给每个短期参考图像，以及

— 将变量LongTermPicNum的值赋给每个长期参考图像。

参考图像是通过 8.4.2.1 节中定义的参考索引来定位的。参考索引是一个指向参考图像列表的索引。当解码一个 P 或 SP 条带时，只有一个参考图像列表 RefPicList0，当解码一个 B 条带时，除 RefPicList0 还有一个独立的参考图像列表 RefPicList1。

在每一条带解码开始的时候，参考图像列表 RefPicList0 和 RefPicList1（用于 B 条带）由以下步骤得到：

— 初始参考图像列表RefPicList0和RefPicList1（用于B条带）按8.2.4.2所述得到。

— 初始参考图像列表RefPicList0和RefPicList1（用于B条带）按8.2.4.3所述修改。

注 — 8.2.4.3节所定义的参考图像列表重排序过程允许RefPicList0以及RefPicList1（对B条带）的内容以一种灵活的方式被修改。特别的，一个被标记为"用于参考"的图像也可能被插入到RefPicList0，而对于B条带的RefPicList1，该图像甚至可以不在按照8.2.4.2节规定推导出的初始参考图像列表中。

修改过的参考图像列表 RefPicList0 中条目总数目为 num_ref_idx_l0_active_minus1 + 1，对 B 条带来说，修改过的参考图像列表 RefPicList1 中条目总数目为 num_ref_idx_l1_active_minus1 + 1。一个参考图像可能会在一个以上的 RefPicList0 或 RefPicList1 的索引中出现。

8.2.4.1 图像编号的解码过程

在调用 8.2.4 节的参考图像列表重建过程或 8.2.5 节的已解码参考图像标记过程时，调用本过程。

变量 FrameNum, FrameNumWrap, PicNum, LongTermFrameIdx 和 LongTermPicNum 用于 8.2.4.2 中参考图像列表的初始化过程，8.2.4.3 中的参考图像列表修改过程，8.2.5 中的解码参考图像标识过程。

对每一个短期参考图像，变量 FrameNum 和 FrameNumWrap 由如下所示过程赋值。首先，FrameNum 设置为与已经解码的相应短期参考图像条带头部语法元素 frame_num 相等。变量 FrameNumWrap 由如下方法得到：

```
if( FrameNum > frame_num )
  FrameNumWrap = FrameNum − MaxFrameNum                    (8-28)
else
  FrameNumWrap = FrameNum
```

其中 8-28 中 frame_num 的值为当前图像条带头的 frame_num 值。

每个长期参考图像都具有一个 LongTermFrameIdx 值（按照 8.2.5 中的规定来进行分配）。

为每一个短期参考图像分配一个变量 PicNum，为每一个长期参考图像分配一个变量 LongTermPicNum。这些变量的值由当前图像的 field_pic_flag 和 bottom_field_flag 决定，它们的设置方式如下所示：

— 如果field_pic_flag等于0，应用如下过程：

— 对每一个短期参考帧或互补参考场对：

PicNum = FrameNumWrap (8-29)

— 对每一个长期参考帧或长期互补参考场对：

LongTermPicNum = LongTermFrameIdx (8-30)

注 — 当解码一帧图像时，MbaffFrameFlag的值对8.2.4.2, 8.2.4.3和 8.2.5的推导过程没有影响。

— 否则(field_pic_flag等于1)，应用如下规则：

— 对每一个短期参考场以下规则适用：

— 如果参考场和当前场有相同的奇偶性

PicNum = 2 * FrameNumWrap + 1 (8-31)

— 否则（参考场与当前场的奇偶性相反），

PicNum = 2 * FrameNumWrap (8-32)

— 对每一个长期参考场，应用如下规则：

— 如果参考场和当前场有相同的奇偶性

LongTermPicNum = 2 * LongTermFrameIdx + 1 (8-33)

— 否则（参考场与当前场的奇偶性相反），

LongTermPicNum = 2 * LongTermFrameIdx (8-34)

8.2.4.2 参考图像列表的初始化过程

在解码 P、SP 或 B 条带头的时候，调用此初始化过程。

如 8.2.4.2.1 到 8.2.4.2.5 中所述，RefPicList0 和 RefPicList1 具有初始值。

当 8.2.4.2.1 节至 8.2.4.2.5 节定义的 RefPicList0 或 RefPicList1 中条目的数目分别比 num_ref_idx_l0_active_minus1 + 1 或 num_ref_idx_l1_active_minus1 + 1 大时，从参考图像列表中删除超过 num_ref_idx_l0_active_minus1 + 1 或 num_ref_idx_l1_active_minus1 + 1 的位置的条目。

当 8.2.4.2.1 节至 8.2.4.2.5 节定义的 RefPicList0 或 RefPicList1 中条目的数目分别比 num_ref_idx_l0_active_minus1 + 1 或 num_ref_idx_l1_active_minus1 + 1 小时，将剩下的初始参考图像列表中的条目设置为“非参考图像”。

8.2.4.2.1 帧中P, SP条带的参考图像列表的初始化过程

在解码一个已编码帧 P 或 SP 条带时，调用此初始化过程。

当调用这个过程时，至少有一个参考帧或者互补参考场对当时被标记为"用于短期参考"或者“用于长期参考”。

参考图像列表 RefPicList0 被排序，从而使短期参考帧和短期互补参考场对的索引值比长期参考帧和长期互补参考场对的索引值小。

短期参考帧和短期互补参考场对的顺序以具有最大 PicNum 值的帧或参考场对开始，按递减顺序一直到具有最小 PicNum 值的参考帧或互补参考场对。

长期参考帧和长期互补参考场对的顺序以具有最小 LongTermPicNum 值的帧或参考场对开始，按递增顺序一直到具有最大 LongTermPicNum 值的参考帧或互补参考场对。

注 — 解码一个帧时，不管MbaffFrameFlag为何值，非成对参考场都将不被使用。

例如，当被标记为“用于短期参考”的三个参考帧，分别具有的 PicNum 值等于 300，302，303 并且两个标记为“用于长期参考”的参考帧，LongTermPicNum 值分别为 0 和 3 时，则初始索引顺序为：

— RefPicList0[0] 设为等于 PicNum = 303的短期参考图像，

— RefPicList0[1] 设为等于PicNum = 302的短期参考图像，

— RefPicList0[2] 设为等于 PicNum = 300的短期参考图像，

— RefPicList0[3] 设为等于 LongTermPicNum = 0的长期参考图像，和

— RefPicList0[4] 设为等于 LongTermPicNum = 3的长期参考图像。

8.2.4.2.2 场中P, SP条带的参考图像列表的初始化过程

当解码一个已编码场中 P 或 SP 条带时，调用本初始化过程。

参考图像列表 RefPicList0 中的每一个场在参考图像列表 RefPicList0 中都具有独立的索引值。

注 — 当解码一个场时，参考图像列表中的可用参考图像数至少是解码一个在解码顺序中具有同样位置的帧的可用参考图像数的两倍。

两个已经排好顺序的参考帧列表 refFrameList0ShortTerm 和 refFrameList0LongTerm 按如下方式得到。为形成帧列表、已解码帧、互补参考场对、非成对互补参考场和有一个场被标记为“用于短期参考”或“用于长期参考”的参考帧都被认为是参考帧。

— 所有包含一个或多个标记为“用于短期参考”场的帧，均包含在短期参考帧列表refFrameList0ShortTerm中。若当前场是互补参考场对中的第二场（解码顺序），并且第一场被标记为“用于短期参考”时，则该第一场包含在短期参考帧列表refFrameList0ShortTerm中。refFrameList0ShortTerm的顺序以具有最大FrameNumWrap值的帧开始，以递减顺序直到具有最小FrameNumWrap值的帧为止。

— 所有包含一个或多个帧被标记为“用于长期参考”场的帧，均包含在长期参考帧列表LongTermFrameIdx中。当当前场是互补参考场对中的第二场（解码顺序），并且第一场被标记为“用于长期参考”时，则该第一场包含在长期参考帧列表refFrameList0LongTerm中。refFrameList0LongTerm的顺序以具有最小LongTermFrameIdx值的帧开始，以递增顺序直到具有最大LongTermFrameIdx值的帧为止。

调用 8.2.4.2.5 节中定义的过程，refFrameList0ShortTerm 和 refFrameList0LongTerm 为输入，将输出分配给 RefPicList0。

8.2.4.2.3 帧中B条带的参考图像列表的初始化过程

在解码一个已编码帧 B 条带时，调用本初始化过程

为了便于产生参考图像列表 RefPicList0 和 RefPicList1，后面涉及到的术语参考条目是指解码后的参考帧或者互补参考场对。

在调用这个过程的时候，至少有一个参考条目当前被标记为"用于短期参考"或者"用于长期参考"。

对 B 条带，参考图像列表 RefPicList0 和 RefPicList1 中的短期参考条目的顺序和由 PicOrderCnt( )给出的输出顺序有关。当 pic_order_cnt_type 等于 0 的时候，8.2.5.2 中定义的被标记为“不存在”的参考图像在 RefPicList0 或者 RefPicList1 任何一个当中都不会存在。

注 1 — 当gaps_in_frame_num_value_allowed_flag等于1时，编码器应该利用参考图像列表的再排序功能来保证解码过程的正确运行（尤其是当pic_order_cnt_type等于0的时候，这时不能认为PicOrderCnt( )是“不存在”帧）。

对参考图像列表 RefPicList0 进行排序，从而使短期参考条目的索引值比长期参考条目的索引值低。排序方式如下：

— 假设entryShortTerm是一个变量，它的范围包括所有的当前标记为"用于短期参考"的参考条目。当entryShortTerm的一些值的PicOrderCnt(entryShortTerm)小于PicOrderCnt(CurrPic)时，将entryShortTerm的某些值按照PicOrderCnt(entryShortTerm)的降序方式放在refPicList0的起始位置上。然后entryShortTerm的所有剩余值（如果存在）按照PicOrderCnt(entryShortTerm)升序的方式附加到refPicList0。

— 对长期参考条目进行排序，从具有最低LongTermPicNum值的长期参考条目开始，然后按照升序的方式直到具有最高LongTermPicNum值的长期参考条目。

参考图像列表RefPicList1被排序从而使短期参考条目的索引值比长期参考条目的索引值低。排序过程如下：

— 假设entryShortTerm是一个变量，它的范围包括所有的当前标记为" 用于短期参考 "的参考条目。当entryShortTerm的某些值的PicOrderCnt(entryShortTerm)大于PicOrderCnt(CurrPic)时，将entryShortTerm的这些值按照PicOrderCnt(entryShortTerm)的升序方式放在在refPicList1的起始位置。然后entryShortTerm的所有剩余值（如果存在）按照PicOrderCnt(entryShortTerm)降序的方式附加到refPicList1。

— 将长期参考条目排序，从具有最小LongTermPicNum值的长期参考帧或互补参考场对开始，以递增顺序直到具有最大LongTermPicNum值的长期参考条目。

— 当参考图像列表RefPicList1具有一个以上条目并且和参考图像列表RefPicList0相同时，交换RefPicList0和RefPicList1的前两个条目。

注 2 — 一个未配对的参考场不能用于帧间预测（与MbaffFrameFlag的值无关）。

8.2.4.2.4 场中B条带的参考图像列表的初始化过程

在解码一个已编码场 B 条带时，调用该初始化过程。

当解码一个场时，储存的参考帧的每一个场都作为单独的参考图像并且有唯一的索引值。参考图像列表 RefPicList0 和 RefPicList1 中短期参考图像的顺序依赖于由 PicOrderCnt( )给出的输出顺序。当 pic_order_cnt_type 等于 0 的时候，按照 8.2.5.2 节中定义的方式标记为“不存在”的参考图像不包括在 RefPicList0 或者 RefPicList1 中的任何一个内。

注 1 — 当gaps_in_frame_num_value_allowed_flag等于1时，编码器应该利用参考图像列表的再排序功能来保证解码过程的正确运行（尤其是当pic_order_cnt_type等于0的时候，不能认为PicOrderCnt( )是“不存在”帧）。

注 2 — 当解码一个场时, 参考图像列表中的可用参考图像数，至少是解码一个在解码顺序中具有同样位置的帧的可用参考图像数的两倍。

三个已排序的参考帧列表 refFrameList0ShortTerm, refFrameList1ShortTerm 和 refFrameListLongTerm 可以按照如下方法得到。为形成这些帧列表，后面涉及到的术语参考条目是指解码参考帧，互补参考场对和非成对参考场。当 pic_order_cnt_type 为 0 的时候，术语参考条目没有涉及到 8.2.5.2 中定义的标记为“不存在”的帧。

— 设entryShortTerm是一个变量，它的范围包括所有的当前标记为“用于短期参考”的参考条目。当entryShortTerm中的某些值的PicOrderCnt(entryShortTerm)小于或者等于PicOrderCnt(CurrPic)时，将entryShortTerm的这些值按照PicOrderCnt(entryShortTerm)的降序方式放在refFrameList0ShortTerm的起始位置上。然后entryShortTerm的所有剩余值（如果存在）按照PicOrderCnt(entryShortTerm)升序的方式附加到refFrameList0ShortTerm。

注 3 — 当当前场遵循解码的顺序时，一个编码场fldPrev和当前场一起构成了一个互补参考场对，采用在前一个句子中描述的排序方式，利用PicOrderCnt(fldPrev)把fldPrev放入到refFrameList0ShortTerm列表中。

— 设entryShortTerm是一个变量，它的范围包括所有的当前标记为“用于短期参考”的参考条目。当entryShortTerm的某些值的PicOrderCnt(entryShortTerm)大于PicOrderCnt(CurrPic)，将entryShortTerm的这些值按照PicOrderCnt(entryShortTerm)的升序方式在refFrameList1ShortTerm的起始位置。然后entryShortTerm的所有剩余值（如果存在）按照PicOrderCnt(entryShortTerm)降序的方式附加到refFrameList1ShortTerm。

注 4 — 当当前场遵循解码的顺序时，一个编码场fldPrev和当前场一起构成了一个互补参考场对，利用PicOrderCnt(fldPrev)把fldPrev放入到refFrameList0ShortTerm列表中，并且采用在前一个语句描述的排序方式。

— refFrameListLongTerm的顺序以具有最小LongTermFrameIdx的参考条目开始，以递增顺序一直到具有最大LongTermFrameIdx的参考索引为止。

注 5 — 当当前图像的互补场被标记为“用于长期参考”时，它将被放入到refFrameListLongTerm列表中。把只有一个场被标记为“用于长期参考”的参考条目放入到refFrameListLongTerm列表中。

调用 8.2.4.2.5 节所定义的过程，输入为 refFrameList0ShortTerm 和 refFrameListLongTerm，并且以 RefPicList0 为输出。

调用 8.2.4.2.5 节所定义的过程，refFrameList1ShortTerm 和 refFrameListLongTerm 为输入，并且以 RefPicList0 为输出。

当参考图像列表 RefPicList1 具有一个以上条目并且和参考图像列表 RefPicList0 相同时， 交换 RefPicList0 和 RefPicList1 的前两个条目。

8.2.4.2.5 场中参考图像列表的初始化过程

本过程的输入为 refFrameListXShortTerm (X 可为 0 或 1) 和 refFrameListLongTerm 。

参考图像列表 RefPicListX 被排序以使短期参考场的索引值总是比长期参考场的索引值低。给定 refFrameListXShortTerm 和 refFrameListLongTerm，RefPicListX 由如下过程得到：

— 从与当前场具有相同奇偶性的场（如果存在）开始，对短期参考场进行排序，其方法为从有序的帧列表refFrameListXShortTerm中通过交替选择具有不同奇偶性的场作为参考场。当一个参考帧的一个场还没有解码或被标记为“用于短期参考”时，忽略丢失的场，并且把refFrameListXShortTerm帧列表中具有相应奇偶性的下一个可用已存参考场插入到RefPicListX中。当已经排好序的refFrameListXShortTerm帧列表中没有更多的具有交替奇偶性的短期参考场时，把接下来的奇偶性可用、还没有被索引的场按照它们在已经排好序的refFrameListXShortTerm帧列表中具有的顺序插入到RefPicListX中。

— 从与当前场具有相同奇偶性的场（如果存在）开始，对长期参考场进行排序，其方法为从有序的帧列表refFrameListLongTerm中通过交替选择具有不同奇偶性的场作为参考场。当一个参考帧的一个场还没有解码或被标记为“用于长期参考”时，忽略丢失的场，并且把refFrameListLongTerm帧列表中具有相应奇偶性的下一个可用已存参考场插入到RefPicListX中。当已经排好序的refFrameListLongTerm帧列表中没有更多的

具有交替奇偶性的长期参考场时，把剩下的奇偶性可用、还没有被索引的场按照它们在已经排好序的帧refFrameListLongTerm帧列表中具有的顺序插入到RefPicListX中。

8.2.4.3 参考图像列表的重排序过程

当 ref_pic_list_reordering_flag_l0 等于 1 时，应用如下过程：

— 令 refIdxL0 为指向 RefPicList0 的一个索引，初值为零。

— 对应的语法元素reordering_of_pic_nums_idc按他们在比特流中出现的顺序处理。对每一个语法元素都按下方法处理。

— 如果reordering_of_pic_nums_idc等于0或1，则调用8.2.4.3.1中的定义的过程，refIdxL0作为输入，输出赋值给refIdxL0。

— 否则，如果reordering_of_pic_nums_idc等于2，调用8.2.4.3.2中定义的过程，refIdxL0作为输入，输出赋值给refIdxL0。

— 否则（reordering_of_pic_nums_idc等于3），参考图像列表RefPicList0重排序过程结束。

当 ref_pic_list_reordering_flag_l1 等于 1 时，应用下列规定：

— 设 refIdxL1 为指向 RefPicList1 的一个索引。初值为零。

— 对应的语法元素reordering_of_pic_nums_idc按他们在比特流中出现的顺序处理。对每一个语法元素都按下方法处理。

— 如果reordering_of_pic_nums_idc等于1或0，则调用8.2.4.3.1中定义的过程，refIdxL1作为输入，输出赋给refIdxL1。

— 否则，如果reordering_of_pic_nums_idc等于2，调用8.2.4.3.2中定义的过程，refIdxL1作为输入，输出赋给refIdxL1。

— 否则（reordering_of_pic_nums_idc等于3），参考图像列表RefPicList1重排序过程结束。

8.2.4.3.1 短期参考图像列表的重排序过程

本过程的输入为一个索引 RefPicListX (X 为 0 或 1)。

本过程的输出为一个增量索引 refIdxLX。

变量 picNumLXNoWrap 由以下方法得到：

— 如果reordering_of_pic_nums_idc等于0

```
if( picNumLXPred − ( abs_diff_pic_num_minus1 + 1 ) < 0 )
    picNumLXNoWrap = picNumLXPred − ( abs_diff_pic_num_minus1 + 1 ) + MaxPicNum        (8-35)
else
    picNumLXNoWrap = picNumLXPred − ( abs_diff_pic_num_minus1 + 1 )
```

— 否则（reordering_of_pic_nums_idc等于1）

```
if( picNumLXPred + ( abs_diff_pic_num_minus1 + 1 ) >= MaxPicNum )
    picNumLXNoWrap = picNumLXPred + ( abs_diff_pic_num_minus1 + 1 ) − MaxPicNum        (8-36)
else
    picNumLXNoWrap = picNumLXPred + ( abs_diff_pic_num_minus1 + 1 )
```

picNumLXPred 是变量 picNumLXNoWrap 的预测值。当本节所规定的过程在条带中第一次调用时（也就是说，当语法 ref_pic_list_reordering( ) 中第一次出现 reordering_of_pic_nums_idc 等于 0 或 1），picNumL0Pred 和 picNumL1Pred 初始值设为 CurrPicNum，当每一个 picNumLXNoWrap 都赋完值后，将 picNumLXNoWrap 的值赋给 picNumLXPred。

变量 picNumLX 由以下过程得到：

```
if( picNumLXNoWrap > CurrPicNum )
    picNumLX = picNumLXNoWrap − MaxPicNum                                             (8-37)
else
    picNumLX = picNumLXNoWrap
```

picNumLX 应该等于标记为“用于短期参考”的参考图像 PicNum，不应该等于标记为“不存在”的短期参考图像的 PicNum。

下面过程将一个具有短期序号 picNumLX 的图像放在索引位置 refIdxLX，将剩余图像的位置移动到列表的后部，并且将 refIdxLX 的值增加 1。

```
for( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx-－)
    RefPicListX[ cIdx ] = RefPicListX[ cIdx − 1]
RefPicListX[ refIdxLX++ ] = picNumLX
nIdx = refIdxLX
for( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )                    (8-38)
    if( LongTermEntry( RefPicListX[ cIdx ] ) || RefPicListX[ cIdx ] != picNumLX )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]
```

这里函数 PicNumF(RefPicListX[cIdx])由以下方式得到：

— 如果图像RefPicListX[cIdx] 被标记为"用于短期参考"， PicNumF(RefPicListX[cIdx]) 为图像RefPicListX[cIdx]的 PicNum值。

— 否则 (图像RefPicListX[cIdx]未被标记为"用于短期参考")， PicNumF(RefPicListX[cIdx])等于MaxPicNum。

注 1 — MaxPicNum的值不能等于picNumLX。

注 2 — 在这个伪代码程序内，RefPicListX列表的长度被暂时置为比最终列表所需要的长度多一个元素。本过程执行完后，只需要保留通过列表的num_ref_idx_lX_active_minus1得到的元素0。

8.2.4.3.2 长期参考图像列表的重排序过程

本过程的输入为一个索引 RefPicListX (X 为 0 或 1)。

本过程的输出为一个增量索引 refIdxLX。

下面的过程将一个长期序号为 long_term_pic_num 的图像放在索引位置 refIdxLX，将剩余图像的位置移动到列表的后部，并且将 refIdxLX 的值增加 1。

```
for( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx-－)
    RefPicListX[ cIdx ] = RefPicListX[ cIdx − 1]
RefPicListX[ refIdxLX++ ] = LongTermPicNum                                                 (8-39)
nIdx = refIdxLX
for( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )
    if( !LongTermEntry( RefPicListX[ cIdx ] ) || RefPicListX[ cIdx ] != LongTermPicNum )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]
```

通过下述方式可以得到函数 LongTermPicNumF(RefPicListX[cIdx])。

— 如果图像RefPicListX[cIdx]被标记为“用于长期参考”，LongTermPicNumF(RefPicListX[cIdx])为图像RefPicListX[cIdx]的LongTermPicNum

— 否则（图像RefPicListX[cIdx]没有被标记为“用于长期参考”），LongTermPicNumF(RefPicListX[cIdx])等于2 * (MaxLongTermFrameIdx + 1)。

注 1 — 2 * (MaxLongTermFrameIdx + 1)的值不能等于long_term_pic_num。

注 2 — 在这个伪代码程序中，列表RefPicListX的长度被暂时设置为比最终列表所需要的长度多一个元素。在这个程序执行完以后，只需要保留通过列表的num_ref_idx_lX_active_minus1得到的元素0。

8.2.5 已解码参考图像标记过程

当 nal_ref_idc 的值不等于 0 时，为解码图像调用本过程。

注 — 当nal_ref_idc等于0，也应该调用8.2.5.2中定义的frame_num间隔的解码过程，如第8节的规定。

解码图像的 nal_ref_idc 值如果不为 0，并且被当作参考图像，可能会标记为“用于短期参考”或“用于长期参考”。对一个已解码参考帧，它的两个场的标记和帧相同。对于一个互补参考场对，它的标记也和它的两个场一样。被标记为"用于短期参考帧"的图像通过它的 FrameNum 来识别，如果它是一个场，那么通过它的奇偶性来识别。被标记为"用于长期参考"的图像通过 LongTermFrameIdx 来识别，如果它是一个场，则通过它的奇偶性来识别。

当解码一个帧时，标记为“用于短期参考”或者“用于长期参考”的帧或者互补场对可以作为帧间预测的一个参考，直至帧、互补场对或者它的一个构成场被标记为“不用于参考”。在解码一个场时，一个标记为“用于短期参考”或者“用于长期参考”的场可以作为帧间预测的一个参考，直到该场被标记为“不用于参考”。

图像可以通过 8.2.5.3 定义的滑动窗参考图像标记过程— 一种先进先出机制，或 8.2.5.4 定义的自适应存储控制参考图像标记过程— 一种定制自适应标记操作，标记为"不用于参考"。

一个短期参考图像通过变量 FrameNum 和 FrameNumWrap 以它的图像号码 PicNum 来识别，从而用于解码过程。一个长期参考图像通过它的长期图像号码 LongTermPicNum 来识别从而用在解码过程中。如果当前的图像不是 IDR 图像，那么调用 8.2.4.1 中的过程来定义变量 FrameNum、FrameNumWrap、PicNum 以及 LongTermPicNum 的赋值情况。

8.2.5.1 已解码参考图像标记过程操作步骤

已解码参考图像标记过程按下列步骤顺序进行：

1．当前图像的所有条带都已被解码。

2．根据当前图像是否是 IDR 图像，应用如下规则：

— 如果当前图像是IDR图像，应用如下规则：

  — 所有参考图像需要被标记为"不用于参考"

  — 根据long_term_reference_flag的不同取值，应用下列规则：

    — 如果long_term_reference_flag等于0，则该IDR图像将被标记为"用于短期参考"并且MaxLongTermFrameIdx 设为“非长期帧索引”。

    — 否则（long_term_reference_flag等于1），该IDR图像需要被标记为"用于长期参考"。LongTermFrameIdx应被置为0，MaxLongTermFrameIdx应设为0。

— 否则（当前图像不是IDR图像），应用下列规则：

  — 如果adaptive_ref_pic_marking_mode_flag等于0，则调用8.2.5.3所规定的过程。

  — 否则（adaptive_ref_pic_marking_mode_flag等于1），调用8.2.5.4所规定的过程。

3．在当前图像不是 IDR 图像并且没有被标记为"用于长期参考"时（memory_management_control_operation 等于 6），则当前图像将被标记为 "用于短期参考"。

在当前已解码参考图像标记过程完成后，至少有 1 场标记为“用于参考”的帧的总数，加上至少有一场被标记为“用于参考”的互补参考场对的数目，再加上标记为“用于参考”的非成对场数目，不应该大于 Max(num_ref_frames, 1)。

8.2.5.2 frame_num间隙的解码过程

当 frame_num 不等于 PrevRefFrameNum 并且不等于(PrevRefFrameNum + 1) % MaxFrameNum 时，调用本过程。

注 1 — 虽然该过程是作为8.2.5节（它定义了一个只有在nal_ref_idc不等于0时才调用的过程）的一小节来定义的，但是在nal_ref_idc等于0时，可能也要调用这个过程（参见第8节的规定）。关于这个部分在这个建议书 | 国际标准中的位置是其有历史原因的。

注 2 — 只有当frame_num_value_allowed_flag内的间隔等于1时，才可以为一个一致的比特流调用这个过程。如果frame_num_value_allowed_flag等于0，而且frame_num不等于PrevRefFrameNum也不等于(PrevRefFrameNum + 1) % MaxFrameNum，那么解码过程应该推断出此时存在图像丢失。

当调用本过程时，除当前图像的 frame_num 值，“不存在”图像的 frame_num 值的集合通过等式 7-21 中的 UnusedShortTermFrameNum 得到。

解码过程为属于“不存在”图像的 frame_num 的每一个值产生和标记一个帧，顺序中的 UnusedShortTermFrameNum 通过等式 7-21 产生，使用 8.2.5.3 规定的“滑动窗”图像标记过程。生成的帧也标记为“不存在”和“用于短期参考”。比特流中包含的数据不应该会有导致引用帧间预测过程中标识的“不存在”的帧，和短期参考图像的参考图像列表重排序命令中的帧（见 8.2.4.3.1），或者从 LongTermFrameIdx 到短期参考图像的分配过程中的帧（见 8.2.5.4.3）。

当 pic_order_cnt_type 不等于 0 的时候，调用 8.2.1 中用于图像顺序排列的解码过程，得到每个“不存在”帧的 TopFieldOrderCnt 和 BottomFieldOrderCnt。在为特定的“不存在”帧调用 8.2.1 中的过程时，认为当前的图像的 frame_num 等于 UnusedShortTermFrameNum，nal_ref_idc 不等于 0，nal_unit_type 不能于 5，field_pic_flag 等于 0，adaptive_ref_pic_marking_mode_flag 等于 0，delta_pic_order_cnt[0]（如果需要）等于 0，delta_pic_order_cnt[1]（如果需要）等于 0。

注 3 — 如果在帧间预测过程中，在短期参考图像（8.2.4.3.1）的参考图像列表的重排序命令中，或者在从一个 LongTermFrameIdx 到一个短期参考图像的赋值过程（8.2.5.4.3）中涉及到了任何一个属于“不存在”图像的 frame_num值，解码过程应该可以推断处一个非故意的图像丢失。当把一个不等于3存储器管理控制操作应用到一个标记为“不存在”的帧的时候，解码过程不应当推断为一个非故意的图像丢失。

8.2.5.3 已解码参考图像的滑动窗标记过程

在 adaptive_ref_pic_marking_mode_flag 等于 0 时，调用本过程。

根据如下当前图像的性质，应用下列规则：

— 如果当前编码场是一个互补参考场对的第二个场，并且第一个场已经被标记为“用于短期参考”时，当前图像也应该被标记为“用于短期参考”。

— 否则，应用下列规则：

— 令numShortTerm 为参考图像，互补参考场对和至少有一个场被标记为“用于短期参考”的非成对参考场的总数。令numLongTerm为参考图像，互补参考场对和至少有一个场被标记为“用于长期参考”的非成对参考场的总数。

— 当numShortTerm+numLongTerm等于num_ref_frames时，numShortTerm大于0的条件应该满足，并且短期参考图像，互补参考场对和具有最小FrameNumWrap值的非成对参考图像被标记为“不用于参考”。当它是帧或者互补参考场对时，它的两个场都应该被标记为“不用于参考”。

8.2.5.4 自适应存储器控制的已解码图像标记过程

当 adaptive_ref_pic_marking_mode_flag 等于 1 时，调用本过程。

值为 1 到 6 的 memory_management_control_operation 命令在当前图像已经解码后按照它们在比特流中出现的先后顺序进行处理。对 memory_management_control_operation 命令中的每一个，根据 memory_management_control_operation 的值，来调用在 8.2.5.4.1 至 8.2.5.4.6 中所定义的某一个过程。memory_management_control_operation 等于 0 的命令说明 memory_management_control_operation 命令已经结束。

存储器管理控制操作可以应用到以下图像类型：

— 如果field_pic_flag等于0，则memory_management_control_operation应用于所说明的帧或互补参考场对。

— 否则（field_pic_flag等于1），memory_management_control_operation应用于单个参考场。

8.2.5.4.1 将短期图像标记为“未用于参考”的过程

当 memory_management_control_operation 等于 1 时，调用本过程。

设 picNumX 由下式得到：

$$picNumX = CurrPicNum - (difference\_of\_pic\_nums\_minus1 + 1). \quad (8\text{-}40)$$

根据 field_pic_flag 的不同取值，picNumX 的值用于将一个短期图像标记为“未用于参考”，如下：

— 如果field_pic_flag等于0，则这个由picNumX指定的短期参考帧或参考图像场对和它的两个场都被标记为“未用于参考”。

— 否则（field_pic_flag等于1），这个由picNumX指定的短期参考场被标记为“未用于参考”。当这个参考图像是一个参考帧或一个互补参考场对的一部分时，这个参考帧或互补参考场对也将被标记为“未用于参考”， 但是其他场的标记不能改变。

8.2.5.4.2 将长期参考图像标记为“不用于参考”的过程

当 memory_management_control_operation 等于 2 时，调用本过程。

根据 field_pic_flag 的不同取值，LongTermPicNum 的值用于将一个长期参考图像标记为“不用于参考”，规定如下：

— 如果field_pic_flag为0，则LongTermPicNum等于long_term_pic_num的长期参考帧或长期互补参考场对和它的两个场都将被标记为“不用于参考”。

— 否则（field_pic_flag为1），LongTermPicNum等于long_term_pic_num的长期参考场标记为“不用于参考”。当这个参考图像是参考帧或互补参考场对的一部分时，这个参考帧或互补参考场对也将被标记为“不用于参考”，但是其他场的标记不能改变。

8.2.5.4.3 赋LongTermFrameIdx给短期参考图像的过程

当 memory_management_control_operation 等于 3 时，调用本过程。

给定语法元素 difference_of_pic_nums_minus1，变量 picNumX 通过 8.2.5.4.1 节得到的。picNumX 应指向标记为“用于短期参考”但不是“不存在”的一个帧或互补参考场对或非成对参考场。

当值为 long_term_frame_idx 的 LongTermFrameIdx 赋给了一个长期参考帧或长期互补参考场对时，则这个互补参考场对和它的两个场都要被标记为“不用于参考”。 当 LongTermFrameIdx 已经被赋给了一个非成对参考场并且这个场不是被 PicNumX 确定的图像的互补参考场时，则该场被标记为“不用于参考”。

取决于 field_pic_flag 的值，LongTermFrameIdx 的值用于将一个“用于短期参考”图像标记为“用于长期参考”如下：

— 如果field_pic_flag等于0，则PicNumX所确定的短期参考帧或互补参考场对和它的两个场的标记都将从“用于短期参考”转化为“用于长期参考”并且LongTermFrameIdx的值等于long_term_frame_idx 。

— 否则（field_pic_flag等于1），PicNumX所确定的短期参考场的标记将从“用于短期参考”变为“用于长期参考”，并且LongTermFrameIdx的值等于long_term_frame_idx。当该场是一个参考帧和互补参考场对的一部分，并且同一个参考帧或互补参考场对中的另外一场被标记为“用于长期参考”，则该参考帧或互补参考场对也将被标记为“用于长期参考”，且设置LongTermFrameIdx等于long_term_frame_idx。

8.2.5.4.4 MaxLongTermFrameIdx的解码过程

当 memory_management_control_operation 等于 4 时，调用本过程。

所有 LongTermFrameIdx 大于 max_long_term_frame_idx_plus1 − 1，并且标记为"用于长期参考"的图像将被标记为 “不用于参考”。

变量 MaxLongTermFrameIdx 的值通过以下过程得到：

— 如果max_long_term_frame_idx_plus1等于0，MaxLongTermFrameIdx将设为“非长期帧索引”。

— 否则（max_long_term_frame_idx_plus1 的值大于 0），MaxLongTermFrameIdx 将设为 max_long_term_frame_idx_plus1 − 1。

注 — 等于4的存储管理控制操作命令可以用来标记一个长期参考图像为“不用于参考”。传送max_long_term_frame_idx_plus1的频率在本建议书 | 国际标准中不做规定。然而，编码器应该在收到错误信息时发送一个等于4的存储管理控制操作的命令，例如帧内更新请求消息。

8.2.5.4.4.1 将所有参考图像标记为“不用于参考”并将MaxLongTermFrameIdx 设置为“非长期帧索引”

当 memory_management_control_operation 等于 5 时，调用本过程。

所有参考图像标记为“不用于参考”并且变量 MaxLongTermFrameIdx 设置为“非长期帧索引”。

8.2.5.4.5 为当前图像分配长期参考索引值的过程

当 memory_management_control_operation 等于 6 时，调用本过程。

当值等于 long_term_frame_idx 的变量 LongTermFrameIdx 已经被赋给了长期参考帧或长期互补参考场对时，则这个帧或互补参考场对和它的两个场都要被标记为“不用于参考”。当 LongTermFrameIdx 已经赋给了一个非成对参考场并且这个场不是当前图像的互补参考场时，该场被标记为“不用于参考”。

当前图像被标记为"用于长期参考" 并且将 LongTermFrameIdx 赋值为 long_term_frame_idx 。

当 field_pic_flag 等于 0 时，两个场都被标记为"用于长期参考"并且 LongTermFrameIdx 等于 long_term_frame_idx。

当 field_pic_flag 等于 1，并且当前图像在解码顺序上是一个互补参考场对的第二场（按照解码顺序）且其第一场也标记为"用于长期参考"时，也将该互补参考场对标记为"用于长期参考"并且 LongTermFrameIdx 等于 long_term_frame_idx 。

在对当前解码参考图像进行标记后，至少有一个场标记为“用于参考”的帧的总数，加上至少一个场标记为“用于参考”的互补参考场对的个数，再加上标记为“用于参考”的非成对场数，不应大于 Max(num_ref_frames, 1)。

注 — 在某些情况下，上述语句意味着对于等于6的语法元素memory_management_control_operation 与等于1、2或4的 memory_management_control_operation在解码参考图像标记语法中出现的顺序的约束。

8.3 帧内预测过程

宏块类型为 I 和 SI 时调用该过程。

本过程的输入是去块效应滤波过程之前的重建样点值，以及对于 Intra_N×N 预测模式（其中 N×N 等于 4×4 或 8×8）下相邻宏块的 IntraNxNPredMode 值。

本过程的输出为：

— 如果宏块预测类型为Intra_4x4 或 Intra_8x8时，输出为去块效应滤波过程之前的重建样点值和宏块 $pred_C$中的色度预测样点值（当chroma_format_idc不为0时），其中C为Cb或Cr。

— 否则，如果mb_type不等于I_PCM，输出为宏块中像素的亮度预测值$pred_L$以及宏块中的色度预测样点值$pred_C$（当chroma_format_idc不为0时），其中C为Cb或Cr。

— 否则（如果mb_type等于I_PCM），输出为去块效应滤波过程之前的重建亮度和色度样点值（当chroma_format_idc不为0时）。

MvCnt 的值设置为 0。

根据 mb_type 值的不同，应用以下过程：

— 如果mb_type等于I_PCM，调用8.3.5节定义的过程；

— 否则（如果mb_type不等于I_PCM），应用以下过程：

— 对于亮度分量，帧内预测模式下的解码过程如下：

  — 如果宏块预测模式为Intra_4x4，应用8.3.1节定义的过程；

  — 否则，如果宏块预测模式为Intra_8x8，应用8.3.2节定义的过程；

  — 否则（如果宏块预测模式为Intra_16x16），应用8.3.3节定义的过程；

  — 8.3.4节定义了帧内预测模式下色度分量的的解码过程，仅当chroma_format_idc不等于0（单色模式）时，才调用该过程。

帧内预测过程中使用的样点值没有经过任何去块效应滤波操作的更改。

8.3.1 亮度样点的Intra_4x4预测过程

当宏块预测模式为 Intra_4x4 时，调用本过程。

本过程的输入是相邻宏块或宏块对的 Intra4x4PredMode (如果可用)值或 Intra8x8PredMode(如果可用)值。

一个宏块的亮度分量包括 16 个 4x4 亮度样点块，这些块通过调用 6.4.3 节定义的 4x4 亮度块逆扫描过程进行逆扫描。

对于一个 luma4x4BlkIdx = 0..15 的宏块亮度分量的所有 4x4 块，调用 8.3.1.1 节所规定的 Intra4x4PredMode 的获取过程，在这个过程中，相邻宏块的 luma4x4BlkIdx 以及已经得到的（按照解码顺序）Intra4x4PredMode 和 Intra8x8PredMode 作为输入，输出是 Intra4x4PredMode[luma4x4BlkIdx]。

对于每一个使用 luma4x4BlkIdx = 0..15 作为索引的 4x4 亮度块：

1．调用 8.3.1.2 节所规定的 Intra_4x4 样点预测过程被调用，输入是 luma4x4BlkIdx 和去块效应滤波过程之前（按照解码顺序）根据相邻亮度块构建的样点，输出是 Intra_4x4 亮度预测样点值 $pred4x4_L[x, y]$ ，x, y = 0..3。

2．调用 6.4.3 定义的逆 4x4 亮度扫描过程，可以得到当前宏块中索引为 luma4x4BlkIdx 的 4x4 亮度块的左上角样点的位置，这个过程的输入是 luma4x4BlkIdx，输出值赋给(xO, yO)， x, y = 0..3。

$$pred_L[xO + x, yO + y] = pred4x4_L[x, y] \quad (8\text{-}41)$$

3．调用 8.5 中定义的位于去块效应滤波器之前的变换系数解码过程和图像构建过程，输入是 $pred_L$ 和 luma4x4BlkIdx，输出是为当前 4x4 亮度块 S'_L 构建的样点值。

8.3.1.1 Intra4x4PredMode的推导过程

本过程的输入是 4x4 亮度块的索引 luma4x4BlkIdx 和先前（按照解码顺序）已经得到的相邻宏块的变量序列 Intra4x4PredMode (如果可用) 和 Intra8x8PredMode(如果可用)。

本过程的输出是变量 Intra4x4PredMode[luma4x4BlkIdx]。

表 8-2 定义了 Intra4x4PredMode[luma4x4BlkIdx]的值和相应的名称

表 8-2－Intra4x4PredMode[luma4x4BlkIdx]以及相关名称的规范

Intra4x4PredMode[luma4x4BlkIdx]	Intra4x4PredMode[luma4x4BlkIdx]名称
0	Intra_4x4_Vertical (预测模式)
1	Intra_4x4_Horizontal (预测模式)
2	Intra_4x4_DC (预测模式)
3	Intra_4x4_Diagonal_Down_Left (预测模式)
4	Intra_4x4_Diagonal_Down_Right (预测模式)
5	Intra_4x4_Vertical_Right (预测模式)
6	Intra_4x4_Horizontal_Down (预测模式)
7	Intra_4x4_Vertical_Left (预测模式)
8	Intra_4x4_Horizontal_Up (预测模式)

图 8-1 阐述了 Intra4x4PredMode[luma4x4BlkIdx]的值为 0、1、2、3、4、5、6、7 和 8，这些值分别代表不同预测方向，如图 8-1 所示。

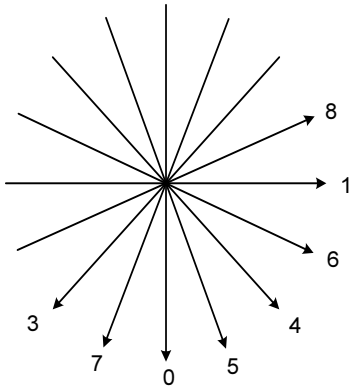


图 8-1－Intra_4x4各预测模式的方向（资料性）

Intra4x4PredMode[luma4x4BlkIdx] 由以下方式得到：

— 调用6.4.8.3节所定义的过程，输入是luma4x4BlkIdx ，输出分别赋值给mbAddrA、luma4x4BlkIdxA、mbAddrB以及luma4x4BlkIdxB；

— 变量dcPredModePredictedFlag由以下方式得到：

— 如果以下任何条件为真，dcPredModePredictedFlag设为1

— 宏块地址为mbAddrA的宏块不可用；

— 宏块地址为mbAddrB的宏块不可用；

— 宏块地址为mbAddrA的宏块可用，并且以帧间预测方式进行编码、constrained_intra_pred_flag为1；

— 宏块地址为mbAddrB的宏块可用，并且以帧间预测方式进行编码、constrained_intra_pred_flag为1；

— 否则，dcPredModePredictedFlag设为0。

— 变量intraMxMPredModeN（N=A或B）由以下方式得到：

— 如果dcPredModePredictedFlag为1，或者宏块地址为mbAddrN的宏块不是以Intra_4x4或Intra_8x8预测方式编码的，则intraMxMPredModeN设为2(Intra_4x4_DC 预测模式)；

— 否则（dcPredModePredictedFlag为0并且（宏块地址为mbAddrN的宏块是Intra_4x4预测方式或者宏块地址为mbAddrN的宏块是以Intra_8x8预测方式编码的）），应用以下规则：

— 如果宏块地址为mbAddrN的宏块是以Intra_4x4预测方式编码，则intraMxMPredModeN设为 Intra4x4PredMode[luma4x4BlkIdxN]，这里Intra4x4PredMode是分配给宏块mbAddrN的变量阵列；

— 否则（宏块地址为mbAddrN的宏块是以Intra_8x8预测方式编码的），intraMxMPredModeN设为 Intra8x8PredMode[luma4x4BlkIdxN >> 2]，这里Intra8x8PredMode是分配给宏块mbAddrN的变量阵列；

— Intra4x4PredMode[luma4x4BlkIdx] 由以下程序得到：

```
predIntra4x4PredMode = Min( intraMxMPredModeA, intraMxMPredModeB )
if( prev_intra4x4_pred_mode_flag[ luma4x4BlkIdx ] )
    Intra4x4PredMode[ luma4x4BlkIdx ] = predIntra4x4PredMode
else                                                                    (8-42)
    if( rem_intra4x4_pred_mode[ luma4x4BlkIdx ] < predIntra4x4PredMode )
        Intra4x4PredMode[ luma4x4BlkIdx ] = rem_intra4x4_pred_mode[ luma4x4BlkIdx ]
    else
        Intra4x4PredMode[ luma4x4BlkIdx ] = rem_intra4x4_pred_mode[ luma4x4BlkIdx ] + 1
```

8.3.1.2 Intra_4x4 样点预测

对宏块中每个预测模式为 Intra_4x4 的 4x4 亮度块的预测模式调用本过程，位于本过程之后的是 4x4 亮度块去块效应之前的变换系数解码过程和图像构建过程。

本过程的输入是 4x4 亮度块的索引 luma4x4BlkIdx。

本过程的输出是索引为 luma4x4BlkIdx 的 4x4 亮度块的预测样点值 $pred4x4_L[x, y]$，x, y = 0..3。

当前宏块内索引为 luma4x4BlkIdx 的 4x4 亮度块的左上角样点的位置是根据 6.4.3 所定义的逆 4x4 亮度扫描过程得到，这个过程的输入是 luma4x4BlkIdx，输出是(xO, yO)。

去块效应滤波之前的 13 个相邻重建亮度样点值 p[x, y] 由以下方式得到，x = –1, y = –1..3 并且 x = 0..7, y = –1：

— 亮度位置(xN, yN)规定如下：

xN = xO + x (8-43)

yN = yO + y (8-44)

— 调用6.4.9所定义的相邻位置的推导过程，输入是(xN, yN)，输出是mbAddrN和 (xW, yW)。

— 每一个样点p[x, y]由以下方式得到，其中x = –1, y = –1..3并且 x = 0..7, y = –1：

— 如果以下条件中有任何一个成立，则样点p[x, y]被标记为“不可用于Intra_4x4预测”：

— mbAddrN；

— 宏块mbAddrN以帧间预测方式进行编码并且constrained_intra_pred_flag为1；

— 宏块mbAddrN的mb_type等于SI，constrained_intra_pred_flag等于1，并且当前宏块的mb_type不等于SI；

— x大于3并且luma4x4BlkIdx等于3或11。

— 否则，p[x, y]被标记为“可用于Intra_4x4预测”，并且p[x, y]的值等于宏块mbAddrN中(xW, yW)亮度位置上的亮度样点。

当样点 p[x, –1]，x = 4..7 被标记为“不可用于 Intra_4x4 预测”，并且 p[3, –1]被标记为“可用于 Intra_4x4 预测”时，用 p[3, –1]的样点值代替 p[x, –1]，x = 4..7 的样点值，并且 p[x, –1]，x = 4..7 被标记为“可用于 for Intra_4x4 预测”。

注 — 在下一个块解码之前，每个块都被认为已经加入到图像阵列中。

根据 Intra4x4PredMode[luma4x4BlkIdx]的不同值，调用 8.3.1.2.1 到 8.3.1.2.9 所规定的 Intra_4x4 预测模式之一。

8.3.1.2.1 Intra_4x4_Vertical预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 0 时，使用这种 Intra_4x4 预测模式。

这种预测方式仅能在样点 p[x, –1]，x = 0..3 被标记为“可用于 Intra_4x4 预测”时使用。

样点预测值 $pred4x4_L[x, y]$，x, y = 0..3 如下：

$pred4x4_L[x, y] = p[x, –1]$，x, y = 0..3 (8-45)

8.3.1.2.2 Intra_4x4_Horizontal预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 1 时，调用这种 Intra_4x4 预测模式。

这种预测方式仅能在样点 p[–1, y]，y = 0..3 被标记为“可用于 Intra_4x4 预测”时使用。

样点预测值 $pred4x4_L[x, y]$，x, y = 0..3 如下：

$$pred4x4_L[x, y] = p[-1, y] \quad x,y = 0..3 \tag{8-46}$$

8.3.1.2.3 Intra_4x4_DC预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 2 时，调用这种 Intra_4x4 预测模式。

样点预测值 $pred4x4_L[x, y]$，x, y = 0..3 按照如下方式得到：

— 如果所有样点p[x, −1]，x = 0..3和p[−1, y]，y = 0..3被标记为“可用于Intra_4x4预测”，则样点预测值 $pred4x4_L[x, y]$，x, y = 0..3以如下方式得到：

$$pred4x4_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + 4) >> 3 \tag{8-47}$$

— 否则，如果p[x, −1]，x = 0..3中的任何样点被标识为“不可用于Intra_4x4预测”，并且所有样点 p[−1, y]，y = 0..3被标记为“可用于Intra_4x4预测”，则样点预测值$pred4x4_L[x, y]$，x, y = 0..3以如下方式得到：

$$pred4x4_L[x, y] = (p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + 2) >> 2 \tag{8-48}$$

— 否则，如果p[−1, y]，y = 0..3中的任何样点被标识为“不可用于Intra_4x4预测”，并且所有样点 p[x, −1]，x = 0..3被标记为“可用于Intra_4x4预测”，样点预测值$pred4x4_L[x, y]$，x, y = 0..3以如下方式得到：

$$pred4x4_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + 2) >> 2 \tag{8-49}$$

— 否则（p[x, −1]，x = 0..3中的某些样点被标识为“不可用于Intra_4x4预测”，并且p[−1, y]，y = 0..3中的某些样点被标识为“不可用于Intra_4x4预测”），则样点预测值$pred4x4_L[x, y]$、x, y = 0..3以如下方式得到：

$$pred4x4_L[x, y] = (1 << (BitDepth_Y - 1)) \tag{8-50}$$

注 — 一个4x4亮度块总是可以使用这种预测方式。

8.3.1.2.4 Intra_4x4_Diagonal_Down_Left预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 3 时，调用这种 Intra_4x4 预测模式。

这种预测方式仅能在样点 p[x, −1]，x = 0..7 被标记为“可用于 Intra_4x4 预测”时使用。

样点预测值 $pred4x4_L[x, y]$、x, y = 0..3 以如下方式得到：

— 如果x等于3并且y等于3，

$$pred4x4_L[x, y] = (p[6, -1] + 3 * p[7, -1] + 2) >> 2 \tag{8-51}$$

— 否则（x,y均不等于3），

$$pred4x4_L[x, y] = (p[x + y, -1] + 2 * p[x + y + 1, -1] + p[x + y + 2, -1] + 2) >> 2 \tag{8-52}$$

8.3.1.2.5 Intra_4x4_Diagonal_Down_Right预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 4 时，调用这种 Intra_4x4 预测模式。

这种预测方式仅能在样点 p[x, −1]，x = 0..3 和 p[−1, y]，y = −1..3 被标记为“可用于 Intra_4x4 预测”时使用。

样点预测值 $pred4x4_L[x, y]$，x, y = 0..3 以如下方式得到：

— 如果x大于y，则

$pred4x4_L[x, y] = (p[x - y -2, -1] + 2 * p[x- y -1, -1] + p[x-y, -1] + 2) >> 2$ (8-53)

— 否则，如果x小于y

$pred4x4_L[x, y] = (p[-1, y - x - 2] + 2 * p[-1, y - x - 1] + p[-1, y - x] + 2) >> 2$ (8-54)

— 否则（x等于y）

$pred4x4_L[x, y] = (p[0, -1] + 2 * p[-1, -1] + p[-1, 0] + 2) >> 2$ (8-55)

8.3.1.2.6 Intra_4x4_Vertical_Right预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 5 时，调用这种 Intra_4x4 预测模式。

这种预测方式仅能在样点 p[x, −1]，x = 0..3 和 p[−1, y]，y = −1..3 被标记为“可用于 Intra_4x4 预测”时使用。

设 zVR= 2 * x − y。

预测样点 $pred4x4_L[x, y]$，x、y=0,3 的值由正式导出：

— 如果zVR等于0、2、4或6，则

$pred4x4_L[x, y] = (p[x - (y >> 1) -1, -1] + p[x - (y >> 1), -1] + 1) >> 1$ (8-56)

— 否则，如果zVR等于1、3或5，则

$pred4x4_L[x, y] = (p[x - (y >> 1) -2, -1] + 2 * p[x - (y >> 1) - 1, -1] + p[x - (y >> 1), -1] + 2) >> 2$ (8-57)

— 否则，如果zVR等于−1，则

$pred4x4_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2$ (8-58)

— 否则（zVR等于−2或−3），

$pred4x4_L[x, y] = (p[-1, y - 1] + 2 * p[-1, y - 2] + p[-1, y - 3] + 2) >> 2$ (8-59)

8.3.1.2.7 Intra_4x4_Horizontal_Down预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 6 时，调用这种 Intra_4x4 预测模式。

这种预测方式仅能在样点 p[x, −1]，x = 0..3 和 p[−1, y]，y = −1..3 被标记为“可用于 Intra_4x4 预测”时使用。

设 zHD= 2 * y − x。

样点预测值 $pred4x4_L[x, y]$，x, y = 0..3 以如下方式得到：

— 如果zHD等于0、2、4或6，则

$pred4x4_L[x, y] = (p[-1, y - (x >> 1) - 1] + p[-1, y - (x >> 1)] + 1) >> 1$ (8-60)

— 否则，如果zHD等于1、3或5，则

$pred4x4_L[x, y] = (p[-1, y - (x >> 1) - 2] + 2 * p[-1, y - (x >> 1) - 1] + p[-1, y - (x >> 1)] + 2) >> 2$ (8-61)

— 否则，如果zHD等于−1，

$pred4x4_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2$ (8-62)

— 否则（zHD等于−2或−3），则

$pred4x4_L[x, y] = (p[x - 1, -1] + 2 * p[x - 2, -1] + p[x - 3, -1] + 2) >> 2$ (8-63)

8.3.1.2.8 Intra_4x4_Vertical_Left预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 7 时，调用这种 Intra_4x4 预测模式。

这种预测方式仅能在样点 p[x, −1]，x = 0..7 被标记为“可用于 Intra_4x4 预测”时使用。

样点预测值 $pred4x4_L[x, y]$，x, y = 0..3 以如下方式得到：

— 如果y等于0或2，

$pred4x4_L[x, y] = (p[x + (y >> 1), -1] + p[x + (y >> 1) + 1, -1] + 1) >> 1$ (8-64)

— 否则（y等于1或3）

$pred4x4_L[x, y] = (p[x + (y >> 1), -1] + 2 * p[x + (y >> 1) + 1, -1] + p[x + (y >> 1) + 2, -1] + 2) >> 2$ (8-65)

8.3.1.2.9 Intra_4x4_Horizontal_Up预测模式的规范

在 Intra4x4PredMode[luma4x4BlkIdx]等于 8 时，调用这种 Intra_4x4 预测模式。

这种预测方式仅能在样点 p[−1, y]，y = 0..3 被标记为“可用于 Intra_4x4 预测”时使用。

设变量 zHU 等于 x + 2*y。

样点预测值 $pred4x4_L[x, y]$，x, y = 0..3 以如下方式得到：

— 如果zHU等于0、2或4，

$pred4x4_L[x, y] = (p[-1, y + (x >> 1)] + p[-1, y + (x >> 1) + 1] + 1) >> 1$ (8-66)

— 否则，如果zHU等于1或3，

$pred4x4_L[x, y] = (p[-1, y + (x >> 1)] + 2 * p[-1, y + (x >> 1) + 1] + p[-1, y + (x >> 1) + 2] + 2) >> 2$ (8-67)

— 否则，如果zHU等于5，

$pred4x4_L[x, y] = (p[-1, 2] + 3 * p[-1, 3] + 2) >> 2$ (8-68)

— 否则（zHU大于5）

$pred4x4_L[x, y] = p[-1, 3]$ (8-69)

8.3.2 亮度样点的Intra_8x8预测过程

当宏块预测模式为 Intra_8x8 时，调用本过程。

本过程的输入是来自相邻宏块或宏块对的 Intra4x4PredMode (如果可用)的值或 Intra8x8PredMode(如果可用)的值。

本过程的输出是 8x8 亮度样点阵列，它们是宏块 $pred_L$ 的预测样点的 16x16 亮度样点阵列的一部分。

一个宏块的亮度分量包括 4 个 8x8 亮度样点块。这些块使用 6.4.4 节规定的逆 8x8 亮度块扫描过程进行逆扫描。

对于一个宏块亮度分量中所有 luma8x8BlkIdx = 0..3 的 8x8 亮度块，调用 8.3.2.1 节规定的 Intra8x8PredMode 的推导过程，在这个过程中，luma8x8BlkIdx 以及已经得到的（按照解码顺序）相邻宏块的 Intra8x8PredMode 和 Intra8x8PredMode 作为输入，输出是变量 Intra8x8PredMode[luma8x8BlkIdx]。

对于每一个索引号为 luma8x8BlkIdx = 0..3 的 8x8 亮度块，应用如下规则。

— 调用8.3.2.2节所定义的Intra_8x8样点预测过程，输入是luma8x8BlkIdx 和去块效应滤波过程之前的构建的样点（按照解码顺序），输出是Intra_8x8亮度预测样点值$pred8x8_L$[x, y]， x, y = 0..7。

— 调用6.4.4节规定的逆8x8亮度扫描过程，可以得到索引为luma8x8BlkIdx的8x8亮度块的左上角样点的位置，这个过程的输入是luma8x8BlkIdx，输出赋值给(xO, yO)，x, y = 0..7。

$pred_L$[xO + x, yO + y] = $pred8x8_L$[x, y] (8-70)

— 调用8.5节所阐述的去块效应滤波器之前的变换系数解码过程和图像重构建过程，输入是$pred_L$和luma8x8BlkIdx，输出是当前8x8 亮度块 S'_L的构建样点值。

8.3.2.1 Intra8x8PredMode的推导过程

本过程的输入是 8x8 亮度块的索引 luma8x8BlkIdx 和已经得到的（按照解码顺序）相邻宏块的变量阵列 Intra4x4PredMode (如果可用) 和 Intra8x8PredMode(如果可用)。

本过程的输出是变量 Intra8x8PredMode[luma8x8BlkIdx]。

表 8-3 规定了 Intra8x8PredMode[luma8x8BlkIdx]的取值和相关名称。

表 8-3－Intra8x8PredMode[luma8x8BlkIdx]的取值和相关名称

Intra8x8PredMode[luma8x8BlkIdx]	Intra8x8PredMode[luma8x8BlkIdx]的名称
0	Intra_8x8_Vertical (预测模式)
1	Intra_8x8_Horizontal (预测模式)
2	Intra_8x8_DC (预测模式)
3	Intra_8x8_Diagonal_Down_Left (预测模式)
4	Intra_8x8_Diagonal_Down_Right (预测模式)
5	Intra_8x8_Vertical_Right (预测模式)
6	Intra_8x8_Horizontal_Down (预测模式)
7	Intra_8x8_Vertical_Left (预测模式)
8	Intra_8x8_Horizontal_Up (预测模式)

Intra8x8PredMode[luma8x8BlkIdx]由以下过程得到。

— 调用 6.4.8.2 节规定的过程，输入是 luma8x8BlkIdx，输出是 mbAddrA、luma8x8BlkIdxA、 mbAddrB 以及 luma8x8BlkIdxB；

— 变量dcPredModePredictedFlag由以下方式得到：

  — 如果以下任何条件为真，dcPredModePredictedFlag设为1

    — 宏块地址为mbAddrA的宏块不可用；

    — 宏块地址为mbAddrB的宏块不可用；

    — 宏块地址为mbAddrA的宏块可用，并且以帧间预测方式进行编码，constrained_intra_pred_flag为1；

    — 宏块地址为mbAddrB的宏块可用，并且以帧间预测方式进行编码、constrained_intra_pred_flag为1；

  — 否则，dcPredModePredictedFlag设为0。

— 对于N=A或B，变量intraMxMPredModeN由以下方式得到：

— 如果dcPredModePredictedFlag等于1或者（宏块地址为mbAddrN的宏块不是以Intra_4x4或Intra_8x8预测方式编码），则intraMxMPredModeN设为2(Intra_8x8_DC 预测模式)；

— 否则（dcPredModePredictedFlag为0并且（宏块地址为mbAddrN的宏块是Intra_4x4预测方式编码或者宏块地址为mbAddrN的宏块是Intra_8x8预测方式编码）），应用如下规则：

— 如果地址为mbAddrN的宏块是Intra_8x8预测方式编码，则intraMxMPredModeN设为等于Intra8x8PredMode[luma8x8BlkIdxN]，这里Intra8x8PredMode是分配给宏块mbAddrN的变量阵列；

— 否则（宏块地址为mbAddrN的宏块是以Intra_4x4预测方式编码），intraMxMPredModeN按照下式得到，其中Intra4x4PredMode是为地址为mbAddrN的宏块分配的变量阵列：

intraMxMPredModeN = Intra4x4PredMode[luma8x8BlkIdxN * 4 + n] (8-71)

这里 Intra4x4PredMode 是宏块地址为 mbAddrN 的宏块的变量序列。

变量 n 由以下方式得到：

— 如果N为A，根据变量MbaffFrameFlag、变量luma8x8BlkIdx、当前宏块和宏块地址mbAddrN的不同取值，应用如下规定：

— 如果MbaffFrameFlag等于1，当前宏块是帧编码宏块，宏块mbAddrN是场编码宏块、luma8x8BlkIdx等于2，则n等于3；

— 否则（MbaffFrameFlag等于0，或当前宏块是场编码宏块，或宏块mbAddrN是帧编码宏块、或luma8x8BlkIdx不等于2），则n等于1。

— 否则（N等于B），n设为等于2。

— 最后，给定 intraMxMPredModeA 和 intraMxMPredModeB，变量 Intra8x8PredMode[luma8x8BlkIdx]由以下方式得到：

```
predIntra8x8PredMode = Min( intraMxMPredModeA, intraMxMPredModeB )
if( prev_intra8x8_pred_mode_flag[ luma8x8BlkIdx ] )
    Intra8x8PredMode[ luma8x8BlkIdx ] = predIntra8x8PredMode
else                                                                    (8-72)
    if( rem_intra8x8_pred_mode[ luma8x8BlkIdx ] < predIntra8x8PredMode )
        Intra8x8PredMode[ luma8x8BlkIdx ] = rem_intra8x8_pred_mode[ luma8x8BlkIdx ]
    else
        Intra8x8PredMode[ luma8x8BlkIdx ] = rem_intra8x8_pred_mode[ luma8x8BlkIdx ] + 1
```

8.3.2.2 Intra_8x8样点预测

为一个宏块中的每个预测模式为 Intra_8x8 的 8x8 亮度块调用本过程，在本过程之后是变换系数解码过程和 8x8 亮度块去块效应之前的图像构建过程。

本过程的输入是 8x8 亮度块的索引 luma8x8BlkIdx。

本过程的输出是索引为 luma8x8BlkIdx 的 8x8 亮度块的预测样点值 $pred8x8_L[x, y]$, x, y = 0..7。

当前宏块内索引为 luma8x8BlkIdx 的 8x8 亮度块的左上角样点的位置是由 6.4.4 所定义的逆 8x8 亮度块扫描过程得到的，输入是 luma8x8BlkIdx，输出赋给(xO, yO)。

去块效应滤波过程之前构建的 25 个相邻的亮度样点值 p[x, y] 由以下方式得到，这里 x = –1, y = –1..7 并且 x = 0..15, y = –1：

— 亮度位置(xN, yN)通过如下公式得到：

xN = xO + x (8-73)

yN = yO + y (8-74)

— 调用6.4.9所定义的相邻位置的推导过程，输入是(xN, yN)，输出是mbAddrN和 (xW, yW)。

— 每一个样点p[x, y]由以下方式得到，这里x = –1, y = –1..7并且 x = 0..15, y = –1：

— 如果以下条件中有任何一个成立，则样点p[x, y]被标记为“不可用于Intra_8x8预测”：

— mbAddrN不可用；

— 宏块mbAddrN以帧间预测方式进行编码，并且constrained_intra_pred_flag为1；

— 否则，p[x, y]被标记为“在 Intra_8x8预测时可用”，宏块mbAddrN中(xW, yW)亮度位置的亮度样点值赋给p[x, y]。

当样点 p[x, –1]，x = 8..15 被标记为“在 Intra_8x8 预测时不可用”，并且 p[7, –1]被标记为“可用于ntra_8x8 预测”时，用 p[7, –1]的样点值代替 p[x, –1]、x =8..15 的样点值，p[x, –1]、x = 8..15 被标记为“可用于 Intra_8x8 预测”。

注 — 在解码下一个块之前，每个块都被认为已经加入到图像阵列中。

调用 8.3.2.2.1 所定义的用于 Intra_8x8 样点预测的参考样点滤波过程，在这个过程中，p[x, y] x = –1, y = –1..7 和 x = 0..15, y = –1 (如果可用) 作为输入， p'[x, y] x = –1, y = –1..7 和 x = 0..15, y = –1 作为输出。

根据 Intra8x8PredMode[luma8x8BlkIdx]，调用 8.3.2.2.2 到 8.3.2.2.10 所定义的 Intra_8x8 预测模式之中的一个。

8.3.2.2.1 Intra_8x8样点预测的参考样点滤波过程

本过程的输入是 Intra_8x8 样点预测的参考样点 p[x, y] x = –1，y = –1..7 和 x = 0..15, y = –1（如果可用）。

本过程的输出是 Intra_8x8 样点预测时的滤波参考样点 p'[x, y] x = –1, y = –1..7 和 x = 0..15, y = –1。

当 p[x, –1]，x = 0..7 的所有样点都被标记为“可用于 Intra_8x8 预测”时，应用以下过程：

— p'[0, –1]的值由以下方式得到：

— 如果p[–1, –1]被标记为“可用于Intra_8x8预测”，则

p'[0, –1] = (p[–1, –1] + 2 * p[0, –1] + p[1, –1] + 2) >> 2 (8-75)

— 否则（p[–1, –1]被标记为“不可用于Intra_8x8预测时”，）则

p'[0, –1] = (3 * p[0, –1] + p[1, –1] + 2) >> 2 (8-76)

— p'[x, –1]，x = 1..7的值由以下方式得到：

p'[x, –1] = (p[x–1, –1] + 2 * p[x, –1] + p[x+1, –1] + 2) >> 2 (8-77)

当 p[x, –1]，x = 7..15 的所有样点都被标记为“可用于 Intra_8x8 预测”时，应用以下过程。

— p'[x, –1]，x = 8..14的值由以下方式得到：

p'[x, –1] = (p[x–1, –1] + 2 * p[x, –1] + p[x+1, –1] + 2) >> 2 (8-78)

— p'[15, –1]的值由以下方式得到：

p'[15, –1] = (p[14, –1] + 3 * p[15, –1] + 2) >> 2 (8-79)

当 p[−1, −1]被标记为“可用于 Intra_8x8 预测”时，p'[−1, −1]的值由以下方式得到：

— 如果p[0, −1] 被标记为“不可用于Intra_8x8预测”，或 p[−1, 0] 被标记为“不可用于Intra_8x8预测”，以下方式适用：

— 如果p[0, −1]被标记为“可用于Intra_8x8预测”，则p[−1, −1]为

$$p'[-1, -1] = (3 * p[-1, -1] + p[0, -1] + 2) >> 2 \quad (8\text{-}80)$$

— 否则（p[0, −1] 被标记为“不可用于Intra_8x8预测”，并且p[−1, 0] 被标记为“可用于Intra_8x8预测”），则p[−1, −1]为

$$p'[-1, -1] = (3 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad (8\text{-}81)$$

— 否则（样点p[0, −1]标记为“可用于Intra_8x8预测”并且样点p[−1, 0]标记为“可用于Intra_8x8预测”），p'[−1, −1]通过如下公式得到：

$$p'[-1, -1] = (p[0, -1] + 2 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad (8\text{-}82)$$

当所有 p[−1, y]，y = 0..7 的样点被标记为“可用于是 Intra_8x8 预测”时，应用如下规则：

— p'[−1, 0]的值由以下方式得到：

— 如果p[−1, −1]被标记为“可用于Intra_8x8预测”，则 p'[−1, 0] 为

$$p'[-1, 0] = (p[-1, -1] + 2 * p[-1, 0] + p[-1, 1] + 2) >> 2 \quad (8\text{-}83)$$

— 否则（如果p[−1, −1]被标记为“不可用于Intra_8x8预测”），则 p'[−1, 0]为

$$p'[-1, 0] = (3 * p[-1, 0] + p[-1, 1] + 2) >> 2 \quad (8\text{-}84)$$

— p'[−1, y]，y = 1..6的值为：

$$p'[-1, y] = (p[-1, y-1] + 2 * p[-1, y] + p[-1, y+1] + 2) >> 2 \quad (8\text{-}85)$$

— p'[−1, 7]的值为：

$$p'[-1 ,7] = (p[-1, 6] + 3 * p[-1, 7] + 2) >> 2 \quad (8\text{-}86)$$

8.3.2.2.2 Intra_8x8_Vertical预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 0 时，调用这种 Intra_8x8 预测模式。

仅在样点 p[x, −1]，x = 0..7 被标记为“可用于 Intra_8x8 预测”时，使用这种预测方式。

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 的值为：

$$pred8x8_L[x, y] = p'[x, -1], \ x, y = 0..7 \quad (8\text{-}87)$$

8.3.2.2.3 Intra_8x8_Horizontal预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 1 时，调用这种 Intra_8x8 预测模式。

仅在样点 p[−1, y]，y = 0..7 被标记为“可用于 Intra_8x8 预测”时，使用这种预测方式。

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 为：

$$pred8x8_L[x, y] = p'[-1, y], \ x, y = 0..7 \quad (8\text{-}88)$$

8.3.2.2.4 Intra_8x8_DC预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 2 时，调用这种 Intra_8x8 预测模式。

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 由以下方式得到：

— 如果所有p[x, −1]，x = 0..7 以及p[−1, y]，y = 0..7 的样点都被标记为“可用于是Intra_8x8预测”，则样点预测值$pred8x8_L[x, y]$，x, y = 0..7由下式得到：

$$pred8x8_L[x,y] = (\sum_{x'=0}^{7} p'[x',-1] + \sum_{y'=0}^{7} p'[-1,y'] + 8) \quad 4 \tag{8-89}$$

— 否则，如果任何p[x, −1]，x = 0..7 被标记为“不可用于Intra_8x8预测”并且所有p[−1, y]，y = 0..7 样点被标记为“可用于Intra_8x8预测”，则样点预测值$pred8x8_L[x, y]$，x, y = 0..7由下式得到：

$$pred8x8_L[x,y] = (\sum_{y'=0}^{7} p'[-1,y'] + 4) \quad 3 \tag{8-90}$$

— 否则，如果任何p[x, −1]，x = 0..7被标记为“不可用于Intra_8x8预测”，并且所有 p[−1, y]，y = 0..7 被标记为“可用于Intra_8x8预测”），则预测样点值$pred8x8_L[x, y]$，x, y = 0..7由以下方式得到：

$$pred8x8_L[x,y] = (\sum_{x'=0}^{7} p'[x',-1] + 4) \quad 3 \tag{8-91}$$

— 否则（p[x, −1]，x = 0..7 以及 p[−1, y ，y = 0..7中的某些样点被标记为“不可用于Intra_8x8预测”], 则样点预测值$pred8x8_L[x, y]$，x, y = 0..7由下式得到：

$$pred8x8_L[x, y] = (1 << (BitDepth_Y - 1)) \tag{8-92}$$

注 — 一个8x8亮度块经总使用这种方式进行预测。

8.3.2.2.5 Intra_8x8_Diagonal_Down_Left预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 3 时，调用这种 Intra_8x8 预测模式。

仅在样点 p[x, −1]，x = 0..15 被标记为“可用于 Intra_8x8 预测”时，使用这种预测方式。

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 由以下方式得到：

— 如果x 等于 7 并且y 等于 7，

$$pred8x8_L[x, y] = (p'[14, -1] + 3 * p'[15, -1] + 2) >> 2 \tag{8-93}$$

— 否则 (x 不等于 7 或者y不等于 7)，

$$pred8x8_L[x, y] = (p'[x + y, -1] + 2 * p'[x + y + 1, -1] + p'[x + y + 2, -1] + 2) >> 2 \tag{8-94}$$

8.3.2.2.6 Intra_8x8_Diagonal_Down_Right预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 4 时，调用这种 Intra_8x8 预测模式。

仅在样点 p[x, −1]，x = 0..7 以及 p[−1, y]，y = −1..7 被标记为“可用于 Intra_8x8 预测”时，使用这种预测方式。

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 由以下方式得到：

— 如果 x 大于 y，

$$pred8x8_L[x, y] = (p'[x - y - 2, -1] + 2 * p'[x - y - 1, -1] + p'[x - y, -1] + 2) >> 2 \tag{8-95}$$

— 否则如果 x 小于 y，

$pred8x8_L[x, y] = (p'[-1, y - x - 2] + 2 * p'[-1, y - x - 1] + p'[-1, y - x] + 2) >> 2$ (8-96)

— 否则 (x等于y),

$pred8x8_L[x, y] = (p'[0, -1] + 2 * p'[-1, -1] + p'[-1, 0] + 2) >> 2$ (8-97)

8.3.2.2.7 Intra_8x8_Vertical_Right预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 5 时，调用这种 Intra_8x8 预测模式。

仅在样点 p[x, −1]，x = 0..7 以及 p[−1, y]，y = −1..7 被标记为“可用于 Intra_8x8 预测”时，使用这种预测方式。

设变量 zVR 等于 2 * x − y

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 由以下方式得到：

— 如果zVR 等于0, 2, 4, 6, 8, 10, 12或14

$pred8x8_L[x, y] = (p'[x - (y >> 1) - 1, -1] + p'[x - (y >> 1), -1] + 1) >> 1$ (8-98)

— 否则，如果 zVR等于1, 3, 5, 7, 9, 11或 13

$pred8x8_L[x, y] = (p'[x - (y >> 1) - 2, -1] + 2 * p'[x - (y >> 1) - 1, -1] + p'[x - (y >> 1), -1] + 2) >> 2$ (8-99)

— 否则，如果zVR等于 −1

$pred8x8_L[x, y] = (p'[-1, 0] + 2 * p'[-1, -1] + p'[0, -1] + 2) >> 2$ (8-100)

— 否则（zVR等于 −2, −3, −4, −5, −6或−7）

$pred8x8_L[x, y] = (p'[-1, y - 2*x - 1] + 2 * p'[-1, y - 2*x - 2] + p'[-1, y - 2*x - 3] + 2) >> 2$ (8-101)

8.3.2.2.8 Intra_8x8_Horizontal_Down预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 6 时，调用这种 Intra_8x8 预测模式。

仅在样点 p[x, −1]，x = 0..7 以及 p[−1, y]，y = −1..7 被标记为“可用于 Intra_8x8 预测”时使用这种预测方式。

设变量 zHD 等于 2 * y − x

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 由以下方式得到：

— 如果zHD等于 0, 2, 4, 6, 8, 10, 12或14

$pred8x8_L[x, y] = (p'[-1, y - (x >> 1) - 1] + p'[-1, y - (x >> 1)] + 1) >> 1$ (8-102)

— 否则，如果zHD 等于 1, 3, 5, 7, 9, 11或13

$pred8x8_L[x, y] = (p'[-1, y - (x >> 1) - 2] + 2 * p'[-1, y - (x >> 1) - 1] + p'[-1, y - (x >> 1)] + 2) >> 2$ (8-103)

— 否则，如果zHD等于−1,

$pred8x8_L[x, y] = (p'[-1, 0] + 2 * p'[-1, -1] + p'[0, -1] + 2) >> 2$ (8-104)

— 否则（zHD 等于 −2, −3, −4, −5, −6, −7）

$pred8x8_L[x, y] = (p'[x - 2*y - 1, -1] + 2 * p'[x - 2*y - 2, -1] + p'[x - 2*y - 3, -1] + 2) >> 2$ (8-105)

8.3.2.2.9 Intra_8x8_Vertical_Left预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 7 时，使用这种 Intra_8x8 预测模式。

仅在样点 p[x, −1]，x = 0..15 被标记为“可用于 Intra_8x8 预测”时使用这种预测方式。

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 由以下方式得到：

— 如果 y 等于0, 2, 4 或 6

$$pred8x8_L[x, y] = (p'[x + (y >> 1), -1] + p'[x + (y >> 1) + 1, -1] + 1) >> 1 \quad (8\text{-}106)$$

— 否则（y 等于 1, 3, 5, 7）

$$pred8x8_L[x, y] = (p'[x + (y >> 1), -1] + 2 * p'[x + (y >> 1) + 1, -1] + p'[x + (y >> 1) + 2, -1] + 2) >>2 \quad (8\text{-}107)$$

8.3.2.2.10 Intra_8x8_Horizontal_Up预测模式的规范

在 Intra8x8PredMode[luma8x8BlkIdx]等于 8 时，调用这种 Intra_8x8 预测模式。

仅在样点 p[−1, y]，y = 0..7 被标记为“可用于 Intra_8x8 预测”时，使用这种预测方式。

设变量 zHU 等于 x + 2 * y。

样点预测值 $pred8x8_L[x, y]$，x, y = 0..7 由以下方式得到：

— 如果 zHU 等于0, 2, 4, 6, 8, 10或12

$$pred8x8_L[x, y] = (p'[-1, y + (x >> 1)] + p'[-1, y + (x >> 1) + 1] + 1) >> 1 \quad (8\text{-}108)$$

— 否则，如果 zHU等于1, 3, 5, 7, 9或11

$$pred8x8_L[x, y] = (p'[-1, y + (x >> 1)] + 2 * p'[-1, y + (x >> 1) + 1] + p'[-1, y + (x >> 1) + 2] + 2) >>2 \quad (8\text{-}109)$$

— 否则，如果 zHU等于 13，

$$pred8x8_L[x, y] = (p'[-1, 6] + 3 * p'[-1, 7] + 2) >> 2 \quad (8\text{-}110)$$

— 否则（zHU 大于 13）

$$pred8x8_L[x, y] = p'[-1, 7] \quad (8\text{-}111)$$

8.3.3 亮度样点的Intra_16x16预测过程

当宏块预测模式为 Intra_16x16 时调用本过程。本过程定义了当前宏块的帧内预测亮度样点的推导过程。

本过程的输出是当前宏块的帧内亮度样点预测值 $pred_L[x, y]$。

去块效应滤波过程之前的 33 个相邻重建亮度样点 p[x, y]，x = −1, y = −1..15，以及 x = 0..15, y = −1 由以下方式得到。

— 调用6.4.9节所定义的得到样点相邻位置的推导过程，该过程中以 (x, y)即(xN, yN) 作为输入 ，mbAddrN 和 (xW, yW) 作为输出。

— p[x, y] 其中x = −1，y = −1..15 和x = 0..15，y = −1中的每个样点均由以下方式得到：

— 如果任一下列条件为真， 则样点 p[x, y]被标记为“不可用于tra_16x16预测”

— mbAddrN 不可用，

— 宏块mbAddrN以帧内方式编码并且constrained_intra_pred_flag等于 1，

— 宏块 mbAddrN 的mb_type 等于 SI，并且constrained_intra_pred_flag 等于 1。

— 否则 p[x, y]被标记为“可用于Intra_16x16预测”，并且当前宏块mbAddrN 内(xW, yW) 位置上的亮度样点被赋值给p[x, y]。

设 $pred_L$[x, y]，x, y = 0..15 表示 16x16 亮度宏块中的样点。

表 8-4 规定了 Intra_16x16 预测模式。

表 8-4－Intra16x16PredMode以及相关名称的规范

Intra16x16PredMode	Intra16x16PredMode 的名称
0	Intra_16x16_Vertical (预测模式)
1	Intra_16x16_Horizontal (预测模式)
2	Intra_16x16_DC (预测模式)
3	Intra_16x16_Plane (预测模式)

根据 Intra16x16PredMode，可以调用 8.3.3.1 到 8.3.3.4 节中规定的任何一种 Intra_16x16 预测模式。

8.3.3.1 Intra_16x16_Vertical预测模式的规范

仅在样点 p[x, −1]，x = 0..15 被标记为“可用于 Intra_16x16 预测”时，使用这种 Intra_16x16 预测方式。

$$pred_L[x, y] = p[x, -1],\ x, y = 0..15 \tag{8-112}$$

8.3.3.2 Intra_16x16_Horizontal预测模式的规范

仅在样点 p[−1, y]，y = 0..15 被标记为“可用于 Intra_16x16 预测”时，使用这种 Intra_16x16 预测方式。

$$pred_L[x, y] = p[-1, y],\ x, y = 0..15 \tag{8-113}$$

8.3.3.3 Intra_16x16_DC预测模式的规范

根据相邻样点是否被标记为“可用于 Intra_16x16 预测”，Intra_16x16 预测模式定义如下：

— 如果所有的相邻样点p[x, −1]，x = 0..15 以及p[−1, y]，y = 0..15 被标记为“可用于Intra_16x16预测”，宏块内所有亮度样点预测值如下：

$$pred_L[x, y] = \left(\sum_{x'=0}^{15} p[x', -1] + \sum_{y'=0}^{15} p[-1, y'] + 16\right) \gg 5,\ x, y = 0..15 \tag{8-114}$$

— 否则，如果任何相邻样点p[x, −1]，x = 0..15 被标记为“不可用于Intra_16x16预测”并且所有相邻样点 p[−1, y]，y = 0..15 被标记为“可用于Intra_16x16预测时”，宏块内所有亮度样点预测值如下：

$$pred_L[x, y] = \left(\sum_{y'=0}^{15} p[-1, y'] + 8\right) \gg 4,\ x, y = 0..15 \tag{8-115}$$

— 否则，如果任何相邻样点p[−1, y]，y = 0..15 被标记为“不可用于Intra_16x16预测”并且所有相邻样点 p[x, −1]，x = 0..15 被标记为“可用于Intra_16x16预测”，宏块内所有亮度样点预测值如下：

$$pred_L[x, y] = \left(\sum_{x'=0}^{15} p[x', -1] + 8\right) \gg 4,\ x, y = 0..15 \tag{8-116}$$

— 否则（某些相邻样点 p[x, −1]，x = 0..15 和p[−1, y]，y = 0..15 被标记为“不可用于Intra_16x16预测”），当前宏块内所有亮度样点预测值如下：

$pred_L[x, y] = (1 << (BitDepth_Y − 1)),\ x, y = 0..15$ (8-117)

8.3.3.4 Intra_16x16_Plane预测模式的规范

仅在样点 p[x, −1]，x = −1..15 以及 p[−1, y]，y = 0..15 被标记为“可用于 Intra_16x16 预测”时，使用这种 Intra_16x16 预测方式。

$pred_L[x, y] = Clip1_Y((a + b * (x − 7) + c * (y − 7) + 16) >> 5)$，x，y = 0..15 （8-118）

其中：

$a = 16 * (p[−1, 15] + p[15, −1])$ (8-119)

$b = (5 * H + 32) >> 6$ (8-120)

$c = (5 * V + 32) >> 6$ (8-121)

H 和 V 由等式 8-122 和 8-123 定义

$$H = \sum_{x'=0}^{7} (x'+1) * (p[8+x', -1] - p[6-x', -1])$$ (8-122)

$$V = \sum_{y'=0}^{7} (y'+1) * (p[-1, 8+y'] - p[-1, 6-y'])$$ (8-123)

8.3.4 色度样点的帧内预测过程

对 I 和 SI 宏块类型调用本过程。该过程规定了当前宏块的帧内预测色度样点的推导过程。

本过程的输出是当前宏块的帧内预测色度样点 $pred_{Cb}[x, y]$和 $pred_{Cr}[x, y]$。

宏块的两个色度块（Cb 和 Cr）使用相同的预测方式。预测方式单独应用到两个色度块。为每个色度块调用本节所定义的过程。在本节以下的部分，色度块特指是两色度块之一，使用下标 C 代替 Cb 或 Cr。

去块效应滤波过程之前构建的相邻色度样点 p[x, y]（这里 x = −1, y = −1..MbHeightC − 1 以及 x = 0..MbWidthC − 1, y = −1）由以下方式得到。

— 调用6.4.9节所定义的相邻位置的推导过程可以用来推导色度位置，色度位置以 (x, y)即(xN, yN) 作为输入，mbAddrN 和 (xW, yW) 作为输出。

— 每个样点p[x, y]均由以下方式得到：

  — 如果以下任何条件成立，则样点p[x, y]标记为“不可用于帧内色度预测”。

    — mbAddrN 不可用，

    — 宏块mbAddrN 以帧内模式预测编码，并且 constrained_intra_pred_flag等于 1，

    — 宏块 mbAddrN 的mb_type 等于 SI 、constrained_intra_pred_flag 等于1，并且当前宏块的mb_type不等于SI。

  — 否则，p[x, y]被标记为“可用于帧内色度预测”，宏块mbAddrN内 色度位置(xW, yW)上分量C的色度样点分配给p[x, y]。

设 $pred_C$[x, y]，x = 0..MbWidthC − 1，y = 0..MbHeightC − 1 代表色度块样点预测值。

表 8-5 描述帧内色度预测模式。

表 8-5－帧内色度预测模式描述以及相应名称的规范

intra_chroma_pred_mode	intra_chroma_pred_mode 名称
0	Intra_Chroma_DC (预测模式)
1	Intra_Chroma_Horizontal (预测模式)
2	Intra_Chroma_Vertical (预测模式)
3	Intra_Chroma_Plane (预测模式)

根据 intra_chroma_pred_mode，调用 8.3.4.1 到 8.3.4.4 节中定义的某一种帧间色度预测模式。

8.3.4.1 Intra_Chroma_DC预测模式的规范

当 intra_chroma_pred_mode 等于 0 时，调用这种帧内色度预测方式。

对于每一个索引为 chroma4x4BlkIdx = 0..(1 << (chroma_format_idc + 1)) − 1 的 4x4 的色度块样点，应用如下规则：

— 根据chroma_format_idc的取值不同，索引为chroma4x4BlkIdx 的4x4色度块的左上角位置由以下方式得到：

— 如果chroma_format_idc等于1或2，则

xO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (8-124)

yO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (8-125)

— 否则（chroma_format_idc等于3），下列适用

xO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 0) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 0) (8-126)

yO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 1) (8-127)

— 如果(xO, yO) 等于(0, 0) 或者 xO 和 yO 大于 0，样点预测值 $pred_C$[x + xO, y + yO]，x, y = 0..3 由如下方式得到：

— 如果所有p[x + xO, −1]，x = 0..3 以及p[−1, y +yO]，y = 0..3 的样点标记为“可用于帧内色度预测”， 样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3 为

$$pred_C[x+xO, y+yO] = \left(\sum_{x'=0}^{3} p[x'+xO, -1] + \sum_{y'=0}^{3} p[-1, y'+yO] + 4 \right) \quad 3 \text{ , x, y = 0..3.} \tag{8-128}$$

— 否则，如果任何p[x + xO, −1]，x = 0..3 的样点标记为“不可用于帧内色度预测”并且所有p[−1, y +yO]，y = 0..3的样点被标记为“可用于帧内色度预测”，样点预测值 $pred_C$[x + xO, y + yO]，x, y = 0..3 为

$$pred_C[x+xO, y+yO] = \left(\sum_{y'=0}^{3} p[-1, y'+yO] + 2 \right) \quad 2 \text{ , x, y = 0..3.} \tag{8-129}$$

— 否则，如果任何p[−1, y +yO]，y = 0..3的样点被标记为“不可用于帧内色度预测”并且所有p[x + xO, −1]，x = 0..3的样点被标记为“可用于帧内色度预测”，样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3为：

$$pred_C[x+xO, y+yO] = \left(\sum_{x'=0}^{3} p[x'+xO, -1] + 2 \right) \quad 2,\ x, y = 0..3. \tag{8-130}$$

— 否则 (p[x + xO, −1]，x = 0..3与p[−1, y +yO]，y = 0..3中的某些样点被标记为“不可用于帧内色度预测”)，样点预测值 $pred_C$[x + xO, y + yO]，x, y = 0..3为

$$pred_C[x+xO, y+yO] = (1 << (BitDepth_C - 1)),\ x, y = 0..3. \tag{8-131}$$

— 否则，如果xO 大于0，且yO等于 0, 样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3 为：

— 如果所有p[x + xO, −1]，x = 0..3的样点被标记为“可用于帧内色度预测”，样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3 为

$$pred_C[x+xO, y+yO] = \left(\sum_{x'=0}^{3} p[x'+xO, -1] + 2 \right) \quad 2,\ x, y = 0..3. \tag{8-132}$$

— 否则，如果任何样点p[−1, y +yO]，y = 0..3 被标记为“可用于帧内色度预测”， 样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3 为

$$pred_C[x+xO, y+yO] = \left(\sum_{y'=0}^{3} p[-1, y'+yO] + 2 \right) \quad 2,\ x, y = 0..3. \tag{8-133}$$

— 否则，(p[x + xO, −1]，x = 0..3以及 p[−1, y +yO]，y = 0..3中的某些样点被标记为“不可用于帧内色度预测”)，样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3 为

$$pred_C[x+xO, y+yO] = (1 << (BitDepth_C - 1)),\ x, y = 0..3. \tag{8-134}$$

— 否则(xO 等于0 并且yO大于 0)，样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3为：

— 如果所有p[−1, y +yO]，y = 0..3 的样点标记为“可用于帧内色度预测”，样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3 为：

$$pred_C[x+xO, y+yO] = \left(\sum_{y'=0}^{3} p[-1, y'+yO] + 2 \right) \quad 2,\ x, y = 0..3. \tag{8-135}$$

— 否则，如果任何样点p[x + xO, −1]，x = 0..3 被标记为“可用于帧内色度预测”，样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3为

$$pred_C[x+xO, y+yO] = \left(\sum_{x'=0}^{3} p[x'+xO, -1] + 2 \right) \quad 2,\ x, y = 0..3. \tag{8-136}$$

— 否则 (p[x + xO, −1]，x = 0..3 以及 p[−1, y +yO]，y = 0..3中的某些样点被标记为“不可用于帧内色度预测”)，样点预测值$pred_C$[x + xO, y + yO]，x, y = 0..3为

$$pred_C[x+xO, y+yO] = (1 << (BitDepth_C - 1)),\ x, y = 0..3. \tag{8-137}$$

8.3.4.2 Intra_Chroma_Horizontal预测模式的规范

在 intra_chroma_pred_mode 等于 1 时，调用这种帧内色度预测方式。

仅当样点 p[−1, y]，y = 0..MbHeightC − 1 被标记为“可用于帧内色度预测”时，应用这种预测模式。

样点预测值 $pred_C$[x, y] 如下：

$pred_C[x, y] = p[-1, y]$，x = 0..MbWidthC − 1 和 y = 0..MbHeightC − 1 (8-138)

8.3.4.3 Intra_Chroma_Vertical预测模式的规范

在 intra_chroma_pred_mode 等于 2 时，调用这种帧内色度预测方式。

仅当样点 p[x, −1]，x = 0..MbWidthC − 1 被标记为“可用于帧内色度预测”时，应用这种预测模式。

样点预测值 $pred_C$[x, y] 如下：

$pred_C[x, y] = p[x, -1]$，x = 0..MbWidthC − 1 和 y = 0..MbHeightC − 1 (8-139)

8.3.4.4 Intra_Chroma_Vertical预测模式的规范

在 intra_chroma_pred_mode 等于 3 时，调用这种帧内色度预测方式。

仅当样点 p[x, −1]，x = 0..MbWidthC − 1 以及 p[−1, y] y = −1..MbHeightC − 1 被标记为“可用于帧内色度预测”。

样点预测值 $pred_C$[x, y]推导如下：

设变量 xCF 等于 4 * (chroma_format_idc == 3)和 yCF 等于 4 * (chroma_format_idc != 1)。

$pred_C[x, y] = Clip1_C((a + b * (x - 3 - xCF) + c * (y - 3 - yCF) + 16) >> 5)$,

x = 0..MbWidthC − 1 和 y = 0..MbHeightC − 1 (8-140)

其中：

a = 16 * (p[−1, MbHeightC − 1] + p[MbWidthC − 1, −1]) (8-141)

b = ((34 − 29 * (chroma_format_idc == 3)) * H + 32) >> 6 (8-142)

c = ((34 − 29 * (chroma_format_idc != 1)) * V + 32) >> 6 (8-143)

H 和 V 的定义如下：

$$H = \sum_{x'=0}^{3+xCF} (x'+1) * (p[4 + xCF + x', -1] - p[2 + xCF - x', -1]) \quad (8\text{-}144)$$

$$V = \sum_{y'=0}^{3+yCF} (y'+1) * (p[-1, 4 + yCF + y'] - p[-1, 2 + yCF - y']) \quad (8\text{-}145)$$

8.3.5 I_PCM宏块的样点构建过程

在 mb_type 等于 I_PCM 时，调用本过程。

变量 dy 由以下方式得到。

— 如果变量MbaffFrameFlag等于1，且当前宏块是场宏块，则dy等于2。

— 否则（变量MbaffFrameFlag等于0或者当前宏块是帧宏块），则dy等于1。

当前宏块的左上角亮度样点的位置是由 6.4.1 所定义的逆宏块扫描过程得到的，这个过程的输入是 CurrMbAddr，输出赋给(xP, yP)。

去块效应滤波之前的构建亮度样点由以下方式得到：

```
for( i = 0; i < 256; i++ )
  S'L[ xP + ( i % 16 ), yP + dy * ( i / 16 ) ] ] = pcm_sample_luma[ i ]                    (8-146)
```

当 chroma_format_idc 不等于 0 (单色)时，去块效应之前的重建色度样点由下式得到：

```
for( i = 0; i < MbWidthC * MbHeightC; i++ ) {
  S'Cb[ ( xP / SubWidthC ) + ( i % MbWidthC ),
        ( ( yP + SubHeightC − 1 ) / SubHeightC ) + dy * ( i / MbWidthC ) ] =
         pcm_sample_chroma[ i ]                                                          (8-147)
  S'Cr[ ( xP / SubWidthC ) + ( i % MbWidthC ),
        ( ( yP + SubHeightC − 1 ) / SubHeightC ) + dy * ( i / MbWidthC ) ] =
           pcm_sample_chroma[ i + MbWidthC * MbHeightC ]
}
```

8.4 帧间预测过程

当解码 P、B 类型的宏块时，调用本过程。

本过程的输出为当前宏块的帧间预测样点，包含一个 16×16 亮度阵列 $pred_L$ 和 2 个（当 chroma_format_idc 不等于 0，即单色时）8×8 色度样点阵列 $pred_{Cr}$ 和 $pred_{Cb}$，色度分量分别为 Cb 和 Cr。

宏块的分割模式由该宏块的 mb_type 规定。宏块的每个分割块均有一个索引值 mbPartIdx 作为标识。如果宏块分割是由与子宏块相等的分割块构成，则每个子宏块可以进一步分割为子宏块分割块，子宏块的分割模式由 sub_mb_type 规定的。每个子宏块分割均有一个索引值 subMbPartIdx 作为标识。如果宏块分割块不是由子宏块构成，则 subMbPartIdx 等于 0。

对于每个宏块分割块或每个子宏块分割块，有如下规定。

函数 MbPartWidth( ), MbPartHeight( ), SubMbPartWidth( )和 SubMbPartHeight( )给出了宏块分割块和子宏块分割块的宽度和高度，它们的定义分别见表 7-13, 表 7-14, 表 7-17 和表 7-18。

宏块分割块的索引 mbPartIdx 的取值范围规定如下：

— 如果mb_type等于B_Skip或B_Direct_16x16，则mbPartIdx在0…3之间取值 。

— 否则（mb_type不等于B_Skip或B_Direct_16x16），mbPartIdx在0…NumMbPart(mb_type) − 1之间取值。

每个 mbPartIdx 值所对应的宏块中的每个宏块分割块或其子宏块分割块的相关变量 partWidth 和 partHeight 推导过程如下：

— 如果mb_type不等于P_8x8, P_8x8ref0, B_Skip, B_Direct_16x16或B_8x8，则subMbPartIdx等于0，并且partWidth和partHeight按下式得到：

```
partWidth = MbPartWidth( mb_type )                                                       (8-148)
partHeight = MbPartHeight( mb_type )                                                     (8-149)
```

— 否则，如果mb_type等于P_8x8或P_8x8ref0，或者mb_type等于B_8x8且sub_mb_type[mbPartIdx]不等于B_Direct_8x8，subMbPartIdx在0…NumSubMbPart(sub_mb_type) − 1之间取值，并且partWidth和partHeight按照下式得到：

```
partWidth = SubMbPartWidth( sub_mb_type[ mbPartIdx ] )                                   (8-150)
partHeight = SubMbPartHeight( sub_mb_type[ mbPartIdx ] ).                                (8-151)
```

— 否则（mb_type等于B_Skip 或B_Direct_16x16，或者mb_type等于B_8x8且sub_mb_type[mbPartIdx]等于B_Direct_8x8），subMbPartIdx在0…3之间取值，并且partWidth和partHeight按照下式得到：

partWidth = 4 (8-152)

partHeight = 4 (8-153)

当 chroma_format_idc 不等于 0（单色）时，变量 partWidthC 和 partHeightC 的值按照下式得到：

partWidthC = partWidth / SubWidthC (8-154)

partHeightC = partHeight / SubHeightC (8-155)

在 8.4.1 节所定义的过程被调用之前，令变量 MvCnt 的初始值等于 0。

宏块分割 mbPartIdx 和子宏块分割 subMbPartIdx 的帧间预测过程依次包含下列步骤：

1．8.4.1 节所定义的运动矢量分量和参考索引的推导过程。

该过程的输入为：

— 宏块分割mbPartIdx，

— 子宏块分割subMbPartIdx。

该过程的输出为：

— 亮度运动矢量mvL0和mvL1，以及chroma_format_idc不为0（单色）时的色度运动矢量mvCL0、mvCL1。

— 参考索引refIdxL0和refIdxL1

— 指示预测列表使用的标志位predFlagL0和predFlagL1

— 子宏块分割块的运动矢量个数subMvCnt

2．变量 MvCnt 以 subMvCnt 为步长递增。

3．8.4.2 节定义的帧间预测样点的解码过程。

输入为：

— 宏块分割mbPartIdx

— 子宏块分割subMbPartIdx

— 规定亮度和色度（如果可用）分割宽度和高度的变量partWidth, partHeight, partWidthC （如果可用）和 partHeightC（如果可用）

— 亮度运动矢量mvL0和mvL1，以及chroma_format_idc不为0（单色）时的色度运动矢量mvCL0、mvCL1

— 参考索引refIdxL0和refIdxL1

— 指示预测列表使用的标志位的predFlagL0和predFlagL1

输出为：

— 帧间预测样点（pred）：它是预测得到的一个亮度样点的(partWidth) × (partHeight)的阵列 $predPart_L$，以及当chroma_format_idc不为0（单色）时预测得到的2个色度样点(partWidthC) × (partHeightC)色度阵列$predPart_{Cr}$、$predPart_{Cb}$，每个色度分量Cb和Cr都对应一个。

为便于在后期解码时的变量推导过程中使用，部分变量赋值如下：

MvL0[mbPartIdx][subMbPartIdx] = mvL0 (8-156)

MvL1[mbPartIdx][subMbPartIdx] = mvL1 (8-157)

RefIdxL0[mbPartIdx] = refIdxL0 (8-158)

RefIdxL1[mbPartIdx] = refIdxL1 (8-159)

PredFlagL0[mbPartIdx] = predFlagL0 (8-160)

PredFlagL1[mbPartIdx] = predFlagL1 (8-161)

宏块分割块中左上角样点与所处宏块左上角样点之间的相对位置，可以通过调用 6.4.2.1 中定义的反向宏块分割扫描过程得到，mbPartIdx 作为输入，输出为(xP, yP)。

宏块子分割块中左上角样点与所处宏块分割块左上角样点之间的相对位置，可以通过调用 6.4.2.2 中定义的反向子宏块分割块扫描过程得到，subMbPartIdx 作为输入，输出为(xS, yS)。

宏块的预测可以通过将分割块或子宏块分割块的预测样点放置在宏块中的适当位置上来构造，如下：

变量 $pred_L$[xP + xS + x, yP + yS + y]按下式得到，其中 x = 0 .. partWidth − 1, y = 0 .. partHeight − 1

$pred_L$[xP + xS + x, yP + yS + y] = $predPart_L$[x, y] (8-162)

当 chroma_format_idc 不等于 0（单色）时，变量 $pred_C$（其中，x = 0..partWidthC − 1, y = 0..partHeightC − 1，$pred_C$ 和 $predPart_C$ 中的 C 分别换成 Cb 或 Cr）按下式得到：

$pred_C$[xP / SubWidthC + xS / SubWidthC + x, yP / SubHeightC + yS / SubHeightC + y] = $predPart_C$[x, y] (8-163)

8.4.1 运动矢量分量和参考索引的推导过程

本过程的输入为：

— 宏块分割块mbPartIdx，

— 子宏块分割块subMbPartIdx。

本过程的输出为：

— 亮度运动矢量mvL0和mvL1，以及色度运动矢量mvCL0和mvCL1

— 参考索引refIdxL0和refIdxL1

— 指示预测列表使用的标志位predFlagL0和predFlagL1

— 表示宏块子分割块运动矢量个数的变量subMvCnt

变量 mvL0 和 mvL1 以及 refIdxL0 和 refIdxL1 的推导，规定如下：

— 如果mb_type等于P_Skip，则调用8.4.1.1节定义的在P或SP条带中跳过宏块的亮度运动矢量的导出过程，该过程的输出为亮度运动矢量mvL0和参考索引refIdxL0，且predFlagL0设置为1。mvL1和refIdxL1被标记为不可用，且predFlagL1设置为0。子分割块运动矢量个数变量subMvCnt设置为1。

— 否则，如果mb_type等于B_Skip或B_Direct_16x16，或者sub_mb_type[mbPartIdx]等于B_Direct_8x8，调用8.4.1.2节中定义的B条带中B_Skip, B_Direct_16x16和B_Direct_8x8宏块的亮度运动矢量导出过程，mbPartIdx和subMbPartIdx 作为输入，输出为亮度运动矢量mvL0, mvL1、参考索引refIdxL0, refIdxL1、子分割块的运动矢量个数subMvCnt以及指示预测列表使用的标志位predFlagL0和predFlagL1。

— 否则，对于变量predFlagLX、mvLX、refIdxLX和Pred_LX，以及语法元素ref_idx_lX和mvd_lX（其中的X可以用0或1替换），应用下列规则：

1．变量 refIdxLX 和 predFlagLX 推导如下：

— 如果 MbPartPredMode(mb_type, mbPartIdx) 或 SubMbPredMode(sub_mb_type[mbPartIdx]) 等于 Pred_LX 或等于BiPred，

refIdxLX = ref_idx_lX[mbPartIdx] (8-164)

predFlagLX = 1 (8-165)

— 否则，变量refIdxLX和 predFlagLX 取值如下：

refIdxLX = –1 (8-166)

predFlagLX = 0 (8-167)

2．表示子分割块运动矢量个数的变量 subMvCnt 等于 predFlagL0 + predFlagL1。

3．变量 currSubMbType 推导如下：

— 如果宏块类型为B_8x8，则currSubMbType的值等于sub_mb_type[mbPartIdx]。

— 否则（宏块类型不是B_8x8），currSubMbType的值设定为“未指定（na）”。

4．当 predFlagLX 等于 1 时，调用 8.4.1.3 节中定义的亮度运动矢量预测值的推导过程，mbPartIdx、subMbPartIdx、refIdxLX 和 currSubMbType 作为输入，输出为 mvpLX。亮度运动矢量按下式得到：

mvLX[0] = mvpLX[0] + mvd_lX[mbPartIdx][subMbPartIdx][0] (8-168)

mvLX[1] = mvpLX[1] + mvd_lX[mbPartIdx][subMbPartIdx][1] (8-169)

对色度运动矢量的导出过程，应用下述规定，即：当 predFlagLX（X 为 0 或 1）等于 1 时，调用 8.4.1.4 节定义的色度运动矢量导出过程，mvLX 和 refIdxLX 作为输入，输出为 mvCLX。

8.4.1.1 P和SP条带中跳过宏块的亮度运动矢量推导过程

当 mb_type 等于 P_Skip 时，调用本过程。

本过程的输出为运动矢量 mvL0 和参考索引 refIdxL0。

跳过模式下宏块的参考索引 refIdxL0 按下式取值：

refIdxL0 = 0. (8-170)

对于 P_Skip 类型宏块的运动矢量 mvL0 的推导，应用下列规则：

— 调用8.4.1.3.2节定义的过程，其中mbPartIdx设置为0，subMbPartIdx设置为0，currSubMbType设置为“未指定（na）”，listSuffixFlag设置为0，作为输入。输出结果赋值给mbAddrA, mbAddrB, mvL0A, mvL0B, refIdxL0A和refIdxL0B。

— 对变量mvL0有如下规定：

— 如果下述条件中的任何一个为真，则运动矢量mvL0的两个分量均等于0：

— mbAddrA不可用

— mbAddrB不可用

— refIdxL0A等于0，且mvL0A的两个分量都等于0

— refIdxL0B等于0，且mvL0B的两个分量都等于0

— 否则，调用8.4.1.3节定义的亮度运动矢量预测值的导出过程，其中mbPartIdx = 0, subMbPartIdx = 0, refIdxL0和currSubMbType = "na" 作为输入， 输出结果赋值给mvL0。

注 — 输出直接赋值给mvL0，因为预测值等于运动矢量的实际值。

8.4.1.2 B_Skip, B_Direct_16x16和B_Direct_8x8模式下亮度运动矢量的推导过程

当 mb_type 等于 B_Skip or B_Direct_16x16，或者 sub_mb_type[mbPartIdx]等于 B_Direct_8x8 时，调用本过程。

本过程的输入为 mbPartIdx 和 subMbPartIdx。

本过程的输出为参考索引 refIdxL0、refIdxL1，运动矢量 mvL0 和 mvL1，子分割块的运动矢量个数 subMvCnt，以及指示预测列表使用的标志位 predFlagL0 和 predFlagL1。

推导过程由条带头部语法里比特流中出现的 direct_spatial_mv_pred_flag（见 7.3.3）确定，规定如下：

— 如果direct_spatial_mv_pred_flag等于1，则本过程输出结果的推导过程参照空域直接预测模式。

— 否则（direct_spatial_mv_pred_flag等于0），本过程输出结果的推导过程参照时域直接预测模式。

空域和时域直接预测模式都使用了 8.4.1.2.1 节中定义的共同位置（co-located）运动矢量和参考索引。

运动矢量和参考索引推导如下：

— 如果采用空域直接预测模式，则采用8.4.1.2.2节规定的直接运动矢量和参考索引预测模式，输出为subMvCnt。

— 否则（采用时域直接预测模式），采用8.4.1.2.3节规定的直接运动矢量和参考索引预测模式，且变量subMvCnt推导如下：

— 如果subMbPartIdx等于0，则subMvCnt 等于2。

— 否则（subMbPartIdx不等于0），则Otherwise subMvCnt 等于0。

8.4.1.2.1 共同位置4z4子宏块分割块的推导过程

本过程的输入为 mbPartIdx 和 subMbPartIdx。

本过程的输出为图像 colPic，共同位置宏块 mbAddrCol，运动矢量 mvCol，参考索引 refIdxCol 以及变量 vertMvScale （可能的取值为 One_To_One, Frm_To_Fld 或 Fld_To_Frm）。

当 RefPicList1[0]为一个帧或者一个互补的场对时，令 firstRefPicL1Top 和 firstRefPicL1Bottom 分别为 RefPicList1[0]中的顶场和底场，且令下列变量取值如下：

topAbsDiffPOC = Abs(DiffPicOrderCnt(firstRefPicL1Top, CurrPic)) (8-171)

bottomAbsDiffPOC = Abs(DiffPicOrderCnt(firstRefPicL1Bottom, CurrPic)) (8-172)

变量 colPic 指出了包含共同位置宏块的图像，如表 8-6。

表 8-6－变量colPic的规范

field_pic_flag	RefPicList1[0] is …	mb_field_decoding_flag	附加条件	colPic
1	已解码帧中的一场			包含 RefPicList1[0]的帧
	一个已解码的场			RefPicList1[0]
0	一个已解码的帧			RefPicList1[0]
	一个互补场对	0	topAbsDiffPOC < bottomAbsDiffPOC	firstRefPicL1Top
			topAbsDiffPOC >= bottomAbsDiffPOC	firstRefPicL1Bottom
		1	(CurrMbAddr & 1) == 0	firstRefPicL1Top
			(CurrMbAddr & 1) != 0	firstRefPicL1Bottom

当 direct_8x8_inference_flag 等于 1 时，subMbPartIdx 按下式取值：

subMbPartIdx = mbPartIdx (8-173)

令 PicCodingStruct(X)为含参数 X 的函数，X 可以取 CurrPic 或 colPic。该函数定义如表 8-7。

表 8-7－函数PicCodingStruct(X)的规范

X 编码时使用的 with field_pic_flag 值	mb_adaptive_frame_field_flag	PicCodingStruct(X)
1		FLD
0	0	FRM
0	1	AFRM

以 luma4x4BlkIdx = mbPartIdx * 4 + subMbPartIdx 作为输入，调用 6.4.3 定义的反向 4×4 亮度块扫描过程，将输出的结果(x, y)赋值给(xCol, yCol)。

表 8-8 按照下列 2 个步骤规定了共同位置宏块地址 mbAddrCol，yM 和变量 vertMvScale 的取值：

1．根据 PicCodingStruct(CurrPic)和 PicCodingStruct(colPic)确定的宏块地址 mbAddrX 定义。

注 — CurrPic和colPic的图像编码类型不可能为(FRM, AFRM)，也不可能为(AFRM, FRM)，因为这些图像编码类型必须用IDR图像分隔开。

2．根据 mb_field_decoding_flag 和变量 fieldDecodingFlagX 确定的 mbAddrCol, yM 和 vertMvScale 的定义，fieldDecodingFlagX 推导如下：

— 如果图像colPic中的宏块mbAddrX是一个场宏块，则fieldDecodingFlagX的取值为1

— 否则（图像colPic中的宏块mbAddrX是一个帧宏块），fieldDecodingFlagX的取值为0。

表 8-8 中未做规定的值，表示相应变量值与其在当前表格中所处的行无关。

mbAddrCol 取值为 CurrMbAddr，或者取下列值之一：

mbAddrCol1 = 2 * PicWidthInMbs * (CurrMbAddr / PicWidthInMbs) +
(CurrMbAddr % PicWidthInMbs) + PicWidthInMbs * (yCol / 8) (8-174)

mbAddrCol2 = 2 * CurrMbAddr + (yCol / 8) (8-175)

mbAddrCol3 = 2 * CurrMbAddr + bottom_field_flag (8-176)

mbAddrCol4 = PicWidthInMbs * (CurrMbAddr / (2 * PicWidthInMbs)) +
(CurrMbAddr % PicWidthInMbs) (8-177)

mbAddrCol5 = CurrMbAddr / 2 (8-178)

mbAddrCol6 = 2 * (CurrMbAddr / 2) + ((topAbsDiffPOC < bottomAbsDiffPOC) ? 0 : 1) (8-179)

mbAddrCol7 = 2 * (CurrMbAddr / 2) + (yCol / 8) (8-180)

表 8-8－mbAddrCol, yM和vertMvScale取值的规范

PicCodingStruct(CurrPic)	PicCodingStruct(colPic)	mbAddrX	mb_field_decoding_flag	FieldDecodingFlagX	mbAddrCol	yM	vertMvScale
FLD	FLD				CurrMbAddr	yCol	One_To_One
	FRM				mbAddrCol1	(2 * yCol) % 16	Frm_To_Fld
	AFRM	2*CurrMbAddr		0	mbAddrCol2	(2 * yCol) % 16	Frm_To_Fld
				1	mbAddrCol3	yCol	One_To_One
FRM	FLD				mbAddrCol4	8 * ((CurrMbAddr / PicWidthInMbs) % 2) + 4 * (yCol / 8)	Fld_To_Frm
	FRM				CurrMbAddr	yCol	One_To_One
AFRM	FLD		0		mbAddrCol5	8 * (CurrMbAddr % 2) +4 * (yCol / 8)	Fld_To_Frm
			1		mbAddrCol5	yCol	One_To_One
	AFRM	CurrMbAddr	0	0	CurrMbAddr	yCol	One_To_One
				1	mbAddrCol6	8 * (CurrMbAddr % 2) + 4 * (yCol / 8)	Fld_To_Frm
		CurrMbAddr	1	0	mbAddrCol7	(2 * yCol) % 16	Frm_To_Fld
				1	CurrMbAddr	yCol	One_To_One

令 mbPartIdxCol 为共同位置宏块分割块的索引，subMbPartIdxCol 为共同位置子宏块分割块的索引。那么图像 colPic 中宏块 mbAddrCol 内覆盖样点（ xCol, yM）的分割块应该被指定为 mbPartIdxCol，且图像 colPic 中宏块分割块 mbPartIdxCol 内覆盖样点（ xCol, yM）的子宏块分割块应该被指定为 subMbPartIdxCol。

指示预测使用的标志位 predFlagL0Col 和 predFlagL1Col 分别设置为 PredFlagL0[mbPartIdxCol] 和 PredFlagL1[mbPartIdxCol]。这些指示预测使用的标志位是为图像 colPic 中的宏块分割块 mbAddrCol\mbPartIdxCol 分配的。

运动矢量 mvCol 和参考索引 refIdxCol 推导如下：

— 如果宏块mbAddrCol是在帧内宏块预测模式下编码的，或者指示预测列表使用的标志位predFlagL0Col和predFlagL1Col都等于0，则运动矢量mvCol的两个分量等于0，refIdxCol等于－1。

— 否则，应用如下规则：

  — 如果predFlagL0Col等于1，则运动矢量mvCol和参考索引refIdxCol分别等于MvL0[mbPartIdxCol][subMbPartIdxCol] 和 RefIdxL0[mbPartIdxCol]，这些值是为图像colPic内部（子）宏块分割块mbAddrCol\mbPartIdxCol\subMbPartIdxCol的运动矢量mvCol和参考索引refIdxL1分配的。

  — 否则（predFlagL0Col等于0，且predFlagL1Col等于1），运动矢量mvCol和参考索引refIdxCol分别等于MvL1[mbPartIdxCol][subMbPartIdxCol]和RefIdxL1[mbPartIdxCol]，这些值是为图像colPic内部（子）宏块分割块mbAddrCol\mbPartIdxCol\subMbPartIdxCol的运动矢量mvCol和参考索引refIdxCol分配的。

8.4.1.2.2 空域直接模式下亮度运动矢量和参考索引的推导过程

当 direct_spatial_mv_pred_flag 等于 1，且下列条件之一为真时，调用本过程：

— mb_type等于B_Skip

— mb_type等于B_Direct_16x16

— sub_mb_type[mbPartIdx]等于B_Direct_8x8

本过程的输入为 mbPartIdx 和 subMbPartIdx。

本过程的输出为参考索引 refIdxL0, refIdxL1，运动矢量 mvL0 和 mvL1，子分割块运动矢量的个数 subMvCnt，以及指示预测列表使用的标志位 predFlagL0 和 predFlagL1。

参考索引 refIdxL0 和 refIdxL1 以及变量 directZeroPredictionFlag 按照下列步骤导出：

1．令变量 currSubMbType 等于 sub_mb_type[mbPartIdx]。

2．调用 8.4.1.3.2 节定义的过程，mbPartIdx = 0, subMbPartIdx = 0, currSubMbType 和 listSuffixFlag = 0 作为输入，输出赋值给运动矢量 mvL0N 和参考索引 refIdxL0N，其中的 N 用 A、B 和 C 替代。

3．调用 8.4.1.3.2 节定义的过程，mbPartIdx = 0, subMbPartIdx = 0, currSubMbType 和 listSuffixFlag = 1 作为输入，输出被赋值给运动矢量 mvL1N 和参考索引 refIdxL1N，其中的 N 用 A、B 和 C 替代。

注 1 — 运动矢量mvL0N, mvL1N和参考索引refIdxL0N, refIdxL1N的值对同一个宏块中的所有4×4子宏块分割块都相同。

4．参考索引 refIdxL0, refIdxL1 和 directZeroPredictionFlag 推导如下：

refIdxL0 = MinPositive(refIdxL0A, MinPositive(refIdxL0B, refIdxL0C)) (8-181)

refIdxL1 = MinPositive(refIdxL1A, MinPositive(refIdxL1B, refIdxL1C)) (8-182)

directZeroPredictionFlag = 0 (8-183)

其中

$$\mathrm{MinPositive}(x,y)=\begin{cases}\mathrm{Min}(x,y) & \text{如果}x>=0\text{和}y>=0\\ \mathrm{Max}(x,y) & \text{其他情况}\end{cases} \quad (8\text{-}184)$$

5．当参考索引 refIdxL0 和 refIdxL1 均小于 0 时，

refIdxL0 = 0 (8-185)

refIdxL1 = 0 (8-186)

directZeroPredictionFlag = 1 (8-187)

以给定的 mbPartIdx, subMbPartIdx 作为输入，调用 8.4.1.2.1 节定义的过程，输出结果赋值给 refIdxCol 和 mvCol。

变量 colZeroFlag 的推导过程如下：

— 如果下列所有条件均为真，则 colZeroFlag 等于 1：

— RefPicList1[0]当前标志为"用于短期参考"。

— refIdxCol等于0。

— 运动矢量的两个分量mvCol[0]和mvCol[1]均在–1到1范围内，度量的单位规定如下：

— 如果共同位置宏块为帧宏块，则mvCol[0]和mvCol[1]的单位等于1/4亮度帧样点的单位；

— 否则（共同位置宏块为场宏块），mvCol[0]和mvCol[1]的单位等于1/4亮度场样点的单位。

注 2 — 为判定上述情况，在当前宏块为帧宏块而共同位置宏块为场宏块，或者当前宏块为场宏块而共同位置宏块为帧宏块时，mvCol[1]的值没有经过缩放以使用当前宏块运动矢量的单位。这一点与8.4.1.2.3中所规定的时域直接模式中mvCol[1]的使用不同，在那里通过等式8-190 或等式8-191对共同位置的运动矢量进行缩放，以便使用与当前宏块相同的运动矢量单位。

— 否则，colZeroFlag 等于 0。

运动矢量 mvLX （X 等于 0 或 1）推导过程如下：

— 如果下列任何一个条件为真，则运动矢量 mvLX 的两个分量均等于 0：

— directZeroPredictionFlag等于1

— refIdxLX 小于0

— refIdxLX等于0且colZeroFlag等于1

— 否则，调用8.4.1.3节定义的过程，mbPartIdx = 0, subMbPartIdx = 0, refIdxLX和currSubMbType作为输入，输出赋值给mvLX。

注 3 — 调用8.4.1.3节过程时返回的运动矢量mvLX对于同一宏块中的所有4×4子宏块分割均相等。

表示预测使用情况的标志位 predFlagL0 和 predFlagL1 应该按照表 8-9 的规定取值。

表 8-9－表示预测使用情况的标志位取值

refIdxL0	refIdxL1	predFlagL0	predFlagL1
>= 0	>= 0	1	1
>= 0	< 0	1	0
< 0	>= 0	0	1

变量 subMvCnt 推导如下：

— 如果subMbPartIdx不等于0或者direct_8x8_inference_flag等于0，则subMvCnt取值等于0。

— 否则（subMbPartIdx 等于 0 且 direct_8x8_inference_flag 等于 1），subMvCnt 等于 predFlagL0 + predFLagL1。

8.4.1.2.3 时域直接模式下亮度运动矢量和参考索引的推导过程

当 direct_spatial_mv_pred_flag 等于 0 且下列条件中的任何一个为真时，调用本过程：

— mb_type等于B_Skip

— mb_type等于B_Direct_16x16

— sub_mb_type[mbPartIdx]等于B_Direct_8x8。

本过程的输入为 mbPartIdx 和 subMbPartIdx。

本过程的输出为运动是矢量 mvL0 和 mvL1，参考索引 refIdxL0 和 refIdxL1，以及预测列表使用标志位 predFlagL0 和 predFlagL1。

以给定的 mbPartIdx, subMbPartIdx 作为输入，调用 8.4.1.2.1 定义的过程，输出分别作为 colPic, mbAddrCol, mvCol, refIdxCol 和 vertMvScale 的值。

参考索引 refIdxL0 和 refIdxL1 推导如下：

refIdxL0 = ((refIdxCol < 0) ? 0 : MapColToList0(refIdxCol)) (8-188)

refIdxL1 = 0 (8-189)

注 1 — 如果当前宏块为场宏块，则refIdxL0和refIdxL1指的是一系列的场；否则（当前宏块为帧宏块），refIdxL0和refIdxL1指的是一系列的帧或者互补参考场对。

令 refPicCol 为在解码图像 colPic 中的共同位置宏块 mbAddrCol 时，参考索引 refIdxCol 所指的一个帧、场或者互补场对。函数 MapColToList0(refIdxCol)定义如下：

— 如果vertMvScale等于One_To_One，应用如下规则：

  — 如果field_pic_flag等于0，且当前宏块为场宏块，则

    — 令refIdxL0Frm为当前参考图像列表RefPicList0中的最小索引值，该参考图像列表所指的为包含场refPicCol的帧或互补场对。RefPicList0应该包括含有场refPicCol的帧或互补场对。MapColToList0( )的返回值定义如下：

      — 如果refIdxCol所指的场与当前宏块具有相同的奇偶性，则MapColToList0(refIdxCol)返回参考索引(refIdxL0Frm << 1)。

      — 否则（refIdxCol所指的场具有与当前宏块相反的奇偶性），则MapColToList0(refIdxCol)返回参考索引((refIdxL0Frm << 1)+1)。

  — 否则（field_pic_flag 等于1，或者当前宏块为帧宏块），MapColToList0(refIdxCol)返回指向refPicCol的当前参考图像列表RefPicList0中的最小参考索引值refIdxL0，RefPicList0应该包含refPicCol。

— 否则，如果vertMvScale等于Frm_To_Fld，则

  — 如果field_pic_flag等于0，令refIdxL0Frm为当前参考图像列表RefPicList0中值最小的索引，该索引指向refPicCol。 MapColToList0(refIdxCol)返回参考索引(refIdxL0Frm << 1)。RefPicList0应该包含refPicCol。

  — 否则（field_pic_flag等于1），MapColToList0(refIdxCol)返回当前参考图像列表RefPicList0中值最小的索引refIdxL0， 该索引指向refPicCol中与当前图像CurrPic奇偶性相同的一个场。RefPicList0应该包含refPicCol中与当前图像CurrPic奇偶性相同的场。

— 否则（vertMvScale等于Fld_To_Frm），MapColToList0(refIdxCol)返回当前参考图像列表RefPicList0中值最小的索引refIdxL0，该索引指向包含refPicCol的帧或互补场对。RefPicList0应该包含含有场refPicCol的帧或互补场对。

注 2 — 一个已解码图像在含有共同位置宏块的图像解码过程中用作参考，被标记为“用于短期参考”，则当它在被用于当前宏块的直接模式下帧间预测参考之前，其标记可能已改为“用于长期参考”。

根据 vertMvScale 的值，对 mvCol 的垂直分量修改如下：

— 如果vertMvScale等于Frm_To_Fld，则

mvCol[1] = mvCol[1] / 2 (8-190)

— 否则，如果vertMvScale等于Fld_To_Frm，则

mvCol[1] = mvCol[1] * 2 (8-191)

— 否则（vertMvScale等于 One_To_One）， mvCol[1]保持不变。

变量 currPicOrField、pic0 和 pic1 推导如下：

— 如果field_pic_flag等于0，且当前宏块为场宏块，则

  — currPicOrField为当前图像CurrPic中与当前宏块具有相同奇偶性的场。

  — pic1为RefPicList1[0]中与当前宏块具有相同奇偶性的场。

  — 变量pic0推导如下：

    — 如果refIdxL0 % 2等于0，则pic0为RefPicList0[refIdxL0 / 2]中与当前宏块具有相同奇偶性的场。

    — 否则（refIdxL0 % 2不等于0），pic0为RefPicList0[refIdxL0 / 2]中与当前宏块有相反奇偶性的场。

— 否则（field_pic_flag等于1，或者当前宏块为帧宏块），currPicOrField为当前图像CurrPic，pic1为已解码参考图像RefPicList1[0]，pic0为已解码参考图像RefPicList0[refIdxL0]。

当前宏块中每个 4×4 子宏块分割块的两个运动矢量 mvL0 和 mvL1 推导过程如下：

注 3 — 比较常见的情形是许多4×4子宏块分割块共用相同的运动矢量和参考图像。这时，时域直接模式运动补偿能够按照比4×4更大的单元计算帧间预测样点的值。例如，当direct_8x8_inference_flag等于1时，至少宏块中每个8×8亮度样点所在的象限共用相同的运动矢量和参考图像。

— 如果参考索引refIdxL0指向一个长期参考图像，或DiffPicOrderCnt(pic1, pic0)等于0，则用于直接模式分割块的运动矢量mvL0和mvL1推导如下：

mvL0 = mvCol (8-192)

mvL1 = 0 (8-193)

— 否则，运动矢量mvL0, mvL1作为共同位置子宏块分割块的运动矢量mvCol的缩放值，按照下面规定推导（参见图8-2）：

tx = (16 384 + Abs(td / 2)) / td (8-194)

DistScaleFactor = Clip3(−1024, 1023, (tb * tx + 32) >> 6) (8-195)

mvL0 = (DistScaleFactor * mvCol + 128) >> 8 (8-196)

mvL1 = mvL0 − mvCol (8-197)

其中，tb 和 td 按下式导出：

tb = Clip3(−128, 127, DiffPicOrderCnt(currPicOrField, pic0)) (8-198)

td = Clip3(−128, 127, DiffPicOrderCnt(pic1, pic0)) (8-199)

注 4 — mvL0和mvL1不能超过附件A规定的范围。

表示预测使用情况的标志位 predFlagL0 和 predFlagL1 均置为 1。

图 8-2 表示当前图像在时间上位于列表 0 中的参考图像和列表 1 中的参考图像之间时，时域直接模式运动矢量的推导过程。

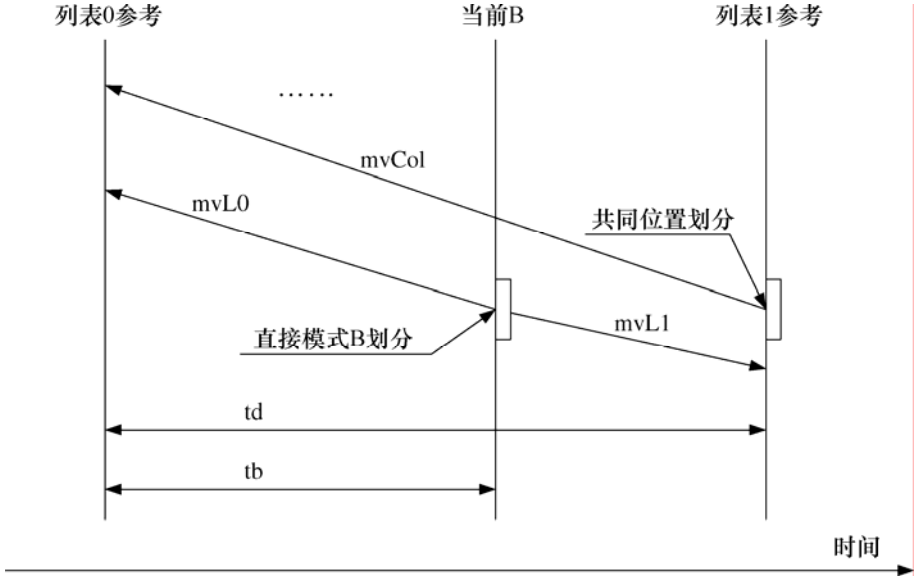


图 8-2－时域直接模式运动矢量推导举例（资料性）

8.4.1.3 亮度运动矢量预测值的推导过程

本过程的输入为：

— 宏块分割块索引mbPartIdx，

— 子宏块分割块索引subMbPartIdx，

— 当前分割块的参考索引refIdxLX（X为0或1），

— 变量currSubMbType。

本过程的输出为运动矢量 mvLX（X 为 0 或 1）的预测值 mvpLX。

以 mbPartIdx、subMbPartIdx、currSubMbType 和 listSuffixFlag = X（X 为 0 和 1，refIdxLX 分别为 refIdxL0 或 refIdxL1）为输入，调用 8.4.1.3.2 节定义的相邻块运动数据推导过程，mbAddrN\mbPartIdxN\subMbPartIdxN、参考索引 refIdxLXN 以及运动矢量 mvLXN（N 可以用 A、B 或 C 替代）作为输出。

除非下列条件之一为真，以 mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN（N 以 A、B 或 C 替代）为输入，调用 8.4.1.3.1 定义的中值亮度运动矢量推导过程，mvpLX 作为输出：

— MbPartWidth(mb_type)等于16， MbPartHeight(mb_type)等于8， mbPartIdx等于0， refIdxLXB等于refIdxLX，

mvpLX = mvLXB (8-200)

— MbPartWidth(mb_type)等于16， MbPartHeight(mb_type) 等于8， mbPartIdx等于1， refIdxLXA等于refIdxLX，

mvpLX = mvLXA (8-201)

— MbPartWidth(mb_type)等于8， MbPartHeight(mb_type) 等于16， mbPartIdx等于0， refIdxLXA等于refIdxLX，

mvpLX = mvLXA (8-202)

— MbPartWidth(mb_type) 等于8，MbPartHeight(mb_type) 等于16，mbPartIdx等于1，refIdxLXC等于refIdxLX，

mvpLX = mvLXC (8-203)

图 8-3 表示上述非中值预测。

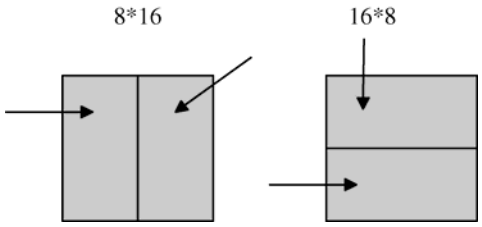


图 8-3－带方向的分段预测（资料性）

8.4.1.3.1 中值亮度运动矢量预测值的推导过程

本过程的输入为：

— 相邻的分割块mbAddrN\mbPartIdxN\subMbPartIdxN（其中的N用A, B或C替代），

— 相邻分割块的运动矢量mvLXN（其中的N用A, B或C替代），

— 相邻分割块的参考索引refIdxLXN（其中的N用A, B或C替代），以及

— 当前分割块的参考索引refIdxLX。

本过程的输出为运动矢量的预测值 mvpLX。

变量 mvpLX 的推导过程如下：

— 当分割块mbAddrB\mbPartIdxB\subMbPartIdxB和mbAddrC\mbPartIdxC\subMbPartIdxC均不可用，且mbAddrA\mbPartIdxA\subMbPartIdxA可用时，

mvLXB = mvLXA (8-204)

mvLXC = mvLXA (8-205)

refIdxLXB = refIdxLXA (8-206)

refIdxLXC = refIdxLXA (8-207)

— 根据参考索引 refIdxLXA、refIdxLXB 和 refIdxLXC 的不同取值，遵照下列规定：

— 如果一个且仅有一个参考索引refIdxLXA、refIdxLXB或refIdxLXC等于当前分割块的参考索引refIdxLX，则遵照下列规定。令refIdxLXN为等于refIdxLX的参考索引，运动矢量mvLXN赋值给运动矢量的预测值mvpLX：

mvpLX = mvLXN (8-208)

— 否则，运动矢量预测值mvpLX的每个分量由运动矢量mvLXA、mvLXB和mvLXC对应分量的中值给出：

mvpLX[0] = Median(mvLXA[0], mvLXB[0], mvLXC[0]) (8-209)

mvpLX[1] = Median(mvLXA[1], mvLXB[1], mvLXC[1]) (8-210)

8.4.1.3.2 相邻分割块运动矢量数据的推导过程

本过程的输入为：

— 宏块分割块索引mbPartIdx，

— 子宏块分割块索引subMbPartIdx，

— 当前子宏块的类型currSubMbType，

— 列表后缀标志listSuffixFlag

本过程的输出为（N 用 A、B 或 C 取代）：

— 代表相邻分割块的mbAddrN\mbPartIdxN\subMbPartIdxN，

— 相邻分割块的运动矢量mvLXN，以及

— 相邻分割块的参考索引refIdxLXN

对于包含字符串“LX”的变量名，其中的 X 等于 listSuffixFlag 的值。

分割块 mbAddrN\mbPartIdxN\subMbPartIdxN（N 用 A、B 或 C 取代）依次按照下列步骤推导：

1．令 mbAddrD\mbPartIdxD\subMbPartIdxD 为规定额外相邻分割块的变量。

2．调用 6.4.8.5 节定义的过程，mbPartIdx, currSubMbType 和 subMbPartIdx 作为输入，输出赋值给 mbAddrN\mbPartIdxN\subMbPartIdxN（其中 N 以 A、B、C 或 D 替代）。

3．当分割块 mbAddrC\mbPartIdxC\subMbPartIdxC 不可用时，则

mbAddrC = mbAddrD (8-211)

mbPartIdxC = mbPartIdxD (8-212)

subMbPartIdxC = subMbPartIdxD (8-213)

运动矢量 mvLXN 和参考索引 refIdxLXN（其中 N 以 A、B 或 C 替代）推导如下：

— 如果宏块分割块或子宏块分割块mbAddrN\mbPartIdxN\subMbPartIdxN不可用，或者mbAddrN在帧内预测模式下编码，或者mbAddrN\mbPartIdxN\subMbPartIdxN的predFlagLX等于0，则mvLXN的两个分量均置为0且refIdxLXN置为−1。

— 否则，遵照如下规定：

— 使运动矢量 mvLXN 和参考索引 refIdxLXN 分别等于 MvLX[mbPartIdxN][subMbPartIdxN] 和 RefIdxLX[mbPartIdxN]，即赋值给（子）宏块分割块mbAddrN\mbPartIdxN\subMbPartIdxN的运动矢量mvLX和参考索引refIdxLX。

— 变量mvLXN[1]和refIdxLXN进一步处理如下：

— 如果当前宏块为场宏块，且宏块mbAddrN为帧宏块，则

mvLXN[1] = mvLXN[1] / 2 (8-214)

refIdxLXN = refIdxLXN * 2 (8-215)

— 否则，如果当前宏块为帧宏块且宏块mbAddrN为场宏块，那么

mvLXN[1] = mvLXN[1] * 2 (8-216)

refIdxLXN = refIdxLXN / 2 (8-217)

— 否则，运动矢量垂直分量mvLXN[1]和参考索引refIdxLXN保持不变。

8.4.1.4 色度运动矢量的推导过程

仅当 chroma_format_idc 不等于 0 时（单色），调用本过程。

本过程的输入为亮度运动矢量 mvLX 和参考索引 refIdxLX。

本过程的输出为色度运动矢量 mvCLX。

色度运动矢量由对应的亮度运动矢量导出。

色度运动矢量水平分量的精度为 1 ÷ (4 * SubWidthC)，垂直分量的精度为 1 ÷ (4 * SubHeightC)。

注 — 例如，当使用4:2:0色度格式时，由于亮度运动矢量的单位是亮度样点单位的1/4，色度分量的水平和垂直分辨率是亮度的1/2，则色度运动矢量的单位为色度样点单位的1/8，比如一个值为1的色度运动矢量代表一个1/8色度样点位移。例如，当亮度运动矢量用于8×16亮度样点块时，相应的4:2:0色度格式下色度矢量用于4×8色度样点块；当亮度矢量用于4×4亮度块时，对应的4:2:0色度格式下色度矢量用于2×2色度样点块。

对于运动矢量 mvCLX 的推导，应用如下规则：

— 如果chroma_format_idc不等于1，或者当前宏块为帧宏块，则亮度运动矢量mvCLX的水平和垂直分量按下式得到：

mvCLX[0] = mvLX[0] (8-218)

mvCLX[1] = mvLX[1] (8-219)

— 否则（chroma_format_idc等于1，且当前宏块为场宏块），只有色度运动矢量mvCLX[0]的水平分量使用8-218式推导。亮度运动矢量mvCLX[1]的垂直分量则取决于当前场或当前宏块以及（由参考索引refIdxLX所指的）参考图像的奇偶性。mvCLX[1]根据表8-10的规定由mvLX[1]导出。

表 8-10－场编码模式下色度矢量垂直分量的导出过程

奇偶性		**mvCLX[1]**
参考图像（refIdxLX）	当前场（图像/宏块）	
顶场	底场	mvLX[1] + 2
底场	顶场	mvLX[1] − 2
其他		mvLX[1]

8.4.2 帧间预测样点的解码过程

本过程的输入为：

— 宏块分割块mbPartIdx，

— 子宏块分割块subMbPartIdx，

— 规定亮度和色度（如果存在）分割块宽度和高度的变量partWidth, partHeight, partWidthC（如果存在）和partHeightC （如果存在），

— 亮度运动矢量mvL0和mvL1，以及当chroma_format_idc不等于0（单色）时的色度运动矢量mvCL0和mvCL1，

— 参考索引refIdxL0和refIdxL1，

— 表示预测列表使用的标志位predFlagL0和predFlagL1，

本过程的输出为：

— 帧间预测样点predPart，包括一个(partWidth) × (partHeight)的亮度样点阵列预测值$predPart_L$，以及当chroma_format_idc不等于0（单色）时的两个(partWidthC)x(partHeightC) 阵列的色度样点预测值$predPart_{Cb}$, $predPart_{Cr}$，分别为色度分量Cb和Cr。

令 $predPartL0_L$ 和 $predPartL1_L$ 为预测得到的(partWidth)x(partHeight)亮度样点值阵列，并且当 chroma_format_idc 不等于 0（单色）时，令 $predPartL0_{Cb}$, $predPartL1_{Cb}$, $predPartL0_{Cr}$ 和 $predPartL1_{Cr}$ 为预测得到的(partWidthC)x(partHeightC)色度样点值阵列。

变量 predFlagLX, RefPicListX, refIdxLX, refPicLX, predPartLX 中的 LX 以 L0 或 L1 替换，对它们的规定如下。

当 predFlagLX 等于 1 时，应用如下规则：

— 调用8.4.2.1节定义的过程得到一个参考图像，它包含一个经过排序的二维亮度样点阵列$refPicLX_L$，以及当chroma_format_idc不等于0（单色）时两个经过排序的二维色度样点阵列$refPicLX_{Cb}$和$refPicLX_{Cr}$的参考图像。以给定的refIdxLX和RefPicListX作为过程的输入。

— 调用8.4.2.2节定义的过程，得到阵列$predPartLX_L$和当chroma_format_idc不等于0（单色）时的阵列$predPartLX_{Cb}$以及$predPartLX_{Cr}$。以mbPartIdx\subMbPartIdx规定的当前分割块、运动矢量mvLX, mvCLX（如果可用）和参考阵列$refPicLX_L$, $refPicLX_{Cb}$（如果可用）以及$refPicLX_{Cr}$（如果可用）作为过程的输入。

对可以用 L、Cb（如果可用）或 Cr（如果可用）替换的 C，通过调用 8.4.2.3 节定义的过程推导 C 分量的预测样点阵列 $predPart_C$，该过程的输入为以 mbPartIdx 和 subMbPartIdx 所规定的当前分割块、阵列 $predPartL0_C$ 和 $predPartL1_C$，以及 predFlagL0 和 predFlagL1。

8.4.2.1 参考图像选择过程

本过程的输入为参考索引 refIdxLX 。

本过程的输出为一个参考图像，包含一个二维亮度样点阵列 $refPicLX_L$ 和两个二维色度样点阵列 $refPicLX_{Cb}$ 和 $refPicLX_{Cr}$。

根据 field_pic_flag 的不同取值，参考图像列表 RefPicListX（其定义见 8.2.4）的组成如下：

— 如果 field_pic_flag 等于 1，则 RefPicListX 中的每一项均为一个参考场或参考帧中的一个场。

— 否则（field_pic_flag等于0），RefPicListX中的每一项均为一个参考帧或者一个互补参考场对。

对于参考图像的推导过程，有如下规定：

— 如果field_pic_flag等于1，则输出为参考场或参考帧中的场RefPicListX[refIdxLX]。该场由一个($PicWidthInSamples_L$)x($PicHeightInSamples_L$)的亮度样点阵列组成，在chroma_format_idc不等于0时还包括两个($PicWidthInSamples_C$)x($PicHeightInSamples_C$)的色度样点阵列$refPicLX_{Cb}$和$refPicLX_{Cr}$。

— 否则（field_pic_flag 等于 0），

— 如果当前宏块为帧宏块，则输出为参考帧或互补参考场对RefPicListX[refIdxLX]。该参考帧或互补参考场对包括亮度样点$refPicLX_L$ 的($PicWidthInSamples_L$)x($PicHeightInSamples_L$)阵列，在chroma_format_idc不等于0（单色）时，还包括两个色度样点$refPicLX_{Cb}$和$refPicLX_{Cr}$阵列($PicWidthInSamples_C$)x($PicHeightInSamples_C$)。

— 否则（当前宏块为场宏块），下列适用

— 令refFrame为参考帧或互补参考场对RefPicListX[refIdxLX / 2]。

— 对refFrame中的场进行如下选择：

— 如果refIdxLX % 2等于0，则输出为refFrame中与当前宏块具有相同奇偶性的场。

— 否则（refIdxLX % 2等于1），输出为refFrame中与当前宏块具有相反奇偶性的场。

— 输出的参考场或参考帧中的场由一个(PicWidthInSamples$_L$)x(PicHeightInSamples$_L$ / 2)的亮度样点阵列组成，在chroma_format_idc不等于0（单色）时还包含2个(PicWidthInSamples$_L$)x(PicHeightInSamples$_L$ / 2)的色度样点阵列refPicLX$_{Cb}$和refPicLX$_{Cr}$。

参考图像样点阵列 refPicLX$_L$、refPicLX$_{Cb}$（如果可用）和 refPicLX$_{Cr}$（如果可用）与通过调用 8.7 定义的过程所得到的已解码样点阵列 S_L、S_{Cb}（如果可用）和 S_{Cr}（如果可用）相对应,得到的 S_L、S_{Cb} 和 S_{Cr} 用于于先前解码的参考场，或参考帧，或互补参考场对，或参考帧中的一个场。

8.4.2.2 非整数样点的内插过程

本过程的输入为：

— 由分割块索引mbPartIdx指定的当前分割块，及其子宏块分割块索引subMbPartIdx，

— 该分割块的高度partHeight和宽度partWidth，以亮度样点为单位，

— 亮度运动矢量mvLX，以1/4亮度样点为单位，

— 色度运动矢量mvCLX，以1/8色度样点为单位，以及

— 选定的参考图像样点阵列refPicLX$_L$，refPicLX$_{Cb}$和refPicLX$_{Cb}$ 。

本过程的输出为：

— 一个(partWidth)x(partHeight)的预测亮度样点预测值的阵列predPartLX$_L$，以及

— 当chroma_format_idc不等于0（单色）时，两个(partWidthC)x(partHeightC)的色度样点预测值阵列predPartLX$_{Cb}$和predPartLX$_{Cr}$。

令(xA$_L$, yA$_L$)为当前分割块左上角样点与给定的二维亮度样点阵列左上角亮度样点的相对位置，以整样点为单位给出，该当前分割块由索引 mbPartIdx\subMbPartIdx 指出。

令(xInt$_L$, yInt$_L$)为以整样点为单位给出的亮度位置，(xFrac$_L$, yFrac$_L$)为以 1/4 样点为单位给出的偏移量。这些变量仅在本小节内使用，用来定义参考样点阵列 refPicLX$_L$、refPicLX$_{Cb}$（如果可用）和 refPicLX$_{Cr}$（如果可用）内一般非整数位置。

对于每个预测亮度样点阵列 predPartLX$_L$ 内的亮度样点位置（0 <= x_L < partWidth, 0 <= y_L < partHeight），对应的预测样点值 predPartLX$_L$[x_L, y_L]的推导过程如下：

— 变量xInt$_L$, yInt$_L$, xFrac$_L$和 yFrac$_L$根据下列公式确定：

$$xInt_L = xA_L + (mvLX[0] >> 2) + x_L \quad (8\text{-}220)$$

$$yInt_L = yA_L + (mvLX[1] >> 2) + y_L \quad (8\text{-}221)$$

$$xFrac_L = mvLX[0] \& 3 \quad (8\text{-}222)$$

$$yFrac_L = mvLX[1] \& 3 \quad (8\text{-}223)$$

— 亮度样点的预测值predPartLX$_L$[x_L, y_L]通过调用8.4.2.2.1节定义的过程得到，该输入为给定的(xInt$_L$, yInt$_L$), (xFrac$_L$, yFrac$_L$)和refPicLX$_L$。

当 chroma_format_idc 不等于 0（单色）时，有下述规定。

令(xInt$_C$, yInt$_C$)为以整样点为单位给出的色度样点位置，(xFrac$_C$, yFrac$_C$)为以 1/8 样点为单位给出的偏移量。这些变量仅在本小节内部使用，用来定义参考样点阵列 refPicLX$_{Cb}$和 refPicLX$_{Cr}$内一般非整数样点位置。

对于每个预测亮度样点阵列 predPartLX$_{Cb}$ 和 predPartLX$_{Cr}$ 内的亮度样点位置（0 <= x_C < partWidthC, 0 <= y_C < partHeightC），对应的预测样点值 predPartLX$_{Cb}$[x_C, y_C]和 predPartLX$_{Cr}$[x_C, y_C]的推导过程如下：

— 根据chroma_format_idc的取值，变量xInt$_C$, yInt$_C$, xFrac$_C$和yFrac$_C$的推导过程如下：

— 如果chroma_format_idc等于1，则

$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 3) + x_C$ (8-224)

$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 3) + y_C$ (8-225)

$xFrac_C = mvCLX[0] \& 7$ (8-226)

$yFrac_C = mvCLX[1] \& 7$ (8-227)

— 否则，如果chroma_format_idc等于2，

$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 3) + x_C$ (8-228)

$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 2) + y_C$ (8-229)

$xFrac_C = mvCLX[0] \& 7$ (8-230)

$yFrac_C = (mvCLX[1] \& 3) << 1$ (8-231)

— 否则（chroma_format_idc等于3），

$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 2) + x_C$ (8-232)

$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 2) + y_C$ (8-233)

$xFrac_C = (mvCLX[0] \& 3) << 1$ (8-234)

$yFrac_C = (mvCLX[1] \& 3) << 1$ (8-235)

— 预测样点值$predPartLX_{Cb}[x_C, y_C]$通过调用8.4.2.2.2节定义的过程得到，此时该过程的输入为给定的$(xInt_C, yInt_C)$, $(xFrac_C, yFrac_C)$和$refPicLX_{Cb}$。

— 预测样点值$predPartLX_{Cr}[x_C, y_C]$通过调用8.4.2.2.2节定义的过程得到，此时该过程的输入为给定的$(xInt_C, yInt_C)$, $(xFrac_C, yFrac_C)$和$refPicLX_{Cr}$。

8.4.2.2.1 亮度样点的内插过程

本过程的输入为：

— 亮度位置$(xInt_L, yInt_L)$，以整样点为单位，

— 亮度位置偏移量$(xFrac_L, yFrac_L)$，以非整数样点为单位，以及

— 选定参考图像$refPicLX_L$中的亮度样点阵列

本过程的输出为一个预测亮度样点的值 $predPartLX_L[x_L, y_L]$。

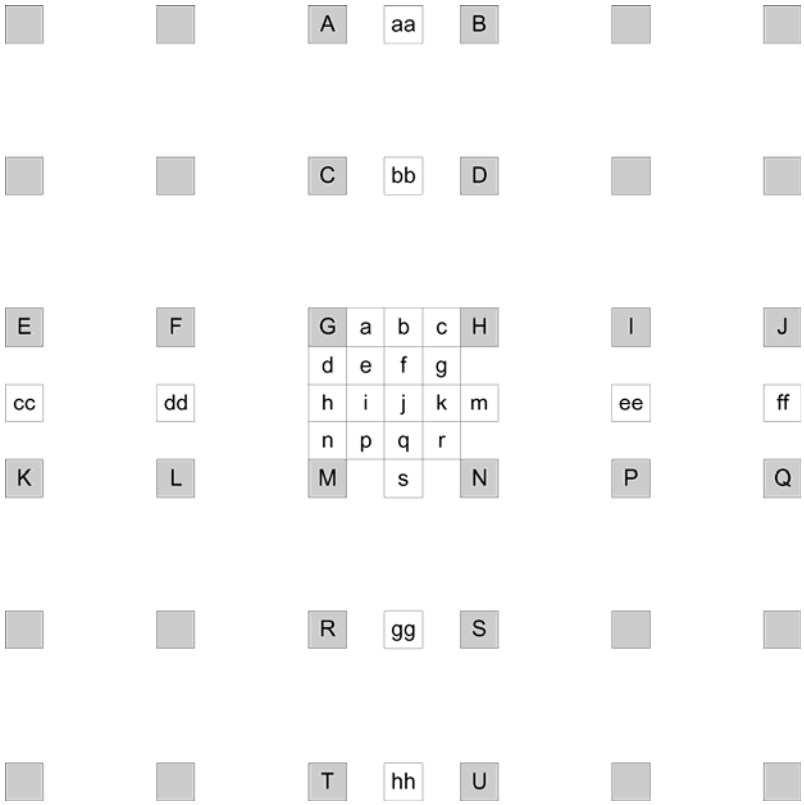


图 8-4－1/4样点亮度内插时的整数样点（标有大写字母的阴影块）和非整数样点位置（标有小写字母的白色块）

表示有效参考图像亮度阵列高度的变量 $refPicHeightEffective_L$ 推导如下：

— 如果 MbaffFrameFlag 等于 0，或者 mb_field_decoding_flag 等于 0，则置 $refPicHeightEffective_L$ 为等于 $PicHeightInSamples_L$。

— 否则（MbaffFrameFlag 等于 1 且 mb_field_decoding_flag 等于 1），则置 $refPicHeightEffective_L$ 为等于 $PicHeightInSamples_L$ / 2。

在图 8-4 中，标有大写字母的阴影块表示给定二维亮度样点阵列 $refPicLX_L$ 中整样点位置上的亮度样点。这些样点可能会被用来预测亮度样点值 $predPartLX_L[x_L, y_L]$。亮度样点阵列 $refPicLX_L$ 中的每个(xZ_L, yZ_L)位置根据下式导出。其中的 Z 可能为 A, B, C, D, E, F, G, H, I, J, K, L, M, N, P, Q, R, S, T 和 U 中的一个。

$$xZ_L = Clip3(0, PicWidthInSamples_L - 1, xInt_L + xDZ_L)$$
$$yZ_L = Clip3(0, refPicHeightEffective_L - 1, yInt_L + yDZ_L) \quad (8\text{-}236)$$

表 8-11 定义了不同 Z 所对应的(xDZ_L, yDZ_L)的值。

表 8-11－整样点亮度位置差分值

Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	P	Q	R	S	T	U
xDZ_L	0	1	0	1	–2	–1	0	1	2	3	–2	–1	0	1	2	3	0	1	0	1
yDZ_L	–2	–2	–1	–1	0	0	0	0	0	0	1	1	1	1	1	1	2	2	3	3

假设亮度样点‘A’到‘U’位于整样点位置(xA_L, yA_L)到(xU_L, yU_L)上 ，则非整数样点位置上的亮度样点‘a’到‘s’按照下列规则导出。半样点位置上的亮度预测值应该使用系数为（ 1, –5, 20, 20, –5, 1）的 6 拍滤波器计算得到。1/4 样点位置上的亮度预测值应该通过计算整数和半样点位置的平均值得到。每个非整数位置的推导过程描述如下：

— 为了计算标记为b的半样点位置上的样点值，首先应该通过对临近的整数位置样点进行水平方向6拍滤波，计算得到中间值b_1。为了计算半样点位置上的样点值h，首先应该通过对临近的整数位置样点进行垂直方向6拍滤波，计算得到中间值h_1：

$$b_1 = (E - 5 * F + 20 * G + 20 * H - 5 * I + J) \quad (8\text{-}237)$$

$$h_1 = (A - 5 * C + 20 * G + 20 * M - 5 * R + T) \quad (8\text{-}238)$$

b 和 h 的最终预测值应该通过下列式子得到：

$$b = Clip1_Y((b_1 + 16) >> 5) \quad (8\text{-}239)$$

$$h = Clip1_Y((h_1 + 16) >> 5) \quad (8\text{-}240)$$

— 为了计算半样点位置上的样点值j，首先应该通过对临近的半整数位置样点进行水平或垂直方向（二者得到的结果相等）6拍滤波，计算得到中间值j_1：

$$j_1 = cc - 5 * dd + 20 * h_1 + 20 * m_1 - 5 * ee + ff，或 \quad (8\text{-}241)$$

$$j_1 = aa - 5 * bb + 20 * b_1 + 20 * s_1 - 5 * gg + hh \quad (8\text{-}242)$$

其中，中间值 aa、bb、gg、s_1和 hh 应该使用与 b_1相同的方法进行水平 6 拍滤波导出，cc、dd、ee、m_1和 ff 应该使用与 h_1相同的方法进行水平 6 拍滤波导出。j 的最终预测值应该通过下式得到：

$$j = Clip1_Y((j_1 + 512) >> 10) \quad (8\text{-}243)$$

— 最终预测值s和m应该采用与b和h相同的导出方法根据s_1和m_1按照下式得到：

$$s = Clip1_Y((s_1 + 16) >> 5) \quad (8\text{-}244)$$

$$m = Clip1_Y((m_1 + 16) >> 5) \quad (8\text{-}245)$$

— 1/4样点位置a, c, d, n, f, i, k和q上的值应该通过计算两个最接近的整数和半整数位置的平均值（只入不舍）得到：

$$a = (G + b + 1) >> 1 \quad (8\text{-}246)$$

$$c = (H + b + 1) >> 1 \quad (8\text{-}247)$$

$$d = (G + h + 1) >> 1 \quad (8\text{-}248)$$

$$n = (M + h + 1) >> 1 \quad (8\text{-}249)$$

$$f = (b + j + 1) >> 1 \quad (8\text{-}250)$$

$$i = (h + j + 1) >> 1 \quad (8\text{-}251)$$

$$k = (j + m + 1) >> 1 \quad (8\text{-}252)$$

$$q = (j + s + 1) >> 1. \quad (8\text{-}253)$$

— 1/4 样点位置 e, g, p 和 r 上的值应该通过计算对角线方向上两个最接近的半整数位置的平均值（只入不舍）得到：

$$e = (b + h + 1) >> 1 \quad (8\text{-}254)$$

$$g = (b + m + 1) >> 1 \quad (8\text{-}255)$$

$$p = (h + s + 1) >> 1 \quad (8\text{-}256)$$

$$r = (m + s + 1) >> 1. \quad (8\text{-}257)$$

以非整数样点为单位的亮度偏移位置（$xFrac_L$, $yFrac_L$）规定了哪个生成的亮度整数和非整数位置上的值被指定为 $predPartLX_L[x_L, y_L]$的值。这样的赋值根据表 8-12 进行。$predPartLX_L[x_L, y_L]$的取值应该作为输出。

表 8-12－亮度预测样点predPartLX$_L$[x_L, y_L]的取值

$xFrac_L$	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
$yFrac_L$	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
predPartLX$_L$[x_L, y_L]	G	d	h	n	a	e	i	p	b	f	j	q	c	g	k	r

8.4.2.2.2 色度样点的内插过程

仅当 chroma_format_idc 不等于 0（单色）时，调用本过程。

本过程的输入为：

— 以整样点为单位的色度位置 ($xInt_C$, $yInt_C$)，

— 以非整数样点为单位的色度位置偏移量($xFrac_C$, $yFrac_C$)，以及

— 选定参考图像中的色度分量样点refPicLX$_C$。

本过程的输出为一个色度预测样点值 predPartLX$_C$[x_C, y_C]。

图 8-5 中，位置 A、B、C 和 D 代表给定的二维亮度样点阵列 refPicLX$_C$ 中整样点位置上的亮度样点。

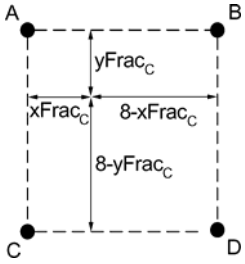


图 8-5－色度内插中的非整数样点位置与周围整数位置样点A、B、C和D的关系

变量 refPicHeightEffective$_C$ 为有效参考图像色度阵列的高度，导出过程如下：

— 如果 MbaffFrameFlag 等于 0，或者 mb_field_decoding_flag 等于 0，则 refPicHeightEffective$_C$ 等于 PicHeightInSamples$_C$。

— 否则（MbaffFrameFlag 等于 1 且 mb_field_decoding_flag 等于 1），refPicHeightEffective$_C$ 等于 PicHeightInSamples$_C$ / 2。

色度样点预测值 predPartLX$_C$[x_C, y_C]的生成用到了等式 8-258 到 8-265 定义的坐标。

$xA_C = Clip3(0, PicWidthInSamples_C - 1, xInt_C)$ (8-258)

$xB_C = Clip3(0, PicWidthInSamples_C - 1, xInt_C + 1)$ (8-259)

$xC_C = Clip3(0, PicWidthInSamples_C - 1, xInt_C)$ (8-260)

$xD_C = Clip3(0, PicWidthInSamples_C - 1, xInt_C + 1)$ (8-261)

$yA_C = Clip3(0, refPicHeightEffective_C - 1, yInt_C)$ (8-262)

$yB_C = Clip3(0, refPicHeightEffective_C - 1, yInt_C)$ (8-263)

$yC_C = Clip3(0, refPicHeightEffective_C - 1, yInt_C + 1)$ (8-264)

$yD_C = Clip3(0, refPicHeightEffective_C - 1, yInt_C + 1)$ (8-265)

假设 A、B、C 和 D 为等式 8-258 到 8-265 定义的整样点位置上的色度样点，色度样点的预测值 $predPartLX_C[x_C, y_C]$按下式推导：

$$predPartLX_C[x_C, y_C] = ((8 - xFrac_C) * (8 - yFrac_C) * A + xFrac_C * (8 - yFrac_C) * B + (8 - xFrac_C) * yFrac_C * C + xFrac_C * yFrac_C * D + 32) >> 6 \quad (8\text{-}266)$$

8.4.2.3 样点的加权预测过程

本过程的输入为：

— mbPartIdx：由分割块索引指定的当前分割块

— subMbPartIdx：子宏块分割块索引

— predFlagL0和predFlagL1：表示预测列表使用的标志

— $predPartLX_L$：一个(partWidth)x(partHeight)的亮度样点预测值阵列（根据predFlagL0和predFlagL1的取值，LX可用L0和L1替换）

— 当chroma_format_idc不等于0（单色）时的$predPartLX_{Cb}$和$predPartLX_{Cr}$：(partWidthC)x(partHeightC)的色度样点预测值阵列，分别为色度分量Cb和Cr（根据predFlagL0和predFlagL1的取值，LX可用L0和L1替换）

本过程的输出为：

— $predPart_L$：一个(partWidth)x(partHeight)的亮度样点预测阵列，以及

— 当chroma_format_idc不等于0（单色）时的$predPart_{Cb}$和$predPart_{Cr}$：(partWidthC)x(partHeightC)的色度样点预测值阵列，分别为色度分量Cb和Cr。

对于P或SP条带中predFlagL0等于1的宏块或分割块，应用下列规则：

— 如果weighted_pred_flag等于0，则调用8.4.2.3.1节中定义的缺省样点加权预测过程，其输入和输出与本节所规定的相同。

— 否则（weighted_pred_flag等于1），调用8.4.2.3.2节中定义的显式样点加权预测过程，其输入和输出与本节所规定的相同。

对于B条带中predFlagL0或predFlagL1等于1的宏块或分割块，应用下列规则：

— 如果weighted_bipred_idc等于0，则调用8.4.2.3.1节中定义的缺省样点加权预测过程，其输入和输出与本节所规定的相同。

— 否则（weighted_bipred_idc等于1），调用8.4.2.3.2节中定义的显式样点加权预测过程，其输入和输出与本节所规定的相同。

— 否则（weighted_bipred_idc等于2），

  — 如果predFlagL0等于1且predFlagL1等于1，则调用8.4.2.3.2节定义的显式样点加权预测过程，其输入和输出与本节所规定的相同。

  — 否则（(predFlagL0或predFlagL1等于1，但不是同时等于1），调用8.4.2.3.1节定义的缺省样点加权预测过程，其输入和输出与本节所规定的相同。

8.4.2.3.1 缺省的样点加权预测过程

本过程的输入与 8.4.2.3 节中定义的相同。

本过程的输出与 8.4.2.3 节中定义的相同。

根据所推导的预测块的可用组成不同，应用下列规则：

— 如果推导的是亮度样点预测值$predPart_L[x, y]$，则应用如下规则且令C等于L，x等于0…partWidth − 1，y等于0…partHeight − 1。

— 否则，如果推导的是色度分量Cb样点的预测值$predPart_{Cb}[x, y]$，应用如下规则且令C等于Cb，x等于0 .. partWidthC − 1，y等于0 .. partHeightC − 1。

— 否则（推导的是色度分量Cr预测值$predPart_{Cr}[x, y]$），则应用如下规则且令C等于Cr，x等于0 .. partWidthC − 1，y等于0 .. partHeightC − 1。

预测样点值推导过程如下：

— 如果对于当前分割块，predFlagL0等于1，且predFlagL1等于0，则

$$predPart_C[x, y] = predPartL0_C[x, y] \quad (8\text{-}267)$$

— 否则，如果对于当前分割块，predFlagL0等于0且predFlagL1等于1，那么

$$predPart_C[x, y] = predPartL1_C[x, y] \quad (8\text{-}268)$$

— 否则（对于当前分割块，predFlagL0和predFlagL1都等于1），

$$predPart_C[x, y] = (predPartL0_C[x, y] + predPartL1_C[x, y] + 1) >> 1。 \quad (8\text{-}269)$$

8.4.2.3.2 样点的加权预测过程

本过程的输入与 8.4.2.3 节所定义的过程相同。

本过程的输出与 8.4.2.3 节所定义的过程相同。

根据所推导的预测块的可用组成不同，应用下列规则：

— 如果推导的是亮度样点预测值$predPart_L[x, y]$，则应用如下规则且令C等于L，x等于0…partWidth − 1，y等于0…partHeight − 1。

— 否则，如果推导的是色度分量Cb样点的预测值$predPart_{Cb}[x, y]$，应用如下规则且令C等于Cb，x等于0 .. partWidthC − 1，y等于0 .. partHeightC − 1。

— 否则（推导的是色度分量Cr预测值$predPart_{Cr}[x, y]$），应用如下规则且令C等于Cr，x等于0 .. partWidthC − 1，y等于0 .. partHeightC − 1。

预测样点值推导过程如下：

— 如果分割块mbPartIdx\subMbPartIdx的predFlagL0等于1并且predFlagL1等于0，则最终预测值$predPart_C[x, y]$推导如下：

$$\begin{aligned} &\text{if(logWD >= 1)} \\ &\quad predPart_C[x, y] = Clip1_C(((predPartL0_C[x, y] * w_0 + 2^{logWD - 1}) >> logWD) + o_0) \\ &\text{else} \\ &\quad predPart_C[x, y] = Clip1_C(predPartL0_C[x, y] * w_0 + o_0) \end{aligned} \quad (8\text{-}270)$$

— 否则，如果分割块mbPartIdx\subMbPartIdx的predFlagL0等于0，并且predFlagL1等于1，则最终预测值$predPart_C[x, y]$推导如下：

$$\begin{aligned} &\text{if(logWD >= 1)} \\ &\quad predPart_C[x, y] = Clip1_C(((predPartL1_C[x, y] * w_1 + 2^{logWD - 1}) >> logWD) + o_1) \\ &\text{else} \\ &\quad predPart_C[x, y] = Clip1_C(predPartL1_C[x, y] * w_1 + o_1) \end{aligned} \quad (8\text{-}271)$$

— 否则（分割块mbPartIdx\subMbPartIdx的predFlagL0和predFlagL1都等于1），最终预测值$predPart_C[x, y]$推导如下：

$$\begin{aligned} predPart_C[x, y] = Clip1_C(((predPartL0_C[x, y] * w_0 + predPartL1_C[x, y] * w_1 + 2^{logWD}) >> \\ (logWD + 1)) + ((o_0 + o_1 + 1) >> 1)) \end{aligned} \quad (8\text{-}272)$$

上述预测样点推导中用到的变量的导出过程如下：

— 如果weighted_bipred_idc等于2且slice_type等于B，则按照下列规定进行隐式加权预测：

$$logWD = 5 \quad (8\text{-}273)$$

$o_0 = 0$ (8-274)

$o_1 = 0$ (8-275)

w_0和w_1推导如下：

— 变量currPicOrField，pic0和pic1的推导过程如下：

— 如果field_pic_flag等于0，且当前宏块为场宏块，则

— currPicOrField为当前图像CurrPic中与当前宏块具有相同奇偶性的场。

— 变量pic0推导如下：

— 如果refIdxL0 % 2等于0，则pic0为RefPicList0[refIdxL0 / 2]中与当前宏块具有相同奇偶性的场。

— 否则（refIdxL0 % 2不等于0），pic0为RefPicList0[refIdxL0 / 2]中与当前宏块具有相反奇偶性的场。

— 变量pic1推导如下：

— 如果refIdxL1 % 2等于0，则pic1为RefPicList0[refIdxL1 / 2]中与当前宏块具有相同奇偶性的场。

— 否则（refIdxL1 % 2不等于0），pic1为RefPicList0[refIdxL1 / 2]中与当前宏块具有相反奇偶性的场。

— 否则（field_pic_flag等于1，或者当前宏块为帧宏块），currPicOrField为当前图像CurrPic，pic1为RefPicList1[refIdxL1]，且pic0为RefPicList0[refIdxL0]。

— 变量tb, td, tx和DistScaleFactor根据currPicOrField, pic0, pic1的值分别由等式8-198，8-199，8-194以及8-195导出。

— 如果DiffPicOrderCnt(pic1, pic0)等于0，或者pic1和pic0其中之一或两者都被标志为“用于长期参考”，或者(DistScaleFactor >> 2) < –64 or (DistScaleFactor >> 2) > 128，则w_0和w_1推导如下：

$w_0 = 32$ (8-276)

$w_1 = 32$ (8-277)

— 否则，

$w_0 = 64 - (DistScaleFactor >> 2)$ (8-278)

$w_1 = DistScaleFactor >> 2$ (8-279)

— 否则（P或SP条带中的weighted_pred_flag等于1，或者B条带中的weighted_bipred_idc等于1），则按照下列规定进行显式加权预测：

— 变量refIdxL0WP和refIdxL1WP推导如下：

— 如果MbaffFrameFlag等于1且当前宏块为场宏块，则

refIdxL0WP = refIdxL0 >> 1 (8-280)

refIdxL1WP = refIdxL1 >> 1 (8-281)

— 否则（MbaffFrameFlag等于0，或当前宏块为帧宏块），

refIdxL0WP = refIdxL0 (8-282)

refIdxL1WP = refIdxL1 (8-283)

— 变量logWD, w_0, w_1, o_0以及o_1推导如下：

— 如果$predPart_C$[x, y]的C用L替换，代表亮度样点，则

logWD = luma_log2_weight_denom (8-284)

w_0 = luma_weight_l0[refIdxL0WP] (8-285)

w_1 = luma_weight_l1[refIdxL1WP] (8-286)

o_0 = luma_offset_l0[refIdxL0WP] * (1 << ($BitDepth_Y$ − 8)) (8-287)

o_1 = luma_offset_l1[refIdxL1WP] * (1 << ($BitDepth_Y$ − 8)) (8-288)

— 否则（如果$predPart_C$[x, y]中的C用Cb或Cr替换，分别代表两个分量的色度样点，则对于Cb，iCbCr = 0；对于Cr，iCbCr = 1。）

logWD = chroma_log2_weight_denom (8-289)

w_0 = chroma_weight_l0[refIdxL0WP][iCbCr] (8-290)

w_1 = chroma_weight_l1[refIdxL1WP][iCbCr] (8-291)

o_0 = chroma_offset_l0[refIdxL0WP][iCbCr] * (1 << ($BitDepth_C$ − 8)) (8-292)

o_1 = chroma_offset_l1[refIdxL1WP][iCbCr] * (1 << ($BitDepth_C$ − 8)) (8-293)

当进行显式加权预测，且分割块 mbPartIdx\subMbPartIdx 的 predFlagL0 和 predFlagL1 都等于 1 时，应该遵照下列规定：

−128 <= w_0 + w_1 <= ((logWD == 7) ? 127 : 128) (8-294)

注 — 对于隐式模式的加权预测，权重w_0和w_1的取值需在−64…128之间，并且，尽管没有明确指出它们也必须始终满足式8-294所表示的约束。而对于logWD等于7的显式模式的加权预测，当一个或两个权重w_0或w_1等于128（由于luma_weight_l0_flag，luma_weight_l1_flag，chroma_weight_l0_flag，或chroma_weight_l1_flag等于0），则另外的一个权重(w_1或w_0)应该为负值，以便满足式8-294的规定（从而，另外的标志位luma_weight_l0_flag, luma_weight_l1_flag，chroma_weight_l0_flag，或chroma_weight_l1_flag应该等于1）。

8.5 位于去块效应滤波过程之前的变换系数解码过程以及图像重建过程

本过程的输入是 Intra16x16DCLeve（如果可用），Intra16x16ACLevel（如果可用），LumaLevel（如果可用），LumaLevel8x8（如果可用），ChromaDCLevel（如果可用），ChromaACLevel（如果可用），以及可能的用于当前宏块以及可用分量 $pred_L$、$pred_{Cb}$或者 $pred_{Cr}$的帧间或者帧内预测样点数组。

注 1 — 当采用Intra_4x4 (或者Intra_8x8)预测模式来解码宏块的时候，因为对于每个4x4 (或者8x8)的亮度块，第8.3.1节（或者8.3.2）中定义的亮度样点的Intra_4x4 (或者Intra_8x8)预测过程以及在本节中定义的过程是重复的，所以宏块预测数组可能不完整。

本过程的输出是用于可应用分量 S'_L, S'_{Cb}或者 S'_{Cr}.的去块效应滤波过程的前面的重建样点数组。

注 2 — 当采用Intra_4x4 (或者Intra_8x8)预测模式来解码一个宏块时，因为对于每个4x4 (或者8x8)的亮度块，第8.3.1节（或者8.3.2）中定义的亮度样点的Intra_4x4 (或者Intra_8x8)预测过程以及在本节中定义的过程是重复的，所以位于去块效应滤波过程之前的由宏块重建的样点数组的亮度分量可能不完整。

本节定义了位于去块效应滤波过程前面的变换系数解码以及图像重建过程。

把当前的宏块编码为 P_Skip 或 B_Skip 时，当前宏块的 LumaLevel, LumaLevel8x8, ChromaDCLevel, ChromaACLevel 的所有值都置 0。

当 residual_colour_transform_flag 等于 1 时，调用第 8.5.13 节中定义的残差色彩变换过程。

8.5.1 对用于4x4亮度残差块的变换解码过程的规范

这个定义适用于 transform_size_8x8_flag 等于 0 的情况。

当前的宏块预测模式不是 Intra_16x16 时，变量 LumaLevel 包含了用于亮度变换系数的幅值。对于一个由 luma4x4BlkIdx = 0..15 来索引的 4x4 亮度块，定义了下述操作步骤。

1．调用 8.5.5 中描述的反向变换系数扫描过程，LumaLevel[luma4x4BlkIdx]为输入，两维数组 c 为输出。

2．调用 8.5.10 中定义的残差 4x4 块的缩放以及变换过程，c 为输入，r 为输出。

3．当 residual_colour_transform_flag 置 1 时，变量 $R_{Y,ij}$ 置为 r_{ij}，r_{ij} 中 i, j = 0..3，并且在第 8.5.13 节中定义的残差色彩变换过程完成之前暂停该过程，在残差色彩变换过程完成以后，将变量 r_{ij} 置为 $R_{G,ij}$，其中 i, j = 0..3，然后再继续该过程。

4．通过调用第 6.4.3 节中描述的反 4x4 亮度块扫描过程来推导在宏块中索引为 luma4x4BlkIdx 的一个 4x4 亮度块左上角样点的位置，把 luma4x4BlkIdx 作为输入，输出分配给(xO, yO)。

5．通过下述方式得到元素为 u_{ij} 的 4x4 数组 u，其中 i, j = 0..3。

$$u_{ij} = Clip1_Y(pred_L[xO + j, yO + i] + r_{ij}) \quad (8\text{-}295)$$

当 qpprime_y_zero_transform_bypass_flag 为 1，QP'_Y 为 0 时，比特流不应该包含可以导致通过方程式 8-295 计算出来的 u_{ij} 值不等于 $pred_L[xO + j, yO + i] + r_{ij}$ 的数据。

6．调用第 8.5.12 节中定义的位于去块效应滤波过程之前的图像重建过程，luma4x4BlkIdx 和 u 为输入。

8.5.2 对用于Intra_16x16宏块预测模式的亮度样点的变换解码过程的规范

当当前的宏块预测模式是 Intra_16x16 时，变量 Intra16x16DCLevel 以及 Intra16x16ACLevel 包含用于亮度变换系数的幅值。按照下面的步骤进行变换系数解码：

1．解码宏块的所有 4x4 亮度块的亮度 DC 变换系数。

a．调用 8.5.5 中描述的反变换系数扫描过程，Intra16x16DCLevel 为输入，2 维数组 c 作为输出。

b．调用 8.5.8 中描述的用于 Intra_16x16 宏块类型的亮度 DC 变换系数的缩放以及变换过程，c 为输入，dcY 为输出。

2．为由 luma4x4BlkIdx = 0..15 索引的一个 4x4 的亮度块定义了下述的步骤。

a．获得一个具有 16 个条目的变量 LumaList。该 lumaList 的第一个条目是来自数组 dcY 的相应的值。图 8-6 给出了数组 dcY 到 luma4x4BlkIdxfe 的索引的分配。小方块中的两个数字指得是 dcYij 中的索引 i 和 j，在大方块中的数字是指 luma4x4BlkIdx。

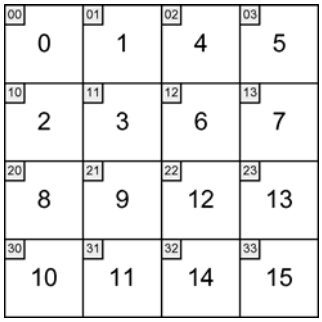


图 8-6－dcY到luma4x4BlkIdx的索引的分配

lumaList 中索引为 k = 1..15 的元素的定义为，

lumaList[k] = Intra16x16ACLevel[luma4x4BlkIdx][k − 1] (8-296)

b．调用第 8.5.5 节中描述的反变换系数扫描过程，lumaList 为输入，2 维数组 c 为输出。

c．调用第 8.5.10 节中描述的用于残差 4x4 块的缩放以及变换过程，c 为输入，r 为输出。

d．当 residual_colour_transform_flag 置 1 时，变量 $R_{Y,ij}$ 置为 rij，rij 中 i, j = 0..3，并且在第 8.5.13 节中定义的残差色彩变换过程完成之前暂停该过程，在残差色彩变换过程完成以后，将变量 rij 置为 RG,ij，其中 i, j = 0..3，然后再继续该过程。

e．通过调用第 6.4.3 节中描述的反 4x4 亮度块扫描过程来推导在宏块中索引为 luma4x4BlkIdx 的一个 4x4 亮度块左上角样点的位置，把 luma4x4BlkIdx 作为输入，输出分配给(xO, yO)。

f．通过下述方式推导出元素为 u_{ij} 的 4x4 数组 u，其中 i, j = 0..3。

$u_{ij} = Clip1_Y(pred_L[xO + j, yO + i] + r_{ij})$ (8-297)

当 qpprime_y_zero_transform_bypass_flag 为 1，QP'_Y 为 0 时，比特流不应该包含可以导致通过 8-297 计算出来的 u_i 的一个值不等于 $pred_L[xO + j, yO + i] + r_{ij}$ 的数据。

g．调用第 8.5.12 节中的位于去块效应滤波过程前面的图像重建过程，luma4x4BlkIdx 和 u 为输入。

8.5.3 对用于8x8亮度残差块的变换解码过程的规范

这个规范适用于 transform_size_8x8_flag 为 1 的情况。

变量 LumaLevel8x8[luma8x8BlkIdx]中包含了用于索引为 luma8x8BlkIdx 的亮度 8x8 块的亮度变换系数的幅值，其中 luma8x8BlkIdx = 0..3。

为通过 luma8x8BlkIdx = 0..3 索引的一个 8x8 亮度块定义了如下的步骤。

1．调用第 8.5.6 节中描述的用于亮度变换系数的反扫描过程，LumaLevel8x8[luma8x8BlkIdx]为输入，2 维数组 c 维输出。

2．为残差 8x8 块调用 8.5.11 中描述的缩放和变换过程，c 为输入，r 为输出。

3．当 residual_colour_transform_flag 置 1 时，变量 $R_{Y,ij}$ 置为 rij，rij 中 i, j = 0..7，并且在第 8.5.13 节中定义的残差色彩变换过程完成之前暂停该过程，在残差色彩变换过程完成以后，将变量 rij 置为 RG,ij，其中 i, j = 0..7，然后再继续该过程。

4．通过调用第 6.4.4 节中描述的反 8x8 亮度块扫描过程来推导在宏块中索引为 luma8x8BlkIdx 的一个 8x8 亮度块左上角样点的位置，luma8x8BlkIdx 为输入，输出分配给(xO, yO)。

5．元素为 u_{ij} 的 8x8 数组 u 通过如下方式得到，其中 i, j = 0..7。

$$u_{ij} = Clip1_Y(pred_L[xO + j, yO + i] + r_{ij}) \quad (8\text{-}298)$$

当 qpprime_y_zero_transform_bypass_flag 为 1，QP'_Y 为 0 时，比特流不应该包含能够导致通过 8-298 计算出来的 u_{ij} 的一个值不等于 $pred_L[xO + j, yO + i] + r_{ij}$ 的数据。

6．调用第 8.5.12 节中描述的位于去块效应滤波过程之前的图像重建过程，luma8x8BlkIdx 和 u 为输入。

8.5.4 对色度样点变换解码过程的定义

为每个色度分量 Cb 以及 Cr 分别调用该过程。

对于每个色度分量，变量 ChromaDCLevel[iCbCr]和 ChromaACLevel[iCbCr]包含适用于色度变换系数的两个分量的幅值，其中，对于 Cb，将 iCbCr 置 0，对于 Cr，将 iCbCr 置 1。

将变量 numChroma4x4Blks 置为(MbWidthC / 4) * (MbHeightC / 4)。

对于每个色度分量，按照下述步骤处理变换解码：

1．解码通过宏块的 iCbCr 来索引的分量的 4x4 色度块的 numChroma4x4Blks 色度 DC 变换系数。

a．根据变量 chroma_format_idc 来执行下面的内容。

— 如果chroma_format_idc为1，通过下面的方式将反光栅扫描过程应用到ChromaDCLevel上来得到2x2的数组c。

$$c = \begin{bmatrix} ChromaDCLevel[iCbCr][0] & ChromaDCLevel[iCbCr][1] \\ ChromaDCLevel[iCbCr][2] & ChromaDCLevel[iCbCr][3] \end{bmatrix} \quad (8\text{-}299)$$

— 否则，如果chroma_format_idc为2，那么通过下面的方式将反光栅扫描过程应用到ChromaDCLevel上，从而得到2x4的数组c。

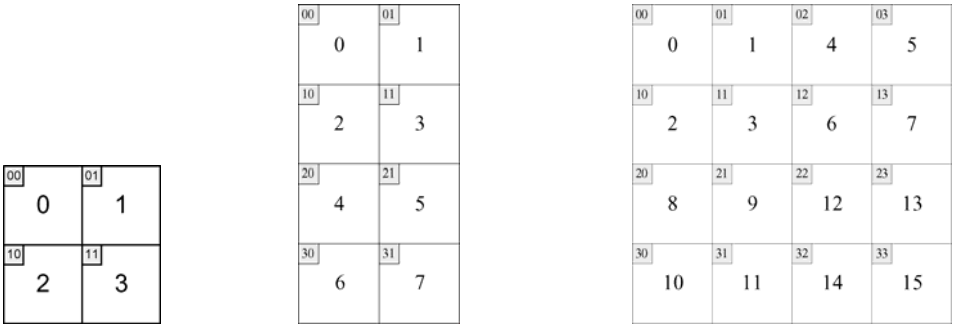
$$c = \begin{bmatrix} ChromaDCLevel[iCbCr][0] & ChromaDCLevel[iCbCr][2] \\ ChromaDCLevel[iCbCr][1] & ChromaDCLevel[iCbCr][5] \\ ChromaDCLevel[iCbCr][3] & ChromaDCLevel[iCbCr][6] \\ ChromaDCLevel[iCbCr][4] & ChromaDCLevel[iCbCr][7] \end{bmatrix} \quad (8\text{-}300)$$

— 否则（chroma_format_idc 为 3），调用第 8.5.5 节中定义的用于变换系数的反扫描过程，ChromaDCLevel[iCbCr]为输入，两维4x4数组c为输出。

b．调用第 8.5.9 节中定义的用于色度 DC 变换系数的缩放和变换过程，c 为输入，dcC 为输出。

2．为由 iCbCr 索引的分量的每个 4x4 色度块定义了如下的步骤，这些 4x4 色度块由 chroma4x4BlkIdx = 0..numChroma4x4Blks − 1 来索引。

a．获取具有 16 个条目的变量 chromaList。chromaList 的第一个条目是数组 dcC 中相应的值。图 8-7 给出了从数组 dcC 到 chroma4x4BlkIdx 的索引分配情况。小方块中的两个数字是指 dcC_{ij} 中的索引 i 和 j，大方块中的数字是指 chroma4x4BlkIdx。



a b c

图 8-7－从dcC到chroma4x4BlkIdx的索引分配情况：
(a) chroma_format_idc等于1, (b) chroma_format_idc等于 2, (c) chroma_format_idc等于3

在 chromaList 中索引为 k = 1..15 的元素可以表示为，

chromaList[k] = ChromaACLevel[chroma4x4BlkIdx][k − 1] (8-301)

b．调用第 8.5.9 节中描述的用于变换系数的反扫描过程，chromaList 为输入，2 维数组 c 为输出。

c．调用第 8.5.10 节中定义的用于残差 4x4 块的缩放和变换过程，c 为输入，r 为输出。

d．根据变量 chroma_format_idc，宏块中索引为 chroma4x4BlkIdx 的一个 4x4 色度块的左上角样点的位置通过下述方式得到。

— 如果chroma_format_idc为1或者2，那么采用下面的方式。

xO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (8-302)

yO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (8-303)

— 否则（chroma_format_idc为3），采用下面的方式。

xO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 0) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 0) (8-304)

yO =InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 1) (8-305)

e．当 residual_colour_transform_flag 为 1 时，将变量 xO'置为 xO % (4 << transform_size_8x8_flag)，变量 yO'置为 yO % (4 << transform_size_8x8_flag)，并且执行下述内容。

— 如果为色度分量Cb调用本过程，那么变量RCb,mn置为rij ，其中i, j = 0..3， m = xO' + i, n = yO' + j，而且在第8.5.13节中定义的残差色彩变换过程完成之前，暂停本过程，在这个残差色彩变换过程完成以后，将变量rij置为RB,mn，其中i, j = 0..3, m = xO' + i, n = yO' + j，并继续该过程。

— 否则（为色度分量Cr调用本过程），将变量RCr,mn置为rij ，其中i, j = 0..3， m = xO' + i, n = yO' + j，而且在第8.5.13节中定义的残差色彩变换过程完成之前，暂停本过程，在这个残差色彩变换过程完成以后，将变量rij置为RR,mn，其中i, j = 0..3, m = xO' + i, n = yO' + j，并继续该过程。

f．通过下面的方式得到具有元素 u_{ij} 的 4x4 数组，其中 i, j = 0..3，

$u_{ij} = Clip1_C(pred_C[xO + j, yO + i] + r_{ij})$ (8-306)

当 qpprime_y_zero_transform_bypass_flag 为 1，QP'_Y 为 0 时，比特流不应该包含会导致通过方程式 8-306 计算出来的 u_{ij} 的一个值不等于 $pred_C[xO + j, yO + i] + r_{ij}$ 的数据。

g．调用第 8.5.12 节中位于去块效应滤波过程前面的图像重建过程，chroma4x4BlkIdx 和 u 为输入。

8.5.5 用于变换系数的反扫描过程

本过程的输入是一个 16 个值的列表。

本过程的输出是一个变量 c，它包含一个 4x4 值的 2 维数组。在变换系数的情况中，这些 4x4 值代表变换块中分配给位置的幅值。在把反扫描过程应用到一个缩放列表的时候，输出变量 c 包含一个 2 维数组，该数组代表一个 4x4 缩放矩阵。

用于变换系数的反扫描过程把变换系数幅值的顺序映射到变换系数幅值位置。表 8-13 定义了两种映射方式：反 Z 型扫描以及反域扫描。反 Z 型应该用在帧宏块中的变换系数上，而反域扫描应该用在域宏块的变换系数上。

用于缩放列表的反扫描过程将缩放列表条目的顺序映射到相应的缩放矩阵中的位置。对于该映射，应该采用反 Z 型扫描。

图 8-8 给出了这个扫描的情况。

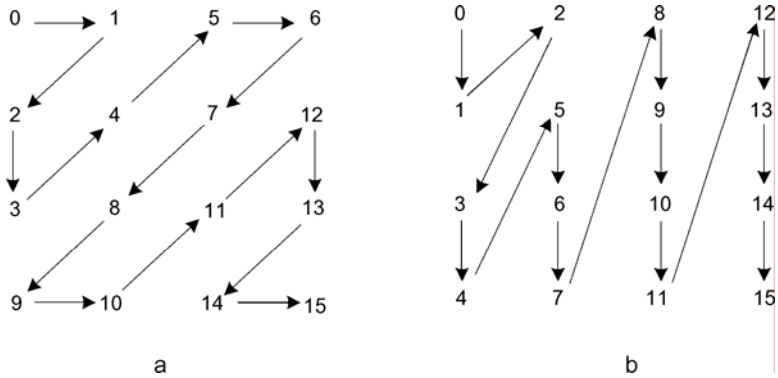


图 8-8－4x4块扫描 (a)Z型扫描(b)域扫描 (资料性)

表 8-13 提供了从 16 个元素的输入列表中的索引 idx 到 2 维数组 c 的索引 i 和 j 的映射。

表 8-13－对用在Z型以及域扫描的从idx到c_{ij}的映射的规范

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
zig-zag	c_{00}	c_{01}	c_{10}	c_{20}	c_{11}	c_{02}	c_{03}	c_{12}	c_{21}	c_{30}	c_{31}	c_{22}	c_{13}	c_{23}	c_{32}	c_{33}
field	c_{00}	c_{10}	c_{01}	c_{20}	c_{30}	c_{11}	c_{21}	c_{31}	c_{02}	c_{12}	c_{22}	c_{32}	c_{03}	c_{13}	c_{23}	c_{33}

8.5.6 8x8的亮度变换系数的反扫描过程

本过程的输入是一个 64 个值的列表。

本过程的输出是一个变量 c，它包含一个 2 维的 8x8 个值的数组。在变换系数情况下，这些 8x8 值代表分配给变换块中位置的幅值。如果把反扫描过程应用到一个缩放列表中，输出变量 c 包含一个 2 维数组，该数组代表一个 8x8 的缩放矩阵。

用于变换系数的反扫描过程将变换系数幅值的顺序映射到变换系数幅值位置。表 8-14 描述了两种映射方式：反 8x8Z 型扫描以及反 8x8 域扫描。反 8x8Z 型扫描应该用于帧宏块的变换系数，反 8x8 域扫描应该用在域宏块的变换系数上。

用于缩放列表的反扫描过程将缩放列表条目的顺序映射到缩放矩阵中相应的位置：对于这种映射方式，应该采用 Z 型扫描。

图 8-9 说明了这种扫描方式。

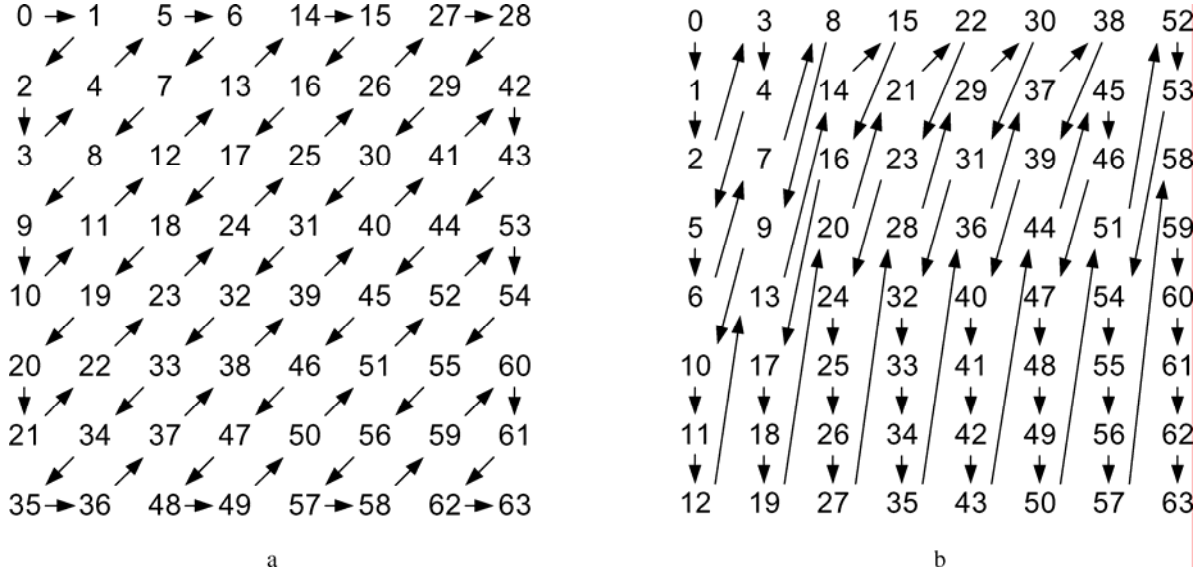


图 8-9－8x8块扫描(a) 8x8Z型扫描 (b) 8x8域扫描 (资料性)

表 8-14 提供了从 64 个元素的输入列表的索引 idx 到 2 维数组 c 的索引 i 和 j 的映射。

表 8-14－对用于8x8Z型和8x8域扫描的从idx到c_{ij}的映射的规范

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
zig-zag	c_{00}	c_{01}	c_{10}	c_{20}	c_{11}	c_{02}	c_{03}	c_{12}	c_{21}	c_{30}	c_{40}	c_{31}	c_{22}	c_{13}	c_{04}	c_{05}
域	c_{00}	c_{10}	c_{20}	c_{01}	c_{11}	c_{30}	c_{40}	c_{21}	c_{02}	c_{31}	c_{50}	c_{60}	c_{70}	c_{41}	c_{12}	c_{03}

表 8-14 (续) －对用于8x8Z型和8x8域扫描的从idx到c_{ij}的映射的规范

idx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
zig-zag	c_{14}	c_{23}	c_{32}	c_{41}	c_{50}	c_{60}	c_{51}	c_{42}	c_{33}	c_{24}	c_{15}	c_{06}	C_{07}	c_{16}	c_{25}	c_{34}
域	c_{22}	c_{51}	c_{61}	c_{71}	c_{32}	c_{13}	c_{04}	c_{23}	c_{42}	c_{52}	c_{62}	c_{72}	C_{33}	c_{14}	c_{05}	c_{24}

表 8-14 (续)－对用于8x8Z型和8x8域扫描的从idx到c_{ij}的映射的规范

idx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
zig-zag	c_{43}	c_{52}	c_{61}	c_{70}	c_{71}	c_{62}	c_{53}	c_{44}	c_{35}	c_{26}	c_{17}	c_{27}	C_{36}	c_{45}	c_{54}	c_{63}
域	c_{43}	c_{53}	c_{63}	c_{73}	c_{34}	c_{15}	c_{06}	c_{25}	c_{44}	c_{54}	c_{64}	c_{74}	C_{35}	c_{16}	c_{26}	c_{45}

表8-14 (结束的)－对用于8x8Z型和8x8域扫描的从idx到c_{ij}的映射的规范

idx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
zig-zag	c_{72}	c_{73}	c_{64}	c_{55}	c_{46}	c_{37}	c_{47}	c_{56}	c_{65}	c_{74}	c_{75}	c_{66}	c_{57}	c_{67}	c_{76}	c_{77}
域	c_{55}	c_{65}	c_{75}	c_{36}	c_{07}	c_{17}	c_{46}	c_{56}	c_{66}	c_{76}	c_{27}	c_{37}	c_{47}	c_{57}	c_{67}	c_{77}

8.5.7 色度量化参数和缩放功能的推导过程

本过程的输出是：

— QP_C：每个色度分量Cb和Cr的色度量化参数。

— QS_C：每个色度分量Cb和Cr的用于解码SP和SI条带（如果可用）的附加的色度量化参数。

注 1 — QP量化参数值QP_Y 和QS_Y总是在$QpBdOffset_Y$ 到 51这个范围内，包括$QpBdOffset_Y$和 51。QP量化参数值QP_C和QS_C总是在$QpBdOffset_C$ 到 51这个范围内，包括$QpBdOffset_C$ 和51。

一个色度分量的QP_C值由QP_Y当前的值以及 chroma_qp_index_offset (用于 Cb) 或者 second_chroma_qp_index_offset (用于 Cr)的值决定。

注 2 — 缩放方程式定义为，QP_Y每增加6，等价的变换系数幅值缩放因子翻倍。因此，QP_Y值每增加1，用于缩放的因子大概增加12%。

每个色度分量的 QP_C 值根据被表示为 qP_I 的索引来决定，参见表 8-15。

通过下述方式得到每个色度分量的变量 qP_{Offset}。

— 如果色度分量是Cb分量，那么qP_{Offset}被定义为

$$qP_{Offset} = chroma\_qp\_index\_offset \quad (8\text{-}307)$$

— 否则（色度分量是Cr分量），qP_{Offset}被定义为

$$qP_{Offset} = second\_chroma\_qp\_index\_offset \quad (8\text{-}308)$$

每个色度分量的 qP_I 值通过下述方式获得

$$qP_I = Clip3(-QpBdOffset_C, 51, QP_Y + qP_{Offset}) \quad (8\text{-}309)$$

通过下面的方式得到色度分量的 QP'_C 值

$$QP'_C = QP_C + QpBdOffset_C \tag{8-310}$$

通过下述的方式得到色度分量的 $BitDepth'_C$ 值

$$BitDepth'_C = BitDepth_C + residual\_colour\_transform\_flag \tag{8-311}$$

表 8-15—作为qP_I函数的QP_C的规范

qP_I	<30	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
QP_C	$=qP_I$	29	30	31	32	32	33	34	34	35	35	36	36	37	37	37	38	38	38	39	39	39	39

当前的条带是一个 SP 或者 SI 条带时，通过上面的过程可以得到 QSc，用 QS_Y 代替 QP_Y，QS_C 代替 QP_C。

函数 LevelScale(m, i, j)的定义如下。

— 4x4矩阵weightScale(i, j)定义如下。

  — 通过下述方式得到变量mbIsInterFlag。

    — 如果当前的宏块是采用帧间宏块预测模式来编码，将mbIsInterFlag置1。

    — 否则（当前的宏块通过帧内宏块预测模式编码），mbIsInterFlag置0。

  — 通过下述方式得到变量iYCbCr。

    — 如果输入数组c和一个亮度残差块相关，那么iYCbCr置0。

    — 否则，如果输入数组c和一个色度残差块相关，而且色度分量为Cb，那么iYCbCr置1。

    — 否则，（输入数组c和一个色度残差块相关，而且色度分量为Cr），那么iYCbCr为2。

  — 调用第 8.5.5 节中描述的用于变换系数的反扫描过程，ScalingList4x4[iYCbCr + ((mbIsInterFlag == 1) ? 3 : 0)]为输入，输出分配给4x4矩阵weightScale。

$$LevelScale(m, i, j) = weightScale(i, j) * normAdjust(m, i, j) \tag{8-312}$$

其中

$$normAdjust(m,i,j) = \begin{cases} v_{m0} & (i\%2, j\%2)\text{等于}(0,0), \\ v_{m1} & (i\%2, j\%2)\text{等于}(1,1), \\ v_{m2} & \text{其他情况} \end{cases} \tag{8-313}$$

V 的第一个和第二个下标分别代表行和列的索引，该矩阵的定义如下：

$$v = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 18 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 16 & 25 & 20 \\ 18 & 29 & 23 \end{bmatrix}. \tag{8-314}$$

函数 LevelScale8x8(m, i, j)的定义如下：

— 8x8的矩阵weightScale8x8(i, j)的定义如下。

— 通过下述方式得到变量mbIsInterFlag。

— 如果当前的宏块通过帧间宏块预测模式编码，将mbIsInterFlag置1。

— 否则（当前的宏块通过帧内宏块预测模式编码），将mbIsInterFlag置0。

— 调用第8.5.6节中描述的用于8x8亮度变换系数的反扫描过程，ScalingList8x8[mbIsInterFlag]为输入，输出分配给8x8矩阵weightScale8x8。

$$\text{LevelScale8x8}(m, i, j) = \text{weightScale8x8}(i, j) * \text{normAdjust8x8}(m, i, j) \tag{8-315}$$

其中

$$\text{normAdjust8}\times 8(\text{m,i,j}) = \begin{cases} v_{m0} & (i\%4, j\%2)\text{等于}(0,0), \\ v_{m1} & (i\%2, j\%2)\text{等于}(1,1), \\ v_{m2} & (i\%4, j\%4)\text{等于}(2,2), \\ v_{m3} & (i\%4, j\%2)\text{等于}(0,1)\text{或}(i\%2, j\%4)\text{等于}(1,0), \\ v_{m4} & (i\%4, j\%4)\text{等于}(0,2)\text{或}(i\%4, j\%4)\text{等于}(2,0), \\ v_{m5} & \text{其他情况} \end{cases} \tag{8-316}$$

其中 V 的两个下标分别代表行和列的索引，该矩阵的描述如下。

$$v = \begin{bmatrix} 20 & 18 & 32 & 19 & 25 & 24 \\ 22 & 19 & 35 & 21 & 28 & 26 \\ 26 & 23 & 42 & 24 & 33 & 31 \\ 28 & 25 & 45 & 26 & 35 & 33 \\ 32 & 28 & 51 & 30 & 40 & 38 \\ 36 & 32 & 58 & 34 & 46 & 43 \end{bmatrix}. \tag{8-317}$$

8.5.8 用于Intra_16x16宏块类型的亮度DC变换系数的缩放和变换过程

本过程的输入是用于 Intra_16x16 宏块的亮度 DC 变换系数的变换系数幅值，它是一个 4x4 的数组 c，该数组的元素为 c_{ij}，i 和 j 形成了一个 2 维的频率指数。

本过程的输出是 Intra_16x16 宏块的亮度 4x4 块的 16 级的 DC 值，这些值形成了一个 4x4 的数组 dcY，dcY 的元素为 dcY_{ij}。

根据 qpprime_y_zero_transform_bypass_flag 以及 QP'_Y 的值，执行下述内容。

— 如果 qpprime_y_zero_transform_bypass_flag 为 1，QP'_Y 为 0，那么通过 $dcY_{ij} = c_{ij}$ 来得到 dcY, 其中 i, j = 0..3。 (8-318)

— 否则（qpprime_y_zero_transform_bypass_flag为0，或者QP'_Y不为0），本过程接下来的内容定义了输出情况。

用于 4x4 亮度 DC 变换系数的反变换定义如下：

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{8-319}$$

比特流不应该包含会导致 f 的任何一个元素 f_{ij}（其中 i, j = 0..3）的值超出从$-2^{(7 + BitDepth_Y)}$ 到 $2^{(7 + BitDepth_Y)} -1$ 之间整数值范围的数据，这个范围包含$-2^{(7 + BitDepth_Y)}$ 和 $2^{(7 + BitDepth_Y)} -1$。

反变换完成以后，按照下述方式执行缩放功能。

— 如果QP'_Y大于等于36，那么缩放的结果如下

$$dcY_{ij} = (f_{ij} * LevelScale(QP'_Y \% 6, 0, 0)) << (QP'_Y/6 - 6),\ i,j = 0..3 \tag{8-320}$$

— 否则（QP'_Y小于36），缩放结果如下

$$dcY_{ij} = (f_{ij} * LevelScale(QP'_Y \% 6, 0, 0) + 2^{5 - QP'_Y/6}) >> (6 - QP'_Y/6),\ i,j = 0..3 \tag{8-321}$$

比特流不应该包含能够导致 dcY 的任何一个元素 dcY_{ij}（i, j = 0..3）超出$-2^{(7 + BitDepth_Y)}$ 到 $2^{(7 + BitDepth_Y)} -1$ 之间整数值范围的数据，这个范围包括$-2^{(7 + BitDepth_Y)}$ 和 $2^{(7 + BitDepth_Y)} -1$。

注 1 — 当熵编码模式标志为0，QP'_Y 小于10，简表标号为66，77或者88，那么可以被c的元素c_{ij}代表的值的范围不足以代表dcY的元素dcY_{ij}的整个范围的值，这些值对为任何可能的采用帧内16x16宏块类型的源图像的内容形成一个精确的近似值可能是必须的。

注 2 — 因为在对方程式8-321执行右移以后为dcY的dcY_{ij}加上了范围限制，因此在执行右移之前，解码器应该支持更大的数值范围。

8.5.9 用于色度DC变换系数的缩放和变换过程

本过程的输入是用于宏块的一个色度分量的色度 DC 变换系数的变换系数幅值值，这些值形成一个（MbWidthC / 4）x(MbHeightC / 4)数组 c，c 的元素为 c_{ij}，其中 i 和 j 形成了一个 2 维的频率指数。

本过程的输出是缩放的 DC 值，这些值形成一个(MbWidthC / 4)x(MbHeightC / 4)数组 dcC，它的元素为 dcC_{ij}。

根据 qpprime_y_zero_transform_bypass_flag 和 QP'_Y 的值来执行下面的步骤。

— 如果qpprime_y_zero_transform_bypass_flag为1， QP'Y为0，通过$dcC_{ij} = c_{ij}$ 来得到输出dcC，其中 i=0..(MbWidthC / 4) − 1，j = 0..(MbHeightC / 4) − 1。 (8-322)

— 否则（qpprime_y_zero_transform_bypass_flag为0或者QP'_Y不为0），通过本过程下面的内容来得到输出。

根据变量 chroma_format_idc 定义了下面的反变换。

— 如果chroma_format_idc为1，用于2x2色度DC变换系数的反变换的定义如下

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{8-323}$$

— 否则，如果 chroma_format_idc 为 2，用于 2x4 色度 DC 变换系数的反变换定义如下

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ c_{20} & c_{21} \\ c_{30} & c_{31} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{8-324}$$

— 否则（chroma_format_idc为3），用于4x4色度DC变换系数的反变换定义如下。

— 如果residual_colour_transform_flag为1而且当前的宏块预测模式MbPartPredMode(mb_type, 0)是Intra_4x4或者Intra_8x8，那么用于4x4色度DC变换系数的反变换定义为

$$f_{ij} = c_{ij} << 2,\ i, j = 0..3 \tag{8-325}$$

— 否则，用于4x4色度DC变换系数的反变换定义为

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{8-326}$$

比特流不应该包含能够导致 f 的任何一个元素 f_{ij} 超出从$-2^{(7+\mathrm{BitDepth'_C})}$到 $2^{(7+\mathrm{BitDepth'_C})}-1$ 之间整数值范围的数据，该范围包括$-2^{(7+\mathrm{BitDepth'_C})}$ 和 $2^{(7+\mathrm{BitDepth'_C})}-1$。

反变换过程完成以后，根据变量 chroma_format_idc 来执行缩放过程。

— 如果chroma_format_idc为1，缩放后的结果为

$$dcC_{ij} = ((f_{ij} * \mathrm{LevelScale}(QP'_C \,\%\, 6, 0, 0)) << (QP'_C / 6)) \;>>\; 5, \quad i,j = 0, 1 \tag{8-327}$$

— 如果chroma_format_idc为2，执行下面的内容。

— 通过下述方式得到变量$QP'_{C,DC}$。

$$QP'_{C,DC} = QP'_C + 3 \tag{8-328}$$

— 根据$QP'_{C,DC}$的值来执行下面的内容。

— 如果$QP'_{C,DC}$≥36，那么可以通过下式得到缩放后的结果。

$$dcC_{ij} = (f_{ij} * \mathrm{LevelScale}(QP'_{C,DC} \,\%6, 0, 0)) << (QP'_{C,DC} / 6 - 6), \; i = 0..3, \; j = 0, 1 \tag{8-329}$$

— 否则（$QP'_{C,DC}$＜36＝，通过下式得到缩放后的结果。

$$dcC_{ij} = (f_{ij} * \mathrm{LevelScale}(QP'_{C,DC} \,\%\, 6, 0, 0) + 2^{5-QP'_{C,DC}/6}) \;>>\; (6 - QP'_{C,DC} / 6), \; i = 0..3, j = 0,1 \tag{8-330}$$

— 否则（chroma_format_idc为3），执行下面的内容。

— 如果QP'_C≥36，那么缩放后的结果可以通过下式得到

$$dcC_{ij} = (f_{ij} * \mathrm{LevelScale}(QP'_C \,\%6, 0, 0)) << (QP'_C / 6 - 6), \quad i,j = 0..3. \tag{8-331}$$

— 否则（QP'_C＜36，通过下述方式得到缩放后的结果。

$$dcC_{ij} = (f_{ij} * \mathrm{LevelScale}(QP'_C \,\%\, 6, 0, 0) + 2^{5-QP'_C/6}) >> (6 - QP'_C / 6), i,j, = 0..3 \tag{8-332}$$

比特流不应该包含可以导致 dcC 的任何一个元素 dcC_{ij} 超出从$-2^{(7+\mathrm{BitDepth'_C})}$到 $2^{(7+\mathrm{BitDepth'_C})}-1$ 之间整数值范围的数据，这个范围包括$-2^{(7+\mathrm{BitDepth'_C})}$ 和 $2^{(7+\mathrm{BitDepth'_C})}-1$。

注 1 — 当熵编码模式标志为0，QP'_C小于4，简表标号为66，77或者88，那么可以被c的元素cij代表的值的范围不足以代表dcC的元素$dcCij_j$的整个范围的值，这些值对为任何可能源图像的内容形成一个精确的近似值可能是必须的。

注 2 — 因为在对方程式8-327、 8-330或者8-332执行右移以后为dcC的dcCij加上了范围限制，因此在执行右移之前，解码器应该支持更大的数值范围。

8.5.10 用于残差4x4块的缩放和变换过程

本过程的输入是一个具有元素 c_{ij} 的 4x4 数组 c，它可能是和亮度分量的一个残差块相关的数组或者是和一个色度分量的残差块相关的数组。

本过程的输出是残差样点值，它们构成一个 4x4 的数组 r，该数组的元素为 r_{ij}。

根据 qpprime_y_zero_transform_bypass_flag 以及 QP'_Y 的值来执行下面的内容。

— 如果qpprime_y_zero_transform_bypass_flag 为1，QP'_Y为0，输出可以表示为

$$r_{ij} = c_{ij}, \; i, j = 0..3 \tag{8-333}$$

— 否则（qpprime_y_zero_transform_bypass_flag为0或者QP'_Y不为0），接下来的内容决定了该过程的输出。

通过下述方式得到变量 bitDepth。

— 如果输入数组c和一个亮度残差块相关，将bitDepth置为$BitDepth_Y$。

— 否则（输入数组c和一个色度残差块相关），将bitDepth置为 $BitDepth'_C$。

比特流不应该包含可以导致 c 的任何一个元素 c_{ij}（i, j = 0..3）超出从$-2^{(7 + bitDepth)}$到 $2^{(7 + bitDepth)} - 1$ 之间整数值范围的数据，该范围包括$-2^{(7 + bitDepth)}$ 和 $2^{(7 + bitDepth)} - 1$。

通过下述方式得到变量 sMbFlag。

— 如果mb_type等于SI或者宏块预测模式为一个SP条带内的帧内模式，将sMbFlag置1。

— 否则（mb_type不等于SI同时宏块预测模式也是一个SP条带内的帧内模式），将sMbFlag置0。

通过下述方式得到变量 qP。

— 如果输入数组c和一个亮度残差块相关而且sMbFlag为0

$$qP = QP'_Y \tag{8-334}$$

— 否则，如果输入数组c和一个亮度残差块相关而且sMbFlag为1，

$$qP = QS_Y \tag{8-335}$$

— 否则，如果输入数组c和一个色度残差块相关而且sMbFlag为0

$$qP = QP'_C \tag{8-336}$$

— 否则（如果输入数组c和一个色度残差块相关而且sMbFlag为1）

$$qP = QS_C \tag{8-337}$$

对 4x4 块变换系数幅值 c_{ij} 的缩放处理如下。

— 如果下面所有的条件都为真

— i为0

— j为0

— c和一个用Intra_16x16预测模式编码的亮度残差块相关或者和一个色度残差块相关

通过下述方式得到变量 d_{00}

$$d_{00} = c_{00} \tag{8-338}$$

— 否则，执行下面的内容。

— 如果qP≥24，通过下述方式得到缩放后的结果。

$$d_{ij} = (c_{ij} * LevelScale(qP \% 6, i, j)) << (qP / 6 - 4), \ i,j = 0..3 \text{ 上述标注除外} \tag{8-339}$$

否则（qP＜24＝），通过下述方式可以得到缩放后的结果。

$$d_{ij} = (c_{ij} * LevelScale(qP \% 6, i, j) + 2^{3-qP/6}) >> (4 - qP / 6), \ \ i,j = 0..3 \text{ 上述标注除外} \tag{8-340}$$

比特流不应该包含可以导致 d 的任何一个元素 d_{ij}（i, j = 0..3）超出从$-2^{(7 + bitDepth)}$ 到 $2^{(7 + bitDepth)} - 1$ 之间整数值范围的数据，该范围包括$-2^{(7 + bitDepth)}$ 和 $2^{(7 + bitDepth)} - 1$。

变换过程应该通过在数学等价于下述内容的方式将缩放变换系数的块转换成为一个输出样点的块。

首先，通过如下所示的一个 1 维反变换将缩放变换系数的每行（水平的）进行变换。

一组中间值的计算方法如下。

$e_{i0} = d_{i0} + d_{i2},\ i = 0..3$ (8-341)

$e_{i1} = d_{i0} - d_{i2},\ i = 0..3$ (8-342)

$e_{i2} = (d_{i1} >> 1) - d_{i3},\ i = 0..3$ (8-343)

$e_{i3} = d_{i1} + (d_{i3} >> 1),\ i = 0..3$ (8-344)

比特流不应该包括会导致 e 的任何一个元素 e_{ij}（i, j = 0..3）超出从$-2^{(7 + bitDepth)}$到 $2^{(7 + bitDepth)} - 1$ 之间整数值范围的数据，这个范围包括$-2^{(7 + bitDepth)}$和 $2^{(7 + bitDepth)} - 1$。

然后，通过这些中间值来计算变换后的结果。

$f_{i0} = e_{i0} + e_{i3,}\ i = 0..3$ (8-345)

$f_{i1} = e_{i1} + e_{i2},\ i = 0..3$ (8-346)

$f_{i2} = e_{i1} - e_{i2},\ i = 0..3$ (8-347)

$f_{i3} = e_{i0} - e_{i3},\ i = 0..3$ (8-348)

比特流不应该包括会导致 f 的任何一个元素 f_{ij}（i, j = 0..3）超出从$-2^{(7 + bitDepth)}$到 $2^{(7 + bitDepth)} - 1$ 之间整数值范围的数据，这个范围包括$-2^{(7 + bitDepth)}$和 $2^{(7 + bitDepth)} - 1$。

然后，通过下述的 1 维反变换对得到的矩阵的每列（纵向）进行变换。

通过下述方式计算出一组中间值。

$g_{0j} = f_{0j} + f_{2j,}\ j = 0..3$ (8-349)

$g_{1j} = f_{0j} - f_{2j},\ j = 0..3$ (8-350)

$g_{2j} = (f_{1j} >> 1) - f_{3j},\ j = 0..3$ (8-351)

$g_{3j} = f_{1j} + (f_{3j} >> 1),\ j = 0..3$ (8-352)

比特流不应该包含可以导致 g 的任何一个元素 g_{ij} 超出从$-2^{(7 + bitDepth)}$ 到 $2^{(7 + bitDepth)} - 1$ 之间整数值范围的数据，该范围包括$-2^{(7 + bitDepth)}$和 $2^{(7 + bitDepth)} - 1$。

然后，通过这些中间值按照下述方式计算出变换的结果。

$h_{0j} = g_{0j} + g_{3j},\ j = 0..3$ (8-353)

$h_{1j} = g_{1j} + g_{2j},\ j = 0..3$ (8-354)

$h_{2j} = g_{1j} - g_{2j},\ j = 0..3$ (8-355)

$h_{3j} = g_{0j} - g_{3j},\ j = 0..3$ (8-356)

比特流不应该包含可以导致 h 的任何一个元素 h_{ij} 超出从$-2^{(7 + bitDepth)}$到 $2^{(7 + bitDepth)} - 33$ 之间整数值范围的数据，该范围包括$-2^{(7 + bitDepth)}$ 和 $2^{(7 + bitDepth)} - 33$。

在执行完 1 维横向以及 1 维纵向反变换从而产生一个变换样点的数组之后，通过下述方式可以得到最终重建的残差样点值。

$r_{ij} = (h_{ij} + 2^5) >> 6,\ i, j = 0..3$ (8-357)

8.5.11 用于残差8x8亮度块的缩放和变换过程

本过程的输入是一个具有元素 c_{ij} 的 8x8 的数组 c，这个数组和亮度分量的一个 8x8 的残差块相关。

本过程的输出是残差样点值，它构成一个元素为 r_{ij} 的 8x8 的数组 r。

根据 qpprime_y_zero_transform_bypass_flag 和 QP'_Y 的值来执行下面的内容。

— 如果qpprime_y_zero_transform_bypass_flag为1，QP'_Y为0，通过下述方式得到输出

$r_{ij} = c_{ij},\ i, j = 0..7$ (8-358)

— 否则（qpprime_y_zero_transform_bypass_flag为0或者QP'_Y不为0），本过程下面的内容定义了输出。

比特流不应该包含可以导致 c 的任何一个元素 c_{ij}（i, j = 0..7）超出了从$-2^{(7 + BitDepth_Y)}$到 $2^{(7 + BitDepth_Y)} - 1$ 之间整数值范围的数据，该范围包括$-2^{(7 + BitDepth_Y)}$和 $2^{(7 + BitDepth_Y)} - 1$。

用于 8x8 块变换系数幅值 c_{ij} 的缩放过程按照下面的方式进行。

— 如果$QP'_Y \geq 36$，缩放后的结果可以表示为

$d_{ij} = (c_{ij} * LevelScale8x8(QP'_Y \% 6, i, j)) << (QP'_Y / 6 - 6),\ i,j = 0..7$ (8-359)

— 否则（$QP'_Y < 36$=，缩放后的结果可以表示为

$d_{ij} = (c_{ij} * LevelScale8x8(QP'_Y \% 6, i, j)) + 2^{5 - QP'_Y/6}) >> (6 - QP'_Y /6),\ i,j = 0..7$ (8-360)

比特流不应该包含可以导致 d 的任何一个元素 d_{ij}（i, j = 0..7）超出从$-2^{(7 + BitDepth_Y)}$ 到 $2^{(7 + BitDepth_Y)} - 1$ 之间整数值范围的数据，该范围包括$-2^{(7 + BitDepth_Y)}$和 $2^{(7 + BitDepth_Y)} - 1$。

变换过程应该通过数学上等价于下述情况的方法将缩放变换系数的块转换成为一个输出样点的块。

首先，通过一个 1 为反变换方式变换缩放变换系数的每一行（横向）。

— 通过下述方式得到一组中间值

$e_{i0} = d_{i0} + d_{i4},\ i = 0..7$ (8-361)

$e_{i1} = - d_{i3} + d_{i5} - d_{i7} - (d_{i7} >> 1),\ i = 0..7$ (8-362)

$e_{i2} = d_{i0} - d_{i4},\ i = 0..7$ (8-363)

$e_{i3} = d_{i1} + d_{i7} - d_{i3} - (d_{i3} >> 1),\ i = 0..7$ (8-364)

$e_{i4} = (d_{i2} >> 1) - d_{i6},\ i = 0..7$ (8-365)

$e_{i5} = - d_{i1} + d_{i7} + d_{i5} + (d_{i5} >> 1),\ i = 0..7$ (8-366)

$e_{i6} = d_{i2} + (d_{i6} >> 1),\ i = 0..7$ (8-367)

$e_{i7} = d_{i3} + d_{i5} + d_{i1} + (d_{i1} >> 1),\ i = 0..7$ (8-368)

— 通过中间值e_{ij}计算出第二组中间结果f_{ij}，如下

$f_{i0} = e_{i0} + e_{i6},\ i = 0..7$ (8-369)

$f_{i1} = e_{i1} + (e_{i7} >> 2) ,\ i = 0..7$ (8-370)

$f_{i2} = e_{i2} + e_{i4},\ i = 0..7$ (8-371)

$f_{i3} = e_{i3} + (e_{i5} >> 2) ,\ i = 0..7$ (8-372)

$f_{i4} = e_{i2} - e_{i4},\ i = 0..7$ (8-373)

$f_{i5} = (e_{i3} >> 2) - e_{i5},\ i = 0..7$ (8-374)

$f_{i6} = e_{i0} - e_{i6},\ i = 0..7$ (8-375)

$f_{i7} = e_{i7} - (e_{i1} >> 2) ,\ i = 0..7$ (8-376)

— 然后，通过中间值f_{ij}可以计算出变换后的结果g_{ij}。

$g_{i0} = f_{i0} + f_{i7},\ i = 0..7$ (8-377)

$g_{i1} = f_{i2} + f_{i5},\ i = 0..7$ (8-378)

$g_{i2} = f_{i4} + f_{i3},\ i = 0..7$ (8-379)

$g_{i3} = f_{i6} + f_{i1},\ i = 0..7$ (8-380)

$g_{i4} = f_{i6} - f_{i1},\ i = 0..7$ (8-381)

$g_{i5} = f_{i4} - f_{i3},\ i = 0..7$ (8-382)

$g_{i6} = f_{i2} - f_{i5},\ i = 0..7$ (8-383)

$g_{i7} = f_{i0} - f_{i7},\ i = 0..7$ (8-384)

然后，通过同样的 1 维反变换方式对得到矩阵的每列（垂直方向）进行变换。

— 通过水平变换值g_{ij}计算出一组中间值h_{ij}。

$h_{0j} = g_{0j} + g_{4j},\ j = 0..7$ (8-385)

$h_{1j} = -g_{3j} + g_{5j} - g_{7j} - (g_{7j} >> 1),\ j = 0..7$ (8-386)

$h_{2j} = g_{0j} - g_{4j},\ j = 0..7$ (8-387)

$h_{3j} = g_{1j} + g_{7j} - g_{3j} - (g_{3j} >> 1),\ ,\ j = 0..7$ (8-388)

$h_{4j} = (g_{2j} >> 1) - g_{6j},\ j = 0..7$ (8-389)

$h_{5j} = -g_{1j} + g_{7j} + g_{5j} + (g_{5j} >> 1),\ j = 0..7$ (8-390)

$h_{6j} = g_{2j} + (g_{6j} >> 1),\ j = 0..7$ (8-391)

$h_{7j} = g_{3j} + g_{5j} + g_{1j} + (g_{1j} >> 1),\ j = 0..7$ (8-392)

通过中间值 h_{ij} 计算出第二组中间值 k_{ij}。

$k_{0j} = h_{0j} + h_{6j},\ j = 0..7$ (8-393)

$k_{1j} = h_{1j} + (h_{7j} >> 2),\ j = 0..7$ (8-394)

$k_{2j} = h_{2j} + h_{4j},\ j = 0..7$ (8-395)

$k_{3j} = h_{3j} + (h_{5j} >> 2),\ j = 0..7$ (8-396)

$k_{4j} = h_{2j} - h_{4j},\ j = 0..7$ (8-397)

$k_{5j} = (h_{3j} >> 2) - h_{5j},\ j = 0..7$ (8-398)

$k_{6j} = h_{0j} - h_{6j},\ j = 0..7$ (8-399)

$k_{7j} = h_{7j} - (h_{1j} >> 2),\ j = 0..7$ (8-400)

— 然后，通过这些中间值k_{ij}计算出变换后的结果m_{ij}。

$m_{0j} = k_{0j} + k_{7j},\ j = 0..7$ (8-401)

$m_{1j} = k_{2j} + k_{5j},\ j = 0..7$ (8-402)

$m_{2j} = k_{4j} + k_{3j},\ j = 0..7$ (8-403)

$m_{3j} = k_{6j} + k_{1j},\ j = 0..7$ (8-404)

$m_{4j} = k_{6j} - k_{1j},\ j = 0..7$ (8-405)

$m_{5j} = k_{4j} - k_{3j}, j = 0..7$ (8-406)

$m_{6j} = k_{2j} - k_{5j}, j = 0..7$ (8-407)

$m_{7j} = k_{0j} - k_{7j}, j = 0..7$ (8-408)

比特流不应该包含可以导致 e_{ij}, f_{ij}, g_{ij}, h_{ij}（i 和 j 在 0 到 7 之间，包括 0 和 7）中的任何一个元素超出从 $-2^{(7 + BitDepth_Y)}$ 到 $2^{(7 + BitDepth_Y)} - 1$ 之间整数值范围的数据，该范围包括$-2^{(7 + BitDepth_Y)}$和 $2^{(7 + BitDepth_Y)} - 1$。

比特流不应该包含可以导致任何一个元素 m_{ij}（i 和 j 在 0 到 7 范围内，包括 0 和 7）超出从$-2^{(7 + BitDepth_Y)}$到 $2^{(7 + BitDepth_Y)} - 33$ 之间整数值范围的数据，该范围包括$-2^{(7 + BitDepth_Y)}$和 $2^{(7 + BitDepth_Y)} - 33$。

通过执行 1 维的横向以及 1 维的纵向反变换从而生成一个变换后样点的数组后，通过下述方式得到最终重建的残差样点值。

$r_{ij} = (m_{ij} + 2^5) >> 6,\ i, j = 0..7$ (8-409)

8.5.12 去块效应滤波过程前面的图像重建过程

这个过程的输入是

— luma4x4BlkIdx或者chroma4x4BlkIdx或者luma8x8BlkIdx

— 一个元素是u_{ij}的样点数组u，它可以是一个4x4亮度块或者一个4x4色度块或者一个8x8亮度块。

通过调用第 6.4.1 节中描述的反宏块扫描过程（CurrMbAddr 为输入，输出分配给 (xP, yP)）来获得当前宏块左上角亮度样点的位置。

当 u 是一个亮度块时，对于亮度块的每个样点 u_{ij}，执行下述内容。

— 根据块u的尺寸来执行下述内容。

  — 如果u是一个4x4亮度块，通过调用第6.4.3节中描述的反4x4亮度块扫描过程（luma4x4BlkIdx为输入，输出分配给(xO, yO)，变量nE置为4）可以得到在宏块中的索引为luma4x4BlkIdx的4x4亮度块左上角样点的位置。

  — 否则（u是一个8x8亮度块），通过调用6.4.4中定义的反8x8亮度块扫描过程（luma8x8BlkIdx为输入，将输出分配给(xO, yO)，变量nE置为8）可以得到在宏块中的索引为luma8x8BlkIdx的8x8亮度块左上角样点的位置。

— 根据变量MbaffFrameFlag以及当前宏块，执行下述内容。

  — 如果MbaffFrameFlag为1，当前宏块为域宏块，那么

$S'_L[xP + xO + j, yP + 2 * (yO + i)] = u_{ij}, i, j = 0..nE - 1$ (8-410)

  — 否则（MbaffFrameFlag为0或者当前的宏块是帧宏块）

$S'_L[xP + xO + j, yP + yO + i] = u_{ij}, i, j = 0..nE - 1$ (8-411)

当 u 是一个色度块时，对于 4x4 色度块的每个样点 u_{ij}，执行下面的内容。

— Cb色度分量中变量S'_C中的下表C被Cb所代替，Cr色度分量中变量S'_C中的下表C被Cr所代替。

— 根据变量chroma_format_idc，通过下述方式可以得到一个4x4色度块左上角的位置，该色度块在宏块中的索引为chroma4x4BlkIdx。

  — 如果chroma_format_idc为1或者2，执行下面的内容。

xO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (8-412)

yO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (8-413)

— 否则（chroma_format_idc为3），执行下面的内容。

$$xO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 0) + InverseRasterScan(chroma4x4BlkIdx \% 4, 4, 4, 8, 0) \quad (8\text{-}414)$$

$$yO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) + InverseRasterScan(chroma4x4BlkIdx \% 4, 4, 4, 8, 1) \quad (8\text{-}415)$$

— 根据变量MbaffFrameFlag以及当前的宏块，执行下面的内容。

— 如果MbaffFrameFlag为1而且当前的宏块是一个域宏块，那么

$$S'_C[(xP / subWidthC) + xO + j, ((yP + SubHeightC - 1) / SubHeightC) + 2 * (yO + i)] = u_{ij}, \quad i, j = 0..3 \quad (8\text{-}416)$$

— 否则（MbaffFrameFlag为0或者当前的宏块是一个帧宏块），

$$S'_C[(xP/ subWidthC) + xO + j, (yP / SubHeightC) + yO + i] = u_{ij}, \ i, j = 0..3 \quad (8\text{-}417)$$

8.5.13 残差色彩变换过程

当 residual_colour_transform_flag 为 1 时，调用本过程。

调用该过程以后，在推导 $R_{Y,ij}$, $R_{Cb,ij}$, 和 $R_{Cr,ij}$ 没有完成之前暂时中止这个过程，其中 i, j = 0..ijMax，ijMax 的定义如下。

— 如果transform_size_8x8_flag为0，那么将变量ijMax置为3。

— 否则（transform_size_8x8_flag为1），那么将变量ijMax置为7。

在重新运行这个过程以后，在第 8.5.1、8.5.2.0, 或者 8.5.4 节中定义的相关过程开始运行之前，所有的值 $R_{Y,ij}$、$R_{Cb,ij}$ 以及 $R_{Cr,ij}$ 都应该是可用的，其中 i, j = 0..ijMax。

对于每个 i, j = 0..ijMax，通过下述方式计算残差色彩变换。

$$t = R_{Y,ij} - (R_{Cb,ij} >> 1) \quad (8\text{-}418)$$

$$R_{G,ij} = t + R_{Cb,ij} \quad (8\text{-}419)$$

$$R_{B,ij} = t - (R_{Cr,ij} >> 1) \quad (8\text{-}420)$$

$$R_{R,ij} = R_{B,ij} + R_{Cr,ij} \quad (8\text{-}421)$$

注 — 残差色彩变换和方程式E-30到E-33中定义的YCgCo变换相似。但是，残差色彩变换运行在解码过程的解码残差差别数据上，而不是作为一个后加工步骤来运行，该步骤不在本规范定义的解码过程内。

8.6 SP条带或者SI宏块中P宏块的解码过程

当解码一个 SP 条带中的 P 宏块类型或者一个 SI 条带中的 SI 宏块类型时，调用本过程。

本过程的输入是预测残差变换系数幅值以及当前宏块的预测样点。

本过程的输出是位于去块效应滤波过程之前的当前宏块的解码样点。

本小节为 SP 条带中的 P 宏块类型以及 SI 条带中的 SI 宏块类型定义了变换系数解码过程以及图像重建过程。

注 — SP条带利用帧间预测编码方式来使用序列中的临时冗余，它所采用的方法和P条带编码所采用的方法相似。但是，和P条带编码不同的是，即使是使用了完全不同的参考图像，SP条带编码也可以接受一个条带的完全相同的

重建。SI条带利用空间预测，它采用的方式和I条带采用的方式相似。SI条带编码和相应SP条带一样可以接受相同的重建。SP和SI条带的属性有助于为比特流变换、插接、随即访问、快速转发、快速反转以及差错恢复提供帮助。

一个 SP 条带包括按照 I 宏块类型或者 P 宏块类型编码的宏块。

一个 SI 条带包括按照 I 宏块类型或者 SI 宏块类型编码的宏块。

调用 8.5 中定义的位于去块效应滤波器过程前面的变换系数解码过程以及图像重建过程，这些过程用于 SI 条带中 I 宏块类型。通过下述方式解码 SI 宏块类型。

当当前宏块被编码为 P_Skip 时，将用于当前宏块的所有值 LumaLevel、ChromaDCLevel、ChromaACLevel 都置为 0。

8.6.1 用于非切换图像的SP解码过程

当解码 SP 条带（其中 sp_for_switch_flag 为 0）中的 P 宏块时，调用本过程。

本过程的输入是第 8.4 节中描述的当前宏块的帧间预测样点以及预测残差变换系数幅值。

本过程的输出是位于去块效应滤波过程之前的当前宏块的解码样点。

本节适用于 sp_for_switch_flag 为 0 的 SP 条带中的所有宏块，但是不适用于宏块预测模式等于 Intra_4x4 或者 Intra_16x16 的情况。本节不应用到 SI 条带上。

8.6.1.1 亮度变换系数解码过程

本过程的输入是第 8.4 节中定义的当前宏块 $pred_L$ 的帧间预测亮度样点、预测残差变换系数幅值、LumaLevel 以及 4x4 亮度块 luma4x4BlkIdx 的索引。

通过调用第 6.4.3 节中描述的反 4x4 亮度块扫描过程可以得到当前宏块中索引为 luma4x4BlkIdx 的 4x4 亮度块的左上角样点的位置，该过程的输入为 luma4x4BlkIdx，输出分配给(x, y)。

假设变量 p 为一个预测样点的 4x4 数组，通过如下方式得到元素 p_{ij}。

$$p_{ij} = pred_L[\, x + j, y + i \,], \; i, j = 0..3 \tag{8-422}$$

变换变量 p 从而根据下面的方式生成变换系数 c^p。

$$c^p = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \tag{8-423}$$

调用第 8.5.5 节中描述的反变换系数扫描过程，LumaLevel[luma4x4BlkIdx]为输入，元素是 c_{ij}^r 的 2 维数组 c 为输出。

通过量化参数 QP_Y 对预测残差变换系数 c^r 进行缩放，并且把它加入到预测块 c^p 的变换系数中，其中 i, j = 0..3。

$$c_{ij}^s = c_{ij}^p + (((c_{ij}^r * LevelScale(QP_Y \% 6, i, j) * A_{ij}) << (QP_Y / 6)) >> 10) \tag{8-424}$$

方程式 8-312 定义了 LevelScale(m, i, j)，A_{ij} 的定义如下：

$$A_{ij} = \begin{cases} 16 & (i,j) \in \{(0,0), (0,2), (2,0), (2,2)\}, \\ 25 & (i,j) \in \{(1,1), (1,3), (3,1), (3,3)\}, \\ 20 & \textbf{其他情况} \end{cases} \tag{8-425}$$

用在下面公式中的函数 LevelScale2(m, i, j)定义如下：

$$\text{LevelScale2}(m,i,j) = \begin{cases} w_{m0} & (i,j) \in \{(0,0),(0,2),(2,0),(2,2)\}, \\ w_{m1} & (i,j) \in \{(1,1),(1,3),(3,1),(3,3)\}, \\ w_{m2} & \text{其他情况} \end{cases} \tag{8-426}$$

w 的第一个和第二个下标分别代表矩阵的行和列的索引，这个矩阵的描述如下：

$$w = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix} \tag{8-427}$$

通过一个量化参数 QS_Y 对得到的总数 c^S 进行量化，其中 i, j = 0..3，如下所示。

$$c_{ij} = \text{Sign}(c_{ij}^s) * ((\text{Abs}(c_{ij}^s) * \text{LevelScale2}(QS_Y \% 6, i, j) + (1 << (14 + QS_Y / 6))) >> (15 + QS_Y / 6)) \tag{8-428}$$

调用第 8.5.10 节中定义的用于残差 4x4 块的缩放和变换过程，c 为输入，r 为输出。

通过下述方式得到元素为 u_{ij} 的 4x4 数组 u。

$$u_{ij} = \text{Clip1}_Y(r_{ij}), \ i, j = 0..3 \tag{8-429}$$

调用第 8.5.12 节中描述的位于去块效应滤波过程之前的图像重建过程，luma4x4BlkIdx 和 u 为输入。

8.6.1.2 色度变换系数解码过程

本过程的输入是 8.4 节中描述的当前宏块的帧间预测色度样点、预测残差变换系数幅值、ChromaDCLevel 和 ChromaACLevel。

本过程被调用两次：一次用于 Cb 分量，另外一次用于 Cr 分量。对于 Cb 分量，用 Cb 代替 C，对于 Cr 分量，用 Cr 代替 C。假设 iCbCr 选择了当前色度分量。

对于通过 chroma4x4BlkIdx 来索引的当前色度分量的每个 4x4 块（chroma4x4BlkIdx 等于 0..3），执行下述内容。

— 通过下述方式得到在宏块中索引为chroma4x4BlkIdx的一个4x4色度块左上角样点的位置。

$$x = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \tag{8-430}$$

$$y = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \tag{8-431}$$

— 假设p是一个4x4的预测样点的数组，通过下式得到元素p_{ij}。

$$p_{ij} = \text{pred}_C[x + j, y + i], \ i, j = 0..3 \tag{8-432}$$

— 通过方程式8-423对4x4数组p进行变换从而得到变换系数c^p(chroma4x4BlkIdx)。

— 得到一个具有16个条目的变量chromaList。chromaList[0]被置为0，索引k = 1..15的chromaList[k]的定义如下。

$$\text{chromaList}[k] = \text{ChromaACLevel}[\text{iCbCr}][\text{chroma4x4BlkIdx}][k - 1] \tag{8-433}$$

— 调用第8.5.5节中描述的反变换系数扫描过程，chromaList为输入，4x4的数组c为输出。

— 利用量化参数QP_C对预测残差变换系数进行缩放，并且添加到预测块c^p的变换系数中，其中i, j = 0..3，但是i = 0, j = 0的情况是个例外，具体操作如下。

$$c_{ij}^s = c_{ij}^p(chroma4x4BlkIdx) + (((c_{ij}^r * LevelScale(QP_C \% 6, i, j) * A_{ij}) << (QP_C / 6)) >> 10) \quad (8\text{-}434)$$

— 利用一个量化参数QS_C对通过前一个步骤得到的总数c^s进行量化，其中i, j = 0..3，对特例i = 0, j = 0的情况的处理方式如下。在本节接下来的部分描述了由此而得到的c_{00}(chroma4x4BlkIdx)。

$$c_{ij}(chroma4x4BlkIdx) = (Sign(c_{ij}^s) * (Abs(c_{ij}^s) * LevelScale2(QS_C \% 6, i, j) + (1 << (14 + QS_C / 6)))) >> (15 + QS_C / 6) \quad (8\text{-}435)$$

— 调用8.5.10中定义的用于残差4x4块的缩放以及变换过程，c(chroma4x4BlkIdx)为输入，r为输出。

— 通过下述方式可以得到元素为u_{ij}的4x4的数组u。

$$u_{ij} = Clip1_C(r_{ij}), \ i, j = 0..3 \quad (8\text{-}436)$$

— 调用第8.5.12节中位于去块效应滤波过程之前的图像重建过程，chroma4x4BlkIdx和u为输入。

DC 变换系数幅值 c_{00}(chroma4x4BlkIdx)的推导情况定义如下。将宏块的通用分量的 4 个预测色度 4x4 块的 DC 变换系数组装到一个 2x2 的矩阵中，该矩阵的元素为 c_{00}^p(chroma4x4BlkIdx)，并且把一个 2x2 变换应用到 DC 变换系数中，如下所示。

$$dc^p = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00}^p(0) & c_{00}^p(1) \\ c_{00}^p(2) & c_{00}^p(3) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8\text{-}437)$$

通过量化参数 QP 把色度 DC 预测残差变换系数幅值，k = 0..3 的 ChromaDCLevel[iCbCr][k]进行缩放，并且把它们加入到预测 DC 变换系数中，如下所示。

$$dc_{ij}^s = dc_{ij}^p + (((ChromaDCLevel[iCbCr][j * 2 + i] * LevelScale(QP_C \% 6, 0, 0) * A_{00}) << (QP_C / 6)) >> 9), \ i, j = 0, 1 \quad (8\text{-}438)$$

通过量化参数 QS_C 对 2x2 数组 dc^s 进行量化，如下所示。

$$dc_{ij}^r = (Sign(dc_{ij}^s) * (Abs(dc_{ij}^s) * LevelScale2(QS_C \% 6, 0, 0) + (1 << (15 + QS_C / 6)))) >> (16 + QS_C / 6), \ i, j = 0, 1 \quad (8\text{-}439)$$

元素为 f_{ij}，其中 i, j = 0..1，的 2x2 数组 f 的推导情况如下。

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} dc_{00}^r & dc_{01}^r \\ dc_{10}^r & dc_{11}^r \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8\text{-}440)$$

对 f 的元素 f_{ij} 的缩放如下所示。

$$c_{00}(j * 2 + i) = ((f_{ij} * LevelScale(QS_C \% 6, 0, 0)) << (QS_C / 6)) >> 5, \ i, \ j = 0, 1 \quad (8\text{-}441)$$

8.6.2 用于变换图像的SP和SI条带解码过程

当解码 SP 条带中的 P 宏块类型，此时 sp_for_switch_flag 为 1，以及解码 SI 条带中的 SI 宏块类型时调用本过程。

本过程的输入是预测残差变换系数幅值以及用于当前宏块的预测样点数组 $pred_L$, $pred_{Cb}$ 和 $pred_{Cr}$。

8.6.2.1 亮度变换系数解码过程

本过程的输入是预测亮度样点 $pred_L$ 以及亮度预测残差变换系数幅值 LumaLevel。

按照 8.6.1.1 中描述的方式推导出元素为 p_{ij} 的 4x4 数组 p，同时根据方程式 8-422 对该数组进行变换从而产生变换系数 c^p。然后利用量化参数 QS_Y 对这些变换系数进行量化，如下所示：

$$c_{ij}^s = Sign(c_{ij}^p) * ((Abs(c_{ij}^p) * LevelScale2(QS_Y \% 6, i, j) + (1 << (14 + QS_Y / 6))) >> (15 + QS_Y / 6)), \quad i, j = 0..3 \tag{8-442}$$

调用第 8.5.5 节中描述的反变换系数扫描过程，LumaLevel[luma4x4BlkIdx]为输入，元素是 c_{ij}^r 的 2 维数组 c^r 为输出。

通过下式推导出元素为 c_{ij} 的 4x4 数组 c，其中 i, j = 0..3。

$$c_{ij} = c_{ij}^r + c_{ij}^s, \ i, j = 0..3 \tag{8-443}$$

调用第 8.5.10 节中定义的用于残差 4x4 块的缩放和变换过程，c 为输入，r 为输出。

通过下式推导出元素为 u_{ij} 的 4x4 数组 u。

$$u_{ij} = Clip1_Y(r_{ij}), \ i, j = 0..3 \tag{8-444}$$

调用第 8.5.12 节中位于去块效应滤波过程之前的图像重建过程，luma4x4BlkIdx 和 u 为输入。

8.6.2.2 色度变换系数解码过程

本过程的输入是第 8.4 节中通用宏块的预测色度样点以及预测残差变换系数幅值、ChromaDCLevel 和 ChromaACLevel。

本过程被调用了两次：一次用于 Cb 分量，另外一次用于 Cr 分量。对于 Cb 分量，用 Cb 代替 C，对于 Cr 分量，用 Cr 代替 C。假设 iCbCr 选择的是当前色度分量。

对于利用 chroma4x4BlkIdx 来进行索引的当前色度分量的每个 4x4 块，chroma4x4BlkIdx 等于 0..3，执行下面的内容。

1．按照第 8.6.1.2 节中描述的方式推导出元素为 p_{ij} 的 4x4 数组 p，其中 i, j = 0..3，并且根据方程式 8-423 对该数组进行变换从而生成变换系数 c^p(chroma4x4BlkIdx)。然后利用量化参数 QS_C 对这些变换系数进行量化，其中 i, j = 0..3，但是 i = 0, j = 0 的组合是个例外，对于该组合的处理方式如下所示。在本节的接下来的内容中描述了对 c_{00}^p(chroma4x4BlkIdx)的处理。

$$c_{ij}^s = (Sign(c_{ij}^p(chroma4x4BlkIdx)) * (Abs(c_{ij}^p(chroma4x4BlkIdx)) * LevelScale2(QS_C \% 6, i, j) + (1 << (14 + QS_C / 6)))) >> (15 + QS_C / 6) \tag{8-445}$$

— 推导出具有16个条目的变量chromaList。将chromaList[0]置为0，索引k = 1..15的chromaList[k]的定义如下。

$$chromaList[k] = ChromaACLevel[iCbCr][chroma4x4BlkIdx][k - 1] \tag{8-446}$$

— 调用第8.5.5节中描述的反变换系数扫描过程，chromaList为输入，元素为c_{ij}^r(chroma4x4BlkIdx)的2维数组c^r(chroma4x4BlkIdx)为输出。

— 按照下述方式推导出元素为c_{ij}(chroma4x4BlkIdx)的4x4数组c(chroma4x4BlkIdx)，其中i, j = 0..3，但是i = 0, j = 0的组合情况除外，将在接下来的内容中描述c_{00}(chroma4x4BlkIdx)的推导过程。

$$c_{ij}(chroma4x4BlkIdx) = c_{ij}^r(chroma4x4BlkIdx) + c_{ij}^s \tag{8-447}$$

— 调用8.5.10节中定义的用于残差4x4块的缩放与变换过程，c(chroma4x4BlkIdx)为输入，r为输出。

— 元素为u_{ij}的4x4数组u的推导如下。

$$u_{ij} = Clip1_C(r_{ij}),\ i, j = 0..3 \tag{8-448}$$

— 调用8.5.12节中的位于去块效应滤波过程之前的图像重建过程，chroma4x4BlkIdx和u为输入。

DC 变换系数幅值 c_{00}(chroma4x4BlkIdx)的推导的定义如下。把宏块的当前分量的 4 个预测 4x4 色度块的 DC 变换系数 c_{00}^{p}(chroma4x4BlkIdx)组装倒一个 2x2 矩阵中，根据方程式 8-436 将一个 2x2 的变换应用到这些块的 DC 变换系数中从而得到 DC 变换系数 dc_{ij}^{p}。

然后通过量化参数 QS_C 对这些 DC 变换系数进行量化，如下所示：

$$dc_{ij}^{s} = (Sign(dc_{ij}^{p}) * (Abs(dc_{ij}^{p}) * LevelScale2(QS_C \% 6, 0, 0) + (1 << (15 + QS_C / 6)))) >> (16 + QS_C / 6),\ i, j = 0, 1 \tag{8-449}$$

将解析后的色度 DC 预测残差变换系数 ChromaDCLevel[iCbCr][k]加入到预测块的这些量化的 DC 变换系数中，其中 k = 0..3，如下所示：

$$dc_{ij}^{r} = dc_{ij}^{s} + ChromaDCLevel[iCbCr][j * 2 + i],\ i, j = 0, 1 \tag{8-450}$$

通过方程式 8-440 可以得到元素为 f_{ij} 的 2x2 的数组 f，其中 i, j = 0..1。

按照如下方式复制元素为 f_{ij} 的 2x2 的数组 f，其中 i,j=0…1。

$$c_{00}(j * 2 + i) = f_{ij},\ i, j = 0, 1 \tag{8-451}$$

8.7 去块效应滤波过程

应该将有条件的滤波应用到一幅图象的所有的 NxN（对于亮度来讲，N=4 或者 N=8；对于色度来讲，N=6）块边缘，但是图像边界的边缘以及任何被 disable_deblocking_filter_idc 终止了去块效应滤波过程的边缘例外，这些情况的定义如下。对于整个解码图像，在位于去块效应滤波过程（在 8.5 和 8.6 节中有定义）之前的图像重建过程完成以后，在宏块的基础上执行这个滤波过程，按照宏块地址增加的顺序为一幅图像中的所有的宏块执行滤波过程。

注 1 — 在每个宏块的去块效应滤波过程运行之前，因为是在位于去块效应滤波过程之前的图像重建过程完成以后才开始对整个解码图像执行去块效应滤波过程，所以宏块或者上述宏块对（如果存在任何一对）以及宏块或者位于当前宏块左边的宏块对（如果存在任何一对）的解锁样点通常都是可用的。但是，当disable_deblocking_filter_idc等于2的时候，为了决定需要滤波那个边缘，在去块效应滤波过程操作的步骤中，认为位于不同条带的宏块不可用。

分别为亮度和色度分量调用去块效应滤波过程。对于每个宏块以及每个分量，先滤波纵向的边缘，从宏块的左侧的边缘开始，按照几何学的顺序向宏块的右侧边缘处理，然后滤波横向边缘，从宏块的上部边缘开始，按照几何学的顺序向宏块的下部方向进行处理。图 8-10 给出了一个宏块的边缘，可以将该宏块的边缘解释为亮度或者色度的边缘。

当把图 8-10 中的边缘看作是亮度边缘时，根据 transform_size_8x8_flag 来执行下面的内容。

— 如果transform_size_8x8_flag为0，实粗线以及虚粗线的亮度边缘都要滤波。

— 否则（transform_size_8x8_flag为1），只滤波实粗线的亮度边缘。

当把图 8-10 中的边缘看作是色度边缘时，根据 chroma_format_idc 来执行下面的内容。

— 如果chroma_format_idc为1（4:2:0格式），只滤波实粗线色度边缘。

— 否则，如果chroma_format_idc为2（4:2:2格式），滤波实粗线垂直方向上的色度边缘以及实粗线和虚粗线水平方向上的色度边缘。

— 否则，如果 chroma_format_idc 为 3（4:4:4 格式），实粗线以及虚粗线色度边缘都滤波。

— 否则（chroma_format_idc 为 0（单色）），那么没有色度边缘需要滤波。

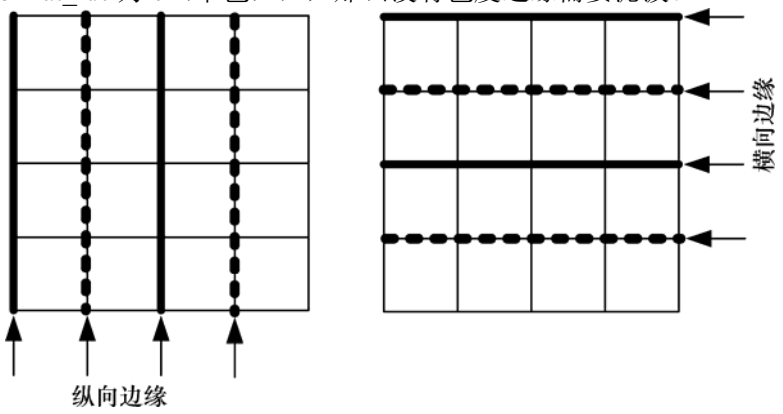


图 8-10－需要滤波的宏块边界

对于在值 0..PicSizeInMbs − 1 上处理的当前宏块地址 CurrMbAddr，执行下述内容。

1．调用 1.4.8.1 中定义的邻接宏块的推导过程，输出分配给 mbAddrA 和 mbAddrB。

2．通过下述方式得到变量 ieldModeMbFlag、filterInternalEdgesFlag、filterLeftMbEdgeFlag 以及 filterTopMbEdgeFlag。

— 变量fieldModeMbFlag的推导方式如下。

— 如果下面的任何一个条件都为真，那么把fieldModeMbFlag 置为1。

— field_pic_flag等于1。

— MbaffFrameFlag 等于1，而且宏块CurrMbAddr是一个域宏块。

— 否则，把fieldModeMbFlag置为0。

— 通过下述方式得到变量filterInternalEdgesFlag。

— 如果包含了宏块CurrMbAddr的条带的disable_deblocking_filter_idc为1，那么将变量filterInternalEdgesFlag置为0；

— 否则（包含了宏块CurrMbAddr的条带的disable_deblocking_filter_idc不等于1），将变量filterInternalEdgesFlag置为1。

— 通过下述方式得到变量filterLeftMbEdgeFlag。

— 如果下面的任何一个条件都为真，那么将变量filterLeftMbEdgeFlag置为0。

— MbaffFrameFlag等于0而且CurrMbAddr % PicWidthInMbs也为0。

— MbaffFrameFlag等于1，而且 (CurrMbAddr >> 1) % PicWidthInMbs等于0。

— 包含宏块CurrMbAddr的条带中的disable_deblocking_filter_idc等于1。

— 包含宏块CurrMbAddr的条带中的disable_deblocking_filter_idc等于2而且宏块mbAddrA不可用。

— 否则，将变量filterLeftMbEdgeFlag置为1。

— 通过下述方式得到变量filterTopMbEdgeFlag。

— 如果下面任何一个条件都为真，那么将变量filterTopMbEdgeFlag置为0。

— MbaffFrameFlag 等于0而且CurrMbAddr小于PicWidthInMbs。

— MbaffFrameFlag等于1，(CurrMbAddr >> 1)＜PicWidthInMbs，同时宏块 CurrMbAddr 是一个域宏块。

— MbaffFrameFlag等于1， (CurrMbAddr >> 1) ＜PicWidthInMbs，宏块CurrMbAddr是一个帧宏块，而且CurrMbAddr % 2等于0。

— 包含宏块CurrMbAddr的条带中的disable_deblocking_filter_idc等于1。

— 包含宏块CurrMbAddr的条带中的disable_deblocking_filter_idc等于2而且宏块mbAddrB不可用。

— 否则，将变量filterTopMbEdgeFlag置为1。

3．假定变量 fieldModeMbFlag、filterInternalEdgesFlag、filterLeftMbEdgeFlag 以及 filterTopMbEdgeFlag，按照下述方式控制去块效应滤波过程。

— 当filterLeftMbEdgeFlag等于1时，对左侧垂直亮度边缘的滤波情况如下。

— 调用8.7.1节中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 1、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (0, k)$（k = 0..15）作为输入，S'_L为输出。

— 当filterInternalEdgesFlag等于1时，对内部垂直亮度边缘滤波的情况如下所示。

— 当transform_size_8x8_flag等于0时，调用8.7.1中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 1、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (4, k)$（k = 0..15）为输入，S'_L为输出。

— 调用8.7.1节中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 1、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (8, k)$（k = 0..15）为输入，S'_L为输出。

— 当transform_size_8x8_flag等于0时，调用8.7.1中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 1、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (12, k)$（k = 0..15）为输入，S'_L为输出。

— 当filterTopMbEdgeFlag等于1时，滤波上部水平亮度边缘的情况如下所示。

— 如果MbaffFrameFlag等于1，(CurrMbAddr % 2) 等于0，CurrMbAddr ≥ 2 * PicWidthInMbs，宏块CurrMbAddr是一个帧宏块同时宏块(CurrMbAddr – 2 * PicWidthInMbs + 1)是一个域宏块，那么执行下面的内容。

— 调用8.7.1节中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 0、fieldModeFilteringFlag = 1以及$(xE_k, yE_k) = (k, 0)$（k = 0..15）为输入，S'_L为输出。

— 调用8.7.1节中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 0、fieldModeFilteringFlag = 1以及$(xE_k, yE_k) = (k, 1)$（k = 0..15）为输入，S'_L为输出。

— 否则，调用8.7.1节中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 0、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (k, 0)$（k = 0..15）为输入，S'_L为输出。

— 当filterInternalEdgesFlag等于1时，对内部横向亮度边缘的滤波情况定义如下。

— 当transform_size_8x8_flag等于0时，调用8.7.1节中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 0、fieldModeFilteringFlag = fieldModeMbFlag以及 $(xE_k, yE_k) = (k, 4)$（k = 0..15）为输入，S'_L为输出。

— 调用8.7.1节中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 0、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (k, 8)$（k = 0..15）为输入，S'_L为输出。

— 当transform_size_8x8_flag等于0时，调用8.7.1节中定义的过程，其中chromaEdgeFlag = 0、verticalEdgeFlag = 0 、 fieldModeFilteringFlag = fieldModeMbFlag 以 及 $(xE_k, yE_k) = (k, 12)$（k = 0..15）为输入，S'_L为输出。

— 对两种色度分量的滤波，Cb中iCbCr = 0，Cr中iCbCr = 1，执行下述内容。

— 当filterLeftMbEdgeFlag等于1时，对左侧垂直色度边缘滤波的情况定义如下。

— 调用8.7.1节中定义的过程，把chromaEdgeFlag = 1 、 iCbCr 、 verticalEdgeFlag = 1 、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (0, k)$（k = 0..MbHeightC − 1）作为输入；S'_C为输出，其中对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 当filterInternalEdgesFlag等于1时，对于内部垂直色度边缘的滤波情况定义如下。

— 调用8.7.1节中定义的过程， chromaEdgeFlag = 1 、 iCbCr 、 verticalEdgeFlag = 1 、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (4, k)$（k = 0..MbHeightC − 1）作为输入；S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 当chroma_format_idc等于3时，调用8.7.1节中定义的过程，把chromaEdgeFlag = 1、iCbCr、verticalEdgeFlag = 1、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (8, k)$（k = 0..MbHeightC − 1）作为输入，S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 当chroma_format_idc等于3时，调用8.7.1节中定义的过程，把chromaEdgeFlag = 1、iCbCr 、 verticalEdgeFlag = 1 、 fieldModeFilteringFlag = fieldModeMbFlag 以 及 $(xE_k, yE_k) = (12, k)$（k = 0..MbHeightC − 1）作为输入，S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 当filterTopMbEdgeFlag等于1时，对上部横向色度边缘滤波情况的定义如下。

— 如果MbaffFrameFlag等于1，(CurrMbAddr % 2) 等0，CurrMbAddr ≥ 2 * PicWidthInMbs ，宏块CurrMbAddr是一个帧宏块同时宏块(CurrMbAddr − 2 * PicWidthInMbs + 1)是一个域宏块，那么执行下述内容。

— 调用8.7.1节中定义的过程，其中chromaEdgeFlag = 1 、 iCbCr 、 verticalEdgeFlag = 0、fieldModeFilteringFlag = 1以及 $(xE_k, yE_k) = (k, 0)$（k = 0..MbWidthC − 1）为输入，S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 调用8.7.1节中定义的过程，其中chromaEdgeFlag = 1 、 iCbCr 、 verticalEdgeFlag = 0 、 fieldModeFilteringFlag = 1以及 $(xE_k, yE_k) = (k, 1)$（k = 0..MbWidthC − 1）为输入，S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 否则，调用8.7.1节中定义的过程，chromaEdgeFlag = 1、iCbCr、verticalEdgeFlag = 0、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (k, 0)$（k = 0..MbWidthC − 1）为输入，S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 当filterInternalEdgesFlag等于1时，对内部横向色度边缘的滤波情况的定义如下。

— 调用8.7.1节中定义的过程， chromaEdgeFlag = 1 、 iCbCr 、 verticalEdgeFlag = 0 、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (k, 4)$（k = 0..MbWidthC − 1）为输入，S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 当chroma_format_idc不等于1时，调用8.7.1节中定义的过程，chromaEdgeFlag = 1、iCbCr、verticalEdgeFlag = 0、fieldModeFilteringFlag = fieldModeMbFlag以及$(xE_k, yE_k) = (k, 8)$

（k = 0..MbWidthC − 1）为输入，S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

— 当chroma_format_idc不等于1时，调用8.7.1节中定义的过程，chromaEdgeFlag = 1、iCbCr、verticalEdgeFlag = 0、fieldModeFilteringFlag = fieldModeMbFlag以及 (xE_k, yE_k) = (k, 12)（k = 0..MbWidthC − 1）为输入，S'_C为输出，对于iCbCr = 0的情况，用Cb代替C；对于iCbCr = 1的情况，用Cr代替C。

注 2 — 如果在一个帧宏块的顶部横向边缘内应用域模式的滤波（fieldModeFilteringFlag等于1），那么在顶部和底部宏块边界上应用纵向滤波可能会涉及到一些超越了内部块边缘的样点也同样采用帧模式进行内部地滤波。

注 3 — 例如，在4:2:0色度格式下，如果transform_size_8x8_flag等于0，那么应用下述内容。3个横向亮度边缘，用于Cb的1个纵向色度边缘，用于Cr的一个横向色度边缘采用对于一个宏块来讲属于内部的滤波方式。如果将域模式的滤波（fieldModeFilteringFlag等于1）应用到一个帧宏块的顶部边缘，那么在帧宏块以及上述宏块对之间的2个横向亮度，Cb的2个横向色度边缘，Cr的2个横向色度边缘采用域模式的滤波。认为对于全部5个横向亮度边缘，用于Cb的3个横向色度边缘，用于Cr的3个横向色度边缘需要在帧宏块的控制之下进行滤波。在其他所有的情况中，认为最多有4个横向亮度，用于Cb的2个横向色度边缘，用于Cr的2个横向色度边缘需要在一个特定的宏块控制之下进行滤波。

最后，分别将数组 S'_L, S'_{Cb}, S'_{Cr} 分配给数组 S_L, S_{Cb}, S_{Cr}（这些数组代表解码后的图像）。

8.7.1 用于块边缘的滤波过程

本过程的输入是 chromaEdgeFlag、色度分量索引 iCbCr (此时 chromaEdgeFlag 等于 1)、verticalEdgeFlag、fieldModeFilteringFlag 以及一组 nE 样点相对于宏块 CurrMbAddr 左上角的位置 (xE_k, yE_k)（ k = 0..nE – 1）。样点位置（xE_k, yE_k）组代表了样点在一个纵向边缘的右侧位置（此时，verticalEdgeFlag 等于 1），或者在一个横向边缘下面（此时，verticalEdgeFlag 等于 0）的位置。

通过下述方式得到变量 nE。

— 如果chromaEdgeFlag等于0，那么将nE置为16。

— 否则（chromaEdgeFlag等于1），将nE置为MbHeightC : MbWidthC （ verticalEdgeFlag == 1 。）

通过如下方式对一个定义了一个亮度或者色度样点数组的变量进行推导。

— 如果chromaEdgeFlag等于0，s'代表一个当前图像的亮度样点数组S'_L。

— 否则，如果chromaEdgeFlag等于1，iCbCr等于0，那么s'代表当前图像的色度分量Cb的色度样点数组S'_{Cb}。

— 否则（chromaEdgeFlag 等于1，iCbCr等于1），s’代表当前图像的色度分量Cr的色度样点数组S'_{Cr}。

通过下述方式得到变量 dy。

— 如果fieldModeFilteringFlag等于1，MbaffFrameFlag等于1，将dy置为2。

— 否则（fieldModeFilteringFlag等于0，MbaffFrameFlag等于0），将dy置为1。

通过调用 6.4.1 节中描述的反宏块扫描过程可以推导出宏块 CurrMbAddr 左上角亮度样点的位置，该过程把 mbAddr = CurrMbAddr 作为输入，将输出分配给(xI, yI)。

通过下述方式得到变量 xP 和 yP。

— 如果chromaEdgeFlag等于0，将xP置为xI ，yP置为yI。

— 否则（chromaEdgeFlag等于1），将xP置为xI / SubWidthC ，yP置为(yI + SubHeightC−1) / SubHeightC。

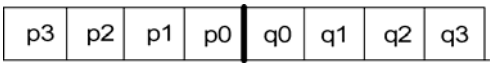


图 8-11－描述一个4x4块横向或者纵向边界的样点的惯例

对于每个样点位置(xE_k, yE_k)（k = 0 .. nE − 1），执行下述内容。

— 对一个4x4块的横向或者纵向边缘的一个8个样点组进行滤波，这8个样点可以表示为p_i和q_i（i = 0..3），如图 8-11所示。边缘位于p_0和q_0之间，对p_i和q_i（i = 0..3）的定义如下所示。

— 如果verticalEdgeFlag等于1

$$q_i = s'[\ xP + xE_k + i,\ yP + dy * yE_k\ ] \quad (8\text{-}452)$$

$$p_i = s'[\ xP + xE_k - i - 1,\ yP + dy * yE_k\ ] \quad (8\text{-}453)$$

否则(verticalEdgeFlag 等于 0),

$$q_i = s'[\ xP + xE_k,\ yP + dy * (\ yE_k + i\ ) - (yE_k \% 2\ )\ ] \quad (8\text{-}454)$$

$$p_i = s'[\ xP + xE_k,\ yP + dy * (\ yE_k - i - 1\ ) - (yE_k \% 2\ )\ ] \quad (8\text{-}455)$$

— 调用8.7.2节中定义的过程，把样点值p_i和q_i (i = 0..3)、chromaEdgeFlag、verticalEdgeFlag以及fieldModeFilteringFlag作为输入，把输出分配给滤波后的结果样点值p'_i和q'_i（i = 0..2）。

— 在样点数组中，用相应的滤波后得到的样点值p'_i和q'_i（i = 0..2）来代替输入的样点值p_i和q_i (i = 0..2)，如下所示。

— 如果 verticalEdgeFlag 等于1,

$$s'[\ xP + xE_k + i,\ yP + dy * yE_k\ ] = q'_i \quad (8\text{-}456)$$

$$s'[\ xP + xE_k - i - 1,\ yP + dy * yE_k\ ] = p'_i \quad (8\text{-}457)$$

— 否则 (verticalEdgeFlag等于0),

$$s'[\ xP + xE_k,\ yP + dy * (\ yE_k + i\ ) - (\ yE_k \% 2\ )\ ] = q'_i \quad (8\text{-}458)$$

$$s'[\ xP + xE_k,\ yP + dy * (\ yE_k - i - 1\ ) - (\ yE_k \% 2\ )\ ] = p'_i \quad (8\text{-}459)$$

8.7.2 用于一个横向或者纵向块边缘的一组样点的滤波过程

本过程的输入是将要被滤波的一个边缘上的一组样点的输入样点值 p_i 和 q_i（i=0..3）、chromaEdgeFlag、verticalEdgeFlag 以及 fieldModeFilteringFlag。

本过程的输出是滤波后得到的样点值 p'_i 和 q'_i（i = 0..2）。

通过如下方式得到依赖内容的边界滤波强度变量 bS。

— 如果chromaEdgeFlag等于0，那么调用8.7.2.1节中定义的用于依赖内容的边界滤波强度的推导过程，其中p_0、q_0以及verticalEdgeFlag为输入，输出分配给bS。

— 否则（chromaEdgeFlag等于1），应该将用于滤波一个横向或者纵向色度边缘的一组样点的bS值分别置为滤波一个横向或者纵向亮度边缘的bS值，该bS在同一个域的亮度数组内的(SubWidthC * x, SubHeightC * y)

位置包含了亮度样点，其中(x, y)是同一个域内色度数组中色度样点q_0的位置。

调用 8.7.2.2 节中定义的过程，p_0、q_0、p_1、q_1、chromaEdgeFlag 和 bS 为输入，将输出分配给 filterSamplesFlag、indexA、α和β。

根据变量 filterSamplesFlag 的情况来执行下面的内容。

— 如果filterSamplesFlag等于1，执行下述内容。

  — 如果bS＜4，调用8.7.2.3节中定义的过程，p_i和q_i (i = 0..2)、chromaEdgeFlag、bS、β和indexA为输入，将输出分配给p'_i和q'_i (i = 0..2)。

  — 否则（bS等于4），调用8.7.2.4节中定义的过程，p_i和q_i (i = 0..3)、chromaEdgeFlag、 α和β为输入，把输出分配给p'_i和q'_i (i = 0..2)。

— 否则（filterSamplesFlag等于0），用相应的输入样点p_i和q_i代替滤波后得到样点值p'_i和q'_i (i = 0..2)。

对于 $i = 0..2, p'_i = p_i$ (8-460)

对于 $i = 0..2, q'_i = q_i$ (8-461)

8.7.2.1 亮度的依赖内容的边界滤波强度的推导过程

本过程的输入是一个将要被滤波的边缘的一组样点的输入样点值 p_0 和 q_0 以及 verticalEdgeFlag。

本过程的输出是变量 bS。

通过如下方式可以得到变量 mixedModeEdgeFlag。

— 如果MbaffFrameFlag等于1，样点p_0和q_0位于不同的宏块对中，其中一个是域宏块对，另外一个是帧宏块对，将mixedModeEdgeFlag置为1。

— 否则，将mixedModeEdgeFlag置0。

通过下述方式得到变量 bS。

— 如果块边缘也是一个宏块边缘而且下面的任何一个条件都为真，值为4的bS应该为输出：

  — 样点p_0和q_0都在帧宏块内，而且两个样点p_0和q_0之中的两个或者一个在采用帧内宏块预测模式编码的宏块内。

  — 样点p_0和q_0都在帧宏块内，而且两个样点p_0和q_0之中的两个或者一个都在slice_type等于SP或者SI的条带内。

  — MbaffFrameFlag等于1或者field_pic_flag等于1，verticalEdgeFlag等于1同时p_0或者q_0之中的两个或者一个在通过一个内部宏块预测模式编码的宏块中。

  — MbaffFrameFlag等于1或者field_pic_flag等于1，verticalEdgeFlag等于1同时两个样点p_0和q_0之中的两个或者一个在slice_type等于SP或者SI的条带的一个宏块中。

— 否则，如果下面的任何一个条件为真，等于3的bS值应该为输出。

  — mixedModeEdgeFlag等于0，两个样点p_0或者q_0之中的一个或者两个都在利用帧内宏块预测模式编码的宏块中。

  — mixedModeEdgeFlag等于0，两个样点p_0或者q_0之中的一个或者两个都在slice_type等于SP或者SI的条带的一个宏块中。

  — mixedModeEdgeFlag等于1，verticalEdgeFlag等于0，两个样点p_0或者q_0之中的一个或者两个都在利用帧内宏块预测模式编码的宏块中。

  — mixedModeEdgeFlag等于1，verticalEdgeFlag等于0同时两个样点p_0和q_0之中的两个或者一个在slice_type等于SP或者SI的条带的一个宏块中。

— 否则，如果下面的条件为真，等于2的bS应该为输出：

  — 包含样点p_0的亮度块或者包含样点q_0的亮度块包含非零的变换系数幅值。

— 否则，如果下面的任何一个条件为真，等于1的bS值应该为输出：

  — mixedModeEdgeFlag等于1

  — mixedModeEdgeFlag等于0，而且是对包含样点p_0的宏块/子宏块而不是对包含样点q_0的宏块/子宏块采用不同的参考图像或者不同数量的运动矢量。

  注 1 — 两个宏块/子宏块采用的参考图像是相同还是不同只是根据被参考图像而定，而无需考虑是否通过将一个索引放置到参考图像列表0中或者将索引放置到参考图像列表1中来形成一个预测，同时也无需考虑在一个参考图像列表内索引的位置相同与否。

  — mixedModeEdgeFlag等于0，一个运动矢量用来预测包含样点p_0的宏块/子宏块，一个运动矢量用来预测包含样点q_0的宏块/子宏块，运动矢量采用的横向或者纵向分量之间的绝对差别≥4（它的单位是1/4亮度帧样点）

  — mixedModeEdgeFlag等于0，利用两个运动矢量以及两个不同的参考图像来预测包含样点p_0的宏块/子宏块，利用用于上述两个参考图像的两个运动矢量来预测包含样点q_0的宏块/子宏块，用于预测用在相同的参考图像上的两个宏块/子宏块的两个运动矢量的横向或者纵向分量之间的决对差别≥4（它的单位是1/4亮度帧样点）。

  — mixedModeEdgeFlag等于0，用于同一个参考图像的两个运动矢量用来预测包含样点p_0的宏块/子宏块，用于同一个参考图像的两个运动矢量用来预测包含样点q_0的宏块/子宏块，而且下面的两个条件都为真：

    — 用于预测两个宏块/子宏块的运动矢量列表0中横向和纵向分量的绝对差别≥4（它的单位是1/4亮度帧样点）或者用于预测两个宏块/子宏块的运动矢量列表1中横向和纵向分量的绝对差别≥4（它的单位是1/4亮度帧样点）。

    — 用于预测包括样点p_0的宏块/子宏块的运动矢量列表0的横向和纵向分量之间的绝对差别以及用于预测包括样点q_0的宏块/子宏块的运动矢量列表1的横向和纵向分量之间的绝对差别≥4（它的单位是1/4亮度帧样点），或者用于预测包含样点p_0的宏块/子宏块的运动矢量列表1的横向或者纵向分量之间的绝对差别以及用于预测包含样点q_0的宏块/子宏块的运动矢量列表0的横向或者纵向分量之间的绝对差别≥4（它的单位是1/4亮度帧样点）。

    注 2 — 一个单位为1/4亮度帧样点的4个纵向差别是单位为1/4亮度域样点的2个差别。

— 否则，值为0的bS应该为输出。

8.7.2.2 每个块边缘的门限的推导过程

本过程的输入是将要被滤波的一个边缘的一组样点的输入样点值 p_0、q_0、p_1 和 q_1、chromaEdgeFlag 以及 bS，对于这组输入样点，参见 8.7.2 中的定义。

本过程的输出是变量 filterSamplesFlag（它是指是否滤波了输入样点），indexA 的值以及门限变量α和β的值。

假设 qP_p 和 qP_q 是定义了用于宏块的量化参数的变量，这些宏块分别包括了样点 p_0 和 q_0。通过下述方式得到变量 qP_z（分别用 p 或者 q 代替 z）。

— 如果chromaEdgeFlag等于0，执行下面的内容。

  — 如果宏块包括的样点z_0是一个I_PCM宏块，将qP_z置0。

  — 否则（宏块包括的样点z_0不是一个I_PCM宏块），将qP_z设置为包括样点z_0的宏块的QP_Y的值。

— 否则（chromaEdgeFlag等于1），执行下面的内容。

— 如果包含样点z_0的宏块是一个I_PCM宏块，那么将qP_z置为对应QP_Y的值为0的QP_C值，参见第8.5.7节的定义。

— 否则（宏块包括的样点z_0不是一个I_PCM宏块），将qP_z置为对应包含样点z_0的宏块的QP_Y值的QP_C值，参见8.5.7节中的定义。

令 qP_{av} 是一个定义了一个平均量化参数的变量，通过下述方式可以得到它。

$$qP_{av} = (qP_p + qP_q + 1) >> 1 \quad (8\text{-}462)$$

注 — 在SP和SI条带中，得到qP_{av}值的方式和在其它条带类型中采用的方式相同。来自方程式7-28的QS_Y没有用在去块效应滤波中。

令 indexA 是一个用于访问α 表 (表 8-16)以及 t_{C0} 表 (表 8-17)的变量，这个表格用于边缘滤波，同时 bS＜4，参见 8.7.2.3 节中的定义，假设 indexB 是用于访问β表(表 8-16)的变量。变量 indexA 和 indexB 的推导方式如下所示，其中 FilterOffsetA 和 FilterOffsetB 的值是在 7.4.3 中定义的那些用于包含样点 q_0 的宏块的条带的变量的值。

$$indexA = Clip3(0, 51, qP_{av} + FilterOffsetA) \quad (8\text{-}463)$$

$$indexB = Clip3(0, 51, qP_{av} + FilterOffsetB) \quad (8\text{-}464)$$

表 8-16 定义了依赖 indexA 和 indexB 的值的变量α'和β'。根据 chromaEdgeFlag，通过下述方式可以得到相应的门限变量α和β。

— 如果chromaEdgeFlag等于0,，

$$\alpha = \alpha' * (1 << (BitDepth_Y - 8)) \quad (8\text{-}465)$$

$$\beta = \beta' * (1 << (BitDepth_Y - 8)) \quad (8\text{-}466)$$

— 否则(chromaEdgeFlag等于1),，

$$\alpha = \alpha' * (1 << (BitDepth_C - 8)) \quad (8\text{-}467)$$

$$\beta = \beta' * (1 << (BitDepth_C - 8)) \quad (8\text{-}468)$$

变量 filterSamplesFlag 的推导方式如下

$$filterSamplesFlag = (bS != 0 \ \&\& \ Abs(p_0 - q_0) < \alpha \ \&\& \ Abs(p_1 - p_0) < \beta \ \&\& \ Abs(q_1 - q_0) < \beta) \quad (8\text{-}469)$$

表 8-16－来自indexA和indexB的依赖偏移量的门限变量α'和β'的推导

	indexA (α')或 indexB (β')																								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
α'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	5	6	7	8	9	10	12	13
β'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	3	3	3	3	4	4	4

表 8-16 (结束)－来自indexA和indexB的依赖偏移量的门限变量α'和β'的推导

	indexA (α')或 indexB (β')																									
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
α'	15	17	20	22	25	28	32	36	40	45	50	56	63	71	80	90	101	113	127	144	162	182	203	226	255	255
β'	6	6	7	7	8	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15	16	16	17	17	18	18

8.7.2.3 bS＜4的情况下的边缘滤波过程

本过程的输入是将要被滤波的一个边缘的一组样点中的输入样点值 p_i 和 q_i (i = 0..2)、chromaEdgeFlag,、bS,、β和 indexA，参见 8.7.2 中定义的输入样点组。

本过程的输出是滤波输入样点值组后得到的样点值 p'_i 和 q'_i (i = 0..2)。

通过下述方式得到滤波后得到的样点 p'_0 和 q'_0。

$$\Delta = \text{Clip3}(-t_C, t_C, ((((q_0 - p_0) << 2) + (p_1 - q_1) + 4) >> 3)) \quad (8\text{-}470)$$

$$p'_0 = \text{Clip1}(p_0 + \Delta) \quad (8\text{-}471)$$

$$q'_0 = \text{Clip1}(q_0 - \Delta) \quad (8\text{-}472)$$

由下面的内容决定门限 t_C。

— 如果chromaEdgeFlag等于0。

$$t_C = t_{C0} + ((a_p < \beta) ? 1 : 0) + ((a_q < \beta) ? 1 : 0) \quad (8\text{-}473)$$

— 否则(chromaEdgeFlag 等于1)，

$$t_C = t_{C0} + 1 \quad (8\text{-}474)$$

根据 indexA 和 bS 的值，表 8-17 给出了变量 t'_{C0} 的定义。根据 chromaEdgeFlag，通过下述方式得到相应的门限变量 t_{C0}。

— 如果chromaEdgeFlag等于 0，

$$t_{C0} = t'_{C0} * (1 << (\text{BitDepth}_Y - 8)) \quad (8\text{-}475)$$

— 否则(chromaEdgeFlag等于1),

$$t_{C0} = t'_{C0} * (1 << (\text{BitDepth}_C - 8)) \quad (8\text{-}476)$$

假设 a_p 和 a_q 是两个门限变量，它们的定义如下

$$a_p = \text{Abs}(p_2 - p_0) \quad (8\text{-}477)$$

$$a_q = \text{Abs}(q_2 - q_0) \quad (8\text{-}478)$$

通过下述方式得到滤波后的样点 p'_1

— 如果chromaEdgeFlag等于0同时a_p＜β，

$$p'_1 = p_1 + \text{Clip3}(-t_{C0}, t_{C0}, (p_2 + ((p_0 + q_0 + 1) >> 1) - (p_1 << 1)) >> 1) \quad (8\text{-}479)$$

— 否则(chromaEdgeFlag等于1或者a_p≥β)，

$$p'_1 = p_1 \quad (8\text{-}480)$$

通过如下方式得到滤波后的样点

— 如果chromaEdgeFlag等于0同时$a_q < \beta$，

$$q'_1 = q_1 + \text{Clip3}(-t_{C0}, t_{C0}, (q_2 + ((p_0 + q_0 + 1) >> 1) - (q_1 << 1)) >> 1) \quad (8\text{-}481)$$

— 否则(chromaEdgeFlag等于1或者 $a_q \geq \beta$)，

$$q'_1 = q_1 \quad (8\text{-}482)$$

通常把滤波后得到的样点 p'_2 和 q'_2 置为输入样点 p_2 和 q_2。

$$p'_2 = p_2 \quad (8\text{-}483)$$

$$q'_2 = q_2 \quad (8\text{-}484)$$

表 8-17－作为indexA和bS 函数的变量t'_{C0}的值

	indexA																									
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
bS = 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
bS = 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
bS = 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

表 8-17 (结束)－作为indexA和bS 函数的变量t'_{C0}的值

	indexA																									
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
bS = 1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13
bS = 2	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5	5	6	7	8	8	10	11	12	13	15	17
bS = 3	1	2	2	2	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13	14	16	18	20	23	25

8.7.2.4 bS等于4的情况下的边缘滤波过程

本过程的输入是将要被滤波的一个边缘的一组样点中的输入样点值 p_i 和 q_i (i = 0..3)、变量 chromaEdgeFlag 以及这组样点的门限变量α和β的值，参见 8.7.2 节中的定义。

本过程的输出是滤波这组输入样点值后得到的样点值 p'_i 和 q'_i (i = 0..2)。

假设 a_p 和 a_q 分别是方程式 8-477 和 8-478 中定义的 2 个门限变量，参见 8.7.2.3 节中的描述。

通过下述方式可以得到滤波后得到的样点值 p'_i (i = 0..2)。

— 如果chromaEdgeFlag等于0而且下述条件有效

$$a_p < \beta \ \&\& \ \text{Abs}(p_0 - q_0) < ((\alpha >> 2) + 2) \quad (8\text{-}485)$$

然后可以通过下述方式得到变量 p'_0、p'_1 和 p'_2。

$$p'_0 = (p_2 + 2*p_1 + 2*p_0 + 2*q_0 + q_1 + 4) >> 3 \quad (8\text{-}486)$$

$$p'_1 = (p_2 + p_1 + p_0 + q_0 + 2) >> 2 \quad (8\text{-}487)$$

$p'_2 = (2*p_3 + 3*p_2 + p_1 + p_0 + q_0 + 4) >> 3$ (8-488)

— 否则（chromaEdgeFlag等于1或者方程式8-484中的条件无效），那么通过下述方式得到变量p'_0、p'_1和p'_2。

$p'_0 = (2*p_1 + p_0 + q_1 + 2) >> 2$ (8-489)

$p'_1 = p_1$ (8-490)

$p'_2 = p_2$ (8-491)

通过如下方式得到滤波后的样点 q'_i (i = 0..2)。

— 如果chromaEdgeFlag等于0而且下面的条件有效，

$a_q < \beta$ && $Abs(p_0 - q_0) < ((\alpha >> 2) + 2)$ (8-492)

那么可以通过下述方式得到变量 q'_0、q'_1 和 q'_2

$q'_0 = (p_1 + 2*p_0 + 2*q_0 + 2*q_1 + q_2 + 4) >> 3$ (8-493)

$q'_1 = (p_0 + q_0 + q_1 + q_2 + 2) >> 2$ (8-494)

$q'_2 = (2*q_3 + 3*q_2 + q_1 + q_0 + p_0 + 4) >> 3$ (8-495)

— 否则（chromaEdgeFlag等于1或者方程式8-491中的条件无效），那么通过下述方式得到变量q'_0、q'_1和q'_2

$q'_0 = (2*q_1 + q_0 + p_1 + 2) >> 2$ (8-496)

$q'_1 = q_1$ (8-497)

$q'_2 = q_2$ (8-498)

9 解析过程

本过程的输入为 RBSP 的比特。

本过程的输出为语法元素值。

7.3 节的语法表格中的语法元素描述符等于 e(v)，me(v)，se(v)，te(v) (见 9.1 节), ce(v) (见 9.2 节)，或者 ae(v) (见 9.3 节)时，调用本过程。

9.1 指数哥伦布编码的解析过程

7.3 节的语法表格中的语法元素描述符等于 e(v)，me(v)，se(v)，te(v)时，调用本过程。对于 7.3.4 和 7.3.5 节中的语法元素，仅在 entropy_coding_mode_flag 等于 0 时，调用本过程。

本过程的输入是 RBSP 的比特。

本过程的输出是语法元素值。

编码为 ue(v)，me(v)，或者 se(v) 的语法元素是指数哥伦布编码。编码为 te(v)的语法元素是舍位指数哥伦布编码。这些语法元素的解析过程是由比特流当前位置比特开始读取，包括非 0 比特，直至 leading_bits 的数量为 0。具体过程如下所示：

```
leadingZeroBits = −1;
for( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )
```

变量 codeNum 按照如下方式赋值：

$codeNum = 2^{leadingZeroBits} - 1 + read\_bits(leadingZeroBits)$

这里 read_bits(leadingZeroBits)的返回值使用高位在先的二进制无符号整数表示。

表 9-1 给出了指数哥伦布编码的结构，将比特串分为“前缀”和“后缀”两个部分。“前缀”通过上面计算 leadingZeroBits 的伪码解析，在表 9-1 的比特串栏中以 0 或者 1 表示；“后缀”通过 codeNum 的计算解析，在表 9-1 中以 x_i 表示，I 的范围时从 0 到 leadingZeroBits − 1，每个 x_i 可以取 0 或者 1。

表 9-1－带有“前缀”和“后缀”比特的比特串和codeNum范围分配 (资料性)

比特串格式	CodeNum 范围
1	0
0 1 x_0	1-2
0 0 1 x_1 x_0	3-6
0 0 0 1 x_2 x_1 x_0	7-14
0 0 0 0 1 x_3 x_2 x_1 x_0	15-30
0 0 0 0 0 1 x_4 x_3 x_2 x_1 x_0	31-62
…	…

表 9-2 举例说明了 codeNum 值对应的比特串分配。

表 9-2－ue(v)的指数哥伦布比特串和codeNum (资料性)

比特串	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
…	…

语法元素的值取决于描述符，遵守如下规定：

— 如果语法元素编码为ue(v)，语法元素值等于codeNum。

— 否则，如果语法元素编码为se(v)，语法元素值通过调用9.1.1节所规定的有符号指示哥伦布编码的映射过程取得，codeNub作为输入。

— 否则，如果语法元素编码为me(v)，语法元素值通过调用9.1.2节所规定的已编码块模式的映射过程取得，codeNum作为输入。

— 否则，如果语法元素编码为te(v)，首先应该判断语法元素的范围。语法元素的范围可以是在0和x之间，x大于或者等于1，语法元素的值按照如下方法取得：

  — 如果x大于1，codeNum 和语法元素值的取得应该和ue(v)相同。

  — 否则(x 等于1)，与语法元素值相等的codeNum的解析过程由如下计算方法取得：

```
b = read_bits( 1 )
codeNum = !b
```

9.1.1 有符号指数哥伦布编码的映射过程

按照 9.1 节规定，本过程的输入是 codeNum。

本过程的输出是 se(v)的值。

表 9-3 中给出了分配给 codeNum 的语法元素值的规则，语法元素值按照绝对值的升序排列，负值按照其绝对值参与排列，但列在绝对值相等的正值之后。

表 9-3－有符号指数哥伦布编码语法元素se(v)值与codeNum的对应

codeNum	语法元素值
0	0
1	1
2	−1
3	2
4	−2
5	3
6	−3
k	$(-1)^{k+1}$ Ceil(k÷2)

9.1.2 已编码块模式的映射过程

按照 9.1 节的规定本过程的输入为 codeNum。

本过程的输出为 me(v)的值。

表 9-4 显示了 coded_block_pattern 和 codeNum 的对应关系，宏块预测模式等于 Intra_4x4, Intra_8x8 或者 Inter 时，coded_block_pattern 的值不同。

表 9-4－codeNum对应的宏块预测模式coded_block_pattern值

(a) chroma_format_idc不等于 0

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
0	47	0
1	31	16
2	15	1
3	0	2
4	23	4
5	27	8
6	29	32
7	30	3
8	7	5
9	11	10
10	13	12
11	14	15
12	39	47
13	43	7
14	45	11
15	46	13
16	16	14
17	3	6
18	5	9
19	10	31
20	12	35
21	19	37
22	21	42
23	26	44
24	28	33
25	35	34
26	37	36
27	42	40
28	44	39
29	1	43
30	2	45
31	4	46
32	8	17
33	17	18
34	18	20
35	20	24
36	24	19
37	6	21
38	9	26
39	22	28
40	25	23
41	32	27
42	33	29
43	34	30

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
44	36	22
45	40	25
46	38	38
47	41	41

(b) chroma_format_idc等于 0

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
0	15	0
1	0	1
2	7	2
3	11	4
4	13	8
5	14	3
6	3	5
7	5	10
8	10	12
9	12	15
10	1	7
11	2	11
12	4	13
13	8	14
14	6	6
15	9	9

9.2 变换系数幅值的CAVLC解析过程

当 7.3.5.3.1 节规定的解析语法元素的描述符等于 ce(v)，并且 entropy_coding_mode_flag 等于 0 时，调用本过程。

本过程的输入比特来自于条带数据，非 0 变换系数幅值 maxNumCoeff 的最大数量，当前变换系数幅值块的亮度索引 luma4x4BlkIdx 或者色度索引 chroma4x4BlkIdx。

本过程的输出是 coeffLevel 列表，包含了索引为 luma4x4BlkIdx 的亮度块或者索引为 chroma4x4BlkIdx 的色度块的变换系数幅值。

本过程遵守如下步骤：

1．所有索引从 0 到 maxNumCoeff − 1 的变换系数幅值，在 coeffLevel 列表中设为 0。

2．非 0 变换系数幅值 TotalCoeff(coeff_token)的总数和拖尾变换系数幅值 TrailingOnes(coeff_token)的数量按照如下过程解析 coeff_token (见 9.2.1 节)得到：

— 如果非0变换系数幅值TotalCoeff(coeff_token) 等于0，返回包含0 coeffLevel列表，不需要执行更多步骤。

— 否则，执行如下步骤：

a．非 0 变换系数幅值通过解析 trailing_ones_sign_flag，level_prefix 和 level_suffix (见 9.2.2 节)得到。

b．在每个非 0 变换系数幅值之前的 0 变换系数的游程通过解析 total_zeros 和 run_before (见 9.2.3 节)得到。

c．幅值和游程信息归到 coeffLevel 列表中（见 9.2.4 节）。

9.2.1 变换系数幅值和拖尾比特总数的解析过程

本过程的输入比特来自于条带数据，非 0 传送系数幅值 maxNumCoeff 的最大数量，当前变换块的亮度索引 luma4x4BlkIdx 或者色度索引 chroma4x4BlkIdx。

本过程的输出是 TotalCoeff(coeff_token) 和 TrailingOnes(coeff_token)。

语法元素 coeff_token 使用表 9-5 右边六列中的六个 VLC 之一解码。每个 VLC 规定了给定码字 coeff_token 的 TotalCoeff(coeff_token) 和 TrailingOnes(coeff_token)。VLC 的选择取决于变量 nC，nC 按照如下规则取值：

— 如果调用CAVLC解析过程是为了ChromaDCLevel，nC 的值如下：

  — 如果chroma_format_idc等于1，nC 等于 −1；

  — 如果 chroma_format_idc等于 2，nC等于−2；

  — 否则， (chroma_format_idc 等于 3)，nC 等于 0。

— 否则，应用如下过程：

  — 当调用CAVLC解析过程是为了Intra16x16DCLevel时，luma4x4BlkIdx 设为0。

  — 变量blkA 和 blkB的值如下：

    — 如果CAVLC解析过程的调用是为了Intra16x16DCLevel，Intra16x16ACLevel或者LumaLevel，将会调用 6.4.8.3 节所规定的过程，输入为 luma4x4BlkIdx，输出赋值给 mbAddrA， mbAddrB，luma4x4BlkIdxA 和 luma4x4BlkIdxB。mbAddrA\luma4x4BlkIdxA 所规定的4x4 色度块赋值给 blkA，mbAddrB\luma4x4BlkIdxB所规定的的4x4色度块赋值给blkB。

    — 否则（CAVLC解析过程的调用是为了ChromaACLevel），调用6.4.8.4节所规定的过程，输入为 chroma4x4BlkIdx，输出赋值给 mbAddrA, mbAddrB, chroma4x4BlkIdxA 和 chroma4x4BlkIdxB。mbAddrA\iCbCr\chroma4x4BlkIdxA所规定的4x4色度块赋值给blkA，mbAddrB\iCbCr\chroma4x4BlkIdxB所规定的4x4色度块赋值给blkB。

  — 设nA和nB分别为在当前宏块左侧和上侧的变换系数幅值blkA和blkB中的非0变换系数幅值数量（由 TotalCoeff(coeff_token)给出）。

  — 用N代替A和B，在mbAddrN，blkN和nN 中应用如下过程：

    — 如果如下任意条件为真，N 设为0。

    — mbAddrN 不可用。

      — 当前宏块使用内预测模式编码，constrained_intra_pred_flag等于1，mbAddrN使用中间预测模式编码，并且使用条带数据划分（nal_unit_type的范围为由2到4）。

    — 宏块mbAddrN有mb_type等于P_Skip或者B_Skip。

      — 由于CodedBlockPatternLuma或者CodedBlockPatternChroma的相应比特等于0，相邻块blkN的所有AC剩余变换系数幅值等于0。

  — 否则，如果mbAddrN是I_PCM宏块，nN 设为16。

    — 否则，nN等于相邻宏块blkN的TotalCoeff(coeff_token)值。

注1 — 使用TotalCoeff(coeff_token)推导出的nA和nB值不包括中间16x16宏块的DC变换系数幅值，或者色度块内的DC变换系数幅值，原因是这些变换系数幅值单独解码。当位于左侧或者上方的块属于一个内16x16宏块，或者是一个色度块，nA和nB是已解码的非0AC变换系数幅值的数量。

注2 — 当解析Intra16x16DCLevel时，nA和nB的值是基于相邻的4x4块的非0变换系数幅值数量，而不是相邻的16x16块的非0 DC变换系数幅值数量。

  — 给定nA 和nB的值，变量C的推导过程如下：

— 如果mbAddrA和mbAddrB都可用，变量nC 等于 (nA + nB + 1) >> 1。

— 否则(mbAddrA 不可用或者mbAddrB不可用)，变量nC等于 nA + nB。

通过解码 coeff_token 得到的 TotalCoeff(coeff_token)值的范围应该在 0 到 maxNumCoeff之间。

表 9-5— 映射到TotalCoeff(coeff_token) 和TrailingOnes(coeff_token) 的coeff_token

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC = = −1	nC = = −2
0	0	1	11	1111	0000 11	01	1
0	1	0001 01	0010 11	0011 11	0000 00	0001 11	0001 111
1	1	01	10	1110	0000 01	1	01
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00	0001 110
1	2	0001 00	0011 1	0111 1	0001 01	0001 10	0001 101
2	2	001	011	1101	0001 10	001	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11	0000 0011 1
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011	0001 100
2	3	0000 101	0010 01	0111 0	0010 10	0000 010	0001 011
3	3	0001 1	0101	1100	0010 11	0001 01	0000 1
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10	0000 0011 0
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011	0000 0010 1
2	4	0000 0101	0001 01	0101 1	0011 10	0000 0010	0001 010
3	4	0000 11	0100	1011	0011 11	0000 000	0000 01
0	5	0000 0000 111	0000 0100	0001 011	0100 00	-	0000 0001 11
1	5	0000 0001 10	0000 110	0100 0	0100 01	-	0000 0001 10
2	5	0000 0010 1	0000 101	0100 1	0100 10	-	0000 0010 0
3	5	0000 100	0011 0	1010	0100 11	-	0001 001
0	6	0000 0000 0111 1	0000 0011 1	0001 001	0101 00	-	0000 0000 111
1	6	0000 0000 110	0000 0110	0011 10	0101 01	-	0000 0000 110
2	6	0000 0001 01	0000 0101	0011 01	0101 10	-	0000 0001 01
3	6	0000 0100	0010 00	1001	0101 11	-	0001 000
0	7	0000 0000 0101 1	0000 0001 111	0001 000	0110 00	-	0000 0000 0111
1	7	0000 0000 0111 0	0000 0011 0	0010 10	0110 01	-	0000 0000 0110
2	7	0000 0000 101	0000 0010 1	0010 01	0110 10	-	0000 0000 101
3	7	0000 0010 0	0001 00	1000	0110 11	-	0000 0001 00
0	8	0000 0000 0100 0	0000 0001 011	0000 1111	0111 00	-	0000 0000 0011 1
1	8	0000 0000 0101 0	0000 0001 110	0001 110	0111 01	-	0000 0000 0101
2	8	0000 0000 0110 1	0000 0001 101	0001 101	0111 10	-	0000 0000 0100
3	8	0000 0001 00	0000 100	0110 1	0111 11	-	0000 0000 100
0	9	0000 0000 0011 11	0000 0000 1111	0000 1011	1000 00	-	
1	9	0000 0000 0011 10	0000 0001 010	0000 1110	1000 01	-	
2	9	0000 0000 0100 1	0000 0001 001	0001 010	1000 10	-	
3	9	0000 0000 100	0000 0010 0	0011 00	1000 11	-	
0	10	0000 0000 0010 11	0000 0000 1011	0000 0111 1	1001 00	-	
1	10	0000 0000 0010 10	0000 0000 1110	0000 1010	1001 01	-	
2	10	0000 0000 0011 01	0000 0000 1101	0000 1101	1001 10	-	
3	10	0000 0000 0110 0	0000 0001 100	0001 100	1001 11	-	
0	11	0000 0000 0001 111	0000 0000 1000	0000 0101 1	1010 00	-	

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC = = −1	nC = = −2
1	11	0000 0000 0001 110	0000 0000 1010	0000 0111 0	1010 01	-	
2	11	0000 0000 0010 01	0000 0000 1001	0000 1001	1010 10	-	
3	11	0000 0000 0011 00	0000 0001 000	0000 1100	1010 11	-	
0	12	0000 0000 0001 011	0000 0000 0111 1	0000 0100 0	1011 00	-	
1	12	0000 0000 0001 010	0000 0000 0111 0	0000 0101 0	1011 01	-	
2	12	0000 0000 0001 101	0000 0000 0110 1	0000 0110 1	1011 10	-	
3	12	0000 0000 0010 00	0000 0000 1100	0000 1000	1011 11	-	
0	13	0000 0000 0000 1111	0000 0000 0101 1	0000 0011 01	1100 00	-	
1	13	0000 0000 0000 001	0000 0000 0101 0	0000 0011 1	1100 01	-	
2	13	0000 0000 0001 001	0000 0000 0100 1	0000 0100 1	1100 10	-	
3	13	0000 0000 0001 100	0000 0000 0110 0	0000 0110 0	1100 11	-	
0	14	0000 0000 0000 1011	0000 0000 0011 1	0000 0010 01	1101 00	-	
1	14	0000 0000 0000 1110	0000 0000 0010 11	0000 0011 00	1101 01	-	
2	14	0000 0000 0000 1101	0000 0000 0011 0	0000 0010 11	1101 10	-	
3	14	0000 0000 0001 000	0000 0000 0100 0	0000 0010 10	1101 11	-	
0	15	0000 0000 0000 0111	0000 0000 0010 01	0000 0001 01	1110 00	-	
1	15	0000 0000 0000 1010	0000 0000 0010 00	0000 0010 00	1110 01	-	
2	15	0000 0000 0000 1001	0000 0000 0010 10	0000 0001 11	1110 10	-	
3	15	0000 0000 0000 1100	0000 0000 0000 1	0000 0001 10	1110 11	-	
0	16	0000 0000 0000 0100	0000 0000 0001 11	0000 0000 01	1111 00	-	
1	16	0000 0000 0000 0110	0000 0000 0001 10	0000 0001 00	1111 01	-	
2	16	0000 0000 0000 0101	0000 0000 0001 01	0000 0000 11	1111 10	-	
3	16	0000 0000 0000 1000	0000 0000 0001 00	0000 0000 10	1111 11	-	

9.2.2 幅值信息的解析过程

本过程的输入比特来自于条带数据，非 0 变换系数幅值 TotalCoeff(coeff_token)的数量，结尾变换系数幅值(coeff_token)的数量。

本过程的输出是包含了变换系数幅值的幅值列表。

索引 i 初始值设为 0。然后应用如下迭代过程 TrailingOnes(coeff_token)次，用以解码结尾变换系数幅值（如果有的话）。

— 1-bit 语法元素trailing_ones_sign_flag的解码和求值如下：

  — 如果trailing_ones_sign_flag等于0，level[i]等于值 +1。

  — 否则 (trailing_ones_sign_flag is equal to 1)，level[i]等于值−1。

— 索引I加1。

按照结尾变换系数幅值的解码，变量 suffixLength 的初始化过程如下：

  — 如果TotalCoeff(coeff_token)大于10，并且TrailingOnes(coeff_token)小于3，suffixLength等于1。

  — 否则 (TotalCoeff(coeff_token)小于或等于 10，或者TrailingOnes(coeff_token)等于3)，suffixLength等于0。

接下来的过程迭代应用 (TotalCoeff(coeff_token) − TrailingOnes(coeff_token)) 次，以解码剩余幅值 （如果有的话）：

— 语法元素level_prefix按照9.2.2.1节的规定解码。

  — 除以下两种情况外，变量levelSuffixSize等于变量suffixLength。

— 当level_prefix 等于14 且suffixLength等于 0时，levelSuffixSize 等于 4。

— 当level_prefix大于等于15时，levelSuffixSize等于 level_prefix − 3。

— 语法元素level_suffix 的解码如下：

  — 如果levelSuffixSize大于0，语法元素level_suffix的解码使用levelSuffixSize比特的无符号整数u(v)表示。

  — 否则(levelSuffixSize 等于0)，语法元素level_suffix 应该为0。

— 变量levelCode等于(Min(15, level_prefix) << suffixLength) + level_suffix。

— 当Wlevel_prefix 大于等于15，并且suffixLength等于0时，levelCode 递增15。

— 当level_prefix大于等于16时，levelCode递增(1<<(level_prefix − 3)) − 4096。

  — 当索引i 等于TrailingOnes(coeff_token) ，并且TrailingOnes(coeff_token)小于3时，levelCode递增2。

— 变量level[i]按照如下步骤推导：

  — 如果levelCode是偶数，赋(levelCode + 2) >> 1给level[i]。

  — 否则 (levelCode是一个奇数)，赋(−levelCode − 1) >> 1给level[i]。

— 当suffixLength等于0时，suffixLength等于1。

  — 当level[i]的绝对值大于 (3 << (suffixLength − 1)) ，并且suffixLength小于6时，suffixLength递增1。

— 索引i 递增1。

9.2.2.1 level_prefix的解析过程

本过程的输入比特来自条带数据。

输出是 level_prefix。

该语法元素的解析由如下过程组成：从比特流的当前位置开始向上读取，包括第一个非 0 比特，计算为 0 的 leading bits 数量。该过程应采取如下等同的步骤：

```
leadingZeroBits = −1
for( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )
level_prefix = leadingZeroBits
```

表 9-6 给出了 level_prefix 的码字示例表格。

表 9-6－level_prefix的码字表格 (资料性)

level_prefix	比特串
0	1
1	01
2	001
3	0001
4	0000 1
5	0000 01
6	0000 001
7	0000 0001
8	0000 0000 1
9	0000 0000 01
10	0000 0000 001
11	0000 0000 0001
12	0000 0000 0000 1
13	0000 0000 0000 01
14	0000 0000 0000 001
15	0000 0000 0000 0001
…	…

9.2.3 游程信息的解析过程

本过程的输入比特来自于条带数据，非 0 变换系数幅值 TotalCoeff(coeff_token)的数量，非 0 变换系数幅值maxNumCoeff 的最大值。

本过程的输出是非 0 变换系数幅值前的 0 变换系数幅值游程列表。

索引 I 初始置为 0。

变量 zerosLeft 通过如下过程推导：

— 如果非0变换系数幅值TotalCoeff(coeff_token)的数量和非0变换系数幅值maxNumCoeff的最大值相等，变量zerosLeft 等于 0。

— 否则(非0变换系数幅值TotalCoeff(coeff_token)的数量小于非0变换系数幅值maxNumCoeff的最大值)，解码total_zeros，zerosLeft I等于total_zeros。

用于解码 total_zeros 的 VLC 的推导过程如下：

如果 maxNumCoeff 等于 4，使用表 9-9 (a)规定的 VLC 之一。

— 否则，如果maxNumCoeff 等于 8，使用表9-9 (b)规定的VLC之一。

— 否则 (maxNumCoeff 不等于4，也不等于8)， 使用表9-7和表9-8的VLC。

然后迭代应用如下过程(TotalCoeff(coeff_token) − 1)次：

— 变量 run[i]按照如下过程推导：

— 如果 zerosLeft 大于0，根据表9-10 解码run_before，置zerosLeft. run[i]等于 run_before。

— 否则 (zerosLeft 等于0)，run[i] 等于 0。

— zerosLeft 减去run[i] 后所得的值赋于zerosLeft。该减法的结果应该大于或等于0。

— 索引 i 递增1。

最后 zerosLeft 的值赋给 run[i]。

表 9-7－TotalCoeff(coeff_token) 1到 7的4x4块total_zeros表格

total_zeros	TotalCoeff(coeff_token)						
	1	2	3	4	5	6	7
0	1	111	0101	0001 1	0101	0000 01	0000 01
1	011	110	111	111	0100	0000 1	0000 1
2	010	101	110	0101	0011	111	101
3	0011	100	101	0100	111	110	100
4	0010	011	0100	110	110	101	011
5	0001 1	0101	0011	101	101	100	11
6	0001 0	0100	100	100	100	011	010
7	0000 11	0011	011	0011	011	010	0001
8	0000 10	0010	0010	011	0010	0001	001
9	0000 011	0001 1	0001 1	0010	0000 1	001	0000 00
10	0000 010	0001 0	0001 0	0001 0	0001	0000 00	
11	0000 0011	0000 11	0000 01	0000 1	0000 0		
12	0000 0010	0000 10	0000 1	0000 0			
13	0000 0001 1	0000 01	0000 00				
14	0000 0001 0	0000 00					
15	0000 0000 1						

表 9-8－TotalCoeff(coeff_token) 8到 15的4x4块total_zeros表格

total_zeros	TotalCoeff(coeff_token)							
	8	9	10	11	12	13	14	15
0	0000 01	0000 01	0000 1	0000	0000	000	00	0
1	0001	0000 00	0000 0	0001	0001	001	01	1
2	0000 1	0001	001	001	01	1	1	
3	011	11	11	010	1	01		
4	11	10	10	1	001			
5	10	001	01	011				
6	010	01	0001					
7	001	0000 1						
8	0000 00							

表 9-9－色度DC 2x2和2x4块的total_zeros表格

(a) 色度 DC 2x2块k (4:2:0 色度样点)

total_zeros	TotalCoeff(coeff_token)		
	1	2	3
0	1	1	1
1	01	01	0
2	001	00	
3	000		

(b)色度 DC 2x4 块(4:2:2色度样点)

total_zeros	TotalCoeff(coeff_token)						
	1	2	3	4	5	6	7
0	1	000	000	110	00	00	0
1	010	01	001	00	01	01	1
2	011	001	01	01	10	1	
3	0010	100	10	10	11		
4	0011	101	110	111			
5	0001	110	111				
6	0000 1	111					
7	0000 0						

表 9-10－run_before表格

run_before	zerosLeft						
	1	2	3	4	5	6	>6
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
4	-	-	-	000	001	010	011
5	-	-	-	-	000	101	010
6	-	-	-	-	-	100	001
7	-	-	-	-	-	-	0001
8		-	-	-	-	-	00001
9	-	-	-	-	-	-	000001
10	-	-	-	-	-	-	0000001
11	-	-	-	-	-	-	00000001
12	-	-	-	-	-	-	000000001
13	-	-	-	-	-	-	0000000001
14	-	-	-	-	-	-	00000000001

9.2.4 组合幅值和游程信息

本过程的输入为变换系数幅值列表（称为幅值），游程列表（称为游程）和非 0 变换系数幅值 TotalCoeff(coeff_token)。

本过程的输出为变换系数幅值的 coeffLevel 列表。

变量 coeffNum 等于 −1，且索引 i 等于 (TotalCoeff(coeff_token)−1)。应用如下迭代过程 TotalCoeff(coeff_token)次：

— coeffNum增加run[i] + 1。

— coeffLevel[coeffNum]等于level[i]。

— 索引 i 增加1。

9.3 条带数据的CABAC 解析过程

当 entropy_coding_mode_flag 等于 1 时，解析具有 7.3.4 和 7.3.5 节的描述符 ae(v)语法元素需要调用本过程。

本过程的输入是语法元素值和预先解析语法元素值的请求。

本过程的输出是语法元素值。

当开始解析 7.3.4 节的条带数据时，按照 9.3.1 节的规定调用 CABAC 解析过程的初始化过程。

语法元素的解析过程如下：

对于每个被请求的语法元素的二值化表示，推导过程按照 9.3.2 的描述进行。

语法元素和已解析的二进制序列的二值化表示决定了解码流程按照 9.3.3 节的描述进行。

语法元素的二进制序列中的每一个二进制码，都可以使用 binIdx 索引，上下文索引 ctxIdx 可以通过 9.3.3.2 节推导出来。

对于每个 ctxIdx，其算术解码过程都是按照 9.3.3.2 节的规定调用的。

已解析的二进制码的结果序列(b_0 .. b_{binIdx})需要和每个二进制码解码后的二进制序列所给出的二进制码串序列相比较。如果序列值和给定序列匹配，语法元素就会被赋于相应值。

如果语法元素值的请求的处理是为了语法元素 mb_type，并且 mb_type 的解码值等于 I_PCM，在按照 9.3.1.2 节的规定解码 pcm_alignment_zero_bit，all pcm_sample_luma 和 pcm_sample_chroma 的任何一个之后，都需要进行解码引擎的初始化处理。

CABAC 的解析过程如图 9-1 所示的流程，图中 SE 代表语法元素。

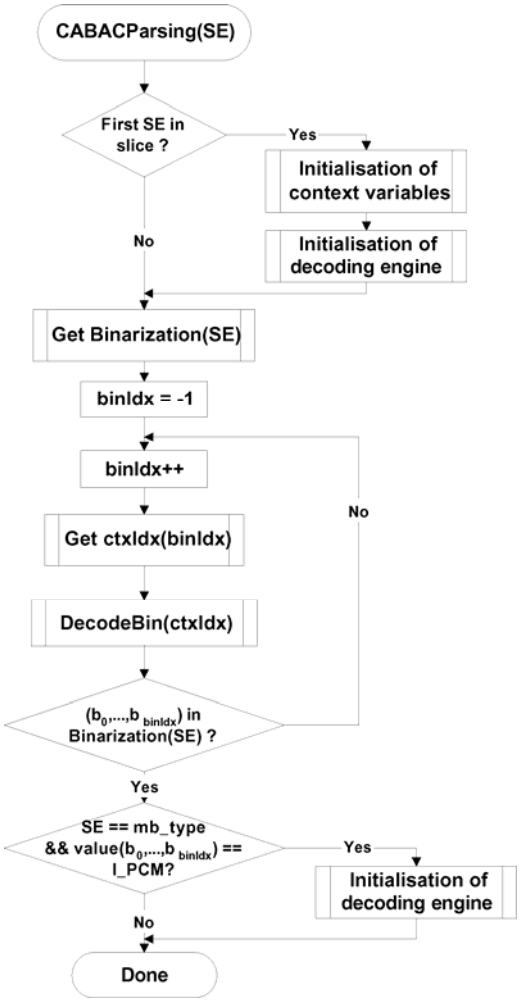


图 9-1－语法元素SE的CABAC解析过程举例 (资料性)

9.3.1 初始化过程

本过程的输出为初始化后的 CABAC 内部变量。

当开始解析 7.3.4 节的条带数据时，调用 9.3.1.1 和 9.3.1.2 节的处理过程。

在为类型为 I_PCM 的宏块解码 pcm_alignment_zero_bit，所有 pcm_sample_luma 和 pcm_sample_chroma 的任何一个之后，应该调用 9.3.1.2 节的处理。

9.3.1.1 上下文变量的初始化过程

本过程的输出是初始化了的 CABAC 上下文变量，该变量通过 ctxIdx 索引。

表 9-12 到 9-23 包含了在初始化上下文变量时使用的变量 n 和 m 值，在 7.3.4 和 7.3.5 节，除了 end-of-slice 标志外，分配给所有的语法元素。

对于每个上下文变量，应该初始化两个变量 pStateIdx 和 valMPS。

注1 — 变量pStateIdx对应于概率状态索引，变量 valMPS对应于最大可能，9.3.3.2中有更进一步的描述。

在初始化时赋于 pStateIdx 和 valMPS 的两个值通过 $SliceQP_Y$ 推导出来，推导过程见等式 7-27。给定两个值(m,n)。

1．preCtxState = Clip3(1, 126, ((m * Clip3(0, 51, $SliceQP_Y$)) >> 4) + n)

2．if(preCtxState <= 63) {

pStateIdx = 63 − preCtxState

valMPS = 0

} else {

pStateIdx = preCtxState − 64

valMPS = 1

}

在表 9-11 中，列出了所有条带类型需要初始化的 ctxIdx。同时包含了初始化过程中需要的 m 和 n 的值的表格编号也在表中列出。对于 P、SP 和 B 条带类型，初始化过程也取决于 cabac_init_idc 语法元素的值。需要注意的是，语法元素名字并不影响初始化过程。

表 9-11－所有条带类型初始化过程所需的ctxIdx和语法元素联合列表

	语法元素	表	条带类型			
			SI	I	P, SP	B
slice_data( )	mb_skip_flag	表 9-13 表 9-14			11-13	24-26
	mb_field_decoding_flag	表 9-18	70-72	70-72	70-72	70-72
macroblock_layer( )	mb_type	表 9-12 表 9-13 表 9-14	0-10	3-10	14-20	27-35
	transform_size_8x8_flag	表 9-16	na	399-401	399-401	399-401
	coded_block_pattern (luma)	表 9-18	73-76	73-76	73-76	73-76
	coded_block_pattern (chroma)	表 9-18	77-84	77-84	77-84	77-84
	mb_qp_delta	表 9-17	60-63	60-63	60-63	60-63
mb_pred( )	prev_intra4x4_pred_mode_flag	表 9-17	68	68	68	68
	rem_intra4x4_pred_mode	表 9-17	69	69	69	69
	prev_intra8x8_pred_mode_flag	表 9-17	na	68	68	68
	rem_intra8x8_pred_mode	表 9-17	na	69	69	69
	intra_chroma_pred_mode	表 9-17	64-67	64-67	64-67	64-67
mb_pred( ) and sub_mb_pred( )	ref_idx_l0	表 9-16			54-59	54-59
	ref_idx_l1	表 9-16				54-59
	mvd_l0[ ][ ][0]	表 9-15			40-46	40-46
	mvd_l1[ ][ ][0]	表 9-15				40-46
	mvd_l0[ ][ ][1]	表 9-15			47-53	47-53

	mvd_l1[ ][ ][1]	表 9-15				47-53
sub_mb_pred( )	sub_mb_type	表 9-13 表 9-14			21-23	36-39
residual_block_cabac( )	coded_block_flag	表 9-18	85-104	85-104	85-104	85-104
	significant_coeff_flag[ ]	表 9-19 表 9-22 表 9-24 表 9-24	105-165 277-337	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450
	last_significant_coeff_flag[ ]	表 9-20 表 9-23 表 9-24 表 9-24	166-226 338-398	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459
	coeff_abs_level_minus1[ ]	表 9-21 表 9-24	227-275	227-275 426-435	227-275 426-435	227-275 426-435

注2 — 等于 276的ctxIdx与规定了I PCM宏块类型的end_of_slice_flag和mb类型是有关的。9.3.3.2.4节所规定的解码处理可以应用在ctxIdx等于276时。不过，该解码过程也可以按照9.3.3.2.1节规定的解码过程执行。在这种情况下，和ctxIdx等于276相关的初始值可以规定为pStateIdx = 63和valMPS = 0，这里pStateIdx = 63意味着没有合适的概率状态。

表 9-12－ctxIdx从0到10时变量m和n的值

初始化变量	**ctxIdx**										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
m	20	2	3	20	2	3	−28	−23	−6	−1	7
n	−15	54	74	−15	54	74	127	104	53	54	51

表 9-13－ctxIdx从11到23时变量m和n的值

cabac_init_idc 值	初始化变量	ctxIdx												
		11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
0	**m**	23	23	21	1	0	−37	5	−13	−11	1	12	−4	17
	n	33	2	0	9	49	118	57	78	65	62	49	73	50
1	**m**	22	34	16	−2	4	−29	2	−6	−13	5	9	−3	10
	n	25	0	0	9	41	118	65	71	79	52	50	70	54
2	**m**	29	25	14	−10	−3	−27	26	−4	−24	5	6	−17	14
	n	16	0	0	51	62	99	16	85	102	57	57	73	57

表 9-14－ctxIdx从24到39时变量m和n的值

cabac_init_idc 值	初始化变量	ctxIdx															
		24	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**
0	**m**	18	9	29	26	16	9	−46	−20	1	−13	−11	1	−6	−17	−6	9
	n	64	43	0	67	90	104	127	104	67	78	65	62	86	95	61	45
1	**m**	26	19	40	57	41	26	−45	−15	−4	−6	−13	5	6	−13	0	8
	n	34	22	0	2	36	69	127	101	76	71	79	52	69	90	52	43
2	**m**	20	20	29	54	37	12	−32	−22	−2	−4	−24	5	−6	−14	−6	4
	n	40	10	0	0	42	97	127	117	74	85	102	57	93	88	44	55

表 9-15－ctxIdx从40到53时变量m和n的值

cabac_init_idc 值	初始化变量	ctxIdx													
		40	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**
0	**m**	−3	−6	−11	6	7	−5	2	0	−3	−10	5	4	−3	0
	n	69	81	96	55	67	86	88	58	76	94	54	69	81	88
1	**m**	−2	−5	−10	2	2	−3	−3	1	−3	−6	0	−3	−7	−5
	n	69	82	96	59	75	87	100	56	74	85	59	81	86	95
2	**m**	−11	−15	−21	19	20	4	6	1	−5	−13	5	6	−3	−1
	n	89	103	116	57	58	84	96	63	85	106	63	75	90	101

表 9-16－ctxIdx从54到59，从399到401时变量m和n的值

cabac_init_idc 值	初始化变量	ctxIdx								
		54	**55**	**56**	**57**	**58**	**59**	**399**	**400**	**401**
I 条带	**m**	na	na	na	na	na	na	31	31	25
	n	na	na	na	na	na	na	21	31	50
0	**m**	−7	−5	−4	−5	−7	1	12	11	14
	n	67	74	74	80	72	58	40	51	59
1	**m**	−1	−1	1	−2	−5	0	25	21	21
	n	66	77	70	86	72	61	32	49	54
2	**m**	3	−4	−2	−12	−7	1	21	19	17
	n	55	79	75	97	50	60	33	50	61

表 9-17－ctxIdx从60到69时变量m和n的值

初始化变量	ctxIdx									
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**
m	0	0	0	0	−9	4	0	−7	13	3
n	41	63	63	63	83	86	97	72	41	62

表 9-18－ctxIdx从70到104时变量m和n的值

ctxIdx	I 和 SI 条带		Cabac_init_idc 值						ctxIdx	I 和 SI 条带		Cabac_init_idc 值					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
70	0	11	0	45	13	15	7	34	**88**	−11	115	−13	108	−4	92	5	78
71	1	55	−4	78	7	51	−9	88	**89**	−12	63	−3	46	0	39	−6	55
72	0	69	−3	96	2	80	−20	127	**90**	−2	68	−1	65	0	65	4	61
73	−17	127	−27	126	−39	127	−36	127	**91**	−15	84	−1	57	−15	84	−14	83
74	−13	102	−28	98	−18	91	−17	91	**92**	−13	104	−9	93	−35	127	−37	127
75	0	82	−25	101	−17	96	−14	95	**93**	−3	70	−3	74	−2	73	−5	79
76	−7	74	−23	67	−26	81	−25	84	**94**	−8	93	−9	92	−12	104	−11	104
77	−21	107	−28	82	−35	98	−25	86	**95**	−10	90	−8	87	−9	91	−11	91
78	−27	127	−20	94	−24	102	−12	89	**96**	–30	127	−23	126	−31	127	−30	127
79	−31	127	−16	83	−23	97	−17	91	**97**	−1	74	5	54	3	55	0	65
80	−24	127	−22	110	−27	119	−31	127	**98**	−6	97	6	60	7	56	−2	79
81	−18	95	−21	91	−24	99	−14	76	**99**	−7	91	6	59	7	55	0	72
82	−27	127	−18	102	−21	110	−18	103	**100**	−20	127	6	69	8	61	−4	92
83	−21	114	−13	93	−18	102	−13	90	**101**	−4	56	−1	48	−3	53	−6	56
84	−30	127	−29	127	−36	127	−37	127	**102**	−5	82	0	68	0	68	3	68
85	−17	123	−7	92	0	80	11	80	**103**	−7	76	−4	69	−7	74	−8	71
86	−12	115	−5	89	−5	89	5	76	**104**	−22	125	−8	88	−9	88	−13	98
87	−16	122	−7	96	−7	94	2	84									

表 9-19－ctxIdx从105到165时变量m和n的值

ctxIdx	I 和 SI 条带		Cabac_init_idc 值						ctxIdx	I 和 SI 条带		Cabac_init_idc 值					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
105	−7	93	−2	85	−13	103	−4	86	**136**	−13	101	5	53	0	58	−5	75
106	−11	87	−6	78	−13	91	−12	88	**137**	−13	91	−2	61	−1	60	−8	80
107	−3	77	−1	75	−9	89	−5	82	**138**	−12	94	0	56	−3	61	−21	83
108	−5	71	−7	77	−14	92	−3	72	**139**	−10	88	0	56	−8	67	−21	64
109	−4	63	2	54	−8	76	−4	67	**140**	−16	84	−13	63	−25	84	−13	31
110	−4	68	5	50	−12	87	−8	72	**141**	−10	86	−5	60	−14	74	−25	64
111	−12	84	−3	68	−23	110	−16	89	**142**	−7	83	−1	62	−5	65	−29	94
112	−7	62	1	50	−24	105	−9	69	**143**	−13	87	4	57	5	52	9	75
113	−7	65	6	42	−10	78	−1	59	**144**	−19	94	−6	69	2	57	17	63
114	8	61	−4	81	−20	112	5	66	**145**	1	70	4	57	0	61	−8	74
115	5	56	1	63	−17	99	4	57	**146**	0	72	14	39	−9	69	−5	35
116	−2	66	−4	70	−78	127	−4	71	**147**	−5	74	4	51	−11	70	−2	27
117	1	64	0	67	−70	127	−2	71	**148**	18	59	13	68	18	55	13	91
118	0	61	2	57	−50	127	2	58	**149**	−8	102	3	64	−4	71	3	65
119	−2	78	−2	76	−46	127	−1	74	**150**	−15	100	1	61	0	58	−7	69
120	1	50	11	35	−4	66	−4	44	**151**	0	95	9	63	7	61	8	77
121	7	52	4	64	−5	78	−1	69	**152**	−4	75	7	50	9	41	−10	66
122	10	35	1	61	−4	71	0	62	**153**	2	72	16	39	18	25	3	62
123	0	44	11	35	−8	72	−7	51	**154**	−11	75	5	44	9	32	−3	68
124	11	38	18	25	2	59	−4	47	**155**	−3	71	4	52	5	43	−20	81
125	1	45	12	24	−1	55	−6	42	**156**	15	46	11	48	9	47	0	30
126	0	46	13	29	−7	70	−3	41	**157**	−13	69	−5	60	0	44	1	7
127	5	44	13	36	−6	75	−6	53	**158**	0	62	−1	59	0	51	−3	23
128	31	17	−10	93	−8	89	8	76	**159**	0	65	0	59	2	46	−21	74
129	1	51	−7	73	−34	119	−9	78	**160**	21	37	22	33	19	38	16	66
130	7	50	−2	73	−3	75	−11	83	**161**	−15	72	5	44	−4	66	−23	124
131	28	19	13	46	32	20	9	52	**162**	9	57	14	43	15	38	17	37
132	16	33	9	49	30	22	0	67	**163**	16	54	−1	78	12	42	44	−18
133	14	62	−7	100	−44	127	−5	90	**164**	0	62	0	60	9	34	50	−34
134	−13	108	9	53	0	54	1	67	**165**	12	72	9	69	0	89	−22	127
135	−15	100	2	53	−5	61	−15	72									

表 9-20－ctxIdx从166到226时变量m和n的值

ctxIdx	I 和 SI 条带		Cabac_init_idc 值						ctxIdx	I 和 SI 条带		Cabac_init_idc 值					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
166	24	0	11	28	4	45	4	39	**197**	26	−17	28	3	36	−28	28	−3
167	15	9	2	40	10	28	0	42	**198**	30	−25	28	4	38	−28	24	10
168	8	25	3	44	10	31	7	34	**199**	28	−20	32	0	38	−27	27	0
169	13	18	0	49	33	−11	11	29	**200**	33	−23	34	−1	34	−18	34	−14
170	15	9	0	46	52	−43	8	31	**201**	37	−27	30	6	35	−16	52	−44
171	13	19	2	44	18	15	6	37	**202**	33	−23	30	6	34	−14	39	−24
172	10	37	2	51	28	0	7	42	**203**	40	−28	32	9	32	−8	19	17
173	12	18	0	47	35	−22	3	40	**204**	38	−17	31	19	37	−6	31	25
174	6	29	4	39	38	−25	8	33	**205**	33	−11	26	27	35	0	36	29
175	20	33	2	62	34	0	13	43	**206**	40	−15	26	30	30	10	24	33
176	15	30	6	46	39	−18	13	36	**207**	41	−6	37	20	28	18	34	15
177	4	45	0	54	32	−12	4	47	**208**	38	1	28	34	26	25	30	20
178	1	58	3	54	102	−94	3	55	**209**	41	17	17	70	29	41	22	73
179	0	62	2	58	0	0	2	58	**210**	30	−6	1	67	0	75	20	34
180	7	61	4	63	56	−15	6	60	**211**	27	3	5	59	2	72	19	31
181	12	38	6	51	33	−4	8	44	**212**	26	22	9	67	8	77	27	44
182	11	45	6	57	29	10	11	44	**213**	37	−16	16	30	14	35	19	16
183	15	39	7	53	37	−5	14	42	**214**	35	−4	18	32	18	31	15	36
184	11	42	6	52	51	−29	7	48	**215**	38	−8	18	35	17	35	15	36
185	13	44	6	55	39	−9	4	56	**216**	38	−3	22	29	21	30	21	28
186	16	45	11	45	52	−34	4	52	**217**	37	3	24	31	17	45	25	21
187	12	41	14	36	69	−58	13	37	**218**	38	5	23	38	20	42	30	20
188	10	49	8	53	67	−63	9	49	**219**	42	0	18	43	18	45	31	12
189	30	34	−1	82	44	−5	19	58	**220**	35	16	20	41	27	26	27	16
190	18	42	7	55	32	7	10	48	**221**	39	22	11	63	16	54	24	42
191	10	55	−3	78	55	−29	12	45	**222**	14	48	9	59	7	66	0	93
192	17	51	15	46	32	1	0	69	**223**	27	37	9	64	16	56	14	56
193	17	46	22	31	0	0	20	33	**224**	21	60	–1	94	11	73	15	57
194	0	89	−1	84	27	36	8	63	**225**	12	68	–2	89	10	67	26	38
195	26	−19	25	7	33	−25	35	–18	**226**	2	97	–9	108	−10	116	−24	127
196	22	−17	30	−7	34	−30	33	–25									

表9-21－ctxIdx从227到275时变量m和n的值

ctxIdx	I 和 SI 条带		Cabac_init_idc 值						ctxIdx	I 和 SI 条带		Cabac_init_idc 值					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
227	−3	71	−6	76	−23	112	−24	115	**252**	−12	73	−6	55	−16	72	–14	75
228	−6	42	−2	44	−15	71	−22	82	**253**	−8	76	0	58	−7	69	–10	79
229	−5	50	0	45	−7	61	−9	62	**254**	−7	80	0	64	−4	69	−9	83
230	−3	54	0	52	0	53	0	53	**255**	−9	88	−3	74	−5	74	−12	92
231	−2	62	−3	64	−5	66	0	59	**256**	−17	110	−10	90	−9	86	−18	108
232	0	58	−2	59	−11	77	−14	85	**257**	−11	97	0	70	2	66	−4	79
233	1	63	−4	70	−9	80	−13	89	**258**	−20	84	−4	29	−9	34	−22	69
234	−2	72	−4	75	−9	84	−13	94	**259**	−11	79	5	31	1	32	−16	75
235	−1	74	−8	82	−10	87	−11	92	**260**	−6	73	7	42	11	31	−2	58
236	−9	91	−17	102	−34	127	−29	127	**261**	−4	74	1	59	5	52	1	58
237	−5	67	−9	77	−21	101	−21	100	**262**	−13	86	−2	58	−2	55	−13	78
238	−5	27	3	24	−3	39	−14	57	**263**	−13	96	−3	72	−2	67	−9	83
239	−3	39	0	42	−5	53	−12	67	**264**	−11	97	−3	81	0	73	−4	81
240	−2	44	0	48	−7	61	−11	71	**265**	−19	117	−11	97	−8	89	−13	99
241	0	46	0	55	−11	75	−10	77	**266**	−8	78	0	58	3	52	−13	81
242	−16	64	−6	59	−15	77	−21	85	**267**	−5	33	8	5	7	4	−6	38
243	−8	68	−7	71	−17	91	−16	88	**268**	−4	48	10	14	10	8	−13	62
244	−10	78	−12	83	−25	107	−23	104	**269**	−2	53	14	18	17	8	−6	58
245	−6	77	−11	87	−25	111	−15	98	**270**	−3	62	13	27	16	19	−2	59
246	−10	86	−30	119	−28	122	−37	127	**271**	−13	71	2	40	3	37	−16	73
247	−12	92	1	58	−11	76	−10	82	**272**	−10	79	0	58	−1	61	−10	76
248	−15	55	−3	29	−10	44	−8	48	**273**	−12	86	−3	70	−5	73	−13	86
249	−10	60	−1	36	−10	52	−8	61	**274**	−13	90	−6	79	−1	70	−9	83
250	−6	62	1	38	−10	57	−8	66	**275**	−14	97	−8	85	−4	78	−10	87
251	−4	65	2	43	−9	58	−7	70									

表9-22－ctxIdx从277到337时变量m和n的值

ctxIdx	I 和 SI 条带		Cabac_init_idc 值						ctxIdx	I 和 SI 条带		Cabac_init_idc 值					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
277	−6	93	−13	106	−21	126	−22	127	**308**	−16	96	−1	51	−16	77	−10	67
278	−6	84	−16	106	−23	124	−25	127	**309**	−7	88	7	49	−2	64	1	68
279	−8	79	−10	87	−20	110	−25	120	**310**	−8	85	8	52	2	61	0	77
280	0	66	−21	114	−26	126	−27	127	**311**	−7	85	9	41	−6	67	2	64
281	−1	71	−18	110	−25	124	−19	114	**312**	−9	85	6	47	−3	64	0	68
282	0	62	−14	98	−17	105	−23	117	**313**	−13	88	2	55	2	57	−5	78
283	−2	60	−22	110	−27	121	−25	118	**314**	4	66	13	41	−3	65	7	55
284	−2	59	−21	106	−27	117	−26	117	**315**	−3	77	10	44	−3	66	5	59
285	−5	75	−18	103	−17	102	−24	113	**316**	−3	76	6	50	0	62	2	65
286	−3	62	−21	107	−26	117	−28	118	**317**	−6	76	5	53	9	51	14	54
287	−4	58	−23	108	−27	116	−31	120	**318**	10	58	13	49	−1	66	15	44
288	−9	66	−26	112	−33	122	−37	124	**319**	−1	76	4	63	−2	71	5	60
289	−1	79	−10	96	−10	95	−10	94	**320**	–1	83	6	64	−2	75	2	70
290	0	71	−12	95	−14	100	−15	102	**321**	−7	99	−2	69	−1	70	−2	76
291	3	68	−5	91	−8	95	−10	99	**322**	−14	95	−2	59	−9	72	−18	86
292	10	44	−9	93	−17	111	−13	106	**323**	2	95	6	70	14	60	12	70
293	−7	62	−22	94	−28	114	−50	127	**324**	0	76	10	44	16	37	5	64
294	15	36	−5	86	−6	89	−5	92	**325**	−5	74	9	31	0	47	−12	70
295	14	40	9	67	−2	80	17	57	**326**	0	70	12	43	18	35	11	55
296	16	27	−4	80	−4	82	−5	86	**327**	−11	75	3	53	11	37	5	56
297	12	29	−10	85	−9	85	−13	94	**328**	1	68	14	34	12	41	0	69
298	1	44	−1	70	−8	81	−12	91	**329**	0	65	10	38	10	41	2	65
299	20	36	7	60	−1	72	−2	77	**330**	−14	73	−3	52	2	48	−6	74
300	18	32	9	58	5	64	0	71	**331**	3	62	13	40	12	41	5	54
301	5	42	5	61	1	67	−1	73	**332**	4	62	17	32	13	41	7	54
302	1	48	12	50	9	56	4	64	**333**	−1	68	7	44	0	59	−6	76
303	10	62	15	50	0	69	−7	81	**334**	−13	75	7	38	3	50	−11	82
304	17	46	18	49	1	69	5	64	**335**	11	55	13	50	19	40	−2	77
305	9	64	17	54	7	69	15	57	**336**	5	64	10	57	3	66	−2	77
306	−12	104	10	41	−7	69	1	67	**337**	12	70	26	43	18	50	25	42
307	−11	97	7	46	−6	67	0	68									

表 9-23－ctxIdx从338到398时变量m和n的值

ctxIdx	I 和 SI 条带		Cabac_init_idc 值						ctxIdx	I 条带 SI 条带		Cabac_init_idc 值					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
338	15	6	14	11	19	−6	17	−13	**369**	32	−26	31	−4	40	−37	37	−17
339	6	19	11	14	18	−6	16	−9	**370**	37	−30	27	6	38	−30	32	1
340	7	16	9	11	14	0	17	−12	**371**	44	−32	34	8	46	−33	34	15
341	12	14	18	11	26	−12	27	−21	**372**	34	−18	30	10	42	−30	29	15
342	18	13	21	9	31	−16	37	−30	**373**	34	−15	24	22	40	−24	24	25
343	13	11	23	−2	33	−25	41	−40	**374**	40	−15	33	19	49	−29	34	22
344	13	15	32	−15	33	−22	42	−41	**375**	33	−7	22	32	38	−12	31	16
345	15	16	32	−15	37	−28	48	−47	**376**	35	−5	26	31	40	−10	35	18
346	12	23	34	−21	39	−30	39	−32	**377**	33	0	21	41	38	−3	31	28
347	13	23	39	−23	42	−30	46	−40	**378**	38	2	26	44	46	−5	33	41
348	15	20	42	−33	47	−42	52	−51	**379**	33	13	23	47	31	20	36	28
349	14	26	41	−31	45	−36	46	−41	**380**	23	35	16	65	29	30	27	47
350	14	44	46	−28	49	−34	52	−39	**381**	13	58	14	71	25	44	21	62
351	17	40	38	−12	41	−17	43	−19	**382**	29	−3	8	60	12	48	18	31
352	17	47	21	29	32	9	32	11	**383**	26	0	6	63	11	49	19	26
353	24	17	45	−24	69	−71	61	−55	**384**	22	30	17	65	26	45	36	24
354	21	21	53	−45	63	−63	56	−46	**385**	31	−7	21	24	22	22	24	23
355	25	22	48	−26	66	−64	62	−50	**386**	35	−15	23	20	23	22	27	16
356	31	27	65	−43	77	−74	81	−67	**387**	34	−3	26	23	27	21	24	30
357	22	29	43	−19	54	−39	45	−20	**388**	34	3	27	32	33	20	31	29
358	19	35	39	−10	52	−35	35	−2	**389**	36	−1	28	23	26	28	22	41
359	14	50	30	9	41	−10	28	15	**390**	34	5	28	24	30	24	22	42
360	10	57	18	26	36	0	34	1	**391**	32	11	23	40	27	34	16	60
361	7	63	20	27	40	−1	39	1	**392**	35	5	24	32	18	42	15	52
362	−2	77	0	57	30	14	30	17	**393**	34	12	28	29	25	39	14	60
363	−4	82	−14	82	28	26	20	38	**394**	39	11	23	42	18	50	3	78
364	−3	94	−5	75	23	37	18	45	**395**	30	29	19	57	12	70	−16	123
365	9	69	−19	97	12	55	15	54	**396**	34	26	22	53	21	54	21	53
366	−12	109	−35	125	11	65	0	79	**397**	29	39	22	61	14	71	22	56
367	36	−35	27	0	37	−33	36	−16	**398**	19	66	11	86	11	83	25	61
368	36	−34	28	0	39	−36	37	−14									

表 9-24－ctxIdx从402到459时变量m和n的值

ctxIdx	I 条带		Cabac_init_idc 值						ctxIdx	I 条带		Cabac_init_idc 值					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
402	−17	120	−4	79	−5	85	−3	78	**431**	−2	55	−12	56	−9	57	−12	59
403	−20	112	−7	71	−6	81	−8	74	**432**	0	61	−6	60	−6	63	−8	63
404	−18	114	−5	69	−10	77	−9	72	**433**	1	64	−5	62	−4	65	−9	67
405	−11	85	−9	70	−7	81	−10	72	**434**	0	68	−8	66	−4	67	−6	68
406	−15	92	−8	66	−17	80	−18	75	**435**	−9	92	−8	76	−7	82	−10	79
407	−14	89	−10	68	−18	73	−12	71	**436**	−14	106	−5	85	−3	81	−3	78
408	−26	71	−19	73	−4	74	−11	63	**437**	−13	97	−6	81	−3	76	−8	74
409	−15	81	−12	69	−10	83	−5	70	**438**	−15	90	−10	77	−7	72	−9	72
410	−14	80	−16	70	−9	71	−17	75	**439**	−12	90	−7	81	−6	78	−10	72
411	0	68	−15	67	−9	67	−14	72	**440**	−18	88	−17	80	−12	72	−18	75
412	−14	70	−20	62	−1	61	−16	67	**441**	−10	73	−18	73	−14	68	−12	71
413	−24	56	−19	70	−8	66	−8	53	**442**	−9	79	−4	74	−3	70	−11	63
414	−23	68	−16	66	−14	66	−14	59	**443**	−14	86	−10	83	−6	76	−5	70
415	−24	50	−22	65	0	59	−9	52	**444**	−10	73	−9	71	−5	66	−17	75
416	−11	74	−20	63	2	59	−11	68	**445**	−10	70	−9	67	−5	62	−14	72
417	23	−13	9	−2	17	−10	9	−2	**446**	−10	69	−1	61	0	57	−16	67
418	26	−13	26	−9	32	−13	30	−10	**447**	−5	66	−8	66	−4	61	−8	53
419	40	−15	33	−9	42	−9	31	−4	**448**	−9	64	−14	66	−9	60	−14	59
420	49	−14	39	−7	49	−5	33	−1	**449**	−5	58	0	59	1	54	−9	52
421	44	3	41	−2	53	0	33	7	**450**	2	59	2	59	2	58	−11	68
422	45	6	45	3	64	3	31	12	**451**	21	−10	21	−13	17	−10	9	−2
423	44	34	49	9	68	10	37	23	**452**	24	−11	33	−14	32	−13	30	−10
424	33	54	45	27	66	27	31	38	**453**	28	−8	39	−7	42	−9	31	−4
425	19	82	36	59	47	57	20	64	**454**	28	−1	46	−2	49	−5	33	−1
426	−3	75	−6	66	−5	71	−9	71	**455**	29	3	51	2	53	0	33	7
427	−1	23	−7	35	0	24	−7	37	**456**	29	9	60	6	64	3	31	12
428	1	34	−7	42	−1	36	−8	44	**457**	35	20	61	17	68	10	37	23
429	1	43	−8	45	−2	42	−11	49	**458**	29	36	55	34	66	27	31	38
430	0	54	−5	48	−2	52	−10	56	**459**	14	67	42	62	47	57	20	64

9.3.1.2 算术解码引擎的初始化过程

本过程的调用在解码条带的第一个宏块之前，或者解码类型为 I_PCM 宏块的

本过程的输出为初始化的解码引擎注册器 codIRange 和 codIOffset

算术解码引起的状态通过变量 codIRange 和 codIOffset 表示。在算术解码处理的初始化过程中，codIRange 设

为 0x01FE，codIOffset 设为 read_bits(9)的返回值，使用高位在先的无符号整数的 9 比特二进制表示。

比特流中不能包含会导致 codIOffset 等于 0x01FE 或者 0x01FF 的数据。

注 — 本建议书 | 国家标准中的算术解码引擎描述使用16比特注册器精度。不过变量codIRange和 codIOffset的最小精度为9比特。

9.3.2 二值化过程

本过程的输入是请求语法元素。

本过程的输出是二值化的语法元素，maxBinIdxCtx，ctxIdxOffset 和 bypassFlag。

表 9-25 规定了二值化过程的类型和每个语法元素相关的 maxBinIdxCtx 和 ctxIdxOffset。

9.3.2.1 到 9.3.2.4 节规定了一元(U)二值化过程，舍位一元(TU) 二值化过程，串联一元/ k 阶顺序 Exp-Golomb (UEGk) 二值化过程和固定长度（FL）二值化过程。其他二值化过程见 9.3.2.5 到 9.3.2.7。

除 I 条带以外，9.3.2.5 节规定的语法元素 mb_type 的二值化表示，包含了由前缀和后缀比特串串联给出的二进制码字串。9.3.2.3 节规定 UEGk 二值化表示，可以用到语法元素 mvd_lX (X = 0, 1) 和 coeff_abs_level_minus1，以及 coded_block_pattern 的二值化表示，也包含了由前缀和后缀比特串串联给出的二进制码字串。对于这些二值过程，前缀和后缀分别使用 binIdx 变量索引，更详细的规定见 9.3.3 节。前缀比特串和后缀比特串的两个序列分别是指二值化表示的前缀部分和后缀部分。

表 9-25 中列出了与所有二值化表示或者语法元素的部分二值化表示相关的特定上下文索引偏移(ctxIdxOffset)变量值和 maxBinIdxCtx 变量值。当表 9-25 中给出了这两个变量的值时，那么在上面一行的是前缀部分，而下面一行则是后缀部分。

DecodeBypass 过程和变量 bypassFlag 的使用可以通过如下过程推导：

— 如果ctxIdxOffset没有被赋值，那么在表9-25中相应的二值化表示或者二值化表示部分标为“na”，相应的二值化表示列或者二值化表示的前缀/后缀部分位串中的所有二进制码应该通过调用9.3.3.2.3节规定的DecodeBypass过程进行解码。在这种情况下，bypassFlag等于1，这里bypassFlag用来指示解析该二进制码的值应该使用DecodeBypass过程。

— 否则，对于表9-25中给出的每个binIdx直到特定的MaxBinIdxCtx值，变量ctxIdx的具体值在9.3.3节中规定，bypassFlag等于0。

上下文索引 ctxdex 值的范围从 0 到 459。分配给 ctxIdxOffset 的值规定了分配给语法元素的相应二值化或二值化部分的 ctxIdx 范围的最小值。

ctxIdx = ctxIdxOffset = 276 分配给 syntax element end_of_slice_flag 和 mb_type 二进制码，mb_type 二进制码规定了 I_PCM 宏块类型，更进一步的规定见 9.3.3.1。为解析相应比特流中的二进制码的值，应该应用 9.3.3.2.4 节规定的觉得解码结束的算术解码过程。

注 — I条带中的mb类型二进制码和SI条带中mb_type前缀二进制码和binIdx的值相符，使用相同的ctxIdx。Mb_type前缀的最后的二进制码，与P，SP和B条带中的mb_type的前缀的第一个二进制码应该共享相同的ctxIdx。

表 9-25－语法元素和二进制序列的相关类型，maxBinIdxCtx和ctxIdxOffset

语法元素	二级制序列类型	maxBinIdxCtx	ctxIdxOffset
mb_type (仅 SI 条带)	前缀和后缀如 9.3.2.5 节的规定	前缀: 0 后缀: 6	前缀: 0 后缀: 3
mb_type (仅 I 条带)	按 9.3.2.5 节规定	6	3
mb_skip_flag (仅 P, SP 条带)	FL, cMax=1	0	11
mb_type (P, SP slices only)	前缀和后缀如 9.3.2.5 节的规定	前缀: 2 后缀: 5	前缀: 14 后缀: 17
sub_mb_type (仅 P, SP 条带)	按 9.3.2.5 节规定	2	21
mb_skip_flag (仅 B 条带)	FL, cMax=1	0	24
mb_type (仅 B 条带)	前缀和后缀如 9.3.2.5 节的规定	前缀: 3 后缀: 5	前缀: 27 后缀: 32
sub_mb_type (仅 B 条带)	按 9.3.2.5 节规定	3	36
mvd_l0[ ][ ][0], mvd_l1[ ][ ][0]	前缀和后缀由 UEG3 给出 signedValFlag=1, uCoff=9	前缀: 4 后缀: na	前缀: 40 后缀: na (使用 DecodeBypass)
mvd_l0[ ][ ][1], mvd_l1[ ][ ][1]		前缀: 4 后缀: na	前缀: 47 后缀: na (使用 DecodeBypass)
ref_idx_l0, ref_idx_l1	U	2	54
mb_qp_delta	按 9.3.2.7 节规定	2	60
intra_chroma_pred_mode	TU, cMax=3	1	64
prev_intra4x4_pred_mode_flag, prev_intra8x8_pred_mode_flag	FL, cMax=1	0	68
rem_intra4x4_pred_mode, rem_intra8x8_pred_mode	FL, cMax=7	0	69
mb_field_decoding_flag	FL, cMax=1	0	70
coded_block_pattern	前缀和后缀如 9.3.2.6 节的规定	前缀: 3 后缀: 1	前缀: 73 后缀: 77
coded_block_flag	FL, cMax=1	0	85
significant_coeff_flag (帧编码 ctxBlockCat < 5)	FL, cMax=1	0	105
last_significant_coeff_flag (块，ctxBlockCat < 5)	FL, cMax=1	0	166
coeff_abs_level_minus1 (ctxBlockCat < 5 的块)	前缀和后缀由 UEG0 给出 signedValFlag=0, uCoff=14	前缀: 1 后缀: na	前缀: 227 后缀: na, (使用 DecodeBypass)
coeff_sign_flag	FL, cMax=1	0	na, (使用 DecodeBypass)
end_of_slice_flag	FL, cMax=1	0	276
significant_coeff_flag (域编码块，ctxBlockCat < 5)	FL, cMax=1	0	277

语法元素	二级制序列类型	maxBinIdxCtx	ctxIdxOffset
last_significant_coeff_flag (域编码块，ctxBlockCat < 5)	FL, cMax=1	0	338
transform_size_8x8_flag	FL, cMax=1	0	399
significant_coeff_flag (帧编码块，ctxBlockCat == 5)	FL, cMax=1	0	402
last_significant_coeff_flag 帧编码块， ctxBlockCat == 5)	FL, cMax=1	0	417
coeff_abs_level_minus1 (ctxBlockCat == 5 的块)	前缀和后缀由 UEG0 给出 signedValFlag=0, uCoff=14	前缀: 1 后缀: na	前缀: 426 后缀: na, (使用 DecodeBypass)
significant_coeff_flag (域编码块，ctxBlockCat == 5)	FL, cMax=1	0	436
last_significant_coeff_flag (域编码块，ctxBlockCat == 5)	FL, cMax=1	0	451

9.3.2.1 一元（U）二值化过程

本过程的输入是语法元素的 U 二值化请求。

本过程的输出是语法元素的 U 二值化表示。

如果语法元素带有 synElVal 值，其二进制码串使用长度为 synElVal + 1 的 BinIdx 索引。小于 synElVal 的 binIdx 码等于 0。等于 synElVal 的 binIdx 码也等于 0。

表 9-26 给出了语法元素的一元二进制序列的码串。

表 9-26－一元二值化表示的二进制码串(资料性)

语法元素值	二进制码串					
0 (I_NxN)	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
…						
binIdx	0	1	2	3	4	5

9.3.2.2 舍位一元 (TU) 二值化过程

本过程的输入是语法元素的 TU 二值化请求和 cMax。

本过程的输出是语法元素的 TU 二值化表示。

对于小于 cMax 的语法元素值（无符号整数），调用 9.3.2.1 节规定的 U 二值化过程。对于等于 cMax 的语法元素值，二进制码串的全都码字为 1，长度为 cMax。

注 — 在cMax值等于被解码语法元素最大可能值时，调用 TU二值化过程。

9.3.2.3 串联的一元/k阶顺序哥伦布指数(UEGk) 二值化过程

本过程的输入是语法元素 UEGk 的二值化请求，signedValFlag 和 uCoff。

本过程的输出是语法元素的 UEGk 的二值化表示

UEGk 码串用来连接前缀和后缀码串。9.3.2.2 节规定了语法元素值 synElVal 的前缀部分 Min(uCoff, Abs(synElVal))的 TU 二值化过程，二值化表示的前缀通过调用该过程来实现。

UEGk 码串通过如下过程推导：

— 如果如下条件的一个为真，带有synElVal 值的语法元素的码串仅包含前缀码串。

  — signedValFlag 等于0，前缀比特串不是长度为uCoff、所有比特为1的比特串。

  — signedValFlag 等于1，前缀比特串是包含了单个比特值为0的比特串。

— 否则，语法元素值为synElVal的UEGk前缀部分通过如下伪码过程得到：

```
if( Abs( synElVal ) >= uCoff ) {
  sufS = Abs( synElVal ) − uCoff
  stopLoop = 0
  do {
      if( sufS >= ( 1 << k ) ) {
          put( 1 )
          sufS = sufS − ( 1<<k )
          k++
      } else {
          put( 0 )
          while( k− − )
              put( ( sufS >> k ) & 0x01 )
          stopLoop = 1
      }
    } while( !stopLoop )
  }
  if( signedValFlag && synElVal != 0)
      if( synElVal > 0 )
          put( 0 )
      else
          put( 1 )
```

注 — k阶指数哥伦布编码（EGk）规定使用相反意思的1和0代表9.1节规定的一元0阶指数哥伦布编码。

9.3.2.4 固定长度 (FL)二值化过程

本过程的输入是语法元素的 FL 二值化请求和 cMax。

本过程的输出是语法元素的 FL 二值化 表示。

FL 二值化通过使用语法元素值的 fixedLength 比特、无符号整数码串构建，这里 fixedLength = Ceil(Log2(cMax + 1))。FL 的二值化表示的二进制码索引可以是在 binIdx = 0 时和最小首位相关，随着 binIdx 值增加到最大首位比特。

9.3.2.5 宏块类型和子宏块类型的二值化过程

本过程的输出是语法元素 mb_type 或者 sub_mb_type。

本过程的输入是语法元素的二值化表示。

I 条带中的宏块类型的二值化方法见表 9-27。

对于 SI 条带中的宏块类型，二值化表示包含了如下的前缀和后缀串联得到的码串。

前缀码串由一个单比特构成，通过 b_0 = ((mb_type == SI) ? 0 : 1)规定。对于 b_0 等于 0 的语法元素值，码串仅包含前缀比特串。对于 b_0 等于 1 的语法元素值，二值化表示由前缀 b_0 和后缀比特串串联组成，后缀值在表 9-27 中规定，通过 SI 条带中的 mb_type 减 1 得到的值索引。

表 9-27—I条带中的宏块类型二值化

mb_type 的值（名称）	二进制码串						
0 (I_4x4)	0						
1 (I_16x16_0_0_0)	1	0	0	0	0	0	
2 (I_16x16_1_0_0)	1	0	0	0	0	1	
3 (I_16x16_2_0_0)	1	0	0	0	1	0	
4 (I_16x16_3_0_0)	1	0	0	0	1	1	
5 (I_16x16_0_1_0)	1	0	0	1	0	0	0
6 (I_16x16_1_1_0)	1	0	0	1	0	0	1
7 (I_16x16_2_1_0)	1	0	0	1	0	1	0
8 (I_16x16_3_1_0)	1	0	0	1	0	1	1
9 (I_16x16_0_2_0)	1	0	0	1	1	0	0
10 (I_16x16_1_2_0)	1	0	0	1	1	0	1
11 (I_16x16_2_2_0)	1	0	0	1	1	1	0
12 (I_16x16_3_2_0)	1	0	0	1	1	1	1
13 (I_16x16_0_0_1)	1	0	1	0	0	0	
14 (I_16x16_1_0_1)	1	0	1	0	0	1	
15 (I_16x16_2_0_1)	1	0	1	0	1	0	
16 (I_16x16_3_0_1)	1	0	1	0	1	1	
17 (I_16x16_0_1_1)	1	0	1	1	0	0	0
18 (I_16x16_1_1_1)	1	0	1	1	0	0	1
19 (I_16x16_2_1_1)	1	0	1	1	0	1	0
20 (I_16x16_3_1_1)	1	0	1	1	0	1	1
21 (I_16x16_0_2_1)	1	0	1	1	1	0	0
22 (I_16x16_1_2_1)	1	0	1	1	1	0	1
23 (I_16x16_2_2_1)	1	0	1	1	1	1	0
24 (I_16x16_3_2_1)	1	0	1	1	1	1	1
25 (I_PCM)	1	1					
binIdx	0	1	2	3	4	5	6

P 和 SP 条带中的 P 宏块类型和 B 条带的 B 宏块的二进制化方案在表 9-28 中规定。

在 P 和 SP 条带中的 I 宏块类型二进制码串，对应 mb_type 的值从 5 到 30，包含了前缀和后缀的串联，这里前缀包含了表 9-28 中规定的值为 1 的单个比特，后缀在 9-27 中规定，通过 mb_type 减 5 所得的值索引。

mb_type 不允许等于 4 (P_8x8ref0) 。

对于 B 条带中的 I 宏块类型， (mb_type 值从 23 到 48)，二值化表示由表 9-28 规定的前缀和表 9-27 规定的后缀串联而成，通过 mb_type 减 23 所得的值索引。

表 9-28－在P，SP和B条带中的宏块二值化

条带类型	mb_type 值（类型）	二进制码串						
P, SP 条带	0 (P_L0_16x16)	0	0	0				
	1 (P_L0_L0_16x8)	0	1	1				
	2 (P_L0_L0_8x16)	0	1	0				
	3 (P_8x8)	0	0	1				
	4 (P_8x8ref0)	na						
	5 to 30 (Intra, 只有前缀)	1						
B 条带	0 (B_Direct_16x16)	0						
	1 (B_L0_16x16)	1	0	0				
	2 (B_L1_16x16)	1	0	1				
	3 (B_Bi_16x16)	1	1	0	0	0	0	
	4 (B_L0_L0_16x8)	1	1	0	0	0	1	
	5 (B_L0_L0_8x16)	1	1	0	0	1	0	
	6 (B_L1_L1_16x8)	1	1	0	0	1	1	
	7 (B_L1_L1_8x16)	1	1	0	1	0	0	
	8 (B_L0_L1_16x8)	1	1	0	1	0	1	
	9 (B_L0_L1_8x16)	1	1	0	1	1	0	
	10 (B_L1_L0_16x8)	1	1	0	1	1	1	
	11 (B_L1_L0_8x16)	1	1	1	1	1	0	
	12 (B_L0_Bi_16x8)	1	1	1	0	0	0	0
	13 (B_L0_Bi_8x16)	1	1	1	0	0	0	1
	14 (B_L1_Bi_16x8)	1	1	1	0	0	1	0
	15 (B_L1_Bi_8x16)	1	1	1	0	0	1	1
	16 (B_Bi_L0_16x8)	1	1	1	0	1	0	0
	17 (B_Bi_L0_8x16)	1	1	1	0	1	0	1
	18 (B_Bi_L1_16x8)	1	1	1	0	1	1	0
	19 (B_Bi_L1_8x16)	1	1	1	0	1	1	1
	20 (B_Bi_Bi_16x8)	1	1	1	1	0	0	0
	21 (B_Bi_Bi_8x16)	1	1	1	1	0	0	1
	22 (B_8x8)	1	1	1	1	1	1	
	23 to 48 (Intra, 只有前缀)	1	1	1	1	0	1	
binIdx		0	1	2	3	4	5	6

P，SP 和 B 条带中的 sub_mb_type 二进制序列在表 9-29 中给出。

表 9-29－P，SP和B条带中的sub_mb_type二进制序列

条带类型	sub_mb_type 值（名称）	二进制码串					
P, SP 条带	0 (P_L0_8x8)	1					
	1 (P_L0_8x4)	0	0				
	2 (P_L0_4x8)	0	1	1			
	3 (P_L0_4x4)	0	1	0			
B 条带	0 (B_Direct_8x8)	0					
	1 (B_L0_8x8)	1	0	0			
	2 (B_L1_8x8)	1	0	1			
	3 (B_Bi_8x8)	1	1	0	0	0	
	4 (B_L0_8x4)	1	1	0	0	1	
	5 (B_L0_4x8)	1	1	0	1	0	
	6 (B_L1_8x4)	1	1	0	1	1	
	7 (B_L1_4x8)	1	1	1	0	0	0
	8 (B_Bi_8x4)	1	1	1	0	0	1
	9 (B_Bi_4x8)	1	1	1	0	1	0
	10 (B_L0_4x4)	1	1	1	0	1	1
	11 (B_L1_4x4)	1	1	1	1	0	
	12 (B_Bi_4x4)	1	1	1	1	1	
binIdx		0	1	2	3	4	5

9.3.2.6 编码块模式的二值化过程

本过程的输入是语法元素 coded_block_pattern 的二值化请求。

本过程的输出是语法元素的二值化表示。

coded_block_pattern 的二值化表示由前缀和后缀部分（如果存在）组成。二值化表示的前缀部分由 cMax = 15 的 CodedBlockPatternLuma 的 FL 二值化表示给出。当 chroma_format_idc 不等于 0 时，后缀部分存在，包含了 cMax = 2 的 CodedBlockPatternChroma 的 TU 二值化表示。语法元素 coded_block_pattern 值和 CodedBlockPatternLuma 值之间的关系在 7.4.5 节中规定。

9.3.2.7 mb_qp_delta的二值化表示

本过程的输入是语法元素 mb_qp_delta 的二值化请求。

本过程的输出是语法元素的二值化表示。

mb_qp_delta 的二进制码串通过语法元素 mb_qp_delta 的映射值的 U 二值化表示推导得出，在有符号 mb_qp_delta 值和它的映射值之间的分配原则见表 9-3。

9.3.3 解码处理流程

本过程的输入是 9.3.2 中定义的语法请求的二进制串、maxBinIdxCtx、bypassFlag 和 ctxIdxOffset。

本过程的输出是语法元素的值。

本过程规范了每一个语法元素的二进制串是如何解析的。

当解析每一比特时，得到的比特串与语法元素的所有二进制串进行比较，并应用如下规则：

— 如果比特串与其中一个二进制串相同，则二进制串对应的值就是输出值。

— 否则解析下一比特。

每当解析一位时，变量 binIdx 增 1；当解析第一个二进制位时，binIdx 设为 0。

当语法元素对于的二进制串包含前缀和后缀部分时，变量 binIdx 在二进制串的每一部分的第一比特（前缀和后缀）都被置位 0。这种情况下，解析前缀串之后，如 9.3.2.3 和 9.3.2.5 所规范，对后缀的解析依赖于前缀解析的结果。注意，对于语法元素 coded_block_pattern 的二进制串，如 9.3.2.6 所定义，后缀比特串的出现与长度为 4 的前缀比特无关。

对于变量 bypassFlag，应用如下规则：

— 如果bypassFlag为1，对于比特流中的值应使用9.3.3.2.3中定义的bypass解码过程。

— 否则（bypassFlag值为0），每一个二进制符号通过下面顺序的两步来解析：

1．给定 binIdx、maxBinIdxCtx 和 ctxIdxOffset，由 9.3.3.1 中定义的过程导出 ctxIdx。

2．给定 ctxIdx，通过 9.3.3.2 中定义的过程解码比特流。

9.3.3.1 ctxIdx的推导过程

本过程的输入是 binIdx、maxBinIdxCtx 和 ctxIdxOffset。

本过程的输出是 ctxIdx。

表 9-30 给出了对于所有 ctxIdxOffset，binIdx 对应的 ctxIdx 增量（ctxIdxInc），除了那些与语法元素 coded_block_flag，significant_coeff_flag，last_significant_coeff_flag 和 coeff_abs_level_minus1 相关的值。

对于一个特定 binIdx 要得到 ctxIdx，需要首先决定给出的二进制串中的 ctxIdxOffset。CtxIdx 通过下面的过程得到：

— 如果ctxIdxOffset在表9-30中，binIdx对应的ctxIdx是ctxIdxOffset和ctxIdxInc的和，可以在表9-30中找到。如果表9-30列出了多于一个的binIdx值， binIdx对应的ctxIdx的值则在括号中的章节号中进一步给出。

— 否则（ctxIdxOffset没有在表9-30中列出），则ctxIdx为下列变量之和：ctxIdxOffset和表9-31中的ctxIdxBlockCatOffset（ctxBlockCat）和ctxIdxInc（ctxBlockCat）。9.3.3.1.3给出了使用哪一个ctxBlockCat。9.3.3.1.1.9 给出了 coded_block_flag 语法元素为时对 ctxIdxInc 的赋值；9.3.3.1.3 给出了语法元素为 significant_coeff_flag、last_significant_coeff_flag和 coeff_abs_level_minus1时对ctxIdxInc的赋值。

所有 binIdx 大于 BinIdxCtx 的二进制值应使用被赋值到 maxBinIdxCtx 的 ctxIdx 来解析。

在表 9-30 中所有标记为 na 的表项对应与不存在对应 ctxIdxOffset 的 binIdx 的值。

ctxIdx＝276 被用于 mb_type 的 binIdx 用于指示 I_PCM 模式。当从比特流中解析到这个值时，应当使用 9.3.3.2.4 中规定的算法。

表9-30－除了关于语法元素coded_block_flag、 significant_coeff_flag、last_significant_coeff_flag和coeff_abs_level_minus1之外，对于所有的ctxIdxOffset，binIdx对应的ctxIdxInc

ctxIdxOffset	binIdx						
	0	**1**	**2**	**3**	**4**	**5**	**>= 6**
0	0,1,2 (9.3.3.1.1.3 节)	na	na	na	na	na	Na
3	0,1,2 (9.3.3.1.1.3 节)	ctxIdx=276	3	4	5,6 (9.3.3.1.2 节)	6,7 (9.3.3.1.2 节)	7
11	0,1,2 (9.3.3.1.1.1 节)	na	na	na	na	na	Na
14	0	1	2,3 (9.3.3.1.2 节)	na	na	na	Na
17	0	ctxIdx=276	1	2	2,3 (9.3.3.1.2 节)	3	3
21	0	1	2	na	na	na	Na
24	0,1,2 (9.3.3.1.1.1 节)	na	na	na	na	na	Na
27	0,1,2 (9.3.3.1.1.3 节)	3	4,5 (9.3.3.1.2 节)	5	5	5	5
32	0	ctxIdx=276	1	2	2,3 (9.3.3.1.2 节)	3	3
36	0	1	2,3 (9.3.3.1.2 节)	3	3	3	Na
40	0,1,2 (9.3.3.1.1.7 节)	3	4	5	6	6	6
47	0,1,2 (9.3.3.1.1.7 节)	3	4	5	6	6	6
54	0,1,2,3 (9.3.3.1.1.6 节)	4	5	5	5	5	5
60	0,1 (9.3.3.1.1.5 节)	2	3	3	3	3	3
64	0,1,2 (9.3.3.1.1.8 节)	3	3	na	na	na	Na
68	0	na	na	na	na	na	Na
69	0	0	0	na	na	na	Na
70	0,1,2 (9.3.3.1.1.2 节)	na	na	na	na	na	Na
73	0,1,2,3 (9.3.3.1.1.4 节)	0,1,2,3 (9.3.3.1.1.4 节)	0,1,2,3 (9.3.3.1.1.4 节)	0,1,2,3 (9.3.3.1.1.4 节)	na	na	Na
77	0,1,2,3 (9.3.3.1.1.4 节)	4,5,6,7 (9.3.3.1.1.4 节)	na	na	na	na	Na
276	0	na	na	na	na	na	Na
399	0,1,2 (9.3.3.1.1.10 节)	na	na	na	na	na	Na

表 9-31 给出了语法元素 coded_block_flag、significant_coeff_flag、last_significant_coeff_flag 和 coeff_abs_level_minus1 的 ctxBlockCat 的 ctxIdxBlockCatOffset 值。CtxBlockCat 定义在表 9-33。

表9-31－语法元素coded_block_flag、significant_coeff_flag、last_significant_coeff_flag和coeff_abs_level_minus1的ctxBlockCat的ctxIdxBlockCatOffset值

语法元素	ctxBlockCat (见表 9-33)					
	0	**1**	**2**	**3**	**4**	**5**
coded_block_flag	0	4	8	12	16	na
significant_coeff_flag	0	15	29	44	47	0
last_significant_coeff_flag	0	15	29	44	47	0
coeff_abs_level_minus1	0	10	20	30	39	0

9.3.3.1.1 使用相邻语法元素的ctxIdxInc的赋值

9.3.3.1.1.1 给出了语法元素 mb_skip_flag 的 ctxIdxInc 的推导过程。

9.3.3.1.1.2 给出了语法元素 mb_field_decoding_flag 的 ctxIdxInc 的推导过程。

9.3.3.1.1.3 给出了语法元素 mb_type 的 ctxIdxInc 的推导过程。

9.3.3.1.1.4 给出了语法元素 coded_block_pattern 的 ctxIdxInc 的推导过程。

9.3.3.1.1.5 给出了语法元素 mb_qp_delta 的 ctxIdxInc 的推导过程。

9.3.3.1.1.6 给出了语法元素 ref_idx_l0 和 ref_idx_l1 的 ctxIdxInc 的推导过程。

9.3.3.1.1.7 给出了语法元素 mvd_l0 和 mvd_l1 的 ctxIdxInc 的推导过程。

9.3.3.1.1.8 给出了语法元素 intra_chroma_pred_mode 的 ctxIdxInc 的推导过程。

9.3.3.1.1.9 给出了语法元素 coded_block_flag 的 ctxIdxInc 的推导过程。

9.3.3.1.1.10 给出了语法元素 transform_size_8x8_flag 的 ctxIdxInc 的推导过程。

9.3.3.1.1.1 语法元素mb_skip_flag的ctxIdxInc的推导过程

本过程的输出是 ctxIdxInc。

当 MbaffFrameFlag 为 1 且当前宏块对中高地址宏块地址为 2* (CurrMbAddr / 2)的 mb_field_decoding_flag 还没有被解码时，则使用 7.4.4 中定义的语法元素 mb_field_decoding_flag 导出规则。

调用 6.4.8.1 中定义的相邻宏块推导规则，其输出被赋值到 mbAddrA 和 mbAddrB。

变量 condTermFlagN (N 为 A 或 B)的值推导如下：

— 如果mbAddrN不可用或宏块mbAddrN的mb_skip_flag等于1，则condTermFlagN被置为0；

— 否则condTermflagN置为1。

变量 ctxIdxInc 为：

ctxIdxInc = condTermFlagA + condTermFlagB (9-1)

9.3.3.1.1.2 语法元素mb_skip_flag的ctxIdxInc的推导过程

本过程的输出时 ctxIdxInc。

相邻宏块地址和它们在 MBAFF 帧中的可用性（见 6.4.7）的推导过程被调用并且其输出被赋予 mbAddrA 和 mbAddrB。

当宏块 mbAddrN 和 mbAddrN+1 都有 mb_type 等于 P_Skip 或 B_Skip 时，定义在 7.4.4 中的语法元素 mb_field_decoding_flag 的推断结果应应用到 mbAddrN。

变量 condTermFlagN（N 为 A 或 B）推导如下：

— 如果任何下列条件为真，condTermFlagN应置为0，

  — MbAddrN不可用

  — 宏块mbAddrN是一个帧宏块

— 否则，condTermFlagN置为1。

变量 ctxIdxInc 为：

ctxIdxInc = condTermFlagA + condTermFlagB (9-2)

9.3.3.1.1.3 语法元素mb_type的ctxIdxInx的推导过程

本过程的输出是 ctxIdxOffset。

本过程的输出是 ctxIdxInc。

6.4.8.1 中的相邻宏块的推导过程被调用并且其输出被赋予 mbAddrA 和 mbAddrB。

变量 condTermFlagN（N 为 A 或 B）推导如下：

— 如果以下任何条件为真，则condTermFlagN置为0

  — mbAddrN不可用

  — ctxIdxOffset等于0且宏块mbAddrN的mb_type等于SI

  — ctxIdxOffset等于3且宏块mbAddrN的mb_type等于I_NxN

    — ctxIdxOffset等于27且宏块mbAddrN的mb_type等于P_Skip、B_Skip或B_Direct_16×16

— 否则，condTermflagN置为1。

变量 ctxIdxInc 为：

ctxIdxInc = condTermFlagA + condTermFlagB (9-3)

9.3.3.1.1.4 语法元素coded_block_pattern的ctxIdxInc的推导过程

本过程的输入是 ctxIdxOffset 和 binIdx。

本过程的输出是 ctxIdxInc。

根据变量 ctxIdxOffset 不同的值，应用下列规则：

— 如果ctxIdxOffset等于73，则应用下列规则，

  — 使用luma8x8BlkIdx＝binIdx做为输入，调用在6.4.8.2节定义的相邻8x8亮度块的推导过程，其输出赋值给mbAddrA，mbAddrB，luma8x8BlkIdxA和luma8x8BlkIdxB。

  — 变量condTermFlagN（N为A或B）的导出过程为：

  — 如果任何下列条件为真，则condTermFlagN置为0

    — mbAddrN不可用

    — 宏块mbAddrN的mb_type等于I_PCM

    — 宏块mbAddrN不是当前宏块CurrMbAddr并且宏块mbAddrN没有mb_type等于P_Skip或B_Skip，并且对于宏块mbAddrN的CodedBlockPatternLuma的值和((CodedBlockPatternLuma >> luma8x8BlkIdxN) & 1)不等于0

— 宏块mbAddrN是当前宏块CurrMbAddr并且前一个解码的k= luma8x8BlkIdxN的coded_block_pattern的二进制值b_k不等于0。

  — 否则，confTermFlagN置为1。

— 变量ctxIdxInc定义为：

ctxIdxInc = condTermFlagA + 2 * condTermFlagB (9-4)

— 否则 (ctxIdxOffset 不等于 77), 则

— 调用6.4.8.1中的相邻宏块的推导过程，其输出赋值到mbAddrA和mbAddrB.

— 变量condTermFlagN（N为A或B）的导出过程为：

— 如果mbAddrN可用且宏块mbAddrN的mb_type等于I_PCM，condTermFlagN置为1

— 否则，如果任何下列条件为真，condTermFlagN置为0

— mbAddrN不可用或宏块mbAddrN具有mb_type等于P_Skip或B_Skip

— binIdx等于0且宏块mbAddrN的codedBlockPatternChroma等于0。

— binIdx等于1且宏块mbAddrN的。CodedBlockPatternChroma不等于2

— 否则，condTermFlagN置为1。

— 变量ctxIdxInc为：

ctxIdxInc = condTermFlagA + 2 * condTermFlagB + ((binIdx == 1) ? 4 : 0) (9-5)

注 — 当宏块使用Intra_16x16预测模式时，宏块的CodedBlockPatternLuma和CodedBlockPatternChroma值应从表7-11的mb_type中得到。

9.3.3.1.1.5 语法元素mb_qp_delta的ctxIdxInc的推导过程

本过程的输出为 ctxIdxInc。

令 prevMbAddr 为在解码顺序中当前宏块的前一宏块地址，当当前宏块为条带中的第一个宏块时，prevMbAddr 被标记为不可用。

令变量 ctxIdxInc 推导如下。

— 如果下列条件为真，ctxIdxInc被置为0

— prevMbAddr不可用或宏块prevMbAddr具有mb_type等于P_Skip或B_Skip

— 宏块的mb_type等于I_PCM

— 宏块 prevMbAddr 不是使用 Intra_16x16 预测模式编码的，且宏块 prevMbAddr 的CodedBlockPatternLuma和CodedBlockPatternChroma等于0

— 宏块prevMbAddr的mb_qp_delta等于0

— 否则，ctxIdxInc置为1。

9.3.3.1.1.6 语法元素ref_idx_l0和ref_idx_l1的ctxIdxInc的推导过程

本过程的输入为 mbPartIdx。

本过程的输出为 ctxIdxInc。

本节中 ref_idx_lX 和 Pred_LX 解释为：

— 如果本过程被调用用于ref_idx_l0的导出过程，ref_idx_lX为ref_idx_l0且Pred_LX为Pred_L0

— 否则（本过程用于ref_idx_l1的导出），ref_idx_lX为ref_idx_l1且Pred_LX为Pred_L1

令 currSubMbType 等于 sub_mb_type[mbPartIdx]。

调用 6.4.8.5 中的相邻划分的推导过程时，输入为 mbPartIdx，currSubMbType 和 subMbPartIdx = 0，输出被赋值到 mbAddrA\mbPartIdxA 和 mbAddrB\mbPartIdxB。

对于宏块 mbAddrN，ref_idx_lX[mbPartIdxN] (N 为 A 或 B)规定语法元素，令变量 refIdxZeroFlagN 为

— 如果MbaffFrameFlag等于1，当前宏块是一个帧宏块，且宏块mbAddrN是场宏块，则

refIdxZeroFlagN = ((ref_idx_lX[mbPartIdxN] > 1) ? 0 : 1) (9-6)

— 否则,

refIdxZeroFlagN = ((ref_idx_lX[mbPartIdxN] > 0) ? 0 : 1) (9-7)

令变量 predModeEqualFlag 如下：

— 如果宏块mbAddrN的mb_type等于P_8x8或B_8x8，则，

  — 如果 SubMbPredMode(sub_mb_type[mbPartIdxN]) 不等于 Pred_LX 且不等于 BiPred，则 predModeEqualFlag置为0，其中sub_mb_type指定了宏块mbAddrN的语法元素。

— 否则，predModeEqualFlag置为1

— 否则，下列适用。

  — 如果 MbPartPredMode(mb_type, mbPartIdxN) 不等于 Pred_LX 且不等于 BiPred，则 predModeEqualFlag置为0，其中sub_mb_type指定了宏块mbAddrN的语法元素。

  — 否则，predModeEqualFlag置为1

令变量 condTermFlagN(N 为 A 或 B)为：

— 如果任何下列条件为真，则condTermFlagN置为0

  — mbAddrN不可用

  — 宏块mbAddrN的mb_type等于P_Skip或B_Skip

  — 宏块mbAddrN编码为帧内预测模式

  — predModeEqualFlag等于0

  — refIdxZeroFlagN等于1

— 否则，condTermFlagN置为1

变量 ctxIdxInc 为：

ctxIdxInc = condTermFlagA + 2 * condTermFlagB (9-8)

9.3.3.1.1.7 语法元素mvd_l0和mvd_l1的ctxIdxInc的推导过程

本过程的输入为 mbPartIdx，subMbPartIdx 和 ctxIdxOffset。

本过程的输出为 ctxIdxInc。

在本节中 mvd_lX 和 Pred_LX 的含义如下：

— 如果为了导出mvc_l0而调用本过程，mcd_1X的含义为mvd_l0，Pred_LX的含义为Pred_L0。

— 否则，（如果为了导出mvc_l1而调用本过程），mcd_1X的含义为mvd_l1，Pred_LX的含义为Pred_L1。

令 currSubMbType 等于 sub_mb_type[mbPartIdx]。

调用 6.4.8.5 中的相邻划分的推导过程时，输入为 mbPartIdx，currSubMbType 和 subMbPartIdx = 0，输出被赋值到 mbAddrA\mbPartIdxA 和 mbAddrB\mbPartIdxB。

令变量 compIdx 为：

— 如果ctxIdxOffset等于40，则compIdx等于0

— 否则，（ctxIdxOffset等于47），则compIdx等于1

令变量 predModeEqualFlag 为：

— 如果宏块mbAddrN的mb_type等于P_8x8或B_8x8，则

— 如果 SubMbPredMode(sub_mb_type[mbPartIdxN]) 不等于 Pred_LX 且不等于 BiPred，则predModeEqualFlag置为0，其中sub_mb_type为宏块mbAddrN的语法元素。

— 否则，predModeEqualFlag置为1。

— 否则，下列适用。

— 如果 MbPartPredMode(mb_type, mbPartIdxN) 不等于 Pred_LX 且不等于 BiPred，则predModeEqualFlag置为0，其中mb_type为宏块mbAddrN的语法元素。

— 否则，predModeEqualFlag置为1。

令变量 absMvdCompN(N 为 A 或 B)为：

— 如果任何下列条件为真，则absMvdCompN置为0

— mbAddrN不可用

— 宏块mbAddrN的mb_type等于P_Skip或B_Skip

— 宏块mbAddrN编码为帧内预测模式

— predModeEqualFlag等于0

— 否则，下列适用。

— 如果compIdx等于1，MbaffFrameFlag等于1，当前宏块是一个帧宏块，且宏块mbAddrN是一个场宏块

$$absMvdCompN = Abs(mvd\_lX[mbPartIdxN][subMbPartIdxN][compIdx]) * 2 \quad (9\text{-}9)$$

— 否则，如果compIdx等于1，MbaffFrameFlag等于1，当前宏块是一个场宏块，且宏块mbAddrN是一个帧宏块

$$absMvdCompN = Abs(mvd\_lX[mbPartIdxN][subMbPartIdxN][compIdx]) / 2 \quad (9\text{-}10)$$

— 否则,

$$absMvdCompN = Abs(mvd\_lX[mbPartIdxN][subMbPartIdxN][compIdx]) \quad (9\text{-}11)$$

变量 ctxIdxInc 为：

— 如果(absMvdCompA + absMvdCompB)小于3，ctxIdxInc置为0。

— 否则，如果(absMvdCompA + absMvdCompB)大于32，ctxIdxInc置为2。

— 否则，如果(absMvdCompA + absMvdCompB)在3到32之间，ctxIdxInc置为1。

9.3.3.1.1.8 语法元素intra_chroma_pred_mode的ctxIdxInc的推导过程

本过程的输出是 ctxIdxInc。

6.4.8.1 中定义的相邻宏块的导出过程被调用且输出赋值到 mbAddrA 和 mbAddrB。

令变量 condTermFlagN（N 为 A 或 B）为：

— 如果任何下列条件为真，则condTermFlagN置为0

  — mbAddrN不可用

  — 宏块mbAddrN的mb_type等于I_PCM

  — 宏块mbAddrN编码为帧内预测模式

  — 宏块mbAddrN的intra_chroma_pred_mode为0

— 否则condTermFlagN置为1

变量 ctxIdxInc 为：

ctxIdxInc = condTermFlagA + condTermFlagB (9-12)

9.3.3.1.1.9 语法元素coded_block_flag的ctxIdxInc的推导过程

本过程的输入是 ctxBlockCat 和：

— 如果ctxBlockCat等于0，没有其它输入

— 否则，如果ctxBlockCat等于1或2，则是luma4x4blkIdx

— 否则，如果ctxBlockCat等于3，则是色度元素索引iCbCr

— 否则（ctxBlockCat等于4），则是chroma4x4BlkIdx和亮度元素索引iCbCr

本过程的输出是 ctxIdxInc（ctxBlockCat）。

令变量 transBlockN（N 为 A 或 B）为：

— 如果ctxBlockCat等于0，下列适用。

  — 在6.4.8.1中规定的相邻宏块的推导过程被调用且输出被赋值的mbAddrN（N为A或B）

  — 变量transBlockN的推导过程为：

    — 如果mbAddrN可用且宏块mbAddrN编码为Intra_16x16预测模式，则宏块mbAddrN的亮度直流块赋值到transBlockN

    — 否则，transBlockN被标记为不可用。

— 否则，如果ctxBlockCat等于1或2，下列适用。

  — 6.4.8.3中规定的相邻4x4亮度块的导出过程被调用，其输入为luma4x4BlkIdx，其输出被赋值到mbAddrN和luma4x4BlkIdxN（N为A或B）。

  — 变量transBlockN的推导过程为：

    — 如果mbAddrN可用，宏块mbAddrN没有被跳过，宏块mbAddrN的mb_type不等于I_PCM, 宏块mbAddrN的((CodedBlockPatternLuma >> (luma4x4BlkIdxN >>2)) & 1)不等于0，宏块mbAddrN的transform_size_8x8标记等于0，则索引为luma4x4BlkIdxN的宏块mbAddrN的4x4亮度块被赋值到transBlockN。

    — 否则，如果mbAddrN可用，宏块mbAddrN没有被跳过，宏块mbAddrN的((CodedBlockPatternLuma >> (luma4x4BlkIdxN >>2)) & 1) 不等于0，宏块mbAddrN的transform_size_8x8_flag等于1，则宏块mbAddrN的索引为(luma4x4BlkIdxN >> 2)的8x8亮度块赋值给transBlockN。

    — 否则，transBlockN被标记为不可用

  — 否则，如果ctxBlockCat等于3，下列适用。

    — 6.4.8.1中规定的相邻宏块的推导过程被调用且输出被赋值到mbAddrN（N为A或B）；

    — 变量transBlockN推导过程为：

— 如果mbAddrN可用，宏块mbAddrN没有被忽略，宏块mbAddrN的mb_type不等于I_PCM，宏块mbAddrN中的CodedBlockPatternChroma不等于0，则宏块mbAddrN的色度元素iCbCr中的色度直流块被赋值到transBlockN。

— 否则，transBlockN被标记为不可用。

— 否则（ctxBlockCat等于4），下列适用。

— 6.4.8.4中定义的相邻4x4色度块推导过程被调用，其输入为chroma4x4BlkIdx，其输出被赋值到mbAddrN，chroma4x4BlkIdxN（N为A或B）

— 变量transBlockN推导过程为

— 如果mbAddrN可用，宏块mbAddrN没有mb_type等于P_Skip、B_Skip或I_PCM，宏块mbAddrN的mb_type不等于I_PCM，且宏块mbAddrN的CodedBlockPatternChroma等于2，则宏块mbAddrN的色度元素的chroma4x4BlkIdxN的4x4色度块被赋值到transBlockN。

— 否则，transBlockN标记为不可用。

设变量condTermFlagN（N为A或B）为：

— 如果任何下列条件为真，则condTermFlagN被置为0

— mbAddrN不可用且当前宏块编码为帧间预测模式

— mbAddrN可用且transBlockN不可用且宏块的mb_type不等于I_PCM

— 当前宏块编码为帧内预测模式，constrained_intra_pred_flag等于1，宏块mbAddrN可用且编码为帧间预测模式，且使用条带数据划分（nal_unit_type范围为[2,4]）

— 否则，如果任何下列条件为真，condTermFlagN置为1

— mbAddrN不可用且当前宏块编码为帧内预测模式

— 宏块mbAddrN的mb_type等于I_PCM

— 否则，condTermN等于用于宏块mbAddrN解码的变换块transBlockN的coded_block_flag值。

变量 ctxIdxInc(ctxBlockCat)为：

ctxIdxInc(ctxBlockCat) = condTermFlagA + 2 * condTermFlagB (9-13)

9.3.3.1.1.10 语法元素transform_size_8x8_flag的ctxIdxInc的推导过程

本过程的输出是 ctxIdxInc。

6.4.8.1 中定义的相邻宏块的推导过程被调用且输出赋值到 mbAddrA 和 mbAddrB。

设变量 condTermFlagN（N 为 A 或 B）为：

— 如果任何下列条件为真，condTermFlagN置为0

— mbAddrN不可用

— 宏块mbAddrN的transform_size_8x8_flag等于0

— 否则，condTermFlagN置为1。

变量 ctxInxInc 为：

ctxIdxInc = condTermFlagA + condTermFlagB (9-14)

9.3.3.1.2 使用前一二进制解码值的ctxIdxInc的赋值过程

本过程的输入是 ctxIdxOffset 和 binIdx。

本过程的输出是 ctxIdxInc。

表 9-32 包含对于 ctxIdxOffset 和 binIdx 给定的值对应的 ctxIdxInc 的值。

对于 ctxIdxOffset 和 binIdx 的每一个值，通过使用某些前面解码的二进制值($b_0, b_1, b_2,\ldots, b_k$)可以得到 ctxIdxInc 的值，其中索引 k 小于 binIdx 的值。

表 9-32－ctxIdxOffset和binIdx到ctxIdxInc的对应值

CtxIdxOffset 的值	binIdx	ctxIdxInc
3	4	(b_3 != 0) ? 5: 6
	5	(b_3 != 0) ? 6: 7
14	2	(b_1 != 1) ? 2: 3
17	4	(b_3 != 0) ? 2: 3
27	2	(b_1 != 0) ? 4: 5
32	4	(b_3 != 0) ? 2: 3
36	2	(b_1 != 0) ? 2: 3

9.3.3.1.3 语法元素significant_coeff_flag，last_significant_coeff_flag和coeff_abs_level_minus1的ctxIdxInc的赋值过程

本过程的输入是 ctxIdxOffset 和 binIdx。

本过程的输出是 ctIdxInc。

语法元素 significant_coeff_flag，last_significant_coeff_flag 和 coeff_abs_level_minus1 以及对于依赖于不同块分类的 coded_block_flag 的 ctxIdxInc 的赋值过程是由变量 ctxBlockCat 标记。这些块分类在表 9-33 中给出。

表 9-33－不同块ctxBlockCat的规定

块描述	maxNumCoeff	ctxBlockCat
亮度 DC 变换系数等级块（即，7.4.5.3 中描述的 Intra16x16DCLevel 列表）	16	0
亮度 AC 变换系数等级块（即，在 7.4.5.3 中描述的 Intra16x16ACLevel[i]列表）	15	1
16 亮度变换系数等级块（即，在 7.4.5.3 中描述的 LumaLevel[i]列表）	16	2
色度 DC 变换系数等级块	4 * NumC8x8	3
色度 AC 变换系数等级块	15	4
64 亮度变换系数等级块（即，在 7.4.5.3 中描述的 LumaLevel8x8[i]列表）	64	5

设变量 levelListIdx 为 7.4.5.3 中描述的变换系数等级列表的索引。

当 ctxBlockCat<5 和 ctxBlockCat!=3 时，对于块中的语法元素 significant_coeff_flag 和 last_significant_coeff_flag，变量 ctxIdxInc 为：

ctxIdxInc = levelListIdx (9-15)

其中 levelListIdx 范围为[0,maxNumCoeff－2]

当 ctxBlockCat＝＝3，对于块中的语法元素 significant_coeff_flag 和 last_significant_coeff_flag，变量 ctxIdxInc 为：

ctxIdxInc = Min(levelListIdx / NumC8x8, 2) (9-16)

其中，levelListIdx 范围为[0,4×NumC8x8-2]

当 ctxBlockCat＝＝5，对于 8x8 亮度块中的语法元素 significant_coeff_flag 和 last_significant_coeff_flag，表 9-34 中规定了对于给定 levelListIdx 值时的 ctxIdxInc 的值，其中 levelListIdx 范围为[0,62]。

表 9-34－当ctxBlockCat ＝＝ 5时扫描位置到ctxIdxInc的映射

levelListIdx	significant_coeff_flag 的 ctxIdxInc (帧编码的宏块)	significant_coeff_flag 的 ctxIdxInc (场编码的宏块)	last_significant_coeff_flag 的 ctxIdxInc	levelListIdx	significant_coeff_flag 的 ctxIdxInc (帧编码的宏块)	significant_coeff_flag 的 ctxIdxInc (场编码的宏块)	last_significant_coeff_flag 的 ctxIdxInc
0	0	0	0	**32**	7	9	3
1	1	1	1	**33**	6	9	3
2	2	1	1	**34**	11	10	3
3	3	2	1	**35**	12	10	3
4	4	2	1	**36**	13	8	3
5	5	3	1	**37**	11	11	3
6	5	3	1	**38**	6	12	3
7	4	4	1	**39**	7	11	3
8	4	5	1	**40**	8	9	4
9	3	6	1	**41**	9	9	4
10	3	7	1	**42**	14	10	4
11	4	7	1	**43**	10	10	4
12	4	7	1	**44**	9	8	4
13	4	8	1	**45**	8	13	4
14	5	4	1	**46**	6	13	4
15	5	5	1	**47**	11	9	4
16	4	6	2	**48**	12	9	5
17	4	9	2	**49**	13	10	5
18	4	10	2	**50**	11	10	5
19	4	10	2	**51**	6	8	5
20	3	8	2	**52**	9	13	6
21	3	11	2	**53**	14	13	6
22	6	12	2	**54**	10	9	6
23	7	11	2	**55**	9	9	6
24	7	9	2	**56**	11	10	7
25	7	9	2	**57**	12	10	7

levelListIdx	significant_coeff_flag 的 ctxIdxInc (帧编码的宏块)	significant_coeff_flag 的 ctxIdxInc (场编码的宏块)	last_significant_coeff_flag 的 ctxIdxInc	levelListIdx	significant_coeff_flag 的 ctxIdxInc (帧编码的宏块)	significant_coeff_flag 的 ctxIdxInc (场编码的宏块)	last_significant_coeff_flag 的 ctxIdxInc
26	8	10	2	**58**	13	14	7
27	9	10	2	**59**	11	14	7
28	10	8	2	**60**	14	14	8
29	9	11	2	**61**	10	14	8
30	8	12	2	**62**	12	14	8
31	7	11	2				

设 numDecodAbsLevelEql 为绝对值等于 1 的累积解码变换系数等级；且令 numDecodAbsLevelGt1 为绝对值大于 1 的累积的解码变换系数等级。两个数值都是当前解码过程中关于同一个变换系数块的值。因此，对于 coeff_abs_level_minus1 的解码，其 ctxIdxInc 的值依赖于下列的 binIdx 的值。

— 如果binIdx等于0，ctxIdxInc为：

ctxIdxInc = ((numDecodAbsLevelGt1 != 0) ? 0: Min(4, 1 + numDecodAbsLevelEq1)) (9-17)

— 否则（binIdx大于0），ctxIdxInc为

ctxIdxInc = 5 + Min(4 − (ctxBlockCat == 3), numDecodAbsLevelGt1) (9-18)

9.3.3.2 算术解码过程

本过程的输入是 9.3.3.1 节导出的 bypassFlag, ctxIdx，算术解码引擎中的状态变量 codIRange 和 codIOffset。

本过程的输出是二进制码值。

图9-2为对于一个单独的二进制码值的整个算术解码处理过程。对于每一个二进制码值，上下文索引ctxIdx作为DecodeBin(ctxIdx)解码过程的一个参数，规定如下。

— 如果bypassFlag等于1，调用9.3.3.2.3中规定的DecodeBypass()。

— 否则，如果bypassFlag等于0并且ctxIdx等于276，调用9.3.3.2.4中定义的DecodeTerminate()。

— 否则(bypassFlag等于0且ctxIdx不等于276)，调用9.3.3.2.1中规定的DecodeDecision()。

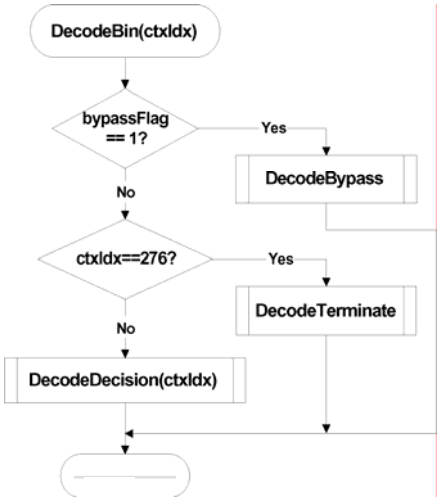


图 9-2－对于单个二进制码值的算术解码过程概要（资料性）

注 — 算术解码是基于递归间隔细分的原则。对于一个二进制判决(0,1)，给出一个概率估计p(0)和p(1)=1−p(0)，对于一个初始给定的范围为codIRange的码字子间隔，相应的将被再细分到范围为p(0) * codIRange和codIRange − p(0) * codIRange的两个子间隔中。根据被观察到的判决，相应的子间隔被做为新的码字间隔，并且指向那个间隔的二进制码串将成为观察到的二进制判决序列。这对于区分最大概率符号（MPS）和最小概率符号（LPS）是有用的，这样二进制判决被看作是一个对MPS或LPS的判决过程，而不是0或1。在这些术语中，每一个上下文由LPS的概率p_{LPS}和MPS的值(valMPS)规定，其不是0就是1。

本建议书 | 国际标准中的算术核心引擎有三个明显的属性：

— 对于LPS的概率p_{LPS}，概率估计由64个不同代表不同概率状态{ p_{LPS}(pStateIdx) | 0 <= pStateIdx < 64 }的有限状态机的执行得到。其中索引为pStateIdx = 0的概率状态对应于LPS概率值为0.5，随着LPS概率下降，则索引值增大。

— 范围codIRange表示编码引擎的状态被量化到一个小的集合{Q_1,…,Q_4}中，集合中为计算新的间隔范围时的预置的量化值。对包含所有64x4个预计算的Q_i * p_{LPS}(pStateIdx)乘积值的排序可以得到一个无相乘运算的codIRange * p_{LPS}(pStateIdx)的近似值。

— 对于符合均匀分布概率假设的语法元素或其中的部分有一个单独的简化或的编码或解码旁路过程。

9.3.3.2.1 二进制判决的算术解码过程

本过程的输入是 ctxIdx，codIRange 和 codIOffset。

本过程的输出是解码的值 binVal 和更新的变量 codIRange 和 codIOffset。

图 9-3 给出了解码单一判决（DecodeDecision）的流程图。

1．变量 codIRangeLPS 的值为：

— 给出当前codIRange值，变量qCodIRangeIdx为：

qCodIRangeIdx =(codIRange >> 6) & 0x03 (9-19)

— 给出qCodIRangeIdx和ctxIdx的pStateIdx，则表9-35中规定的变量rangeTabLPS赋值给codIRangeLPS：

codIRangeLPS = rangeTabLPS[pStateIdx][qCodIRangeIdx] (9-20)

2．变量 codIRange 置为等于 codIRange－codIRangeLPS，且：

— 如果 codIOffset 大于或等于 codIRange，变量 binVal 置为 1−valMPS，codIOffset 减去 codIRange，且 codIRange置为等于codIRangeLPS。

— 否则，变量binVal置为valMPS。

得到 binVal 的值，则应执行 9.3.3.2.1.1 中规定的状态转移。根据 codIRange 的当前值，应执行 9.3.3.2.2 中规定的重归一化。

9.3.3.2.1.1 状态转移过程

本过程的输入是当前 pStateIdx，解码的值 binVal 和 ctxIdx 的上下文变量的 valMPS 值。

本过程的输出是更新的 pStateIdx 和 ctxIdx 的上下文变量的 valMPS 值。

根据解码的 binVal 值，两个变量 pStateIdx 和 valMPS 的更新过程为：

```
if( binVal  ==  valMPS )
    pStateIdx = transIdxMPS( pStateIdx )
else {                                                                    (9-21)
    if( pStateIdx  ==  0 )
        valMPS = 1 − valMPS
    pStateIdx = transIdxLPS( pStateIdx )
}
```

表 9-36 规定了 valMPS 和 1−valMPS 解码后相应 transIdxMPS()和 transIdxLPS 的转换规则。

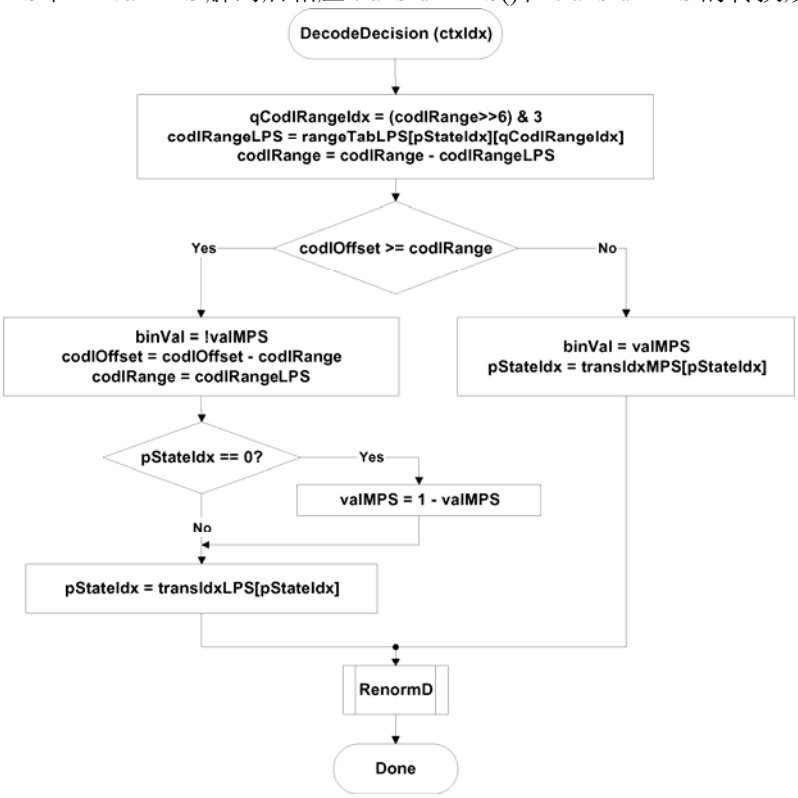


图 9-3－解码判决流程图

表 9-35－ pStateIdx和qCodIRangeIdx对应的rangeTabLPS值

pStateIdx	qCodIRangeIdx				pStateIdx	qCodIRangeIdx			
	0	1	2	3		0	1	2	3
0	128	176	208	240	**32**	27	33	39	45
1	128	167	197	227	**33**	26	31	37	43
2	128	158	187	216	**34**	24	30	35	41
3	123	150	178	205	**35**	23	28	33	39
4	116	142	169	195	**36**	22	27	32	37
5	111	135	160	185	**37**	21	26	30	35
6	105	128	152	175	**38**	20	24	29	33
7	100	122	144	166	**39**	19	23	27	31
8	95	116	137	158	**40**	18	22	26	30
9	90	110	130	150	**41**	17	21	25	28
10	85	104	123	142	**42**	16	20	23	27
11	81	99	117	135	**43**	15	19	22	25
12	77	94	111	128	**44**	14	18	21	24
13	73	89	105	122	**45**	14	17	20	23
14	69	85	100	116	**46**	13	16	19	22
15	66	80	95	110	**47**	12	15	18	21
16	62	76	90	104	**48**	12	14	17	20
17	59	72	86	99	**49**	11	14	16	19
18	56	69	81	94	**50**	11	13	15	18
19	53	65	77	89	**51**	10	12	15	17
20	51	62	73	85	**52**	10	12	14	16
21	48	59	69	80	**53**	9	11	13	15
22	46	56	66	76	**54**	9	11	12	14
23	43	53	63	72	**55**	8	10	12	14
24	41	50	59	69	**56**	8	9	11	13
25	39	48	56	65	**57**	7	9	11	12
26	37	45	54	62	**58**	7	9	10	12
27	35	43	51	59	**59**	7	8	10	11
28	33	41	48	56	**60**	6	8	9	11
29	32	39	46	53	**61**	6	7	9	10
30	30	37	43	50	**62**	6	7	8	9
31	29	35	41	48	**63**	2	2	2	2

表 9-36－状态转移表

pStateIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
transIdxLPS	0	0	1	2	2	4	4	5	6	7	8	9	9	11	11	12
transIdxMPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pStateIdx	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
transIdxLPS	13	13	15	15	16	16	18	18	19	19	21	21	22	22	23	24
transIdxMPS	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
pStateIdx	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
transIdxLPS	24	25	26	26	27	27	28	29	29	30	30	30	31	32	32	33
transIdxMPS	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
pStateIdx	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
transIdxLPS	33	33	34	34	35	35	35	36	36	36	37	37	37	38	38	63
transIdxMPS	49	50	51	52	53	54	55	56	57	58	59	60	61	62	62	63

9.3.3.2.2 算术解码引擎的重归一化过程

本过程的输入是来自条带数据的比特和变量 codIRange 和 codIOffset。

本过程的输出是更新的变量 codIRange 和 codIOffset。

图 9-4 给出了重归一化的流程图。codIRange 的当前值首先与 0x0100 比较，后面的步骤如下。

— 如果codIRange大于或等于0x0100，则不需要重归一化且过程RenormD终止；

— 否则（codIRange小于0x0100），进入在归一化循环。再这个循环中，codIRange的值被乘以2，例如，左移一位且通过使用read_bits(1)把单比特移动到codIOffset。

在完成这个过程之后，比特流中不应包含可以导致 codIOffset 大于或等于 codIRange 的数据。

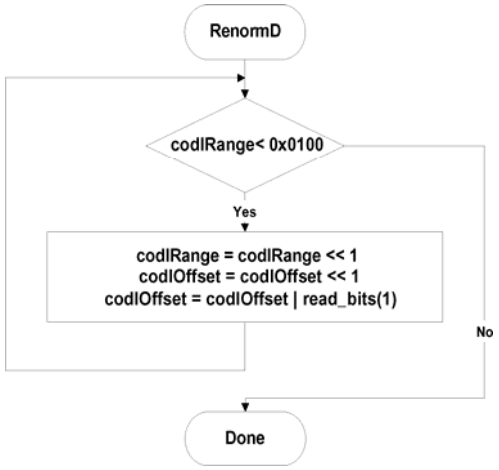


图 9-4－重归一化流程图

9.3.3.2.3 二进制判决的解码旁路过程

本过程的输入是来自条带数据的比特和变量 codIRange 和 codIOffset。

本过程的输出是更新的变量 codIOffset 和解码值 binVal。

当 bypassFlag 等于 1 时解码旁路过程被调用。图 9-5 给出了相应过程的流程图。

首先，codIOffset 被乘以 2。即，左移一位且通过使用 read_bits(1)把单比特移动到 codIOffset，然后，codIOffset 的值于 docIRange 进行比较并且采取以下规定的步骤：

— 如果codIOffset大于或等于codIRange，变量binVal置为1且codIOffset自减codIRange。

— 否则（codIOffset小于codIRange），变量binVal置为0。

在完成这个过程之后，比特流中不应包含可以导致 codIOffset 大于或等于 codIRange 的数据。

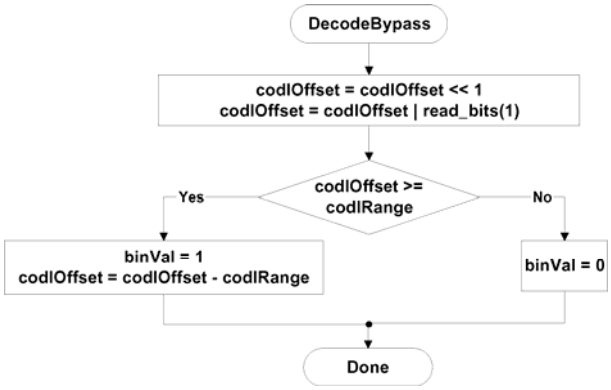


图 9-5－解码旁路过程流程图

9.3.3.2.4 结束前的二进制判决解码过程

本过程的输入是条带数据中的比特和变量 codIRange 和 codIOffset。

本给出的输出是更新后的变量 codIRange 和 codIOffset 和解码值 binVal。

本解码过程用于解码 end_of_slice_flag 和对应于 ctxIdx 等于 276 的 I_PCM 模式的二进制数据。图 9-6 给出了相应的解码流程，规定如下。

首先，codIRange 的值自减 2。然后，codIOffset 的值与 docIRange 进行比较并且采取如下步骤：

— 如果codIOffset大于或等于codIRange，变量binVal置为1，不使用重归一化过程，且CABAC解码结束。寄存codIOffset中插入的最后一比特等于1。当解码end_of_slice_flag，寄存codIOffset中插入的最后一比特解释为rbsp_stop_one_bit。

— 否则（codIOffset小于codIRange），变量binVal置为0且执行9.3.3.2.2.中规定的重归一化过程。

注 — 本过程也可使用ctxIdx=276时的DecodeDecision(ctxIdx)实现。在这种情况下，解码值等于1，下7个比特将被DecodeDecision(ctxIdx)读入，解码处理不得不相应调整比特流指针使之能正确解码后续的语法元素。

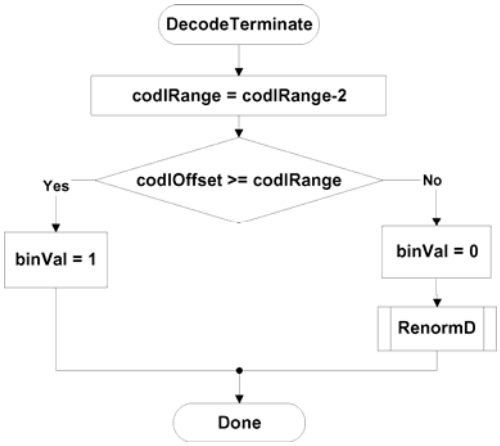


图 9-6－结束前的解码判决流程图

9.3.4 算术解码过程 (资料性)

本节不是本建议书 | 国际标准的组成部分。

本过程的输入是要进行编码或者写入的决定。

本过程的输出是写入到 RBSP 的比特。

本节主要描述了一个算术编码引擎，该引擎与 9.3.3.2 节的解码引擎相匹配。编码引擎本质上是和解码引擎同步的，比如说，呼叫过程以相同的顺序进行。本节将分别描述如下过程：InitEncoder，EncodeDecision，EncodeBypass，与 InitDecoder 相对应的 EncodeTerminate，DecodeDecision，DecodeBypass 和 DecodeTerminate。算术编码引擎的状态使用变量 codILow 表示，codILow 指向子间隔的较低末端，变量 codIRange 说明了该子间隔的范围。

9.3.4.1 算术解码引擎的初始化过程(资料性)

本节不是本建议书 | 国际标准的组成部分。

在解码条带的第一个宏块之前，和解码任一类型为 I_PCM 的宏块的 pcm_alignment_zero_bit 、all pcm_sample_luma 及 pcm_sample_chroma 数据之后，调用本过程。

本过程的输出是算术解码引擎的 codILow，codIRange，firstBitFlag，bitsOutstanding 和 symCnt 值。

在解码器初始化过程中，codILow 设为 0，codIRange 设为 0x01FE。然后 firstBitFlag 设为 1，bitsOutstanding 和 symCnt 计数器设为 0。

注 — codILow的最小注册器精度为10比特，CodIRange的为9比特。计数器bitsOutstanding和symCnt的精度应该设为足够大，以防止相关注册器的溢出。当MaxBinCountInSlice指示在条带编码的二进制判定的最大总数，变量bitsOutstanding和symCnt的最小注册器精度由Ceil(Log2(MaxBinCountInSlice + 1))给出。

9.3.4.2 二进制判定的编码过程(资料性)

本节不是本建议书 | 国际标准的组成部分。

本过程的输入是上下文索引 ctxIdx，待编码的 binVal 值和变量 codIRange，codILow 及 symCnt。

本过程的输出是变量 codIRange，codILow 和 symCnt。

图 9-7 所示为一个单独判定的流程图。在第一步中，变量 codIRangeLPS 通过如下过程推导出来。

给定 codIRange 的当前值，codIRange 映射到通过等式 9-19 计算出的 codIRange 量化值的索引 qCodIRangeIdx。

qCodIRangeIdx 值、与 ctxIdx 有关的 pStateIdx 值用来决定表 9-35 规定的变量 rangeTabLPS 值，该值被分配给 codIRangeLPS。codIRange − codIRangeLPS 的值分配给 codIRange。

在第二步，比较 binVal 值和 ctxIdx 相关的 valMPS 值。当 binVal 与 valMPS 不同时，将 codIRange 加到 codILow，codIRange 的值设为与 codIRangeLPS 相等。给定已编码的判定，状态变换遵守 9.3.3.2.1.1 的规定。取决于 codIRange 的当前值，重归一化按照 9.3.4.3 的规定进行。最后，变量 symCnt 加 1。

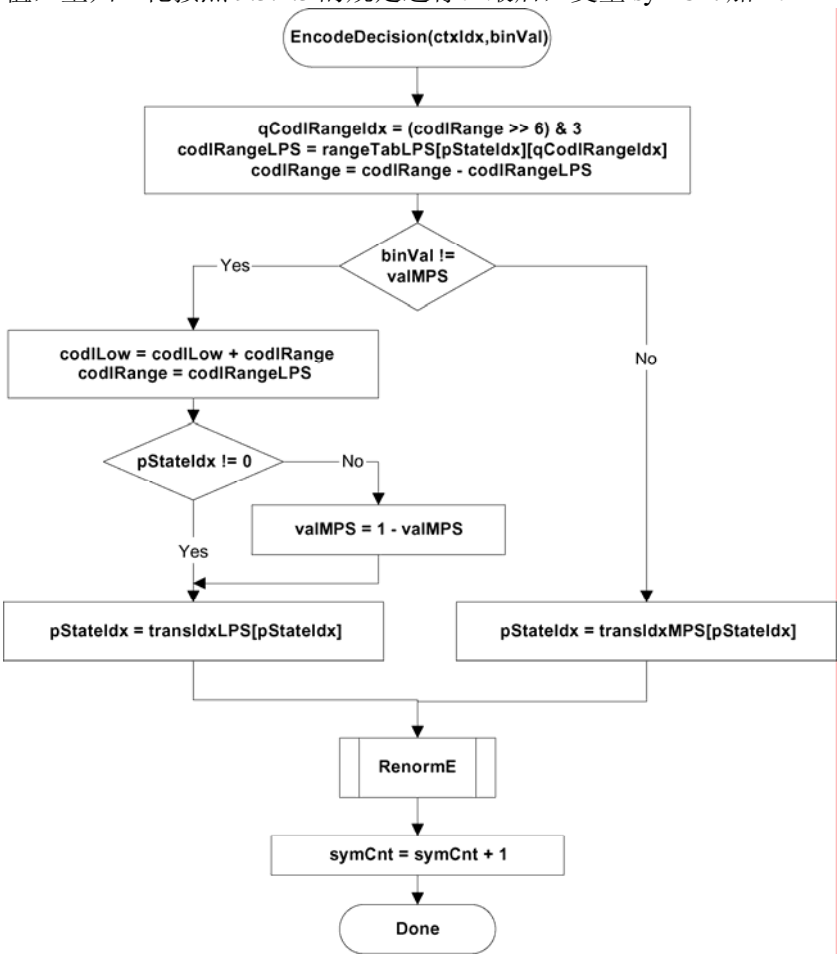


图 9-7—判定编码的流程图

9.3.4.3 算术解码引擎中的重归一化过程(资料性)

本节不是本建议书 | 国际标准的组成部分。

本过程的输入是变量 codIRange，codILow, firstBitFlag 和 bitsOutstanding。

本过程的输出是写入到 RBSP 的 0 或者多个字节，以及更新后的变量 codIRange, codILow，firstBitFlag 和 bitsOutstanding。

重归一化过程如图 9-8 所示。

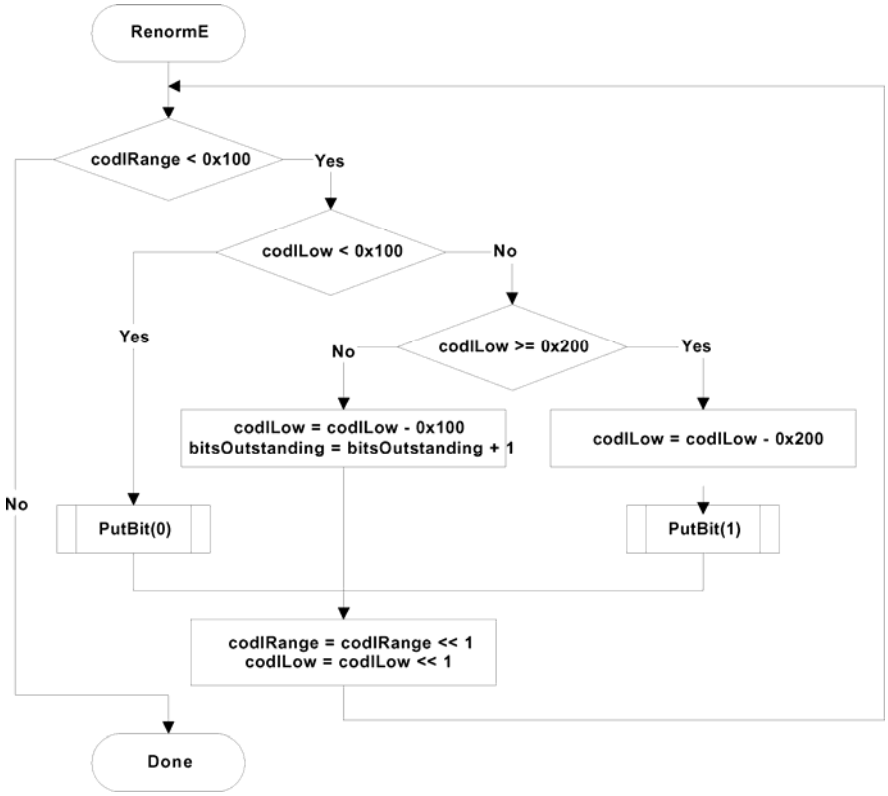


图 9-8— 编码器重归一化流程图

图 9-9 中的 PutBit( )提供了转页控制。它使用函数 WriteBits(B, N)，在 B 值时将 N 比特写入到比特流中，并且将比特流指针前移 N 比特。该函数假设比特流指示器存在，并且可通过解码过程将指针指向要写入到比特流的下一比特。

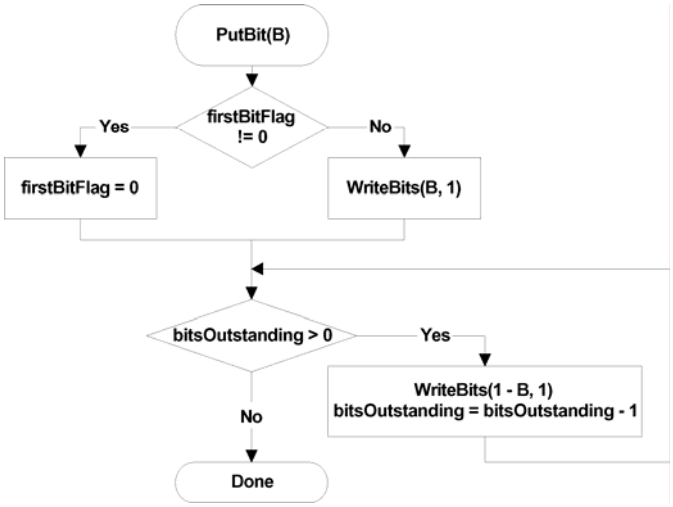


图 9-9—PutBit(B)流程图

9.3.4.4 二进制判定的旁路解码过程 (资料性)

本节不是本建议书 | 国际标准的组成部分。

本过程的输入是变量 binVal，codILow，codIRange，bitsOutstanding 和 symCnt。

本过程的输出是写入到 RBSP 的 1 个字节，以及更新后的变量 codILow，bitsOutstanding 和 symCnt。

本编码过程应用到所有 bypassFlag 等于 1 的二进制判定。重归一化也包含到本过程的规定中，如图 9-10 所示。

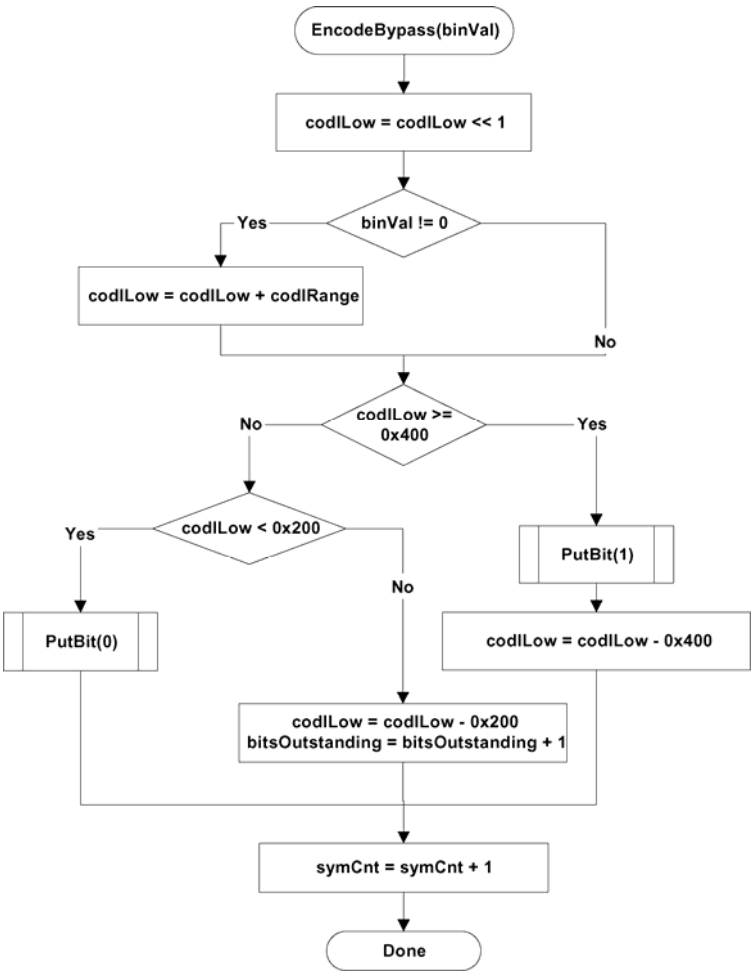


图 9-10－旁路编码流程图

9.3.4.5 结束前的二进制判定的编码过程 (资料性)

本节不是本建议书 | 国际标准的组成部分。

本过程的输入是变量 binVal，codIRange, codILow，bitsOutstanding 和 symCnt。

本过程的输出是写入到 RBSP 的 0 或者多个字节，以及更新后的变量 codIRange, codILow，bitsOutstanding 和 symCnt。

编码流程如图 9-11 所示，在 ctxIdx 等于 276 时， end_of_slice_flag 和指示 I_PCM mb_type 的二进制码的编码都需使用该流程。

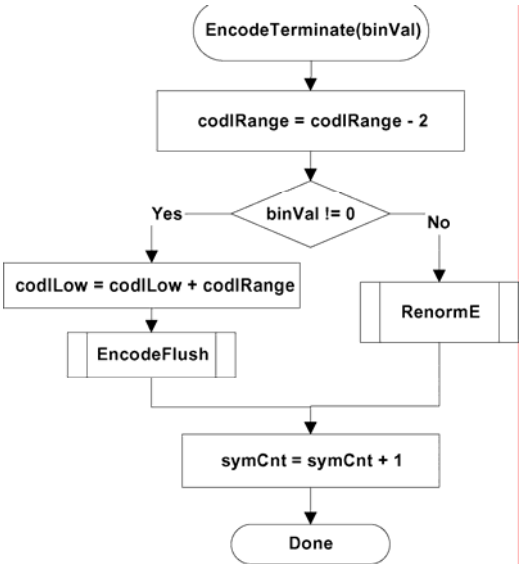


图 9-11－结束前的判定编码流程图

当要编码的 binVal 值等于 1 时，结束 CABAC 编码，应用图 9-12 所示的流程。在该流程中，由 WriteBits(B, N)写入的最后一个比特等于 1。当解码 end_of_slice_flag 时，最后一个比特为 rbsp_stop_one_bit。

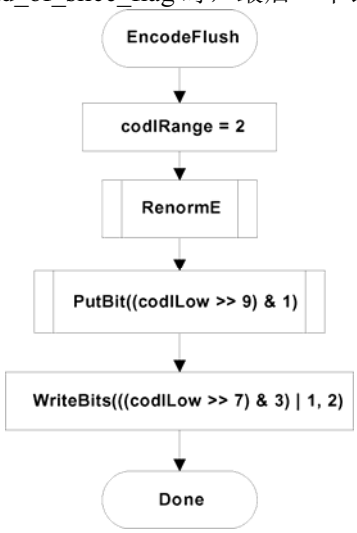


图 9-12－结束时的flush流程图

9.3.4.6 字节填充过程(资料性)

本节不是本建议书 | 国际标准的组成部分。

在对图像的最后一个条带的最后一个宏块的编码完成并且封装后，调用本过程。

本过程的输入是图像中所有 VCL NAL 单元的 NumBytesInVclNALunits 比特数量，图像中宏块 PicSizeInMbs 的数量，图像所有 VCL NAL 单元的内容编码后所得得 BinCountsInNALunits 二进制标识的数量。

本过程的输出是添加到 NAL 单元的 0 或者更多字节。

设变量 k 等于 Ceil((Ceil(3 * (32 * BinCountsInNALunits − (RawMbBits * PicSizeInMbs) ÷ 1024) − NumBytesInVclNALunits) ÷ 3)。根据变量 k 应用如下过程：

— 如果k小于等于0，没有cabac_zero_word 添加到NAL单元。

— 否则(k 大于 0)，在封装后 3 字节 0x000003 添加到 NAL 单元 k 次，这里两字节 0x0000 代表了 cabac_zero_word，第三个字节0x03代表了一个emulation_prevention_three_byte。

附 件 A

简表与级别

(本附件是本建议书 | 国际标准的组成部分)

简表与级别规定了对比特流的限制，因此也限制了比特流解码所需的能力。简表与级别可能也用于指示独立解码应用之间的互操作点。

注1 — 本建议书 | 国际标准不包括解码器上单独可选的“选项”，因为这会增加互操作的难度。

每一个简表定义了一个算法特征的子集，并限定所有与该简表一致的解码器都必须支持。

注2 — 不要求编码器使用某个简表中的任何特征子集。

每一个级别定义了对本建议书 | 国际标准中的语法要素取值的限制集合。相同的级别定义集合用于所有的简表，但单独的应用对所支持的简表可能支持不同的级别。一般来说，对于特定的一个简表，不同的级别对应于解码器的不同处理过程的负载和消耗的存储容量。

A.1 视频解码器能力的需求

与本建议书 | 国际标准一致的视频解码器的能力由对某些特定的视频流进行解码来定义，这些视频流满足本附件所规定的简表与级别的限制。对于每一个简表，所能支持的级别也应进行表述。

本附件规定了对于语法元素 profile_idc 与 level_idc 特定的值。Profile_idc 与 level_idc 的其它值保留给 ITU|ISO/IEC 备用。

注 — 解码器不应因为 profile_idc 或 level_idc 的取值落在本建议书 | 国际标准所规定的值之间，就推定这个值代表所代表的能力处于规定好的简表与级别之间。因为这样做将使得 ITU-T|ISO/IEC 失去对保留值的分配限制。

A.2 简表

A.2.1 基准简表

比特流若与基准简表相一致，应遵如下限制：

— 只可能有I与P条带类型存在；

— NAL单元流不应包含取值范围为2到4的nal_unit_type，包括2与4；

— 序列参数集中frame_mbs_only_flag应为1；

— 语法元素 chroma_format_idc， bit_depth_luma_minus8，bit_depth_chroma_minus8，qpprime_y_zero_transform_bypass_flag，seq_scaling_matrix_present_flag不应出现在序列参数集中；

— 图像参数集中的参数weighted_pred_flag 与weighted_bipred_idc的取值应该为0；

— 图像参数集中的参数entropy_coding_mode_flag的取值应为0；

— 图像参数集中slice_groups_minus1取值应该在0到7之间，包括0与7；

— 语法元素transform_8x8_mode_flag，pic_scaling_matrix_present_flag，与 second_chroma_qp_index_offset 不应在图像参数集中出现；

— 语法元素level_prefix的取值不应大于15；

— 应满足A.3节中对于基准简表所定义的级别规定。

profile_idc 的取值为 66 时，比特流与基准简表相一致。

与基准简表某一级别相一致的解码器应可以对所有满足 profile_idc 等于 66 或 constraint_set0 等于 1，且 level_idc 和 constraint_set3_flag 所表征的级别小于或等于该级别的流进行解码。

A.2.2 主要简表

比特流若与主要简表相一致，则应遵循：

— 只可能有I、P和B条带类型存在；

— NAL单元流不应包含从2到4取值的nal_unit_type，包括2与4；

— 不允许任意的条带顺序；

— 语法元素hroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, qpprime_y_zero_transform_bypass_flag, 与 seq_scaling_matrix_present_flag不应在序列参数集中出现。

— 图像参数集中应该具有只等于0的num_slice_groups_minus1；

— 图像参数集中应该具有只等于0的redundant_pic_cnt_present_flag；

— 语法元素transform_8x8_mode_flag, pic_scaling_matrix_present_flag, 与 second_chroma_qp_index_offset不应出现在图像参数集中；

— 语法元素level_prefix不应大于15（如果出现时）；

— 应满足A.3节中对于主要简表所定义的级别的规定。

profile_idc 的取值为 77 时，比特流与主要简表相一致。

与主要简表某一级别相一致的解码器应可以对所有满足 profile_idc 等于 77 或 constraint_set1 等于 1，且 level_idc 和 constraint_set3_flag 所表征的级别小于或等于该级别的流进行解码。

A.2.3 扩展简表

与扩展简表相一致的比特流应符合如下规定：

— 序列参数集中应包括取值为1的direct_8x8_inference_flag；

— 语法元素 chroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, qpprime_y_zero_transform_bypass_flag, 与 seq_scaling_matrix_present_flag不应包含在序列参数集中；

— 图像参数集应包括取值为0的entropy_coding_mode_flag参数；

— 图像参数集应包括取值范围0-7的num_slice_groups_minus1参数，包括0与7；

— 语法元素transform_8x8_mode_flag, pic_scaling_matrix_present_flag, 与 second_chroma_qp_index_offset不应出现在图像参数集中；

— 语法元素level_prefix不应大于15（如果出现）；

— 应满足A.3中对于扩展简表定义的级别限制。

profile_idc 的取值为 88 时，比特流与扩展简表相一致。

与扩展简表某一级别相一致的解码器应可以对所有满足 profile_idc 等于 88 或 constraint_set2 等于 1，且 level_idc 所表征的级别小于或等于该级别的流进行解码。

与扩展简表某一级别相一致的解码器也应可以对所有满足 profile_idc 等于 66 或 constraint_set0 为 1，且 level_idc 及 constraint_set3_flag 所表征的级别小于或等于该级别的流进行解码。

A.2.4 高级简表

与高级简表相一致的比特流符合下列规定：

— 只有I，P与B条带；

— NAL单元流不应包含取值为2到4的nal_unit_type参数，包括2和4；

— 不允许任意的条带顺序；

— 图像参数集中应有只为0的num_slice_groups_minus1参数；

— 图像参数集中应有只为0的redundant_pic_cnt_present_flag参数；

— 序列参数集中应有取值为0到1，包括0与1的chroma_format_idc参数；

— 列参数集中应有只为0的bit_depth_luma_minus8参数；

— 序列参数集中应有只为0的bit_depth_chroma_minus8参数；

— 序列参数集中应有只为0的qpprime_y_zero_transform_bypass_flag参数。

— 应满足A.3节中规定的高级简表的级别限制。

profile_idc 取值为 100 时，比特流与高级简表相一致。与高级简表某一级别相一致的解码器应可以对所有满足 level_idc 与 Constraint_set3_flag 所表征的级别小于或等于该级别的流进行解码，或下列两个条件均为真：

— Profile_idc为77或100，或

— Constraint_set1_flag为1。

A.2.5 高级10简表

与高级 10 简表相一致的比特流符合下列规定：

— 只有I，P与B条带；

— NAL单元流不应包含取值为2到4，包括2与4的nal_unit_type参数；

— 不允许任意条带顺序；

— 图像参数集中应有只为0的num_slice_groups_minus1参数；

— 图像参数集中应有只为0的redundant_pic_cnt_present_flag参数；

— 序列参数集中应有取值在0到1之间，包括0与1的chroma_format_idc参数；

— 序列参数集中应有取值在0到2之间，包括0与2的bit_depth_luma_minus8参数；

— 序列参数集中应有取值在0到2之间，包括0与2的bit_depth_chroma_minus8参数；

— 序列参数集中应有只为0的qpprime_y_zero_transform_bypass_flag参数；

— 应满足A.3节中规定的高级10简表的级别限制。

profile_idc 取值为 110 时，比特流与高级 10 简表相一致。与高级 10 简表某一级别相一致的解码器应可以对所有满足 level_idc 与 Constraint_set3_flag 所表征的级别小于或等于该级别的流进行解码，或下列两个条件均为真：

— Profile_idc为77、100或110，或

— Constraint_set1_flag为1。

A.2.6 高级 4:2:2 简表

与高级 4:2:2 简表相一致的比特流应遵循如下限制：

— 只有I，P与B条带；

— NAL单元流不应包含取值为2到4，包括2与4的nal_unit_type参数；

— 不允许任意的条带顺序；

— 图像参数集中应有只为0的num_slice_groups_minus1参数；

— 图像参数集中应有只为0的redundant_pic_cnt_present_flag参数；

— 序列参数集中应有取值在0到2之间，包括0与2的chroma_format_idc参数；

— 序列参数集中应有取值在0到2之间，包括0与2的bit_depth_luma_minus8参数；

— 序列参数集中应有取值在0到2之间，包括0与2的bit_depth_chroma_minus8参数；

— 序列参数集中应有只为0的qpprime_y_zero_transform_bypass_flag参数；

— 应满足A.3一节中规定的高级4:2:2简表的级别限制。

profile_idc 取值为 122 时，比特流与高级 4:2:2 简表相一致。与高级 4:2:2 简表某一级别相一致的解码器应可以对所有满足 level_idc 与 Constraint_set3_flag 所表征的级别小于或等于该级别的流进行解码，或下列两个条件均为真：

— Profile_idc为77、100、110或122，或

— Constraint_set1_flag为1。

A.2.7 高级4:4:4简表

与高级 4:4:4 简表相一致的比特流应遵循如下限制：

— 只有I，P与B条带；

— NAL单元流不应包含取值为2到4，包括2与4的nal_unit_type参数；

— 不允许任意的条带顺序；

— 图像参数集中应有只为0的num_slice_groups_minus1参数；

— 图像参数集中应有只为0的redundant_pic_cnt_present_flag参数；

— 序列参数集中应有取值在0到4之间，包括0与4的bit_depth_luma_minus8参数；

— 序列参数集中应有取值在0到4之间，包括0与4的bit_depth_chroma_minus8参数；

— 应满足A.3节中规定的高级4:4:4简表的级别限制。

profile_idc 取值为 144，比特流与高级 4:4:4 简表相一致。与高级 4:4:4 简表某一级别相一致的解码器应可以对所有满足 level_idc 与 Constraint_set3_flag 所表征的级别小于或等于该级别的流进行解码，或下列两个条件均为真：

— Profile_idc为77、100、110、122或144，或

— Constraint_set1_flag为1。

A.3 级别

为了便于表述本附件中的限制，做如下规定：

— 令访问单元n为解码顺序中的第n个被访问的单元，第1个访问单元为访问单元0。

— 令图像n为基本编码图像，或与访问单元n的相对应的解码图像。

A.3.1 基准、主要及扩展简表中通用的级别限制

令变量 fR 根据下列条件得到：

— 如果图像n是一帧，fR等于1 ÷ 172。

— 否则（图像n为场），fR等于1 ÷ (172 * 2)。

与基准、主要或扩展简表的某一级别相一致的比特流应满足如下规定：

a) 按照C.1.2节中的规定，把访问单元n从CPB中进行移除的标称时间，满足$t_{r,n}(n) - t_r(n - 1)$大于或等于 Max(PicSizeInMbs ÷ MaxMBPS, fR)，这里MaxMPS对应于图像n−1在表A-1中规定的值。PicSizeInMbs为图像n-1里宏块号。

b) C.2.2节中规定的来自DPB的连续图像输出时间差满足$\Delta t_{o,dpb}(n) >= Max(PicSizeInMbs \div MaxMBPS, fR)$，这里MaxMPS为对应于表A-1里规定的图像n-1的值，PicSizeInMbs为图像n宏块号，提供图像n已输出及不是比特流最后一幅图像等信息。

c) 访问单元0的NumBytesInNALunit变量之和小于或等于$384 \times (PicSizeInMbs + MaxMBPS \times (t_r(0) - t_{r,n}(0))) \div MinCR$，这里MaxMBPS与MinCR为表A-1中规定的值，用于图像0。PicSizeInMbs为图像0中宏块号。

d) 访问单元n（n>0）的NumBytesInNALunit变量之和小于或等于$384 * MaxMBPS * (t_r(n) - t_r(n-1)) \div MinCR$，这里MaxMBPS与MinCR为表A-1中规定的值，应用于图像n。

e) PicWidthInMbs * FrameHeightInMbs <= MaxFS，这里MaxFS在表A-1中说明。

f) PicWidthInMbs <= Sqrt(MaxFS * 8)

g) FrameHeightInMbs <= Sqrt(MaxFS * 8)

h) max_dec_frame_buffering <= MaxDpbSize，这里MaxDpbSize等于Min(1024 * MaxDPB / (PicWidthInMbs * FrameHeightInMbs * 384), 16)，MaxDPB在表A-1中给出，单位为1024。

i) 对于VCL HTD参数，BitRate[SchedSelIdex] <= 1000 * MaxBR且CpbSize[SchedSelIdx] <= 1000 * MaxCPB至少作为SchedSelIdx的一个值，这里当vlc_hrd_parameter_present_flag为1时，等式E-37中给出BitRate[SchedSelIdex]，等式E-38给出CpbSize[SchedSelIdx]。MaxBR与MaxCPB在表A-1中1000bits/s与1000bits中分别规定。对于至少一个取值在0至cpb_cnt_minus1之间，包括0与1的SchedSelIdx，比特流应满足这些条件。

j) 对于NAL HRD参数，BitRate[SchedSelIdx] <= 1200 * MaxBR与CpbSize[SchedSelIdx] <= 1200 * MaxCPB至少作为一个ShedSelIdx的值，这里当nal_hard_parameters_present_flag为1时，BitRate[SchedSelIdx]由等式E-37给出，CpbSize[SchedSelIdx]由等式E-38给出。MaxBR与MaxCPB在表A-1中以1200bits/s和1200bits为单位分别规定。对于至少一个取值在0至cpb_cnt_minus1之间，包括0与1的SchedSelIdx比特流应满足这些条件。

k) 亮度运动矢量的垂直运动矢量分量范围不会超出以亮度帧样点为单位的MaxVmvR，这里MaxVmvR在表A-1中规定。

注1 — 当色度格式idc等于1且当前宏块为场宏块时，色度运动向量的运动矢量向量范围可能会超过以亮度帧样点为单位的MaxVmvR，这取决于色度运动矢量计算的方法，如8.4.1.4节中所述。

l. 水平运动矢量范围不会超过范围−2048至2047.75，包括−2048与2047.75，单位为亮度样点。

m. 解码顺序中每两个连续宏块运动向量（同样适用于解码顺序中所有来自于条带的最后一个宏块与下一个条带的第一个宏块，并且特别适用于一帧图像的最后一条带的最后一宏块与下一帧图像第一条带的第一宏块）不超过MaxMvsPer2Mb，这里MaxMvsPer2Mb在表A-1中定义。对于每一宏块，运动矢量数量为宏块帧间或帧内预测过程后变量MvCnt的值。

n) 对于任一宏块，macroblock_layer()数据中的比特数不能大于3200。macroblock_layer()数据中的比特数依赖于entropy_coding_mode_flag，其计算如下：

— 如果entropy_coding_mode_flag为0，macroblock_layer()数据中的比特数由宏块的macroblock_layer( )语法结构里的比特数得到。

— 否则（entropy_coding_mode_flag为1）当解析与宏块相关的marcoblock_lay()时，对于一个宏块macroblock_layer( ) 数据的比特数由read_bits(1)被调用的次数给出，如9.3.3.2.2与9.3.3.2.3所述。

表 A-1 说明了每一个级别的限制，表中所有标为“−”的条目表明没有相关的限制。为级别能力比较的目的，如果一个级别比另一级别更接近于表 A-1 的表头（表底），该级别应认为比另一级别更低（高）。

与比特流相一致的级别应在语法元素 level_idc 与 constraint_set3_flag 里指明，如下所示：

— 如果level_idc等于11且constraint_set3_flag也等于1，表明级别为1b级

— 否则（level_idc不等于11，或constraint_set3_flag不等于1），level_idc应设为等于表A-1中级别号的10倍的值，且constraint_set3_flag应设为0。

表 A-1－级别限制

级别号	最大宏块处理速率 MaxMBPS（MB/s）	最大帧长 MaxFS (MB)	最大解码图像缓存大小 MaxDPB (4:2:0 为 1024 字节)	最大视频比特率 MaxBR (1000 bits/s, 1200 bits/s, cpbBrVclFactor bits/s, 或 cpbBrNalFact 或 bits/s)	最大 CPB 大小 MaxCPB (1000 比特, 1200 比特, cpbBrVclFactor 比特，或 cpbBrNalFactor 比特)	垂直 MV 分量范围 MaxVmvR (亮度帧样点)	最小压缩率 MinCR	每两连续 MB 最大运动矢量数 MaxMvsPer2Mb
1	1 485	99	148.5	64	175	[-64,+63.75]	2	-
1b	1 485	99	148.5	128	350	[−64,+63.75]	2	-
1.1	3 000	396	337.5	192	500	[−128,+127.75]	2	-
1.2	6 000	396	891.0	384	1 000	[−128,+127.75]	2	-
1.3	11 880	396	891.0	768	2 000	[−128,+127.75]	2	-
2	11 880	396	891.0	2 000	2 000	[−128,+127.75]	2	-
2.1	19 800	792	1 782.0	4 000	4 000	[−256,+255.75]	2	-
2.2	20 250	1 620	3 037.5	4 000	4 000	[−256,+255.75]	2	-
3	40 500	1 620	3 037.5	10 000	10 000	[−256,+255.75]	2	32
3.1	108 000	3 600	6 750.0	14 000	14 000	[−512,+511.75]	4	16
3.2	216 000	5 120	7 680.0	20 000	20 000	[−512,+511.75]	4	16
4	245 760	8 192	12 288.0	20 000	25 000	[−512,+511.75]	4	16
4.1	245 760	8 192	12 288.0	50 000	62 500	[−512,+511.75]	2	16
4.2	522 240	8 704	13 056.0	50 000	62 500	[−512,+511.75]	2	16
5	589 824	22 080	41 400.0	135 000	135 000	[−512,+511.75]	2	16
5.1	983 040	36 864	69 120.0	240 000	240 000	[−512,+511.75]	2	16

表 A-1 中带有非整形级别编号的级别认为是“中间级别”。

注2 — 所有的级别有相同的情形，但是某些应用可能只选用整数编号的级别。

参考性章节 A.3.4 通过几个范例图像格式显示了这些限制对于帧速率的影响。

A.3.2 对于高级、高级10、高级4:2:2与高级4:4:4简表中通用的级别限制

变量 fR 从如下计算：

— 如果图像n是一帧，fR为 1 ÷ 172；

— 否则（图像n为一场），fR设为 1 ÷ (172 * 2)。

比特流在某一等级满足高级、高级 10、高级 4:2:2 与高级 4:4:4 简表应满足下述的限制：

a) 按照C.1.2节所规定的来自于CPB访问单元n（n>0）的标称移除时间满足大于$t_{r,n}(n) - t_r(n- 1)$或等于$\Delta t_{o,dpb}(n) >= Max(PicSizeInMbs \div MaxMBPS, fR)$，这里MaxMBPS为表A-1中所规定的应用于图像n-1值，PicSizeInMbs为图像n-1宏块的数目。

b) 按照C.2.2节所规定的来自于DPB的图像连续输出时间差满足$\Delta t_{o,dpb}(n) >= Max(PicSizeInMbs \div MaxMBPS, fR)$，这里MaxMBPS为表A-1中所规定的应用于图像n-1值，PicSizeInMbs为图像n中宏块的数目，该数目说明了图像n为输出图像且不为输出比特流的最后一帧图像。

c) PicWidthInMbs * FrameHeightInMbs <= MaxFS，MaxFS于表A-1中规定。

d) PicWidthInMbs <= Sqrt(MaxFS * 8)

e) FrameHeightInMbs <= Sqrt(MaxFS * 8)

f) max_dec_frame_buffering <= MaxDpbSize ， 这 里 MaxDpbSize 等 于 Min(1024 * MaxDPB / (PicWidthInMbs * FrameHeightInMbs * 384), 16)，MaxDPB在表A-1中规定。

g) 垂直运动矢量分量范围不会超出以亮度帧样点为单位的MaxVmvR，这里MaxVmvR在表A-1中规定。

h) 水平运动向量范围不会超出以亮度帧样点为单位的-2048至2047.75，包括2048与2047.75。

i) 在解码顺序中，每两个连续运动矢量的数目（同样适用解码顺序中于取自条带的最后一个宏块与下一个条带的第一个宏块的运动矢量总数）不会超过MaxMvxPer2Mb，MaxMvsPer2Mb在表A-1中规定，对于每一个宏块，运动矢量的数目为对宏块进行帧内或帧间预测结束后，变量MvCnt的值。

j) 对于任一宏块，macroblock_layer()数据中的比特数不能大于128 + RawMbBits。macroblock_layer()数据中的比特数依赖于entropy_coding_mode_flag，其计算如下：

— 如果entropy_coding_mode_flag为0， macroblock_layer()数据中的比特数由宏块的macroblock_layer( )语法结构里的比特数得到。

— 否则（(entropy_coding_mode_flag为1）一个宏块的macroblock_layer()数据由解析与宏块相关的marcoblock_lay()时，read_bits(1)被调用的次数给出，如9.3.3.2.2与9.3.3.2.3所述。

表 A-1 规定了每一级别的限制，标为“−”的条目表示没有相应的限制。

与比特流相一致的级别应在语法元素 level_idc 里指明，如下所示：

— 如果level_idc等于9表明级别为1b级；

— 否则（level_idc不等于9），level_idc应设为等于表A-1中级别号的10倍的值。

A.3.3 与简表相关的级别限制

a) 与主要、高级、高级10、高级4:2:2或高级4:4:4简表相一致的比特流里，访问单元0的移除时间应满足条件：图像0中条带数目小于或等于 (PicSizeInMbs + MaxMBPS * (t_r(0) - $t_{r,n}$(0))) ÷ SliceRate，这里SliceRate为表A-4所规定的应用于图像0的值。

b) 与主要、高级、高级10、高级4:2:2或高级4:4:4简表相一致的比特流里,访问单元n与n-1（n>1），的连续移除时间之差应满足图像n的条带数目小于或等于MaxMBPS * (t_r(n) − t_r(n−1)) ÷ SliceRate，SliceRate为表A-4规定的应用于图像0的值。

c) 与主要、高级、高级10、高级4:2:2或高级4:4:4简表相一致的比特流里，对于表A-4所规定各级别，序号参数集应该包括direct_8x8_inference_flag等于1。

注1 — direct_8x8_inference_flag与基准简表不相关，因其不允许B条带类型（在A.2.1节中规定），对于扩展简表的所有级别，direct_8x8_inference_flag等于1（在A.2.3节中规定）。

d) 与主要、高级、高级10、高级4:2:2或高级4:4:4或扩展简表相一致的比特流里，对于表A-4所规定的关于主要、高级、高级10、高级4:2:2或高级4:4:4简表，及表A-5中规定的扩展简表的各级别，参数集应有设置为1的frame_mbs_only_flag。

注2 — frame_mbs_only_flag对于基准简表的所有级别均为1（A.2.1规定）。

e) 与主要、高级、高级10、高级4:2:2或高级4:4:4或扩展简表相一致的比特流中，对于表A-4所规定的关于主要、高级、高级10、高级4:2:2或高级4:4:4简表，及表A-5中规定的扩展简表中， MiniLumaBiPredSize显示为8x8的各级别，B宏块中的sub_mb_type的值应不等于B_Bi_8x4，B_Bi_4x8或 B_Bi_4x4。

f) 与基准与扩展简表相一致的比特流中，对于所有如8.4.2.2.1节中规定的过程调用，带有mb_type等于P_8x8, P_8x8ref0 或 B_8x8宏块中($xInt_{max}$ − $xInt_{min}$ + 6) * ($yInt_{max}$ − $yInt_{min}$ + 6) <= MaxSubMbRectSize，用于

生成每一个8x8子宏块预测亮度样点阵列，这里NumSubMbPart(sub_mb_type) > 1，MaxSubMbRectSize对于基准简表在表A-3中规定，对于扩展简表在表A-5中规定。

— $xInt_{mi}$为所有子宏块亮度样点预测中$xInt_L$的最小值；

— $xInt_{max}$为所有子宏块亮度样点预测中$xInt_L$的最大值；

— $yInt_{min}$为所有子宏块亮度样点预测中$yInt_L$的最小值；

— $yInt_{max}$为所有子宏块亮度样点预测中$yInt_L$的最大值；

g) 与主要、高级、高级10、高级4:2:2或高级4:4:4简表相一致的比特流里，对于NAL HRD参数，至少一个SchedSelIdx 的值， BitRate[SchedSelIdx] <= cpbBrNalFactor * MaxBR 且 CpbSize[SchedSelIdx] <= cpbBrVclFactor * MaxCPB，这里当vcl_hrd_parameters_present_flag 为1时，cpbBrVclFactor在表A-2中规定，BitRate[SchedSelIdx]在等式E-37中规定，CpbSize[SchedSelIdx]在等式E-38中规定。MaxBR与MaxCPB在表A-1中规定，单位分别为cpbBrVclFactor bits/s与cpbBrVclFactor bits。此比特流对于至少一个取值在0到cpb_cnt_minus1，包括0与cpb_cnt_minus1的SchedSelIdx应满足这些条件。

h) 与主要、高级、高级10、高级4:2:2或高级4:4:4简表相一致的比特流里，对于NAL HRD参数，至少一个SchedSelIdx 的值， BitRate[SchedSelIdx] <= cpbBrNalFactor * MaxBR 且 CpbSize[SchedSelIdx] <= cpbBrNalFactor * MaxCPB，这里当vcl_hrd_parameters_present_flag 为1时，cpbBrVclFactor为表A-2中规定，BitRate[SchedSelIdx]在等式E-37中规定，CpbSize[SchedSelIdx]在等式E-38中规定。MaxBR与MaxCPB在表A-1中规定，单位分别为cpbBrVclFactor bits/s与cpbBrVclFactor bits。此比特流对于至少一个取值在0到cpb_cnt_minus1，包括0与cpb_cnt_minus1的SchedSelIdx应满足这些条件。

表 A-2－cpbBrVclFactor 与 cpbBrNalFactor的规定

简表	cpbBrVclFactor	cpbBrNalFactor
高级	1 250	1 500
高级 10	3 000	3 600
高级 4:2:2	4 000	4 800
高级 4:4:4	4 000	4 800

A.3.3.1 基准简表限制

表 A-3 规定了每一级别的限制，这些级别都针对于满足基准简表的比特流。表 A-3 中标为“–”的条目，表示没有相应的限制。

表 A-3－基准简表级别限制

级别号	MaxSubMbRectSize
1	576
1b	576
1.1	576
1.2	576
1.3	576
2	576
2.1	576
2.2	576
3	576
3.1	-
3.2	-
4	-
4.1	-
4.2	-
5	-
5.1	-

A.3.3.2 主要、高级、高级10、高级4:2:2或高级4:4:4简表限制

表 A-4 规定了针对于与主要、高级、高级 10、高级 4:2:2 或高级 4:4:4 简表相一致的比特流的每一级别的限制，表 A-4 中标为“–”的条目，表示没有相应的限制。

表 A-4－主要、高级、高级10、高级4:2:2或高级4:4:4简表中级别的限制

级别号	SliceRate	MinLumaBiPredSize	direct_8x8_inference_flag	frame_mbs_only_flag
1	-	-	-	1
1b	-	-	-	1
1.1	-	-	-	1
1.2	-	-	-	1
1.3	-	-	-	1
2	-	-	-	1
2.1	-	-	-	-
2.2	-	-	-	-
3	22	-	1	-
3.1	60	8x8	1	-
3.2	60	8x8	1	-
4	60	8x8	1	-
4.1	24	8x8	1	-
4.2	24	8x8	1	1
5	24	8x8	1	1
5.1	24	8x8	1	1

A.3.3.3 扩展简表限制

表 A-5 规定了针对于与扩展简表相一致的比特流的每一级别的限制，表 A-5 中标为“–”的条目，表示没有相应的限制。

表 A-5—扩展简表级别限制

级别号	MaxSubMbRectSize	MinLumaBiPredSize	frame_mbs_only_flag
1	576	-	1
1b	576	-	1
1.1	576	-	1
1.2	576	-	1
1.3	576	-	1
2	576	-	1
2.1	576	-	-
2.2	576	-	-
3	576	-	-
3.1	-	8x8	-
3.2	-	8x8	-
4	-	8x8	-
4.1	-	8x8	-
4.2	-	8x8	1
5	-	8x8	1
5.1	-	8x8	1

A.3.4 帧速率的级别限制的影响（参考性）

本小节不是本建议书丨国际标准的组成部分。

表 A-6－某些例子中最大帧速率（帧每秒）

级别					**1**	**1b**	**1.1**	**1.2**	**1.3**	**2**	**2.1**
最大帧大小 (宏块):					**99**	**99**	**396**	**396**	**396**	**396**	**792**
宏块/秒					**1 485**	**1 485**	**3 000**	**6 000**	**11 880**	**11 880**	**19 800**
宏块大小(样点):					25 344	25 344	101 376	101 376	101 376	101 376	202 752
最大样点/秒:					380 160	380 160	768 000	1 536 000	3 041 280	3 041 280	5 068 800
格式	**亮度宽度**	**亮度高度**	**总 MBs**	**亮度样点**							
SQCIF	**128**	**96**	48	12 288	30.9	30.9	62.5	125.0	172.0	172.0	172.0
QCIF	**176**	**144**	99	25 344	15.0	15.0	30.3	60.6	120.0	120.0	172.0
QVGA	**320**	**240**	300	76 800	-	-	10.0	20.0	39.6	39.6	66.0
525 SIF	**352**	**240**	330	84 480	-	-	9.1	18.2	36.0	36.0	60.0
CIF	**352**	**288**	396	101 376	-	-	7.6	15.2	30.0	30.0	50.0
525 HHR	**352**	**480**	660	168 960	-	-	-	-	-	-	30.0
625 HHR	**352**	**576**	792	202 752	-	-	-	-	-	-	25.0
VGA	**640**	**480**	1 200	307 200	-	-	-	-	-	-	-
525 4SIF	**704**	**480**	1 320	337 920	-	-	-	-	-	-	-
525 SD	**720**	**480**	1 350	345 600	-	-	-	-	-	-	-
4CIF	**704**	**576**	1 584	405 504	-	-	-	-	-	-	-
625 SD	**720**	**576**	1 620	414 720	-	-	-	-	-	-	-
SVGA	**800**	**600**	1 900	486 400	-	-	-	-	-	-	-
XGA	**1024**	**768**	3 072	786 432	-	-	-	-	-	-	-
720p HD	**1280**	**720**	3 600	921 600	-	-	-	-	-	-	-
4VGA	**1280**	**960**	4 800	1 228 800	-	-	-	-	-	-	-
SXGA	**1280**	**1024**	5 120	1 310 720	-	-	-	-	-	-	-
525 16SIF	**1408**	**960**	5 280	1 351 680	-	-	-	-	-	-	-
16CIF	**1408**	**1152**	6 336	1 622 016	-	-	-	-	-	-	-
4SVGA	**1600**	**1200**	7 500	1 920 000	-	-	-	-	-	-	-
1080 HD	**1920**	**1088**	8 160	2 088 960	-	-	-	-	-	-	-
2Kx1K	**2048**	**1024**	8 192	2 097 152	-	-	-	-	-	-	-
2Kx1080	**2048**	**1088**	8 704	2 228 224	-	-	-	-	-	-	-
4XGA	**2048**	**1536**	12 288	3 145 728	-	-	-	-	-	-	-
16VGA	**2560**	**1920**	19 200	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	-	-	-	-	-	-	-
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	-	-	-	-	-	-	-
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	-	-	-	-	-	-
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	-	-	-	-	-	-

表 A-6 （续）一某些例子中最大帧速率（帧每秒）

级别:					**2.2**	**3**	**3.1**	**3.2**	**4**	**4.1**	**4.2**
最大帧大小 (宏块):					**1 620**	**1 620**	**3 600**	**5 120**	**8 192**	**8 192**	**8 704**
宏块/秒					**20 250**	**40 500**	**108 000**	**216 000**	**245 760**	**245 760**	**522 240**
宏块大小(样点):					414 720	414 720	921 600	1 310 720	2 097 152	2 097 152	2 228 224
最大样点/秒:					5 184 000	10 368 000	27 648 000	55 296 000	62 914 560	62 914 560	133 693 440
格式	**亮度宽度**	**亮度高度**	**总 MBs**	**亮度样点**							
SQCIF	**128**	**96**	48	12 288	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QCIF	**176**	**144**	99	25 344	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QVGA	**320**	**240**	300	76 800	67.5	135.0	172.0	172.0	172.0	172.0	172.0
525 SIF	**352**	**240**	330	84 480	61.4	122.7	172.0	172.0	172.0	172.0	172.0
CIF	**352**	**288**	396	101 376	51.1	102.3	172.0	172.0	172.0	172.0	172.0
525 HHR	**352**	**480**	660	168 960	30.7	61.4	163.6	172.0	172.0	172.0	172.0
625 HHR	**352**	**576**	792	202 752	25.6	51.1	136.4	172.0	172.0	172.0	172.0
VGA	**640**	**480**	1 200	307 200	16.9	33.8	90.0	172.0	172.0	172.0	172.0
525 4SIF	**704**	**480**	1 320	337 920	15.3	30.7	81.8	163.6	172.0	172.0	172.0
525 SD	**720**	**480**	1 350	345 600	15.0	30.0	80.0	160.0	172.0	172.0	172.0
4CIF	**704**	**576**	1 584	405 504	12.8	25.6	68.2	136.4	155.2	155.2	172.0
625 SD	**720**	**576**	1 620	414 720	12.5	25.0	66.7	133.3	151.7	151.7	172.0
SVGA	**800**	**600**	1 900	486 400	-	-	56.8	113.7	129.3	129.3	172.0
XGA	**1024**	**768**	3 072	786 432	-	-	35.2	70.3	80.0	80.0	172.0
720p HD	**1280**	**720**	3 600	921 600	-	-	30.0	60.0	68.3	68.3	145.1
4VGA	**1280**	**960**	4 800	1 228 800	-	-	-	45.0	51.2	51.2	108.8
SXGA	**1280**	**1024**	5 120	1 310 720	-	-	-	42.2	48.0	48.0	102.0
525 16SIF	**1408**	**960**	5 280	1 351 680	-	-	-	-	46.5	46.5	98.9
16CIF	**1408**	**1152**	6 336	1 622 016	-	-	-	-	38.8	38.8	82.4
4SVGA	**1600**	**1200**	7 500	1 920 000	-	-	-	-	32.8	32.8	69.6
1080 HD	**1920**	**1088**	8 160	2 088 960	-	-	-	-	30.1	30.1	64.0
2Kx1K	**2048**	**1024**	8 192	2 097 152	-	-	-	-	30.0	30.0	63.8
2Kx1080	**2048**	**1088**	8 704	2 228 224	-	-	-	-	-	-	60.0
4XGA	**2048**	**1536**	12 288	3 145 728	-	-	-	-	-	-	-
16VGA	**2560**	**1920**	19 200	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	-	-	-	-	-	-	-
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	-	-	-	-	-	-	-
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	-	-	-	-	-	-
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	-	-	-	-	-	-

表A-6（结束）－某些例子中最大帧速率（帧每秒）

级别:					**5**	**5.1**
最大帧大小 (宏块):					**22 080**	**36 864**
宏块/秒					**589 824**	**983 040**
宏块大小(样点):					**5 652 480**	**9 437 184**
最大样点/秒:					**150 994 944**	**251 658 240**
格式	**亮度宽度**	**亮度高度**	**总 MBs**	**亮度样点**		
SQCIF	**128**	**96**	48	12 288	172.0	172.0
QCIF	**176**	**144**	99	25 344	172.0	172.0
QVGA	**320**	**240**	300	76 800	172.0	172.0
525 SIF	**352**	**240**	330	84 480	172.0	172.0
CIF	**352**	**288**	396	101 376	172.0	172.0
525 HHR	**352**	**480**	660	168 960	172.0	172.0
625 HHR	**352**	**576**	792	202 752	172.0	172.0
VGA	**640**	**480**	1 200	307 200	172.0	172.0
525 4SIF	**704**	**480**	1 320	337 920	172.0	172.0
525 SD	**720**	**480**	1 350	345 600	172.0	172.0
4CIF	**704**	**576**	1 584	405 504	172.0	172.0
625 SD	**720**	**576**	1 620	414 720	172.0	172.0
SVGA	**800**	**600**	1 900	486 400	172.0	172.0
XGA	**1024**	**768**	3 072	786 432	172.0	172.0
720p HD	**1280**	**720**	3 600	921 600	163.8	172.0
4VGA	**1280**	**960**	4 800	1 228 800	122.9	172.0
SXGA	**1280**	**1024**	5 120	1 310 720	115.2	172.0
525 16SIF	**1408**	**960**	5 280	1 351 680	111.7	172.0
16CIF	**1408**	**1152**	6 336	1 622 016	93.1	155.2
4SVGA	**1600**	**1200**	7 500	1 920 000	78.6	131.1
1080 HD	**1920**	**1088**	8 160	2 088 960	72.3	120.5
2Kx1K	**2048**	**1024**	8 192	2 097 152	72.0	120.0
2Kx1080	**2048**	**1088**	8 704	2 228 224	67.8	112.9
4XGA	**2048**	**1536**	12 288	3 145 728	48.0	80.0
16VGA	**2560**	**1920**	19 200	4 915 200	30.7	51.2
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	27.2	45.3
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	26.7	44.5
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	30.0
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	26.7

应注意以下的情况：

— 本建议书｜国际标准为帧大小可变的规范，表A-6中的具体帧大小只是用于示范作用。

— 如表A-6中所示，“525”表示使用525模拟扫描线环境（大约有480线可视图像的范围）的典型用法，“625”表示用于表示使用625模拟扫描线环境（大约有576线可视图像的范围）的典型用法。

— XGA也称为SVGA，4SVGA也称为UXGA，16XGA也称为4Kx3K，CIF也称为625SIF，625HHR也称为2CIF 、半625 D-1，半625 ITU-R BT.601，525SD也称为525D-1、525 ITU-R BT.601，625SD也称为625 ITU-R BT.601。

— 给定的帧速率对于逐进扫描模式为正确的。对于帧高为32整数倍的情况下，使用交错视频，编码帧速率也是正确的。

附 件 B

字节流的格式

(本附件是本建议书 | 国际标准的组成部分)

本附件规定了字节流格式的语法与语义，这些字节流的规定用于按照顺序的字节或比特流，传送部分或全部 NAL 单元流，在该流中 NAL 单元边界的定位应该能够从数据类型中识别，如 ITU-T H.222.0 建议书| ISO/IEC 13818-1 系统或 ITU-T H.320 建议书系统。对于面向比特的传送，字节流中的比特顺序起始于第一个字节的 MSB，处理至第一个字节的 LSB，接着是第二个字节的 MSB，以此类推。

字节流格式由字节流 NAL 单元语法结构序列构成。每一字节流 NAL 单元语法结构包含有一个起始编码前缀，后面跟随一个 nal_unit(NumBytesInNALunit)语法结构。它可能（一定情况下，它应该）包含一个额外的 zero_byte 语法元素。它也可能包含一个或多个额外的 trailing_zero_8bits 语法元素。当为第一个字节流 NAL 单元时，也可能包含一个或多个额外的 leading_zero_8bits 语法元素。

B.1 字节流NAL单元语法与语义

B.1.1 字节流NAL单元语法

byte_stream_nal_unit(NumBytesInNALunit) {	C	描述符
while(next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
leading_zero_8bits /*等于 0x00 */		f(8)
if(next_bits(24) != 0x000001)		
zero_byte /* equal to 0x00 */		f(8)
start_code_prefix_one_3bytes /* 等于 0x000001 */		f(24)
nal_unit(NumBytesInNALunit)		
while(more_data_in_byte_stream( ) && next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
trailing_zero_8bits /* 等于 0x00 */		f(8)
}		

B.1.2 字节流NAL单元语义

字节流 NAL 单元的顺序应遵循 NAL 单元里包含的 NAL 单元解码顺序（见 7.4.1.2 节）。当字节流 NAL 单元里包含 NAL 单元（见 7.4.1.2.3 节）时，每一字节流 NAL 单元的内容与相同的访问单元相关联。

leading_zero_8bits 为 0x00。

注 — leading_zero_8bits语法元素只能在流的第一个字节流NAL单元里出现，因为（如B.1.1节语法图所示）任一在NAL单元语法结构后等于0x00字节，出现在一个四字节序列0x00000001前（见7.4.1.2.3节）。

zero_byte 为一个等于 0x00 的单字节。

当下述任一个条件满足时，应有 zero_byte 语法元素。

— nal_unit( )里的nal_unit_type等于7（设置了序列参数）或8（设置了图像参数）。

— 字节流NAL单元语法结构在解码顺序时包含一个访问单元的第一个NAL单元，如7.4.1.2.3节所规定。

start_code_prefix_one_3bytes 为一个 3 字节的固定值序列，等于 0x000001，该语法元素称为起始码前缀。

trailing_zero_8bits 为一个等于 0x00 的字节。

B.2 字节流NAL单元解码过程

本过程的输入由构成一组有序字节组成，该字节流由一个字节流 NAL 单元语法结构序列组成。

本过程的输出由 NAL 单元语法结构序列构成。

在解码过程开始时，解码器把其当前的位置初始化为字节流的起始位置。然后提取，并丢弃每一个 leading_zero_8bits 语法元素（如果存在的话），移动当前位置至某一时刻的字节处，直到比特流的当前位置紧接的四个字节为四字节序列 0x00000001。

解码器此时重复执行下述按步骤的过程，对字节流中每一个 NAL 单元语法结构进行提取与解码，直到字节流结尾（由未规定的方式判决），并且字节流中最后一个 NAL 单元也已经解码：

1．当字节流里的紧接的四个字节构成四字节序列 0x00000001，对比特流中下一个字节（为 zero_byte 语法元素）进行提取并丢弃时，字节流的当前位置设为紧接被丢弃的字节的字节位置。

2．提取与丢弃比特流中下一个三字节序列（为 start_code_prefix_one_3bytes），且比特流当前位置设为此紧接被丢弃的 3 字节序列的字节的位置。

3．NumBytesInNALunit 设为自当前字节位置起至下述条件前的位置的最后一个字节，且包括最后一个字节的编号。

a．一个三字节序列的排列等于0x000000，或

b．一个三字节序列的排列等于0x000001，或

c．字节流的结束，由未规定的方式判决。

4．NumBytesInNALunit 字节从比特流中移除，字节流的当前位置前移 NumBytesInNALunit 字节。这个字节序列为 nal_unit(NumBytesInNALunit)，并用 NAL 单元解码过程进行解码。

5．当字节流中的当前位置不为字节流的结尾（由未规定的方式判决），且字节流中一个字节不是等于 0x000001 开始的三字节序列，也不是等于 0x00000001 开始的四字节序列。解码器提取并丢弃每一个 trailing_zero_8bits 语法元素，移动字节流中的当前位置到某一时刻的一个字节处，直到字节流里的当前位置接下的四个字节构成四字节的序列 0x00000001 或已至字节流的结尾（由未规定的方式判决）。

B.3 解码器字节定界恢复（参考性）

本小节不是本建议书｜国际标准的组成部分。

很多应用向解码器提供采用字节本身定界的数据，这样不需要本节所描述的检测面向比特的字节定界。

当可以判定比特流中的位置是否为字节定界时，解码器称为可以对比特流进行字节定界的。当一个解码器不能与编码器字节流的字节定界时，解码器可能会检测输入的比特流中的二进制图像'00000000 00000000 00000000 00000001'（31 个连续比特等于 0 后，1 个比特等于 1）。紧随这个图样的比特为起始码前缀定界字节的第一个比特。直到检测到这个图样，解码器将与编码器按字节定界，并定位于字节流中 NAL 单元的起始位置。

当与编码器的字节对位后，解码器可以从流入的字节流里查找三字节的序列 0x000001 与 0x000003。

当三字节序列 0x000001 被检测到时，意味着起始编码前缀。

当检测到三字节序列 0x000003，第三字节（0x03）如果是 mulation_prevention_three_byte 时，将按 7.4.1 节所规定被丢弃。

当检测到比特流语法的错误（比如 orbidden_zero_bit 的非 0 值或 7.4.1 中所规定禁止使用的三字节序列之一或四字节序列之一）时，解码器可能认为检测到的情况意味着字节定界的丢失，丢弃所有比特数据，直到重新检测到比特流中稍后的字节定界，如本节所述。

附 件 C

假定参考解码器

(本附件是本建议书｜国际标准的组成部分)

本附件规定了假定参考解码器（HRD）及它如何用来检查比特流与解码器一致性。

对于本建议书｜国际标准，有两类比特流是 HRD 一致性检查的对象：第一类比特流称为类型 I 比特流，为 NAL 单元流，只包含有 VCL NAL 单元与在比特流里对于所有访问单元的填充 NAL 单元数据。第二类比特流，称为类型 II 比特流，除了包含有 VCL NAL 单元与在比特流里对于所有访问单元的填充数据 NAL 单元，至少还包含一个下述的内容：

— 附加的不为填充数据NAL单元的非VAL NAL单元；

— 所有leading_zero_8bits，zero_byte，start_code_prefix_one_3bytes和 trailing_zero_8bits语法元素，这些语法元素构成来自于NAL单元流的字节流（如附件B所规定）。

图 C-1 显示了 HRD 检查的比特流类型一致性点。

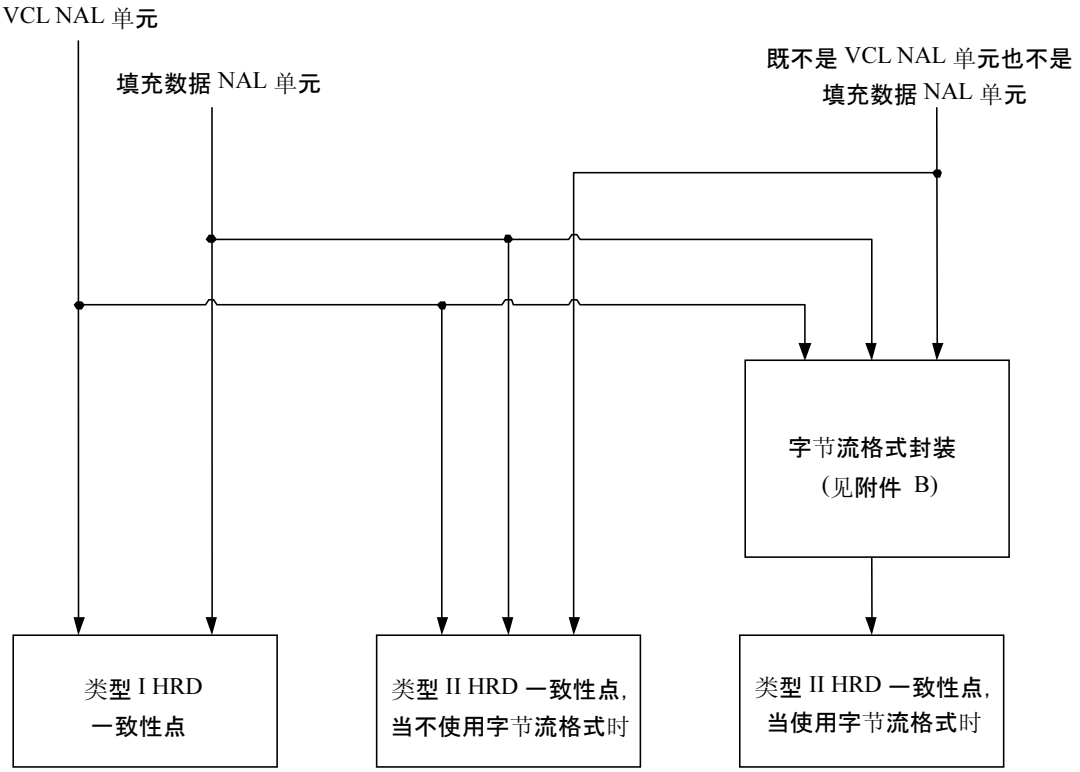


图 C-1－字节流的结构与用于HRD一致性检查的NAL单元流

第 7 节语义与附件 D 与 E 中规定了 HRD 所需要的非 VCL NAL 单元的语法元素（或它们对某些语法元素的默认值）。

有两类 HRD 参数集可供使用。HRD 参数集通过视频可用性信息传送，如 E.1 与 E.2 节所规定，视频可用性信息是序列参数集语法结构的一部分。

为了检查使用 HRD 比特流的一致性，所有的涉及 VCL NAL 单元的序列参数集与图像参数集，及对应于缓存周期与图像定时 SEI 消息应及时传送到 HRD，或者以比特流形式（非 VCL NAL 单元），或者以其它本建议书｜国际标准未规定的方式。

在附件 C、D 与 E 中，当这些 NAL 单元（或仅是某些）以本建议书 | 国际标准未定义的方式传送到解码器（或到 HRD），对于非 VCL NAL 单元“存在”的规定也应得到满足。为了对比特计数，只有实际在比特流中出现的比特才能进行计数。

注1 — 作为一个例子，非VAL NAL单元与比特流中的NAL单元同步，采用通过比特流携带以外的方式，可以通过指定一个比特流中的两点来实现，如果编码器决定在比特流中传送时，在此两点间，非VCL NAL单元可能在比特流中出现。

当非 VCL NAL 单元的内容为某种应用，通过某些方式在比特流中传送时，非 VCL NAL 单元内容的表现不需要使用本附件规定的同样语法。

注2 — 当HRD信息包含在比特流中时，可能仅靠比特流中的这些信息用来检证比特流是否与本节所述需求相一致。当HRD信息没有出现在比特流中时，比如所有独立的类型I比特流的情况里，一致性可能只用于验证何时由本建议书 | 国际标准未规定的方式提供HRD数据。

HRD 包含编码图像缓存（CPB）、实时解码过程、解码图像缓存（DPB）及输出裁切，如图 C-2 所示。

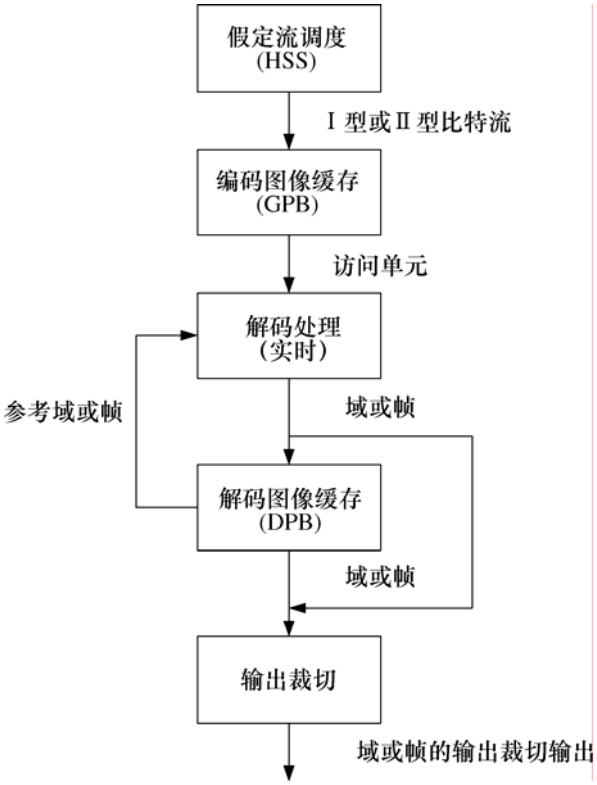


图 C-2－HRD缓存模型

CPB 大小（比特数）为 CpbSize[SchedSelIdx]，DPB 的大小（帧数）为 Max(1, max_dec_frame_buffering)。

HRD 按照下文所述进行操作。由 HSS 传送与访问单元关联的数据，这些数据按规定的到达时间表注入 CPB 中。与每一个访问单元相关联的数据在 CPB 移除时间到达时，被移除并由实时解码过程进行实时解码。每一解码的图像在其 CPB 移除时间到达时放入到 DPB 中，除非要在 CPB 移除时间时输出，并且是非参考图像。当一幅图像放入 DPB，在稍后 DPB 输出时间时或在其标记为“未用于参考”的时间，从 DPB 中移除。

CPB 的操作在 C.1 节中规定，实时解码操作在第 8 与第 9 节中规定。DPB 的操作在 C.2 节中规定。输出的裁切在 C.2.2 节中规定。

HSS 与 HRD 信息在 E.1.1、E.1.2、E.2.1 与 E.2.2 规定，这些信息主要是关于枚举出的传送进度表数目与其相关的比特率和缓存器大小。HRD 按 D.1.1 与 D.2.1 节中描述的缓存周期 SEI 消息的规定进行初始化。从 CPB 移除访问单元的时间与从 DPB 输出的时间按 D.1.2 与 D.2.2 节中描述的图像定时 SEI 消息的规定进行。所有与具体访问单元相关联的定时信息应先于访问单元的 CPB 移除时间到达。

使用 HRD 检查比特流与解码器的一致性，如 C.3 与 C.4 节所分别规定。

注3 — 如下假设保证了一致性：所有用于产生比特流的帧速率和时钟与比特流里的传送的值完好吻合，但在实际环境中，每一个参数与传送值或规定值均会有所差别。

本附件中所有计算均使用实际值，这样就不会引起舍入误差的传播。比如 CPB 里的比特数在一访问单元前或移除后不是必须为整数。

变量 t_c 称为一个时钟记号，推导如下。

$$t_c = num\_units\_in\_tick \div time\_scale \quad (C\text{-}1)$$

如下内容为本附件的限定条件。

— 使访问单元n在解码顺序中为第n个访问单元，第一个访问单元为0；

— 使图像n为主要编码图像或访问单元n的主要解码图像。

C.1 编码图像的缓存操作(CPB)

本节中规定独立应用于每一个出现的 CPB 参数集及图 C-1 中示类型 I 与类型 2 一致性点。

C.1.1 比特流到达的定时

HRD 可以初始化任一个缓存周期 SEI 消息。在初始化前，CPB 为空。

注 — 在初始化后，HRD不能再被随后的缓存周期SEI消息进行初始化。

每一个访问单元都使用访问单元 n 来引用，这里用编号 n 来标识特定的访问单元。与缓存周期 SEI 消息相关，并初始化 CPB 的访问单元称为访问单元 0。数值 n 会在解码过程中，因随后每一个访问单元增加 1。

访问单元 n 的第一个比特进入 CPB 的时刻称为起始到达时间 $t_{ai}(n)$。

访问单元的起始到达时间的由如下方法得出：

— 如果访问单元访为访问单元0，则$t_{ai}(0) = 0$；

— 否则（访问单元为访问单元n，n>0），按照如下规则：

— 如果cbr_flag[SchedSelIdx]等于1，访问单元n的起始到达时间等于访问单元n-1的最后到达时间（如下所示），即：

$$t_{ai}(n) = t_{af}(n - 1) \quad (C\text{-}2)$$

— 否则（cbr_flag[SchedSelIdx]等于0），访问单元n的起始到达时间：

$$t_{ai}(n) = Max(t_{af}(n - 1), t_{ai,earliest}(n)) \quad (C\text{-}3)$$

这里$t_{ai,earliest}(n)$推导如下：

— 如果访问单元n不为后续缓存周期的第一个访问单元，$t_{ai,earliest}(n)$经

$$t_{ai,earliest}(n) = t_{r,n}(n) - (initial\_cpb\_removal\_delay[SchedSelIdx] + initial\_cpb\_removal\_delay\_offset[SchedSelIdx]) \div 90000 \quad (C\text{-}4)$$

导出， $t_{r,n}(n)$ 为访问单元应从CPB中移除的时刻，如C.1.2节所规定，initial_cpb_removal_delay[SchedSelIdx]与initial_cpb_removal_delay_offset[SchedSelIdx] 为前一缓存周期SEI消息的规定的值。

— 否则（访问单元n为后续访问单元中的第一个），$t_{ai,earliest}(n)$导出如下：

$$t_{ai,earliest}(n) = t_{r,n}(n) - (initial\_cpb\_removal\_delay[SchedSelIdx] \div 90000) \quad (C\text{-}5)$$

initial_cpb_removal_delay[SchedSelIdx]为与访问单元 n 相关的缓存周期 SEI 消息中所定义的。

访问单元 n 的最终到达时间由下式导得：

$$t_{af}(n) = t_{ai}(n) + b(n) \div BitRate[SchedSelIdx] \quad (C\text{-}6)$$

这里 b(n)为比特为单位访问列表 n 的大小，对 VCL NAL 单元与类型 I 一致性点的填充数据 NAL 单元或对类型 II 一致性点的类型 II 比特流中所有比特进行计数，类型 I 与类型 II 的一致性点如图 C-1 所示。

SchedSelIdx，BitRate[SchedSelIdx]与 CpbSize[SchedSelIdx]的值受下述条件约束：

— 如果访问单元n与访问单元n-1为不同编码视频序列的一部分，两编码视频序列的激活序列参数集的内容不同，HSS从SchedSelIdx值中选择SchedSelIdx中SchedSelIdx1的值，SchedSelIdx包含访问单元n的编码视频序列，该访问单元n产生了两编码视频序列中第二个编码视频序列（此序列包含访问单元n）的BitRate[SchedSelIdx1]或 CpbSize[SchedSelIdx1]参数。BitRate[SchedSelIdx1]或CpbSize[SchedSelIdx1]的值。对于 SchedSelIdx 中 SchedSelIdx0 的值， BitRate[SchedSelIdx1] 或 CpbSize[SchedSelIdx1] 可能与 BitRate[SchedSelIdx0] 或CpbSize[SchedSelIdx0]的值不同，SchedSelIdx的SchedSelIdx0的值用于包含访问单元n-1的编码视频序列。

— 否则，HSS继续依照SchedSelIdx、BitRate[SchedSelIdx] 与 CpbSize[SchedSelIdx]的前一值进行操作。

当HSS选择不同于前一访问单元的BitRate[SchedSelIdx]或CpbSize[SchedSelIdx]的值时，将应用下述条件：

— 变量BitRate[SchedSelIdx]在$t_{ai}(n)$生效；

— 变量CpbSize[SchedSelIdx]如下所述生效：

— 如果CpbSize[SchedSelIdx]的新值超过原CPB的大小，将在$t_{ai}(n)$产生作用；

— 否则CpbSize[SchedSelIdx]的新值将在$t_r(n)$产生作用。

C.1.2 编码图像的移除的定时

对于访问单元 0，规定从 CPB 的移除时间定义如下：

$$t_{r,n}(0) = initial\_cpb\_removal\_delay[SchedSelIdx] \div 90000 \quad (C\text{-}7)$$

对于不由 HRD 发起的缓存周期的第一个访问单元，从 CPB 的移除时间定义如下：

$$t_{r,n}(n) = t_{r,n}(n_b) + t_c * cpb\_removal\_delay(n) \quad (C\text{-}8)$$

$t_{r,n}(n_b)$为前一个缓存周期的第一个访问单元的移除时间，cpb_removal_delay(n)为与访问单元 n 相关的图像定时 SEI 消息中规定的 cpb_removal_delay 值。

当一个访问单元 n 为缓存周期的第一个访问单元时，n_b在访问单元 n 移除时设为 n。

缓存周期内，不为第一个访问单元的访问单元 n 的移除时间 $t_{r,n}(n)$由下式给出：

$$t_{r,n}(n) = t_{r,n}(n_b) + t_c * cpb\_removal\_delay(n) \quad (C\text{-}9)$$

$t_{r,n}(n_b)$为当前缓存周期第一个访问单元规定的移除时间，cpb_removal_delay(n)为与访问单元 n 相关的图像 SEI 消息中规定的 cpb_removal_delay 的值。

访问单元 n 的移除时间定义如下：

— 如果low_delay_hrd_flag等于0或$t_{r,n}(n) >= t_{af}(n)$，访问单元n的移除时间规定为：

$$t_r(n) = t_{r,n}(n) \quad \text{(C-10)}$$

— 否则（low_delay_hrd_flag等于1或$t_{r,n}(n) < t_{af}(n)$），移除时间规定如下：

$$t_r(n) = t_{r,n}(n) + t_c * Ceil((t_{af}(n) - t_{r,n}(n)) \div t_c) \quad \text{(C-11)}$$

注 — 后一种情况表明访问单元n，b(n)的尺寸很大，可以防止规定移除时间的移除。

C.2 解码图像缓存的操作(DPB)

解码图像缓存包含有帧缓存，每一帧缓存可能包括一个解码帧，一解码补偿场对或单（非成对）解码场，这些内容标记为“用于参考”（参考图像）或为以后的输出（录像或延时图像）保留。在初始化之前，DPB 为空（CPB 的填充度设为 0）。本节下述各小节中的步骤均在 $t_r(n)$时刻按照列出的顺序实时发生。

C.2.1 frame_num 间隔的解码与“不存在”帧的存储

如果可行，frame_num 里的间隔由解码过程检测，对生成帧进行标记并插入到 DPB 中，如下文所述：

frame_num 里的间隔由解码过程检测出来，产生的帧按照 8.2.5.2 节的规定进行标记。

标记每一个生成帧后，由滑动窗过程标记为“未用参考”每一图像 m 从 DPB 中移除，图像 m 也标记为“不存在”或其 DPB 输出时间小于或等于当前 CPB 的移除时间，将从 DPB 中移除，即，$t_{o,dpb}(m) <= t_r(n)$。当帧缓存里一帧或最后一场从 DPB 缓存里移除，DPB 填充度减一。“不存在”帧插入到 DPB，DPB 填充度加 1。

C.2.2 图像解码与输出

解码图像 n，其 DPB 输出时间 $t_{o,dpb}(n)$由下式导出：

$$t_{o,dpb}(n) = t_r(n) + t_c * dpb\_output\_delay(n) \quad \text{(C-12)}$$

当前图像的输出规定如下：

— 如果$t_{o,dpb}(n) = t_r(n)$，当前图像输出；

注 — 当前图像为参考图像时，将被存储在DPB。

— 否则（$t_{o,dpb}(n) > t_r(n)$），当前图像稍后输出，并将存储于DPB（如C.2.4节所述），在$t_{o,dpb}(n)$时输出，除非在$t_{o,dpb}(n)$时间前，解码过程或no_output_of_prior_pics_flag等于1，指明不能输出。

输出图像应被裁切，裁切使用序列参数集所规定的裁切矩形。

当图像 n 为一输出图像，且不为输出比特流的最后一帧时，$\Delta t_{o,dpb}(n)$的值定义为：

$$\Delta t_{o,dpb}(n) = t_{o,dpb}(n_n) - t_{o,dpb}(n) \quad \text{(C-13)}$$

n_n 为输出顺序中图像 n 后的那幅图像。

解码图像要临时进行存储（不是在 DPB 中）。

C.2.3 在当前图像可能插入前，从DPB中移除图像

在当前图像可能插入前，从 DPB 中移除图像的过程如下所述：

— 如果解码图像为IDR图像，如下所述：

— 所有在DPB中的参考图像标记为“未用于参考”如8.2.5.1节所述；

— 当IDR图像不是第一幅解码IDR图像，且从序列参数集中导出的PicWidthInMbs或FrameHeightInMbs或max_dec_frame_buffering的值与前面序列激活的序列参数集中的PicWidthInMbs或FrameHeightInMbs或max_dec_frame_buffering的值不同时，不考虑no_output_of_prior_pics_flag的实际值，no_output_of_prior_pics_flag通过HRD可以推断出等于1。

注 — 同与PicWidthInMbs或FrameHeightInMbs的改变相关的HRD相比，解码器操作应尝试更好地处理帧或DPB大小的变化。

— 当no_output_of_prior_pics_flag设为1或推断将为1时，所有的DPB中帧缓存清空而不输出，DPB填充度设为0。

— 否则（解码图像不是IDR图像），如下所述：

— `如果当前图像的条带头包括了等于5的memory_management_control_operation，所有DPB中的参考图像标记为“未用作参考”；

— 否则（当前图像的条带头不包括等于5的memory_management_control_operation），解码参考图像标记过程如8.2.5节所规定进行调用。

所有 DPB 中的图像 m，若下述条件为真，从 DPB 中移除。

— 图像m标记为“未用做参考”或图像m为非参考图像。当图像为参考帧，仅当它的两场均被标为“未用做参考”，它会被标记为“未用做参考”。

— 当图像m标记为“不存在”，或其DPB输出时间小于或等于CPB中当前图像n的移除时间，即$t_{o,dpb}(m) <= t_r(n)$。

当帧缓存中的一帧或最后一场从 DPB 中移除时，DPB 的填充度减一。

C.2.4 当前解码图像标记与存储

C.2.4.1 DPB里标记与存储一参考解码图像

当当前图像为参考图像时，它在 DPB 中存储如下：

— 如果当前解码图像为补偿参考场对中的第二场（在解码顺序中），场对中的第一场仍在DPB中，当前解码图像如第一场一样存在同一帧缓存中。

— 否则当前解码图像存在一空的帧缓存中，DPB填充度加1。

C.2.4.2 向DPB中存储非参考图像

如果当前图像为非参考图像且当前图像 n 有 $t_{o,dpb}(n) > t_r(n)$，按下述存储在 DPB 中：

— 如果当前解码图像为一补偿非参考场对中的第二场，第一场仍在DPB中，当前解码图像与第一场存于同一帧缓存中。

— 否则当前解码图像存于一空帧缓存中，DPB的填充度加1。

C.3 比特流一致性

编码数据的比特流若与本建议书｜国际标准一致，需满足如下要求。

按照语法、语意与本建议书｜国际标准规定的限制构成的比特流不在本附件范围内。

由 HRD 测试的比特流规定如下：

对于类型 I 比特流，进行的测试数等于 cpb_cnt_minus1 + 1，这里 cpb_cnt_minus1 或者是在 vcl_hrd_parameters_present_flag 之后的 hrd_parameters( )的语法元素，或使用由本建议书｜国际标准中未规定方式的应用来决定。一个测试对在 vcl_hrd_parameters_present_flag 之后 hrd_parameters( )所规定的每一种比特率与 CPB 大小的组合进行。每一测试由图 C-1 所示的类型 I 的一致性点所指导。

对于类型 II 比特流，有两个测试集。第一个集的测试号等于 cpb_cnt_minus1 + 1，这里 cpb_cnt_minus1 + 1 或者是在 vcl_hrd_parameters_present_flag 之后的 hrd_parameters( )的语法元素，或由本建议书｜国际标准内未规定的其它方式所判定。一个测试对每一种比特率与 CPB 大小的组合进行。每一个测试由图 C-1 所示的类型 I 一致性点所指导。对于这些测试，只有 VCL 与填充数据 NAL 单元用于输入比特率与 CPB 存储的计数。

对于类型 II，第二个集的测试号，等于 cpb_cnt_minus1 + 1，这里 cpb_cnt_minus1 + 1 或者是在 vcl_hrd_parameters_present_flag 之后的 hrd_parameters( )的语法元素，或者由本建议书｜国际标准内未规定的其它方式所判定。一个测试对每一种比特率都与在 vcl_hrd_parameters_present_flag 之后 hrd_parameters( )所规定的 CPB 大小组合都进行。每一个测试由图 C-1 所示的类型 II 一致性点所引导。对于这些测试，所有（类型 II NAL 单元流的）NAL 单元或（字节流）的所有字节用于输入比特率与 CPB 存储的计数。

注1 — 在VBR情况下，相同initial_cpb_removal_delay[SchedSelIdx], BitRate[SchedSelIdx], 和CpbSize[SchedSelIdx]的值，对于图C-1所示的类型II一致性点，由SchedSelIdx建立的NAL HRD参数同样足够来建立用于图C-1所示类型I的一致性点的VCL HRD一致性。这是因为满足类型I一致性点数据流是满足类型II一致性点数据流的子集，同时对于VBR情况下，CPB允许为空并保持空的状态，直到计划中下一个图像开始到达。比如，当类型II一致性点提供NAL HRD参数时，此时类型II一致性点不仅满足A.3.1节的项目j或A.3.3节（根据使用的简表）的项目i中简表一致性所需的NAL HRD参数范围集，也满足A.3.1的项目i或A.3.3节项目h中简表一致性所需的VCL HRD参数范围集，对于类型I一致性点，VCL HRD的一致性保证能满足在A.3.1节中所述项目i的范围内。

对于一致的比特流，下述所有情况应均被满足：

— 对于每一与缓存周期SEI消息想关的访问单元n，n>0，有$\Delta t_{g,90}(n)$参数，其规定如下：

$$\Delta t_{g,90}(n) = 90000 * (t_{r,n}(n) - t_{af}(n - 1)) \quad \text{(C-14)}$$

initial_cpb_removal_delay[SchedSelIdx]的取值受如下限制：

— 如果cbr_flag[SchedSelIdx]为0，

$$\text{initial\_cpb\_removal\_delay[SchedSelIdx]} <= \text{Ceil}(\Delta t_{g,90}(n)) \quad \text{(C-15)}$$

— 否则（cbr_flag[SchedSelIdx]不为0，）

$$\text{Floor}(\Delta t_{g,90}(n)) <= \text{initial\_cpb\_removal\_delay[SchedSelIdx]} <= \text{Ceil}(\Delta t_{g,90}(n)) \quad \text{(C-16)}$$

注2 — 在每一图像的移除时间，在CPB中准确的比特数可能与被选择对HRD进行初始化的缓存周期SEI消息有关。HRD可能由任一个缓存周期SEI参数初始化，不论选择哪一个缓存周期SEI消息来对HRD进行初始化，编码器必须考虑到这些情况保证所有规定的限制必须被遵守。

— CPB溢出流规定为CPB里的比特总数大于CPB的大小，CPB不应溢出。

— CPB下溢流规定为$t_{r,n}(n)$小于$t_{af}(n)$，当low_delay_hrd_flag等于0，CPB不应下溢。

— 图像从CPB移除的时间（从解码顺序中的第二幅图像开始），应满足$t_{r,n}(n)$与$t_r(n)$的限制，如A.3.1到A.3.3节对比特流的简表与级别所述。

— 任一解码图像放入DPB中之后，DPB的填充度应小于或等于DPB大小，DPB的大小限制由附件A、附件D与附件E中对比特流内的简表与级别规定。

— 当需要预测时，所有参考图像应在DPB中。每一图像在它DPB输出时间时均应在DPB中，除非它根本就不在DPB中，或在输出前由C.2节中规定的某种过程移除。

— $\Delta t_{o,dpb}(n)$由等式C-13给出，它是输出顺序中一幅图像时间与之后下一幅图像输出时间之差，该值应满足A.3.1中所述对于对比特流规定的简表与级别的限制。

C.4 解码一致性

与本建议书｜国际标准相一致的解码器需满足以下要求。

假设 VCL NAL 单元中的序列参数集与图像参数集，适当的缓存周期与图像定时 SEI 消息或以比特流形式（非 VCL NAL 单元）或以本建议书｜国际标准内未定义的外部方式，及时传送到解码器，如果解码器声称与某一特定简表和级别相一致，那么它应该可以成功解码所有 C.3 节中为解码器一致性所规定的比特流。

解码器可以宣称满足的一致性类型有两种，输出定时一致性与输出顺序一致性。

为检查解码器的一致性，与 C.3 节规定的简表与级别一致的测试比特流通过假想流调度（HSS）发送至 HRD 与被测解码器（DUT）。所有由 HRD 输出的图像也应由 DUT 输出，对于 HRD 输出的每一幅图像，由 DUT 输出的样点的值应与 HRD 输出样点的值相等。

对于输出定时解码器一致性，HSS 操作如上所述，传送进度表只从 SchedSelIdx 的值的子集中选取，对于 SchedSelIdx，比特率与 CPB 的大小如附件 A 中对规定简表与级别所描述的一样进行限制，或者带有插入的进度表，如下文所述，对于插入的进度表，比特率与 CPB 的大小如附件 A 中对规定简表与级别所描述的一样进行限制。同一传送进度表同时用于 HRD 与 DUT。

当 HRD 参数与缓存周期 SEI 消息中带有大于 0 的 cpb_cnt_minus1 时，解码器应可以对比特流进行解码，比特流由使用插入传送进度表的 HSS 操作传送，插入进度表的规定包括峰值速率 r，CPB 大小 c（r），与起始 CPB 移除延时 delay (f(r) , r)分别如下所规定：

$$\alpha = (r - BitRate[SchedSelIdx - 1]) \div (BitRate[SchedSelIdx] - BitRate[SchedSelIdx - 1]), \quad \text{(C-17)}$$

$$c(r) = \alpha * CpbSize[SchedSelIdx] + (1 - \alpha) * CpbSize[SchedSelIdx-1], \quad \text{(C-18)}$$

$$f(r) = \alpha * initial\_cpb\_removal\_delay[SchedSelIdx] * BitRate[SchedSelIdx] + (1 - \alpha) * initial\_cpb\_removal\_delay[SchedSelIdx - 1] * BitRate[SchedSelIdx - 1] \quad \text{(C-19)}$$

对于任何 SchedSelIdx > 0 与 r，这样的 BitRate[SchedSelIdx − 1] <= r <= BitRate[SchedSelIdx]，r 与 c（r）在附件 A 所规定对于某简表或级别的最大比特速率范围之内。

注1 — 起始cpb_removal_delay[SchedSelIdx]可能因缓存周期不同而不同，需要重新进行计算。

对于输出定时解码器一致性，如上所述的 HRD 会被使用，图像输出的定时（与第一比特传送时间相关）对于 HRD 与 DUT 是相同的，为一个固定时延。

对于输出顺序解码器一致性，HSS 按照 DUT 的要求传送比特流到 DUT，意味着只有当 DUT 需要更多比特进行处理时，HSS 才传送比特（按解码顺序）。

注2 — 这意味着对于该测试，DUT的编码图像缓存应该和最大访问单元的尺寸一样小。

使用如下所述的一个修改过的解码器，HSS 由一个进度表向 HRD 传送比特流，此进度表在比特流中规定比特率与 CPB 的大小，如附件 A 中所规定。图像输出顺序对于 HRD 与 DUT 应是一致的。

对于输出顺序解码器一致性，HRD CPB 大小对于所选的进度表等于 CpbSize[SchedSelIdx]，DPB 的大小等于 MaxDpbSize。HRD 从 CPB 的移除时间等于最后一比特的到达时间，解码也立即开始。HRD 的 DPB 操作如下文所述。

C.4.1 DPB输出顺序操作

解码图像缓存包含帧缓存，每一帧缓存可能包含一个解码帧，一个解码补偿场对或一单独的（非成对）标记为“用于参考”或为以后输出做保留（录像图像）的解码场。在 HRD 初始化时，DPB 的填充度以帧为单位衡量，设为 0。以后步骤，当一访问单元从 CPB 移除时，都同时发生，并按照列出的顺序进行。

C.4.2 frame_num 里间隔的解码与“不存在”图像的存储

如果可行，frame_num 由解码处理过程检测到，“不存在”帧的必要数目按照等式 7-21 的 UnusedShortTermFrameNum 值的产生推出，并按 8.2.5.2 节的规定进行标记。帧缓存中包含一个帧或一补偿场对或非成对场，这些内容标记为“不需输出”与“未用于参考”，此时帧缓存清空（没有输出），DPB 的填充度减去帧缓存清除的数目。每一“不存在”帧按如下所述存于 DPB：

— 当没有空帧缓存（比如，DPB的填充度等于DPB的大小）时，C.4.5.3节中规定的排除处理过程将被重复调用，直到有一个用于存储“不存在”帧的空的帧缓存。

— “不存在”帧存于空的帧缓存中，标记为“不需输出”，DPB的填充度加1。

C.4.3 图像解码

主编码图像 n 进行解码并临时进行存储（不在 DPB 中）。

C.4.4 在当前图像可能插入前，从DPB中移除图像

在当前图像可能插入前，从 DPB 中移除图像，如下所述：

— 如果解码图像是一个IDR图像，应用如下过程：

— 所有DPB里的参考图像标记为“未用于参考”，如8.2.5节所述。

— 当IDR图像不是第一幅解码IDR图像，且从活动的序列参数集中导出的PicWidthInMbs值或FrameHeightInMbs值或max_dec_frame_buffering的值与之前序列激活的序列参数集中的PicWidthInMbs或FrameHeightInMbs或max_dec_frame_buffering的值不同时，no_output_of_prior_pics_flag可以通过HRD推出为1，不需要考虑no_output_of_prior_pics_flag的值。

注 — 与HRD改变PicWidthInMbs或FrameHeightInMbs值相比，解码器应尝试更好地处理帧或DPB大小的变化。

— 当no_output_of_prior_pics_flag设为1或推断将为1时，所有的DPB中帧缓存清空，缓存中的图像不输出，DPB填充度设为0。

— 否则（解码图像不为一IDR图像），调用解码参考图像标记处理过程，如8.2.5节所述。帧缓存包含一个帧或一补偿场对或非成对场，这些内容标记为“不需输出”与“未用于参考”，此时帧缓存清空（没有输出），DPB的填充度减去帧缓存清除的数目。

当当前图像包含值为 5 的 memory_management_control_operation，或者是 no_output_of_prior_pics_flag 不等于 1 且不会推断出为 1 的 IRD 图像时，进行下述步骤；

1．帧缓存包含一个帧或一补偿场对或非成对场，这些内容标记为“不需输出”与“未用于参考”，此时帧缓存清空（没有输出），DPB 的填充度减去帧缓存清除的数目。

2．所有 DPB 中非空帧缓存由重复调用的 C.4.5.3 节规定的排除过程进行清除，DPB 的填充度设为 0。

C.4.5 当前解码图像标记存储

C.4.5.1 在DPB里存储与标记一参考解码图像

当当前图像为一参考图像时，它如下所述存储于 DPB 中：

— 如果当前解码图像为补偿参考场对中的第二场（在解码顺序中），场对中的第一场仍在DPB中，当前解码图像如第一场一样存在同一帧缓存中，并标记为“需要输出”。

— 否则进行如下操作。

— 当没有空帧缓存（比如，DPB的填充度等于DPB的大小）时，C.4.5.3节中规定的排除处理过程将被重复调用，直到有一个用于存储“不存在”帧的空的帧缓存。

— 当前解码图像存储在空的帧缓存中，并标记为“需要输出”，DPB的填充度加1。

C.4.5.2 在DPB里存储与标记一幅非参考解码图像

当当前图像为非参考图像时，进行下述操作：

— 如果当前解码图像为一补偿非参考场对中的第二场，第一场仍在DPB中，当前解码图像与第一场存在同一帧缓存中，并标为“需要输出”。

— 否则，进行重复进行下述操作，直到解码图像被裁切并输出或存于DPB中：

— 如果没有空的帧缓存（也就是DPB的填充度等于DPB的大小），进行如下操作。

— 如果当前图像没有低于DPB中标为“需要输出”的所有图像的PicOrderCnt( )值，执行C.4.5.3节所描述的排除过程。

— 否则（当前图像PicOrderCnt()值低于所有DPB里的标为“需要输出”图像），对当前图像进行裁切，使用的裁切矩形在序列参数集里定义，经过裁切的图像进行输出。

— 否则（有空的帧缓存，比如DPB的填充度小于DPB的大小），当前解码图像存于空的帧缓存中，标为“需要输出”，DPB的填充度减1。

C.4.5.3 排除过程

在下述情况下调用排除过程：

— 没有空的帧缓存（比如，DPB的填充度等于DPB的大小），需要一个空的帧缓存存储“不存在”帧，如C.4.2节所述；

— 当前图像为一IDR图像，且no_output_of_prior_pics_flag不等于1，也不会等于1，如C.4.4节所述；

— 当前图像memory_management_control_operation等于5，如C.4.4节所述；

— 没有空的帧缓存（比如DPB的填充度等于DPB的大小），且需要一个空的帧缓存存储解码参考图像（非IDR），如C.4.5.1节所述；

— 没有空的帧缓存（比如DPB的填充度等于DPB的大小），且当前图像不是非参考图像，不是补偿非参考场对中的第二场，DPB中的图像按输出顺序先于当前非参考图像，图像标记为“需要输出”时，如C.4.5.2节所述，则需要一个空的缓存存储当前图像。

排除过程包括如下步骤：

— 按如下条件进行选择图像或输出的第一个补偿参考场对：

— 选中的帧缓存包含的图像，具有DPB中所有标记为“需要输出”图像中最小的PicOrderCnt( )值。

— 如果帧缓存包含的补偿非参考场对的两场均标记为“需要输出”，且两场有相同的PicOrderCnt()，补偿参考场的第一个作为首先输出的对象。

— 否则，如果帧缓存包含的补偿参考场对的两场均标记为“需要输出”，且两场有相同的PicOrderCnt()，整个补偿参考场作为首先输出的对象。

— 否则，在帧缓存中有最小PicOrderCnt()值的图像最先输出。

— 如果一幅单独的图像作为最先排除，这图像用序列的序列参数集中规定的裁切矩形进行裁切，经裁切的图像排除，这幅图像标记为“不需输出”。

— 否则，（一个补偿参考场对作为最先排除），补偿参考场对的两场均经裁切，裁切所用的矩形在序列的序列参数集中规定，补偿参考场对中的两场裁切后同时排除，两场均标记为“不需输出”。

— 帧缓存中包括裁切后的图像或补偿参考场对，并允许进行排除，当下述任一个条件满足，帧缓存清空，DPB的填充度减1。

— 帧缓存里包含一个非参考非成对场；

— 帧缓存里包含非参考帧；

— 帧缓存包含一个补偿非参考场对，此参考场对中两场均标为“不用作输出”；

— 帧缓存包含一个非成对参考场，标为“不用作参考”；

— 帧缓存包含一个参考帧，此参考帧中两场均标为“不用作参考”

— 帧缓存包含一个补偿参考场对，此参考场中两场均标为“不用于参考”与“不需要输出”。

附 件 D

辅助增强信息

(本附件是本建议书 | 国际标准的组成部分)

本附件规定辅助增强信息（SEI）载荷的语法和语义。

SEI 消息在解码、显示或其它过程中起辅助作用。不过在解码处理时，构建亮度或者色度样值是不需要 SEI 消息的。在本建议书 | 国际标准中，不需要处理输出指令一致性信息的一致性解码器（参见附件 C 对一致性的规定）。在检查比特流一致性和输出定时解码器的一致性时，需要某些 SEI 消息信息。

在附件 D 中，对于 SEI 消息出现的规定，在这些消息通过其它非本建议书 | 国际标准规定的方式传送给解码器时同样适用。当 SEI 消息在比特流中出现时，应该遵守 7.3.2.3，7.4.2.3 节和本附件的规定。当 SEI 消息的内容通过某些方式（不是出现在比特流中）传送时，SEI 消息的内容的表示不需要使用本建议书规定的相同语法。当计算比特数时，只有实际出现在比特流中的比特才会被计算在内。

D.1 SEI 载荷语法

sei_payload(payloadType, payloadSize) {	C	描述符
if(payloadType == 0)		
buffering_period(payloadSize)	5	
else if(payloadType == 1)		
pic_timing(payloadSize)	5	
else if(payloadType == 2)		
pan_scan_rect(payloadSize)	5	
else if(payloadType == 3)		
filler_payload(payloadSize)	5	
else if(payloadType == 4)		
user_data_registered_itu_t_t35(payloadSize)	5	
else if(payloadType == 5)		
user_data_unregistered(payloadSize)	5	
else if(payloadType == 6)		
recovery_point(payloadSize)	5	
else if(payloadType == 7)		
dec_ref_pic_marking_repetition(payloadSize)	5	
else if(payloadType == 8)		
spare_pic(payloadSize)	5	
else if(payloadType == 9)		
scene_info(payloadSize)	5	
else if(payloadType == 10)		
sub_seq_info(payloadSize)	5	
else if(payloadType == 11)		
sub_seq_layer_characteristics(payloadSize)	5	
else if(payloadType == 12)		
sub_seq_characteristics(payloadSize)	5	
else if(payloadType == 13)		
full_frame_freeze(payloadSize)	5	
else if(payloadType == 14)		
full_frame_freeze_release(payloadSize)	5	
else if(payloadType == 15)		
full_frame_snapshot(payloadSize)	5	
else if(payloadType == 16)		
progressive_refinement_segment_start(payloadSize)	5	
else if(payloadType == 17)		
progressive_refinement_segment_end(payloadSize)	5	
else if(payloadType == 18)		
motion_constrained_slice_group_set(payloadSize)	5	
else if(payloadType == 19)		
film_grain_characteristics(payloadSize)	5	
else if(payloadType == 20)		
deblocking_filter_display_preference(payloadSize)	5	
else if(payloadType == 21)		
stereo_video_info(payloadSize)	5	
else		

reserved_sei_message(payloadSize)	5	
if(!byte_aligned( )) {		
bit_equal_to_one /* equal to 1 */	5	f(1)
while(!byte_aligned( ))		
bit_equal_to_zero /* equal to 0 */	5	f(1)
}		
}		

D.1.1 缓冲周期SEI消息语法

buffering_period(payloadSize) {	C	描述符
seq_parameter_set_id	5	ue(v)
if(NalHrdBpPresentFlag) {		
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
initial_cpb_removal_delay[SchedSelIdx **]**	5	u(v)
initial_cpb_removal_delay_offset[SchedSelIdx **]**	5	u(v)
}		
}		
if(VclHrdBpPresentFlag) {		
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
initial_cpb_removal_delay[SchedSelIdx **]**	5	u(v)
initial_cpb_removal_delay_offset[SchedSelIdx **]**	5	u(v)
}		
}		
}		

D.1.2 图像定时SEI消息语法

pic_timing(payloadSize) {	C	描述符
if(CpbDpbDelaysPresentFlag) {		
cpb_removal_delay	**5**	**u(v)**
dpb_output_delay	**5**	**u(v)**
}		
if(pic_struct_present_flag) {		
pic_struct	**5**	**u(4)**
for(i = 0; i < NumClockTS ; i++) {		
clock_timestamp_flag[i]	**5**	**u(1)**
if(clock_timestamp_flag[i]) {		
ct_type	**5**	**u(2)**
nuit_field_based_flag	**5**	**u(1)**
counting_type	5	u(5)
full_timestamp_flag	5	u(1)
discontinuity_flag	5	u(1)
cnt_dropped_flag	5	u(1)
n_frames	5	u(8)

if(full_timestamp_flag) {		
seconds_value /* 0..59 */	5	u(6)
minutes_value /* 0..59 */	5	u(6)
hours_value /* 0..23 */	5	u(5)
} else {		
seconds_flag	5	u(1)
if(seconds_flag) {		
seconds_value /* range 0..59 */	5	u(6)
minutes_flag	5	u(1)
if(minutes_flag) {		
minutes_value /* 0..59 */	5	u(6)
hours_flag	5	u(1)
if(hours_flag)		
hours_value /* 0..23 */	5	u(5)
}		
}		
}		
if(time_offset_length > 0)		
time_offset	5	i(v)
}		
}		
}		
}		

D.1.3 泛扫描矩形 SEI 消息语法

Pan_scan_rect(payloadSize) {	C	描述符
pan_scan_rect_id	5	ue(v)
pan_scan_rect_cancel_flag	5	u(1)
if(!pan_scan_rect_cancel_flag) {		
pan_scan_cnt_minus1	5	ue(v)
for(i = 0; i <= pan_scan_cnt_minus1; i++) {		
pan_scan_rect_left_offset[i **]**	5	se(v)
pan_scan_rect_right_offset[i **]**	5	se(v)
pan_scan_rect_top_offset[i **]**	5	se(v)
pan_scan_rect_bottom_offset[i **]**	5	se(v)
}		
pan_scan_rect_repetition_period	5	ue(v)
}		
}		

D.1.4 填充载荷SEI 消息语法

filler_payload(payloadSize) {	C	描述符
for(k = 0; k < payloadSize; k++)		
ff_byte /* equal to 0xFF */	5	f(8)
}		

D.1.5 ITU-T T.35建议书登记的用户数据的SEI 消息语法

user_data_registered_itu_t_t35(payloadSize) {	C	描述符
itu_t_t35_country_code	5	b(8)
if(itu_t_t35_country_code != 0xFF)		
i = 1		
else {		
Itu_t_t35_country_code_extension_byte	5	b(8)
i = 2		
}		
do {		
itu_t_t35_payload_byte	5	b(8)
i++		
} while(i < payloadSize)		
}		

D.1.6 用户数据未注册SEI语法

User_data_unregistered(payloadSize) {	C	描述符
uuid_iso_iec_11578	5	u(128)
for(i = 16; i < payloadSize; I++)		
user_data_payload_byte	5	b(8)
}		

D.1.7 恢复点SEI 消息语法

recovery_point(payloadSize) {	C	描述符
recovery_frame_cnt	5	ue(v)
exact_match_flag	5	u(1)
broken_link_flag	5	u(1)
changing_slice_group_idc	5	u(2)
}		

D.1.8 解码参考图像标记重复SEI 消息语法

dec_ref_pic_marking_repetition(payloadSize) {	C	描述符
original_idr_flag	5	u(1)
original_frame_num	5	ue(v)
if(!frame_mbs_only_flag) {		
original_field_pic_flag	5	u(1)
if(original_field_pic_flag)		
original_bottom_field_flag	5	u(1)
}		
dec_ref_pic_marking( )	5	
}		

D.1.9 备用图像SEI消息语法

spare_pic(payloadSize) {	C	描述符
target_frame_num	5	ue(v)
spare_field_flag	5	u(1)
if(spare_field_flag)		
target_bottom_field_flag	5	u(1)
num_spare_pics_minus1	5	ue(v)
for(i = 0; i < num_spare_pics_minus1 + 1; i++) {		
delta_spare_frame_num[i **]**	5	ue(v)
if(spare_field_flag)		
spare_bottom_field_flag[i **]**	5	u(1)
spare_area_idc[i **]**	5	ue(v)
if(spare_area_idc[i] = = 1)		
for(j = 0; j < PicSizeInMapUnits; j++)		
spare_unit_flag[i **][** j **]**	5	u(1)
else if(spare_area_idc[i] = = 2) {		
mapUnitCnt = 0		
for(j=0; mapUnitCnt < PicSizeInMapUnits; j++) {		
zero_run_length[i **][** j **]**	5	ue(v)
mapUnitCnt += zero_run_length[i][j] + 1		
}		
}		
}		
}		

D.1.10 场景信息SEI 语法

scene_info(payloadSize) {	C	描述符
scene_info_present_flag	5	u(1)
if(scene_info_present_flag) {		
scene_id	5	ue(v)
scene_transition_type	5	ue(v)
if(scene_transition_type > 3)		
second_scene_id	5	ue(v)
}		
}		

D.1.11 子序列信息SEI消息语法

sub_seq_info(payloadSize) {	C	描述符
sub_seq_layer_num	5	ue(v)
sub_seq_id	5	ue(v)
first_ref_pic_flag	5	u(1)
leading_non_ref_pic_flag	5	u(1)
last_pic_flag	5	u(1)
sub_seq_frame_num_flag	5	u(1)
if(sub_seq_frame_num_flag)		
sub_seq_frame_num	5	ue(v)
}		

D.1.12 子序列层特性SEI 消息语法

sub_seq_layer_characteristics(payloadSize) {	C	描述符
num_sub_seq_layers_minus1	5	ue(v)
for(layer = 0; layer <= num_sub_seq_layers_minus1; layer++) {		
accurate_statistics_flag	5	u(1)
average_bit_rate	5	u(16)
average_frame_rate	5	u(16)
}		
}		

D.1.13 子序列特性SEI消息语法

sub_seq_characteristics(payloadSize) {	C	描述符
sub_seq_layer_num	5	ue(v)
sub_seq_id	5	ue(v)
duration_flag	5	u(1)
if(duration_flag)		
sub_seq_duration	5	u(32)
average_rate_flag	5	u(1)
if(average_rate_flag) {		
accurate_statistics_flag	5	u(1)
average_bit_rate	5	u(16)
average_frame_rate	5	u(16)
}		
num_referenced_subseqs	5	ue(v)
for(n = 0; n < num_referenced_subseqs; n++) {		
ref_sub_seq_layer_num	5	ue(v)
ref_sub_seq_id	5	ue(v)
ref_sub_seq_direction	5	u(1)
}		
}		

D.1.14 全帧冻结SEI 消息语法

full_frame_freeze(payloadSize) {	C	描述符
full_frame_freeze_repetition_period	5	ue(v)
}		

D.1.15 全帧冻结解除SEI 消息语法

full_frame_freeze_release(payloadSize) {	C	描述符
}		

D.1.16 全帧快照SEI消息语法

full_frame_snapshot(payloadSize) {	C	描述符
snapshot_id	5	ue(v)
}		

D.1.17 逐步细化段开始SEI消息语法

progressive_refinement_segment_start(payloadSize) {	C	描述符
progressive_refinement_id	5	ue(v)
num_refinement_steps_minus1	5	ue(v)
}		

D.1.18 逐步细化段结束SEI消息语法

progressive_refinement_segment_end(payloadSize) {	C	描述符
progressive_refinement_id	5	ue(v)
}		

D.1.19 运动受限条带组集SEI消息语法

motion_constrained_slice_group_set(payloadSize) {	C	描述符
num_slice_groups_in_set_minus1	5	ue(v)
for(i = 0; i <= num_slice_groups_in_set_minus1; i++)		
slice_group_id[i **]**	5	u(v)
exact_sample_value_match_flag	5	u(1)
pan_scan_rect_flag	5	u(1)
if(pan_scan_rect_flag)		
pan_scan_rect_id	5	ue(v)
}		

D.1.20 胶片颗粒特性SEI消息语法

film_grain_characteristics(payloadSize) {	C	描述符
film_grain_characteristics_cancel_flag	5	u(1)
if(!film_grain_**characteristics**_cancel_flag) {		
model_id	5	u(2)
separate_colour_description_present_flag	5	u(1)
if(separate_colour_description_present_flag) {		
film_grain_bit_depth_luma_minus8	5	u(3)
film_grain_bit_depth_chroma_minus8	5	u(3)
film_grain_full_range_flag	5	u(1)
film_grain_colour_primaries	5	u(8)
film_grain_transfer_characteristics	5	u(8)
film_grain_matrix_coefficients	5	u(8)
}		
blending_mode_id	5	u(2)
log2_scale_factor	5	u(4)
for(c = 0; c < 3; c++)		
comp_model_present_flag[c]	5	u(1)
for(c = 0; c < 3; c++)		
if(comp_model_present_flag[c]) {		
num_intensity_intervals_minus1[c **]**	5	u(8)
num_model_values_minus1[c **]**	5	u(3)
for(i = 0; i <= num_intensity_intervals_minus1[c]; i++) {		
intensity_interval_lower_bound[c **][** i **]**	5	u(8)
intensity_interval_upper_bound[c **][** i **]**	5	u(8)
for(j = 0; j <= num_model_values_minus1[c]; j++)		
comp_model_value[c **][** i **][** j **]**	5	se(v)
}		
}		
film_grain_characteristics_repetition_period	5	ue(v)
}		
}		

D.1.21 去块效应滤波器显示优选项SEI消息语法

deblocking_filter_display_preference(payloadSize) {	C	描述符
deblocking_display_preference_cancel_flag	5	u(1)
if(!deblocking_display_preference_cancel_flag) {		
display_prior_to_deblocking_preferred_flag	5	u(1)
dec_frame_buffering_constraint_flag	5	u(1)
deblocking_display_preference_repetition_period	5	ue(v)
}		
}		

D.1.22 立体视频信息SEI 消息语法

stereo_video_info(payloadSize) {	C	描述符
field_views_flag	5	u(1)
if(field_views_flag)		
top_field_is_left_view_flag	5	u(1)
else {		
current_frame_is_left_view_flag	5	u(1)
next_frame_is_second_view_flag	5	u(1)
}		
left_view_self_contained_flag	5	u(1)
right_view_self_contained_flag	5	u(1)
}		

D.1.23 保留SEI消息语法

reserved_sei_message(payloadSize) {	C	描述符
for(i = 0; i < payloadSize; i++)		
reserved_sei_message_payload_byte	5	b(8)
}		

D.2 SEI载荷语义

D.2.1 缓冲周期SEI消息的语义

当 NalHrdBpPresentFlag 或 VclHrdBpPresentFlag 等于 1 时，一个缓冲周期 SEI 消息可以关联到比特流中的任一访问单元。同时，每个 IDR 访问单元应该关联一个缓冲周期 SEI 消息，每个关联了恢复指针 SEI 消息的访问单元也应该关联一个缓冲周期 SEI 消息。

注 — 在某些应用中，可能要求缓冲周期SEI消息频繁出现。

在解码序列中，两个缓冲周期 SEI 消息之间的一组访问单元称为一个缓冲周期。

seq_parameter_set_id 表示序列参数集标号，包含序列 HRD 属性。seq_parameter_set_id 的值应与一个主编码图像引用的图像参数组中的 seq_parameter_set_id 的值相等，该主编码图像是与本缓冲周期 SEI 消息相关联的。seq_parameter_set_id 的取值范围是大于等于 0，小于等于 31。

initial_cpb_removal_delay[SchedSelIdx] 表示在 HRD 初始化后的第一个缓冲周期，第 SchedSelIdx 个 CPB 的时间延迟，这个延迟是从与该缓冲周期 SEI 消息相关联的访问单元中的编码数据的第一个比特到达，到其数据从 CPB 中删除的时间间隔。这个语法元素的比特长度由 initial_cpb_removal_delay_length_minus1 + 1 给出。initial_cpb_removal_delay[SchedSelIdx]单位是 90kHz 的时钟（即 1/90000 秒），其取值不能为 0，也不能超过 90000 * (CpbSize[SchedSelIdx] ÷ BitRate[SchedSelIdx])，即以 90kHz 的时钟为单位的 CPB 时间容限。

initial_cpb_removal_delay_offset[SchedSelIdx]，与 initial_cpb_removal_delay[SchedSelIdx]组合用于第 SchedSelIdx 个 CPB，表示对 CPB 的编码访问单元的初始发送时间。initial_cpb_removal_delay_offset[SchedSelIdx]也以 90kHz 的时钟为单位。它的语法元素是一个定长码，其比特长度由 initial_cpb_removal_delay_length_minus1 + 1 决定。这个语法元素不被解码器使用，而只是附件 C 中规定的发送调度程序的需要。

对全部视频编码序列，不论 SchedSelIdx 的取值，initial_cpb_removal_delay[SchedSelIdx] 与 initial_cpb_removal_delay_offset[SchedSelIdx]的和是一个常数。

D.2.2 图像定时SEI消息的语义

在比特流中，图像定时 SEI 消息的出现应遵循下述原则：

— 如果CpbDpbDelaysPresentFlag等于 1 ，或pic_struct_present_flag等于1，视频编码序列的每个访问单元中都要有图像定时SEI消息。

— 否则（CpbDpbDelaysPresentFlag等于0且pic_struct_present_flag等于0），视频编码序列的任一访问单元中都不应出现图像定时SEI消息。

cpb_removal_delay 表示从 CPB 中删除与最近的一个缓冲周期 SEI 消息相关的访问单元以后，到与图像定时 SEI 消息相关的访问单元被删除为止，等待的时钟跳动数（见 E.2.1）。该值还用来计算对 HSS 来说进入 CPB 的访问单元数据最早可能的到达时间，见附件 C 的规定。它的语法元素是一个定长码，其长度由 cpb_removal_delay_length_minus1 + 1.决定。cpb_removal_delay 是计数器 $2^{(cpb_removal_delay_length_minus1 + 1)}$的余数。

比特流中第一幅图像的 cpb_removal_delay 的值应为 0。

dpb_output_delay 用于计算图像的 DPB 输出时间。它表示，从访问单元自 CPB 删除至解码图像自 DPB 输出（见 C.2）需要等待的时钟记录数。

注1 — 一幅图像在输出时并不从DPB中删除，而是被标记为“用于短期参考”或“用于长期参考”。

注2 — 对一幅解码图像只有一个DPB输出延时。

语法元素 dpb_output_delay 的比特长度是 dpb_output_delay_length_minus1 + 1，当 max_dec_frame_buffering 等于 0 时，dpb_output_delay 应为 0。

输出时间由 dpb_output_delay 推导出。C.2 节中规定的输出定时一致解码器输出的任一幅图像的输出时间，应超过解码顺序中后续任何一个视频编码序列的所有图像的输出时间。

按解码顺序，一个补充非参考场对中，从第二场的 dpb_output_delay 推导出的输出时间应该超过该补充非参考场对的第一场的 dpb_output_delay 推导出的输出时间。

由本语法元素的值确定的图像输出顺序，应当与 C.4.1 至 C.4.5 条规定的 PicOrderCnt( )的值确定的图像输出顺序相同。一种特例情况是，一个补充参考场对的两场有相同的 PicOrderCnt( )值，它们的输出时间不同。

对那些不是由 C.4.5 规定的释放（bumping）过程输出的图像，因为在解码顺序上，它们在一个 no_output_of_prior_pics_flag 等于 1 或推断为 1 的 IDR 图像之前，所以在同一视频解码序列中任何一幅 memory_management_control_operation 等于 5 的图像之后的所有图像，由 dpb_output_delay 导出的输出时间应当随着 PicOrderCnt( )值的增加而增加。

pic_struct 表示一幅图像应显示为一帧还是一场或更多场，如表 D-1。双倍帧（pic_struct 等于 7）表示该帧应连续显示两次，而三倍帧（pic_struct 等于 8）表示该帧应连续显示三次。

注3 — 双倍帧使显示方便，例如，一个25p的视频信号显示在50p的显示器上，以及29.97p的视频信号显示在59.94p的显示器上。每隔一帧组合使用双倍帧和三倍帧使23.98p的视频信号方便地显示在59.94p的显示器上。

表 D-1－pic_struct的解释

值	图像显示含义	限制条件	NumClockTS
0	帧	field_pic_flag 应为 0	1
1	顶场	field_pic_flag 应为 1, bottom_field_flag 应为 0	1
2	底场	field_pic_flag 应为 1, bottom_field_flag 应为 1	1
3	顶场，底场，按顺序	field_pic_flag 应为 0	2
4	底场，顶场，按顺序	field_pic_flag 应为 0	2
5	顶场，底场，重复顶场，按顺序	field_pic_flag 应为 0	3
6	底场，顶场，重复底场，按顺序	field_pic_flag 应为 0	3
7	双倍帧	field_pic_flag 应为 0 fixed_frame_rate_flag 应为 1	2
8	三倍帧	field_pic_flag 应为 0 fixed_frame_rate_flag 应为 1	3
9..15	保留		

NumClockTS 是由 pic_struct 决定的，见表 D-1。对一幅图像，最多可以有 NumClockTS 组时戳信息，每组由 clock_timestamp_flag[i]标识。时戳信息组根据 pic_struct 的内容用于关联图像的对应场或帧。

时戳信息语法元素的内容说明源时间，拍摄时间或理想的播放时间。这个时间由下式计算：

$$clockTimestamp = ((hH * 60 + mM) * 60 + sS) * time\_scale + nFrames * (num\_units\_in\_tick * (1 + nuit\_field\_based\_flag)) + tOffset, \quad (D\text{-}1)$$

以时钟记录数为单位，时钟频率等于 time_scale Hz，对于一些未说明时间的点 clockTimestamp 等于 0。输出顺序和 DPB 输出定时不受 clockTimestamp 值的影响。当 pic_struct 等于 0 的两帧或更多帧在输出顺序上连续且有相等的 clockTimestamp 值时，其含义是这些帧表现的是同样的内容，其中最后输出的一帧是首选的图像。

注4 — clockTimestamp定时指示有助于在刷新频率不能与DPB输出时间很好匹配的设备上显示。

clock_timestamp_flag[i **]** 值为 1 时表示随后马上有很多时戳语法元素出现。值为 0 时表示不存在相关联的时戳语法元素。当 NumClockTS 大于 1 且存在一个以上的 i 值使 clock_timestamp_flag[i]等于 1 时，clockTimestamp 的值不能随着 i 值的增大而减小。

ct_type 表示源资料的扫描类型，见表 D-2。

一个编码帧的两场可以有不同的 ct_type 值。

当在输出顺序上连续的奇偶两场的 clockTimestamp 相等时，若两场的 ct_type 值等于 0（逐行）或等于 2（未知），表示这两场来自同一原始逐行帧。当输出顺序连续的两场中任一场的 ct_type 值为 1（隔行）时，则两场应有不同的 clockTimestamp 值。

表 D-2－ct_type与原图像扫描的对应关系

值	原图像扫描方式
0	逐行
1	隔行
2	未知
3	保留

nuit_field_based_flag 用于根据方程式 D-1 计算 clockTimestamp。

counting_type 表示 n_frames 的值丢弃的方法，见表 D-3。

表 D-3 － counting_type值的定义

值	解释
0	不丢弃 n_frames 计数器的值，且不用 time_offset
1	不丢弃 n_frames 计数器的值
2	丢弃 n_frames 计数器中单独的 0 值
3	丢弃 n_frames 计数器中单独的 MaxFPS−1 值
4	当 seconds_value 等于 0 且 minutes_value 不是 10 的整数倍时，丢弃最小的（值为 0 和 1）两个 n_frames 计数器
5	丢弃 n_frames 计数器中单独的未指明值
6	丢弃未指明个 n_frames 计数器中未指明的值
7..31	保留

full_timestamp_flag 等于 1 时表示 n_frames 语法元素后面有 seconds_value，minutes_value，和 hours_value。等于 0 时表示 n_frames 语法元素后面只有 seconds_flag。

discontinuity_flag 等于 0 时表示 clockTimestamp 的当前值与按照输出顺序其前一个时戳计算得到的 clockTimestamp 的值的差异，可以解释为源时间与拍摄关联的帧或场时间之间的时差。discontinuity_flag 等于 1 时表示不能做上述解释。当 discontinuity_flag 等于 0 时，clockTimestamp 的值应该大于等于按 DPB 输出顺序其前图像的所有存在的 clockTimestamp 值。

cnt_dropped_flag 表示依据 counting_type 说明的计算方式丢弃 n_frames 中的一个或几个值。

n_frames 表示用于计算 clockTimestamp 的 nFrames 的值。n_frames 应小于：

$$MaxFPS = Ceil(time\_scale \div num\_units\_in\_tick) \quad (D\text{-}2)$$

注 5 — n_frames 是一个基于帧的计数器。对特定场的定时指示，时间偏移应用于表明每个场的具体 clockTimestamp。

当 counting_type 等于 2 且 cnt_dropped_flag 等于 1 时，n_frames 应该等于 1，且按照输出顺序其前面的图像的 n_frames 值不能为 0，除非 discontinuity_flag 等于 1。

注6 — 当counting_type等于2时，如果采用固定的非整数帧率（例如每秒12.5帧，time_scale等于25，并且num_units等于2和nuit_field_based_flag等于0），方程式D-1中的tOffset需要的值越来越大。通过计数时偶尔地丢弃n_frames值等于0来避免这种情况，例如n_frames计数从0到12后seconds_value加一，接着n_frames计数从1到12后seconds_value加一，n_frames再计数从0到12。

当 counting_type 等于 3 且 cnt_dropped_flag 等于 1 时，n_frames 应该等于 0，且按照输出顺序其前面的图像的 n_frames 值不能等于 MaxFPS − 1，除非 discontinuity_flag 等于 1。

注7 — 当counting_type等于3时，如果采用固定的非整数帧率（例如每秒12.5帧，time_scale等于25，并且num_units等于2和nuit_field_based_flag等于0），方程式D-1中的tOffset需要的值越来越大。通过计数时偶尔地丢弃n_frames值等于MaxFPS来避免这种情况，例如n_frames计数从0到12后seconds_value加一，接着n_frames计数从0到11后seconds_value加一，n_frames再计数从0到12。

当 counting_type 等于 4 且 cnt_dropped_flag 等于 1 时，n_frames 应该等于 2，且 sS 的规定值应能为 0，mM 的规定值不应为 10 的整数倍，按照输出顺序其前面的图像的 n_frames 值不能为 0 或 1 除非 discontinuity_flag 等于 1。

注8 — 当counting_type等于4时，如果采用固定的非整数帧率（例如每秒30000÷1001帧，time_scale等于60000，并且num_units等于1001和nuit_field_based_flag等于1），方程式D-1中的tOffset需要的值越来越大。通过计数时偶尔地丢弃n_frames值等于MaxFPS来避免这种情况，例如n_frames从0计数到29后seconds_value加一，n_frames继续从0计数到29，直到seconds_value等于0且minutes_value不是10的整数倍时，n_frames再从2计数到29，然后seconds_value加一，n_frames继续从0计数到29。这是业内公认的计数方法，常被称作“NTSC丢帧”计数法。

当 counting_type 等于 5 或 6 且 cnt_dropped_flag 等于 1 时，n_frames 应该等于按照输出顺序其前面的图像的 n_frames 值对 MaxFPS 的模再加 1，除非 discontinuity_flag 等于 1。

注9 — 当counting_type等于5或6时，如果采用固定的非整数帧率，方程式D-1中的tOffset需要的值越来越大。通过计数时偶尔地丢弃某些n_frames值可以避免这种情况。counting_type等于5或6时被丢弃的具体n_frames值未详细规定。

seconds_flag 等于 1 时表示如果 full_timestamp_flag 等于 0，seconds_value 和 minutes_flag 存在。等于 0 时表示 seconds_value 和 minutes_flag 不存在。

seconds_value 表示用于计算 clockTimestamp 的 sS 值。seconds_value 的值应在 0 到 59 的范围内。当 seconds_value 不存在时，用解码顺序中前一个 seconds_value 作为 sS 的值计算 clockTimestamp。

minutes_flag 等于 1 时表示如果 full_timestamp_flag 等于 0 且 seconds_flag 等于 1，minutes_value 和 hours_flag 存在。等于 0 时表示 minutes_value 和 hours_flag 不存在。

minutes_value 表示用于计算 clockTimestamp 的 mM 值。minutes_value 的值应在 0 到 59 的范围内。当 minutes_value 不存在时，用解码顺序中前一个 minutes_value 作为 mM 的值计算 clockTimestamp。

hours_flag 等于 1 时表示如果 full_timestamp_flag 等于 0 且 seconds_flag 和 minutes_flag 等于 1，hours_value 存在。

hours_value 表示用于计算 clockTimestamp 的 hH 值。hours_value 的值应在 0 到 23 的范围内。当 hours_value 不存在时，用解码顺序中前一个 hours_value 作为 hH 的值计算 clockTimestamp。

time_offset 表示用于计算 clockTimestamp 的 tOffset 值。表示 time_offset 的比特数应等于 time_offset_length。当 time_offset 不存在时，用 0 值作为 tOffset 计算 clockTimestamp。

D.2.3 泛扫描矩形SEI消息的语义

泛扫描矩形 SEI 消息的语法元素规定一个矩形相对于序列参数集的裁剪矩形的坐标。这个矩形的每个坐标由相对于亮度采样格栅的 1/16 采样间隔为单位表示。

pan_scan_rect_id 包含一个识别号码，用于表示泛扫描矩形的用途（例如表示该矩形是在特定显示设备上放映的区域，或是含有场景中的特定角色的区域）。pan_scan_rect_id 的值应在 0 到 $2^{32} - 1$ 的范围内。

pan_scan_rect_id 的值从 0 到 255 和从 512 到 $2^{31}-1$ 可以用于应用决定的场景。pan_scan_rect_id 的值从 256 到 511 和从 2^{31} 到 $2^{32}-1$ 为未来的应用被 ITU-T | ISO/IEC 保留。解码器遇到 pan_scan_rect_id 的值在从 256 到 511 或从 2^{31} 到 $2^{32}-1$ 的范围内时，应忽略它（丛比特流中删除并丢弃）。

pan_scan_rect_cancel_flag 等于 1 时表示本 SEI 消息取消输出顺序中所有在前的泛扫描矩形 SEI 消息的存留。pan_scan_rect_cancel_flag 等于 0 时表示随后是泛扫描矩形信息。

pan_scan_cnt_minus1 表示 SEI 消息中出现的泛扫描矩形的数量。pan_scan_cnt_minus1 的值应在 0 至 2 之间。pan_scan_cnt_minus1 等于 0 表示一个泛扫描矩形用于解码图像的所有场。pan_scan_cnt_minus1 等于 1 表示有两个泛扫描矩形，第一个用于输出顺序上图像的第一场，第二个用于输出顺序上图像的第二场。pan_scan_cnt_minus1 等于 2 表示有三个泛扫描矩形，第一个用于输出顺序上图像的第一场，第二个用于输出顺序上图像的第二场，第三个重复第一个用于输出顺序上图像的第三场。

pan_scan_rect_left_offset[i], **pan_scan_rect_right_offset[i]**, **pan_scan_rect_top_offset[i]**, 和 **pan_scan_rect_bottom_offset[i]** 表示泛扫描矩形位置的有符号整数值，单位是相对于亮度采样格栅的 1/16 采样间隔。这四个语法元素的每个的值应在-2^{31} 到 $2^{31}-1$ 的范围内。

泛扫描矩形区域由取值从 16*CropUnitX * frame_crop_left_offset + pan_scan_rect_left_offset[i] 到 16 * (16 * PicWidthInMbs − CropUnitX * *frame_crop_right_offset) + pan_scan_rect_right_offset[i] − 1 的帧水平坐标，和取值从 16 *CropUnitY * frame_crop_top_offset + pan_scan_rect_top_offset[i] 到 16 * (16 * PicHeightInMbs − CropUnitY * frame_crop_bottom_offset) + pan_scan_rect_bottom_offset[i] − 1 垂直坐标表示。16 * CropUnitX * frame_crop_left_offset + pan_scan_rect_left_offset[i] 的值应小于等于 16 * (16 * PicWidthInMbs − CropUnitX * frame_crop_right_offset) + pan_scan_rect_right_offset[i] − 1 ； 而 16 CropUnitY * frame_crop_top_offset + pan_scan_rect_top_offset[i] 的值应小于等于 16 * (16 * PicHeightInMbs − CropUnitY * frame_crop_bottom_offset) + pan_scan_rect_bottom_offset[i] − 1。

当泛扫描矩形区域包括裁剪矩形以外的样值时，裁剪矩形以外的区域可以用合成的内容（例如黑色或灰色的视频内容）填充。

pan_scan_rect_repetition_period 表示泛扫描矩形 SEI 消息的持续性，还可以指定一个图像顺序计数间隔，在此间隔后，比特流中出现另一个具有相同 pan_scan_rect_id 值的泛扫描矩形 SEI 消息，或者到达视频编码序列的末尾。pan_scan_rect_repetition_period 的值应在 0 到 16384 的范围内。如果 pan_scan_cnt_minus1 的值大于 0，pan_scan_rect_repetition_period 的值不能大于 1。

pan_scan_rect_repetition_period 等于 0 表示泛扫描矩形的信息只用于当前的解码图像。

pan_scan_rect_repetition_period 等于 1 表示泛扫描矩形的信息一直存在于输出直到满足下面任一个条件。

— 一个新的视频编码序列开始。

— 输出的一个访问单元中的图像包含pan_scan_rect_id值相同的泛扫描矩形SEI消息，PicOrderCnt( ) 的值大于PicOrderCnt(CurrPic)。

pan_scan_rect_repetition_period 等于 0 或等于 1 预示着存在或不存在 pan_scan_rect_id 值相同的泛扫描矩形 SEI 消息。

pan_scan_rect_repetition_period 大于 1 表示泛扫描矩形的信息一直存在直到满足下面任一个条件。

— 一个新的视频编码序列开始。

— 输出的一个访问单元中的图像包含pan_scan_rect_id值相同的泛扫描矩形SEI消息，图像的 PicOrderCnt( ) 的值大于 PicOrderCnt(CurrPic)，小于等于 PicOrderCnt(CurrPic) + pan_scan_rect_repetition_period。

pan_scan_rect_repetition_period 大于 1 预示，对输出的访问单元中的一幅图像，应有另一个 pan_scan_rect_id 值相同的泛扫描矩形 SEI 消息，图像的 PicOrderCnt( )的值大于 PicOrderCnt(CurrPic)，小于等于 PicOrderCnt(CurrPic)

+pan_scan_rect_repetition_period，除非比特流结束或者新的视频编码序列开始而没有输出这幅图像。

D.2.4 填充载荷SEI消息语义

本消息包含一系列值为 0xFF 的填充字节，可以丢弃。

ff_byte 应是值为 0xFF 的字节。

D.2.5 ITU-T T.35建议书登记的用户数据的SEI消息语义

本消息包含 ITU-T T.35 建议书登记的用户数据，其内容不由本建议书规定。

itu_t_t35_country_code 应为 1 个字节，值是 ITU-T T.35 建议书附件 A 规定的国家码。

itu_t_t35_country_code_extension_byte 应为 1 个字节，值是 ITU-T T.35 建议书附件 B 规定的国家码。

itu_t_t35_payload_byte 应为 1 个字节，包含 ITU-T T.35 建议书登记的用户数据。

ITU-T T.35 建议书的终端提供者编码和终端提供者导向编码应包含在 itu_t_t35_payload_byte 的最初一个或几个字节中，由发布终端提供者编码的管理员规定其格式。itu_t_t35_payload_byte 数据的剩余字节应是具有语法和语义的数据，其语法语义由 ITU-T T.35 建议书的国家码和终端提供者编码确定的实体规定。

D.2.6 未登记的用户数据的SEI消息语义

本消息包含未登记的用户数据，由 UUID 标识，其内容不由本建议书规定。

uuid_iso_iec_11578 应有一个由 UUID 指定的值，依照 ISO/IEC 11578:1996 附件 A 的规程。

user_data_payload_byte 应为一个字节的数据，其语法和语义由 UUID 生成器指明。

D.2.7 恢复点SEI消息语义

恢复点 SEI 消息帮助解码器确定，在解码器开始随机访问或解码器遇到序列中断的链接以后，解码过程生成能够合格显示的图像的时间。当解码过程从一个解码顺序中与恢复点 SEI 消息关联的访问单元开始时，所有此 SEI 消息指明的输出顺序中恢复点以后的解码图像是内容正确的或大致正确的。由恢复点 SEI 消息关联的图像之前的随机访问单元生成的解码图像在内容上不一定是正确的，直到指定的恢复点。从与恢复点 SEI 消息关联的访问单元开始的解码过程操作可以包含对解码图像缓冲区中不存在的图像的引用。

另外，通过使用 broken_link_flag，恢复点 SEI 消息可以为解码器指示比特流中一些可能在显示时导致严重视觉假象的图像的位置，甚至当解码过程在解码序列中前一个 IDR 访问单元的位置开始时。

注1 — broken_link_flag可以为解码器指示一个位置点，该点以后对一些图像的解码过程可能导致对解码时虽然能够得到，但是不是比特流原始编码时引用的图像的引用（例如，由于在比特流生成过程中执行了拼接操作）。

恢复点由以 frame_num 的增量为单位的计数器来表示，其值是 SEI 消息所处位置的当前访问单元的 frame_num 之后的增量。

注2 — 当比特流中有HRD信息时，关联了恢复点SEI消息的访问单元应关联一个缓冲周期SEI消息，从而在随机访问后建立HRD缓冲模型的初始值。

recovery_frame_cnt 指定输出图像在输出顺序中的恢复点。从参考图像的输出位置以后，所有输出顺序中的解码图像的内容是正确的或大致正确的，参考图像的 frame_num 等于当前访问单元的 VCL NAL 单元的 frame_num 加上 recovery_frame_cnt 再对 MaxFrameNum 取模运算。recovery_frame_cnt 的值在 0 到 MaxFrameNum − 1 的范围内。

exact_match_flag 表示在与恢复点 SEI 消息关联的访问单元处开始的解码过程输出的特定恢复点之后的解码图像，是否应该是一个与 NAL 单元流中的前一个 IDR 访问单元位置处开始的解码过程生成的图像精确匹配的图

像。值为 0 表示不一定精确匹配，为 1 表示精确匹配。

当解码从恢复点 SEI 消息开始时，所有对不可获得的参考图像的引用应当推断为对只包含用内部宏块预测方式编码的宏块，样点值为 Y 等于 128，Cb 和 Cr 等于 128（中度灰）的图像的引用，目的是确定与 exact_match_flag 的取值的一致性。

注3 — 当执行随机访问时，解码器应推断对不可获得的参考图像的引用是对只包含内部宏块，样点值为Y等于128，Cb和Cr等于128（中度灰）的图像的引用，而不考虑exact_match_flag的取值。

当 exact_match_flag 等于 0 时，恢复点处近似图像的质量由编码过程决定，不由本建议书 | 国际标准规定。

broken_link_flag 表示在恢复点 SEI 消息处 NAL 单元流的链接是否出现中断。它的语义如下。

— 如果broken_link_flag等于1，在前一个IDR访问单元位置处开始的解码过程生成的图像可能包含不希望的视觉假象，以致于在输出顺序中关联恢复点SEI消息的访问单元之后的解码图像不可显示，直到指定的输出顺序中的恢复点。

— 否则（broken_link_flag等于0），没有预示会出现潜在的视觉假象。

不论 broken_link_flag 的取值，在输出顺序中指定恢复点以后的图像在内容上是正确的或大致正确的。

注4 — 当一个子序列信息SEI消息与一个broken_link_flag等于1的恢复点SEI消息联合出现，且sub_seq_layer_num等于0时，sub_seq_id应与最近的一个在恢复点SEI消息的位置之前解码的sub_seq_layer_num等于0的sub_seq_id不同。当broken_link_flag等于0时，子序列层0的sub_seq_id应保持不变。

changing_slice_group_idc 等于 0 表示当主编码图像的所有宏块在变换条带组周期内，也就是从恢复点 SEI 消息关联的访问单元到指定恢复点期间被解码时，在输出顺序中恢复点以后解码图像内容正确或大致正确。当变换条带组周期内任一主编码图像的 num_slice_groups_minus1 等于 0 时，changing_slice_group_idc 应等于 0。

当 changing_slice_group_idc 等于 1 或 2 时，num_slice_groups_minus1 应等于 1，且宏块对条带映射类型 3、4 或 5 应被用于变换条带组周期内的每个主编码图像。

changing_slice_group_idc 等于 1 表示在变换条带组周期内，没有条带组 0 覆盖的解码宏块以外的采样值被用于条带组 0 内任一宏块的内部预测。同时，changing_slice_group_idc 等于 1 还表示当变换条带组周期内条带组 0 的所有宏块被解码时，输出顺序中指定恢复点以后的所有解码图像的内容是正确或大致正确的，而不论变换条带组周期内条带组 1 的宏块是否被解码。

changing_slice_group_idc 等于 2 表示在变换条带组周期内，没有条带组 1 覆盖的解码宏块以外的采样值被用于条带组 1 内任一宏块的内部预测。同时，changing_slice_group_idc 等于 2 还表示当变换条带组周期内条带组 1 的所有宏块被解码时，输出顺序中指定恢复点以后的所有解码图像的内容是正确或大致正确的，而不论变换条带组周期内条带组 0 的宏块是否被解码。

changing_slice_group_idc 应在 0 到 2 的范围内。

D.2.8 解码参考图像标记重复SEI消息的语义

解码参考图像标记重复 SEI 消息用于重复解码参考图像标记的语法结构，此语法结构位于解码顺序中较早图像的条带头中。

original_idr_flag 当解码参考图像标记语法结构最初在一幅 IDR 图像中出现时应等于 1。当解码参考图像标记语法结构最初在一幅非 IDR 图像中出现时，original_idr_flag 应等于 0。

original_frame_num 应等于重复的解码参考图像标记语法结构最初出现所属的图像的 frame_num。由 original_frame_num 指示的图像是具有指定 frame_num 值的前一幅编码图像。memory_management_control_operation 等于 5 的图像的 original_frame_num 的值通常应为 0 。

original_field_pic_flag 应等于重复的解码参考图像标记语法结构最初出现所属的图像的 field_pic_flag。

original_bottom_field_flag 应等于重复的解码参考图像标记语法结构最初出现所属的图像的 bottom_field_flag。

dec_ref_pic_marking( ) 应包含 frame_num 为 original_frame_num 的图像的解码参考图像标记语法结构的拷贝。用于规范重复的 dec_ref_pic_marking( ) 语法结构的 nal_unit_type 应是 frame_num 为 original_frame_num 的图像的 slice header(s)的 nal_unit_type（也就是，7.3.3.3 条中的 nal_unit_type，在 original_idr_flag 等于 1 时应等于 5，在 original_idr_flag 等于 0 时应不等于 5）。

D.2.9 备用图像SEI消息的语义

本 SEI 消息表示被称作备用条带组映射单元的条带组映射单元，在一个或多个解码参考图像中，与被称作目标图像的指定解码图像中的相同位置条带组映射单元类似。一个备用条带组映射单元可以用于在目标图像中代替一个解码错误的同位置的条带组映射单元。一幅包含备用条带组映射单元的图像称作备用图像。

对所有备用图像 SEI 消息标定的备用图像，frame_mbs_only_flag 值应等于同一个 SEI 消息的目标图像的 frame_mbs_only_flag 值。SEI 消息中的备用图像应该遵守如下限制：

— 如果目标图像为解码的场，同一SEI消息中的所有备用图像均为解码场。

— 否则（目标图像为一解码帧），同一SEI消息中的所有图像均为解码帧。

对所有备用图像 SEI 消息标定的备用图像，其 pic_width_in_mbs_minus1 和 pic_height_in_map_units_minus1 的值，应对应地等于同一个 SEI 消息的目标图像的 pic_width_in_mbs_minus1 和 pic_height_in_map_units_minus1 值。与本消息关联的图像（参见 7.4.1.2.3 节的规定）在解码顺序中应在目标图像之后出现。

target_frame_num 表示目标图像的 frame_num。

spare_field_flag 等于 0 表示目标图像和备用图像是解码帧。spare_field_flag 等于 1 表示目标图像和备用图像是解码场。

target_bottom_field_flag 等于 0 表示目标图像是顶场。target_bottom_field_flag 表示目标图像是底场。

目标图像是一幅解码参考图像，其对应的主编码图像在解码顺序中位于当前图像的前面。而主编码图像的 frame_num，field_pic_flag (存在时) 和 bottom_field_flag (存在时)的值对应等于 target_frame_num, spare_field_flag 和 target_bottom_field_flag。

num_spare_pics_minus1 表示指定目标图像的备用图像数量。备用图像数等于 num_spare_pics_minus1 + 1。num_spare_pics_minus1 的取值范围是 0 到 15。

delta_spare_frame_num[i] 用于标示包含第 i 套备用条带组映射单元的备用图像，下文称作第 I 幅备用图像，详细说明如下。delta_spare_frame_num[i]的值应在 0 到 MaxFrameNum − 1 − !spare_field_flag 的范围内。

第 i 幅备用图像的 frame_num，即 spareFrameNum[i]，通过下式得到：

```
candidateSpareFrameNum = target_frame_num − !spare_field_flag
for ( i = 0; i <= num_spare_pics_minus1; i++ ) {
  if( candidateSpareFrameNum < 0 )
    candidateSpareFrameNum = MaxFrameNum − 1
  spareFrameNum[ i ] = candidateSpareFrameNum − delta_spare_frame_num[ i ]          (D-3)
  if( spareFrameNum[ i ] < 0 )
    spareFrameNum[ i ] = MaxFrameNum + spareFrameNum[ i ]
```

candidateSpareFrameNum = spareFrameNum[i] − !spare_field_flag

}

spare_bottom_field_flag[i] 等于 0 表示第 i 幅备用图像是顶场。spare_bottom_field_flag[i]等于 1 表示第 i 幅备用图像是底场。

第 0 幅备用图像是一幅解码参考图像，其对应的主编码图像在解码顺序中位于目标图像的前面，而主编码图像的 frame_num，field_pic_flag (存在时) 和 bottom_field_flag (存在时)的值对应等于 spareFrameNum[0]，spare_field_flag 和 spare_bottom_field_flag[0]。第 i 幅备用图像是一幅解码参考图像，其对应的主编码图像在解码顺序中位于第 i-1 幅备用图像的前面，而主编码图像的 frame_num，field_pic_flag (存在时) 和 bottom_field_flag (存在时)的值对应等于 spareFrameNum[i]，spare_field_flag 和 spare_bottom_field_flag[i]。

spare_area_idc[i] 表示用来标识在第 i 幅备用图像中备用条带组映射单元的方法。spare_area_idc[i]应在 0 到 2 的范围内取值。spare_area_idc[i]等于 0 表示第 i 幅备用图像中的所有条带组映射单元是备用单元。spare_area_idc[i]等于 1 表示用语法元素 spare_unit_flag[i][j]的值标识备用条带组映射单元。spare_area_idc[i]等于 2 表示用语法元素 zero_run_length[i][j]导出 spareUnitFlagInBoxOutOrder[i][j]，具体参见下述。

spare_unit_flag[i][j] 等于 0 表示第 i 幅备用图像的光栅扫描顺序中的第 j 个条带组映射单元是备用单元。spare_unit_flag[i][j]等于 1 表示第 i 幅备用图像的光栅扫描顺序中的第 j 个条带组映射单元不是备用单元。

zero_run_length[i][j] 在 spare_area_idc[i]等于 2 时用于导出 spareUnitFlagInBoxOutOrder[i][j]的值。在这种情况下，对每幅备用图像，由 spareUnitFlagInBoxOutOrder[i][j]确定的备用条带组映射单元以逆时针 box-out 的顺序出现。spareUnitFlagInBoxOutOrder[i][j]等于 0 表示第 i 幅备用图像的逆时针 box-out 顺序中的第 j 个条带组映射单元是备用单元。spareUnitFlagInBoxOutOrder[i][j]等于 1 表示第 i 幅备用图像的逆时针 box-out 顺序中的第 j 个条带组映射单元不是备用单元。

当 spare_area_idc[0]等于 2 时，spareUnitFlagInBoxOutOrder[0][j]由下式计算：

```
for( j = 0, loop = 0; j < PicSizeInMapUnits; loop++ ) {
  for( k = 0; k < zero_run_length[ 0 ][ loop ]; k++ )
    spareUnitFlagInBoxOutOrder[ 0 ][ j++ ] = 0                    (D-4)
  spareUnitFlagInBoxOutOrder[ 0 ][ j++ ] = 1
}
```

当 spare_area_idc[i]等于 2 且 i 值大于 0 时，spareUnitFlagInBoxOutOrder[i][j]由下式计算：

```
for( j = 0, loop = 0; j < PicSizeInMapUnits; loop++ ) {
  for( k = 0; k < zero_run_length[ i ][ loop ]; k++ )
    spareUnitFlagInBoxOutOrder[ i ][ j ] = spareUnitFlagInBoxOutOrder[ i − 1 ][ j++ ]    (D-5)
  spareUnitFlagInBoxOutOrder[ i ][ j ] = !spareUnitFlagInBoxOutOrder[ i − 1 ][ j++ ]
}
```

D.2.10 场景信息SEI消息的语义

场景和场景转换在这里定义为输出顺序上连续的一组图像。

注1 — 一个场景里的解码图像一般有相近的内容。场景信息SEI消息用场景标识符标注图像和指示场景改变。这个消息描述被标注的图像的源图像是如何创建的。解码器可以用这个信息选择合适的算法以掩盖传输错误。例如，一个特定的算法可以掩盖属于渐变场景中的图像的传输错误。此外，场景信息SEI消息还可以在由应用决定的方式下应用，比如在编码序列中检索一个场景。

场景信息 SEI 消息根据解码顺序为所有图像作标注，从与本 SEI 消息关联的主编码图像（见 7.4.1.2.3），到下一个与场景信息 SEI 消息关联的主编码图像之前，或到比特流的最后一个访问单元。这些图像在此被称作目标图像。

scene_info_present_flag 等于 0 表示目标图像所属的场景或场景转换未指定。scene_info_present_flag 等于 1 表示目标图像属于同一个场景或场景变换。

scene_id 标识目标图像所属的场景。当目标图像的 scene_transition_type 值小于 4，且在输出顺序上其前一幅图像的 scene_transition_type 值小于 4，且目标图像的 scene_id 的值与其前前一幅图像的 scene_id 的值相同时，表示目标图像的源场景和其前一幅图像的源场景被编码器认为是同一个场景。当 scene_transition_type 的值大于 3，且在输出顺序上其前一幅图像的 scene_transition_type 值小于 4，且目标图像的 scene_id 的值与其前前一幅图像的 scene_id 的值相同时，表示目标图像的源场景之一和其前一幅图像的源场景被编码器认为是同一个场景。当 scene_id 的值不等于其前前一幅图像的 scene_id 时，表示目标图像的源场景和输出顺序上其前一幅图像的源场景被编码器认为是不同的场景。

scene_id 的值应在 0 到 $2^{32}-1$ 的范围内。scene_id 的值在 0 到 255 和 512 到 $2^{31}-1$ 时，可以由应用决定其使用。scene_id 的值在 256 到 511，和 2^{31} to $2^{32}-1$ 的范围，被 ITU-T | ISO/IEC 保留给未来使用。解码器在遇到 scene_id 的值在 256 到 511，和 2^{31} 到 $2^{32}-1$ 的范围时，应忽略之（从比特流中删除丢弃）。

scene_transition_type 指定目标图像涉及的场景变换（如果存在）的类型。scene_transition_type 的有效值见表 D-4。

表 D-4－scene_transition_type 的值

值	解释
0	无变换
1	淡出
2	淡入
3	从或至固定色彩的未指明的变换
4	叠化，一幅图像渐隐，而另一幅图像在其后渐显
5	换景；用一条线穿过显示器以转换图像
6	两个场景的未指明的混合

当 scene_transition_type 大于 3 时，目标图像包括由它自己的 scene_id 标记的场景，和由 second_scene_id（见下文）标记的输出顺序上下一个场景的内容。术语“the current scene”用来表示由 scene_id 标记的场景。术语“the next scene”表示由 second_scene_id 标记的场景。对输出顺序上后续的任何图像，其 scene_id 不必等于本 SEI 消息的 second_scene_id。

场景变换类型具体描述如下：

“无变换”表示目标图像不涉及渐变的场景变换。

注2 — 当两个输出顺序相连的图像的场景变换类型都是0但有不同的scene_id值时，两个图像间发生场景的切换。

“淡出”表示目标图像是输出顺序中涉及淡出场景变换的一个图像序列的一部分，就是说，场景的亮度值逐渐达到 0，同时场景的色度值逐渐达到 128。

注3 — 当被标注的两幅图像属于相同的场景变换，场景的变换类型是淡出时，后输出的那幅比前面的暗。

“淡入”表示目标图像是输出顺序中涉及淡出场景变换的一个图像序列的一部分，就是说，场景的亮度值逐渐远离 0，同时场景的色度值逐渐远离 128。

注4 — 当被标注的两幅图像属于相同的场景变换，场景的变换类型是淡入时，后输出的那幅比前面的亮。

“叠化”表示在编码前每个目标图像的样值是计算本场景的一幅图像和下一场景一幅图像的 co-located 样值的加权和而生成的。本场景的权重从全部逐渐减少到 0，反之下一场景的权重从 0 逐渐增加到全部。当被标注的两

幅图像属于同一场景变换，且它们的 scene_transition_type 是“叠化”时，输出顺序上后一幅图像的当前场景的权重小于前一幅的，而后一幅图像的下一场景的权重大于前一幅的。

“换景”表示在编码前每个目标图像的样值中，一些是通过复制本场景的图像的 co-located 样值生成的，剩下的那些是通过复制下一场景的图像的 co-located 样值生成的。当被标注的两幅图像属于同一场景变换，且它们的 scene_transition_type 是“换景”时，输出顺序上后一幅图像复制自下一场景的样值数大于前一幅的。

second_scene_id 标识目标图像所属的场景变换中的下一场景。second_scene_id 的值应不等于 scene_id 的值。second_scene_id 的值应不等于输出顺序中前一幅图像的 scene_id 的值。当输出顺序中下一幅图像被标注的 scene_transition_type 值小于 4，且其 second_scene_id 值与 scene_id 值相同时，表示编码器认为目标图像的源场景之一与输出顺序中下一幅图像的源场景是同一个场景。当 second_scene_id 的值不等于 scene_id 的值或输出顺序中下一幅图像的 second_scene_id（如果存在）的值时，表示编码器认为目标图像与输出顺序中下一幅图像来自不同的源场景。

当一幅图像的 scene_id 值等于输出顺序中下一幅图像的 scene_id 值，且两幅图像的 scene_transition_type 都小于 4 时，表示编码器认为这两幅图像来自相同的源场景。当一幅图像的 scene_id 、scene_transition_type 和 second_scene_id（如果存在）的值对应等于输出顺序中下一幅图像的 scene_id 、scene_transition_type 和 second_scene_id，且 scene_transition_type 的值大于 0 时，表示编码器认为这两幅图像来自相同的源场景变换。

second_scene_id 的值在 0 到 $2^{32}-1$ 的范围内，second_scene_id 的值在 0 到 255 和 512 到 $2^{31}-1$ 时，可以由应用决定其使用。second_scene_id 的值在 256 到 511，和 2^{31} 到 $2^{32}-1$ 的范围，被 ITU-T | ISO/IEC 保留给未来使用。解码器在遇到 second_scene_id 的值在 256 到 511，和 2^{31} 到 $2^{32}-1$ 的范围时，应忽略之（从比特流中删除丢弃）。

D.2.11 子序列信息SEI消息语义

子序列信息 SEI 消息用于说明一幅图像在由子序列层和子序列组成的数据相关结构中的位置。

一个子序列层包含一个序列中的编码图像的子集。子序列层以非负整数编号。层编号大的层是层编号小的层的上层。这些层按照互相之间的从属关系分等级排列，这样某一层的任一图像不能根据任何其上层的任一图像去预测。

注1 — 换一种说法，第0层的任一图像不能根据第1层或更上层的任一图像预测，第1层的图像可以根据第0层预测，第2层的图像可以根据第0和1层预测，等等。

注2 — 主观质量随着解码的层数增加而提高。

子序列是子序列层中的一组编码图像。一幅图像应属于且只属于一个子序列层和一个子序列。子序列中的任一幅图像不能根据同一子序列层或更上层的其他子序列去预测。第 0 层的子序列不依赖不属于该子序列的任一图像即可独立解码。

子序列信息 SEI 消息关系到当前的访问单元。该访问单元中的主编码图像在此指当前图像。

子序列信息 SEI 消息只有在被与子序列信息 SEI 消息关联的图像引用的序列参数组中的 gaps_in_frame_num_value_allowed_flag 等于 1 时才存在。

sub_seq_layer_num 指明当前图像的子序列层编号。当 sub_seq_layer_num 大于 0 时，内存管理控制操作不可以在当前图像的任一条带头中使用。在当前图像属于一个子序列，该子序列在输出顺序中的第一幅图像是一幅 IDR 图像时，sub_seq_layer_num 应等于 0。对一个不成对的参考场，sub_seq_layer_num 应等于 0。sub_seq_layer_num 应在 0 到 255 的范围内。

sub_seq_id 指明一个层内的子序列。在当前图像属于一个子序列，该子序列的第一幅图像是一幅 IDR 图像时，sub_seq_id 的值应等于 IDR 图像的 idr_pic_id 值。sub_seq_id 应在 0 到 255 的范围内。

first_ref_pic_flag 等于 1 表示当前图像是子序列在解码顺序中的第一幅参考图像。在当前图像不是子序列在解码顺序中的第一幅参考图像时，first_ref_pic_flag 应等于 0。

leading_non_ref_pic_flag 等于 1 表示当前图像是子序列的任一参考图像在解码顺序中的前一幅非参考图像或者子序列没有参考图像。在当前图像是一幅参考图像或当前图像是子序列的至少一幅参考图像之后的非参考图像时，leading_non_ref_pic_flag 应等于 0。

last_pic_flag 等于 1 表示当前图像是子序列在解码顺序中，包括所有参考图像和非参考图像的最后一幅图像。在当前图像不是子序列在解码顺序中的最后一幅图像时，last_pic_flag 应等于 0。

当前图像按如下方式分配给子序列：

— 在满足下面的一个或多个条件下，当前图像是子序列在解码顺序中的第一幅图像。

  — 在解码顺序中没有较前面的图像sub_seq_id 和 sub_seq_layer_num值与当前图像相同。

  — leading_non_ref_pic_flag的值等于1，且在解码顺序中前面与当前图像有相同sub_seq_id 和 sub_seq_layer_num值的图像的leading_non_ref_pic_flag的值等于0。

  — first_ref_pic_flag 的值等于 1，且在解码顺序中前面与当前图像有相同sub_seq_id 和 sub_seq_layer_num值的图像的leading_non_ref_pic_flag的值等于0。

  — 在解码顺序中前面与当前图像有相同sub_seq_id 和sub_seq_layer_num值的图像的last_pic_flag的值等于1。

— 否则，当前图像和在解码顺序中与当前图像有相同sub_seq_id 和sub_seq_layer_num值的图像属于同一个子序列。

sub_seq_frame_num_flag 等于 0 表示 sub_seq_frame_num 不存在。sub_seq_frame_num_flag 等于 1 表示 sub_seq_frame_num 存在。

sub_seq_frame_num 对一个子序列输出顺序中的第一幅参考图像和其后的所有非参考图像应等于 0。sub_seq_frame_num 的进一步规定如下。

— 如果当前图像不是一个互补的场对中的第二场，sub_seq_frame_num应以子序列中前一参考图像的MaxFrameNum为模，在此范围内逐一递增。

— 否则（当前图像是一个互补的场对中的第二场），sub_seq_frame_num的值应等于互补的场对中第一场的sub_seq_frame_num值。

sub_seq_frame_num 应在 0 到 MaxFrameNum 的范围内取值。

在当前图像是一幅 IDR 图像时，子序列层 0 应开始一个新的子序列。这样，sub_seq_layer_num 应为 0，sub_seq_id 应与子序列层 0 的前一个子序列不同，first_ref_pic_flag 应为 1，leading_non_ref_pic_flag 应等于 0。

当一个互补场对的两个编码场都存在子序列信息 SEI 消息时，两者存在的 sub_seq_layer_num，sub_seq_id，leading_non_ref_pic_flag 和 sub_seq_frame_num 的值应是一致的。

当一个互补场对只有一个编码场有子序列信息 SEI 消息时，存在的 sub_seq_layer_num，sub_seq_id，leading_non_ref_pic_flag 和 sub_seq_frame_num 的值对互补场对中的另一个编码场也是适用的。

D.2.12 子序列层特征SEI消息的语义

子序列层特征 SEI 消息描述子序列层的特征。

num_sub_seq_layers_minus1 加 1 表示在本序列中的子序列层的编号。num_sub_seq_layers_minus1 应在 0 到 255 范围内取值。

一对 average_bit_rate 和 average_frame_rate 表征每个子序列层。第一对 average_bit_rate 和 average_frame_rate 表示子序列层 0 的特征。如果存在第二对 average_bit_rate 和 average_frame_rate，它表示子序列层 0 和 1 的共同特征。每一对 average_bit_rate 和 average_frame_rate 表示范围从层 0 到层循环计数器指示的层数的子序列层的特征。该值从其被解码到其更新值被解码的时间内有效。

accurate_statistics_flag 等于 1 表示 average_bit_rate 和 average_frame_rate 的值是统计修正值的舍入。accurate_statistics_flag 等于 0 表示 average_bit_rate 和 average_frame_rate 的值是估计值，可能与修正值有些偏差。

当 accurate_statistics_flag 等于 0 时，用于计算 average_bit_rate 和 average_frame_rate 的值的近似值的质量由编码过程规定，不由本建议书 | 国际标准规定。

average_bit_rate 表示以每秒 1000 比特为单位的比特速率。所有在子序列层以上的范围内的 NAL 单元都在计算考虑之列。平均比特率根据本建议书 | 国际标准附件 C 中说明的访问单元删除时间得到。在下文中，bTotal 是子序列层特征 SEI 消息之后（包括当前访问单元的 NAL 单元的比特），下一个（输出顺序中）包含子序列层特征 SEI 消息的访问单元（如果存在）或流结束（否则）之前的所有 NAL 单元的比特数。t_1 是以秒为单位的当前访问单元删除时间，t_2 是下一个（输出顺序中）包含子序列层特征 SEI 消息的访问单元（如果存在）或流结束（否则）之前的最后一个访问单元的删除时间 （以秒为单位）。

当 accurate_statistics_flag 等于 1 时，应满足下列条件。

— 如果t_1不等于t_2，下面的条件应为真

$$\text{average\_bit\_rate} == \text{Round}(\text{bTotal} \div ((t_2 - t_1) * 1000))) \quad \text{(D-6)}$$

— 否则（t_1等于t_2），下面的条件应为真

$$\text{average\_bit\_rate} == 0 \quad \text{(D-7)}$$

average_frame_rate 表示以帧数/(256 秒)为单位的目标子序列的平均帧率。目标子序列的所有 NAL 单元都在计算考虑之列。下文中 fTotal 为当前图像(包括)与下一子序列层特征 SEI 消息（如果有的话）或流的结尾（其它）间帧、补偿场对与非成对场的数目，t_1 是以秒为单位的子序列的当前访问单元的删除时间，t_2 是以秒为单位的子序列的最后一个访问单元的删除时间。

当 accurate_statistics_flag 等于 1 时，下述条件应得到满足：

— 如果t_1不等于t_2，下述条件为真：

$$\text{average\_frame\_rate} == \text{Round}(\text{fTotal} * 256 \div (t_2 - t_1)) \quad \text{(D-8)}$$

— 否则（t_1等于t_2），下述条件为真：

$$\text{average\_frame\_rate} == 0 \quad \text{(D-9)}$$

D.2.13 子序列特征SEI消息的语义

子序列特征 SEI 消息描述子序列的特征。它还描述子序列之间的内部预测相关性。本消息应包含在它所应用的子序列的解码顺序中的第一个访问单元中。在这里这个子序列称作目标子序列。

sub_seq_layer_num 指明当前图像的子序列层编号。sub_seq_layer_num 应在 0 到 255 的范围内。

sub_seq_id 标识目标子序列。sub_seq_id 应在 0 到 65535 的范围内。

duration_flag 等于 0 表示目标子序列的持续时间未规定。

sub_seq_duration 表示以 90kHz 时钟的时钟记录数为单位的目标子序列的持续时间。

average_rate_flag 等于 0 表示目标子序列的平均比特率和平均帧率未规定。

accurate_statistics_flag 表示 average_bit_rate 和 average_frame_rate 的数值的可信度。accurate_statistics_flag 等于 1 表示 average_bit_rate 和 average_frame_rate 的数值是由统计修正值舍入得到的。accurate_statistics_flag 等于 0 表示 average_bit_rate 和 average_frame_rate 的数值是估计的，可能偏离统计修正值。

average_bit_rate 表示以每秒 1000 比特为单位的比特速率。所有在子序列层以上的范围内的 NAL 单元都在计算考虑之列。平均比特率根据 C.1.2 小节说明的访问单元删除时间得到。在下文中，nB 是子序列中所有 NAL 单元的比特数。t_1 是以秒为单位的子序列的第一个访问单元（输出顺序中）的删除时间，t_2 是以秒为单位的子序列的最后一个访问单元（输出顺序中）的删除时间。

当 accurate_statistics_flag 等于 1 时，应满足下列条件。

— 如果t_1不等于t_2，下面的条件应为真

$$\text{average\_bit\_rate} == \text{Round}(nB \div ((t_2 - t_1) * 1000)) \quad \text{(D-10)}$$

— 否则（t_1等于t_2），下面的条件应为真

$$\text{average\_bit\_rate} == 0 \quad \text{(D-11)}$$

average_frame_rate 表示以帧数/(256 秒)为单位的目标子序列的平均帧率。目标子序列的所有 NAL 单元都在计算考虑之列。平均帧率根据 C.1.2 小节说明的访问单元删除时间得到。在下文中，fC 是子序列中帧，互补场对和不成对场的总数。t_1 是以秒为单位的子序列的第一个访问单元（输出顺序中）的删除时间，t_2 是以秒为单位的子序列的最后一个访问单元（输出顺序中）的删除时间。

当 accurate_statistics_flag 等于 1 时，下述条件应被满足：

— 如果t_1不等于t_2，下面的条件应为真

$$\text{average\_frame\_rate} == \text{Round}(fC * 256 \div (t_2 - t_1)) \quad \text{(D-12)}$$

— 否则（t_1等于t_2），下面的条件应为真

$$\text{average\_frame\_rate} == 0 \quad \text{(D-13)}$$

num_referenced_subseqs 表示包含用于对目标子序列中图像进行内部预测的参考图像的子序列的数量。num_referenced_subseqs 应在 0 到 255 的范围内。

ref_sub_seq_layer_num，**ref_sub_seq_id**，和 **ref_sub_seq_direction** 标注包含用于对目标子序列中图像进行内部预测的参考图像的子序列。由 ref_sub_seq_direction 决定下面的使用方法。

— 如果ref_sub_seq_direction等于0，一组由sub_seq_id等于ref_sub_seq_id的子序列组成的候选子序列，位于sub_seq_layer_num等于ref_sub_seq_layer_num的子序列层中，它在解码顺序中的第一幅图像在目标子序列之前。

— 否则（ref_sub_seq_direction等于1），一组由sub_seq_id等于ref_sub_seq_id的子序列组成的候选子序列，位于sub_seq_layer_num等于ref_sub_seq_layer_num的子序列层中，它在解码顺序中的第一幅图像在目标子序列之后。

作为目标子序列参考的子序列是候选子序列组中的子序列，它的第一幅图像在输出顺序中离目标子序列的第一幅图像最近。

D.2.14 全帧冻结SEI消息语义

全帧冻结 SEI 消息指示当前图像和输出顺序中满足特定条件的后续图像不应影响显示的内容。在任何访问单元中出现全帧冻结 SEI 消息不能超过一个。

full_frame_freeze_repetition_period 表示全帧冻结 SEI 消息的持续时间，还可以表示一个图像顺序计数间隔，在其间比特流中出现另一个全帧冻结 SEI 消息或全帧冻结解除 SEI 消息，或视频编码序列结束。full_frame_freeze_repetition_period 的值在 0 到 16384 之间。

full_frame_freeze_repetition_period 等于 0 表示全帧冻结 SEI 消息只作用于当前解码图像。

full_frame_freeze_repetition_period 等于 1 表示全帧冻结 SEI 消息在输出顺序中持续存在直至下面任一条件为真。

— 一个新的视频编码序列开始。

— 包含全帧冻结SEI消息或全帧冻结解除SEI消息的访问单元中的一幅图像输出，其PicOrderCnt( )大于PicOrderCnt(CurrPic)。

full_frame_freeze_repetition_period 大于 1 表示全帧冻结 SEI 消息在输出顺序中持续存在直至下面任一条件为真。

— 一个新的视频编码序列开始。

— 包含全帧冻结SEI消息或全帧冻结解除SEI消息的访问单元中的一幅图像输出，其PicOrderCnt( )大于PicOrderCnt(CurrPic)且小于等于PicOrderCnt(CurrPic) + full_frame_freeze_repetition_period。

full_frame_freeze_repetition_period 大于 1 预示着对输出的访问单元的 PicOrderCnt( ) 大于 PicOrderCnt(CurrPic)且小于等于 PicOrderCnt(CurrPic)+ full_frame_freeze_repetition_period，的一幅图像，存在另一个全帧冻结 SEI 消息，除非比特流结束或未输出这样一幅图像而新的视频编码序列开始。

D.2.15 全帧冻结解除SEI消息语义

全帧冻结解除 SEI 消息取消输出顺序中当前图像之前的图像发出的全帧冻结 SEI 消息的作用。全帧冻结解除 SEI 消息表示当前图像和输出顺序中的后续图像将影响显示的内容。

在任何访问单元中出现全帧冻结解除 SEI 消息不能超过一个。全帧冻结解除 SEI 消息不能在包含全帧冻结 SEI 消息的访问单元中出现。如果出现全帧冻结 SEI 消息的访问单元包含互补场对中的一场（互补场对中的两场 PicOrderCnt(CurrPic)相等），那么互补的两个访问单元中不能出现全帧冻结解除 SEI 消息。

D.2.16 全帧快照SEI消息语义

全帧快照 SEI 消息表示当前帧被应用程序标记用作视频内容的一个静止图像快照。

snapshot_id 表示快照的识别号。snapshot_id 应在 0 到 $2^{32} - 1$ 的范围内。

snapshot_id 的值在 0 到 255 和 512 到 $2^{31} - 1$ 时，可以由应用决定其使用。snapshot_id 的值在 256 到 511，和 2^{31} 到 $2^{32} - 1$ 的范围，被 ITU-T | ISO/IEC 保留给未来使用。解码器在遇到 snapshot_id 的值在 256 到 511，和 2^{31} 到 $2^{32} - 1$ 的范围时，应忽略之（从比特流中删除丢弃）。

D.2.17 逐步细化段开始SEI消息语义

逐步细化段开始 SEI 消息表示连续的一组编码图像的开始，被标记的当前图像后面是一个细化当前图像质量的一幅或多幅图像序列，而不是一个对连续运动场景的描述。

被标记的该组连续编码图像应连续出现，直到下列条件之一为真。当下列条件之一为真时，被解码的下一条带不再属于被标记的该组连续编码图像。

1．被解码的下一条带属于一幅 IDR 图像。

2．num_refinement_steps_minus1 大于 0 且被解码的下一条带的 frame_num 是 (currFrameNum + num_refinement_steps_minus1 + 1) % MaxFrameNum，此处 currFrameNum 是包含本 SEI 消息的访问单元中图像的 frame_num 值。

3．num_refinement_steps_minus1 为 0 且一个 progressive_refinement_id 与本 SEI 消息相同的逐步细化段结束 SEI 消息被解码。

被标记的连续编码图像组内的解码顺序应与它们的输出顺序相同。**progressive_refinement_id** 表示逐步细化操作的识别编号。progressive_refinement_id 应在 0 到 $2^{32} - 1$ 的范围内。

progressive_refinement_id 的值在 0 到 255 和 512 到 $2^{31} - 1$ 时，可以由应用决定其使用。progressive_refinement_id 的值在 256 到 511，和 2^{31} 到 $2^{32} - 1$ 的范围，被 ITU-T | ISO/IEC 保留给未来使用。解码器在遇到 progressive_refinement_id 的值在 256 到 511，和 2^{31} 到 $2^{32} - 1$ 的范围时，应忽略（从比特流中删除丢弃）。

num_refinement_steps_minus1 表示被标记的连续编码图像组内参考帧的数目，如下所述。

— 如果num_refinement_steps_minus1等于0，被标记的连续编码图像组内参考帧的数目未知。

— 否则，被标记的连续编码图像组内参考帧的数目等于num_refinement_steps_minus1 + 1。

num_refinement_steps_minus1 应在 0 到 MaxFrameNum − 1 的范围内。

D.2.18 逐步细化段结束SEI消息语义

逐步细化段结束 SEI 消息表示连续的一组编码图像的结束。逐步细化段开始 SEI 消息标记一幅图像作为初始图像，后续是一个细化初始图像质量的一幅或多幅图像序列，至本图像结束。

progressive_refinement_id 表示逐步细化操作的识别编号。progressive_refinement_id 应在 0 到 $2^{32} - 1$ 的范围内。

逐步细化段结束 SEI 消息表示由有相同 progressive_refinement_id 的逐步细化段开始 SEI 消息启动的一个逐步细化段的结束。

progressive_refinement_id 的值在 0 到 255 和 512 到 $2^{31} - 1$ 时，可以由应用决定其使用。progressive_refinement_id 的值在 256 到 511，和 2^{31} 到 $2^{32} - 1$ 的范围，被 ITU-T | ISO/IEC 保留给未来使用。解码器在遇到 progressive_refinement_id 的值在 256 到 511，和 2^{31} 到 $2^{32} - 1$ 的范围时，应忽略之（从比特流中删除丢弃）。

D.2.19 运动受限条带组集SEI消息语义

本 SEI 消息表示跨越条带组边界的内部预测受如下限制。如果存在，根据 7.4.1.2.3 小节，本消息应只在它相关联的 IDR 访问单元处出现。

本 SEI 消息的目标图像集包含从相关联的主编码 IDR 图像开始，到下一个主编码 IDR 图像前（不包括该 IDR 图像），或比特流中后续不再有主编码 IDR 图像的最后一个主编码图像为止，所有顺序相连的主编码图像。条带组集是一个或多个条带组的集合，用语法元素 slice_group_id[i]予以标识。

本 SEI 消息预示，对目标图像集中的每幅图像，内部预测过程受如下限制：条带组集以外的样值，和由条带组集以外的一个或多个样值得到的局部样点位置处的样值，不用于内部预测条带组集内的任何样值。

num_slice_groups_in_set_minus1 加 1 表示条带组集中条带组的数目。num_slice_groups_in_set_minus1 的允许范围是 0 到 num_slice_groups_minus1。num_slice_groups_minus1 的允许范围参见附件 A。

slice_group_id[i] 标识条带组集中的条带组。其允许的范围是 0 到 num_slice_groups_in_set_minus1。slice_group_id[i]语法元素的长度是 Ceil(Log2(num_slice_groups_minus1 + 1))比特。

exact_sample_value_match_flag 等于 0 表示，目标图像组内，如果不属于条带组集的宏块未被解码，条带组集里的每个样点的值不一定严格地与所有宏块被解码后的样值相同。exact_sample_value_match_flag 等于 1 表示，目标图像组内，如果不属于条带组集的宏块未被解码，条带组集里的每个样点的值应严格地与所有宏块被解码后的样值相同。

注1 — 当目标图像组中所有条带的disable_deblocking_filter_idc等于2时，exact_sample_value_match_flag应等于1。

pan_scan_rect_flag 等于 0 表示 pan_scan_rect_id 不存在。pan_scan_rect_flag 等于 1 表示 pan_scan_rect_id 存在。

pan_scan_rect_id 表示指定的条带组集至少覆盖目标图像集内 pan_scan_rect_id 标示的泛扫描矩形。

注2 — 若干约束运动条带组集SEI消息可以关联到同一个IDR图像。因此，在一个目标图像集中，可以有多个有效的条带组集。

注3 — 条带组集中的条带组的大小、形状、位置可能在目标图像集内有所变化。

D.2.20 胶片颗粒特征SEI消息语义

本 SEI 消息为解码器提供一个颗粒合成的参数化模型。例如，一个编码器可以用胶片颗粒特征 SEI 消息描述原始视频资料中出现的，被预处理滤波技术消除了的胶片颗粒。在显示过程中对解码图像进行胶片颗粒的模拟合成是可选的，不影响本建议书 | 国际标准中规定的解码过程。如果在显示过程中对解码图像执行胶片颗粒的模拟合成，不要求执行合成的方法与胶片颗粒特征 SEI 消息提供的胶片颗粒的参数化模型相同。

注1 — 本建议书 | 国际标准不规定显示过程。

film_grain_characteristics_cancel_flag 等于 1 表示本 SEI 消息取消输出顺序中之前的胶片颗粒特征 SEI 消息的作用。film_grain_characteristics_cancel_flag 等于 0 表示其后是胶片颗粒模型信息。

model_id 标识胶片颗粒仿真模型，见表 D-5。model_id 的值应在 0 到 1 的范围内。

表 D-5－model_id值

值	解释
0	频率过滤
1	自动消退
2	保留
3	保留

separate_colour_description_present_flag 等于 1 表示胶片颗粒特征 SEI 消息语法结构中存在描述明确的颜色空间的胶片颗粒特征。separate_colour_description_present_flag 等于 0 表示 SEI 消息中的胶片颗粒特征的颜色描述与视频编码序列相同，参见 E.2.1 小节。

注2 — 如果separate_colour_description_present_flag等于1，本SEI消息中规定的胶片颗粒特征的颜色空间可能与E.2.1小节规定的视频编码序列的颜色空间不同。

film_grain_bit_depth_luma_minus8 加 8 表示在本 SEI 消息中用于规定胶片颗粒特征的亮度部分的比特深度。如果胶片颗粒特征 SEI 消息中不存在 film_grain_bit_depth_luma_minus8，应推断 film_grain_bit_depth_luma_minus8 的值等于 bit_depth_luma_minus8。

filmGrainBitDepth[0]的值由下式计算

filmGrainBitDepth[0] = film_grain_bit_depth_luma_minus8 + 8 (D-14)

film_grain_bit_depth_chroma_minus8 加 8 表示在本 SEI 消息中用于规定胶片颗粒特征的 Cb 和 Cr 部分的比特深度。如果胶片颗粒特征 SEI 消息中不存在 film_grain_bit_depth_chroma_minus8，应推断 film_grain_bit_depth_chroma_minus8 的值等于 bit_depth_chroma_minus8。

filmGrainBitDepth[c]（c = 1 和 2）的值由下式计算

filmGrainBitDepth[c] = film_grain_bit_depth_chroma_minus8 + 8，c = 1, 2 (D-15)

film_grain_full_range_flag 与 E.2.1 小节规定的 video_full_range_flag 语法元素有相同的语义，特例如下。

— film_grain_full_range_flag表示胶片颗粒特征的颜色空间由本SEI消息规定，而不是用视频编码序列的颜色空间。

— 当胶片颗粒特征SEI消息中不存在film_grain_full_range_flag时，应推断film_grain_full_range_flag的值等于video_full_range_flag。

film_grain_colour_primaries 与E.2.1小节规定的colour_primaries语法元素有相同的语义，特例如下。

— film_grain_colour_primaries表示胶片颗粒特征的颜色空间由本SEI消息规定，而不是用视频编码序列的颜色空间。

— 如果胶片颗粒特征SEI消息中不存在film_grain_colour_primaries，应推断film_grain_colour_primaries的值等于colour_primaries。

film_grain_transfer_characteristics 与 E.2.1 小节规定的 transfer_characteristics 语法元素有相同的语义，特例如下。

— film_grain_transfer_characteristics表示胶片颗粒特征的颜色空间由本SEI消息规定，而不是用视频编码序列的颜色空间。

— 如果胶片颗粒特征SEI消息中不存在film_grain_transfer_characteristics，应推断film_grain_transfer_characteristics的值等于transfer_characteristics。

film_grain_matrix_coefficients 与 E.2.1 小节规定的 matrix_coefficients 语法元素有相同的语义，特例如下。

— film_grain_matrix_coefficients表示胶片颗粒特征的颜色空间由本SEI消息规定，而不是用视频编码序列的颜色空间。

— 如果胶片颗粒特征SEI消息中不存在film_grain_matrix_coefficients，应推断film_grain_matrix_coefficients的值等于matrix_coefficients。

— film_grain_matrix_coefficients的允许值不受chroma_format_idc的限制。

应推断胶片颗粒特征 SEI 消息规定的胶片颗粒特征的 chroma_format_idc 等于 3（4:4:4）。

注3 — 因为执行用于显示处理的胶片颗粒生成功能不要求明确具体的方法，如果解码器愿意，它可以变换减少模型的色度信息去仿真其他色度格式（4:2:0或4:2:2）的胶片颗粒，而不是在执行颗粒生成之前变换增加视频解码信息（方法不由本建议书 | 国际标准规定）。

blending_mode_id 表示将模拟胶片颗粒与解码图像混合的方式，见表 D-6。blending_mode_id 应在 0 到 1 的范围内。

表 D-6－blending_mode_id值

值	描述
0	加性
1	乘积
2	保留
3	保留

混合方式根据 blending_mode_id 由下述方法确定。

— 如果blending_mode_id等于0，混合方式是加性的如下式

$$I_{grain}[x, y, c] = Clip3(0, (1 << filmGrainBitDepth[c]) - 1, I_{decoded}[x, y, c] + G[x, y, c]) \quad (D\text{-}16)$$

— 否则（blending_mode_id等于1），混合方式是乘积形式的如下式

$$I_{grain}[x, y, c] = Clip3(0, (1 << filmGrainBitDepth[c]) - 1, I_{decoded}[x, y, c] * (1 + G[x, y, c])) \quad (D\text{-}17)$$

此处 $I_{decoded}[x, y, c]$代表在坐标 x, y 处解码图像 $I_{decoded}$的颜色部分 c 的样值，G[x, y, c]是在相同的位置和颜色部分的模拟胶片颗粒的值，而 filmGrainBitDepth[c]是数组 $I_{grain}[x, y, c]$中用定长无符号二进制表示的每个样值的比特数。

log2_scale_factor 表示用于胶片颗粒特征方程的比例因子。

comp_model_present_flag[c] 等于 0 表示胶片颗粒不由第 c 个颜色部分模拟，c 等于 0 指亮度部分，c 等于 1 指 Cb 部分，c 等于 2 指 Cr 部分。comp_model_present_flag[c]等于 1 表示本 SEI 消息中存在为胶片颗粒的第 c 个颜色部分模拟的语法元素。

num_intensity_intervals_minus1[c] 加 1 表示对估计的特定的一组模型值的强度间隔的数量。

注4 — 强度间隔可以重叠以模拟多级的胶片颗粒。

num_model_values_minus1[c] 加 1 表示每个强度间隔中模拟的胶片颗粒出现的模拟值的数量。num_model_values_minus1[c]的值应在 0 到 5 的范围内。

intensity_interval_lower_bound[c][i] 表示对模拟值组应用的强度级别的间隔 i 的下限。

intensity_interval_upper_bound[c][i] 表示对模拟值组应用的强度级别的间隔 i 的上限。

模拟值组的选择根据 model_id 由下述方法确定。

— 如果model_id等于0，每个$I_{decoded}$中16 x16样点块b的平均值，作为b_{avg}，用于根据面向块中所有样值的索引s[j]选择模拟值组。

```
for( i = 0, j = 0; i <= num_intensity_intervals_minus1; i++ )
  if( b_avg >= intensity_interval_lower_bound[ c ][ i ] && b_avg <= intensity_interval_upper_bound[ c ][ i ] ) {
    s[ j ] = i                                                                    (D-18)
    j++
  }
```

— 否则（model_id等于1），生成胶片颗粒的模拟值组对$I_{decoded}$中的每个样值按下式选择。

```
for( i = 0, j = 0; i <= num_intensity_intervals_minus1; i++ )
  if( I_decoded[ x, y, c ] >= intensity_interval_lower_bound[ c ][ i ] &&
      I_decoded[ x, y, c ] <= intensity_interval_upper_bound[ c ][ i ] ) {        (D-19)
   s[ j ] = i
   j++
  }
```

没有进入任何预定的间隔范围的样值不被颗粒生成功能改变。进入不止一个间隔范围的样值会产生多级颗粒。多级颗粒源自对每个强度间隔独立计算颗粒的叠加。

comp_model_value[c][i][j] 表示对颜色部分 c 和强度间隔 i 的各个模拟值。模拟值组根据其 model_id 的值有不同的含义。comp_model_value[c][i][j]的值应受如下限制，且还可能在本小节的其他部分有附加的限制。

— 如果model_id等于0，comp_model_value[c][i][j]应在0到$2^{filmGrainBitDepth[c]} - 1$的范围内。

— 否则（model_id等于1），comp_model_value[c][i][j] 应在$-2^{(filmGrainBitDepth[c] - 1)}$到$2^{(filmGrainBitDepth[c] - 1)} - 1$的范围内。

胶片颗粒合成根据 model_id 的不同采用下列模型。

— 如果model_id等于0，对原始胶片颗粒（c = 0..2，x = 0..$PicWidthInSamples_L$，且 y = 0..$PicHeightInSamples_L$）的模拟可以使用频率滤波模型，如下：

G[x, y, c] = (comp_model_value[c][s][0] * Q[x, y, c] + comp_model_value[c][s][5] * G[x, y, c−1]) >> log2_scale_factor (D-20)

此处 Q[c]是一个 2 维随机过程，以归一化高斯分布生成的随机值 $gaussRv_{ij}$（独立的分布一致的高斯随机变量样值的均值为 0，单位方差）对 16x16 块的 gaussRv 滤波而得到。用于方程式右侧的一个 G[x, y, c−1]的值在 c-1 小于 0 时推定为 0。

注5 — 一个归一化高斯随机值可以由两个独立的，在0到1之间（不等于0）一致分布的随机数值生成，两个随机值以uRv_0和uRv_1表示，用Box-Muller变换描述为：

$$gaussRv_{ij} = \sqrt{-2 * Ln(uRv_0)} * Cos(2 * \pi * uRv_1) \quad \text{(D-21)}$$

此处 Ln(x)是 x 的自然对数（以 e 为底，e 是自然对数底常数 2.718 281 828…），Cos(x)是对以弧度为单位的参数 x 的三角余弦函数运算，π是阿基米德常数 3.141 592 653…。

对块的 gaussRv 带通滤波可以在离散余弦变换域内实现如下：

```
for( y = 0; y < 16; y++ )
  for( x = 0; x < 16; x++ )
    if( ( x < comp_model_value[ c ][ s ][ 3 ]  &&  y < comp_model_value[ c ][ s ][ 4 ] ) ||
        x > comp_model_value[ c ][ s ][ 1 ] || y > comp_model_value[ c ][ s ][ 2 ] )
      gaussRv[ x, y ] = 0
filteredRv = IDCT16x16( gaussRv )
```

(D-22)

此处 IDCT16x16(z)是对 16x16 矩阵为参数的归一化的反离散余弦变换（IDCT），如下：

$$IDCT16x16(z) = r * z * r^T \quad \text{(D-23)}$$

式中上标 T 表示矩阵的转置，r 是 16x16 矩阵其元素 r_{ij} 表达式为：

$$r_{ij} = \frac{((i == 0) ? 1 : \sqrt{2})}{4} Cos\left(\frac{i * (2 * j + 1) * \pi}{32}\right) \quad \text{(D-24)}$$

式中 Cos(x)是对单位为弧度的参数 x 的三角余弦函数操作，π是阿基米德常数 3.141 592 653…。

Q[c]由频率滤波后的块 filteredRv 组成。

注6 — 编码模型的值基于16x16块，但是一个具体的解码器实现可以采用其他的块大小。例如，基于8x8块实现IDCT的解码器，对i等于1到4 的编码模型值comp_model_value[c][s][i]，应把变换组降低为2分之一。

注7 — 为降低拼接频率过滤块filteredRv而导致的明显的块效应，解码器可以在频率过滤块的边界处应用低通滤波。

— 否则（model_id等于1），对原始胶片颗粒（c = 0..2, x = 0..PicWidthInSamples$_L$, 且 y = 0..PicHeight-InSamples$_L$）的模拟可以使用自回归模型，如下：

$$
\begin{aligned}
&G[x, y, c] = (comp\_model\_value[c][s][0] * n[x, y, c] + \\
&comp\_model\_value[c][s][1] * (G[x-1, y, c] + ((comp\_model\_value[c][s][4] * G[x, y-1, c]) >> \\
&\quad log2\_scale\_factor)) + \\
&comp\_model\_value[c][s][3] * (((comp\_model\_value[c][s][4] * G[x-1, y-1, c]) >> \\
&\quad log2\_scale\_factor) + G[x+1, y-1, c]) + \\
&comp\_model\_value[c][s][5] * (G[x-2, y, c] + \\
&\quad ((comp\_model\_value[c][s][4] * comp\_model\_value[c][s][4] * G[x, y-2, c]) >> \\
&\quad\quad (2 * log2\_scale\_factor))) + \\
&comp\_model\_value[c][s][2] * G[x, y, c-1]) >> log2\_scale\_factor
\end{aligned}
\tag{D-25}
$$

此处 n[x, y, c]是归一化高斯分布的一个随机值（对每个 x，y 和 c 值，独立的分布一致的高斯随机变量样值的均值为 0，单位方差）。用于方程式右侧的一个 G[x, y, c]的值在满足下列条件时推定为 0。

— x小于0，

— y小于0，

— x大于等于PicWidthInSamples$_L$，

— c小于0。

comp_model_value[c][i][0] 提供由 model_id 指定的模型的第一个模拟值。comp_model_value[c][i][0]符合公式 D-20 到 D-23 所述生成函数中的高斯噪声的标准偏差条件。

comp_model_value[c][i][1] 提供由 model_id 指定的模型的第二个模拟值。comp_model_value[c][i][1]应大于等于 0 小于 16。

如果胶片颗粒特征 SEI 消息中不出现 comp_model_value[c][i][1]，应作如下推断：

— 如果model_id等于0，应推断comp_model_value[c][i][1]等于8。

— 否则（model_id等于1），应推断comp_model_value[c][i][1]等于0。

comp_model_value[c][i][1]可以作如下解释：

— 如果model_id等于0，comp_model_value[c][i][1]表示用于对16x16随机值块作DCT过滤的水平上限截止频率。

— 否则（model_id等于1），comp_model_value[c][i][1]表示相邻样值(x−1, y)和(x, y−1)的一阶空间相关性。

comp_model_value[c][i][2] 提供由 model_id 指定的模型的第三个模拟值。comp_model_value[c][i][2]应大于等于 0 小于 16。

如果胶片颗粒特征 SEI 消息中不出现 comp_model_value[c][i][2]，应作如下推断：

— 如果model_id等于0，应推断comp_model_value[c][i][2] 等于comp_model_value[c][i][1]。

— 否则（model_id等于1），应推断comp_model_value[c][i][2]等于0。

comp_model_value[c][i][2]可以作如下解释：

— 如果model_id等于0，comp_model_value[c][i][2]表示用于对16x16随机值块作DCT过滤的垂直上限截止频率。

— 否则（model_id等于1），comp_model_value[c][i][2]表示两个相邻的颜色部分的颜色相关性。

comp_model_value[c][i][3] 提供由 model_id 指定的模型的第四个模拟值。comp_model_value[c][i][3]应大于等于 0 小于等于 comp_model_value[c][i][1]。

如果胶片颗粒特征 SEI 消息中不出现 comp_model_value[c][i][3]，应推断其等于 0。

comp_model_value[c][i][3]可以作如下解释：

— 如果model_id等于0，comp_model_value[c][i][3]表示用于对16x16随机值块作DCT过滤的水平下限截止频率。

— 否则（model_id等于1），comp_model_value[c][i][3]表示两个相邻样值(x−1, y−1)和(x+1, y−1)的一阶空间相关性。

comp_model_value[c][i][4] 提供由 model_id 指定的模型的第五个模拟值。comp_model_value[c][i][4] 应大于等于 0 小于等于 comp_model_value[c][i][2]。

当胶片颗粒特征 SEI 消息中不出现 comp_model_value[c][i][4]时，应推断其等于 0。

comp_model_value[c][i][4]可以作如下解释：

— 如果model_id等于0，comp_model_value[c][i][4]表示用于对16x16随机值块作DCT过滤的垂直下限截止频率。

— 否则（model_id等于1），comp_model_value[c][i][4]表示模拟颗粒的高宽比。

comp_model_value[c][i][5] 提供由 model_id 指定的模型的第六个模拟值。如果胶片颗粒特征 SEI 消息中不出现 comp_model_value[c][i][5]，应推断其等于 0。

comp_model_value[c][i][5]可以作如下解释：

— 如果model_id等于0，comp_model_value[c][i][5]表示相邻的颜色部分的颜色相关性。

— 否则（model_id等于1），comp_model_value[c][i][5]表示两个相邻样值(x, y−2)和(x−2, y)的二阶空间相关性。

film_grain_characteristics_repetition_period 表示胶片颗粒特征 SEI 消息的持续时间，还可以表示一个图像顺序计数间隔，在其间比特流中出现另一个胶片颗粒特征 SEI 消息或视频编码序列结束。film_grain_characteristics_repetition_period 的值在 0 到 16384 之间。

film_grain_characteristics_repetition_period 等于 0 表示胶片颗粒特征 SEI 消息只作用于当前解码图像。

film_grain_characteristics_repetition_period 等于 1 表示胶片颗粒特征 SEI 消息在输出顺序中持续存在直至下面任一条件为真。

— 一个新的视频编码序列开始；

— 包含胶片颗粒特征SEI消息的访问单元中的一幅图像输出，其PicOrderCnt( )大于PicOrderCnt(CurrPic)。

film_grain_characteristics_repetition_period 大于 1 表示胶片颗粒特征 SEI 消息在输出顺序中持续存在直至下面任一条件为真。

— 一个新的视频编码序列开始；

— 包含胶片颗粒特征SEI消息的访问单元中的一幅图像输出，其PicOrderCnt( )大于PicOrderCnt(CurrPic)且小于等于PicOrderCnt(CurrPic) + film_grain_characteristics_repetition_period。

film_grain_characteristics_repetition_period 大于 1 预示着对输出的访问单元的 PicOrderCnt( )大于 PicOrderCnt(CurrPic)且小于等于 PicOrderCnt(CurrPic) +film_grain_characteristics_repetition_period 的一幅图像，存在另一个胶片颗粒特征 SEI 消息，除非比特流结束或未输出这样一幅图像而新的视频编码序列开始。

D.2.21 去块效应滤波器显示选项SEI消息语义

本 SEI 消息向解码器提示，编码器希望每个解码图像输出时，用 8.7 小节规定的去块效应滤波器方法裁剪的结果进行显示，还是用 8.5.12 小节规定的去块效应滤波器前图像构建方法的裁剪结果进行显示。

注1 — 显示过程不由本建议书 | 国际标准规定。编码器如何在去块效应滤波器显示参考SEI消息中表达它希望的方法也不由本建议书 | 国际标准规定，而且在去块效应滤波器显示参考SEI消息中表达的倾向性表述不对显示过程强加任何要求。

deblocking_display_preference_cancel_flag 等于 1 表示本 SEI 消息取消输出顺序中之前的去块效应滤波器显示参考 SEI 消息的作用。deblocking_display_preference_cancel_flag 等于 0 表示其后有 display_prior_to_deblocking_preferred_flag 和 deblocking_display_preference_repetition_period。

注2 — 如果没有去块效应滤波器显示参考SEI消息，或收到一个deblocking_display_preference_cancel_flag等于1的去块效应滤波器显示参考SEI消息之后，解码器应推断在每幅解码图像输出时，用8.7小节规定的去块效应滤波器方法裁剪的结果进行显示优先于用8.5.12小节规定的去块效应滤波器前图像构建方法的裁剪结果进行显示。

display_prior_to_deblocking_preferred_flag 等于 1 表示对附件 C 中说明的每幅裁剪输出图像，编码器希望的显示过程（不由本建议书 | 国际标准规定）是用 8.5.12 小节规定的去块效应滤波器前图像构建方法的裁剪结果，而不是用 8.7 小节规定的去块效应滤波器方法裁剪的结果。display_prior_to_deblocking_preferred_flag 等于 0 表示对附件 C 中说明的每幅裁剪输出图像，编码器希望的显示过程（不由本建议书 | 国际标准规定）是用 8.7 小节规定的去块效应滤波器方法裁剪的结果，而不是用 8.5.12 小节规定的去块效应滤波器前图像构建方法的裁剪结果。

注3 — 去块效应滤波器显示参考SEI消息存在与否和display_prior_to_deblocking_preferred_flag的值不对本建议书 | 国际标准规定的解码过程的要求产生影响。确切地说，它只在满足本建议书 | 国际标准规定的解码过程的要求之外提供一个指示，如何以一种可选择的方式执行显示过程（不由本建议书 | 国际标准规定）从而获得增强的视觉质量。采用去块效应滤波器显示参考SEI消息的编码器应在设计时了解，除非编码器限制将附件A规定的DPB容量只供给在用的简表和级别，在显示重排序和延迟的图像时，一些解码器可能没有足够的内存容量，在存储了8.7小节规定的去块效应滤波器方法裁剪的结果以外，再提供8.5.12小节规定的去块效应滤波器前图像构建方法的裁剪结果的存储空间，而这样的解码器因此将不能从倾向性提示中获益。通过限制对DPB容量的使用，一个编码器能够使用附件A规定的DPB容量的至少一半，同时允许解码器使用剩余的容量作为被指示为显示参考的未过滤图像在输出时间到来之前的存储空间。

dec_frame_buffering_constraint_flag 等于 1 表示对 max_dec_frame_buffering 规定的 HRD 解码图像缓冲区（DPB）的帧缓存容量的使用受限制，视频编码序列不可以要求解码图像缓存超过 Max(1, max_dec_frame_buffering) 帧，使过滤或未过滤解码图像（由去块效应滤波器显示参考 SEI 消息标示）在图像定时 SEI 消息的 dpb_output_delay 规定的输出时间能够输出。

dec_frame_buffering_constraint_flag 等于 0 表示对 HRD 解码图像缓冲区（DPB）的帧缓存容量的使用可以如 frame_buffering_constraint_flag 等于 1 的方式限制，也可以不限制。

为了确定当 dec_frame_buffering_constraint_flag 等于 1 时的限制，在任意给定的时间点，包含图像的 DPB 的每一个帧缓存，其帧缓存容量的数量应由下述内容得到。

— 如果帧缓存满足下面两条标准，认为它需要占用两帧的缓存空间容量。

  — 帧缓存中有一帧或一个及一个以上的场被标记为“用于参考”，且

  — 帧缓存中有满足下列两条标准的场。

    — 图像的HRD输出时间大于给定的时间点，且

    — 在去块效应滤波器显示参考SEI消息中已经表示，编码器希望的显示图像裁减结果的构建过程优先采用8.5.12小节规定的去块效应过滤前图像构建方法，而不希望采用8.7小节规定的去块效应滤波器方法。

— 否则，认为帧缓存需要占用DPB中的一帧容量作为缓存空间。

如果 dec_frame_buffering_constraint_flag 等于 1，按照这种办法，被 DPB 中所有帧缓存占用的帧缓存区总容量，在 HRD 对视频解码序列操作的过程中，不应大于 Max(1, max_dec_frame_buffering)。

dec_frame_buffering_constraint_flag 的值在视频解码序列的所有去块效应滤波器显示参考 SEI 消息中应相同。

deblocking_display_preference_repetition_period 表示去块效应滤波器显示参考 SEI 消息的持续时间，还可以表示一个图像顺序计数间隔，在其间比特流中出现另一个去块效应滤波器显示参考 SEI 消息或视频编码序列结束。deblocking_display_preference_repetition_period 的值在 0 到 16384 之间。

deblocking_display_preference_repetition_period 等于 0 表示去块效应滤波器显示参考 SEI 消息只作用于当前解码图像。

deblocking_display_preference_repetition_period 等于 1 表示胶片颗粒特征 SEI 消息在输出顺序中持续存在直至下面任一条件为真。

— 一个新的视频编码序列开始，或

— 包含去块效应滤波器显示参考SEI消息的访问单元中的一幅图像输出，其PicOrderCnt( )大于PicOrderCnt(CurrPic)。

deblocking_display_preference_repetition_period 大于 1 表示去块效应滤波器显示参考 SEI 消息在输出顺序中持续存在直至下面任一条件为真。

— 一个新的视频编码序列开始，或

— 包含胶片颗粒特征SEI消息的访问单元中的一幅图像输出，其PicOrderCnt( )大于PicOrderCnt(CurrPic)且小于等于PicOrderCnt(CurrPic) + deblocking_display_preference_repetition_period。

deblocking_display_preference_repetition_period 大于 1 表示对输出的访问单元的 PicOrderCnt( )大于 PicOrderCnt(CurrPic)且小于等于 PicOrderCnt(CurrPic) + deblocking_display_preference_repetition_period 的一幅图像出现另一个去块效应滤波器显示参考 SEI 消息，除非比特流结束或未输出这样一幅图像而新的视频编码序列开始。

D.2.22 立体视频信息SEI消息语义

本 SEI 消息向解码器指示，整个视频编码序列包含由立体视觉内容组成的图像对。

立体视频信息 SEI 消息只能在视频编码序列的第一个访问单元中出现，在其他任何访问单元中不能出现。

field_views_flag 等于 1 表示当前视频编码序列中的所有图像是场图像，某个特定奇偶性的所有场是立体视觉内容的左视图，相对奇偶性的所有场是右视图。field_views_flag equal 等于 0 表示当前视频编码序列中的所有图像是帧图像，在输出顺序中交替出现的帧各自代表立体视觉的一个视图。field_views_flag 的值在一个视频编码序列的所有立体视频信息 SEI 消息中应相同。

如果立体视频信息 SEI 消息存在且 field_views_flag 等于 1，立体视频对的左视图和右视图应被编码为互补场对，场对在输出顺序中的第一场的显示时间应延迟至与第二场的输出时间相同，且每个独立场中样值的空间位置为显示目的应解释为代表图 6-1 所示的完整图像，而不是如图 6-2 所示的一帧中的特定空间位置场。

注 — 显示过程不由本建议书 | 国际标准规定。

top_field_is_left_view_flag 等于 1 表示视频编码序列中的顶场代表左视图而底场代表右视图。top_field_is_left_view_flag 等于 0 表示视频编码序列中的底场代表左视图而顶场代表右视图。如果存在，在一个视频编码序列中所有立体视频信息 SEI 消息的 top_field_is_left_view_flag 的值应相同。

current_frame_is_left_view_flag 等于 1 表示当前图像是立体视频对的左视图。current_frame_is_left_view_flag 等于 0 表示当前图像是立体视频对的右视图。

next_frame_is_second_view_flag 等于 1 当前图像和输出顺序中它的下一幅图像组成一个立体视频对，当前图像的显示时间应延迟以与其下一幅图像保持一致。next_frame_is_second_view_flag 等于 0 表示当前图像和输出顺序中它的上一幅图像组成一个立体视频对，当前图像的显示时间不因立体视频对的原因而延迟。

left_view_self_contained_flag 等于 1 表示对视频编码序列中左视图图像的解码过程没有参考右视图图像的内部预测。left_view_self_contained_flag 等于 0 表示对视频编码序列中左视图图像的解码过程可能有一些（也可能没有）参考右视图图像的内部预测。在一个视频编码序列中所有立体视频信息 SEI 消息的 left_view_self_contained_flag 的值应相同。

right_view_self_contained_flag 等于 1 表示对视频编码序列中右视图图像的解码过程没有参考左视图图像的内部预测。right_view_self_contained_flag 等于 0 表示对视频编码序列中右视图图像的解码过程可能有一些（也可能没有）参考左视图图像的内部预测。在一个视频编码序列中所有立体视频信息 SEI 消息的 right_view_self_contained_flag 的值应相同。

D.2.23 保留SEI消息语义

本消息由 ITU-T | ISO/IEC 为未来的向后兼容性保留的数据组成。遵从本建议书 | 国际标准的编码器不应发送 SEI 消息直至 ITU-T | ISO/IEC 规定了本消息的用法以后。遵从本建议书 | 国际标准的解码器遇到保留 SEI 消息应丢弃其内容而不影响解码的过程，除非 ITU-T | ISO/IEC 在未来的建议书 | 国际标准中具体规定。

reserved_sei_message_payload_byte 是 ITU-T | ISO/IEC 为未来应用保留的字节。

附 件 E

视频可用性信息

（本附件是本建议书 | 国际标准的组成部分）

本附件规定序列参数组中 VUI 参数的语法和语义。

VUI 参数不是解码过程构建亮度或色度样值所必需的。遵从本建议书 | 国际标准的编码器在输出顺序中不必处理本信息（一致性规范见附件 C）。某些 VUI 参数在检查比特流一致性和保证解码器输出定时一致性时是必需的。

在附件 E 中，规范 VUI 参数的出现当这些参数（或它们的子集）由其他本建议书 | 国际标准未规定的方法传递到解码器（或 HRD）的情况下也应得到满足。当在比特流中出现，VUI 参数应跟随在 7.3.2.1、7.4.2.1 和本附件规定的语法和语义之后。当 VUI 参数的内容不在比特流中出现而由其他方法传递给应用时，VUI 参数内容的表现形式不必采用与本附件规定相同的语法。为统计比特数，只有在比特流中实际存在的正确的比特被计数。

E.1 VUI语法

E.1.1 VUI参数语法

vui_parameters( ) {	C	描述符
aspect_ratio_info_present_flag	0	u(1)
if(aspect_ratio_info_present_flag) {		
aspect_ratio_idc	0	u(8)
if(aspect_ratio_idc == Extended_SAR) {		
sar_width	0	u(16)
sar_height	0	u(16)
}		
}		
overscan_info_present_flag	0	u(1)
if(overscan_info_present_flag)		
overscan_appropriate_flag	0	u(1)
video_signal_type_present_flag	0	u(1)
if(video_signal_type_present_flag) {		
video_format	0	u(3)
video_full_range_flag	0	u(1)
colour_description_present_flag	0	u(1)
if(colour_description_present_flag) {		
colour_primaries	0	u(8)
transfer_characteristics	0	u(8)
matrix_coefficients	0	u(8)
}		
}		
chroma_loc_info_present_flag	0	u(1)
if(chroma_loc_info_present_flag) {		
chroma_sample_loc_type_top_field	0	ue(v)
chroma_sample_loc_type_bottom_field	0	ue(v)
}		
timing_info_present_flag	0	u(1)
if(timing_info_present_flag) {		
num_units_in_tick	0	u(32)
time_scale	0	u(32)
fixed_frame_rate_flag	0	u(1)
}		
nal_hrd_parameters_present_flag	0	u(1)
if(nal_hrd_parameters_present_flag)		
hrd_parameters( )		
vcl_hrd_parameters_present_flag	0	u(1)
if(vcl_hrd_parameters_present_flag)		
hrd_parameters( )		
if(nal_hrd_parameters_present_flag || vcl_hrd_parameters_present_flag)		
low_delay_hrd_flag	0	u(1)
pic_struct_present_flag	0	u(1)
bitstream_restriction_flag	0	u(1)

if(bitstream_restriction_flag) {		
motion_vectors_over_pic_boundaries_flag	0	u(1)
max_bytes_per_pic_denom	0	ue(v)
max_bits_per_mb_denom	0	ue(v)
log2_max_mv_length_horizontal	0	ue(v)
log2_max_mv_length_vertical	0	ue(v)
num_reorder_frames	0	ue(v)
max_dec_frame_buffering	0	ue(v)
}		
}		

E.1.2 HRD参数语法

hrd_parameters( ) {	C	描述符
cpb_cnt_minus1	0	ue(v)
bit_rate_scale	0	u(4)
cpb_size_scale	0	u(4)
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
bit_rate_value_minus1[SchedSelIdx **]**	0	ue(v)
cpb_size_value_minus1[SchedSelIdx **]**	0	ue(v)
cbr_flag[SchedSelIdx **]**	0	u(1)
}		
initial_cpb_removal_delay_length_minus1	0	u(5)
cpb_removal_delay_length_minus1	0	u(5)
dpb_output_delay_length_minus1	0	u(5)
time_offset_length	0	u(5)
}		

E.2 VUI语义

E.2.1 VUI参数语义

aspect_ratio_info_present_flag 等于 1 表示 aspect_ratio_idc 存在。aspect_ratio_info_present_flag 等于 0 表示 aspect_ratio_idc 不存在。

aspect_ratio_idc 表示亮度样值的样点高宽比的取值。表 E-1 给出代码的含义。当 aspect_ratio_idc 的取值表示是 Extended_SAR，样点高宽比由 sar_width 和 sar_height 描述。当 aspect_ratio_idc 语法元素不存在，aspect_ratio_idc 的值应被推定为 0。

表 E-1－样点高宽比标识符的含义

aspect_ratio_idc	高宽比	(资料性) 样值用法
0	未定义	
1	1:1 (“正方形”)	1280x720 16:9 帧，无过扫描 1920x1080 16:9 帧，无过扫描(由 1920x1088 剪切) 640x480 4:3 帧，无过扫描
2	12:11	720x576 4:3 帧，无过扫描 352x288 4:3 帧，无过扫描
3	10:11	720x480 4:3 帧，无水平过扫描 352x240 4:3 帧，无过扫描
4	16:11	720x576 16:9 帧，无水平过扫描 540x576 4:3 帧，无水平过扫描
5	40:33	720x480 16:9 帧，无水平过扫描 540x480 4:3 帧，无水平过扫描
6	24:11	352x576 4:3 帧，无过扫描 480x576 16:9 帧，无水平过扫描
7	20:11	352x480 4:3 帧，无过扫描 480x480 16:9 帧，无水平过扫描
8	32:11	352x576 16:9 帧，无过扫描
9	80:33	352x480 16:9 帧，无过扫描
10	18:11	480x576 4:3 帧，无水平过扫描
11	15:11	480x480 4:3 帧，无水平过扫描
12	64:33	540x576 16:9 帧，无水平过扫描
13	160:99	540x480 16:9 帧，无水平过扫描
14..254	保留	
255	Extended_SAR	

sar_width 表示样点高宽比的水平尺寸（以任意单位）。

sar_height 表示样点高宽比的垂直尺寸（以与 sar_width 相同的任意单位）。

sar_width 和 sar_height 应是互质的或等于 0。当 aspect_ratio_idc 等于 0 或 sar_width 等于 0 或 sar_height 等于 0 时，样点高宽比应被视为本建议书 | 国际标准未定义的。

overscan_info_present_flag 等于 1 表示 overscan_appropriate_flag 存在。当 overscan_info_present_flag 等于 0 或不存在时，视频信号的优选显示方法未定义。

overscan_appropriate_flag 等于 1 表示被剪切的解码图像输出适合以过扫描显示。overscan_appropriate_flag 等于 0 表示被剪切的解码图像输出在向外到图像剪切矩形边缘的整个区域包含重要的可视信息，因此被剪切的解码图像输出不应以过扫描显示。相反地，它应以显示区域和剪切矩形的完全匹配方式显示或以欠扫描显示。

注1 — 例如，overscan_appropriate_flag等于1可以用于娱乐电视节目，或视频会议中人物的现场图像，而overscan_appropriate_flag等于0可以用于计算机屏幕捕捉或保安摄像内容。

video_signal_type_present_flag 等于 1 表示 video_format，video_full_range_flag 和 colour_description_present_flag 存在。video_signal_type_present_flag 等于 0 表示 video_format，video_full_range_flag 和 colour_description_present_flag 不存在。

video_format 表示图像在依本建议书 | 国际标准编码前的制式，见表 E-2 的规定。当 video_format 语法元素不存在，video_format 的值应被推定为 5。

表 E-2－video_format的含义

视频格式	含义
0	组合信号
1	PAL
2	NTSC
3	SECAM
4	MAC
5	未定义的视频制式
6	保留
7	保留

video_full_range_flag 表示黑电平和亮度与色度信号的范围由 E'_Y, E'_{PB}, 和 E'_{PR} 或 E'_R, E'_G, 和 E'_B 模拟信号分量得到。

当 video_full_range_flag 语法元素不存在时，video_full_range_flag 的值应被推定为等于 0。

colour_description_present_flag 等于 1 表示 colour_primaries，transfer_characteristics 和 matrix_coefficients 存在。colour_description_present_flag 等于 0 表示 colour_primaries，transfer_characteristics 和 matrix_coefficients 不存在。

colour_primaries 表示最初的原色的色度坐标，按照 CIE 1931 的规定（见表 E-3），x 和 y 的定义由 SO/CIE 10527 规定。

当 colour_primaries 语法元素不存在时，colour_primaries 的值应被推定为等于 2（色度未定义或由应用决定）。

表 E-3－色彩原色

值	原色			信息性注释
0	保留			为未来使用被 ITU-T / ISO/IEC 保留
1	原色	x	y	ITU-R BT.709-5 建议书
	绿	0.300	0.600	
	蓝	0.150	0.060	
	红	0.640	0.330	
	白 D65	0.3127	0.3290	
2	未定义			图像特征未知或由应用决定
3	保留			
4	原色	x	y	ITU-R BT.470-6 建议书 M 系统
	绿	0.21	0.71	
	蓝	0.14	0.08	
	红	0.67	0.33	
	白 C	0.310	0.316	
5	原色	x	y	ITU-R BT.470-6 建议书 B, G 系统
	绿	0.29	0.60	
	蓝	0.15	0.06	
	红	0.64	0.33	
	白 D65	0.3127	0.3290	
6	原色	x	y	运动图像与电视工程师社团 170M (1999)
	绿	0.310	0.595	
	蓝	0.155	0.070	
	红	0.630	0.340	
	白 D65	0.3127	0.3290	
7	原色	x	y	运动图像与电视工程师社团 240M (1999)
	绿	0.310	0.595	
	蓝	0.155	0.070	
	红	0.630	0.340	
	白 D65	0.3127	0.3290	
8	原色	x	y	普通电影（色彩滤镜用光源 C）
	绿	0.243	0.692 (Wratten 58)	
	蓝	0.145	0.049 (Wratten 47)	
	红	0.681	0.319 (Wratten 25)	
	白 C	0.310	0.316	
9-255	保留			为未来使用被 ITU-T / ISO/IEC 保留

transfer_characteristics 表示源图像的光电转换特性，模拟量范围从 0 到 1 的线性光强度输入 L_c 的函数，见表 E-4。

当 transfer_characteristics 语法元素不存在时，transfer_characteristics 的值应被推定为等于 2（转换特性未定义或由应用决定）。

表 E-4－转换特性

值	转换特性	信息性注释
0	保留	为未来使用被 ITU-T / ISO/IEC 保留
1	$V = 1.099\ L_c^{0.45} - 0.099$ $1 >= L_c >= 0.018$ $V = 4.500\ L_c$ $0.018 > L_c >= 0$	ITU-R BT.709-5 建议书
2	未定义	图像特征未知或由应用决定
3	保留	为未来使用被 ITU-T / ISO/IEC 保留
4	假定显示 gamma 2.2	ITU-R BT.470-6 建议书系统 M
5	假定显示 gamma 2.8	ITU-R BT.470-6 建议书系统 B, G
6	$V = 1.099\ L_c^{0.45} - 0.099$ $1 >= L_c >= 0.018$ $V = 4.500\ L_c$ $0.018 > L_c >= 0$	运动图像与电视工程师社团 170M (1999)
7	$V = 1.1115\ L_c^{0.45} - 0.1115$ for $1 >= L_c >= 0.0228$ $V = 4.0\ L_c$ $0.0228 > L_c >= 0$	运动图像与电视工程师社团 240M (1999)
8	$V = L_c$ $1 > L_c >= 0$	线性转换特性
9	$V = 1.0 - \text{Log10}(L_c) \div 2$ $1 >= L_c >= 0.01$ $V = 0.0$ $0.01 > L_c >= 0$	对数转换特性 (100:1 range)
10	$V = 1.0 - \text{Log10}(L_c) \div 2.5$ for $1 >= L_c >= 0.0031622777$ $V = 0.0$ $0.0031622777 > L_c >= 0$	对数转换特性(316.22777:1 range)
11..255	保留	为未来使用被 ITU-T / ISO/IEC 保留

matrix_coefficients 描述用于根据红、绿、蓝三原色得到亮度和色度信号的矩阵系数，见表 E-5。

matrix_coefficients 不应等于 0，除非下面两个条件都为真

— $BitDepth_C$等于$BitDepth_Y$

— chroma_format_idc等于3 (4:4:4)

在所有其它情况下 matrix_coefficients 等于 0 的使用规范被 ITU-T / ISO/IEC 为未来应用保留。

matrix_coefficients 不应等于 8，除非下面的一个或两个条件为真

— BitDepthC等于BitDepthY

— BitDepthC等于BitDepthY+ 1，且chroma_format_idc等于3 (4:4:4)

在所有其它情况下 matrix_coefficients 等于 8 的使用规范被 ITU-T / ISO/IEC 为未来应用保留。

当 matrix_coefficients 语法元素不存在时，matrix_coefficients 的值应被推定为等于 2。

matrix_coefficients 的解释见如下规定。

— E'_R, E'_G, 和E'_B 是范围从0到1的模拟量。

— 白色是E'_R 等于 1, E'_G等于 1, E'_B也等于 1。

— 黑色是E'_R 等于 0, E'_G等于 0, E'_B也等于 0。

— 如果video_full_range_flag等于0，用下列方程。

  — 如果matrix_coefficients is等于1，4，5，6或7，用下列方程。

$$Y = \text{Round}((1 << (BitDepth_Y - 8)) * (219 * E'_Y + 16)) \quad \text{(E-1)}$$

$$Cb = Round((1 << (BitDepth_C - 8)) * (224 * E'_{PB} + 128)) \quad (E\text{-}2)$$

$$Cr = Round((1 << (BitDepth_C - 8)) * (224 * E'_{PR} + 128)) \quad (E\text{-}3)$$

— 否则，如果matrix_coefficients is等于0或8，用下列方程。

$$R = (1 << (BitDepth_Y - 8)) * (219 * E'_R + 16) \quad (E\text{-}4)$$

$$G = (1 << (BitDepth_Y - 8)) * (219 * E'_G + 16) \quad (E\text{-}5)$$

$$B = (1 << (BitDepth_Y - 8)) * (219 * E'_B + 16) \quad (E\text{-}6)$$

— 否则，如果matrix_coefficients is等于2，matrix_coefficients语法元素解释为未知或由应用决定。

— 否则（matrix_coefficients不等于0，1，2，3，4，5，6，7或8），matrix_coefficients语法元素解释是为未来使用被 ITU-T / ISO/IEC保留。

— 否则，video_full_range_flag等于1，用下列方程。

— 如果matrix_coefficients is等于1，4，5，6或7，用下列方程。

$$Y = Round(((1 << BitDepth_Y) - 1) * E'_Y) \quad (E\text{-}7)$$

$$Cb = Round(((1 << BitDepth_C) - 1) * E'PB + (1 << (BitDepth_C - 1))) \quad (E\text{-}8)$$

$$Cr = Round(((1 << BitDepth_C) - 1) * E'PR + (1 << (BitDepth_C - 1))) \quad (E\text{-}9)$$

— 否则，如果matrix_coefficients is等于0或8，用下列方程。

$$R = ((1 << BitDepth_Y) - 1) * E'_R \quad (E\text{-}10)$$

$$G = ((1 << BitDepth_Y) - 1) * E'_G \quad (E\text{-}11)$$

$$B = ((1 << BitDepth_Y) - 1) * E'_B \quad (E\text{-}12)$$

— 否则，如果matrix_coefficients is等于2，matrix_coefficients语法元素解释为未知或由应用决定。

— 否则（matrix_coefficients不等于0，1，2，3，4，5，6，7或8），matrix_coefficients语法元素解释是为未来使用被 ITU-T / ISO/IEC保留。

— 如果matrix_coefficients不等于0或8，用下列方程。

$$E'_Y = K_R * E'_R + (1 - K_R - K_B) * E'_G + K_B * E'_B \quad (E\text{-}13)$$

$$E'_{PB} = 0.5 * (E'_B - E'_Y) \div (1 - K_B) \quad (E\text{-}14)$$

$$E'_{PR} = 0.5 * (E'_R - E'_Y) \div (1 - K_R) \quad (E\text{-}15)$$

注2 — 此时E'_Y是取值范围从0到1的模拟量，E'_{PB} 和 E'_{PR}是取值范围从−0.5到0.5的模拟量，白色等效地由$E'_Y = 1$, $E'_{PB} = 0$, $E'_{PR} = 0$表示。

— 否则，如果matrix_coefficients等于0，用下列方程。

$$Y = Round(G) \quad \text{(E-16)}$$

$$Cb = Round(B) \quad \text{(E-17)}$$

$$Cr = Round(R) \quad \text{(E-18)}$$

— 否则（matrix_coefficients等于8），用下列方程。

— 如果$BitDepth_C$等于 $BitDepth_Y$，用下列方程。

$$Y = Round(0.5 * G + 0.25 * (R + B)) \quad \text{(E-19)}$$

$$Cb = Round(0.5 * G - 0.25 (R + B)) \quad \text{(E-20)}$$

$$Cr = Round(0.5 * (R - B)) \quad \text{(E-21)}$$

注3 — 为表E-5中以YCgCo命名法表示的目的，方程式E-20和 E-21的Cb和 Cr分别可以称作Cg和Co，上述四个方程式的反变换应如下计算：

$$t = Y - Cb \quad \text{(E-22)}$$

$$G = Y + Cb \quad \text{(E-23)}$$

$$B = t - Cr \quad \text{(E-24)}$$

$$R = t + Cr \quad \text{(E-25)}$$

— 否则($BitDepth_C$ is not equal to $BitDepth_Y$)，用下列方程。

$$Cr = Round(R) - Round(B) \quad \text{(E-26)}$$

$$t = Round(B) + (Cr >> 1) \quad \text{(E-27)}$$

$$Cb = Round(G) - t \quad \text{(E-28)}$$

$$Y = t + (Cb >> 1) \quad \text{(E-29)}$$

注4 — 为表E-5中以YCgCo命名法表示的目的，方程式E-28和 E-26的Cb和 Cr分别可以称作Cg和Co，上述四个方程式的反变换应如下计算：

$$t = Y - (Cb >> 1) \quad \text{(E-30)}$$

$$G = t + Cb \quad \text{(E-31)}$$

$$B = t - (Cr >> 1) \quad \text{(E-32)}$$

$$R = B + Cr \quad \text{(E-33)}$$

表 E-5－矩阵系数

值	矩阵	资料性注释
0	GBR	典型地称作 RGB；见方程式 E-16 到 E-18
1	$K_R = 0.2126$; $K_B = 0.0722$	ITU-R BT.709-5 建议书和 运动图像与电视工程师社团 RP177 (1993)
2	未定义	图像特征未知或由应用决定
3	保留	为未来使用被 ITU-T / ISO/IEC 保留
4	$K_R = 0.30$; $K_B = 0.11$	美国联邦通信委员会 Title 47 Code of Federal Regulations (2003) 73.682 (a) (20)
5	$K_R = 0.299$; $K_B = 0.114$	ITU-R BT.470-6 建议书系统 B, G
6	$K_R = 0.299$; $K_B = 0.114$	运动图像与电视工程师社团 170M (1999)
7	$K_R = 0.212$; $K_B = 0.087$	运动图像与电视工程师社团 240M (1999)
8	YCgCo	见方程式 E-19 到 E-33
9-255	保留	为未来使用被 ITU-T / ISO/IEC 保留

chroma_loc_info_present_flag 等于 1 表示 hroma_sample_loc_type_top_field 和 chroma_sample_loc_type_bottom_field 存在。chroma_loc_info_present_flag 等于 0 表示 hroma_sample_loc_type_top_field 和 chroma_sample_loc_type_bottom_field 不存在。

chroma_sample_loc_type_top_field 和 **chroma_sample_loc_type_bottom_field** 表示色度样值在顶场和底场中的位置，见图 E-1。chroma_sample_loc_type_top_field 和 chroma_sample_loc_type_bottom_field 的值应在 0 到 5 的范围内。当 chroma_sample_loc_type_top_field 和 chroma_sample_loc_type_bottom_field 不存在时，其值应被推定为等于 0。

注5 — 在对渐变的源素材编码时，chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field应有相同的值。

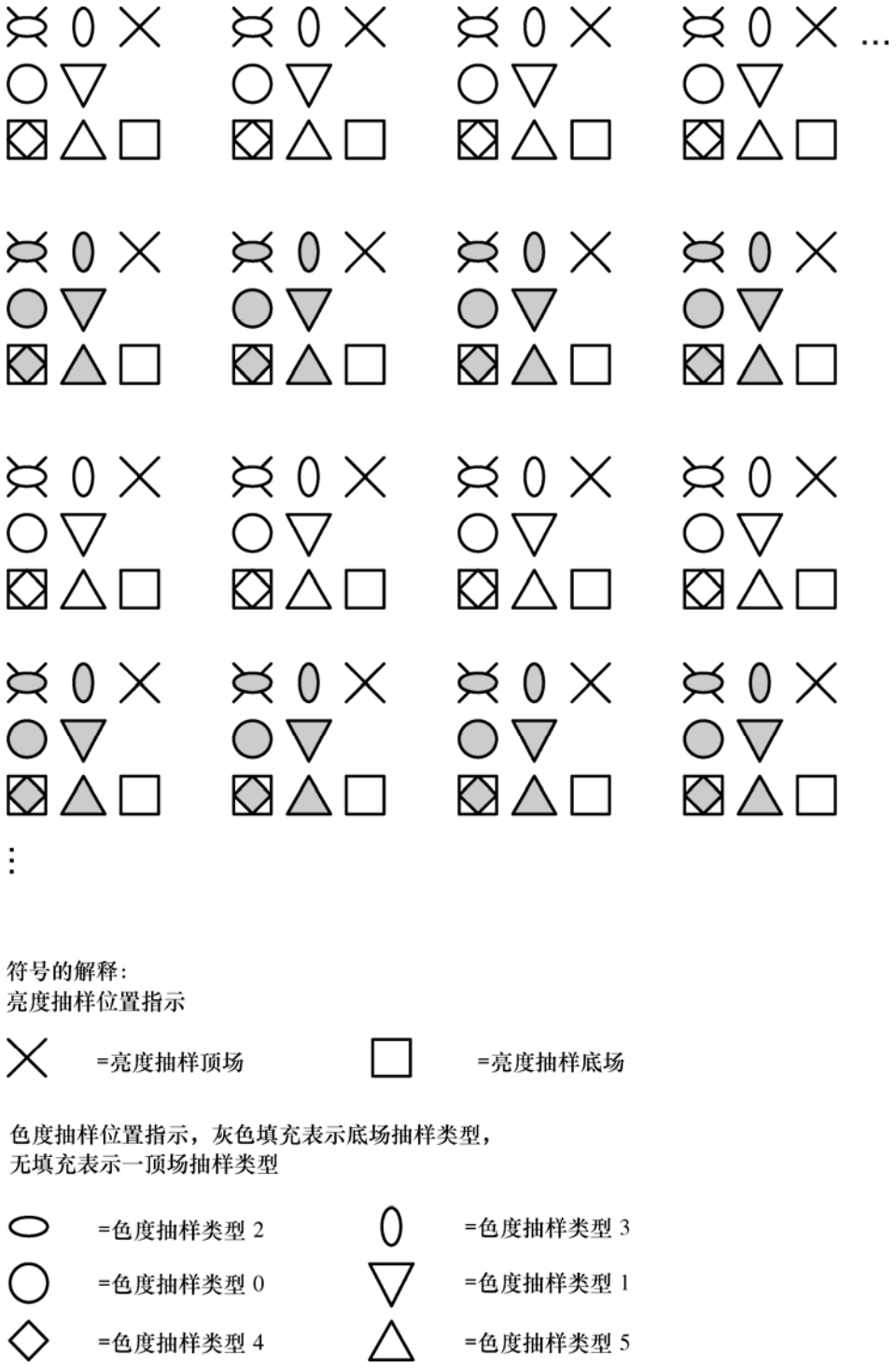


图 E-1 一顶场和底场色度样值的位置是
chroma_sample_loc_type_top_field 和 chroma_sample_loc_type_bottom_field的函数

timing_info_present_flag 等于 1 表示 num_units_in_tick，time_scale 和 fixed_frame_rate_flag 在比特流中存在。timing_info_present_flag 等于 0 表示 num_units_in_tick，time_scale 和 fixed_frame_rate_flag 在比特流中不存在。

num_units_in_tick 是运行在 time_scale Hz 的频率（相应地时钟跳变计数器加一，称作一个时钟跳变）下的时钟的时间单元的数量。num_units_in_tick 应大于 0。一个时钟跳变是在编码数据中可以再现的最小时间间隔。例如，当一个视频信号的时钟频率是 30000 ÷ 1001 Hz 时，time_scale 可以为 30 000 同时 num_units_in_tick 可以为 1001。见方程式 C-1。

time_scale 是在一秒钟内的时间单元的数量。例如，一个以 27MHz 的时钟测量时间的时间坐标系的 time_scale 为 27 000 000。time_scale 应大于 0。

fixed_frame_rate_flag 等于 1 表示在输出顺序上相连的任意两幅图像的 HRD 输出时间的暂时间隔受下述限制。fixed_frame_rate_flag 等于 0 表示对输出顺序上相连的任意两幅图像的 HRD 输出时间的暂时间隔没有该限制。

对输出的每幅图像 n，n 表示输出顺序中的第 n 幅图像，如果图像 n 不是比特流中的最后一幅输出图像，规定$\Delta t_{fi,dpb}(n)$的值为

$$\Delta t_{fi,dpb}(n) = \Delta t_{o,dpb}(n) \div DeltaTfiDivisor \quad (E\text{-}34)$$

此处$\Delta t_{o,dpb}(n)$由方程式 C-13 规定，DeltaTfiDivisor 根据包含图像 n 的视频编码序列的 pic_struct_present_flag，field_pic_flag，和 pic_struct 的值由表 E-6 规定。在表 E-6 标为"-"的条目表示 DeltaTfiDivisor 不依赖相应的语法元素决定。

当包含图像 n 的视频编码序列的 fixed_frame_rate_flag 等于 1 时，在下列任何一个或两个条件对后续图像 n_n 全为真时（n_n 见方程式 C-13 的用法），$\Delta t_{fi,dpb}(n)$的计算值应等于 t_c，见方程式 C-1（用包含图像 n 的视频编码序列的 t_c 的值）。

— 图像n_n与图像n在相同的视频编码序列中。

— 图像n_n在不同的视频编码序列中，包含n_n的视频编码序列的fixed_frame_rate_flag等于1，且两个视频编码序列的num_units_in_tick ÷ time_scale的值相同。

表 E-6－计算$\Delta t_{fi,dpb}(n)$的除数

pic_struct_present_flag	field_pic_flag	pic_struct	DeltaTfiDivisor
0	1	-	1
1	-	1	1
1	-	2	1
0	0	-	2
1	-	0	2
1	-	3	2
1	-	4	2
1	-	5	3
1	-	6	3
1	-	7	4
1	-	8	6

nal_hrd_parameters_present_flag 等于 1 表示 NAL HRD 参数（与类型 II 比特流一致性有关）存在。nal_hrd_parameters_present_flag 等于 0 表示 NAL HRD 参数不存在。

注6 — 当nal_hrd_parameters_present_flag等于0时，不通过本建议书 | 国际标准未规定的某种方法提供NAL HRD参数（包含NAL序列HRD参数信息和所有缓冲周期、图像定时SEI消息）就无法校验比特流的一致性。

当 nal_hrd_parameters_present_flag 等于 1 时，NAL HRD 参数（E.1.2 和 E.2.2 小节）紧跟本标志。

变量 NalHrdBpPresentFlag 如下得到。

— 如果下列任一条件为真，NalHrdBpPresentFlag的值被置为1。

— nal_hrd_parameters_present_flag在比特流中存在且等于1。

— 为比特流中的NAL HRD操作要存在缓冲周期的需求由缓冲周期SEI消息通过本建议书 | 国际标准未规定的某种方法由应用自行确定。

— 否则，NalHrdBpPresentFlag的值应等于0。

vcl_hrd_parameters_present_flag 等于 1 表示 VCL HRD 参数（与所有比特流一致性有关）存在。vcl_hrd_parameters_present_flag 等于 0 表示 VCL HRD 参数不存在。

注7 — 当vcl_hrd_parameters_present_flag等于0时，不通过本建议书 | 国际标准未规定的某种方法提供VCL HRD参数和所有缓冲周期、图像定时SEI消息就无法校验比特流的一致性。

当 vcl_hrd_parameters_present_flag 等于时 1，VCL HRD 参数（E.1.2 和 E.2.2 小节）紧跟本标志。

变量 VclHrdBpPresentFlag 如下得到。

— 如果下列任一条件为真时，VclHrdBpPresentFlag的值被置为1。

— vcl_hrd_parameters_present_flag在比特流中存在且等于1。

— 为比特流中的VCL HRD操作要存在缓冲周期的需求由缓冲周期SEI消息通过本建议书 | 国际标准未规定的某种方法由应用自行确定。

— 否则，VclHrdBpPresentFlag的值应等于0。

变量CpbDpbDelaysPresentFlag如下得到。

— 如果下列任一条件为真，CpbDpbDelaysPresentFlag的值被置为1。

— nal_hrd_parameters_present_flag在比特流中存在且等于1。

— vcl_hrd_parameters_present_flag在比特流中存在且等于1。

— 为比特流中的CPB和DPB输出延迟存在的需求由图像定时SEI消息通过本建议书 | 国际标准未规定的某种方法由应用自行确定。

— 否则，CpbDpbDelaysPresentFlag的值应等于0。

low_delay_hrd_flag 表示附件 C 规定的 HRD 操作模式。如果 fixed_frame_rate_flag 等于 1，low_delay_hrd_flag 应等于 0。

注8 — 当low_delay_hrd_flag等于1时，允许由于访问单元的比特数多而违反标称的CPB删除时间的“大图像”。希望但并不要求这样的“大图像”只是偶尔出现。

pic_struct_present_flag 等于 1 表示图像定时 SEI 消息（D.2.2 小节）包含 pic_struct 语法元素。pic_struct_present_flag 等于 0 表示 pic_struct 语法元素不存在于图像定时 SEI 消息中。当 pic_struct_present_flag 不存在，它的值应推定为 0。

bitstream_restriction_flag 等于 1，表示下面的视频编码序列比特流限制参数存在。bitstream_restriction_flag 等于 0，表示下面的视频编码序列比特流限制参数不存在。

motion_vectors_over_pic_boundaries_flag 等于 0 表示没有样值在图像边界以外，且没有用图像边界以外的一个或几个样值得到的局部样点位置处的样值去内部预测任何样值。motion_vectors_over_pic_boundaries_flag 等于 1 表示等于 1 表示可以把图像边界以外的一个或几个样值用于内部预测。当 motion_vectors_over_pic_boundaries_flag 语法元素不存在时，motion_vectors_over_pic_boundaries_flag 的值应被推定为等于 1。

max_bytes_per_pic_denom 表示在视频编码序列中与任何编码图像关联的 VCL NAL 单元的大小不会超过的一个字节数量。

规定在 NAL 单元流中代表一幅图像的字节数的目的是作为该图像的 VCL NAL 单元数据（也就是，对 VCL NAL 单元的 NumBytesInNALunit 变量的总和）的字节总数。max_bytes_per_pic_denom 的值应在 0 到 16 的范围内。

依据 max_bytes_per_pic_denom 的值适用下面的规定。

— 如果max_bytes_per_pic_denom等于0，表示没有限制。

— 否则（max_bytes_per_pic_denom不等于0），视频编码序列中不出现字节数大于下值的编码图像。

$$(PicSizeInMbs * RawMbBits) \div (8 * max\_bytes\_per\_pic\_denom) \quad (E\text{-}35)$$

当 max_bytes_per_pic_denom 语法元素不存在时，max_bytes_per_pic_denom 的值应被推定为等于 2。

max_bits_per_mb_denom 表示在视频编码序列中任何图像的任何宏块的 macroblock_layer( )数据的编码比特的最大数量。max_bits_per_mb_denom 的值应在 0 到 16 的范围内。

依据 max_bits_per_mb_denom 的值适用下面的规定。

— 如果max_bits_per_mb_denom等于0，表示没有限制。

— 否则（max_bits_per_mb_denom不等于0），视频编码序列中不出现比特数大于下值的编码macroblock_layer( )。

(128 + RawMbBits) ÷ max_bits_per_mb_denom (E-36)

依据 entropy_coding_mode_flag 的值，macroblock_layer( )数据的比特数按如下方法被计算。

— 如果entropy_coding_mode_flag等于0，macroblock_layer( )数据的比特数由对该宏块的macroblock_layer( )语法结构中的比特数给定。

— 否则（entropy_coding_mode_flag等于1），一个宏块的macroblock_layer( )数据的比特数由分析与宏块相关的macroblock_layer( )时read_bits(1)（见9.3.3.2.2和9.3.3.2.3小节）的次数给定。

当 max_bits_per_mb_denom 语法元素不存在时，max_bits_per_mb_denom 的值应被推定为等于 1。

log2_max_mv_length_horizontal 和 **log2_max_mv_length_vertical** 表示对视频编码序列中的所有图像，解码水平和垂直运动矢量分量各自的最大绝对值，以¼亮度样值为单位。一个 n 值表明没有运动矢量分量值会超出-2^n到2^n-1的范围。log2_max_mv_length_vertical 的值应在 0 到 16 的范围内。当 log2_max_mv_length_vertical 不存在时，log2_max_mv_length_vertical 和 log2_max_mv_length_vertical 的值应被推定为等于 16。

注9 — 解码水平和垂直运动矢量分量的最大绝对值还受profile和附件A规定的级别限度（level limits）的限制。

num_reorder_frames 表示视频编码序列中，解码顺序在任一帧、互补场对或不成对场之前，但输出顺序在其后的帧、互补场对或不成对场的数量。num_reorder_frames 的值应在 0 到 max_dec_frame_buffering 的范围内。当 num_reorder_frames 语法元素不存在时，num_reorder_frames 的值应被推定为等于 max_dec_frame_buffering。

max_dec_frame_buffering 表示 HRD 解码图像缓冲区(DPB)所需的大小，以缓冲帧数为单位。视频编码序列不应为在图像定时 SEI 消息的 dpb_output_delay 指定的时刻输出解码图像而请求大小超过 Max(1, max_dec_frame_buffering) 缓冲帧数的解码图像缓冲空间。max_dec_frame_buffering 的值应在 num_ref_frames 到 MaxDpbSize（见 A.3.1 小节的规定）的范围内。当 max_dec_frame_buffering 语法元素不存在时，max_dec_frame_buffering 的值应被推定为等于 MaxDpbSize。

E.2.2 HRD参数语义

cpb_cnt_minus1 加 1 表示在比特流中可选的 CPB 规范的数量。cpb_cnt_minus1 的值应在 0 到 31 的范围内。当 low_delay_hrd_flag 等于 1 时，cpb_cnt_minus1 应等于 0。当 cpb_cnt_minus1 不存在时，应推定它等于 0。

bit_rate_scale （与 with bit_rate_value_minus1[SchedSelIdx]一起）表示第 SchedSelIdx 个 CPB 输入比特率的最大值。

cpb_size_scale （与 cpb_size_value_minus1[SchedSelIdx]一起）表示第 SchedSelIdx 个 CPB 的缓冲空间大小。

bit_rate_value_minus1[SchedSelIdx **]** （与 bit_rate_scale 一起）表示第 SchedSelIdx 个 CPB 输入比特率的最大值。bit_rate_value_minus1[SchedSelIdx] 的值应在 0 到 $2^{32}-2$ 的范围内。对任意 SchedSelIdx 大于 0，bit_rate_value_minus1[SchedSelIdx]应大于 bit_rate_value_minus1[SchedSelIdx − 1]。以 bps 为单位的比特率表达式为

BitRate[SchedSelIdx] = (bit_rate_value_minus1[SchedSelIdx] + 1) * $2^{(6 + bit\_rate\_scale)}$ (E-37)

当 bit_rate_value_minus1[SchedSelIdx] 语法元素不存在时，对 BitRate[SchedSelIdx]的值应作如下推定。

— 如果profile_idc等于66、77或88，BitRate[SchedSelIdx] 的值作为VCL HRD参数应被推定为等于1000 * MaxBR，作为NAL HRD参数应被推定为等于1200 * MaxBR，此处的MaxBR参见A.3.1小节的规定。

— 否则，BitRate[SchedSelIdx] 的值作为VCL HRD参数应被推定为等于cpbBrVclFactor * MaxBR，作为NAL HRD参数应被推定为等于cpbBrNalFactor * MaxBR，此处的cpbBrVclFactor，cpbBrNalFactor，和MaxBR参见A.3.3小节的规定。

cpb_size_value_minus1[SchedSelIdx **]** （与 cpb_size_scale 一起）表示第 SchedSelIdx 个 CPB 的缓冲空间大小。cpb_size_value_minus1[SchedSelIdx] 的值应在 0 到 $2^{32}-2$ 的范围内。对任意 SchedSelIdx 大于 0，cpb_size_value_minus1[SchedSelIdx]应小于等于 cpb_size_value_minus1[SchedSelIdx −1]。

以比特数为单位的 CPB 空间大小为

$$CpbSize[SchedSelIdx] = (cpb\_size\_value\_minus1[SchedSelIdx] + 1) * 2^{(4 + cpb\_size\_scale)} \quad \text{(E-38)}$$

如果 cpb_size_value_minus1[SchedSelIdx] 语法元素不存在，对 CpbSize[SchedSelIdx]的值应作如下推定。

— 如果profile_idc等于66、77或88，CpbSize[SchedSelIdx] 的值作为VCL HRD参数应被推定为等于1000 * MaxCPB，作为NAL HRD参数应被推定为等于1200 * MaxCPB，此处的MaxCPB参见A.3.1小节的规定。

— 否则，CpbSize[SchedSelIdx] 的值作为VCL HRD参数应被推定为等于cpbBrVclFactor * MaxCPB，作为NAL HRD参数应被推定为等于cpbBrNalFactor * MaxCPB，此处的cpbBrVclFactor，cpbBrNalFactor，和 MaxCPB参见A.3.3小节的规定。

cbr_flag[SchedSelIdx **]** 等于 0 表示 HRD 用第 SchedSelIdx 个 CPB 规范解码本比特流，假想流传送调度程序（HSS）工作在间歇比特率模式。br_flag[SchedSelIdx]等于 1 表示 HSS 工作在固定比特率模式（CBR）。当 br_flag[SchedSelIdx] 语法元素不存在时，应推定 cbr_flag 的值等于 0。

initial_cpb_removal_delay_length_minus1 表示缓冲周期 SEI 消息的 initial_cpb_removal_delay[SchedSelIdx] 和 initial_cpb_removal_delay_offset[SchedSelIdx] 语法元素的比特长度。initial_cpb_removal_ delay[SchedSelIdx]和 initial_cpb_removal_delay_offset[SchedSelIdx] 的长度是 initial_cpb_removal_delay_length_ minus1 + 1 。当 initial_cpb_removal_delay_length_minus1 语法元素存在于大于一个的 VUI 参数语法结构内的 hrd_parameters( )语法结构中时，initial_cpb_removal_delay_length_minus1 参数值在两个 hrd_parameters( )语法结构中应相等。当 initial_cpb_removal_delay_length_minus1 语法元素不存在时，应推定它等于 23。

cpb_removal_delay_length_minus1 表示 cpb_removal_delay 语法元素的比特长度。图像定时 SEI 消息的 cpb_removal_delay 语法元素的长度是 cpb_removal_delay_length_minus1 + 1。当 cpb_removal_delay_length_minus1 语法元素存在于大于一个的 VUI 参数语法结构内的 hrd_parameters( ) 语法结构中时，cpb_removal_delay_length_minus1 参数值在两个 hrd_parameters( ) 语法结构中应相等。当 cpb_removal_delay_length_minus1 语法元素不存在时，应推定它等于 23。

dpb_output_delay_length_minus1 表示 dpb_output_delay 语法元素的比特长度。图像定时 SEI 消息的 dpb_output_delay 语法元素的长度是 dpb_output_delay_length_minus1 + 1。当 dpb_output_delay_length_minus1 语法元素存在于大于一个的 VUI 参数语法结构内的 hrd_parameters( )语法结构中时，dpb_output_delay_length_minus1 参数值在两个 hrd_parameters( )语法结构中应相等。当 dpb_output_delay_length_minus1 语法元素不存在时，应推定它等于 23。

time_offset_length 大于 0 表示 time_offset 语法元素的比特长度。time_offset_length 等于 0 表示 time_offset 语法元素不存在。当 time_offset_length 语法元素存在于大于一个的 VUI 参数语法结构内的 hrd_parameters( )语法结构中时，time_offset_length 参数值在两个 hrd_parameters( )语法结构中应相等。当 time_offset_length 语法元素不存在时，应推定它等于 24。

ITU-T 系列建议书

A系列	ITU-T工作的组织
D系列	一般资费原则
E系列	综合网络运行、电话业务、业务运行和人为因素
F系列	非话电信业务
G系列	传输系统和媒质、数字系统和网络
H系列	**视听及多媒体系统**
I系列	综合业务数字网
J系列	有线网络和电视、声音节目及其它多媒体信号的传输
K系列	干扰的防护
L系列	电缆和外部设备其它组件的结构、安装和保护
M系列	电信管理，包括TMN和网络维护
N系列	维护：国际声音节目和电视传输电路
O系列	测量设备的技术规范
P系列	电话传输质量、电话设施及本地线路网络
Q系列	交换和信令
R系列	电报传输
S系列	电报业务终端设备
T系列	远程信息处理业务的终端设备
U系列	电报交换
V系列	电话网上的数据通信
X系列	数据网、开放系统通信和安全性
Y系列	全球信息基础设施、互联网协议问题和下一代网络
Z系列	用于电信系统的语言和一般软件问题

瑞士印刷
2005年，日内瓦